Temporal data

- Stock market data
- Robot sensors
- Weather data
- Biological data: e.g. monitoring fish population.
- Network monitoring
- Weblog data
- Customer transactions
- Clinical data
- EKG and EEG data
- Industrial plan monitoring

Temporal data have a unique structure:

High dimensionality

High feature correlation

Requires special data mining techniques

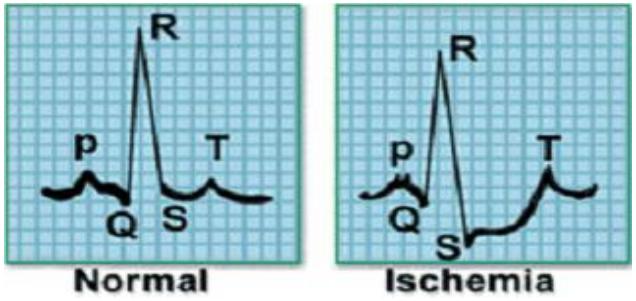
Temporal data

- Sequential data (no explicit time) vs. time series data
  - Sequential data e.g. : Gene sequences (we care about the order, but there is no explicit time!).
- Real valued series vs. symbolic series
  - Symbolic series e.g.: customer transaction logs.
- Regularly sampled vs irregularly sampled time series
  - Regularly sampled time series e.g.: stock data.
  - Irregularly sampled time series e.g.: weblog data, disc accesses.
- Univariate vs multivariate
  - Mulitvarite time series e.g.: EEG data

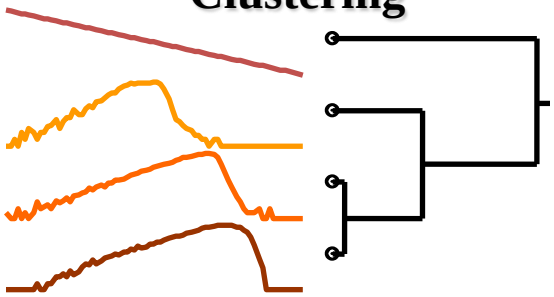
Example: clinical datasets are usually multivariate, real valued and irregularly sampled time series.

Temporal Data Mining Tasks

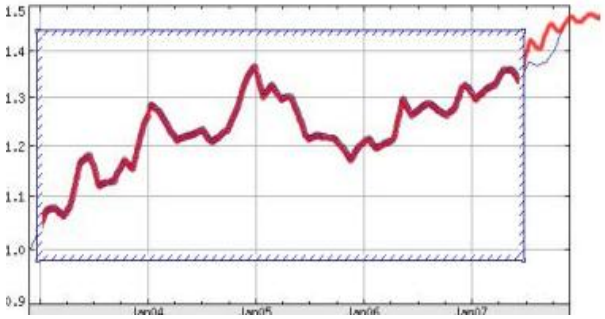
Classification



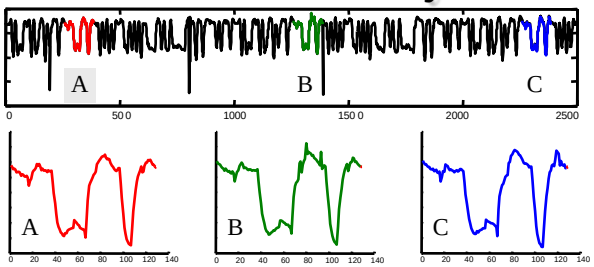
Clustering



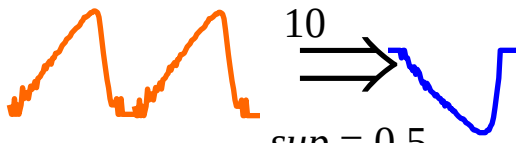
Prediction



Motif Discovery

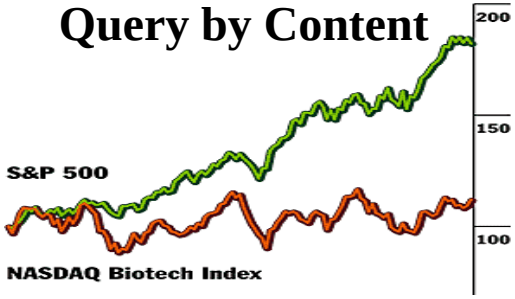


Rule Discovery

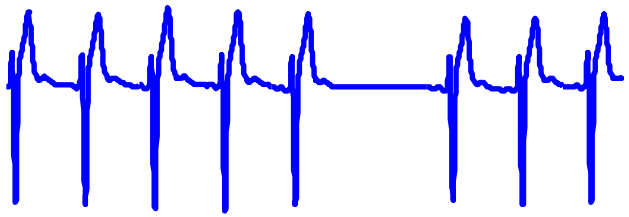


sup = 0.5
conf= 0.6

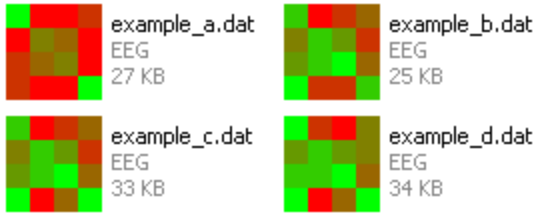
Query by Content



Anomaly Detection



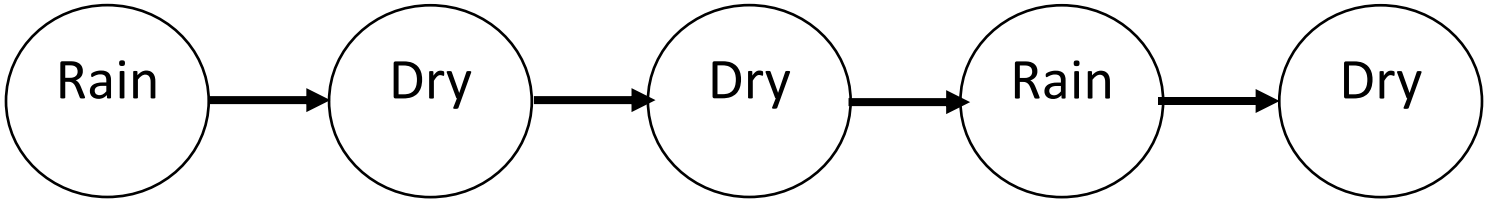
Visualization



Temporal Data Mining

- Hidden Markov Model (HMM)
- Spectral time series representation
  - Discrete Fourier Transform (DFT)
  - Discrete Wavelet Transform (DWT)
- Pattern mining
  - Sequential pattern mining
  - Temporal abstraction pattern mining

