ifis
Institut für Informationssysteme
Technische Universität Braunschweig

Data Warehousing & Mining Techniques

Wolf-Tilo Balke
Mayukh Das
Institut für Informationssysteme
Technische Universität Braunschweig
https://www.tu-braunschweig.de/ifis

Summary

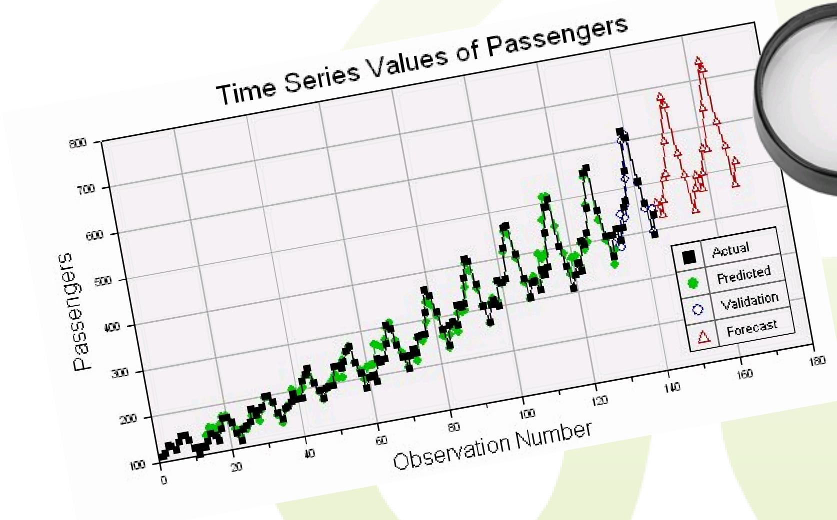
- Association Rule Mining
  - Apriori algorithm, support, confidence, downward closure property
  - Multiple minimum supports solve the "rare-item" problem
  - Head-item problem

10. Data Mining

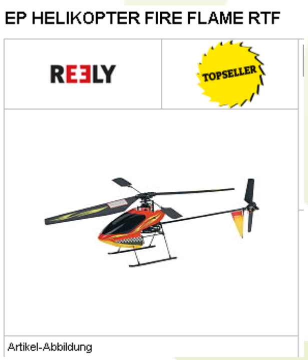
10. Data Mining

10.1 Mining Sequence Patterns

10.2 Mining Time-Series Data

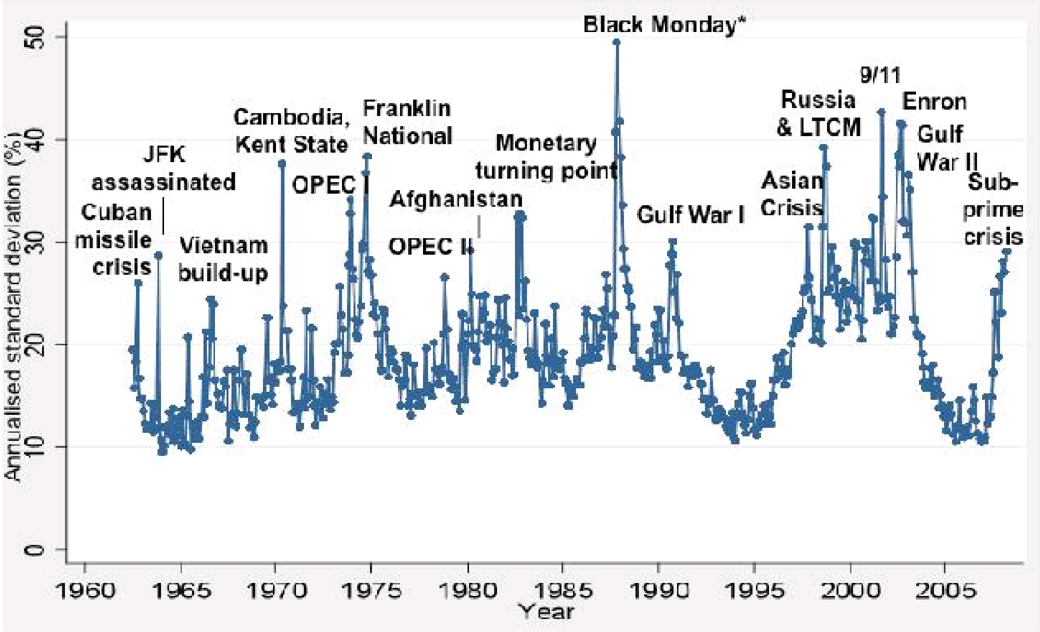


- Sequential pattern mining
  - Mining of frequently occurring ordered events or subsequences as patterns
  - Example
    - Customers who buy helicopter models in some on-line store receive e-mail promotions
      - Regarding batteries
      - Then after a while regarding rotor wings, since most of them will break



10.1 Mining Sequence Patterns

- Sequential pattern mining
  - Applications
    - Customer retention, targeted marketing
    - Ranging from disasters (e.g. earthquakes, wars) to market prediction



10.1 Mining Sequence Patterns

- Mining sequence patterns, vocabulary
  - Let $I=\{I_1, I_2, \ldots, I_p\}$ be the set of all items
  - An **itemset** is a nonempty set of items from I
  - A **sequence S** is an ordered list of events
    - Denoted $<e_1e_2e_3\ldots e_k>$, where event e_1 occurs before e_2 etc.
  - An **event** is an itemset, i.e. an unordered list of items
    - E.g., $(I_2I_1I_3)$, where $I_1, I_2, I_3 \in I$

10.1 Mining Sequence Patterns

– E.g., a customer bought items (abc) at a store. This is an event e_1. Now if later he buys another itemset (ade), representing a second event e_2, we obtain a shopping sequence s
  - $e_1=(abc)$, $e_2=(ade)$
  - $s=<e_1e_2>=<(abc)(ade)>$

– The number of instances of items in a sequence is called the **length** of the sequence
  - Length of s is 6

– A sequence with length k is called a **k-sequence**

10.1 Mining Sequence Patterns

– **Subsequence & supersequence**

- A sequence α=<$a_1a_2\ldots a_n$> is called a **subsequence** of another sequence β=<$b_1b_2\ldots b_m$> denoted $\alpha \sqsubseteq \beta$ (β is called **supersequence** of α)
  if there exist integers $1 \leq j_1 < j_2 < \ldots < j_n \leq m$ such that
  $a_1 \subseteq b_{j1}, a_2 \subseteq b_{j2}, \ldots, a_n \subseteq b_{jn}$
- E.g., if α=<(ab)d> and β=<(abc)(de)> then $\alpha \sqsubseteq \beta$

– **Sequence database**

- A sequence database S is a set of tuples <SID, s>
- E.g., contains the sequences for all customers of the store

10.1 Mining Sequence Patterns

- **Support of a sequence** in a sequence database
  - The support of α in S is the number of tuples in S, containing α
  - $sup_S(\alpha) = |\{<SID,s> | (<SID,s> \in S) \wedge (\alpha \sqsubseteq s)\}|$
- **Frequent sequence**
  - α is a frequent sequence if $sup_S(\alpha) \geq min\_sup$, where min_sup is the **minimum support threshold**
- A frequent sequence is called a **sequence pattern**
  - A sequence pattern of length k is called an **k-pattern**

10.1 Mining Sequence Patterns

- Sequence patterns, example
  - Given
    - I={a, b, c, d, e, f, g}, min_sup=2 and the sequence table
  - Length of <a(abc)(ac)d(cf)> is 9 and although there are 3 ‘a’ items in the first 3 events from record 1, it contributes to the sup(a) with just 1

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

10.1 Mining Sequence Patterns

- Sequence patterns, example
  - <a(bc)df> is a subsequence of the first record
    - <a(bc)df> ⊑ <a(abc)(ac)d(cf)>
  - sup(<(ab)c>) = 2
    - <(ab)c> ⊑ <a(abc)(ac)d(cf)> and <(ab)c> ⊑ <(ef)(ab)(df)cb>
  - If min_sup = 50%, <(ab)c> is a **sequential pattern** or a **3-pattern** (i.e. it has length 3)

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

10.1 Mining Sequence Patterns

- **Challenges** of sequence pattern mining
  - A huge number of possible sequential patterns are hidden in databases
  - A mining algorithm should
    - Find the **complete set of patterns**, when possible, satisfying the minimum support threshold
    - Be **highly efficient**, scalable, involving only a small number of database scans
    - Be able to incorporate various kinds of user-specific constraints

10.1 Mining Sequence Patterns

- **Algorithms**
  - Apriori-based method
    - Generalized Sequential Patterns (GSP)
  - Pattern-growth methods
    - FreeSpan & PrefixSpan
  - Vertical format-based mining
    - Sequential Pattern Discovery using Equivalent classes (SPADE)
  - Mining closed sequential patterns
    - CloSpan

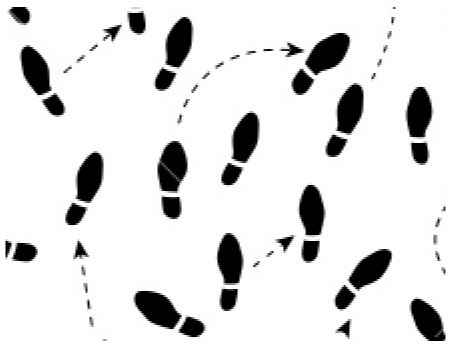
10.1 Mining Sequence Patterns

- **Generalized Sequential Patterns** (GSP)
  - Based on the Apriori property of sequential patterns
    - Downward closure: If a sequence *s* is not frequent then **none** of its super-sequences can be frequent
    - E.g., let min_sup=2; if <hb> is infrequent then <hab> and <(ah)b> are also infrequent!

