České vysoké učení technické v Praze

Fakulta informačních technologií

Katedra teoretické informatiky

Evropský sociální fond
Praha & EU: Investujeme do vaší budoucnosti

MI-PDD – Data Preprocessing module (2011/2012)

Lecture 8: Feature extraction from time series

Pavel Kordík, FIT, Czech Technical University in Prague

Feature Extraction from signals

- Time series
  - Industry, signals from sensors
  - Biological data
  - Financial time series
  - Speech
  - Music
  - ...
- Why we need to describe time series by features?

Examples of time series

A. Far-infrared laser excitation
B. Sleep Apnea
C. Currency exchange rates
D. Particle driven in nonlinear multiple well potentials
E. Variable star data
F. J. S. Bach fugue notes

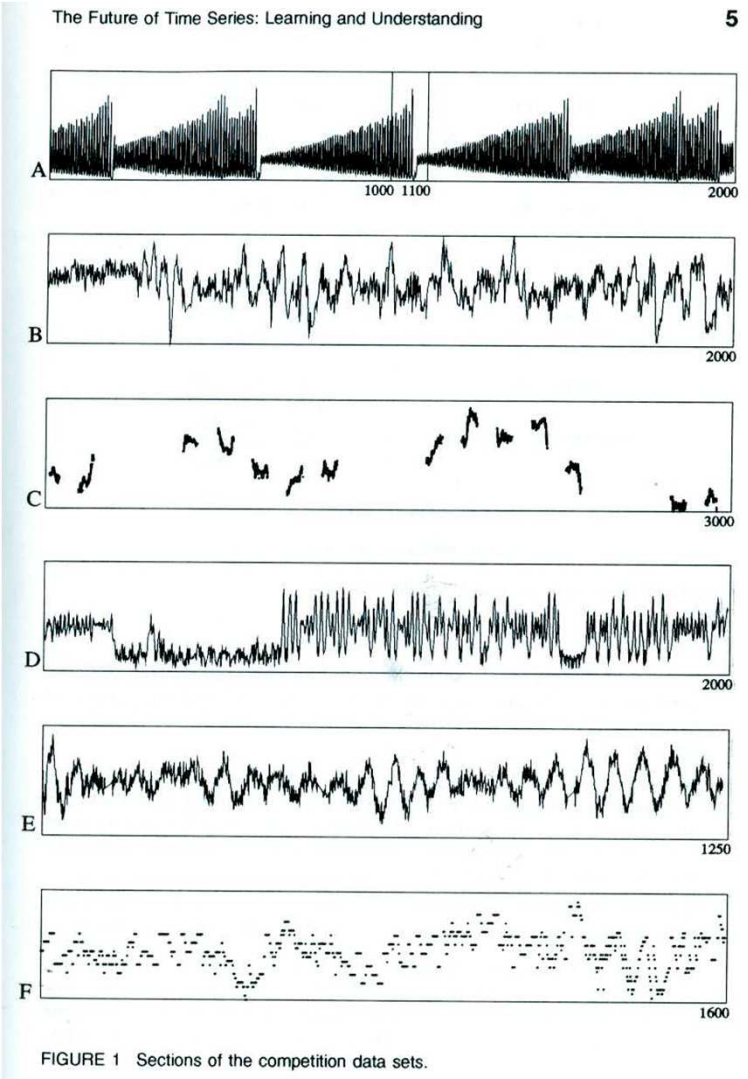
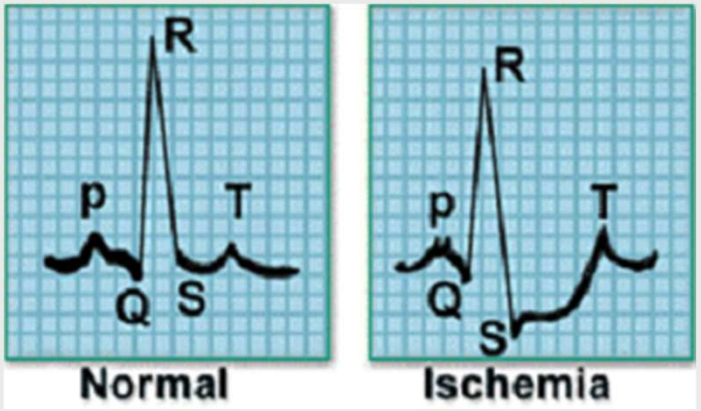


FIGURE 1 Sections of the competition data sets.

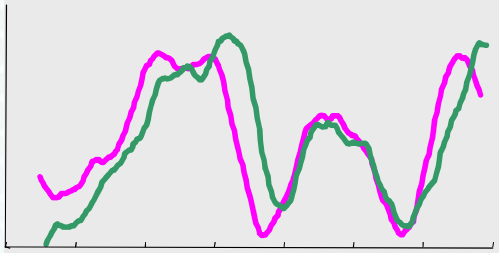
Time series data mining tasks

Clustering

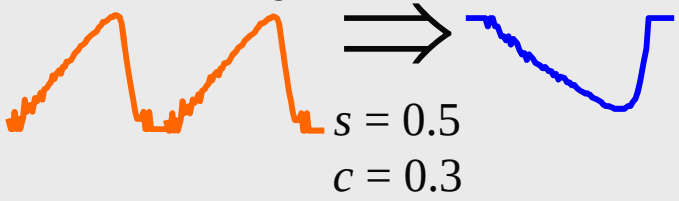
Classification



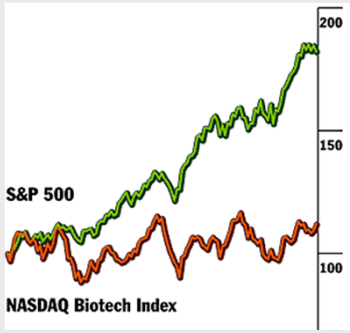
Motif Discovery



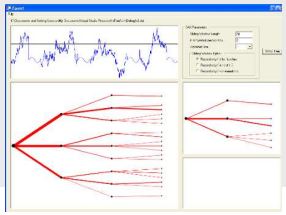
Rule Discovery



Query by Content

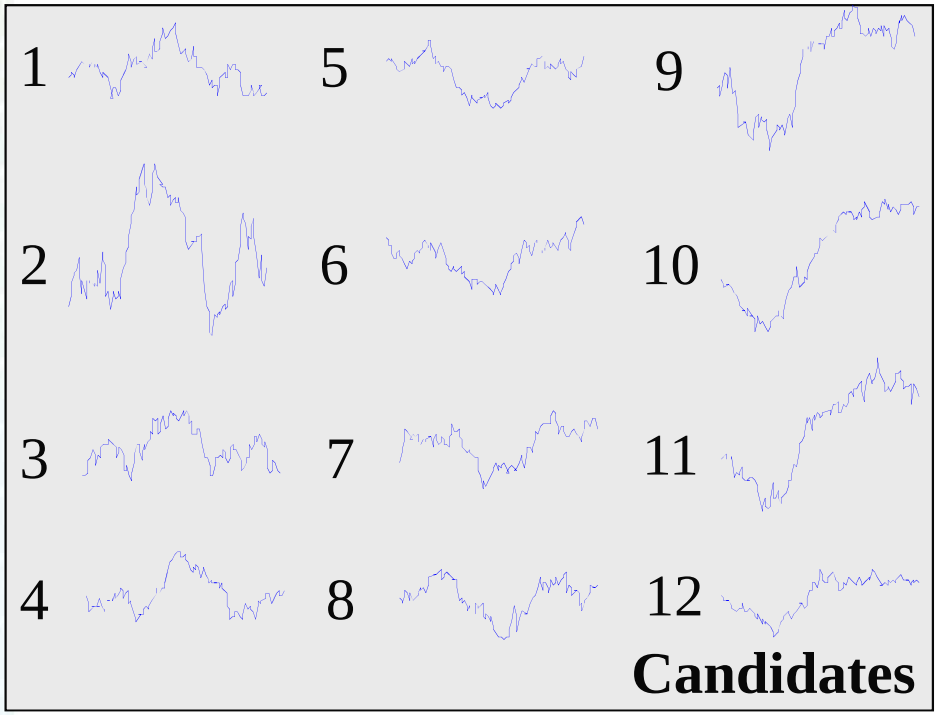


Visualization



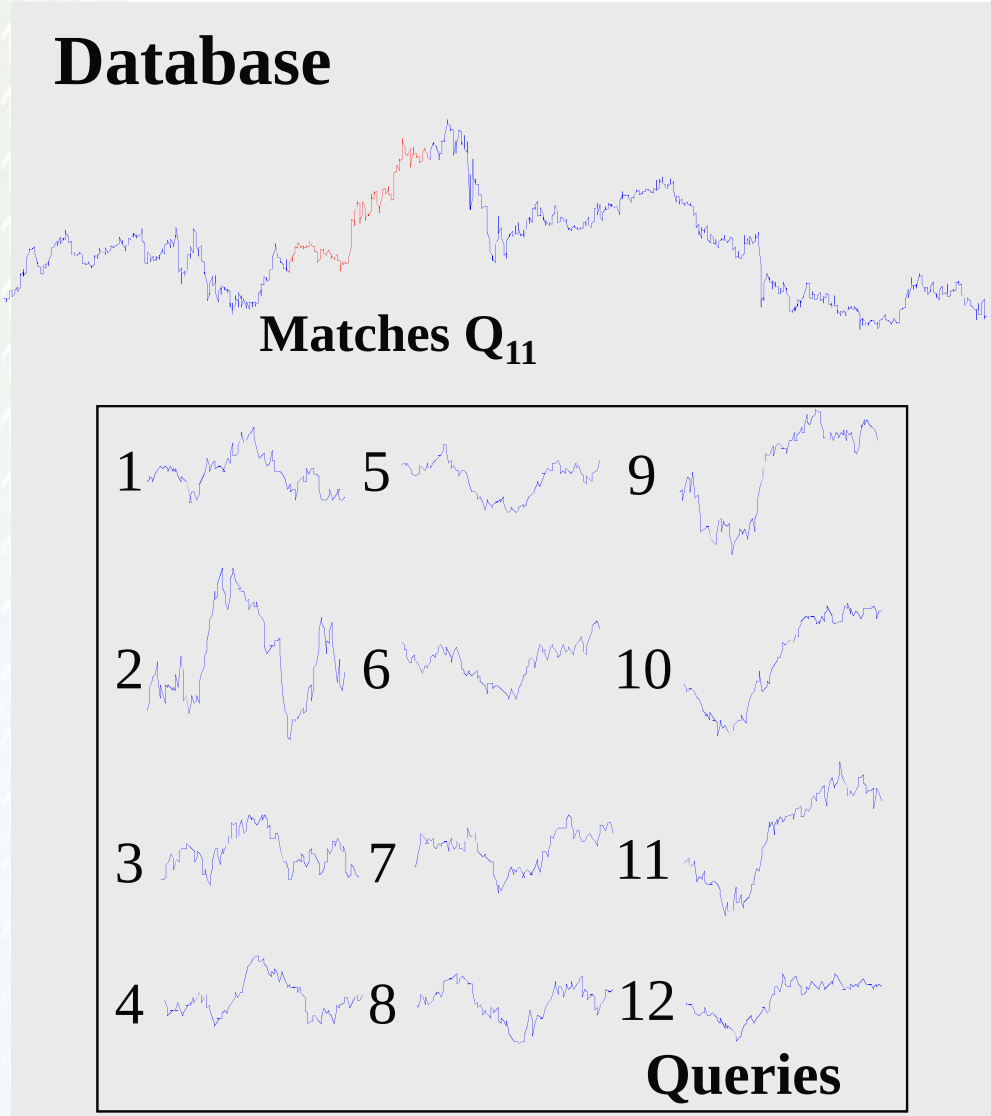
Novelty Detection

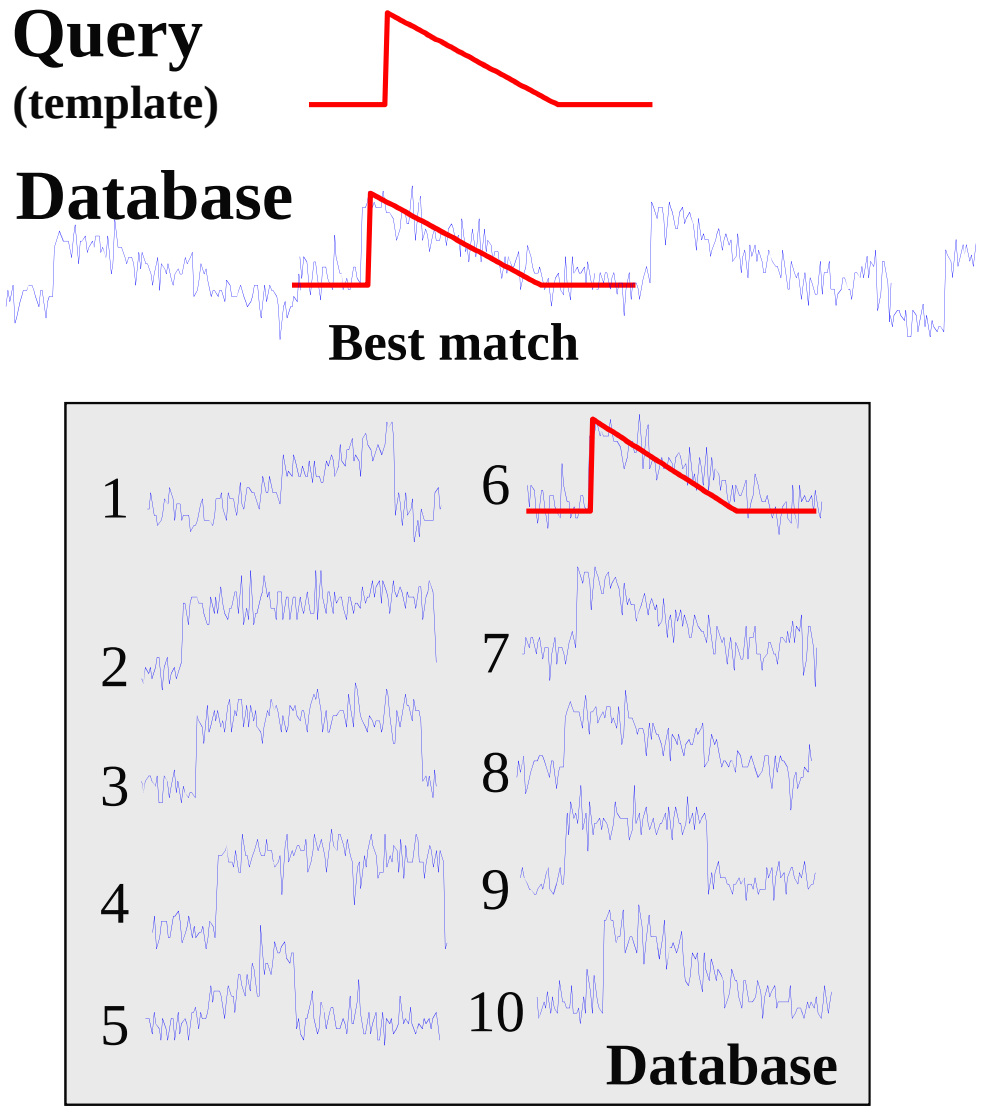
Time Series Filtering



Given a Time Series ***T***, a set of Candidates ***C*** and a distance threshold ***r***, find all subsequences in ***T*** that are within ***r*** distance to any of the candidates in ***C***.

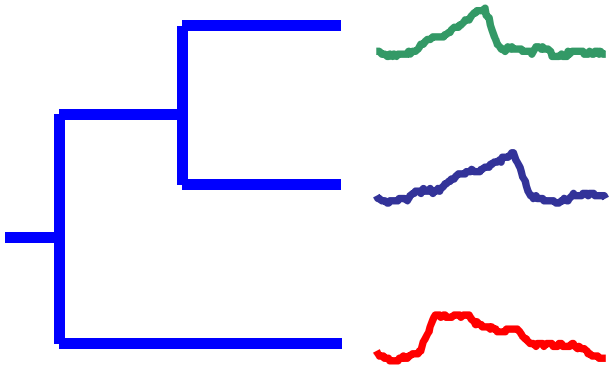
Filtering vs. Querying



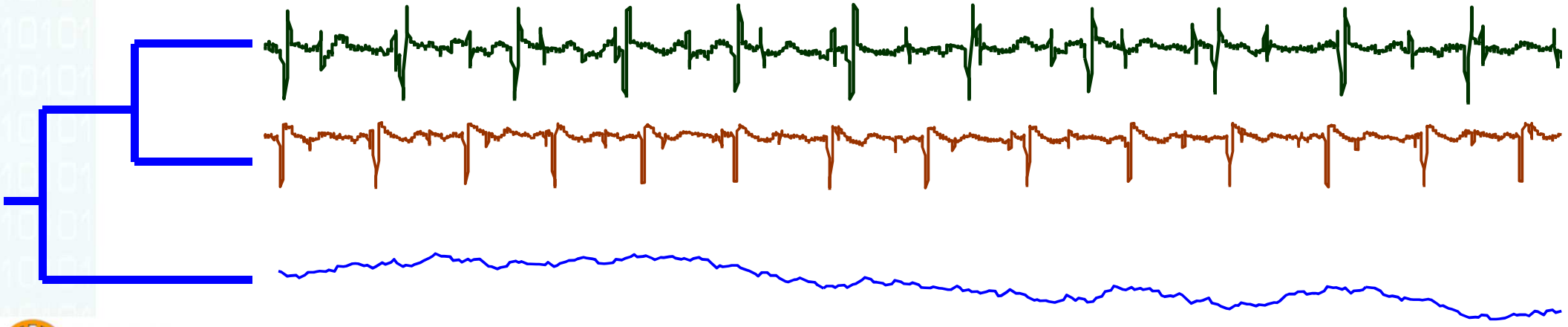


Time series similarity

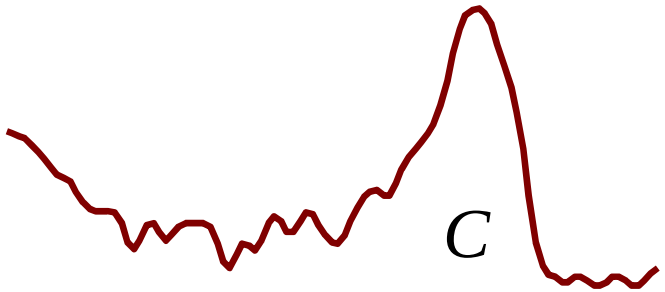
Similarity at the level of *shape*

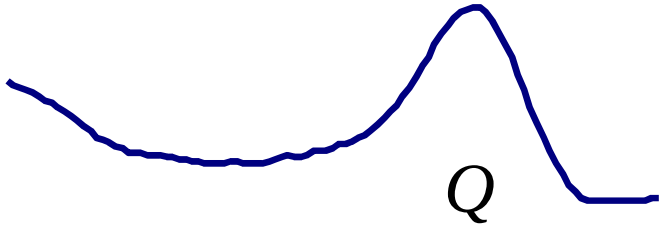


Similarity at the *structural* level



Distance of two signals



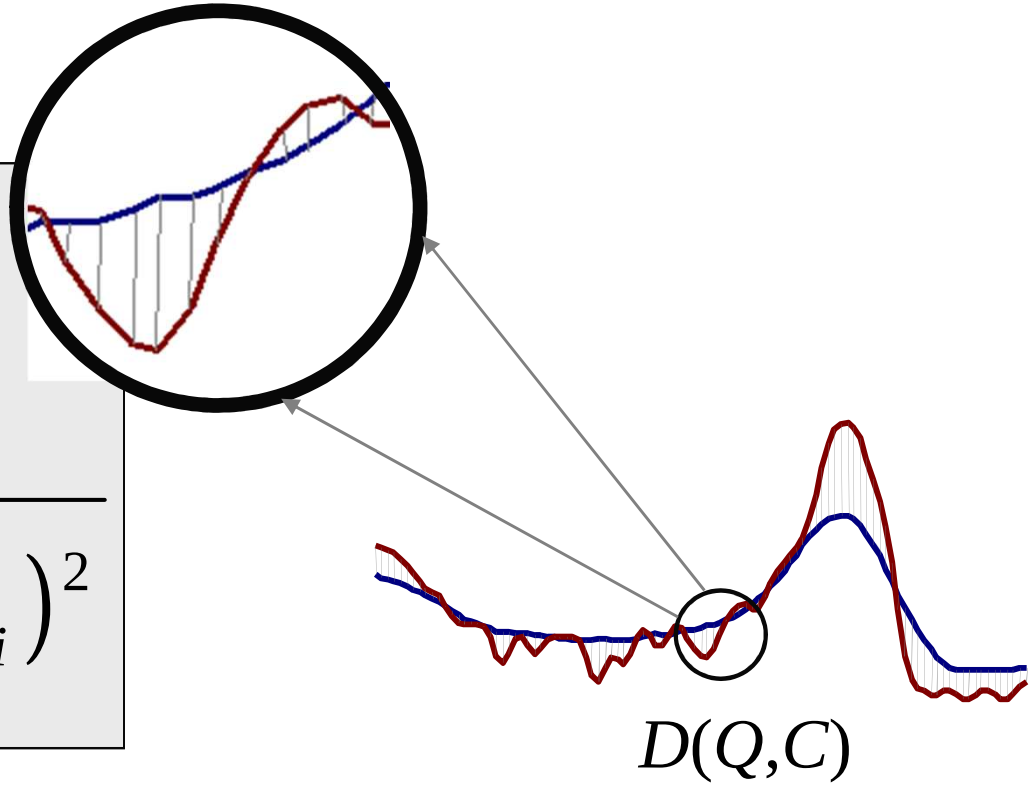


Given two time series:

$$Q = q_1 \ldots q_n$$

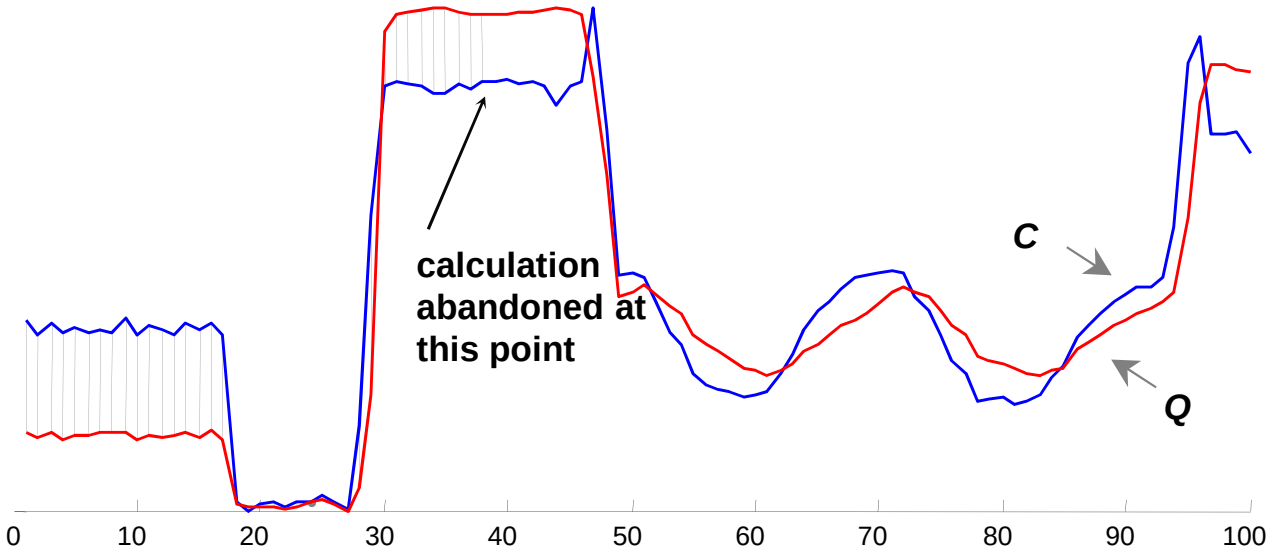
$$C = c_1 \ldots c_n$$

$$D(Q,C) \equiv \sqrt{\sum_{i=1}^{n} (q_i - c_i)^2}$$

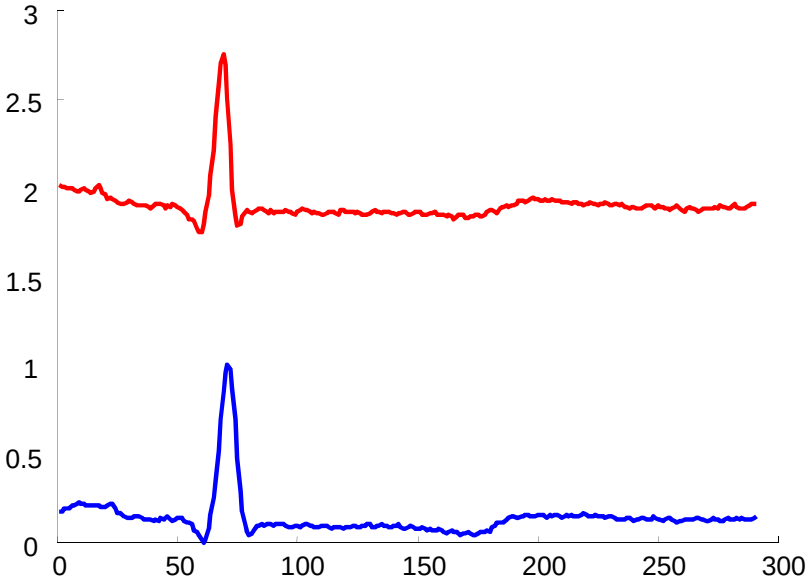


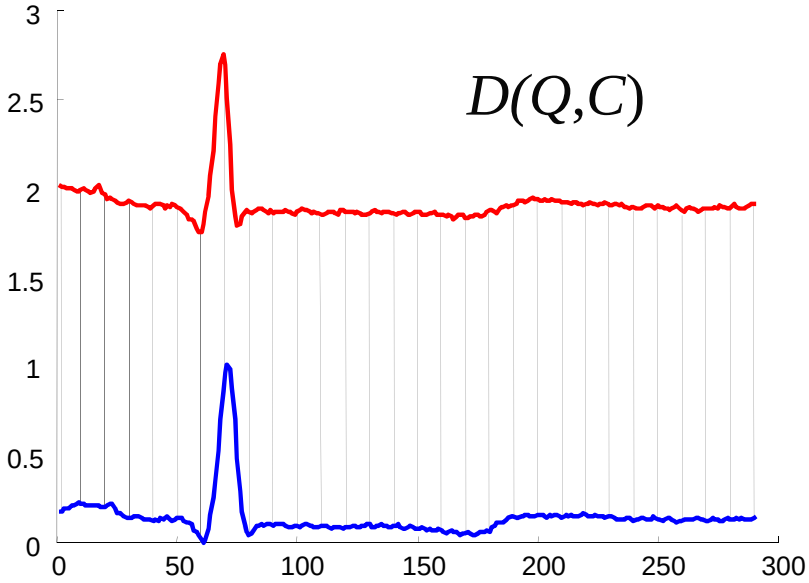
Early Abandon

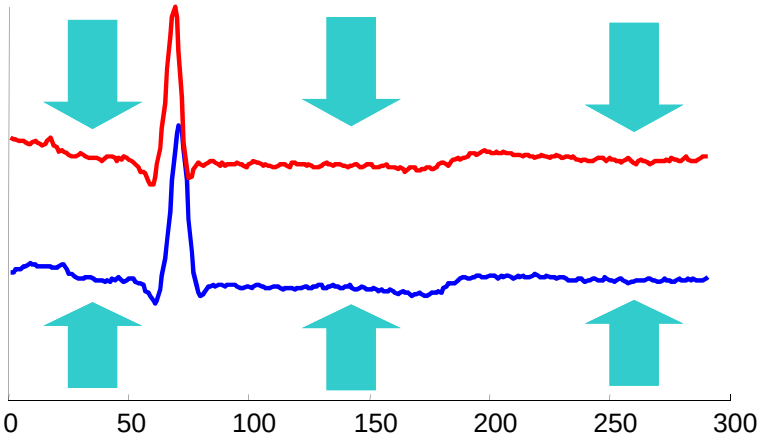
During the computation, if current sum of the squared differences between each pair of corresponding data points exceeds r^2, we can safely stop the calculation.



First translate offsets



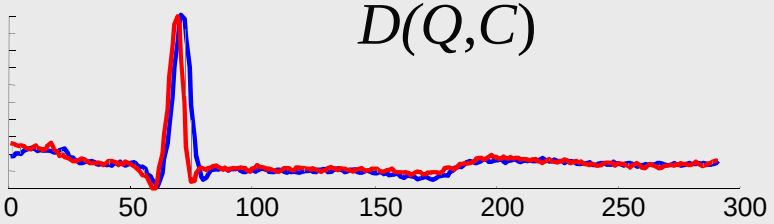




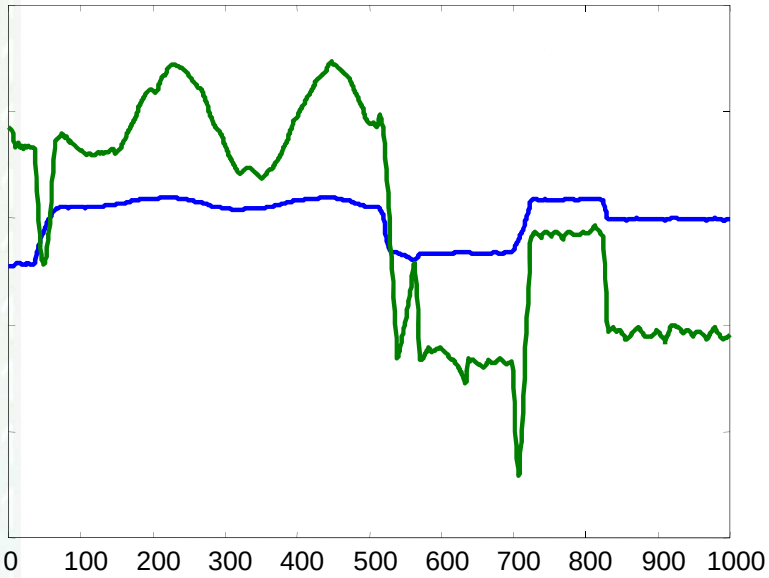
$Q = Q - \text{mean}(Q)$

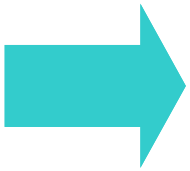
$C = C - \text{mean}(C)$

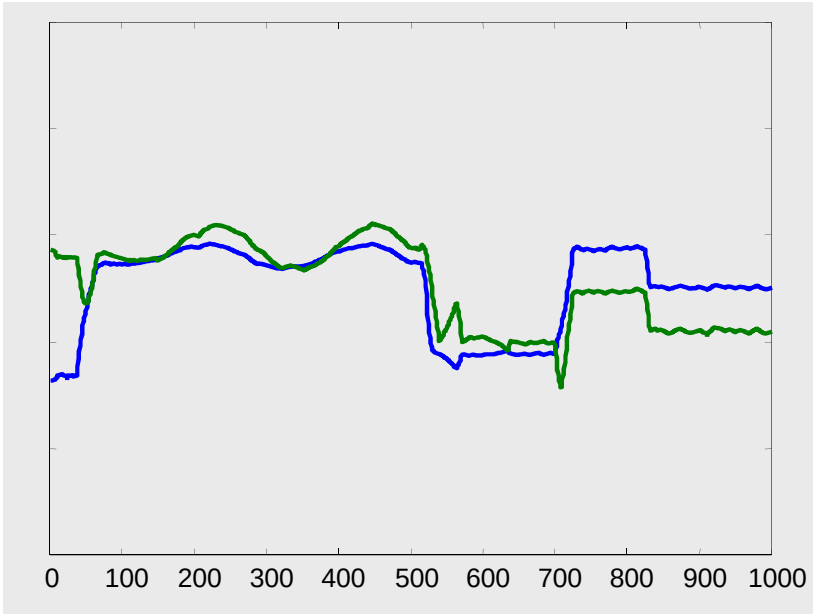
$D(Q,C)$



Then scale (normalize) signals





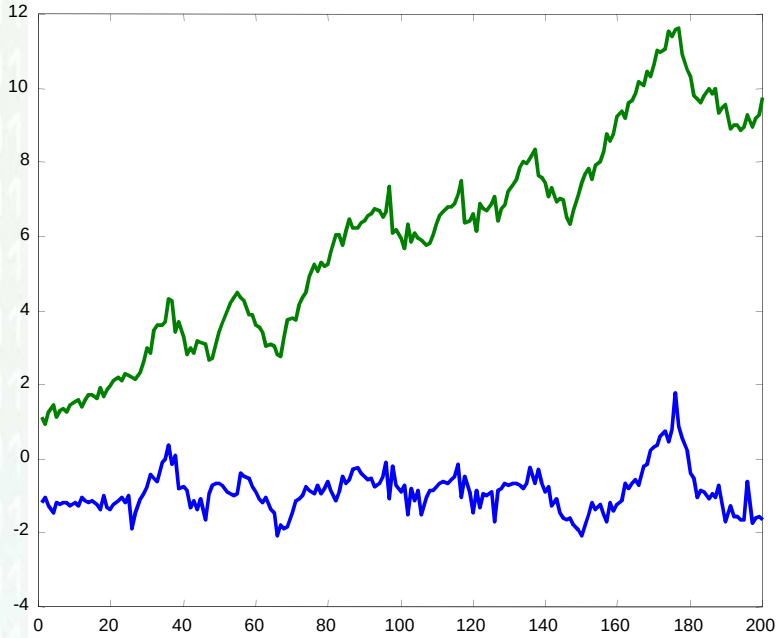


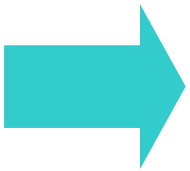
$Q = (Q - \text{mean}(Q)) / \text{std}(Q)$

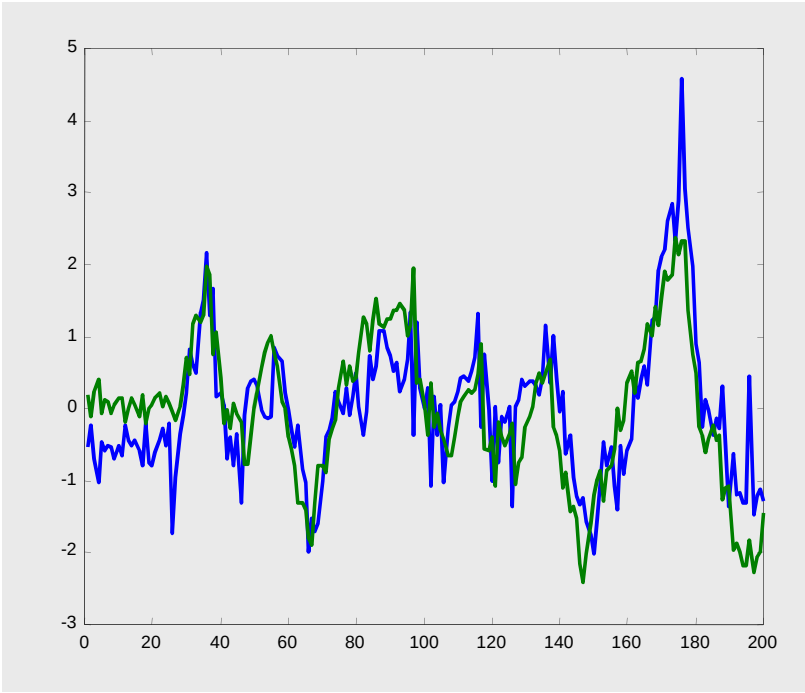
$C = (C - \text{mean}(C)) / \text{std}(C)$

$D(Q,C)$

Then remove trend (optional)