Markov Models

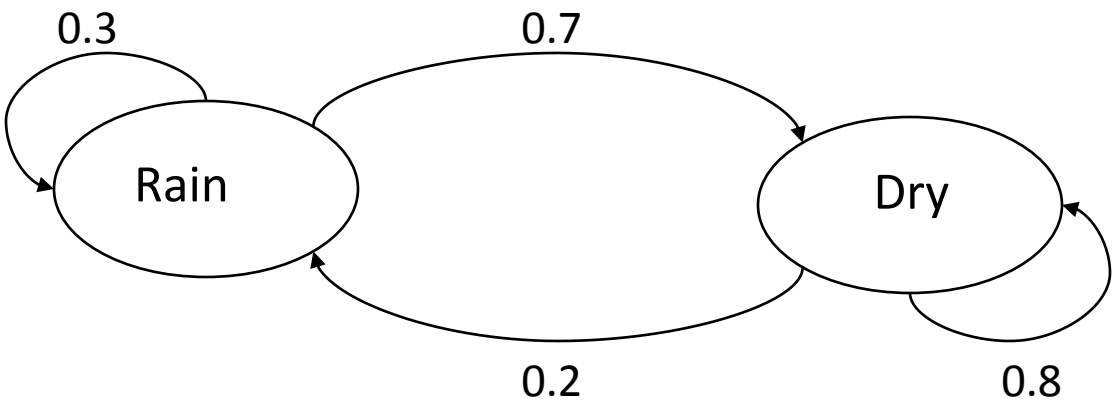


- Set of states: $\{s_1, s_2, \ldots, s_N\}$
- Process moves from one state to another generating a sequence of states: $s_{i1}, s_{i2}, \ldots, s_{ik}, \ldots$
- Markov chain property: probability of each subsequent state depends only on what was the previous state:

$$P(s_{ik} \mid s_{i1}, s_{i2}, \ldots, s_{ik-1}) = P(s_{ik} \mid s_{ik-1})$$

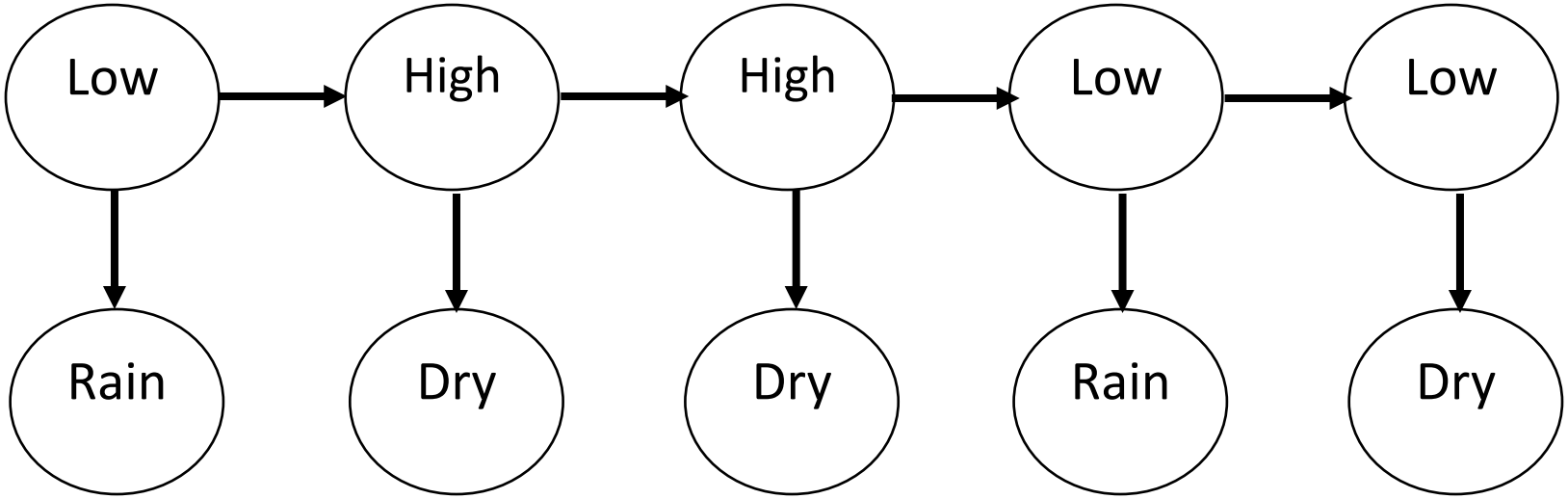
- Markov model parameter
  - transition probabilities: $a_{ij} = P(s_i \mid s_j)$
  - initial probabilities: $\pi_i = P(s_i)$

Markov Model



- Two states : Rain and Dry.
- Transition probabilities: P(Rain|Rain)=0.3 , P(Dry|Rain)=0.7 , P(Rain|Dry)=0.2, P(Dry|Dry)=0.8
- Initial probabilities: say P(Rain)=0.4 , P(Dry)=0.6.
- P({Dry, Dry, Rain, Rain}) =
  P(Dry) P(Dry|Dry) P(Rain|Dry) P(Rain|Rain)
  = 0.6 * 0.8 * 0.2 * 0.3