<hb> is a subset of only record 3

SID	Sequence
1	<(bd)cb(ac)>
2	<(bf)(ce)b(fg)>
3	<(ah)(bf)abf>
4	<(be)(ce)d>
5	<a(bd)bcb(ade)>

10.1 GSP



- GSP algorithm, 2 step description
  - Initial step
    - Every item in the sequence database is a candidate of length 1
  - Generalization
    - Scan database to collect support count for each k length, candidate sequence, and establish the k-patterns
    - Generate candidate sequences of length $(k+1)$ from k-patterns using the Apriori property
    - Repeat this generalization step until no more candidates can be found e.g., there are no more k length frequent sequences

10.1 GSP

min_sup = 2

SID	Sequence
1	<(bd)cb(ac)>
2	<(bf)(ce)b(fg)>
3	<(ah)(bf)abf>
4	<(be)(ce)d>
5	<a(bd)bcb(ade)>

- Initial step
  - All singleton sequences are <a>, <b>, <c>, <d>, <e>, <f>, <g>, <h>
- General step, k = 1
  - Scan database once, count support for candidates
  - <g> and <h> are not 1-patterns since sup(<g>) = 1 < min_sup = 2 sup(<h>) = 1 < min_sup = 2
  - According to the Apriori property: since <g> and <h> are not 1-patterns, they can't form any 2-patterns. So they can be removed!

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

10.1 GSP

– General step, k = 1, generate length 2 candidates

• First generate 2 event candidates

– 6*6 = 36 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>

• Then generate 1 event candidates, each with 2 items

– 6*5/2 = 15 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>						

10.1 GSP

– k = 2, we have 51 2-length candidates
  - After the second table scan we remain with 19 2-patterns
  - Then we generate candidates for length 3, and so on…
  - <(bd)cba> is a 5-pattern, meaning that events (bd), c, b and a were frequent in the table, in this order

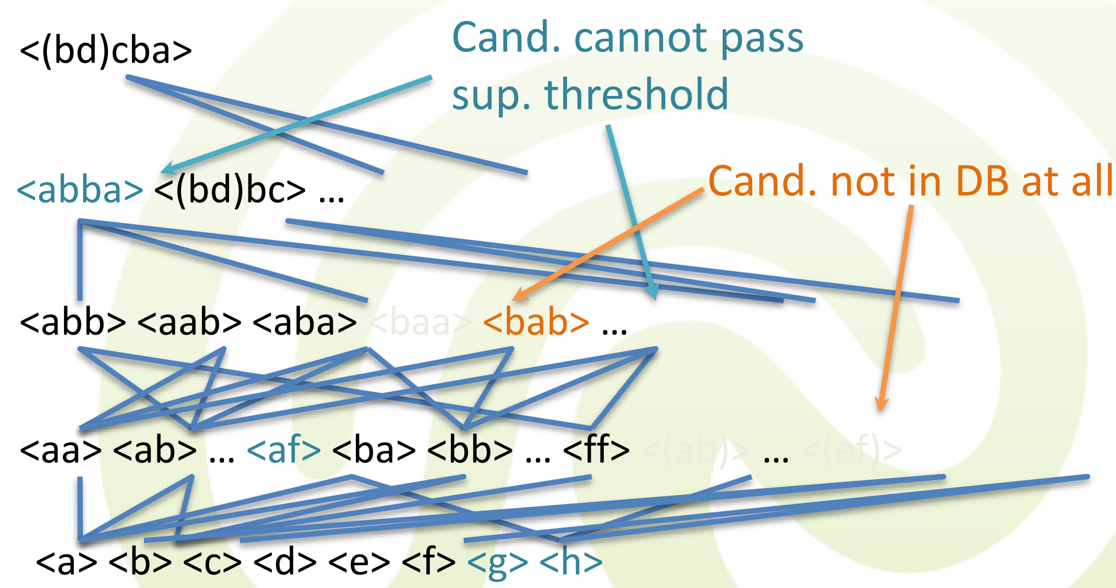
5th scan: 1 cand. 1 5-pattern

4th scan: 8 cand. 6 4-patterns

3rd scan: 47 cand. 19 3-patterns,
20 cand. not in DB at all

2nd scan: 51 cand. 19 2-patterns
10 cand. not in DB at all

1st scan: 8 cand. 6 1-patterns



10.1 GSP

- **Join and Prune**
  - Analogous to Apriori we can use join and prune to reduce the number of candidates C_k
    - Less candidates mean less minimum supports to compute
  - **Prune** works the same way as in the Apriori algorithm
  - For **join** we have to account for the order of the events since they are not simply ordered lexicographically
    - Example: how to join <ab> and <ac>? In Apriori it would be "<abc>" but here we are stuck with <abc> and <acb>

10.1 GSP

- Joining sequences
  - Idea: From the sequences s_1 and s_2 **remove** the **first item from s_1** and **the last item from s_2**. If the remaining subsequences are identical you can join s_1 and s_2.
  - Examples:
    - <bc> and <ca> can be joined to <bca>
    - <abc> and <cbe> can **not** be joined

10.1 GSP

- Drawbacks of GSP
  - A **huge set** of candidate sequences generated
    - Especially 2-item candidate sequence
  - Multiple scans of database needed
    - The length of each candidate grows by one for each database scan
  - Inefficient for mining long sequential patterns
    - Long patterns grow from short patterns
    - The number of short patterns is exponential in the length of mined patterns

10.2 Time-Series Data

- Sequence patterns mining
  - Are ordered events
  - No **concrete notion of time**

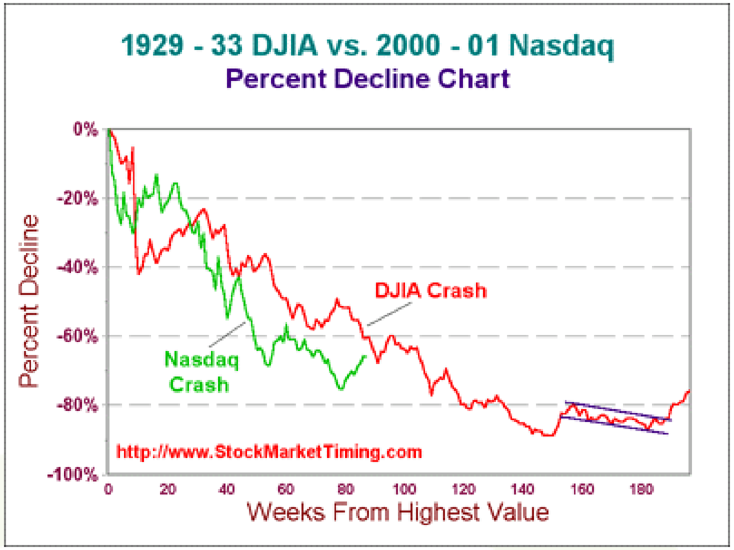
- Combining sequences of events with repeated measurements of **time** (at equal time intervals) we obtain **time-series** data

10.2 Time-Series Data

- **Time-series databases**
  - Time series reveal **temporal behavior** of the underlying mechanism that produced the data
  - Consists of sequences of values or events **changing with time**
  - Data is recorded at **regular intervals**

10.2 Time-Series Data

- Applications
  - Financial
    - Stock market, sales forecasting, inflation
  - Industry
    - Power consumption, workload projections, process and quality control
  - Meteorological
    - Observation of natural phenomena such as precipitation, temperature, wind, earthquakes



10.2 Time-Series Data

- **Goals** of time-series data analysis
  - **Modeling** time-series
    - Get insight into the mechanisms or underlying forces that generate the time series
  - **Forecasting** time-series
    - Predict the future values of the time-series variables
- Methods
  - Trend analysis
  - Similarity search

10.2 Trend Analysis

- **Trend analysis**
  - Application of **statistical techniques** e.g., regression analysis, to make and justify statements about trends in the data
  - Construct a **model**, independent of anything known about the physics of the process, to explain the behavior of the measurement
    - E.g., increasing or decreasing trend, that can be statistically distinguished from random behavior: take daily average temperatures at a given location, from winter to summer

10.2 Trend Analysis

- **Regression analysis** (RA)
  - Popular tool for modeling time series, finding trends and outliers in data sets
  - Analysis of numerical data consisting of values of a **dependent variable** (also called a response variable) and of one or more **independent variables**
    - The dependent variable in the regression equation is modeled as a function of the independent variables, corresponding parameters ("constants“) and an error term

