Data Mining Algorithms

Lecture Course with Tutorials
Summer 2007

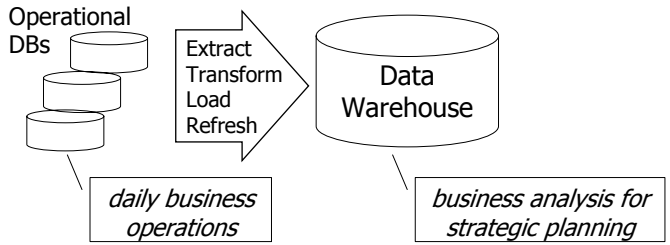
Chapter 2: Data Warehousing
and Data Preprocessing

Chapter 2: Data Warehousing and OLAP Technology for Data Mining

- What is a data warehouse?
- A multi-dimensional data model
- Data warehouse architecture

What is a Data Warehouse?

- Defined in many different ways, but not rigorously.
  - A decision support database that is maintained separately from the organization's operational database(s)



- Data warehousing:
  - The process of constructing and using data warehouses

What is a Data Warehouse?

- Support information processing by providing a solid platform of consolidated, historical data for analysis.
- "A data warehouse is a *subject-oriented*, *integrated*, *time-variant*, and *nonvolatile* collection of data in support of management's decision-making process."—W. H. Inmon

Data Warehouse: *Subject-Oriented*

- Organized around major subjects, such as customer, product, sales.
- Focusing on the modeling and analysis of data for decision makers, not on daily operations or transaction processing.
- Provide a simple and concise view around particular subject issues by excluding data that are not useful in the decision support process.

Data Warehouse: *Integrated*

- Constructed by integrating multiple, heterogeneous data sources
  - relational databases, flat files, on-line transaction records
- Data cleaning and data integration techniques are applied.
  - Ensure consistency in naming conventions, encoding structures, attribute measures, etc. among different data sources
    - E.g., Hotel price: currency, tax, breakfast covered, etc.
  - When data is moved to the warehouse, it is converted.

Data Warehouse: *Time Variant*

- The time horizon for the data warehouse is significantly longer than that of operational systems.
  - Operational database: current value data.
  - Data warehouse data: provide information from a historical perspective (e.g., past 5-10 years)
- Every key structure in the data warehouse
  - Contains an element of time, explicitly or implicitly
  - The key of the original operational data may or may not contain an explicit "time element".

Data Warehouse: *Non-Volatile*

- A physically separate store of data transformed from the operational environment.
- Operational update of data does not occur in the data warehouse environment.
  - Does not require transaction processing, recovery, and concurrency control mechanisms
  - Requires only two operations in data accessing:
    - *initial loading of data* and *access of data*.

Data Warehouse vs. Heterogeneous DBMS

- Traditional heterogeneous DB integration:
  - Build wrappers/mediators on top of heterogeneous databases
  - Query driven approach
    - Distribute a query to the individual heterogeneous sites; a meta-dictionary is used to translate the queries accordingly
    - Integrate the results into a global answer set
- Data warehouse: update-driven
  - Information from heterogeneous sources is integrated in advance and stored in warehouses for direct query and analysis

Data Warehouse vs. Operational DBMS

- **OLTP** (on-line transaction processing)
  - Major task of traditional relational DBMS
  - Day-to-day operations: purchasing, inventory, banking, manufacturing, payroll, registration, accounting, etc.
- **OLAP** (on-line analytical processing)
  - Major task of data warehouse system
  - Data analysis and decision making
- Distinct features (OLTP vs. OLAP):
  - *User and system orientation:* customer vs. market
  - *Data contents:* current & detailed vs. historical & consolidated
  - *Database design:* ER + application vs. star schema + subject
  - *View:* current & local vs. evolutionary & integrated
  - *Access patterns:* update vs. read-only but complex queries

OLTP vs. OLAP

	OLTP	**OLAP**
users	clerk, IT professional	knowledge worker
function	day to day operations	decision support
DB design	application-oriented	subject-oriented
data	current, up-to-date detailed, flat relational isolated	historical, summarized, multidimensional integrated, consolidated
usage	repetitive	ad-hoc
access	read/write index/hash on prim. key	lots of scans
unit of work	short, simple transaction	complex query
# records accessed	tens	millions
# users	thousands	hundreds
DB size	100MB-GB	100GB-TB
metric	transaction throughput	query throughput, response

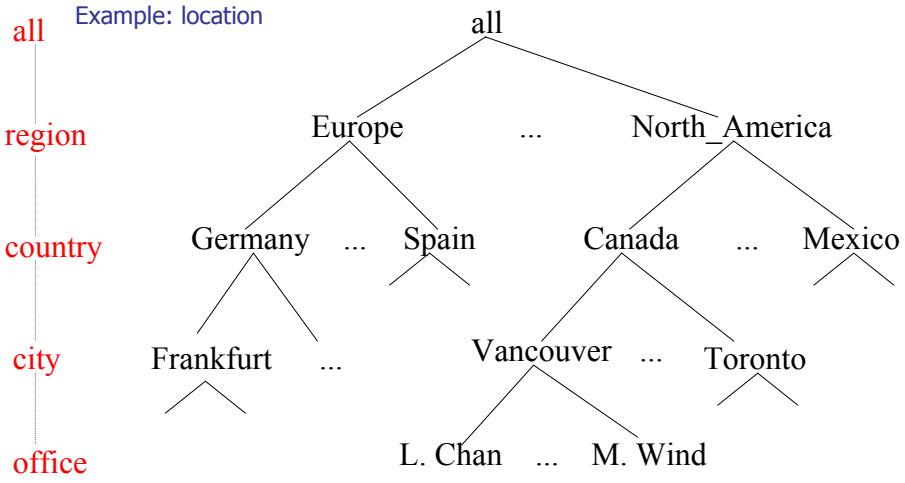
Chap. 2: Data Warehousing and OLAP Technology for Data Mining

- What is a data warehouse?
- A multi-dimensional data model
- Data warehouse architecture

From Tables and Spreadsheets to Data Cubes

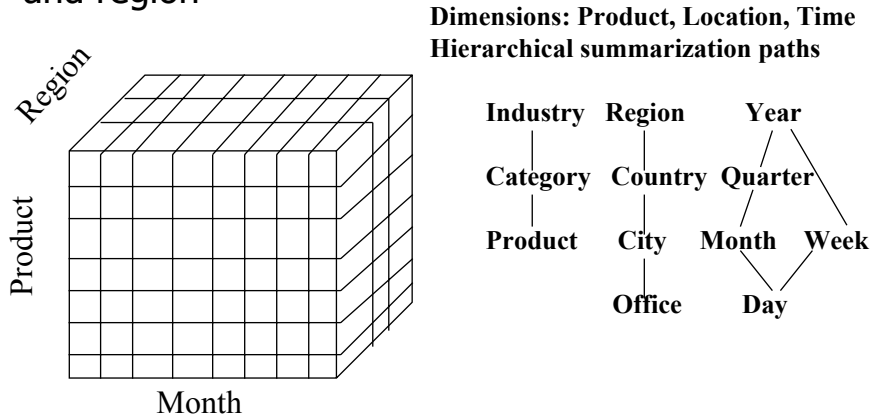
- A data warehouse is based on a multidimensional data model which views data in the form of a data cube
- A data cube, such as sales, allows data to be modeled and viewed in multiple dimensions
  - Dimension tables, such as item (item_name, brand, type), or time(day, week, month, quarter, year)
  - A Fact table that contains
    - measures (*dependent attributes*, e.g., dollars_sold) and
    - keys to each of the related dimension tables (dimensions, *independent attributes*)

Dimensions form Concept Hierarchies



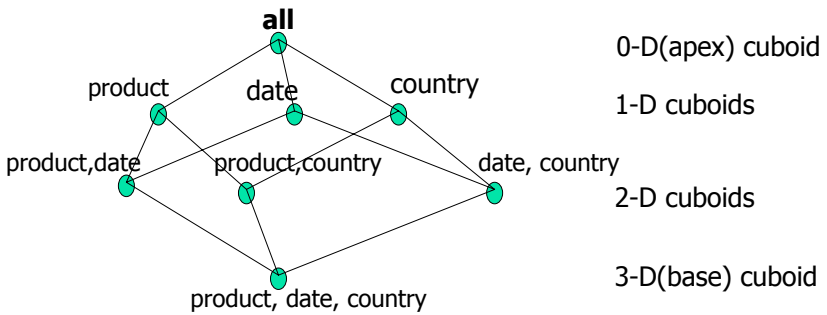
Multidimensional Data

- Sales volume as a function of product, month, and region

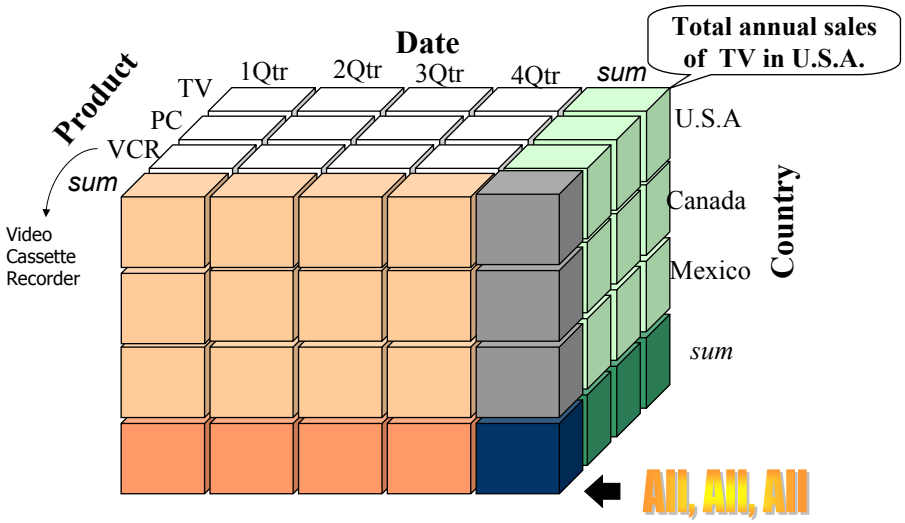


Cube as a Lattice of Cuboids

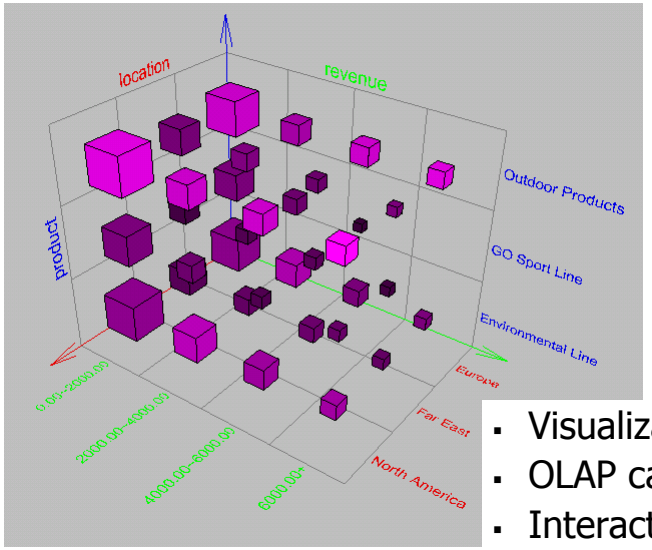
- In data warehousing literature, an *n*-D base cube is called a base cuboid. The top most 0-D cuboid, which holds the highest-level of summarization, is called the apex cuboid. The lattice of cuboids forms a data cube.



A Sample Data Cube



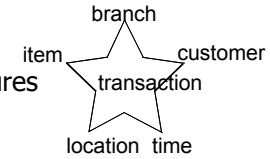
Browsing a Data Cube

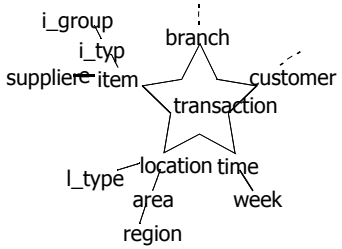


- Visualization
- OLAP capabilities
- Interactive manipulation

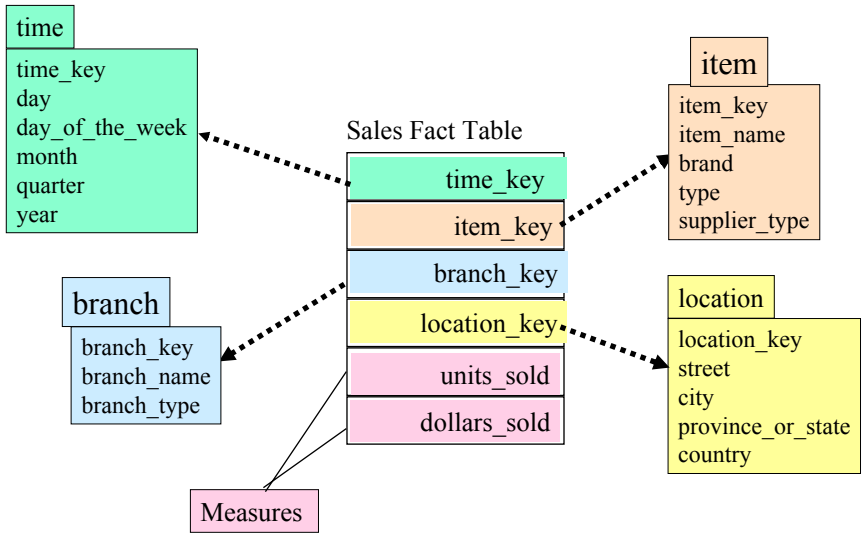
Conceptual Modeling of Data Warehouses

- Modeling data warehouses: dimensions & measures
  - Star schema: A fact table in the middle connected to a set of dimension tables
  - Snowflake schema: A refinement of star schema where some dimensional hierarchy is normalized into a set of smaller dimension tables, forming a shape similar to snowflake
  - Fact constellations: Multiple fact tables share dimension tables, i.e., a collection of stars, called galaxy schema or fact constellation

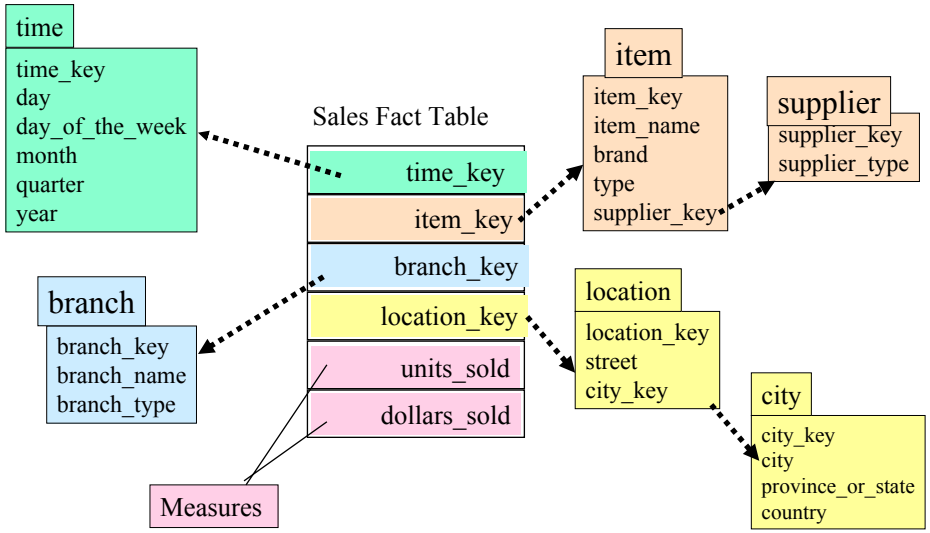




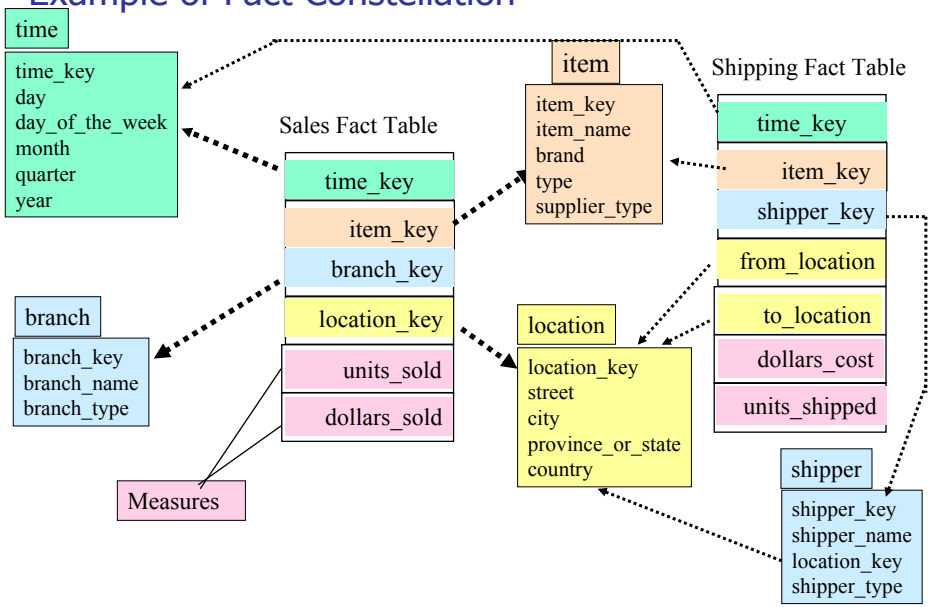
Example of Star Schema



Example of Snowflake Schema



Example of Fact Constellation



Measures: Three Categories

- *distributive:* if the result derived by applying the function to *n* aggregate values is the same as that derived by applying the function on all the data without partitioning.
  - E.g., count(), sum(), min(), max().
- *algebraic:* if it can be computed by an algebraic function with *M* arguments (where *M* is a bounded integer), each of which is obtained by applying a distributive aggregate function.
  - E.g., avg() = sum() / count(); standard_deviation().
- *holistic:* if there is no constant bound on the storage size which is needed to determine / describe a subaggregate.
  - E.g., median(), mode(), rank() [see next slide]

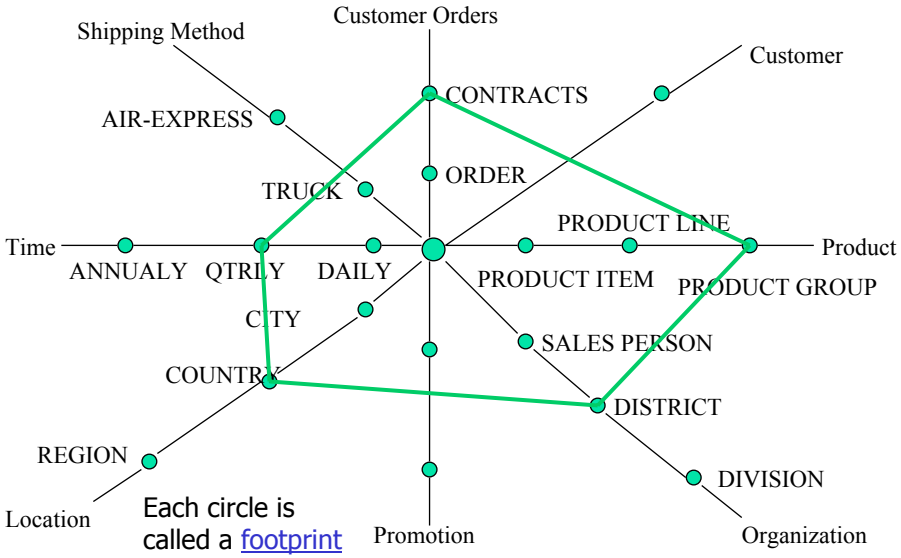
Measures: Examples

- *Distributive Measures*
  - count ($D_1 \cup D_2$) = count (D_1) + count (D_2)
  - sum ($D_1 \cup D_2$) = sum (D_1) + sum (D_2)
  - min ($D_1 \cup D_2$) = min (min (D_1), min (D_2))
  - max ($D_1 \cup D_2$) = max (max (D_1), max (D_2))
- *Algebraic Measures*
  - avg ($D_1 \cup D_2$) = sum ($D_1 \cup D_2$) / count ($D_1 \cup D_2$)
    = (sum (D_1) + sum (D_2)) / (count (D_1) + count (D_2))
- *Holistic Measures*
  - median: value in the middle of a sorted series of values (= 50% quantile)
  - mode: value that appears most often in a set of values
  - rank: k-smallest / k-largest value (cf. quantiles, percentiles)
  - median ($D_1 \cup D_2$) ≠ simple_function (median (D_1) + median (D_2))

Typical OLAP Operations

- Roll up (drill-up): summarize data
  - *by climbing up hierarchy or by dimension reduction*
- Drill down (roll down): reverse of roll-up
  - *from higher level summary to lower level summary or detailed data, or introducing new dimensions*
- Slice and dice:
  - *selection on one (slice) or more (dice) dimensions*
- Pivot (rotate):
  - *reorient the cube, visualization, 3D to series of 2D planes*
- Other operations
  - *drill across: involving (across) more than one fact table*
  - *drill through: through the bottom level of the cube to its back-end relational tables (using SQL)*

A Star-Net Query Model



Chap. 2: Data Warehousing and OLAP Technology for Data Mining

- What is a data warehouse?
- A multi-dimensional data model
- Data warehouse architecture

Why Separate Data Warehouse?