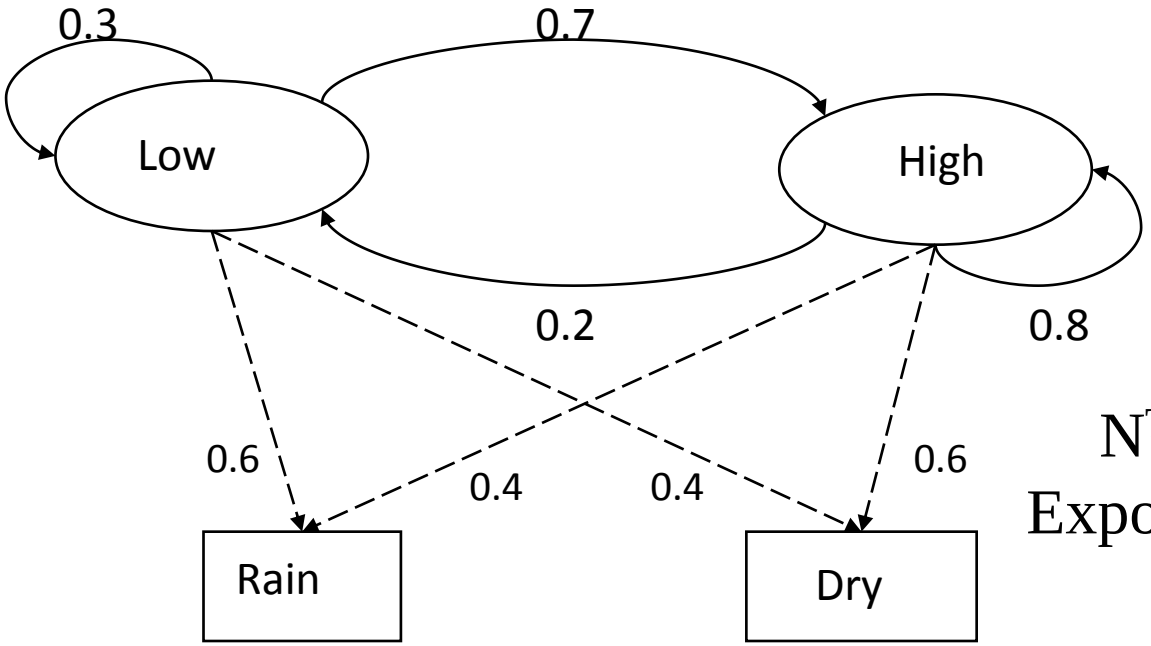
Hidden Markov Model (HMM)



- States are not visible, but each state randomly generates one of M observations (or visible states)
- Markov model parameter: M=(A, B, π)
  - Transition probabilities: $a_{ij} = P(s_i \mid s_j)$
  - Initial probabilities: $\pi_i = P(s_i)$
  - Emission probabilities: $b_i(v_m) = P(v_m \mid s_i)$

Hidden Markov Model (HMM)

Initial probabilities: P(Low)=0.4 , P(High)=0.6 .



N^T possible paths:
Exponential complexity!

P({Dry,Rain}) = P({Dry,Rain} , {Low,Low}) + P({Dry,Rain} , {Low,High}) + P({Dry,Rain} , {High,Low}) + P({Dry,Rain} , {High,High})

where first term is : P({Dry,Rain} , {Low,Low})=
P(Low)*P(Dry|Low)* P(Low|Low)*P(Rain|Low) = 0.4*0.4*0.3*0.6

Hidden Markov Model (HMM)

The Three Basic HMM Problems

- Problem 1 (Evaluation): Given the HMM: M=(A, B, π) and the observation sequence $O=o_1 o_2 \ldots o_K$, calculate the probability that model M has generated sequence O.

Forward algorithm

- Problem 2 (Decoding): Given the HMM: M=(A, B, π) and the observation sequence $O=o_1 o_2 \ldots o_K$, calculate the most likely sequence of hidden states $q_1 \ldots q_K$ that produced O.

Viterbi algorithm

Hidden Markov Model (HMM)

The Three Basic HMM Problems

- Problem 3 (Learning): Given some training observation sequences O and general structure of HMM (numbers of hidden and visible states), determine HMM parameters M=(A, B, π) that best fit the training data, that is maximizes P(O|M).

Baum-Welch algorithm (EM)

Hidden Markov Model (HMM)
Forward algorithm

Use Dynamic programming: Define the forward variable $\alpha_k(i)$ as the joint probability of the partial observation sequence $o_1\, o_2 \ldots o_k$ and that the hidden state at time k is s_i : $\alpha_k(i) = P(o_1\, o_2 \ldots o_k\, , q_k = s_i)$

- Initialization:

  $\alpha_1(i) = P(o_1\, , q_1 = s_i) = \pi_i\, b_i(o_1)$, $1 <= i <= N$.

- Forward recursion:

  Complexity : N^2T operations.

  $\alpha_{k+1}(i) = P(o_1\, o_2 \ldots o_{k+1}\, , q_{k+1} = s_j) =$

  $\Sigma_i\, P(o_1\, o_2 \ldots o_{k+1}\, , q_k = s_i\, , q_{k+1} = s_j) =$

  $\Sigma_i\, P(o_1\, o_2 \ldots o_k\, , q_k = s_i)\, a_{ij}\, b_j(o_{k+1}) =$

  $[\Sigma_i\, \alpha_k(i)\, a_{ij}]\, b_j(o_{k+1})$, $1 <= j <= N$, $1 <= k <= K-1$.

- Termination:

  $P(o_1\, o_2 \ldots o_T) = \Sigma_i\, P(o_1\, o_2 \ldots o_T\, , q_T = s_i) = \Sigma_i\, \alpha_T(i)$

Hidden Markov Model (HMM) Baum-Welch algorithm

If training data has information about sequence of hidden states, then use maximum likelihood estimation of parameters:

$$a_{ij} = P(s_i \mid s_j) = \frac{\text{Number of transitions from state } \mathbf{S}_j \text{ to state } \mathbf{S}_i}{\text{Number of transitions out of state } \mathbf{S}_j}$$

$$b_i(v_m) = P(v_m \mid s_i) = \frac{\text{Number of times observation } \mathbf{V}_m \text{ occurs in state } \mathbf{S}_i}{\text{Number of times in state } \mathbf{S}_i}$$

$$\pi_i = P(s_i) = \text{Number of times state } S_i \text{ occur at time k=1.}$$

Hidden Markov Model (HMM)
Baum-Welch algorithm

Using an initial parameter instantiation, the algorithm iteratively re-estimates the parameters to improve the probability of generating the observations

$$a_{ij} = P(s_i \mid s_j) = \frac{\text{Expected number of transitions from state } \mathbf{S}_j \text{ to state } \mathbf{S}_i}{\text{Expected number of transitions out of state } \mathbf{S}_j}$$

$$b_i(v_m) = P(v_m \mid s_i) = \frac{\text{Expected number of times observation } \mathbf{V}_m \text{ occurs in state } \mathbf{S}_i}{\text{Expected number of times in state } \mathbf{S}_i}$$

$\pi_i = P(s_i)$ = Expected Number of times state S_i occur at time k=1.

