Knowledge Discovery in Databases II

Summer Term 2017

Lecture 3:
Sequence Data

Lectures : Prof. Dr. Peer Kröger, Yifeng Lu
Tutorials: Yifeng Lu

http://www.dbs.ifi.lmu.de/cms/Knowledge_Discovery_in_Databases_II_(KDD_II)

Overview

1. Introduction
2. Sequence Data
3. Time Series Data

Sequence Data

- So far we dealt with mostly structured, “flat” data from relational tables that provide a snapshot of the data at a particular moment (OK, the data can be updated inducing an update of patterns as well …)
- But very often, the world is different: just looking at a snapshot cannot reveal important insights into the (dynamics of) data
- Rather, we need to look at a sequences of snapshots of data to e.g. analyze:
  - How patterns are changing/evolving from one snapshot to the other
  - If certain patterns appear in sequential/periodical fashion
  - If there are “sequential” patterns
  - …

Sequence Data

- ***Sequence Data*** allow for measuring/monitoring phenomena over time (***Time Series Data***) or – more generally – in a given order (of sequential events) without a concrete notion of time
- Examples:
  - Sequence Data: Sequence of purchases
  - Sequential Pattern: Customers buying A are likely to by B within the next 4 transactions

  - Time Series Data: Stock rates over time
  - Pattern: find stocks with similar behavior (over the entire time frame or in a sub-interval of time)

Overview

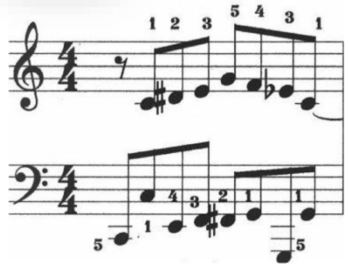
1. Introduction
2. Sequence Data
3. Time Series Data

Sequences

- A sequence S of length n is a mapping of the index set $I_n = \{1,2,\ldots,n\}$ into a domain O:
  $$S: I_n \rightarrow O$$
- The set of all sequences of length n is $O^n = O^{I_n} = \{I_n \rightarrow O\}$
- The set of all sequences over domain O is $O^* = \{I_n \rightarrow O \mid n \in \mathbb{N}_0\}$
- Sequences can be classified by their domain
  - Categorical values (nominal values, alphabets, enumeration types)
  - Continuous values (real numbers)

Sequences

- Examples:
  - Text data $\{a, \dots, Z, 0, \dots 9, \dots\}^*$
  - Video data $images^*$
  - Music data $notes^*$
  - Protein sequences $amino\_acid^* = \{\mathrm{LEU}, \mathrm{ARG}, \dots\}^*$
  - Gene sequences $nucleic\_acid^* = \{\mathrm{C}, \mathrm{G}, \mathrm{A}, \mathrm{T}\}^*$
  - …



- Time series are of course special types of sequences

Similarity Models for Sequences

- The most important question: how to account for the sequential nature of the data???
- We can use similarity models that do the job, e.g.:
- Hamming Distance
  - Simple approach similar to the Euclidean Distance on vector data
  - Naïve alignment of sequences
- Edit Distance
  - Transformation-based approach that measures the edit costs for transforming one sequence into another
  - Byproduct: (Optimal) alignment of sequences
- Longest Common Subsequences (LCSS)
  - Utilization of a third common basis sequence
  - Variant of the edit distance

Hamming Distance

- Hamming Distance counts the number of positions with different elements
  - It thus accounts for the fact that objects are “sequences of some symbols”
- Given two sequences $Q = (q_1, \dots, q_n)$ and $S = (s_1, \dots, s_n)$ of the same length, the Hamming Distance between Q and S is defined as:

$$\mathrm{D}_{Hamming}(Q,S) = \sum_{i=1}^{n} \delta(q_i, s_i) \quad \text{with} \quad \delta(x,y) = \begin{cases} 0 & if\ x = y \\ 1 & else \end{cases}$$

- Example:

Q = t h r e e

| x x | x

S = t r e e .

| = match, x = mismatch

⇨ $\mathrm{D}_{Hamming}(Q,S) = 3$

Drawback:

- Very strict matching similar to the Euclidean Distance
- Similar subsequences are not considered (aligned appropriately)

Hamming Distance: Further Example

- Consider the following sequences (in German):

$$\begin{array}{lcllllllllll} Q & = & T & \text{Ü} & R & S & C & H & L & O & S & S \\ S & = & T & O & R & S & C & H & U & S & S & . \end{array}$$

→ $\mathrm{D}_{Hamming}(Q,S) = 4$

$$\begin{array}{lcllllllllll} Q & = & T & \text{Ü} & R & S & C & H & L & O & S & S \\ R & = & A & B & S & C & H & U & S & S & . & . \end{array}$$

→ $\mathrm{D}_{Hamming}(Q,R) = 10$

- Similarity of subsequences $SCHLOSS$ and $SCHUSS$ is not considered

Edit Distance [L66]

- **Idea**:
  - Dissimilarity between two sequences is defined as the ***minimal*** number of edit operations (insertions, deletions, substitutions) for transforming one sequence into another
- **Example**:
  - Given the following two sequences Q and S, two deletions (◊) and three substitutions (:) are necessary for the transformation
  - Five symbols are unmodified (|):

$$
\begin{array}{cccccccccccc}
Q & = & T & \ddot{U} & R & S & C & H & L & O & S & S \\
 & & \lozenge & : & : & | & | & | & \lozenge & : & | & | \\
S & = & & A & B & S & C & H & & U & S & S
\end{array}
$$

  - $\mathrm{D}_{Edit}(Q,S) = 5$
- The mapping between elements is called **optimal alignment** and the Edit Distance represents the **alignment cost**

Edit Distance: Formal Definition

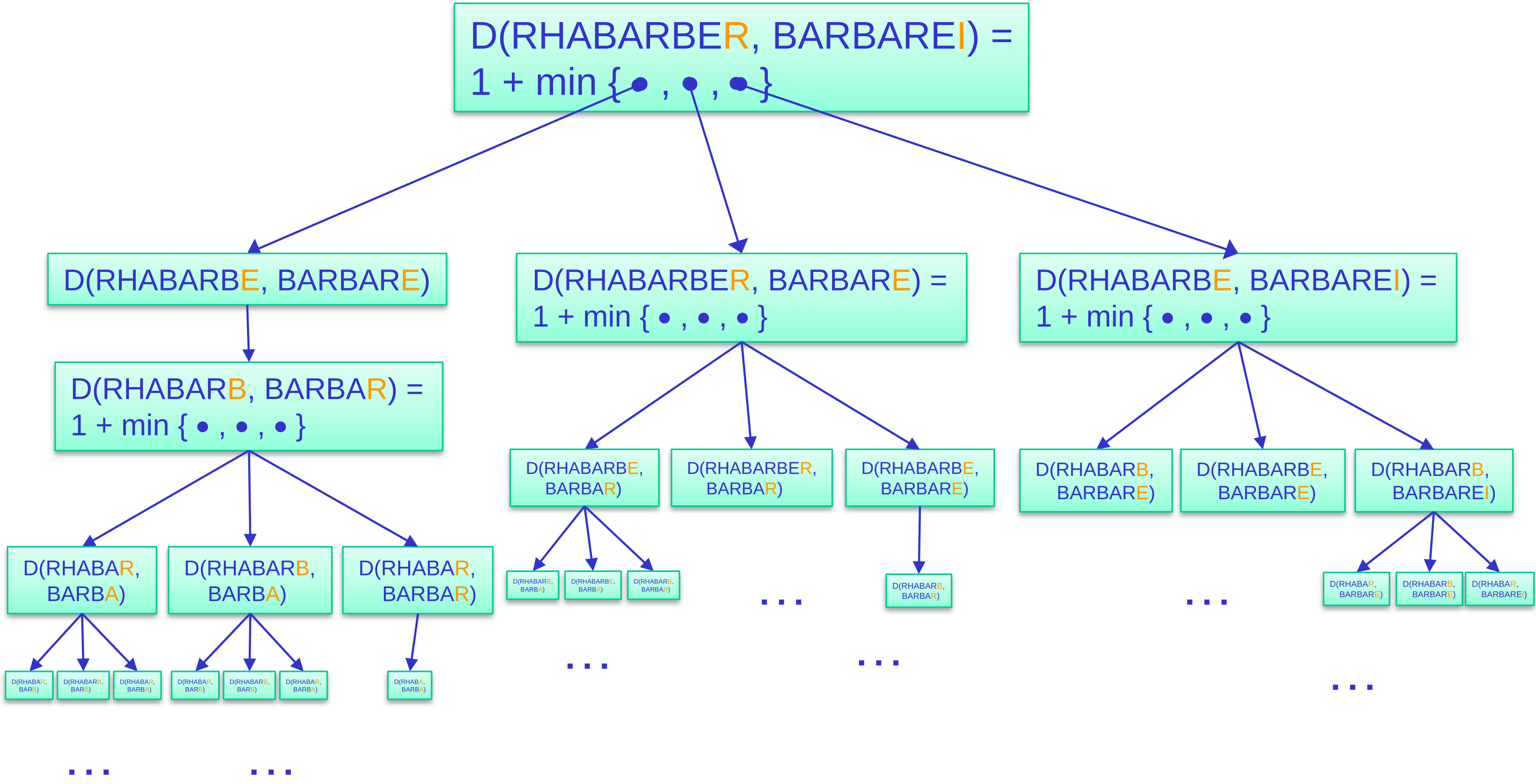
- Given a sequence $Q = (q_1, \dots, q_n)$ let $start(Q) = (q_1, \dots, q_{n-1})$ denote the prefix of Q and $last(Q) = q_n$ the last element of Q.

- Given two sequences $Q = (q_1, \dots, q_n)$ and $S = (s_1, \dots, s_m)$, the **Edit Distance ED** of Q and S is defined as:

$$ED(Q,S) = \begin{cases} n & \text{if } m = 0 \\ m & \text{if } n = 0 \\ ED\big(start(Q), start(S)\big) & \text{if } last(Q) = last(S) \\ 1 + min \begin{cases} ED\big(start(Q), start(S)\big), \\ ED\big(Q, start(S)\big), \\ ED(start(Q), S) \end{cases} & else \end{cases}$$

- Remark: if no insertions or deletions occur, the Edit Distance is equivalent to the Hamming Distance

Naïve Computation of the Edit Distance



D(RHABARBER, BARBAREI) =
1 + min { • , • , • }

D(RHABARBE, BARBARE)

D(RHABARB, BARBAR) =
1 + min { • , • , • }

D(RHABAR, BARBA)
D(RHABARB, BARBA)
D(RHABAR, BARBAR)

D(RHABAR, BARB)
D(RHABARB, BARB)
D(RHABAR, BARBA)
D(RHABAR, BARB)
D(RHABARB, BARB)
D(RHABAR, BARBA)
D(RHABA, BARBA)

. . .
. . .

D(RHABARBER, BARBARE) =
1 + min { • , • , • }

D(RHABARBE, BARBAR)
D(RHABARBER, BARBAR)
D(RHABARBE, BARBARE)

D(RHABARB, BARBA)
D(RHABARBE, BARBA)
D(RHABARB, BARBAR)

. . .

D(RHABARB, BARBAR)

. . .
. . .

D(RHABARBE, BARBAREI) =
1 + min { • , • , • }

D(RHABARB, BARBARE)
D(RHABARBE, BARBARE)
D(RHABARB, BARBAREI)

. . .

D(RHABAR, BARBARE)
D(RHABARB, BARBARE)
D(RHABAR, BARBAREI)

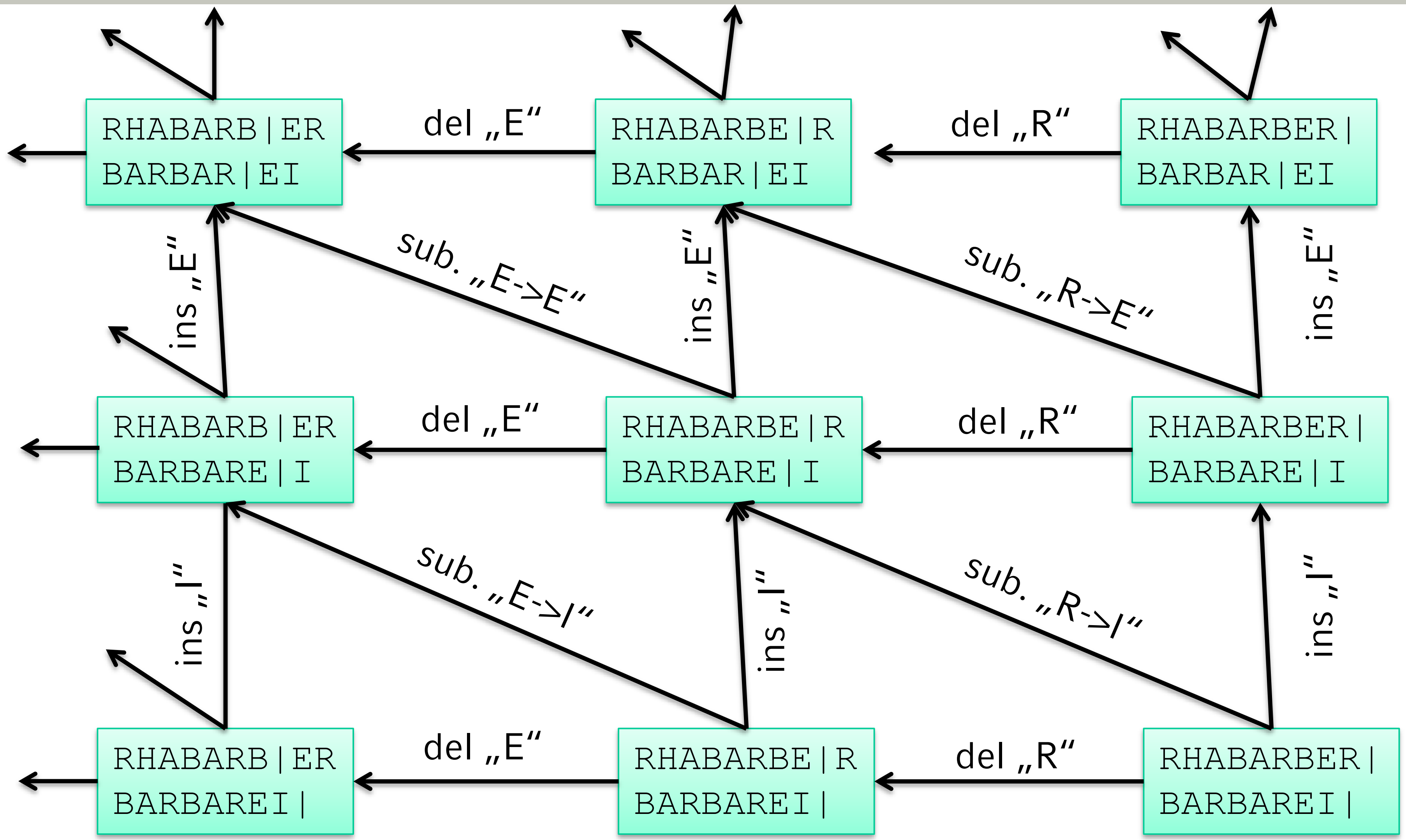
. . .

