# Knowledge Discovery on Time Series Data

**Juan Pedro Caraça-Valente** **november 2018**

# Summary

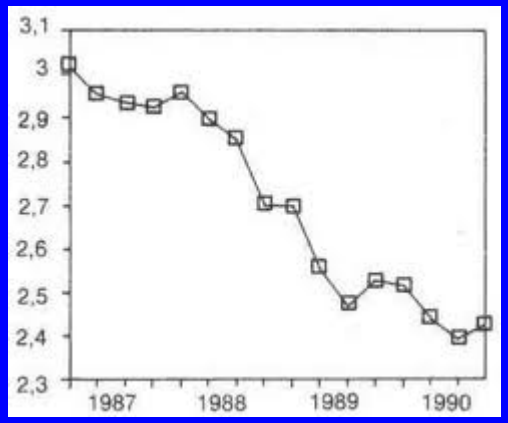
- 1. Introduction
- 2. Basic Techniques
- 3. Distance
- 4. Comparing Time Series
- 5. Search
- 6. Pattern Identification
- 7. Events
- 8. Temporal Abstraction
- 9. Conclusions

# 1. Introduction

# Introduction

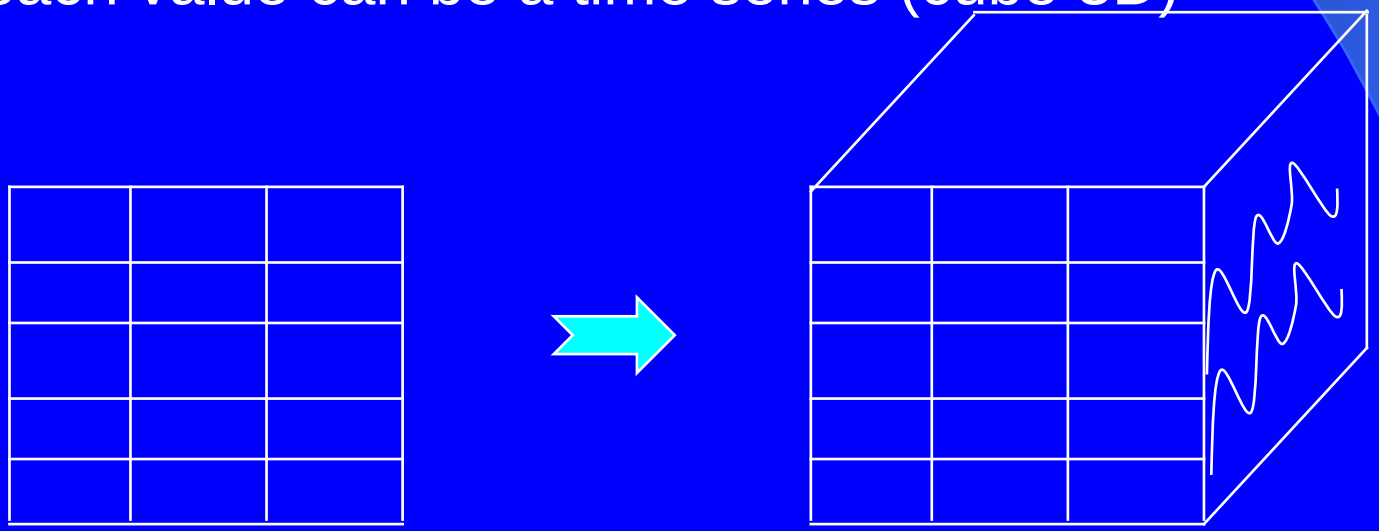
Time Series: *An ordered sequence of values of a variable at equally spaced time intervals*

Why KDD on Time Series: *Because data points taken over time may have an internal structure (such as autocorrelation, trend or seasonal variation) that should be accounted for*.




# Introduction

- KDD on Time Series : Opens a new dimension for Data Mining, objects are no longer a tuple of values for a given set of attributes (table 2D), but instead each value can be a time series (cube 3D)

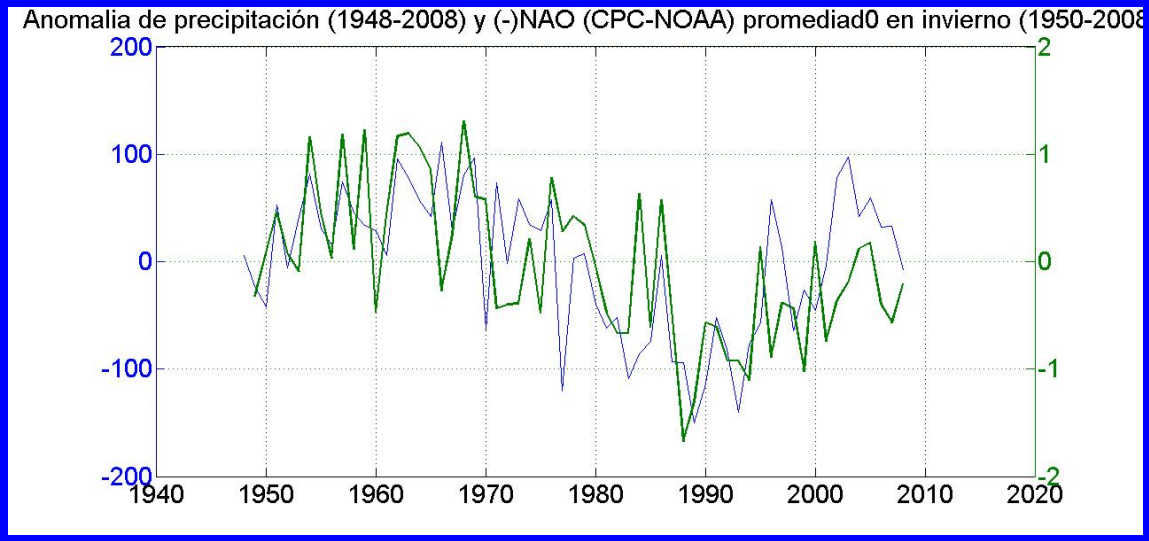


# Domains

- Finances

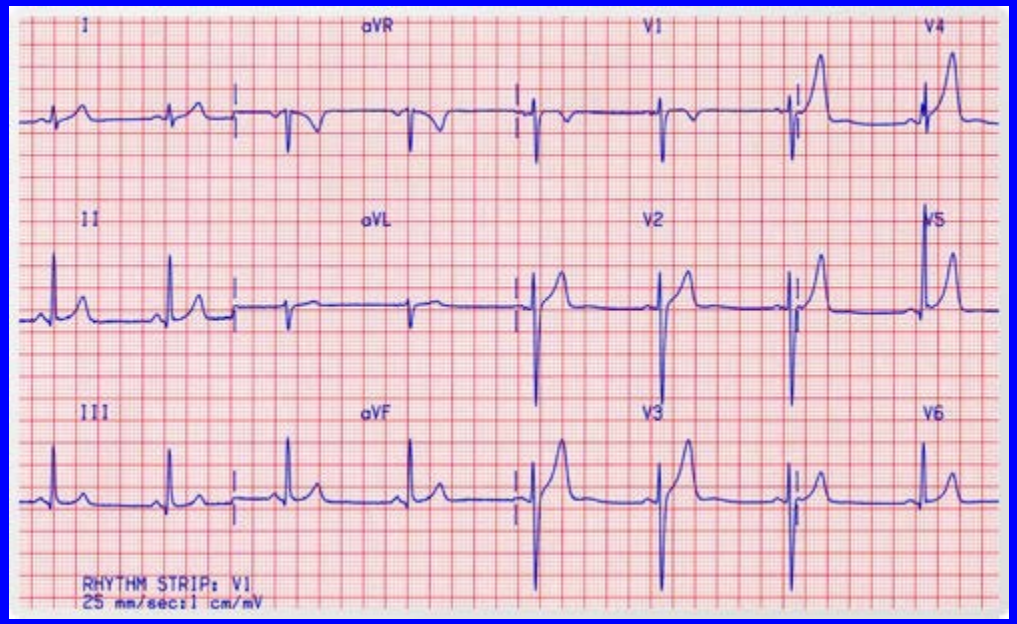
# Domains

- Metheorology



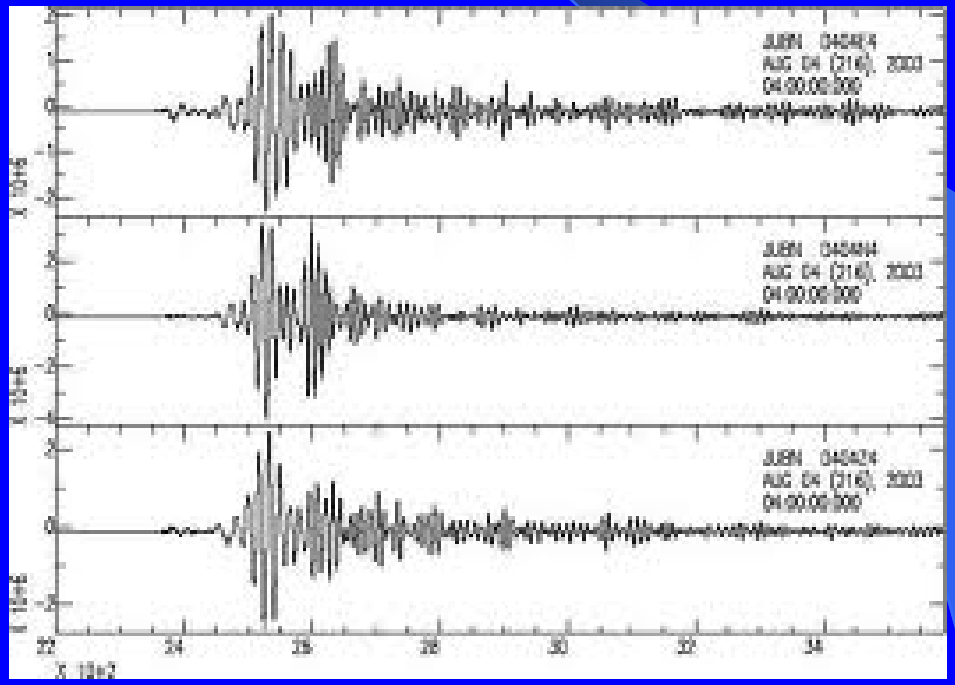
# Domains

- Medicine: Electrocardiogram



# Domains

- Sismology



- Others: Industry, Biology, Communications, etc.

# Aplications of DM to TS

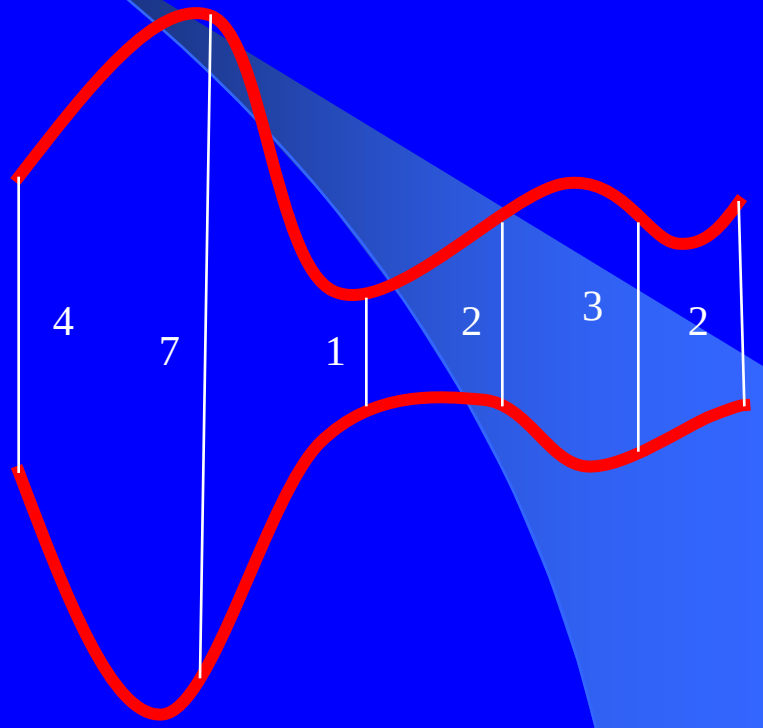
- Find similars series : Marketing, Finances
- Search for a subsequence within another sequence: Music
- Search for patterns that can characterize a population group: Medicine, Fraud detection
- Search for important events: Sismology, Industry
- Build representative models of population groups: Medicine, Traffic Control

# 2. Basic Techniques

# Preliminar Thoughts

- In order to be able to perform any task with time series it is necessary to have a mechanism to compare time series

→ define a distance




$$D=\mathrm{sqrt}(4^2+7^2+1^2+2^2+3^2+2^2)$$

# Problem: Dimensionality Reduction

- Set of time series that represent the stock value of 500 companies during the last 2 months (samples taken every 15min/12hx20days/month)
- Problem: Find the companies with similar behaviour of their stock values in that period of time
- Solution: run a hierarchical clustering algorythm
- Process: Obtain the proximity table 500x500 for the similarity values of each pair of time series (1920 values/TS).