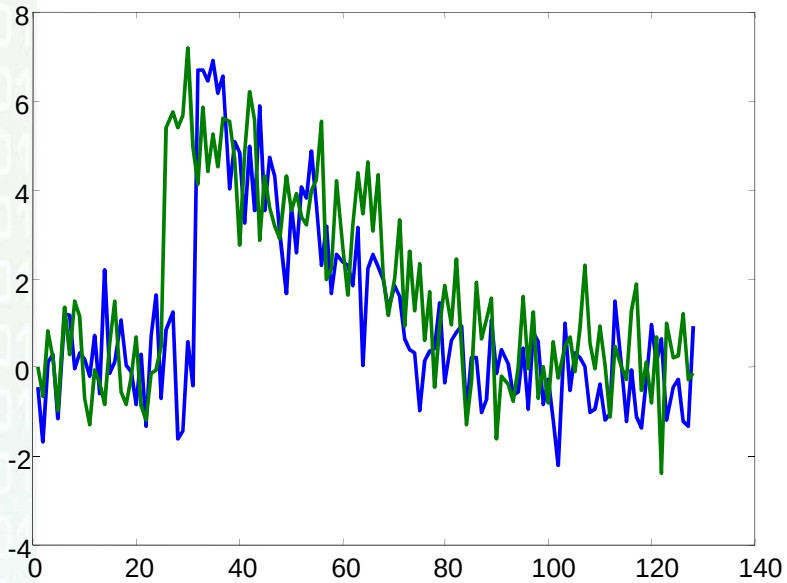


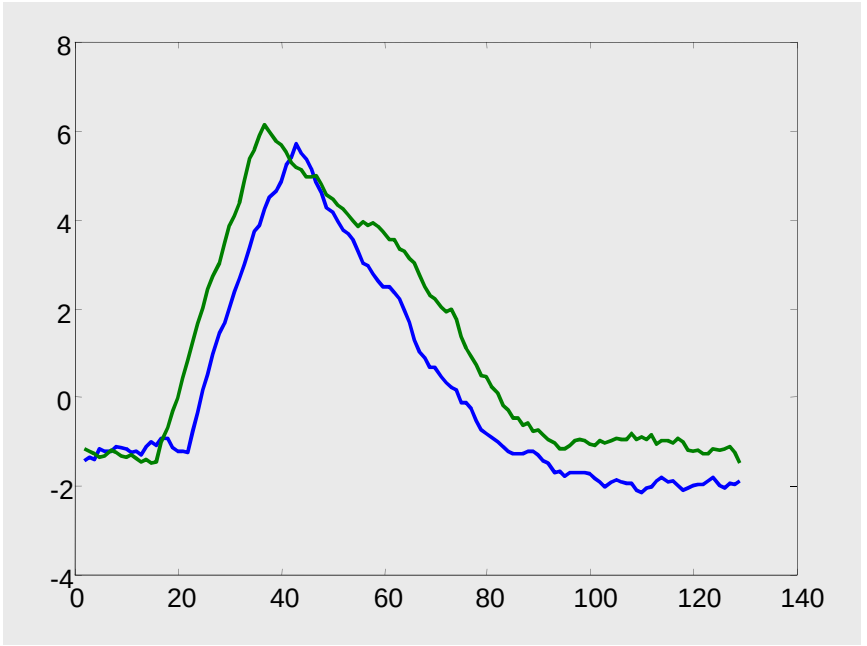
Removed **linear trend**

Removed offset translation

Removed amplitude scaling

Remove noise





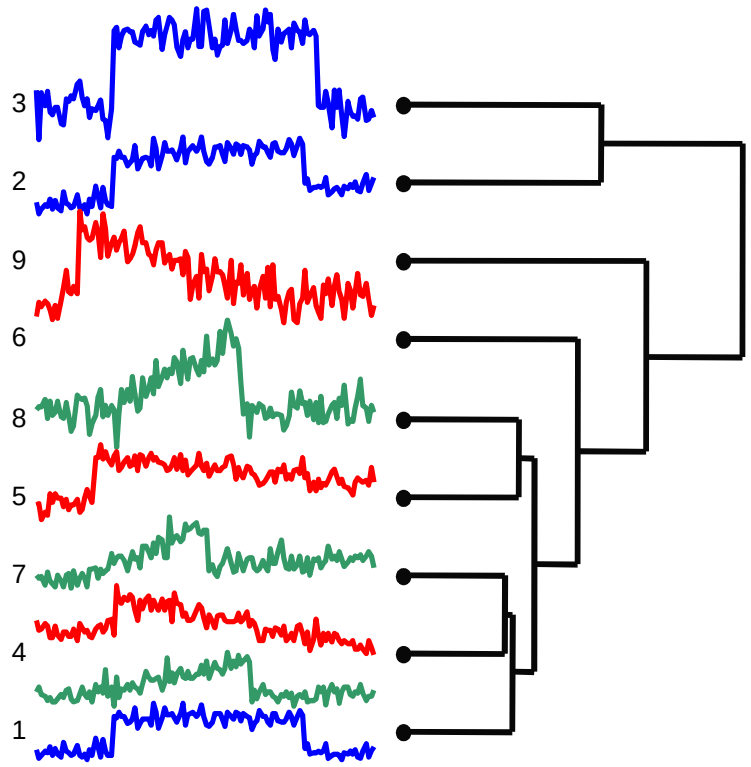
$Q = \text{smooth}(Q)$

$C = \text{smooth}(C)$

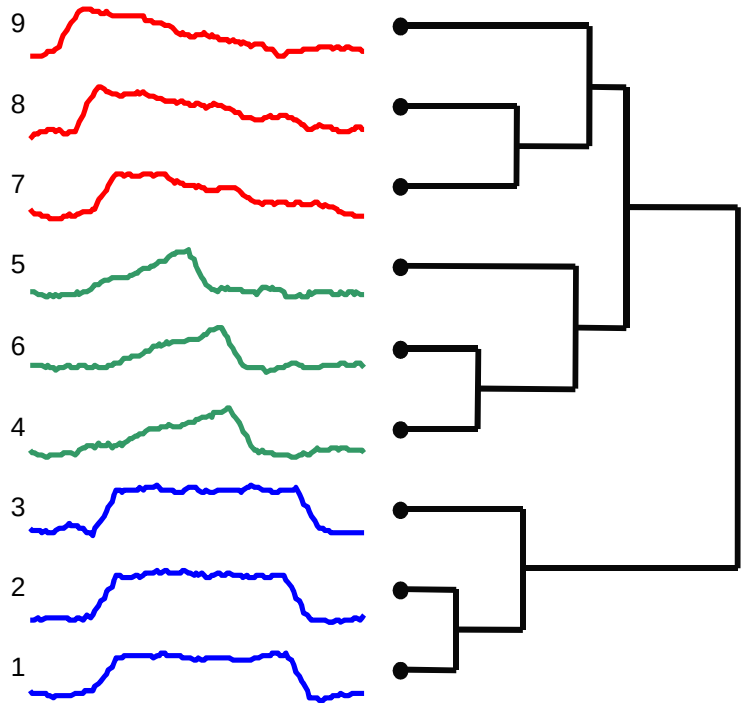
$D(Q,C)$

Agglomerative clustering using Euclidian distance

- Without preprocessing



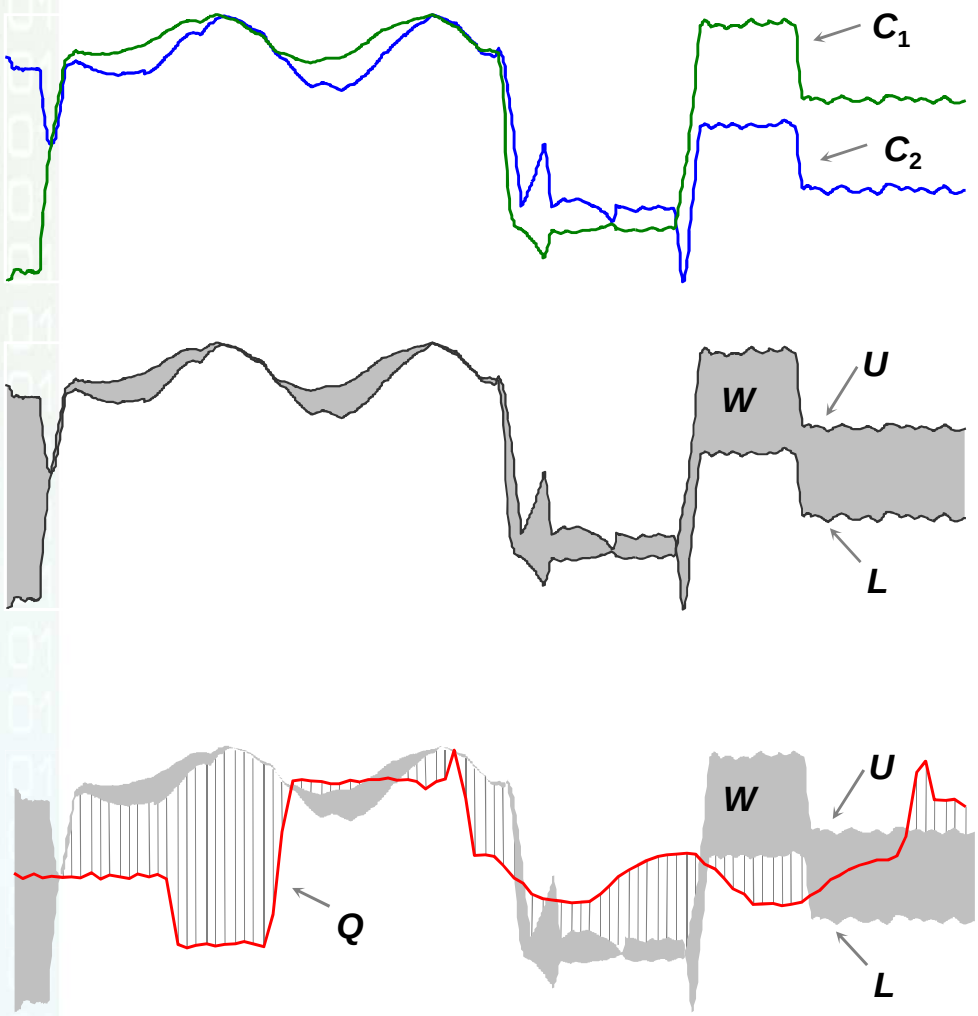
- Noise removed
- Trend removed
- Offset translated
- Signals normalized



Wedge

Li Wei Eamonn Keogh

Computer Science & Engineering Dept.
University of California – Riverside
Riverside, CA 92521
{wli, eamonn}@cs.ucr.edu



Having candidate sequences $C_1, .. , C_k$, we can form two new sequences U and L :

$$U_i = \max(C_{1i}, .. , C_{ki})$$
$$L_i = \min(C_{1i}, .. , C_{ki})$$

They form the smallest possible bounding envelope that encloses sequences $C_1, .. , C_k$.

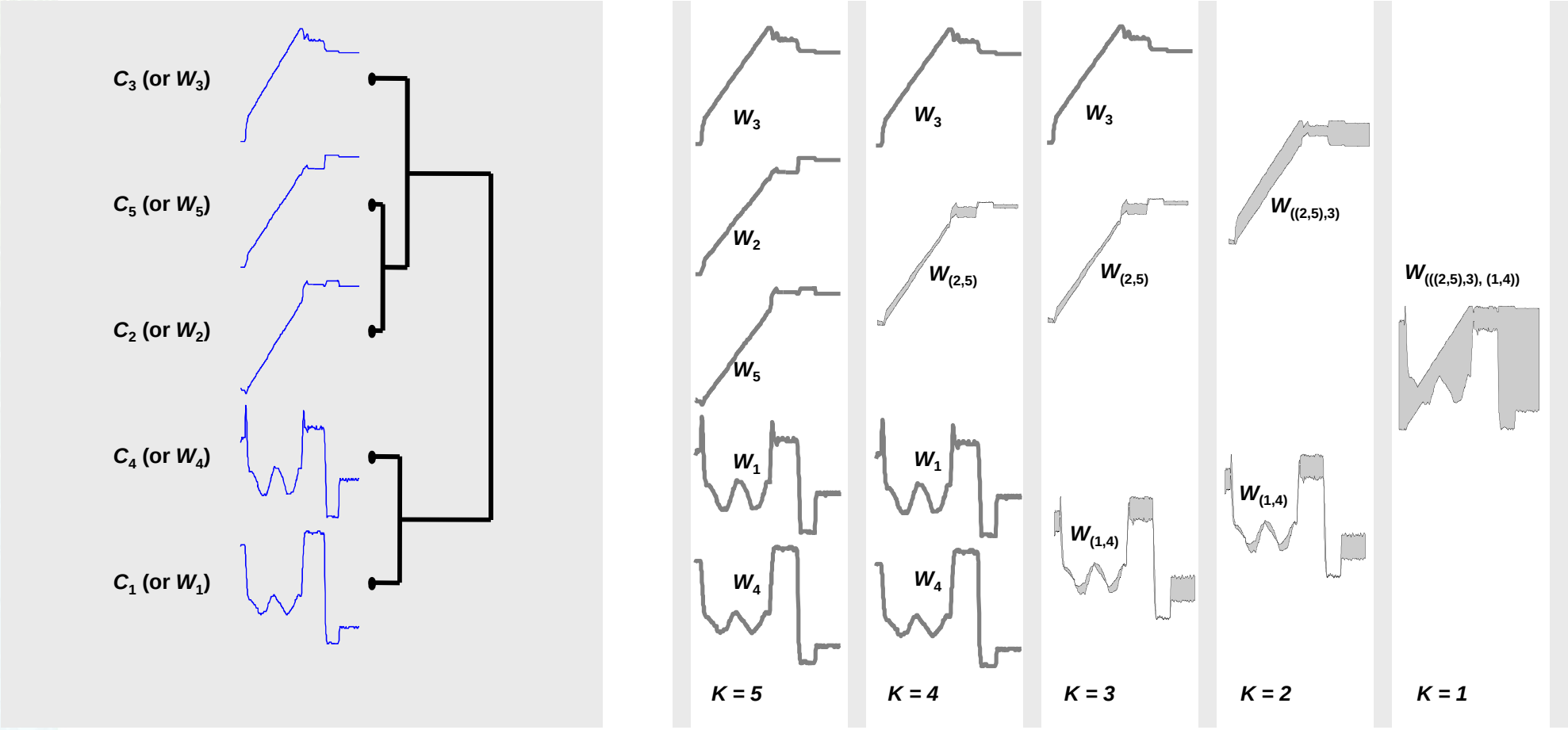
We call the combination of U and L a *wedge*, and denote a wedge as W.

$$W = \{U, L\}$$

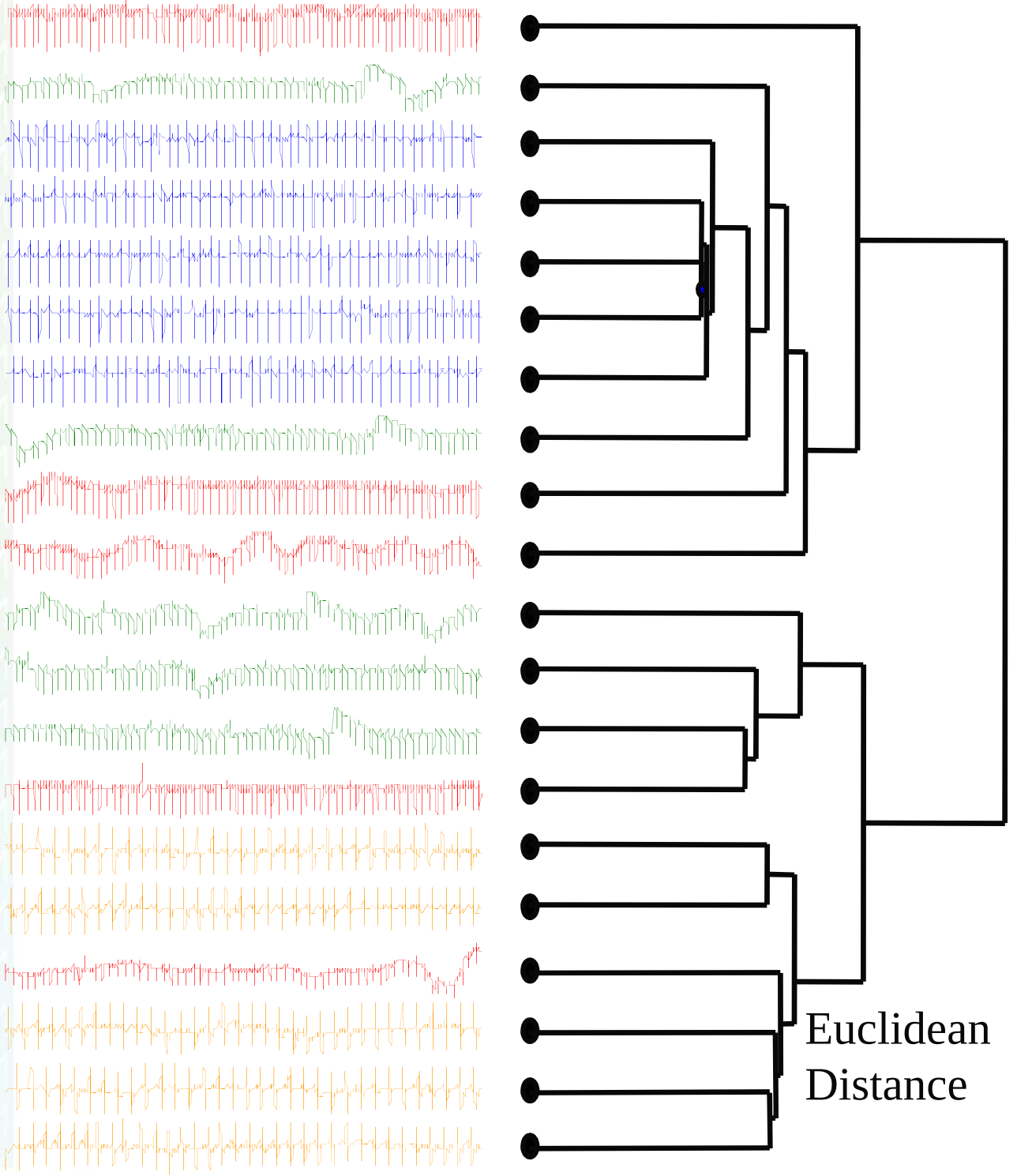
A lower bounding measure between an arbitrary query Q and the entire set of candidate sequences contained in a wedge W:

$$LB\_Keogh(Q,W) = \sqrt{\sum_{i=1}^{n} \begin{cases} (q_i - U_i)^2 & \text{if } q_i > U_i \\ (q_i - L_i)^2 & \text{if } q_i < L_i \\ 0 & \text{otherwise} \end{cases}}$$

Hierarchal Clustering with Wedges



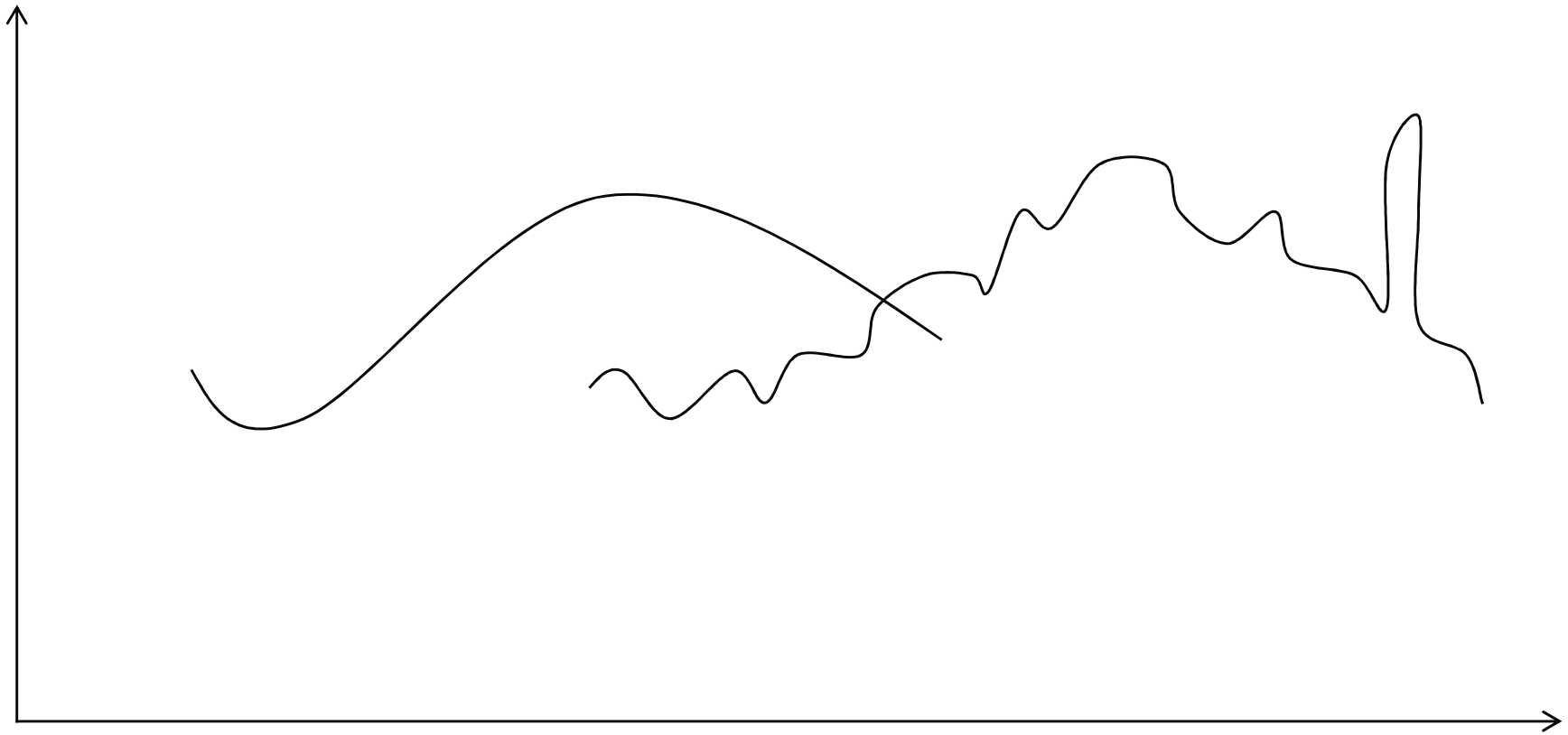
Wedges can be used for filtering or querying.



But!

- Euclidian distance not universally applicable!
- Consider:
  - Phase shift
  - Delays
  - Unsynchronized signals
  - ...

