Machine Learning for Time Series

Lecture 1: Pattern Recognition and Detection

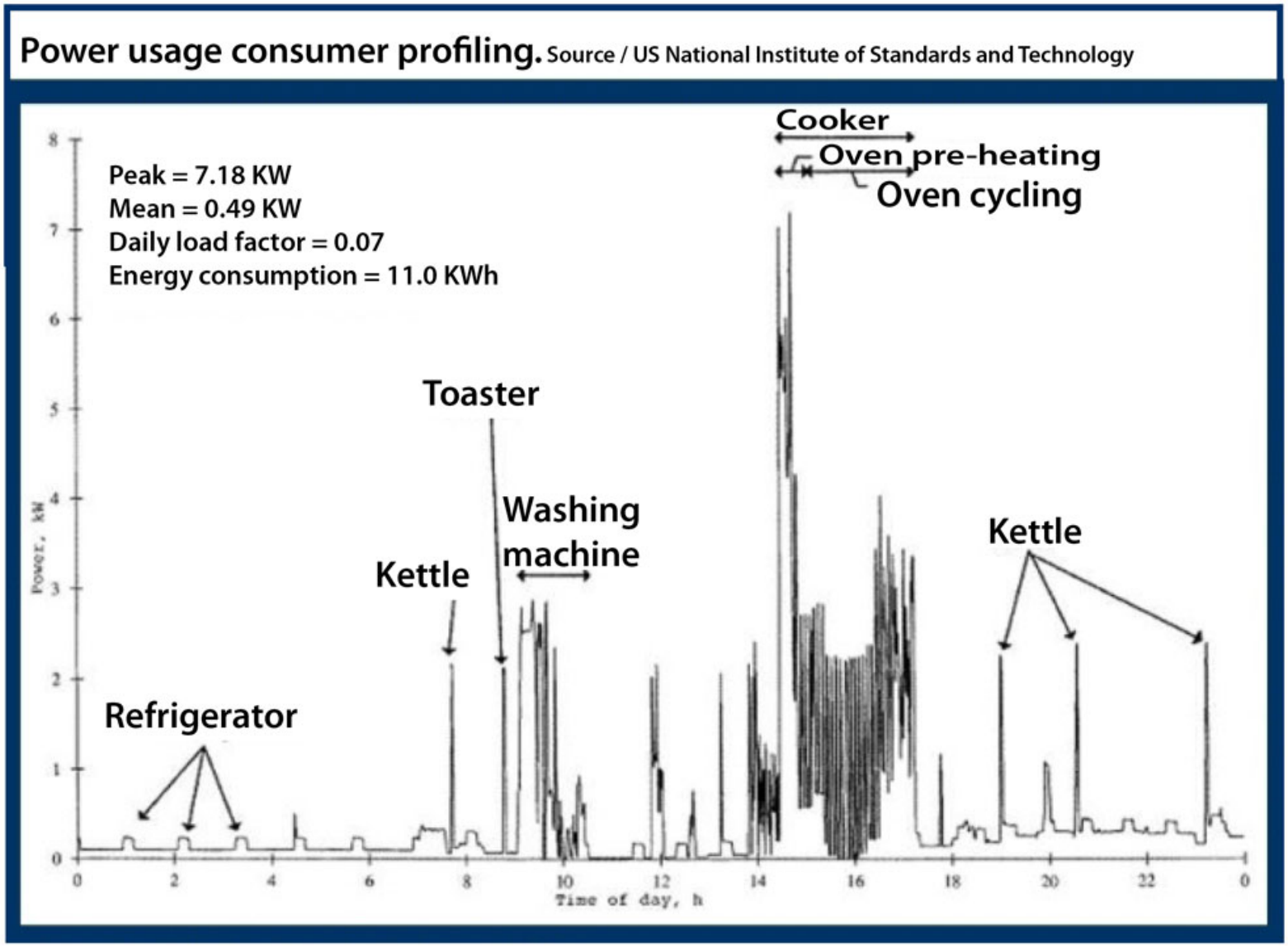
Laurent Oudre
laurent.oudre@ens-paris-saclay.fr

Master MVA
2023-2024

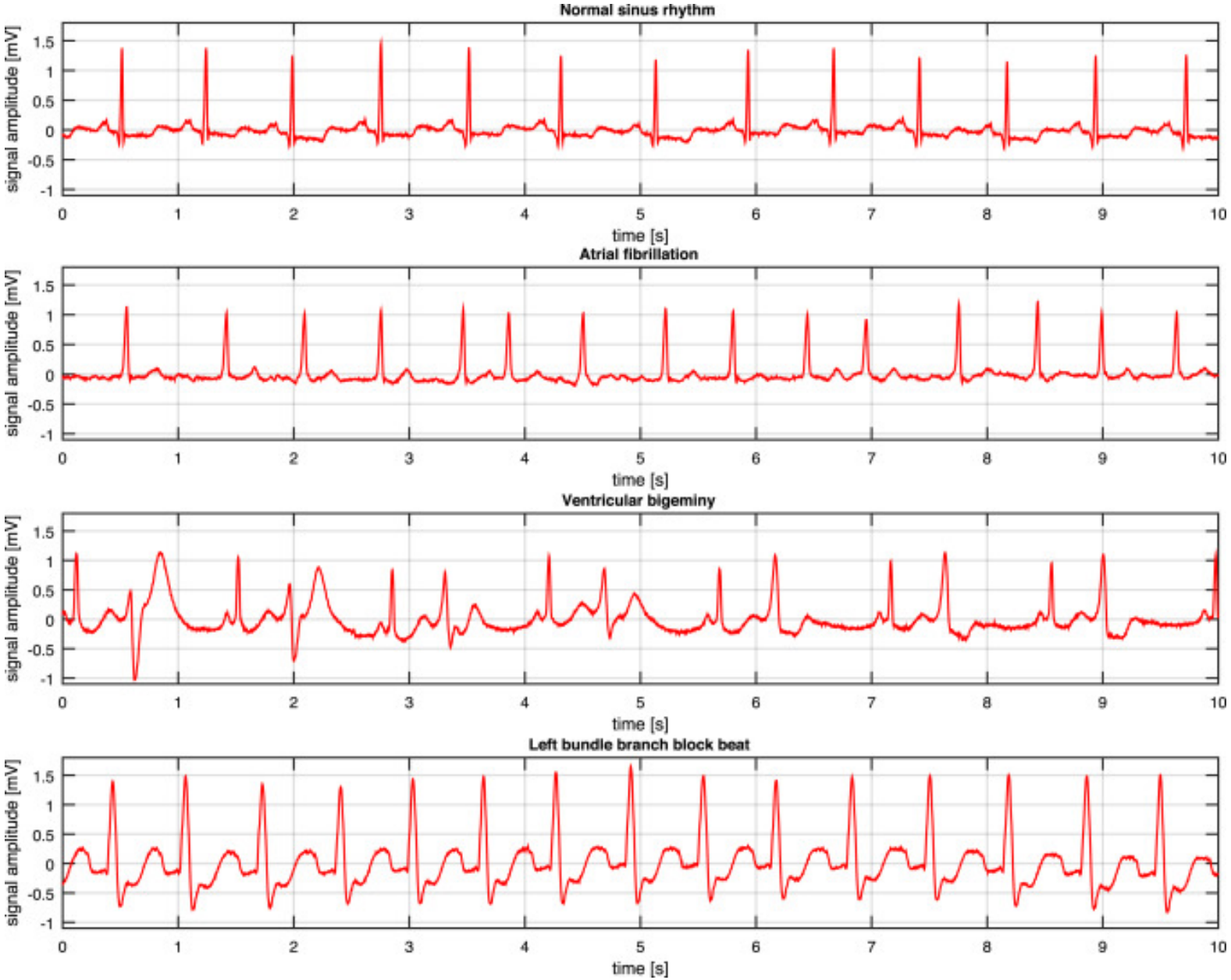
Contents

Contents

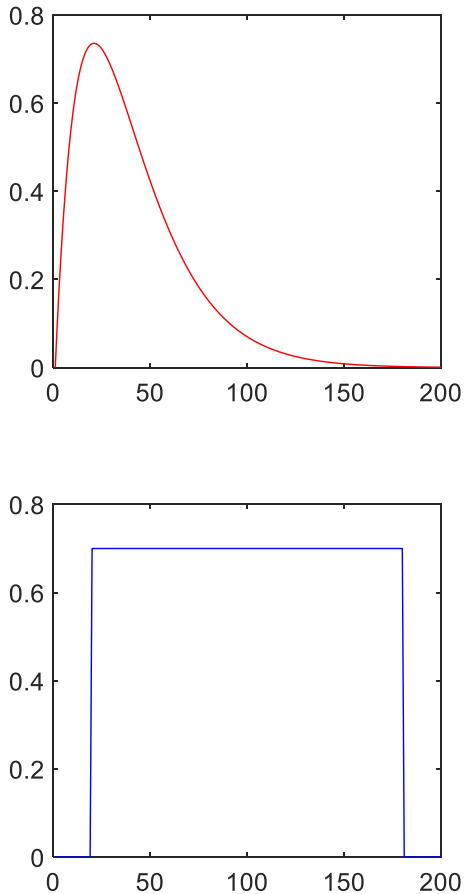
Motivation


Power usage consumer profiling. Source / US National Institute of Standards and Technology
Peak = 7.18 KW
Mean = 0.49 KW
Daily load factor = 0.07
Energy consumption = 11.0 KWh
Cooker
Oven pre-heating
Oven cycling
Toaster
Washing machine
Kettle
Kettle
Refrigerator
Power, kW
Time of day, h

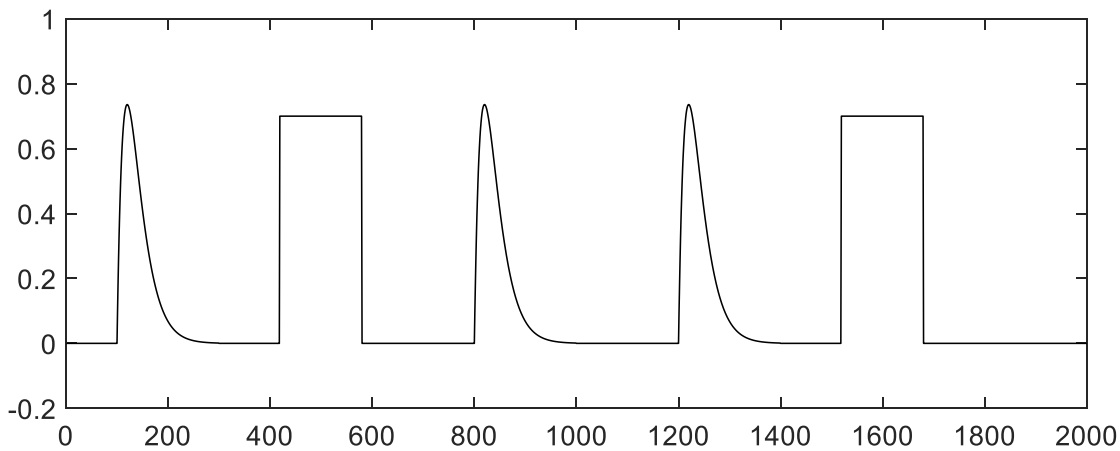
Motivation


Normal sinus rhythm
Atrial fibrillation
Ventricular bigeminy
Left bundle branch block beat
signal amplitude [mV]
time [s]

Problem 1: Pattern Detection

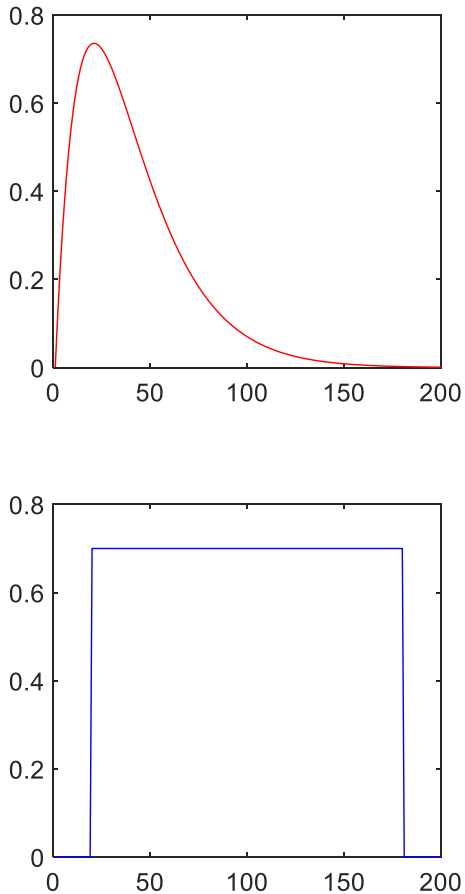


Dictionary of patterns

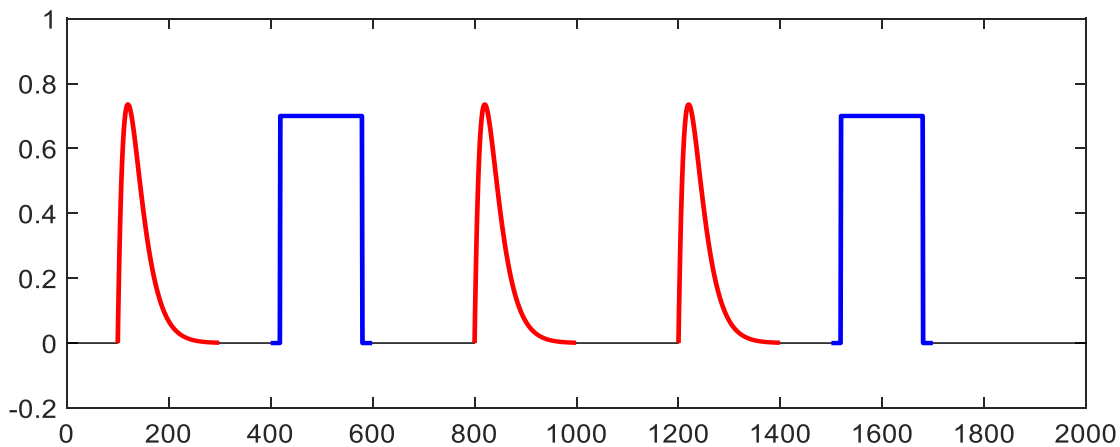


Input time series

Problem 1: Pattern Detection



Dictionary of patterns



Annotated time series

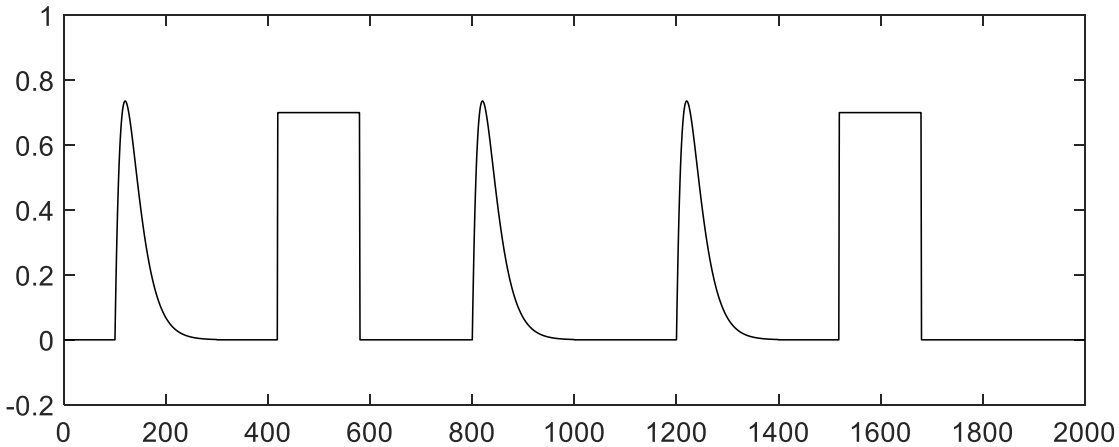
Problem 1: Pattern Detection

Pattern Detection

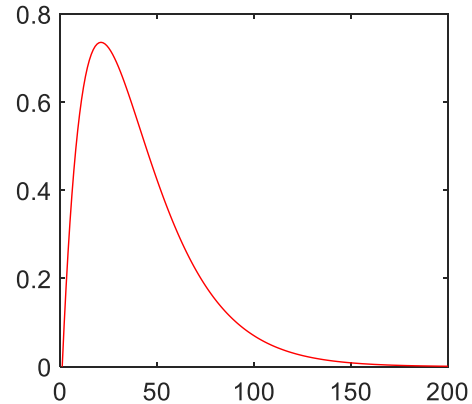
Given a dictionary of patterns, retrieve these patterns in an input time series

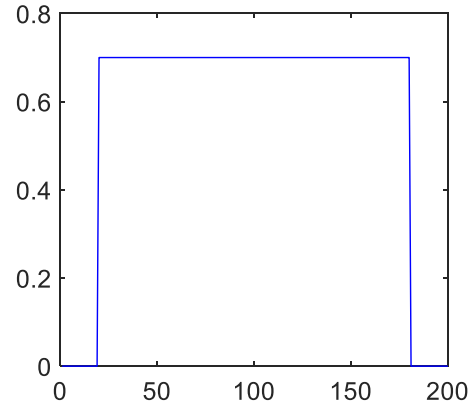
- The patterns/the time series can be multivariate
- The patterns may have different lengths
- The patterns can be annotated, i.e. be linked to a specific phenomenon of interest: in this context, pattern recognition will provide an automated annotation of the input time series