- High performance for both systems, OLTP and OLAP
- **DBMS** — tuned for OLTP
  - access methods
  - indexing
  - concurrency control
  - recovery
- **Warehouse** — tuned for OLAP
  - complex OLAP queries
  - multidimensional view
  - consolidation

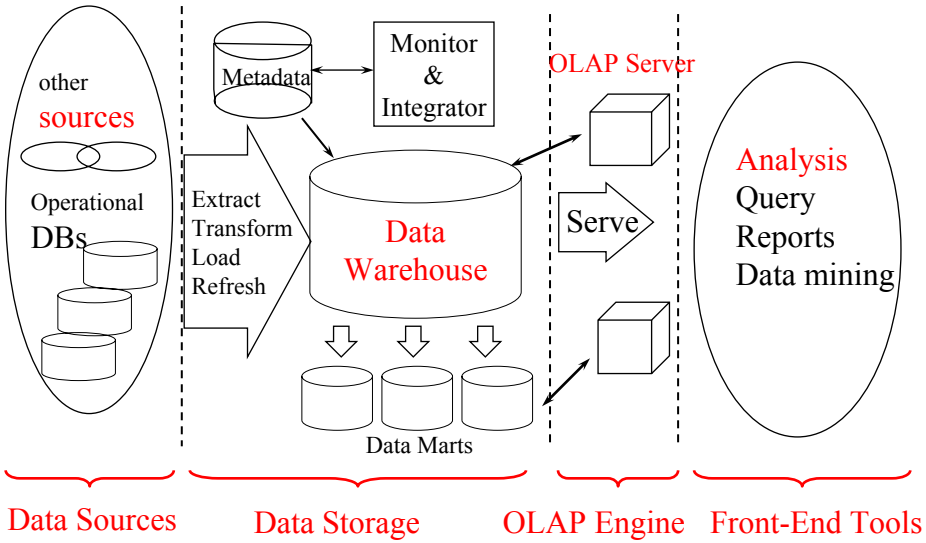
Design of a Data Warehouse: A Business Analysis Framework

- Four views regarding the design of a data warehouse
  - Top-down view
    - allows selection of the relevant information necessary for the data warehouse
  - Data source view
    - exposes the information being captured, stored, and managed by operational systems
  - Data warehouse view
    - consists of fact tables and dimension tables
  - Business query view
    - sees the perspectives of data in the warehouse from the view of end-user

Data Warehouse Design Process

- Top-down, bottom-up approaches or a combination of both
  - Top-down: Starts with overall design and planning (mature)
  - Bottom-up: Starts with experiments and prototypes (rapid)
- From software engineering point of view
  - Waterfall: structured and systematic analysis at each step before proceeding to the next
  - Spiral: rapid generation of increasingly functional systems, short turn around time, quick turn around
- Typical data warehouse design process
  - Choose a business process to model, e.g., orders, invoices, etc.
  - Choose the *grain* (*atomic level of data*) of the business process
  - Choose the dimensions that will apply to each fact table record
  - Choose the measure that will populate each fact table record

Multi-Tiered Architecture



Three Data Warehouse Models

- Enterprise warehouse
  - collects all of the information about subjects spanning the entire organization
- Data Mart
  - a subset of corporate-wide data that is of value to a specific groups of users. Its scope is confined to specific, selected groups, such as marketing data mart
    - Independent vs. dependent (directly from warehouse) data mart
- Virtual warehouse
  - A set of views over operational databases
  - Only some of the possible summary views may be materialized

OLAP Server Architectures

- Relational OLAP (ROLAP)
  - Use relational or extended-relational DBMS to store and manage warehouse data and OLAP middle ware to support missing pieces
  - Include optimization of DBMS backend, implementation of aggregation navigation logic, and additional tools and services
  - greater scalability (?)
- Multidimensional OLAP (MOLAP)
  - Array-based multidimensional storage engine (sparse matrix techniques)
  - fast indexing to pre-computed summarized data
- Hybrid OLAP (HOLAP)
  - User flexibility, e.g., low level: relational, high-level: array

Data Warehouse Back-End Tools and Utilities

- *Data extraction*
  - get data from multiple, heterogeneous, and external sources
- *Data cleaning*
  - detect errors in the data and rectify them when possible
- *Data transformation*
  - convert data from legacy or host format to warehouse format
- *Load*
  - sort, summarize, consolidate, compute views, check integrity, and build indices and partitions
- *Refresh*
  - propagate the updates from the data sources to the warehouse

Summary

- Data warehouse
  - A *subject-oriented*, *integrated*, *time-variant*, and *nonvolatile* collection of data in support of management's decision-making process
- A multi-dimensional model of a data warehouse
  - Star schema, snowflake schema, fact constellations
  - A data cube consists of dimensions & measures
- OLAP operations
  - drilling, rolling, slicing, dicing and pivoting
- OLAP servers
  - ROLAP, MOLAP, HOLAP

References (1)

- S. Agarwal, R. Agrawal, P. M. Deshpande, A. Gupta, J. F. Naughton, R. Ramakrishnan, and S. Sarawagi. On the computation of multidimensional aggregates. In Proc. 1996 Int. Conf. Very Large Data Bases, 506-521, Bombay, India, Sept. 1996.
- D. Agrawal, A. E. Abbadi, A. Singh, and T. Yurek. Efficient view maintenance in data warehouses. In Proc. 1997 ACM-SIGMOD Int. Conf. Management of Data, 417-427, Tucson, Arizona, May 1997.
- R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. In Proc. 1998 ACM-SIGMOD Int. Conf. Management of Data, 94-105, Seattle, Washington, June 1998.
- R. Agrawal, A. Gupta, and S. Sarawagi. Modeling multidimensional databases. In Proc. 1997 Int. Conf. Data Engineering, 232-243, Birmingham, England, April 1997.
- K. Beyer and R. Ramakrishnan. Bottom-Up Computation of Sparse and Iceberg CUBEs. In Proc. 1999 ACM-SIGMOD Int. Conf. Management of Data (SIGMOD'99), 359-370, Philadelphia, PA, June 1999.
- S. Chaudhuri and U. Dayal. An overview of data warehousing and OLAP technology. ACM SIGMOD Record, 26:65-74, 1997.
- OLAP council. MDAPI specification version 2.0. In http://www.olapcouncil.org/research/apily.htm, 1998.
- J. Gray, S. Chaudhuri, A. Bosworth, A. Layman, D. Reichart, M. Venkatrao, F. Pellow, and H. Pirahesh. Data cube: A relational aggregation operator generalizing group-by, cross-tab and sub-totals. Data Mining and Knowledge Discovery, 1:29-54, 1997.

References (2)

- V. Harinarayan, A. Rajaraman, and J. D. Ullman. Implementing data cubes efficiently. In Proc. 1996 ACM-SIGMOD Int. Conf. Management of Data, pages 205-216, Montreal, Canada, June 1996.
- Microsoft. OLEDB for OLAP programmer's reference version 1.0. In http://www.microsoft.com/data/oledb/olap, 1998.
- K. Ross and D. Srivastava. Fast computation of sparse datacubes. In Proc. 1997 Int. Conf. Very Large Data Bases, 116-125, Athens, Greece, Aug. 1997.
- K. A. Ross, D. Srivastava, and D. Chatziantoniou. Complex aggregation at multiple granularities. In Proc. Int. Conf. of Extending Database Technology (EDBT'98), 263-277, Valencia, Spain, March 1998.
- S. Sarawagi, R. Agrawal, and N. Megiddo. Discovery-driven exploration of OLAP data cubes. In Proc. Int. Conf. of Extending Database Technology (EDBT'98), pages 168-182, Valencia, Spain, March 1998.
- E. Thomsen. OLAP Solutions: Building Multidimensional Information Systems. John Wiley & Sons, 1997.
- Y. Zhao, P. M. Deshpande, and J. F. Naughton. An array-based algorithm for simultaneous multidimensional aggregates. In Proc. 1997 ACM-SIGMOD Int. Conf. Management of Data, 159-170, Tucson, Arizona, May 1997.

Data Preprocessing

- Why preprocess the data?
- Data cleaning
- Data integration and transformation
- Data reduction
- Discretization and concept hierarchy generation
- Summary

Why Data Preprocessing?

- Data in the real world is dirty
  - incomplete: lacking attribute values, lacking certain attributes of interest, or containing only aggregate data
  - noisy: containing errors or outliers
  - inconsistent: containing discrepancies in codes or names
- No quality data, no quality mining results!
  - Quality decisions must be based on quality data
  - Data warehouse needs consistent integration of quality data

Multi-Dimensional Measure of Data Quality

- A well-accepted multidimensional view:
  - Accuracy (range of tolerance)
  - Completeness (fraction of missing values)
  - Consistency (plausibility, presence of contradictions)
  - Timeliness (data is available *in time;* data is *up-to-date*)
  - Believability (user's trust in the data; reliability)
  - Value added (data brings some benefit)
  - Interpretability (there is some explanation for the data)
  - Accessibility (data is actually available)
- Broad categories:
  - intrinsic, contextual, representational, and accessibility.

Major Tasks in Data Preprocessing

- Data cleaning
  - Fill in missing values, smooth noisy data, identify or remove outliers, and resolve inconsistencies
- Data integration
  - Integration of multiple databases, data cubes, or files
- Data transformation
  - Normalization and aggregation
- Data reduction
  - Obtains reduced representation in volume but produces the same or similar analytical results
- Data discretization
  - Part of data reduction but with particular importance, especially for numerical data

Data Preprocessing

- Why preprocess the data?
- Data cleaning
- Data integration and transformation
- Data reduction
- Discretization and concept hierarchy generation
- Summary

Data Cleaning

- Data cleaning tasks
  - Fill in missing values
  - Identify outliers and smooth out noisy data
  - Correct inconsistent data

Missing Data

- Data is not always available
  - E.g., many tuples have no recorded value for several attributes, such as customer income in sales data
- Missing data may be due to
  - equipment malfunction
  - inconsistent with other recorded data and thus deleted
  - data not entered due to misunderstanding
  - certain data may not be considered important at the time of entry
  - not register history or changes of the data
- Missing data may need to be inferred.

How to Handle Missing Data?

- *Ignore the tuple:* usually done when class label is missing (not effective when the percentage of missing values per attribute varies considerably.
- Fill in the missing value *manually:* tedious (i.e., boring & time-consuming), infeasible?
- Use a *global constant* to fill in the missing value: e.g., a default value, or “unknown”, a new class?! – not recommended!
- Use the *attribute mean* (average value) to fill in the missing value
- Use the *attribute mean* for all samples belonging to the *same class* to fill in the missing value: smarter
- Use the *most probable value* to fill in the missing value: inference-based such as Bayesian formula or decision tree

Noisy Data

- Noise: random error or variance in a measured variable
- Incorrect attribute values may due to
  - faulty data collection instruments
  - data entry problems
  - data transmission problems
  - technology limitation
  - inconsistency in naming convention
- Other data problems which requires data cleaning
  - duplicate records
  - incomplete data
  - inconsistent data

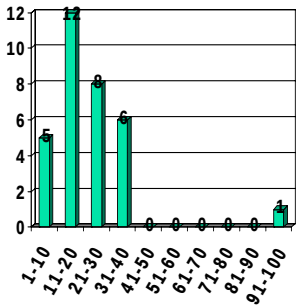
How to Handle Noisy Data?

- Binning method:
  - first sort data and partition into (equi-depth) bins
  - then one can smooth by bin means, smooth by bin median, smooth by bin boundaries, etc.
- Clustering
  - detect and remove outliers
- Combined computer and human inspection
  - detect suspicious values and check by human
- Regression
  - smooth by fitting the data into regression functions

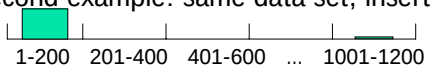
Noisy Data—Simple Discretization (1)

- Equi-width (distance) partitioning:
  - It divides the range into N intervals of equal size: uniform grid
  - if A and B are the lowest and highest values of the attribute, the width of intervals will be: W = (B-A)/N.
  - The most straightforward
  - Shortcoming: outliers may dominate presentation
  - Skewed data is not handled well.
- Example (data sorted, here: 10 bins):

  5, 7, 8, 8, 9, 11, 13, 13, 14, 14, 14, 15, 17, 17, 17, 18, 19, 23, 24, 25, 26, 26, 26, 27, 28, 32, 34, 36, 37, 38, 39, *97*



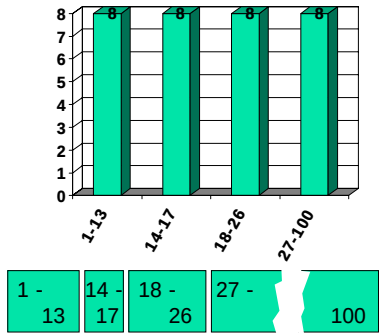
- Second example: same data set, insert 1023



Noisy Data—Simple Discretization (2)

- Equi-height (equi-depth, frequency) partitioning:
  - It divides the range into *N* intervals, each containing approximately same number of samples (*quantile-based approach*)
  - Good data scaling
  - Managing categorical attributes can be tricky.
- Same Example (here: 4 bins):

  5, 7, 8, 8, 9, 11, 13, 13, 14, 14, 14, 15, 17, 17, 17, 18, 19, 23, 24, 25, 26, 26, 26, 27, 28, 32, 34, 34, 36, 37, 37, 38, 39, *97*



- Median = 50%-quantile
  - is more robust against outliers (cf. value 1023 from above)

V-optimal Histograms (1)

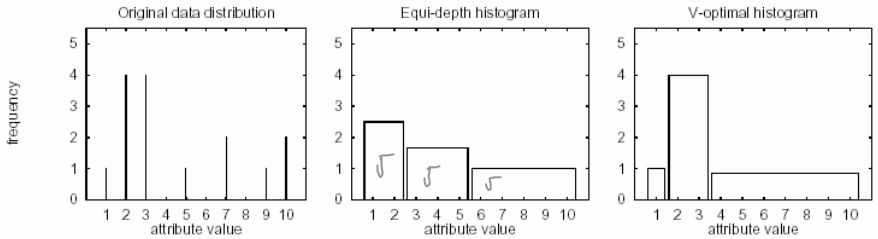
- *V-Optimal:* (variance optimal)
  - Given a fixed number N of buckets, the sum $\sum n_j V_j$ of weighted variances is minimized, where n_j is the number of elements in the j-th bucket and V_j is the variance of the source values (frequencies) in the j-th bucket.
  - Formally:
    - Minimize $\sum_{i=1}^{N} \sum_{j=lb_i}^{ub_i} (f(j) - avg_i)^2$

      where N number of buckets
      lb_i, ub_i lower and upper bounds of i-th bucket
      f(j) number of occurrence of the value j
      avg_i average of frequencies occurring in ith bucket

V. Poosala, Y. E. Ioannidis, P. J. Haas, E. J. Shekita: *Improved Histograms for Selectivity Estimation of Range Predicates*. Proc. ACM SIGMOD Conf. 1996: 294-305
H. V. Jagadish, N. Koudas, S. Muthukrishnan, V. Poosala, K.C. Sevcik, T. Suel, *Optimal histograms with quality guarantees*. Proc. VLDB Conf. 1998: 275-286

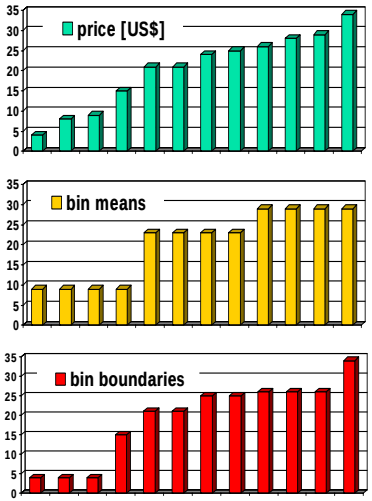
V-optimal Histograms (2)