The algorithm uses iterative expectation-maximization algorithm to find local optimum solution

Temporal Data Mining

- Hidden Markov Model (HMM)
- Spectral time series representation
  - Discrete Fourier Transform (DFT)
  - Discrete Wavelet Transform (DWT)
- Pattern mining
  - Sequential pattern mining
  - Temporal abstraction pattern mining

DFT

- Discrete Fourier transform (DFT) transforms the series from the time domain to the frequency domain.
- Given a sequence x of length n, DFT produces n complex numbers:

$$X_f = \frac{1}{\sqrt{n}} \sum_{i=0}^{n-1} x_i * \exp\left(-\frac{j2\pi if}{n}\right) : f = 0, \dots, n-1$$

  Remember that $\exp(j\phi)=\cos(\phi) + j \sin(\phi)$.

- DFT coefficients (X_f) are complex numbers: $\mathrm{Im}(X_f)$ is sine at frequency f and $\mathrm{Re}(X_f)$ is cosine at frequency f, but X_0 is always a real number.
- DFT decomposes the signal into sine and cosine functions of several frequencies.
- The signal can be recovered exactly by the inverse DFT: $\vec{x} \Leftrightarrow \vec{X}$

DFT

- DFT can be written as a matrix operation where A is a n x n matrix:

$$\vec{X} = A \times \vec{x} \;:\; a_{i,f} = \frac{1}{\sqrt{n}} \exp\left(-\frac{j2\pi i}{n}\right) \;:\; i, f = 0, \ldots, n-1$$

A is column-orthonormal.

Geometric view: view series x as a point in n-dimensional space.

- A does a rotation (but no scaling) on the vector x in n-dimensional complex space:
  - Does not affect the length $\sum_{i=0}^{n-1} |x_i|^2 = \sum_{f=0}^{n-1} |X_f|^2$
  - Does not affect the Euclidean distance between any pair of points $\|\vec{X} - \vec{Y}\| = \|\vec{x} - \vec{y}\|$

DFT

- Symmetry property: $X_f=(X_{n-f})^*$ where * is the complex conjugate, therefore, we keep only the first half of the spectrum.

- Usually, we are interested in the amplitude spectrum ($|X_f|$) of the signal:

$$|X_f| = \sqrt{Re(X_f)^2 + Im(X_f)^2}$$

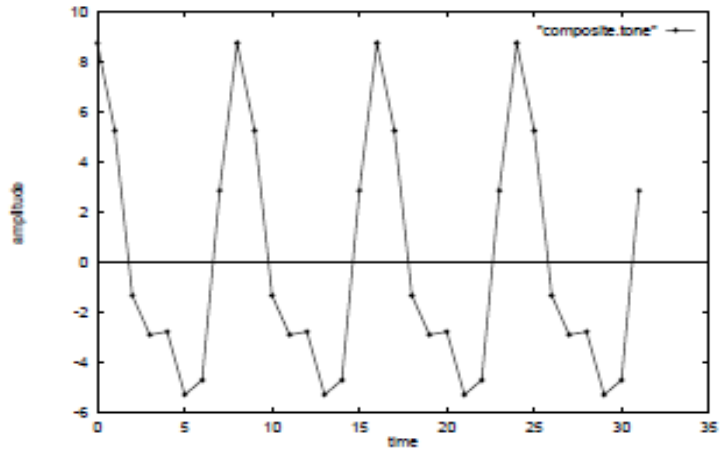
- The amplitude spectrum is insensitive to shifts in the time domain

- Computation:
  - Naïve: $O(n^2)$
  - FFT: $O(n \log n)$

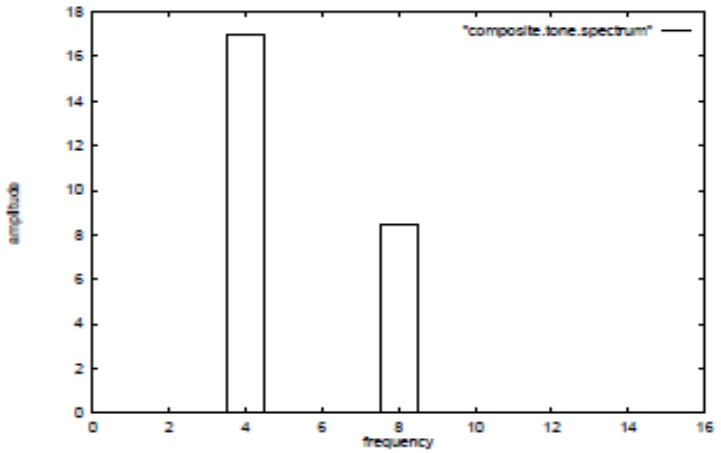
DFT

Example1: $x_i = 6\sin\left(\frac{2\pi 4i}{n} + 0.5\right) + 3\sin\left(\frac{2\pi 8i}{n}\right) : i = 0, \ldots, 31$

