

Knowledge Discovery in Databases II

Summer Term 2018

Lecture 4: Sequence Data

Lectures : Prof. Dr. Peer Kröger, Yifeng Lu

Tutorials: Yifeng Lu

Script © 2015, 2017 Eirini Ntoutsis, Matthias Schubert, Arthur Zimek, Peer Kröger, Yifeng Lu

http://www.dbs.ifi.lmu.de/cms/studium_lehre/lehre_master/kdd218

1. Introduction
2. Sequence Data
3. Time Series Data
4. Spatio-temporal 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
 - ...

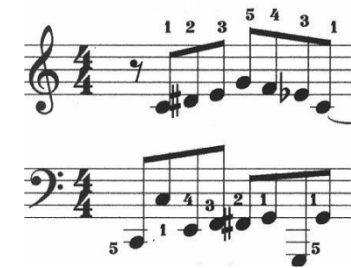
- **Sequence Data** allow for measuring/monitoring phenomena over time (**Time Series Data**) or – more generally – in a given order (of sequential events) without a concrete notion of time
- Examples:
 - Sequence Data: Sequence of purchases
 - Sequential Pattern: Customers buying A are likely to buy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

1. Introduction
2. Sequence Data
3. Time Series Data
4. Spatio-temporal Data

- A sequence S of length n is a mapping of the index set $I_n = \{1, 2, \dots, n\}$ 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)

- Examples:

- Text data $\{a, \dots, Z, 0, \dots 9, \dots\}^*$
- Video data *images**
- Music data *notes**
- Protein sequences *amino_acid** = {LEU, ARG, ... }*
- Gene sequences *nucleic_acid** = {C, G, A, T}*
- ...



- Time series are of course special types of 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 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:

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

- Example:

```

Q = t h r e e
   | x x | x
S = t r e e .
  
```

| = match, x = mismatch

→ $D_{Hamming}(Q, S) = 3$

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):

$Q = T \ \ddot{U} \ R \ S \ C \ H \ L \ O \ S \ S$

$S = T \ O \ R \ S \ C \ H \ U \ S \ S \ .$

$$\Rightarrow D_{Hamming}(Q, S) = 4$$

$Q = T \ \ddot{U} \ R \ S \ C \ H \ L \ O \ S \ S$

$R = A \ B \ S \ C \ H \ U \ S \ S \ . \ .$

$$\Rightarrow D_{Hamming}(Q, R) = 10$$

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

- **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 unmodified ($|$):

Q	=	T	\ddot{U}	R	S	C	H	L	O	S	S
		\diamond	$:$	$:$	$ $	$ $	$ $	\diamond	$:$	$ $	$ $
S	=	A	B	S	C	H			U	S	S

– $D_{Edit}(Q, S) = 5$

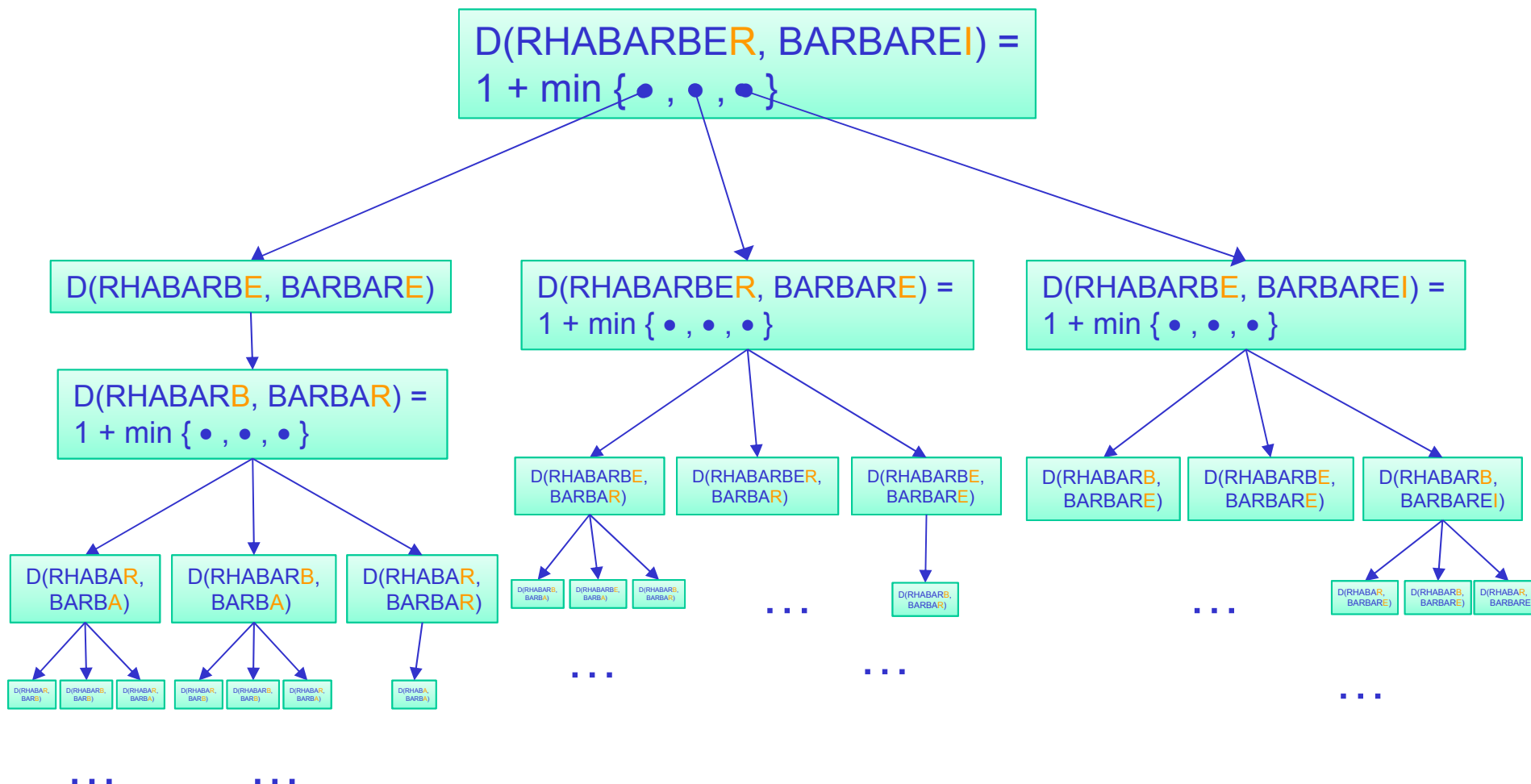
- The mapping between elements is called **optimal alignment** and the Edit Distance represents the **alignment cost**

- 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:

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

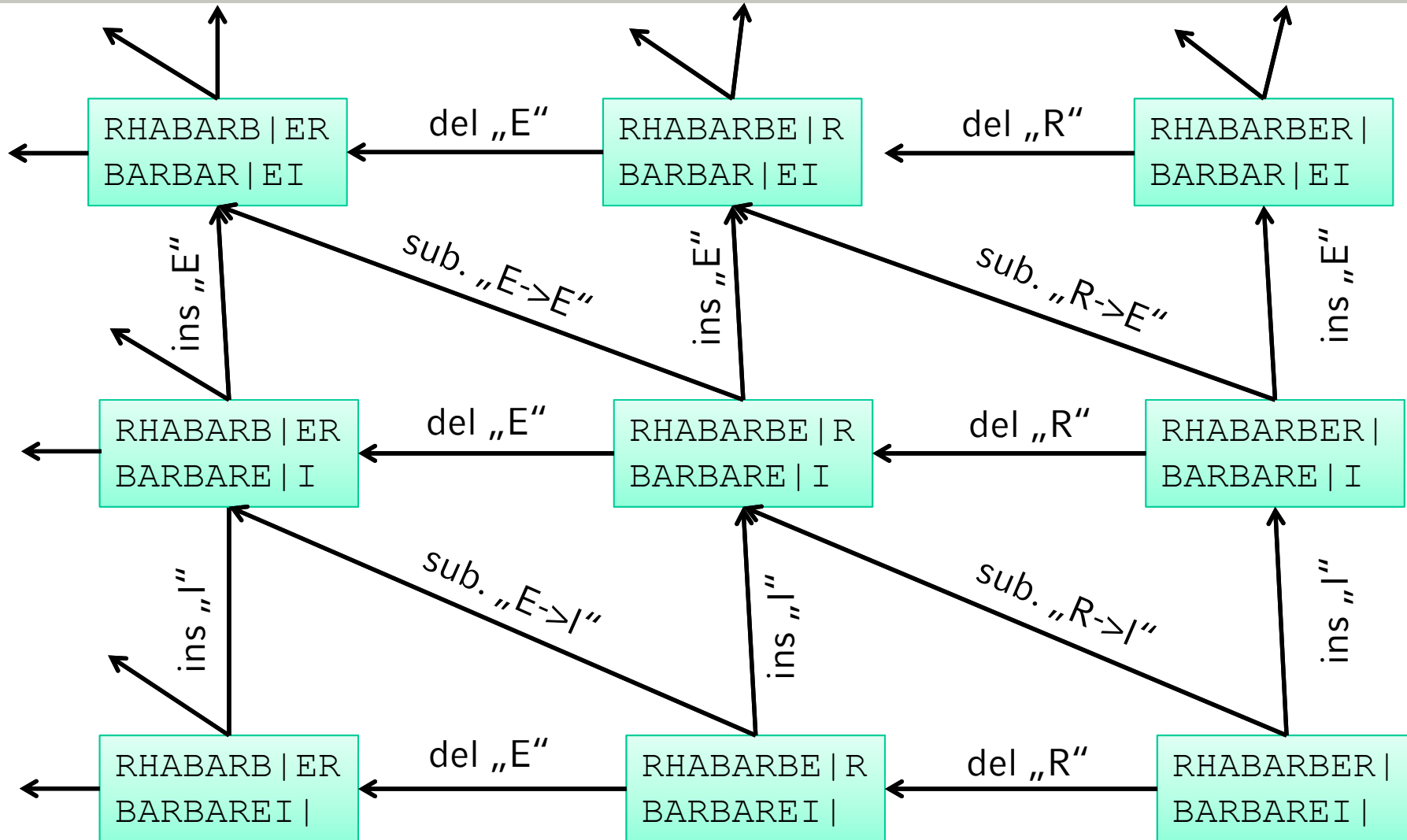
- Remark: if no insertions or deletions occur, the Edit Distance is equivalent to the Hamming Distance

Naïve Computation of the Edit Distance



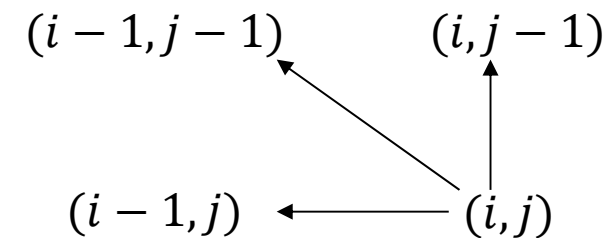
For sequences of lengths n, m , this tree has $\mathcal{O}(3^{n+m})$ nodes

- **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 $3^{1000} \approx 1,322 \cdot 10^{477}$



- 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



- 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)

- 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	4	5	6	7	8	9	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

- **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 δ** between Q and S is defined as:

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

- 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)

- **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 : ACCG**GTCGAGT**GC**GC**GAAGCCGGCCGAA

S : **GTCGTT**CGGAATGCC**GTTGCTCT**GTAA

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:

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

- Properties:**

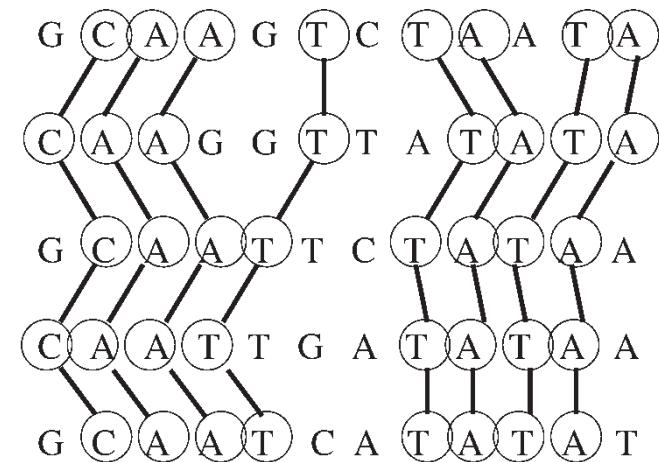
- Computation similar to that of the Edit Distance
- Exponential computation time complexity
- Computation time complexity via dynamic programming lies in $\mathcal{O}(n \cdot m)$

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

- 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 $Q = (q_1, \dots, q_n)$ and $S = (s_1, \dots, s_m)$:

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

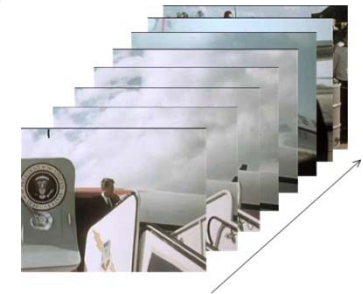
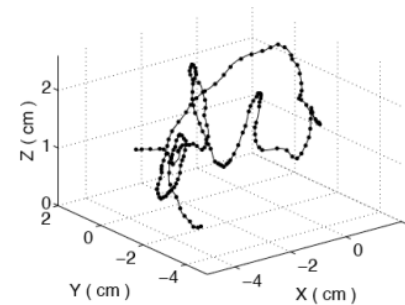
- Generalization of LCSS [S08]:
 - Multiple alignment between several sequences
 - Complexity: $O(2^k n^k)$ for k sequences and n = length of longest sequence



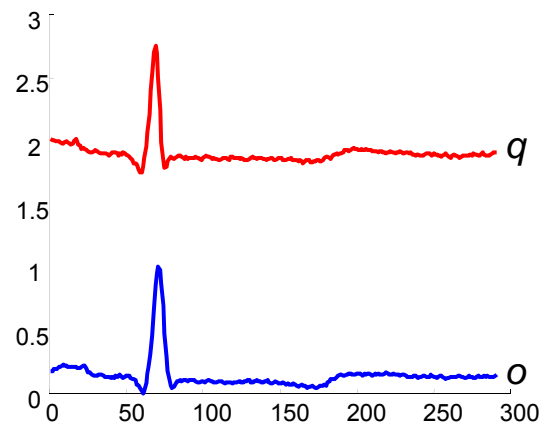
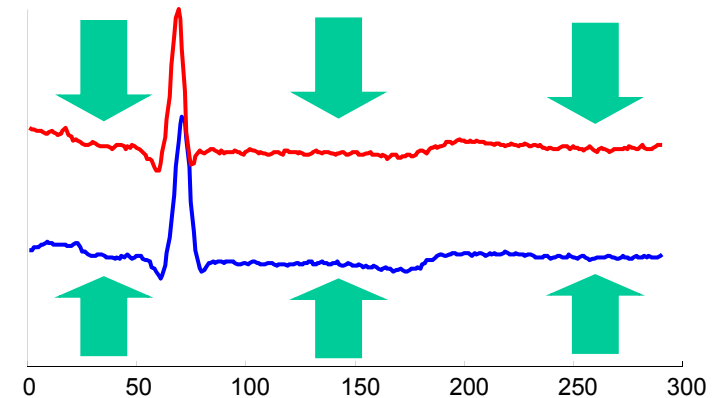
- Distance-based data mining
 - Use one of the similarity measures from above (or variants, or ...)
 - Clustering, outlier detection, classification of sequence data
 - Does not mine sequential patterns but only patterns of similar sequences
- Sequential pattern mining (see previous lecture)
 - Count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