- Example:
  - Equi-depth histogram: frequency
    - 1, 2 = 1 + 4 = 5
    - 3, 4, 5 = 4 + 0 + 1 = 5
    - 6, 7, 8, 9, 10 = 0 + 2 + 0 + 1 + 2 = 5
  - V-optimal histogram: variance
    - Bucket 1: $(1-1)^2 = 0$
    - Bucket 2: $(4-4)^2+(4-4)^2 = 0$
    - Bucket 3: $(0-6/7)^2+(1-6/7)^2+(0-6/7)^2+(2-6/7)^2+(0-6/7)^2+(1-6/7)^2+(2-6/7)^2 = 4{,}9$



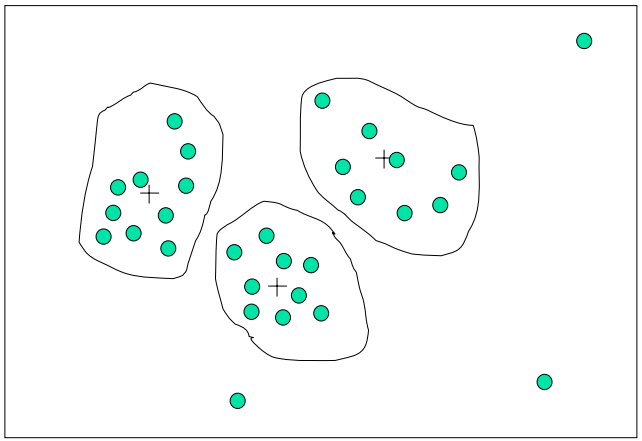
Noisy Data – Binning Methods for Data Smoothing

* Sorted data for price (in dollars):
  4, 8, 9, 15, 21, 21, 24, 25, 26, 28, 29, 34
* Partition into (equi-depth) bins:
  - Bin 1: 4, 8, 9, 15
  - Bin 2: 21, 21, 24, 25
  - Bin 3: 26, 28, 29, 34
* Smoothing by bin means:
  - Bin 1: 9, 9, 9, 9
  - Bin 2: 23, 23, 23, 23
  - Bin 3: 29, 29, 29, 29
* Smoothing by bin boundaries:
  - Bin 1: 4, 4, 4, 15
  - Bin 2: 21, 21, 25, 25
  - Bin 3: 26, 26, 26, 34



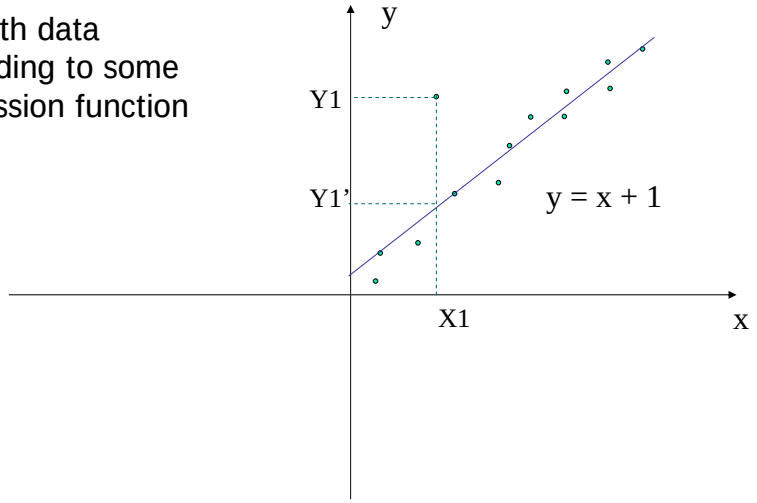
Noisy Data—Cluster Analysis

Detect and remove outliers



Noisy Data—Regression

Smooth data according to some regression function



Data Preprocessing

- Why preprocess the data?
- Data cleaning
- Data integration and transformation
- Data reduction
- Discretization and concept hierarchy generation
- Summary

Data Integration

- Data integration:
  - combines data from multiple, heterogeneous sources into a coherent store
- Schema integration
  - integrate metadata from different sources
  - Entity identification problem: identify real world entities from multiple data sources, e.g., A.cust-id ≡ B.cust-#
- Detecting and resolving data value conflicts
  - for the same real world entity, attribute values from different sources are different
  - possible reasons: different representations, different scales, e.g., metric vs. British units

Handling Redundant Data in Data Integration

- Redundant data occur often when integrating multiple databases
  - The same attribute may have different names in different databases
  - One attribute may be a “derived” attribute in another table, e.g., birthday vs. age; annual revenue
- Redundant data may be able to be detected by correlational analysis
- Careful integration of the data from multiple sources may help reduce/avoid redundancies and inconsistencies and improve mining speed and quality

Data Transformation

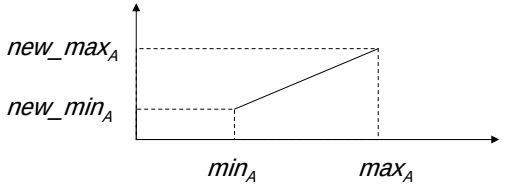
- Smoothing: remove noise from data
- Aggregation: summarization, data cube construction
- Generalization: concept hierarchy climbing
  - e.g., {young, middle-aged, senior} rather than {1...100}
- Normalization: scaled to fall within a small, specified range
  - min-max normalization
  - z-score normalization (= zero-mean normalization)
  - normalization by decimal scaling
- Attribute/feature construction
  - New attributes constructed from the given ones
    e.g., age = years(current_date – birthday)

Data Transformation: min-max Normalization

- min-max normalization

$$v' = (v - min_A)\frac{new\_max_A - new\_min_A}{max_A - min_A} + new\_min_A$$

- transforms data linearly to a new range
  - range outliers may be detected afterwards as well



slope is:

$$\frac{new\_max_A - new\_min_A}{max_A - min_A}$$

Data Transformation: zero-mean Normalization

- zero-mean (z-score) normalization

$$v' = \frac{v - mean_A}{std\_dev_A}$$

  - Leads to mean = 0, std_dev = 1

- Particularly useful if
  - min/max values are unknown
  - Outliers dominate min/max normalization

Data Transformation: Normalization by Decimal Scaling

- Normalization by decimal scaling

$$v' = \frac{v}{10^j}$$ where j is the smallest integer such that $max(|v'|) < 1$

- New data range: $0 <= |v'| < 1$ i.e., $-1 < v' < 1$
  Note that $0.1 \leq max(|v'|)$

- Normalization (in general) is important when considering several attributes in combination.
  - Large value ranges should not dominate small ones of other attributes
  - Example: age 0 ... 100 → 0 ... 1; income 0 ... 1.000.000 → 0 ... 1

Data Preprocessing

- Why preprocess the data?
- Data cleaning
- Data integration and transformation
- Data reduction
- Discretization and concept hierarchy generation
- Summary

Data Reduction Strategies

- Warehouse may store terabytes of data: Complex data analysis/mining may take a very long time to run on the complete data set
- Data reduction
  - Obtains a reduced representation of the data set that is much smaller in volume but yet produces the same (or almost the same) analytical results
- Data reduction strategies
  - Data cube aggregation
  - Dimensionality reduction
  - Numerosity reduction
  - Discretization and concept hierarchy generation

Data Cube Aggregation

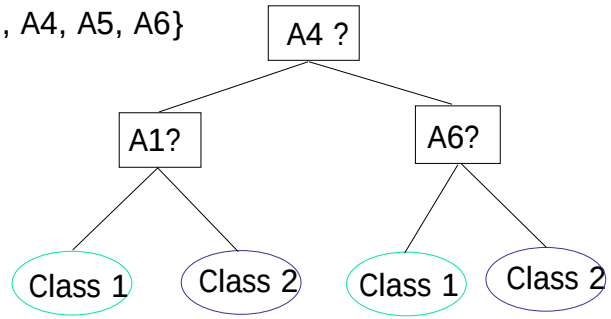
- The lowest level of a data cube
  - the aggregated data for an individual entity of interest
  - e.g., a customer in a phone calling data warehouse.
- Multiple levels of aggregation in data cubes
  - Further reduce the size of data to deal with
- Reference appropriate levels
  - Use the smallest representation which is enough to solve the task
- Queries regarding aggregated information should be answered using data cube, when possible

Dimensionality Reduction

- Feature selection (i.e., attribute subset selection):
  - Select a minimum set of features such that the probability distribution of different classes given the values for those features is as close as possible to the original distribution given the values of all features
  - reduce number of patterns in the patterns, easier to understand
- Heuristic methods (due to exponential # of choices):
  - step-wise forward selection
  - step-wise backward elimination
  - combining forward selection and backward elimination
  - decision-tree induction

Example of Decision Tree Induction

Initial attribute set:
{A1, A2, A3, A4, A5, A6}



-------> Reduced attribute set: {A1, A4, A6}

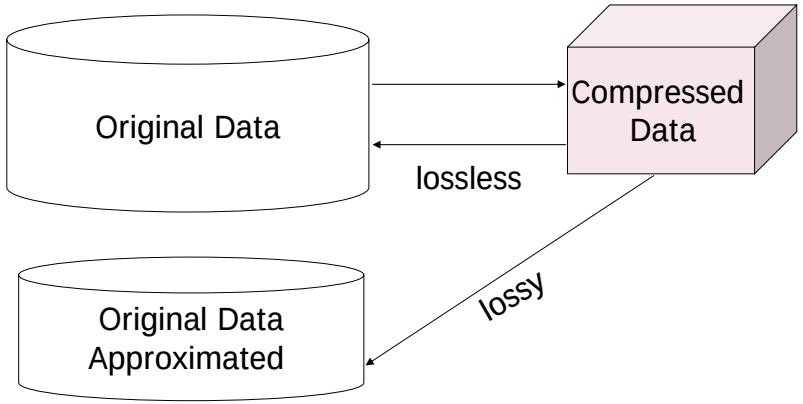
Heuristic Feature Selection Methods

- There are 2^d possible sub-features of d features
- Several heuristic feature selection methods:
  - Best single features under the feature independence assumption: choose by significance tests.
  - Best step-wise feature selection:
    - The best single-feature is picked first
    - Then next best feature condition to the first, ...
  - Step-wise feature elimination:
    - Repeatedly eliminate the worst feature
  - Best combined feature selection and elimination
  - Optimal branch and bound:
    - Use feature elimination and backtracking

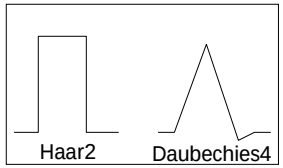
Data Compression

- String compression
  - There are extensive theories and well-tuned algorithms
  - Typically lossless
  - (Limited) manipulation is possible without expansion
- Audio/video compression
  - Typically lossy compression, with progressive refinement (e.g., based on Fourier transform)
  - Sometimes small fragments of signal can be reconstructed without reconstructing the whole
- Time sequence is not audio
  - Typically short and vary slowly with time

Data Compression



Wavelet Transforms



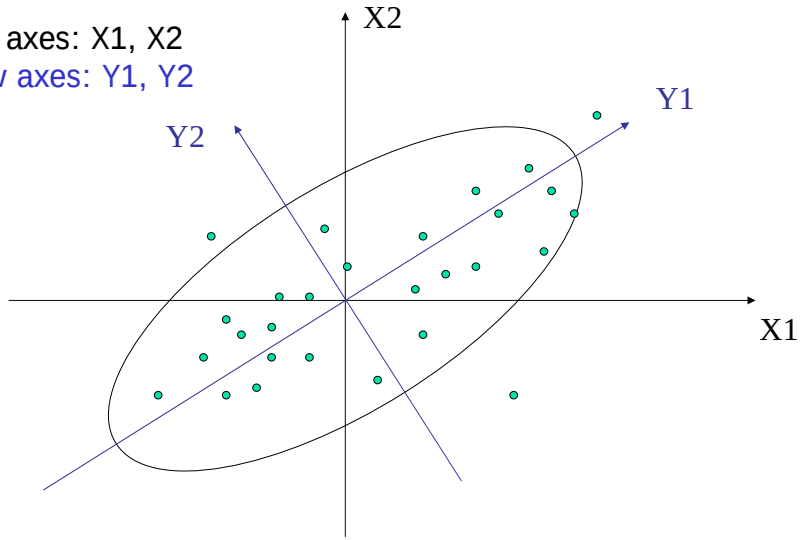
- Discrete wavelet transform (DWT): linear signal processing
- Compressed approximation: store only a small fraction of the strongest of the wavelet coefficients
- Similar to discrete Fourier transform (DFT), but better lossy compression, localized in space
- Method:
  - Length, L, must be an integer power of 2 (padding with 0s, when necessary)
  - Each transform has 2 functions: smoothing, difference
  - Applies to pairs of data, resulting in two set of data of length L/2
  - Applies two functions recursively, until reaches the desired length

Principal Component Analysis (PCA)

- Given *N* data vectors from *k*-dimensions, find $c <= k$ orthogonal vectors that are best used to represent data
  - The original data set is reduced to one consisting of *N* data vectors on *c* principal components (reduced dimensions)
- Each data vector is a linear combination of the *c* principal component vectors
- Works for numeric data only
- Used when the number of dimensions is large
- PCA is also known as Karhunen-Loéve Transform (KLT) or Singular Value Decomposition (SVD)

PCA—Example

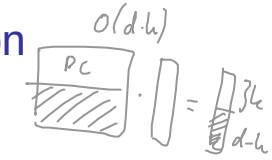
Old axes: X1, X2
new axes: Y1, Y2



PCA—Computation

- Normalization
  - Adapt value ranges of the dimensions
  - Move mean values (center of mass) to origin
- Compute principal components by a numerical method
  - Eigenvectors and Eigenvalues of covariance matrix
  - May use Singular Value Decomposition (SVD)
- Use the PC´s for a base transformation of the data
  - Basic linear algebra operation: multiply matrix to data
- PC´s are sorted in decreasing significance (variance)
  - Use first PC´s as a reduced coordinate system
  - Discard less significant PC directions

Reduction by Random Projection



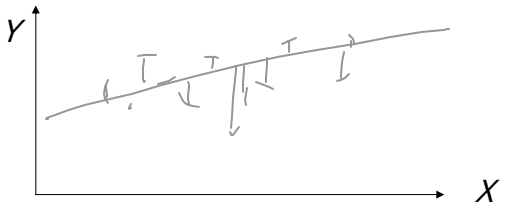
- Characteristics of PCA
  - Optimal reconstruction of data (wrt. linear error)
  - Computation of PCs is $O(n\,d^3)$ for n points, d dimensions
  - Data transformation is $O(ndk)$ for k reduced dimensions
- *Random Projection*
  - Randomly choose k vectors in d dimensions to form a new base
  - The new k-dimensional base is not necessarily orthogonal
- Characteristics of Random Projections
  - Fast precomputation $O(dk)$, Data transformation is $O(ndk)$
  - Reconstruction quality of data is reported to be very good

Numerosity Reduction

- Parametric methods
  - Assume the data fits some model, estimate model parameters, store only the parameters, and discard the data (except possible outliers)
  - Log-linear models: obtain value at a point in m-D space as the product on appropriate marginal subspaces
- Non-parametric methods
  - Do not assume models
  - Major families: histograms, clustering, sampling

Linear Regression

- Basic Idea: Data are modeled to fit a straight line

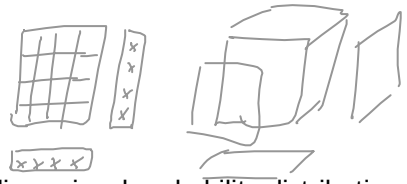


- Approach: $Y = \alpha + \beta X$
  - Two parameters , α and β specify the line and are to be estimated by using the data at hand.
  - Fit the line by applying the least squares method to the known values of *Y1, Y2, ..., X1, X2, ...*

Multiple Regression

- Basic idea: allows a response variable Y to be modeled as a linear function of multidimensional feature vector
- Example: fit $Y = b_0 + b_1 X_1 + b_2 X_2$ to data (X_1, X_2, Y')
  - Many nonlinear functions can be transformed into the above, e.g., $X_1 = f_1(v_1, v_2, v_3)$, $X_2 = f_2(v_1, v_2, v_3)$, i.e., model is fit to 4D data (v_1, v_2, v_3, z)
  - The parameters b_0, b_1, b_2 are determined by the least-squares method

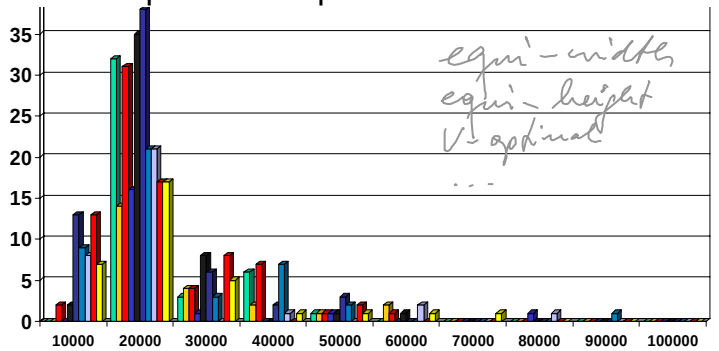
Log-linear Model



- Basic idea
  - Approximate discrete multi-dimensional probability distributions by using lower-dimensional data
- Approach
  - The multi-way table of joint probabilities is approximated by a product of lower-order tables.
  - Combine values of marginal distributions (higher degree of aggregation, “margin” of a cube, coarse cuboid) to approximate less aggregated values (“interior” cell in a data cube, fine-grained cuboid)

Histograms

- A popular data reduction technique
- Divide data into buckets and store average (sum) for each bucket
- Related to quantization problems.



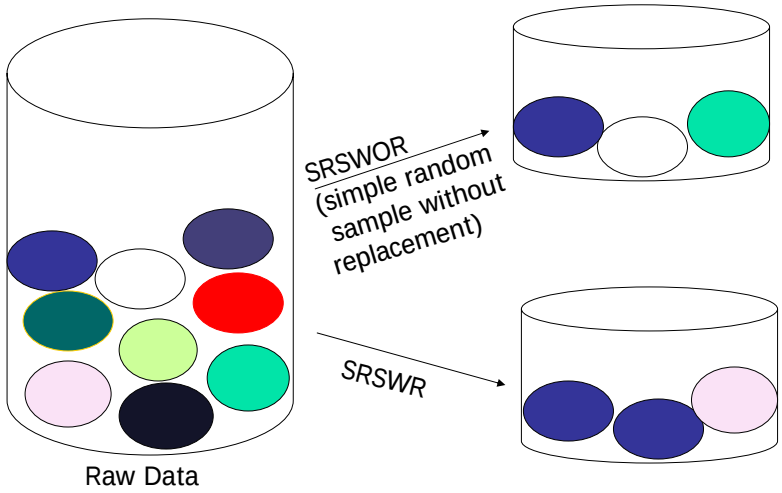
Clustering

- Partition data set into clusters, and one can store cluster representation only
- Can be very effective if data is clustered but not if data is “smeared”
- Can have hierarchical clustering and be stored in multi-dimensional index tree structures
- There are many choices of clustering definitions and clustering algorithms, see later

Sampling

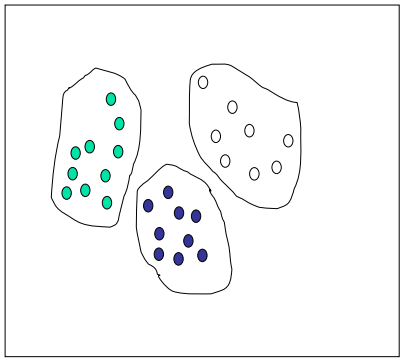
- Allow a mining algorithm to run in complexity that is potentially sub-linear to the size of the data
- Choose a representative subset of the data
  - Simple random sampling may have very poor performance in the presence of skew
- Develop adaptive sampling methods
  - Stratified sampling:
    - Approximate the percentage of each class (or subpopulation of interest) in the overall database
    - Used in conjunction with skewed data
- Sampling may not reduce database I/Os (page at a time).