For sequences of lengths n, m, this tree has $\mathcal{O}(3^{n+m})$ nodes

Acceleration of Computation

- **Analysis**
  - $\mathcal{O}(3^{n+m})$ function calls for sequences of lengths n, m
  - Many calls appear repeatedly
  - There are only $(m+1) \cdot (n+1) = \mathcal{O}(m \cdot n)$ ***different*** recursive calls
- **Solution**
  - Store results of all calls: $\mathcal{O}(m \cdot n)$ space
  - Systematic evaluation with $\mathcal{O}(m \cdot n)$ operations
  - Scheme is called **dynamic programming**
- **Acceleration** (Example: $m, n = 5, 50, 500$)
  - $5 \cdot 5 = 25$ instead of $3^{10} = 59{,}049$
  - $50 \cdot 50 = 2{,}500$ instead of $3^{100} \approx 5{,}154 \cdot 10^{47}$
  - $500 \cdot 500 = 250{,}000$ instead of $3^{1000} \approx 1{,}322 \cdot 10^{477}$

Dynamic Programming Scheme



RHABARB | ER
BARBAR | EI

del „E“

RHABARBE | R
BARBAR | EI

del „R“

RHABARBER |
BARBAR | EI

ins „E“

sub. „E->E“

ins „E“

sub. „R->E“

ins „E“

RHABARB | ER
BARBARE | I

del „E“

RHABARBE | R
BARBARE | I

del „R“

RHABARBER |
BARBARE | I

ins „I“

sub. „E->I“

ins „I“

sub. „R->I“

ins „I“

RHABARB | ER
BARBAREI |

del „E“

RHABARBE | R
BARBAREI |

del „R“

RHABARBER |
BARBAREI |

Dynamic Programming

- Calculation scheme:
  - **Horizontal step**: $(i,j) \rightarrow (i-1,j)$
    - deletion of current character q_i in Q
  - **Vertical step**: $(i,j) \rightarrow (i,j-1)$
    - insertion of character s_i in Q at position i
  - **Diagonal step**: $(i,j) \rightarrow (i-1,j-1)$
    - substitution of current character q_i in Q and s_i in S

$$\begin{tikzcd}
(i-1,j-1) & (i,j-1) \\
(i-1,j) & (i,j) \arrow[l] \arrow[u] \arrow[ul]
\end{tikzcd}$$

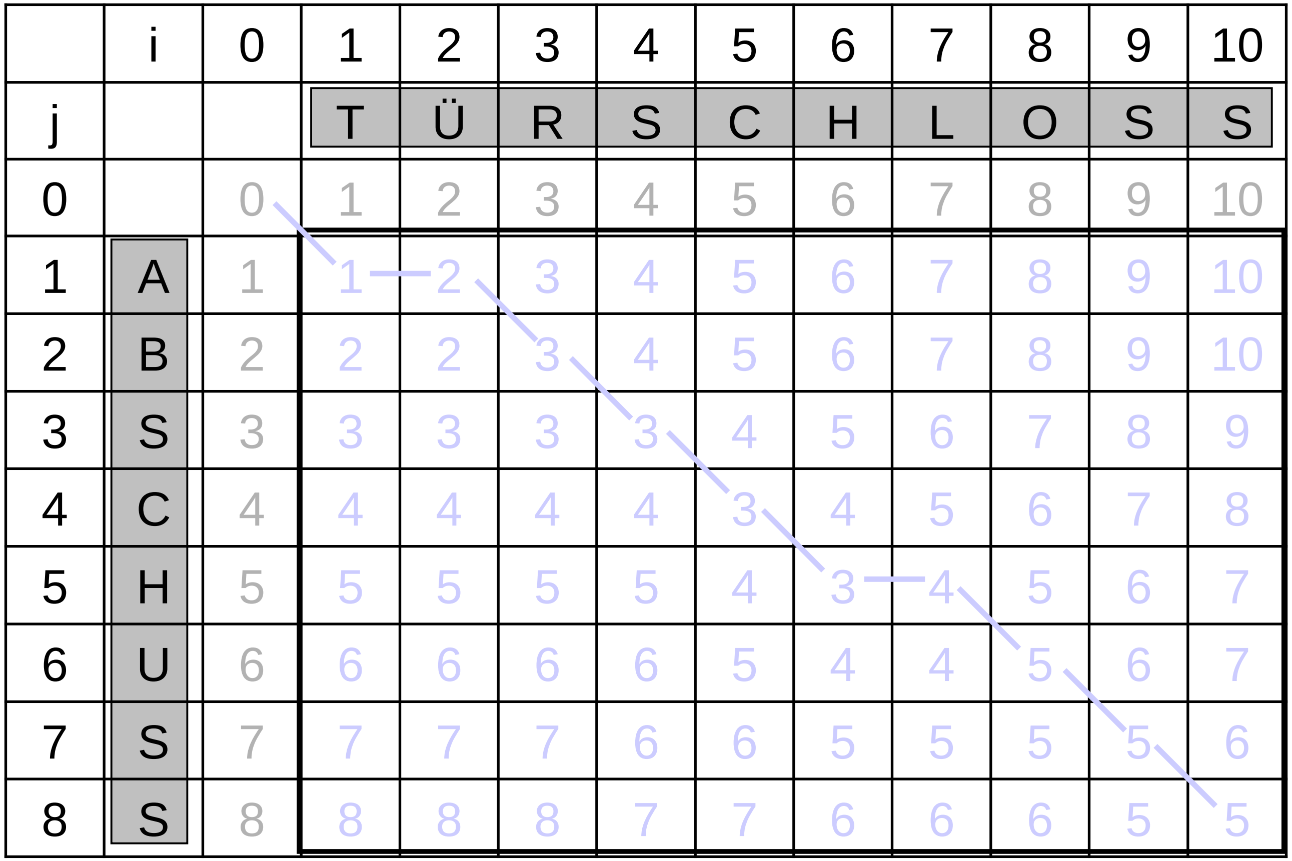
- All possible solutions, i.e. the Edit Distance on subsequences, can be stored within a matrix, following the paradigm of dynamic programming
- A cost minimal path through this matrix from $(0,0)$ to (n,m) yields the Edit Distance (alignment cost and optimal alignment)

  (Note the **non-**determinism: there may be several cost minimal paths/optimal alignments)

- Optimal alignment is obtained by backward reconstruction of the decisions made at every step along the optimal path (decisions can be stored during matrix construction)

Edit Distance: Example of Dynamic Programming

- Computation of the Edit Distance via dynamic programming:



	i	0	1	2	3	4	5	6	7	8	9	10
j			T	Ü	R	S	C	H	L	O	S	S
0		0	1	2	3	4	5	6	7	8	9	10
1	A	1	1	2	3	4	5	6	7	8	9	10
2	B	2	2	2	3	4	5	6	7	8	9	10
3	S	3	3	3	3	3	4	5	6	7	8	9
4	C	4	4	4	4	4	3	4	5	6	7	8
5	H	5	5	5	5	5	4	3	4	5	6	7
6	U	6	6	6	6	6	5	4	4	5	6	7
7	S	7	7	7	7	6	6	5	5	5	5	6
8	S	8	8	8	8	7	7	6	6	6	5	5

```
T Ü R S C H L O S S
: ◊ : | | | ◊ : | |
A   B S C H   U S S
```

Weighted Edit Distance

- **Idea**: Weighting of edit operations via a ground distance
  - Different costs for insertions, deletions, and substitutions
- Given two sequences $Q = (q_1, \dots, q_n)$ and $S = (s_1, \dots, s_m)$, the **Weighted Edit Distance w.r.t. a ground distance $\boldsymbol{\delta}$** between Q and S is defined as:

$$ED_\delta(Q,S) = \begin{cases} 0 & \text{if } n = m = 0 \\ \sum_{i=1}^{n} \delta(q_i, \Diamond) & \text{if } m = 0 \\ \sum_{i=1}^{m} \delta(\Diamond, s_i) & \text{if } n = 0 \\ ED_\delta\big(start(Q), start(S)\big) & \text{if } last(Q) = last(S) \\ min \begin{cases} ED_\delta\big(start(Q), start(S)\big) + \delta\big(last(Q), last(S)\big), \\ ED_\delta\big(Q, start(S)\big) + \delta\big(\Diamond, last(S)\big), \\ ED_\delta(start(Q), S) + \delta(last(Q), \Diamond) \end{cases} & else \end{cases}$$

DATABASE
SYSTEMS
GROUP

Edit Distance: Properties

- The optimal alignment of two sequences is not necessarily unique:

```
B A N A N A          B A N A N A
| | | : ◊ ◊          | ◊ ◊ | | :
B A N D              B     A N D
```

- Edit Distance is a metric
- Weighted Edit Distance is a metric if the ground distance is a metric
- Computation time complexity of a single Edit Distance computation is in $\mathcal{O}(n \cdot m)$ for sequences of lengths n, m
- Common variant: First deletion of a symbol more expensive than repeated deletion (important in bioinformatics)

Longest Common Subsequence (LCSS) [CLR+09]

- **Idea**: Similarity between two sequences Q and S is defined as the length of a third sequence Z which contains elements of Q and S in the same order
  - The longer the sequence Z, the higher the similarity of Q and S and vice versa

- **Example** (DNA sequence):

  Q: ACCGGTCGAGTGCGCGAAGCCGGCCGAA

  S: GTCGTTCGGAATGCCGTTGCTCTGTAA

  One possible solution:

  Z: GTCGTCGGAAGCCGGCCGAA

Definition: Subsequence

- A sequence $Z = (z_1, \dots, z_k)$ is a **subsequence** of sequence $Q = (q_1, \dots q_n)$ if there exists a strictly increasing sequence $i_1, i_2, \dots, i_k$ of indices of Q such that $\forall j = 1,2, \dots, k$ it holds that $q_{i_j} = z_j$
- **Example**:
  - Let $Q = (A, B, C, B, D, A, B)$ be a sequence
  - The sequence $Z = (B, C, D, B)$ is a subsequence of Q
  - The corresponding index sequence is 2,3,5,7

Definition: Common Subsequence

- A sequence $Z = (z_1, \dots, z_k)$ is a **common subsequence** of two sequences $Q = (q_1, \dots q_n)$ and $S = (s_1, \dots, s_m)$ if Z is a subsequence of both Q and S
- **Example**:
  - Let $Q = (A, B, C, B, D, A, B)$ be a sequence
  - Let $S = (B, D, C, A, B, A)$ be another sequence
  - The sequence $Z = (B, C, A)$ is a common subsequence of Q and S
  - However, Z is not the longest common subsequence:
    - $Z' = (B, C, B, A)$
    - $Z'' = (B, D, A, B)$

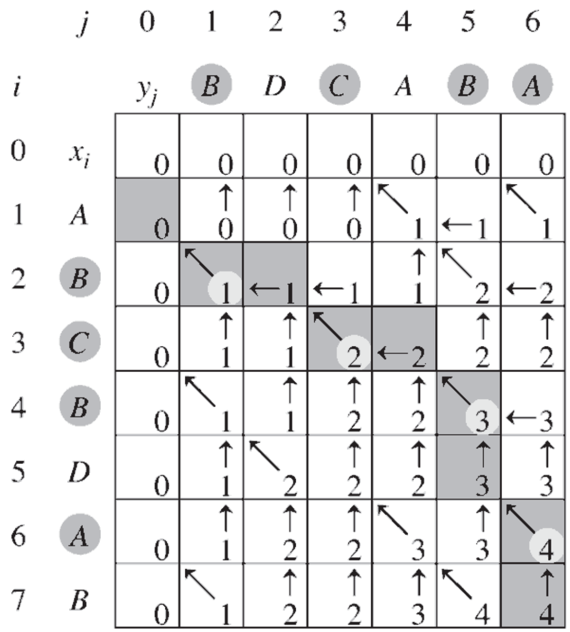
- Given two sequences $Q = (q_1, \dots q_n)$ and $S = (s_1, \dots, s_m)$, the **longest common subsequence problem** is to find a maximum-length common subsequence $Z = (z_1, \dots, z_k)$ of Q and S

Longest Common Subsequence (LCSS)

- Given two sequences $Q = (q_1, \dots q_n)$ and $S = (s_1, \dots, s_m)$, the **longest common subsequence (similarity measure)** is defined as:

$$LCSS(Q,S) = \begin{cases} 0 & if\ n = 0 \vee m = 0 \\ LCSS(Start(Q), Start(S)) + 1 & if\ Last(Q) = Last(S) \\ max \begin{cases} LCSS(Start(Q), S) \\ LCSS(Q, Start(S)) \end{cases} & else \end{cases}$$

- **Properties**:
  - Computation similar to that of the Edit Distance
  - Exponential computation time complexity
  - Computation time complexity via dynamic programming lies in $\mathcal{O}(n \cdot m)$



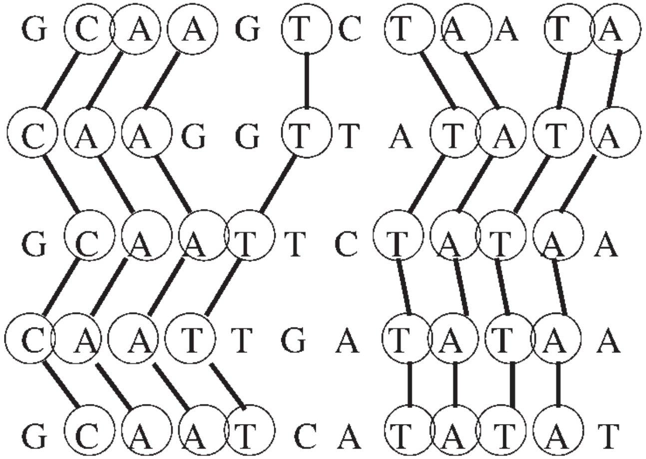
	j	0	1	2	3	4	5	6
i		y_j	B	D	C	A	B	A
0	x_i	0	0	0	0	0	0	0
1	A	0	↑ 0	↑ 0	↑ 0	↖ 1	← 1	↖ 1
2	B	0	↖ 1	← 1	← 1	↑ 1	↖ 2	← 2
3	C	0	↑ 1	↑ 1	↖ 2	← 2	↑ 2	↑ 2
4	B	0	↖ 1	↑ 1	↑ 2	↑ 2	↖ 3	← 3
5	D	0	↑ 1	↖ 2	↑ 2	↑ 2	↑ 3	↑ 3
6	A	0	↑ 1	↑ 2	↑ 2	↖ 3	↑ 3	↖ 4
7	B	0	↖ 1	↑ 2	↑ 2	↑ 3	↖ 4	↑ 4

