Knowledge Discovery in Databases II

Lecture 4 – Sequential Data

Prof. Dr. Peer Kröger, Yifeng Lu

Sommer Semester 2019

Credits:

Based on material of Eirini Ntoutsi, Matthias Schubert,

Arthur Zimek, Peer Kröger, Yifeng Lu

Table of Contents

Kapitel 1: Intorduction i

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, in streams, 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 buy 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)

Kapitel 2: Sequences i

Sequences

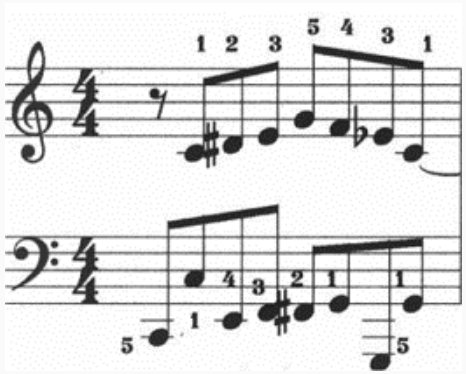
- A sequence S of length n is a mapping of the index set $I_n = \{1, 2, \ldots, n\}$ into a domain O:

$$S : I_n \rightarrow O$$

- 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 $O^* = \{I_n \rightarrow O \,|\, n \in \mathbb{N}_0\}$
- Sequences can be classified by their domain O
  - Categorical values (nominal values, alphabets, enumeration types)
  - Continuous values (real numbers)

Sequence Data

- Examples
  - Text data $\{a, ..., Z, 0, ..., 9, ...\}^*$
  - Video data $images^*$
  - Music data $notes^*$
  - Protein sequences $AminoAcids^*$
  - Gene sequences $\{C, G, A, T\}^*$
  - ...



- Time series are of course special types of sequences

Similarity Models for Sequences

- Very often, distance-based models are employed for sequence data
- 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
    - Naive 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 $Q = (q_1, ..., q_n)$ and $S = (s_1, ..., s_n)$ of the same length, the Hamming distance is defined as:

$$D_{Hamming}(Q, S) = \sum_{i=1}^{n} \delta(q_i, s_i)$$

with

$$\delta(x, y) = \begin{cases} 0 & \text{if } x = y \\ 1 & \text{else} \end{cases}$$

- Use artificial symbols as a fill-up for sequences of different length (see example on the next slide)

Hamming Distance

- Example:

$$
\begin{array}{cccccc}
Q & = & \mathrm{t} & \mathrm{h} & \mathrm{r} & \mathrm{e} & \mathrm{e} \\
 & & | & \mathrm{x} & \mathrm{x} & | & \mathrm{x} \\
S & = & \mathrm{t} & \mathrm{r} & \mathrm{e} & \mathrm{e} & .
\end{array}
$$

(| = match, x = mismatch)

$D_{Hamming}(Q, S) = 3$

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

Hamming Distance

- Example of misaligned subsequences:

Q	=	T	Ü	R	S	C	H	L	O	S	S
		|	x	|	|	|	|	x	x	|	x
S	=	T	O	R	S	C	H	U	S	S	.

Q	=	T	Ü	R	S	C	H	L	O	S	S
		x	x	x	x	x	x	x	x	x	x
R	=	A	B	S	C	H	U	S	S	.	.

$D_{Hamming}(Q, S) = 4$ and $D_{Hamming}(Q, R) = 10$

- Similarity of subsequences SCHLOSS and SCHUSS is not considered (the correct alignment in S is by chance)

Edit Distance

- 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 ($\Diamond$) and three substitutions (:) are necessary for the transformation (five symbols are matches (|))

Q	=	T	Ü	R	S	C	H	L	O	S	S
		$\Diamond$	:	:	|	|	|	$\Diamond$	:	|	|
S	=	.	A	B	S	C	H	.	U	S	S

$D_{Edit}(Q, S) = 5$

- The mapping between elements is called optimal alignment and the Edit Distance represents the alignment cost

Edit Distance

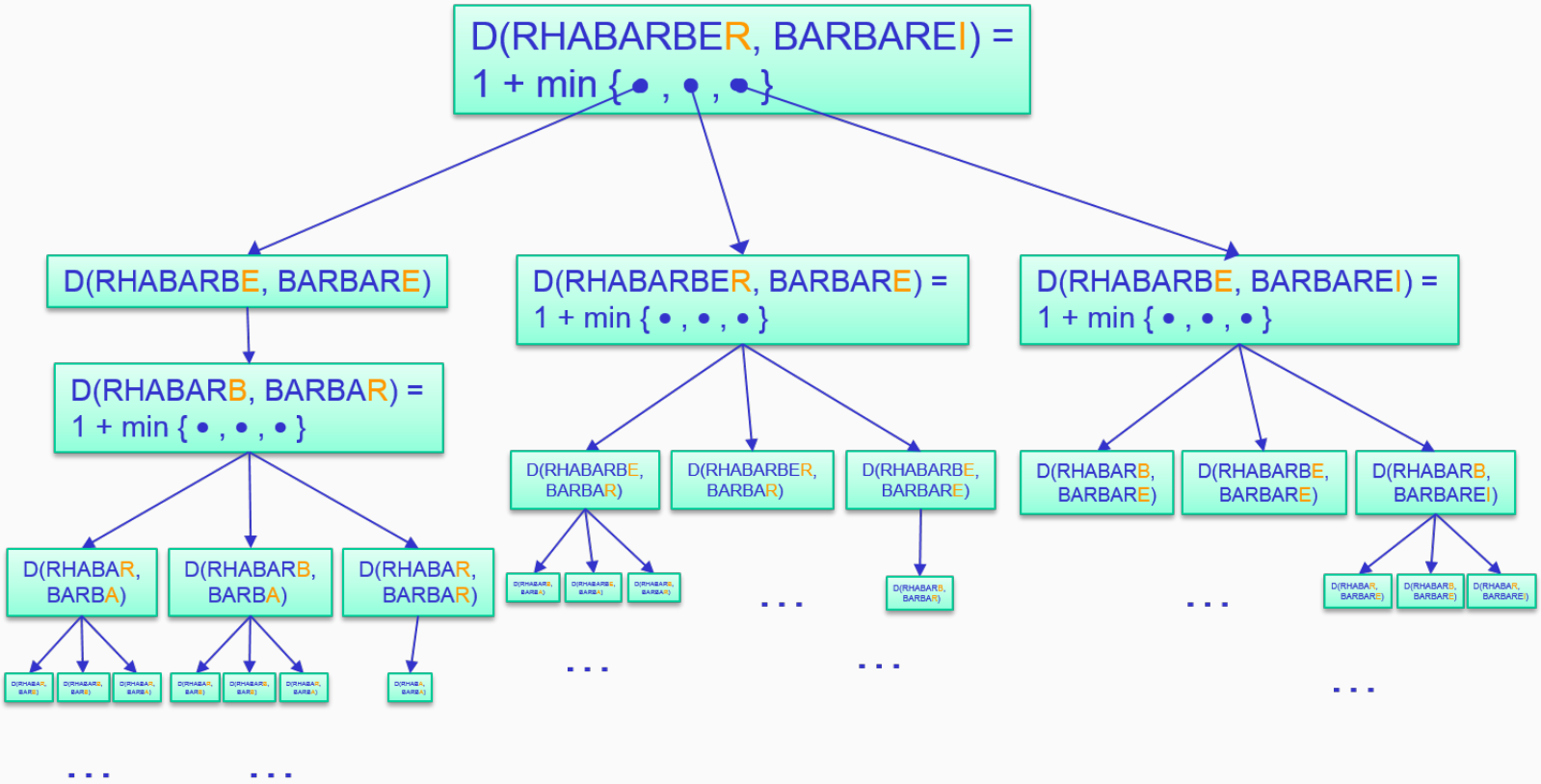
- Formally: Given a sequence $Q = (q_1, ..., q_n)$ let $start(Q) = (q_1, ..., q_{n-1})$ denote the prefix of Q and $last(Q) = q_n$ the last element of Q
- Given two sequences $Q = (q_1, ..., q_n)$ and $S = (s_1, ..., s_m)$, the Edit Distance is defined as

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

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

Edit Distance

- Naive Computation:



- For sequences of lengths n and m this tree has $O(3^{n+m})$ nodes

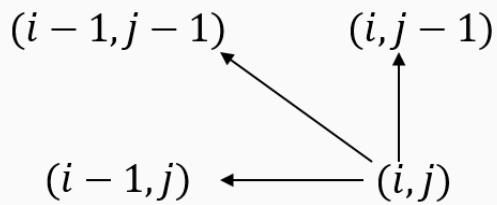
- Analysis:
  - $O(3^{n+m})$ function calls for sequences of lengths n and m
  - But many calls appear repeatedly
  - There are only $(m+1)\cdot(n+1) = O(m\cdot n)$ different recursive calls
- Solution
  - Store results of all calls (which requires $O(m\cdot n)$ space)
  - Systematic evaluation with $O(m\cdot n)$ operations
  - Scheme is called dynamic programming
- Illustration of the acceleration (example $m,n = 5,50,500$)

$5\cdot 5$	=	25	instead of	$3^{10} = 59,049$
$50\cdot 50$	=	$2,500$	instead of	$3^{100} = 5,154\cdot 10^{17}$
$500\cdot 500$	=	$250,000$	instead of	$3^{1000} = 1,322\cdot 10^{477}$

Edit Distance

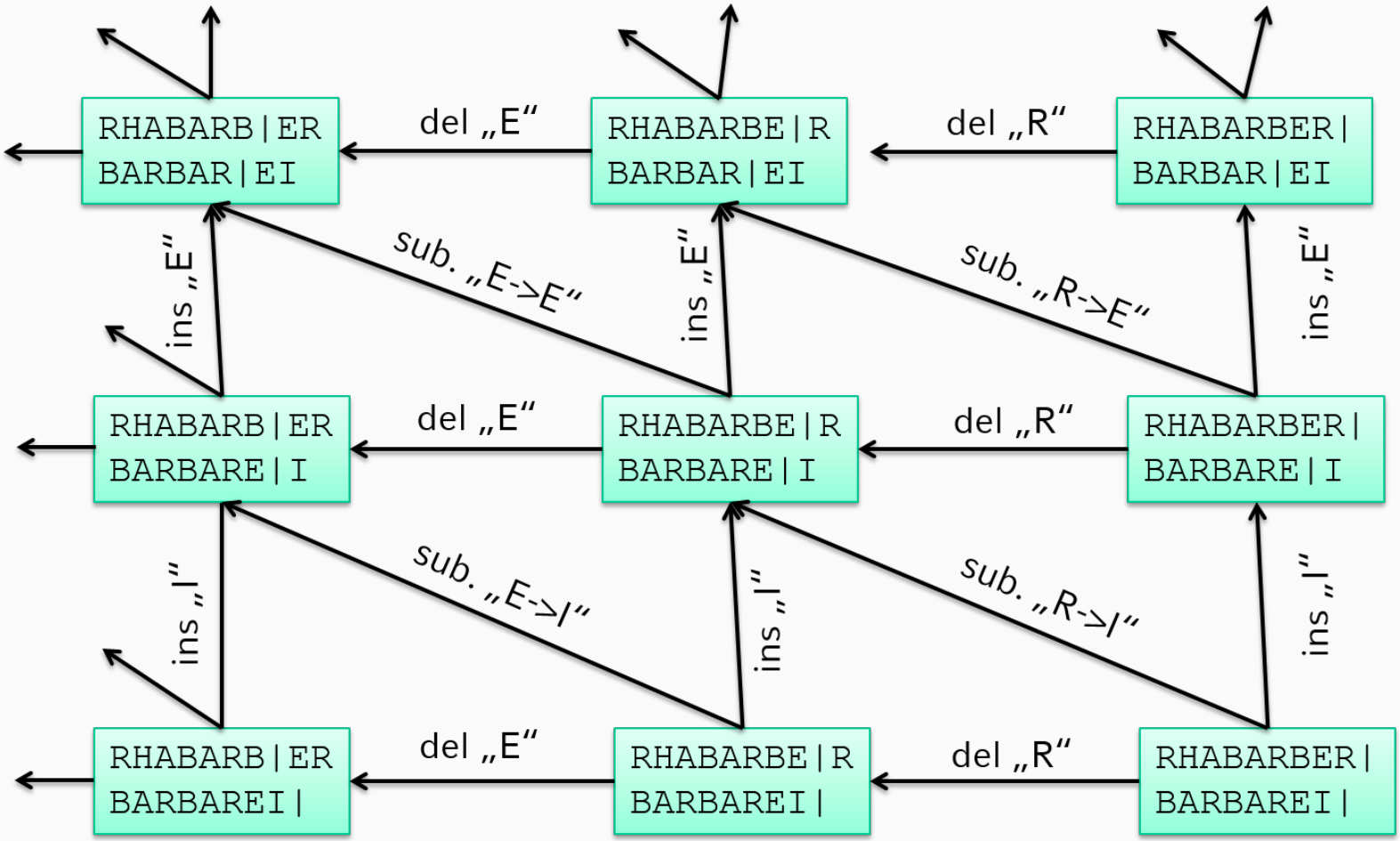
Dynamic Programming calculation scheme

- Horizontal step: $(i,j) \rightarrow (i-1,j)$
  Represents a deletion of the current character q_i in Q
- Vertical step: $(i,j) \rightarrow (i,j-1)$
  Represents an insertion of character s_i in Q at psoition i
- Diagonal step: $(i,j) \rightarrow (i-1,j-1)$
  Represents a substitution or match of the current character sq_i in Q and s_i in S



Edit Distance

- Visualization of the Dynamic Programming scheme:



Edit Distance

- 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: there may be a non-determinism, i.e., 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: computation of the Edit Distance via dynamic programming (one possible alignment path is highlighted)

	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

- Weighting of edit operations via a ground distance allows to define different costs for insertions, deletions, and substitutions
- Given two sequences $Q = (q_1, ..., q_n)$ and $S = (s_1, ..., s_m)$, the weighted Edit Distance w.r.t. a ground distance δ is defined as

$$D^{\delta}_{Edit}(Q,S) = \begin{cases} \sum_{i=1}^{n} \delta(q_i, \lozenge) & \text{if} \quad m = 0 \\ \sum_{i=1}^{m} \delta(\lozenge, s_i) & \text{if} \quad n = 0 \\ D^{\delta}_{Edit}(start(Q), start(S)) & \text{if} \quad last(Q) = last(S) \\ \min \begin{cases} D^{\delta}_{Edit}(start(Q), start(S)) + \delta(last(Q), last(S)), \\ D^{\delta}_{Edit}(Q, start(S)) + \delta(\lozenge, last(S)), \\ D^{\delta}_{Edit}(start(Q), S) + \delta(last(Q), \lozenge) \end{cases} & \text{else} \end{cases}$$

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

Properties of the Edit Distance

- As mentioned before, the optimal alignment of two sequences is not necessarily unique

- 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 $O(n \cdot m)$ for sequences of lengths n and m
- Common variant: First deletion of a symbol more expensive than repeated deletion (important in Bioinformatics)

Longest Common Subsequence (LCSS)

- 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
- The longer the sequence Z, the higher the similarity of Q and S and vice versa (so, attention, this is a similarity measure rather than a distance measure!)
- Example:

  Q: ACCGGTCGAGTGCGCGAAGCCGGCCGAA
  S: GTCGTTCGGAATGCCGTTGCTCTGTAA

  One possible solution is

  Z: GTCGTCGGAAGCCGGCCGAA

Formalization

- A sequence $Z = (z_1, ..., z_k)$ is a subsequence of sequence $Q = (q_1, ..., q_n)$ if there is a strictly increasing sequence $i_1, ..., i_k$ of indices of Q such that $\forall j = 1, ..., k$ it holds that $q_{i_j} = z_j$
- Example:
  - Let Q = (ABCBDAB)
  - The sequence Z = (BCDB) is a subsequence of Q
  - The corresponding index sequence[1] is $2, 3, 5, 7$
- A sequence $Z = (z_1, ..., z_k)$ is a common subsequence of two sequences $Q = (q_1, ..., q_n)$ and $S = (s_1, ..., s_m)$ if Z is a subsequence of both Q and S

[1] Note: in our definition, the index starts at 1 not at 0

Longest Common Subsequence (LCSS)

Formalization (cont.)