Sampling

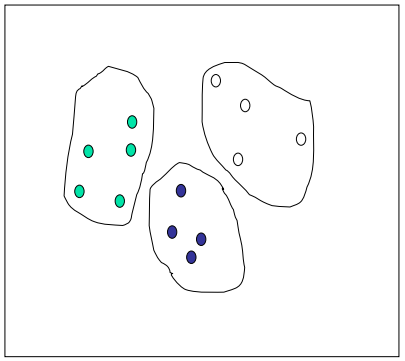


Sampling

Raw Data



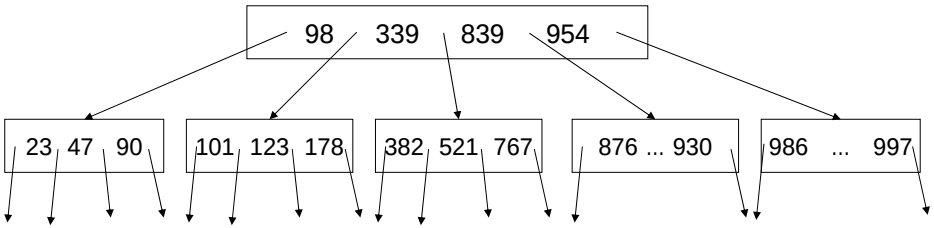
Cluster/Stratified Sample



Hierarchical Reduction

- Use multi-resolution structure with different degrees of reduction
- Hierarchical clustering is often performed but tends to define partitions of data sets rather than “clusters”
- Parametric methods are usually not compatible with hierarchical representation
- Hierarchical aggregation
  - An index tree hierarchically divides a data set into partitions by value range of some attributes
  - Each partition can be considered as a bucket
  - Thus an index tree with aggregates stored at each node is a hierarchical histogram

Example of an Index Tree



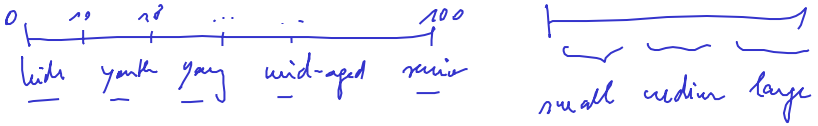
- Different level histograms: 5 bins, 20 bins, ...
- Approximation of equi-depth (“similar-depth”) histograms

Data Preprocessing

- Why preprocess the data?
- Data cleaning
- Data integration and transformation
- Data reduction
- Discretization and concept hierarchy generation
- Summary

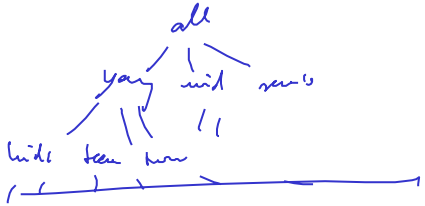
Discretization

- Three types of attributes:
  - Categorical (nominal) — values from an unordered set
  - Ordinal — values from an ordered set
  - Continuous — real numbers
- Discretization:
  - divide the range of a continuous attribute into intervals
  - Some classification algorithms only accept categorical attributes.
  - Reduce data size by discretization
  - Prepare for further analysis



Discretization and Concept Hierachy

- Discretization
  - reduce the number of values for a given continuous attribute by dividing the range of the attribute into intervals. Interval labels can then be used to replace actual data values.
- Concept hierarchies
  - reduce the data by collecting and replacing low level concepts (such as numeric values for the attribute age) by higher level concepts (such as young, middle-aged, or senior).



Discretization and concept hierarchy generation for *numeric* data

- Binning (see above)
- Histogram analysis (see above)
- Clustering analysis (see later)
- Entropy-based discretization
- Segmentation by natural partitioning

Entropy-Based Discretization

- Given a set of samples S, if S is partitioned into two intervals S_1 and S_2 using boundary T, the entropy after partitioning is

$$E(S,T) = \frac{|S_1|}{|S|} Ent(S_1) + \frac{|S_2|}{|S|} Ent(S_2) \qquad Ent(S) = -\sum_{i=1}^{m} p_i \log_2(p_i)$$

- The boundary that minimizes the entropy function over all possible boundaries is selected as a binary discretization.
- Thus, the *Information Gain* *I(S,T)* is maximized:

$$I(S,T) = Ent(S) - E(S,T)$$

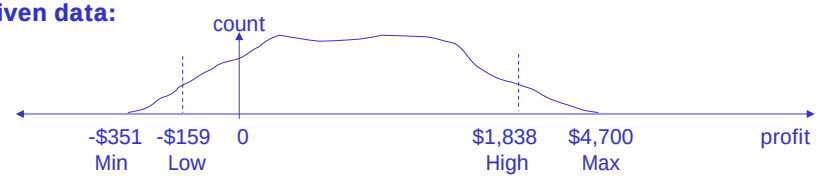
- The process is recursively applied to partitions obtained until some stopping criterion is met, e.g., *I(S,T)* > δ
- Experiments show that it may reduce data size and improve classification accuracy

Segmentation by natural partitioning

- 3-4-5 rule can be used to segment numeric data into relatively uniform, "natural" intervals.
  - If an interval covers 3, 6, 7 or 9 distinct values at the most significant digit, partition the range into 3 equi-width intervals
  - If it covers 2, 4, or 8 distinct values at the most significant digit, partition the range into 4 intervals
  - If it covers 1, 5, or 10 distinct values at the most significant digit, partition the range into 5 intervals

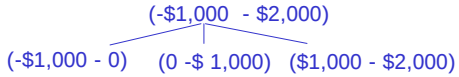
Example of 3-4-5 rule

Given data:



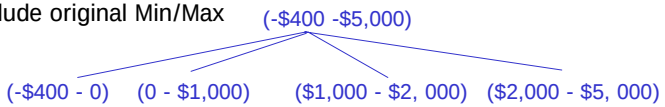
First step:

- **90%-extraction:** Low (5%-tile) = -$159, High (95%-tile) = $1,838
- most significant digit = 1,000 → set Low = –$1,000 and High = $2,000
- → 3 classes

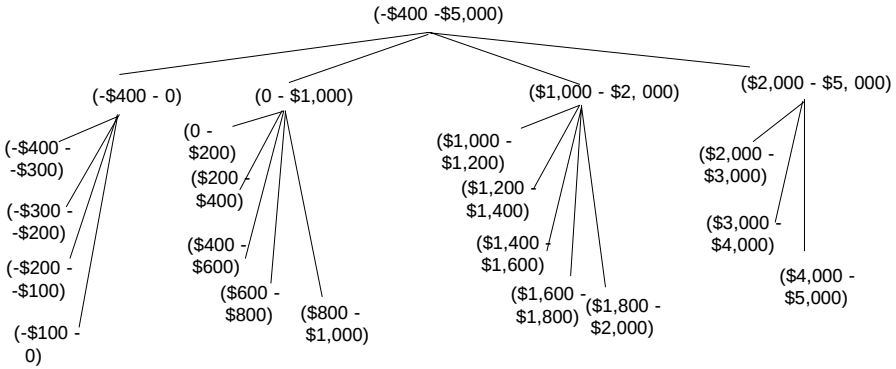


Refinement:

include original Min/Max



Example of 3-4-5 rule—Result

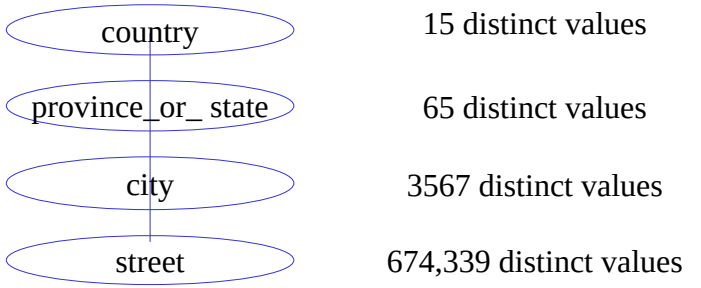


Concept hierarchy generation for *categorical* data

- Alternative approaches to specify concept hierarchies by users or experts:
  - Specify a partial ordering of attributes explicitly at the schema level
    e.g., *street < city < province_or_state < country*
  - Specify a portion of a hierarchy by explicit data grouping
    e.g., {Sweden, Norway, Finland, Danmark} ⊂ Scandinavia,
    {Scandinavia, Baltics, Central Europe, ...} ⊂ Europe
  - Specify a set of attributes, but not their partial ordering
    e.g., {*street, city, province_or_state, country*}
  - Specify only a partial set of attributes
    e.g., {*city*}

Specification of a set of attributes

- Concept hierarchy
  - can be automatically generated based on the number of distinct values per attribute in the given attribute set
  - the attribute with the most distinct values is placed at the lowest level of the hierarchy



  - Counter example: 20 distinct years, 12 months, 7 days_of_the_week but not „year < month < days_of_the_week“ with the latter on top

Data Preprocessing

- Why preprocess the data?
- Data cleaning
- Data integration and transformation
- Data reduction
- Discretization and concept hierarchy generation
- Summary

Summary

- Data preparation is a big issue for both warehousing and mining
- Data preparation includes
  - Data cleaning and data integration
  - Data reduction and feature selection
  - Discretization
- A lot a methods have been developed but still an active area of research

References

- D. P. Ballou and G. K. Tayi. Enhancing data quality in data warehouse environments. Communications of ACM, 42:73-78, 1999.
- Jagadish et al., Special Issue on Data Reduction Techniques. Bulletin of the Technical Committee on Data Engineering, 20(4), December 1997.
- D. Pyle. Data Preparation for Data Mining. Morgan Kaufmann, 1999.
- T. Redman. Data Quality: Management and Technology. Bantam Books, New York, 1992.
- Y. Wand and R. Wang. Anchoring data quality dimensions ontological foundations. Communications of ACM, 39:86-95, 1996.
- R. Wang, V. Storey, and C. Firth. A framework for analysis of data quality research. IEEE Trans. Knowledge and Data Engineering, 7:623-640, 1995.

© for the original version:
- Jörg Sander and Martin Ester
- Jiawei Han and Micheline Kamber

Data Mining Algorithms

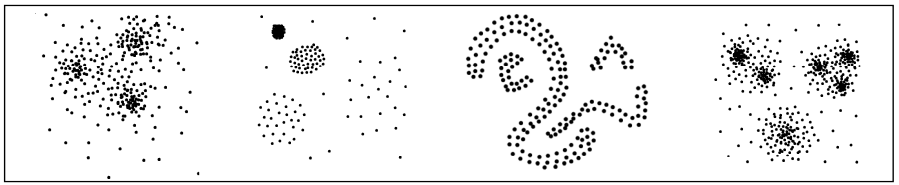
Lecture Course with Tutorials
Summer 2005

Chapter 3: Clustering

Chapter 3: Clustering

What is Clustering?

- Grouping a set of data objects into clusters
  - Cluster: a collection of data objects
    - Similar to one another within the same cluster
    - Dissimilar to the objects in other clusters
- Clustering = ***unsupervised classification*** (no predefined classes)
- Typical usage
  - As a *stand-alone tool* to get insight into data distribution
  - As a *preprocessing step* for other algorithms



Measuring Similarity

- To measure similarity, often a distance function *dist* is used
  - Measures "dissimilarity" between pairs objects *x* and *y*
    - Small distance *dist*(*x*, *y*): objects x and y are more similar
    - Large distance *dist*(*x*, *y*): objects x and y are less similar
- Properties of a distance function
  - $dist(x, y) \geq 0$ (positive semidefinite)
  - $dist(x, y) = 0$ iff $x = y$ (definite) (iff = if and only if)
  - $dist(x, y) = dist(y, x)$ (symmetry)
  - If *dist* is a metric, which is often the case:
    $dist(x, z) \leq dist(x, y) + dist(y, z)$ (triangle inequality)
- Definition of a distance function is highly application dependent
  - May require standardization/normalization of attributes
  - Different definitions for interval-scaled, boolean, categorical, ordinal and ratio variables

Example distance functions (1)

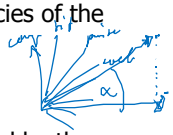
- For standardized numerical attributes, i.e., vectors $x = (x_1, ..., x_d)$ and $y = (y_1, ..., y_d)$ from a d-dimensional vector space:
  - General L_p-Metric (Minkowski-Distance) $d_p(x,y) = \sqrt[p]{\sum_{i=1}^{d} |x_i - y_i|^p}$
  - $p = 2$: Euclidean Distance (cf. Pythagoras) $d_2(x,y) = \sqrt{\sum_{i=1}^{d} (x_i - y_i)^2}$
  - $p = 1$: Manhattan-Distance (city block) $d_1(x,y) = \sum_{i=1}^{d} |x_i - y_i|$
  - $p \to \infty$: Maximum-Metric $d_\infty(x,y) = \max\{|x_i - y_i|, 1 \le i \le d\}$
- For sets x and y: $d_{set}(x,y) = \frac{|x \cup y| - |x \cap y|}{|x \cup y|} = \frac{|x \setminus y \cup y \setminus x|}{|x \cup y|}$

Example distance functions (2)

- For categorical attributes: Hamming distance

$$dist(x,y) = \sum_{i=1}^{d} \delta(x_i, y_i) \text{ where } \delta(x_i, y_i) = \begin{cases} 0 & \text{if } x_i = y_i \\ 1 & \text{else} \end{cases}$$

- For text documents:
  - A document D is represented by a vector $r(D)$ of frequencies of the terms occuring in D, e.g., $r(d) = \{\log(f(t_i, D)), t_i \in T\}$ where $f(t_i, D)$ is the frequency of term t_i in document D



  - The distance between two documents D_1 and D_2 is defined by the cosine of the angle between the two vectors $x = r(D_1)$ and $y = r(D_2)$: $dist(x,y) = 1 - \frac{\langle x, y \rangle}{|x| \cdot |y|}$ where $\langle . , . \rangle$ denotes the inner product and $|.|$ is the length of vectors
  - The cosine distance is semi-definite (e.g., permutations of terms)

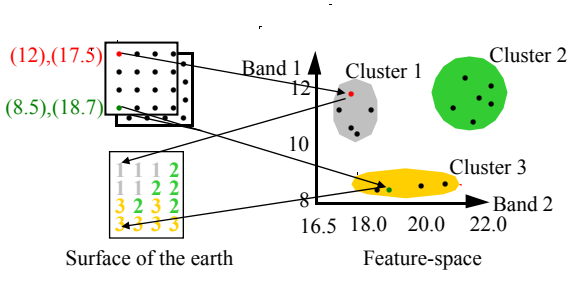
$$\langle x, y \rangle = |x| \cdot |y| \cdot \cos \angle(x,y)$$

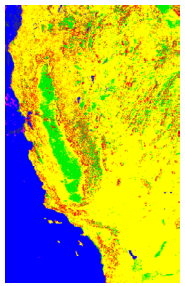
General Applications of Clustering

- Pattern Recognition and Image Processing
- Spatial Data Analysis
  - create thematic maps in GIS (Geographic Information Systems) by clustering feature spaces
  - detect spatial clusters and explain them in spatial data mining
- Economic Science (especially market research)
- WWW
  - Documents (Web Content Mining)
  - Web-logs (Web Usage Mining)
- Biology
  - Clustering of gene expression data

A Typical Application: Thematic Maps

- Satellite images of a region in different wavelengths
  - Each point on the surface maps to a high-dimensional feature vector $p = (x_1, ..., x_d)$ where x_i is the recorded intensity at the surface point in band i.
  - Assumption: each different land-use reflects and emits light of different wavelengths in a characteristic way.





Application: Web Usage Mining

Determine Web User Groups

Sample content of a web log file

```
romblon.informatik.uni-muenchen.de lopa - [04/Mar/1997:01:44:50 +0100] "GET /~lopa/ HTTP/1.0" 200 1364
romblon.informatik.uni-muenchen.de lopa - [04/Mar/1997:01:45:11 +0100] "GET /~lopa/x/ HTTP/1.0" 200 712
fixer.sega.co.jp unknown - [04/Mar/1997:01:58:49 +0100] "GET /dbs/porada.html HTTP/1.0" 200 1229
scooter.pa-x.dec.com unknown - [04/Mar/1997:02:08:23 +0100] "GET /dbs/kriegel_e.html HTTP/1.0" 200 1241
```

Generation of sessions

⇨ Session::= <IP_address, user_id, [URL_1, . . ., URL_k]>

which entries form a single session?

Distance function for sessions: $d(x,y) = \frac{|x \cup y| - |x \cap y|}{|x \cup y|} = \frac{|(x-y) \cup (y-x)|}{|x \cup y|}$

Major Clustering Approaches

- Expectation Maximization
- Partitioning algorithms
  - Find *k* partitions, minimizing some objective function
- Hierarchy algorithms
  - Create a hierarchical decomposition of the set of objects
- Density-based
  - Find clusters based on connectivity and density functions
- Subspace Clustering
- Other methods
  - Grid-based
  - Neural networks (SOM's)
  - Graph-theoretical methods
  - . . .

