Tomáš Horváth

BUSINESS ANALYTICS

Lecture 4

Time Series

Information Systems and Machine Learning Lab

University of Hildesheim

Germany

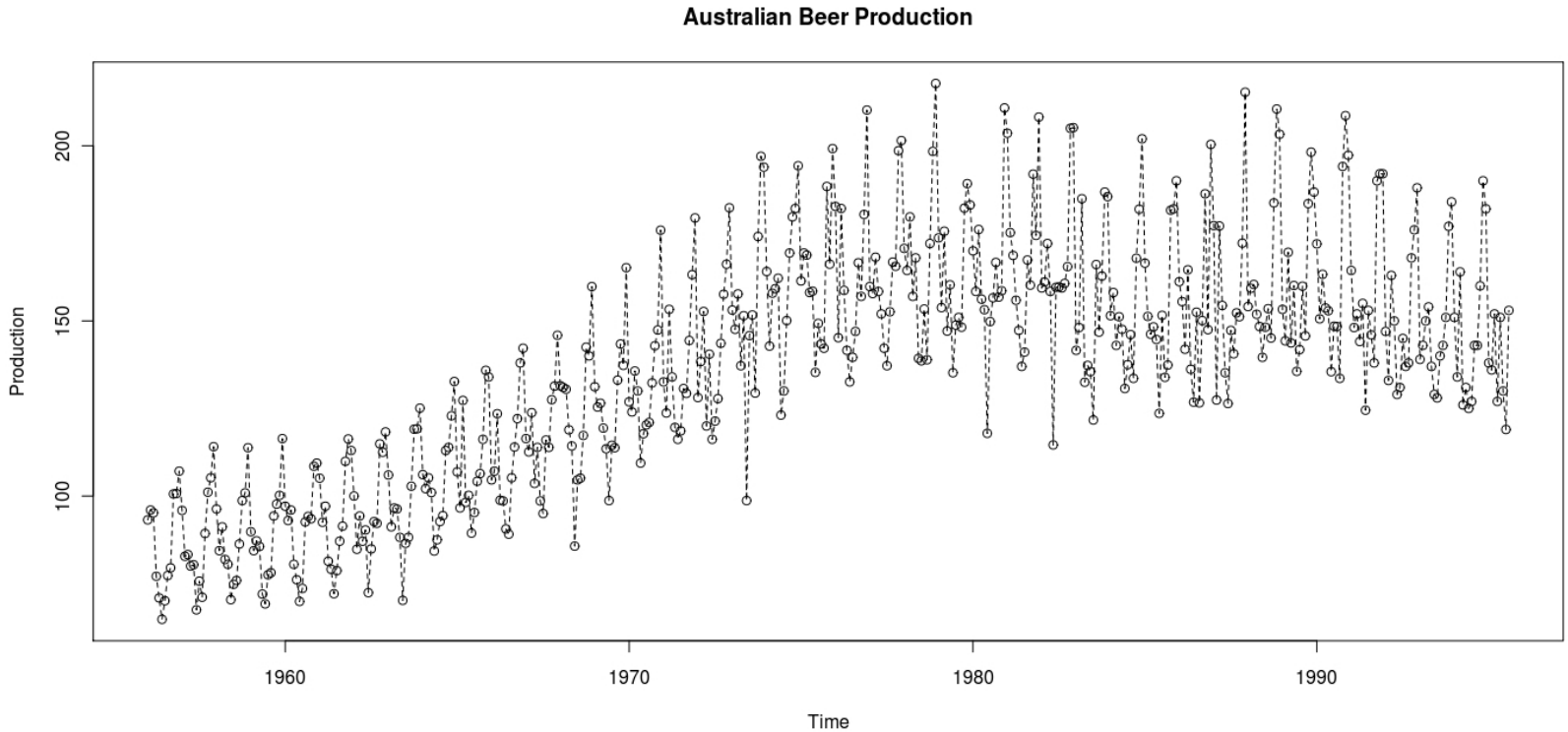
Overview

The aim of this lecture is to get insight to time series mining

- Representation of time series
- Distance measures
- Time series Classification
- Forecasting

Univariate time series

A univariate time series x of length l is a sequence $x = (x_0, x_1, \ldots, x_{l-1})$, where $x_j \in \mathbb{R}$ for each $0 \leq j < l$.

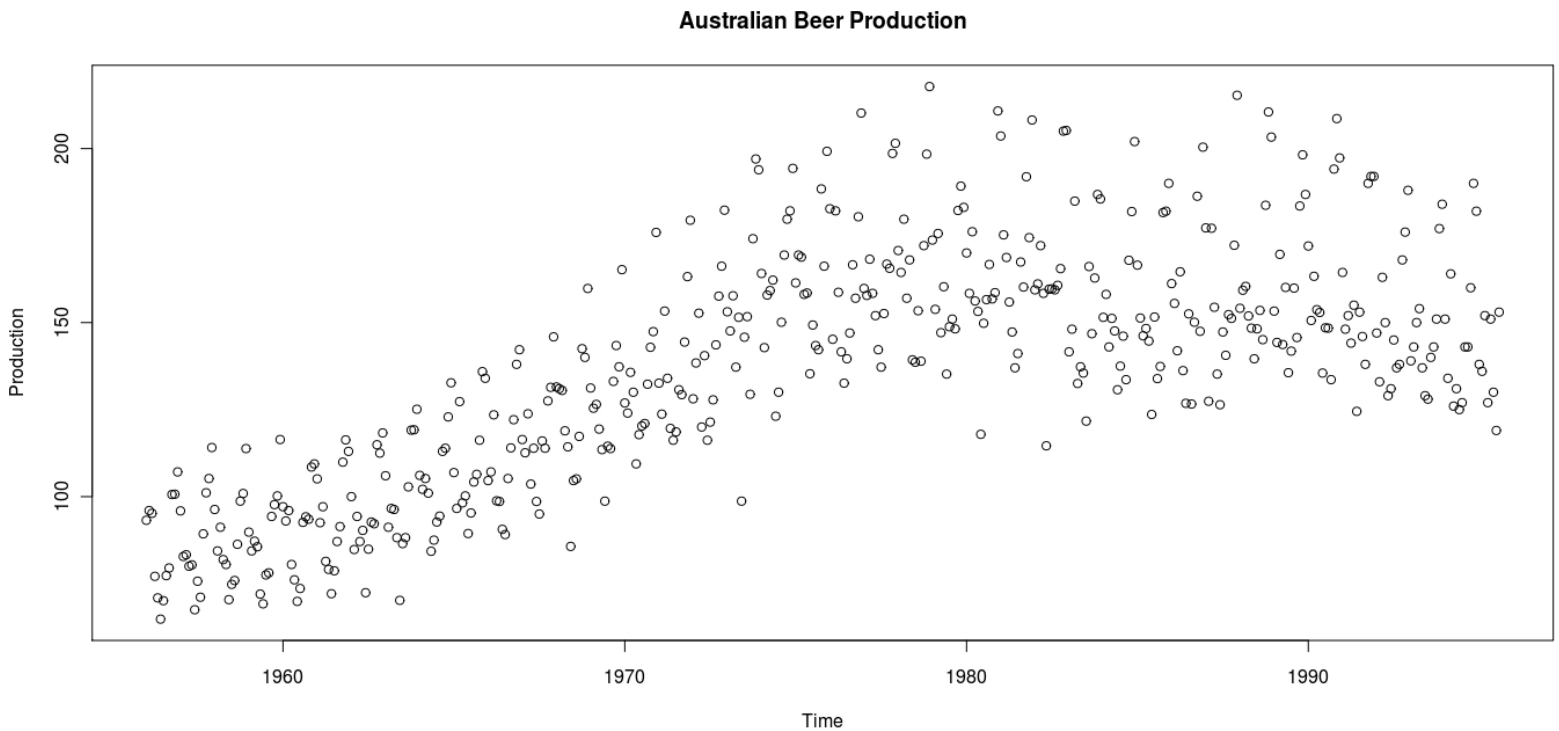


$\mathcal{D} = \{(x^i, c(x^i))\}_{i=1}^n$ is a labeled time series dataset, where $c : \mathcal{T} \to \mathcal{C}$ is a mapping from the set $\mathcal{T}$ of time series to a set $\mathcal{C}$ of classes.

Raw Representation

Listing all the values is not necessary the most approppriate way, especially, when the aim is to classify time series.

- classification algorithms deal with the features of instances
  - approximate/condensed representations of time series are desirable
    - the patterns are usually most interesting than individual points



Different types of representation

Discrete **Fourier**[1] Transform (DFT)

Haar **Wavelet**[1] transformation (DHWT)

Piecewise Linear Approximation (PLA)

- interpolation
- regression

Piecewise Constant Approximation (PCA)

- Piecewise Aggregate Approximation (PAA)
- Adaptive Piecewise Constant Approximation (APCA)

Symbolic Aggregate Approximation (SAX)

[1] Just the basic definitions will be discussed here, since more detailed descriptions are out of the scope of the lecture. See the references at the end of this presentation for more details.

Fourier Series

We can write a **Fourier series** of any continuous, differentiable and *T-periodic*[1] function $f : \mathbb{R} \to \mathbb{C}$ as

$$f(x) = \frac{a_0}{2} + \sum_{k=1}^{\infty} a_k \, cos(k\omega x) + b_k \, sin(k\omega x), \quad \omega = \frac{2\pi}{T}$$

with coefficients $a_k, b_k \in \mathbb{C}$, such that

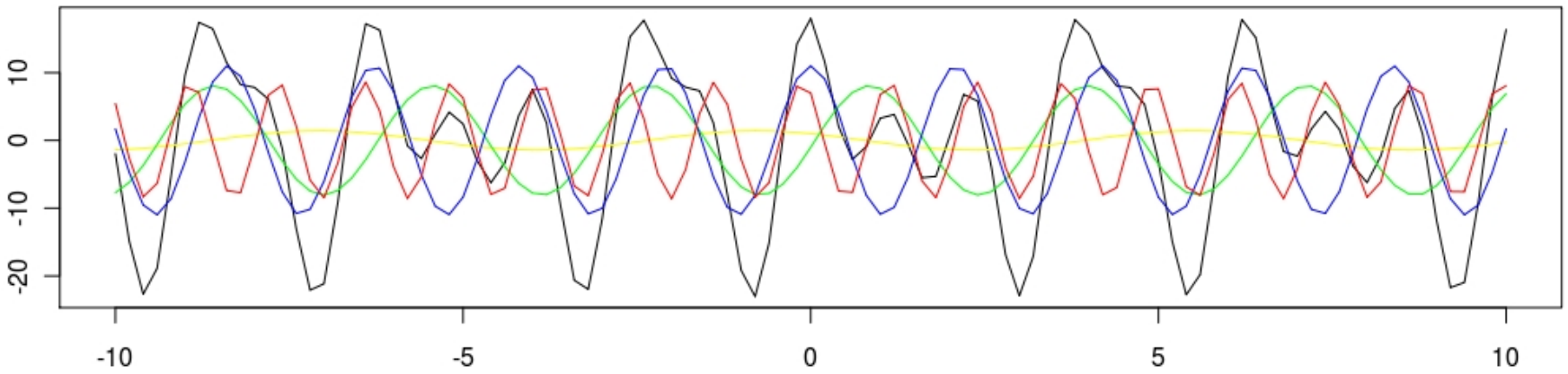
$$a_k = \frac{1}{\frac{\pi}{\omega}} \int_{-\frac{\pi}{\omega}}^{+\frac{\pi}{\omega}} f(x) \, cos(k\omega x) \, dx$$

$$b_k = \frac{1}{\frac{\pi}{\omega}} \int_{-\frac{\pi}{\omega}}^{+\frac{\pi}{\omega}} f(x) \, sin(k\omega x) \, dx$$

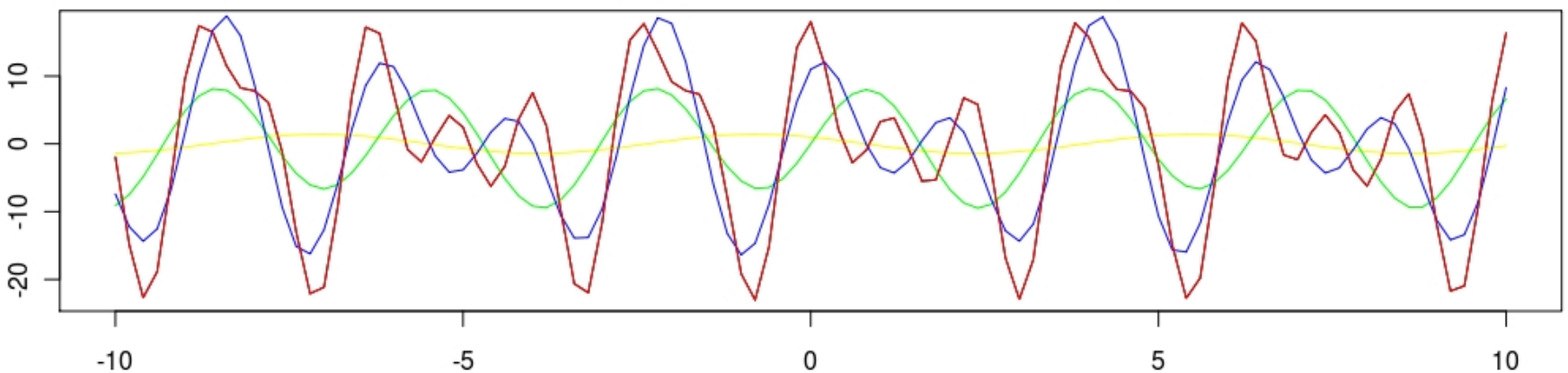
[1] A function $f : \mathbb{R} \to \mathbb{C}$ is T-periodic if $f(x + T) = f(x)$ for each $x \in \mathbb{R}$.