**→ 240M operations → A dimensionality reduction step is needed.**

# Problem: Dimensionality Reduction

- Solution → Use preprocessing techniques that reduce the dimensions of the problem:

  Fourier Transform
  Wavelet Transform
  Segmentation (PAA)
  Other Dimensionality Reduction techniques

# Fourier Transform

- Reduce the dimensions of the problem transforming each time series in a tuple of values.

- Fourier Transform

  Define the function in terms of other functions (sine, cosine):

  – Drastically reduces the amount of information

  – Preserves the properties of the distance between time series

# Fourier Transform

Fourier Transform of a funtion $f(x)$

$$f(x)=\frac{a_0}{2}+\sum_{k=1}^{\infty}(a_n\cos kx+b_n\operatorname{sen} kx),$$

$$\begin{cases} a_k=\dfrac{1}{\pi}\displaystyle\int_0^{2\pi} f(x)\cos kx\ \ dx \\ b_k=\dfrac{1}{\pi}\displaystyle\int_0^{2\pi} f(x)\operatorname{sen} kx\ \ dx, \end{cases}$$

# Fourier Transform

- To obtain an equivalent representation of a time series – $f(x)$ – the first n coeficients are needed (number of timestamps).
- However, with only the first 4 coeficients, 99% of the energy of the TS is preserved, which is enough in the majority of domains.
- Parsival theorem certifies that distance properties are preserved on the transformed domain (frequency).
- Some authors say it is better to take into account (n+1)/2 coeficients

# Fourier Transform

- Comparing Time Series (Agrawal)

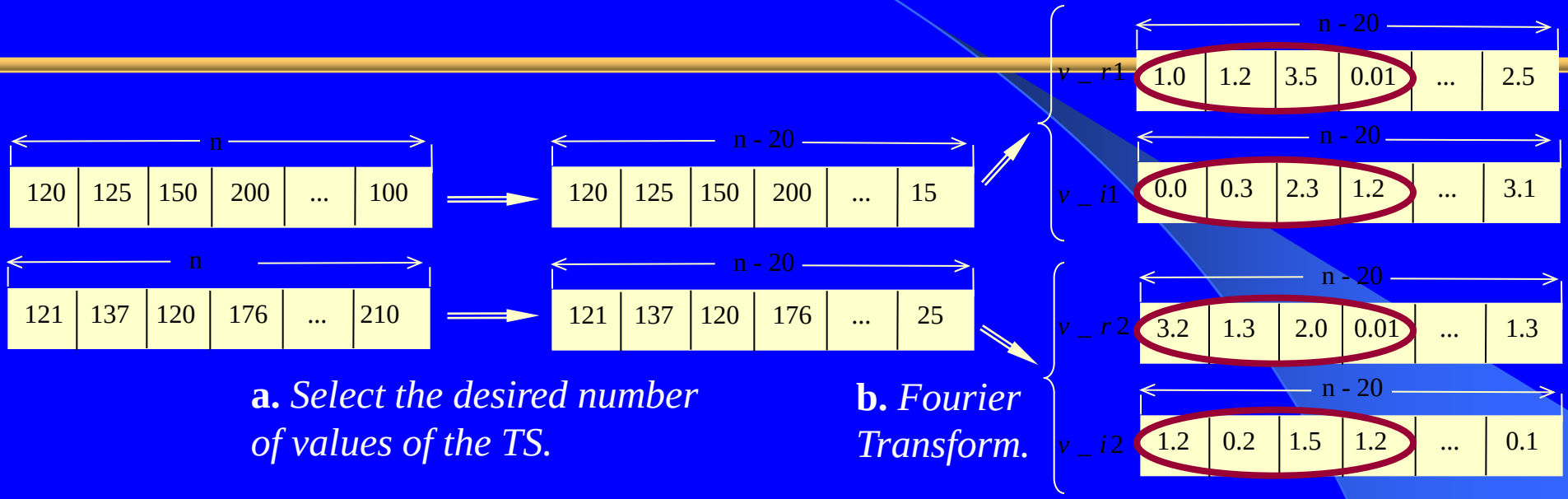
a. Select the desired number of TS data to be compared

b. Calculate the Fourier Transform of the TS

c. Save only the first 4 Fourier coeficients

d. Compute the distance between TS as the square root of the difference between the energy of both series (= euclidean distance)

# Fourier Transform




**a.** *Select the desired number of values of the TS.*

**b.** *Fourier Transform.*

**c.** *Compute the energy difference between both TS.* $\longrightarrow$ $Energy\_total = \sum_{k=1}^{4} (v\_r1[k] - v\_r2[k])^2 + (v\_i1[k] - v\_i2[k])^2$

**d.** *Distance between TS:* $\longrightarrow$ $\boldsymbol{distance} = \sqrt{Energy\_total}$

# Fourier Transform

- Normalization:
  - Normalization of Time Series relative to the average value of the series.

$$X_i^N = \frac{X_i^{SN} - \mu}{\sigma}$$

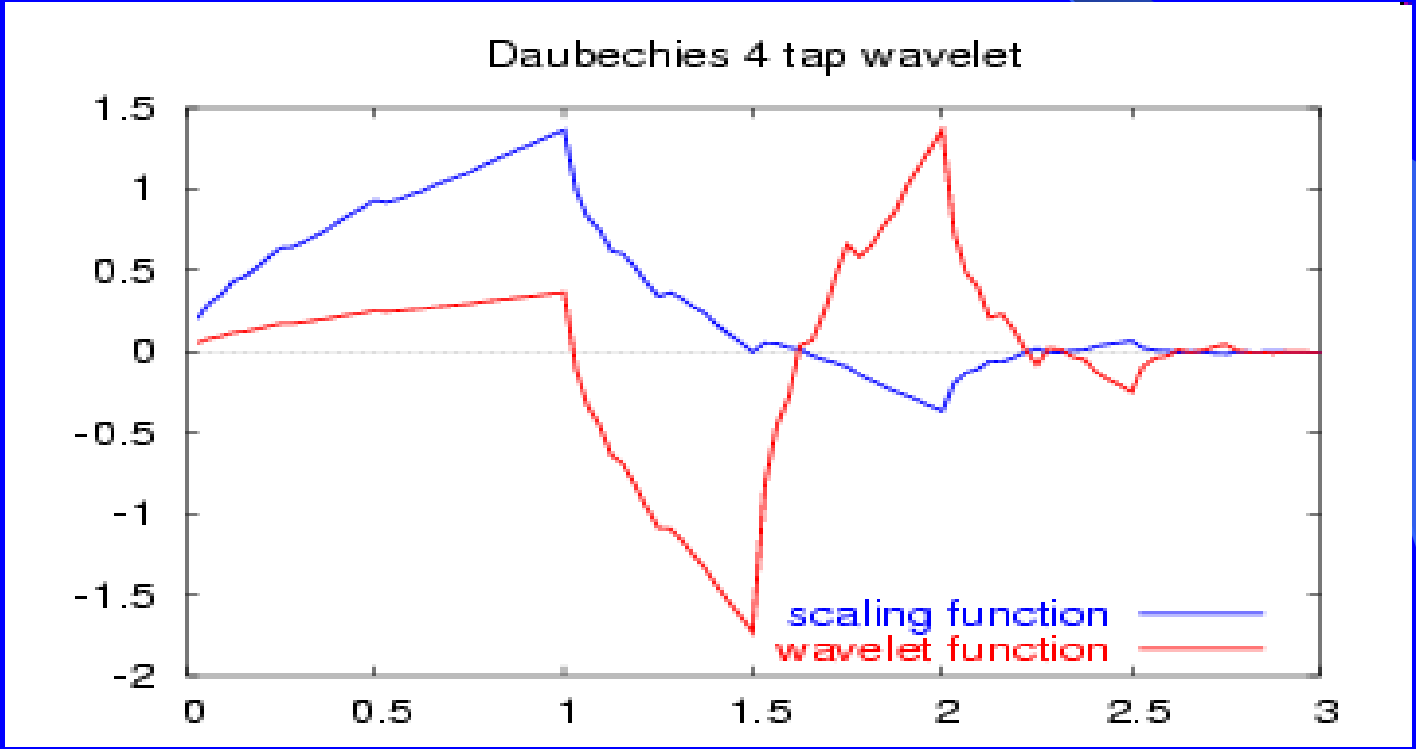
# Wavelet Transform

- Fourier transform works very well with periodic functions but not so well with irregular functions or discontinuities.
- Gabor proposed a windowed Fourier transform which later become the  Wavelet transform

$$W_f(s,\tau) = \int f(t)\psi^*_{s,\tau}(t)\,dt.$$

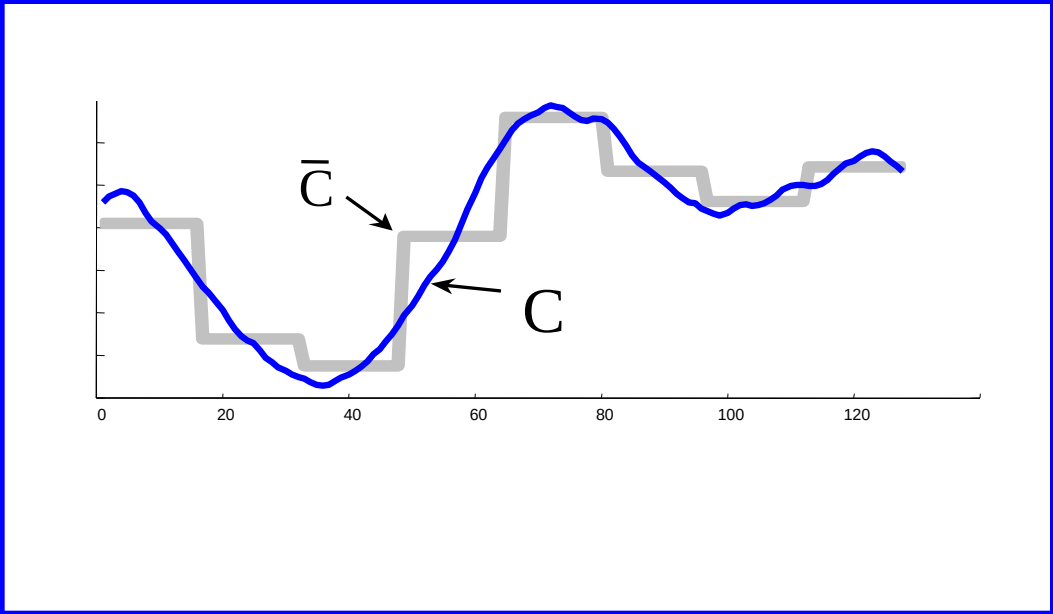
- Wavelets (base functions) are generated through translation and scaling of only one wavelet function

# Wavelet Transform



# Segmentation

- Segmentation applied to TS (PAA):



# 3. Distances

# Distances

- Euclidean
- City – Block
- Time Warping
- Edition distances
  - Needlemann Wunch
  - Levenshtein

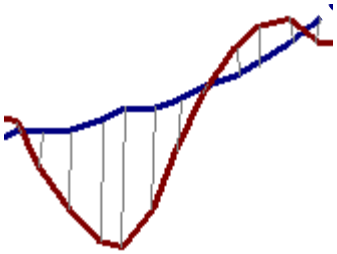
# Euclidean Distance Metric (Keogh)