Chapter 3: Clustering

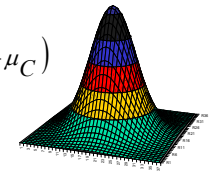
- Introduction to clustering
- Expectation Maximization: a statistical approach
- Partitioning Methods
  - K-Means
  - K-Medoid
  - Choice of parameters: Initialization, Silhouette coefficient
- Density-based Methods: DBSCAN
- Hierarchical Methods
  - Density-based hierarchical clustering: OPTICS
  - Agglomerative Hierarchical Clustering: Single-Link + Variants
- Scaling Up Clustering Algorithms
  - BIRCH, Data Bubbles, Index-based Clustering, GRID Clustering
- Sets of Similarity Queries (Similarity Join)
- Advanced Clustering Topics
  - Incremental Clustering, Generalized DBSCAN, Outlier Detection
- Subspace Clustering

Expectation Maximization (EM)

Basic Notions [Dempster, Laird & Rubin 1977]

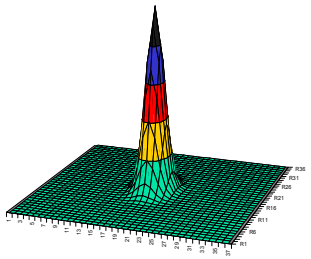
- Consider points $p = (x_1, ..., x_d)$ from a d-dimensional Euclidean vector space
- Each cluster is represented by a probability distribution
- Typically: mixture of Gaussian distributions
- Single distribution to represent a cluster C
  - Center point μ_C of all points in the cluster
  - d x d Covariance matrix Σ_C for the points in the cluster C
- Density function for cluster C:

$$P(x \mid C) = \frac{1}{\sqrt{(2\pi)^d |\Sigma_C|}} \cdot e^{\frac{1}{2} \cdot (x-\mu_C)^T \cdot (\Sigma_C)^{-1} \cdot (x-\mu_C)}$$

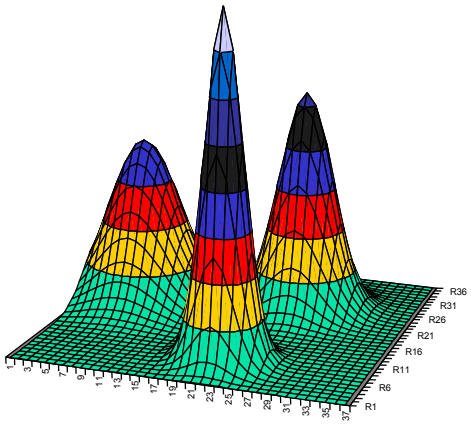


EM: Gaussian Mixture – 2D examples

A single Gaussian density function



A Gaussian mixture model, $k = 3$



EM – Basic Notions

- Density function for clustering $M = \{C_1, \ldots, C_k\}$
  - Estimate the a-priori probability of class C_i, $P(C_i)$, by the relative frequency W_i, i.e., the fraction of cluster C_i in the entire data set D:

$$P(x) = \sum_{i=1}^{k} W_i \cdot P(x \mid C_i)$$

- Assignment of points to clusters
  - A point may belong to several clusters with different probabilities $P(x|C_i)$

$$P(C_i \mid x) = W_i \cdot \frac{P(x \mid C_i)}{P(x)}$$

cf. Bayes Rule:

$$P(C_i \mid x) \cdot P(x) = P(x \mid C_i) \cdot P(C_i)$$

- Maximize *E(M),* a measure for the quality of a clustering M
  - *E(M)* indicates the probability that the data D have been generated by following the distribution model M

$$E(M) = \sum_{x \in D} \log(P(x))$$

EM – Algorithm

ClusteringByExpectationMaximization (point set D, int k)

Generate an initial model M′ = (C_1', …, C_k')

repeat

// (re-) assign points to clusters

For each object x from D and for each cluster (= Gaussian) C_i, compute P(x|C_i), P(x) and P(C_i|x)

// (re-) compute the models

For each Cluster C_i, compute a new model M = {C_1, …, C_k} by recomputing W_i, μ_C and Σ_C

Replace M′ by M

until |E(M) – E(M′)| < ε

return M

EM – Recomputation of Parameters

- Weight W_i of cluster C_i = a-priori probability P(C_i)

$$W_i = \frac{1}{n} \sum_{x \in D} P(C_i \mid x)$$

- Center μ_i of cluster C_i

$$\mu_i = \frac{\sum_{x \in D} x \cdot P(C_i \mid x)}{\sum_{x \in D} P(C_i \mid x)}$$

- Covariance matrix Σ_i of cluster C_i

$$\Sigma_i = \frac{\sum_{x \in D} P(C_i \mid x)(x - \mu_i)(x - \mu_i)^T}{\sum_{x \in D} P(C_i \mid x)}$$

outer product

EM – Discussion

- Convergence to (possibly local) minimum
- Computational effort:
  - O($n \cdot k \cdot$ #iterations)
  - #iterations is quite high in many cases
- Both result and runtime strongly depend on
  - the initial assignment
  - a proper choice of parameter k (= desired number of clusters)
- Modification to obtain a really *partitioning* variant
  - Objects may belong to several clusters
  - Assign each object to the cluster to which it belongs with the highest probability

Chapter 3: Clustering

- Introduction to clustering
- Expectation Maximization: a statistical approach
- Partitioning Methods
  - K-Means
  - K-Medoid
  - Choice of parameters: Initialization, Silhouette coefficient
- Density-based Methods: DBSCAN
- Hierarchical Methods
  - Density-based hierarchical clustering: OPTICS
  - Agglomerative Hierarchical Clustering: Single-Link + Variants
- Scaling Up Clustering Algorithms
  - BIRCH, Data Bubbles, Index-based Clustering, GRID Clustering
- Sets of Similarity Queries (Similarity Join)
- Advanced Clustering Topics
  - Incremental Clustering, Generalized DBSCAN, Outlier Detection
- Subspace Clustering

Partitioning Algorithms: Basic Concept

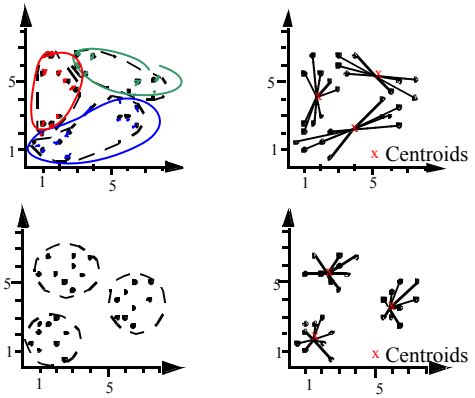
- *Goal:* Construct a partition of a database D of n objects into a set of k clusters minimizing an objective function.
  - Exhaustively enumerating all possible partitions into k sets in order to find the global minimum is too expensive.
- Heuristic methods:
  - Choose k representatives for clusters, e.g., randomly
  - Improve these initial representatives iteratively:
    - Assign each object to the cluster it "fits best" in the current clustering
    - Compute new cluster representatives based on these assignments
    - Repeat until the change in the objective function from one iteration to the next drops below a threshold
- Types of cluster representatives
  - k-means: Each cluster is represented by the center of the cluster
  - k-medoid: Each cluster is represented by one of its objects

Chapter 3: Clustering

- Introduction to clustering
- Expectation Maximization: a statistical approach
- Partitioning Methods
  - K-Means
  - K-Medoid
  - Choice of parameters: Initialization, Silhouette coefficient
- Density-based Methods: DBSCAN
- Hierarchical Methods
  - Density-based hierarchical clustering: OPTICS
  - Agglomerative Hierarchical Clustering: Single-Link + Variants
- Scaling Up Clustering Algorithms
  - BIRCH, Data Bubbles, Index-based Clustering, GRID Clustering
- Sets of Similarity Queries (Similarity Join)
- Advanced Clustering Topics
  - Incremental Clustering, Generalized DBSCAN, Outlier Detection
- Subspace Clustering

K-Means Clustering: Basic Idea

- Objective: For a given *k*, form *k* groups so that the sum of the (squared) distances between the mean of the groups and their elements is minimal.
  - Poor Clustering
  - Optimal Clustering



K-Means Clustering: Basic Notions

- Objects $p = (x^p_1, ..., x^p_d)$ are points in a d-dimensional vector space (the mean of a set of points must be defined)
- *Centroid* μ_C: Mean of all points in a cluster C, $\mu_C = \frac{1}{|C|}\sum_{x_i \in C} x_i$
- Measure for the compactness („Total Distance") of a **cluster** C_j:

$$TD(C_j) = \sqrt{\sum_{p \in C_j} dist(p, \mu_{C_j})^2}$$

- Measure for the compactness of a **clustering**

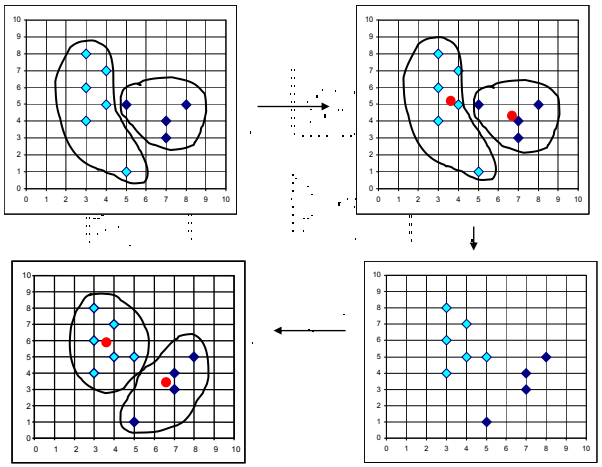
$$TD = \sqrt{\sum_{j=1}^{k} TD^2(C_j)}$$

K-Means Clustering: Algorithm

Given *k*, the *k*-means algorithm is implemented in 4 steps:

1. Partition the objects into *k* nonempty subsets
2. Compute the centroids of the clusters of the current partition. The centroid is the center (mean point) of the cluster.
3. Assign each object to the cluster with the nearest representative.
4. Go back to Step 2, stop when representatives do not change.

K-Means Clustering: Example



K-Means Clustering: Discussion

- Strength
  - Relatively efficient: $O(tkn)$, where n is # objects, k is # clusters, and t is # iterations
  - Normally: $k, t << n$
  - Easy implementation
- Weakness
  - Applicable only when mean is defined
  - Need to specify k, the number of clusters, in advance
  - Sensitive to noisy data and outliers
  - Clusters are forced to have convex shapes
  - Result and runtime are very dependent on the initial partition; often terminates at a *local optimum* – however: methods for a good initialization exist

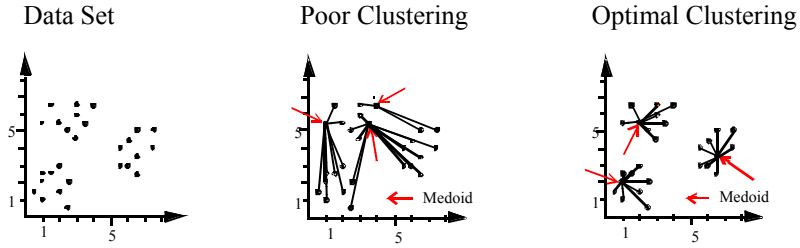
- Several variants of the *k*-means method exist, e.g. ISODATA
  - Extends *k*-means by methods to eliminate very small clusters, merging and split of clusters; user has to specify additional parameters

Chapter 3: Clustering

- Introduction to clustering
- Expectation Maximization: a statistical approach
- Partitioning Methods
  - K-Means
  - K-Medoid
  - Choice of parameters: Initialization, Silhouette coefficient
- Density-based Methods: DBSCAN
- Hierarchical Methods
  - Density-based hierarchical clustering: OPTICS
  - Agglomerative Hierarchical Clustering: Single-Link + Variants
- Scaling Up Clustering Algorithms
  - BIRCH, Data Bubbles, Index-based Clustering, GRID Clustering
- Sets of Similarity Queries (Similarity Join)
- Advanced Clustering Topics
  - Incremental Clustering, Generalized DBSCAN, Outlier Detection
- Subspace Clustering

K-Medoid Clustering: Basic Idea

- Objective: For a given k, find k representatives in the dataset so that, when assigning each object to the closest representative, the sum of the distances between representatives and objects which are assigned to them is minimal.
- Medoid: representative object "in the middle" (cf. median)



K-Medoid Clustering: Basic Notions

- Requires arbitrary objects and a distance function
- *Medoid* m_C: representative object in a cluster C
- Measure for the compactness of a cluster C:

$$TD(C) = \sum_{p \in C} dist(p, m_C)$$

- Measure for the compactness of a clustering

$$TD = \sum_{i=1}^{k} TD(C_i)$$

K-Medoid Clustering: PAM Algorithm

- Partitioning Around Medoids [Kaufman and Rousseeuw, 1990]
- Given *k*, the *k*-medoid algorithm is implemented in 5 steps:
  1. Select *k* objects arbitrarily as medoids (representatives); assign each remaining (non-medoid) object to the cluster with the nearest representative, and compute $TD_{current}$.
  2. For **each** pair (medoid *M*, non-medoid *N*)
     - compute the value $TD_{N\leftrightarrow M}$, i.e., the value of TD for the partition that results when "swapping" *M* with *N*
  3. Select the non-medoid *N* for which $TD_{N\leftrightarrow M}$ is minimal
  4. If $TD_{N\leftrightarrow M}$ is smaller than $TD_{current}$
     - Swap *N* with *M*
     - Set $TD_{current}$:= $TD_{N\leftrightarrow M}$
     - Go back to Step 2
  5. Stop.

K-Medoid Clustering: CLARA and CLARANS

- CLARA [Kaufmann and Rousseeuw,1990]
  - Additional parameter: *numlocal*
  - Draws *numlocal* samples of the data set
  - Applies PAM on each sample
  - Returns the best of these sets of medoids as output
- CLARANS [Ng and Han, 1994)
  - Two additional parameters: *maxneighbor* and *numlocal*
  - At most *maxneighbor* many pairs (medoid *M*, non-medoid *N*) are evaluated in the algorithm.
  - The first pair (*M*, *N*) for which $TD_{N\leftrightarrow M}$ is smaller than $TD_{current}$ is swapped (instead of the pair with the minimal value of $TD_{N\leftrightarrow M}$)
  - Finding the local minimum with this procedure is repeated *numlocal* times.
- Efficiency: runtime(CLARANS) < runtime(CLARA) < runtime(PAM)

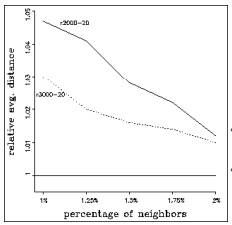
CLARANS Selection of Representatives

CLARANS (objects DB, integer k, real dist, integer numlocal, integer maxneighbor)
  for r **from** 1 **to** numlocal **do**
    Randomly select k objects as medoids
    Let i := 0
    while i < maxneighbor **do**
      Randomly select (Medoid M, Non-medoid N)
      Compute changeOfTD_:= $TD_{N\leftrightarrow M}$ - TD
      if changeOfTD < 0 **then**
        substitute M by N
        TD := $TD_{N\leftrightarrow M}$
        i := 0
      else i:= i + 1
    if TD < TD_best **then**
      TD_best := TD; Store current medoids
  return Medoids

K-Medoid Clustering: Discussion

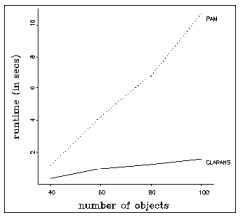
- Strength
  - Applicable to arbitrary objects + distance function
  - Not as sensitive to noisy data and outliers as *k*-means
- Weakness
  - Inefficient
  - Like *k*-means: need to specify the number of clusters *k* in advance, and clusters are forced to have convex shapes
  - Result and runtime for CLARA and CLARANS may vary largely due to the randomization

20 rectangular clusters out of
--- 2000 points
- - 3000 points



TD(CLARANS)

TD(PAM)



Chapter 3: Clustering

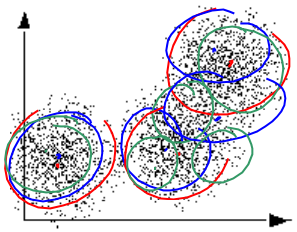
- Introduction to clustering
- Expectation Maximization: a statistical approach
- Partitioning Methods
  - K-Means
  - K-Medoid
  - Choice of parameters: Initialization, Silhouette coefficient
- Density-based Methods: DBSCAN
- Hierarchical Methods
  - Density-based hierarchical clustering: OPTICS
  - Agglomerative Hierarchical Clustering: Single-Link + Variants
- Scaling Up Clustering Algorithms
  - BIRCH, Data Bubbles, Index-based Clustering, GRID Clustering
- Sets of Similarity Queries (Similarity Join)
- Advanced Clustering Topics
  - Incremental Clustering, Generalized DBSCAN, Outlier Detection
- Subspace Clustering

Initialization of Partitioning Clustering Methods

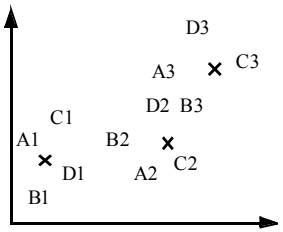
- [Fayyad, Reina and Bradley 1998]
  - Draw m different (small) samples of the dataset
  - Cluster each sample to get m estimates for k representatives $A = (A_1, A_2, \ldots, A_k)$, $B = (B_1, \ldots, B_k)$, ..., $M = (M_1, \ldots, M_k)$
  - Then, cluster the set $DS = A \cup B \cup \ldots \cup M$ m times, using the sets A, B, ..., M as respective initial partitioning
  - Use the best of these m clusterings as initialization for the partitioning clustering of the whole dataset

Initialization of Partitioning Clustering Methods

Example



whole dataset
$k = 3$



DS
$m = 4$ samples A, B, C, D

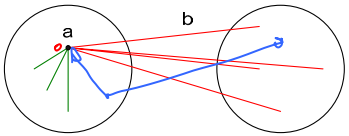
× true cluster centers

Choice of the Parameter k

- Idea for a method:
  - Determine a clustering for each $k = 2, \ldots n\text{-}1$
  - Choose the „best" clustering
- But how can we measure the quality of a clustering?
  - A measure has to be independent of k.
  - The measures for the compactness of a clustering TD^2 and TD are monotonously decreasing with increasing value of k.
- Silhouette-Coefficient [Kaufman & Rousseeuw 1990]
  - Measure for the quality of a k-means or a k-medoid clustering that is independent of k.