Fourier Series: Example

$f(x) = -sin(x) + cos(x) + 8\ sin(2x) - cos(2x) + 11\ cos(3x) - 5\ sin(5x) + 7\ cos(5x)$



$f(x) = -sin(x) + cos(x)$
$f(x) = -sin(x) + cos(x) + 8\ sin(2x) - cos(2x)$
$f(x) = -sin(x) + cos(x) + 8\ sin(2x) - cos(2x) + 11\ cos(3x)$
$f(x) = -sin(x) + cos(x) + 8\ sin(2x) - cos(2x) + 11\ cos(3x) - 5\ sin(5x) + 7\ cos(5x)$



Complex Fourier Series

Euler formula

$$e^{ix} = cos(x) + i \cdot sin(x)$$

We can write a **complex Fourier series** of any continuous, differentiable and T-periodic function $f : \mathbb{R} \to \mathbb{C}$ as

$$f(x) = \sum_{k \in \mathbb{Z}} c_k e^{ik\omega x}, \quad \omega = \frac{2\pi}{T}$$

with complex coefficients $c_k \in \mathbb{C}$, such that

$$c_k = \frac{1}{\frac{2\pi}{\omega}} \int_{-\frac{\pi}{\omega}}^{+\frac{\pi}{\omega}} f(x)\, e^{-ik\omega x}\, dx$$

Fourier transform

A **Fourier transform**[1] of a continuous, differentiable and T-periodic function $f : \mathbb{R} \to \mathbb{C}$ is a mapping $F : \mathbb{R} \to \mathbb{C}$ defined as

$$F(\omega) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{+\infty} f(x)\, e^{-i\omega x}\, dx$$

If F also satisfies some regularity conditions than we can define the **inverze Fourier transform** as

$$f(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{+\infty} F(\omega)\, e^{i\omega x}\, d\omega$$

[1] Called also as a Fourier spectrum of f

Discrete Fourier transform (DFT)

A **Discrete Fourier transform** $F_d(f)$ of a finite, discrete function $f : \{0, 1, \ldots, n-1\} \to \mathbb{C}$ is a mapping $F_d : \{0, 1, \ldots, n-1\} \to \mathbb{C}$ defined as

$$F_d(\omega) = \frac{1}{\sqrt{n}} \sum_{x=0}^{n-1} f(x)\big(cos(2\pi\frac{\omega x}{n}) - i \cdot sin(2\pi\frac{\omega x}{n})\big) = \frac{1}{\sqrt{n}} \sum_{x=0}^{n-1} f(x) e^{-i2\pi\frac{\omega x}{n}}$$

An **inverze discrete Fourier transform** is defined as

$$f(x) = \frac{1}{\sqrt{n}} \sum_{\omega=0}^{n-1} F_d(\omega)\big(cos(2\pi\frac{\omega x}{n}) + i \cdot sin(2\pi\frac{\omega x}{n})\big) = \frac{1}{\sqrt{n}} \sum_{\omega=0}^{n-1} F_d(\omega) e^{i2\pi\frac{\omega x}{n}}$$

Computation of DFT

$f(x)$ consists of real and imaginary parts

- $f(x) = f(x)_{re} + i \cdot f(x)_{im} \in \mathbb{C}$

Thus, we compute[1] $F_d(\omega)$ for each $\omega \in \{0, 1, \ldots, n-1\}$ as

$$
\begin{aligned}
F_d(\omega) &= \frac{1}{\sqrt{n}} \sum_{x=0}^{n-1} f(x)\big(cos(2\pi \tfrac{\omega x}{n}) - i \cdot sin(2\pi \tfrac{\omega x}{n})\big) \\
&= \frac{1}{\sqrt{n}} \sum_{x=0}^{n-1} f(x)_{re}\, cos(2\pi \tfrac{\omega x}{n}) + f(x)_{im} sin(2\pi \tfrac{\omega x}{n}) \\
&+ i \cdot \frac{1}{\sqrt{n}} \sum_{x=0}^{n-1} -f(x)_{re}\, sin(2\pi \tfrac{\omega x}{n}) + f(x)_{im} cos(2\pi \tfrac{\omega x}{n})
\end{aligned}
$$

[1] An efficient way of computation called fast Fourier transform (FFT) can be found in the literature. FFT works well for the sequences of even length, best for lengths of 2^k, where $k \in \mathbb{N}$.

DFT: Example (1)

Sample 16 points from the example before

- $f'(x') = -sin(x') + cos(x') + 8\ sin(2x') - cos(2x') + 11\ cos(3x') - 5\ sin(5x') + 7\ cos(5x')$ for $x' \in \{-8, \ldots, -1, 1, \ldots, 8\}$
- $f'(-8) = 7.828, f'(-1) = -19.175, \ldots, f'(1) = 3.280, \ldots, f'(8) = -6.209$

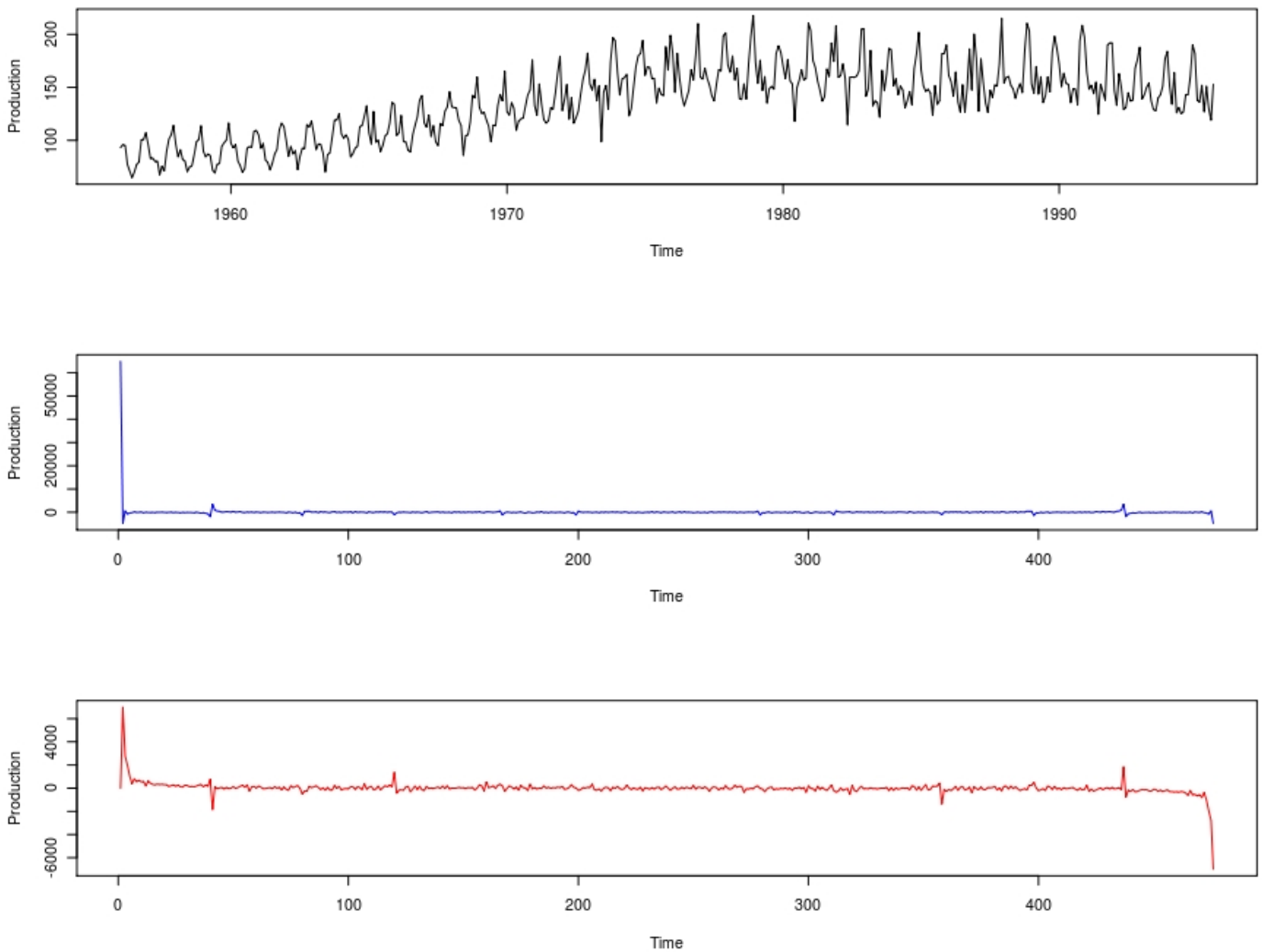
x	$f(x)_{re}$	$f(x)_{im}$
0	7.828	+0.000i
1	-21.142	+0.000i
2	7.533	+0.000i
3	2.436	+0.000i
4	7.524	+0.000i
5	-11.663	+0.000i
6	9.169	+0.000i
7	-19.175	+0.000i
8	3.280	+0.000i
9	0.682	+0.000i
10	-22.918	+0.000i
11	15.738	+0.000i
12	-3.027	+0.000i
13	9.387	+0.000i
14	-2.325	+0.000i
15	-6.209	+0.000i

$$\overset{DFT}{\Longrightarrow}$$

ω	$F_d(\omega)_{re}$	$F_d(\omega)_{im}$
0	-22.883	+0.000i
1	12.733	+4.840i
2	-37.048	+4.288i
3	-19.404	-0.111i
4	24.146	+15.526i
5	1.691	+38.107i
6	50.269	-40.172i
7	23.171	+85.261i
8	37.010	+0.000i
9	23.171	-85.261i
10	50.269	+40.172i
11	1.691	-38.107i
12	24.146	-15.526
13	-19.404	+0.111i
14	-37.048	-4.288i
15	12.733	-4.840i

DFT: Example (2)

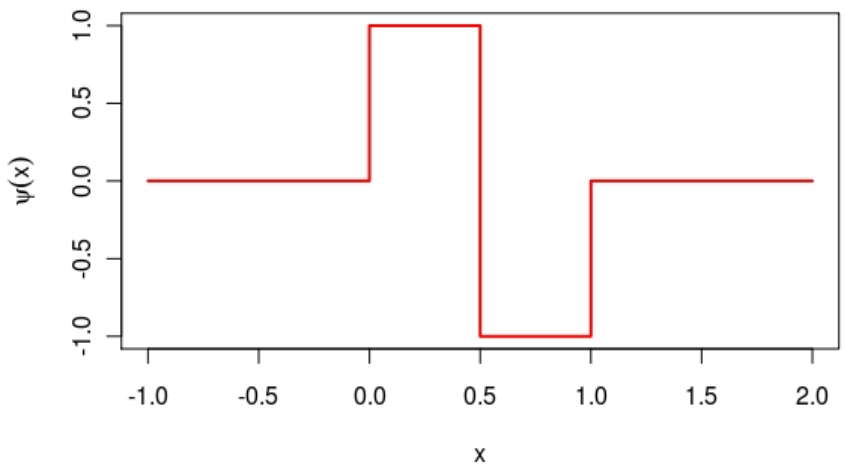
Real and Imaginary parts of DFT on Australian Beer Production data



Haar wavelet and basis functions

The **Haar wavelet** $\psi : \mathbb{R} \to \mathbb{R}$ is defined as

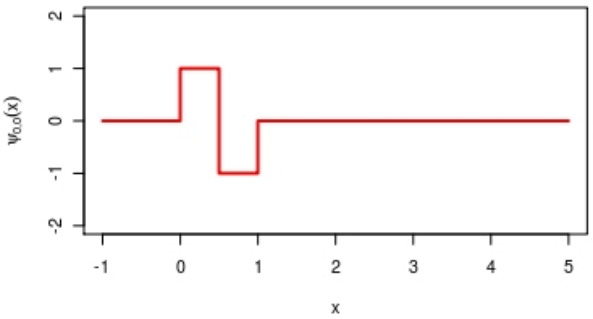
$$\psi(x) = \begin{cases} +1, & x \in \langle 0, \frac{1}{2}) \\ -1, & x \in \langle \frac{1}{2}, 1) \\ 0, & \text{else} \end{cases}$$

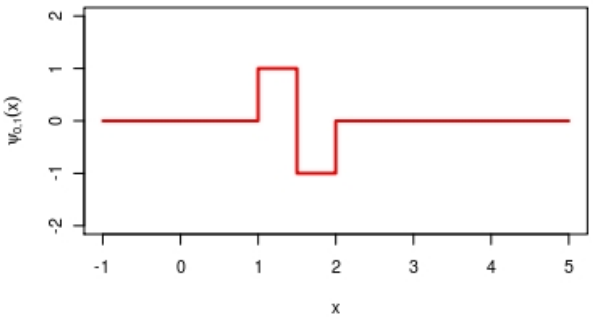


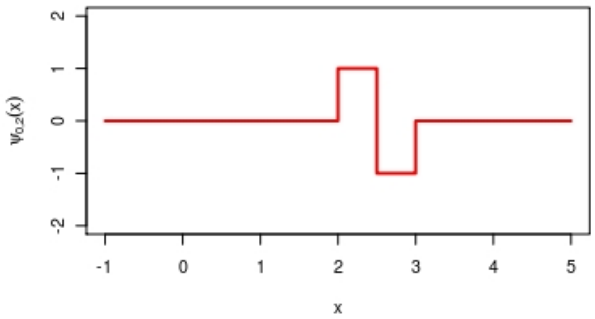
Haar basis functions $\psi_{s,t} : \mathbb{R} \to \mathbb{R}, \quad s, t \in \mathbb{Z}$ are defined as

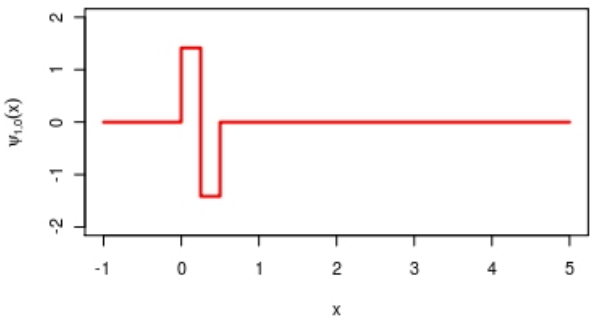
$$\psi_{s,t} = \sqrt{2^s} \cdot \psi(2^s x - t) = \sqrt{2^s} \cdot \begin{cases} +1, & x \in \langle 2^{-s}t, 2^{-s}(t + \frac{1}{2})) \\ -1, & x \in \langle 2^{-s}(t + \frac{1}{2}), 2^{-s}(t + 1)) \\ 0, & \text{else} \end{cases}$$

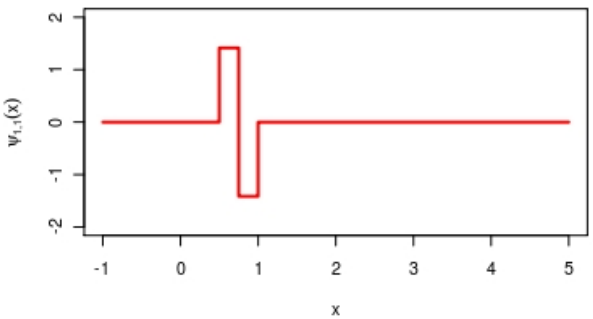
Haar basis functions: Example

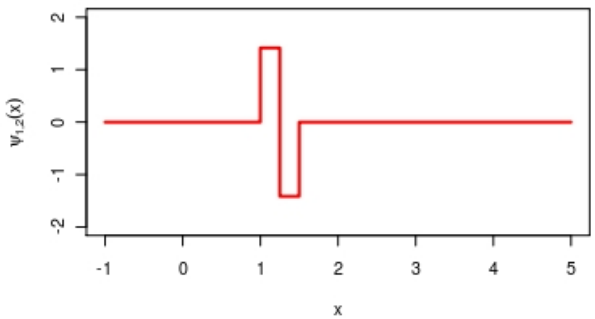


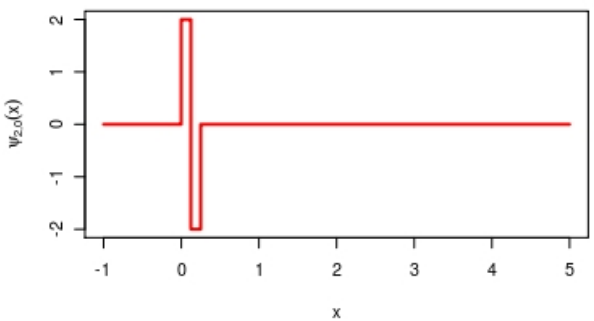


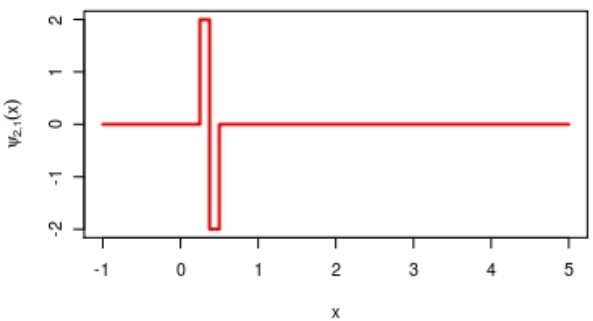


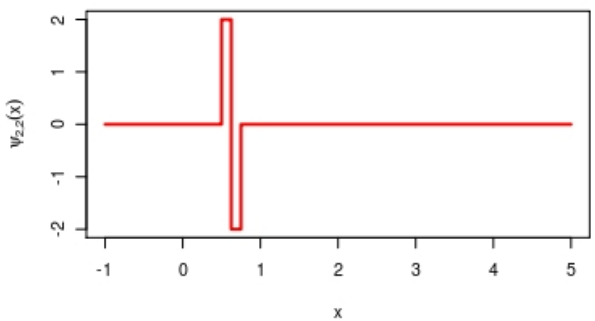












Haar Wavelet representation

Every function $f : \mathbb{R} \to \mathbb{R}$ satisfying some regularity conditions can be written as

$$f(x) = \sum_{s \in \mathbb{Z}} \sum_{t \in \mathbb{Z}} c_{s,t}\, \psi_{s,t}(x)$$

with $c_{s,t} \in \mathbb{R}$, such that

$$c_{s,t} = \sqrt{2^s} \Big(\int_{2^{-s}t}^{2^{-s}(t+\frac{1}{2})} f(x)\, dx - \int_{2^{-s}(t+\frac{1}{2})}^{2^{-s}(t+1)} f(x)\, dx \Big) = \frac{1}{\sqrt{2}} (a_{s+1,2t} - a_{s+1,2t+1})$$

where $a_{s,t}$ can be computed recursively as

$$a_{s,t} = \sqrt{2^s} \int_{2^{-s}t}^{2^{-s}(t+1)} f(x)\, dx = \frac{1}{\sqrt{2}} (a_{s+1,2t} + a_{s+1,2t+1})$$

Discrete Haar Wavelet Transform (DHWT)

A finite, discrete function $f : \{0, 1, \ldots, n-1\} \to \mathbb{R}$, with length $n = 2^k, k \in \mathbb{N}$ can be represented as

$$f(x) = a_{-n,0} + \sum_{s=-n}^{-1} \sum_{t=0}^{2^{n+s}-1} c_{s,t} \cdot \sqrt{2^s} \cdot \psi_{s,t}(x)$$

where the initial values $a_{0,t}$ are the original values of f, i.e. $a_{0,t} = f(t)$.