Example



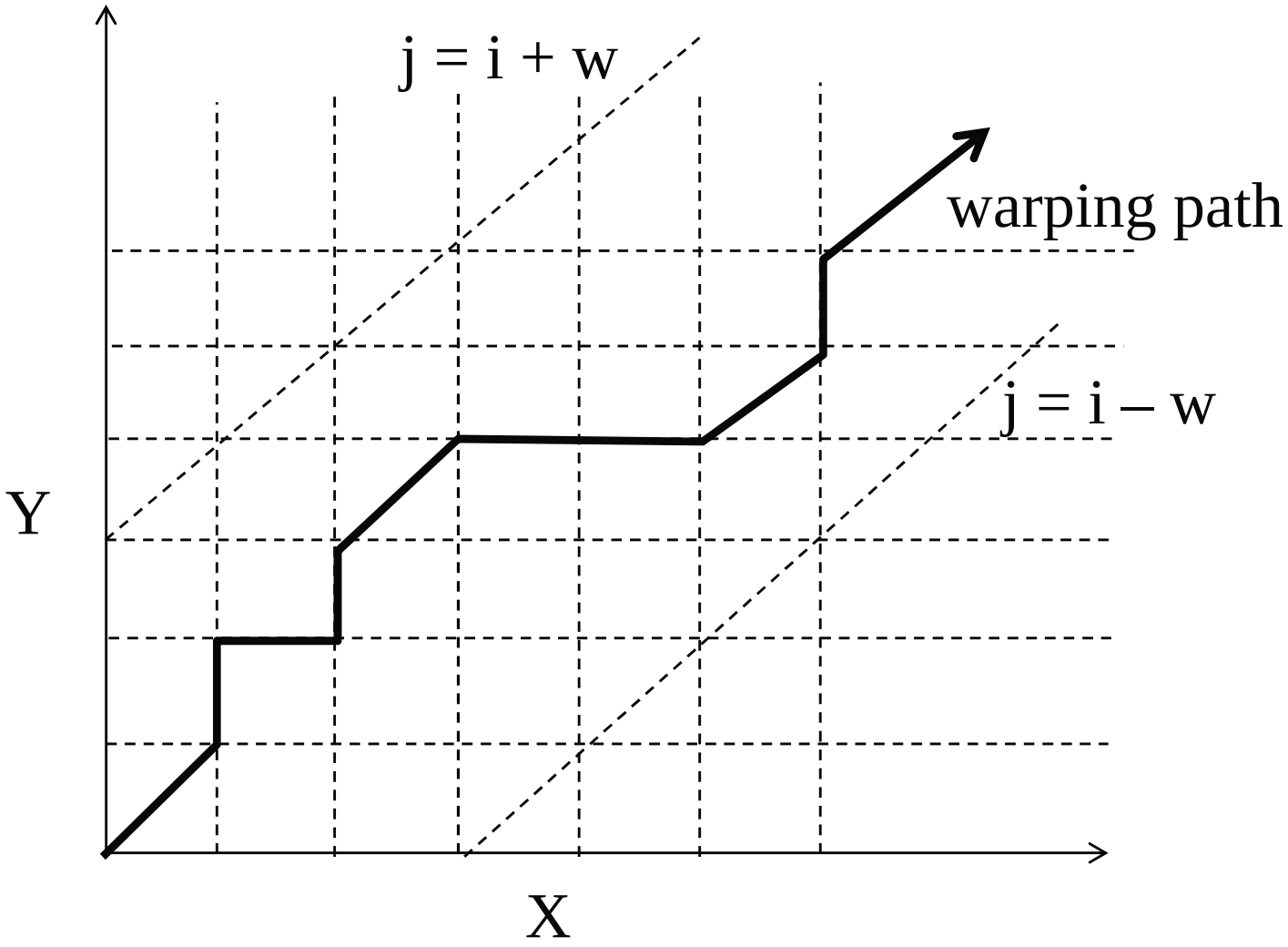
Dynamic Time Warping
[Berndt, Clifford, 1994]

- Allows acceleration-deceleration of signals along the time dimension

- Basic idea
  - Consider X = $x_1, x_2, \ldots, x_n$, and Y = $y_1, y_2, \ldots, y_n$
  - We are allowed to extend each sequence by repeating elements
  - Euclidean distance now calculated between the extended sequences X’ and Y’
  - Matrix M, where $m_{ij} = d(x_i, y_j)$

Dynamic Time Warping

[Berndt, Clifford, 1994]



Restrictions on Warping Paths

- Monotonicity
  - Path should not go down or to the left
- Continuity
  - No elements may be skipped in a sequence
- Warping Window

$$|i - j| <= w$$

Formulation

- Let D(i, j) refer to the dynamic time warping distance between the subsequences

$$x_1, x_2, \ldots, x_i$$
$$y_1, y_2, \ldots, y_j$$

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

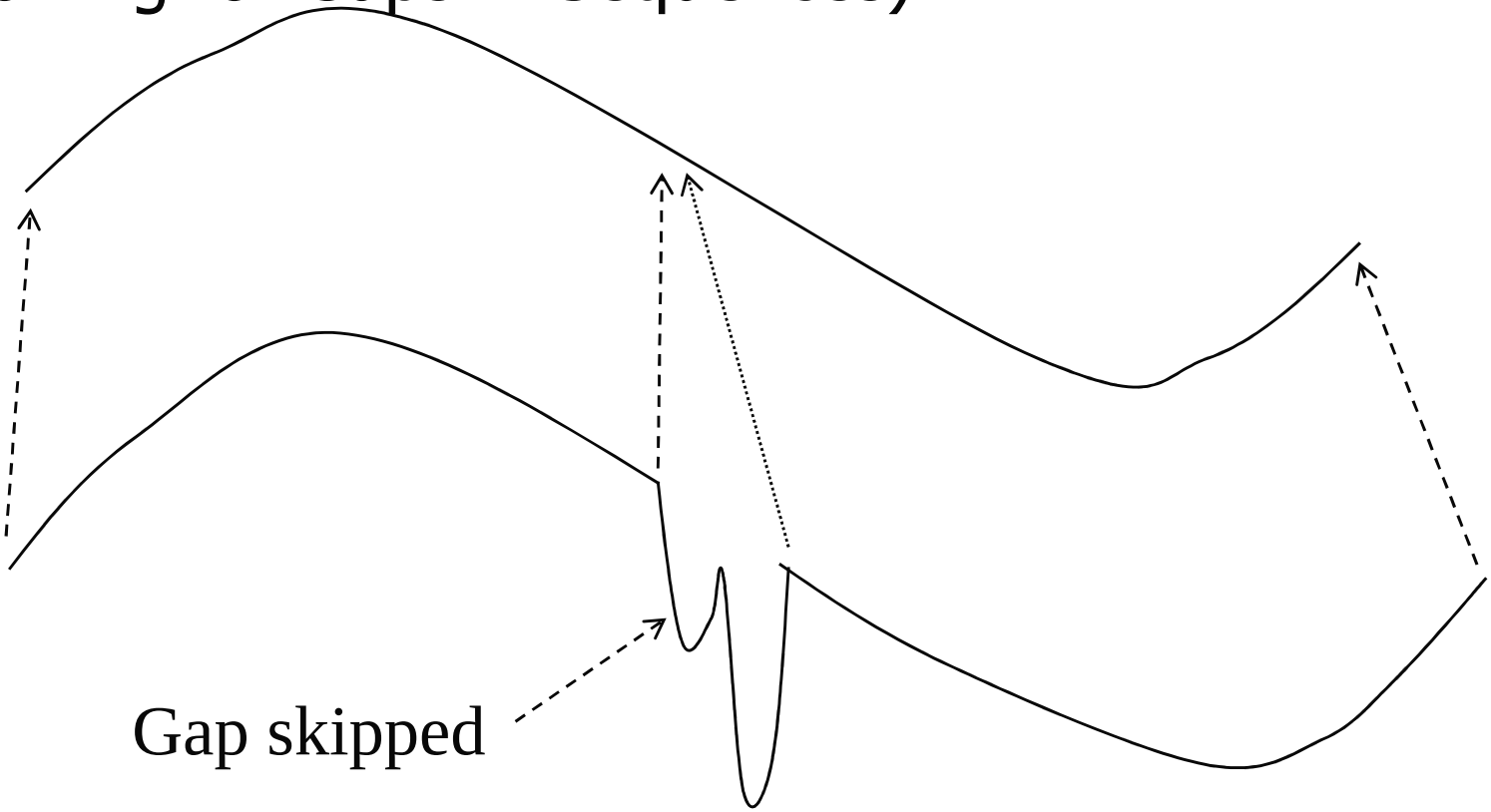
Solution by Dynamic Programming

- Basic implementation = $O(n^2)$ where n is the length of the sequences
  - will have to solve the problem for each (i, j) pair
- If warping window is specified, then O(nw)
  - Only solve for the (i, j) pairs where

  $| i - j | <= w$

Longest Common Subsequence Measures

(Allowing for Gaps in Sequences)



Basic LCS Idea

X = 3, **2**, **5**, **7**, 4, 8, **10**, 7

Y = **2**, **5**, 4, **7**, 3, **10**, 8, 6

LCS = **2**, **5**, **7**, **10**

$$Sim(X,Y) = |LCS| \text{ or } Sim(X,Y) = |LCS| / n$$

Edit Distance is another possibility

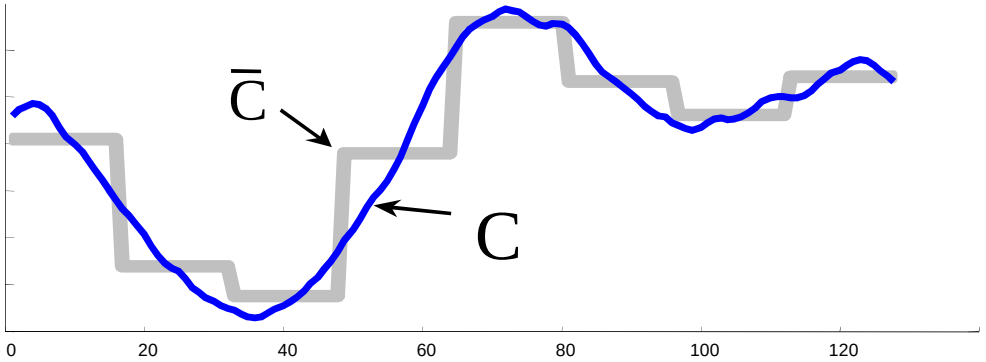
Landmarks
[Perng et. al., 2000]

- Similarity definition much closer to human perception (unlike Euclidean distance)
- A point on the curve is a n-th order landmark if the n-th derivative is 0
  - Thus, local max and mins are first order landmarks
- Landmark distances are tuples (e.g. in time and amplitude) that satisfy the triangle inequality
- Several transformations are defined, such as shifting, amplitude scaling, time warping, etc

PAA and APCA

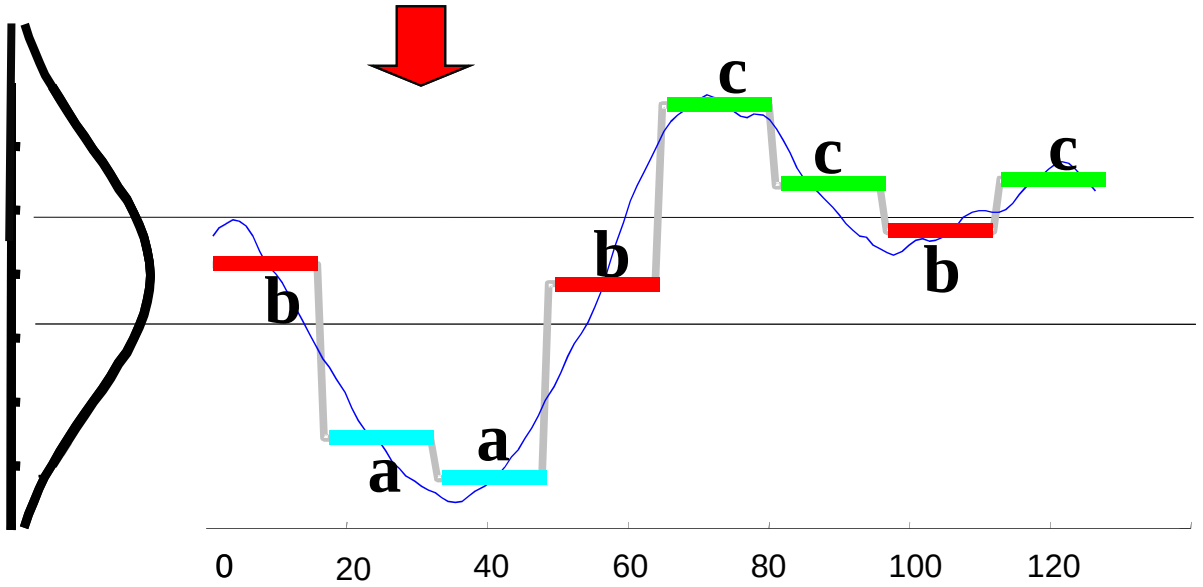
- **P**iecewise **A**ggregate **A**pproximation
  - Segment the time series into equal parts, store the average value for each part.
- **A**daptive **P**iecewise **C**onstant **A**pproximation
  - Parts are of adaptive length

Symbolic Aggregate approXimation



First convert the time series to PAA representation, then convert the PAA to symbols

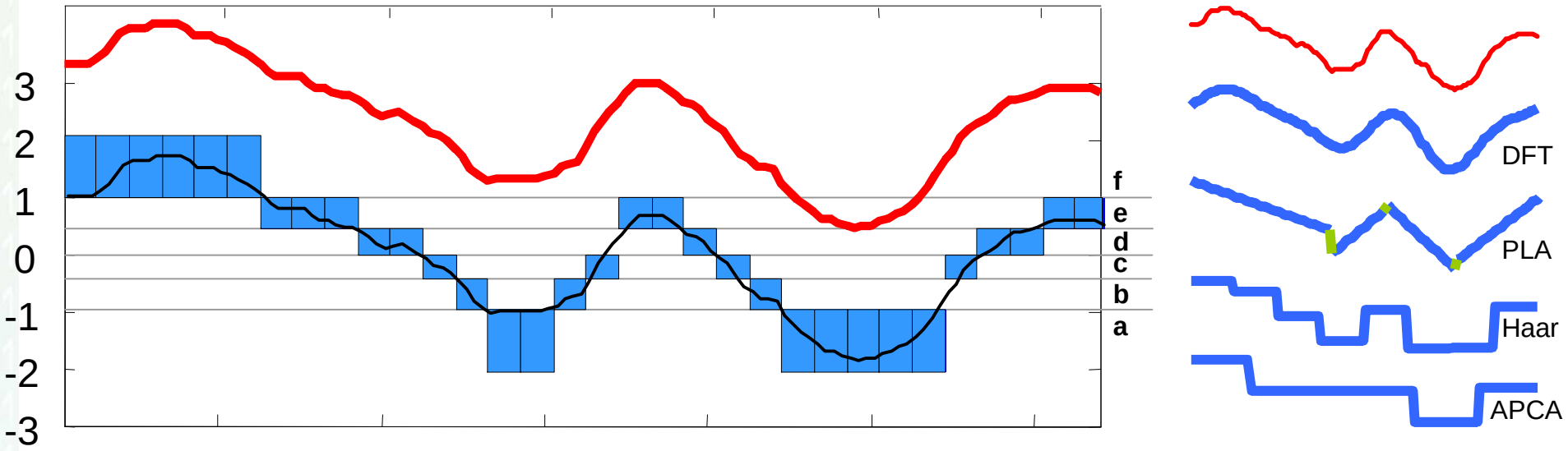
It take linear time



baabccbc

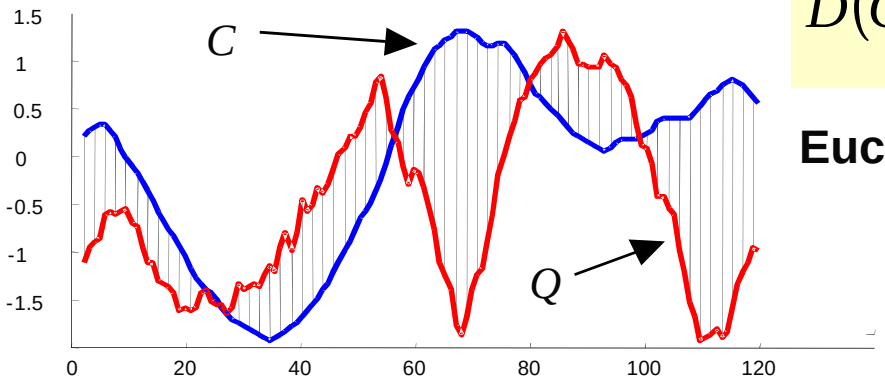
Eamonn Keogh and Jessica Lin
Computer Science & Engineering Department
University of California - Riverside
Riverside,CA 92521
eamonn@cs.ucr.edu

Visual Comparison



A raw time series of length 128 is transformed into the word "**fffffeeeddcbaabceedcbaaaaacddee**."

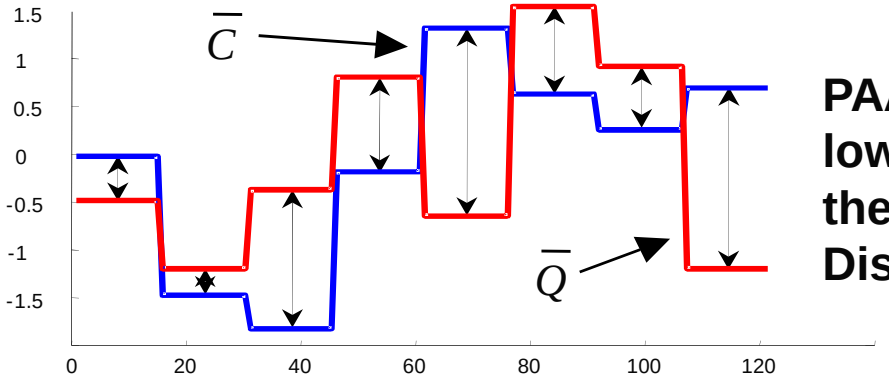
- We can use more symbols to represent the time series since each symbol requires fewer bits than real-numbers (float, double)



$$D(Q,C) \equiv \sqrt{\sum_{i=1}^{n} (q_i - c_i)^2}$$

Euclidean Distance

$$DR(\overline{Q},\overline{C}) \equiv \sqrt{\frac{n}{w}} \sqrt{\sum_{i=1}^{w} (\overline{q}_i - \overline{c}_i)^2}$$



PAA distance lower-bounds the Euclidean Distance

$\hat{C}$ = **baabccbc**

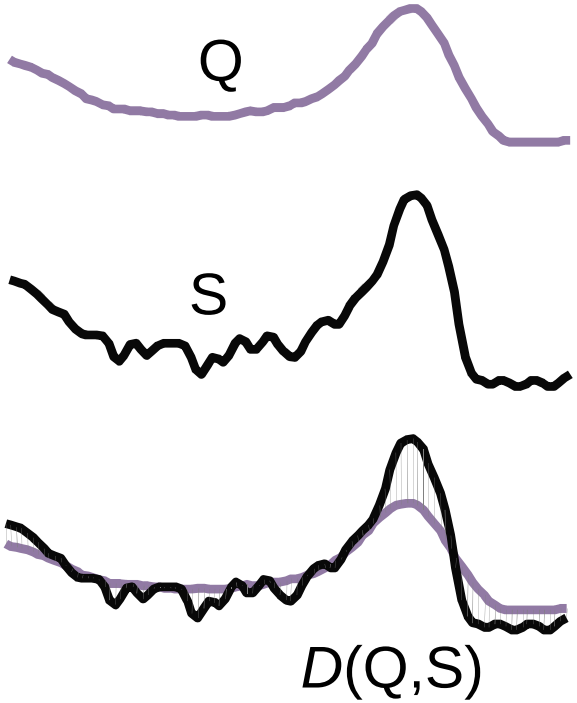
$\hat{Q}$ = **babcacca**

$$MINDIST(\hat{Q},\hat{C}) \equiv \sqrt{\frac{n}{w}} \sqrt{\sum_{i=1}^{w} (dist(\hat{q}_i, \hat{c}_i))^2}$$

dist() can be implemented using a table lookup.

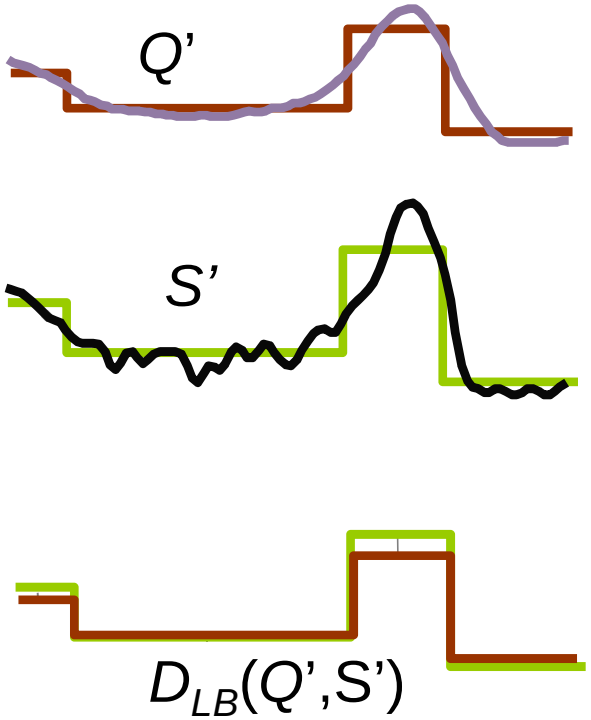
What is lower bounding?

Exact (Euclidean) distance D(Q,S)



$$D(Q,S) \equiv \sqrt{\sum_{i=1}^{n} (q_i - s_i)^2}$$

Lower bounding distance $D_{LB}(Q,S)$



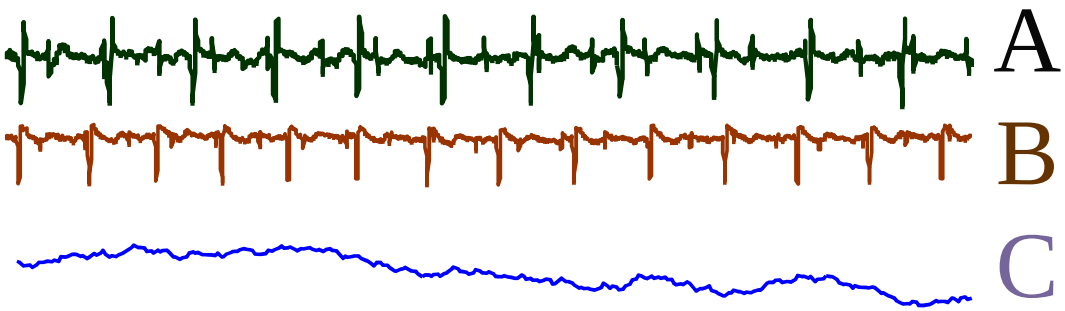
$$D_{LB}(Q',S') \equiv \sqrt{\sum_{i=1}^{M} (sr_i - sr_{i-1})(qv_i - sv_i)^2}$$

Lower bounding means that for all Q and S, we have…

$$D_{LB}(Q',S') \leq D(Q,S)$$

Structure and model based similarity

- Extract *global* features from the time series, create a feature vector, and use these feature vectors to measure similarity and/or classify



Feature \ Time Series	A	B	C
Max Value	11	12	19
Autocorrelation	0.2	0.3	0.5
Zero Crossings	98	82	13
…	…	…	…

ARIMA Models and Forecasting

- If we can describe the way the points in the series are related to each other (the autocorrelations), then we can describe the series using the relationships that we've found
- AutoRegressive Integrated Moving Average Models (ARIMA) are mathematical models of the autocorrelation in a time series
- One way to describe time series

Autocorrelation

- The major statistical tool for ARIMA models is the sample autocorrelation coefficient

$$r_k = \frac{\sum_{t=k+1}^{n}(Y_t - \bar{Y})(Y_{t-k} - \bar{Y})}{\sum_{t=1}^{n}(Y_t - \bar{Y})^2}$$

Autocorrelations

- r_1 indicates how successive values of Y relate to each other,
- r_2 indicates how Y values two periods apart relate to each other,
- and so on.