The silhouette coefficient (1)

- Basic idea:
  - How good is the clustering = how appropriate is the mapping of objects to clusters
  - Elements in cluster should be „similar" to their representative
    → measure the average distance of objects to their representative: *a*
  - Elements in different clusters should be „dissimilar"
    → measure the average distance of objects to alternative cluster (i.e. second closest cluster): *b*



The silhouette coefficient (2)

- *a(o):* average distance between object *o* and the objects in its cluster *A*

$$a(o) = \frac{1}{|C_i|} \sum_{p \in C(o)} dist(o,p)$$

- *b(o):* average distance between object *o* and the objects in its "second closest" cluster *B*

$$b(o) = \min_{C_i \neq C(o)} \left(\frac{1}{|C_i|} \sum_{p \in C_i} dist(o,p) \right)$$

- The silhouette of *o* is then defined as

$$s(o) = \frac{b(o) - a(o)}{\max\{a(o), b(o)\}}$$

- The values of the silhouette coefficient range from –1 to +1

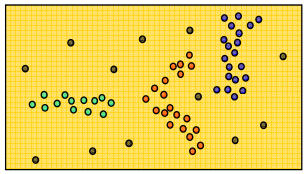
The silhouette coefficient (3)

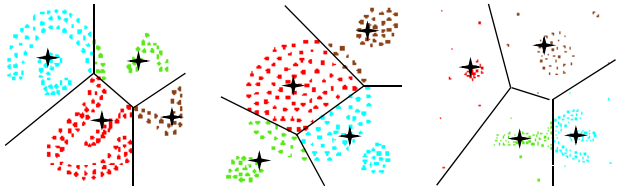
- „Reading" the silhouette coefficient
  - how good is the assignment of *o* to its cluster
    - $s(o) = -1$: bad, on average closer to members of *B*
    - $s(o) = 0$: in-between *A* and *B*
    - $s(o) = 1$: good assignment of *o* to its cluster *A*
- Silhouette Coefficient s_C of a clustering: average silhouette of all objects
    - $0.7 < s_C \leq 1.0$ strong structure, $0.5 < s_C \leq 0.7$ medium structure
    - $0.25 < s_C \leq 0.5$ weak structure, $s_C \leq 0.25$ no structure

Chapter 3: Clustering

Density-Based Clustering

- Basic Idea:
  - Clusters are dense regions in the data space, separated by regions of lower object density
- Why Density-Based Clustering?



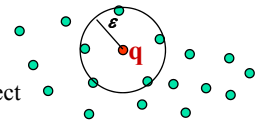


Results of a *k*-medoid algorithm for *k*=4

Different density-based approaches exist (see Textbook & Papers)
Here we discuss the ideas underlying the DBSCAN algorithm

Density Based Clustering: Basic Concept

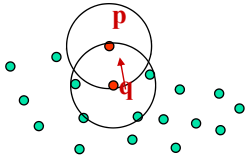
- Intuition for the formalization of the basic idea
  - For any point in a cluster, the local point density around that point has to exceed some threshold
  - The set of points from one cluster is spatially connected
- Local point density at a point *p* defined by two parameters
  - ***ε*** – radius for the neighborhood of point q: $N_\varepsilon(q) := \{p \text{ in data set } D \mid dist(p, q) \leq \varepsilon\}$
  - ***MinPts*** – minimum number of points in the given neighbourhood $N(p)$
- *q* is called a ***core object*** (or core point) w.r.t. ε, *MinPts* if $| N_\varepsilon(q) | \geq MinPts$



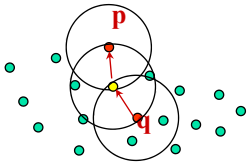
MinPts = 5 → **q** is a core object

Density Based Clustering: Basic Definitions

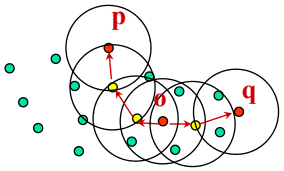
- *p* ***directly density-reachable*** from *q* w.r.t. ε, *MinPts* if
  1) $p \in N_\varepsilon(q)$ and
  2) *q* is a core object w.r.t. ε, *MinPts*



- ***density-reachable***: transitive closure of *directly* density-reachable

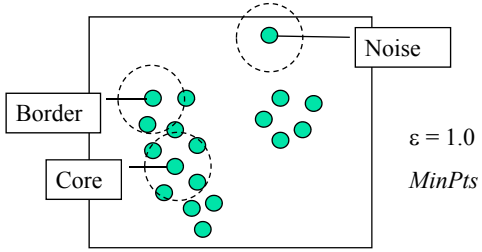


- *p* is ***density-connected*** to a point *q* w.r.t. ε, *MinPts* if there is a point *o* such that both, *p* and *q* are density-reachable from *o* w.r.t. ε, *MinPts*.



Density Based Clustering: Basic Definitions

- ***Density-Based Cluster****:* non-empty subset *S* of database *D* satisfying:
  1) *Maximality*: if *p* is in *S* and *q* is density-reachable from *p* then *q* is in *S*
  2) *Connectivity*: each object in *S* is density-connected to all other objects
- ***Density-Based Clustering*** of a database $D : \{S_1, ..., S_n; N\}$ where
  - $S_1, ..., S_n$: all density-based clusters in the database *D*
  - $N = D \setminus \{S_1, ..., S_n\}$ is called the ***noise*** (objects not in any cluster)



ε = 1.0

MinPts = 5

Density Based Clustering: DBSCAN Algorithm

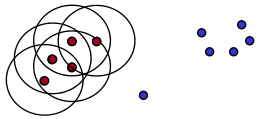
- Density Based Spatial Clustering of Applications with Noise
- Basic Theorem:
  - Each object in a density-based cluster C is density-reachable from any of its core-objects
  - Nothing else is density-reachable from core objects.

```
for each o ∈ D do
    if o is not yet classified then
        if o is a core-object then
            collect all objects density-reachable from o
            and assign them to a new cluster.
        else
            assign o to NOISE
```

  - density-reachable objects are collected by performing successive ε-neighborhood queries.

DBSCAN Algorithm: Example

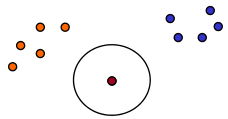
- Parameter
  - ε = 2.0
  - *MinPts* = 3



```
for each o ∈ D do
    if o is not yet classified then
        if o is a core-object then
            collect all objects density-reachable from o
            and assign them to a new cluster.
        else
            assign o to NOISE
```


DBSCAN Algorithm: Example

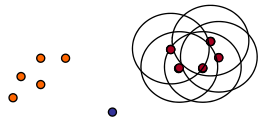
- Parameter
  - ε = 2.0
  - *MinPts* = 3



```
for each o ∈ D do
    if o is not yet classified then
        if o is a core-object then
            collect all objects density-reachable from o
            and assign them to a new cluster.
        else
            assign o to NOISE
```


DBSCAN Algorithm: Example

- Parameter
  - ε = 2.0
  - *MinPts* = 3



```
for each o ∈ D do
    if o is not yet classified then
        if o is a core-object then
            collect all objects density-reachable from o
            and assign them to a new cluster.
        else
            assign o to NOISE
```

DBSCAN Algorithm: Performance

- Runtime complexity: O(*n* * cost for neighborhood query)

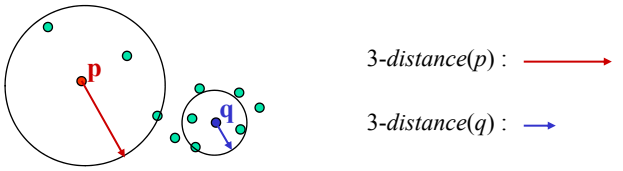
	N_ε-query	DBSCAN
- without support (worst case):	O(*n*)	O(n^2)
- tree-based support (e.g. R*-tree) :	O(log(*n*))	O($n * \log(n)$)
- direct access to the neighborhood:	O(1)	O(*n*)

- Runtime Comparison: DBSCAN (+ R*-tree) ↔ CLARANS

	Time (sec.)									
No. of Points	1,252	2,503	3,910	5,213	6,256	7,820	8,937	10,426	12,512	62,584
DBSCAN	3	7	11	16	18	25	28	33	42	233
CLARANS	758	3,026	6,845	11,745	18,029	29,826	39,265	60,540	80,638	?????

Determining the Parameters ε and *MinPts*

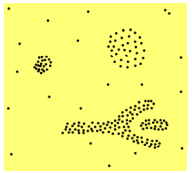
- Cluster: Point density higher than specified by ε and *MinPts*
- Idea: use the point density of the least dense cluster in the data set as parameters – but how to determine this?
- Heuristic: look at the distances to the *k*-nearest neighbors

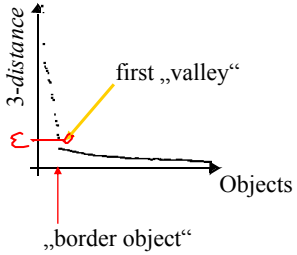


- Function *k-distance*(*p*): distance from *p* to the its *k*-nearest neighbor
- *k-distance plot*: *k*-distances of all objects, sorted in decreasing order

Determining the Parameters ε and *MinPts*

- Example *k*-distance plot
  1. dim = 2 → MinPts = 3
  2. Identify border object
  3. Set ε

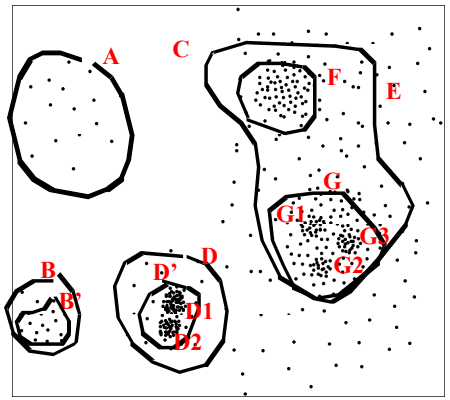


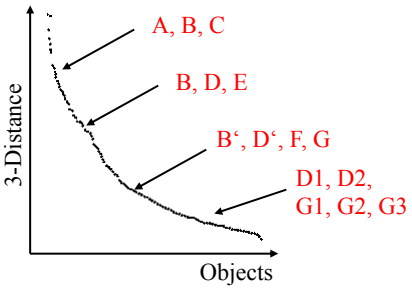


- Heuristic method:
  - Fix a value for *MinPts*
  - (default: 2 × *d* − 1, *d* = dimension of data space)
  - User selects "border object" *o* from the *MinPts-distance* plot; ε is set to *MinPts-distance*(o)

Determining the Parameters ε and *MinPts*

- Problematic example





Density Based Clustering: Discussion

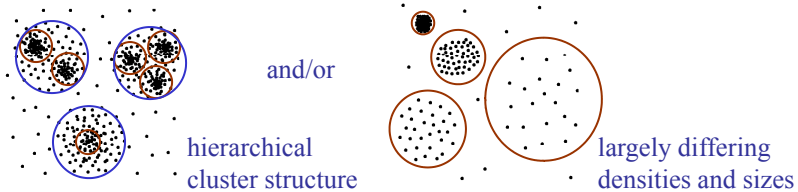
- Advantages
  - Clusters can have arbitrary shape and size, i.e. clusters are not restricted to have convex shapes
  - Number of clusters is determined automatically
  - Can separate clusters from surrounding noise
  - Can be supported by spatial index structures
- Disadvantages
  - Input parameters may be difficult to determine
  - In some situations very sensitive to input parameter setting

Chapter 3: Clustering

- Introduction to clustering
- Expectation Maximization: a statistical approach
- Partitioning Methods
  - K-Means
  - K-Medoid
  - Choice of parameters: Initialization, Silhouette coefficient
- Density-based Methods: DBSCAN
- Hierarchical Methods
  - Density-based hierarchical clustering: OPTICS
  - Agglomerative Hierarchical Clustering: Single-Link + Variants
- Scaling Up Clustering Algorithms
  - BIRCH, Data Bubbles, Index-based Clustering, GRID Clustering
- Sets of Similarity Queries (Similarity Join)
- Advanced Clustering Topics
  - Incremental Clustering, Generalized DBSCAN, Outlier Detection
- Subspace Clustering

From Partitioning to Hierarchical Clustering

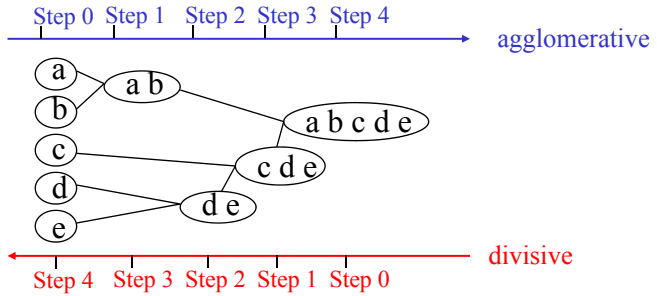
- Global parameters to separate all clusters with a partitioning clustering method may not exist



- Need a hierarchical clustering algorithm in these situations

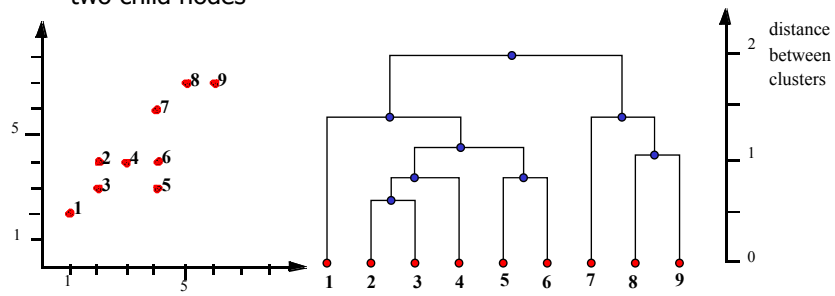
Hierarchical Clustering: Basic Notions

- Hierarchical decomposition of the data set (with respect to a given similarity measure) into a set of nested clusters
- Result represented by a so called *dendrogram* (greek δενδρο = tree)
  - Nodes in the dendrogram represent possible clusters
  - can be constructed bottom-up (agglomerative approach) or top down (divisive approach)



Hierarchical Clustering: Example

- Interpretation of the dendrogram
  - The root represents the whole data set
  - A leaf represents a single objects in the data set
  - An internal node represent the union of all objects in its sub-tree
  - The height of an internal node represents the distance between its two child nodes

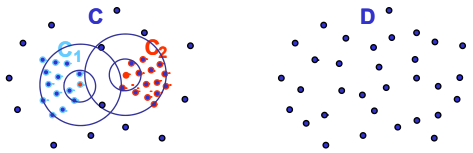


Chapter 3: Clustering

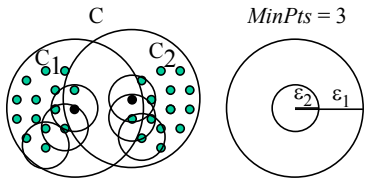
- Introduction to clustering
- Expectation Maximization: a statistical approach
- Partitioning Methods
  - K-Means
  - K-Medoid
  - Choice of parameters: Initialization, Silhouette coefficient
- Density-based Methods: DBSCAN
- Hierarchical Methods
  - Density-based hierarchical clustering: OPTICS
  - Agglomerative Hierarchical Clustering: Single-Link + Variants
- Scaling Up Clustering Algorithms
  - BIRCH, Data Bubbles, Index-based Clustering, GRID Clustering
- Sets of Similarity Queries (Similarity Join)
- Advanced Clustering Topics
  - Incremental Clustering, Generalized DBSCAN, Outlier Detection
- Subspace Clustering

Density-Based Hierarchical Clustering

- *Observation*: Dense clusters are completely contained by less dense clusters



- *Idea*: Process objects in the "right" order and keep track of point density in their neighborhood



Core- and Reachability Distance

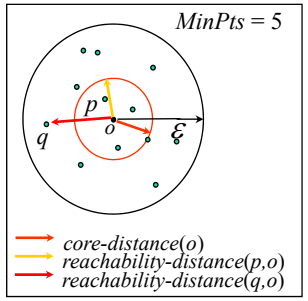
- Parameters: "generating" distance ε, fixed value *MinPts*
- ***core-distance*$_{\varepsilon,MinPts}$*(o)***

  "smallest distance such that *o* is a core object"
  (if that distance is $\leq \varepsilon$; "?" otherwise)

- ***reachability-distance*$_{\varepsilon,MinPts}$*(p, o)***

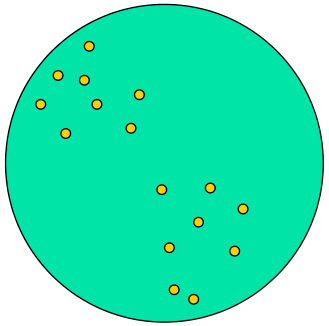
  "smallest distance such that *p* is *directly* density-reachable from *o*"
  (if that distance is $\leq \varepsilon$; "?" otherwise)

$$r\text{-}dist(p,o) = \begin{cases} dist(p,o) & dist(p,o) > c\text{-}dist(o) \\ c\text{-}dist(o) & dist(p,o) < c\text{-}dist(o) \\ undef & dist(p,o) > \varepsilon \end{cases}$$



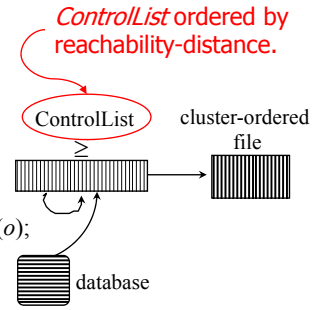
The Algorithm OPTICS

- OPTICS: **O**rdering **P**oints **T**o **I**dentify the **C**lustering **S**tructure
- Basic data structure: controlList
  - Memorize shortest reachability distances seen so far ("distance of a jump to that point")
- Visit each point
  - Make always a shortest jump
- Output:
  - order of points
  - core-distance of points
  - reachability-distance of points



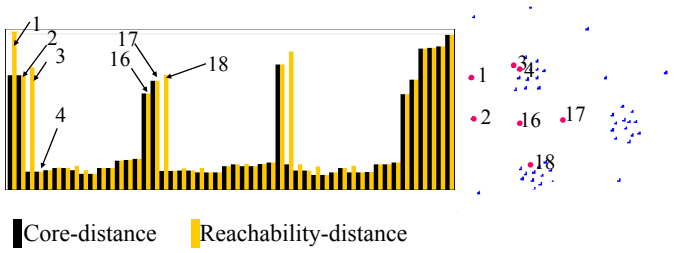
The Algorithm OPTICS

foreach $o \in$ Database
 // initially, *o*.processed = false for all objects *o*
 if *o*.processed = false;
  insert (*o*, "?") into *ControlList;*
 while *ControlList* is not empty
  select first element (*o*, *r-dist*) from *ControlList*;
  retrieve $N_\varepsilon(o)$ and determine *c_dist*= *core-distance*(*o*);
  set *o*.processed = true;
  write (*o*, *r_dist*, *c_dist*) to file;
  if *o* is a core object at any distance $\leq \varepsilon$
   foreach $p \in N_\varepsilon(o)$ not yet processed;
    determine r_dist_p = *reachability-distance*(*p*, *o*);
    if (*p*, _) $\notin$ *ControlList*
     insert (p, r_dist_p) in *ControlList;*
    else if (*p*, *old_r_dist*) $\in$ *ControlList* **and** $r\_dist_p < old\_r\_dist$
     update (p, r_dist_p) in *ControlList*;