Similarity based on LCSS & Generalization

- LCSS provides the length of the longest common subsequence
  - Highly dependent on the length of the analyzed sequences
  - Not a distance function
- **Distance function** based on LCSS between two sequences $Q = (q_1, \ldots q_n)$ and $S = (s_1, \ldots, s_m)$:

$$\mathrm{D}_{LCSS}(Q,S) = 1 - \frac{LCSS(Q,S)}{\min(n,m)}$$

- Generalization of LCSS [S08]:
  - Multiple alignment between several sequences
  - Complexity: $O(2^k n^k)$ for k sequences and n = length of longest sequence



Mining Sequence Data

- Distance-based data mining
  - Use one of the similarity measures from above (or variants, or …)
  - Clustering, outlier detection, classification of sequence data
  - Does not mine sequential patterns but only patterns of similar sequences

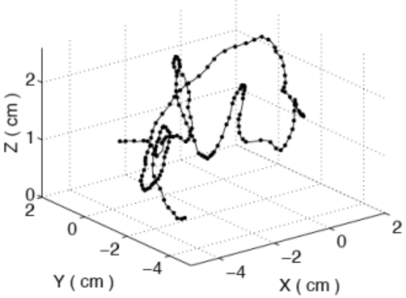
- Sequential pattern mining (see previous lecture)
  - Count the frequency of subsequences in the sequence objects and report the frequent ones (sequential patterns)
  - Relation to (generalization of) frequent item set mining, thus:
  - Algorithms very similar to frequent item set mining

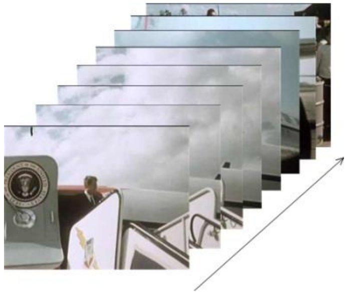
Overview

1. Introduction
2. Sequence Data
3. Time Series Data

Time Series

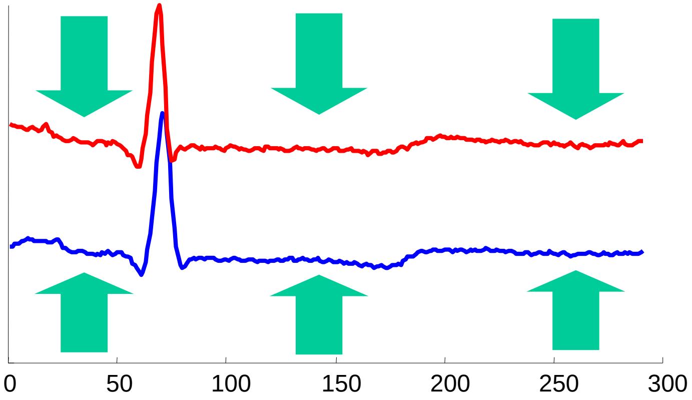
- Time series are a special type of sequences
  - Typically, values that are recorded over time
  - Index set I_n represents specific points in time
- Examples for **univariate time series**:
  - stock prices
  - audio data
  - temperature curves
  - ECG
  - amount of precipitation
- Examples for **multivariate time series**:
  - trajectories (spatial positions)
  - video data (e.g., color histograms)
  - combinations of sensor readings

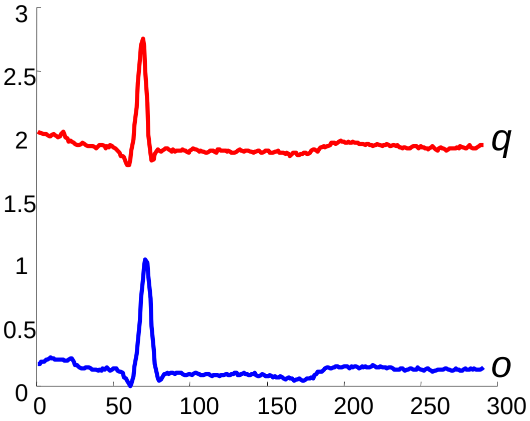




Time Series

- Data Cleaning to remove artefacts, distortion, noise, …
  - Offset Translation (aka “Shifting”)
    - Time series are similar but have different offsets
    - Example: move each time series by its mean *M*

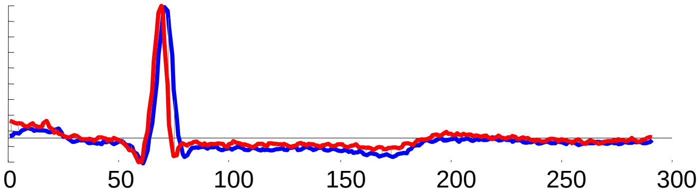




$$q = q - M(q)$$

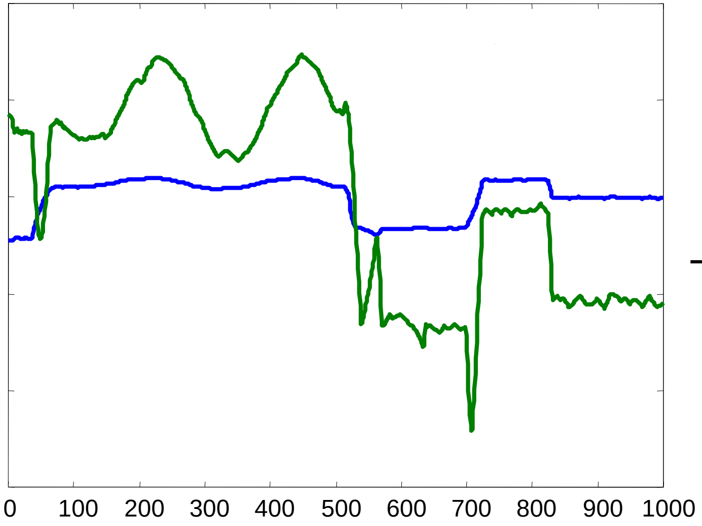
$$\xrightarrow{\qquad\qquad}$$

$$o = o - M(o)$$



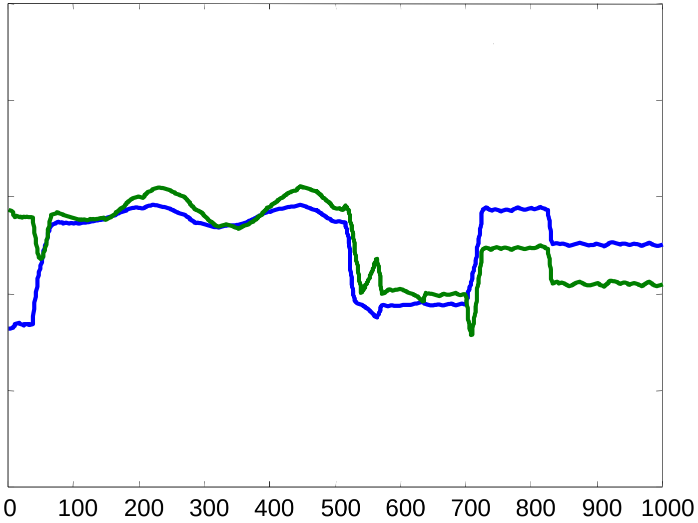
Time Series

- Data Cleaning (cont.)
  - (Amplitude) Scaling
    - Time series have similar trends but have different amplitudes
    - Example: move each time series by its mean *M* and normalize the amplitude by its standard deviation *S* (this is also called "normalization" = shifting + scaling)



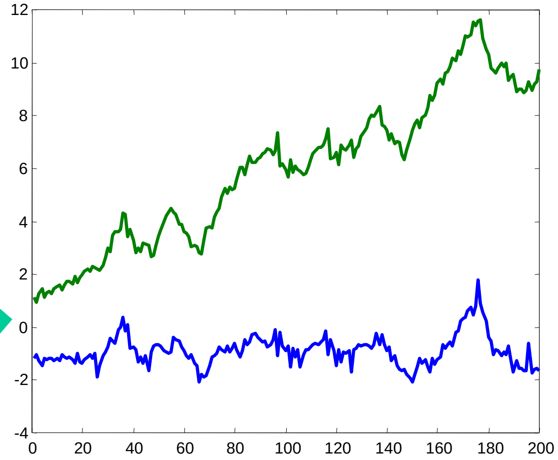
$q = (q - M(q)) / S(q)$

$o = (o - M(o)) / S(o)$

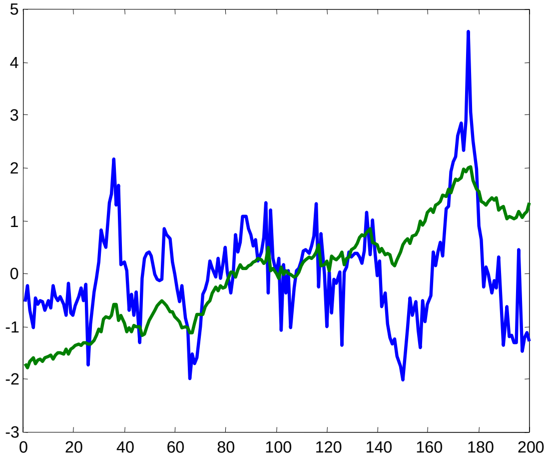


Time Series

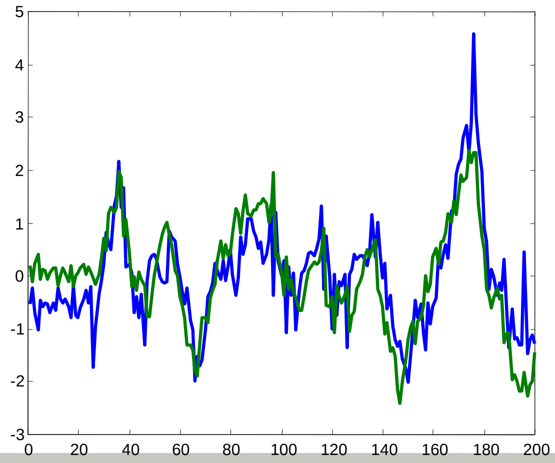
- Data Cleaning (cont.)
  - (Linear) Trend Elimination
    - Similar time series with different trends
    - Determine regression line and move each time series by its regression line
    - Gets complex when an object features more than one trend



Offset Translation + Amplitude Scaling

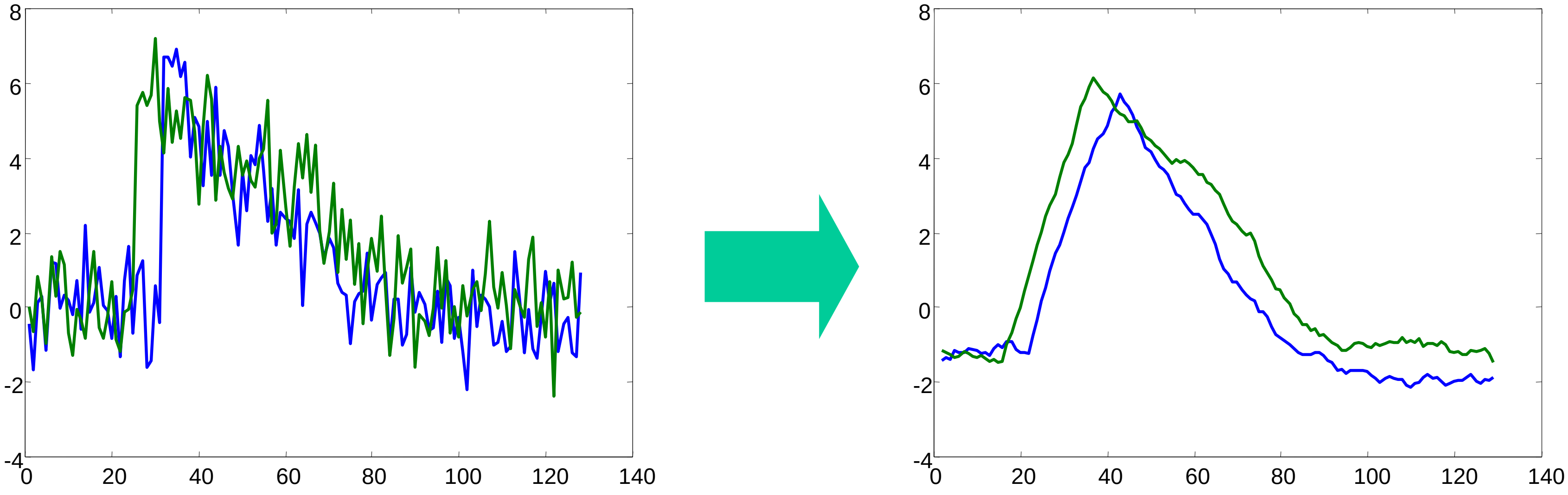


Offset Translation + Amplitude Scaling
+ Trend Elimination



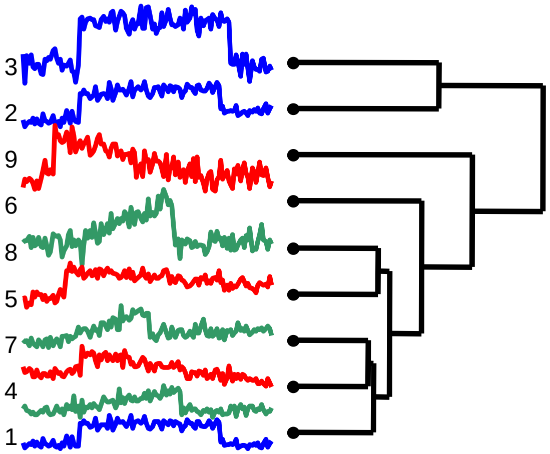
Time Series

- Data Cleaning (cont.)
  - Noise Reduction
    - Similar time series with large noise portion
    - Smoothing: normalization over a range of values (sliding window), e.g. replace *i*-th value v_i with mean value of $2k$ adjacent values $[v_{i-k}, \ldots, v_i, \ldots v_{i+k}]$