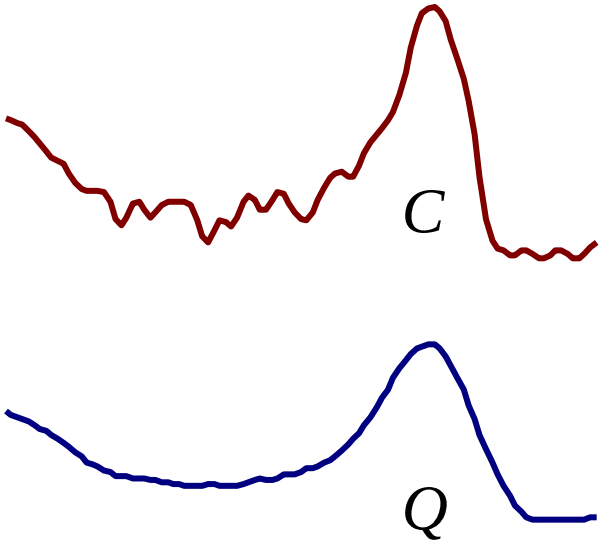
Given two time series

$$Q = q_1 \ldots q_n$$

and

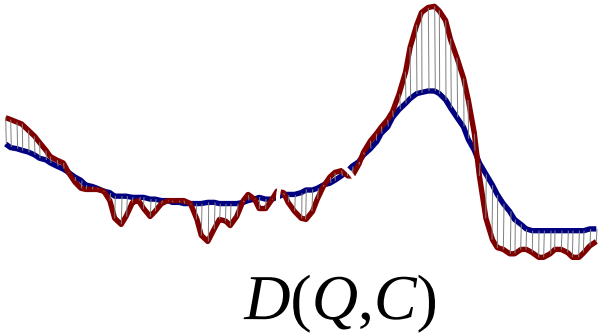
$$C = c_1 \ldots c_n$$





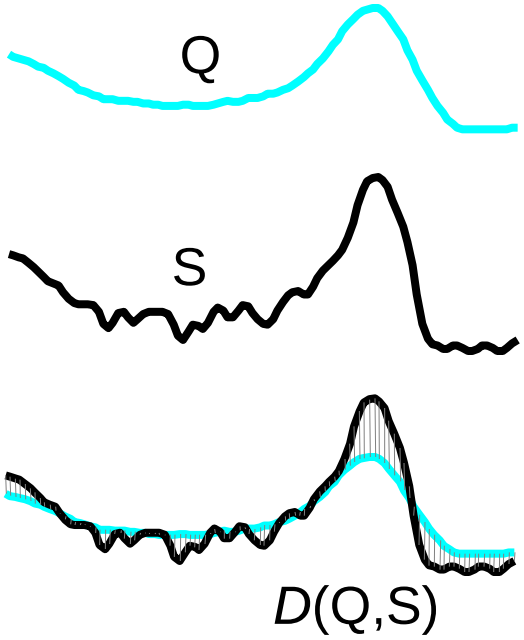

their Euclidean distance:

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



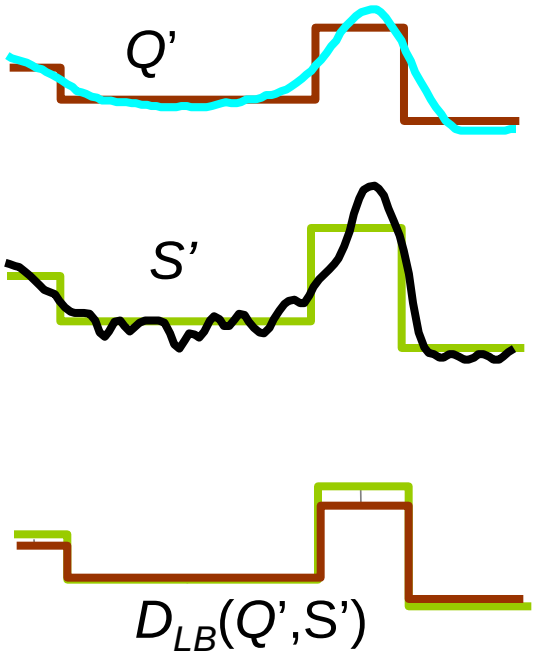
# What is lower bounding?

Exact (Euclidean) distance D(Q,S)




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

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

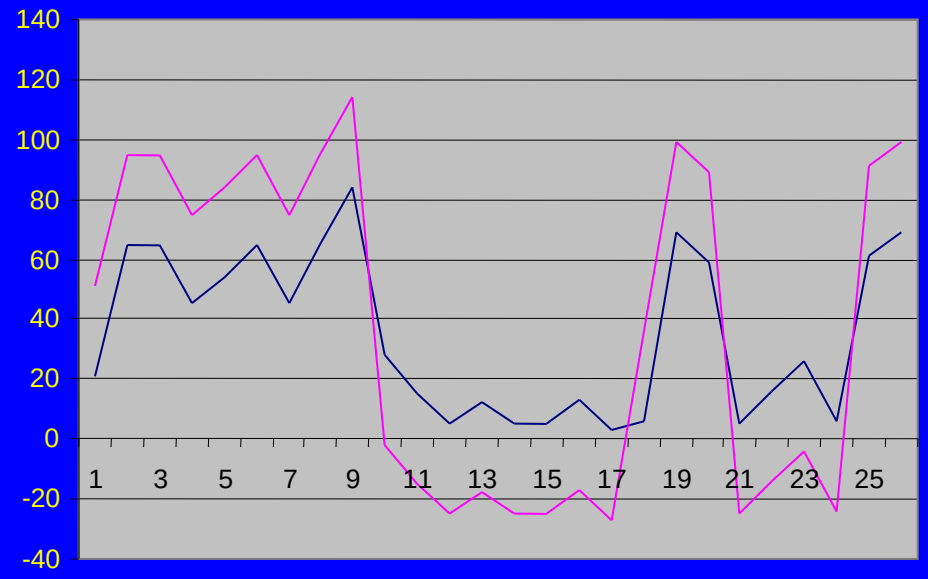



$$D_{LB}(Q',S') \equiv \sqrt{\sum_{i=1}^{M} (sr_i - sr_{i-1})(qv_i - sv_i)^2}$$

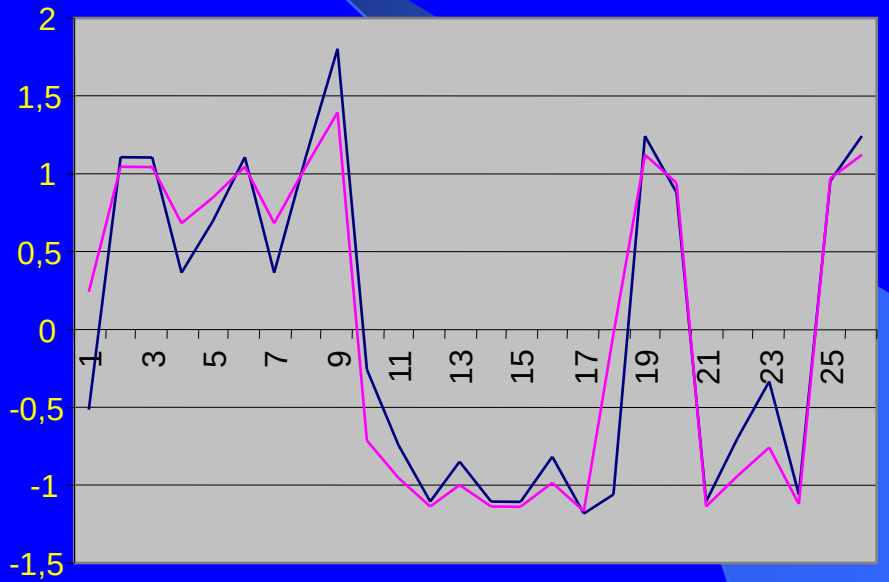
Lower bounding means that for all Q and S, we have…

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

# Normalization




Not NORMALIZATED




NORMALIZATED

# Edition Distances

- Levenshtein Distance
  - Substitution Cost = 1
  - Insertion/Deletion Cost = 1

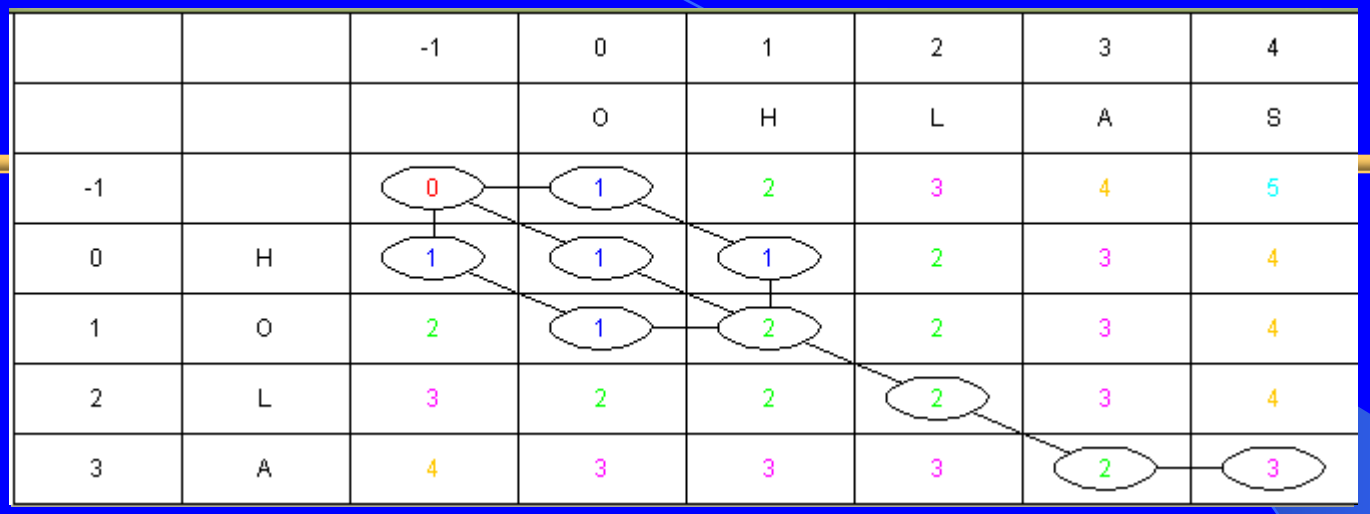
  – Compare
    - U = HOLA
    - W = OHLAS

# Edition Distances

- Dynamic Algorythm

- Compare
  - U = HOLA
  - W = OHLAS

$$D(i,j) = \min \begin{cases} D(i-1,j-1) & \text{si } u_i = w_j \quad \text{//copy} \\ D(i-1,j-1)+1 & \text{si ui != wj} \quad \text{//substitutio} \\ D(i-1,j)+1 & \text{//insertion} \\ D(i,j-1)+1 & \text{//deletion} \end{cases}$$



| | | -1 | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|---|---|
| | | | O | H | L | A | S |
| -1 | | 0 | 1 | 2 | 3 | 4 | 5 |
| 0 | H | 1 | 1 | 1 | 2 | 3 | 4 |
| 1 | O | 2 | 1 | 2 | 2 | 3 | 4 |
| 2 | L | 3 | 2 | 2 | 2 | 3 | 4 |
| 3 | A | 4 | 3 | 3 | 3 | 2 | 3 |

- Compare
  - U = HOLA
  - W = OHLAS

**3 ALLIGNMENTS**

**U → H O L A -**
**W → O H L A S**

**U → H O - L A -**
**W → - O H L A S**

**U → - H O L A -**
**W → O H - L A S**

# DM on TS: What can be done?

- Compare two time series
  - Total comparison
  - Partial Comparison
- Search of subsequences
- Pattern detection in a group of time series
- Event analysis in one time series

# 4. Time Series Comparison

# Comparing TS

- Total Comparison
  - Transforms (Fourier, Wavelet)
  - Based on Transformations
  - Landmarks, Change Points
  - Index Trees (R-trees)
- Partial Comparison
  - MBRs [Faloutsos et al.]