Problem 2: Pattern Extraction



Input time series





Extracted patterns

Problem 2: Pattern Extraction

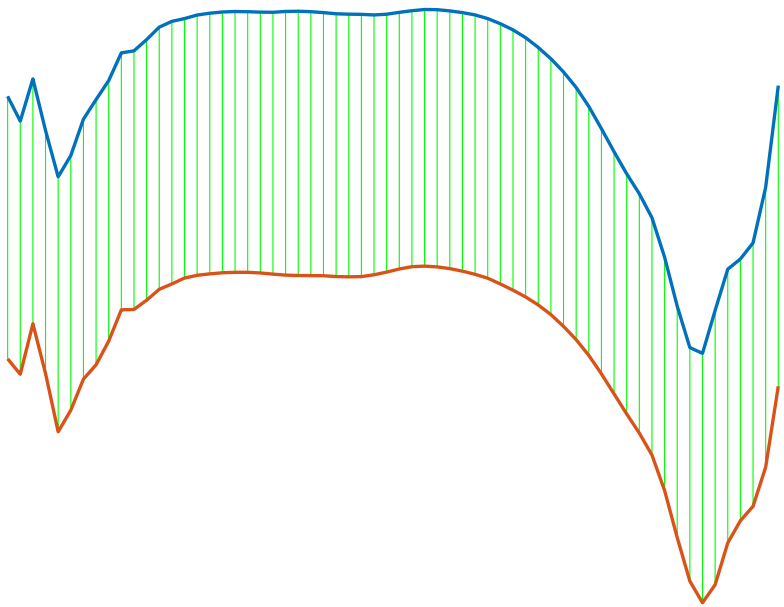
Pattern Extraction

Given an input time series (or a set of time series), learn a dictionary of patterns

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

Contents

Euclidean distance

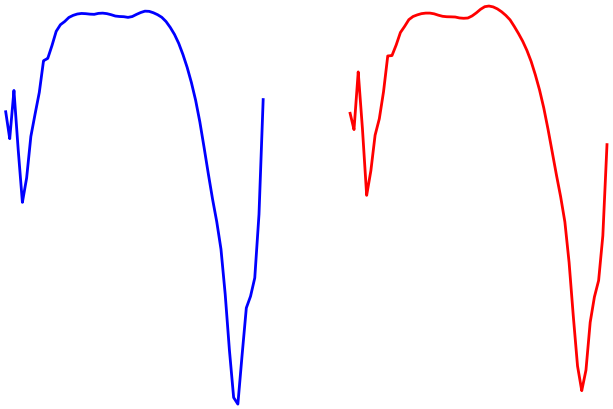


$$d_{EUC}(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{n=1}^{N} (x[n] - y[n])^2}$$

- Sensitive to time shifts, amplitude changes, offsets and dilatation/contraction
- Necessity to have a perfect match between the timelines
- Sensitive to outliers but approximately OK with low AWGN

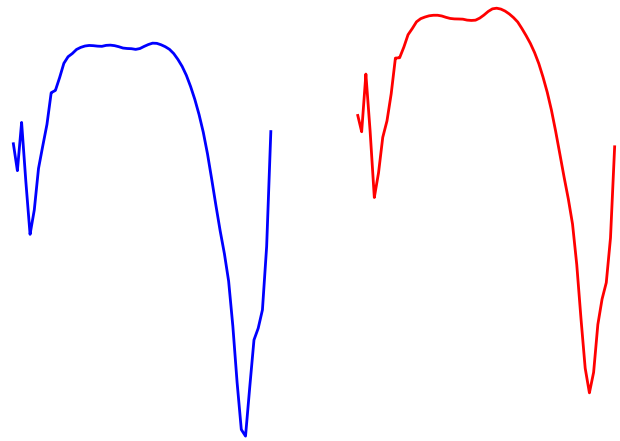
What influences Euclidean distance?

Baseline case



$$d_{EUC} = 3.5$$

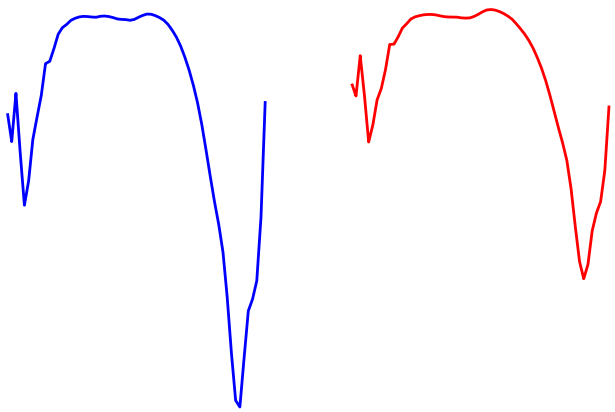
Offset



$$d_{EUC} = 9.6$$

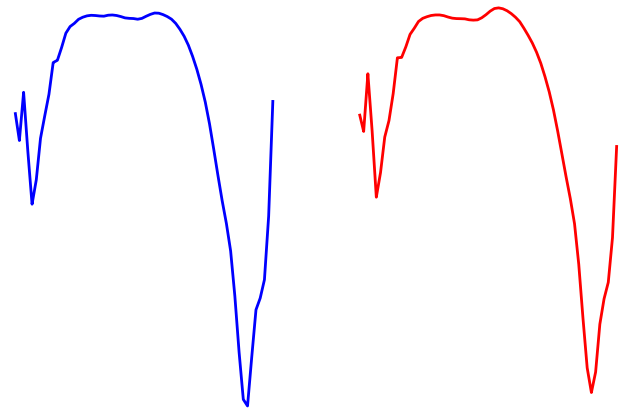
What influences Euclidean distance?

Amplitude shift (70 %)



$d_{EUC} = 12.8$

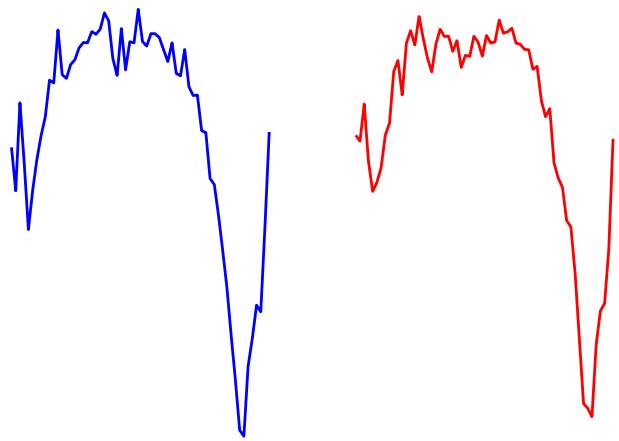
Time shift (1.6 %)



$d_{EUC} = 5.3$

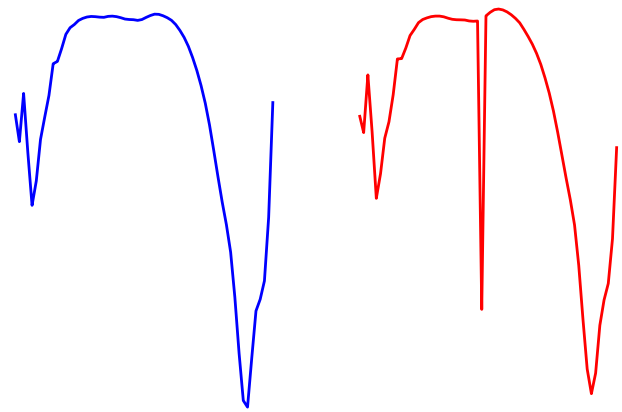
What influences Euclidean distance?

Additive white Gaussian noise



$$d_{EUC} = 6.9$$

Outliers



$$d_{EUC} = 10.3$$

Normalized Euclidean distance

In order to improve the robustness to changes in amplitude/offset, several authors recommand to use a normalized Euclidean distance :

$$d_{nEUC}(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{n=1}^{N} (\tilde{x}[n] - \tilde{y}[n])^2}$$

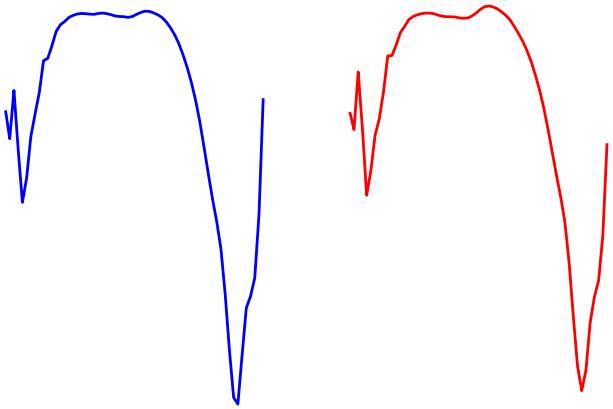
where

$$\tilde{x}[n] = \frac{x[n] - \mu_x}{\sigma_x} \quad \text{z-score normalization}$$

- Still sensitive to time shifts, dilatation/contraction, outliers and timelines
- Normalization may increase the sensitivity with respect to additive noise

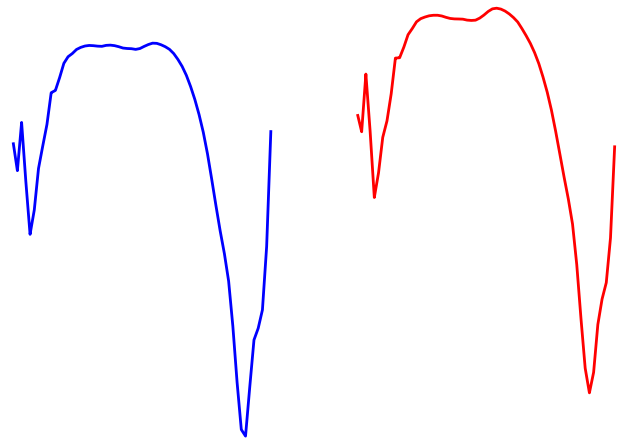
What influences normalized Euclidean distance?

Baseline case



$$d_{nEUC} = 0.8$$

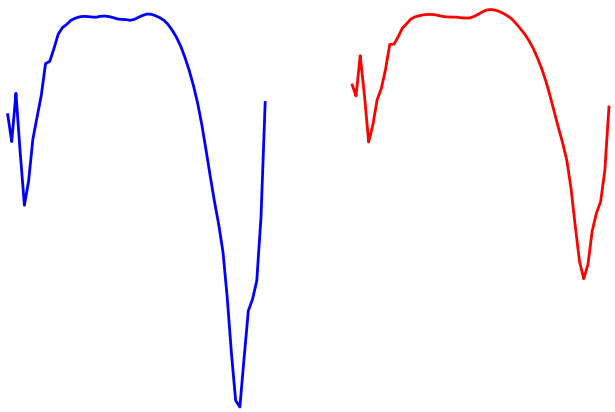
Offset



$$d_{nEUC} = 0.8$$

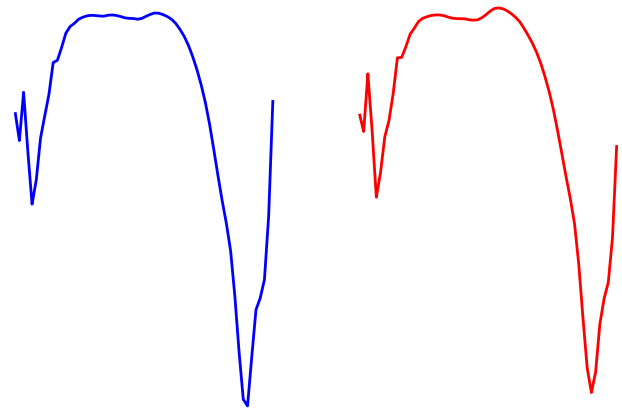
What influences normalized Euclidean distance?

Amplitude shift (70 %)



$d_{nEUC} = 0.8$

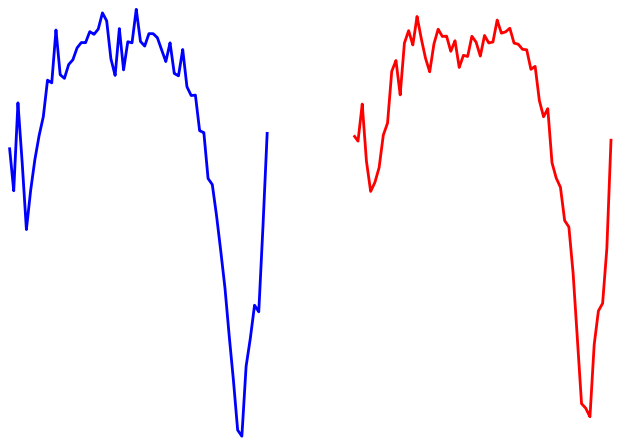
Time shift (1.6 %)



$d_{nEUC} = 1.9$

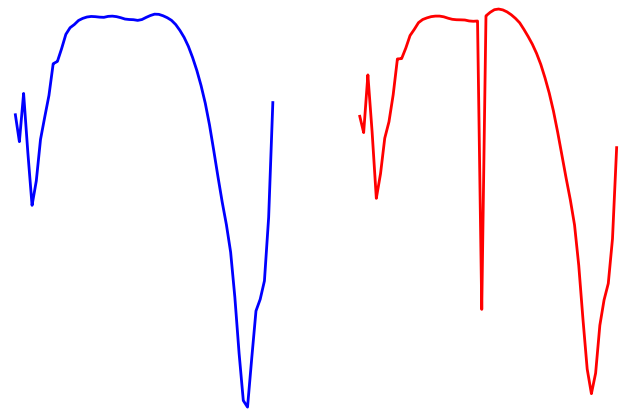
What influences normalized Euclidean distance?

Additive white Gaussian noise



$d_{nEUC} = 1.9$

Outliers



$d_{nEUC} = 2.7$

Open questions

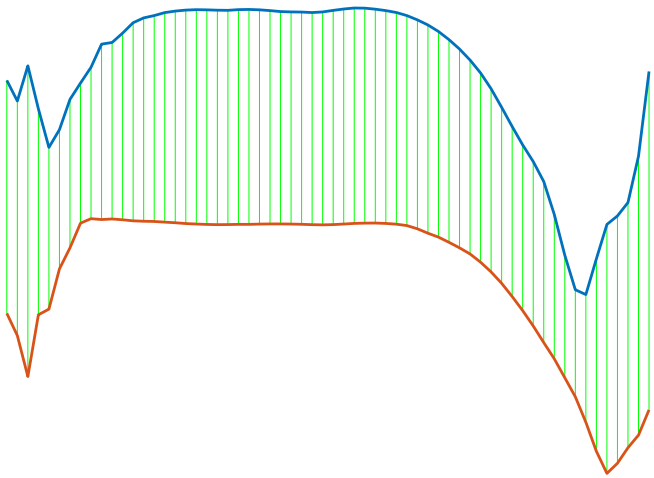
Open questions :

- How do we decrease the sensitivity with respect to timelines (contraction/dilatation, time shifts...)?
- How can we compare two time series of different lengths?
- How do we extend the notion of Euclidean distance to nonlinear timeline modifications?

New notion of distance : Dynamic Time Warping (DTW) [Berndt et al., 1994]

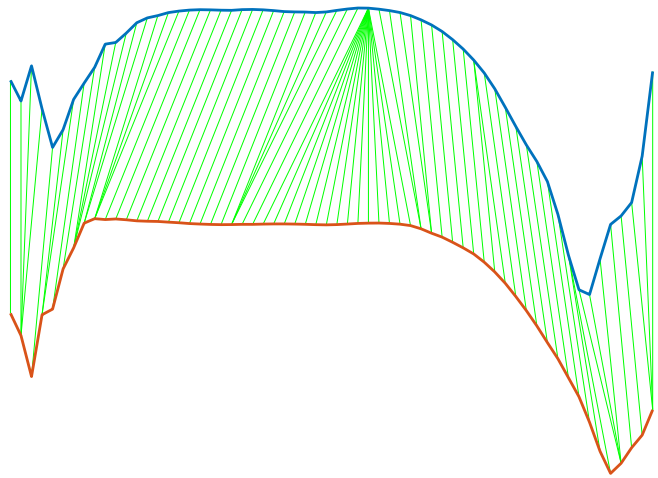
Differences between Euclidean and DTW

Euclidean distance



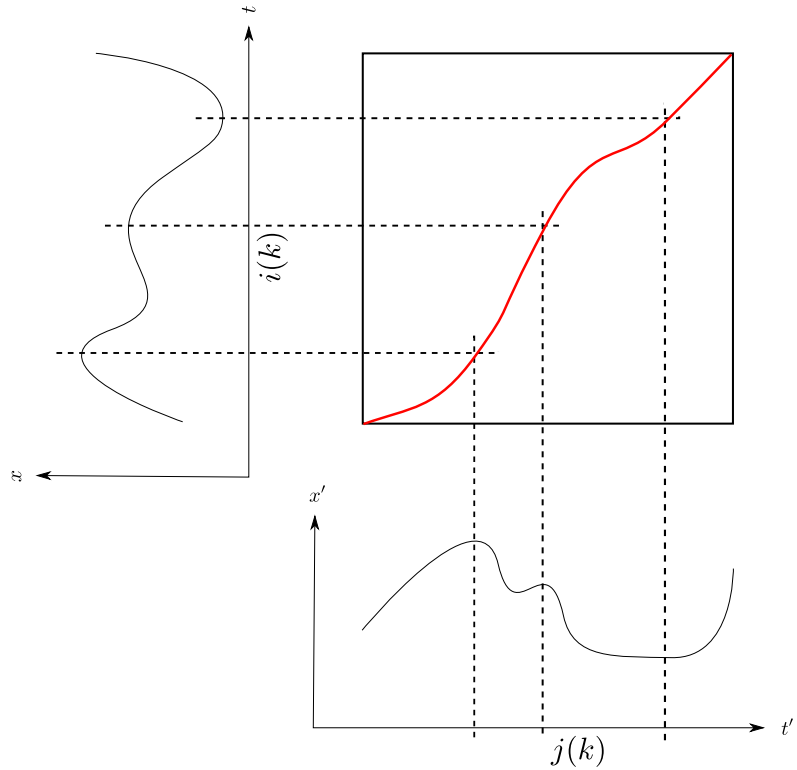
Sample $x[n]$ is associated to sample $y[n]$