10.2 Regression Analysis

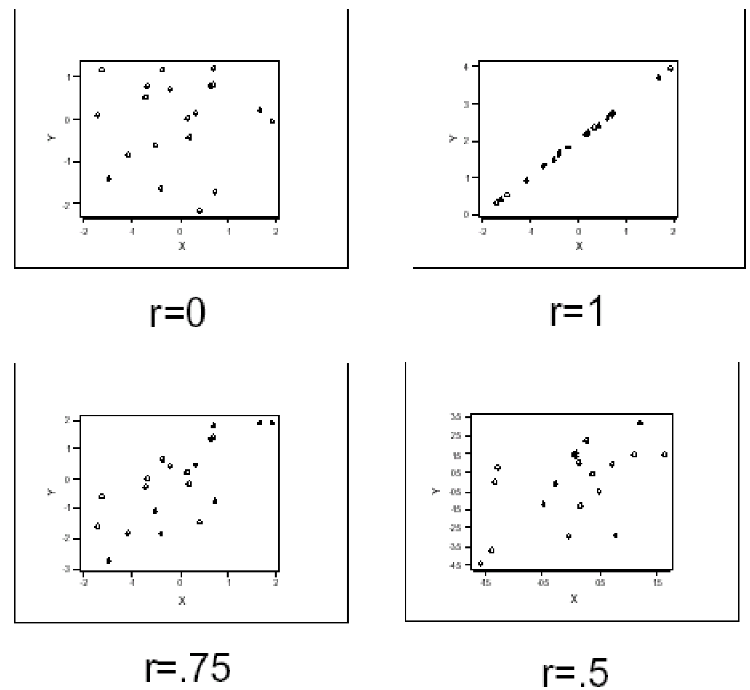
- RA, example: determine appropriate **levels of advertising** for a particular market segment
  - Consider the problem of managing sales of beer at large college campuses
    - Sales over one semester might be influenced by ads in the college paper, ads on the campus radio station, sponsorship of sports-related events, sponsorship of contests, etc.
    - Use data on advertising and promotional expenditures at many different campuses to extract the marginal value of dollars spent in each category

10.2 Regression Analysis

– Set up a model of the following type:
  - sales = b_0 + b_1(print budget) + b_2(radio budget) + b_3(sports promo budget) + b_4(other promo) + error
– This model is called **linear regression analysis**
  - $Y = b_0 + b_1X_1 + b_2X_2 + \ldots + b_nX_n$
  - Y = predicted score
  - b_0 = intercept/origin of regression line
  - b_i = regression coefficient representing unit of change in dependent variable with the increase in 1 unit on X variable

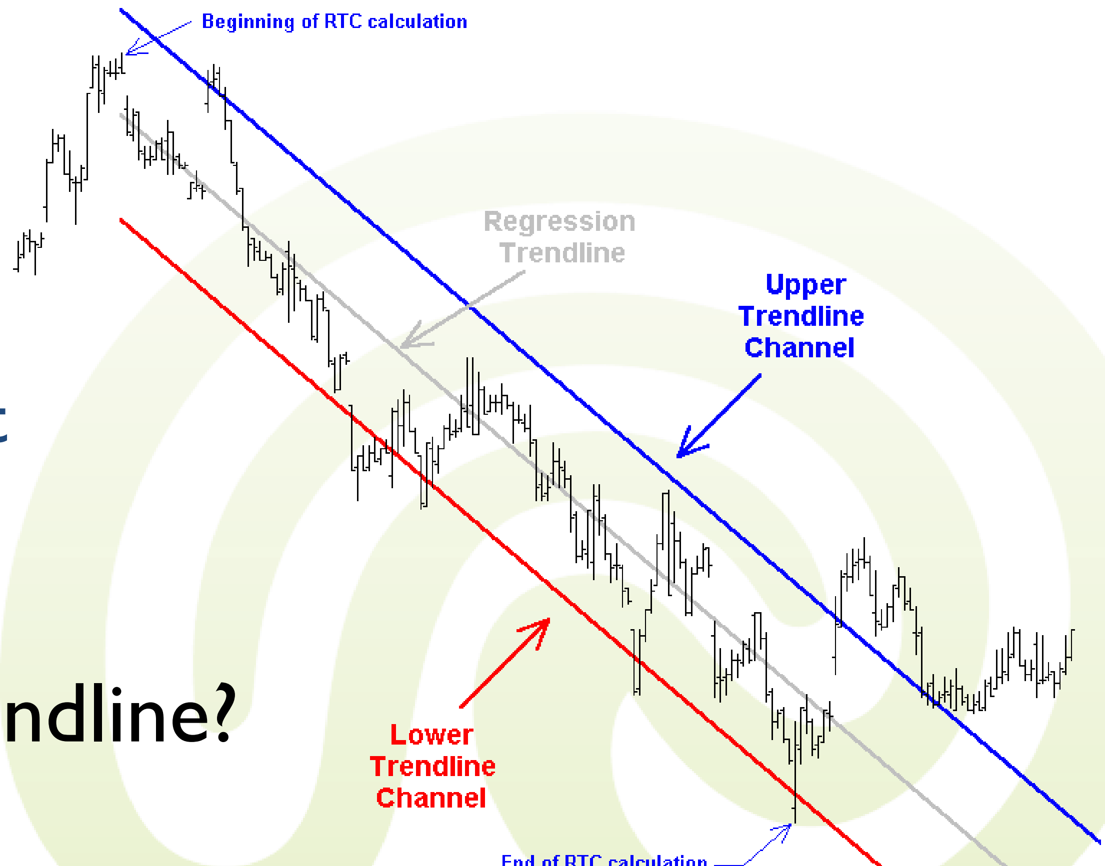
10.2 Regression Analysis

– Correlation (noted R)
  - Refers to the interdependence or co-relationship of variables
  - Reflects the accuracy of the linear relationship between X and Y
  - Lies between -1 and 1 with:
    -1 is anti-correlated
    0 is independent
    1 is linearly correlated



10.2 Regression Analysis

- Regression trend channels (RTC)
  - Very useful in defining and containing the trend of the market
  - When the prices break a well established trend channel, the market usually changes trend
- Upper & Lower trendline?



10.2 Regression Analysis

- What is RTC?
  - The mathematical **standard deviation** of the linear regression
  - Basically it is made up of three parallel lines
    - The center line is the linear regression line
    - This center line is bracketed by two additional lines that represent the +/- standard deviation of the linear regression data

10.2 Regression Analysis

- The linear regression model is the most simple model, but there are others
  - **Nonlinear** regression (the model function is not linear in the parameters), **Bayesian** methods, etc.
- Regression analysis can't capture all trend movements that occur in real-world applications
  - The solution is to **decompose** time-series into **basic movements**

10.2 Trend Analysis

- Basic movements are characteristic **time-series movements** (often called **components**)
  - Trend (T)
    - Reflects the long term progression of the series
  - Seasonal (S)
    - Seasonal fluctuations i.e., almost identical patterns that a time series appears to follow during corresponding months of successive years
  - Cycle (C)
    - Describes regular fluctuations caused by the economic cycle e.g., business cycles
  - Irregular (I)
    - Describes random, irregular influences

10.2 Trend Analysis

- Time-series **decomposition**
  - Additive Model
    - Time-series = T + C + S + I
  - Multiplicative Model
    - Time-series = T × C × S × I
- To perform decomposition we must identify each of the 4 movements in the time-series

10.2 Trend Analysis

- **Trend analysis (T)**, methods
  - The **freehand** method
    - Fit the curve by looking at the graph
    - **Costly** and barely reliable for large-scaled data mining
  - The **least-square** method
    - Find the curve minimizing the sum of the squares of the deviation of points on the curve from the corresponding data points
  - The **moving-average** method
    - Eliminates cyclic, seasonal and irregular patterns
    - Loss of end data
    - Sensitive to outliers

10.2 Trend Analysis

– **Moving average** (MA) of order n

$$\frac{y_1 + y_2 + \cdots + y_n}{n}, \frac{y_2 + y_3 + \cdots + y_{n+1}}{n}, \frac{y_3 + y_4 + \cdots + y_{n+2}}{n}, \cdots$$

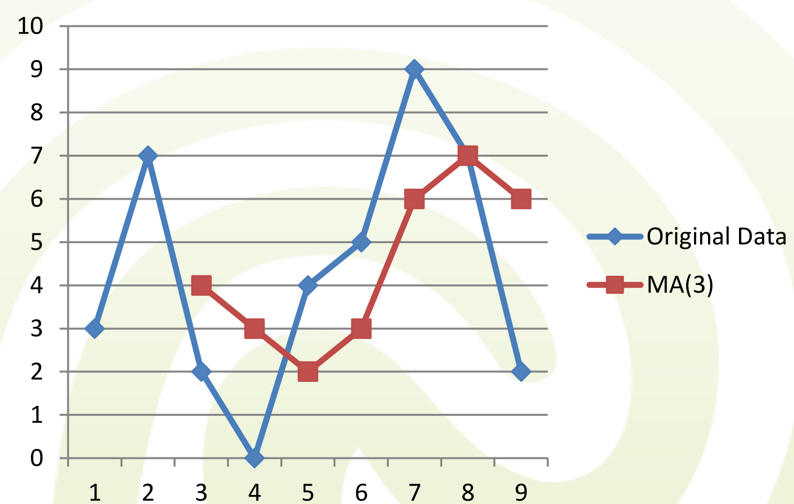
- E.g.,

Original data	MA(3)
3	
7	
2	4
0	3
4	2
5	3
9	6
7	7
2	6

(3+7+2)/3

(7+2+0)/3

…



10.2 Moving Average

– Influence of extreme values can be reduced with **weighted moving average** (WMA)
  - WMA is MA with weights e.g., WMA(3) with (1,4,1) as weights