DHWT allows time series to be viewed in multiple resolutions corresponding to frequencies or spectrums[1]

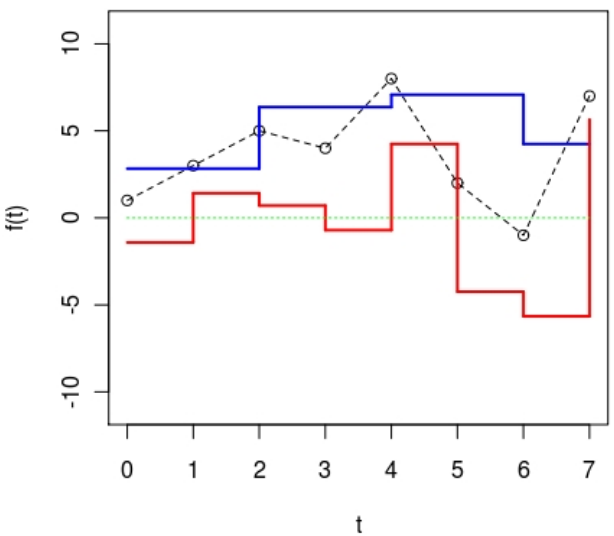
- coefficients $a_{s,t}$ are the smoothed values of f in the corresponding spectrum s
- coefficients $c_{s,t}$ represent the differences in values of f in the corresponding spectrum s

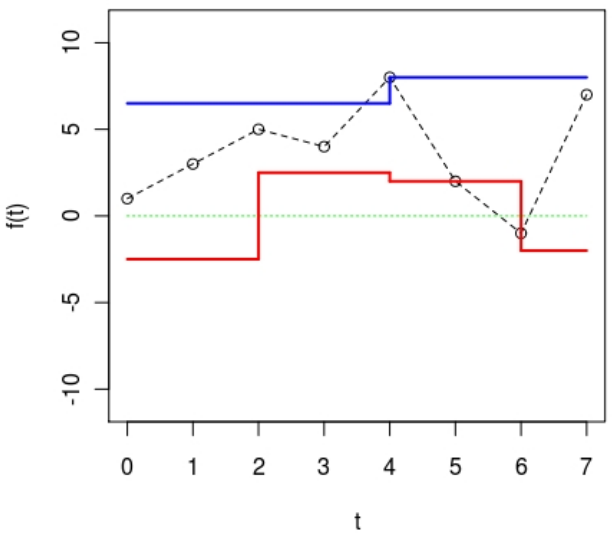
[1] Averages and differences are computed across a window of values.

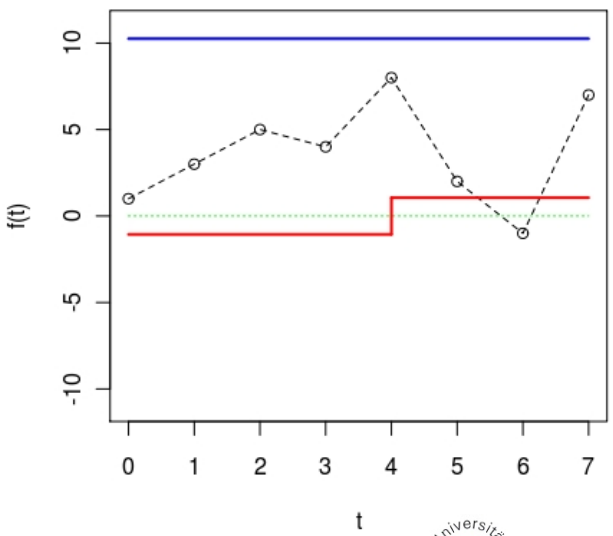
DHWT: Example

t	0	1	2	3	4	5	6	7
$f(t)$	1	3	5	4	8	2	-1	7
$a_{-1,t}$	$\frac{4}{\sqrt{2}}$	$\frac{9}{\sqrt{2}}$	$\frac{10}{\sqrt{2}}$	$\frac{6}{\sqrt{2}}$	–	–	–	–
$c_{-1,t}$	$\frac{-2}{\sqrt{2}}$	$\frac{1}{\sqrt{2}}$	$\frac{6}{\sqrt{2}}$	$\frac{-8}{\sqrt{2}}$	–	–	–	–
$a_{-2,t}$	$\frac{13}{2}$	$\frac{16}{2}$	–	–	–	–	–	–
$c_{-2,t}$	$\frac{-5}{2}$	$\frac{4}{2}$	–	–	–	–	–	–
$a_{-3,t}$	$\frac{29}{2\sqrt{2}}$	–	–	–	–	–	–	–
$c_{-3,t}$	$\frac{-3}{2\sqrt{2}}$	–	–	–	–	–	–	–

$$(1,3,5,4,8,2,-1,7) \rightsquigarrow (\tfrac{29}{2\sqrt{2}}, \tfrac{-3}{2\sqrt{2}}, \tfrac{-5}{2}, \tfrac{4}{2}, \tfrac{-2}{\sqrt{2}}, \tfrac{1}{\sqrt{2}}, \tfrac{6}{\sqrt{2}}, \tfrac{-8}{\sqrt{2}})$$

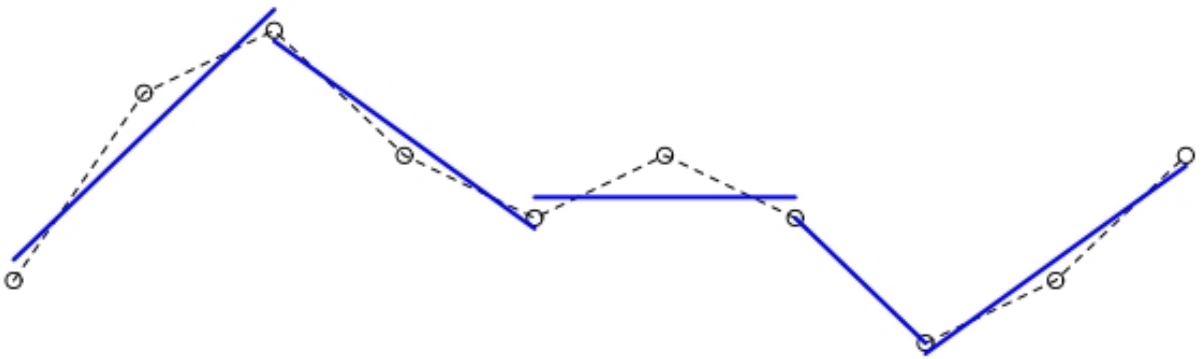




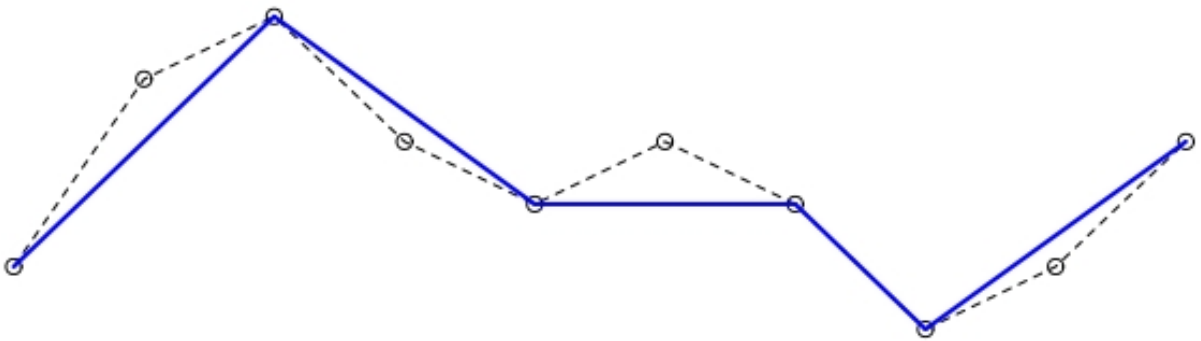


Piecewise Linear Approximation

Regression

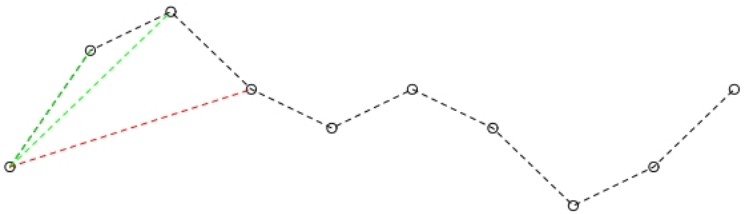
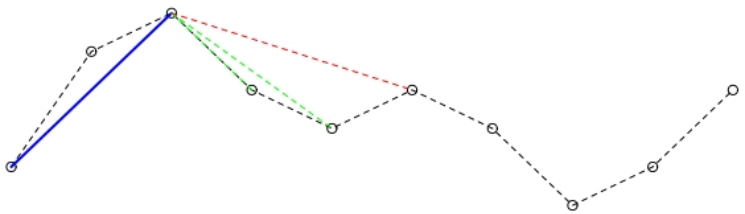
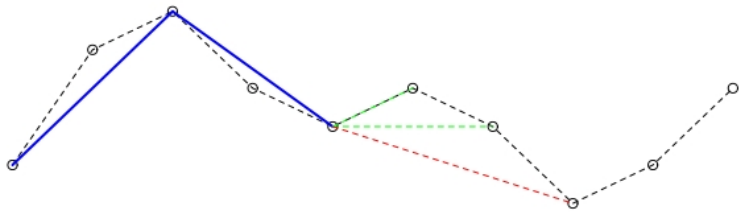
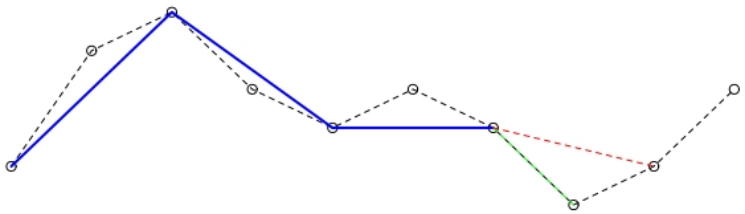
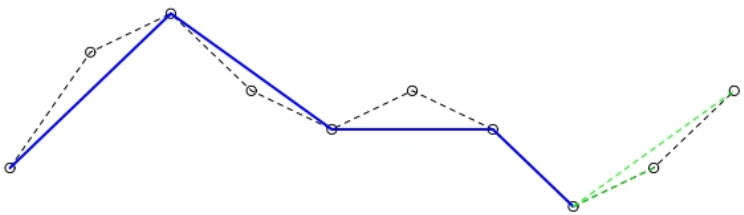
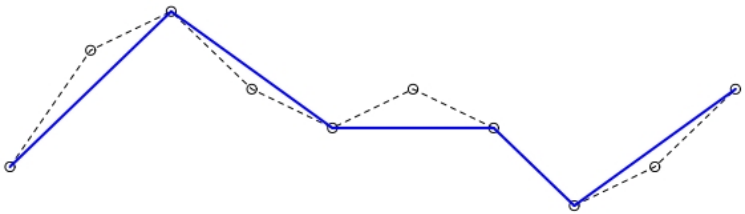


Interpolation



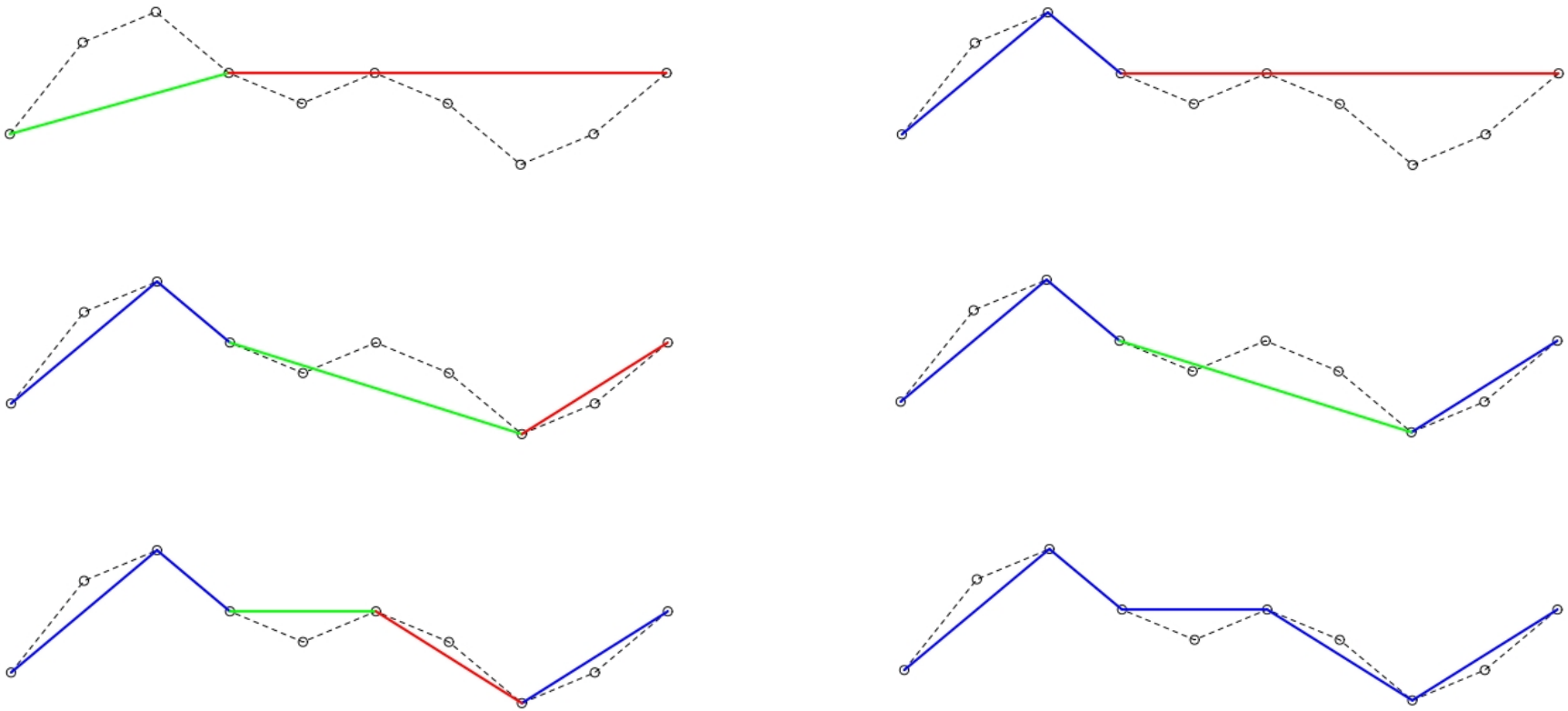
PLA: Sliding Windows

Anchor a left point and approximate data to the right with increasing size of a window while the error is under a given treshold