DTW



Sample $x[i_k]$ is associated to sample $y[j_k]$

Notion of path



- DTW computes a correspondence between the elements of **x** and those of **y**.
- Mapping function called **path** :

$$P = ((i_1, j_1)), \cdots, (i_{K_P}, j_{K_P})) \in (\mathbb{N} \times \mathbb{N})^{K_P}, K_P \in \mathbb{N}$$

$$(i_k, j_k) \in P \Leftrightarrow y[j_k] \text{ is matched with } x[i_k]$$

Choice of the optimal path

- The path P is evaluated through a cost function

$$w(P) = \sum_{k=1}^{K_P} (x[i_k] - y[j_k])^2$$

- The final DTW distance is computed as the minimal value for the cost function

$$d_{DTW}(\mathbf{x}, \mathbf{y}) = \sqrt{\min_{P \in \mathcal{P}} w(P)}$$

- Note that if $K_P = N$ and $i_k = j_k = k$, DTW is exactly equal to the Euclidean distance

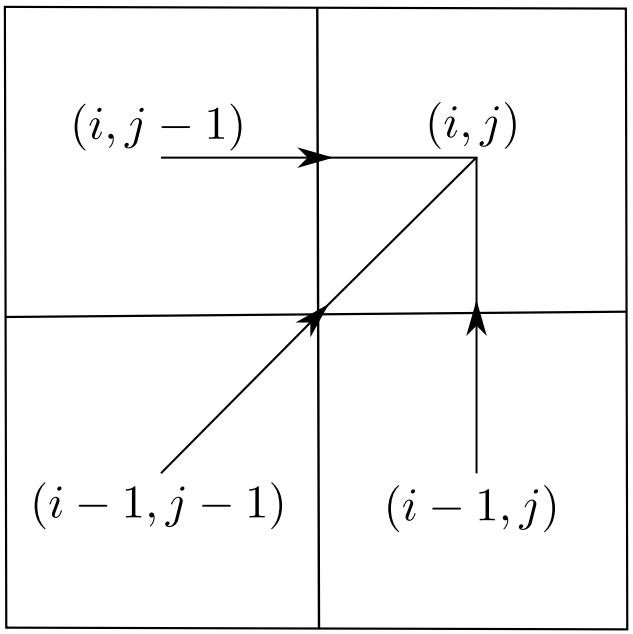
Set of acceptable paths $\mathcal{P}$

How do we choose the set of acceptable paths $\mathcal{P}$? Given two time series $\mathbf{x}$ and $\mathbf{y}$ of lengths M and N, the path P should be:

- **Continuous** $i_k - i_{k-1} \leq 1$ and $j_k - j_{k-1} \leq 1$
  Imagining the plane as a chessboard, the path can only go to the positions a king could;
- **Monotonic** $i_{k-1} \leq i_k$ and $j_{k-1} \leq j_k$
  The path can only go up and right, and diagonally up and right;
- **Bounded** $(i_1, j_1) = (1, 1)$ and $(i_{K_P}, j_{K_P}) = (M, N)$
  The path starts/ends by matching together the first/last elements of both signals

$\mathcal{P}$: set of paths respecting those three restrictions

Set of acceptable paths $\mathcal{P}$



Limited set of indexes to get to (i_k, j_k)

$$(i_{k-1}, j_{k-1}) = \begin{cases} & (i_k - 1, j_k) \\ \text{or} & (i_k, j_k - 1) \\ \text{or} & (i_k - 1, j_k - 1). \end{cases}$$

A recursive problem

- Let $C(i, j)$ be the minimum cost for the path starting at $(1, 1)$, arriving at (i, j), and belonging to $\mathcal{P}$
- It goes that

$$\text{DTW}(\mathbf{x}, \mathbf{y}) = \sqrt{C(M, N)}$$

- So how can we compute $C(M, N)$? By using a **dynamic programming** approach
- By using the properties of the acceptable paths, we can compute this quantity recursively

DTW algorithm

- Initialization:

$$C(1,j) = (x[1] - y[j])^2 \qquad C(i,1) = (x[i] - y[1])^2$$

- How to compute $C(2,2)$? There are three ways for reaching $(2,2)$
  - From $(1,1)$: associated cost $C(1,1) + (x[2] - y[2])^2$
  - From $(1,2)$: associated cost $C(1,2) + (x[2] - y[2])^2$
  - From $(2,1)$: associated cost $C(2,1) + (x[2] - y[2])^2$
- If we only keep the minimum cost, we have

$$C(2,2) = (x[2] - y[2])^2 + \min\{C(1,1), C(1,2), C(2,1)\}$$

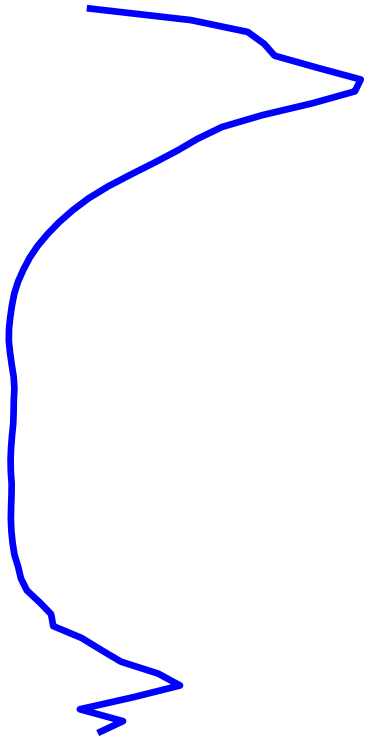
- It is therefore possible to construct the matrix $C(i,j)$ iteratively with increasing i and j

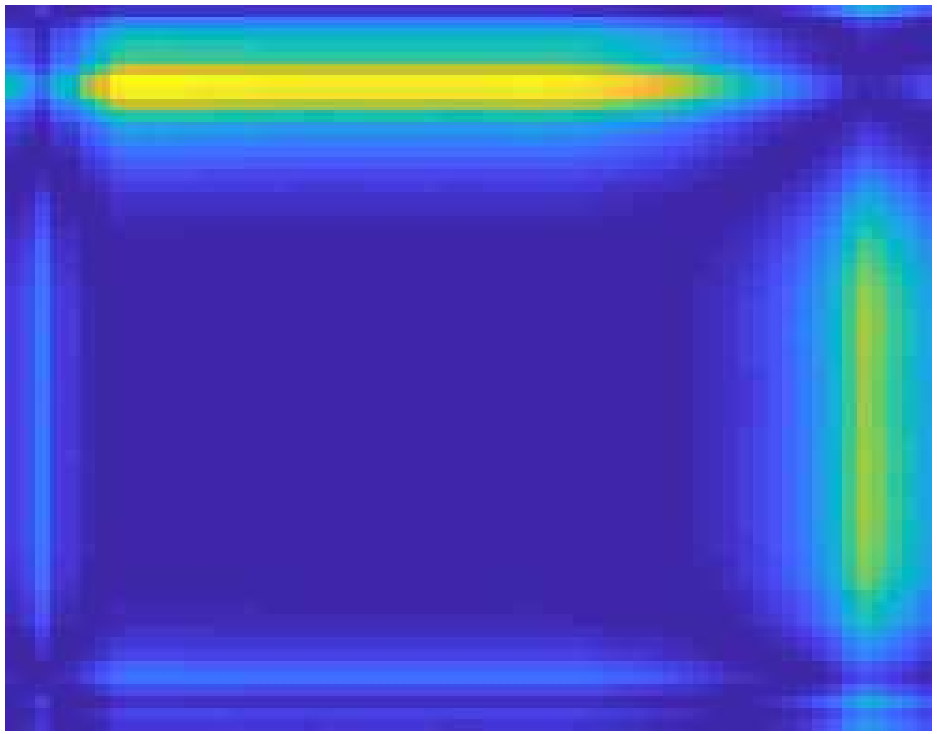
DTW algorithm

Algorithm 1: Dynamic Time Warping

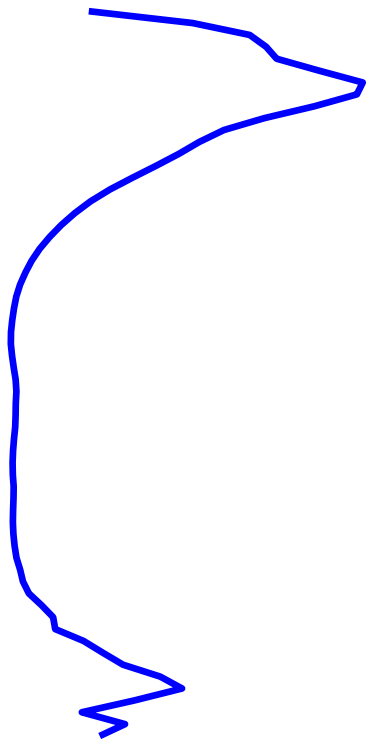
Inputs : Time series $\mathbf{x} \in \mathbb{R}^M$ and $\mathbf{y} \in \mathbb{R}^N$
Output: DTW $d_{DTW}(\mathbf{x}, \mathbf{y})$
$\mathbf{D} = \mathbf{0}_{M \times N}$;
$\mathbf{C} = \mathbf{0}_{M \times N}$;
for $1 \leq i \leq M$, $1 \leq j \leq N$ **do**
  $D(i,j) = (x[i] - y[j])^2$;
end
$C(1,:) = D(1,:)$;
$C(:,1) = D(:,1)$;
for $2 \leq i \leq M$, $2 \leq j \leq N$ **do**
  $C(i,j) = D(i,j) + \min\{C(i-1,j-1), C(i-1,j), C(i,j-1)\}$;
end
$d_{DTW}(\mathbf{x}, \mathbf{y}) = \sqrt{C(M,N)}$;

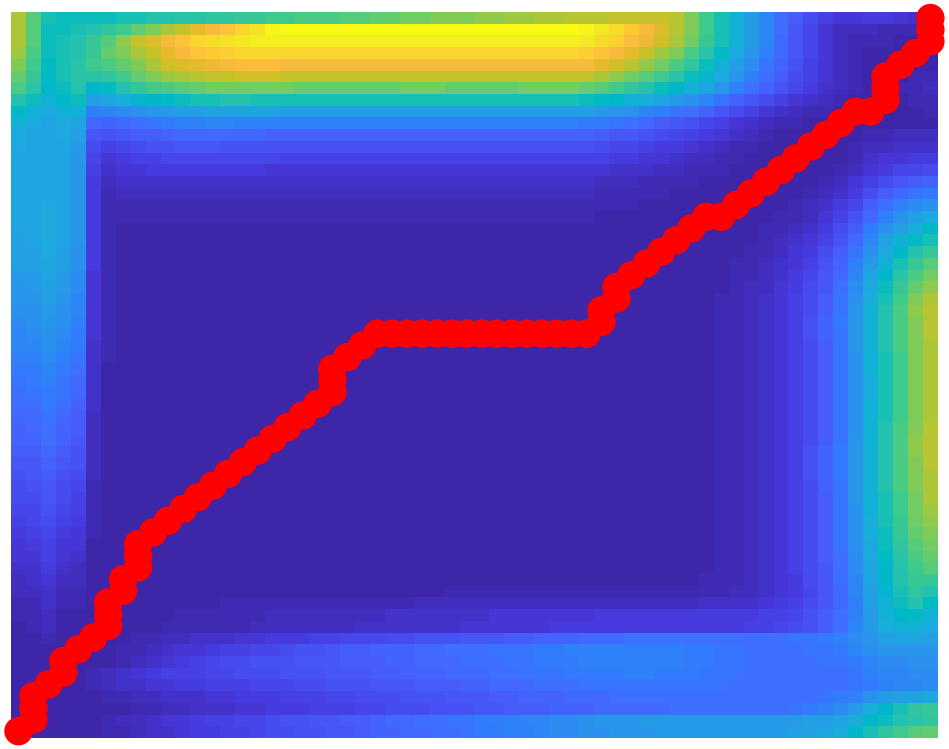
Step 1 : Distance matrix



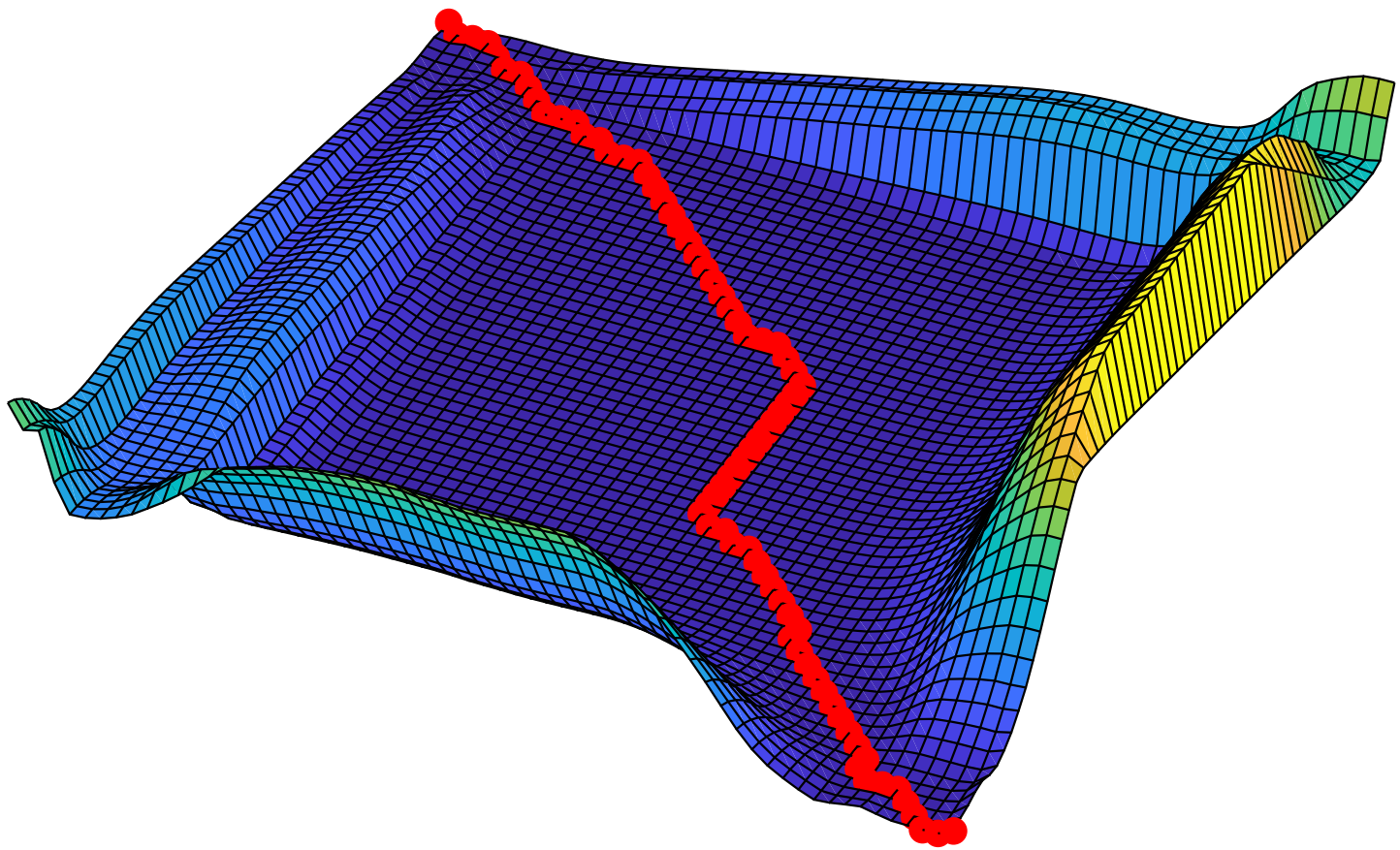


Step 2 : Cumulative distance matrix

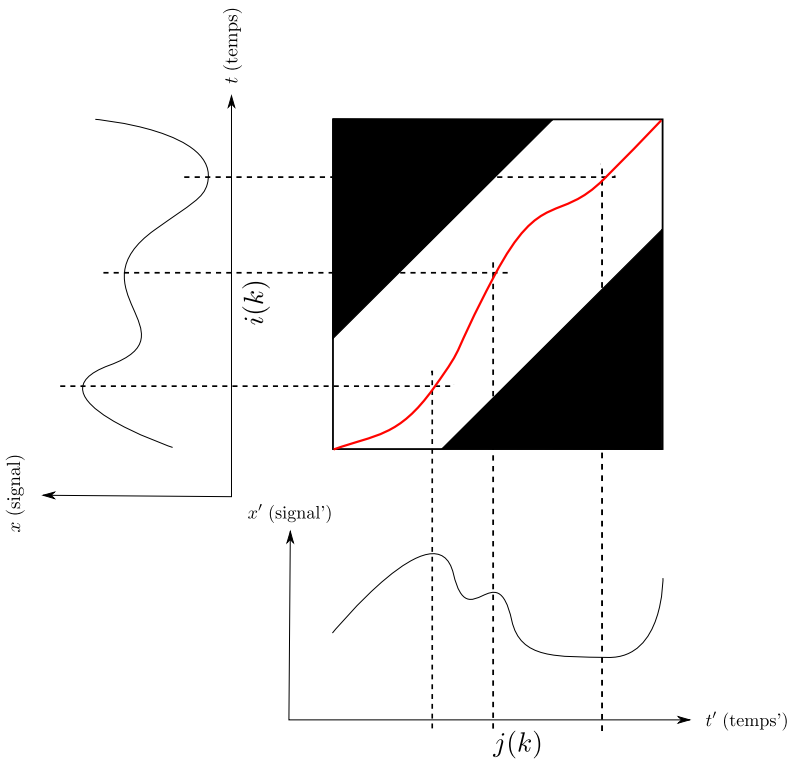




Step 2 : Cumulative distance matrix



Fast computation and variants



- If the timelines of **x** and **y** are assumed to be approximately similar, it is not necessary (nor recommendable) to compute all values of $C(i, j)$
- Constrained variants can be implemented, by for instance only computing $C(i, j)$ if

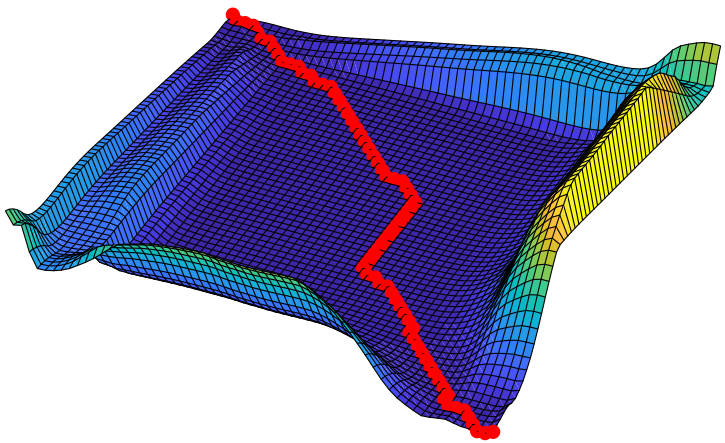
$$|i_k - j_k| < \lambda \text{ (Sakoe-Chiba band)}$$

[Sakoe et al., 1978]

- Other truncations exist such as the Itakura Parallelogram [Itakura, 1975] and depend on the hypothesis on the type of nonlinear timeline variations

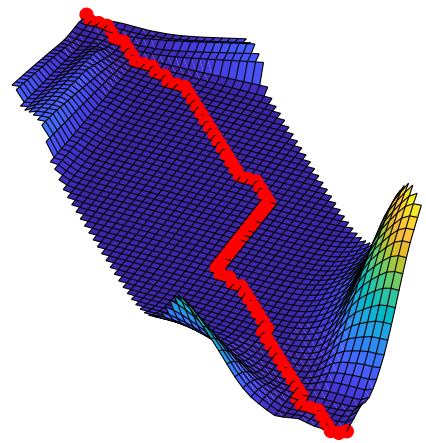
Fast computation and variants

Classic DTW



9 ms

Constrained DTW



2 ms

What influences DTW?