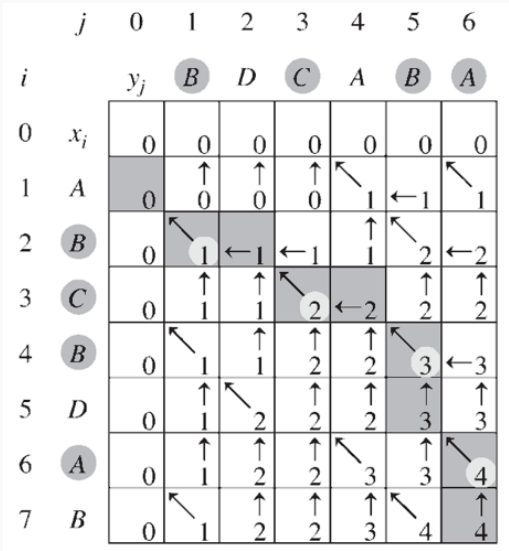
- Given two sequences $Q = (q_1, ..., q_n)$ and $S = (s_1, ..., s_n)$, the longest common subsequence problem is to find a maximum-length common subsequence $Z = (z_1, ..., z_k)$ of Q and S
- Example:
  - Let Q = (ABCBDAB) and S = (BDCABA)
  - The sequence Z = (BCA) is a common subsequence of Q and S but it is not the longest one:
  - Z' = (BCBA) or Z'' = (BDAB) are both the longest common subsequences of Q and S

Formalization (cont.)

- Given two sequences $Q = (q_1, ..., q_n)$ and $S = (s_1, ..., s_m)$, the LCSS similarity is defined as

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

- Properties
  - Computation similar to that of the Edit Distance
  - Computation time complexity via dynamic programming in $O(n \cdot m)$

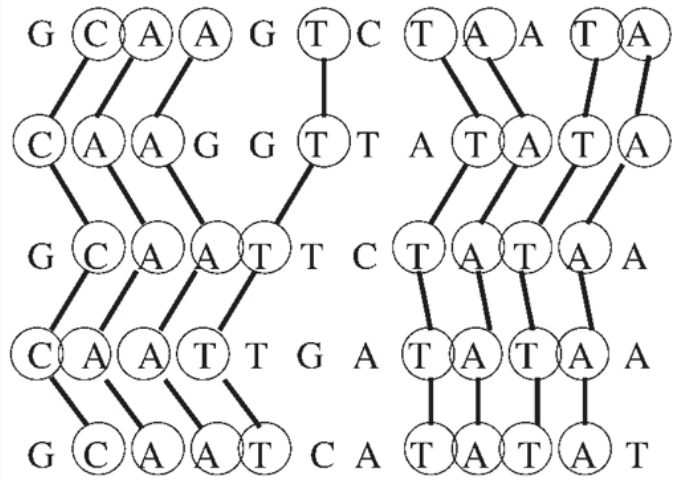


Variants of LCSS

- Downside: LCSS is not a distance measure and highly depends on the length of the analyzed sequences
- Distance function based on LCSS between two sequences $Q = (q_1, ..., q_n)$ and $S = (s_1, ..., s_m)$:

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

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



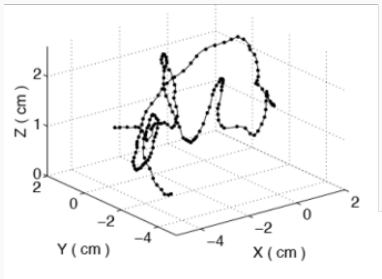
Mining Sequence Data

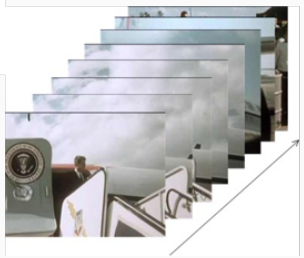
- Distance-based data mining
  - Views entire sequences as single objects/observations
  - Use one of the distance measures from above (or variants, or . . .) to compare sequences
  - Clustering, outlier detection, classification of sequence data
  - Does not mine sequential patterns but only patterns of similar sequences
- Sequential pattern mining
  - Usually searches for frequently occuring sub-patterns within a set of sequences
  - Count the frequency of subsequences (sub-patterns) in the sequence objects and report the frequent ones (sequential patterns)
  - Generalization of frequent item set mining (now order of occurrence matters)
  - Algorithms very similar to frequent item set mining

Kapitel 3: Time Series i

Time Series Data

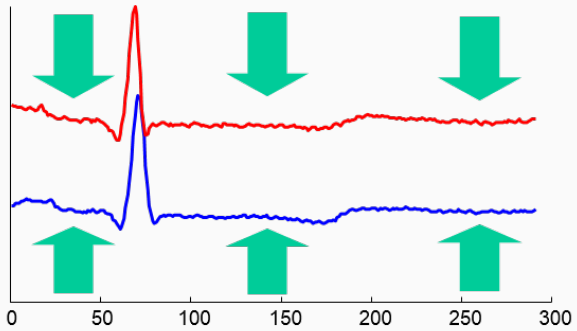
- Time series are a special type of sequences
  - Typically, values that are recorded over time
  - Index set I_n represents specific time points
- Univariate time series
  - stock prices
  - audio data
  - temperature curves
  - ECG
  - amount of precipitation
- Multivariate time series
  - trajectories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

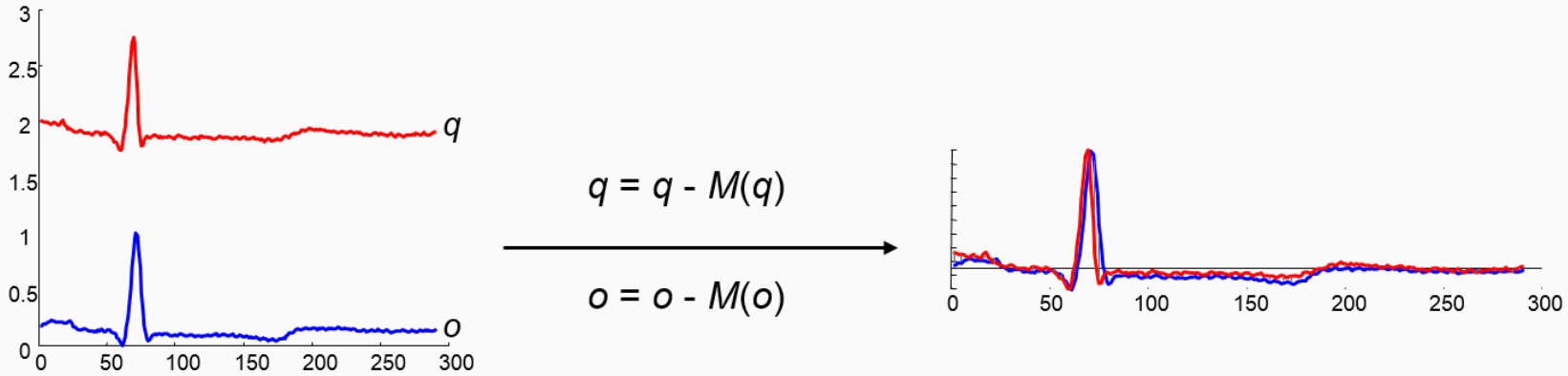




Data cleaning (removing artefacts, distortions, noise, ...)

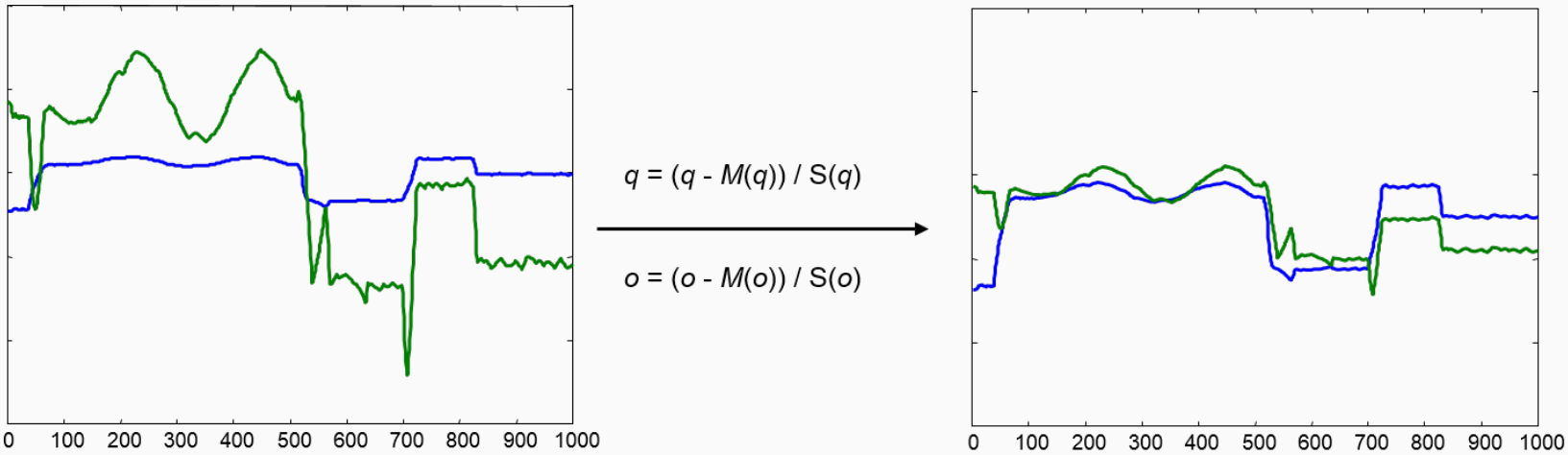
- Offset Translation (aka “Shifting”)
  - Time series are similar but have different offsets
  - Example: move each time series by its mean *M*





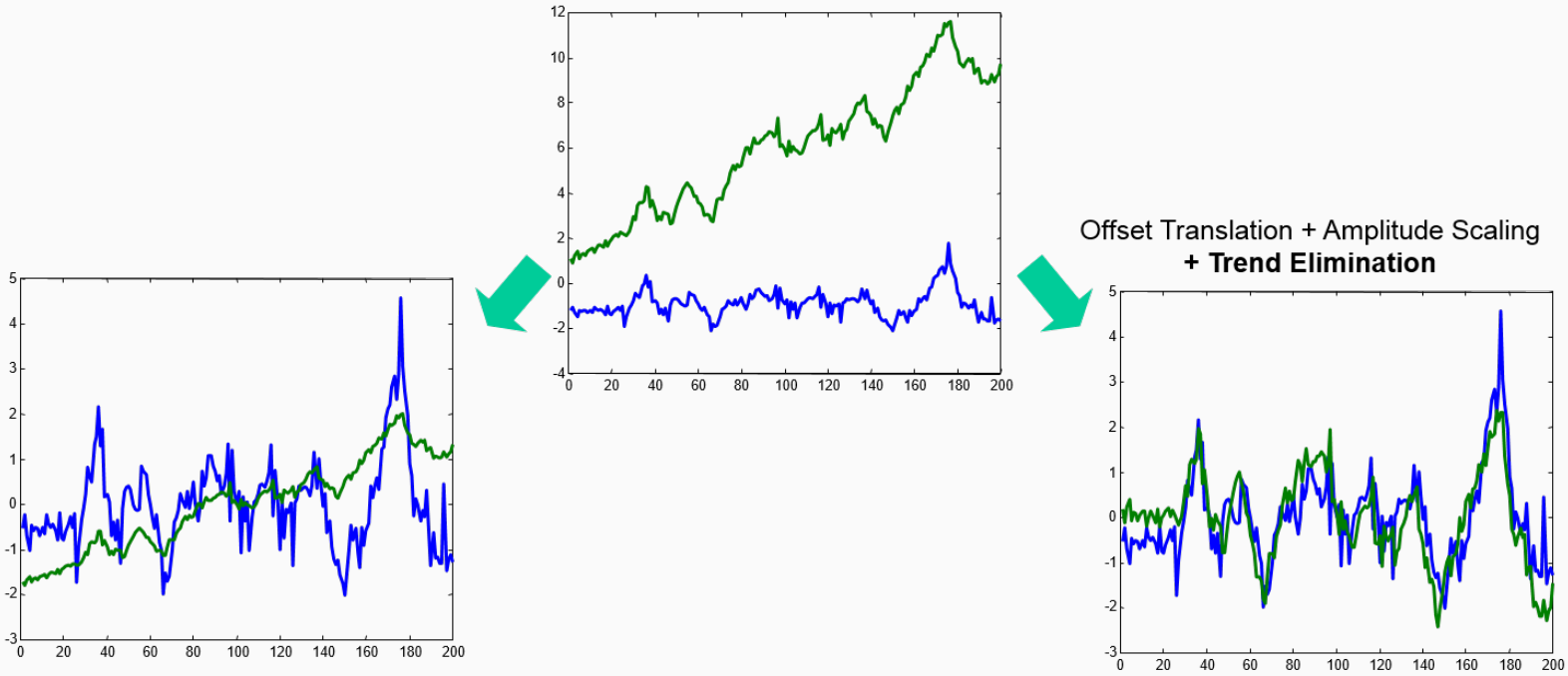
Data cleaning (removing artefacts, distortions, noise, ...)