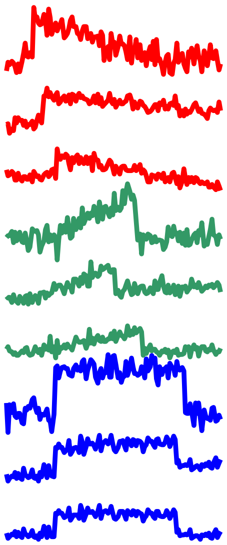


Time Series

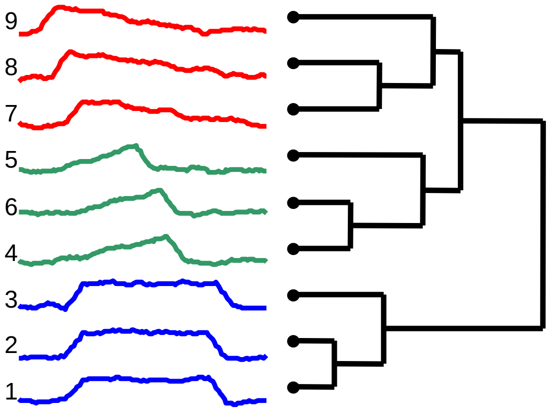
- Data Cleaning: Summary
  - The above mentioned cleaning procedures are common samples (i.e. there are many more types of distortions that might be of interest to be removed)
  - Which cleaning step should be taken? => That heavily depends on the application
  - Example:



Hierarchical (Single-Link) clustering of raw data using euclidean distance



Raw time series from three classes (see color coding)



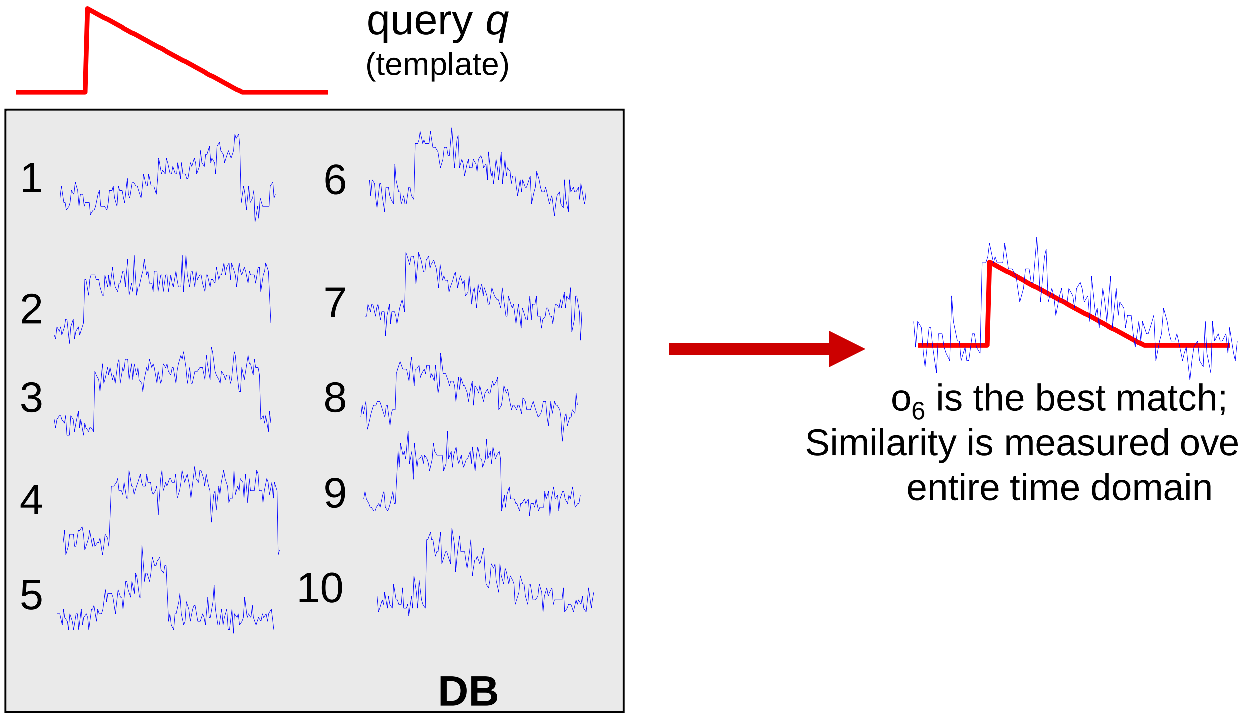
Hierarchical (Single-Link) clustering of raw data using euclidean distance after doing noise reduction, trend elimination, scaling using std. dev. and shifting using mean

Similarity Notions for Time Series

- Some example similarity queries for time series databases [AFS93]:
  - Identify companies with similar pattern of growth
  - Determine products with similar selling patterns
  - Discover stocks with similar movement in stock prices
  - Find if a musical score is similar to one of the copyrighted scores
- Different types of similarity notions:
  - **Whole matching**:
    - Time series are usually assumed to all have the same length
    - Similarity = matching entire time series
  - **Subsequence matching**:
    - Time series may have different lengths
    - Similarity = find the subsequence that has the best match

Similarity Notions for Time Series

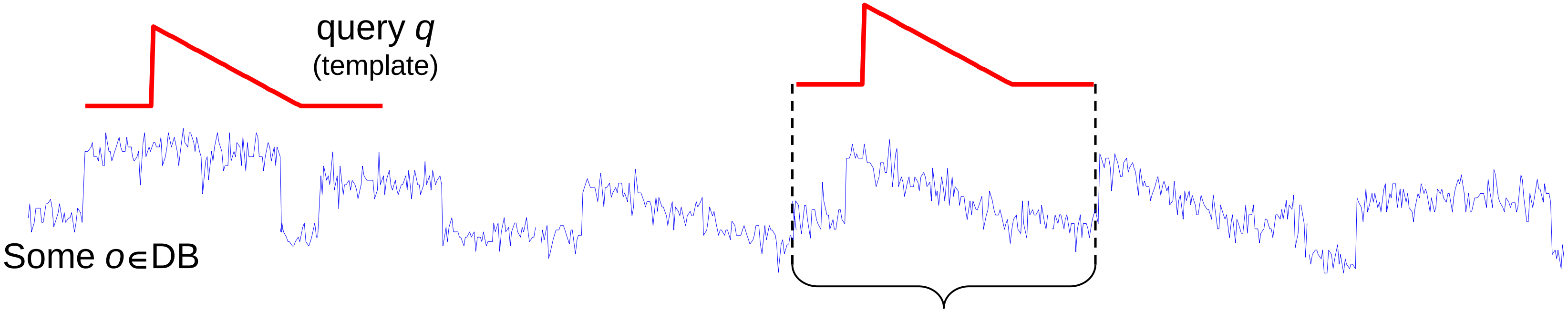
- Illustration with a query template *q*
  - **Whole matching** of *q* to a database of time series



o_6 is the best match;
Similarity is measured over the entire time domain

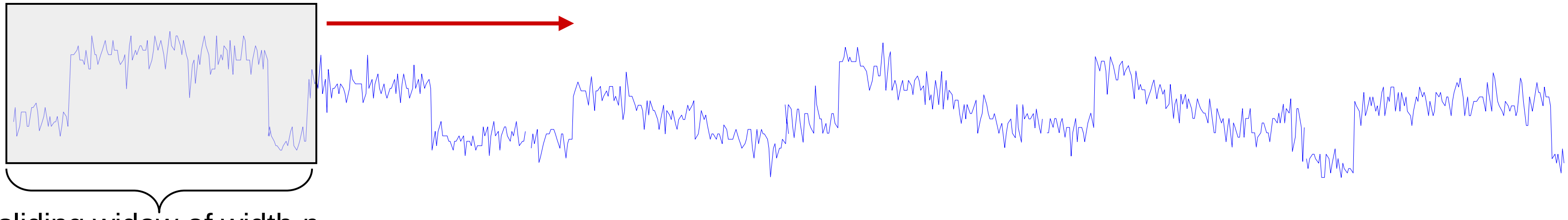
Similarity Notions for Time Series

- Illustration with a query template *q*
  - **Subsequence matching** of *q* to a database of time series



Subsequence with the best matching determines the similarity of *o* and *q*

  - Variant: the length of the (best matching) subsequence is fixed *a priori* to *n*
  - Use a sliding window of width *n* (contents of each window can e.g. be materialized)



sliding widow of width *n*

Similarity Models for Time Series

- Popular similarity measures (among others):
  - Minkowski Distances
  - Uniform Time Warping
  - Dynamic Time Warping
  - Longest Common Subsequences for Time Series
  - Edit Distance on Real Sequence
  - Edit Distance with Real Penalty
  - Shape-based Distance

Minkowski Distances for Time Series

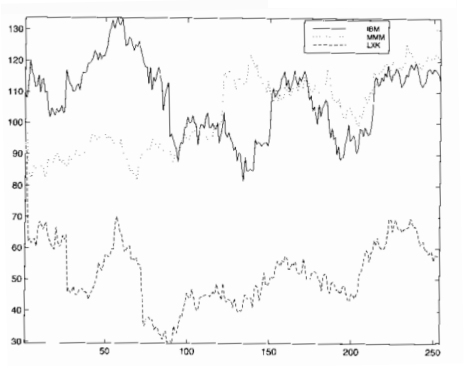
- **Idea**: Representation of a time series $X = (x_1, \dots, x_n)$ as a n-dimensional Euclidean vector
- Given two time series $X = (x_1, \dots, x_n)$ and $Y = (y_1, \dots, y_n)$ of the same length, the Minkowski Distance can be utilized as follows:

$$\mathrm{L}_p(X,Y) = \left(\sum_{i=1}^{n} |x_i - y_i|^p \right)^{\frac{1}{p}}$$

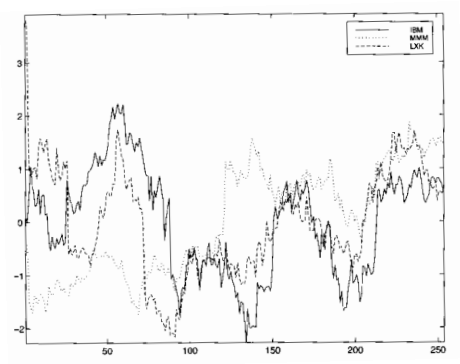
- Properties:
  - $p = 1$: dissimilarities are not emphasized
  - $p = 2$: to be preferred [AFS93]
  - $p = \infty$: distance is attributed to the most dissimilar entries of the time series
- All these variants of the Minkowski Distances are
  - sensitive w.r.t. variations on the time axis
  - are limited to time series having the same baseline, scale, and length

Normalization of Time Series of Fixed Length [SZ04] (1)

- Problems of the Euclidean Distance
  - Two time series can be very similar even though they have different baselines or amplitude scales
- Solution: **Normalization** of time series as explained above (see: preprocessing), e.g.
  - Shifting by the average value (offset translation)
  - Scaling by the standard deviation (amplitude scaling)



Stock price of IBM, LXK, MMM



Normalized stock prices

Normalization of Time Series of Fixed Length [M10] (2)

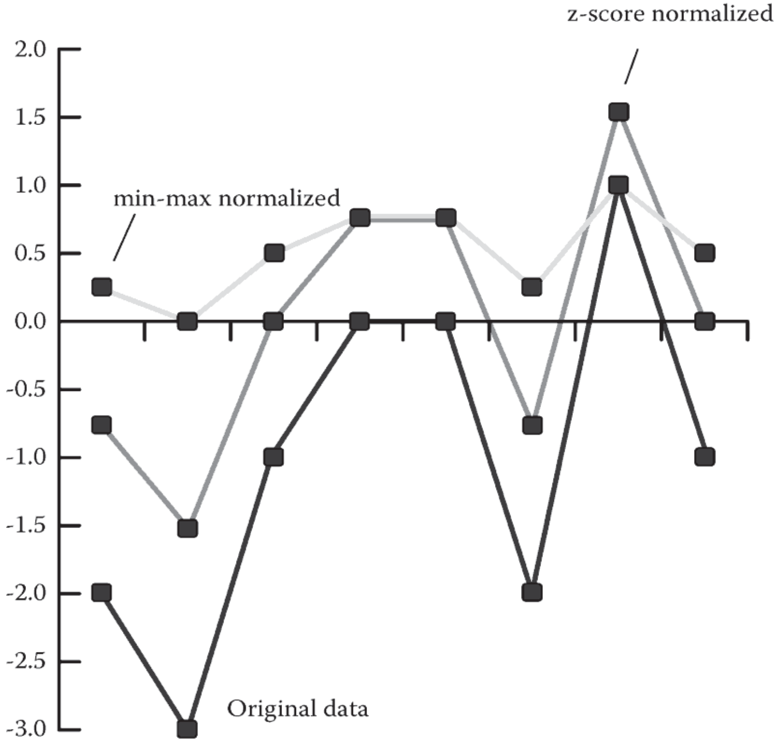
- What we have learned so far is termed **Z-Score Normalization** of a time series $X = (x_1, \dots, x_n)$:
  - shifting by the mean and scaling by the standard deviation
  - $\hat{X} = \frac{X - avg(X)}{std(X)}$

    with $avg(X) = \frac{1}{n} \cdot \sum_{i=1}^{n} x_i$

    and $std(X) = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (x_i - avg(X))^2}$
- Alternative: **Min-max normalization** of a time series $X = (x_1, \dots, x_n)$:
  - $\hat{X} = \frac{X - Max(X)}{Max(X) - Min(X)} (newMax - newMin) + newMin$
- Properties:
  - Z-Score normalization is more robust w.r.t. noise in the data
  - Min-max normalization can be dominated by outliers.

Normalization of Time Series of Fixed Length (3)

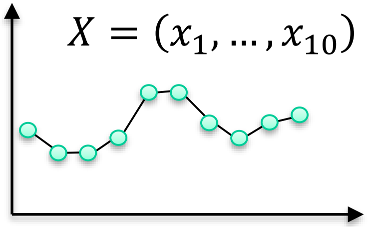
- Example of different normalizations [M10]

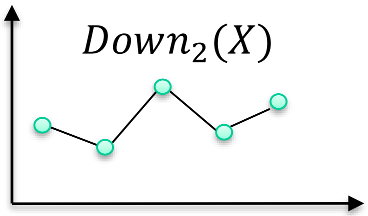


- In most cases, normalization is necessary and should be done before analysis!

