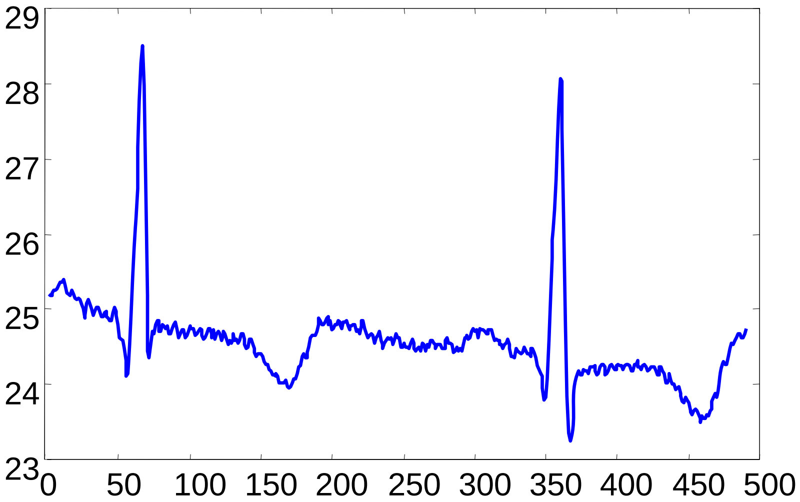
Serie Temporali

Sistemi informativi per le Decisioni

Slide a cura di Prof. Paolo Ciaccia

Time series are everywhere…

- Time series, that is, sequences of observations made through time, are present in everyday's life:
  - Temperature, rainfalls, seismic traces
  - Weblogs
  - Stock prices
  - EEG, ECG, blood pressure
  - Enrolled students at the Engineering Faculty
  - …



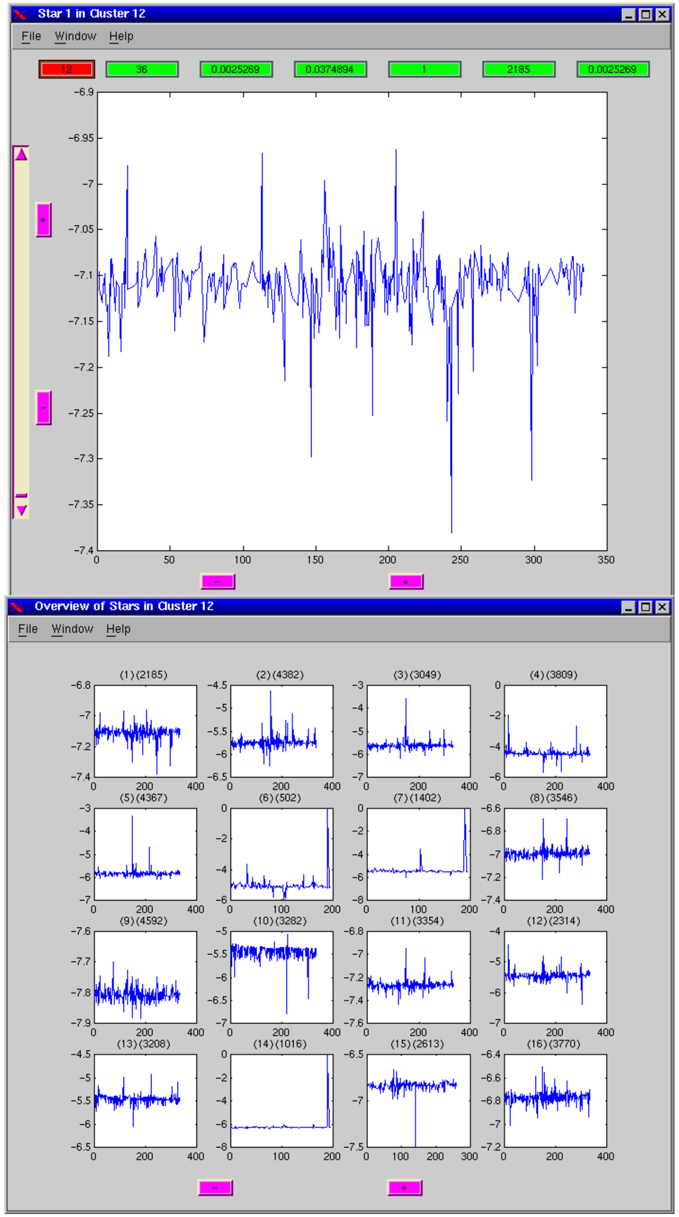
- This as well as many of the following figures/examples are taken from the tutorial given by Eamonn Keogh at SBBD 2002 (XVII Brazilian Symposium on Databases)

www.cs.ucr.edu/~eamonn/

Why is similarity search in t.s.'s important?