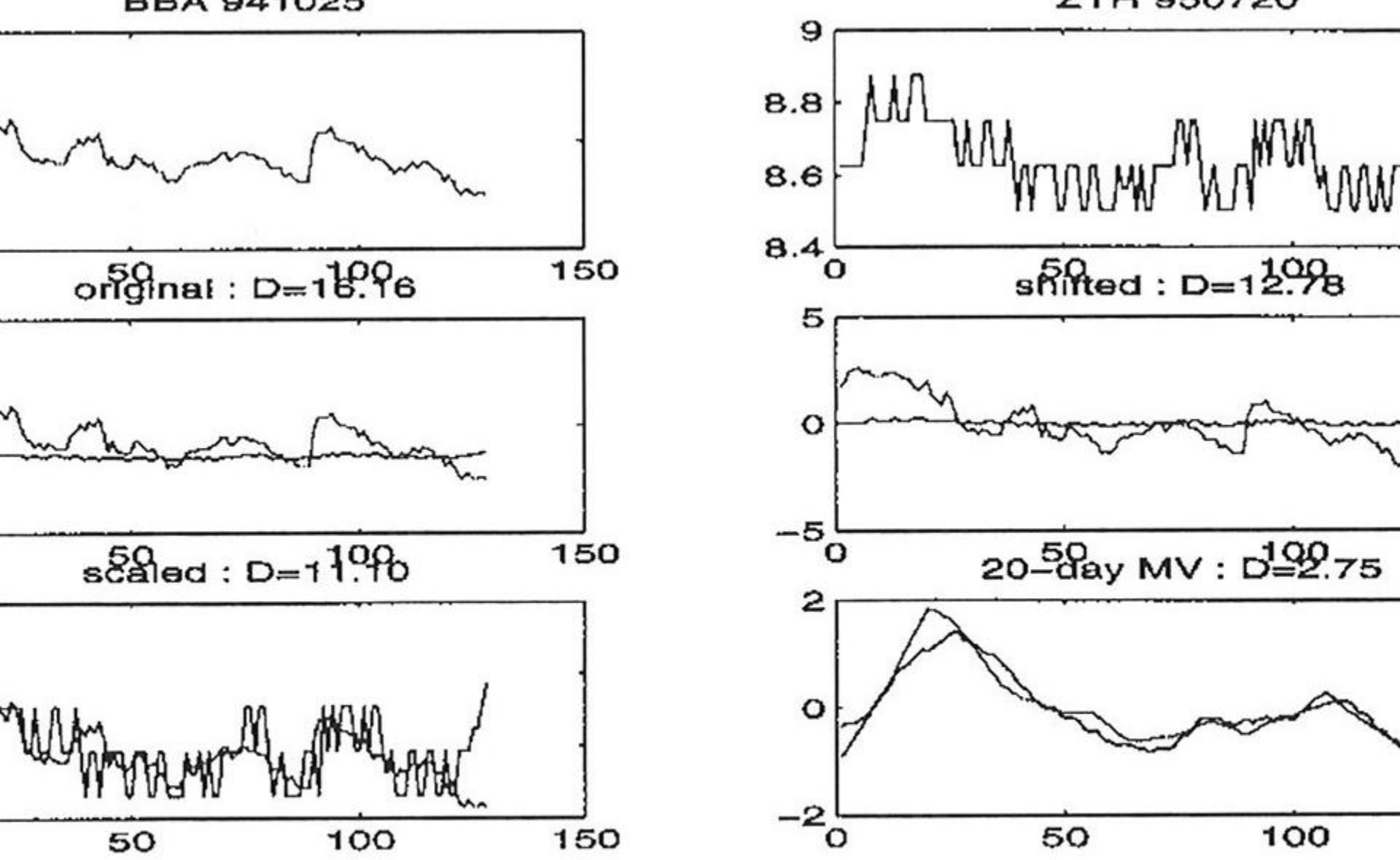
# Comparing TS Based on Transformations

- Fourier transform is efficient but does not allow the user to control in which way are the time series similar (beyond the threshold definition)
- Rafiei and Mendelzon proposed a method to measure the similarity between TS based on a set of operations (transformations) performed on the TS.
  - If, by means of these operations, both series converge, they are considered similar.

# Comparing TS Based on Transformations

- Valid transformations:
  - Moving averages
  - Temporal Scaling
  - Value Scaling
  - Temporal translation
  - Value translation

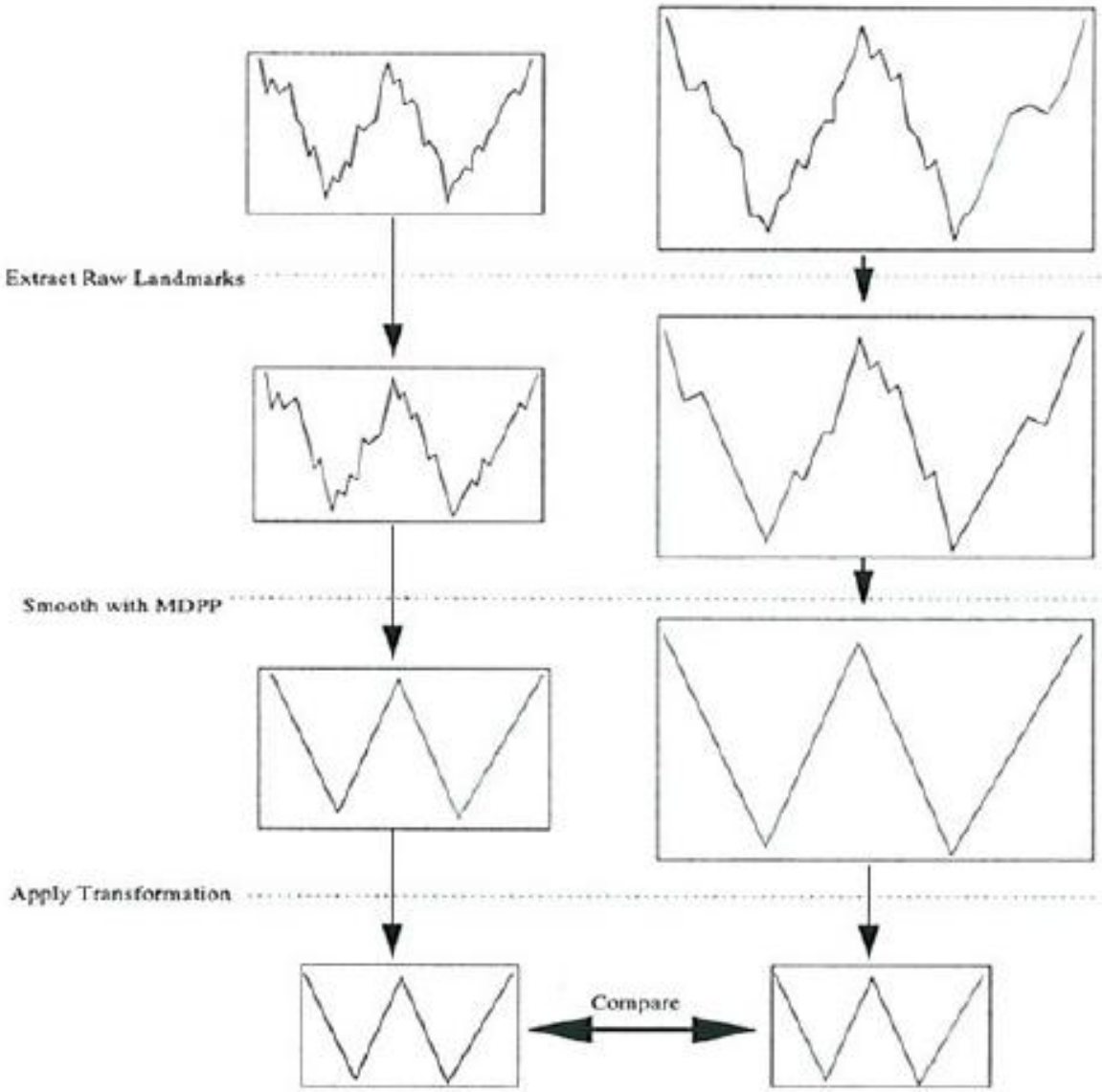
A cost can be assigned to each operation and use the accumulated cost of operations need to transform one serie into the other as the distance between TS.




m left to right, top to bottom: the daily closing price for *The Bombay Co. (BBA* r 128 days, the daily closing price for *Zweig Total Return Fund Inc. (ZTR)* starting fr he two stocks put together, both shifted, both scaled, and the 20-day moving average

# Landmarks (Perng et al.)

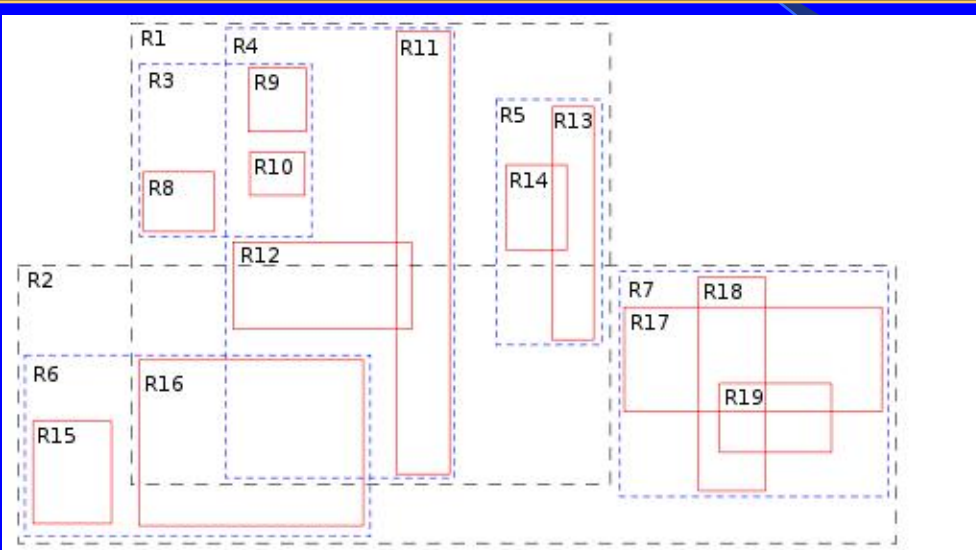
- Landmarks are the outstanding points of a TS
- An order n landmark  is defined as the point where the n-derivative of the TS is zero
- How to use landmarks to compare TS:
  - Landmarks Identification
  - Softing the landmark sequences using the Minimal Distance Percentage Principle
  - Comparison based on transformations

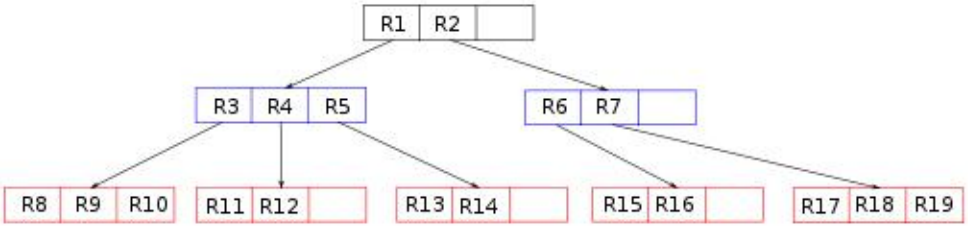

Extract Raw Landmarks
Smooth with MDPP
Apply Transformation
Compare

# R-tree

- These are index structures useful to speed up the search on large volumes of data. R stands for rectangle
- An index is built with the available data through insertion, deletion and search mechanism
- The idea is to find the Minimum Bound Rectangles that encloses the data
- As an aditional advantage, k-nn queries, set intersections, etc. are easy and efficiently done