Scaling Time Series along the Time Axis

- Until now: shifting and scaling is performed on the amplitude axis
- For comparing time series with different lengths, we need scaling of a time series $X = (x_1, \ldots, x_n)$ **along the time axis** as follows:
  - ***ω*-upsampling**:
    - resolution is increased
    - $Up_\omega(x_1, \ldots, x_n) = (z_1, \ldots, z_{n\omega})$ with $z_i = x_{\left\lceil \frac{i}{\omega} \right\rceil}$ and $i = 1 \ldots n\omega$
    - every x_i is repeated ω times
  - ***ω*-downsampling**:
    - resolution is decreased
    - $Down_\omega(x_1, \ldots, x_n) = \left(z_1, \ldots, z_{\left\lfloor \frac{n}{\omega} \right\rfloor}\right)$ with $z_i = x_{i\omega}$ and $i = 1 \ldots \left\lfloor \frac{n}{\omega} \right\rfloor$
    - only multiples of ω are used, i.e. $i \cdot \omega$





Uniform Time Warping (UTW)

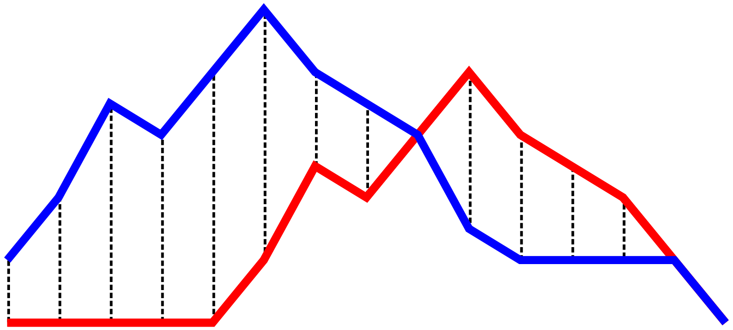
- **Idea**: Scale both time series along the time axis to the same length and utilize the Euclidean Distance
- Given two time series $X = (x_1, \ldots, x_n)$ and $Y = (y_1, \ldots, y_m)$, the **Uniform Time Warping Distance** between X and Y is defined as:

$$\begin{aligned} \mathrm{D}^2_{UTW}(X, Y) &= \frac{\mathrm{L}_2^2\big(Up_m(X), Up_n(X)\big)}{m \cdot n} \\ &= \frac{\sum_{i=1}^{m \cdot n}\big(x_{\lceil i/m \rceil} - y_{\lceil i/n \rceil}\big)^2}{m \cdot n} \end{aligned}$$

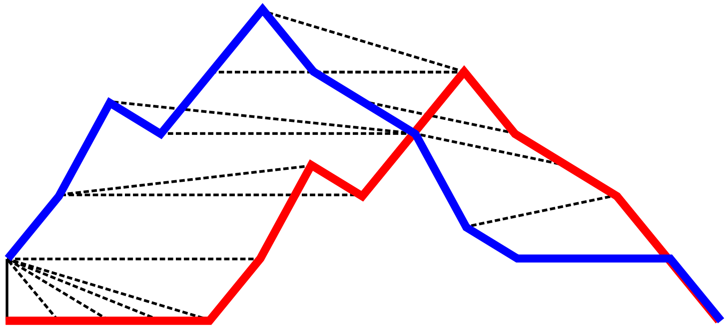
- Instead of upsampling X and Y with m and n, respectively, one could also use their lowest common multiple $\mathrm{LCM}(m, n)$

Dynamic Time Warping

- **Idea:** Allow local (=dynamic) stretching of two time series in order to minimize the distance between them
- Allows comparison of time series of different lengths
- Possible applications:
  - Comparison of hummed songs, handwritten documents, biometric data
- Comparison of the Euclidean Distance, which epitomizes a point-to-point distance, and Dynamic Time Warping



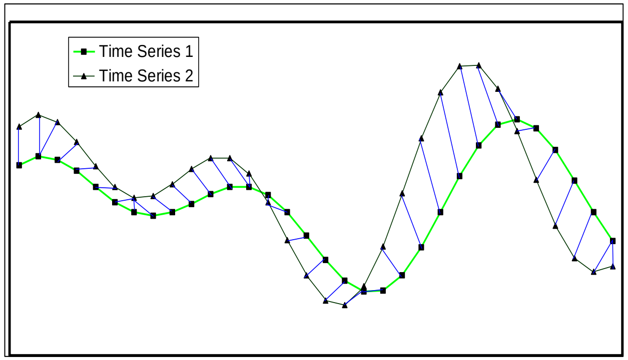
Euclidean Distance



Dynamic Time Warping

Dynamic Time Warping: Formal Definition

- Given a time series $X = (x_1, \dots, x_n)$, let
  - $Start(X) = (x_1, \dots, x_{n-1})$ define the prefix of X
  - $Last(X) = x_n$ define the last element
  - $\emptyset = ()$ define an empty time series



- Given two time series $X = (x_1, \dots, x_n)$ and $Y = (y_1, \dots, y_m)$ and a ground distance δ, the **Dynamic Time Warping Distance** between X and Y is recursively defined as:

$$\begin{aligned}
\mathrm{DTW}_{\delta,p}(\emptyset,\emptyset) &= 0\\
\mathrm{DTW}_{\delta,p}(X,\emptyset) &= \mathrm{DTW}_{\delta,p}(\emptyset,Y) = \infty \text{ for } X,Y \neq \emptyset\\
\mathrm{DTW}_{\delta,p}(X,Y) &= \left(\delta\big(Last(X),Last(Y)\big)^p + \left(\min\left\{\begin{matrix}\mathrm{DTW}_{\delta,p}\big(Start(X),Start(Y)\big)\\ \mathrm{DTW}_{\delta,p}\big(X,Start(Y)\big)\\ \mathrm{DTW}_{\delta,p}\big(Start(X),Y\big)\end{matrix}\right\}\right)^p\right)^{\frac{1}{p}}
\end{aligned}$$

Dynamic Time Warping: Variation of Parameter *p*

- Variation of parameter $p \in \mathbb{R}^+$ yields the following instances
  - $p = 1$:

$$\mathrm{DTW}_{\delta,1}(X,Y) = \delta\big(Last(X), Last(Y)\big) + \min\begin{Bmatrix} \mathrm{DTW}_{\delta,1}\big(Start(X), Start(Y)\big) \\ \mathrm{DTW}_{\delta,1}\big(X, Start(Y)\big) \\ \mathrm{DTW}_{\delta,1}\big(Start(X), Y\big) \end{Bmatrix}$$

  - $p = 2$ (Euclidean variant):

$$\mathrm{DTW}_{\delta,2}(X,Y) = \sqrt{\delta\big(Last(X), Last(Y)\big)^2 + \left(\min\begin{Bmatrix} \mathrm{DTW}_{\delta,2}\big(Start(X), Start(Y)\big) \\ \mathrm{DTW}_{\delta,2}\big(X, Start(Y)\big) \\ \mathrm{DTW}_{\delta,2}\big(Start(X), Y\big) \end{Bmatrix}\right)^2}$$

  - $p \to \infty$:

$$\mathrm{DTW}_{\delta,\infty}(X,Y) = \max\left\{\delta\big(Last(X), Last(Y)\big), \min\begin{Bmatrix} \mathrm{DTW}_{\delta,\infty}\big(Start(X), Start(Y)\big) \\ \mathrm{DTW}_{\delta,\infty}\big(X, Start(Y)\big) \\ \mathrm{DTW}_{\delta,\infty}\big(Start(X), Y\big) \end{Bmatrix}\right\}$$

- Termination cases are the same as on the previous slide

Dynamic Time Warping: Ground Distances and Notations

- Time series are typically real-valued, thus may often choose the ground distance δ as the absolute difference:

$$\delta(x_i, y_i) = |x_i - y_i| = \mathrm{L}_1(x_i, y_i)$$

- One of the most prominent variant of Dynamic Time Warping Distance is the squared Euclidean variant with Manhattan ground distance:

$$\begin{aligned} \mathrm{DTW}^2(X,Y) &= \mathrm{DTW}^2_{\mathrm{L}_1,2}(X,Y) \\ &= |Last(X) - Last(Y)|^2 + \min \begin{Bmatrix} \mathrm{DTW}^2\big(Start(X), Start(Y)\big) \\ \mathrm{DTW}^2\big(X, Start(Y)\big) \\ \mathrm{DTW}^2(Start(X), Y) \end{Bmatrix} \end{aligned}$$

Dynamic Time Warping: Warping Path

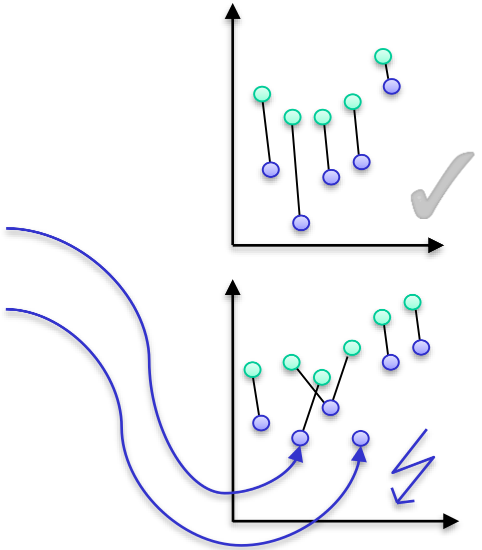
- Dynamic Time Warping aligns two time series to each other
- This element-wise alignment between two time series $X = (x_1, \dots, x_n)$ and $Y = (y_1, \dots, y_m)$ can be expressed by a **warping path** P of indices:

$$P = p_1, \dots, p_L = (p_1^X, p_1^Y), \dots, (p_L^X, p_L^Y)$$

  where $p_i^X \in [1, n]$ and $p_i^Y \in [1, m]$ denote the indices within the times series X and Y

- Properties of a warping path P:
  a) Boundary condition: $p_1 = (1{,}1)$ and $p_L = (n, m)$
  b) Monotonicity: $p_t^X - p_{t-1}^X \geq 0$ and $p_t^Y - p_{t-1}^Y \geq 0$
  c) Continuousness: $p_t^X - p_{t-1}^X \leq 1$ and $p_t^Y - p_{t-1}^Y \leq 1$
  d) The length $|P|$ is bounded by:

$$\max(n, m) \leq |P| \leq n + m - 1$$



Dynamic Time Warping: Warping Path cont'd

- Let $\mathcal{P}$ denote the set of all paths satisfying constraints a) to d)
- The size of $\mathcal{P}$ is exponential
- Let the **cost of a path** $P = p_1, \dots, p_L = (p_1^X, p_1^Y), \dots, (p_L^X, p_L^Y)$ between two time series $X = (x_1, \dots, x_n)$ and $Y = (y_1, \dots, y_m)$ be defined as:

$$cost(P, X, Y) = \sum_{i=1}^{L} \left| x_{p_i^X} - y_{p_i^Y} \right|^2$$

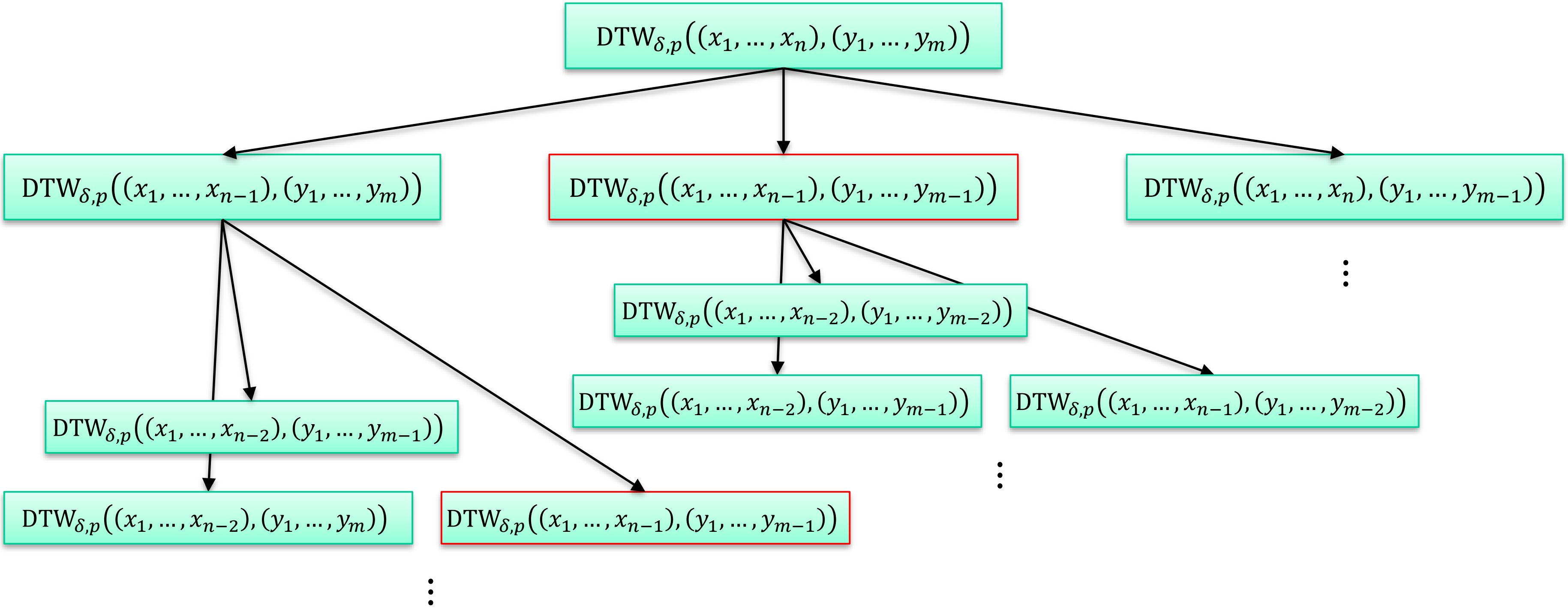
- $\mathrm{DTW}^2(X, Y)$ can be defined by the path with the minimal cost:

$$\mathrm{DTW}^2(X, Y) = \min_{P \in \mathcal{P}} cost(P, X, Y)$$

- For time series with the same length n, the warping path $P = (1,1), \dots, (n, n)$ yields the Euclidean Distance

Naïve Computation of Dynamic Time Warping

- Recursive computation of $\mathrm{DTW}_{\delta,p}$ between two time series $X = (x_1, \dots, x_n)$ and $Y = (y_1, \dots, y_m)$:



$\mathrm{DTW}_{\delta,p}\big((x_1, \dots, x_n), (y_1, \dots, y_m)\big)$

$\mathrm{DTW}_{\delta,p}\big((x_1, \dots, x_{n-1}), (y_1, \dots, y_m)\big)$

$\mathrm{DTW}_{\delta,p}\big((x_1, \dots, x_{n-1}), (y_1, \dots, y_{m-1})\big)$

$\mathrm{DTW}_{\delta,p}\big((x_1, \dots, x_n), (y_1, \dots, y_{m-1})\big)$

⋮

$\mathrm{DTW}_{\delta,p}\big((x_1, \dots, x_{n-2}), (y_1, \dots, y_{m-2})\big)$

$\mathrm{DTW}_{\delta,p}\big((x_1, \dots, x_{n-2}), (y_1, \dots, y_{m-1})\big)$

$\mathrm{DTW}_{\delta,p}\big((x_1, \dots, x_{n-1}), (y_1, \dots, y_{m-2})\big)$

$\mathrm{DTW}_{\delta,p}\big((x_1, \dots, x_{n-2}), (y_1, \dots, y_{m-1})\big)$

⋮

$\mathrm{DTW}_{\delta,p}\big((x_1, \dots, x_{n-2}), (y_1, \dots, y_m)\big)$

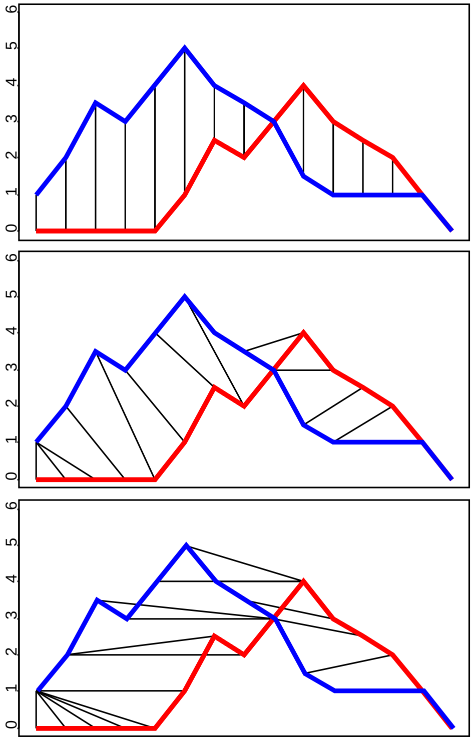
$\mathrm{DTW}_{\delta,p}\big((x_1, \dots, x_{n-1}), (y_1, \dots, y_{m-1})\big)$

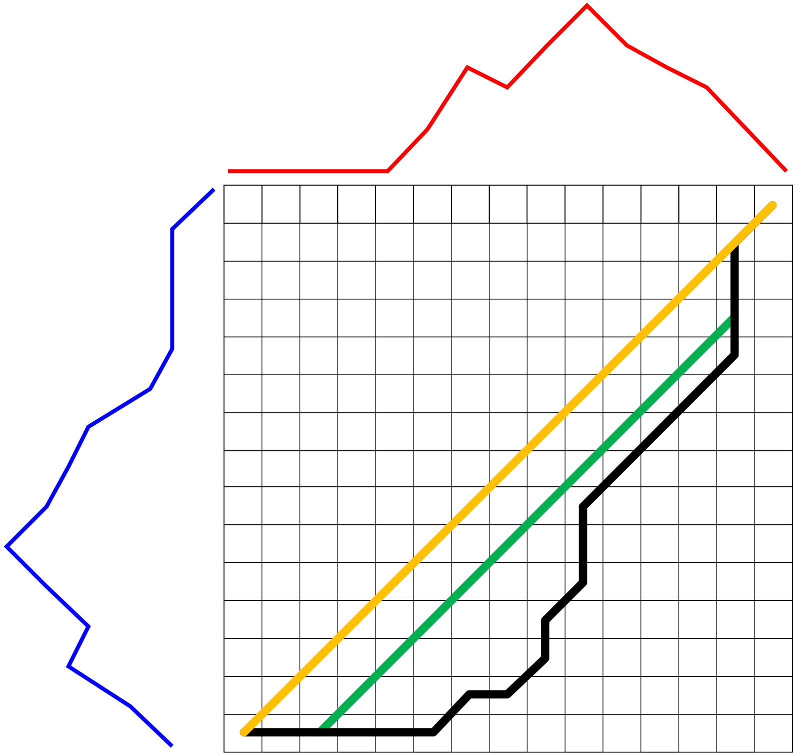
⋮

- Computation time complexity lies in $\mathcal{O}\left(3^{tree\ height}\right) = \mathcal{O}(3^{n+m})$

Dynamic Time Warping: Path in a Matrix

- Any path P between two times series $X = (x_1, \ldots, x_n)$ and $Y = (y_1, \ldots, y_m)$ can be expressed as a path in a $n \times m$ matrix:





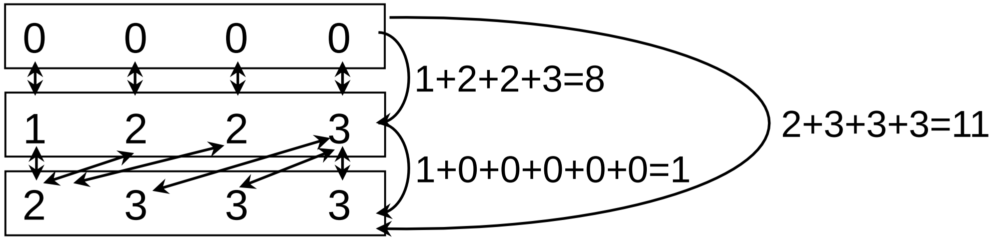
- This matrix is utilized for computing the DTW by Dynamic Programming

Dynamic Time Warping is not a Metric

- DTW does not satisfy the identity of indiscernibles:
  - Time series X: 1 2 2 2
  - Time series Y: 1 1 1 2

$$\Rightarrow DTW(X,Y) = 0$$

- DTW does not satisfy the triangle inequality:
  - Time series X:
  - Time series Y:
  - Time series Z:

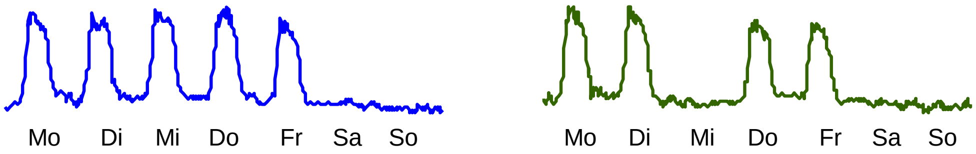


$$\begin{array}{ccccc} DTW(X,Z) & \leq & DTW(X,Y) & + & DTW(Y,Z) \\ 11 & \leq & 8 & + & 1 \end{array} \quad ↯$$

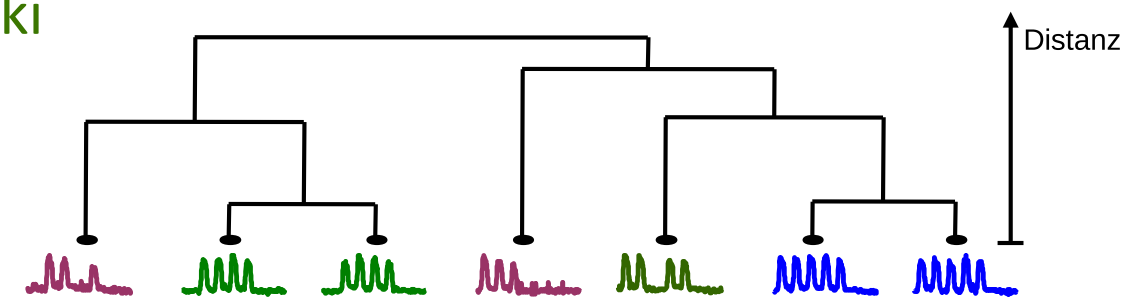
- Reason: replication of elements

Minkowski vs DTW

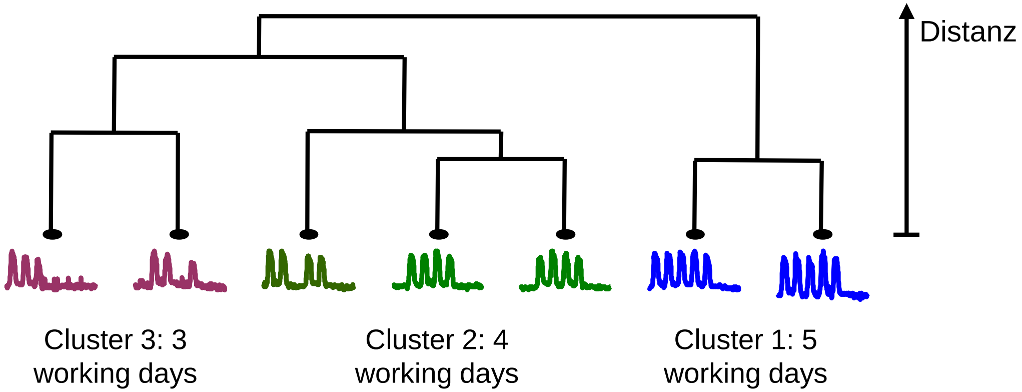
- Comparison on time series that measure the daily network traffic of a company, e.g.:



- Minkowski



- DTW



DTW for Trajectories / Multivariate Time Series [VHG+03]

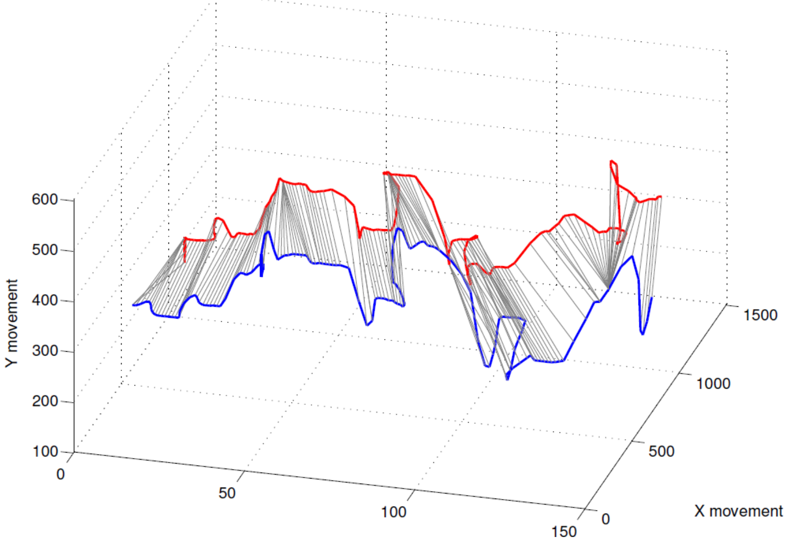
- Up to now: Time series over real numbers
  - Ground distance δ between two elements x_i, y_i of time series X, Y can be chosen as absolute difference:

$$\delta(x_i, y_i) = |x_i - y_i|$$

- Application of DTW to trajectories
  - Trajectories are time series over multidimensional objects, e.g.:

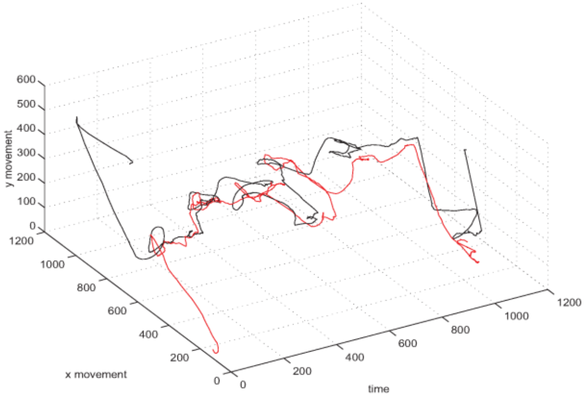
$$X = \big((x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\big)$$

  - Necessary: measurement of temporally ordered points in space
  - Different ground distances $(\mathrm{L}_1, \mathrm{L}_2, \mathrm{L}_\infty)$ for comparison of (x_i, y_i) and $\left(x_j, y_j\right)$
  - ➢ Adaptation of DTW to multidimensional time series is straightforward



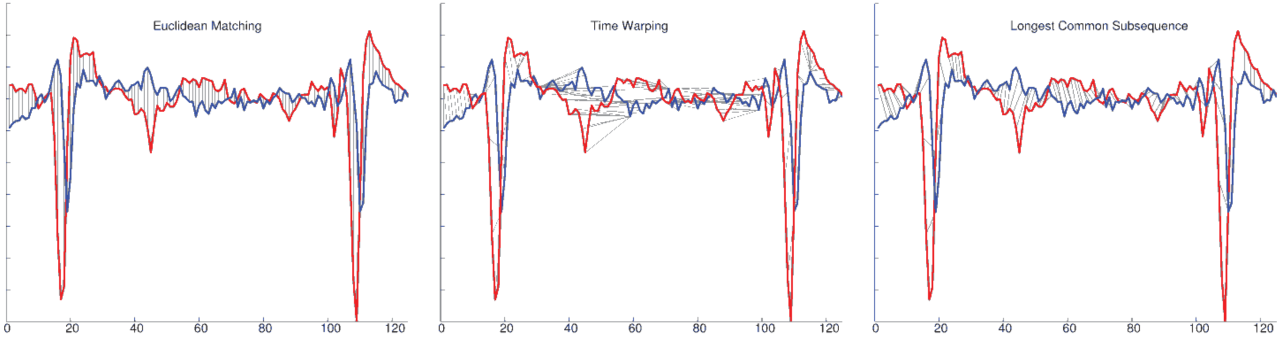
Longest Common Subsequence for Time Series [VKG02, VHG+03]

- Dynamic Time Warping is sensitive to outliers and noise
- Solution: extending LCSS to time series
- A measure tolerant to gaps in the two compared time series



Example 1:

- Two 2D trajectories that contain many outliers at start and end



Example 2:

- Noisy setting where DTW gives many dubious matchings

Feature Extraction in Time Series

- Instead of directly working with the entire time series, we can also extract features from them
- Many feature extraction techniques exist that basically follow two different purposes:
  - Many of them aim at representing time series in a compact way (e.g. as a "shorter" approximation of the original time series) with minimum loss of modelling error