Autoregressive Models

- The autoregressive process of order p is denoted AR(p), and defined by

$$Y_t = \phi_0 + \phi_1 Y_{t-1} + \phi_2 Y_{t-2} + \ldots + \phi_p Y_{t-p} + \varepsilon_t$$

$$Y_t = \sum_{r=1}^{p} \phi_r \, Y_{t-r} + w_t$$

where $\phi_1, \ldots, \phi_p$ are parameters to be estimated and $\{w_t\}$ white noise, a sequence of independent (or uncorrelated) random variables with mean 0 and variance σ^2

An AR(p) model is a **regression model with lagged values** of the dependent variable in the independent variable positions, hence the name **autoregressive** model.

Moving Average Models

- The moving average process of order *q*, denoted MA(q), includes lagged error terms *t*–1 to *t*–*q*, written as

$$Y_t = \mu + w_t - \sum_{r=1}^{q} \theta_r w_{t-r}$$

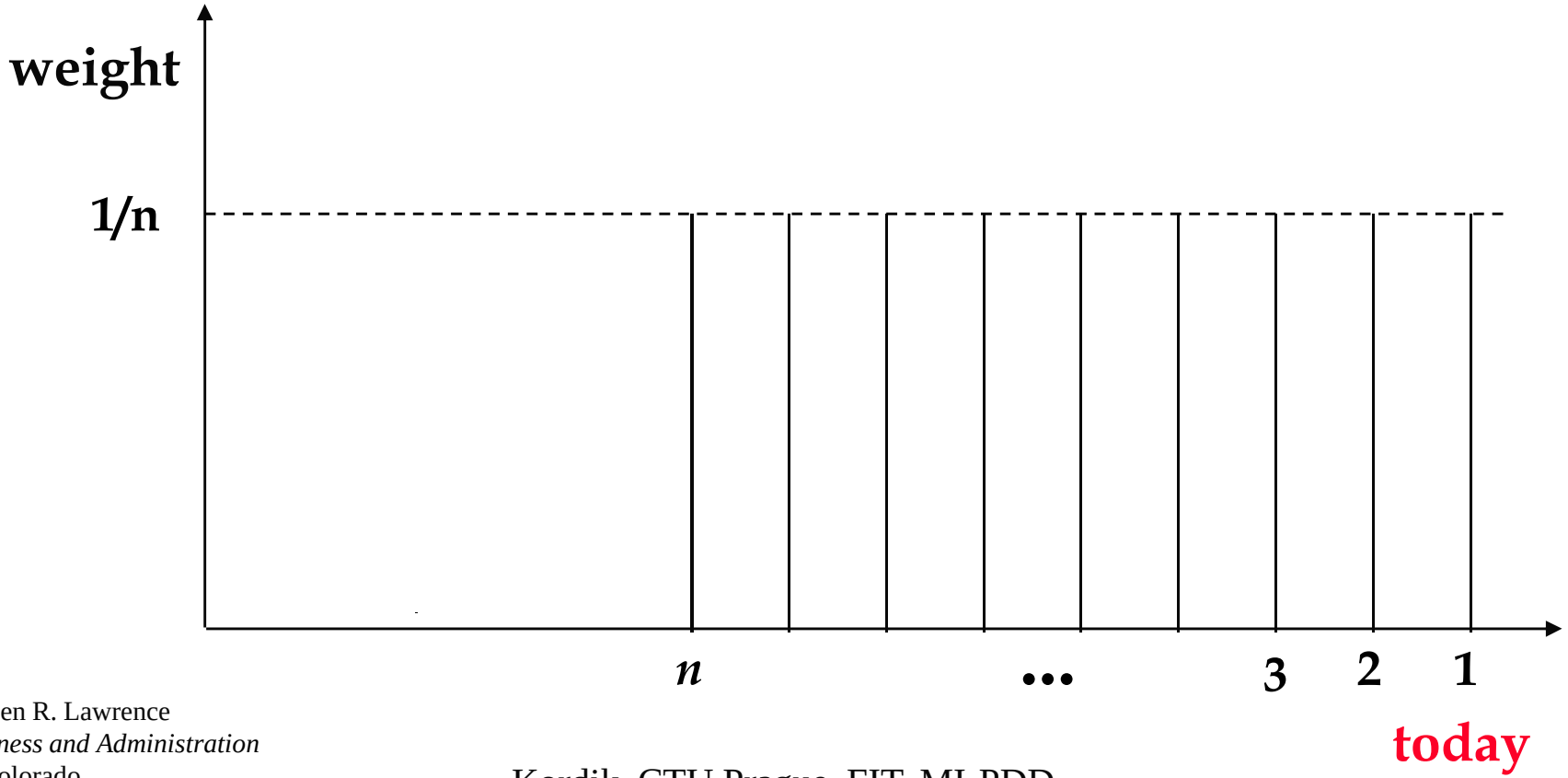
The term Moving Average is historical and should not be confused with the moving average smoothing procedures.

where θ_1 , θ_2 , … , θ_q are the MA parameters and w_t is white noise

An MA(q) model is a regression model with the dependent variable, **Y_t, depending on previous values of the errors** rather than on the variable itself.

Simple Moving Average

- Include *n* most recent observations
- Weight equally
- Ignore older observations



Professor Stephen R. Lawrence
College of Business and Administration
University of Colorado
Boulder, CO 80309-0419

Simple Moving Average

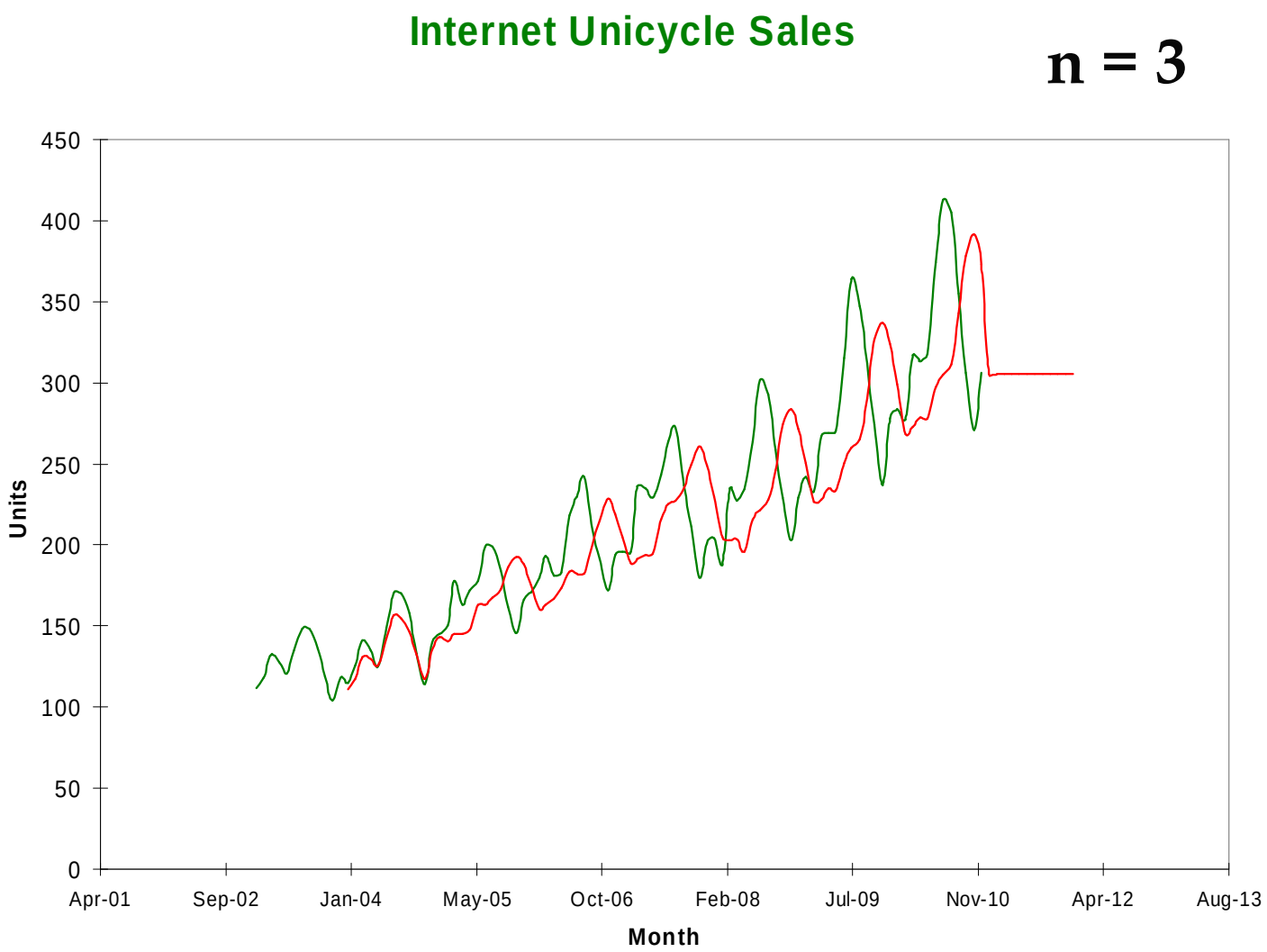
- Forecast F_t is average of n previous observations or actuals D_t:

$$F_{t+1} = \frac{1}{n}(D_t + D_{t-1} + \cdots + D_{t+1-n})$$

$$F_{t+1} = \frac{1}{n}\sum_{i=t+1-n}^{t} D_i$$

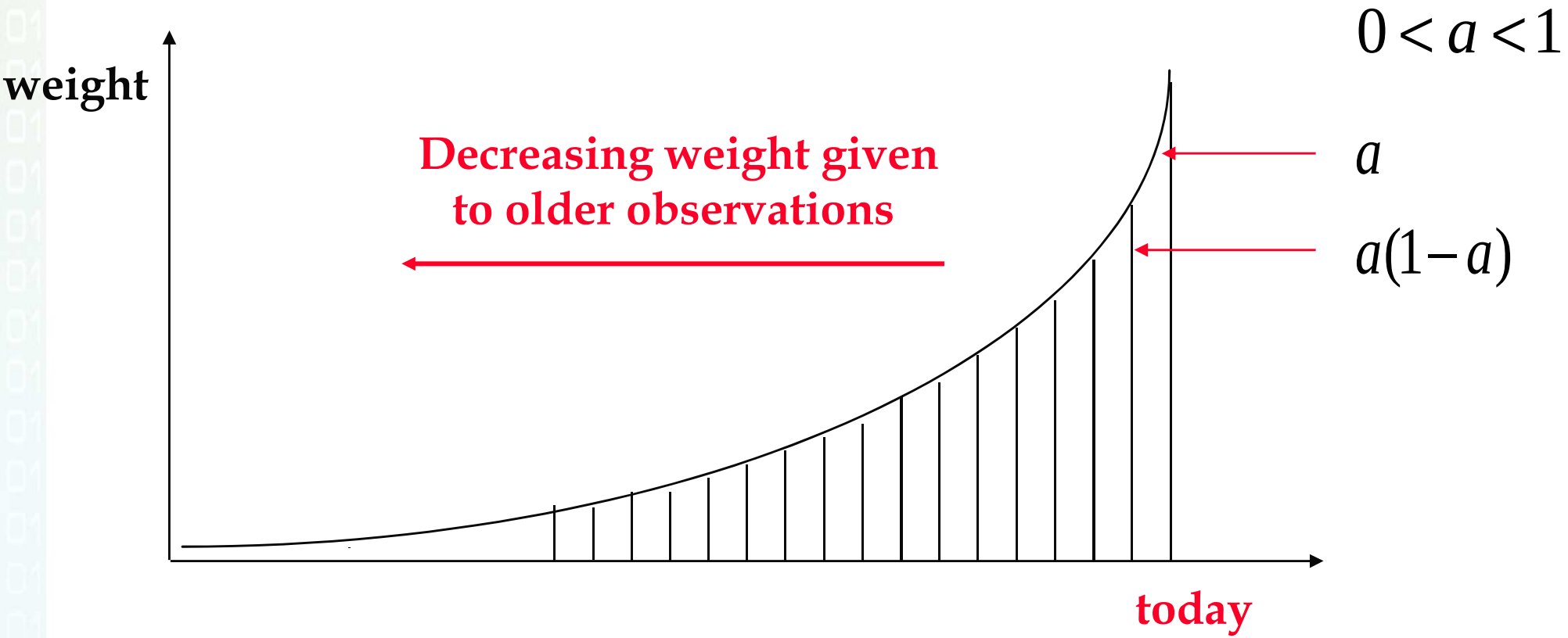
- Note that the n past observations are equally weighted.
- Issues with moving average forecasts:
  - All n past observations treated equally;
  - Observations older than n are not included at all;
  - Requires that n past observations be retained;
  - Problem when 1000's of items are being forecast.

Moving Average



Exponential Smoothing, idea

- Include all past observations
- Weight recent observations much more heavily than very old observations:



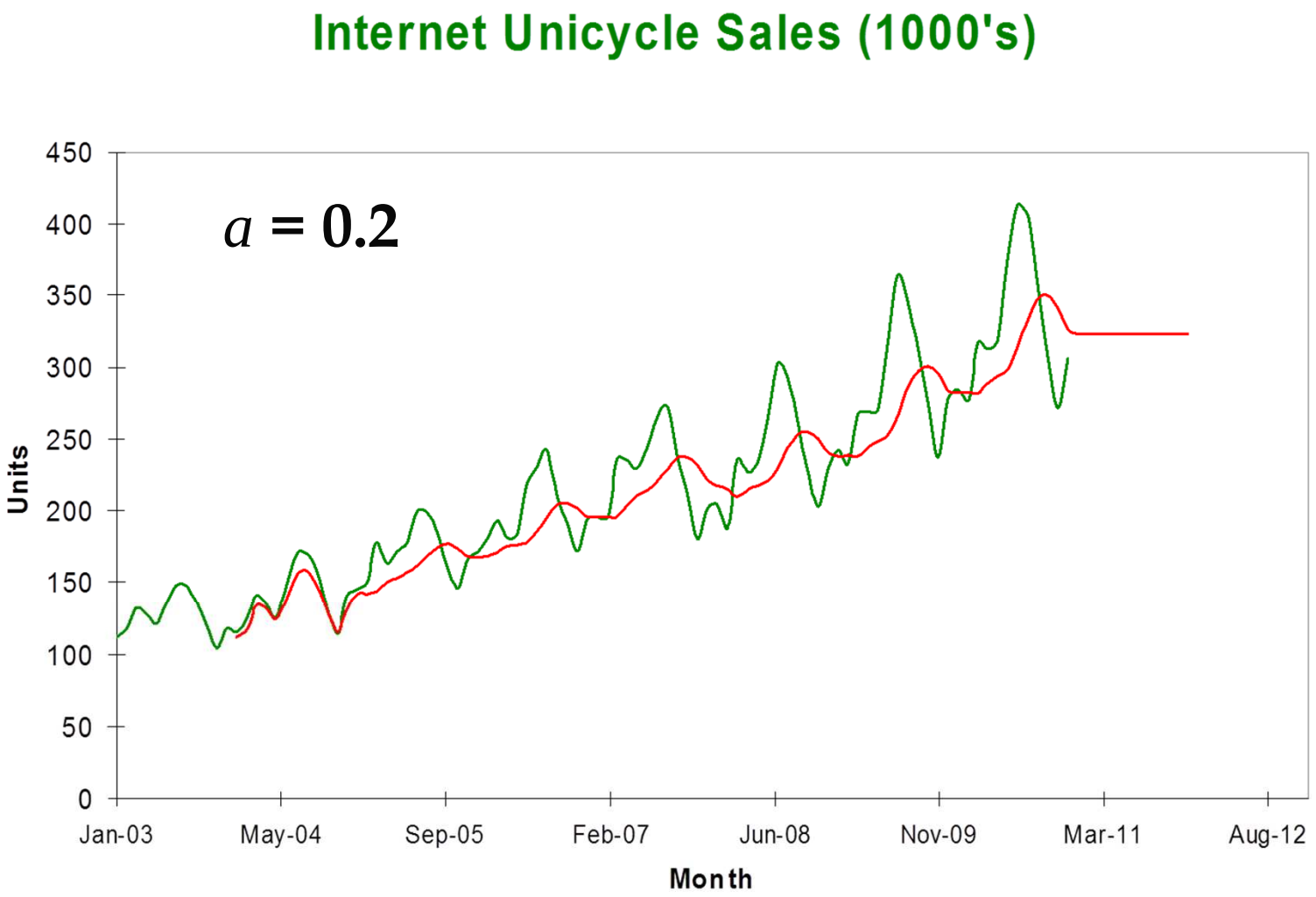
Exponential Smoothing, math

$$F_t = aD_t + a(1-a)D_{t-1} + a(1-a)^2 D_{t-2} + \cdots$$

$$F_t = aD_t + (1-a)F_{t-1}$$

- Thus, new forecast is weighted sum of old forecast and actual demand
- Notes:
  - Only 2 values (D_t and F_{t-1}) are required, compared with n for moving average
  - Parameter a determined empirically (whatever works best)
  - Rule of thumb: $a < 0.5$
  - Typically, $a = 0.2$ or $a = 0.3$ work well

Exponential Smoothing, example



Time honored linear models

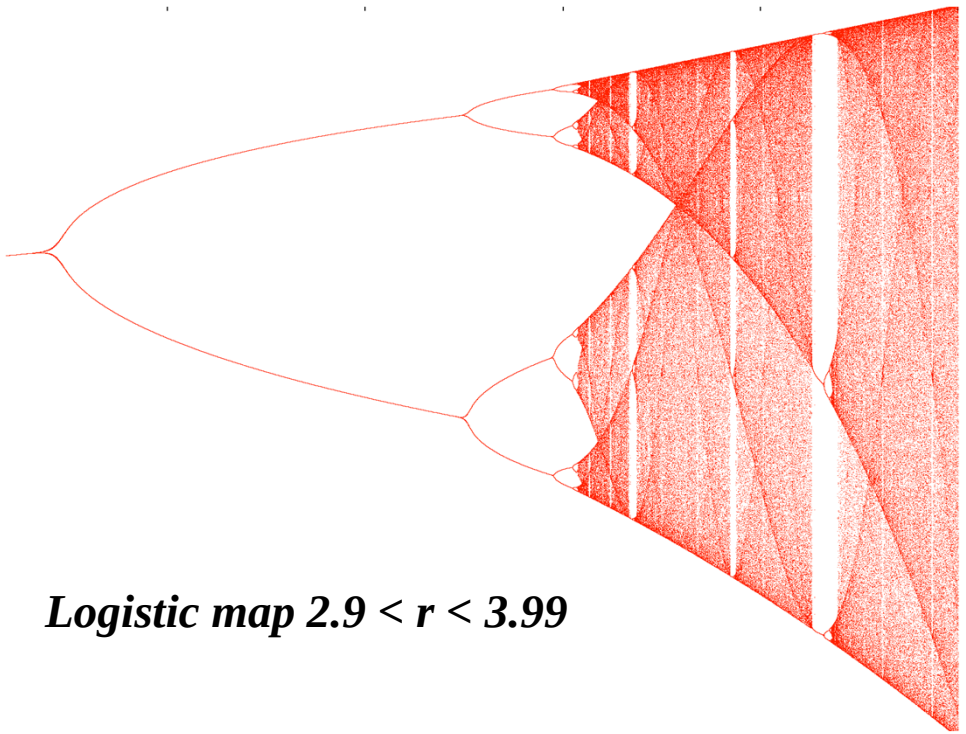
$$y[t+1] = \sum_{i=0}^{N_{AR}} a_i \cdot y[t-i] + \sum_{j=0}^{N_{MA}} b_j \cdot x[t-i]$$