1. Introduction
2. Sequence Data
3. Time Series Data
4. Spatio-temporal Data

- 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

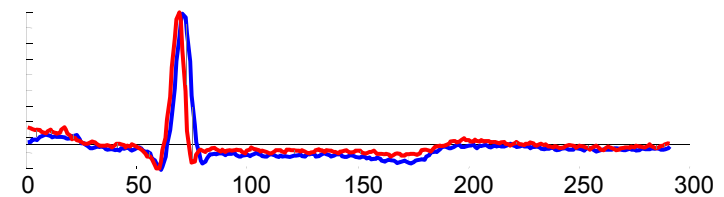


- Data Cleaning to remove artefacts, distortion, noise, ...
 - Offset Translation (aka “Shifting”)
 - Time series are similar but have different offsets
 - Example: move each time series by its mean M

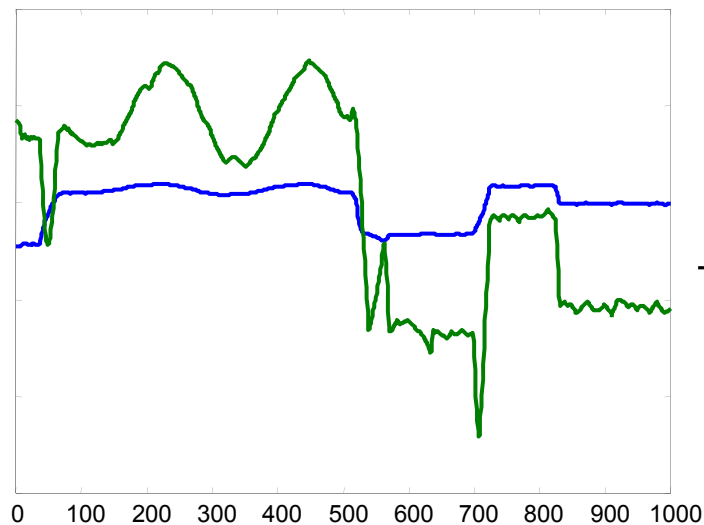


$$q = q - M(q)$$

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

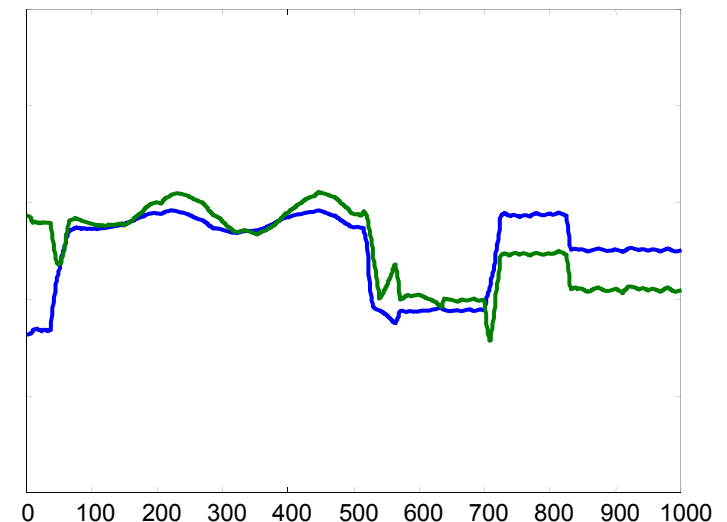


- Data Cleaning (cont.)
 - (Amplitude) Scaling
 - Time series have similar trends but have different amplitudes
 - Example: move each time series by its mean M and normalize the amplitude by its standard deviation S (this is also called “normalization” = shifting + scaling)



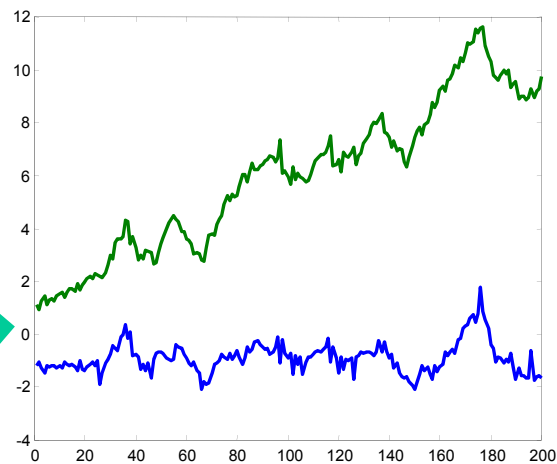
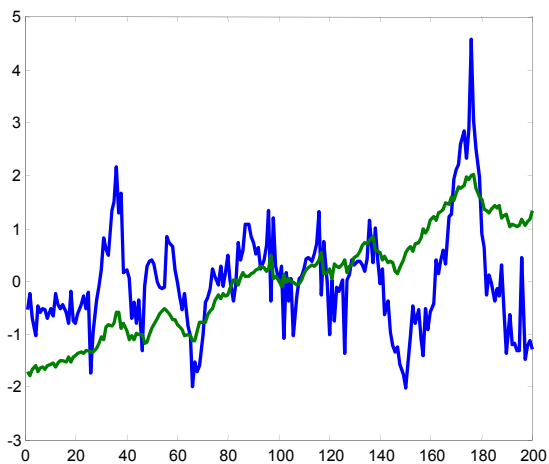
$$q = (q - M(q)) / S(q)$$

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

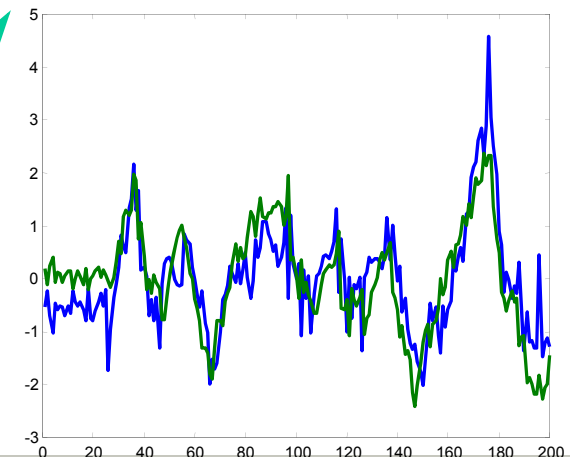


- Data Cleaning (cont.)
 - (Linear) Trend Elimination
 - Similar time series with different trends
 - Determine regression line and move each time series by its regression line
 - Gets complex when an object features more than one trend