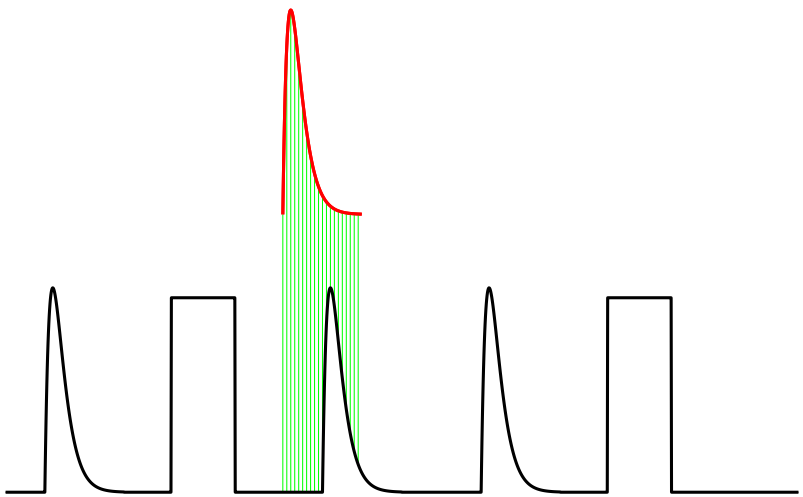
- By definition, DTW is robust to timeline changes
- However, it still needs that the first and last samples are aligned, which can be a limitation in case of time shifts. Some variants exist to avoid this problem.
- DTW is sensitive to scale (can be solved by normalized time series beforehand), noise and outliers

Summary

	Euclidean	Normalized Euclidean	DTW	Normalized DTW
Offset	✗	✓	✗	✓
Amplitude shift	✗	✓	✗	✓
Time shift	✗	✗	≈	≈
Dilatation or contraction	✗	✗	✓	✓
Additive noise	✗	✗	✗	✗
Outlier	✗	✗	✗	✗

Contents

Problem statement



- In order to find a known pattern $\mathbf{p}$ of length N_p in a time series $\mathbf{x}$ of length N, we need to compute all distances

$$d[n] = d(\mathbf{p}, x[n : n + N_p - 1])$$

for $1 \leq n \leq N - N_d + 1$ and where $x[n : n + N_d - 1]$ is the sequence of $\mathbf{x}$ starting at n and of length N_p

- Intuitively, this is a costly operation since the distance needs to be computed approximately N times

Can we compute this quantity with good complexity?

Fast computation for Euclidean distance

$$d[n] = d(\mathbf{p}, x[n : n + N_p - 1])$$

- The sequence $d[n]$ is referred to in the literature as the **distance profile**
- A solution has been found for standard and normalized Euclidean distance. The whole computation is in $\mathcal{O}(N \log N)$ and does not depend on the pattern length N_p [Mueen et al., 2015]
- This solution is based on the properties of FFT (Fast Fourier Transform) that is a fundamental algorithm for computing the Discrete Fourier Transform of a time series in $\mathcal{O}(N \log N)$

Reformulation of the Euclidean distance

- Given a pattern $\mathbf{p}$ of length N_p and a time series $\mathbf{x}$ of length $N > N_p$, we can write

$$d_{EUC}(\mathbf{p}, x[n : n + N_p - 1]) = \sqrt{\sum_{i=n}^{n+N_p-1} x[i]^2 + \sum_{i=1}^{N_p} p[i]^2 - 2\sum_{i=1}^{N_p} x[n + i - 1]p[i]}$$

$$d_{nEUC}(\mathbf{p}, x[n : n + N_p - 1]) = \sqrt{2N_p\left(1 - \frac{\sum_{i=1}^{N_p} x[n + i - 1]p[i] - N_p\mu_x[n]\mu_p}{N_p\sigma_x[n]\sigma_p}\right)}$$

where

$$\mu_x[n] = \frac{1}{N_p}\sum_{i=n}^{n+N_p-1} x[i], \qquad \sigma_x[n] = \sqrt{\frac{1}{N_p}\sum_{i=n}^{n+N_p-1}(x[i] - \mu_x[n])^2}$$

are sliding mean/standard deviations

- All statistics related to $\mathbf{p}$ such as $\sum_{i=1}^{N_p} p[i]^2$, μ_p or σ_p are easy to compute, but those of $x[n : n + N_p - 1]$ change for each index n

Reformulation of the Euclidean distance

- The quantities involving $x[n : n + N_p - 1]$ to be computed are

$$s_x[n] = \sum_{i=n}^{n+N_p-1} x[i]^2$$

$$\mu_x[n] = \frac{1}{N_p} \sum_{i=n}^{n+N_p-1} x[i]$$

$$\sigma_x[n] = \sqrt{\frac{1}{N_p} \sum_{i=n}^{n+N_p-1} (x[i] - \mu_x[n])^2}$$

- In fact, by pre-computing (in linear time) and storing the cumulative terms

$$c_x[n] = \sum_{i=1}^{n} x[i]$$

$$c_x^{(2)}[n] = \sum_{i=1}^{n} x[i]^2$$

all quantities can be computed in a few operations

Reformulation of the Euclidean distance

- The quantities involving $x[n : n + N_p - 1]$ to be computed are

$$s_x[n] = \sum_{i=n}^{n+N_p-1} x[i]^2 \qquad s_x[n] = c_x^{(2)}[n + N_p - 1] - c_x^{(2)}[n - 1]$$

$$\mu_x[n] = \frac{1}{N_p} \sum_{i=n}^{n+N_p-1} x[i] \qquad \mu_x[n] = \frac{1}{N_p} \left(c_x[n + N_p - 1] - c_x[n - 1] \right)$$

$$\sigma_x[n] = \sqrt{\frac{1}{N_p} \sum_{i=n}^{n+N_p-1} (x[i] - \mu_x[n])^2} \qquad \sigma_x[n] = \sqrt{\frac{1}{N_p} s_x[n] - \mu_x[n]^2}$$

- In fact, by pre-computing (in linear time) and storing the cumulative terms

$$c_x[n] = \sum_{i=1}^{n} x[i]$$

$$c_x^{(2)}[n] = \sum_{i=1}^{n} x[i]^2$$

all quantities can be computed in a few operations

Fast computation of the dot product

- The only difficulty is to compute the dot product

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

- We use here two important results from signal processing :
  - Sliding dot product corresponds to the discrete convolution between the time series $\mathbf{x}$ and the time series $\mathbf{p}^{\reflectbox{L}}$ where $\mathbf{p}^{\reflectbox{L}}$ is the time-reverted time series $\mathbf{p}$
  - Convolution in the time domain corresponds to a term-by-term product in the frequency domain. It is always cheaper to compute a convolution product by computing the inverse FFT of the term-by-term product of the FFTs

$$\mathbf{x} * \mathbf{y} = \text{iFFT}\left\{\text{FFT}\left\{\mathbf{x}\right\} \times \text{FFT}\left\{\mathbf{y}\right\}\right\}$$

Fast computation of the dot product

1. Compute the zero-padded time-reversed pattern

$$\mathbf{p}^{\circlearrowleft} = p[N_p]p[N_p - 1] \dots p[1] \underbrace{0 \dots 0}_{N - N_p}$$

2. The dot product can be computed as

$$\mathbf{r}[1 : N - Np + 1] = \text{IFFT}\left\{\text{FFT}\left\{\mathbf{x}\right\} \times \text{FFT}\left\{\mathbf{p}^{\circlearrowleft}\right\}\right\}[N_p : N]$$

Thanks to the fast implementations of FFT, this step is only $\mathcal{O}(N \log N)$

Final computation

- Both computations of sliding Euclidean distance and normalized Euclidean distance can be computed in $\mathcal{O}(N \log N)$
- Normalized Euclidean distance is not more computationally demanding than standard Euclidean distance: the only difference lies in the computation of $\mu_x[n]$ which is linear
- With $N_p = 10^2$ and $N = 10^8$, time computation is only 16 seconds on standard computers

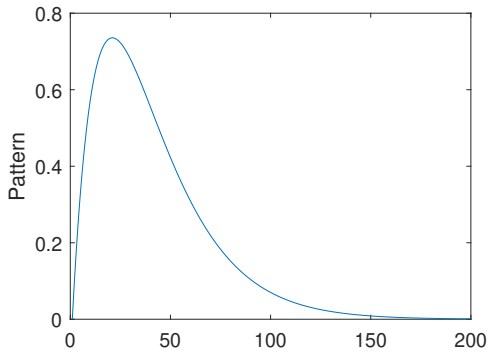
Normalized Euclidean distance profile

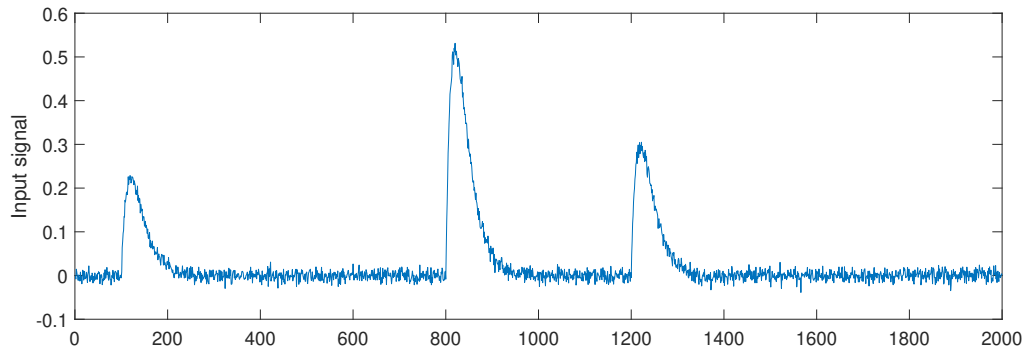
Algorithm 2: Normalized Euclidean distance profile

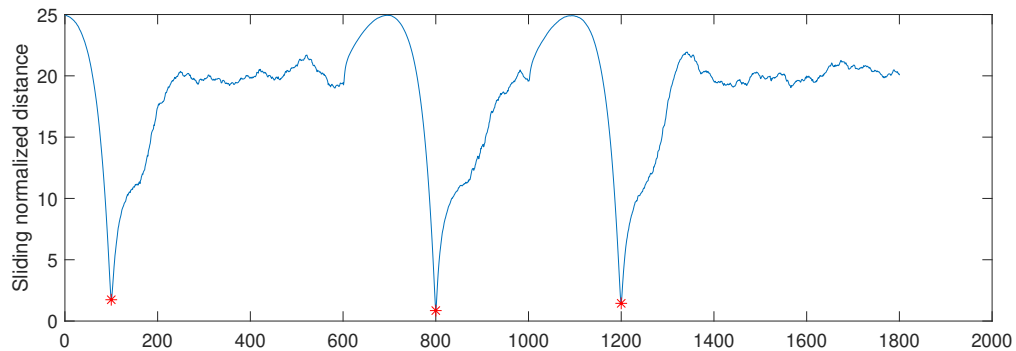
Inputs : Time series $\mathbf{x} \in \mathbb{R}^N$ and pattern $\mathbf{p} \in \mathbb{R}^{N_p}$
Output: Normalized Euclidean distance profile $\mathbf{d} \in \mathbb{R}^{N-N_p+1}$
Normalize $\mathbf{p}$ (zero-mean, unit variance) and compute $\mathbf{p}^{\curvearrowleft} = p[N_p]p[N_p-1]\dots p[1]\underbrace{0\dots 0}_{N-N_p}$;

$\mathbf{c_1} = \mathbf{0}_{N+1}$; $\mathbf{c_2} = \mathbf{0}_{N+1}$;
for $2 \leq i \leq N+1$ **do**
  $c_1(i) = c_1(i-1) + x[i-1]$;
  $c_2(i) = c_2(i-1) + x[i-1]^2$;
end
$\mathbf{r} = \text{iFFT}\left\{\text{FFT}\left\{\mathbf{x}\right\} \times \text{FFT}\left\{\mathbf{p}^{\curvearrowleft}\right\}\right\}$;
$\mathbf{s} = c_2(N_p+1:N+1) - c_2(1:N-N_p+1)$;
$\boldsymbol{\mu} = \left(c_1(N_p+1:N+1) - c_1(1:N-N_p+1)\right)/N_p$;
$\boldsymbol{\sigma} = \sqrt{\mathbf{s}/N_p - \boldsymbol{\mu}^2}$;
$\mathbf{d} = \sqrt{2\left(N_p - r(N_p:N)/\boldsymbol{\sigma}\right)}$;

Results







Patterns can easily be retrieved in the signal by searching for peaks with small distance values

Fast computation for DTW

- The problem is more complex for DTW and will be reformulated as follows :

$$\text{Find } d^* = \min_{1 \leq n \leq N - N_d + 1} d_{DTW}(\mathbf{p}, x[n : n + N_p - 1])$$

- Instead of computing all DTW distances, we are searching for the minimal DTW between the pattern and all subsequences
- Similarly to Euclidean distance, a trick with pre-computed cumulative sums can be used to compute the DTW between renormalized signals (zero mean, unit variance), in order to be robust to amplitude and offsets.
- A very fast solution exist, that is based on several acceleration tricks [Rakthanmanon et al., 2012], one of them being the existence of a lower bound

Finding the minimal DTW distance

The problem here is different than for Euclidean distance: we are only interested in the minimal distance:

- **First trick :** use the lower bound for DTW. If we know that $d_{DTW}(\mathbf{x}, \mathbf{y}) \geq LB(\mathbf{x}, \mathbf{y})$ and that LB is cheap to compute, then when LB is larger than the current minimal distance, it is not worth computing the DTW.
- **Second trick :** even when computing a lower bound LB (which is based on the Euclidean distance - see next slides) or the DTW we can stop when the sum on the first samples exceeds the current minimal distance
- **Other tricks :** other tricks (such as using a pruned computation for the normalization, re-ordering the sequences or using cascade lower bounds) can help to avoid more than 99.9999 % of DTW computations

Lower bounds for DTW

LB_{KimFL} [Kim et al., 2001]

- Let $\mathbf{x}$ and $\mathbf{y}$ be two time series of respective lengths M and N
- Knowing that the first and last points of sequences should be aligned in DTW, i.e. $(i_1, j_1) = (1, 1)$ and $(i_{K_P}, j_{K_P}) = (M, N)$ we know that

$$\begin{aligned} d^2_{DTW}(\mathbf{x}, \mathbf{y}) &= \min_{P \in \mathcal{P}} \sum_{k=1}^{K_P} (x[i_k] - y[j_k])^2 \\ &\geq (x[1] - y[1])^2 + (x[M] - y[N])^2 \end{aligned}$$

$$LB_{KimFL}(\mathbf{x}, \mathbf{y}) = \sqrt{(x[1] - y[1])^2 + (x[M] - y[N])^2}$$

- This naive lower bound is easy to compute and can provide a first rough estimation