- Auto Regressive Moving Average (**ARMA**)
- Many linear estimation techniques based on Least Squares, or Least Mean Squares
- Power spectra, and Autocorrelation characterize such linear systems
- Randomness comes ***only*** from forcing function $x(t)$

Simple nonlinear systems can exhibit chaotic behavior

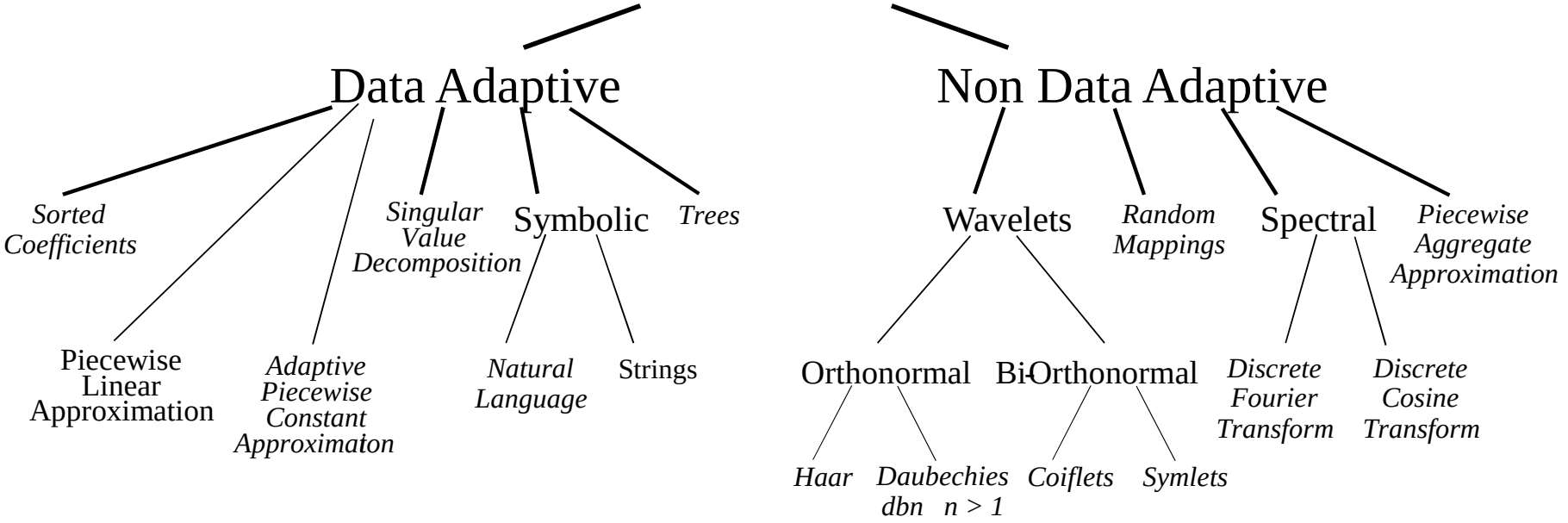
$$x[t+1] = r \cdot x[t](1 - x[t])$$

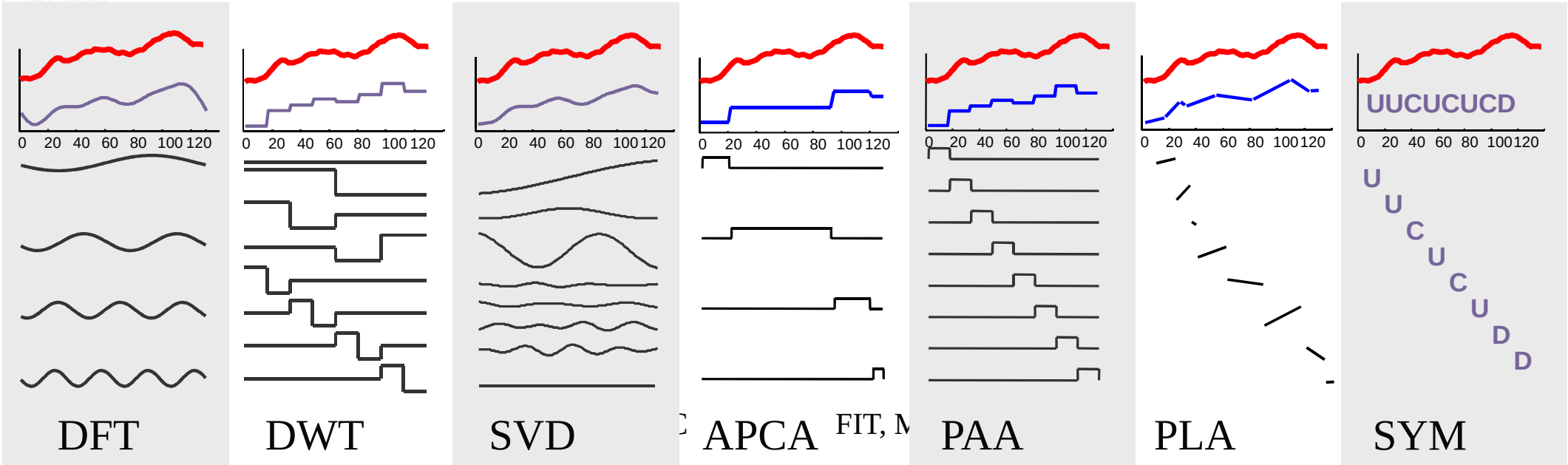
- Spectrum, autocorrelation, characterize ***linear*** systems, not these
- Deterministic chaos looks random to linear analysis methods
- Logistic map is an early example (*Elam 1957*).



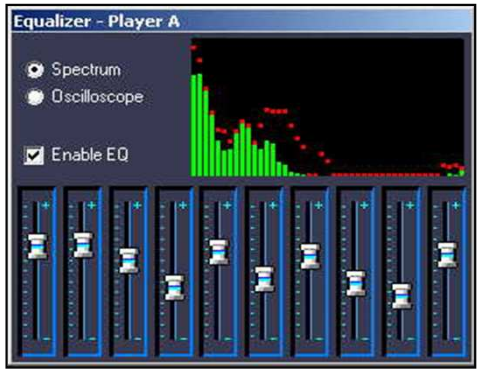
Logistic map 2.9 < r < 3.99

Time Series Representations





Spectrum and Shape Descriptors



Centroid
Rolloff
Flux
Bandwidth
Moments
....

=

M

F

Centroid

Feature Space

Feature vector

Fourier transform

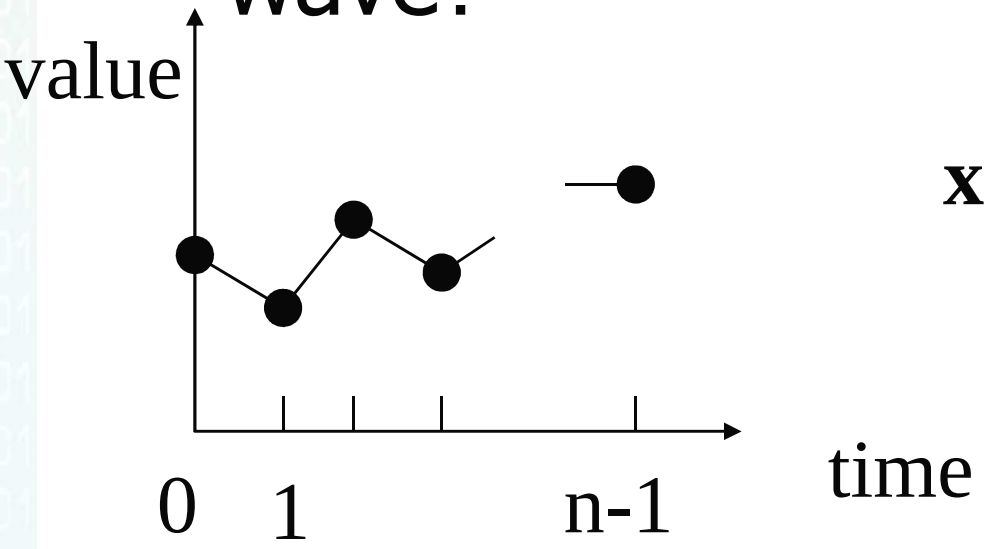
- DFT (Discrete Fourier Transform)
- Transform the data from the time domain to the frequency domain
- Useful for signals with periodicities
  - sales patterns follow seasons;
  - economy follows 50-year cycle (or 10?)
  - temperature follows daily and yearly cycles ...

How does it work?

Based on Slides by D. Gunopulos (UCR)

Decomposes signal to a sum of sine (and cosine) waves.

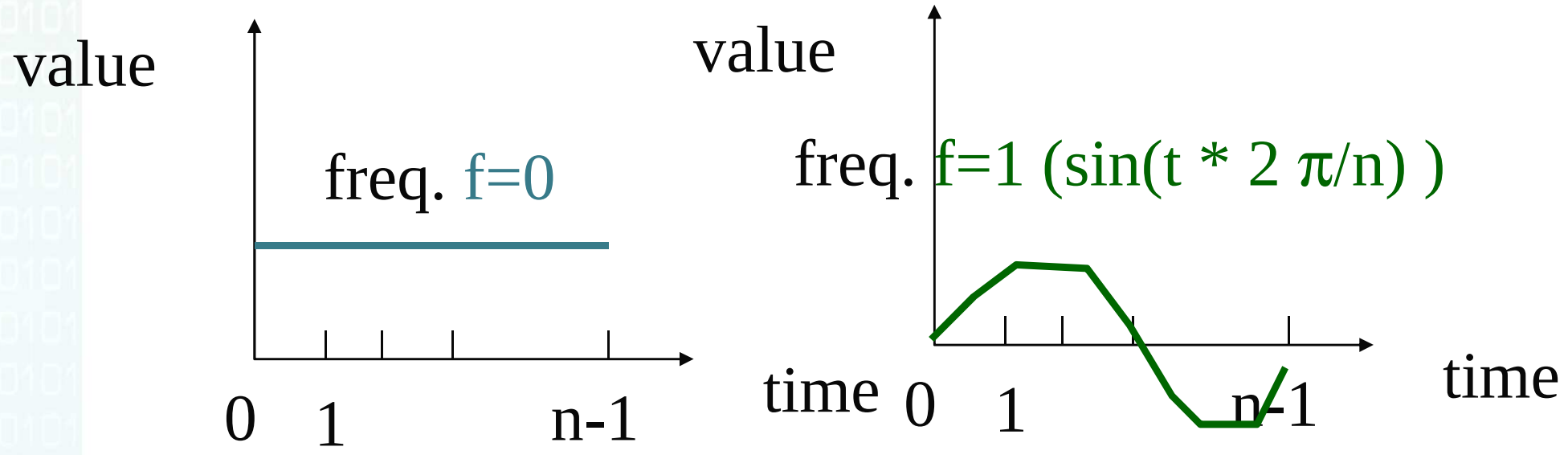
Q:How to assess 'similarity' of **x** with a wave?



$$\mathbf{x} = \{x_0, x_1, \ldots x_{n-1}\}$$

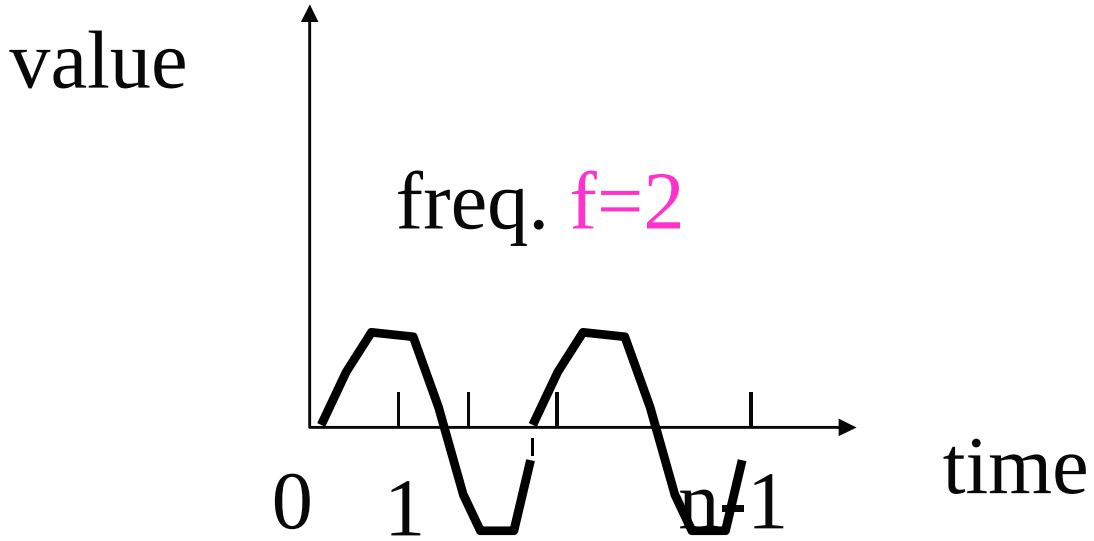
How does it work?

A: consider the waves with frequency 0, 1, ...; use the inner-product (~cosine similarity)



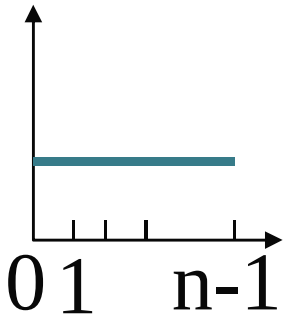
How does it work?

A: consider the waves with frequency 0, 1, ...; use the inner-product (~cosine similarity)

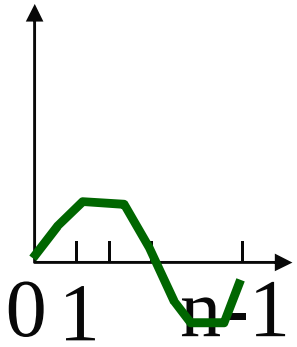


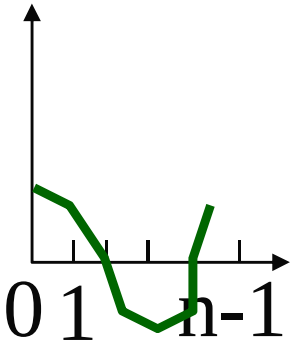
How does it work?

'basis' functions



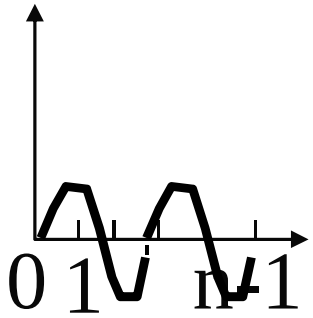
sine, freq =1

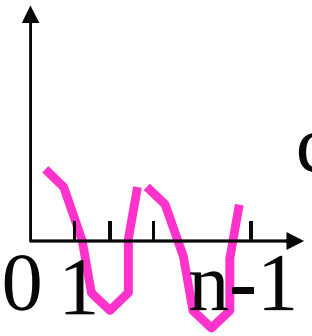




cosine, f=1

sine, freq = 2





cosine, f=2

How does it work?

- Basis functions are actually n-dim vectors, **orthogonal** to each other
- 'similarity' of **x** with each of them: inner product
- DFT: ~ all the similarities of **x** with the basis functions

How does it work?

Since $e^{jf} = cos(f) + j\,sin(f)$ ($j=sqrt(-1)$),

we finally have:

DFT: definition

- Discrete Fourier Transform (n-point):

$$X_f = 1/\sqrt{n} \sum_{t=0}^{n-1} x_t * \exp(-j2\pi tf / n)$$

$$(j = \sqrt{-1})$$

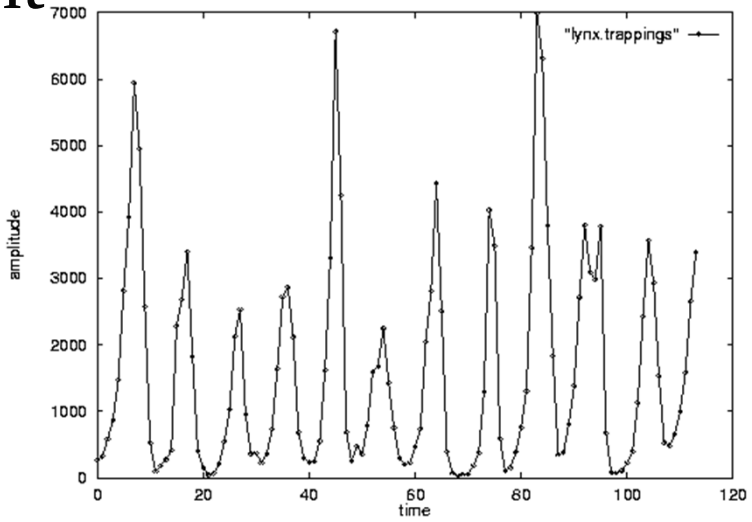
inverse DFT

$$x_t = 1/\sqrt{n} \sum_{t=0}^{n-1} X_f * \exp(+j2\pi tf / n)$$

DFT: Amplitude spectrum

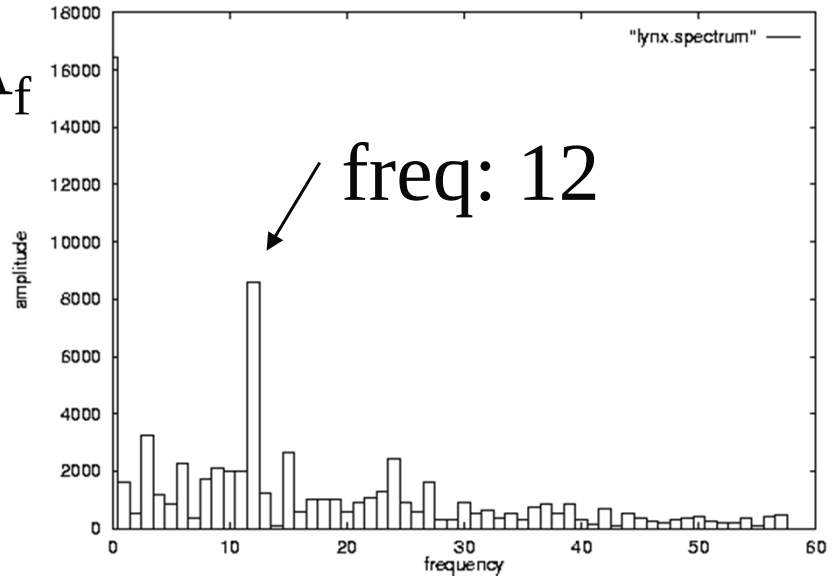
- Amplitude $A_f^2 = \mathrm{Re}^2(X_f) + \mathrm{Im}^2(X_f)$
- Intuition: strength of frequency '*f*'