We show only half the spectrum because of the symmetry



(a) time domain

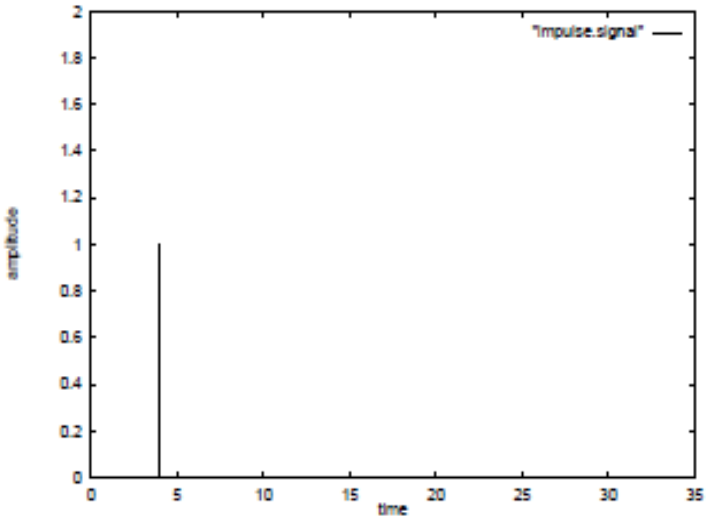


(b) amplitude spectrum

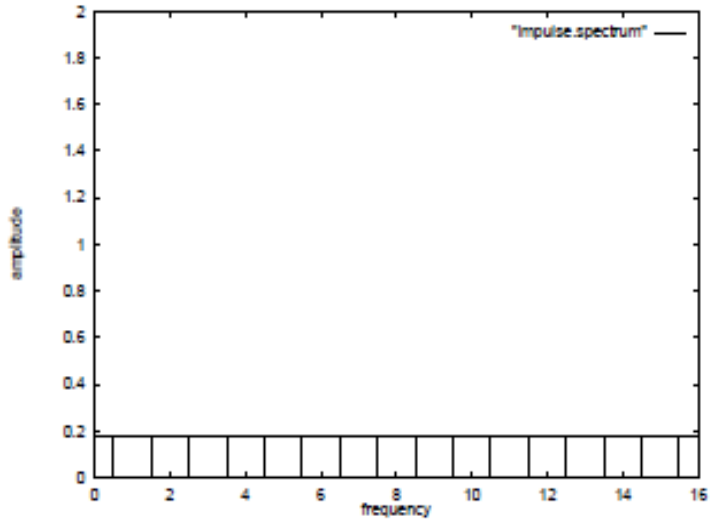
Very good compression!

DFT

Example2: the Dirac delta function.



(a) time domain



(b) amplitude spectrum

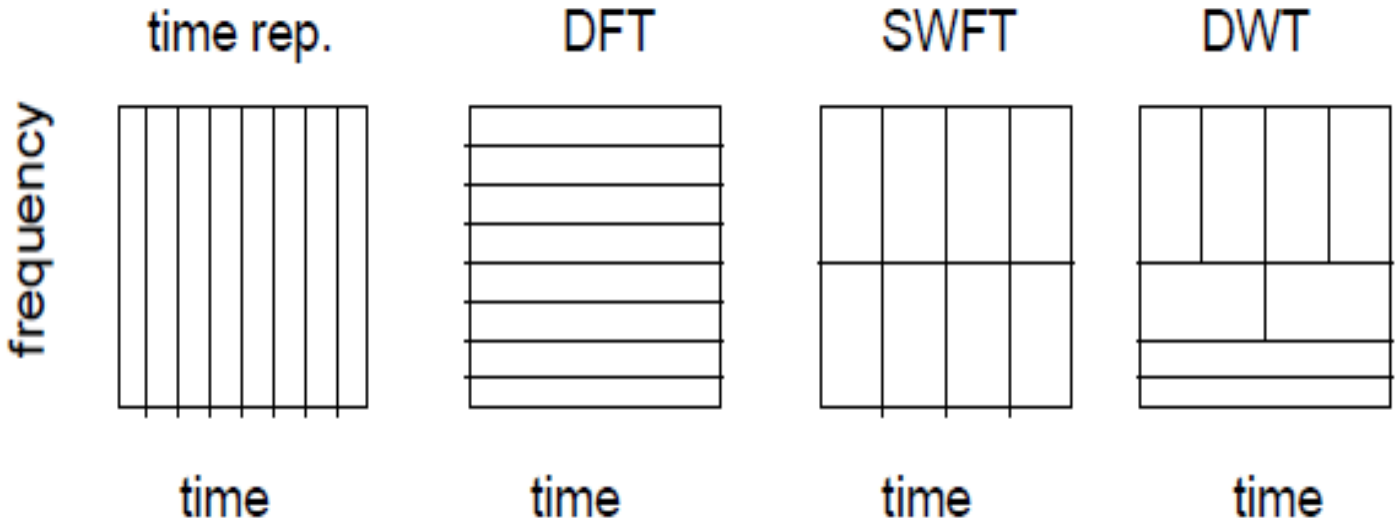
Horrible! The frequency leak problem

SWFT

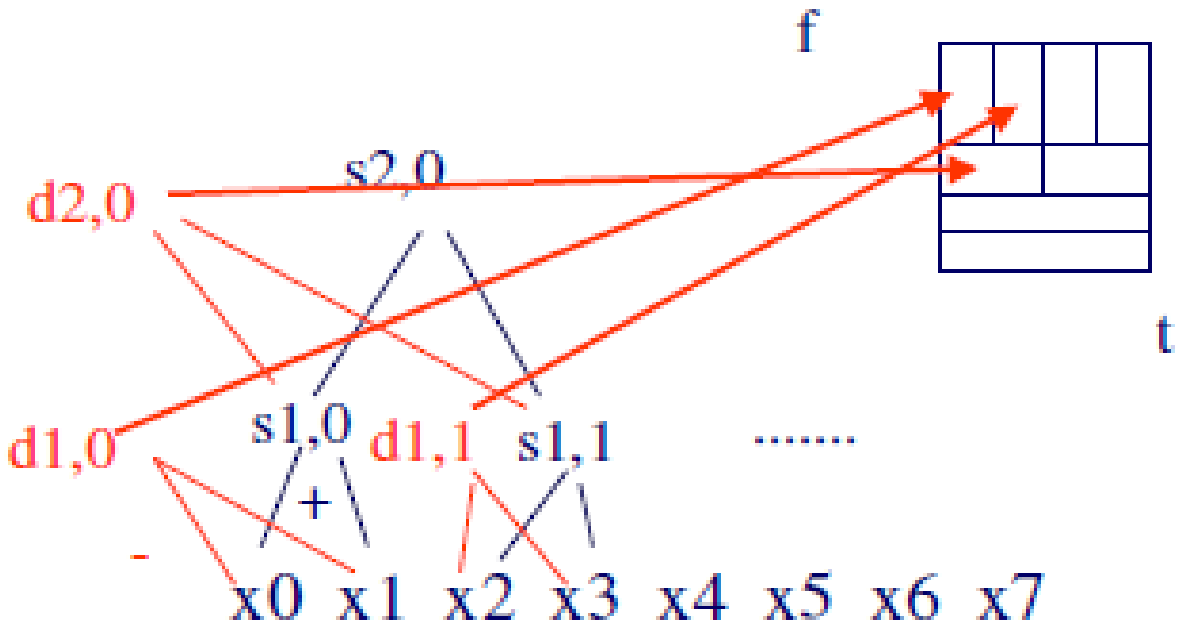
- DFT assumes the signal to be periodic and have no temporal locality: each coefficient provides information about all time points.
- Partial remedy: the Short Window Fourier Transform (SWFT) divides the time sequence into non-overlapping windows of size w and perform DFT on each window.
- The delta function have restricted ‘frequency leak’.
- How to choose the width w?
  - Long w gives good frequency resolution and poor time resolution.
  - Short w gives good time resolution and poor frequency resolution.
- *Solution: let w be variable ⟶ Discrete Wavelet Transform (DWT)*

DWT

- DWT maps the signal into a joint time-frequency domain.
- DWT hierarchically decomposes the signal using windows of different sizes (multi resolution analysis):
  - Good time resolution and poor frequency resolution at high frequencies.
  - Good frequency resolution and poor time resolution at low frequencies.