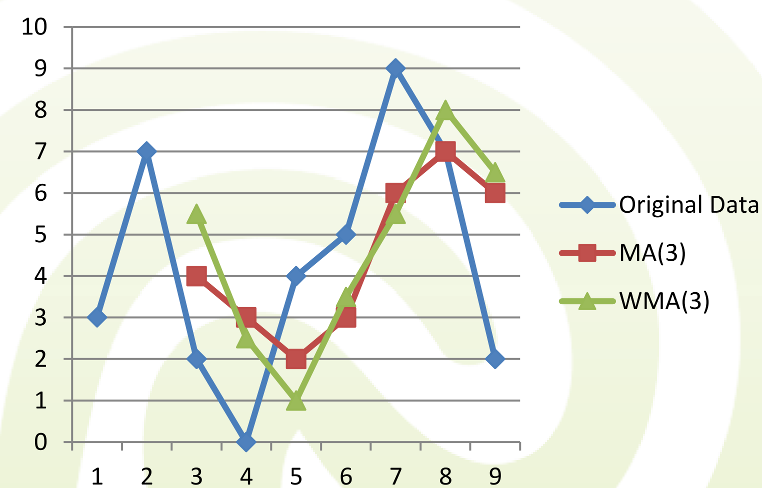
Original data
3
7
2
0
4
5
9
7
2

(3*1+7*4+2*1)/(1+4+1)

(7*1+2*4+0*1)/(1+4+1)

…

WMA(3)
5.5
2.5
1
3.5
5.5
8
6.5



10.2 Moving Average

- Other forms of MA
  - **Cumulative moving average** (CA), also called long running average

$$CA_i = \frac{x_1 + \cdots + x_i}{i}.$$

$$CA_{i+1} = CA_i + \frac{x_{i+1} - CA_i}{i+1}.$$

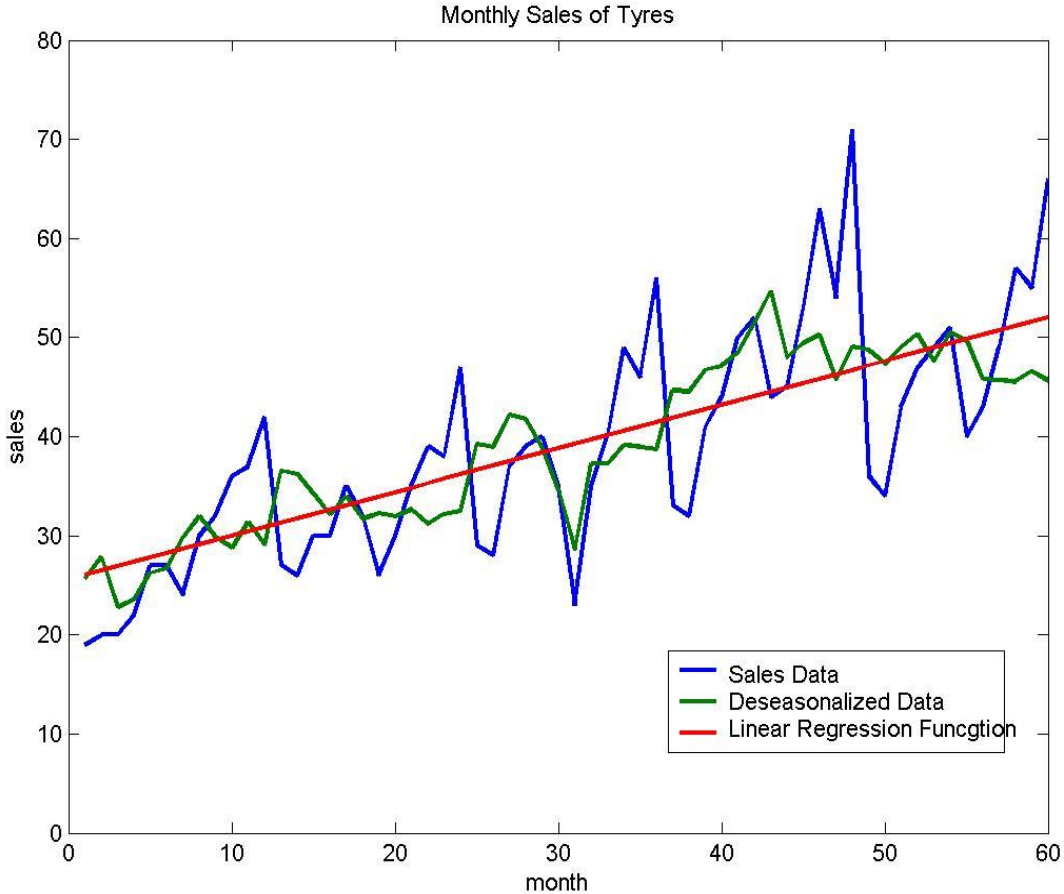
  - **Exponential weighted moving average** (EWMA), applies weighting factors which decrease exponentially
    - Gives much more importance to recent observations while still not discarding older observations entirely

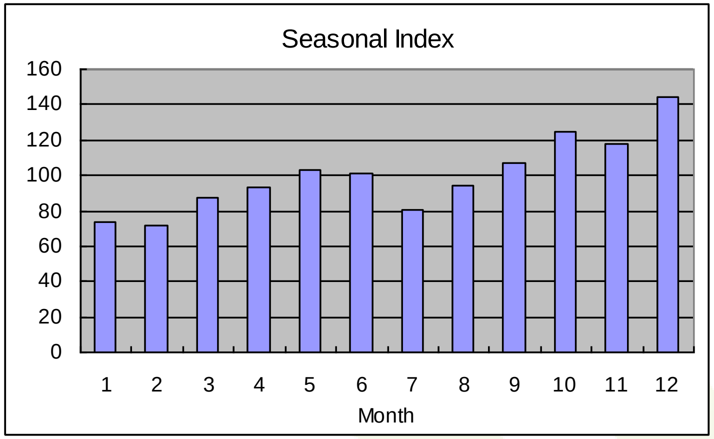
10.2 Trend Analysis

- Estimation of **seasonal variations (S)**
  - Seasonal index
    - Set of numbers showing the relative values of a variable during the months of the year
    - E.g., if the sales during October, November, and December are 80%, 120%, and 140% of the average monthly sales for the whole year, respectively, then 80, 120, and 140 are seasonal index numbers for these months
  - Deseasonalized data
    - Data adjusted for seasonal variations
    - E.g. divide or subtract the original monthly data by the seasonal index numbers for the corresponding months

10.2 Trend Analysis

- Estimation of **seasonal variations (S)**





10.2 Trend Analysis

- Estimation of **cyclic variations (C)**
  - If (approximate) periodicity of cycles occurs, cyclic index can be constructed in much the same manner as seasonal indexes
- Estimation of **irregular variations (I)**
  - By adjusting the data for trend, seasonal and cyclic variations
- With the systematic analysis of the trend, cyclic, seasonal, and irregular components, it is possible to make **long- or short-term predictions** (time-series forecasting) with reasonable quality

10.2 Trend Analysis

- Time-series **forecasting**
  - Finds a mathematical formula that will approximately generate the historical patterns
  - Forecasting models: most popular is **auto-regressive integrated moving average** (ARIMA)
    - ARIMA can be applied in cases where data shows evidence of non-stationarity

10.2 Trend Analysis

- Applications of trend analysis: large corporations selling their products world-wide
  - Products are sold in different countries with different currencies
  - Currency has to be exchanged back and forth
    - The cost of the currency exchange has to be kept under control!
    - Timing is everything in foreign exchange