# R-tree


R1
R4
R11
R3
R9
R5
R13
R10
R14
R8
R12
R2
R7
R18
R17
R6
R16
R19
R15

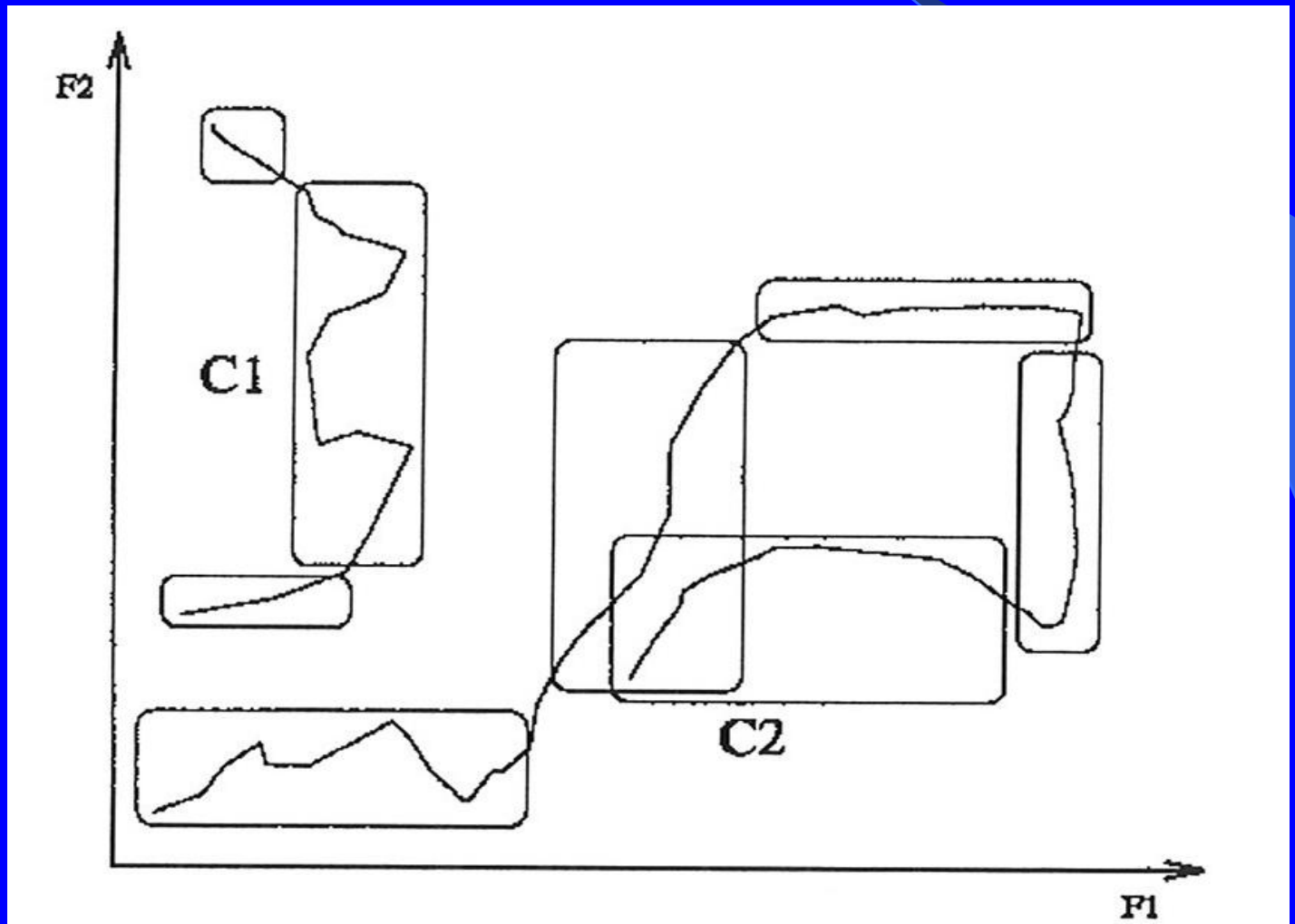

R1
R2
R3
R4
R5
R6
R7
R8
R9
R10
R11
R12
R13
R14
R15
R16
R17
R18
R19

# 5. Subsequence Search

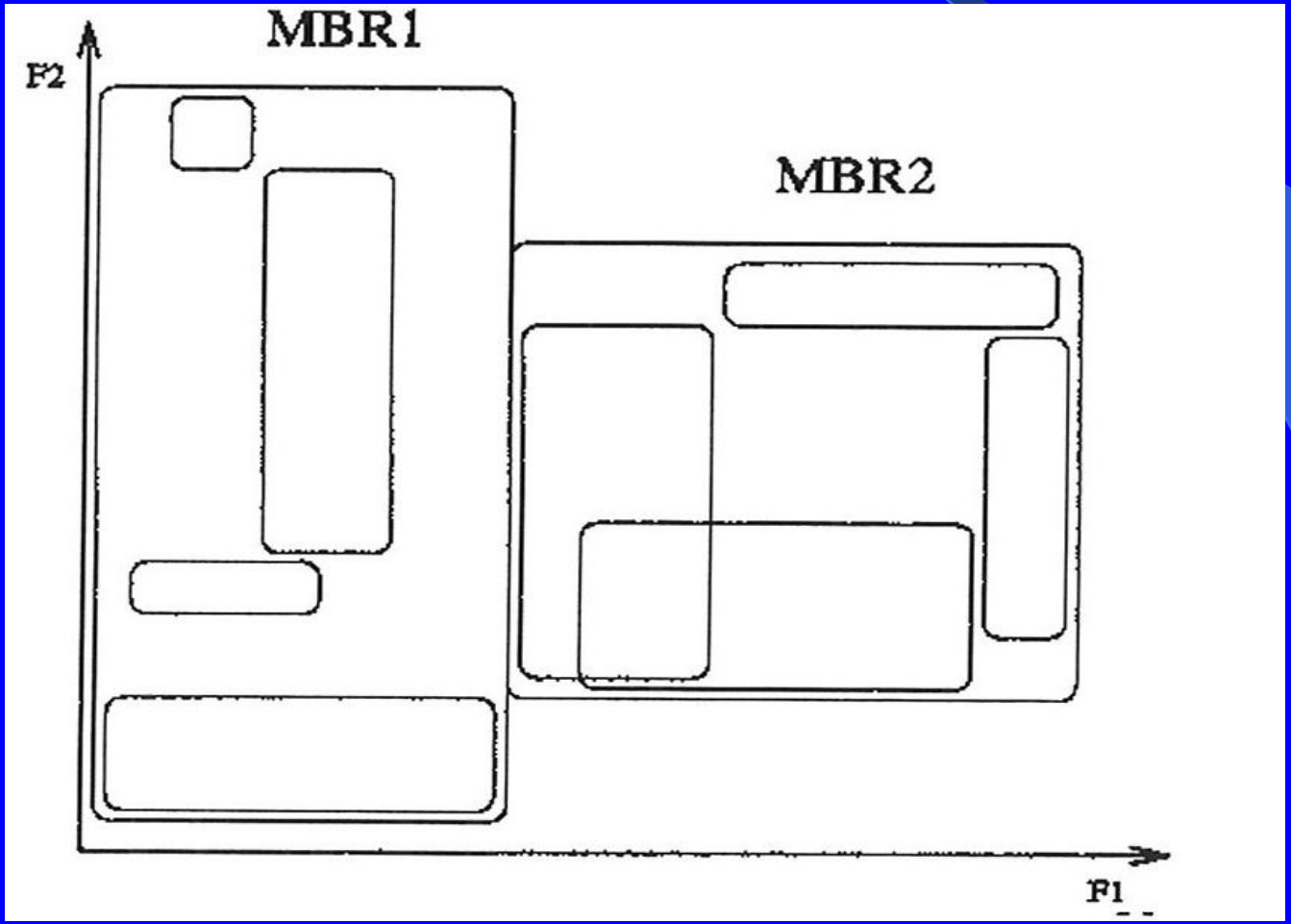
# Subsequence Search

- MBR: Efficient search of subseries in a given TS [Faloutsos et al. 98]
  - Create a moving window and calculate the FT of the subseries within the window ➡ trace
- Kahveci y Singh generalization to subseries search of any size

# Subsequence Search



# Subsequence Search



# 6. Pattern Identification

# Pattern

Pattern is a subsequence that appears a relevant number of times in a given set of TS

- Frequent Pattern [Han et al 2000]
- Partial Periodic Pattern [Han et al 98]
- I4

# Frequent Pattern: Han et al.

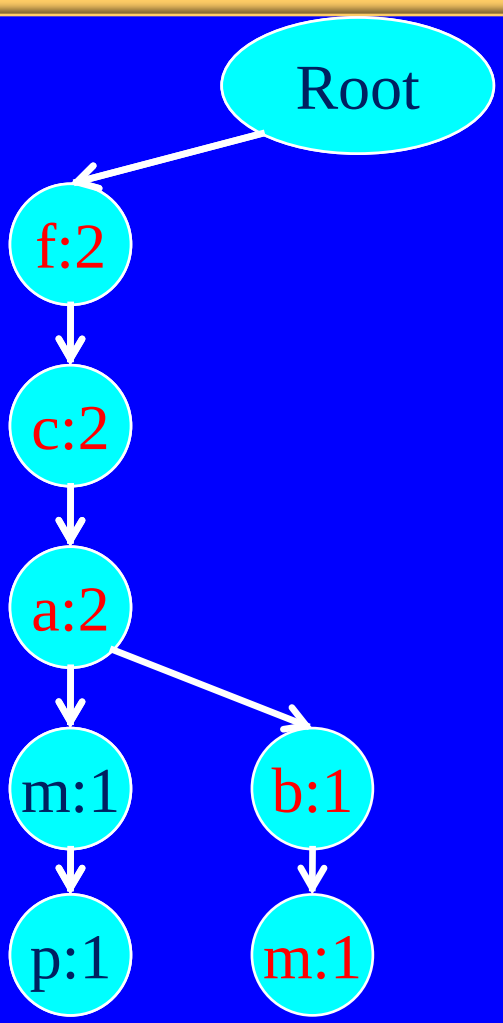
| Serie | Itemset | Frequent Itemset |
|---|---|---|
| S1 | f a c d g i m p | f, c, a, m, p |
| S2 | a b c f l m o | f, c, a, b, m |
| S3 | b f h j o | f, b |
| S4 | b c k s p | c, b, p |
| S5 | a f c e l p m | f, c, a, m p |