DWT: Haar wavelets

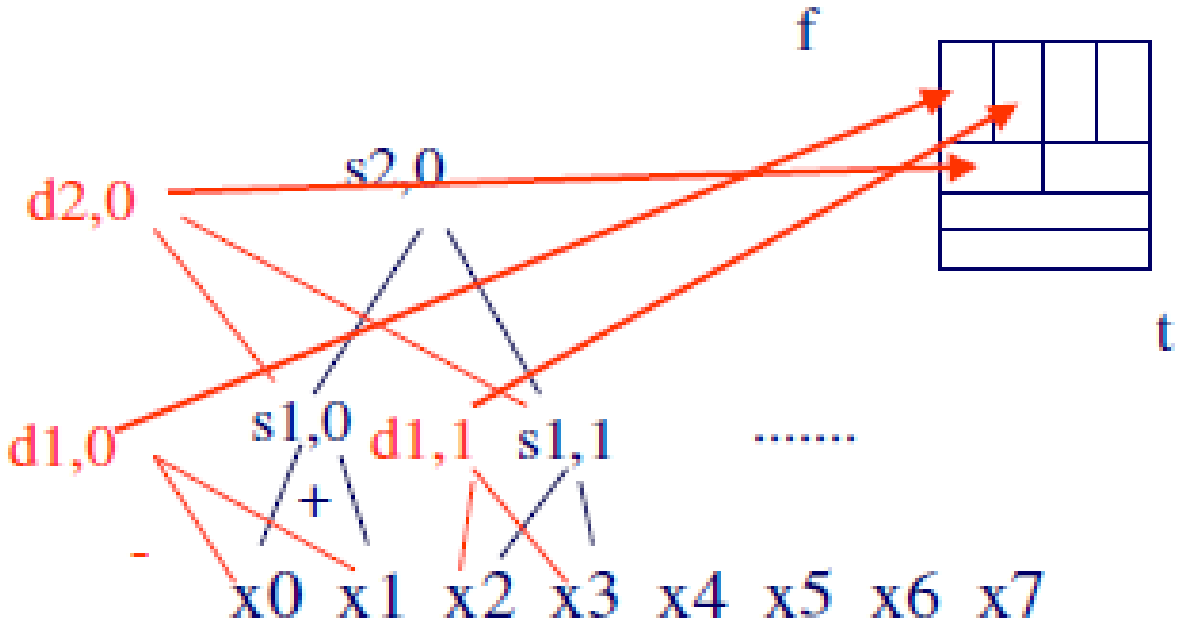


$$d_{l,i} = \frac{1}{\sqrt{2}}\left(s_{l-1,2i} - s_{l-1,2i+1}\right) \quad l = 0,..,L;\ i = 0,\ldots,\frac{n}{2^{l+1}} - 1$$

$$s_{l,i} = \frac{1}{\sqrt{2}}\left(s_{l-1,2i} + s_{l-1,2i+1}\right) \quad l = 0,..,L;\ i = 0,\ldots,\frac{n}{2^{l+1}} - 1$$

Initial condition: $s_{-1,i} = x_i$

DWT: Haar wavelets



Length of the series should be a power of 2: zero pad the series!

The Haar transform: all the difference values $d_{l,i}$ at every level l and offset i (n-1) difference, plus the smooth component $s_{L,0}$ at the last level

Computational complexity is O(n)

DFT and DWT

- Both DFT and DWT are orthonormal transformations → rotation in the space → do not affect the length or the Euclidean distance between the series → *clustering or classification in the transformed space will give the exact same result!*

- DFT/DWT are very useful for dimensionality reduction: usually a small number of low frequency coefficients can approximate well most time series/images.

- DFT/DWT are very useful for query by content using the GEMINI framework:
  - A quick and dirty filter (some false alarms, but no false dismissal).
  - A spatial index (e.g R-tree) using few DFT or DWT coefficients.

Related Time series representations

- Auto-correlation function (ACF)
- Singular Value Decomposition (SVD) [Chan and Fu, 1999].
- Piecewise Aggregate Approximation (PAA) [Yi and Faloutsos , 2000].
- Adaptive Piecewise Constant Approximation (APCA) [Keogh et al. 2001].
- Symbolic Aggregate Approximation (SAX) [Lin et al, 2003].
- Temporal abstractions (discussed later).

No representation is superior for all tasks: problem dependent!

Temporal Data Mining

- Hidden Markov Model (HMM)
- Spectral time series representation
  - Discrete Fourier Transform (DFT)
  - Discrete Wavelet Transform (DWT)
- Pattern mining
  - Sequential pattern mining
  - Temporal abstraction pattern mining

Sequential pattern mining

- A sequence is an ordered list of events, denoted $< e_1\ e_2\ \ldots\ e_L >$.
- Each event e_i is an unordered set of items.
- Given two sequences $\alpha = < a_1\ a_2\ \ldots\ a_n >$ and $\beta = < b_1\ b_2\ \ldots\ b_m >$

  α is called a subsequence of β, denoted as $\alpha \subseteq \beta$, if there exist integers $1 \leq j_1 < j_2 < \ldots < j_n \leq m$ such that $a_1 \subseteq b_{j1}$, $a_2 \subseteq b_{j2}, \ldots, a_n \subseteq b_{jn}$
  - Example: <a(bc)dc> is a *subsequence* of <a(abc)(ac)d(cf)>
- If a sequence contains *l* items, we call it a *l*-sequence
  - Example: <a(bc)dc> is a 5-sequence.
- The support of a sequence α is the number of data sequences that contain α.

Sequential pattern mining

- Given a set of sequences and support threshold, find the complete set of *frequent* subsequences, from which we extract temporal rules.
  - Examples: customers who buy a Canon digital camera are likely to buy an HP color printer within a month.