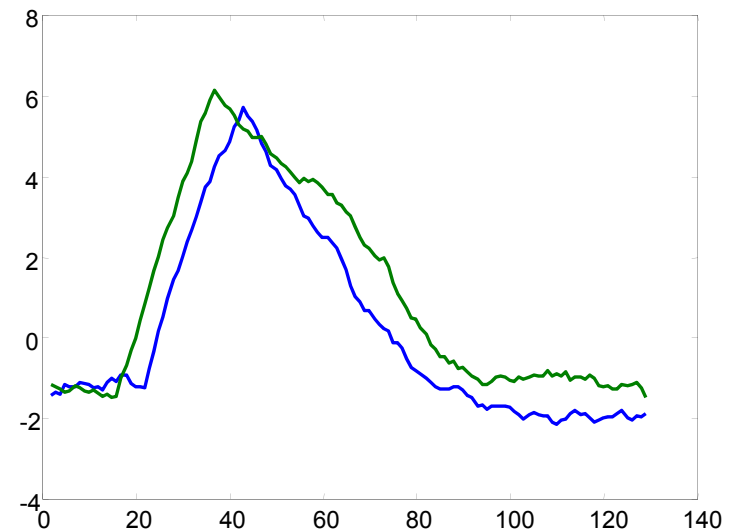
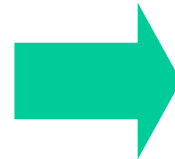
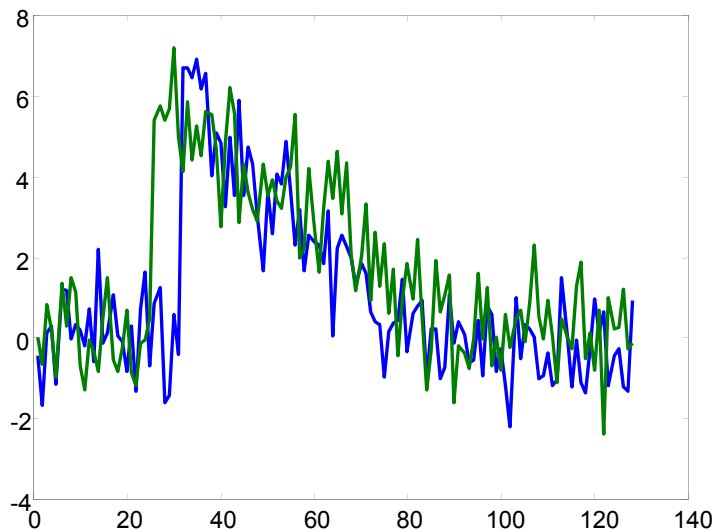
Offset Translation + Amplitude Scaling



Offset Translation + Amplitude Scaling
+ Trend Elimination

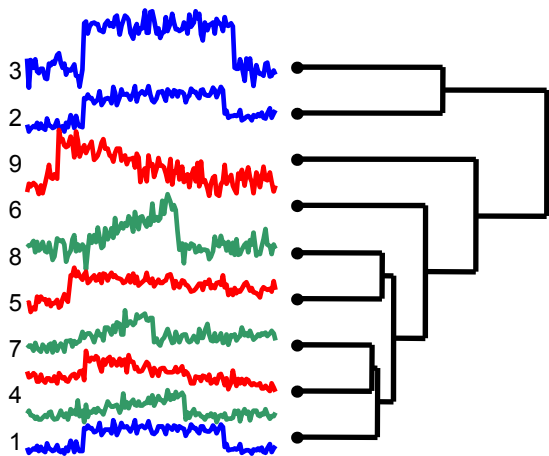


- Data Cleaning (cont.)
 - Noise Reduction
 - Similar time series with large noise portion
 - Smoothing: normalization over a range of values (sliding window), e.g. replace i -th value v_i with mean value of $2k$ adjacent values $[v_{i-k}, \dots, v_i, \dots, v_{i+k}]$

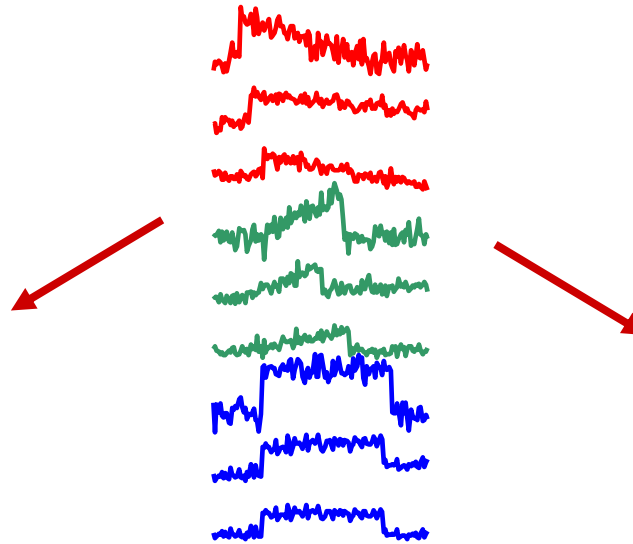


- Data Cleaning: Summary

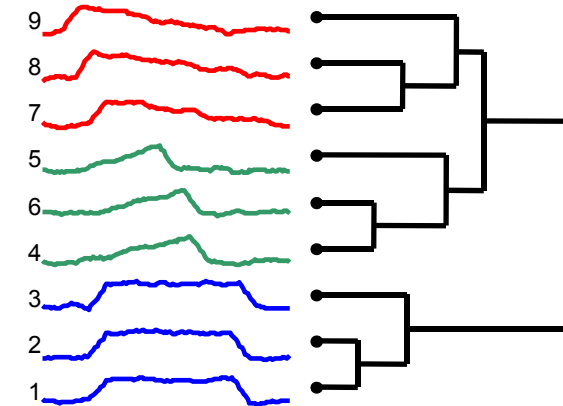
- The above mentioned cleaning procedures are common samples (i.e. there are many more types of distortions that might be of interest to be removed)
- Which cleaning step should be taken? => That heavily depends on the application
- Example:



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



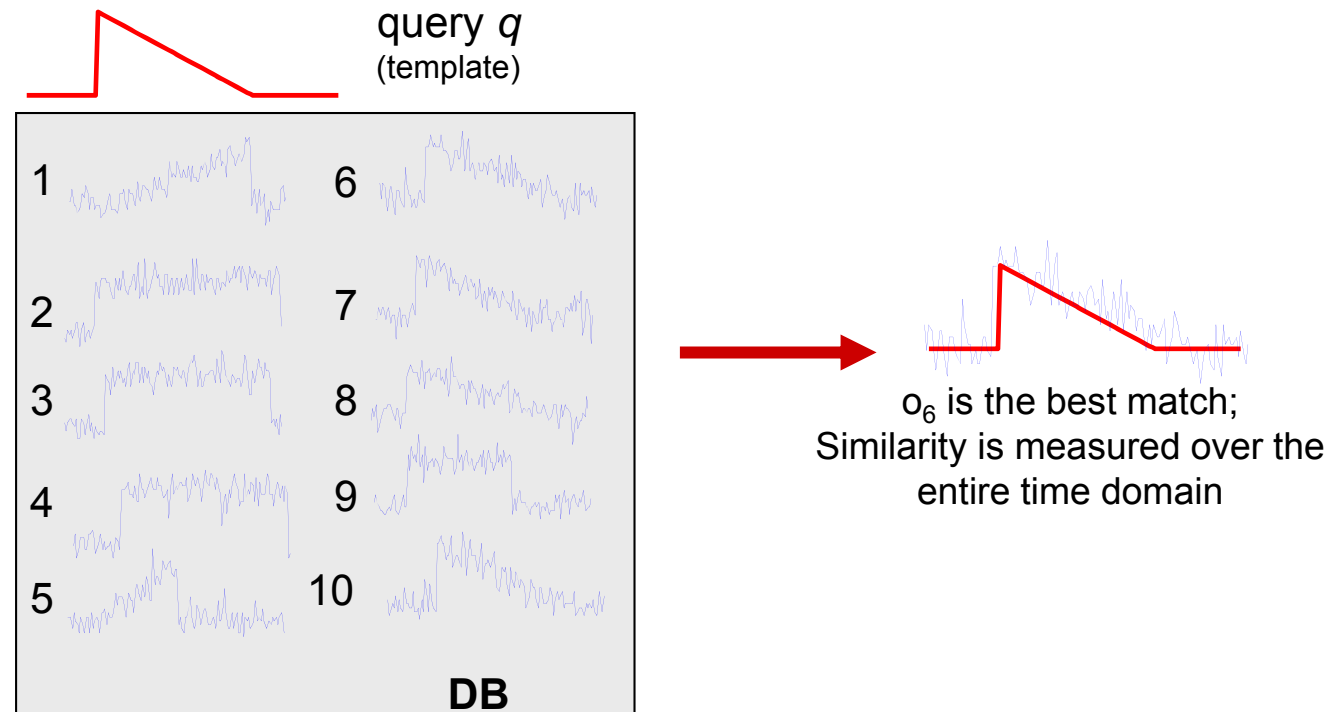
Raw time series from three classes (see color coding)