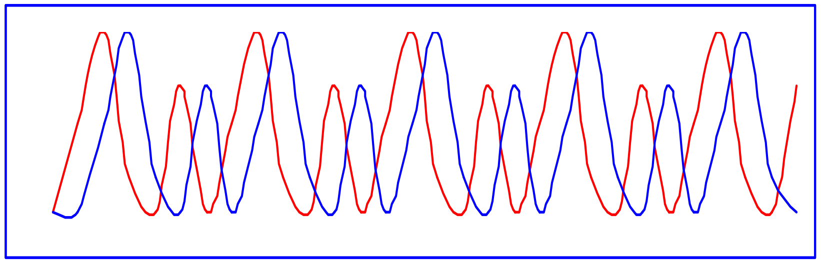
- Consider a large time series DB:
  - 1 hour of ECG data: 1 GByte
  - Typical Weblog: 5 GBytes per week
  - Space Shuttle DB: 158 GBytes
  - MACHO Astronomical DB: 2 TBytes, updated with 3 GBytes a day (20 million stars recorded nightly for 4 years) http://wwwmacho.anu.edu.au/

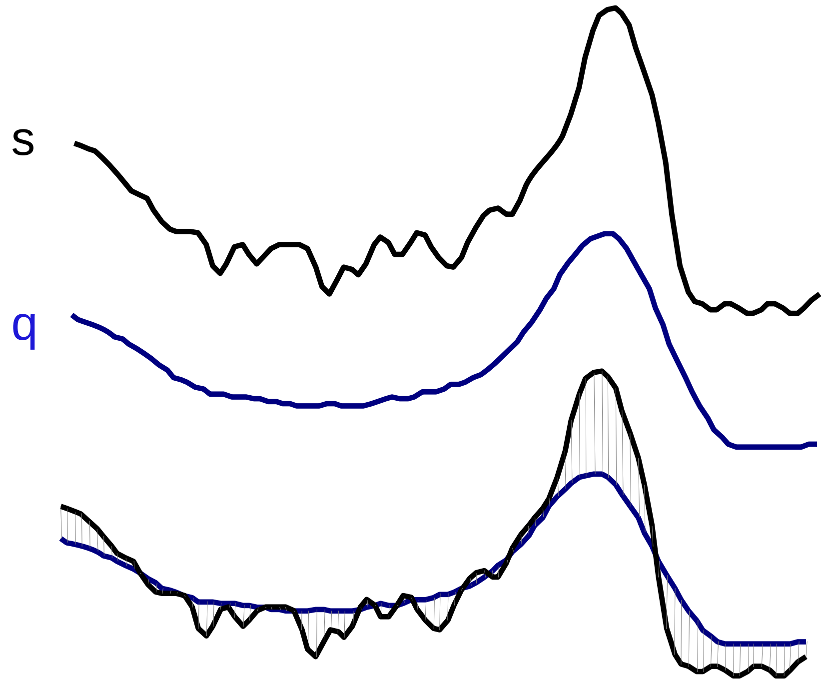
- Similarity search can help you in:
  - Looking for the occurrence of known patterns
  - Discovering unknown patterns
  - Putting "things together" (clustering)
  - Classifying new data
  - Predicting/extrapolating future behaviors
  - …



How to measure similarity

- Given two time series of equal length D, the commonest way to measure their (dis-)similarity is based on Euclidean distance
- However, with Euclidean distance we have to face two basic problems
  - High-dimensionality: (very) large D values
  - Sensitivity to “alignment of values”





- For problem 1. we need to define effective lower-bounding techniques that work in a (much) lower dimensional space
- For problem 2. we will introduce a new similarity criterion

$$L_2(s,q) = \sqrt{\sum_{t=0}^{D-1} (s_t - q_t)^2}$$

Dimensionality reduction: DFT (i)