- (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)



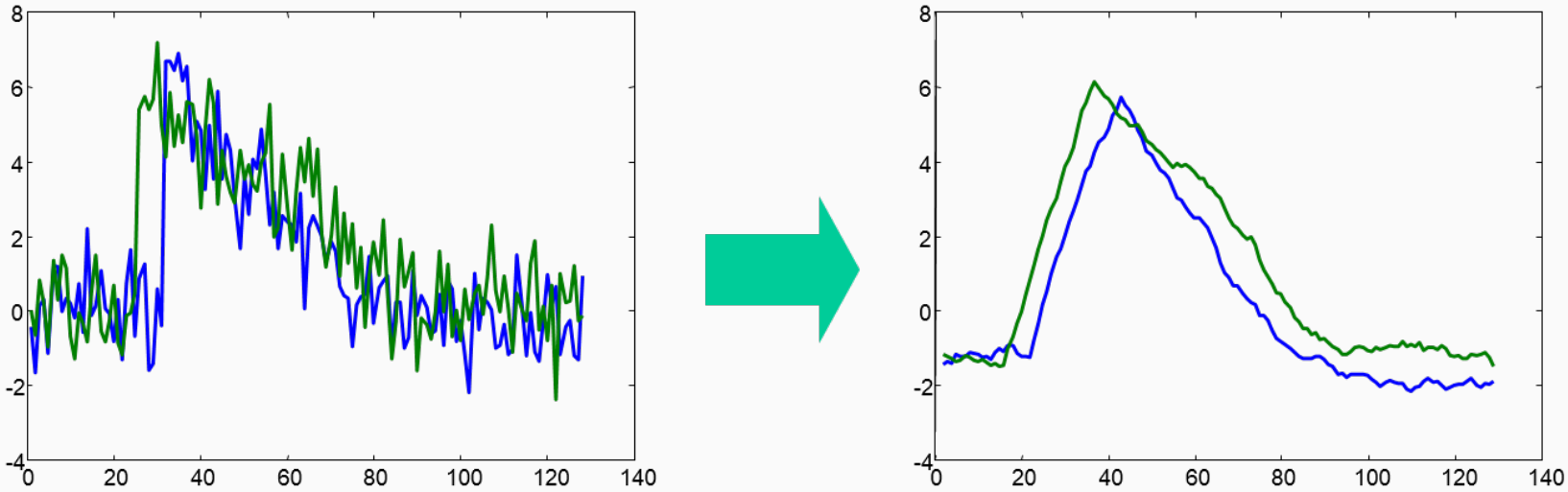
Data cleaning (removing artefacts, distortions, noise, ...)

- (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



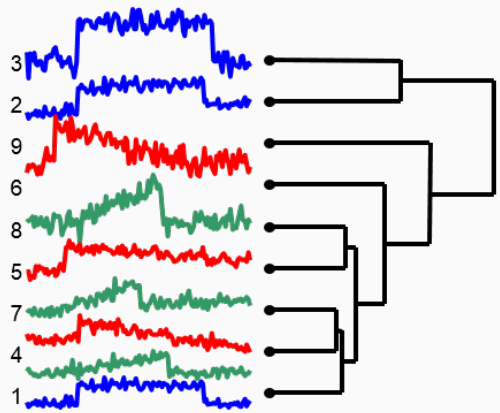
Data cleaning (removing artefacts, distortions, noise, ...)

- 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 $[v_{i-k}, \ldots, v_i, \ldots v_{i+k}]$

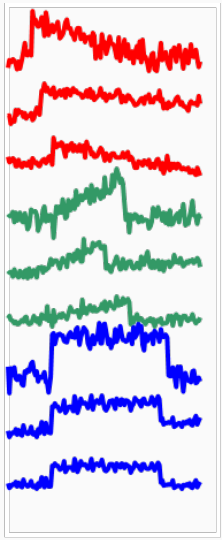


Time Series Data Preprocessing: Summary

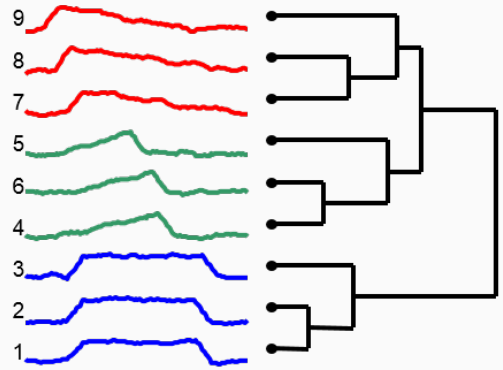
- There are many more types of distortions that might be of interest to be removed
- Application dependent
- Example:



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

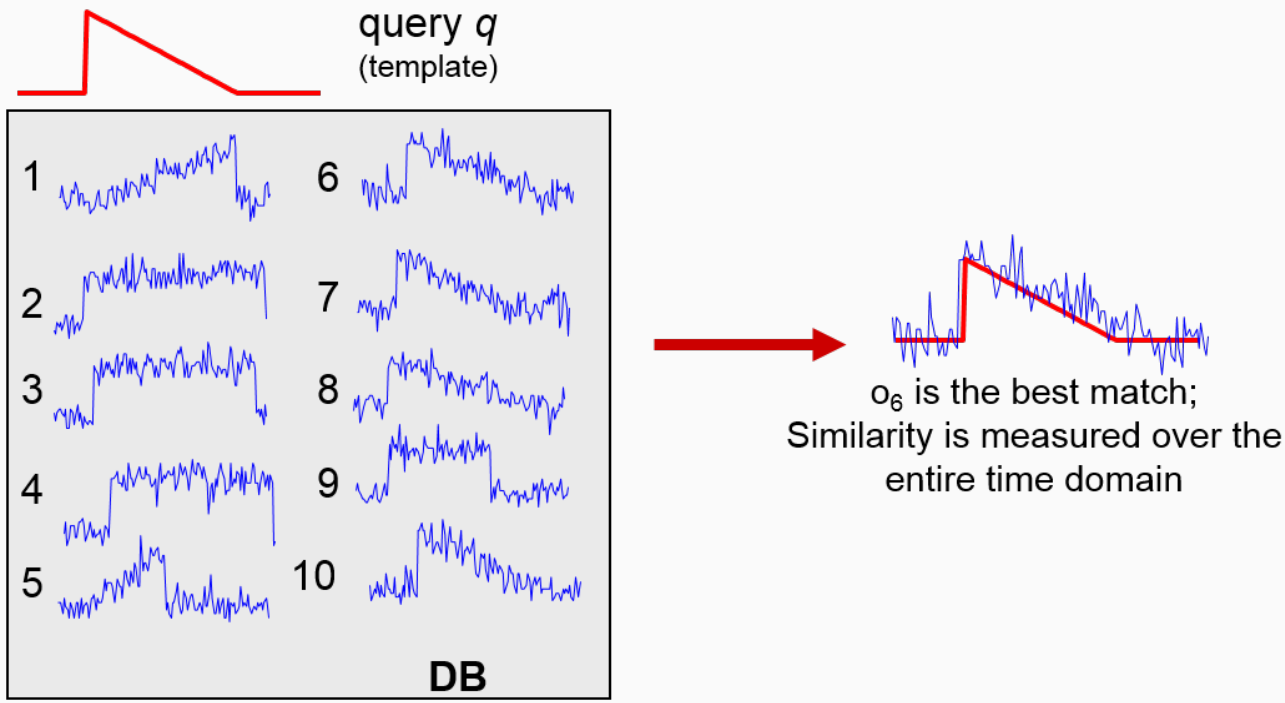
Time Series Similarity

- Some example similarity queries for time series databases
  - 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

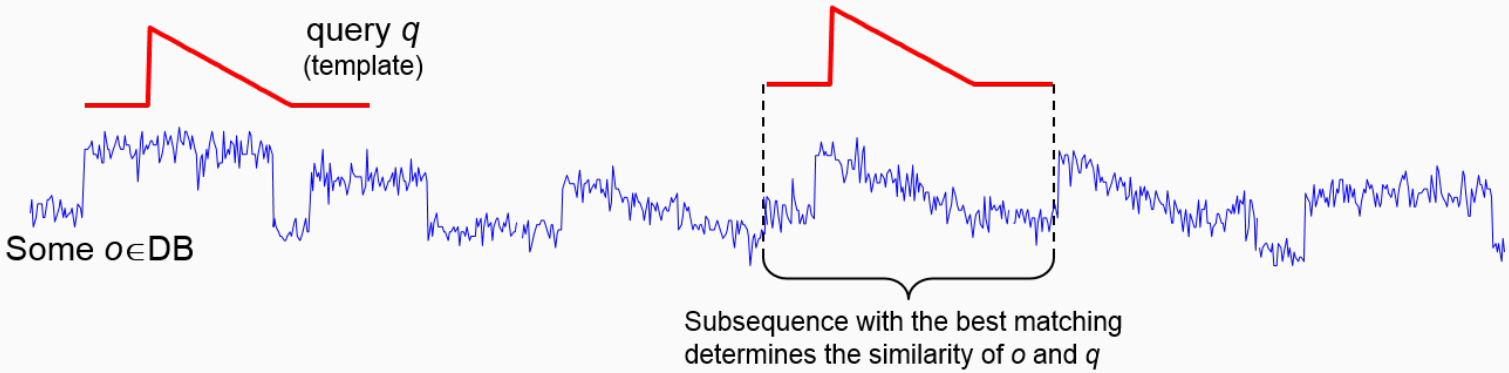
    Similarity = find the subsequence that has the best match

- Illustration of whole matching with a query template q

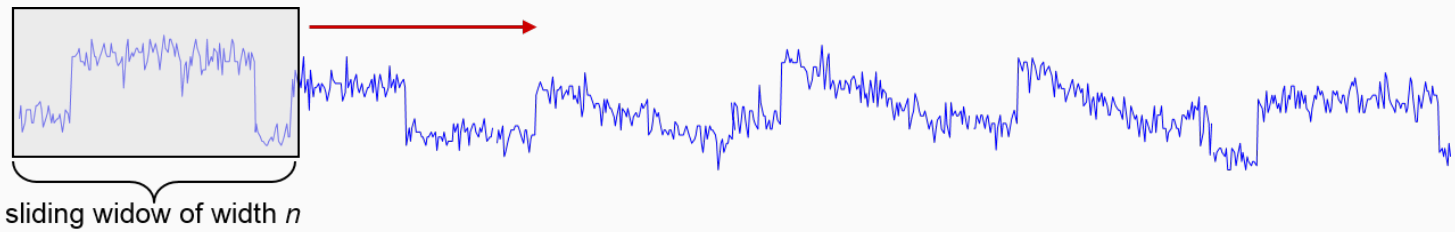


Time Series Similarity

- Illustration of subsequence matching with a query template q



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



Time Series Similarity

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

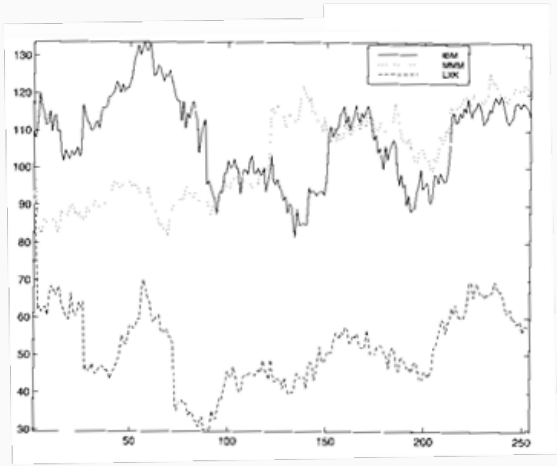
- Idea: Representation of a time series $X = (x_1, ..., x_n)$ as a n-dimensional Euclidean vector
- Given two time series X and Y of the same length n use an L_p-norm to compute the distance between the corresponding vectors

$$L_p(X, Y) = \sqrt[p]{\sum_{i=1}^{n} |x_i - y_i|^p}$$

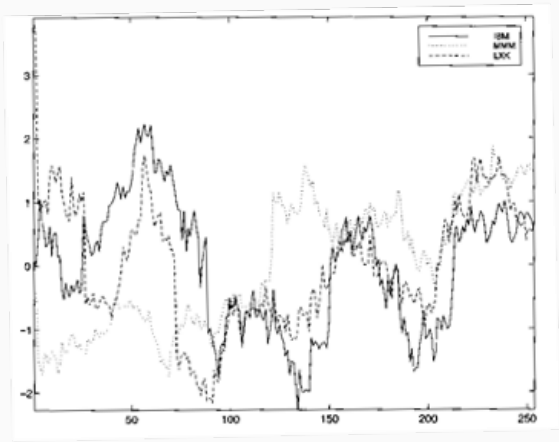
- Properties
  - Each time slot in X is compared to the same time slot in Y
  - Thus, sensitive w.r.t. variations on the time axis
  - Limited to time series having the same baseline, scale, and length

Minkowski Distances for Time Series

- If the two time series X and Y do not have the same length, they could still be considered similar (they have different baselines or amplitude scales)
- If we want to be invariant against some of those effects, normalization of time series (see: “preprocessing” above) can help, e.g.
  - Shifting by the average value (offset translation)
  - Scaling by the standard deviation (amplitude scaling)



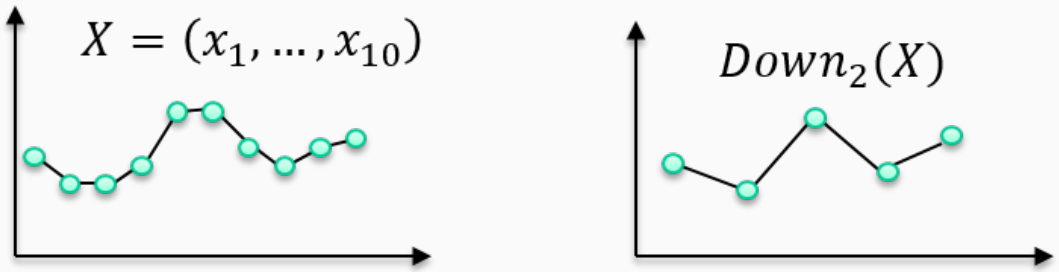
Stock price of IBM, LXK, MMM



Normalized stock prices

Scaling Time Series Along the Time Axis

- Until now: shifting and scaling is performed on the amplitude axis
- That still does not help for time series of different lengths
- For comparing time series with different lengths, we need the scaling of a time series X along the time axis as follows
  - Up-sampling (increase the resolution):
    Every x_i is repeated ω-times, i.e., $Up_\omega(x_1,...,x_n) = (z_1,..,z_{n\cdot\omega})$ where $z_i = x_{\lceil i/\omega \rceil}$ and $i = 1,...,n\cdot\omega$
  - Down-sampling (decrease the resolution):
    Only multiples of ω are used, i.e., $Down_\omega(x_1,...,x_n) = (z_1,..,z_{\lfloor n/\omega \rfloor})$ where $z_i = x_{i\omega}$ and $i = 1,...,\lfloor n/\omega \rfloor$



Uniform Time Warping (UTW)

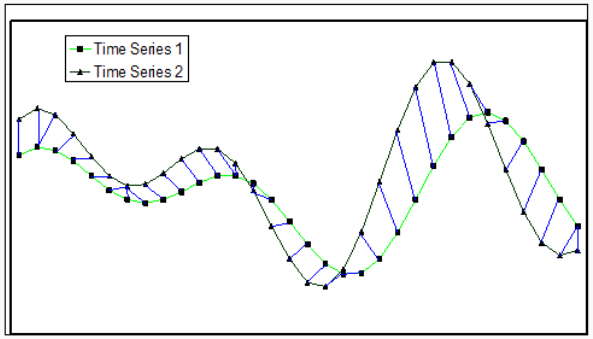
- UIdea: Scale both time series along the time axis to the same length and utilize the Euclidean Distance
- Given two time series $X = (x_1, ..., x_n)$ and $Y = (y_1, ..., y_m)$, the (squared) Uniform Time Warping Distance is defined as:

$$\begin{aligned} D^2_{UTW} &= \frac{L_2^2(Up_m(X), Up_n(Y)}{m \cdot n} \\ &= \frac{\sum_{i=1}^{n \cdot m}(x_{\lceil i/m \rceil} - y_{\lceil i/n \rceil})^2}{m \cdot n} \end{aligned}$$