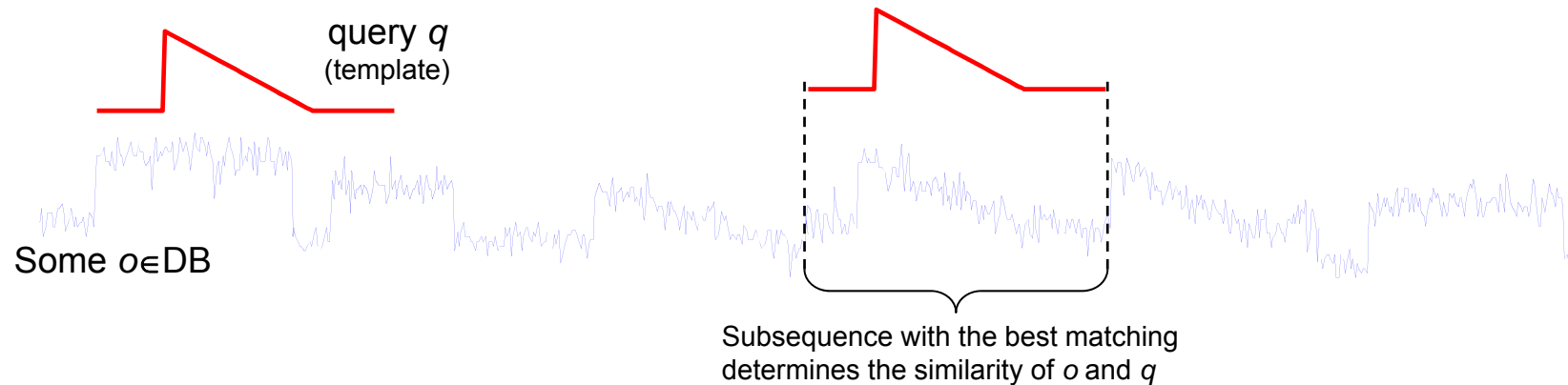
Hierarchical (Single-Link) clustering of raw data using euclidean distance after doing noise reduction, trend elimination, scaling using std. dev. and shifting using mean

- 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

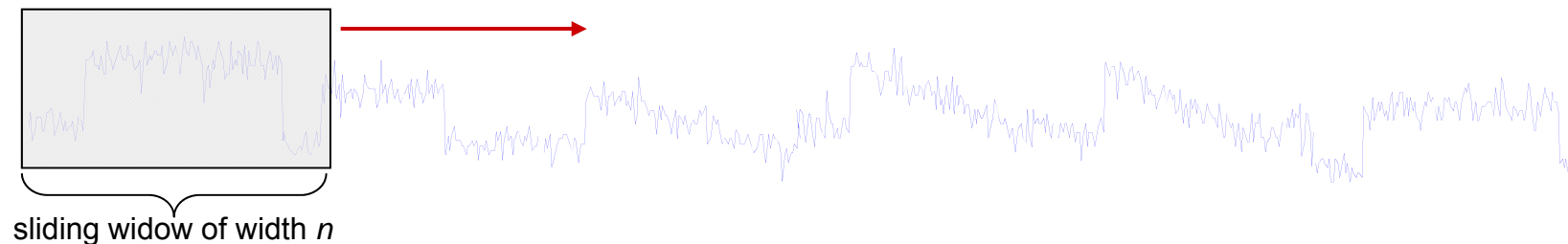
- Illustration with a query template q
 - **Whole matching** of q to a database of time series



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



- 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)



- 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

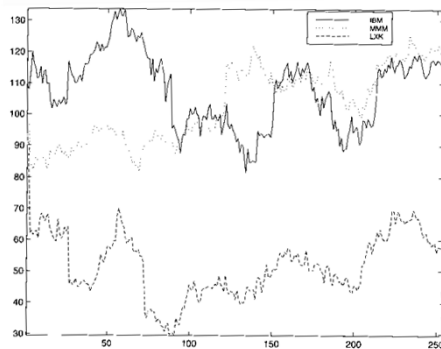
- **Idea:** Representation of a time series $X = (x_1, \dots, x_n)$ as a n-dimensional Euclidean vector
- Given two time series $X = (x_1, \dots, x_n)$ and $Y = (y_1, \dots, y_n)$ of the same length, the Minkowski Distance can be utilized as follows:

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

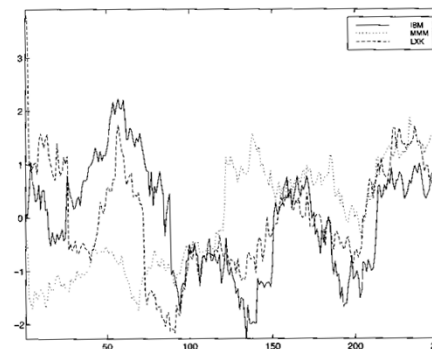
- Properties:
 - $p = 1$: dissimilarities are not emphasized
 - $p = 2$: to be preferred [AFS93]
 - $p = \infty$: distance is attributed to the most dissimilar entries of the time series
- All these variants of the Minkowski Distances are
 - sensitive w.r.t. variations on the time axis
 - are limited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



Normalized stock prices

- 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

$$- \hat{X} = \frac{X - \text{avg}(X)}{\text{std}(X)}$$

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

$$\text{and } \text{std}(X) = \sqrt{\frac{1}{n} \sum_{i=1}^n (x_i - \text{avg}(X))^2}$$

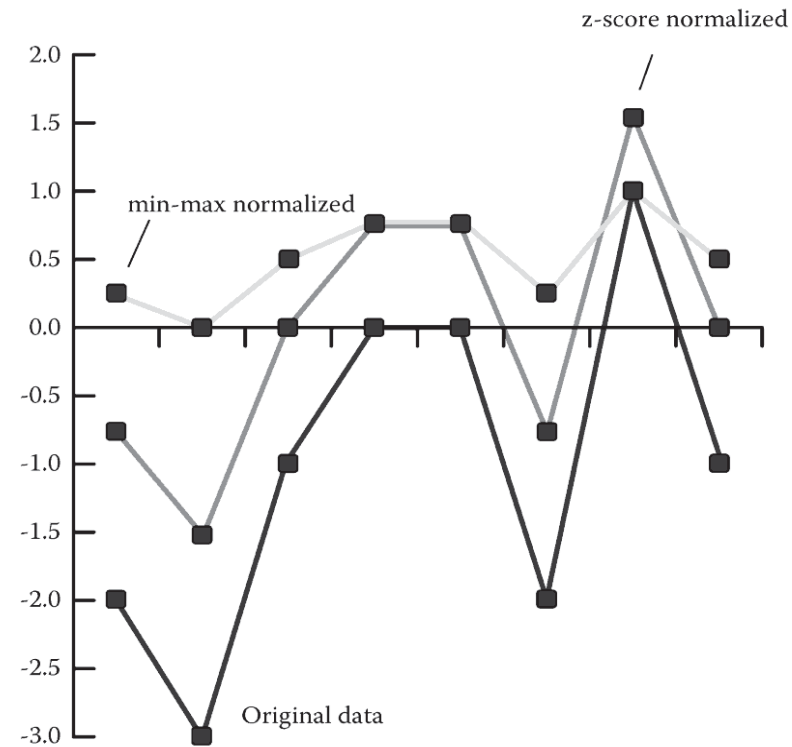
- Alternative: **Min-max normalization** of a time series $X = (x_1, \dots, x_n)$:

$$- \hat{X} = \frac{X - \text{Max}(X)}{\text{Max}(X) - \text{Min}(X)} (\text{newMax} - \text{newMin}) + \text{newMin}$$

- Properties:

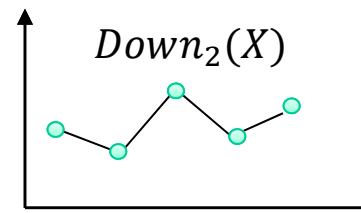
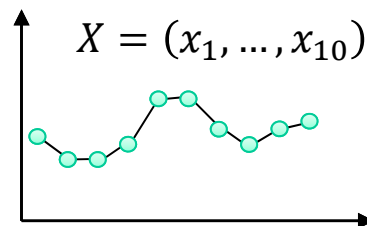
- Z-Score normalization is more robust w.r.t. noise in the data
- Min-max normalization can be dominated by outliers.