A *sequence database*

SID	sequence
1	<a(abc)(ac)d(cf)>
2	<(ad)c(bc)(ae)>
3	<(ef)(ab)(df)cb>
4	<eg(af)cbc>

Given support threshold *min_sup* =2, <(ab)c> is a *sequential pattern* (s is contained in sequences 1 and 3)

Sequential pattern mining
The GSP algorithm

GSP (Generalized Sequential Patterns: [Srikant & Agrawal 96]) is a generalization of Apriori for sequence databases.

Apriori property: If a sequence *S* is not frequent, then none of the super-sequences of *S* are not frequent.

- E.g, <hb> is infrequent so are <hab> and <(ah)b>

- Outline of the method
  - Initially, get all frequent 1-sequences
  - for each level (i.e., sequences of length-k) do
    - generate candidate length-(k+1) sequences from length-k frequent sequences
    - scan database to collect support count for each candidate sequence
  - repeat until no frequent sequence or no candidate can be found

Sequential pattern mining
The GSP algorithm

Finding Length-1 Sequential Patterns

- Initial candidates:
  - <a>, <b>, <c>, <d>, <e>, <f>, <g>, <h>
- Scan database once, count support for candidates

min_sup =2

Seq. ID	Sequence
10	<(bd)cb(ac)>
20	<(bf)(ce)b(fg)>
30	<(ah)(bf)abf>
40	<(be)(ce)d>
50	<a(bd)bcb(ade)>

Cand	Sup
<a>	**3**
<b>	**5**
<c>	**4**
<d>	**3**
<e>	**3**
<f>	**2**
~~<g>~~	1
~~<h>~~	1

Sequential pattern mining
The GSP algorithm

Generating Length-2 Candidates

	<a>	<b>	<c>	<d>	<e>	<f>
<a>	<aa>	<ab>	<ac>	<ad>	<ae>	<af>
<b>	<ba>	<bb>	<bc>	<bd>	<be>	<bf>
<c>	<ca>	<cb>	<cc>	<cd>	<ce>	<cf>
<d>	<da>	<db>	<dc>	<dd>	<de>	<df>
<e>	<ea>	<eb>	<ec>	<ed>	<ee>	<ef>
<f>	<fa>	<fb>	<fc>	<fd>	<fe>	<ff>

Number of candidate 2-sequences is 6*6+6*5/2=51 candidates

	<a>	<b>	<c>	<d>	<e>	<f>
<a>		<(ab)>	<(ac)>	<(ad)>	<(ae)>	<(af)>
<b>			<(bc)>	<(bd)>	<(be)>	<(bf)>
<c>				<(cd)>	<(ce)>	<(cf)>
<d>					<(de)>	<(df)>
<e>						<(ef)>
<f>						

Sequential pattern mining
The GSP algorithm

Candidate generation:

- Example1: join a and b:
  - Sequential pattern mining: ab, ba, (ab)
  - Itemset pattern mining: ab
- Example 2: join ab and ac:
  - Sequential pattern mining: abc, acb, a(bc)
  - Itemset pattern mining: abc

The number of candidates is much larger for sequential pattern mining!

Sequential pattern mining
The GSP algorithm

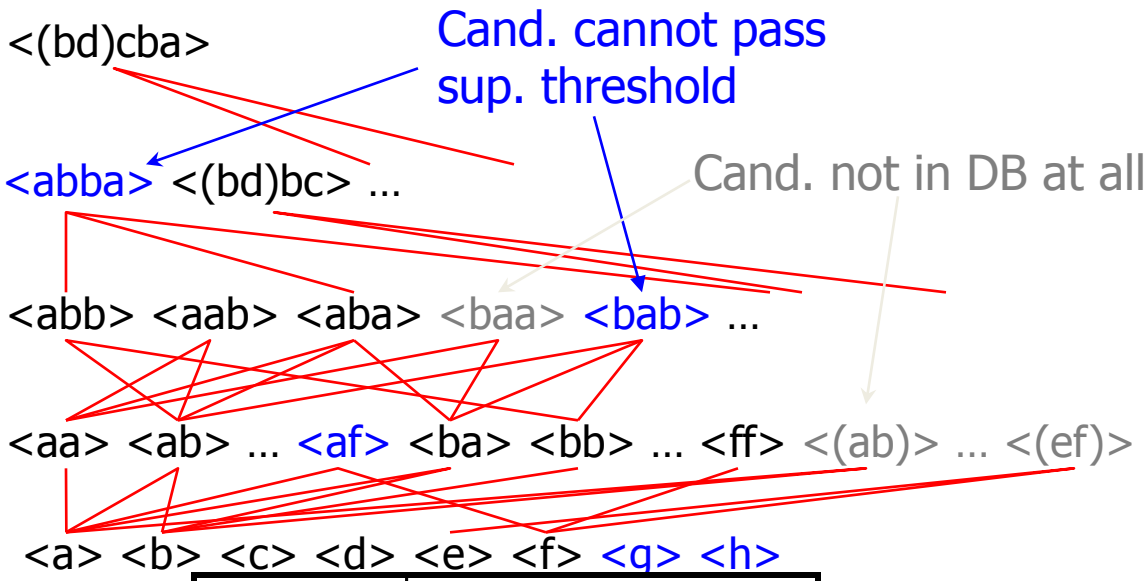
5th scan: 1 cand. 1 length-5 seq. pat.

4th scan: 8 cand. 6 length-4 seq. pat.

3rd scan: 46 cand. 19 length-3 seq. pat.

2nd scan: 51 cand. 19 length-2 seq. pat.

1st scan: 8 cand. 6 length-1 seq. pat.



min_sup =2

Seq. ID	Sequence
10	<(bd)cb(ac)>
20	<(bf)(ce)b(fg)>
30	<(ah)(bf)abf>
40	<(be)(ce)d>
50	<a(bd)bcb(ade)>

Sequential pattern mining

Other sequential pattern mining algorithms:

- SPADE
  - An Apriori-based and vertical data format algorithm.
- PrefixSpan
  - Does not require candidate generation (similar to FP-growth).
- CloSpan:
  - Mining Closed Sequential Patterns.
- Constraint based sequential pattern mining

Temporal abstraction

- Most of the time series representation techniques assume regularly sampled univariate time series data.
- Many real-world temporal datasets (e.g. clinical data) are:
  - Multivariate
  - Irregularly sampled in time
- It is very difficult to directly model this type of data.
- We want to apply methods like sequential pattern mining, but on multivariate time series data.
- Solution: use an abstract (qualitative) description of the series.