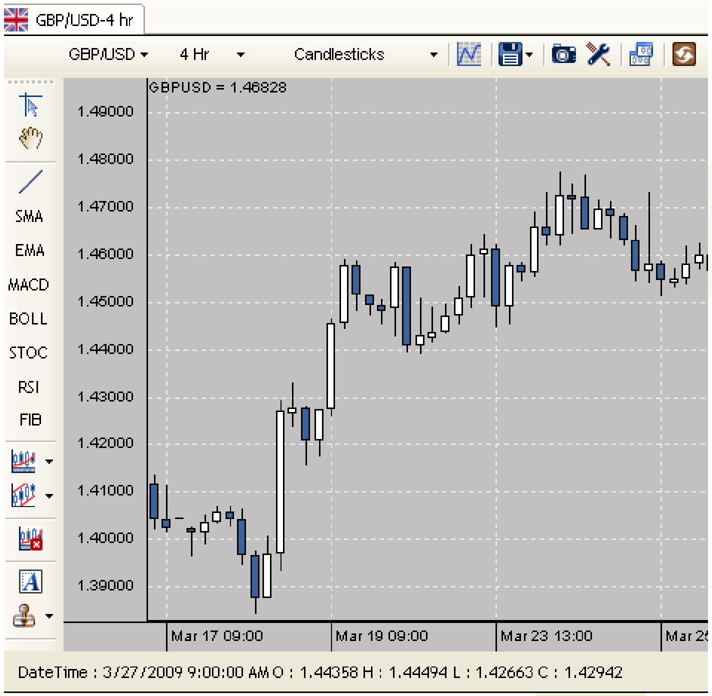
10.2 Trend Analysis

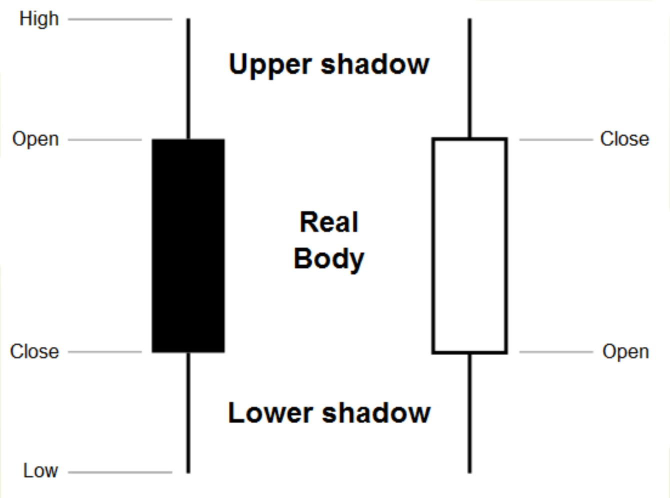
Detour

- Foreign exchange market (FOREX)
  - High data volume
  - Small granularity – ticks milliseconds away
- Transform data to an adequate granularity e.g., 4 hours a candle for FOREX

10.2 Trend Analysis

- Granularity change
  - Use Japanese candlesticks (developed in the 16th century by Japanese rice traders) for data charting

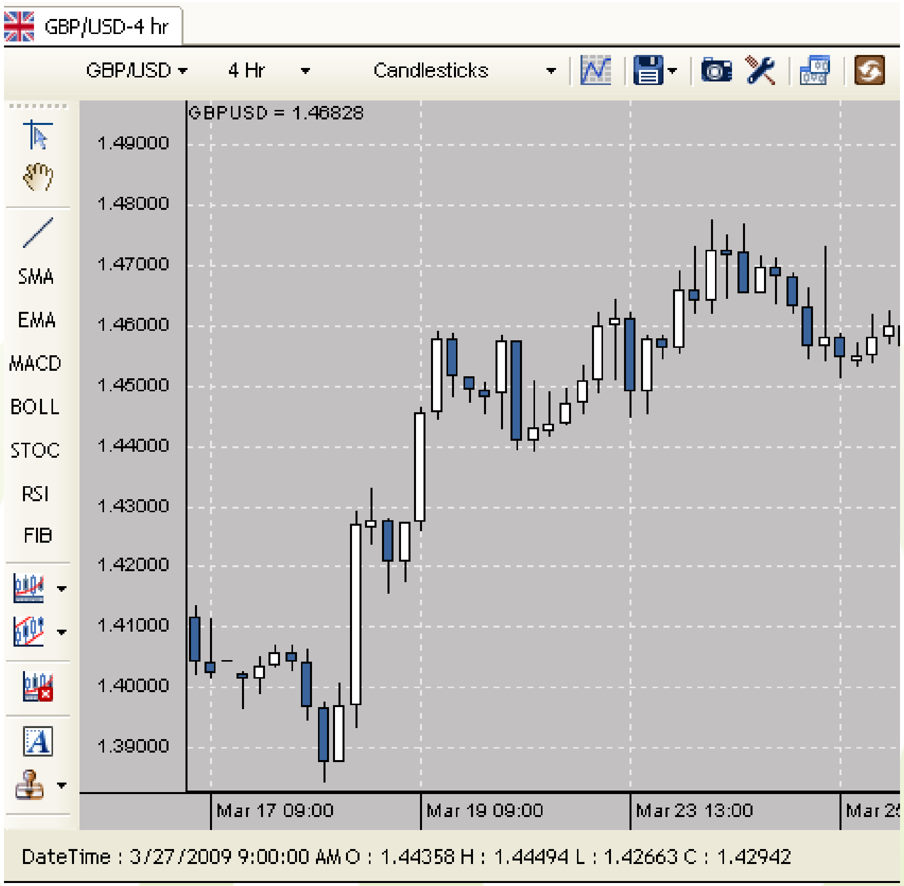




10.2 Trend Analysis

Detour

- When trading the goal is to buy low and sell high!
  - **Use trends to trade!**



10.2 Trend Analysis

Detour

- Why do we need trends? Once we have found a trend, we can:
  - **Open position when in the trend** (buy if it will go up, or sell if it will go down)
  - **Close the position on the trend turns**

10.2 Trend Analysis

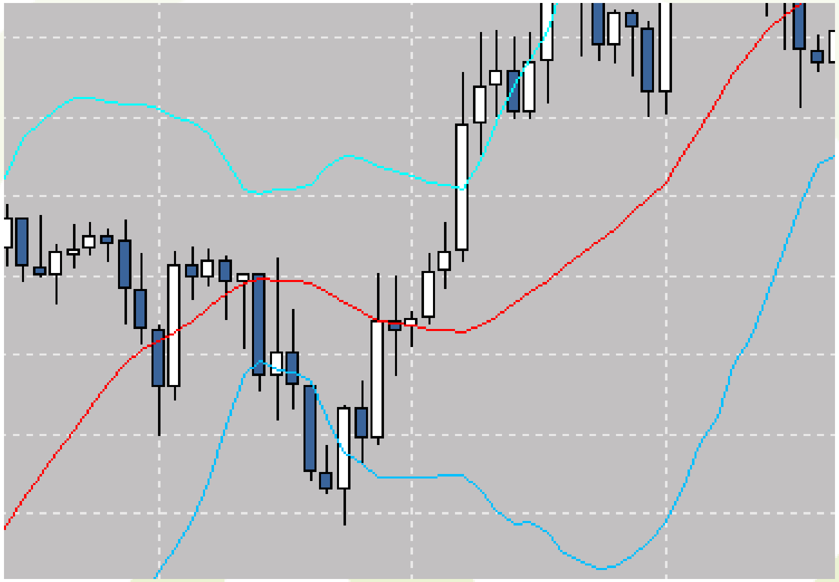
Detour

- Perform smoothing with simple moving average
  - E.g., SMA with window size of 21 bars
- **Trend:** *k* consecutive points on the SMA show constant increase or decrease on Y-axis

10.2 Trend Analysis

Detour

- **Detect turns** using for example Bollinger bands
  - – Calculated based on the moving average
    - N standard deviations up, N down
  - – Useful for detection of over-buy and over-sell



10.2 Trend Analysis

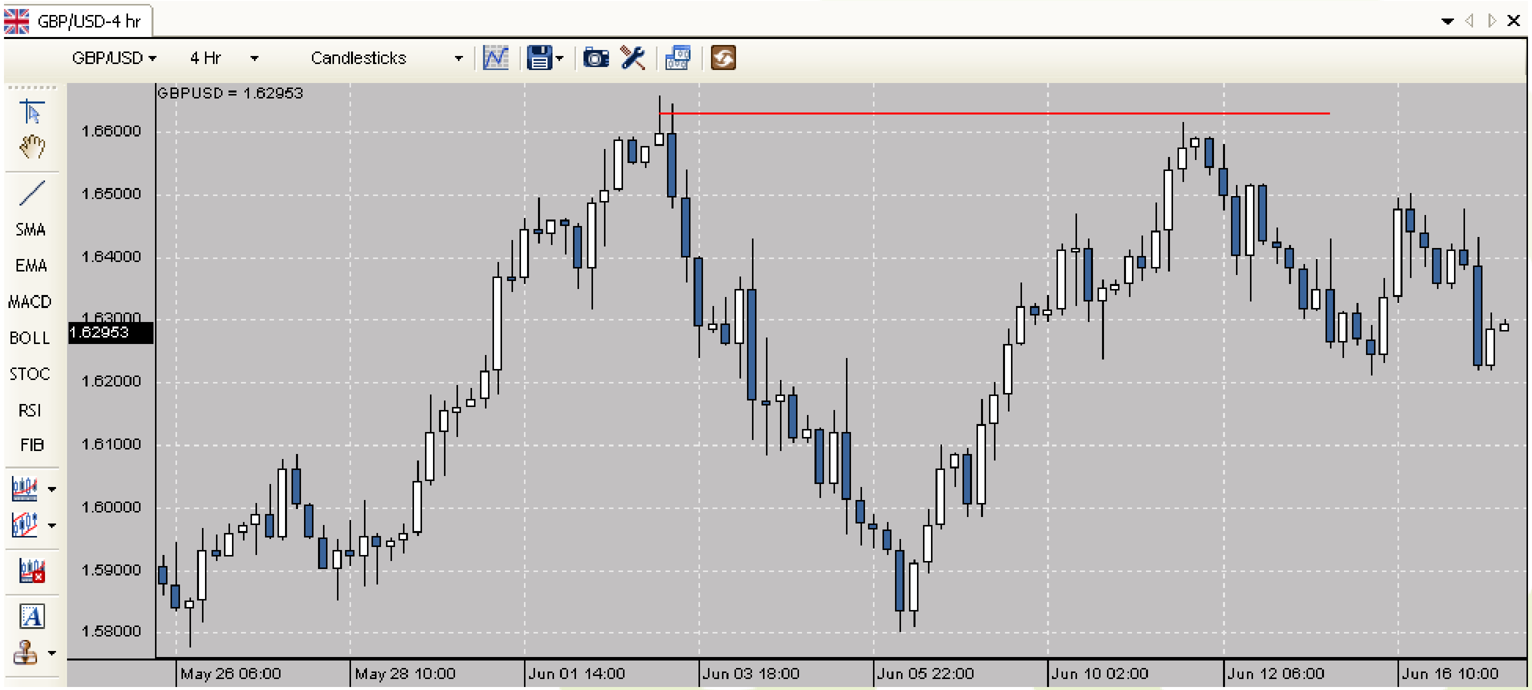
Detour

- Transactions…
  - between the lower band and the SMA show signs of **over-sell** and transactions
  - between SMA and upper band – **over-buy**
  - outside the Bollinger bands – **trend turn**