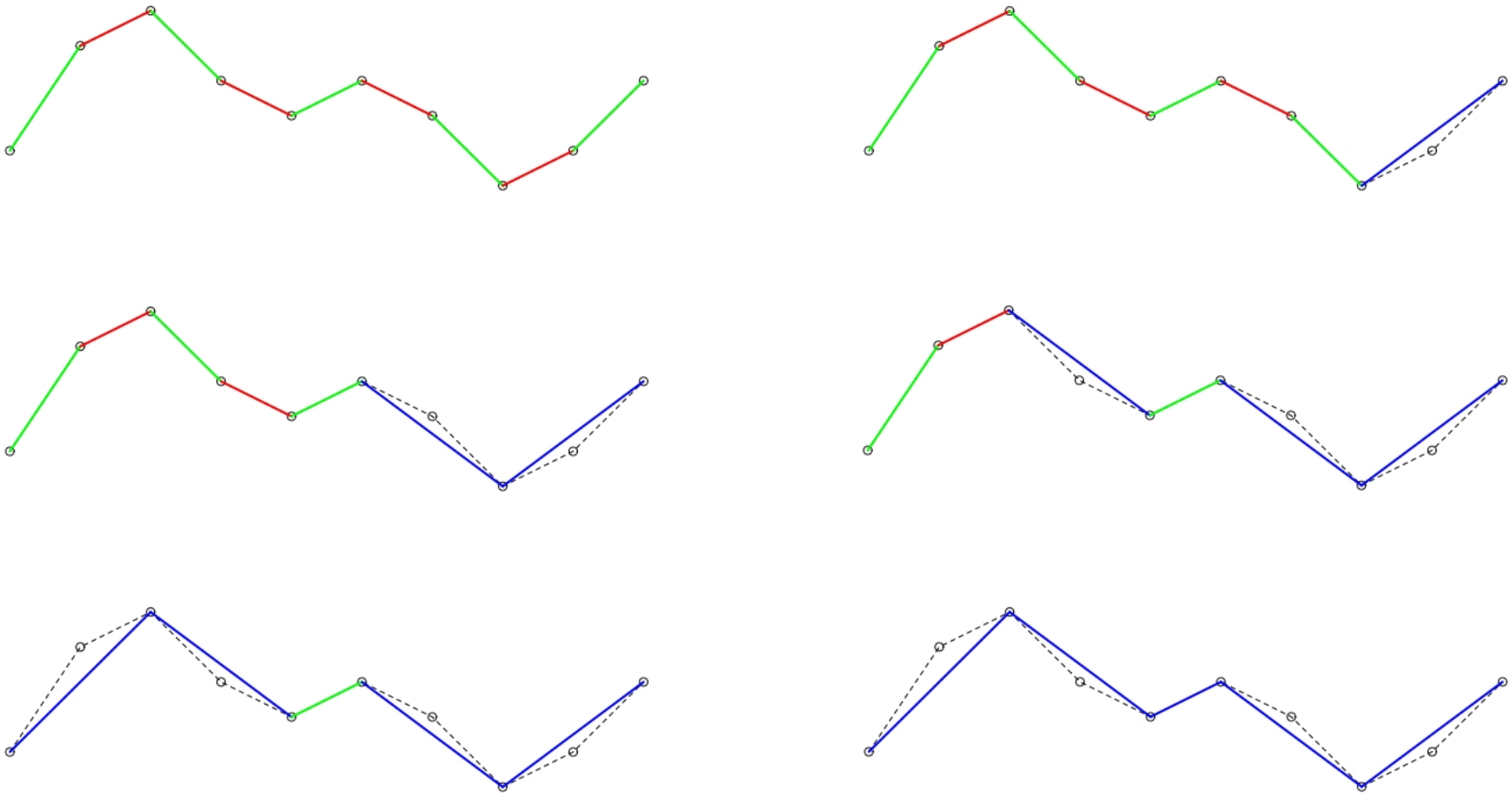
PLA: Top-down approach

Taking into account every possible partition split at the best location recursively while the error is above a given threshold



PLA: Bottom-up approach

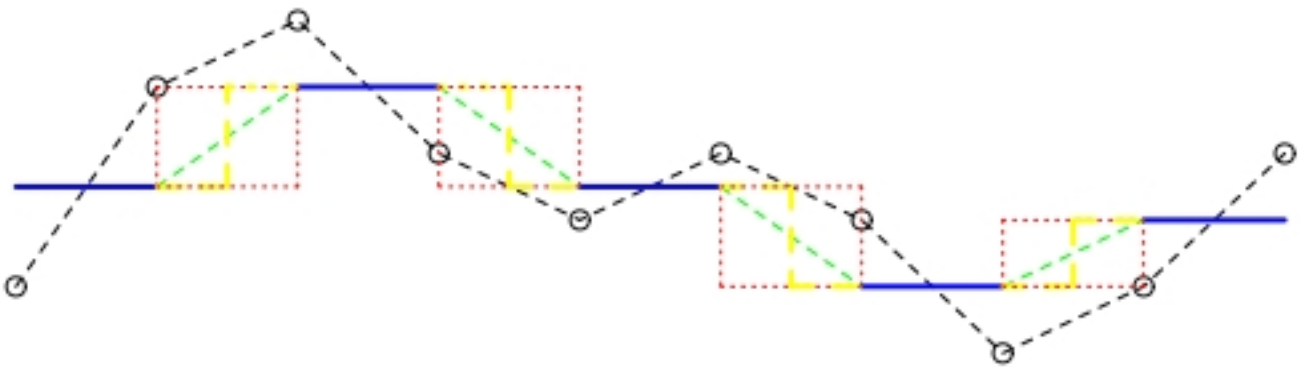
Starting from the finest possible approximation merge segments while the error is above the given threshold



Piecewise Aggregate Approximation

Replace $x = (x_1, \ldots, x_n)$ with a vector $\overline{x} = (\overline{x}_1, \ldots, \overline{x}_N)$, where N is a factor[1] of n and the elements of $\overline{x}$ are computed as

$$\overline{x}_i = \frac{N}{n} \sum_{j=\frac{n}{N}(i-1)+1}^{\frac{n}{N}i} x_j$$

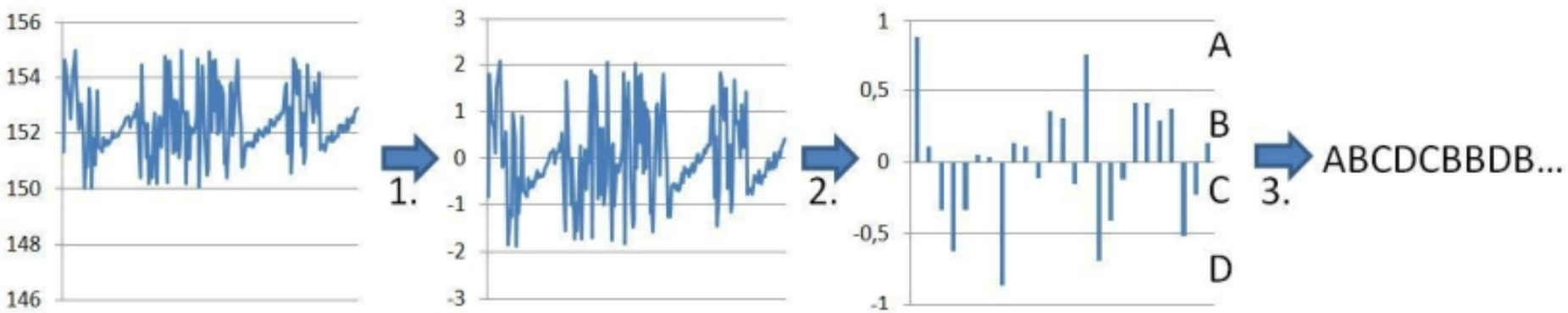


[1] We assume this just for the convenience, for other N it would work, too.

Symbolic Aggregate Approximation

Approach

1. **normalize** time series x to $\mathcal{N}(0,1)$
2. provide **PAA** on the normalized x
3. **discretize** resulting averages of PAA into discrete symbols



SAX: Breakpoints

An important thing is to discretize[1] in a way that symbols of the alphabet $\alpha = \{\alpha_1, \ldots, \alpha_k\}$ are produced with equiprobability.

- It was empirically discovered on more than 50 datasets that normalized subsequences are normally distributed.

Breakpoints are defined as $\beta = \{\beta_1, \ldots, \beta_{k-1}\}$, with $\beta_i < \beta_i + 1$ for all $i, 1 \leq i < k-1$, such that[2]

$$\forall i \in \{0, \ldots, k-1\} \quad \int_{\beta_i}^{\beta_{i+1}} \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} \, dx = \frac{1}{k}$$

where $\beta_0 = -\infty, \beta_k = \infty$.

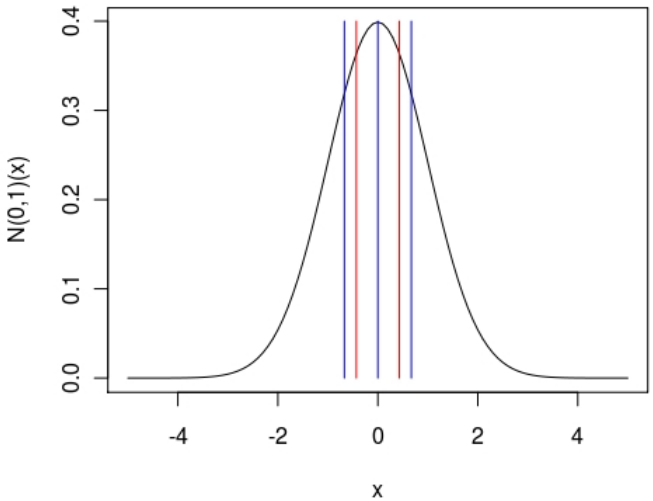
Breakpoints can be found by looking up in a statistical table.

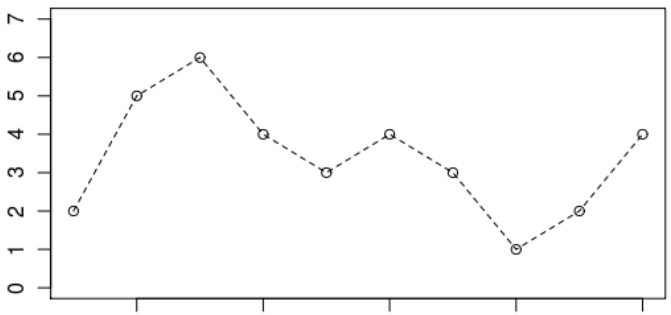
[1] For more details about mapping the averages to symbols, please, see the references.

[2] The area under the $\mathcal{N}(1, 0)$ Gaussian function from β_i to β_{i+1} is equal to $\frac{1}{k}$.

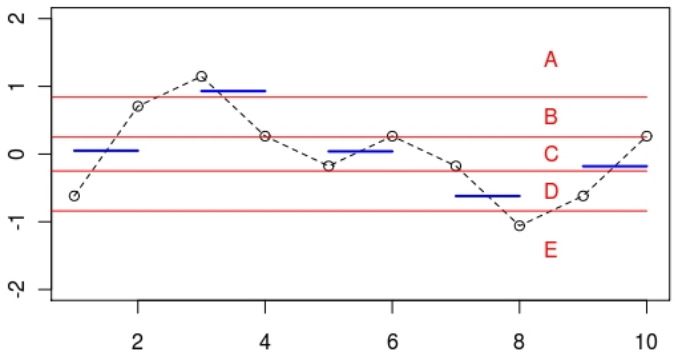
Breakpoints: Example

k	3	4	5	6	7
β_1	-0.43	-0.67	-0.84	-0.97	-1.07
β_2	0.43	0	-0.25	-0.43	-0.57
β_3	–	0.67	0.25	0	-0.18
β_4	–	–	0.84	0.43	0.18
β_5	–	–	–	0.97	0.57
β_6	–	–	–	–	1.07





$$SAX \Longrightarrow$$

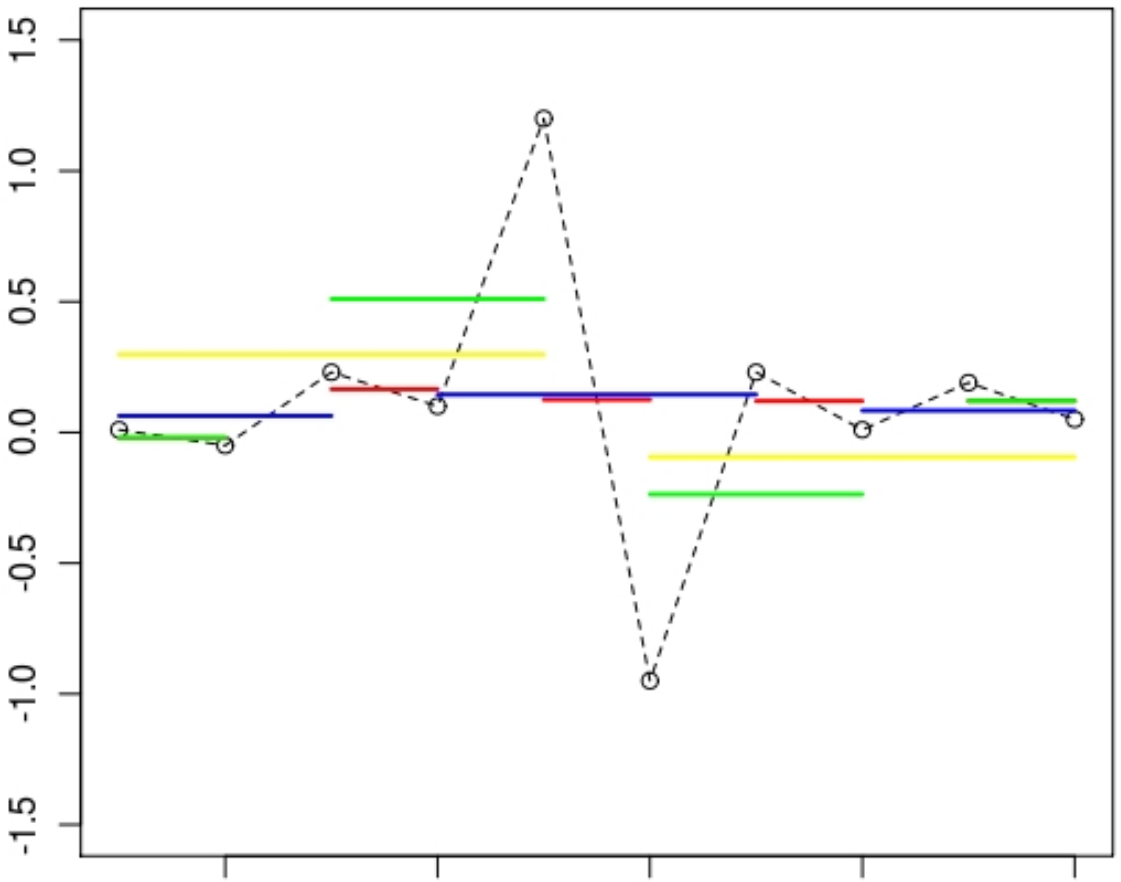


$$(2, 5, 6, 4, 3, 4, 3, 1, 2, 4) \rightsquigarrow (CACDC)$$

Critical points

There are cases, especially in financial time series, when data contain some **critical points**.