Frequent-1Itemset (min=3): f:4, c:4, a:3, b:3, m:3, p:3

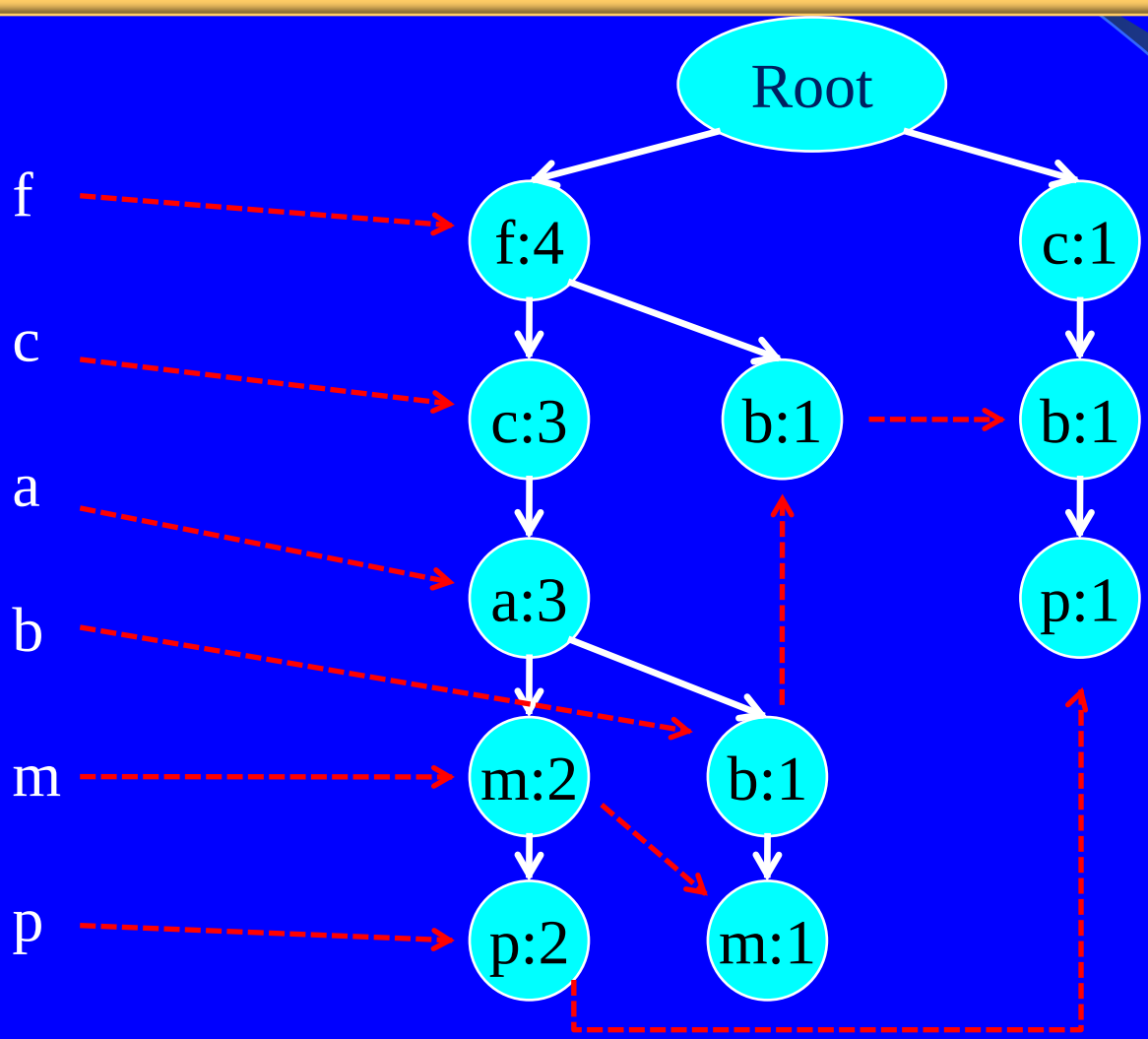
# Building the FP-tree

S1: facdgimp

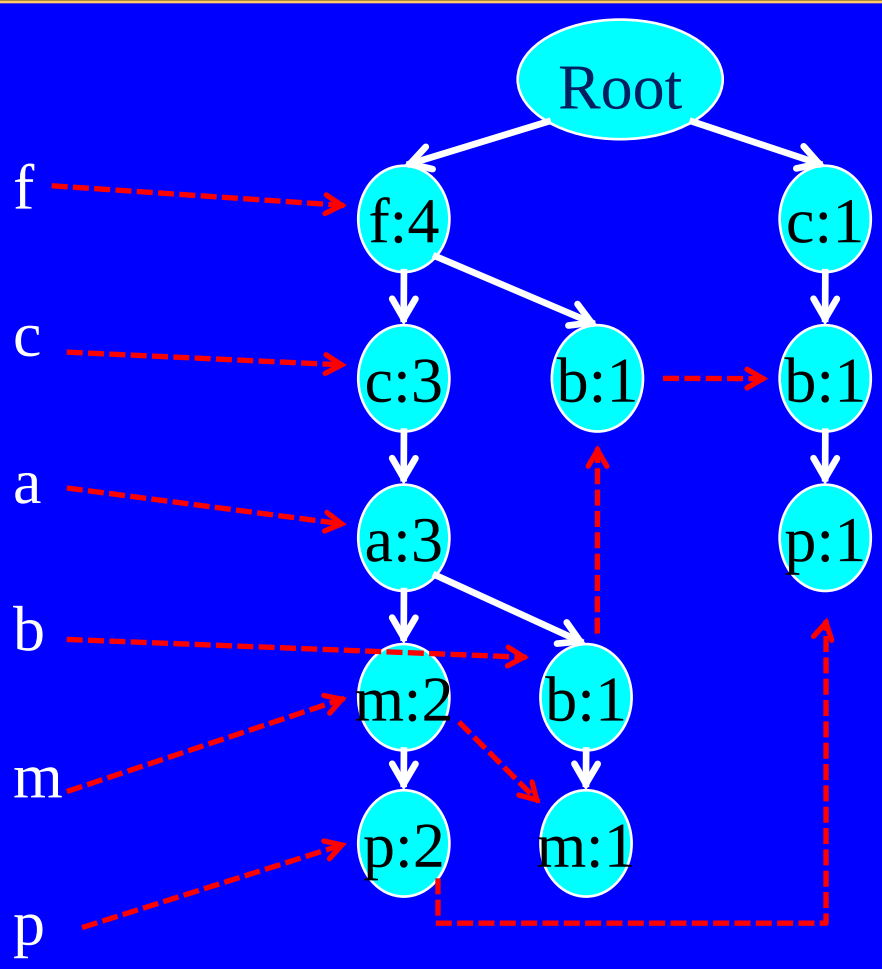
S2: abcflmo



# Final FP-tree



# Using the FP-Tree




| Item | BD Conditional | FP-tree Conditional |
|---|---|---|
| p | {(fcam:2), (cb:1)} | {(c:3)}/p |
| m | {(fca:2), (fcab:1)} | {(fca:3)}/m |
| b | {(fca:1),(f:1),(c:1)} | ∅ |
| a | {(fc:3)} | {(fc:3)}/a |
| c | {(f:3)} | {(f:3)}/c |
| f | ∅ | ∅ |

# Partial Periodic Pattern

a c b a e b a c e d

* is used as a joker

a * b is a length pattern with 2/3 confidence

3: maximun number of periods of length 3

a {b, c} * length=3, confidence = 2/3

min_confidence threshold

# Partial Periodic Pattern

Search Method:

A priori based

1. Find the length(1) frequent itemsets
2. Calculate length(p+1) frequent itemsets based on length(p) frequent itemsets

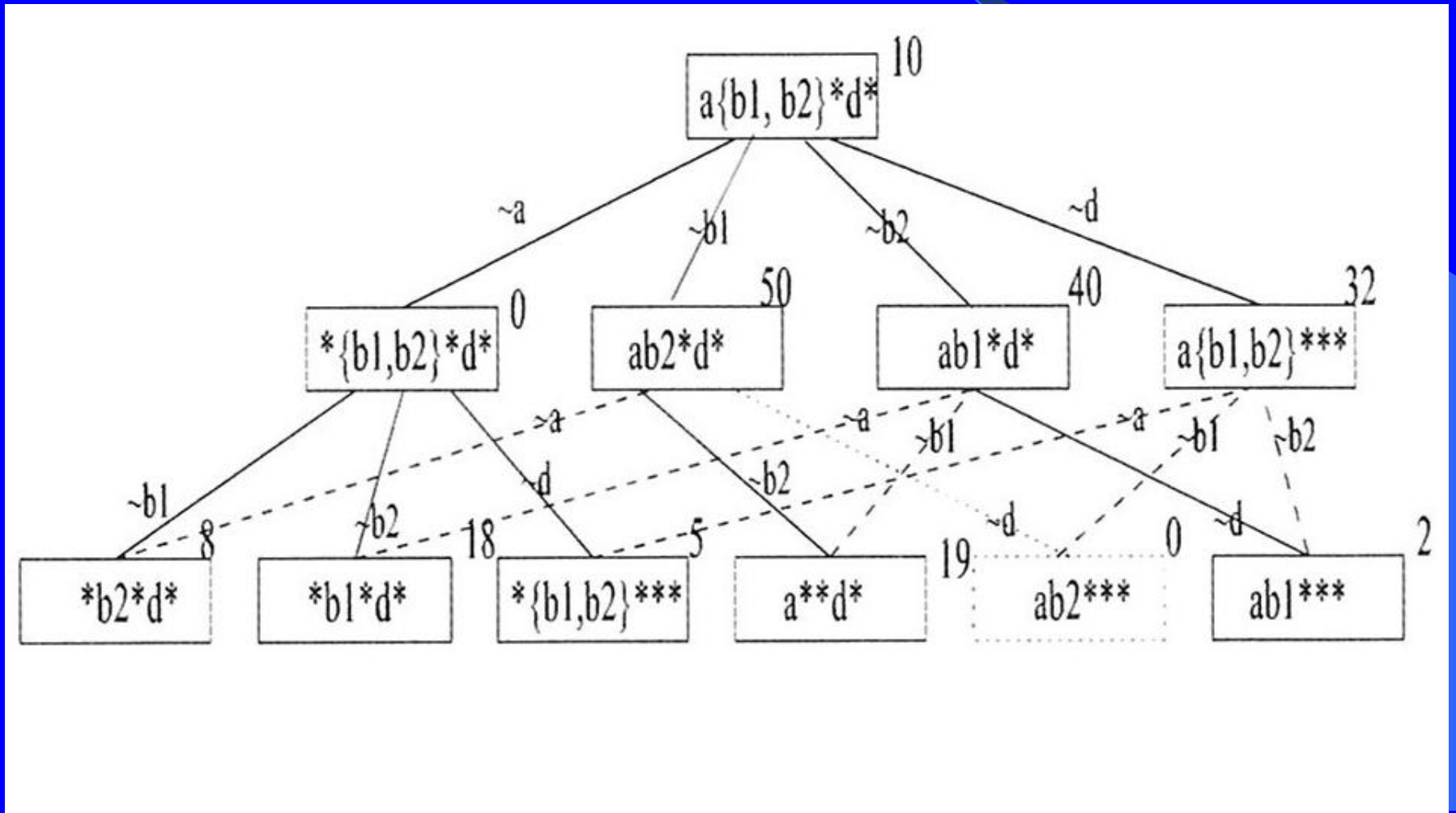
Max-Subpattern Hit Set (Han et al.)

1. Find the length(1) frequent itemsets
2. Calculate the Max-Subpattern-hit tree

# Partial Periodic Pattern

- Max-Subpattern Hit Set (Han et al.)
  - Scan the set of TS to find the patterns of length=1
  - Find the Max Pattern candidate
  - Build the Max-Subpattern Hit Tree

# Arbol de Sub-patrones máximo



# I4: Pattern Search

Pattern: subsequence that repeats frequently in a set of TS
Differences with Han technique:

- Search several series instead of only one
- Finds patterns of any length versus only a given set of lengths
- Continuos data instead of symbolic (discrete)
- Aproximate values instead of joker values

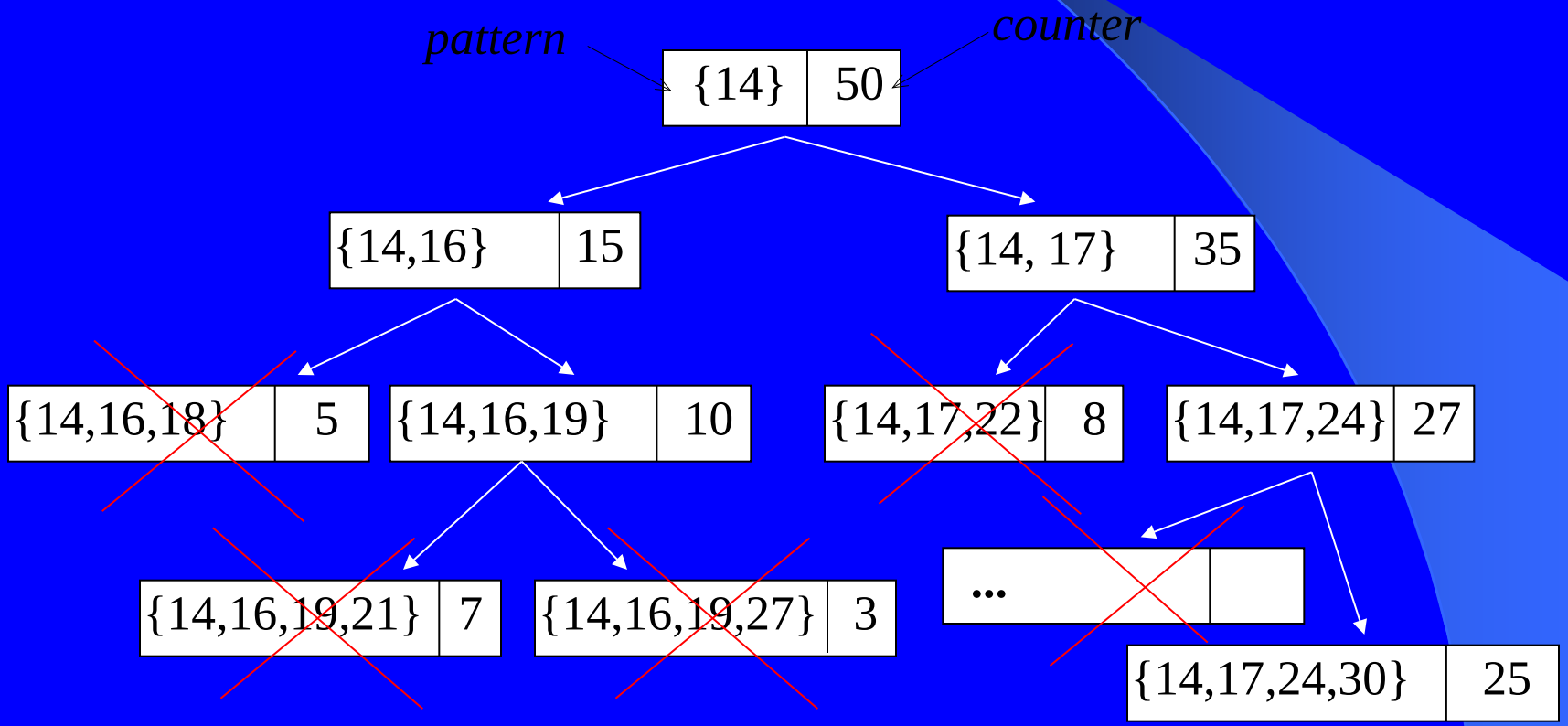
# I4: Pattern Search (discrete)

- Step A.1) Go over the time series forming patterns of length i, inserting them later in a search tree. Count the frequency of each pattern.

- Step A.2) Prune the tree. Those branches that correspond to nonfrequent patterns will be pruned and they will never be considered again.

- Step A.3) Once all the branches have been pruned, or once the length of the pattern surpasses the length of all time series, go over the tree forming the patterns of different length

# I4: Pattern Search (discrete)



# I4: Pattern Search (continuos)

Why similar?