- The first approach to reducing the dimensionality of time series, proposed in [AFS93], was based on Discrete Fourier Transform (DFT)
- Remind: given a time series s, the Fourier coefficients are complex numbers (amplitude,phase), defined as:

$$S_f = \frac{1}{\sqrt{D}} \sum_{t=0}^{D-1} s_t \exp(-j2\pi f t/D) \qquad f = 0,\ldots,D-1$$

- From Parseval theorem we know that DFT preserves the energy of the signal:

$$E(s) = \sum_{t=0}^{D-1} s_t^2 = E(S) = \sum_{f=0}^{D-1} |S_f|^2$$

- Since DFT is a linear transformation we have:

$$L_2(s,q)^2 = \sum_{t=0}^{D-1} (s_t - q_t)^2 = E(s-q) = E(S-Q) = \sum_{f=0}^{D-1} |S_f - Q_f|^2 = L_2(S,Q)^2$$

thus, DFT preserves the Euclidean distance

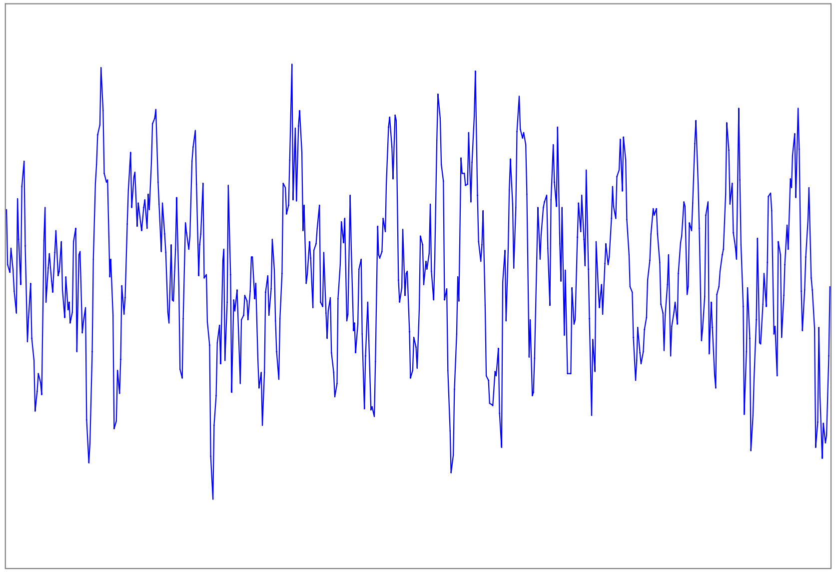
- And? What can we gain from such transformation??

Dimensionality reduction: DFT (ii)

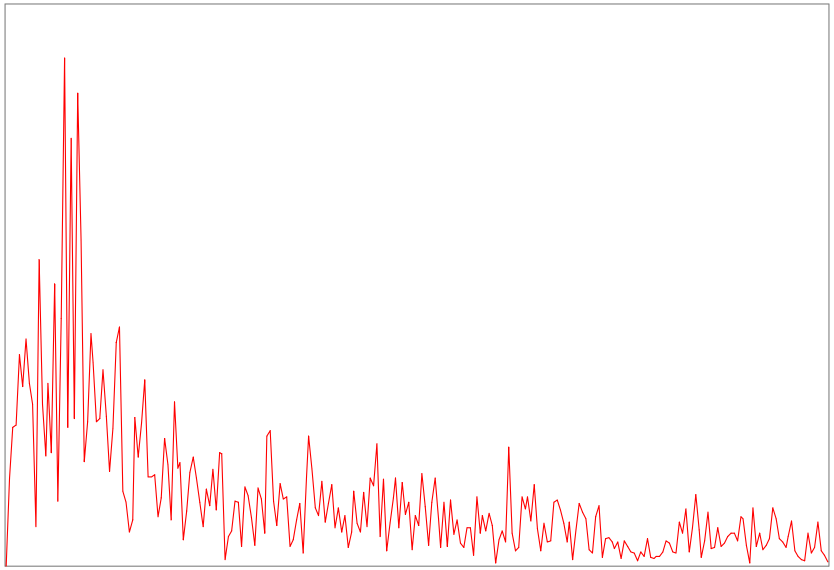
- The key observation is that, by keeping only a small set of Fourier coefficients, we can obtain a good approximation of the original signal
- Why: because most of the energy of many real-world signals concentrates in the low frequencies ([AFS93]):
- More precisely, the energy spectrum ($|S_f|^2$ vs. f) behaves as $O(f^{-b})$, b > 0:
  - b = 2 (random walk or brown noise): used to model the behavior of stock movements and currency exchange rates
  - b > 2 (black noise): suitable to model slowly varying natural phenomena (e.g., water levels of rivers)
  - b = 1 (pink noise): according to Birkhoff's theory, musical scores follow this energy pattern
- Thus, if we only keep the first few coefficients (D' << D) we can achieve an effective dimensionality reduction
  - Note: this is the basic idea used by well-known compression standards, such as JPEG (which is based on Discrete Cosine Transform)