- Example of different normalizations [M10]



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

- 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 $X = (x_1, \dots, x_n)$ **along the time axis** as follows:
 - **ω -upsampling:**
 - resolution is increased
 - $Up_\omega(x_1, \dots, x_n) = (z_1, \dots, z_{n\omega})$ with $z_i = x_{\lfloor \frac{i}{\omega} \rfloor}$ and $i = 1 \dots n\omega$
 - every x_i is repeated ω times
 - **ω -downsampling:**
 - resolution is decreased
 - $Down_\omega(x_1, \dots, x_n) = (z_1, \dots, z_{\lfloor \frac{n}{\omega} \rfloor})$ with $z_i = x_{i\omega}$ and $i = 1 \dots \lfloor \frac{n}{\omega} \rfloor$
 - only multiples of ω are used, i.e. $i \cdot \omega$



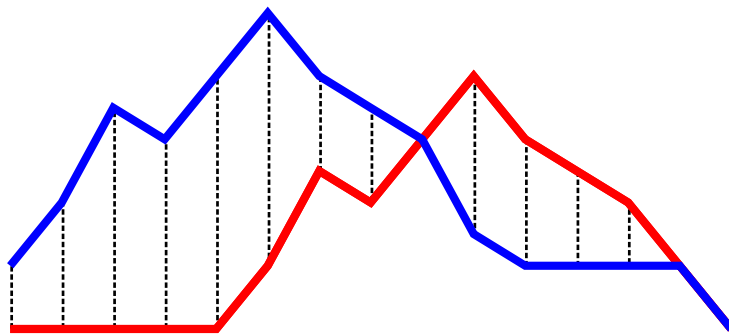
Uniform Time Warping (UTW)

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

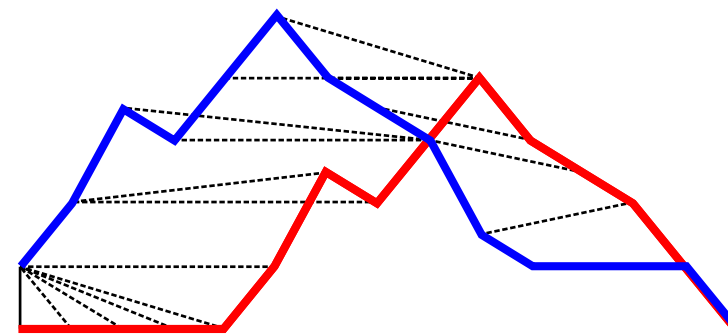
$$\begin{aligned} D_{UTW}^2(X, Y) &= \frac{L_2^2(Up_m(X), Up_n(X))}{m \cdot n} \\ &= \frac{\sum_{i=1}^{m \cdot n} (x_{[i/m]} - y_{[i/n]})^2}{m \cdot n} \end{aligned}$$

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

- **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

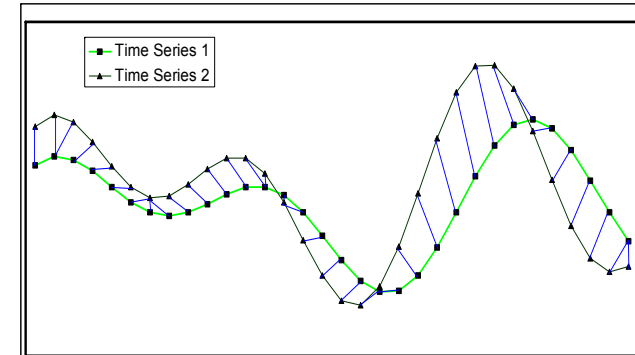


Euclidean Distance



Dynamic Time Warping

- 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:

$$DTW_{\delta,p}(\emptyset, \emptyset) = 0$$

$$DTW_{\delta,p}(X, \emptyset) = DTW_{\delta,p}(\emptyset, Y) = \infty \text{ for } X, Y \neq \emptyset$$

$$DTW_{\delta,p}(X, Y) = \left(\delta>Last(X), Last(Y))^p + \left(\min \left\{ \begin{array}{l} DTW_{\delta,p}(Start(X), Start(Y)) \\ DTW_{\delta,p}(X, Start(Y)) \\ DTW_{\delta,p}(Start(X), Y) \end{array} \right\} \right)^p \right)^{\frac{1}{p}}$$

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

- $p = 1$:

$$DTW_{\delta,1}(X, Y) = \delta(\text{Last}(X), \text{Last}(Y)) + \min \left\{ \begin{array}{l} DTW_{\delta,1}(\text{Start}(X), \text{Start}(Y)) \\ DTW_{\delta,1}(X, \text{Start}(Y)) \\ DTW_{\delta,1}(\text{Start}(X), Y) \end{array} \right\}$$

- $p = 2$ (Euclidean variant):

$$DTW_{\delta,2}(X, Y) = \sqrt{\delta(\text{Last}(X), \text{Last}(Y))^2 + \left(\min \left\{ \begin{array}{l} DTW_{\delta,2}(\text{Start}(X), \text{Start}(Y)) \\ DTW_{\delta,2}(X, \text{Start}(Y)) \\ DTW_{\delta,2}(\text{Start}(X), Y) \end{array} \right\} \right)^2}$$

- $p \rightarrow \infty$:

$$DTW_{\delta,\infty}(X, Y) = \max \left\{ \delta(\text{Last}(X), \text{Last}(Y)), \min \left\{ \begin{array}{l} DTW_{\delta,\infty}(\text{Start}(X), \text{Start}(Y)) \\ DTW_{\delta,\infty}(X, \text{Start}(Y)) \\ DTW_{\delta,\infty}(\text{Start}(X), Y) \end{array} \right\} \right\}$$

- Termination cases are the same as on the previous slide

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

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

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

$$\begin{aligned} DTW^2(X, Y) &= DTW_{L_1, 2}^2(X, Y) \\ &= |Last(X) - Last(Y)|^2 + \min \left\{ \begin{array}{l} DTW^2(Start(X), Start(Y)) \\ DTW^2(X, Start(Y)) \\ DTW^2(Start(X), Y) \end{array} \right\} \end{aligned}$$

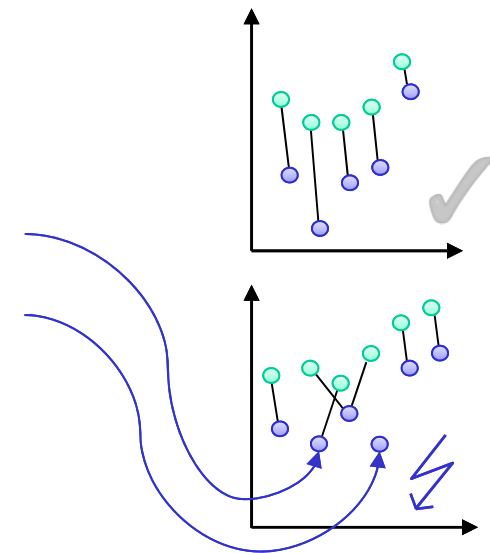
- Dynamic Time Warping aligns two time series to each other
- This element-wise alignment between two time series $X = (x_1, \dots, x_n)$ and $Y = (y_1, \dots, y_m)$ can be expressed by a **warping path** P of indices:

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

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:

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



- 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:

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

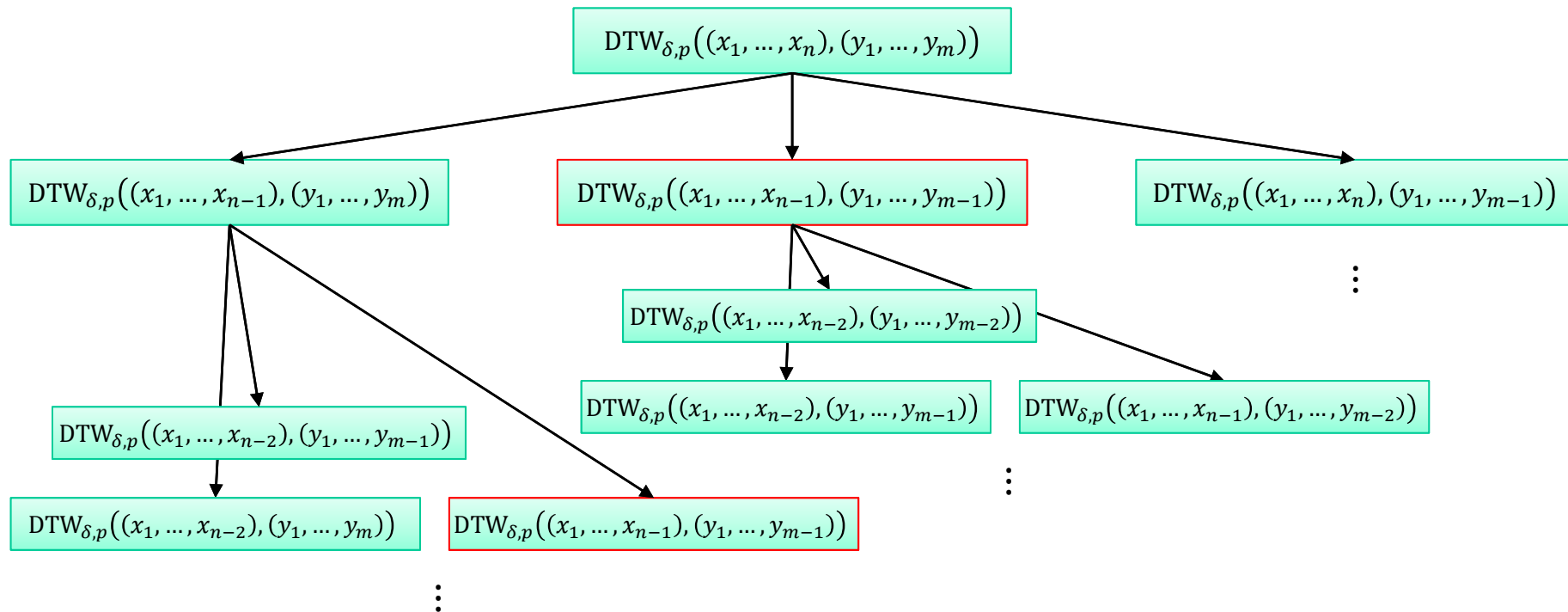
- $\text{DTW}^2(X, Y)$ can be defined by the path with the minimal cost:

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

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

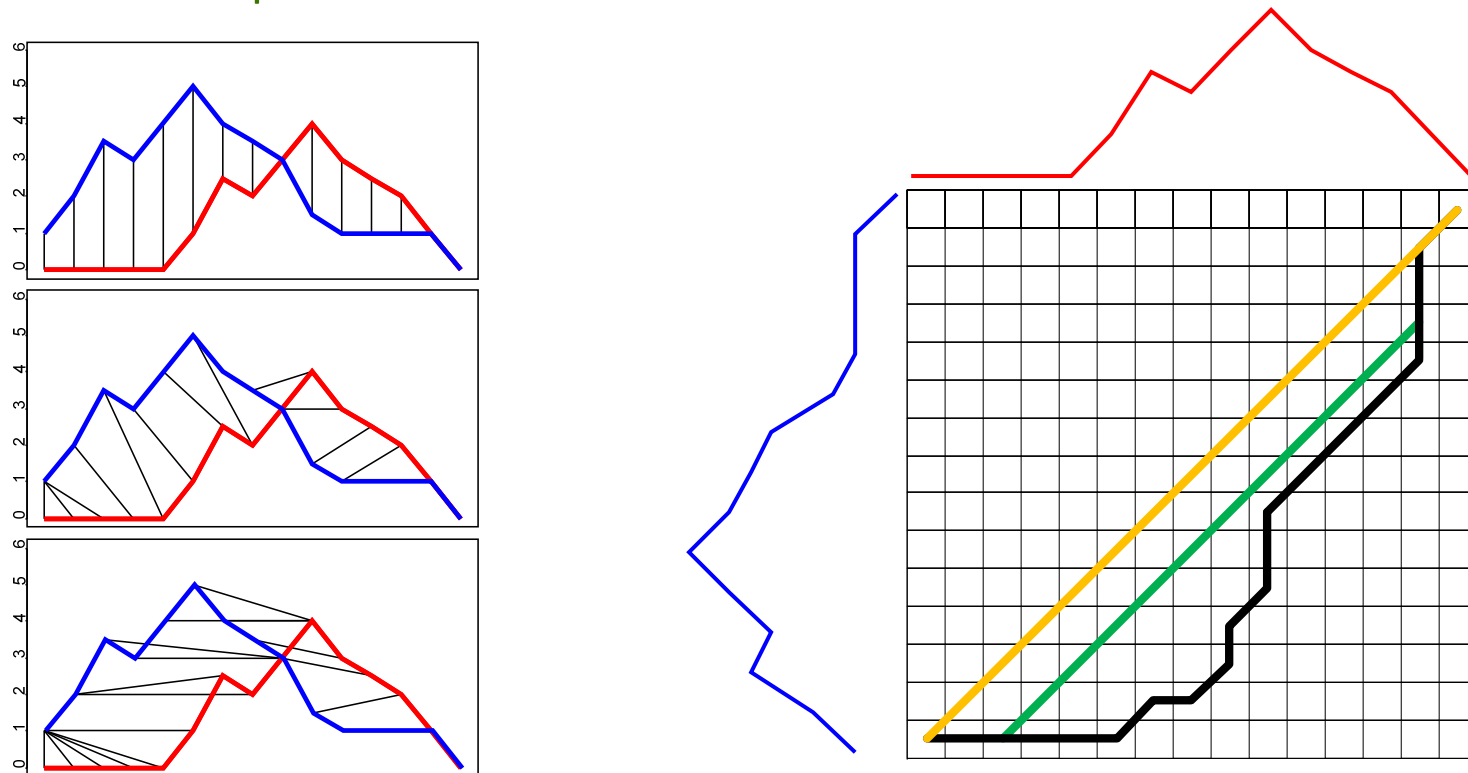
Naïve Computation of Dynamic Time Warping

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



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

- Any path P between two times series $X = (x_1, \dots, x_n)$ and $Y = (y_1, \dots, y_m)$ 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
---	---	---	---

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

– Time series Y :

1	1	1	2
---	---	---	---

- DTW does not satisfy the triangle inequality:

– Time series X :

0	0	0	0
---	---	---	---

– Time series Y :

1	2	2	3
---	---	---	---

– Time series Z :