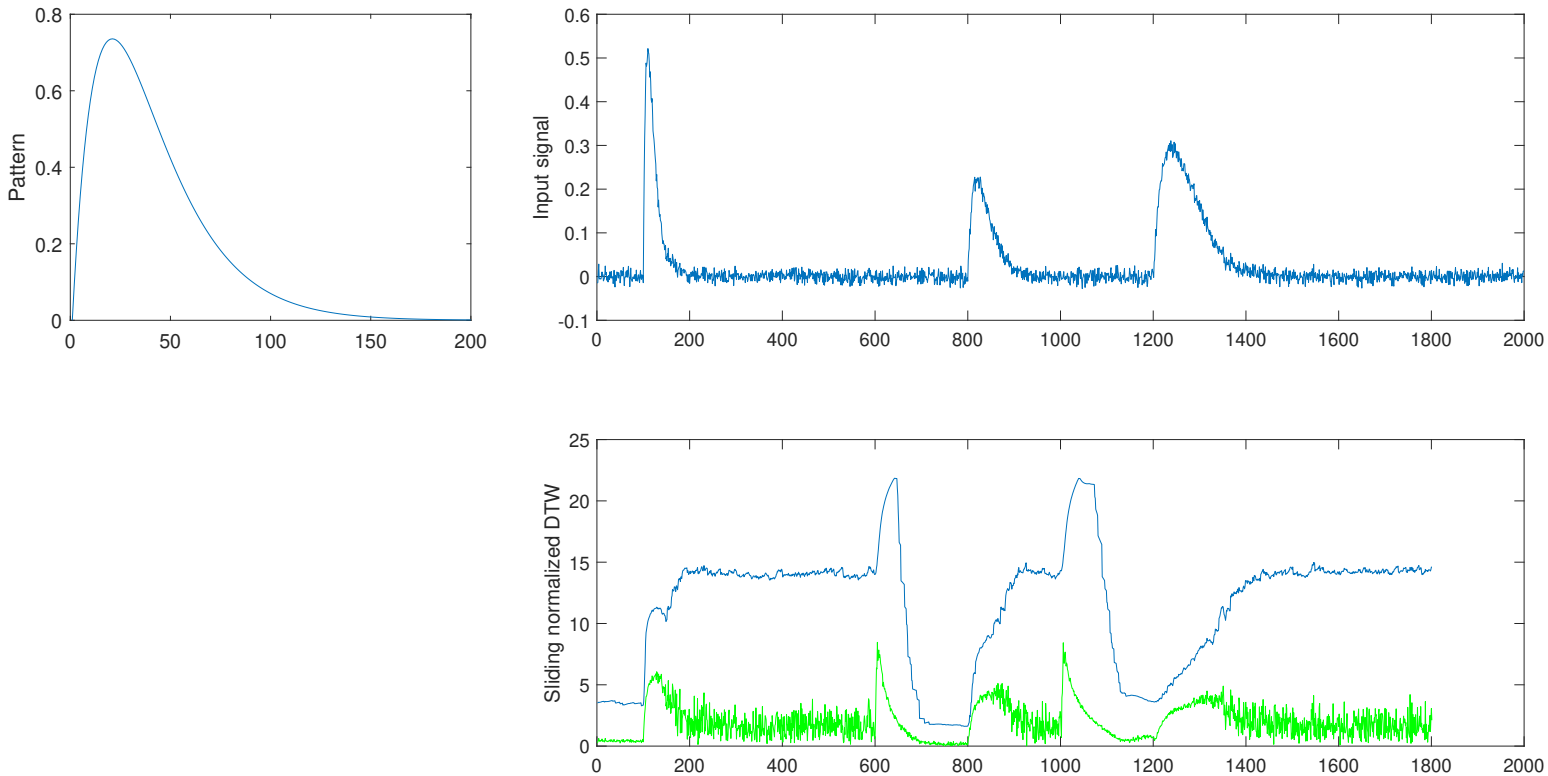
How to use the lower bound property to avoid DTW computations ?

1. Given a pattern $\mathbf{p}$ of length N_p and a time series $\mathbf{x}$ of length $N > N_p$ we first compute $LB_{KimFL}(\mathbf{p}, x[n : n + N_p - 1])$ for each n (it is very cheap !)
2. Then, we process iteratively.
   - 2.1 **Initialization.** We define the set $\mathcal{N}$ of acceptable samples (initialized as the whole $1 \ldots N$ interval), and we initialize the minimum distance value d^* as $+\infty$
   - 2.2 We select (randomly or uniformly) one or a few samples $n_1, n_2, \ldots$ in $\mathcal{N}$ for which we compute the real DTW values : $\text{DTW}(\mathbf{p}, x[n : n + N_p - 1])$ and we update the current minimum distance value d^*
   - 2.3 All samples n for which

$$LB_{KimFL}(\mathbf{p}, x[n : n + N_p - 1]) > d^*$$

     are removed from $\mathcal{N}$
   - 2.4 Reiterate the two previous steps

Results



By computing only one DTW (the first one), 10% of the computations can be avoided (blue = true DTW, green = LB_{KimFL})

Lower bounds for DTW

LB_{Keogh} [Keogh et al., 2004]

- Let **x** and **y** be two time series of respective lengths M and N
- Knowing that we use a constrained DTW with width λ, we know that sample i_k cannot be associated to j_k if $|i_k - j_k| > \lambda$
- Given a signal **x**, we can compute the sliding max/min on the window of possible matches

$$u[j] = \max_{j-\lambda \leq i \leq j+\lambda} x[i] \qquad l[j] = \min_{j-\lambda \leq i \leq j+\lambda} x[i]$$

- Then, using the constrained DTW assumption, we have

$$l[j_k] \leq x[i_k] \leq u[j_k]$$

since $j_k - \lambda \leq i_k \leq j_k + \lambda$

- We therefore have

$$l[j_k] - y[j_k] \leq x[i_k] - y[j_k] \leq u[j_k] - y[j_k]$$

Lower bounds for DTW

LB_{Keogh} [Keogh et al., 2004]

$$l[j_k] - y[j_k] \leq x[i_k] - y[j_k] \leq u[j_k] - y[j_k]$$

- Then, if $y[j_k] < l[j_k]$ we know that

$$(x[i_k] - y[j_k])^2 > (l[j_k] - y[j_k])^2$$

- Similarly when $y[j_k] > u[j_k]$ we know that

$$(x[i_k] - y[j_k])^2 > (u[j_k] - y[j_k])^2$$

- In other cases, we only know that

$$(x[i_k] - y[j_k])^2 \geq 0$$

- We have therefore a lower bound for each $(x[i_k] - y[j_k])^2$, that only depends on the comparison of $y[j_k]$ to $l[j_k]$ or $u[j_k]$

Lower bounds for DTW

LB_{Keogh} [Keogh et al., 2004]

- Knowing that

$$d^2_{DTW}(\mathbf{x}, \mathbf{y}) = \min_{P \in \mathcal{P}} \sum_{k=1}^{K_P} (x[i_k] - y[j_k])^2$$

- We have that

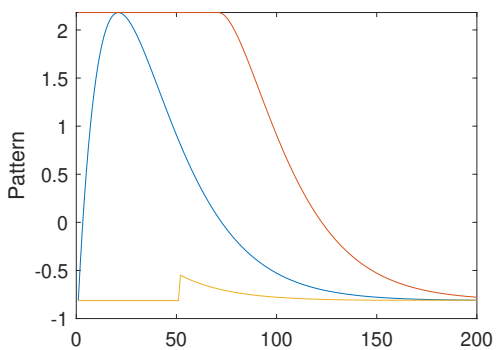
$$d^2_{DTW}(\mathbf{x}, \mathbf{y}) \geq \sum_{\substack{1 \leq j \leq N \\ y[j] < l[j]}} (l[j] - y[j])^2 + \sum_{\substack{1 \leq j \leq N \\ y[j] > u[j]}} (u[j] - y[j])^2$$

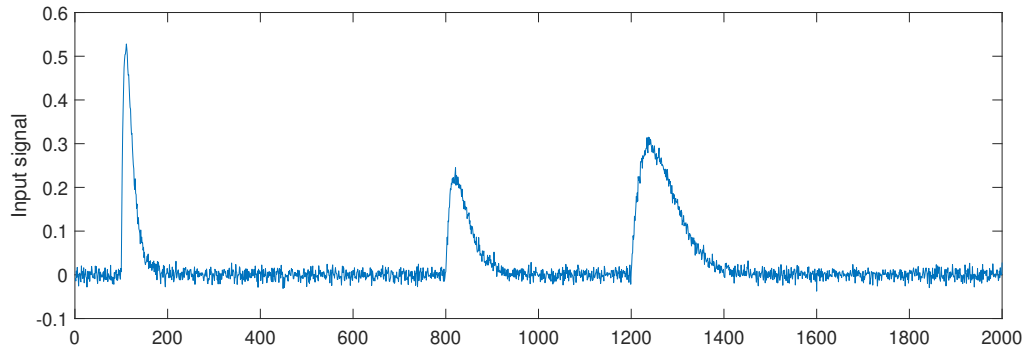
- And finally a new lower bound

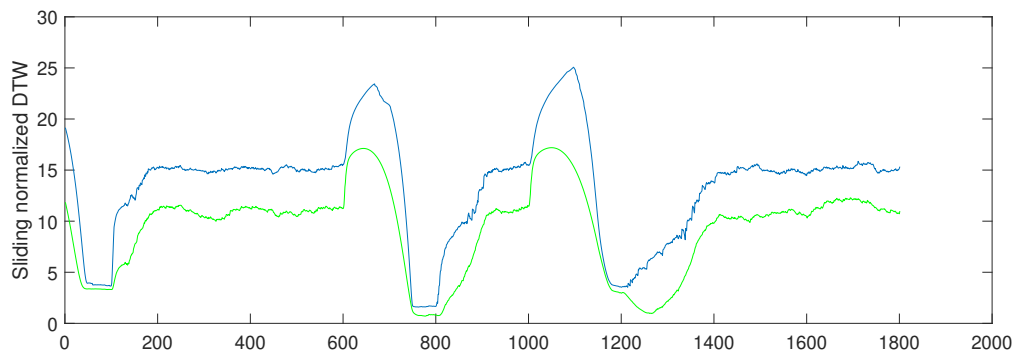
$$LB_{Keogh}(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{\substack{1 \leq j \leq N \\ y[j] < l[j]}} (l[j] - y[j])^2 + \sum_{\substack{1 \leq j \leq N \\ y[j] > u[j]}} (u[j] - y[j])^2}$$

- This lower bound can be used exactly like the previous lower bound

Results







($\lambda = 50$). By computing only the 50 first DTW, 80% of the computations can be avoided (blue = true DTW, green = LB_{Keogh})

Contents

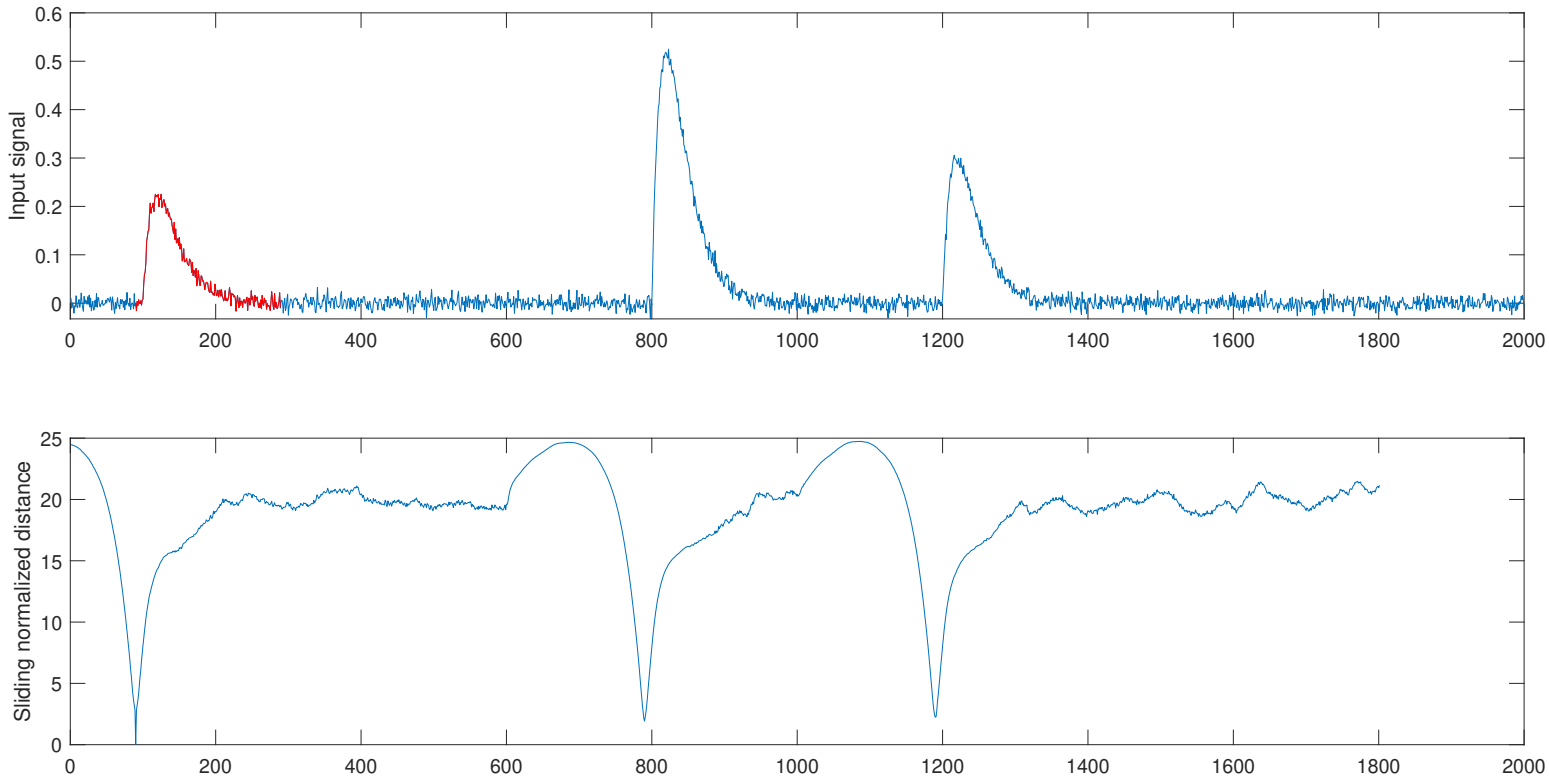
Pattern extraction

- We have seen methods to search for patterns in a time series: these methods need a predefined dictionary of templates
- In practice, one can be interested in the reverse question: how can I detect and extract patterns from a time series ?
- Unsupervised task: no prior knowledge except for the average duration of the researched patterns L
- **What is a pattern ?** Difficult question : repetitive shapes, notion of periodicity, etc.

Distance-based pattern extraction

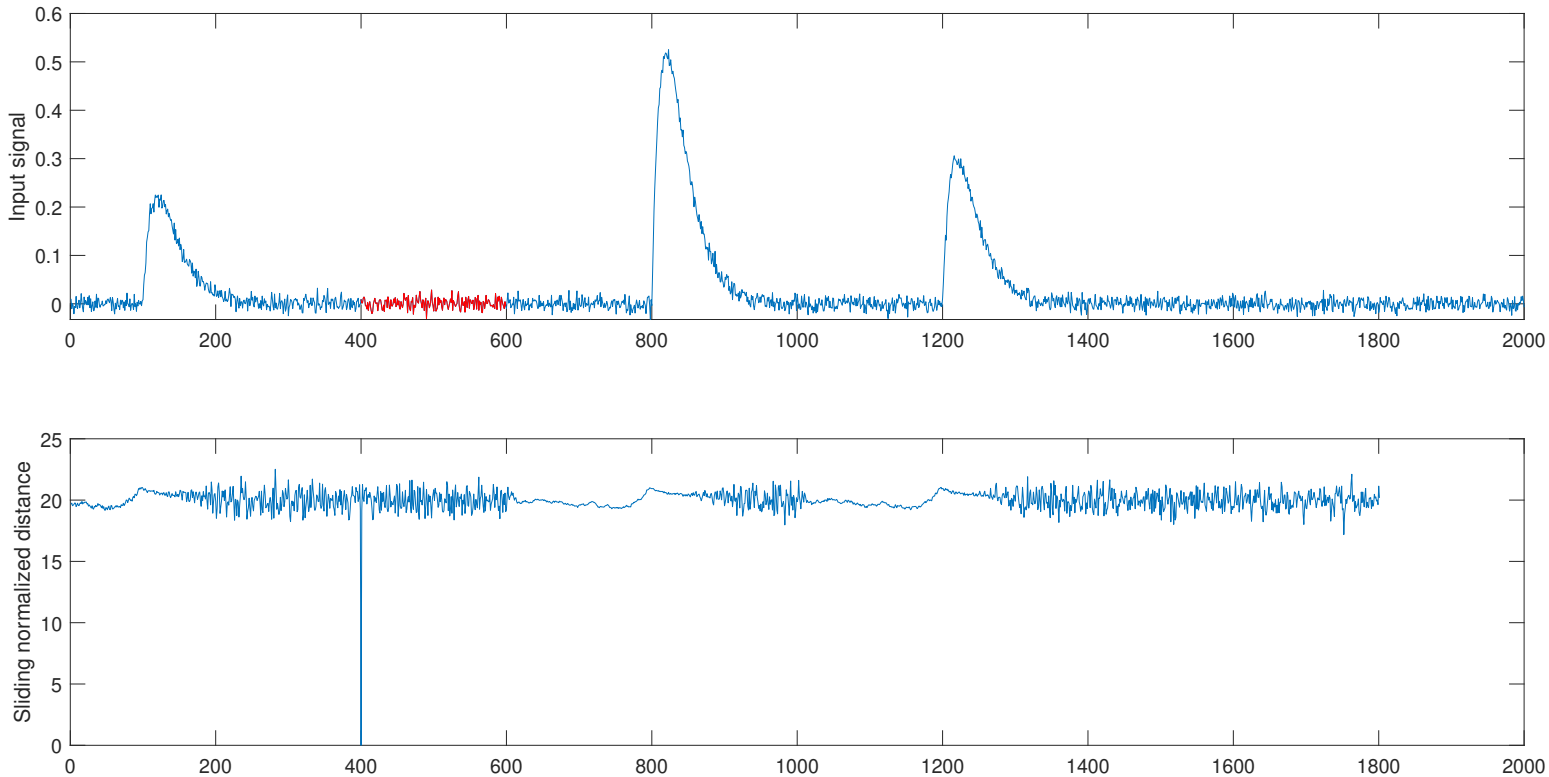
- Intuitively, we can re-use results from the previous part to automatically detect patterns
- A pattern is a subsequence that is likely to be found several times in the whole time series
- By computing a sliding distance between this subsequence and all subsequences in the time series, it should appear clearly that one (or several) others subsequences are very *close*
- **Solution:** use a brute-force algorithm to efficiently compute all distances?