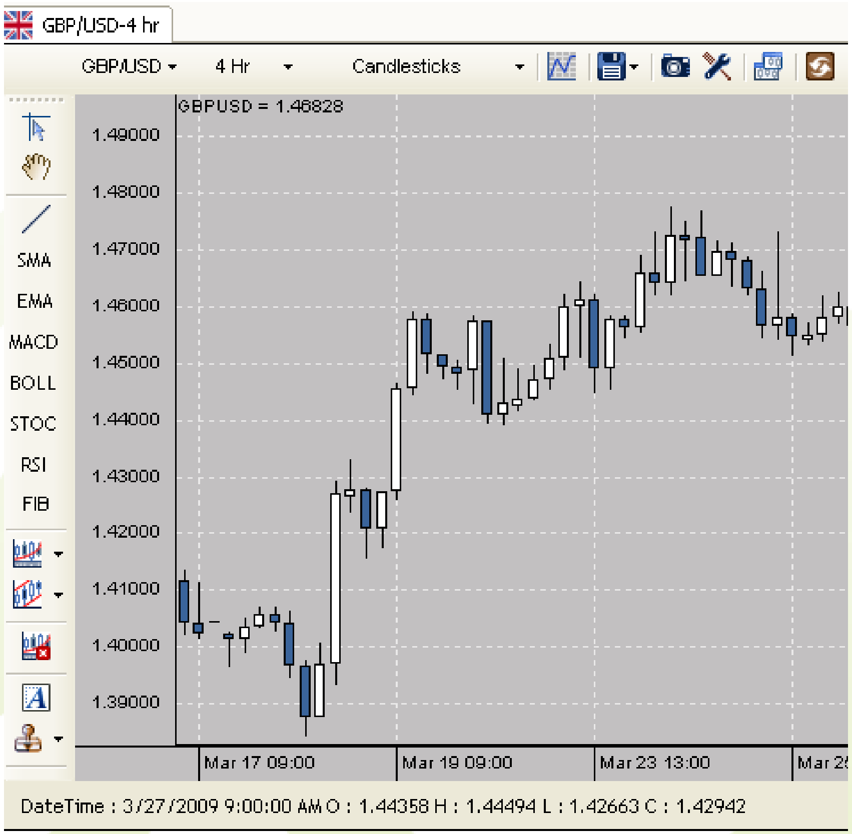
10.2 Trend Analysis

Detour

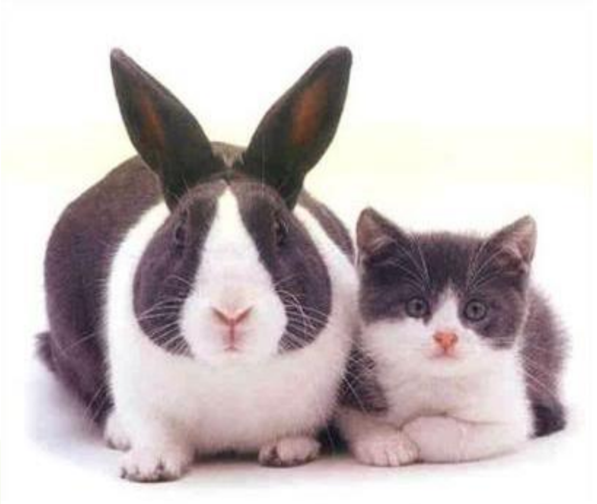
- Psychological pressure of the market
  - **Resistance lines** are determined by the reaction of the market participants to the previous evolution of the data



- And there are many more indicators for **in the trend** and **on trend turns**
  - E.g., **momentum** analysis
    - high momentum shows a powerful trend



10.2 Similarity Search



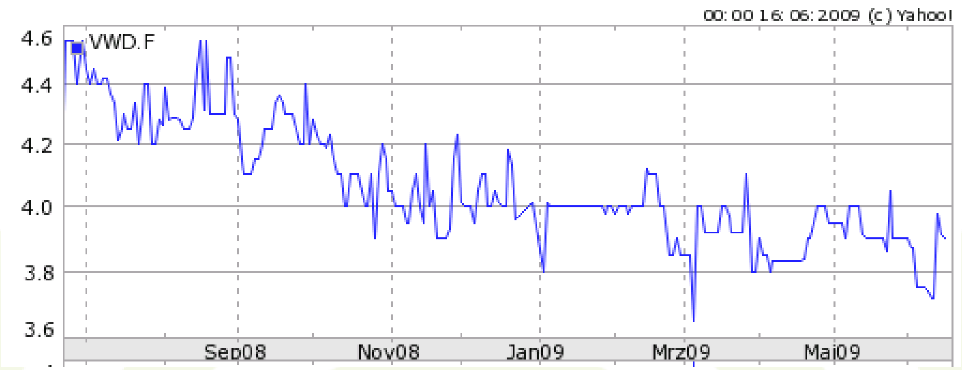
- **Similarity search**
  - Normal database queries find exact matches
  - Similarity search finds data sequences that **differ only slightly** from the given query sequence
- Problem: given a time-series database, identify all the sequences that are **similar** to one another

10.2 Similarity Search

- Typical **applications**
  - Financial market
    - Finding stock items with similar trends
  - Market basket
    - Finding products with similar sales trends
  - Scientific databases
    - Finding periods with similar temperature patterns, finding persons with similar voice clips

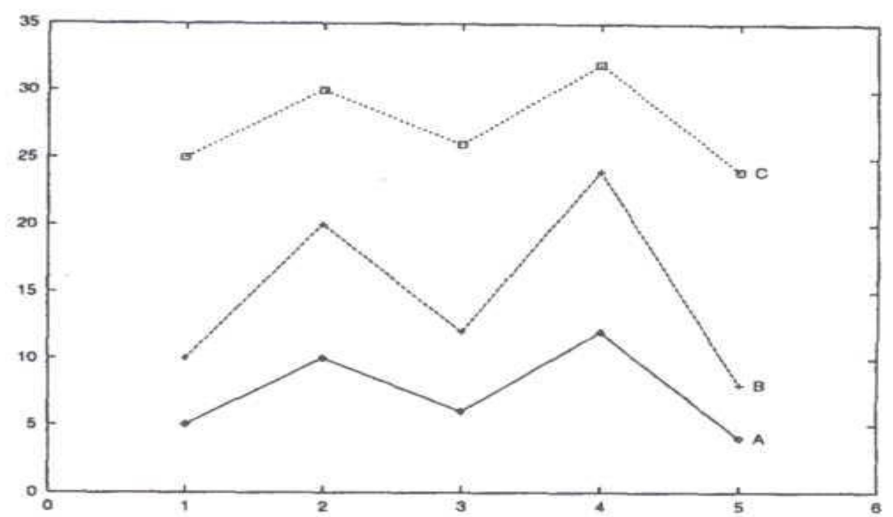
10.2 Similarity Search

- E.g., financial market applications
  - Evolution of VW has implications over all its suppliers
  - If we find similarities between the evolution of VW and Bosch, and if I know VW stock prices will drop due to car sales drops, then I should not buy any Bosch stocks!