- difficult to identify them by PAA, and thus SAX, too.



Extended SAX

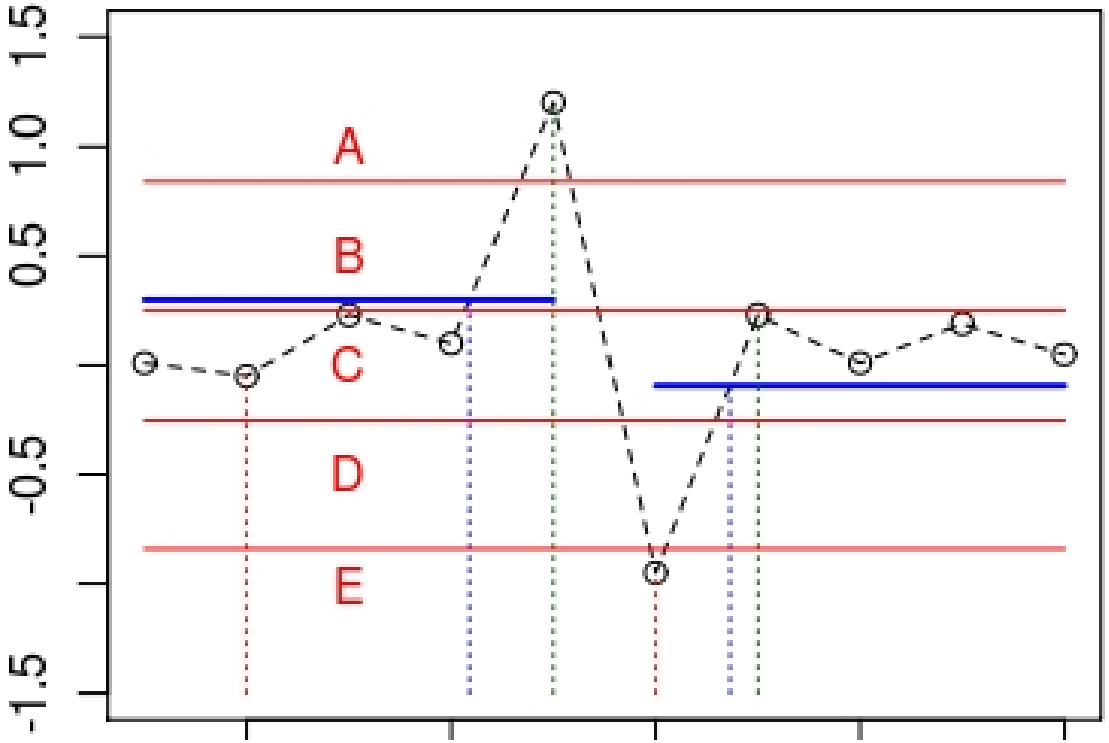
For each segment S_k we also store the symbols s_{min} and s_{max} for the minimal and maximal values of the segment in addition to the symbol s_{mean} representing the mean value of the segment.

$$< s_1, s_2, s_3 >= \begin{cases} < s_{max}, s_{mean}, s_{min} > & \text{if } p_{max} < p_{mean} < p_{min} \\ < s_{min}, s_{mean}, s_{max} > & \text{if } p_{min} < p_{mean} < p_{max} \\ < s_{min}, s_{max}, s_{mean} > & \text{if } p_{min} < p_{max} < p_{mean} \\ < s_{max}, s_{min}, s_{mean} > & \text{if } p_{max} < p_{min} < p_{mean} \\ < s_{mean}, s_{max}, s_{min} > & \text{if } p_{mean} < p_{max} < p_{min} \\ < s_{mean}, s_{min}, s_{max} > & \text{if } p_{mean} < p_{min} < p_{max} \end{cases}$$

where p_{min}, p_{mean} and p_{max} are the positions for the minimal, mean and maximal values in the segment, respectively.

Extended SAX: Example

SAX: "BC"
Extended SAX: "CBAECC"



Static distance measures

For two time series with the **same length**.

- Euclidean distance

$$d_{EU}(x_1, x_2) = \sqrt{\sum_{j=1}^{l}(x_{1j} - x_{2j})^2}$$

- Euclidean distance on the representations of time series

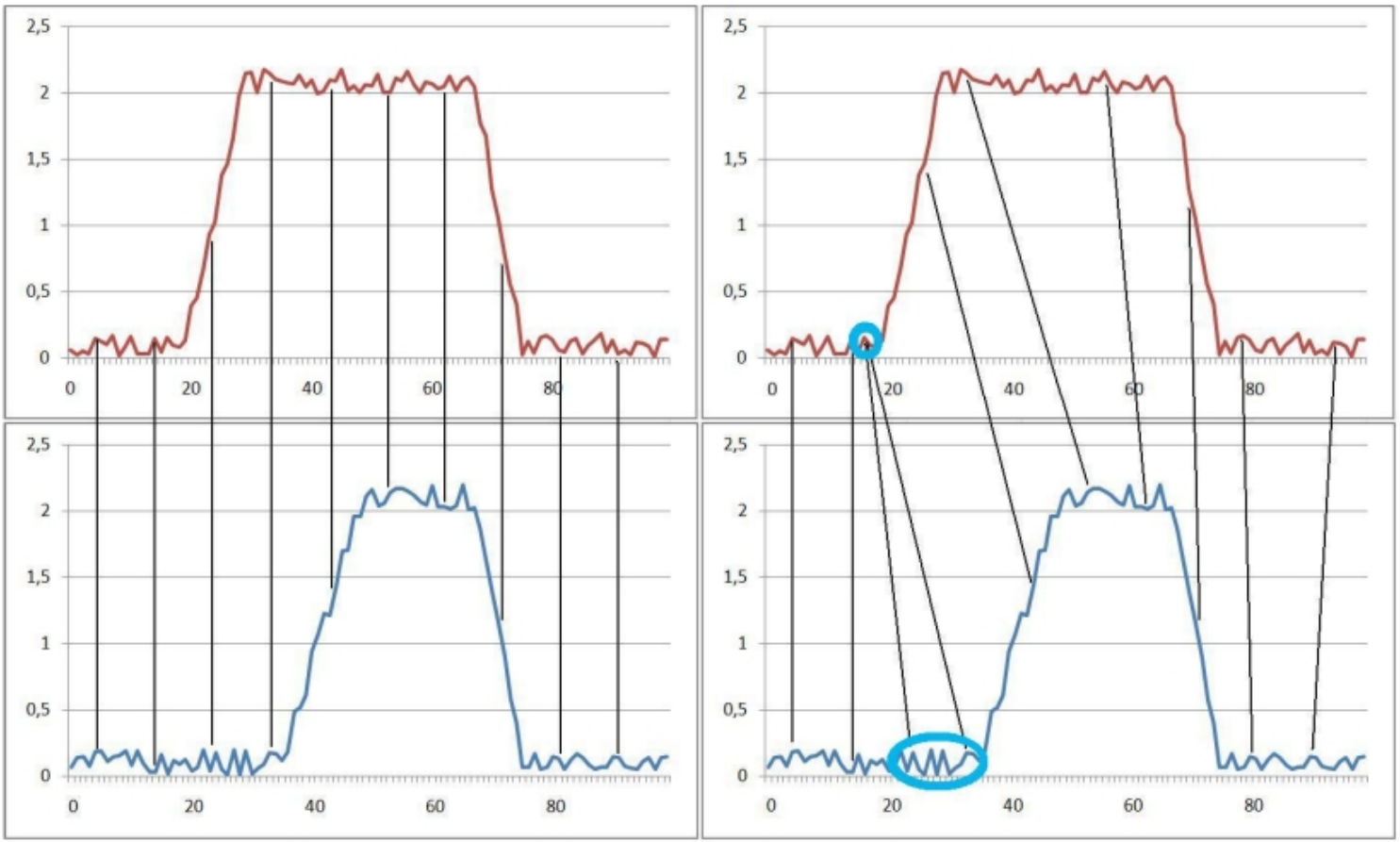
$$d^R_{EU}(x_1, x_2) = \sqrt{\sum_{j=1}^{l}(R(x_1)_j - R(x_2)_j)^2}$$

where $R : \mathcal{T} \to \mathbb{C}$ can be – among other possibilities – defined as

$$R(x) = DFT(x),$$
$$R(x) = DHWT(x) \text{ or}$$
$$R(x) = PAA(x)$$

Dynamic distance measures

Static distance measures compare the values at the same positions, while dynamic distance measures rather compute the so-called **"cost of transformation"** of one time series to another.



Warping path

We have

- sequences $x = (x_1, \ldots, x_n)$ and $y = (y1, \ldots, y_m)$
- **local distance measure** c defined as $c : \mathbb{R} \times \mathbb{R} \to \mathbb{R}_{\geq 0}$
- **cost matrix** $C \in \mathbb{R}^{n \times m}$ defined by $C(i, j) = c(x_i, y_j)$

The goal is to find an alignment between x and y having minimal overall cost[1].

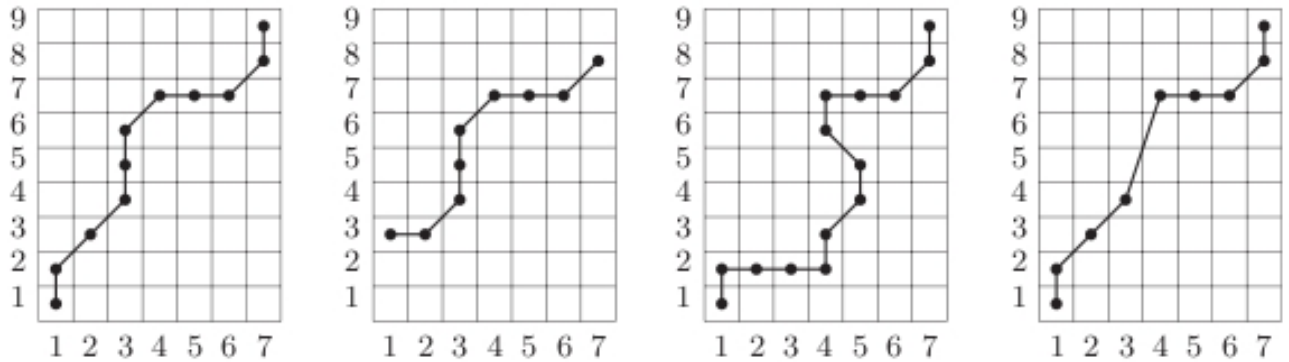
An **(n,m)-warping path** is a sequence $p = (p_1, \ldots, p_L)$ with $p_l = (i_l, j_l)$ for $1 \leq i_l \leq n$, $1 \leq j_l \leq m$ and $1 \leq l \leq L$ satisfying the following conditions:

- *Boundary condition:* $p_1 = (1, 1)$ and $p_L = (n, m)$
- *Monotonicity condition:* $n_1 \leq ... \leq n_L$ and $m_1 \leq ... \leq m_L$
- *Step size condition:* $p_{l+1} - p_l \in \{(1, 0), (0, 1), (1, 1)\}$ for $1 \leq l < L$

[1] Running along a "valley" of low costs within C.

The DTW distance

An (n,m)-warping path defines an alignment between the two time series x and y by assigning the element x_{i_l} of x to the element y_{j_l} of y.



The **total cost** $c_p(x, y)$ of p is defined as

$$c_p(x, y) = \sum_{l=1}^{L} c(x_{i_l}, y_{j_l})$$

The **DTW distance** is defined as

$$DTW(x, y) = min \{ c_p(x, y) \mid p \text{ is an (n,m)-warping path} \}$$

Some properties of DTW

Let $x = (1, 2, 3)$, $y = (1, 2, 2, 3)$ and $z = (1, 3, 3)$ be time series and $c(a, b) = I(a \neq b)$

- c is a metric[1]

DTW does not satisfy triangle inequality

- $DTW(x, y) = 0$, $DTW(x, z) = 1$, $DTW(z, y) = 2$

DTW is generally not unique

- $p_1 = (\,(1, 1), (2, 2), (3, 2), (4, 3)\,)$, $c_{p_1}(x, y) = 2$
- $p_2 = (\,(1, 1), (2, 1), (3, 2), (4, 3)\,)$, $c_{p_2}(x, y) = 2$
- $p_3 = (\,(1, 1), (2, 2), (3, 3), (4, 3)\,)$, $c_{p_3}(x, y) = 2$

[1] i.e. satisfies non-negativity ($c(a, b) \geq 0$), identity ($c(a, b) = 0$ iff $a = b$), symmetry ($c(a, b) = c(b, a)$) and triangular inequality ($c(a, d) \leq c(a, b) + c(b, d)$) conditions

Accumulated cost matrix

One way to determine $DTW(x, y)$ would be to try all the possible warping paths between x and y.

- computationally not feasible – exponential complexity

An **accumulated cost matrix** $D \in \mathbb{R}^{n \times m}$ is defined as

$$D(i, j) = DTW(x_{1:i}, y_{1:j})$$

where $i \in \{1, \ldots, n\}, j \in \{1, \ldots, m\}$ and $x_{1:i}, y_{1:j}$ are **prefix sequences** $(x_1, \ldots, x_i)$ and $(y_1, \ldots, y_j)$, respectively.

Obviously, the following holds

$$DTW(x, y) = D(n, m)$$

Efficient computation of D (1)

Theorem: D satisfies the following identities:

$$D(i,1) = \sum_{k=1}^{i} c(x_k, y_1) \quad \text{for } 1 \leq i \leq n$$

$$D(1,j) = \sum_{k=1}^{j} c(x_1, y_k) \quad \text{for } 1 \leq j \leq m$$

and

$$D(i,j) = min\{D(i-1,j-1),\ D(i-1,j),\ D(i,j-1)\} + c(x_i, y_j)$$

for $1 < i \leq n$ and $1 < j \leq m$.

In particular, $DTW(x,y) = D(n,n)$ can be computed in $O(nm)$.

Efficient computation of D (2)

Proof: If $i \in \{1, \ldots, n\}$ and $j = 1$ then there is only one possible warping path between $x_{1:j}$ and $y_{1:1}$ with a total cost $\sum_{k=1}^{i} c(x_k, y_1)$. Thus, the formula for $D(i, 1)$ is proved as can be analogously proved the formula for $D(1, j)$, too.