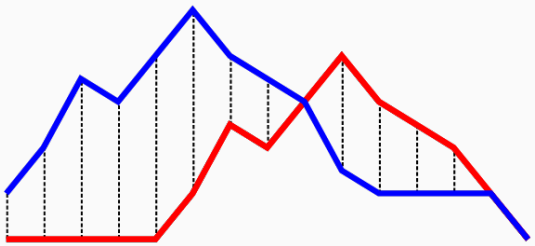
- Instead of up-sampling both X and Y with m and n, respectively, one could also use their lowest common multiple

Dynamic Time Warping (DTW)

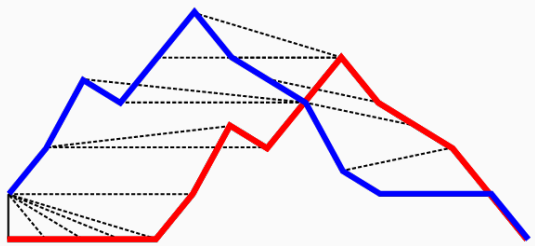
- Idea: Allow local (=dynamic) stretching of two time series in order to minimize the distance between them



- 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

DTW: Formal Definition

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

$$
\begin{aligned}
D^{\delta}_{DTW}(\emptyset, \emptyset) &= 0 \\
D^{\delta}_{DTW}(X, \emptyset) &= D^{\delta}_{DTW}(\emptyset, Y) = \infty \quad \text{for } X, Y \neq \emptyset \\
D^{\delta}_{DTW}(X, Y) &= \left(\delta(Last(X), Last(y))^p + \left(\min \left\{ \begin{array}{c} D^{\delta}_{DTW}(Start(X), Start(Y)) \\ D^{\delta}_{DTW}(X, Start(Y)) \\ D^{\delta}_{DTW}(Start(X), Y) \end{array} \right\} \right)^p \right)^{1/p}
\end{aligned}
$$

- Since time series are typically real-valued, the absolute difference (Manhattan distance. L_1) is often used as the ground distance δ:

$$\delta(x_i, y_j) = |x_i - y_j| = L_1(x_i, y_j)$$

- One of the most prominent variant of DTW is using Manhattan as a ground distance and $p = 2$

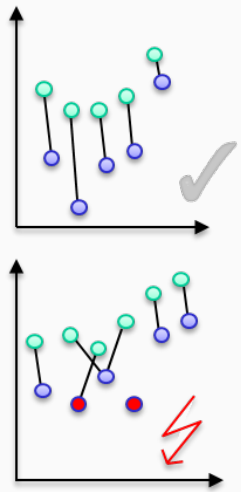
DTW: Warping Path

- DTW computes the alignment two time series $X = (x_1, ..., x_n)$ and $Y = (y_1, ..., y_m)$ represented by a warping path P of indices:

$$P = p_1, ..., p_L = (p_1^X, p_1^Y), ..., (p_L^X, p_L^Y)$$

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

- Properties of a warping path P
  - Boundary condition: $p_1 = (1, 1)$, $p_L = (n, m)$
  - Monotonicity: $p_t^X - p_{t-1}^X \geq 0$ and $p_t^Y - p_{t-1}^Y \geq 0$ (no movement back in time)
  - Continuousness: $p_t^X - p_{t-1}^X \leq 1$ and $p_t^Y - p_{t-1}^Y \leq 0$ (no time slot is missing in P)
  - Length $|P|$ bound: $\max(n, m) \leq |P| \leq n + m + 1$



DTW: Warping Path

- Let $\mathcal{P}$ denote the set of all paths satisfying the four properties of a warping path mentioned above
- The size of $\mathcal{P}$ is exponential in $n+m$
- Let the cost of a path $P = p_1, ..., p_L = (p_1^X, p_1^Y), ..., (p_L^X, p_L^Y)$ between two time series $X = (x_1, ..., x_n)$ and $Y = (y_1, ..., y_m)$ be defined as:

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

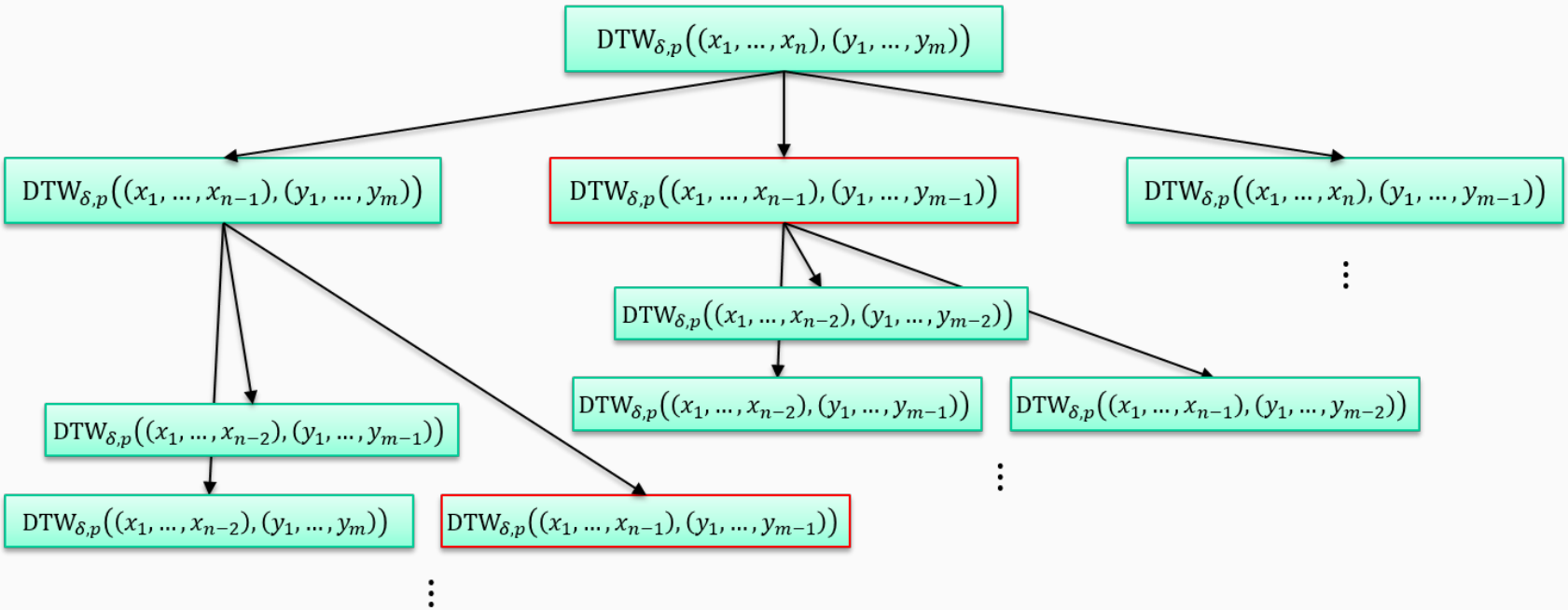
- D^2_{DTW} unsing L_1 as base distance and $p = 2$ can be defined by the path with the minimal cost:

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

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

DTW: Warping Path

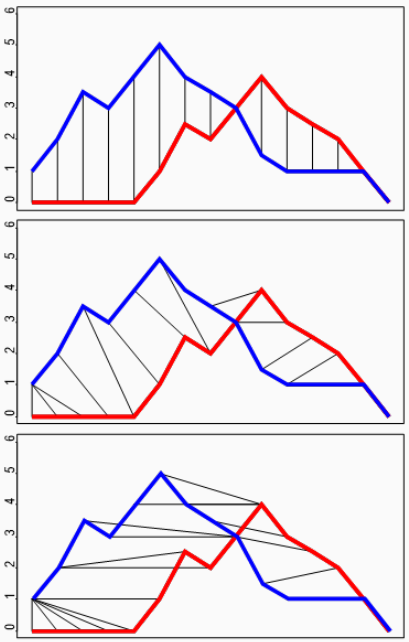
- Recursive computation of the DTW distance is again in $O(3^{n+m})$

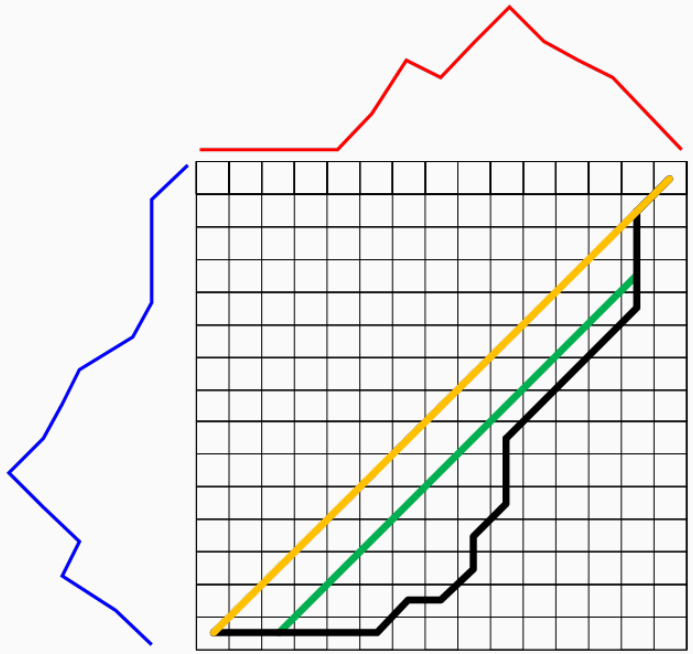


- Looks familiar to the Edit Distance? Absolutely ...

DTW: Warping Path

- Analogously, any path P between X and Y can be expressed as a path in a $|X| \times |Y|$ matrix used for computing the DTW by Dynamic Programming
- In the following example $(0,0)$ is in the lower left corner, so the blue time series is displayed from bottom to top rather than from top to bottom





DTW: Properties

- DTW is not a metric basically because it allows replication of elements
- Example: no identity of indiscernibles

  X: 1 2 2 2

  Y: 1 1 1 2

  $$D_{DTW}(X, Y) = 0$$

- Example: no triangle inequality

  X: 0 0 0 0

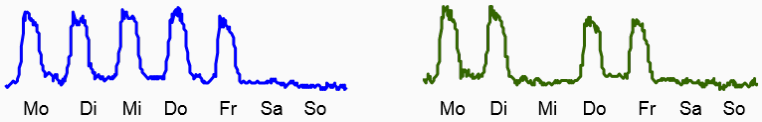
  Y: 1 2 2 3

  Z: 1 3 3 3

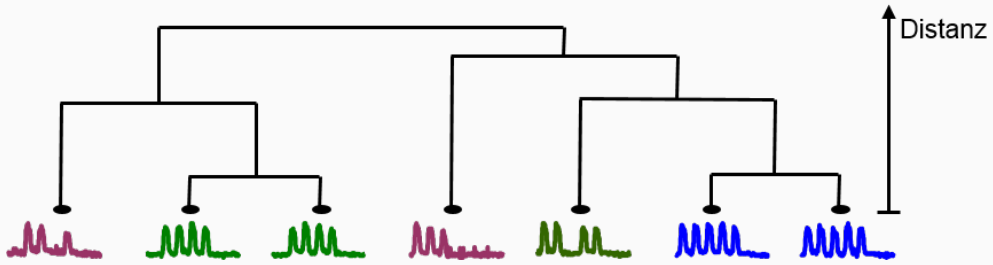
  $$D_{DTW}(X, Z) = 0 \not\leq D_{DTW}(X, Z) = 8 + D_{DTW}(X, Z) = 1$$

Comparison: DTW vs. Minkowski

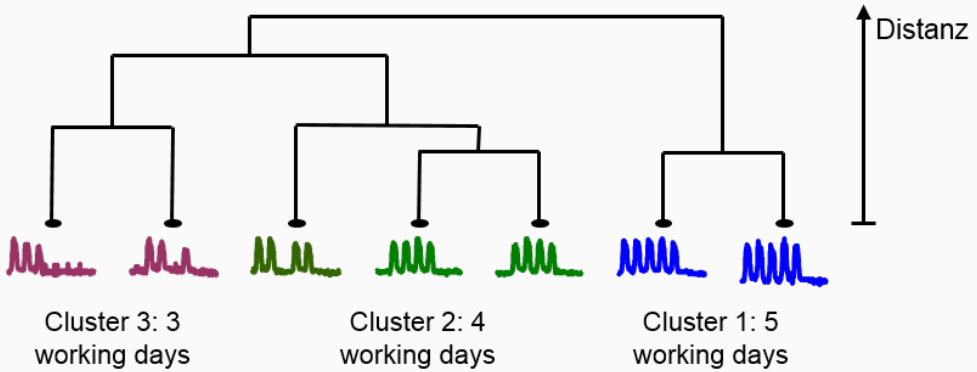
- Comparison on time series that measure the daily network traffic of a company, e.g.



- Minkowski

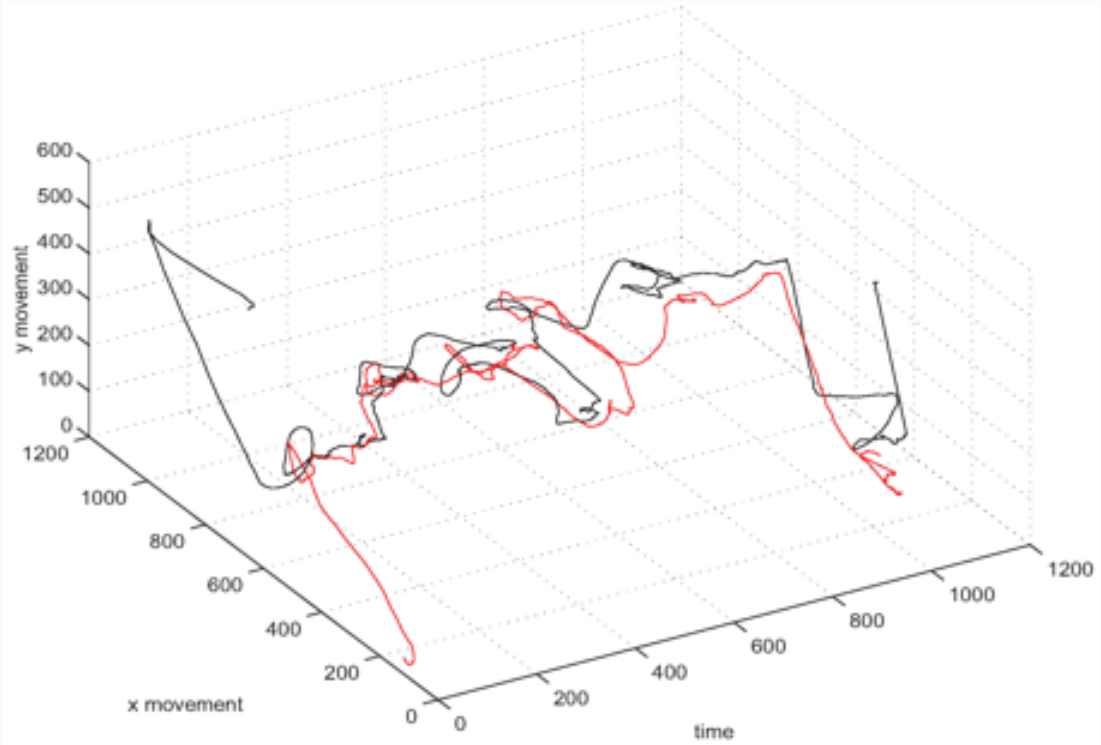


- DTW



LCSS for Time Series

- LCSS is tolerant to gaps in the two compared time series
- Example: trajectories (time series of spatial locations) that contain many outliers at start and end but share a long common route