    => this is mostly done for performance considerations

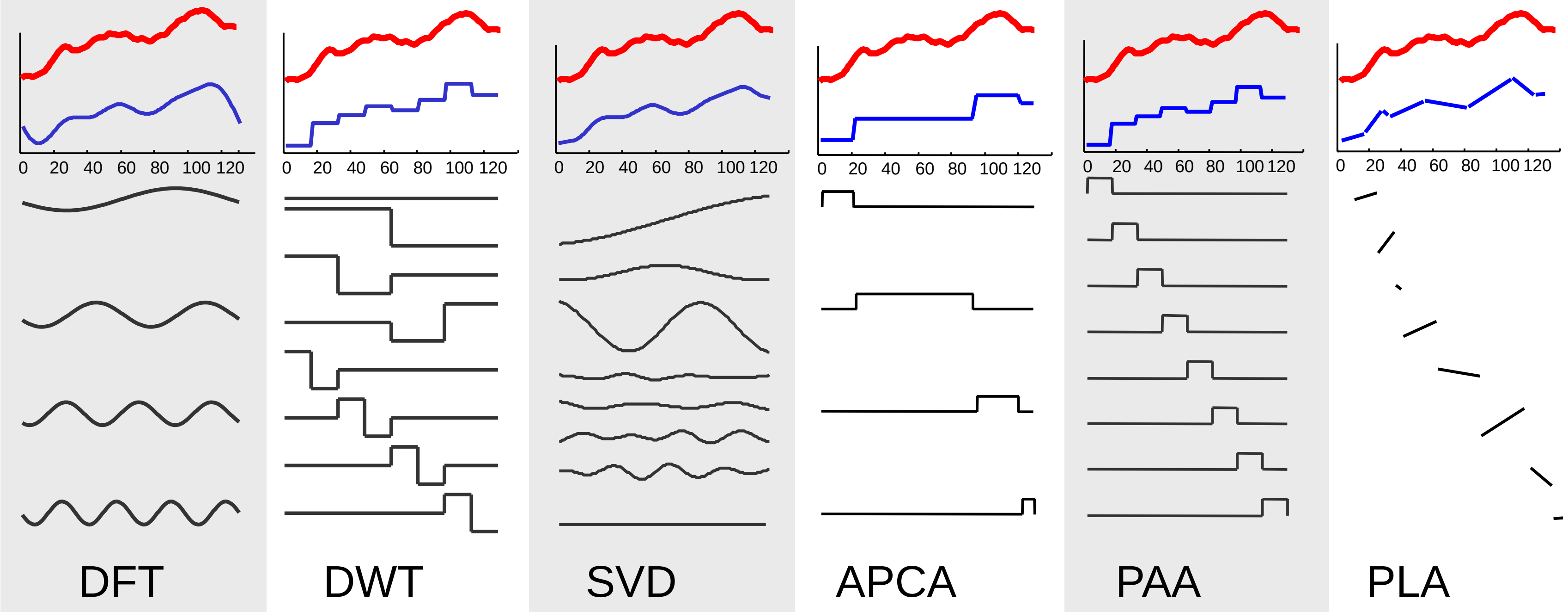
    => approach is closely related to dimensionality reduction/feature selection

    Examples covered here: DFT, DTW, SVD, APCA, PAA, PLA

  - Other model specific properties of the time series relevant to a given application

    Example covered here: threshold-based modelling

Compact Representations: Overview



Agrawal, Faloutsos, Swami. FODO 1993
Faloutsos, Ranganathan, Manolopoulos. SIGMOD 1994

Chan, Fu. ICDE 1999

Korn, Jagadish, Faloutsos. SIGMOD 1997

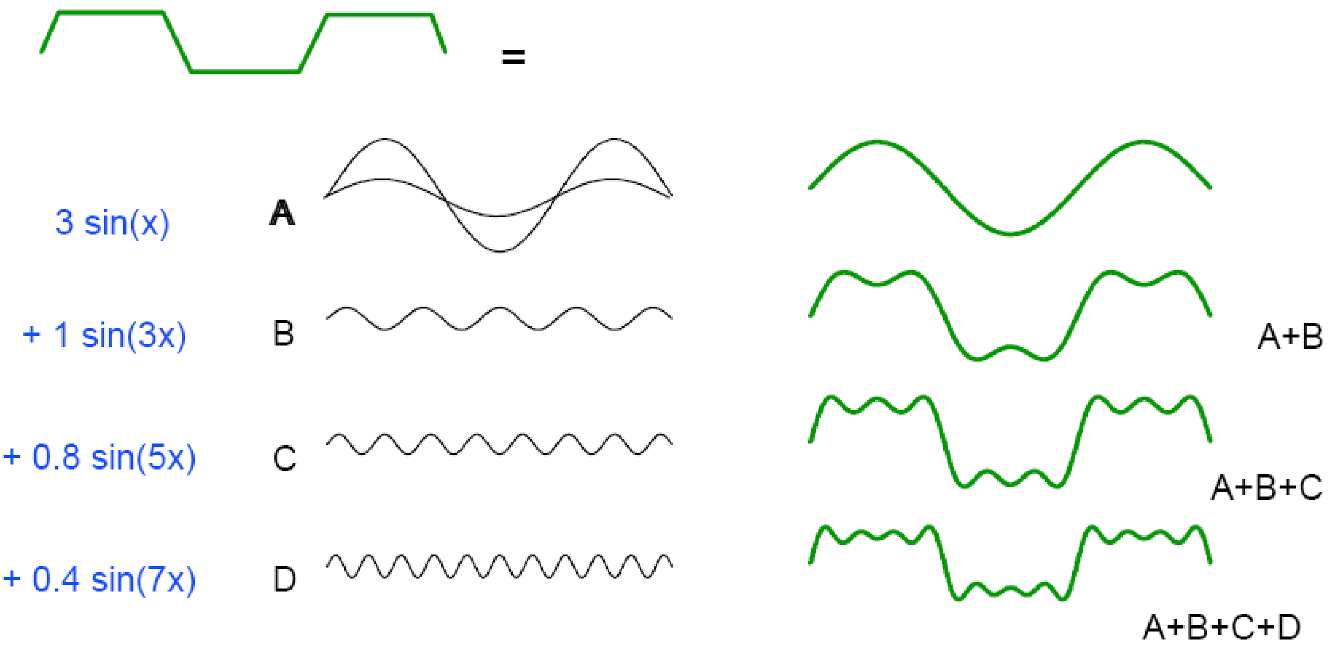
Keogh, Chakrabarti, Pazzani, Mehrotra SIGMOD 2001

Keogh, Chakrabarti, Pazzani, Mehrotra KAIS 2000
Yi, Faloutsos VLDB 2000

Morinaka, Yoshikawa, Amagasa, Uemura, PAKDD 2001

Compact Representations: DFT

- Discrete Fourier Transformation (DFT)
  - Idea:
    - Describe a pereodical function as a weighted sum of periodical base functions with varying frequency
    - Base functions here: sin und cos
  - Example:



Compact Representations: DFT

- Basic foundation: Fouriers Theorem

  Any periodic function can be represented by a sum of sin- and cos-functions of different frequency

  - DFT does not „change“ the function but simply finds a different equivalent representation (and DFT can be reversed)
  - Formally:
    - Let $x = [x_t]$, $t = 0, \ldots, n - 1$ be a time series of length n
    - DFT transforms x into $X = [X_f]$ of n complex numbers with frequencies $f = 0, \ldots, n - 1$ such that

$$X_f = \frac{1}{\sqrt{n}} \sum_{t=0}^{n-1} x_t \cdot e^{\frac{-j2\pi ft}{n}} =$$

$$\underbrace{\frac{1}{\sqrt{n}} \sum_{t=0}^{n-1} x_t \cos\left(\frac{2\pi ft}{n}\right)}_{\text{Realteil}} - \underbrace{j \cdot \frac{1}{\sqrt{n}} \sum_{t=0}^{n-1} x_t \sin\left(\frac{2\pi ft}{n}\right)}_{\text{Imaginärteil}}$$

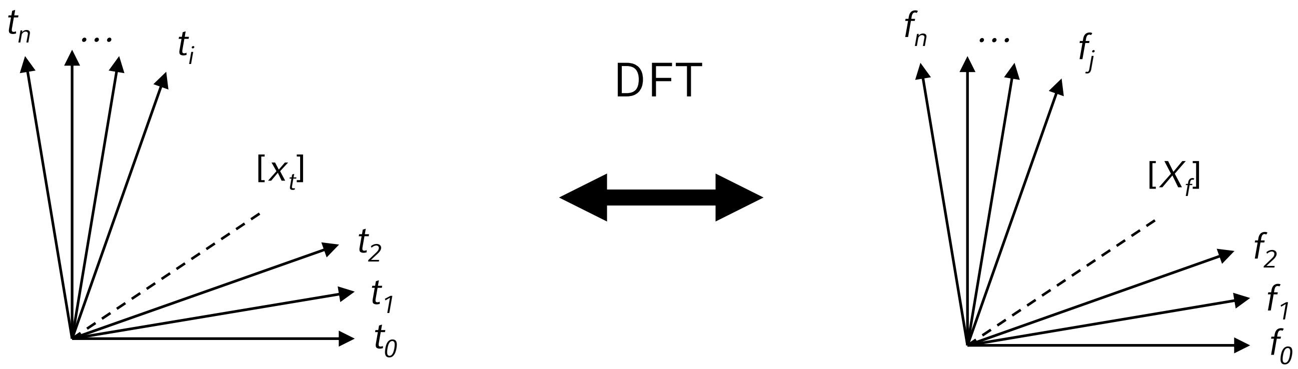
where $j^2 = -1$.

» Realteil is the portion of cosine in frequency f

» Imaginärteil is the portion of sinus in frequency f

Compact Representations: DFT

- DFT can be interpreted as a transformation of the bais vectors (like e.g. PCA):



  - The new axis represent the frequencies

- But how does that help?
  - So far, we transformed an *n*-dimensional time series into an *n*-dimensional vector …

- Well first of all, it holds that the euclidean distance is preserved after DFT, i.e. $|| x - y ||^2 = || X - Y ||^2$

Compact Representations: DFT

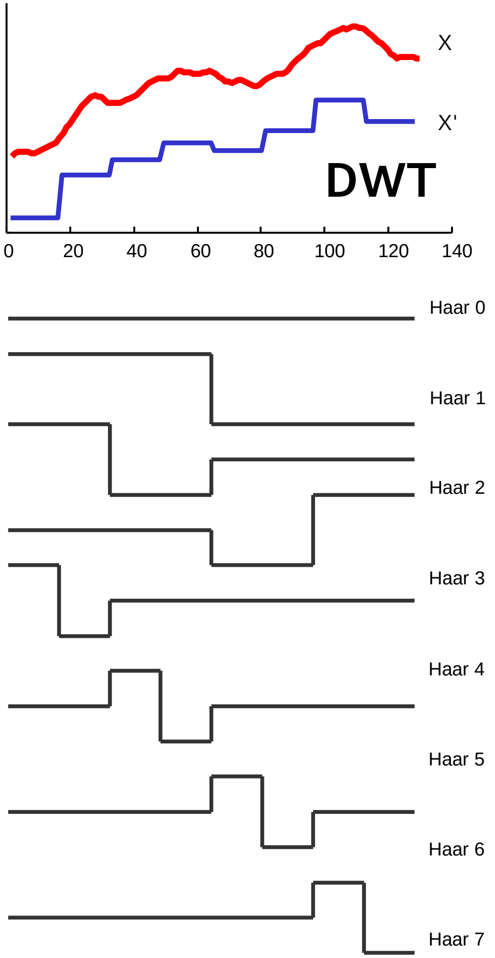
- This follows from Parseval‘s theorem (and the linearity of DFT) which states that the energy of a sequence (= sum of squared amplitudes $E(x) = \|x\|^2 = \sum_{t=0}^{n-1} |x_t|^2$) is preserved, i.e.:

$$\sum_{t=0}^{n-1} |x_t|^2 = \sum_{t=0}^{n-1} |X_f|^2$$

- Now comes the important trick: in practice, the low frequencies (first components) have the highest impact, i.e. contain the most information
- Focusing on the first *c* coeficients is a good choice if we want to reduce the „dimensionality“ of a sequence
- Since $|| x - y ||^2 = || X - Y ||^2$ holds, using only *c* components instead of *n* yields a lower bounding approximation of the Euclidean Distance
- This approximation will be better when using DFT componenents instead of original time stamps

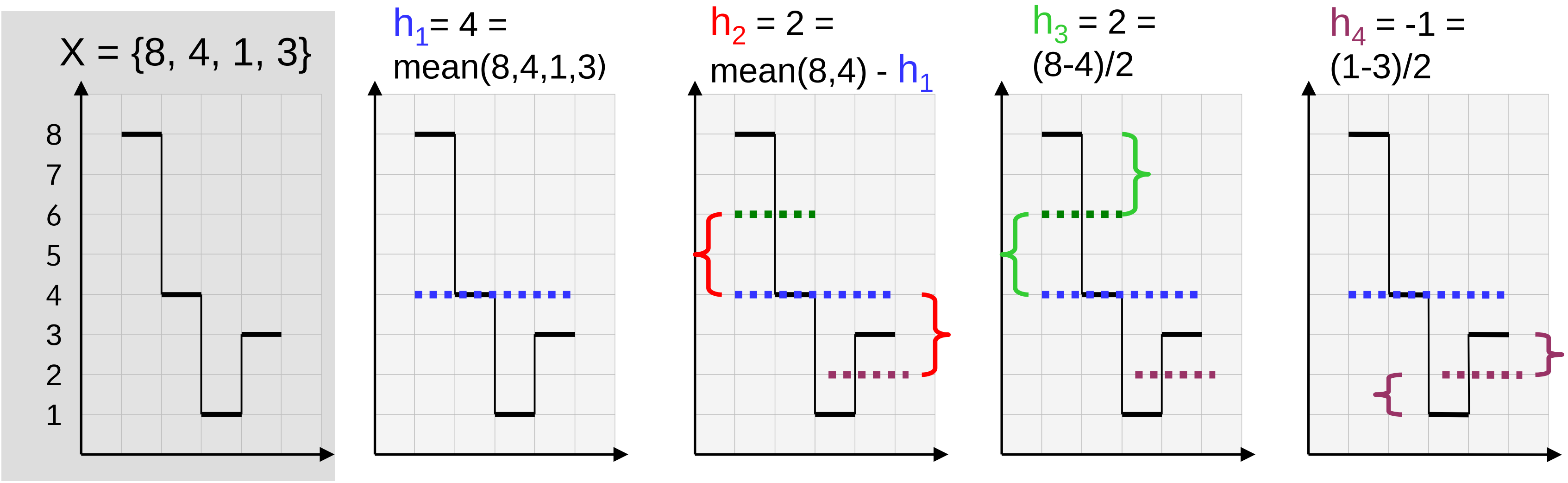
Compact Representations: DWT

- Discrete Wavelet Transformation (DWT)
  - Idea:
    - Represent a time series as a linear combination of base functions (Wavelet-functions)
    - Typically, Haar-Wavelets are used
  - Properties:
    - The more stationary the time series is, the better is the approximation with fewer components
    - Distance on DWT components also lower bounds Euclidean and DTW distance on original tijme series
    - Time series are restricted to be of length 2^i (for any i)