OPTICS: Properties

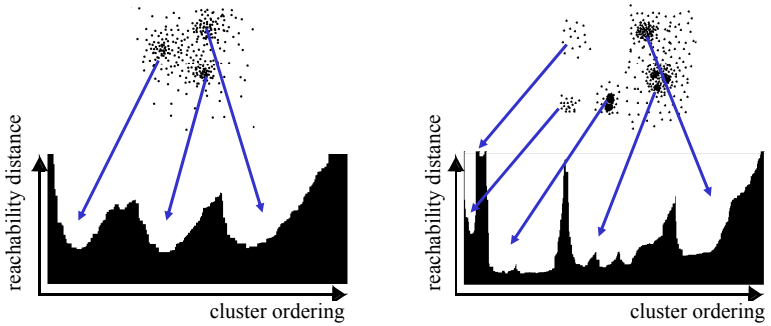
- "Flat" density-based clusters wrt. $\varepsilon^* \leq \varepsilon$ and *MinPts* afterwards:
  - Starts with an object *o* where *c-dist*(*o*) $\leq \varepsilon^*$ and *r-dist*(*o*) $> \varepsilon^*$
  - Continues while *r-dist* $\leq \varepsilon^*$



- Performance: approx. runtime(DBSCAN(ε, *MinPts*))
  - O(*n* * runtime(ε-neighborhood-query))
    - without spatial index support (worst case): O(n^2)
    - e.g. tree-based spatial index support: O($n * \log(n)$)

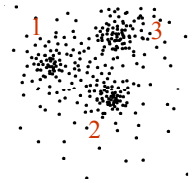
OPTICS: The Reachability Plot

- represents the density-based clustering structure
- easy to analyze
- independent of the dimension of the data



OPTICS: Parameter Sensitivity

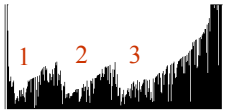
- Relatively insensitive to parameter settings
- Good result if parameters are just "large enough"



MinPts = 10, ε = 10

MinPts = 10, **ε = 5**

***MinPts* = 2**, ε = 10



Chapter 3: Clustering

- Introduction to clustering
- Expectation Maximization: a statistical approach
- Partitioning Methods
  - K-Means
  - K-Medoid
  - Choice of parameters: Initialization, Silhouette coefficient
- Density-based Methods: DBSCAN
- Hierarchical Methods
  - Density-based hierarchical clustering: OPTICS
  - Agglomerative Hierarchical Clustering: Single-Link + Variants
- Scaling Up Clustering Algorithms
  - BIRCH, Data Bubbles, Index-based Clustering, GRID Clustering
- Sets of Similarity Queries (Similarity Join)
- Advanced Clustering Topics
  - Incremental Clustering, Generalized DBSCAN, Outlier Detection
- Subspace Clustering

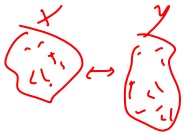
Agglomerative Hierarchical Clustering

1. Initially, each object forms its own cluster
2. Compute all pairwise distances between the initial clusters (objects)
3. Merge the closest pair (A, B) in the set of the current clusters into a new cluster C = A ∪ B
4. Remove A and B from the set of current clusters; insert C into the set of current clusters
5. If the set of current clusters contains only C (i.e., if C represents all objects from the database): STOP
6. Else: determine the distance between the new cluster C and all other clusters in the set of current clusters; go to step 3.

➢ Requires a distance function for clusters (sets of objects)

Single Link Method and Variants

- Given: a distance function *dist*(*p*, *q*) for database objects
- The following distance functions for clusters (i.e., sets of objects) *X* and *Y* are commonly used for hierarchical clustering:



Single-Link: $dist\_sl(X,Y) = \min_{x \in X, y \in Y} dist(x,y)$

Complete-Link: $dist\_cl(X,Y) = \max_{x \in X, y \in Y} dist(x,y)$

Average-Link: $dist\_al(X,Y) = \frac{1}{|X| \cdot |Y|} \cdot \sum_{x \in X, y \in Y} dist(x,y)$

Hierarchical Clustering: Discussion

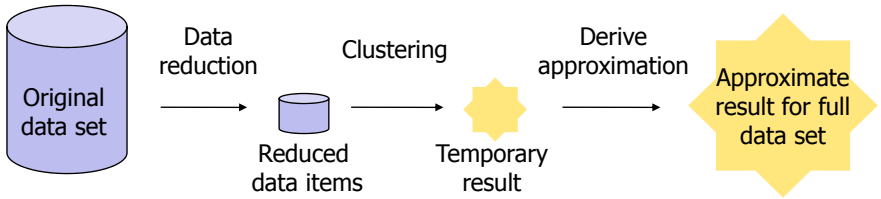
- Advantages
  - Does not require the number of clusters to be known in advance
  - No (standard methods) or very robust parameters (OPTICS)
  - Computes a complete hierarchy of clusters
  - Good result visualizations integrated into the methods
  - A "flat" partition can be derived afterwards (e.g. via a cut through the dendrogram or the reachability plot)
- Disadvantages
  - May not scale well
    - Runtime for the standard methods: $O(n^2 \log n^2)$
    - Runtime for OPTICS: without index support $O(n^2)$

Chapter 3: Clustering

- Introduction to clustering
- Expectation Maximization: a statistical approach
- Partitioning Methods
  - K-Means
  - K-Medoid
  - Choice of parameters: Initialization, Silhouette coefficient
- Density-based Methods: DBSCAN
- Hierarchical Methods
  - Density-based hierarchical clustering: OPTICS
  - Agglomerative Hierarchical Clustering: Single-Link + Variants
- Scaling Up Clustering Algorithms
  - BIRCH, Data Bubbles, Index-based Clustering, GRID Clustering
- Sets of Similarity Queries (Similarity Join)
- Advanced Clustering Topics
  - Incremental Clustering, Generalized DBSCAN, Outlier Detection
- Subspace Clustering

Scaling-Up Clustering Algorithms by using Data Reduction/Summarizations

Basic Idea:



– Small Loss in Accuracy

+ Large Performance Boost

Most simple approach: Random Sampling

Chapter 3: Clustering

- Introduction to clustering
- Expectation Maximization: a statistical approach
- Partitioning Methods
  - K-Means
  - K-Medoid
  - Choice of parameters: Initialization, Silhouette coefficient
- Density-based Methods: DBSCAN
- Hierarchical Methods
  - Density-based hierarchical clustering: OPTICS
  - Agglomerative Hierarchical Clustering: Single-Link + Variants
- Scaling Up Clustering Algorithms
  - BIRCH, Data Bubbles, Index-based Clustering, GRID Clustering
- Sets of Similarity Queries (Similarity Join)
- Advanced Clustering Topics
  - Incremental Clustering, Generalized DBSCAN, Outlier Detection
- Subspace Clustering

BIRCH (1996)

- Birch: **B**alanced **I**terative **R**educing and **C**lustering using **H**ierarchies, by Zhang, Ramakrishnan, Livny (SIGMOD'96)
- Incrementally construct a CF (Clustering Feature) tree, a hierarchical data structure for multiphase clustering
  - Phase 1: scan DB to build an initial in-memory CF tree (a multi-level compression of the data that tries to preserve the inherent clustering structure of the data)
  - Phase 2: use an arbitrary clustering algorithm to cluster the leaf nodes of the CF-tree
- *Scales linearly:* finds a good clustering with a single scan and improves the quality with a few additional scans
- *Weakness:* handles only numeric data, and sensitive to the order of the data record.

Clustering of Data Summarizations: BIRCH

- Basic Idea
  - Construct a partitioning of a data set into "micro-clusters" using an efficient index-like structure
  - Micro-Clusters, i.e., sets of object are described in a compact way by *Clustering Features (CFs)*
  - *CFs* are organized hierarchically in a balanced tree
  - A standard clustering algorithm is then applied to the leaf nodes of the *CF*-tree

BIRCH – Basic Notions

- *Clustering Feature CF* = (*N*, *LS*, *SS*) of a set of points $C = \{X_i\}$
  - $N = |C|$ count(x), number of points in C
  - $LS = \sum_{i=1}^{N} X_i$ sum(x), linear sum of the N data points in C
  - $SS = \sum_{i=1}^{N} X_i^2$ *sum(x*x),* square sum of the N data points in C
- Information stored in *CFs* is sufficient to compute
  - Centroids
  - Measures for the compactness of clusters (e.g., TD, TD^2)
  - Distance measure for clusters

BIRCH – Clustering Feature Tree (CF tree)

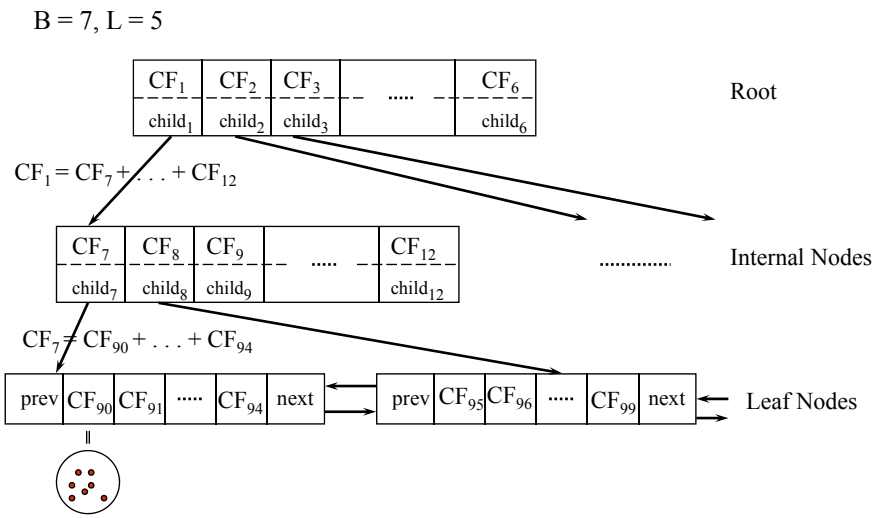
- Additivity theorem for *CFs* $C_1 = (N_1, LS_1, SS_1)$ and $C_2 = (N_2, LS_2, SS_2)$:

  $$CF(C_1 \cup C_2) = CF(C_1) + CF(C_2) = (N_1 + N_2,\ LS_1 + LS_2,\ SS_1 + SS_2)$$

  - CFs can be computed incrementally
- A *CF Tree* with parameters *B, L, T* has the following properties
  - An internal node contains at most *B* entries $[CF_i, child_i]$
  - A leaf node contains at most *L* entries $[CF_i]$
  - The *diameter* of all entries in a leaf node is at most *T*
  - Leaf nodes are connected via *prev* and *next* pointers

- *CF Tree* construction
  - Transform a point *p* into a CF-vector $CF_p = (1, p, p^2)$
  - Insertion of *p* into the tree is analog to insertion into a B^+-tree
  - If the threshold T is violated by the insertion, the corresponding leaf node is split; reorganization of the tree in case of a split is again analog to the reorganization in B^+-trees

BIRCH – Example for a *CF Tree*

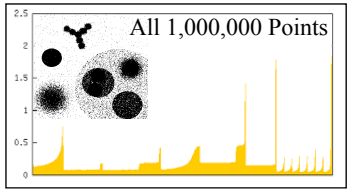


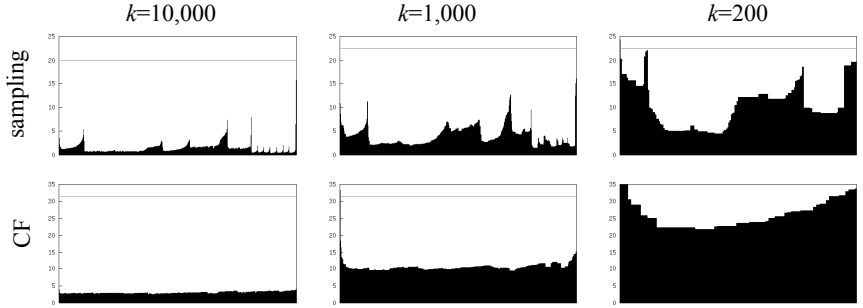
BIRCH – Discussion

- Benefits
  - Compression factor can be tuned to the available main memory
  - Efficient construction of a micro-clustering (O(n))
  - Good clustering result for partitioning iterative-refinement clustering algorithms such as k-means and k-medoid when applied to only the leaf nodes of a CF-tree
- Limitations
  - Only for data from a Euclidean vector space (linear sum, square sum, mean, etc must be defined)
  - Sensitive to the order of the data records
  - How to utilize *CF*s for hierarchical clustering algorithms?

Hierarchical clustering of summarized data items

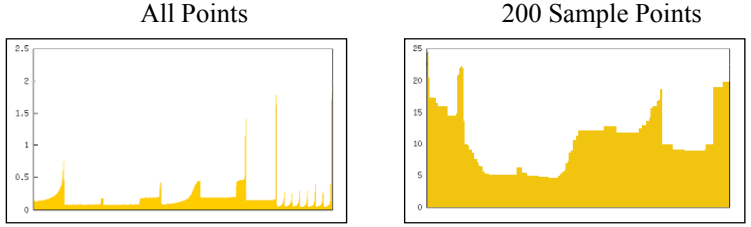
- Compression to *k* objects
  - Absolute sample size has to be very large in order to obtain good result for large data sets
  - CF centers are not suitable





Problems Clustering Summarized Data Items

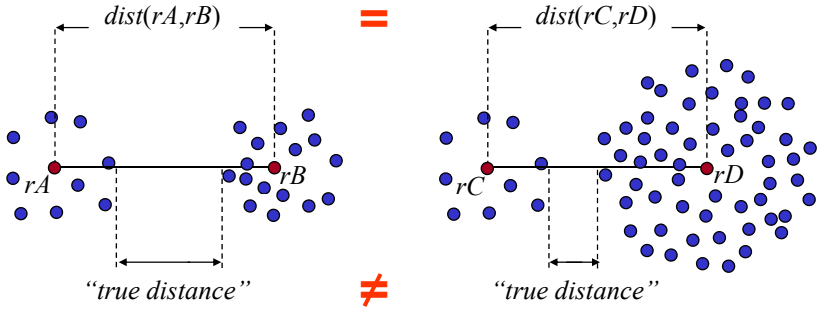
- *Structural Distortion Using High Reduction Rates*



- *Approximation of the Final Result*
  - What are the reachability values in the final reachability plot?
    - Objects can be assigned to their representatives, but the result is a graphical representation, not clusters

Reasons for Structural Distortions I

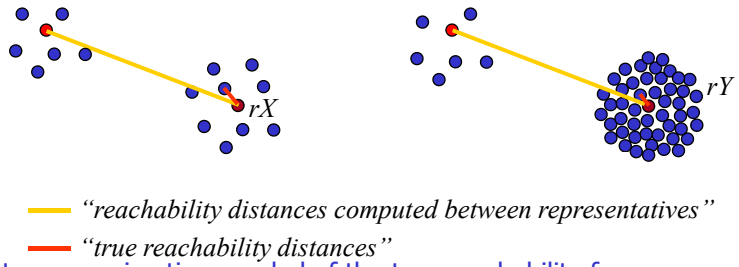
- Distances between sets of original objects are poorly approximated by distance between their representatives



⇒ better approximation needed for the distance between representatives and the points they represent

Reasons for Structural Distortions II

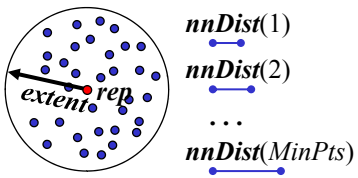
- Reachability distances for representatives poorly approximate reachability distances for represented objects



⇒ better approximation needed of the true reachability for representatives and the points they represent

Data Bubbles

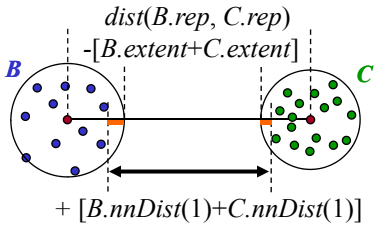
- A Data Bubble for a set of objects X is a tuple **B**=(***n***, ***rep***, ***extent***, ***nnDist***) where
  - ***n*** is the number of objects in X
  - ***rep*** is a representative object for X
  - ***extent*** is an estimation of the "radius" of X
  - ***nnDist*** is a partial function, estimating k-nearest neighbor distances in X (defined at least for k=1, ..., *MinPts*)

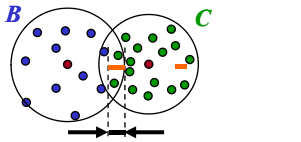


nnDist(1)
nnDist(2)
...
nnDist(*MinPts*)

Distances for Data Bubbles

- ***distance*(*B*, *C*)**





- ***core***- and ***reachability-distance*** for Data Bubbles
  - analog to core and reachability-distance for points
- ***virtual reachability-distance*** for a Data Bubble (Average reachability-distance within a Data Bubble)
  - basically *nnDist*(*MinPts*)

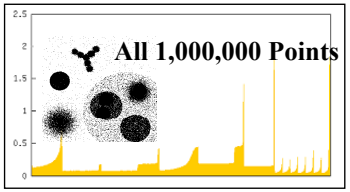
OPTICS on Data Bubbles

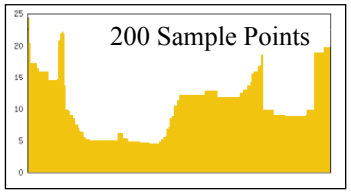
- Generate Data Bubbles
  - Either: Draw a sample of *k* points, assign each object in the database to the closest sample point and compute the Data Bubbles for the resulting *k* sets
  - Or: Use Birch to create a *CF*-tree and generate a Data Bubble for each leaf node (can be computed from the information stored in a CF)
- Apply OPTICS to the set of obtained Data Bubbles using the core- and reachability-distance defined for Data Bubbles
- Before drawing the reachability-plot for the result on the Data Bubbles, expand the Data Bubbles by adding the objects contained in the Data Bubbles using the virtual reachability

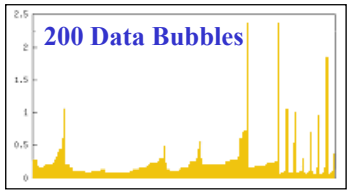
Data Bubbles for *d*-Dimensional Points

- *rep* = center = $\frac{1}{n}\left(\sum_{i=1}^{n} X_i\right) = \frac{LS}{N}$ from CF
- *extent* = average pairwise distance = $\sqrt{\frac{1}{n\cdot(n-1)}\sum_{i=1}^{n}\sum_{j=1}^{n}(X_i - X_j)^2}$
- *knnDist* = expected *knn*-distance = $\sqrt[d]{\frac{k}{n}}\cdot extent$ assuming uniform distribution
- Can be *computed* without much extra cost
  - A *nn*-classification for the final result has to be done anyway
    - Sample + *nn*-classification *before* the clustering
    - Directly from CFs