An example: EEG data

- Sampling rate: 128 Hz



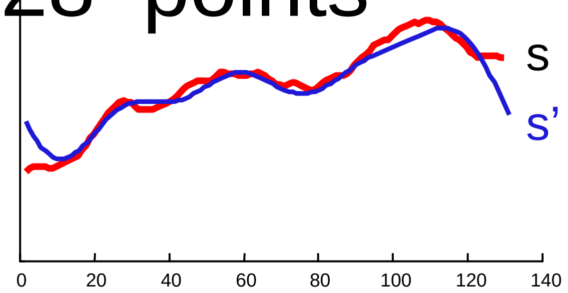
Time series (4 secs, 512 points)

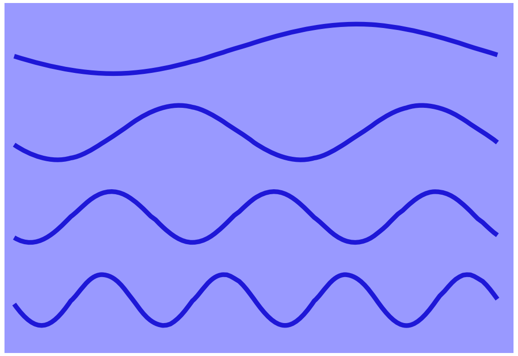


Energy spectrum

Another example

- 128 points





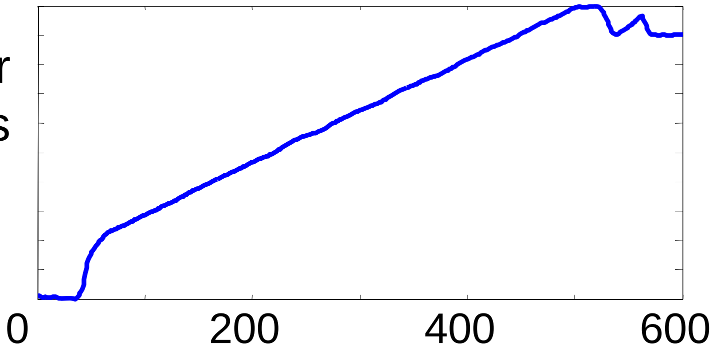
s' = approximation of s with 4 Fourier coefficients

data values	Fourier coefficients	First 4 Fourier coefficients
0.4995	1.5698	**1.5698**
0.5264	1.0485	**1.0485**
0.5523	0.7160	**0.7160**
0.5761	0.8406	**0.8406**
0.5973	0.3709	**0.3709**
0.6153	0.4670	**0.4670**
0.6301	0.2667	**0.2667**
0.6420	0.1928	**0.1928**
0.6515	0.1635	
0.6596	0.1602	
0.6672	0.0992	
0.6751	0.1282	
0.6843	0.1438	
0.6954	0.1416	
0.7086	0.1400	
0.7240	0.1412	
0.7412	0.1530	
0.7595	0.0795	
0.7780	0.1013	
0.7956	0.1150	
0.8115	0.1801	
0.8247	0.1082	
0.8345	0.0812	
0.8407	0.0347	
0.8431	0.0052	
0.8423	0.0017	
0.8387	0.0002	
…	...	