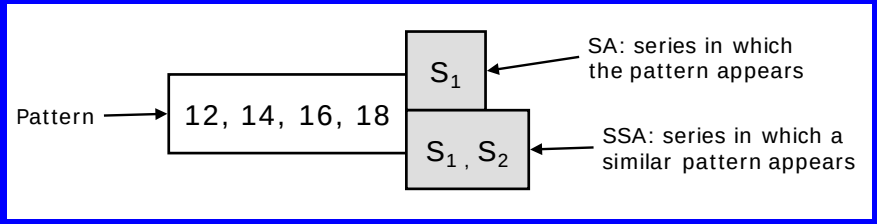
- Noise
- Absolute values are not determinant in continuos domains

Given

- A collection of **M** exercises (time series) from injured patients
- A collection of **M** exercises from non-injured patients
- The maximun distance **d** between similar patterns
- A thereshold **ε** for the confidence of the patterns

# I4: Pattern Search (continuos)

- Step1. Build length(i) patterns (starting with 1).
- Step 2. Calculate the distance between patterns and update each node




- Step 3. Prune the tree. A node will be pruned if:

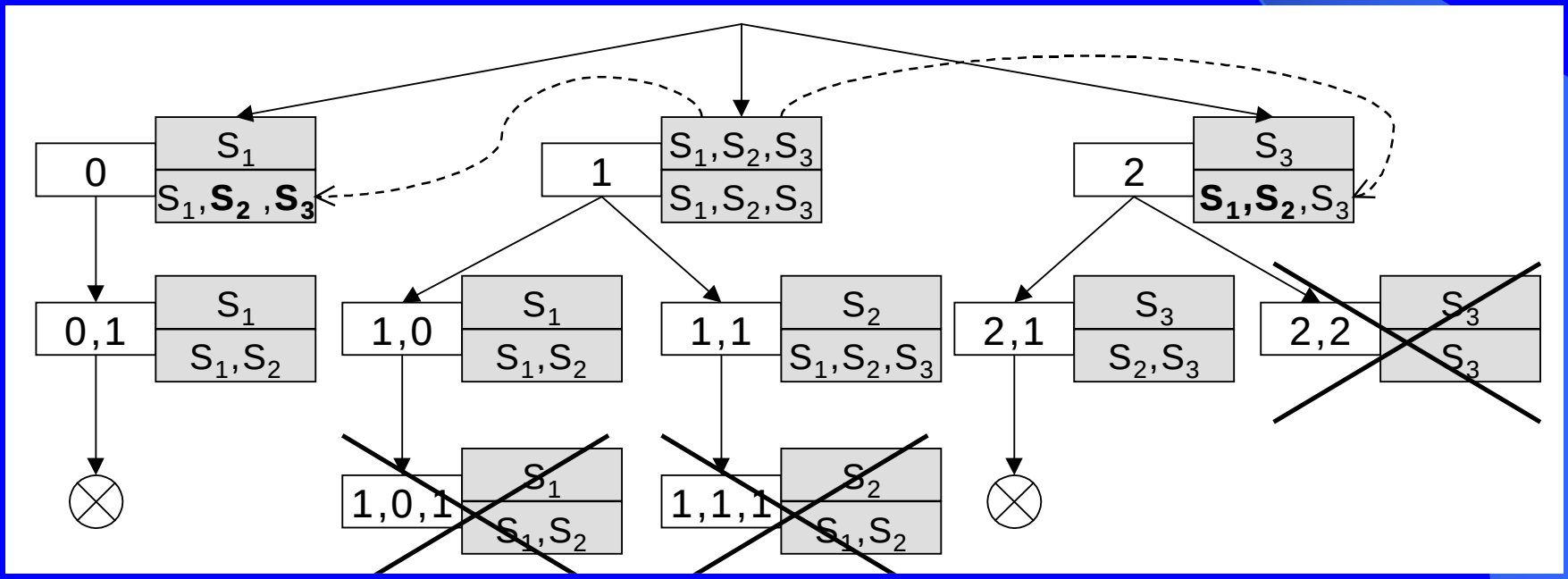
(The pattern of the node is not frequent)

AND

(The distance of the node to every other node of that level is bigger than threshold **d)**

# I4: Pattern Search (continuos)

$S_1 = \{1, 0, 1\}$, $S_2 = \{1, 1, 1\}$, $S_3 = \{2, 2, 1\}$
*min-conf* $\varepsilon = 0.75$ *max-dist* $d = 1$.



# 7. Temporal Abstraction

# Temporal Abstraction

Transform the numerical TS into a symbolic TS to capture the semantics of the TS, as a dimensionality reduction step, etc.

- SDL
- SAX
- I4

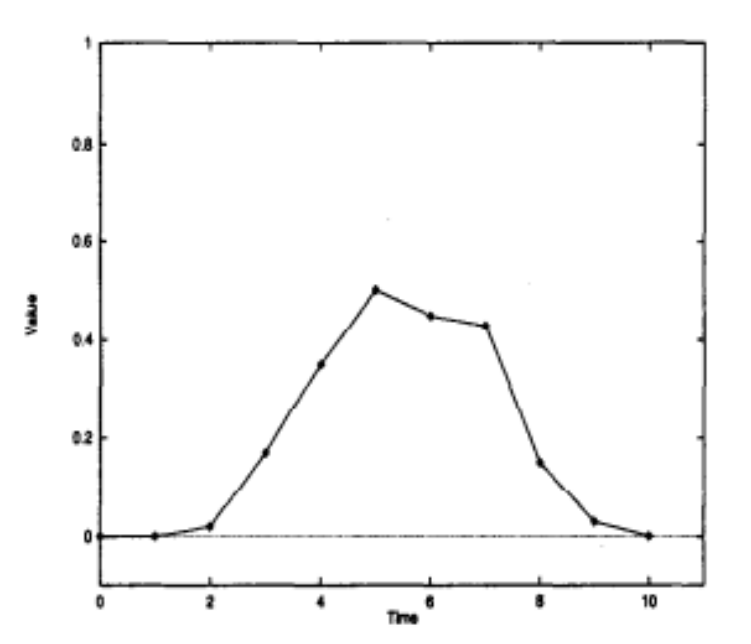
# SDL: Shape Definition Language
## Agrawal et al.

Alphabet (symbol and restrictions on lower and upper bounds differences, and on initial and final values)

| Symbol | Description | *lb* | *ub* | *iv* | *fv* |
|---|---|---|---|---|---|
| `up` | slightly increasing transition | .05 | .19 | `anyvalue` | `anyvalue` |
| `Up` | highly increasing transition | .20 | 1.0 | `anyvalue` | `anyvalue` |
| `down` | slightly decreasing transition | -.19 | -.05 | `anyvalue` | `anyvalue` |
| `Down` | highly decreasing transition | -1.0 | -.19 | `anyvalue` | `anyvalue` |
| `appears` | transition from a zero value to a non-zero value | 0 | 1.0 | `zero` | `nonzero` |
| `disappears` | transition from a non-zero value to a zero value | -1.0 | 0 | `nonzero` | `zero` |
| `stable` | the final value nearly equal to the initial value | -.04 | .04 | `anyvalue` | `anyvalue` |
| `zero` | both the initial and final values are zero | 0 | 0 | `zero` | `zero` |

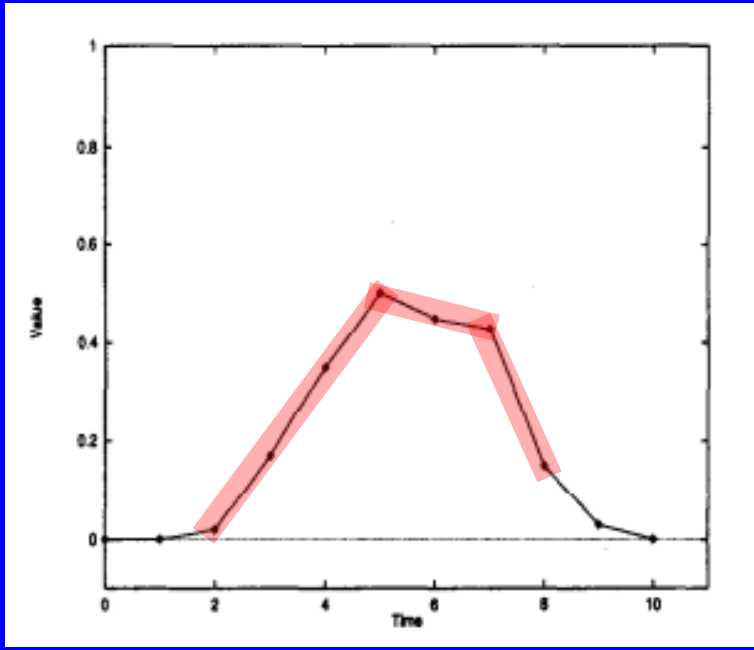
# SDL: Shape Definition Language

(zero appears up up up down stable Down down disappears)

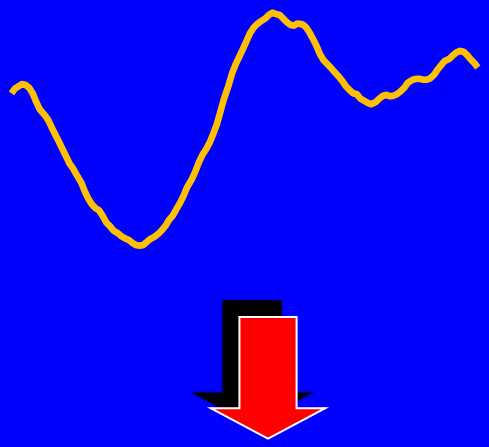


# SDL: Shape Definition Language

(shape pulse( )
(concat up up up (any stable down)(any stable down)
(any down Down))

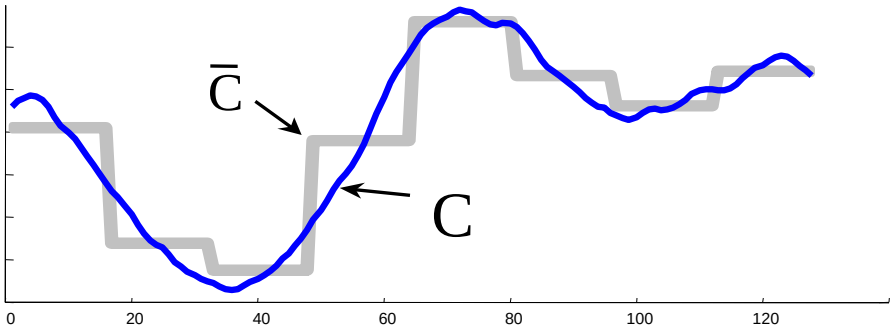


# SAX: Symbolic Aggregate ApproXimation – E.Keogh



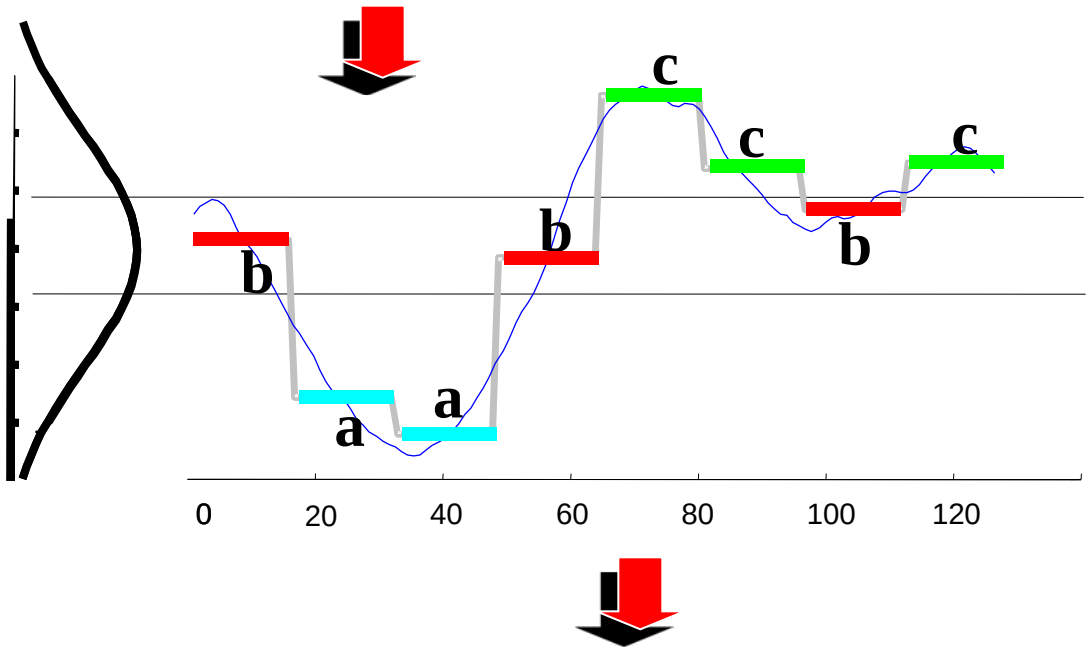
baabccbc

# How do we obtain SAX?

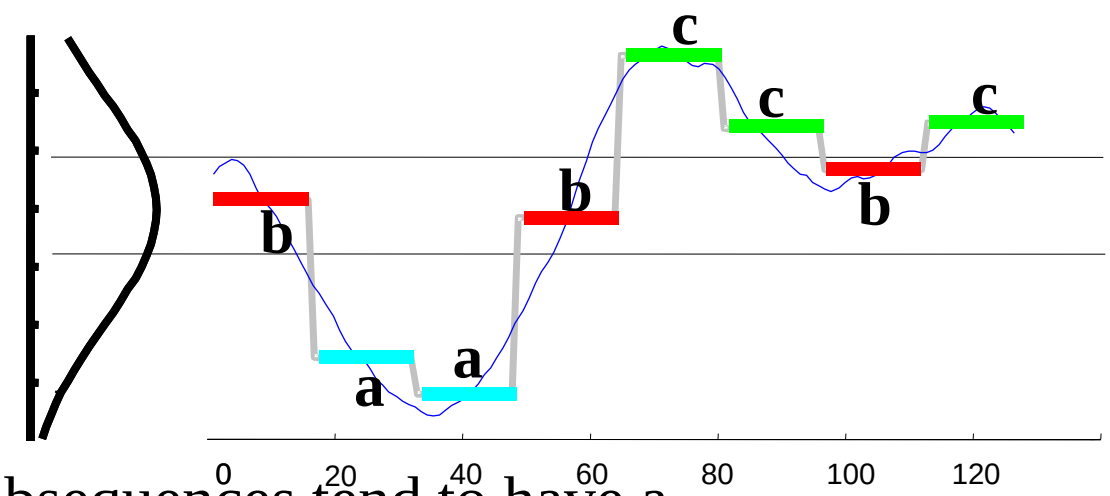



First convert the time series to PAA representation, then convert the PAA to symbols