Example



Here, we computed the distance profile of the sequence displayed in red: it is clear that this sequence appears 3 times in the time series. It might be a pattern!

Example



Here, we computed the distance profile of the sequence displayed in red: no obvious structure in the distance profile (except for the exact correspondence): unlikely to be a pattern!

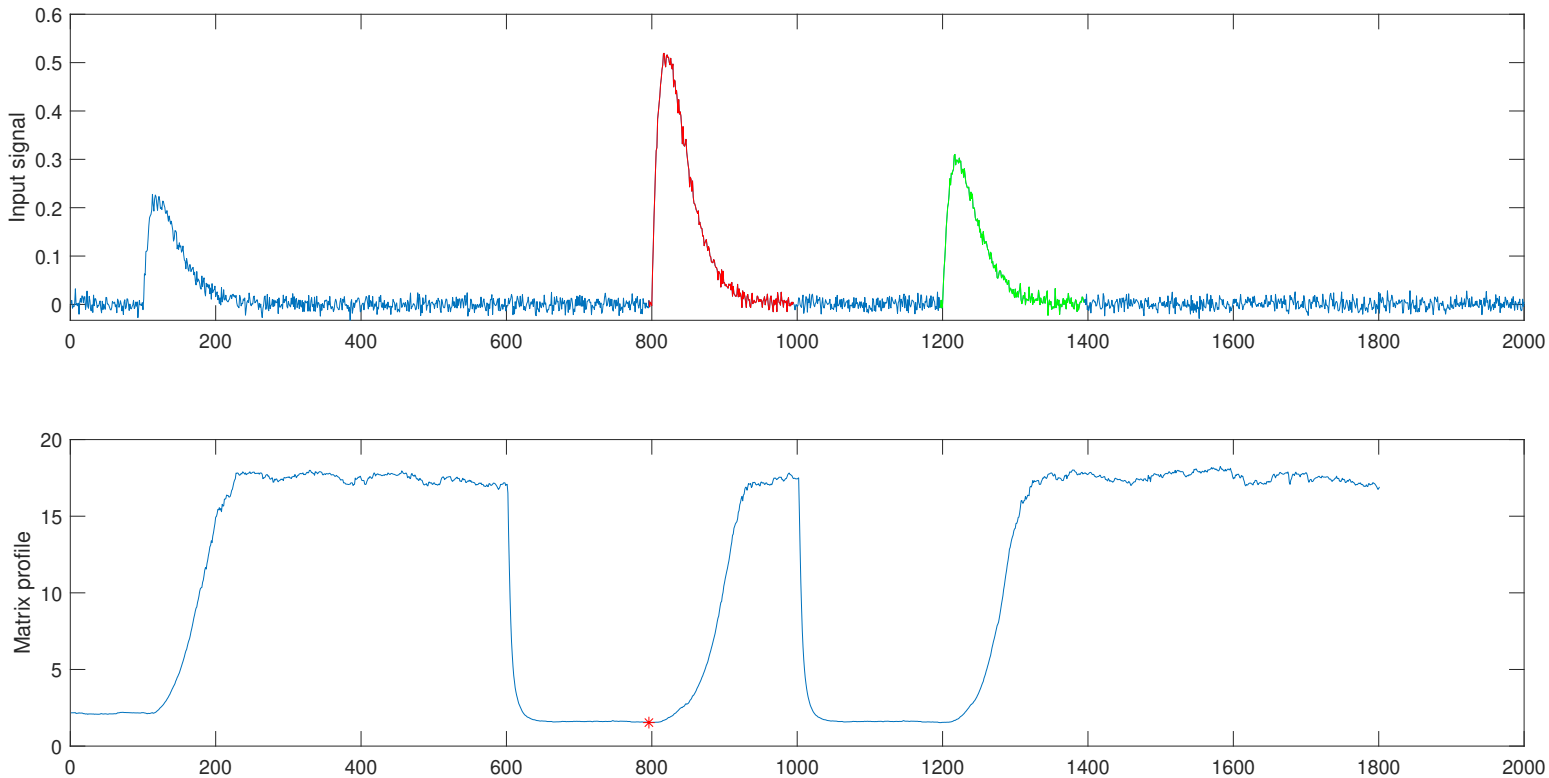
Matrix profile

- In fact, patterns are detectable by only looking at the minimal distance with all subsequences
- Beware of trivial matches ! Only compare with subsequences that do not overlap with the subsequence of interest
- **Matrix profile** [Yeh et al., 2016] : given a pattern length L, compute

$$m[n] = \min_{i>n+L \text{ or } i<n-L} d(x[n : n+L-1], x[i : i+L-1])$$

- Small matrix profiles values indicate that the subsequence has been found elsewhere in the time series, suggesting that it could be a pattern
- Efficient computation with techniques already mentioned (fast sliding distance computation)

Example



In red : subsequence with minimal matrix profile value. In green : closest subsequence (according to the normalized Euclidean distance)

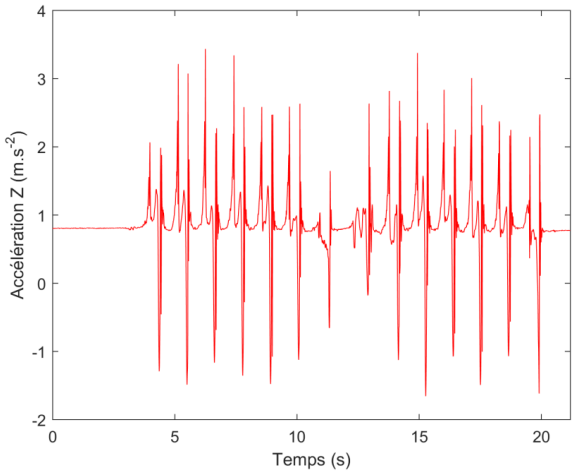
Dictionary-based pattern extraction

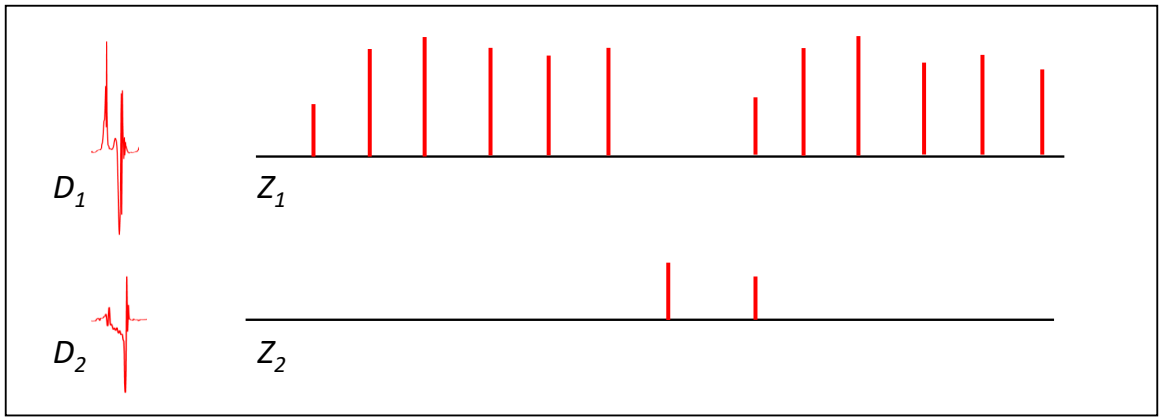
- Another approach consists in formulating this problem as a dictionary learning optimization problem
- Given an input time series $\mathbf{x}$, learn a dictionary of K patterns $\mathbf{d}_k$ of length L
- These patterns can be activated : activations $\mathbf{z}_k$ of length $N - L + 1$

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

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

Convolutional dictionary learning





Convolutional dictionary learning

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

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

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

Optimization problem

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

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

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

Alternated resolution

Alternated resolution of two subproblems

Dictionary learning

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

Convolutional sparse coding

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

Alternated resolution

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

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

Formulation of the gradients

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

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

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

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

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

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

Proximal operators

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

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

Projection on the unit ball

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

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

Soft thresholding operator

Dictionary learning

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

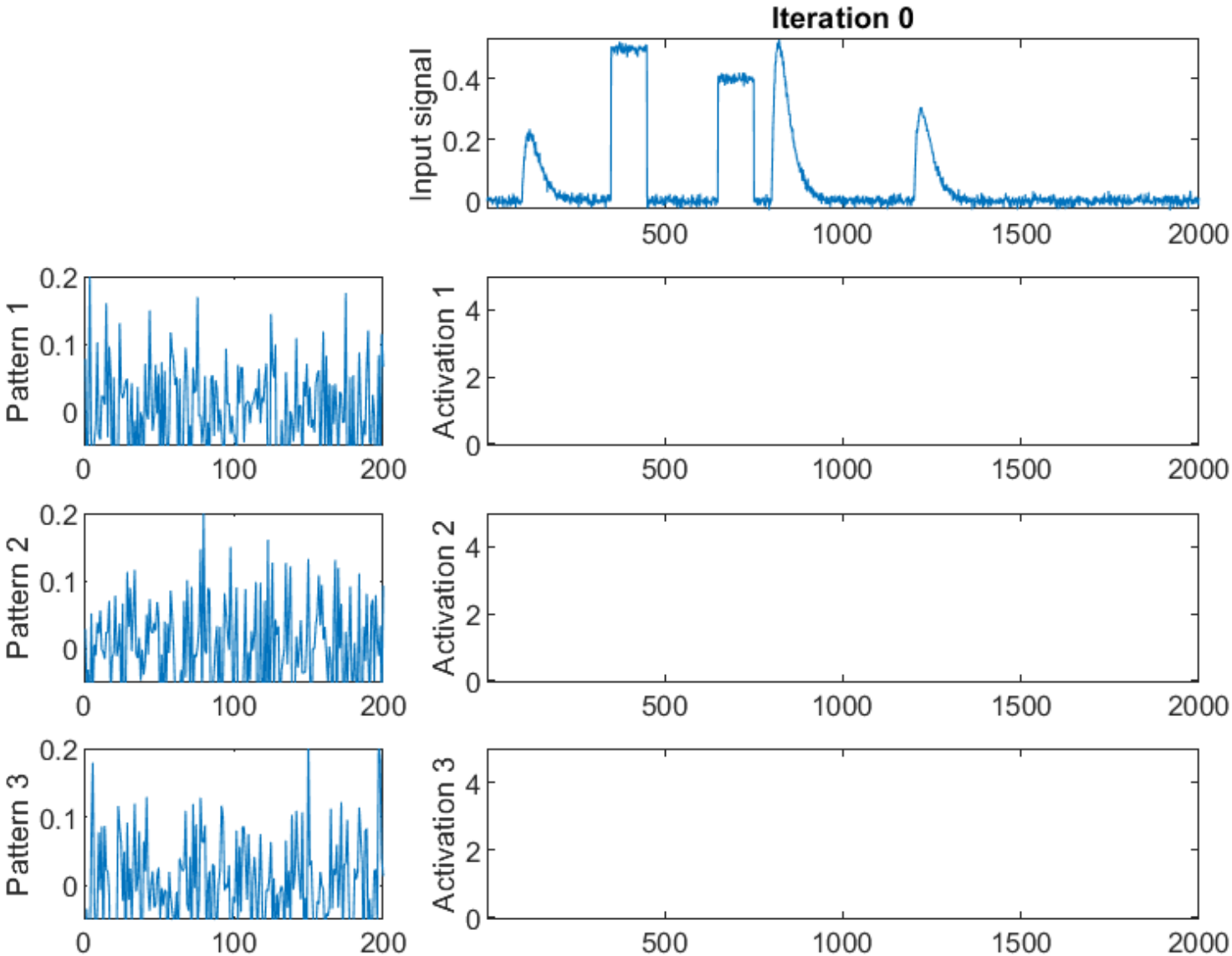
- Basic approach : **Proximal Gradient Descent** with fixed $\mathbf{Z}$
  1. Gradient step :
  $$\mathbf{D} \leftarrow \mathbf{D} - \alpha \nabla_{\mathbf{D}} f(\mathbf{Z}, \mathbf{D})$$
  2. Proximal projection step :
  $$\mathbf{d}_k \leftarrow \text{proj}_{\|.\|_2^2 \leq 1}(\mathbf{d}_k) = \frac{\mathbf{d}_k}{\max(\|\mathbf{d}_k\|_2^2,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

Convolutional sparse coding

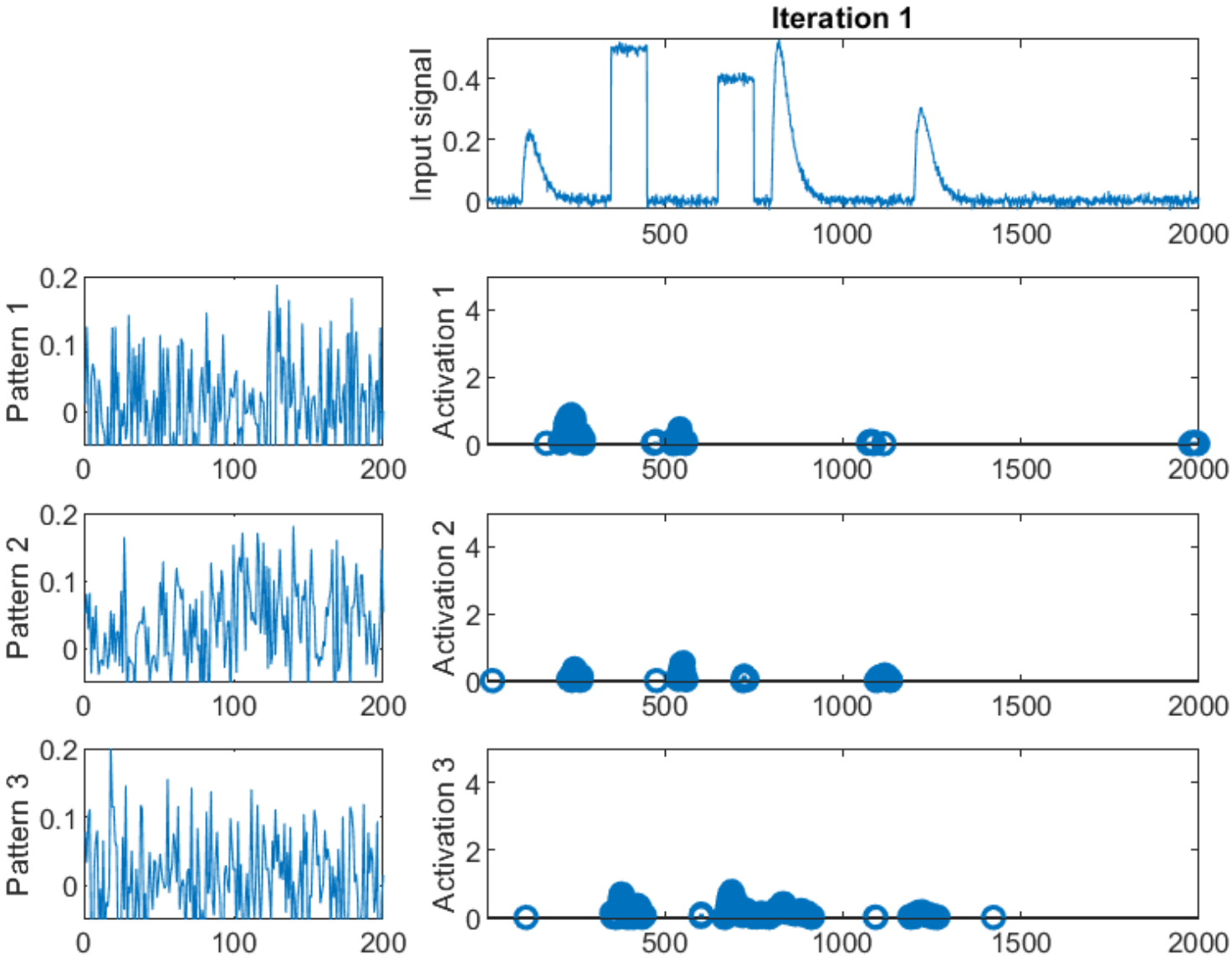
$$\mathbf{Z}^* = \underset{\mathbf{Z}=(\mathbf{z}_1,\dots,\mathbf{z}_K)}{\operatorname{argmin}} \underbrace{\left\| \mathbf{x} - \sum_{k=1}^{K} \mathbf{z}_k * \mathbf{d}_k \right\|_2^2}_{f(\mathbf{Z},\mathbf{D})} + \underbrace{\lambda \sum_{k=1}^{K} \|\mathbf{z}_k\|_1}_{\psi(\mathbf{Z})}$$