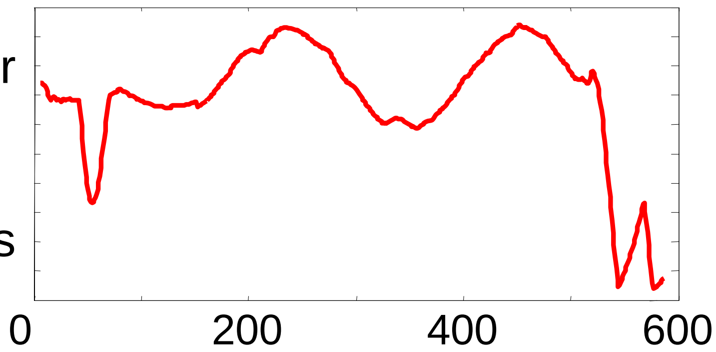
Comments on DFT

- ☺ Can be computed in O(DlogD) time using FFT (provided D is a power of 2)
- ☹ Difficult to use if one wants to deal with sequences of different length
- ☹ Not really amenable to deal with "signals with spots" (time-varying energy)
- An alternative to DFT is to use *wavelets*, which takes a different perspective:
  - A signal can be represented as a sum of contributions, each at a different resolution level
  - Discrete Wavelet Transform (DWT) can be computed in O(D) time
- Experimental results however show that the superiority of DWT w.r.t. DFT is dependent on the specific dataset

Good for wavelets bad for Fourier



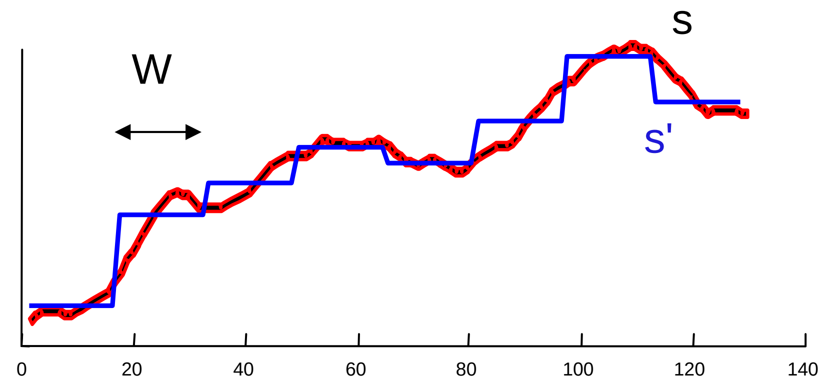
Good for Fourier bad for wavelets



Dimensionality reduction: PAA

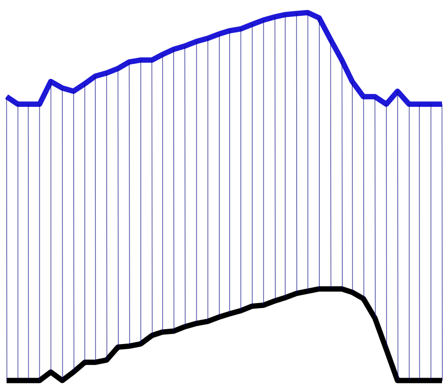
- PAA (Piecewise Aggregate Approximation) [KCP+00,YF00] is a very simple, intuitive and fast (O(D)) method to approximate time series
  - Its performance is comparable to that of DFT and DWT
- We take a window of size W and segment our time series into D' = D/W "pieces" (sub-sequences), each of size W
- For each piece, we compute the average of values, i.e.
- Our approximation is therefore $s' = (s'_1,\ldots,s'_{D'})$
- We have $\sqrt{W} \times L2(s',q') \leq L2(s,q)$
  (arguments generalize those used for the "global average" example)
  - The same can be generalized to work with arbitrary Lp-norms [YF00]

$$s'_i = \frac{\sum_{t=(i-1)\times W}^{i\times W-1} s_t}{W}$$



The “alignment” problem

- Euclidean distance, as well as other Lp-norms, are not robust w.r.t., even small, contractions/expansions of the signal along the time axis
  - E.g., speech signals
- Intuitively, we would need a distance measure that is able to “match” a point of time series s even with “surrounding” points of time series q
  - Alternatively, we may view the time axis as a “stretchable” one
- A distance like this exists, and is called “Dynamic Time Warping” (DTW)!