For $i, j > 1$ let $p = (p1, \ldots, p_L)$ be an optimal warping path for $x_{1:i}$ and $y_{1:j}$.
From the *boundary condition* we have $p_L = (i, j)$.
From the *step size condition* we have
$(a, b) \in \{(i-1, j-1), (i-1, j), (i, j-1)\}$ for $(a, b) = p_{L-1}$.
Since p is an optimal path for $x_{1:i}$ and $y_{1:j}$, so is $(p_1, \ldots, p_{L-1})$ for $x_{1:a}$ and $y_{1:b}$. Because $D(i, j) = c_{(q_1, \ldots, q_{L-1})}(x_{1:a}, y_{1:b}) + c(x_i, y_j)$, the formula for $D(i, j)$ holds. □

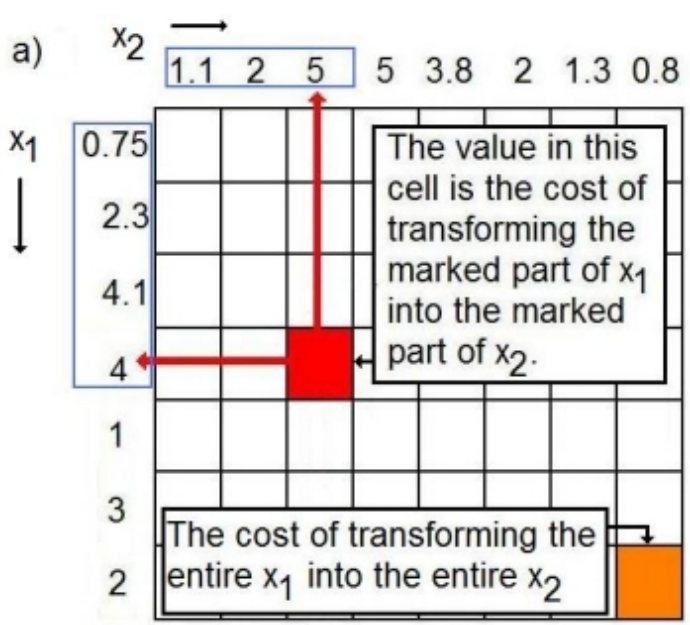
Efficient computation of D (3)

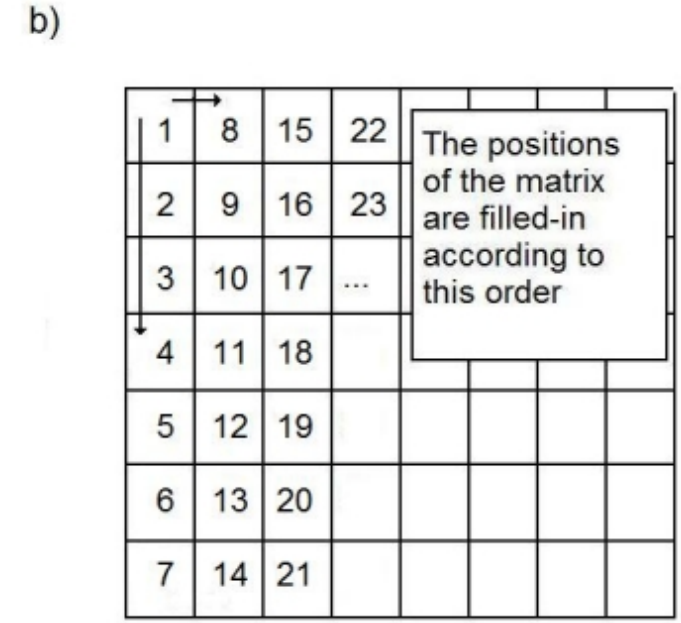
Initialization

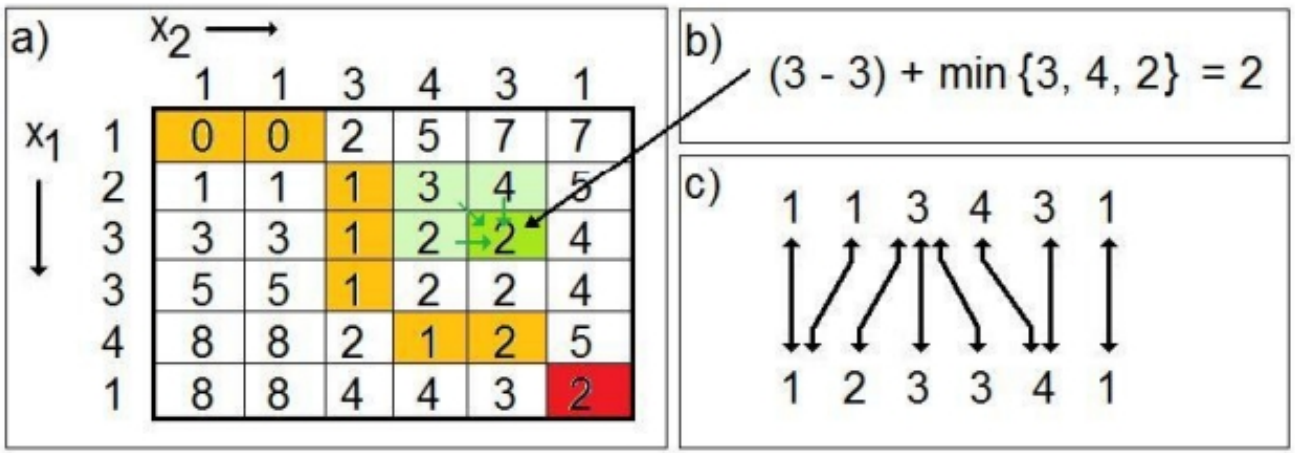
- $D(i, 0) = \infty$ for $1 \leq i \leq n$
- $D(0, j) = \infty$ for $1 \leq j \leq m$
- $D(0, 0) = 0$

The matrix D can be computed in a column-wise or a row-wise fashion. The computation of the whole matrix is needed for getting an optimal warping path.

DTW: Example


a)
x_2
1.1 2 5 5 3.8 2 1.3 0.8
x_1
0.75
2.3
4.1
4
1
3
2
The value in this cell is the cost of transforming the marked part of x_1 into the marked part of x_2.
The cost of transforming the entire x_1 into the entire x_2

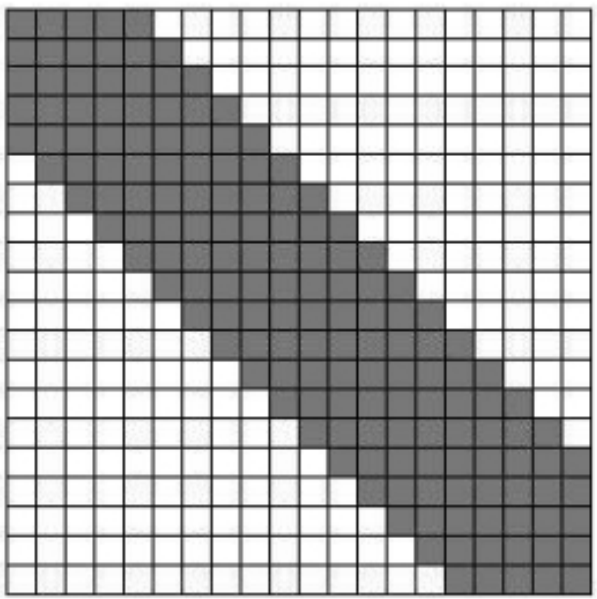

b)
1 8 15 22
2 9 16 23
3 10 17 ...
4 11 18
5 12 19
6 13 20
7 14 21
The positions of the matrix are filled-in according to this order

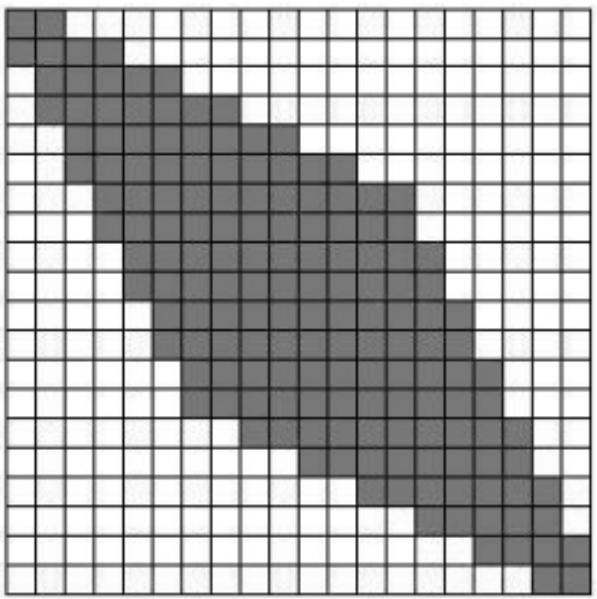

a)
x_2
1 1 3 4 3 1
x_1
1 0 0 2 5 7 7
2 1 1 1 3 4 5
3 3 3 1 2 2 4
3 5 5 1 2 2 4
4 8 8 2 1 2 5
1 8 8 4 4 3 2
b)
(3 - 3) + min {3, 4, 2} = 2
c)
1 1 3 4 3 1
1 2 3 3 4 1

Avoiding the computation of the entire D

Restricting the size of the warping window to a pre-defined constant θ

- $D(i, j)$ is calculated only for those cells (i, j) for which $|i - j| \leq \theta$.
- restricting the computation "near" to the main diagonal of D.
- **constant window size** or **Itakura Parallelogram**





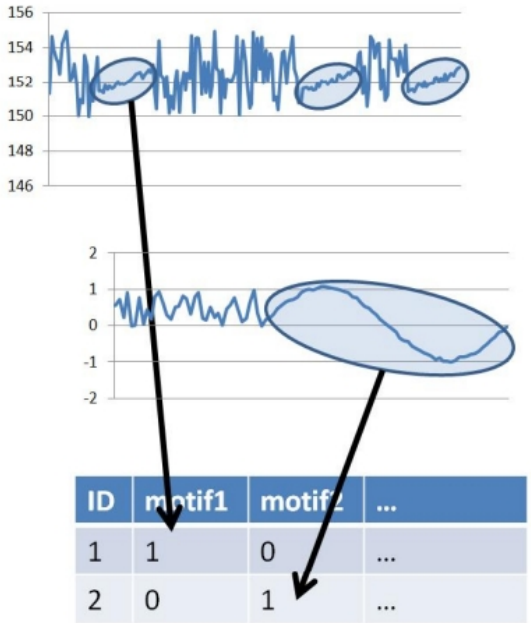
Time-Series Classification

Conventional classification techniques can be used depending on the time-series representation and the distance measure used

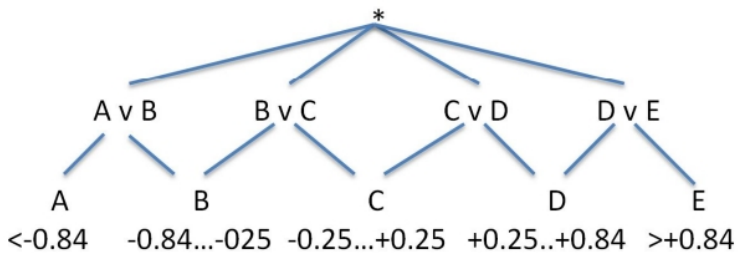
- Instance-based
  - employing DTW
  - empirically showed that using nearest-neighbor classifiers with DTW distance measure performs very well
- Memory-based techniques
  - TS should be represented as vectors of the same length
    - PAA, SAX, first k koefficients of DFT, HWT, ...
    - Other features derived from TS, as e.g. average, min, max, **motifs**, **wildcards**, ...

Motifs & Wildcards

Motifs – characteristical, recurrent patterns in time series



Wildcards – constructed from the taxonomy of symbols

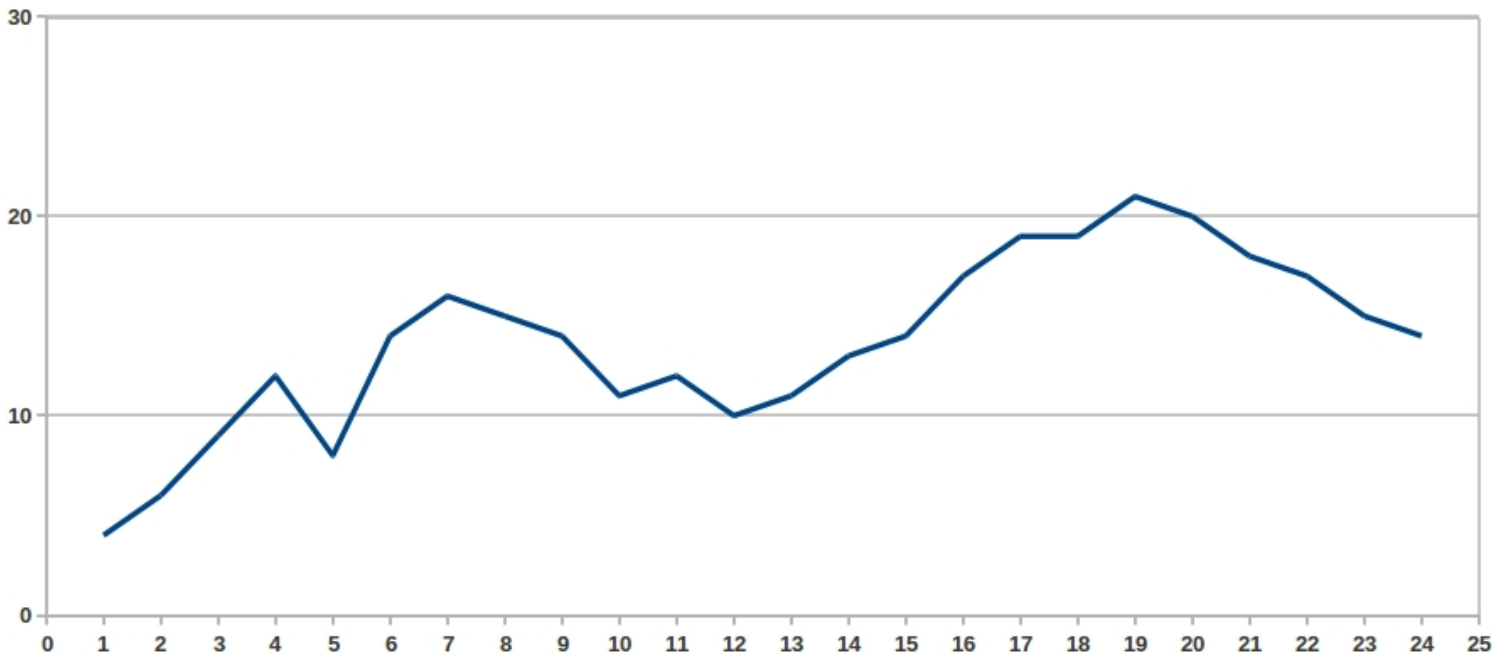


Forecasting: Example

of new customers of a small company

Year	Month											
	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
1	4	6	9	12	8	14	16	15	14	11	12	10
2	11	13	14	17	19	19	21	20	18	17	15	14

How many customers will join the company in March of the Year 3?



Seasonality, Trend & Noise in TS

The main components time series usually consist of are

- Noise
  - a "random" fluctuation in time series
  - we cannot explain, thus, it is hard to predict
- Trend
  - the number of new customers are growing from Year to Year
  - possible to detect
- Seasonality
  - more new customers are joining the company in summer
  - possible to detect

In case of no trend and seasonality, a simple smoothing (e.g. averaging) could be eligible to forecast, while if a trend is present, some regression techniques could be used.