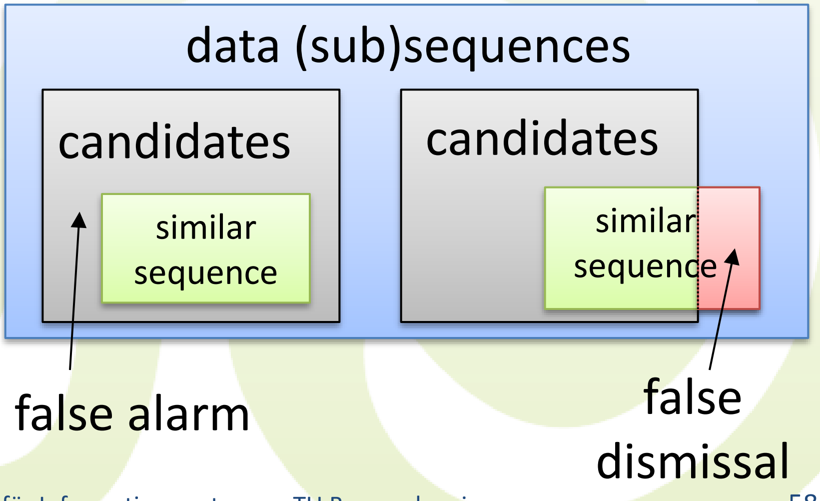
10.2 Similarity Search

- What is similarity?
  - Similarity is some degree of **symmetry** in either analogy and resemblance between two or more concepts or objects
- **Similarity measure**
  - A distance function d(X,Y) e.g., **Euclidean distance**



10.2 Similarity Search

- Issues encountered in similarity search
  - **False alarms**
    - (Sub)sequences returned as candidates, but **not similar** to the query sequence
  - **False dismissals**
    - (Sub)sequences that are similar to the query sequence, but **not returned** as the query result
  - Goal
    - Avoids false dismissals for **correctness**
    - Minimizes false alarms for **efficiency**



10.2 Similarity Search

- **Reduction**
  - Due to large size and high-dimensionality of time-series analysis, reduction is usually the first step
    - Reduction leads not only to smaller storage space but also to faster processing
  - E.g., **Discrete Fourier Transform** (DFT)
    - Concentrates energy in the **first few coefficients**
    - Keep the first few coefficients as representative of the sequence (feature extraction)
      - Based on them, we can compute the **lower bounds of the actual distance**

10.2 Similarity Search

- Two categories of similarity queries
  - **Whole matching**
    - Find a set of **sequences** that is similar to the query sequence
  - **Subsequence matching**
    - Find all sequences that **contain subsequences** that are similar to a given query sequence

10.2 Similarity Search

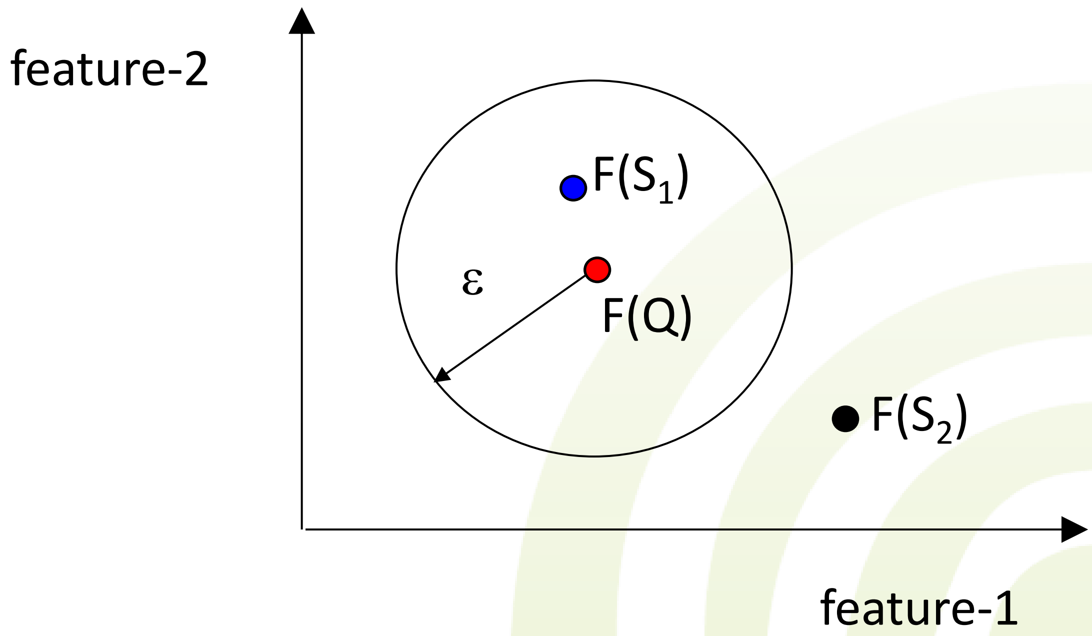
- **Whole matching**, basic idea
  - Uses the Euclidean distance as the similarity measure
  - Employs a multi-dimensional index for efficient search
    - Using the first few Fourier coefficients
    - R-trees, R*-trees can be used as multidimensional indexes
  - Uses a dimensionality-reduction technique for avoiding the curse of dimensionality
    - Data-independent: DFT, DCT, Wavelet transform
  - Guarantees no false dismissal thanks to Parseval's theorem
    - The **distance** between two signals in the **time domain** is the same as their distance in the **frequency domain**

10.2 Whole matching

- **Method**
  - Index building
    - Obtain the DFT coefficients of each sequence in the database
    - Build **a 2k-dimensional index** using the **first k Fourier coefficients** (2k-dimensions are needed because Fourier coefficients are complex numbers)
  - Query processing
    - Obtain the DFT coefficients of the query sequence
    - **Use the 2k-dimensional index to filter** out such sequences that are at most ε distance away from the query sequence
    - **Discards false alarms** by computing the **actual distance** between two sequences

10.2 Whole matching

- Sequences in **multidimensional space**

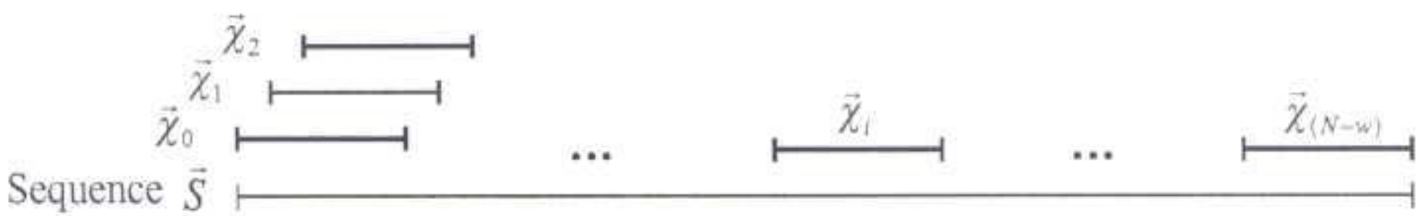


10.2 Similarity Search

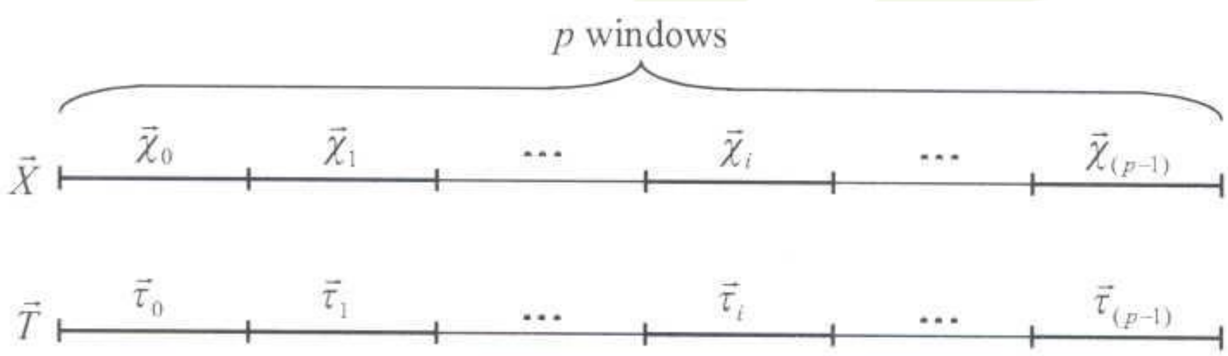
- **Subsequence matching**, basic idea
  - Use the concept of windows
    - Extract a set of **sliding windows** from each sequence
    - Map a window into a point in multi-dimensional space
    - Represent a sequence as a **trail**
    - Divide the trail of each sequence into **subtrails**
    - Represent each subtrail by its minimum bounding rectangle (**MBR**)