Compact Representations: DWT

- Example:
  - Stepwise transformation of time series x = {8,4,1,3} into Haar Wavelet representation H = [4,2,2,-1]



X = {8, 4, 1, 3}

h_1= 4 = mean(8,4,1,3)

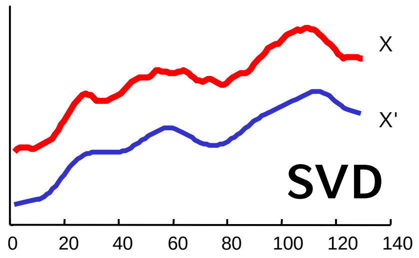
h_2 = 2 = mean(8,4) - h_1

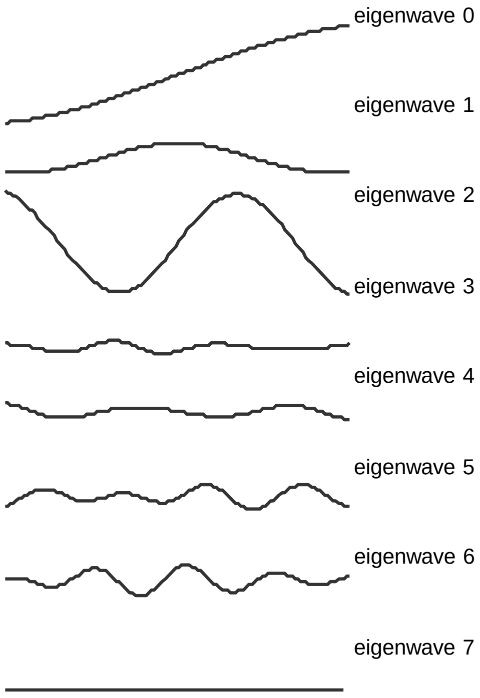
h_3 = 2 = (8-4)/2

h_4 = -1 = (1-3)/2

Compact Representations: SVD

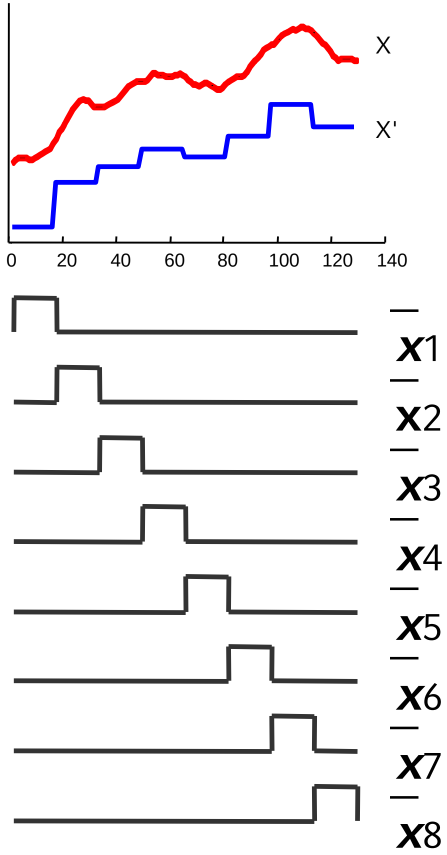
- Singular Value Decomposition (SVD)
  - Idea:
    - Instead of sinus/cosine use Eigen Waves
  - Properties:
    - Minimizes the quadratic approximation error (like PCA and SVD on high dimensional data)
    - The semantics of the components of SVD depends on the actual data while DFT (sin/cos) and DWT (const) are not data dependent
    - In text mining and Information Retrieval, SVD as a feature extraction technique is also know as „Latent Semantic Indexing“





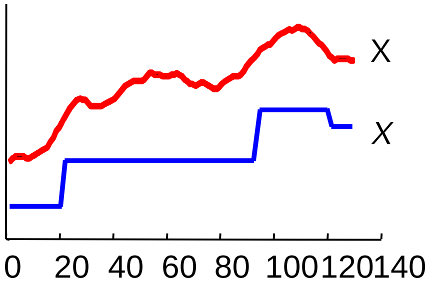
Compact Representations: PAA

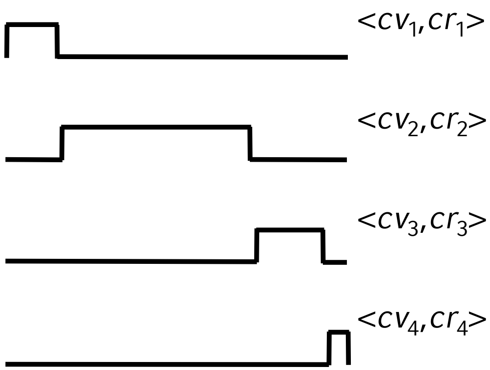
- Piecewise Aggregate Approximation (PAA)
  - Idea:
    - Transform time series into a sequence of box-functions
    - Each box has the same length and approximates the interval by the mean.
  - Properties
    - Lower bounding property
    - Time series may have arbitrary length



Compact Representations: APCA

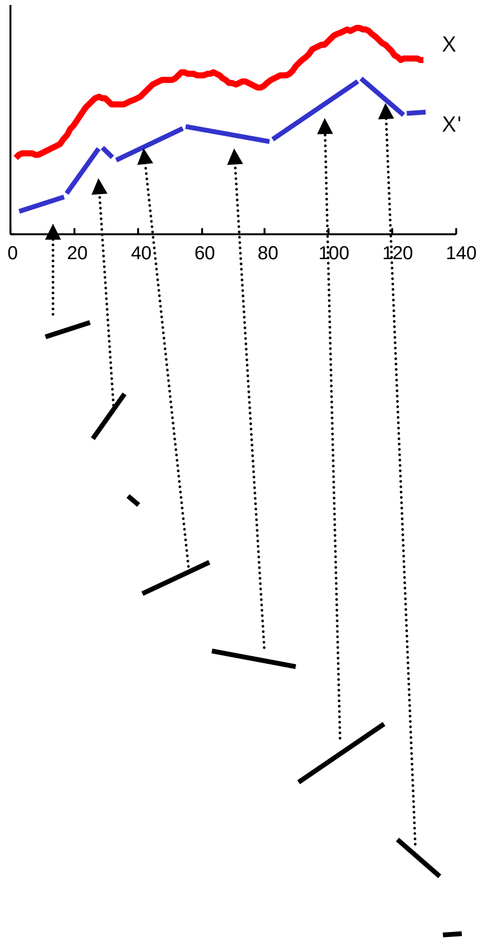
- Extension: Adaptive Piecewise Constant Approximation (APCA)
  - Motivation
    - Time series may have time intervals with a small amount details (small amplitude) and intervals with a large amount of details (large amplitude)
    - PAA cannot account for varying amounts of detail
  - Idea
    - Use boxes of variable length
    - Each segment now requires 2 paramters





Compact Representations: PLA

- Piecewise Linear Approximation (PLA)
  - Idea
    - Transform time series into a sequence of line segments $s = (\text{length}, \text{height}_{start}, \text{height}_{end})$
    - Two consectutive segments need not to be connected
  - Proterties
    - Good approximation dependents on #segments
    - Each component (segment) is a rich approximation but requires more parameters
    - Lower bounds Euclidean and DTW

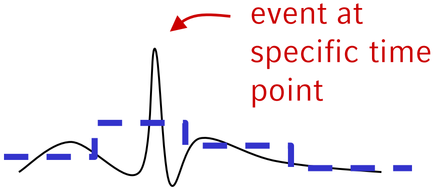


Threshold-based similarity

- An example of a specific feature transformation to model a special notion of similarity of time series is „threshold-based similarity"

[Assfalg, Kriegel, Kröger, Kunath, Pryakhin, Renz. Proc. 10th Int. Conf. on Extending Database Technology (EDBT), 2006]

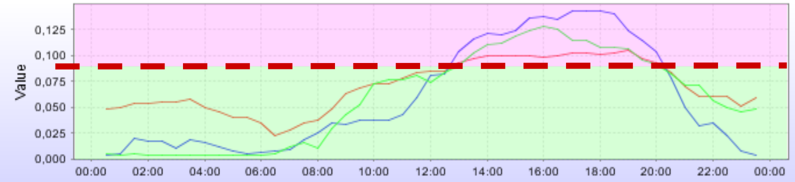
- Basic Idea:
  - In some applications, only significant „events" that are defined by certain amplitudes (or amplitude values) are interesting
  - So far, the feature extraction extracts features modeling certain properties of time intervals but not of amplitude intervals



Aggreagtion over time

Threshold-based similarity

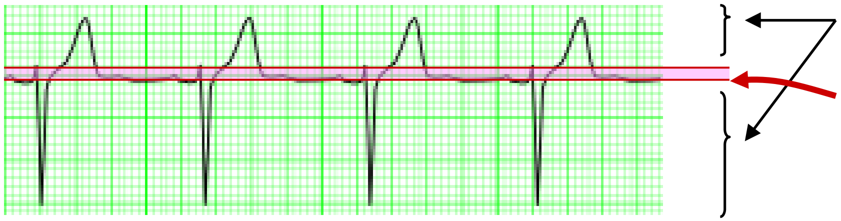
- Sample Applications
  - Environmental Science: analyzing critical ozone concentrations?
    - Find cluster of regions (time series) that exceed the allowed threshold in similar time intervals



Dangerous Ozone Concentration

Normal

  - Medical diagnosis: potential for cardiac infarction?
    - Find clusters of heart rates by focusing on the relevant amplitude intervals



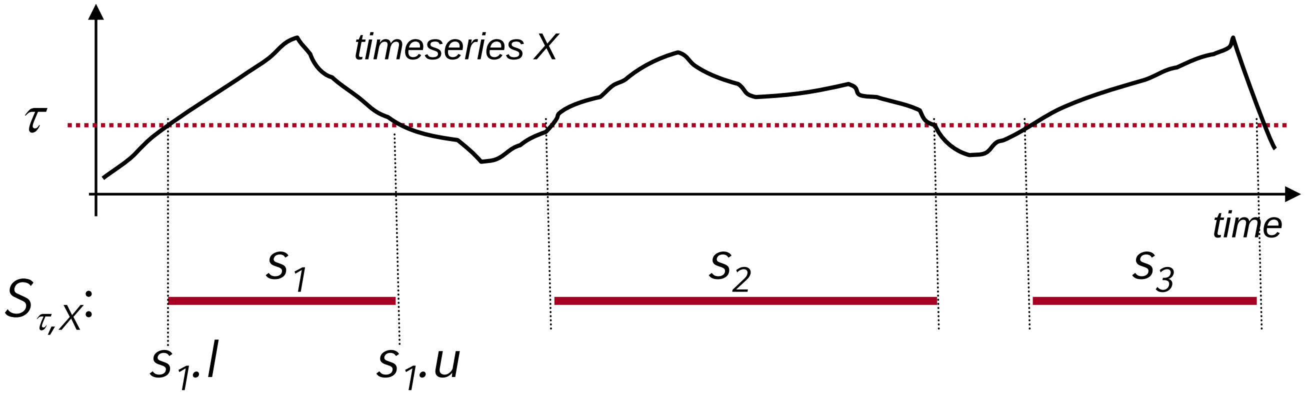
Not relevant for cardic infarction

Relevant for the detection of risk factors for cardic infarction

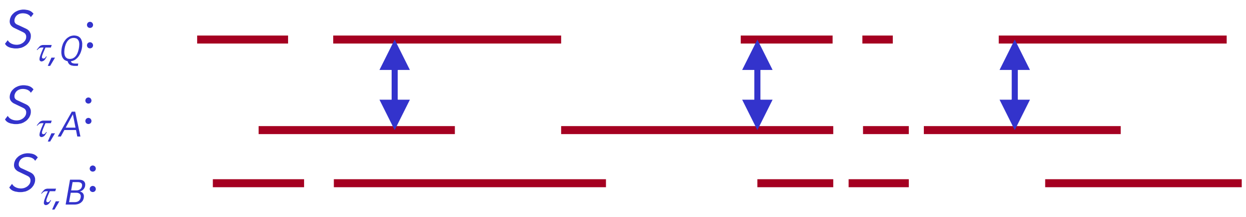
Threshold-based similarity

– Similarity Model

- Time series $X = \langle (x_i,t_i): i = 1..N \rangle$ is transformed into a sequence of intervals $S_{\tau,X} = \{s_j: j = 1..M\}$, such that: $\forall t \in T : (\exists s_j \in S_{\tau,X} : s_j.l < t < s_j.u) \Leftrightarrow x(t) > \tau$.



- Similarity of time series = similarity of sequences of intervals

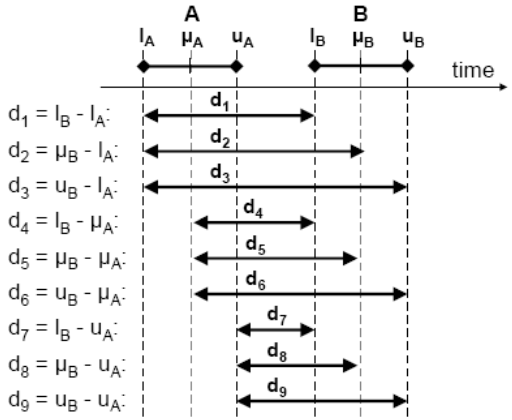


Threshold-based similarity

- Similarity between sequences of intervals?
- First: distance on intervals?
  - Euclidean distance on l- and u-values:

$$d_{\text{int}}(s_1,s_2)=\sqrt{(l_1-l_2)^2+(u_1-u_2)^2}$$

where $s_1=(l_1,u_1)$ und $s_2=(l_2,u_2)$



- Use sum of minimum distance between two sequences of intervals S_X und S_Y

$$d_{TS}(S_X,S_Y)=\frac{1}{2}\cdot\left(\underbrace{\frac{1}{|S_X|}\cdot\sum_{s\in S_X}\min_{t\in S_Y} d_{\text{int}}(s,t)}_{S_X \dashrightarrow S_Y}+\underbrace{\frac{1}{|S_Y|}\cdot\sum_{t\in S_Y}\min_{s\in S_X} d_{\text{int}}(t,s)}_{S_X \dashleftarrow S_Y}\right)$$

Feature Extraction from Time Series

- Round-up:
  - Feature extraction method serve the purpose of
    - Finding a compact representation of the original time series (mostly for performance reasons)
      - Compact representations can be used for approximate similarity computations
      - Some have bounding properties (e.g. lower bounding the exact distance/similarity) that can be used for indexing/pruning

    or

    - Modeling a specialized notion of similarity of a time series for a given application

Spatial-Temporal Data

- Geometric based method
- Semantic based method