Forecasting: A straight approach (1)

Getting the seasonality and the trend

- Compute the **Seasonal Indices**
  1. compute the ratios of the value of each month to the average value of the corresponding Year[1]
  2. compute the average ratios[2] for given months.

# of new customers of a small company

Year	Month											
	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
1	4	6	9	12	8	14	16	15	14	11	12	10
2	11	13	14	17	19	19	21	20	18	17	15	14
SI	0.52	0.67	0.84	1.06	0.94	1.22	1.37	1.29	1.19	1.02	1.00	0.88

- Compute the trend using regression from total Yearly sales

Year (Y)	Total sales (T)
1	131
2	198

$$T = 67 \cdot Y + 64$$

[1] Average values are 10.92 and 16.5 for Years 1 and 2, respectively.

[2] The SI for March is $(9/10.92 + 14/16.5)/2 = 0.84$.

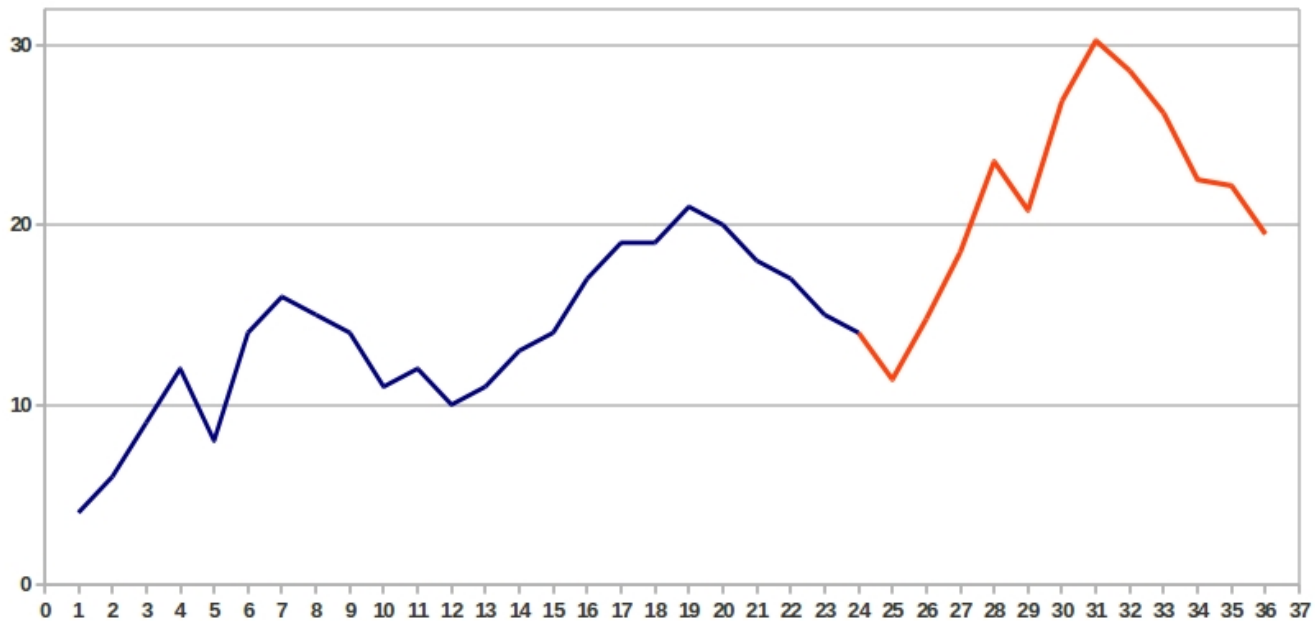
Forecasting: A straight approach (2)

Forecasting for March of the Year 3

1. Predict the total number of new customers for the Year 3

$$T = 67 \cdot 3 + 64 = 265$$

2. compute average monthly value for the Year 3, i.e. $265/12 = 22.08$
3. multiply the average monthly value for the Year 3 with the seasonal index for March to get the forecast, i.e. $22.08 \cdot 0.84 = 18.55$



Smoothing

In our previous example, we computed the trend and the seasonality, however, didn't count with the noise.

- smoothing the time series would be beneficiary for "getting rid" of the noise

The most simplest smoothing techniques are

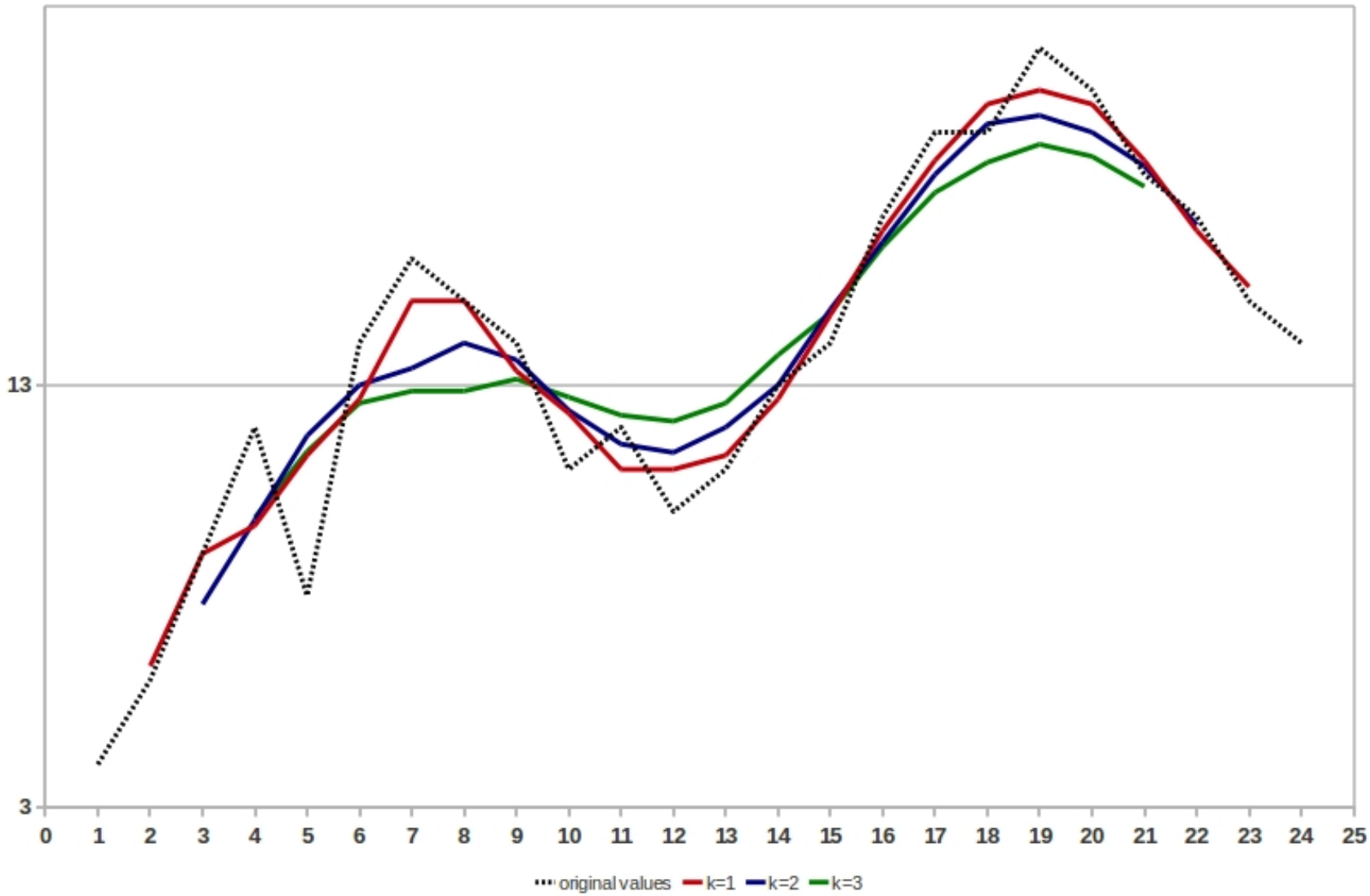
- **Moving Average**

$$\tilde{x}_t = \frac{x_t + x_{t-1} + \cdots + x_{t-n+1}}{N}$$

- **Centered Moving Average**

$$\tilde{x}_t = \frac{x_{t-k} + x_{t-k+1} \cdots + x_t + x_{t+1} + \cdots + x_{t+k}}{2k+1}$$

Moving Average: Example



Single Exponential Smoothing

If there is **no significant trend nor seasonality** in a time series, we can use its average to estimate future values.

- A simple average weights each value equally, however the values which are far from the smoothed one should have less weights.

Single Exponential Smoothing (SES) weight past observations with exponentially decreasing weights, i.e.

$$\tilde{x}_t = \alpha x_{t-1} + (1-\alpha)\tilde{x}_{t-1}$$

where $0 < \alpha \leq 1$, $t \geq 3$ and $\tilde{x}_2 = x_1$.

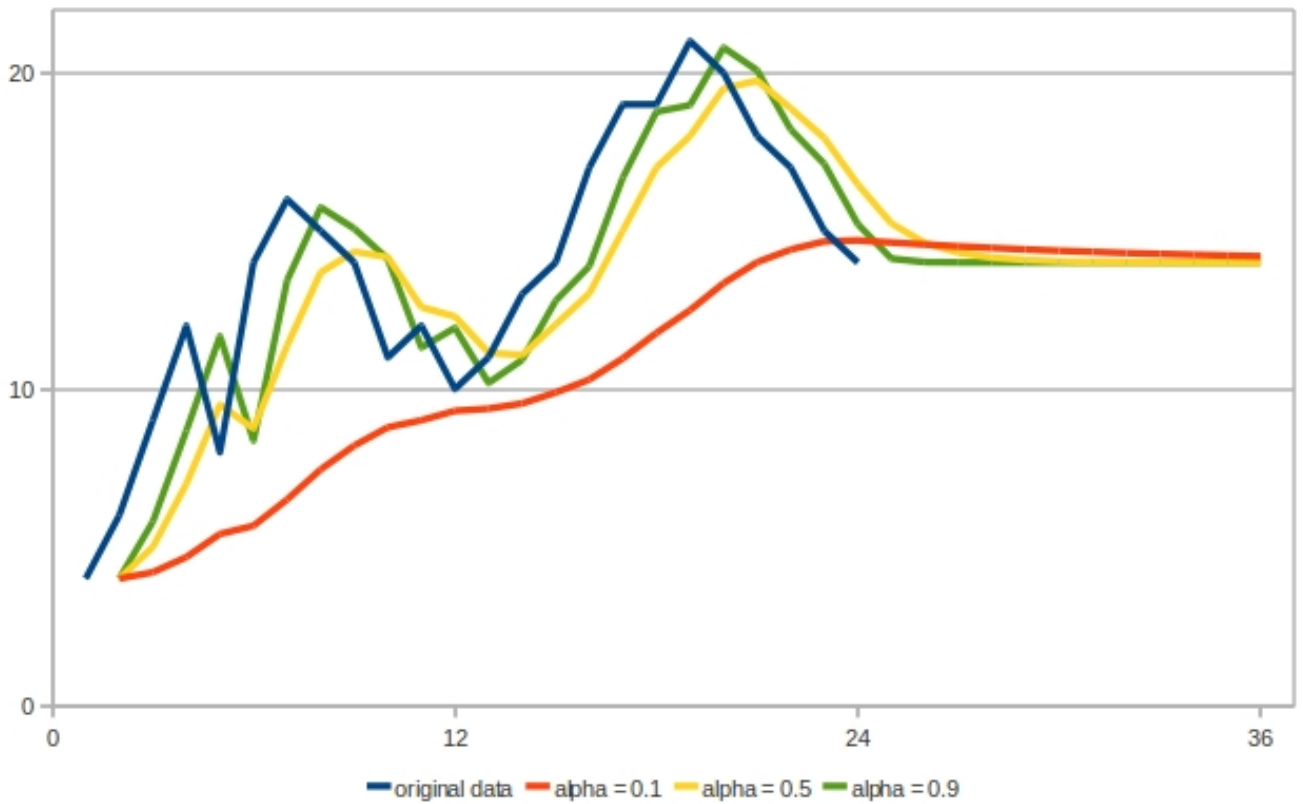
Forecasting with SES is then made in the following way

$$\hat{x}_{t+1} = \alpha x_t + (1-\alpha)\tilde{x}_t = \tilde{x}_t + \alpha(x_t - \tilde{x}_t)$$

SES: Bootstrapping & Example

When there are no actual observations for forecasting, we use the last data point x_n, i.e.

$$\tilde{x}_{n+k} = \alpha x_n + (1-\alpha)\tilde{x}_{n+k-1}, \quad k \geq 1$$



We choose α, which results in a smallest error.

Double Exponential Smoothing

If a **trend is present** in the data, SES doesn't work well. In this case we need to

- adjust the smoothed values to the trend of the previous values, and
- update, and also, smooth the trend simultaneously

Double Exponential Smoothing (DES) is computed as:

$$\tilde{x}_t = \alpha x_t + (1-\alpha)(\tilde{x}_{t-1} + \tilde{r}_{t-1}), \quad 0 \leq \alpha \leq 1$$

$$\tilde{r}_t = \gamma(\tilde{x}_t - \tilde{x}_{t-1}) + (1-\gamma)\tilde{r}_{t-1}, \quad 0 \leq \gamma \leq 1$$

where $t \geq 2$ and $\tilde{r}$ refers to a smoothed trend, initialized e.g. as

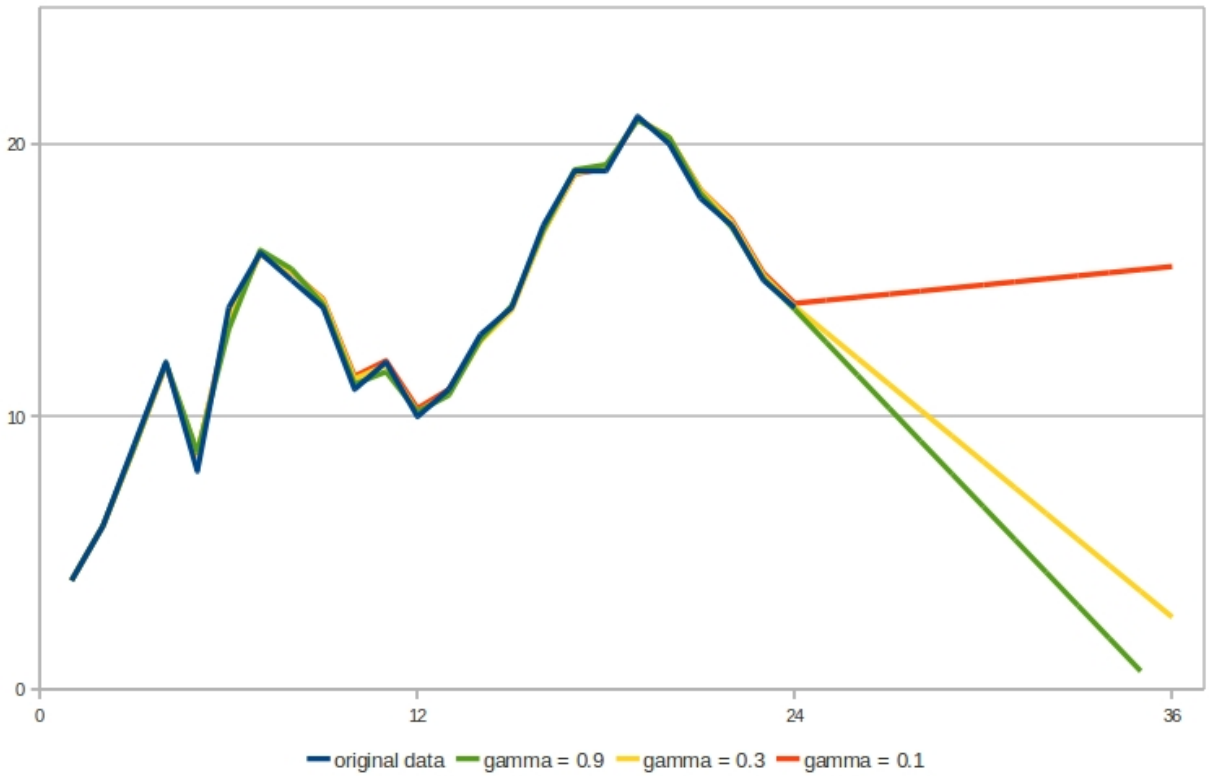
$$\tilde{r}_1 = x_2 - x_1 \qquad \text{or} \qquad \tilde{r}_1 = \frac{x_n - x_1}{n-1}$$