10.2 Subsequence matching

- Sliding window

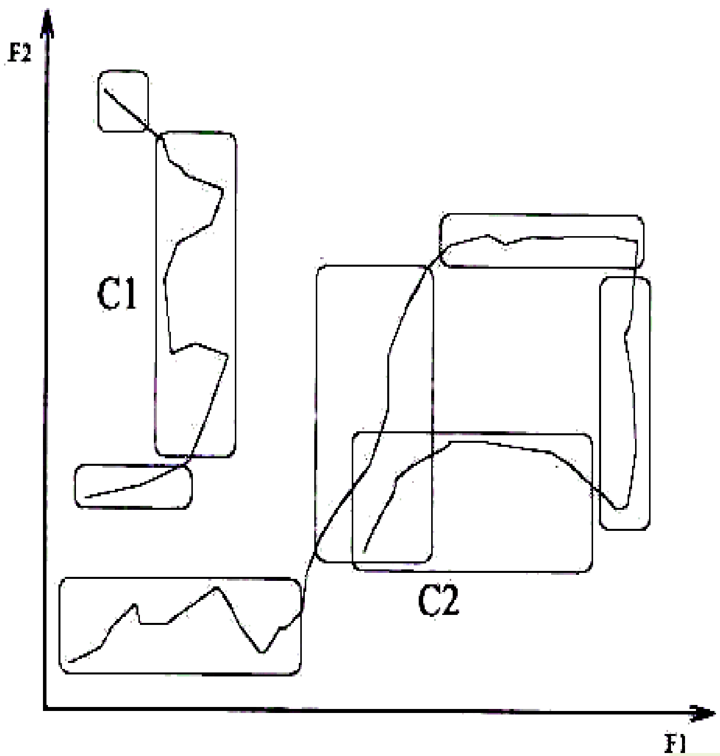


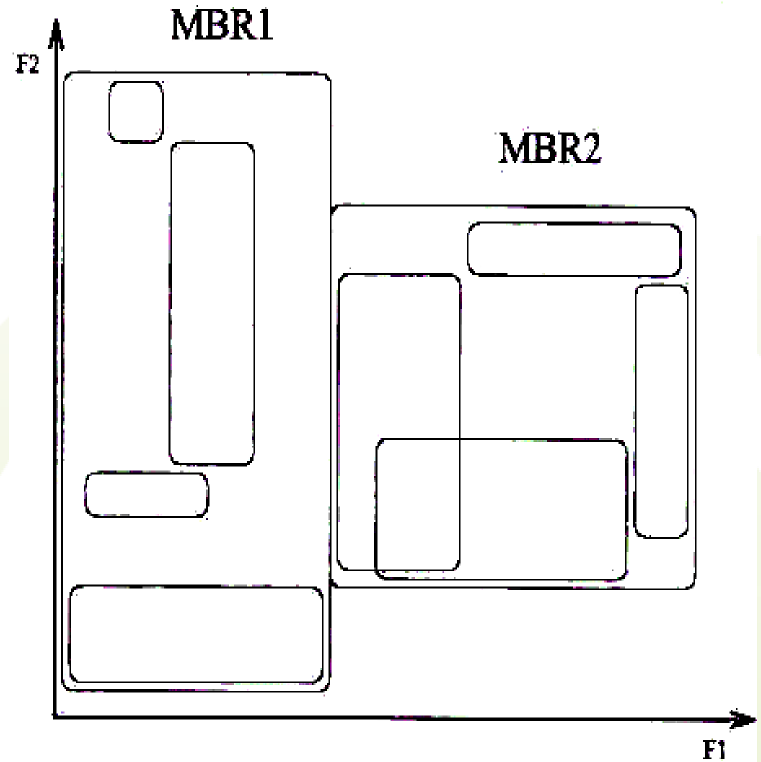
- Window matching



10.2 Subsequence matching

- Trails and their subtrails for sequences





10.2 Subsequence matching

- **Method**
  - Index building
    - Extract **sliding windows** from each sequence in the database
    - Obtain the DFT coefficients of each window
    - Divide the trail corresponding to a sequence into subtrails
    - Build a multi-dimensional index by using the MBRs that cover subtrails (R-Tree)
  - Query processing
    - Extract **p disjoint windows** from a query sequence
    - Obtain the DFT coefficients of each window
    - For each window, use the multi-dimensional index to filter out such sliding windows that are at most $\varepsilon / \sqrt{p}$ distance away from the window
    - **Discard false alarms** by computing the actual distance between the candidate subsequence and query sequence

10.2 Similarity Search

- But what if the two time-series being compared have **different baselines** or **scaling?**
  - E.g., one stock's value can have a baseline of 20€ and fluctuate with a relatively large amplitude (between 15 € and 25 €), while another stock with a baseline of 90 € can fluctuate wit a relatively small amplitude (between 90 € and 110 €)
- What if there are gaps?
- The solution is to apply **transformations**

10.2 Similarity Search

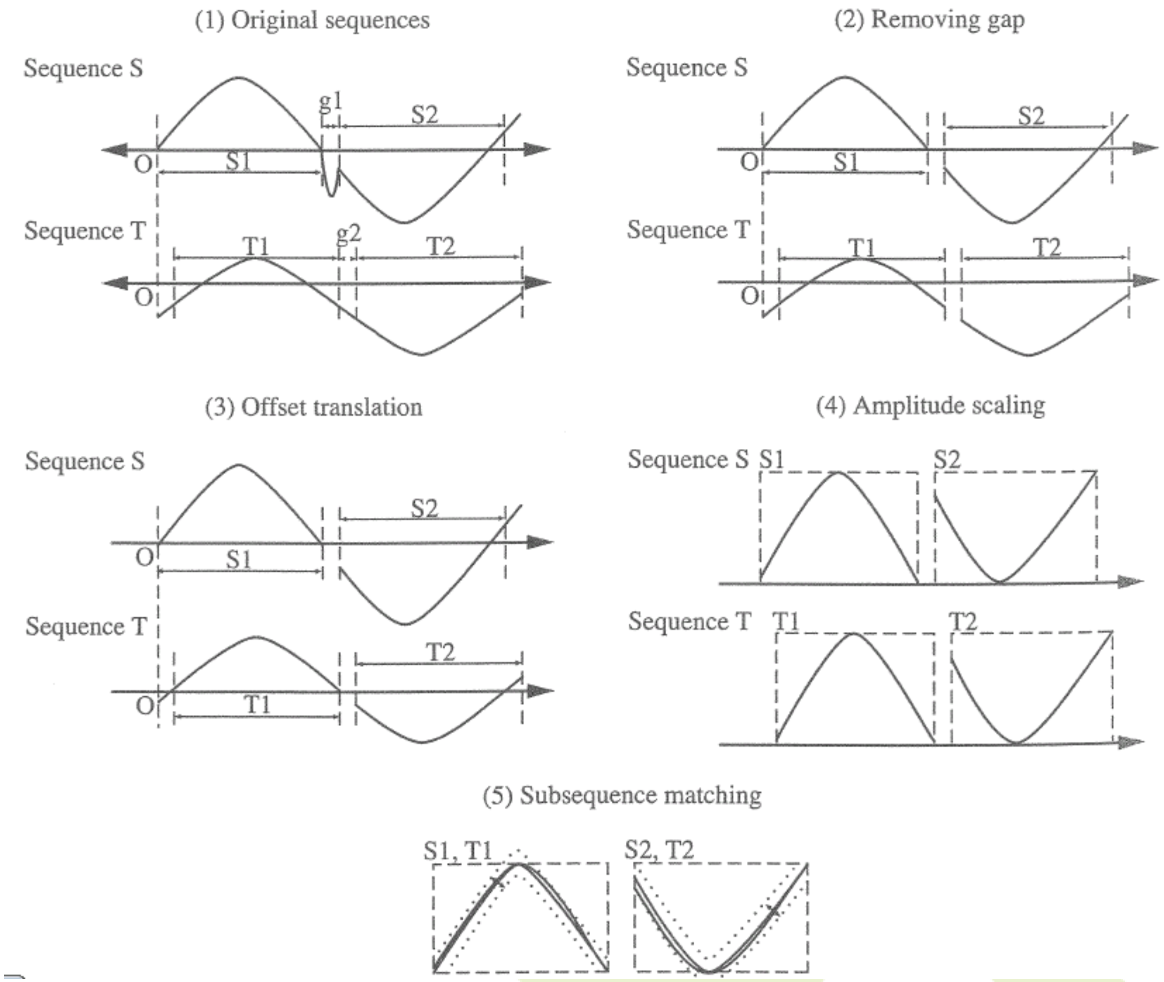
- **Transformation**
  - Provides various similarity models to satisfy specific application needs
  - Classified into:
    - Shifting
    - Scaling
    - Normalization
    - Moving average
    - (Dynamic) Time warping

10.2 Similarity Search

- **Enhanced** Similarity Search Methods
  - **Allow for gaps** within a sequence or differences in offsets or amplitudes
  - Normalize sequences with **amplitude scaling** and **offset translation**
  - Two **subsequences** are considered **similar,** if one lies within **one envelope of ε** width around the other, ignoring outliers
  - Two **sequences** are said to be **similar** if they have enough **non-overlapping**, time-ordered pairs of similar subsequences
  - **Parameters** specified by a user or expert
    - Sliding window size, width of an envelope for similarity, maximum gap, and matching fraction

10.2 Similarity Search

- Similarity model (subsequence)



10.2 Similarity Search

- **Enhanced subsequence matching**, method
  - Index building
    - Extract sliding windows of length w from each sequence in the database
    - Build a w-dimensional index on those windows
  - Query processing
    - Atomic matching
      - Find all pairs of gap-free windows that are similar
    - Window stitching
      - Stitch similar windows to form pairs of longer similar subsequences allowing gaps between window matches
    - Subsequence ordering
      - Linearly order the subsequence matches to determine whether enough similar pieces exist

10.2 Similarity Search

- **Enhanced whole matching**
  – Two sequences X and Y are considered similar, if $D(X, aY+b) \leq \varepsilon$ (after normalization), where a is the scaling constant and b is the shifting constant
- **Query languages?** Still a research question
  – Such a **time-series query language** should be able to:
    - Specify sophisticated queries like:
      – Find all of the sequences that are similar to some sequence in class A, but not similar to any sequence in class B
    - Support range queries, all-pair queries, and nearest neighbor queries

Summary

- Sequence Patterns
  - GSP, based on the Apriori property
- Time-Series
  - Trend Analysis:
    - Basic movements: Trend, Seasonal, Cycle, Irregular
    - Methods: Regression Analysis, Moving Averages, etc.
  - Similarity Search
    - Whole Matching
    - Subsequence Matching

Next lecture

- Data Mining
  - Classification
    - Decision Tree Induction
    - Bayesian Classification
    - Rule-Based Classification