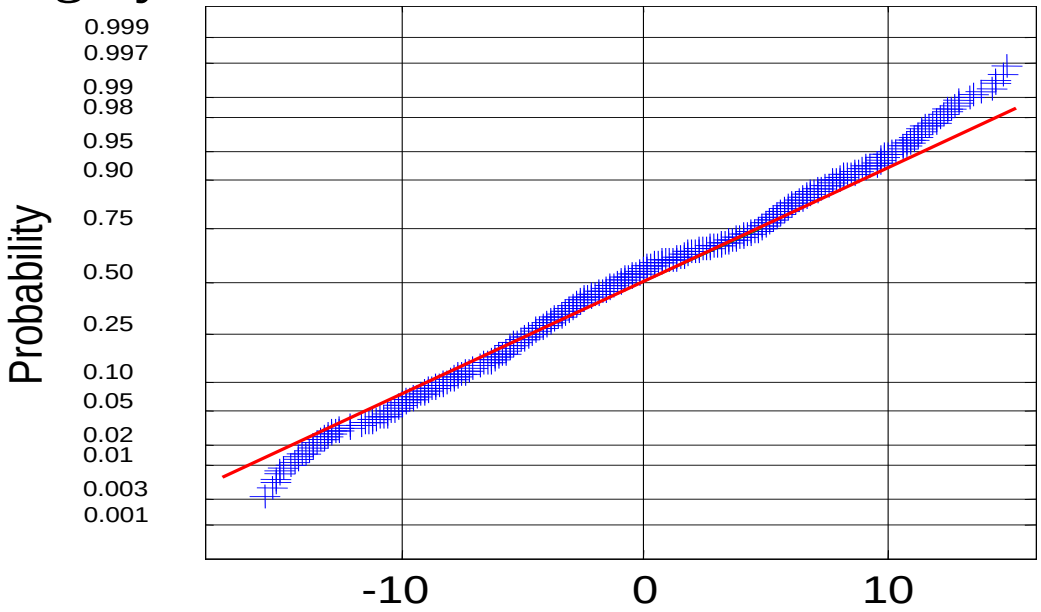
It take linear time




baabccbc

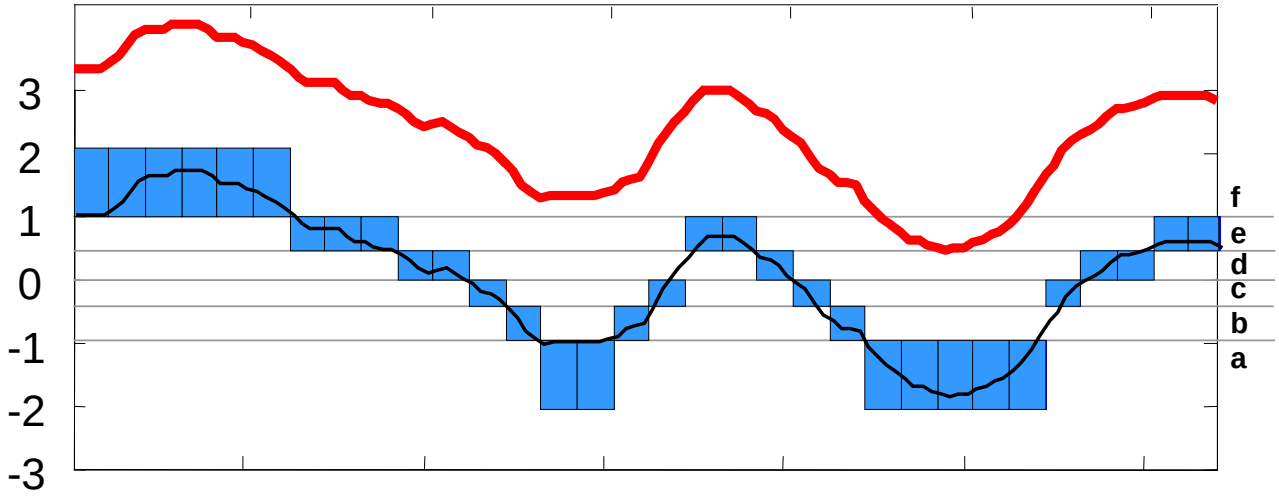


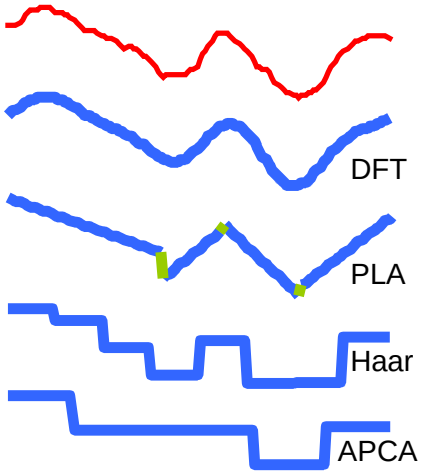

Time series subsequences tend to have a highly Gaussian distribution




A normal probability plot of the (cumulative) distribution of values from subsequences of length 128.

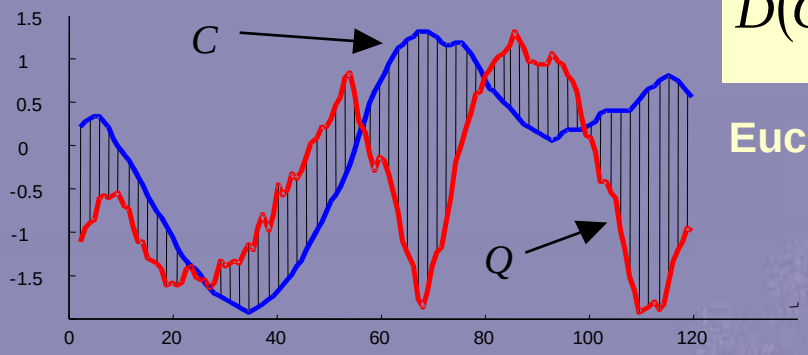
# Visual Comparison




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

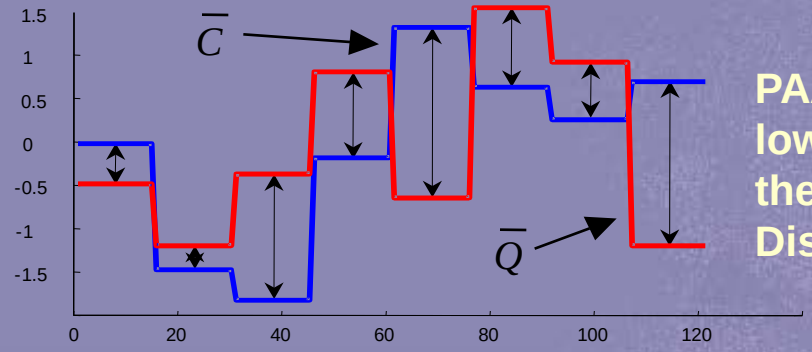
- We can use more symbols to represent the time series since each symbol requires fewer bits than real-numbers (float, double)



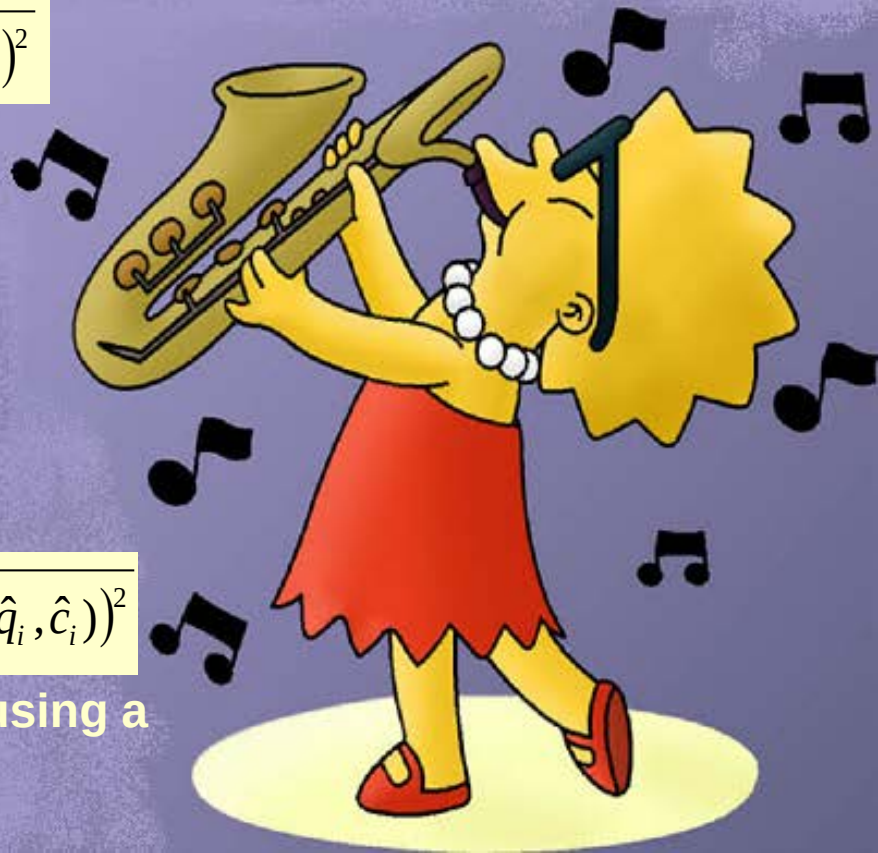

$$D(Q,C) \equiv \sqrt{\sum_{i=1}^{n} (q_i - c_i)^2}$$

**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**



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

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

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

**dist() can be implemented using a table lookup.**

# SEM: Domain Dependent Temporal Abstraction (Santamaria et al.)

- Main objective: the KDD process should use the concepts of the domain, and explain the results obtained using those concepts
- How to know how the expert works
  - Planification of interviews with the expert
  - Know the work environment

# SEM: Domain Dependent Temporal Abstraction (Santamaria et al.)


- Necesario conocer que símbolos son los utilizados por el experto
  - Planificación de Entrevistas con el experto
  - Conocer el entorno de trabajo del experto

# SEM: Domain Dependent Temporal Abstraction (Santamaria et al.)


- Necesario conocer que símbolos son los utilizados por el experto
  - Planificación de Entrevistas con el experto
  - Conocer el entorno de trabajo del experto

# SEM: Domain Dependent Temporal Abstraction (Santamaria et al.)


- Necesario conocer que símbolos son los utilizados por el experto
  - Planificación de Entrevistas con el experto
  - Conocer el entorno de trabajo del experto

# 8. Events

# Eventos

Event is a subsequence within the TS with special value for the interpretation of the TS behaviour

- TSDM
  - Framework for Event Detection on Time Series
  - [Povinelli 99]
- VIIP