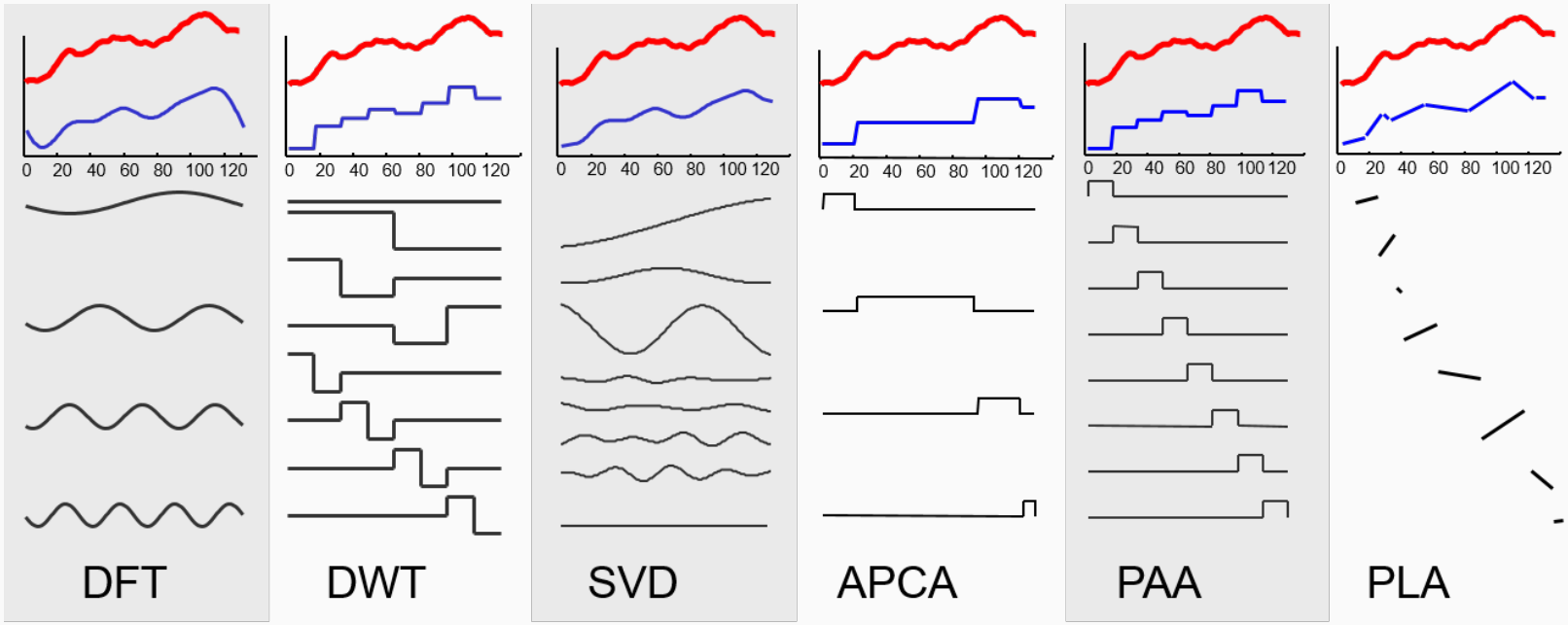


Feature Extraction in Time Series

- Instead of directly working with the entire time series, we can also extract features from them
- Many feature extraction techniques exist that basically follow one of the following two different purposes:
  1. 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
  2. Other model specific properties of the time series relevant to a given application
     - Example covered here: threshold-based modelling

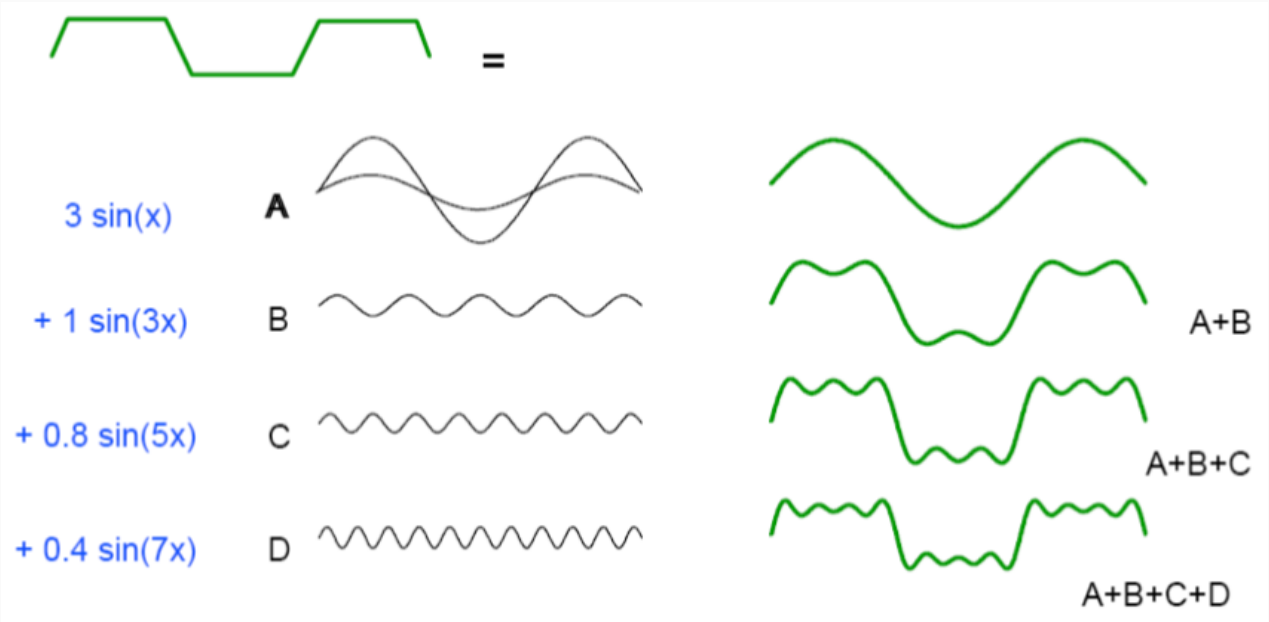
Compact Representations: Overview

- Aim: find a more compact representation (“dimensionality reduction”) of the original time series without loosing too much information
- General approach: represent time series by a combination of base functions



Compact Representations: DFT

- Discrete Fourier Transformation (DFT) is used to describe a periodical function as a weighted sum of periodical base functions (sin and cos) with varying frequency
- Example:



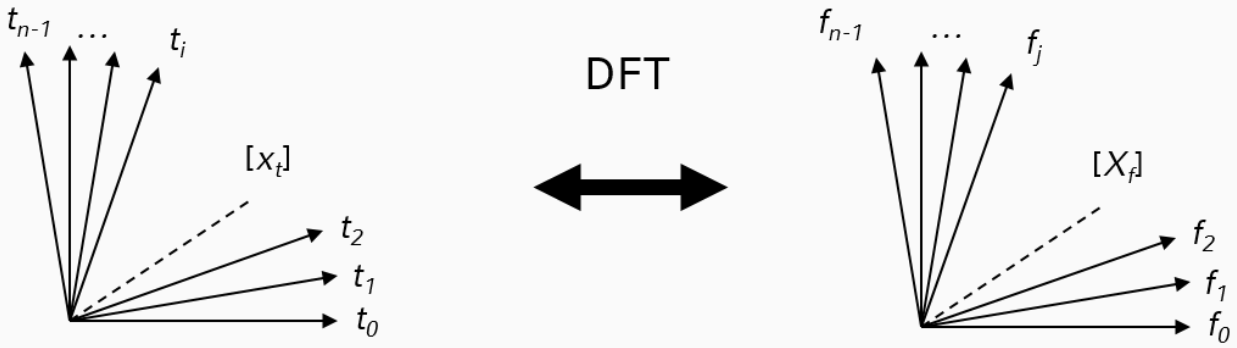
- Fouriers Theorem says that any periodic function can be represented by a sum of sin- and cos-functions of different frequency
- DFT originally finds a different equivalent representation
- DFT transforms a time series $x = [x_t]$ $(t = 0, ..., n-1)$ of length n into $X = [X_f]$ of n complex numbers with frequencies f such that

$$\begin{aligned} X_f &= \frac{1}{\sqrt{n}} \sum_{i=0}^{n-1} x_t \cdot e^{\frac{-j2\pi ft}{n}} \\ &= \underbrace{\frac{1}{\sqrt{n}} \sum_{i=0}^{n-1} x_t \cos(\frac{2\pi ft}{n})}_{\text{real part}} - j \cdot \underbrace{\frac{1}{\sqrt{n}} \sum_{i=0}^{n-1} x_t \sin(\frac{2\pi ft}{n})}_{\text{imaginary part}} \end{aligned}$$

  where $j^2 = -1$
- So the real part is the portion of cosine in f where the imaginary part is the portion of sinus

Compact Representations: DFT

- DFT can be interpreted as a transformation of the basis vectors (like e.g. PCA), the new basis is spanned by the frequencies



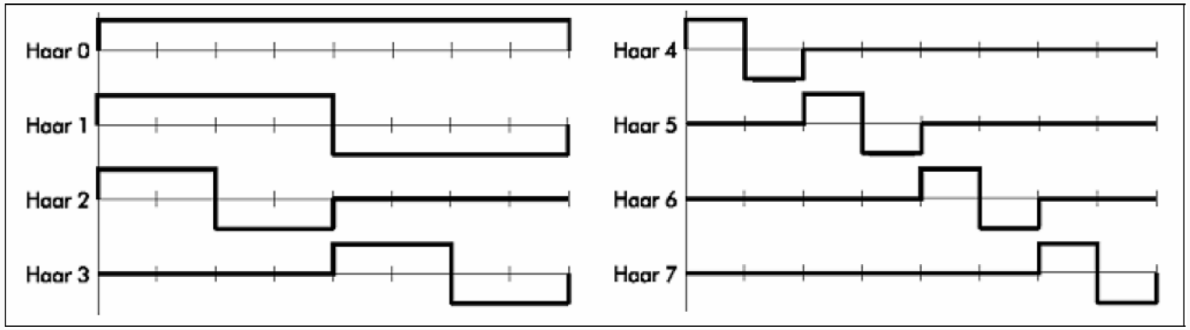
- But how does that help? So far, we transformed an n-dimensional time series into an n-dimensional vector …
- First of all, it holds that the euclidean distance is preserved after DFT, i.e. $\|x \check{} y\|^2 = \|X \check{} Y\|^2$
- This follows from Parseval's theorem (and the linearity of DFT): the energy of a sequence (= sum of squared amplitudes) is preserved by DFT, i.e.: $\sum_t \|x_t\|^2 = \sum_f \|X_f\|^2$

Compact Representations: DFT

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

Compact Representations: DWT

- Discrete Wavelet Transformation (DWT) represents a time series as a linear combination of Wavelet-functions (typically, Haar-Wavelets)



- Similar to DFT, DWT computes n components
- Properties
  - The more stationary the time series is, the better is the approximation with fewer $c < n$ components
  - Distance on DWT components also lower bounds Euclidean and DTW distance on original time series
  - Time series are restricted to be of length 2^i (for any i)

Compact Representations: DWT

- Computing the DWT: for each odd index i, compute mean μ of pairs x_i and x_{i+1} and the offset $x_i - \mu$
- Repeat this for the means until there is only one mean value (the mean of the original time series) left
- Example:

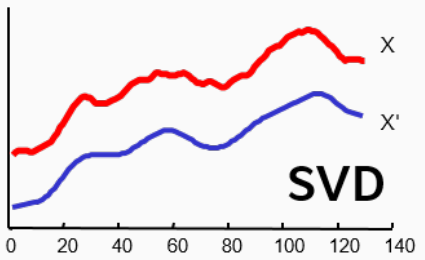
time series X	9	7	3	5	6	10	2	6
mean (1)	8		4		8		4	
offset (1)	1		-1		-2		-2	
mean (2)	6				6			
offset (2)	2				2			
mean (3)	6							
offset (3)	0							

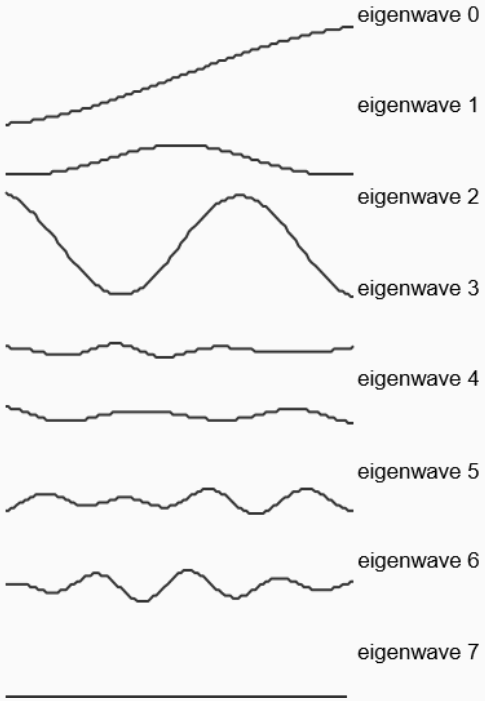
- Keep “last” mean and the offsets in reverse order of computation:

DWT representation of X	6	0	2	2	1	-1	-2	-2

Compact Representations: SVD

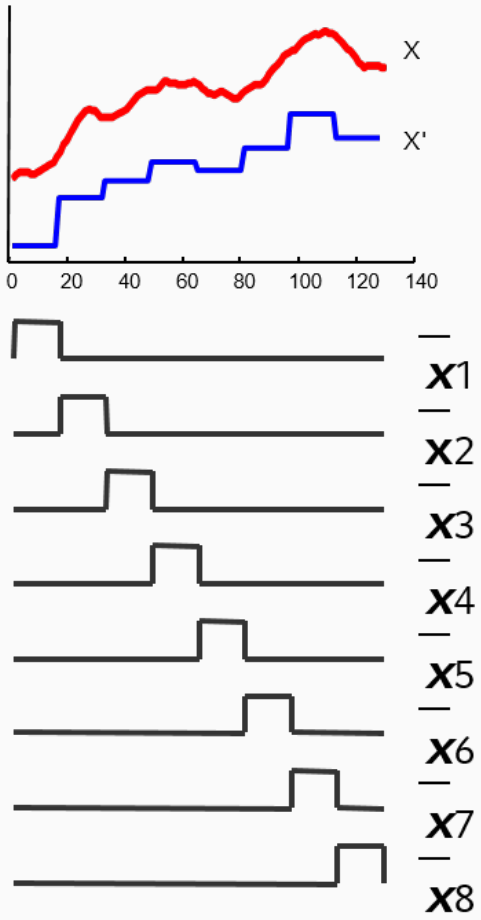
- SVD uses Eigen Waves instead of sinus/cosine as base functions
- Properties:
  - Minimizes the quadratic approximation error (like PCA and SVD on high dimensional data)
  - The semantics of the components of SVD depends on the actual data while DFT (sin/cos base functions) and DWT (constant base functions) are not data dependent
  - In text mining and Information Retrieval, SVD as a feature extraction technique is also know as “Latent Semantic Indexing”