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

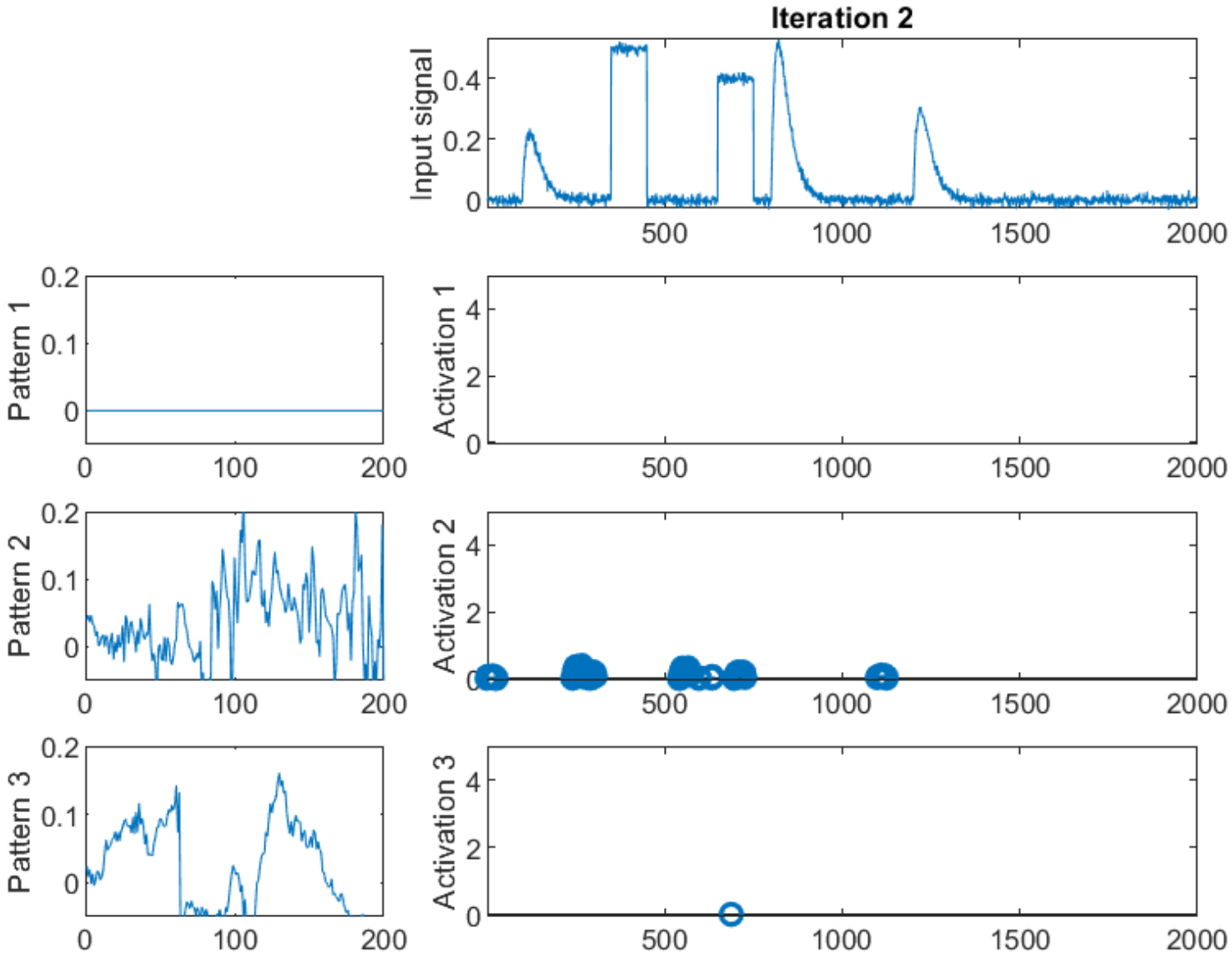
Results


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

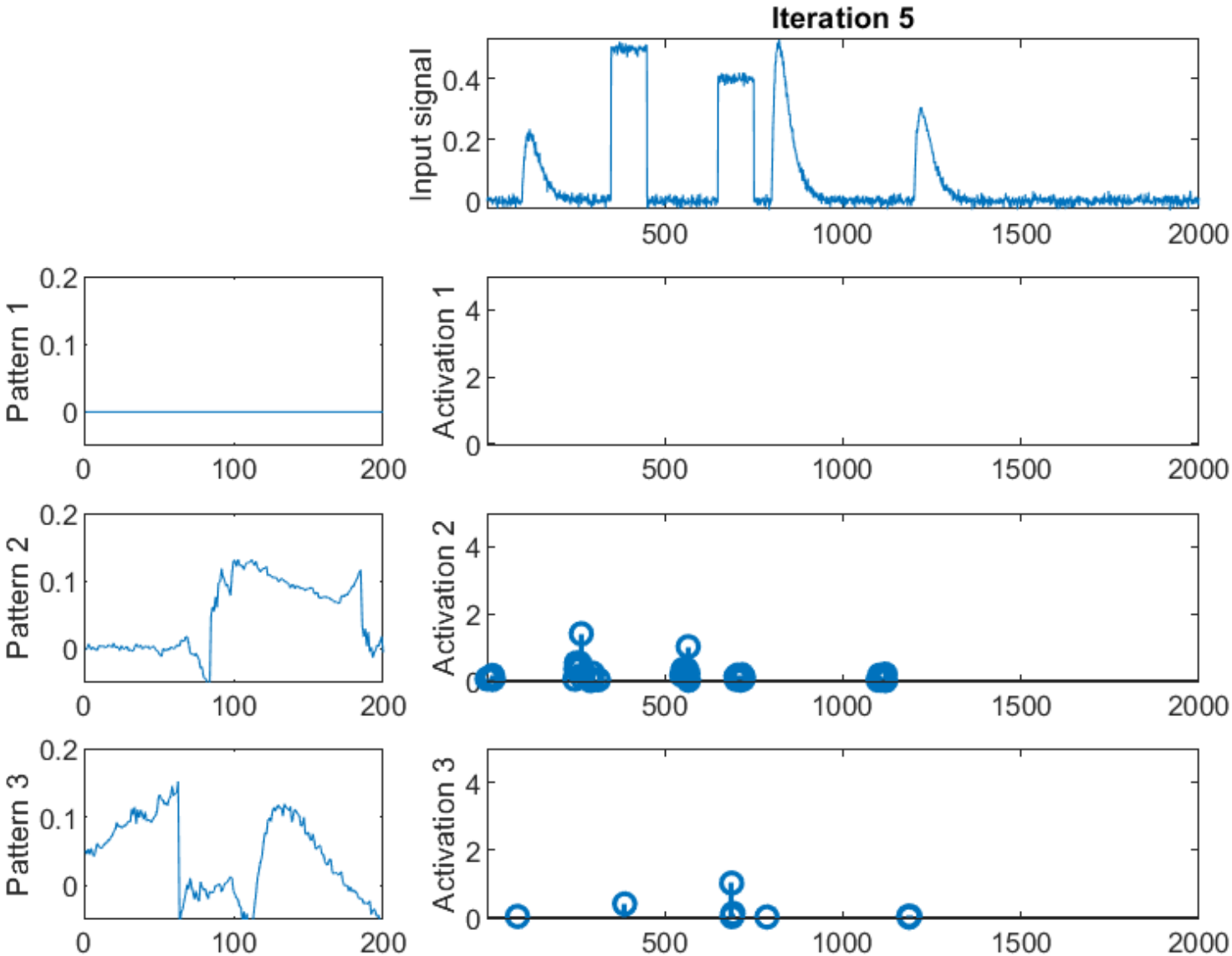
Results


Iteration 1
Input signal
Pattern 1
Activation 1
Pattern 2
Activation 2
Pattern 3
Activation 3

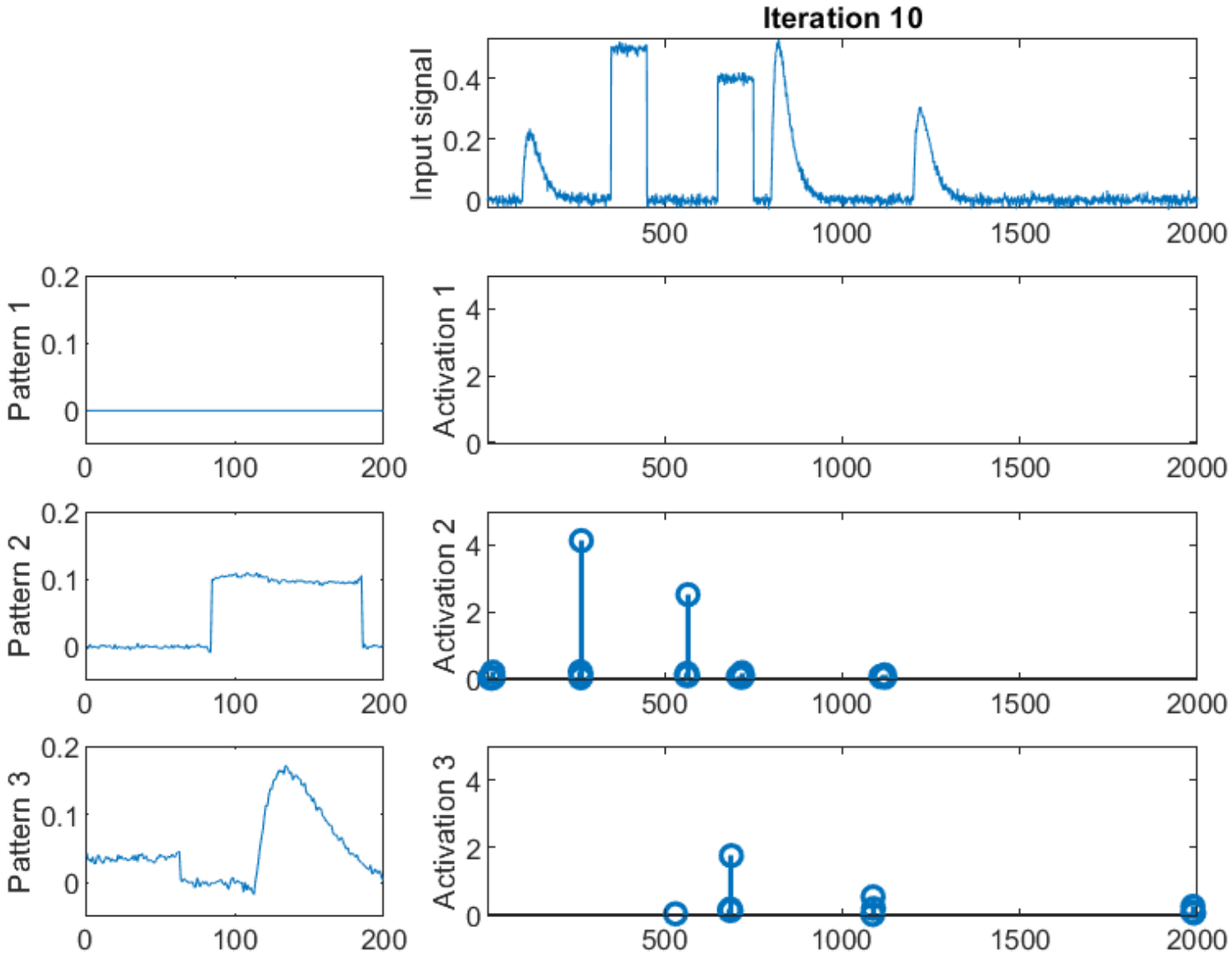
Results


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

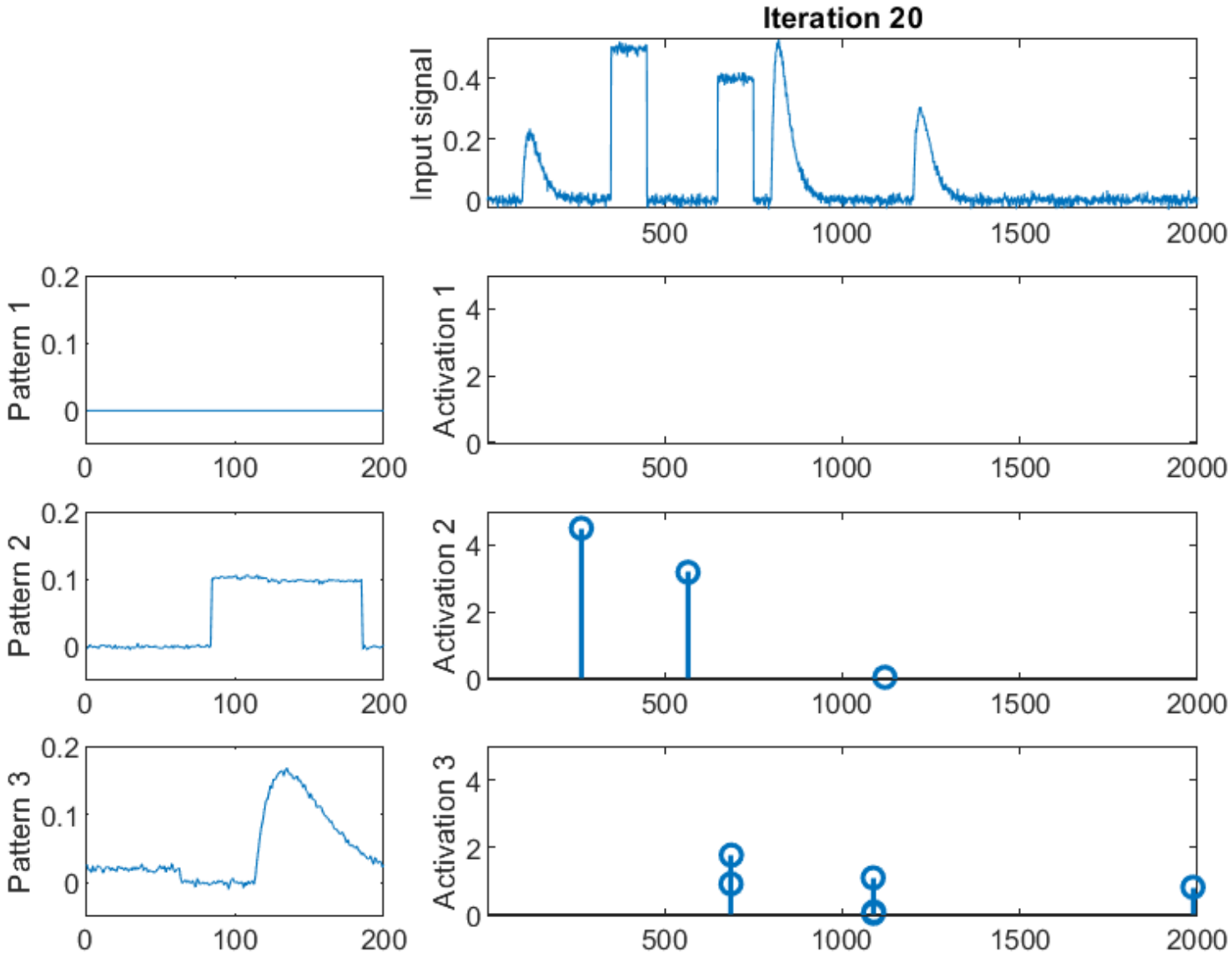
Results


Iteration 5
Input signal
Pattern 1
Activation 1
Pattern 2
Activation 2
Pattern 3
Activation 3

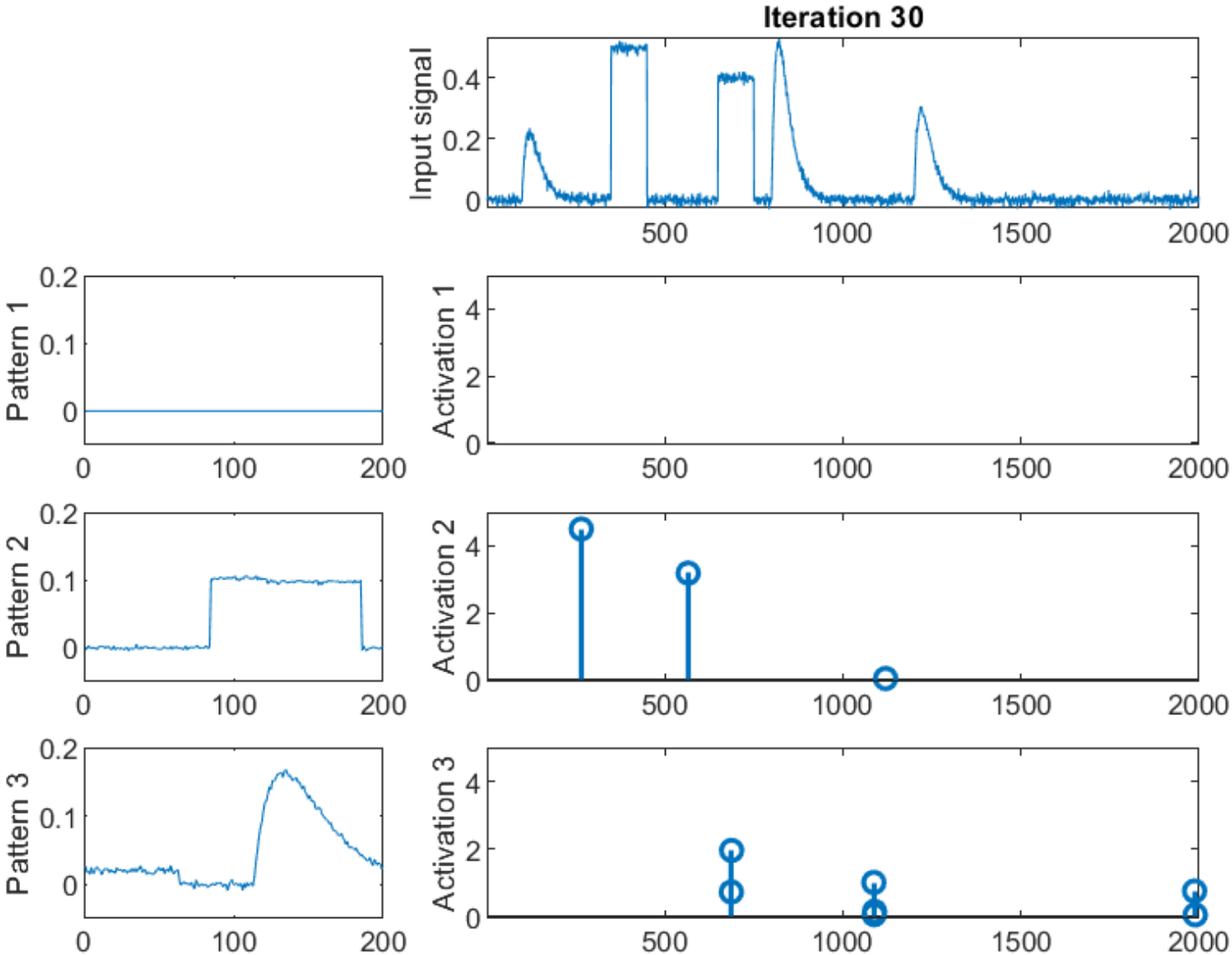
Results


Iteration 10
Input signal
Pattern 1
Activation 1
Pattern 2
Activation 2
Pattern 3
Activation 3