Fixed Time Axis
Sequences are aligned “one to one”

“Warped” Time Axis
Non-linear alignments are possible

How to compute the DTW (i)

- Assume that the two time series s and q have the same length D
  - Note that with DTW this is not necessary anymore!
- Construct a D×D matrix d, whose element $d_{i,j}$ is the distance between s_i and q_j
  - We take $d_{i,j} = (s_i - q_j)^2$, but other possibilities exist (e.g., $|s_i – q_j|$)

D=6	0	1	2	3	4	5
s	1	2	5	4	3	7
q	2	3	2	1	3	4

$L_2(s,q) = \sqrt{29}$

s						
7	25	16	25	36	16	9
3	1	0	1	4	0	1
4	4	1	4	9	1	0
5	9	4	9	16	4	1
2	0	1	0	1	1	4
1	1	4	1	0	4	9
d	**2**	**3**	**2**	**1**	**3**	**4**

q

- The "rules of the game":
  - Start from (0,0) and end in (D-1,D-1)
  - Take one step at a time
  - At each step, move only by increasing i, j, or both
    - i.e., never go back!
  - "Jumps" are not allowed!
  - Sum all distances you have found in the "warping path"

How to compute the DTW (ii)

- The figure shows a possible warping path w, whose “cost” is 21
  - The “Euclidean path” moves only along the main diagonal, and costs 29

s						
7	25	16	25	36	16	**9**
3	1	0	**1**	**4**	**0**	1
4	4	**1**	4	9	1	0
5	9	**4**	9	16	4	1
2	0	**1**	0	1	1	4
1	**1**	4	1	0	4	9
	2	**3**	**2**	**1**	**3**	**4**

warping path w

q

The DTW is the minimum cost among all the warping paths

- But the number of path is exponential in D ☹
- Ok, but we can use dynamic programming, with complexity $O(D^2)$ ☺

How to compute the DTW (iii)

- From the d matrix, incrementally build a new matrix WP, whose elements $wp_{i,j}$ are recursively defined as:

$$wp_{i,j} = d_{i,j} + \min\{wp_{i-1,j}, wp_{i,j-1}, wp_{i-1,j-1}\}$$

s ↑ (rows), q → (columns)

7	25	16	25	36	16	9
3	1	0	1	4	0	1
4	4	1	4	9	1	0
5	9	4	9	16	4	1
2	0	1	0	1	1	4
1	1	4	1	0	4	9
d	**2**	**3**	**2**	**1**	**3**	**4**

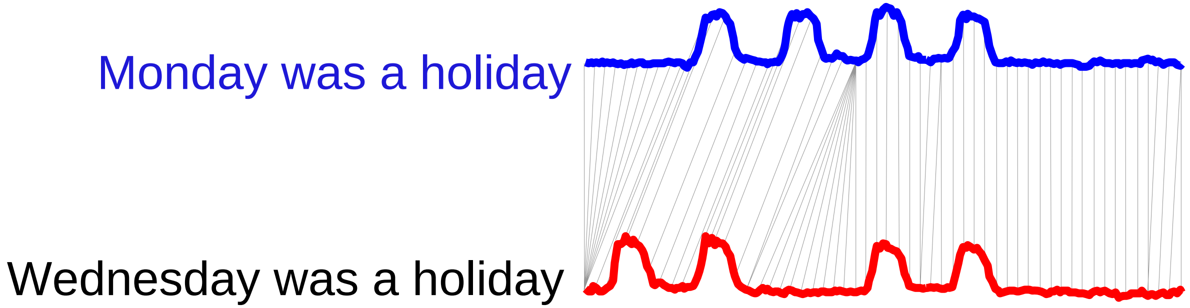
s ↑ (rows), q → (columns)