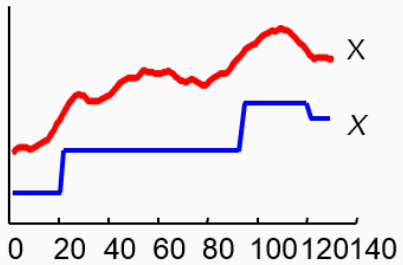
Compact Representations: PAA

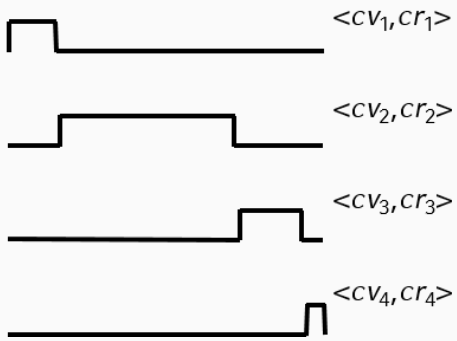
- Piecewise Aggregate Approximation (PAA) transforms time series into a sequence of box-functions
- Each box has the same length and approximates the interval by the mean of the corresponding values within this interval (similar to down-sampling using the mean as representative value)
- Properties:
  - Lower bounding property
  - Time series may have arbitrary length
  - Good if there are repeated intervals with the same length each having values with a similar amplitude



Compact Representations: APCA

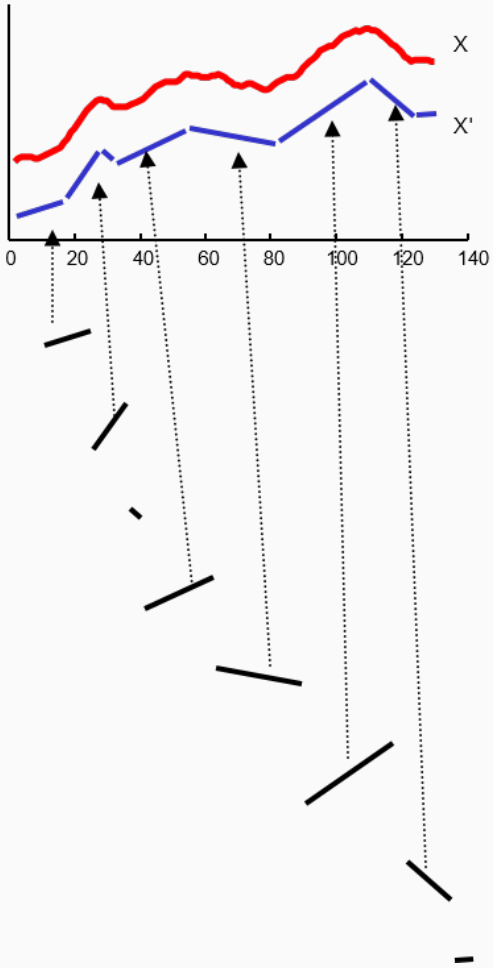
- Adaptive Piecewise Constant Approximation (APCA) is an extension to PAA
- Time series may have time intervals with a small amount of details (small amplitude) and intervals with a large amount of details (large amplitude)
- In that case PAA cannot account for varying amounts of detail
- APCA uses boxers of variable length (each segment now requires 2 parameters, the mean and the length of the interval)





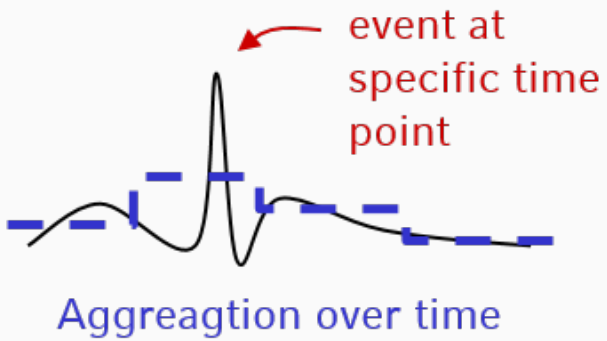
Compact Representations: PLA

- Piecewise Linear Approximation (PLA) is another extension to PAA
- It transforms a time series into a sequence of line segments $s = (length, height_{start}, height_{end})$
- Two consecutive segments need not to be connected
- PLA yields a richer approximation of the intervals but requires more parameters (accuracy of approximation depends on óf segments)



Application-specific Features

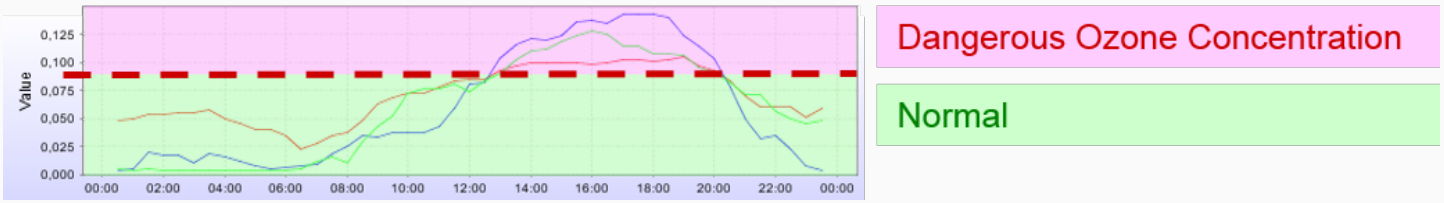
- An example of a specific feature transformation to model a special notion of similarity of time series is “threshold-based similarity”[2]
- Basic Idea
  - In some applications, only significant “events” that are defined by certain amplitudes (or amplitude values) are interesting
  - So far, the feature extraction extracts features modeling certain properties of time intervals but not of amplitude intervals



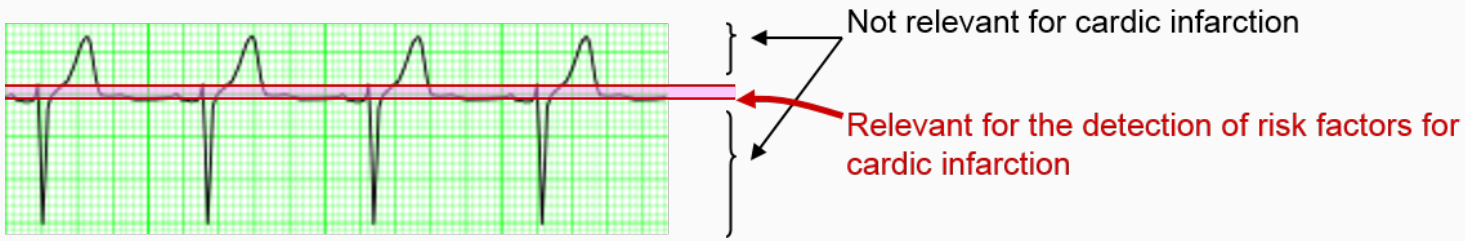
[2] Assfalg, Kriegel, Kröger, Kunath, Pryakhin, Renz. Proc. 10th Int. Conf. on Extending Database Technology (EDBT), 2006

Threshold-based Similarity: Application

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



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

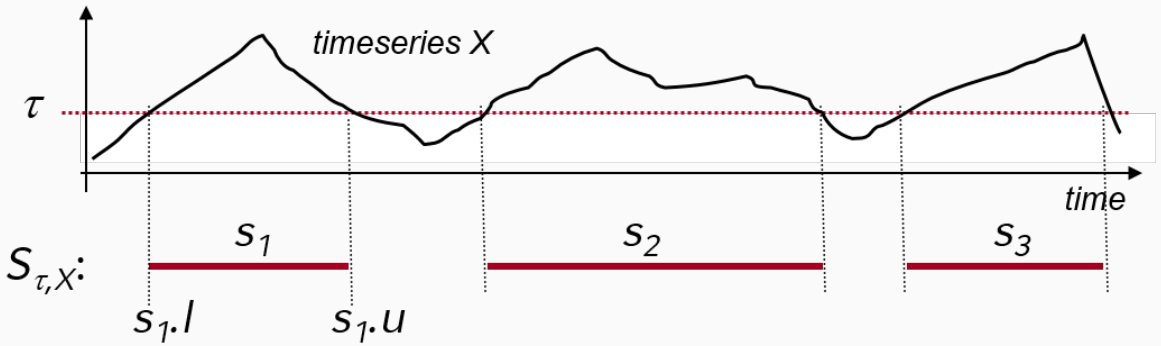


Threshold-based Similarity: Model

- Time series $X = (x_1, ..., x_n)$ is transformed into a sequence of intervals $S_{\tau,X} = (s_1, ...s_m)$, such that:

$$\forall i = 1, ..., n : (\exists s_j \in S_{\tau,X} : s_j.l < i < s_j.u) \Leftrightarrow x_i > \tau$$

($s_j.l$ denotes the start and $s_j.u$ denotes the end of s_j, respectively)

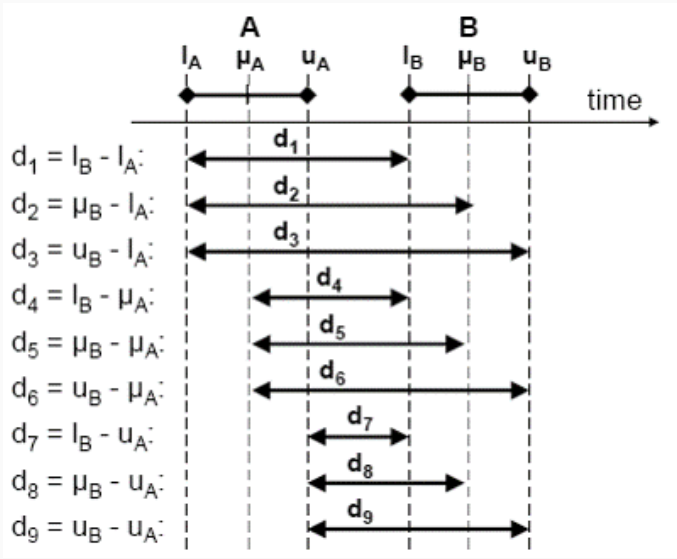


- Similarity of time series = similarity of sequences of intervals

Threshold-based Similarity: Model

- Similarity between sequences of intervals is defined based on a distance on pairs of single intervals
- For single pairs of intervals, the Euclidean distance on the start and end points are used, i.e.

$$d_{int}(s_i, s_j) = \sqrt{(s_i.l - s_j.l)^2 + (s_i.u - s_j.u)^2}$$



- Use the sum of minimum distances between two sequences of intervals S_X und S_Y (representing time series X and Y, resp.):

$$D_{TS}(S_X, S_Y) = \underbrace{\frac{1}{|S_X|} \sum_{s \in S_X} \min_{t \in S_Y} d_{int}(s,t)}_{S_X \longrightarrow S_Y} + \underbrace{\frac{1}{|S_Y|} \sum_{t \in S_Y} \min_{s \in S_X} d_{int}(t,s)}_{S_X \longleftarrow S_Y}$$

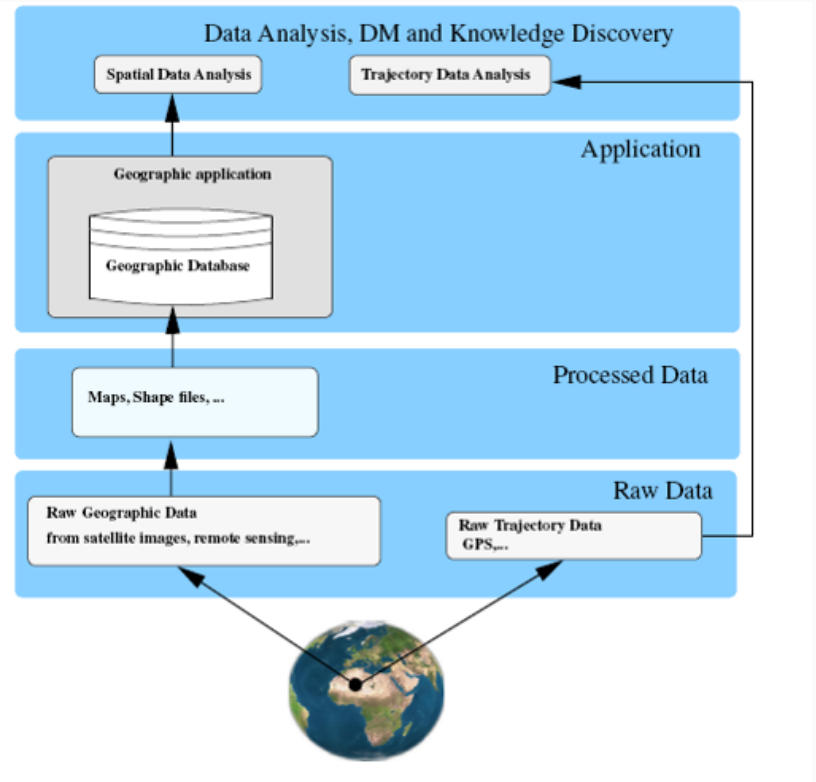
Kapitel 4: Spatio-temporal Data i

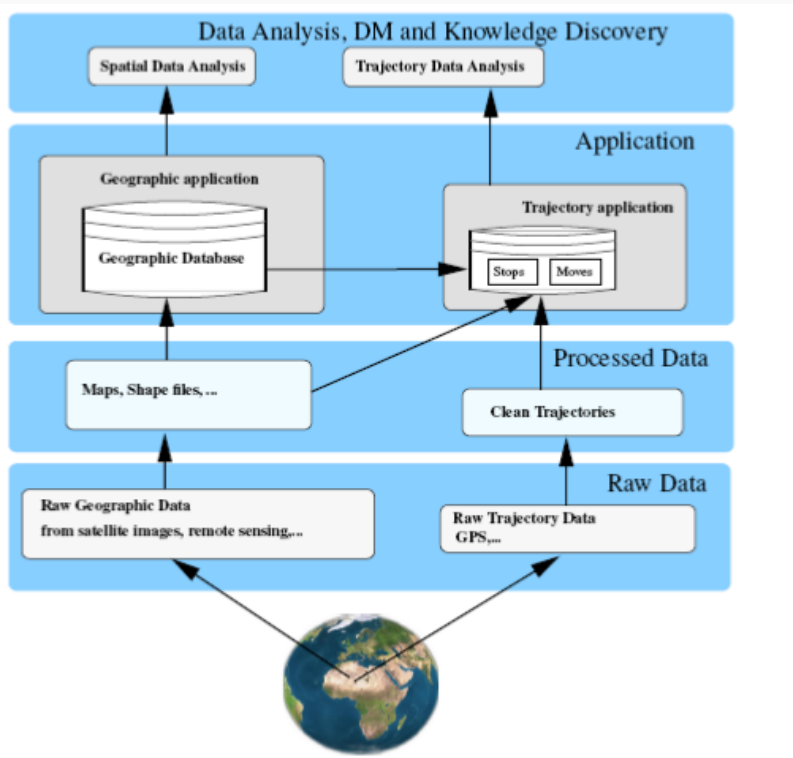
Spatio-temporal Data

- Spatio-temporal data is a special case of time series where (one of) the information recorded at each time point is the location of an object
- A time series over spatial locations is also called “trajectory”
- Often, there is additional information on time slots (e.g. semantic information on the location such as “museum” or “airport” . . .)
- The field of mining spatio-temporal data is wide so this lecture is just a basic (incomplete) snapshot of this field
- However, in general, there are two major approaches to trajectory mining described on the two next slides