count



time

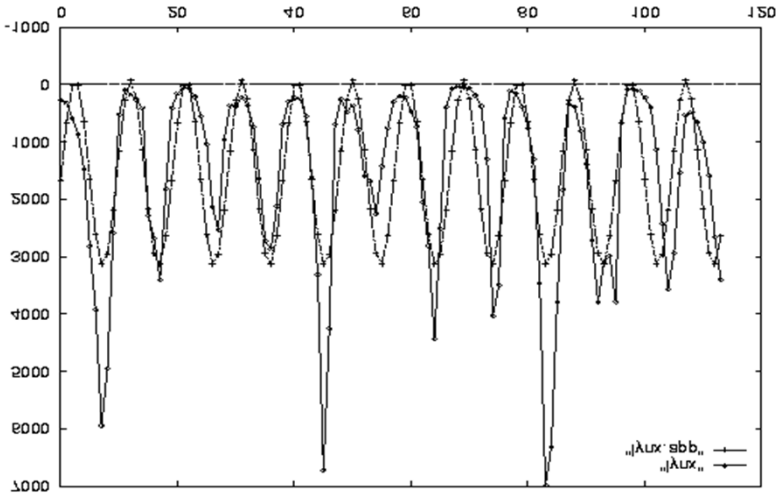
A_f

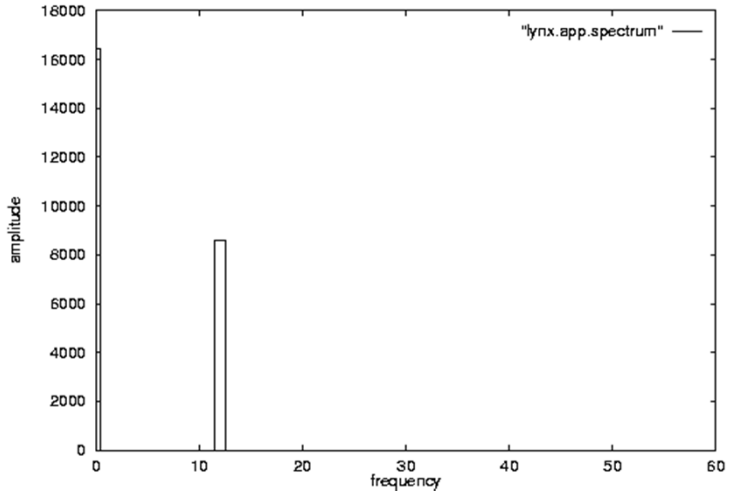


freq. *f*

DFT: Amplitude spectrum

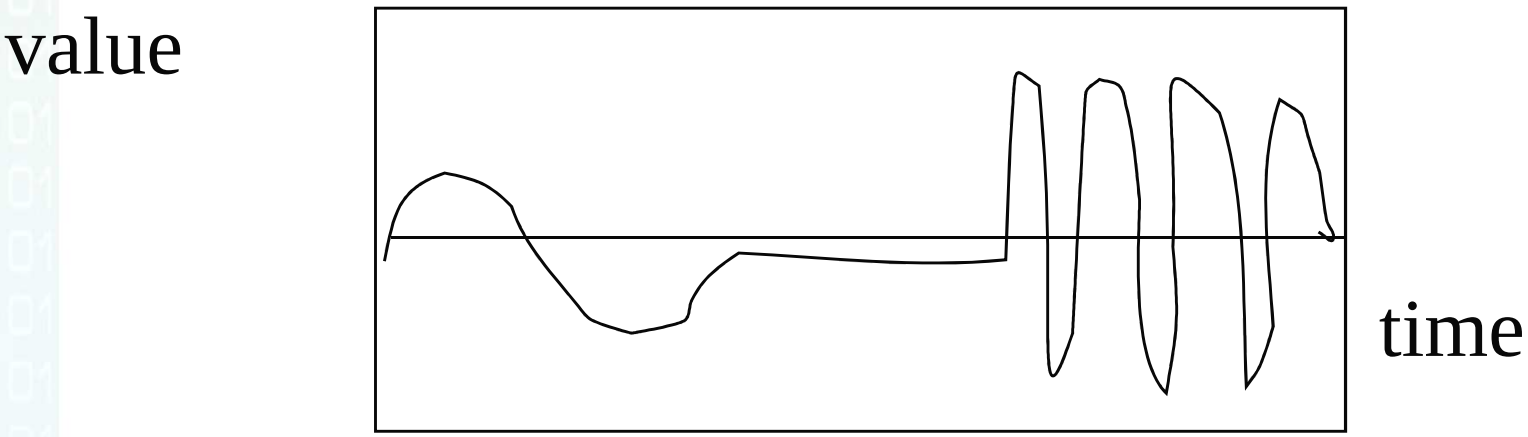
- excellent approximation, with only 2 frequencies!





Varying frequencies

- DFT is great - but, how about compressing opera? (baritone, silence, soprano?)



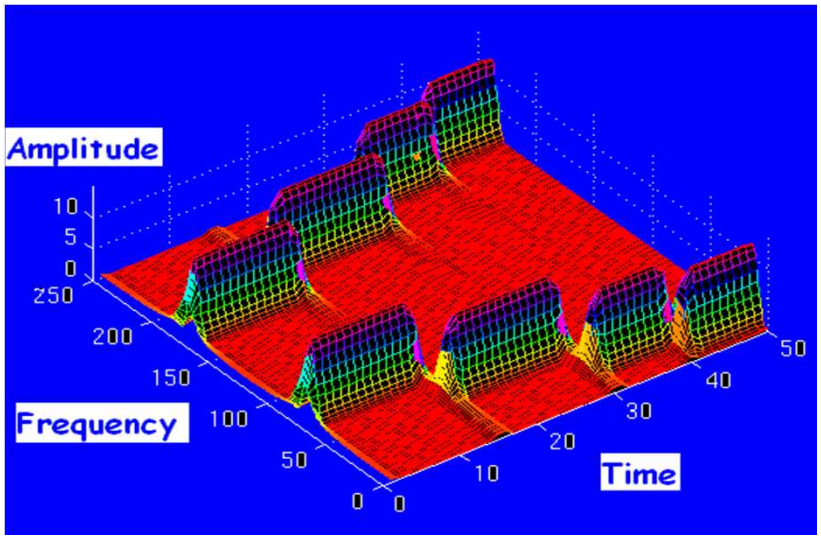
Short time Fourier transform (STFT)

- Solution#1: Short time Fourier transform
  - Apply DFT to sliding window
- But: how short should be the window?

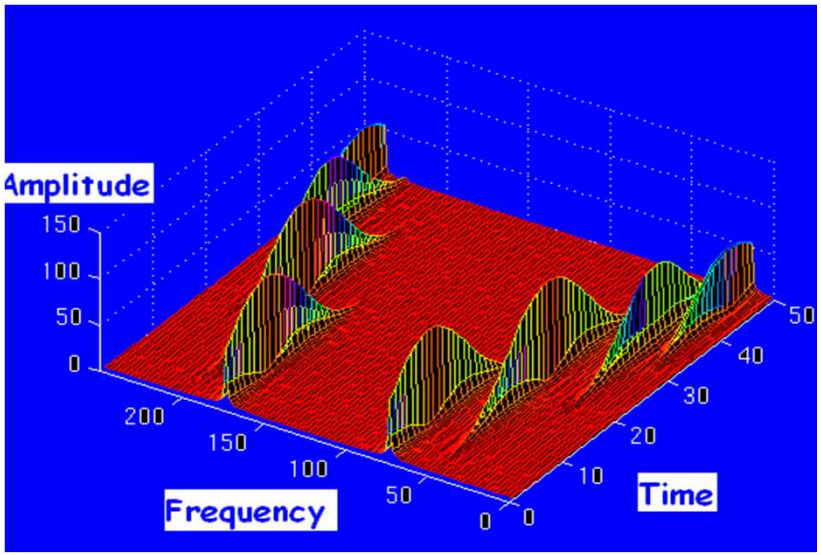
STFT drawbacks

- Unchanged Window
- Dilemma of Resolution
  - Narrow window -> poor frequency resolution
  - Wide window -> poor time resolution
- Heisenberg Uncertainty Principle
  - Cannot know what frequency exists at what time intervals

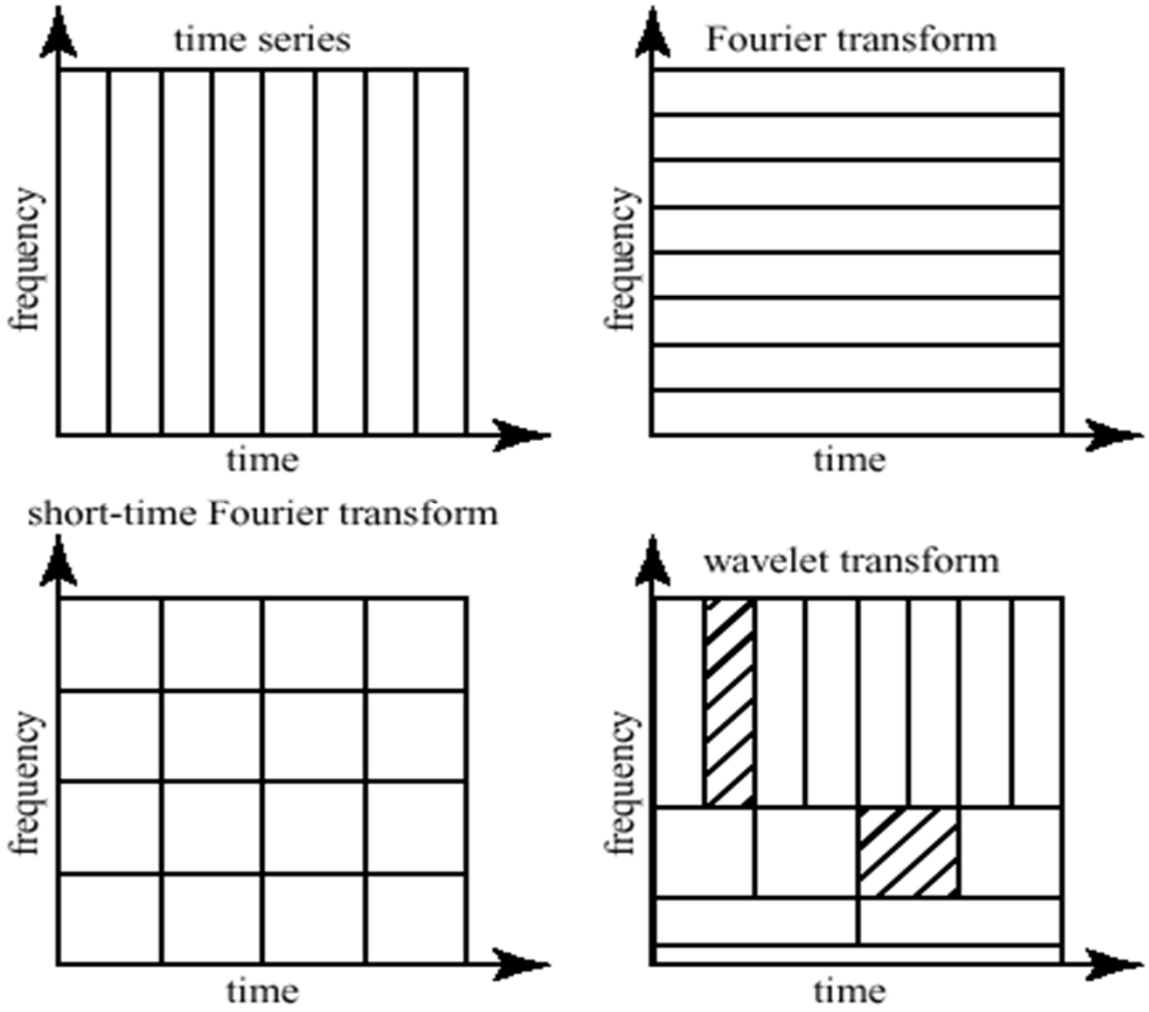
Via Narrow Window



Via Wide Window



Transformations

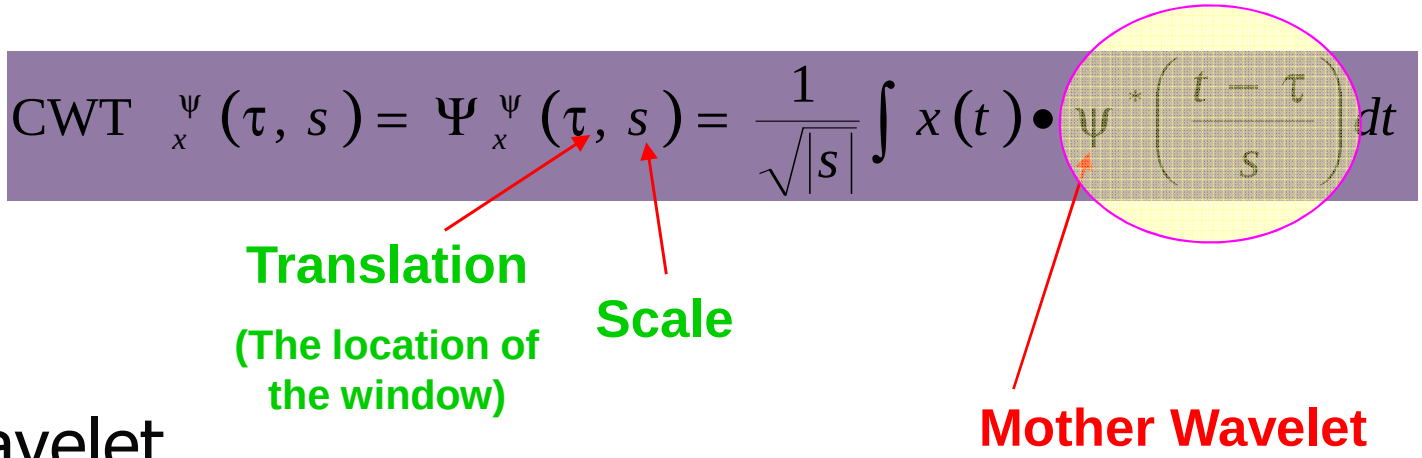


From http://www.cerm.unifi.it/EUcourse2001/Gunther_lecturenotes.pdf, p.10

Wavelet transform

- Split Up the Signal into a Bunch of Signals
- Representing the Same Signal, but all Corresponding to Different Frequency Bands
- Only Providing What Frequency Bands Exists at What Time Intervals

DEFINITION OF CONTINUOUS WAVELET TRANSFORM



- Wavelet
  - Small wave
  - Means the window function is of finite length
- Mother Wavelet
  - A prototype for generating the other window functions
  - All the used windows are its dilated or compressed and shifted versions

SCALE

- Scale
  - S>1: dilate the signal
  - S<1: compress the signal
- Low Frequency -> High Scale -> Non-detailed Global View of Signal -> Span Entire Signal
- High Frequency -> Low Scale -> Detailed View Last in Short Time
- Only Limited Interval of Scales is Necessary

CWT computation

$$\mathrm{CWT}_x^{\Psi}(\tau, s) = \Psi_x^{\Psi}(\tau, s) = \frac{1}{\sqrt{|s|}} \int x(t) \bullet \psi^* \left(\frac{t-\tau}{s} \right) dt$$

Step 1: The wavelet is placed at the beginning of the signal, and set s=1 (the most compressed wavelet);

Step 2: The wavelet function at scale "1" is multiplied by the signal, and integrated over all times;

Step 3: Shift the wavelet to $t = \tau$, and get the transform value at $t = \tau$ and $s = 1$;

Step 4: Repeat the procedure until the wavelet reaches the end of the signal;

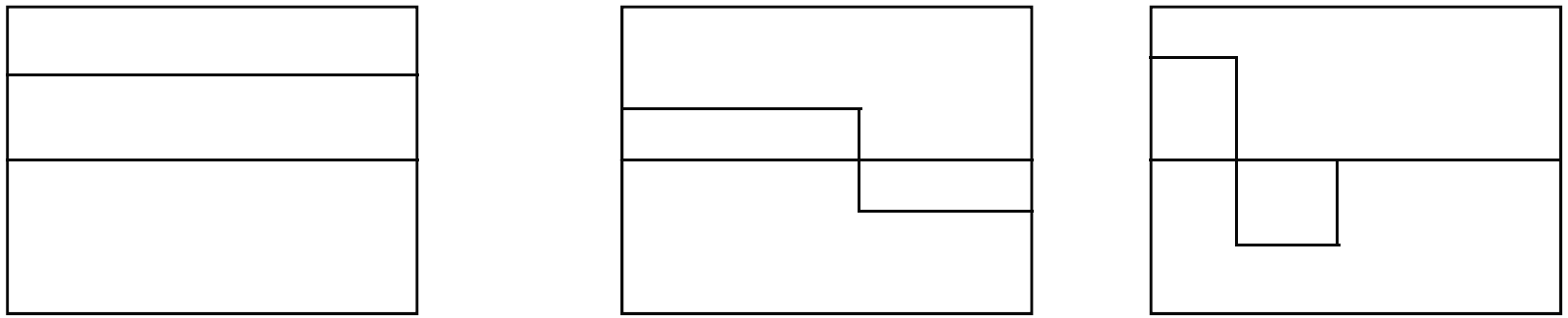
Step 5: Scale s is increased by a sufficiently small value, the above procedure is repeated for all *s;*

Step 6: Each computation for a given *s* fills the single row of the time-scale plane;

Step 7: CWT is obtained if all s are calculated.

Haar Wavelets

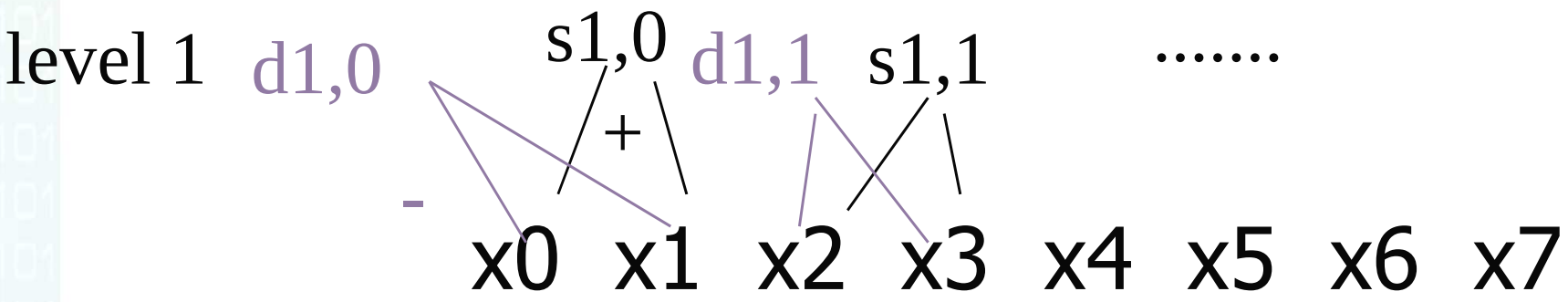
- subtract sum of left half from right half
- repeat recursively for quarters, eightths ...