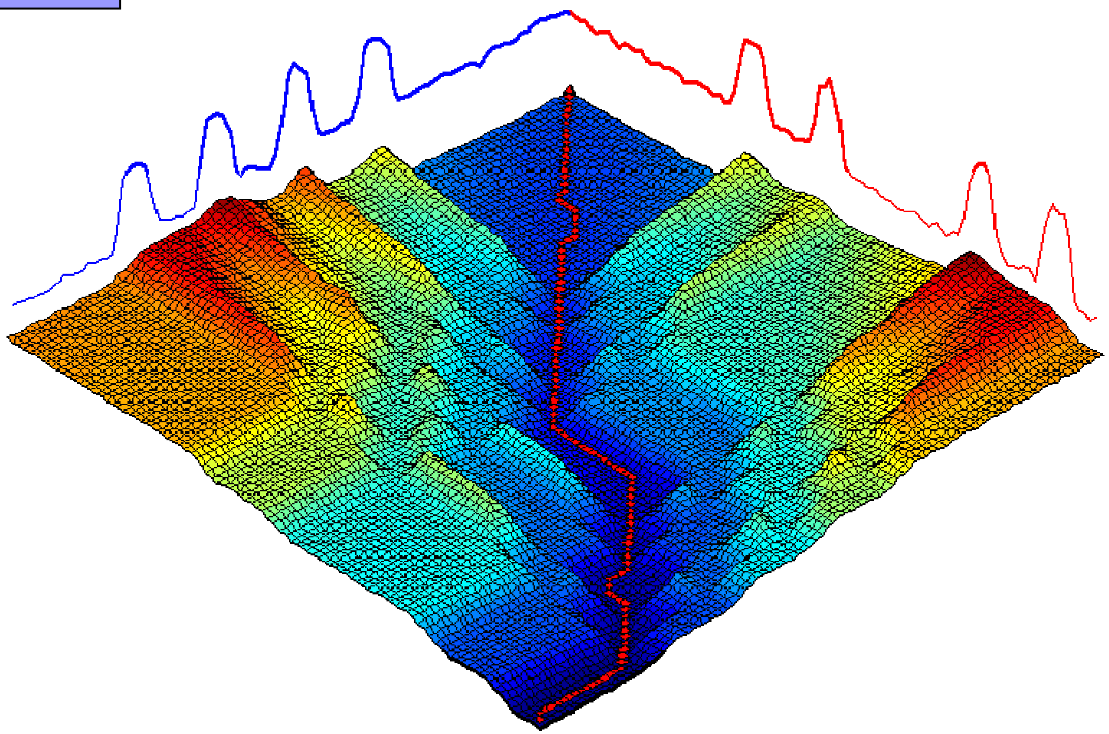
7	40	22	31	43	24	**15**
3	15	6	7	11	8	**6**
4	14	6	9	18	8	**5**
5	10	5	11	18	7	**5**
2	1	**2**	**2**	**3**	**4**	8
1	**1**	5	6	6	10	19
WP	**2**	**3**	**2**	**1**	**3**	**4**

- Then set $\mathbf{d_{DTW}(s,q) = \sqrt{wp_{D-1,D-1}}}$

A real-world graphical example

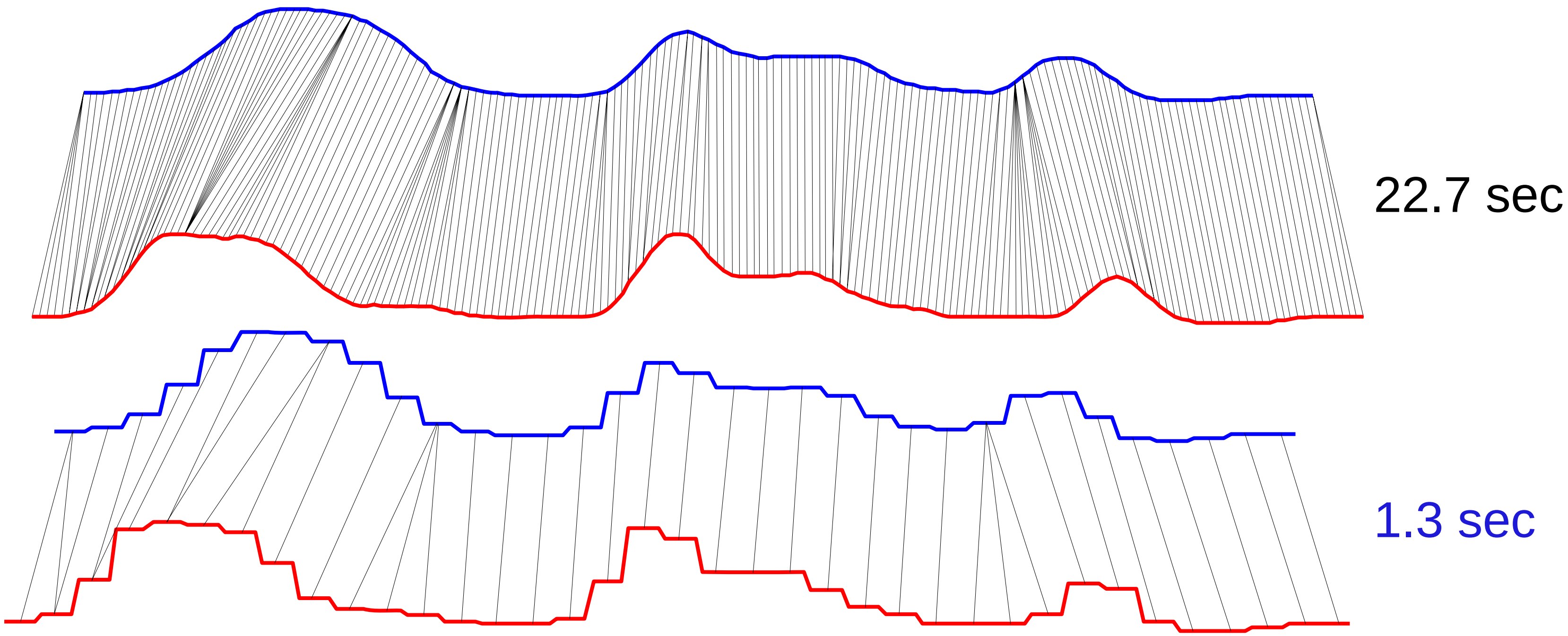
Power-Demand time series
Each sequence corresponds to a week's demand for power in a Dutch research facility in 1997