The m-periods-ahead forecast is computed as

$$\hat{x}_{t+m} = \tilde{x}_t + m \cdot \tilde{r}_t$$

DES: Example

The results of DES for $\alpha = 0.9$ and different values for γ



We choose α, γ, which results in a smallest error when comparing the original series to one-step-ahead forecast

- Since we use the current value of the time series to compute the smoothed value

The Holt-Winters Method

The Holt-Winters method (HWM) is computed as:

$$\tilde{x}_t = \alpha \frac{x_t}{\tilde{s}_{t-L}} + (1-\alpha)(\tilde{x}_{t-1} + \tilde{r}_{t-1}), \quad 0 \leq \alpha \leq 1$$

$$\tilde{r}_t = \gamma(\tilde{x}_t - \tilde{x}_{t-1}) + (1-\gamma)\tilde{r}_{t-1}, \quad 0 \leq \gamma \leq 1$$

$$\tilde{s}_t = \beta \frac{x_t}{\tilde{x}_t} + (1-\beta)\tilde{s}_{t-L}, \quad 0 \leq \beta \leq 1$$

where $t > L$, L is the length of a season and $\tilde{s}$ refers to a sequence of smoothed seasonal indices, initialized[1] e.g. as was showed before.

The trend factor can be initialized[2] as

$$\tilde{r}_L = \frac{1}{L}\left(\frac{x_{L+1} - x_1}{L} + \cdots + \frac{x_{L+L} - x_L}{L}\right)$$

The m-periods-ahead forecast is computed as

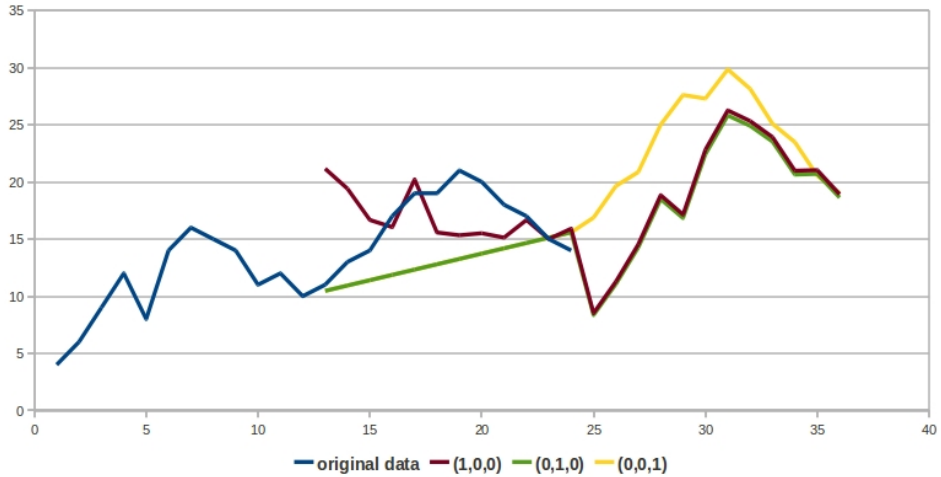
$$\hat{x}_{t+m} = (\tilde{x}_t + m \cdot \tilde{r}_t) \cdot \tilde{s}_{t-L+m}$$

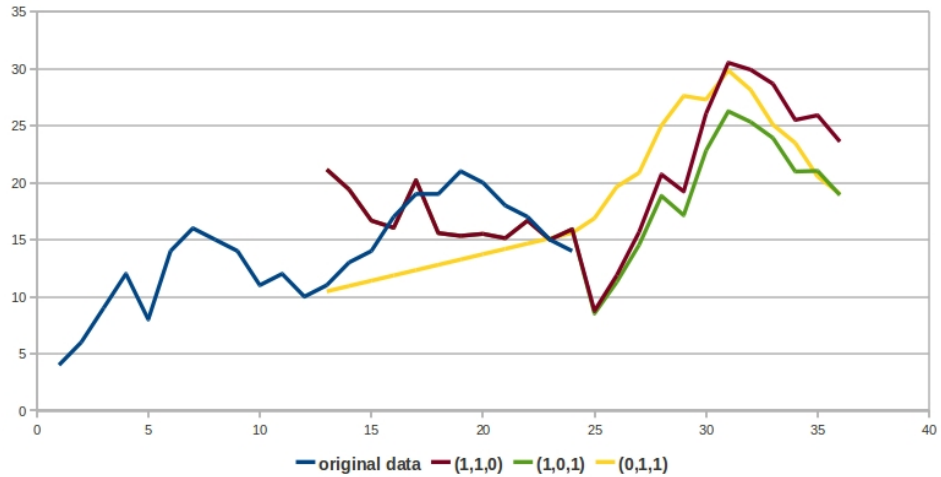
[1] At least one complete season is needed to initialize seasonal indices.

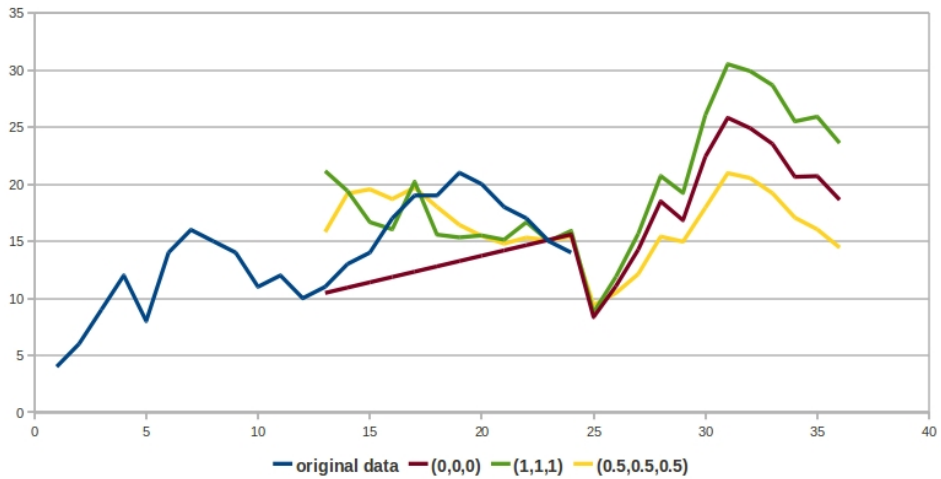
[2] The use of two complete seasons is advisable to initialize the trend factor.

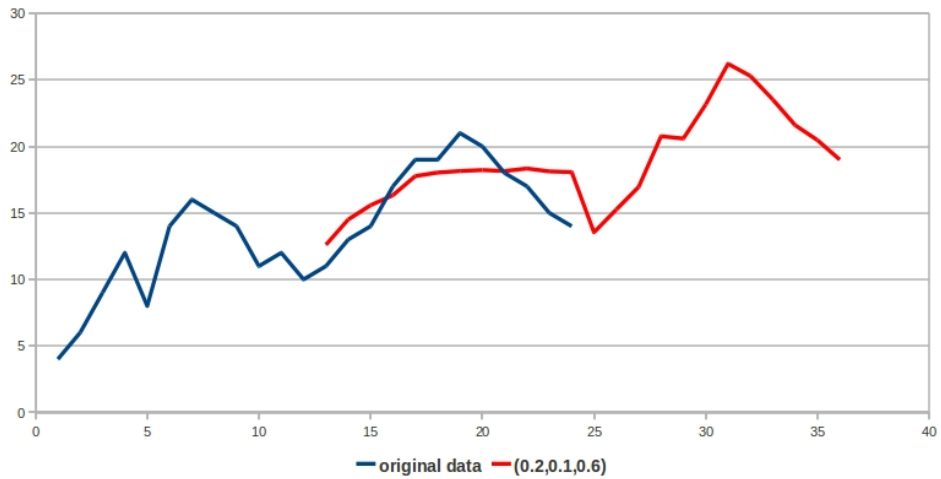
HWM: Example (1)

Results of a simple implementation for different extremes of (α, γ, β)



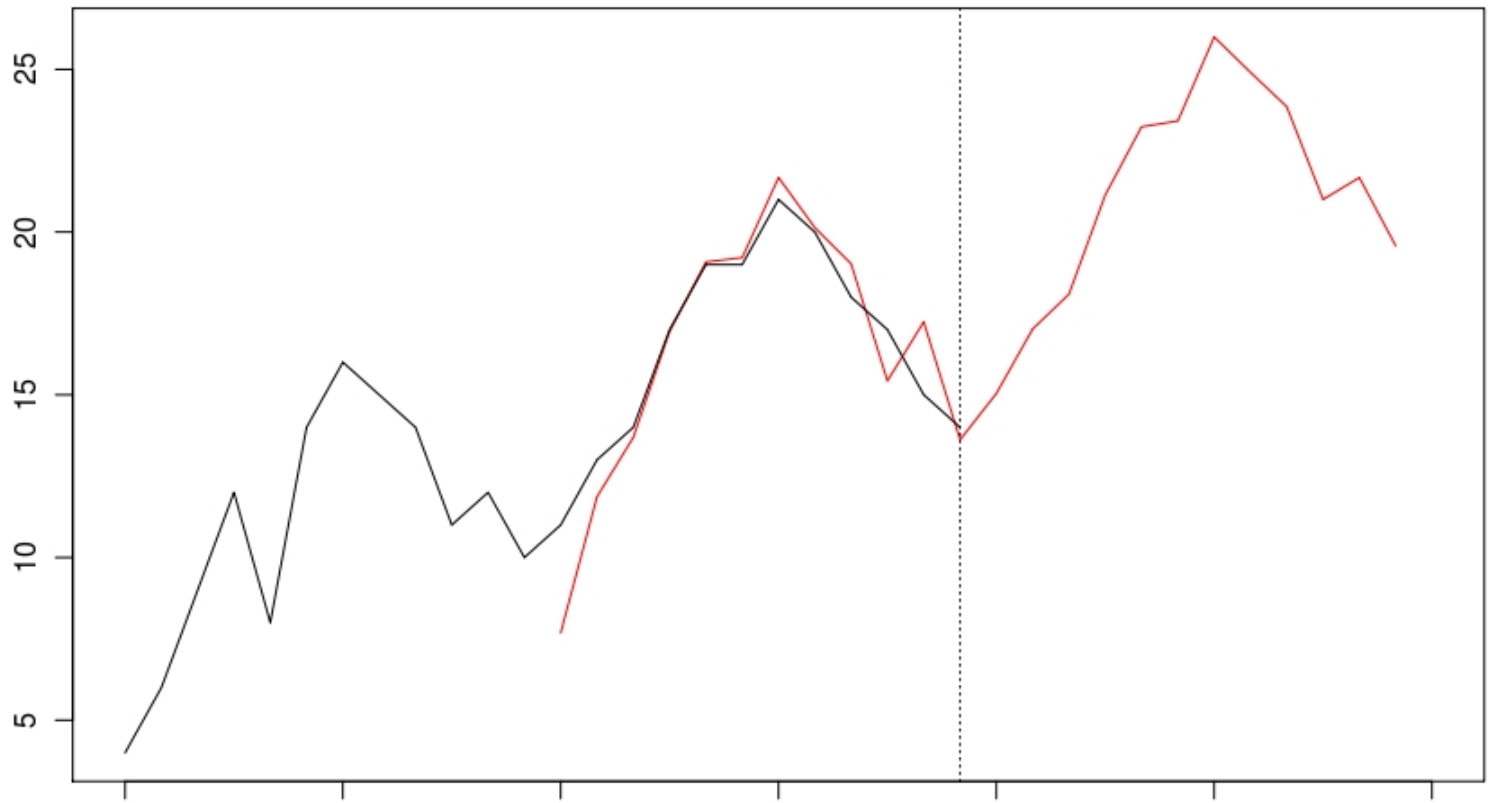






HWM: Example (2)

Different initialization techniques, also the normalization of seasonal factors should be considered when forecasting[1]



[1] See (Chatfield & Yar, 1988) in the references, for more details.

Summary

Representation of time series

- Discrete Fourier & Haar transforms
  - While FT captures the "global" periodic behavior, HT captures both "local and global" character of time series.
- PLA, PAA and SAX
  - One can combine the bottom-up with and Sliding Windows approaches for more efficient computation.
  - Despite its simplicity, PAA is often enough to use. Moreover, its runtime is linear.
  - SAX representation enables us to use "string-processing" techniques such as subsequence matching, etc.
- Also other techniques, for example, Singular Value Decomposition.

Summary

Distance measures

- Static distance measures
  - When time-series are of the same lengths
  - Use different features derived from time series to represent them in a common space
- Dynamic Time Warping
  - Works well even if time series are of different length
  - Computes a "cost of transformation" between two time series
  - Plays an important role in classification of time-series with k-NN
    - Motifs and Wildcards
  - Different variations and speed-ups of DTW
- Forecasting
  - Seasonality, Trend and Noise
  - Simple, Double and the Holt-Winters smoothing
  - Good and well-suited initialization is important
  - Other forecasting methods, as for example ARIMA

References

- Image Processing lecture slides at UHi
  http://www.ismll.uni-hildesheim.de/lehre/ip-08w/script/index_en.html
- E. Keogh, S. Chu, D. Hart and M. Pazzani (2001). An Online Algorithm for Segmenting Time Series. In Proceedings of IEEE International Conference on Data Mining, 289-296.
- J. Lin, E. Keogh, S. Lonardi and B. Chiu (2003). A Symbolic Representation of Time Series, with Implications for Streaming Algorithms. In proceedings of the 8th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery. San Diego, CA.
- B. Lkhagva, Y. Suzuki, K. Kawagoe. Extended SAX: Extension of Symbolic Aggregate Approximation for Financial Time Series Data Representation.
- Meinard Müller. Information Retrieval for Music and Motion, Springer-Verlag, ISBN 978-3-540-74047-6, 2007, 318 p.
- Krisztián Buza. Fusion Methods for Time-Series Classification. PhD thesis, University of Hildesheim, 2011.
- Engineering Statistics Handbook (NIST Sematech)
  http://itl.nist.gov/div898/handbook/
- Ch. Chatfield and M. Yar (1988). Holt-Winters forecasting: some practical issues. The Statistician, 1988 vol. 37, 129-140.
- Time-Critical Decision Making for Business Administration
  http://home.ubalt.edu/ntsbarsh/stat-data/forecast.htm

Thanks for Your attention!

Questions?

horvath@ismll.de