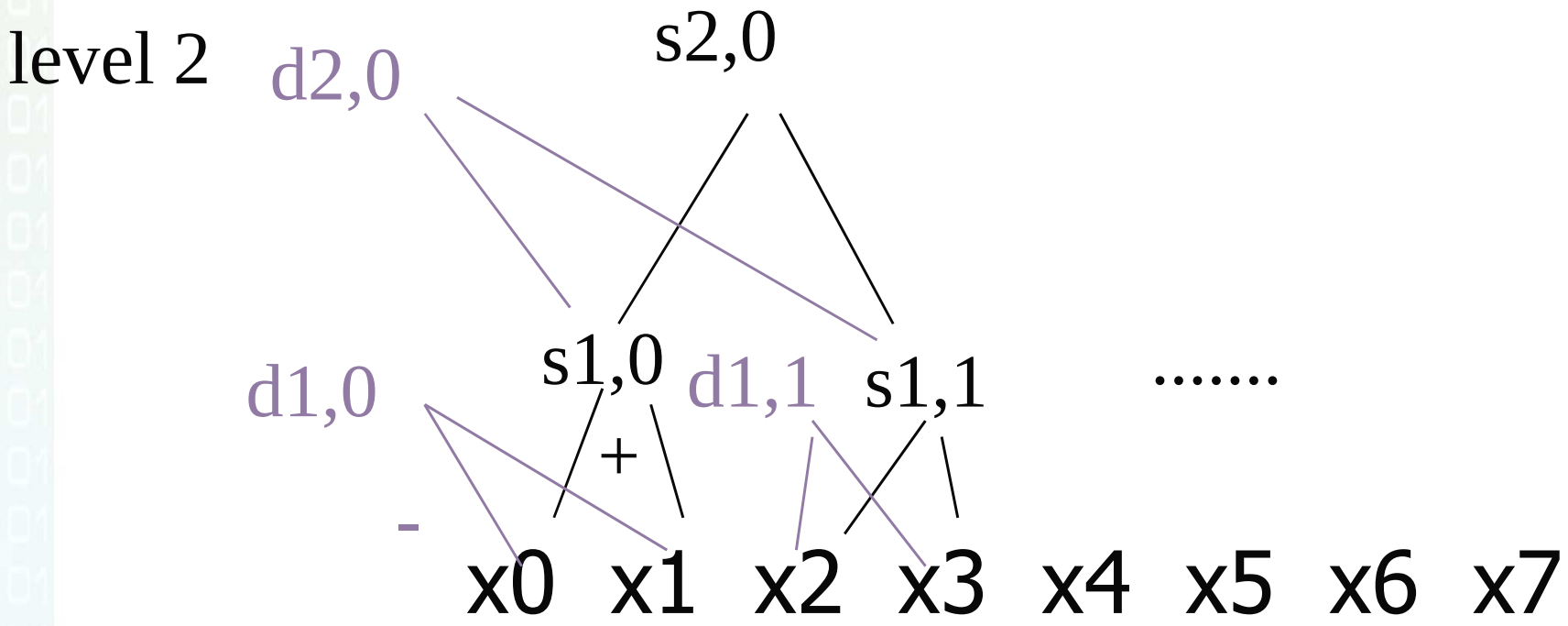
Wavelets - construction

x0 x1 x2 x3 x4 x5 x6 x7

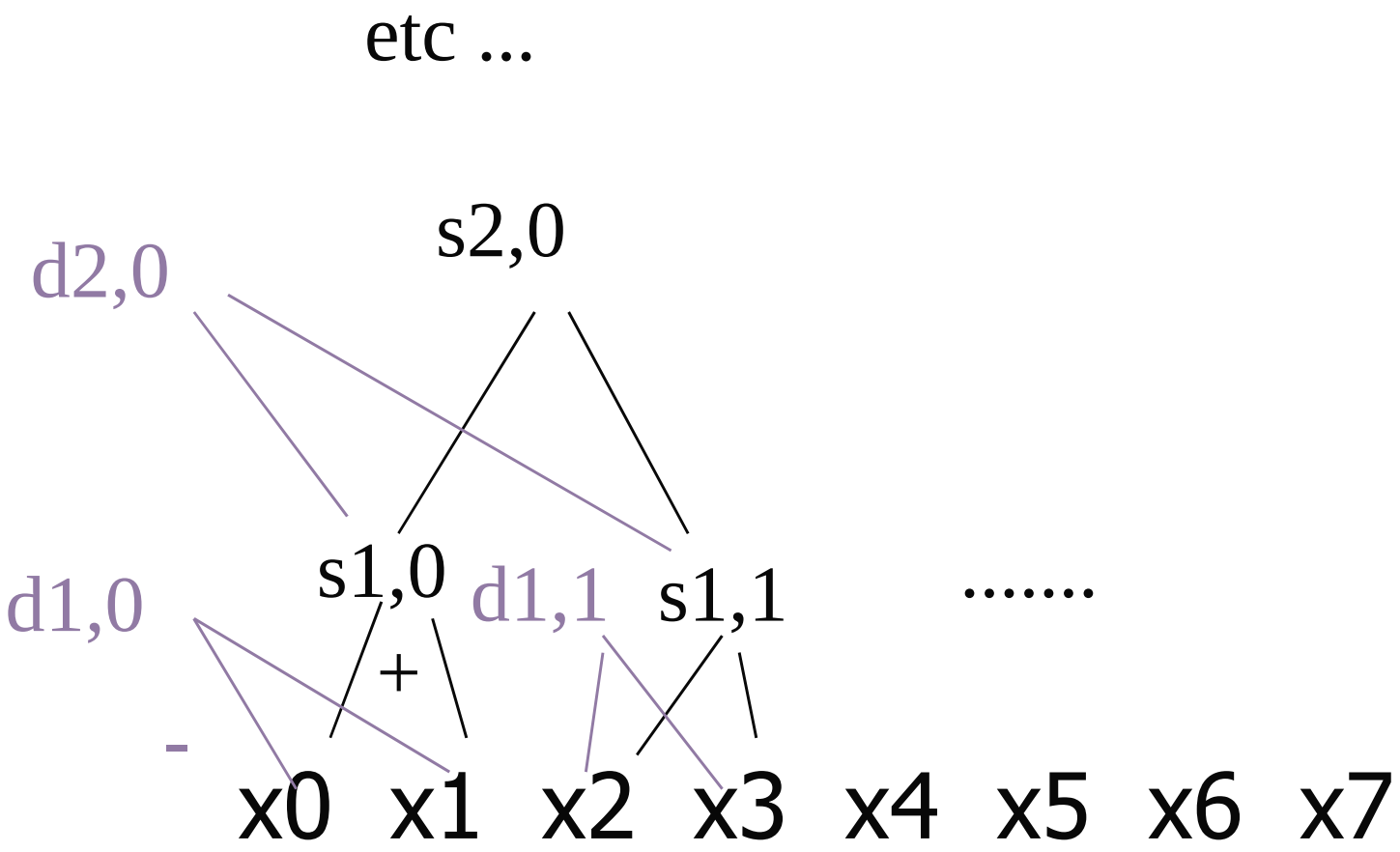
Wavelets - construction



Wavelets - construction



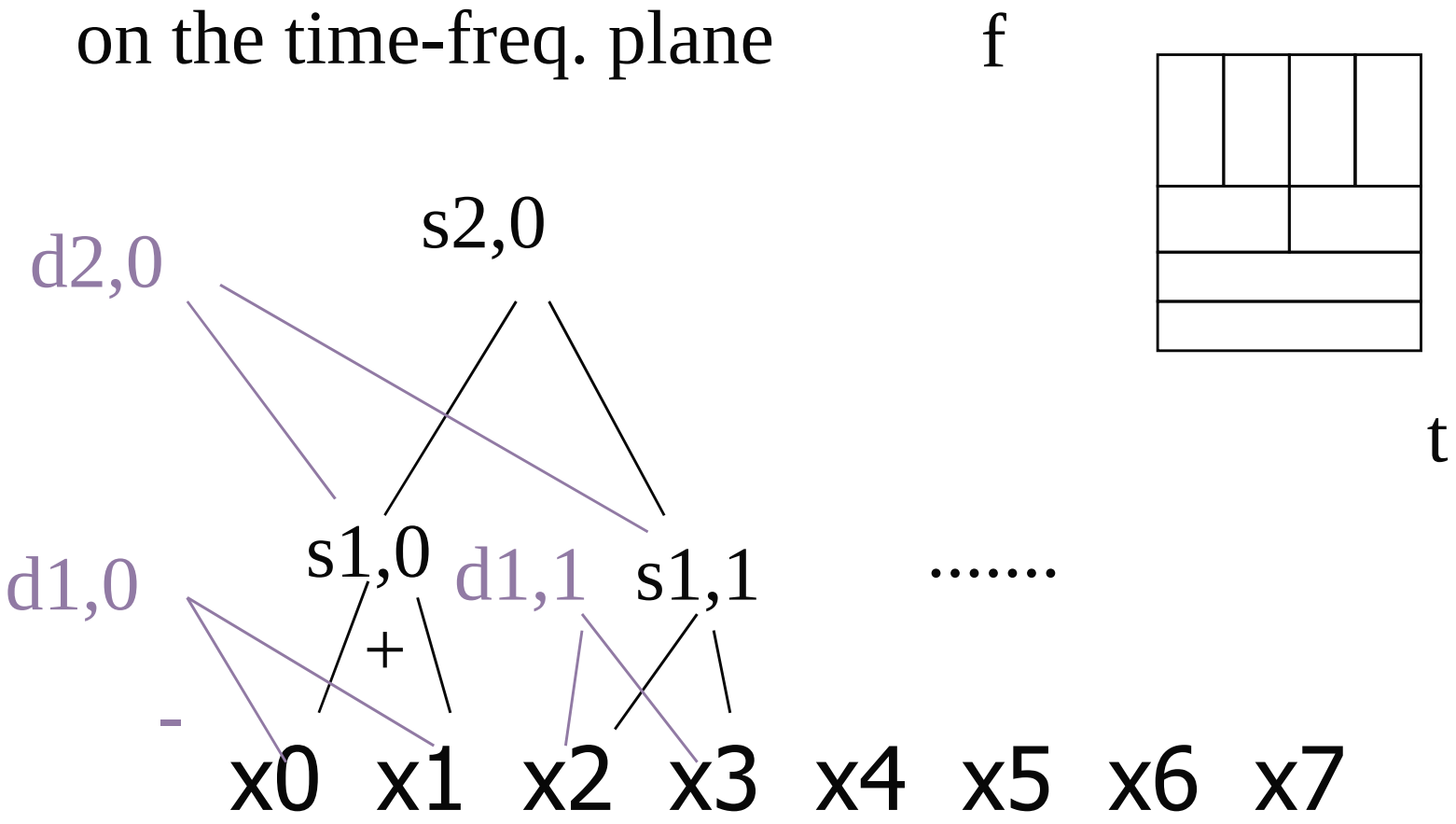
Wavelets - construction



Wavelets - construction

Q: map each coefficient

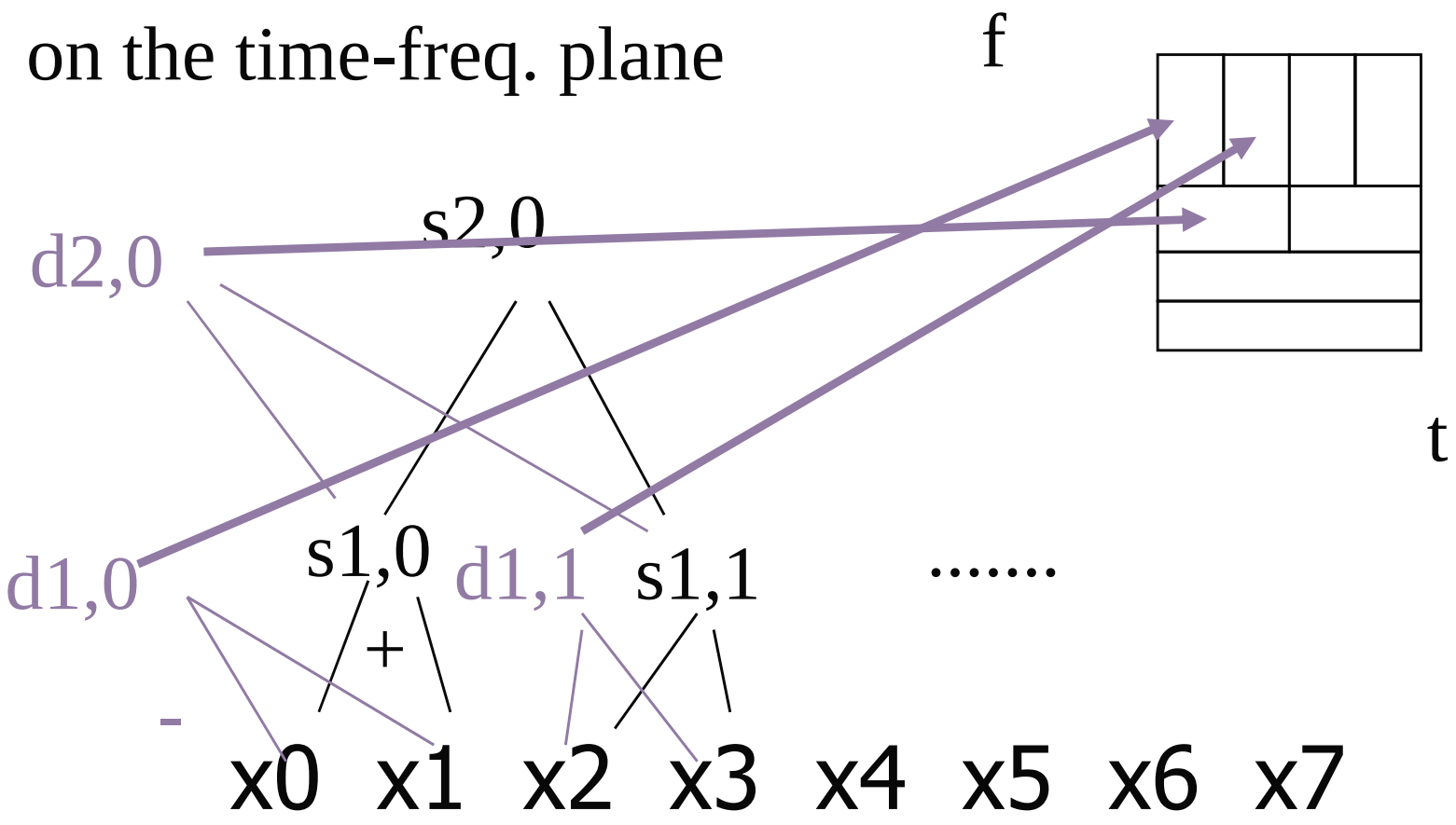
on the time-freq. plane



Wavelets - construction

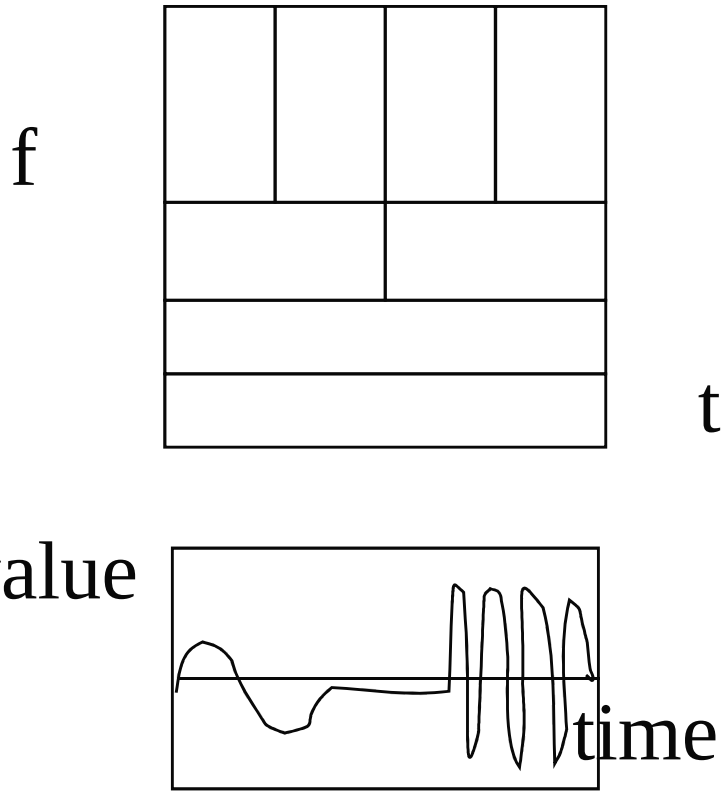
Q: map each coefficient

on the time-freq. plane



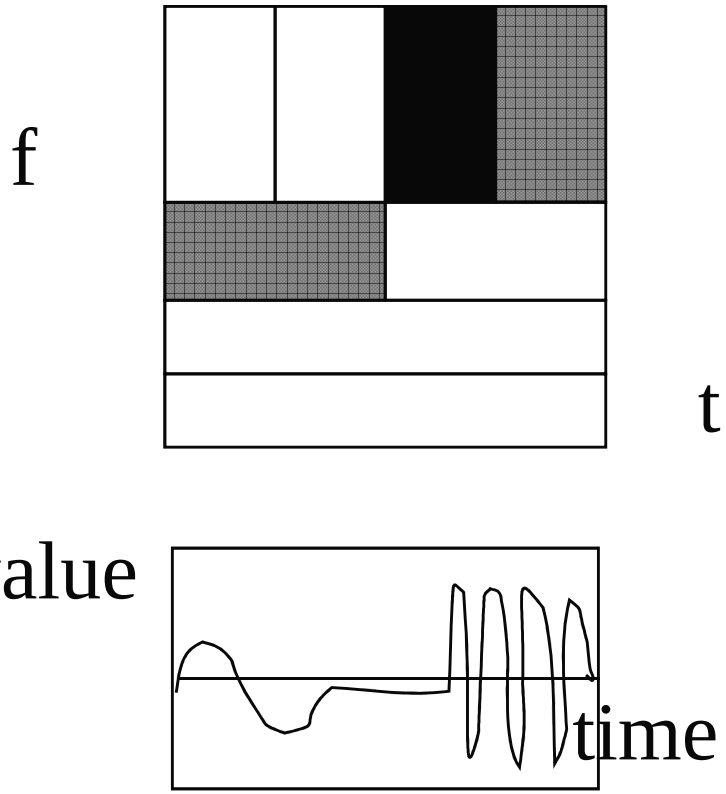
Wavelets - Drill:

- Q: baritone/silence/soprano - DWT?



Wavelets - Drill:

- Q: baritone/soprano - DWT?



Wavelets - construction

Observation1:

- ‘+’ can be some weighted addition
- ‘-’ is the corresponding weighted difference (‘Quadrature mirror filters’)

Observation2: unlike DFT/DCT,

- there are *many* wavelet bases: Haar, Daubechies-4, Daubechies-6, ...

Advantages of Wavelets

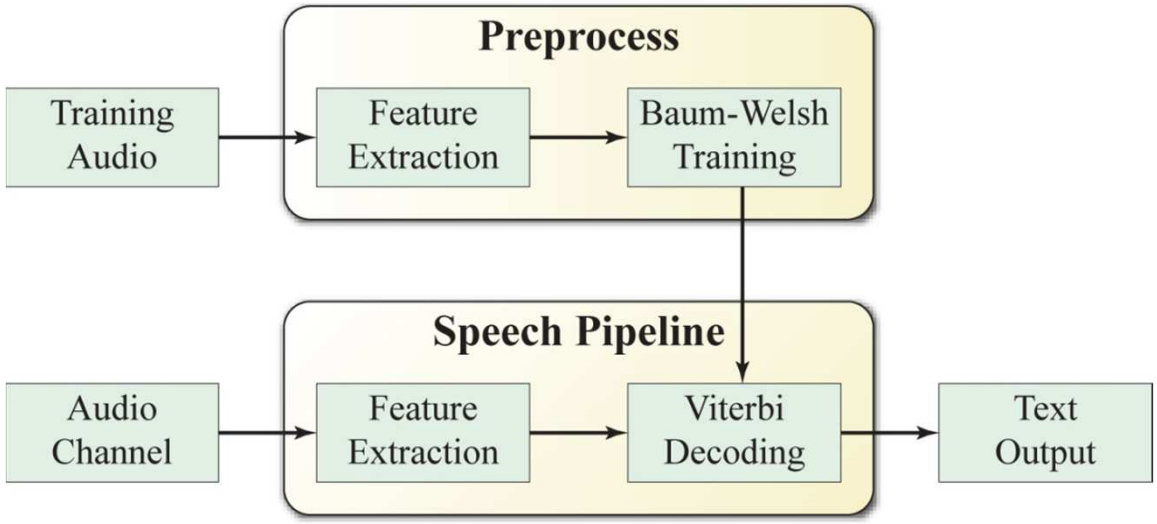
- Better compression (better RMSE with same number of coefficients)
- closely related to the processing of the mammalian eye and ear
- Good for progressive transmission
- handle spikes well
- usually, fast to compute (O(*n*)!)

Feature space

- Keep the d most “important” wavelets coefficients
- Normalize and keep the largest

Speech extraction and recognition

- System Components
  - **Feature Extraction**
    - Convert audio stream into feature vectors
  - Baum-Welsch Training
    - Calibrate HMMs using known data.
  - **Viterbi Decoding**
    - Recognition using HMMs.



How it is implemented at IBM Cell processors?

- Feature Extraction
  - Pipeline (12 stages):
    - Window Extraction
    - Zero Mean
    - Energy Computation
    - Preemphasis Filter
    - Hamming Window
    - Spectrum Computation (FFT)
    - Mel Frequency Computation
    - Cepstrum Computation
    - Discrete Cosine Transform
    - Lifter (Cepstral Filter)
    - Cepstrum Energy Normalization
    - First and Second Temporal Derivatives
  - → **Convert 25 ms frame of speech (with 15 ms overlap) into 39 coefficients (MFCCs).**
- Viterbi Decoding
  - Evaluate likelihood that a HMM produced a particular sequence of frames.
  - Dynamic Programming
  - HMM parameters
    - Left-Right model (only forward and self transitions; transition matrix is upper bi-diagonal)
    - Gaussian Mixture Model (PDFs are approximated by a set of Gaussians)
  - → **Very expensive computation; <u>bottleneck in speech pipeline</u>**

References

[Goldin & Kanellakis 95] On Similarity Queries for Time-Series Data: Constraint Specification and Implementation. CP 1995: 137-153

[Jagadish et al. 95] Similarity-Based Queries. PODS 1995: 36-45

[Rafiei and Mendelson] Querying Time Series Data Based on Similarity. TKDE 12(5): 675-693 (2000)

[Berndt and Clifford 94] Using Dynamic Time Warping to Find Patterns in Time Series. KDD Workshop 1994: 359-370

[Ge and Smyth 2000] Deformable Markov model templates for time-series pattern matching. KDD 2000: 81-90

[Perng et al. 2000] Landmarks: a New Model for Similarity-based Pattern Querying in Time Series Databases. ICDE 2000: 33-42