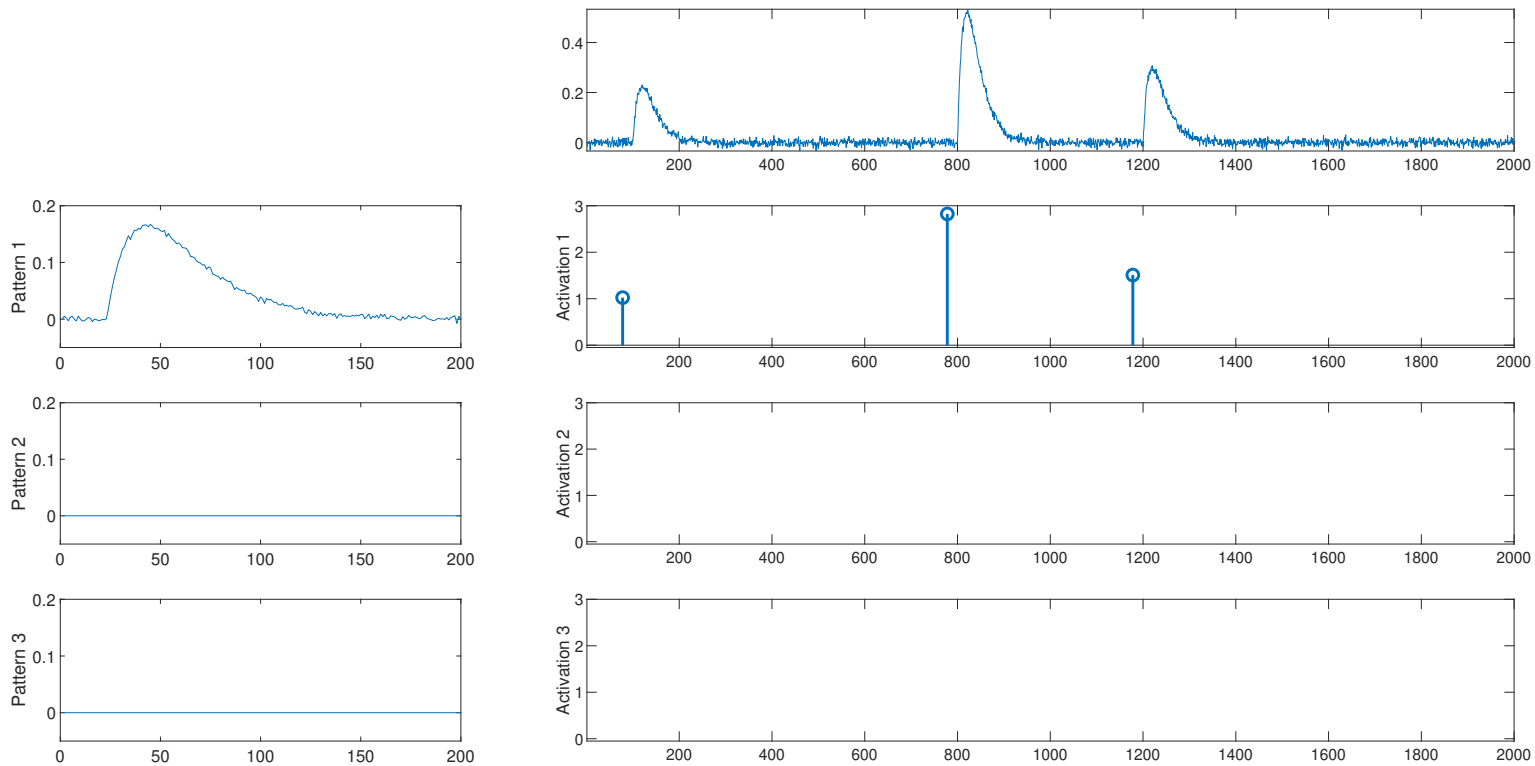
Results


Iteration 20
Input signal
Pattern 1
Activation 1
Pattern 2
Activation 2
Pattern 3
Activation 3

Results

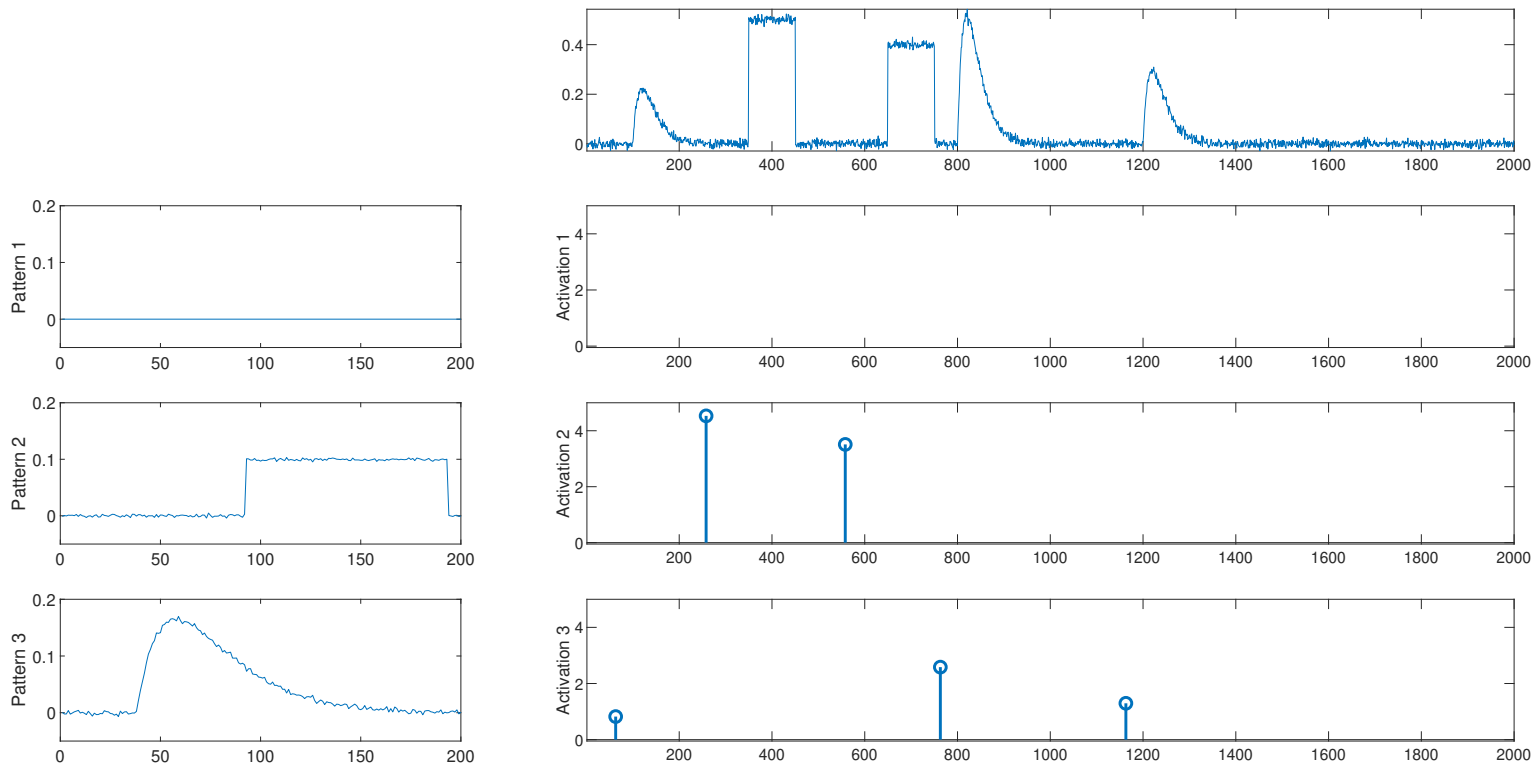

Iteration 30
Input signal
Pattern 1
Activation 1
Pattern 2
Activation 2
Pattern 3
Activation 3

Results



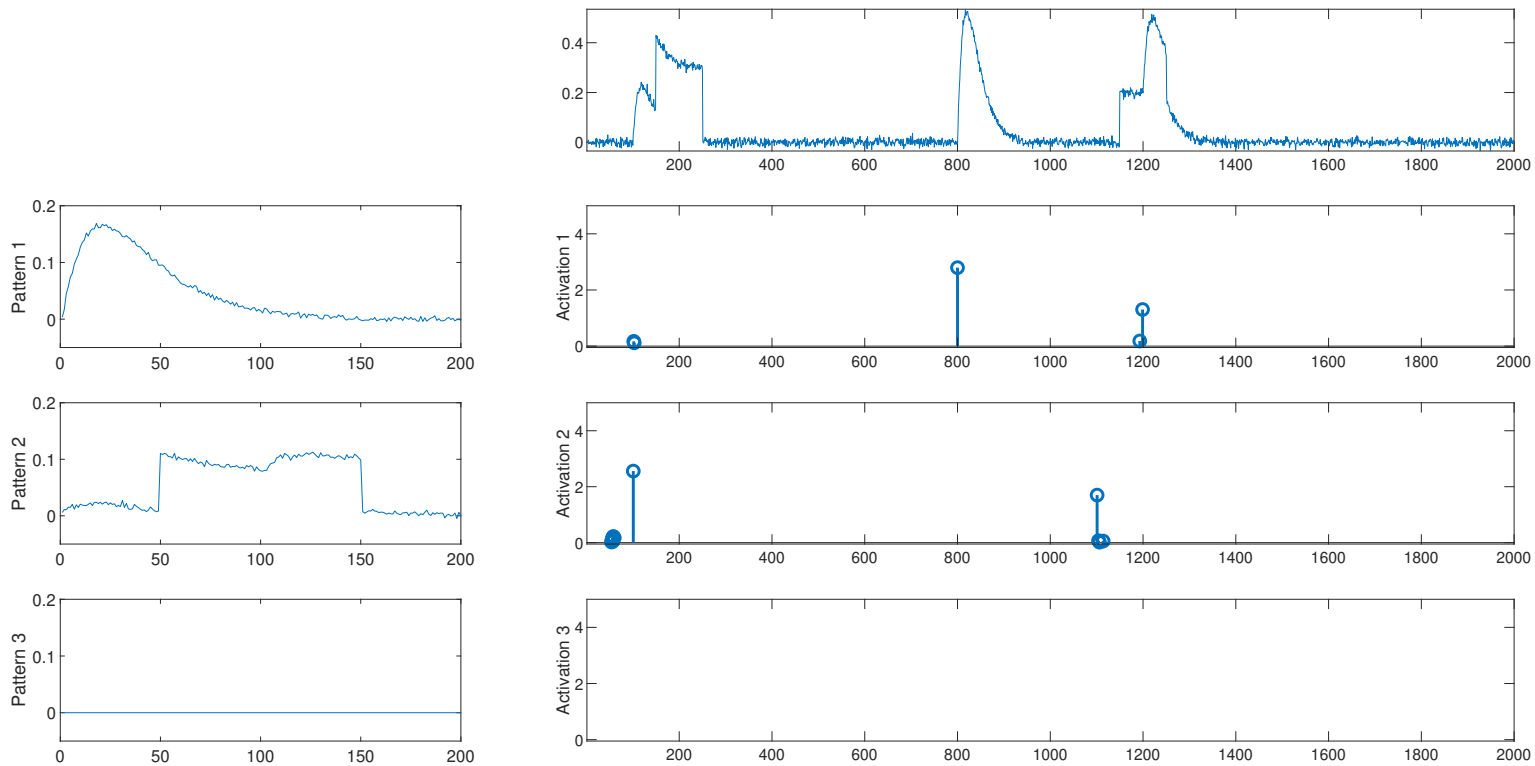
One pattern

Results



Two patterns

Results



Mixture of patterns

Summary

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

References

- Berndt, D. J., & Clifford, J. (1994, July). Using dynamic time warping to find patterns in time series. In KDD workshop (Vol. 10, No. 16, pp. 359-370).
- Sakoe, H., & Chiba, S. (1978). Dynamic programming algorithm optimization for spoken word recognition. IEEE transactions on acoustics, speech, and signal processing, 26(1), 43-49.
- Itakura, F. (1975). Minimum prediction residual principle applied to speech recognition. IEEE Transactions on acoustics, speech, and signal processing, 23(1), 67-72.
- Mueen, A., Viswanathan, K., Gupta, C. K., & Keogh, E. (2015). The fastest similarity search algorithm for time series subsequences under Euclidean distance. http://www.cs.unm.edu/~mueen/FastestSimilaritySearch.html
- Rakthanmanon, T., Campana, B., Mueen, A., Batista, G., Westover, B., Zhu, Q., ... & Keogh, E. (2012, August). Searching and mining trillions of time series subsequences under dynamic time warping. In Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 262-270).
- Kim, S. W., Park, S., & Chu, W. W. (2001, April). An index-based approach for similarity search supporting time warping in large sequence databases. In Proceedings 17th International Conference on Data Engineering (pp. 607-614). IEEE.
- Keogh, E., & Ratanamahatana, C. A. (2004). Exact indexing of dynamic time warping. Knowledge and information systems, 7(3), 358-386.
- Yeh, C. C. M., Zhu, Y., Ulanova, L., Begum, N., Ding, Y., Dau, H. A., ... & Keogh, E. (2016, December). Matrix profile I: all pairs similarity joins for time series: a unifying view that includes motifs, discords and shapelets. In 2016 IEEE 16th international conference on data mining (ICDM) (pp. 1317-1322). Ieee.
- Grosse, R., Raina, R., Kwong, H., & Ng, A. Y. (2007, July). Shift-invariant sparse coding for audio classification. In Proceedings of the Twenty-Third Conference on Uncertainty in Artificial Intelligence (pp. 149-158).
- Wohlberg, B. (2014, May). Efficient convolutional sparse coding. In 2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (pp. 7173-7177). IEEE.
- Mairal, J., Bach, F., Ponce, J., & Sapiro, G. (2010). Online learning for matrix factorization and sparse coding. Journal of Machine Learning Research, 11(1).
- 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.

List of possible topics/projects

- Cuturi, M., & Blondel, M. (2017, July). Soft-dtw: a differentiable loss function for time-series. In International conference on machine learning (pp. 894-903). PMLR.
  A differentiable DTW linked to optimal transport
- Zhao, J., & Itti, L. (2018). shapeDTW: Shape dynamic time warping. Pattern Recognition, 74, 171-184.
  A variant of the DTW that takes local behavior into account
- Le Guen, V., & Thome, N. (2019). Shape and time distortion loss for training deep time series forecasting models. Advances in neural information processing systems, 32.
  How to construct a DTW-based loss for deep learning
- Rakthanmanon, T., & Keogh, E. (2013, May). Fast shapelets: A scalable algorithm for discovering time series shapelets. In proceedings of the 2013 SIAM International Conference on Data Mining (pp. 668-676). Society for Industrial and Applied Mathematics.
  How to automatically extract shapes from time series by using symbolic signal representation.
- Linardi, M., Zhu, Y., Palpanas, T., & Keogh, E. (2020). Matrix profile goes MAD: variable-length motif and discord discovery in data series. Data Mining and Knowledge Discovery
  How to extend the matrix profile approach to variable lengths motifs.
- Yeh, C. C. M., Kavantzas, N., & Keogh, E. (2017, November). Matrix profile vi: meaningful multidimensional motif discovery. In 2017 IEEE international conference on data mining (ICDM) (pp. 565-574). IEEE.
  How to extend the matrix profile approach to multivariate time series
- Alaee, S., Kamgar, K., & Keogh, E. (2020). Matrix Profile XXII: Exact Discovery of Time Series Motifs under DTW. arXiv preprint arXiv:2009.07907.
  How to find patterns using the DTW.
- Hills, J., Lines, J., Baranauskas, E., Mapp, J., & Bagnall, A. (2014). Classification of time series by shapelet transformation. Data Mining and Knowledge Discovery, 28(4), 851-881.
  How to use patterns for time series classification
- Pilastre, B., Silva, G., Boussouf, L., d'Escrivan, S., Rodríguez, P., & Tourneret, J. Y. (2020, May). Anomaly Detection in Mixed Time-Series Using A Convolutional Sparse Representation With Application To Spacecraft Health Monitoring. In ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (pp. 3242-3246). IEEE.
  How to use convolutional dictionary learning for detecting anomaly
- La Tour, T. D., Moreau, T., Jas, M., & Gramfort, A. (2018). Multivariate convolutional sparse coding for electromagnetic brain signals. In Advances in Neural Information Processing Systems (pp. 3292-3302).
  How to use convolutional dictionary learning to study the brain