Mining Spatio-temporal Data

- Geometry-based methods (left) consider only geometrical properties of trajectories, i.e., focus on “location-based” similarity
- Semantic-based methods (right) compute patterns based on the semantics of the data, independent of the specific spatial locations

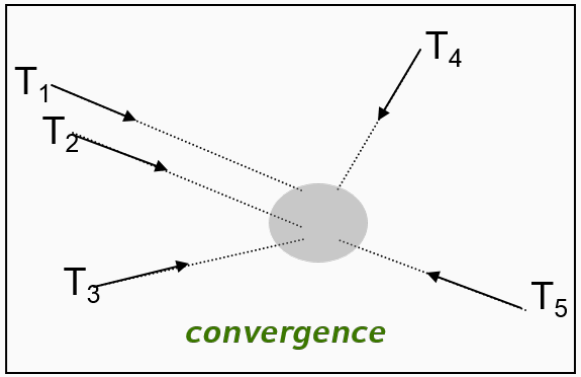




- Laube et al. 2004 proposed five basic patterns based on location, direction, and/or movement
- Several extensions of these patterns have been proposed over time

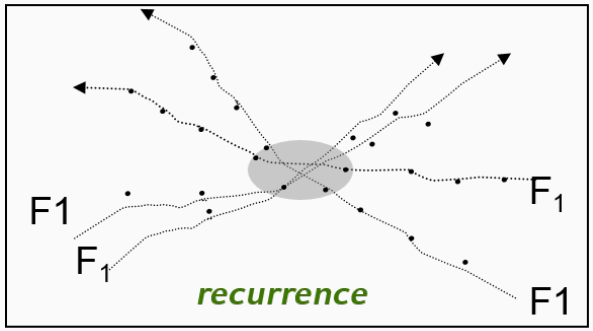
Convergence

At least *m* entities pass through the same circular region of radius *r* (regardless of time)



Recurrence

At least *m* entities visit a circular region of radius *r* at least *k* times



Encounter

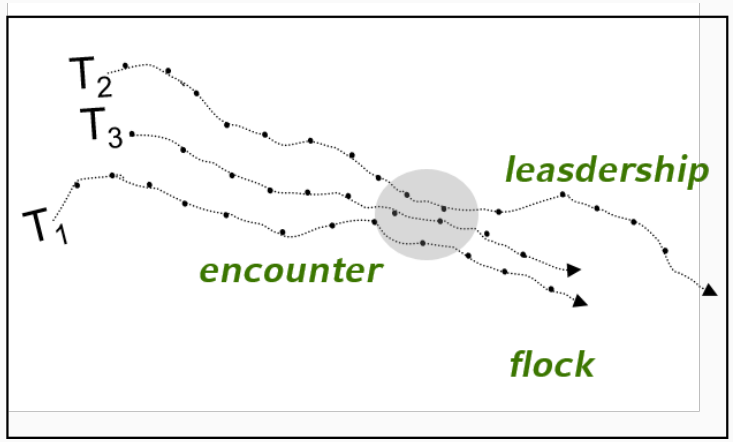
At least m entities are inside the same circular region of radius r, assuming they move with the same speed and direction (e.g. traffic jam at some moment if cars keep moving in the same direction)

Flock pattern

At least m entities are within a region of radius r and move in the same direction during a time interval $\geq s$ (e.g. a traffic jam)

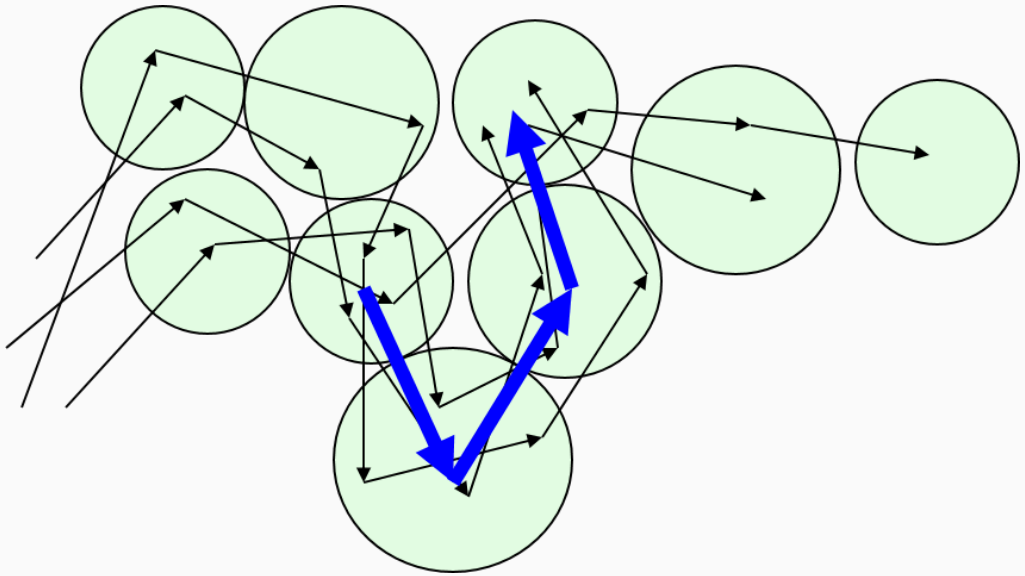
Leadership

At least m entities are within a circular region of radius r, they move in the same direction, and at least one of the entities (the leader) is heading in that direction for at least t time steps. (e.g. bird migration)



Geometry-based Trajectory Mining

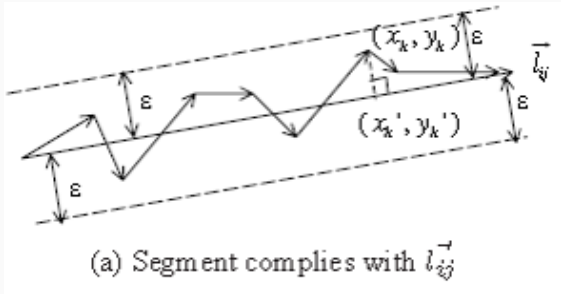
- All these patterns are based on the idea of high frequency patterns (and a kind of density-based notion: minim number of trajectories come together in a region of a specific volume)
- Beside those basic patterns, other frequent patterns have been proposed to be interesting, e.g. frequent followed paths/frequent sequential patterns



Geometry-based Trajectory Mining

Computing frequent sequential patterns (e.g. Cao 2005)

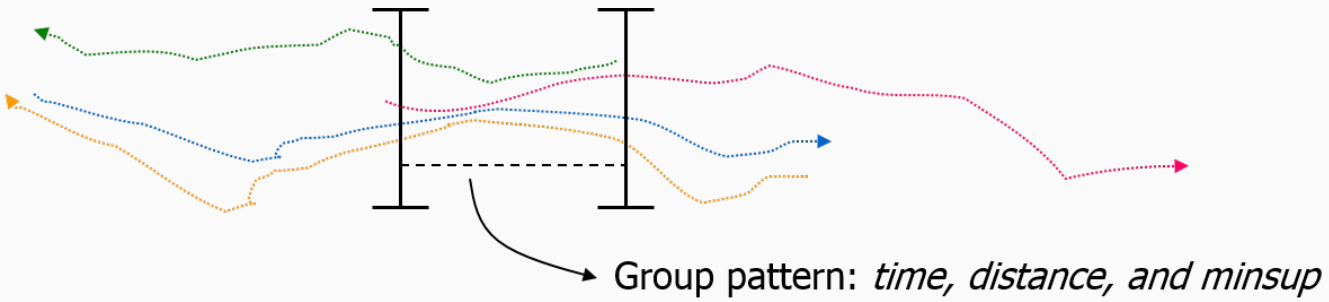
- Transforms each trajectory in a line with several segments
  - A distance tolerance measure is defined
  - All trajectory points inside this distance are summarized in one segment



- Similar segments are grouped
  - Similarity is based on the angle and the spatial length of the segment and, finally on a given distance threshold
  - Each group is represented by a region around the medium segment
- Frequent sequences of regions are computed w.r.t. a *minSup* threshold

Frequent mobile group patterns (Hwang 2005)

- A group pattern is a set of trajectories close to each other (with distance less than a given *minDist*) for a minimal amount of time (*minTime*)
- Direction is not considered
- Use Apriori algorithm to compute frequent groups



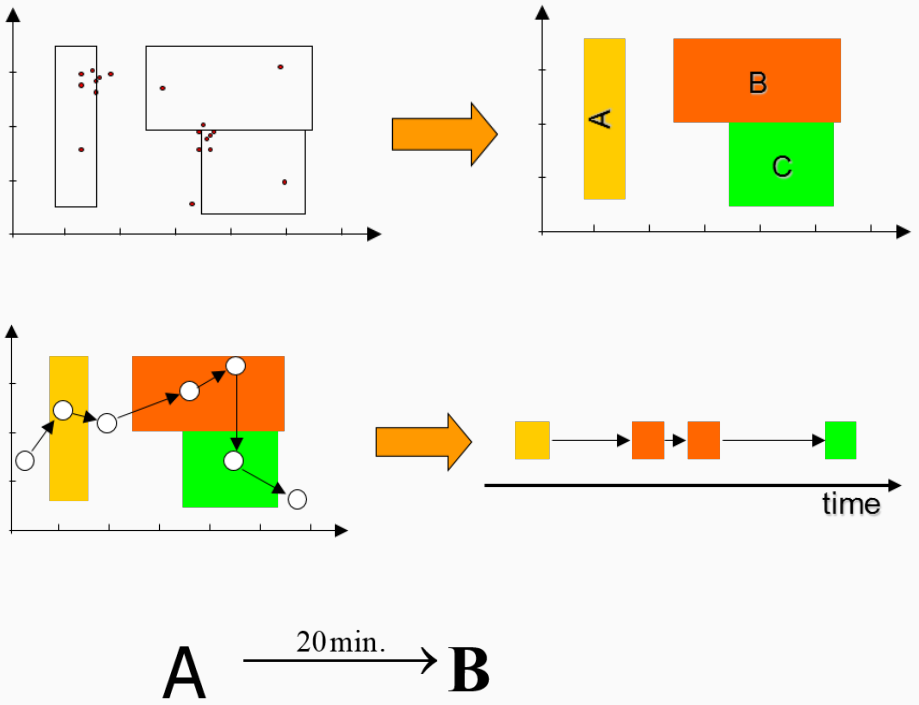
Trajectory Clustering (Han 2007)

- Algorithm TraClus: Group sub-trajectories using a density based clustering algorithm
- Two-step approach
  - Partition each trajectory in line segments with a user defined length L
  - Cluster similar line segments based on spatial proximity of the time points
- Similarity of line segments: Euclidean distance between segments (sub-trajectories)
- Could be anything else, e.g. a measure that also considers time (which is not covered here)

T-Patterns: Sequential Trajectory Pattern Mining (Giannotti 2007)

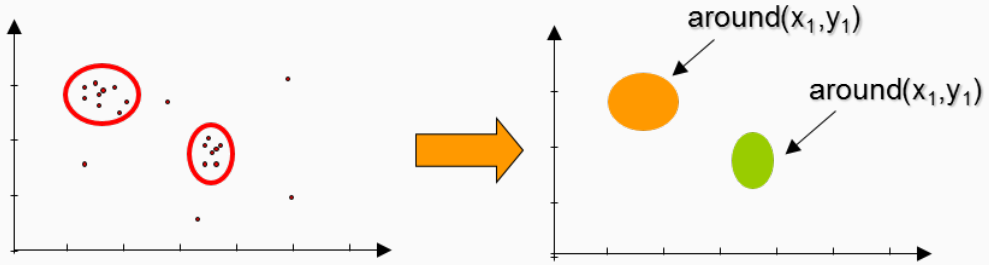
- Describes frequent behavior in terms of visited regions (ROIs) considering both space and time
- General idea (three-step approach):

1. Compute region of interests (ROIs)
2. Transform trajectory into sequ. of ROIs
3. Compute T-Patterns, i.e., sequences of regions visited during the same time intervals

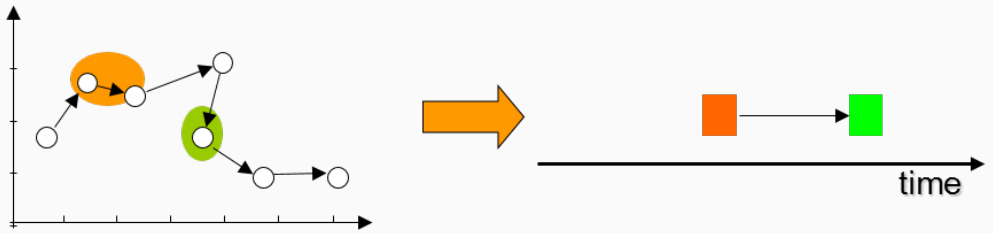


Geometry-based Trajectory Mining

- Step 1: Compute regions of interest (ROIs), i.e., regions with many trajectories (regardless of time)



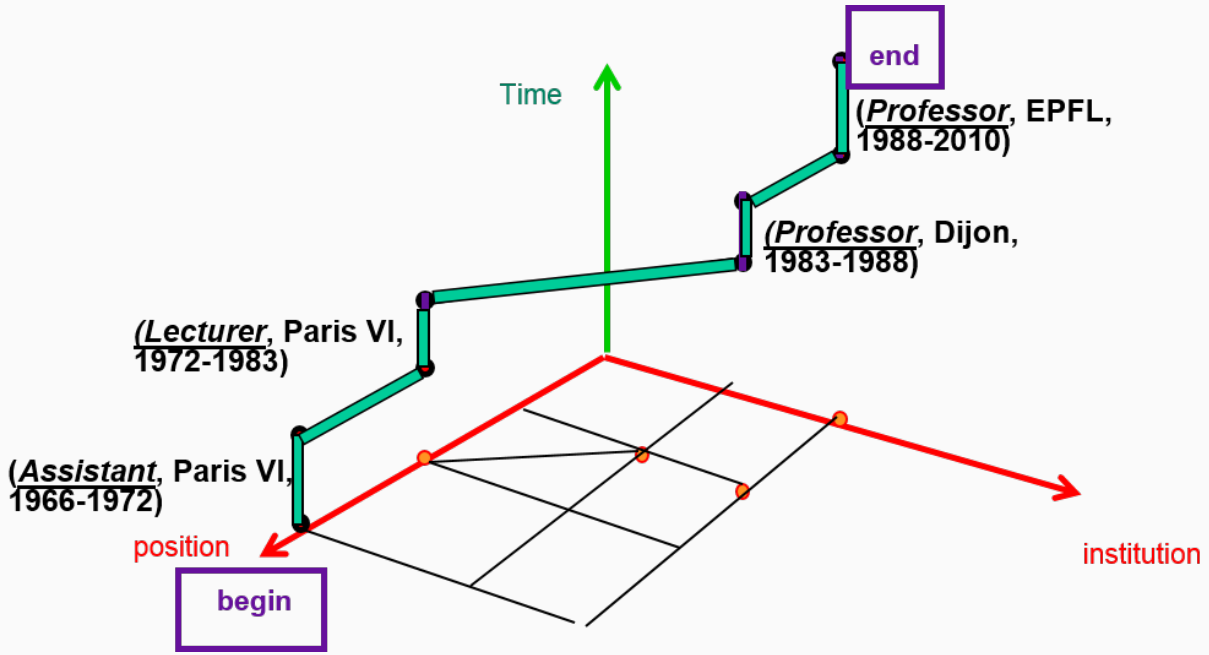
- Step 2: Transform trajectory into a sequence of ROIs: select trajectories intersecting at least with two regions in a sequence and annotate the time traveled between regions