2	3	3	3
---	---	---	---

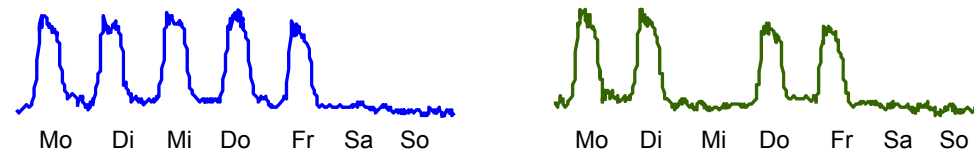
Diagram illustrating DTW alignments and costs:

- Alignment between X and Y : $1+2+2+3=8$
- Alignment between Y and Z : $2+3+3+3=11$
- Alignment between X and Z : $1+0+0+0+0+0=1$

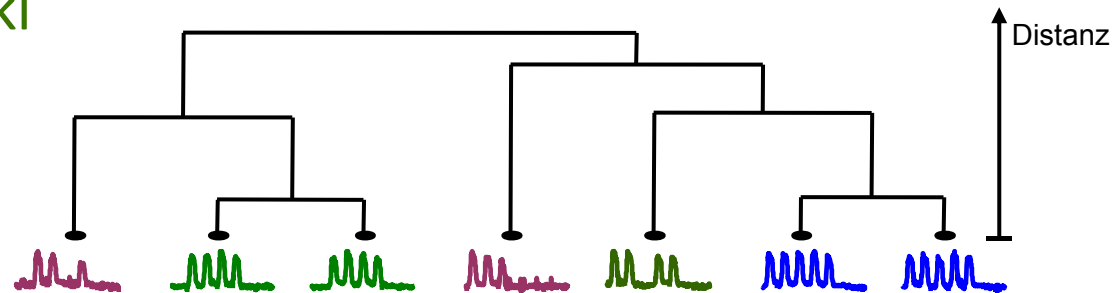
$$\begin{array}{ccccccc}
 DTW(X, Z) & \leq & DTW(X, Y) & + & DTW(Y, Z) & & \\
 11 & \leq & 8 & + & 1 & & \text{⚡}
 \end{array}$$

- Reason: replication of elements

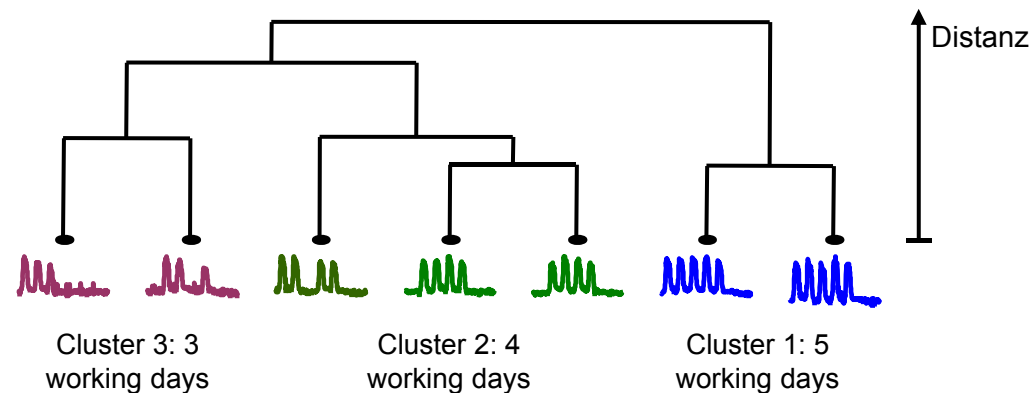
- Comparison on time series that measure the daily network traffic of a company, e.g.:



- Minkowski



- DTW



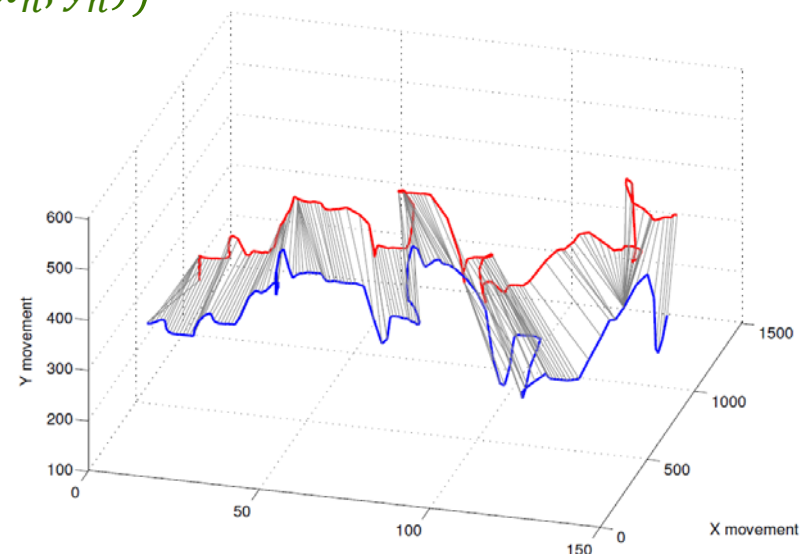
- Up to now: Time series over real numbers
 - Ground distance δ between two elements x_i, y_i of time series X, Y can be chosen as absolute difference:

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

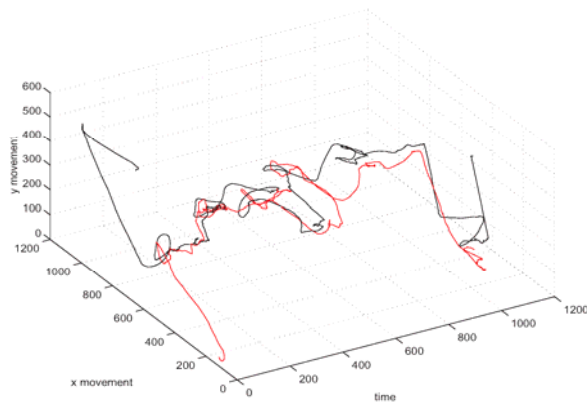
- Application of DTW to trajectories
 - Trajectories are time series over multidimensional objects, e.g.:

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

- Necessary: measurement of temporally ordered points in space
- Different ground distances (L_1, L_2, L_∞) for comparison of (x_i, y_i) and (x_j, y_j)
- Adaptation of DTW to multidimensional time series is straightforward

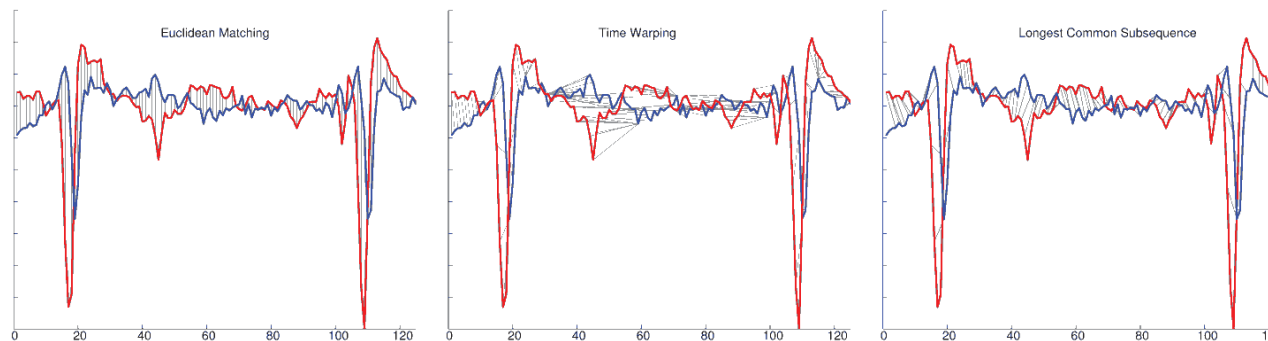


- Dynamic Time Warping is sensitive to outliers and noise
- Solution: extending LCSS to time series
- A measure tolerant to gaps in the two compared time series



Example 1:

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



Example 2:

- Noisy setting where DTW gives many dubious matchings