Fast searching with DTW

- We have now 2 problems to face, if we want to use DTW for searching:
  1. Computing the DTW is very time-consuming
  2. How to index it?
- Both problems can be solved:
  1. Use a lower-resolution approximation of the time series
     - However the method can introduce false dismissals



How to index DTW?

- Using metric trees!
- Unfortunately, DTW is **not** a metric…
- Proof:
  - s=<0,0>
  - t=<1,2>
  - q=<1,2,2>
  - DTW(s,q) = 9 > (DTW(s,t) + DTW(t,q)) = 5 + 0

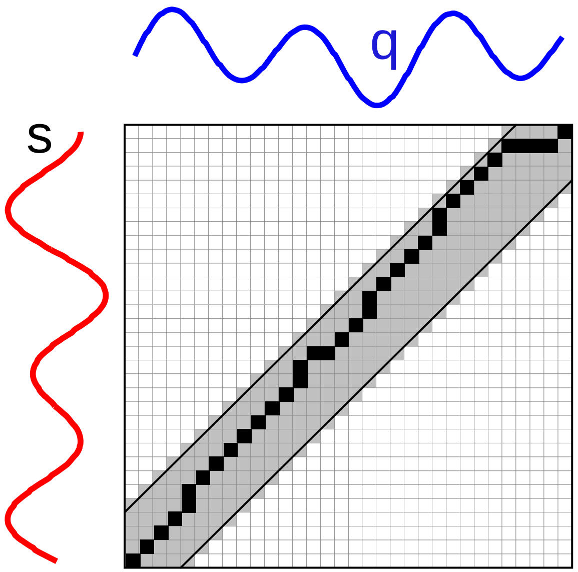
0	2	**5**	**9**
0	**1**	5	9
WP	**1**	**2**	**2**

0	2	**5**
0	**1**	5
WP	**1**	**2**

2	1	**0**	**0**
1	**0**	1	2
WP	**1**	**2**	**2**

Indexing the DTW (sketch) (i)

- An effective indexing technique for DTW has been proposed in [Keo02]
- The method applies only if we have some "global constraint" on the allowed warping paths



The Sakoe-Chiba band of width h=4

s

7				43	24	**17**
3			7	11	8	**8**
4		6	9	18	8	**7**
5	10	5	11	18	**7**	
2	1	**2**	**2**	**3**		
1	**1**	5	6			
WP	**2**	**3**	**2**	**1**	**3**	**4**

q

Our example with h=2

Final considerations

- We have just seen some basic techniques to deal with (large) time series databases

- Other relevant problems exist and have attracted interest, among which:
  - Searching for similar sub-sequences
  - Searching for multi-dimensional time series (i.e., trajectories)