Temporal abstraction

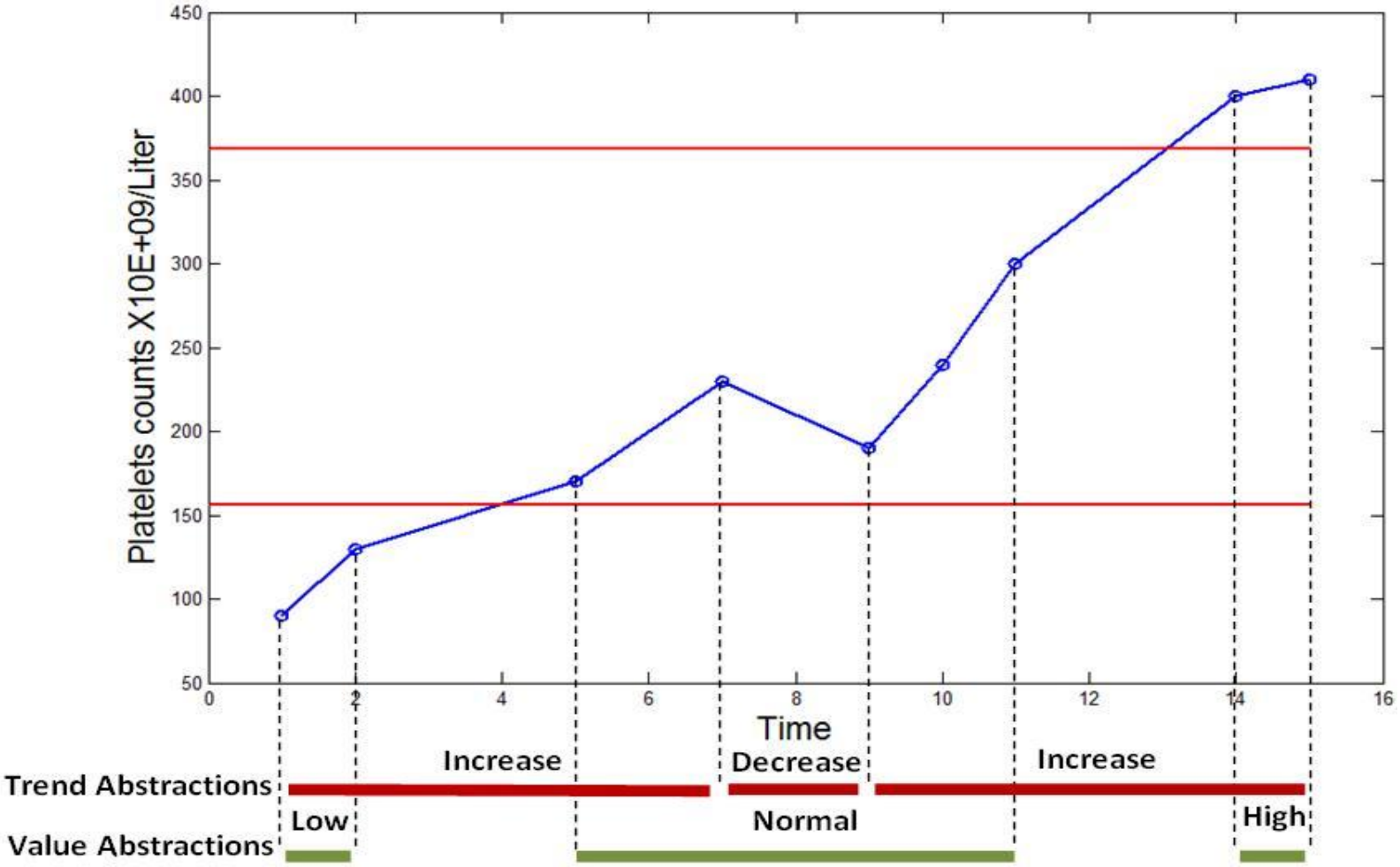
- Temporal abstraction moves from a *time-point* to an *interval-based* representation in a way similar to humans' perception of time series.

- Temporal abstraction converts (multivariate) time series T to state sequences S: $\{(s_1, b_1, e_1), (s_2, b_2, e_2), \ldots, (s_n, b_n, e_n)\}$ where s_i denotes an abstract state, $b_i < e_i$ and $b_i <= b_{i+1}$.

- Abstract states usually defines primitive shapes in the data, e.g.:
  - Trend abstractions: describe the series in terms of it local trends: {increasing, steady, decreasing}
  - Value abstractions: {high, normal, low}.

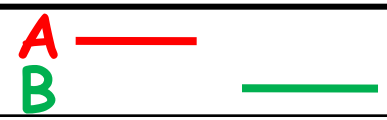
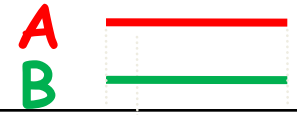
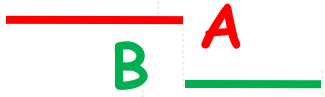
- These states are later combined to form more complex temporal patterns.

Temporal abstraction



Temporal relations

Allen's 13 temporal relations:

	**A before B**	**B after A**
	**A equals B**	**B equals A**
	**A meets B**	**A is-met-by B**
	**A overlaps B**	**A is-overlapped-by B**
	**A during B**	**B contains A**
	**A starts B**	**B is-started-by A**
	**A finishes B**	**B is-finished-by A**

Maybe too specific for some applications: can be simplified to fewer relations

Temporal abstraction patterns

- Combine the *abstract states* using *temporal relations* to form complex *temporal patterns*.
- Temporal pattern can be defined as a sequence of states (intervals) related using temporal relationships.
  - Example: P=low[X] *before* high[Y]
- These temporal patterns can be:
  - User defined [Lucia et al. 2005]
  - Automatically discovered [Hoppner 2001, Batal et al 2009].

Temporal abstraction patterns mining (sketch)

- Sliding window option: interesting patterns can be limited in their temporal extensions.
- More complicated (larger search space) than sequential pattern mining because we have many temporal relations.
- We got Frequent temporal patterns, so what?
  - Extract temporal rules
    - inc[x] overlaps dec[y] ⇒ low[z]: sup=10%, conf=70%.
    - knowledge discovery or prediction
  - Use discriminative temporal patterns for classification
  - Use temporal patterns to define clusters
  - …