- Compute T-Patterns, i.e., sequences of regions visited during the same time intervals

A Conceptual View on Trajectories (Spaccapietra 2008)

- Trajectory is a spatio-temporal object that has generic features (independent of the application) and semantic features (depend on the application)
- Trajectory = travel in abstract space, e.g. 2D career space:

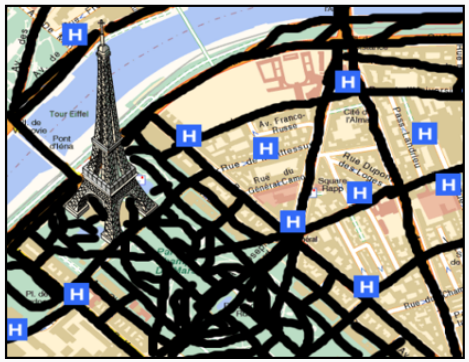


Semantic trajectories

- Idea (Sketch):

Trajectory Samples (x,y,t)

Geographic Data



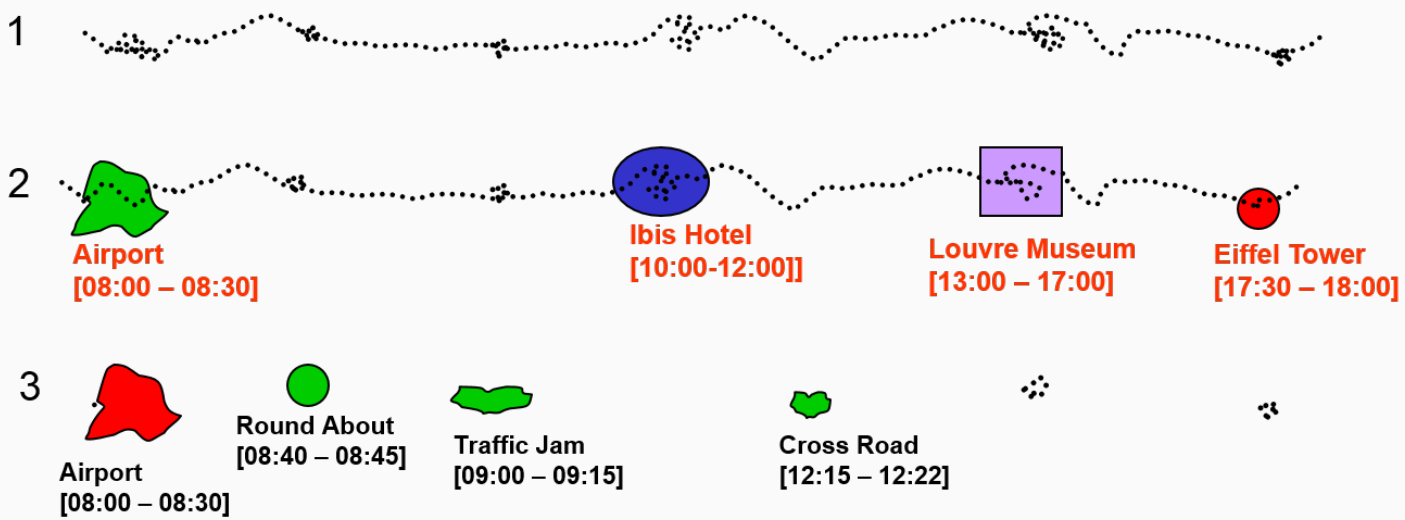
Geographic Data +
Trajectory Data =
Semantic Trajectories

Semantic trajectories (cont.)

- Key challenge: differentiate between stops and moves
- STOPS
  - Important parts of trajectories
  - Where the moving object has stayed for a minimal amount of time
  - Stops are application dependent
  - Tourism application: Hotels, touristic places, airport, . . .
  - Traffic Management Application: Traffic lights, roundabouts, big events. . .
- MOVES
  - Are the parts that are not stops

Semantic trajectories (cont.)

- Stops and moves are independent of the application

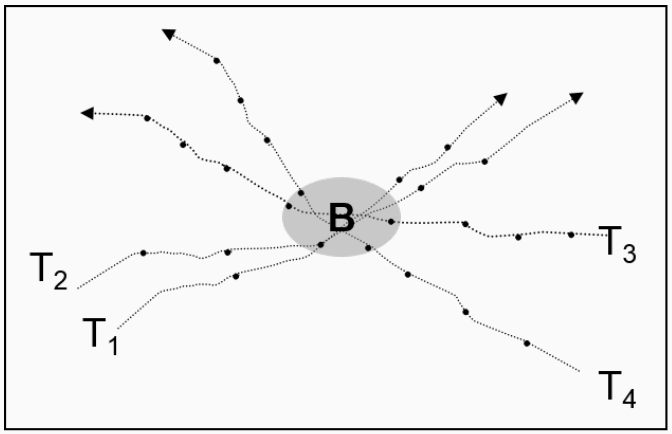


Semantic-based Methods

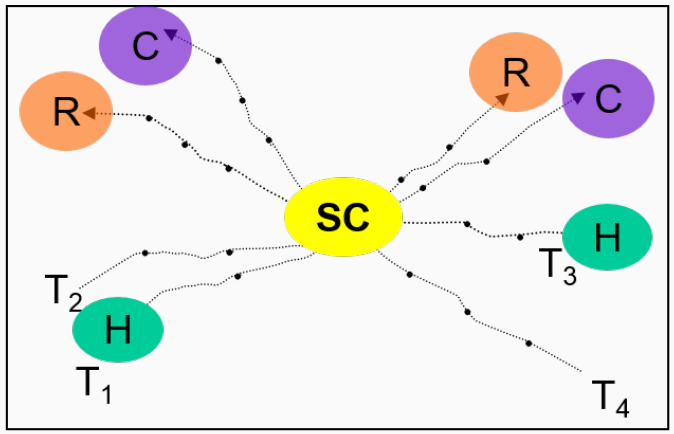
Geometric Patterns enriched by semantics (Bogorny 2008)

- Very little semantics in most trajectory mining approaches (geometry-based approaches)
- Patterns are purely geometrical and hard to interpret
- Thus: enrich geometric patterns with semantic information
- This idea stimulated many approaches on how to add semantics to trajectories

- Example (semantic enrichment)



Geometric Pattern



H Hotel R Restaurant C Cinema

Semantic trajectory Pattern

(a) Hotel to Restaurant, passing by SC
(b) go to Cinema, passing by SC

Stop and Move computation: SMoT (Alvares 2007a)

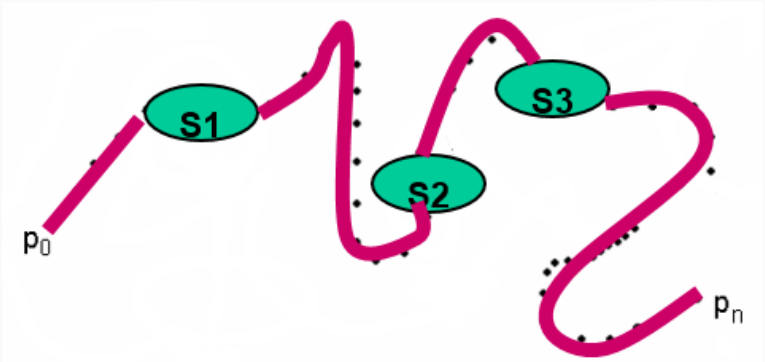
- A candidate stop C is a tuple (R_C, Δ_C), where
  - R_C is the geometry of the candidate stop (spatial feature type)
  - Δ_C is the minimal time duration

  Example: (Hotel, 3 hours)
- An application A is a finite set $A = \{C_1 = (R_{C_1}, Delta_{C_1}), \ldots, C_N = (R_{C_N}, \Delta_{C_N)}\}$ of candidate stops with non-overlapping geometries $R_{C_1}, \ldots, R_{C_N}$

  Example: {(Hotel, 3 hours), (Museum, 1 hour)}

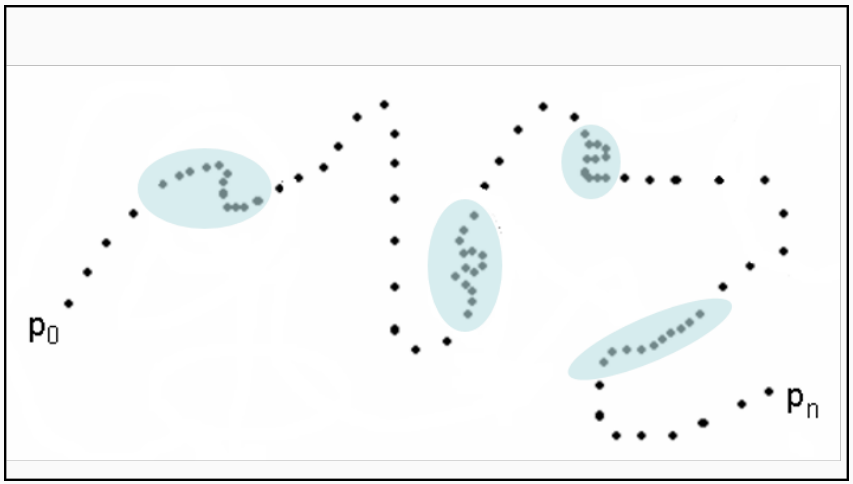
Semantic-based Methods

- A stop of a trajectory T with respect to an application A is a tuple (R_{C_k}, t_j, t_{j+n}), such that T intersects R_{C_k} at all time slots between t_j and t_{j+n} and $|t_{j+n} \check{} t_j| \geq \Delta_{C_k} Ck$
- A move of T with respect to A is
  - a maximal contiguous subtrajectory of T, i.e.,
    - between the starting point and the first stop of T OR
    - between two consecutive stops of T OR
    - between the last stop of T and the ending point of T
  - or the trajectory T itself, if T has no stops



Cluster-based-SMoT (Palma 2008)

- Cluster based: cluster trajectories based on speed
- “Low speed” is assumed to mean “important place”
- Algorithm similar to SMoT but clusters trajectory points first and adds semantics to clusters



Further Readings

- Huiping Cao, Nikos Mamoulis, David W. Cheung: Discovery of Periodic Patterns in Spatiotemporal Sequences. IEEE Trans. Knowl. Data Eng. 19(4): 453-467 (2007)
- Panos Kalnis, Nikos Mamoulis, Spiridon Bakiras: On Discovering Moving Clusters in Spatio-temporal Data. SSTD, 364-381 (2005)
- Florian Verhein, Sanjay Chawla: Mining spatio-temporal patterns in object mobility databases. Data Min. Knowl. Discov. 16(1): 5-38 (2008)
- Florian Verhein, Sanjay Chawla: Mining Spatio-temporal Association Rules, Sources, Sinks, Stationary Regions and Thoroughfares in Object Mobility Databases. DASFAA, 187-201 (2006)
- Changqing Zhou, Dan Frankowski, Pamela J. Ludford, Shashi Shekhar, and Loren G. Terveen. Discovering personally meaningful places: An interactive clustering approach. ACM Trans. Inf. Syst., 25(3), 2007.
- Cao, H., Mamoulis, N., and Cheung, D. W. (2005). Mining frequent spatio-temporal sequential patterns. In: Proceedings of the Fifth IEEE International Conference on Data Mining (ICDM), pages 82–89, Washington, DC, USA, 2005.

Further Readings

- Laube, P. and Imfeld, S. (2002). Analyzing relative motion within groups of trackable moving point objects. In Egenhofer, M. J. and Mark, D. M., editors, GIScience, volume 2478 of Lecture Notes in Computer Science, pages 132–144. Springer.
- Laube, P., Imfeld, S., and Weibel, R. (2005a). Discovering relative motion patterns in groups of moving point objects. International Journal of Geographical Information Science, 19(6):639–668.
- Laube, P., van Kreveld, M., and Imfeld, S. (2005b). Finding REMO: Detecting Relative Motion Patterns in Geospatial Lifelines. Springer.
- Lee, J.-G., Han, J., and Whang, K.-Y. (2007). Trajectory clustering: a partition-and-group framework. In Chan, C. Y., Ooi, B. C., and Zhou, A., editors, SIGMOD Conference, pages 593–604. ACM Press.
- Li, Y., Han, J., and Yang, J. (2004). Clustering moving objects. In: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining (SIGKDD), pages 617–622, New York, NY, USA, 2004. ACM Press.
- Nanni, M. and Pedreschi, D. (2006). Time-focused clustering of trajectories of moving objects. Journal of Intelligent Information Systems, 27(3):267–289.

Further Readings

- Gudmundsson, J. and van Kreveld, M. J. (2006). Computing longest duration flocks in trajectory data. In [de By and Nittel 2006], pages 35–42.
- Gudmundsson, J., van Kreveld, M. J., and Speckmann, B. (2007). Efficient detection of patterns in 2d trajectories of moving points. GeoInformatica, 11(2):195–215.
- Hwang, S.-Y., Liu, Y.-H., Chiu, J.-K., and Lim, E.-P. (2005). Mining mobile group patterns: A trajectory-based approach. In Ho, T. B., Cheung, D. W.-L., and Liu, H., editors, PAKDD, volume 3518 of Lecture Notes in Computer Science, pages 713–718. Springer.
- Cao, H., Mamoulis, N., and Cheung, D. W. (2006). Discovery of collocation episodes in spatiotemporal data. In ICDM, pages 823–827. IEEE Computer Society.
- Bogorny, V. ; Wachowicz, M. A Framework for Context-Aware Trajectory Data Mining. In: Longbing Cao, Philip S. Yu, Chengqi Ahang, Huaifeng Zhang. (Org.). Domain Driven Data Mining: Domain Problems and Applications. 1 ed. : Springer, 2008a.
- Bogorny, V., Kuijpers, B., and Alvares, L. O. (2008b). St-dmql: a semantic trajectory data mining query language. International Journal of Geographical Information Science. Taylor and Francis, 2008.

Further Readings

- Palma, A. T; Bogorny, V.; Kuijpers, B.; Alvares, L.O. A Clustering-based Approach for Discovering Interesting Places in Trajectories. In: 23rd Annual Symposium on Applied Computing, (ACM-SAC'08), Fortaleza, Ceara, 16-20 March (2008) Brazil. pp. 863-868.
- Spaccapietra, S., Parent, C., Damiani, M. L., de Macedo, J. A., Porto, F., and Vangenot, C. (2008). A conceptual view on trajectories. Data and Knowledge Engineering, 65(1):126–146.
- Alvares, L. O., Bogorny, V., de Macedo, J. F., and Moelans, B. (2007a). Dynamic modeling of trajectory patterns using data mining and reverse engineering. In Twenty-Sixth International Conference on Conceptual Modeling - ER2007 - Tutorials, Posters, Panels and Industrial Contributions, volume 83, pages 149–154. CRPIT.
- Alvares, L. O., Bogorny, V., Kuijpers, B., de Macedo, J. A. F., Moelans, B., and Vaisman, A. (2007b). A model for enriching trajectories with semantic geographical information. In ACM-GIS, pages 162–169, New York, NY, USA. ACM Press.