Evaluation



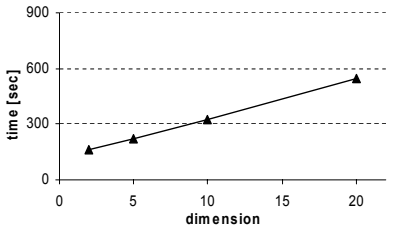




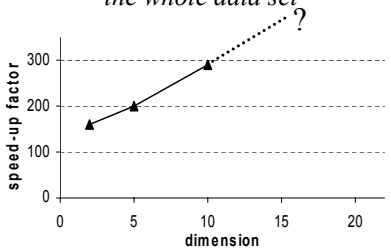
- **Speed-Up = 150 !!!**

Runtime Comparison wrt. Dimension

Runtime for OPTICS on Data Bubbles



Speed-up compared to OPTICS on the whole data set

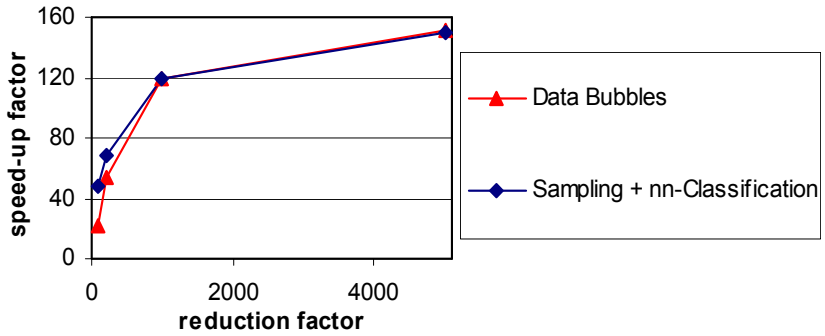


Test databases:

- 1 million objects
- 15 randomly generated Gaussian clusters of random sizes
- 1,000 Data Bubbles, using sampling

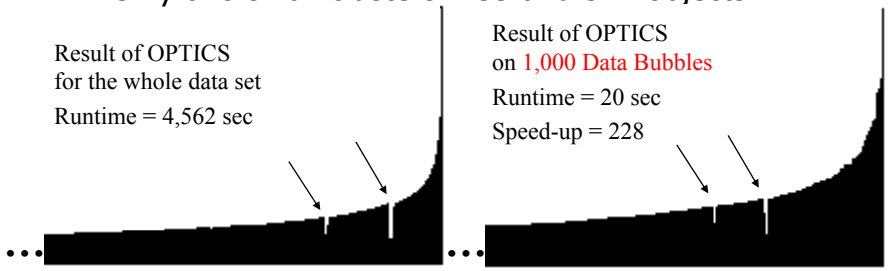
Data Bubbles vs. Sampling + nn-Classification

- No significant difference in runtime



Challenging Data Bubbles

- Corel image collection
  - features: first order moments in the HSV color scheme (Hue, Saturation, Value)
  - 68,040 feature vectors
  - Only two small clusters: 253 and 527 objects



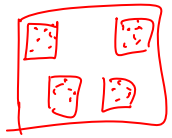
Data Bubbles – Discussion

- Benefits
  - Data Bubbles scale-up hierarchical clustering, too
  - Can be based on Sampling or Clustering Features
  - Allows for extremely high reduction factors
    - ➢ High performance boost
  - Can recover even very small clusters
    - ➢ Small loss in accuracy
- Limitations
  - Only for data from Euclidean vector spaces
    - for general metric data (example: sequence alignment): how to generate Data Bubbles, i.e., *center (medoids?)*, *extent*, and *nnDist* efficiently?

Database Techniques to Improve Runtime Efficiency – Objective

- So far
  - Small data sets
  - Main memory resident
- Now
  - Huge amounts of data that do not fit into main memory
  - Data from secondary storage (e.g., for concurrent use)
    - Access to data is very time-consuming when compared to main memory
  - Efficient algorithms required
    - i.e., runtime should not exceed $O(n \log n)$
- Scalability of clustering algorithms
  - BIRCH, Data Bubbles, Index-based Sampling, Grid Clustering

Database Techniques to Improve Runtime Efficiency – Basic Ideas

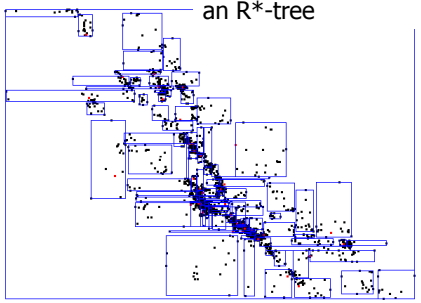


- Employ spatial index structures or related methods
- Index structures build clusters in a preliminary sense
  - Objects in close spatial neighborhood tend to be stored on the same data page in the index
- Index structures are efficient
  - Very simple heuristics for clustering
- Fast access methods for different similarity queries
  - E.g., range queries or k-nearest neighbor queries

Index-based Sampling – Method

- Proposed by Ester, Kriegel & Xu (1995)
- Intuition: Pages of certain index structures have a fixed capacity, i.e. they can store the same number of points
  - In sparse data space areas: larger regions
  - In dense data space areas : smaller regions
  - → Representatives from regions are a good sample of the data distribution as a whole
- Build a spatial index on the data (e.g., an R*-tree)
- Select representatives from the data pages of the index
- Apply a clustering algorithm to the set of representatives
- Transfer the clustering structure to the entire database

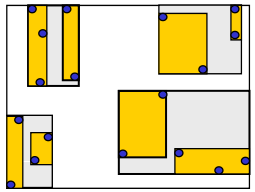
Sample data pages of an R*-tree



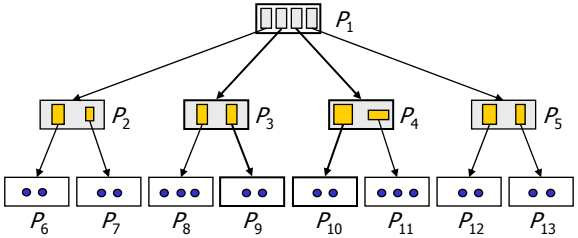
Index-based Sampling – R-Tree structure

- R-Tree: Balanced hierarchical decomposition of a multi-dimensional data space

Multidimensional points in the data space



Data and directory pages in secondary memory (disk)



Index-based Sampling – Transfer Sample Clustering to Database

- How to transfer the sample clustering to the database?
- For *k*-means or *k*-medoid algorithm:
  - Adopt the representatives of the sample clusters for the entire database (centroids, medoids)
- For density-based algorithms:
  - Create a representation for each cluster (e.g., MBR = Minimum Bounding Rectangle)
  - Assign the objects to the closest cluster
- For hierarchical algorithms:
  - Hierarchical representations are difficult to generate (dendrogram or reachability diagram)

Index-based Sampling – Selection of Representatives

- How many objects to be selected from each data page?
  - Depends on clustering method
  - Depends on distribution of data points
- Useful heuristics for CLARANS: one object per data page
  - Good trade-off between quality of clustering and runtime efficiency
- Which objects should be selected?
  - Depends on clustering method and data distribution, too.
  - Simple heuristics: select the „most central" object from a data page

Index-based Sampling

Experimental comparison for CLARANS

- Runtime of CLARANS is $O(n^2)$ for n database objects
- Clustering quality does not increase for more than 1024 representatives
- 1024 representatives seem to trade-off quality and efficiency well

Range Queries for Density-Based Clustering

- Basic operation for DBSCAN and OPTICS
  - Determine the ε-neighborhood of each object o in the database
- Efficient support of ε-range queries by using indexing structures
  - Spatial index structures: R-tree, X-tree
  - Metric index structures: M-tree, slim tree
- Recall runtime complexity for algorithms DBSCAN and OPTICS

	single range query	overall algorithm
without index	$O(n)$	$O(n^2)$
when using index	$O(\log n)$	$O(n \log n)$
using direct access	$O(1)$	$O(n)$

- high-dimensional data spaces cause problems for spatial indexes

GRID Clustering – Approach

- Method proposed by Schikuta (1996)
- Ingredients
  - Manage data by a spatial indexing structure (here: grid file)
  - Consider data pages of spatial index as cluster candidates
  - Density of points in a region r of one or more data pages

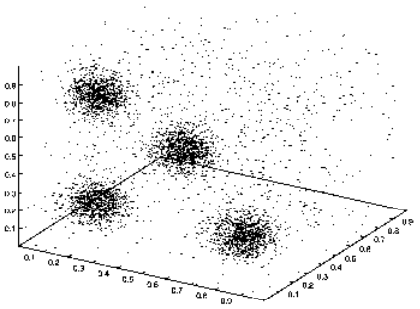
$$density(r) = \frac{count(r)}{volume(r)}$$

- Recursively merge page regions
  - Use page regions having a high density as cluster seeds
  - Merge clusters recursively with neighboring page regions that have a lower density
  - i.e., density-based clustering

GRID Clustering – Method

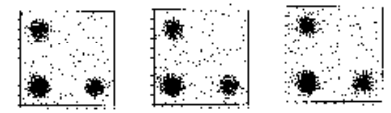
- Start with data page having the highest point density as cluster *r*
- Iteratively determine the (directly or indirectly) neighboring data pages *s* that have a density less or equal to the density of cluster *r*
  - merge these pages *s* with cluster *r*
  - recompute the new value of *density(r)*
- If there are only neighboring pages having a higher density than cluster *r*, then create a new cluster starting at the page that has the highest density among the data pages not yet considered
- When including the information about the merging order the result may be visualized as a dendrogram

GRID Clustering – Example

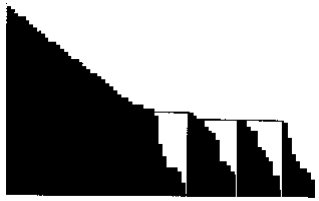


3-dimensional data points

2D projections of 3D data:



Dim1:Dim2 Dim1:Dim3 Dim2:Dim3



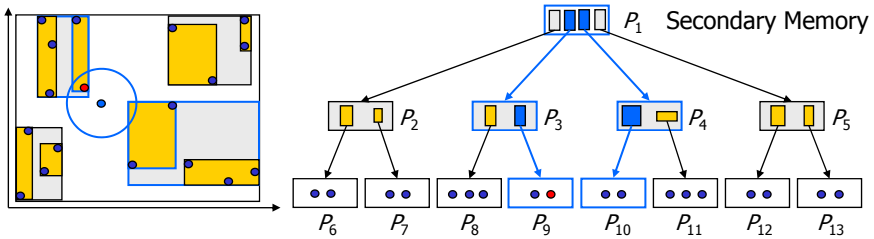
Resulting dendrogram

Chapter 3: Clustering

- Introduction to clustering
- Expectation Maximization: a statistical approach
- Partitioning Methods
  - K-Means
  - K-Medoid
  - Choice of parameters: Initialization, Silhouette coefficient
- Density-based Methods: DBSCAN
- Hierarchical Methods
  - Density-based hierarchical clustering: OPTICS
  - Agglomerative Hierarchical Clustering: Single-Link + Variants
- Scaling Up Clustering Algorithms
  - BIRCH, Data Bubbles, Index-based Clustering, GRID Clustering
- Sets of Similarity Queries (Similarity Join)
- Advanced Clustering Topics
  - Incremental Clustering, Generalized DBSCAN, Outlier Detection
- Subspace Clustering

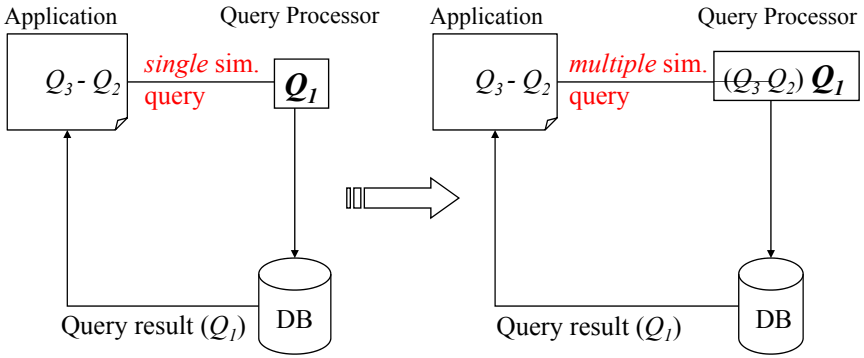
Supporting Sets of Similarity-Queries

- Compute similarity (self-) join
- Many DM algorithms, e.g., density-based clustering, generate large numbers of similarity queries that have to be processed
  - The algorithm generates a whole set of queries in one iteration.
  - However, only one is executed at a time



Potentials for Optimization

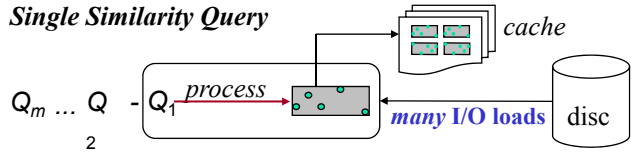
- Idea: Process sets of similarity queries simultaneously



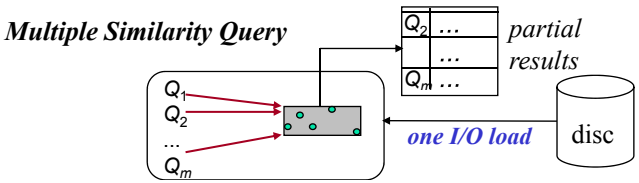
Techniques to Improve Efficiency

- Make "better usage" of the cache
  - Reduction of I/O loads
    - Input/Output operations = disk accesses
    - Single random disk access: ca. 8-19 msec.
- Try to avoid distance calculations
  - Reduction of CPU cost

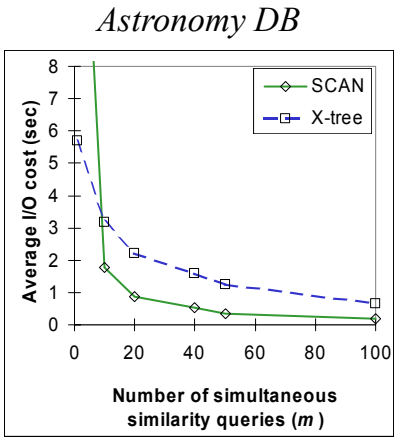
Reduction of the I/O cost

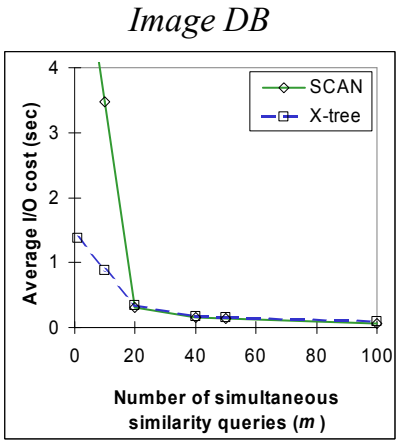


Better "caching" strategy: Process a loaded data page immediately for all queries in the set and store partial results instead of whole pages.



Reduction of the I/O cost – Evaluation





Reduction of the CPU cost (1)

- Given
  - Set of query objects $Q_1, \ldots, Q_m$ (with query ranges)
  - Database objects (from "relevant" pages)

$$A = \begin{pmatrix} & Q_1 & \cdots & Q_m \\ Q_1 & 0 & & \\ \vdots & & \ddots & \\ Q_m & & & 0 \end{pmatrix}$$

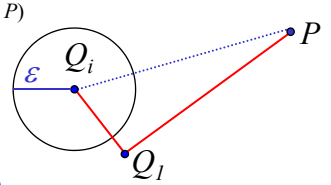
- Basic Procedure
  - Compute distances between $Q_1, \ldots, Q_m$
  - Compute *distance*(P, Q_i) only if it cannot be avoided by
    - a previously computed *distance*(P, Q_j), $j < i$,
    - plus the *triangle inequality*
- Important for complex distance functions
  - L_p in high-dimensional spaces
  - Quadratic Form Dist., Earth Mover's Distance, Edit Distance

Reduction of the CPU cost (2)

Avoiding expensive distance calculations

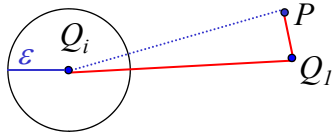
From triangle inequality: $dist(Q_i, P) + dist(Q_1, Q_i) \geq dist(Q_1, P)$

$$dist(Q_i, P) \geq \underbrace{dist(Q_1, P) - dist(Q_1, Q_i)}$$



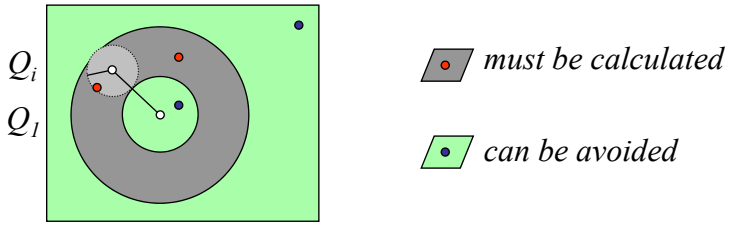
avoided if $\ldots > QueryRange(Q_i)$

$$dist(Q_i, P) \geq \overbrace{dist(Q_1, Q_i) - dist(Q_1, P)}$$



Reduction of the CPU cost (3)

- Area of avoidable and not avoidable distance computations



- Total CPU cost for m queries

$$C^m_{CPU} = \frac{(m-1)*m}{2} * time(distance\_calculation)$$
$$+ triangle\_trials * time(triangle\_inequality\_evaluation)$$
$$+ not\_avoided * time(distance\_calculation)$$

Reduction of the CPU cost – Evaluation

Astronomy DB

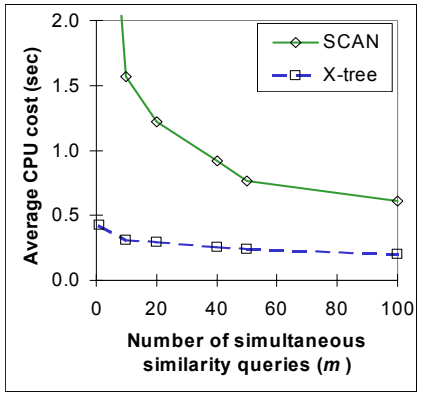
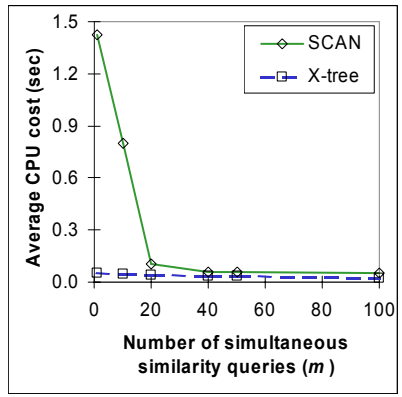
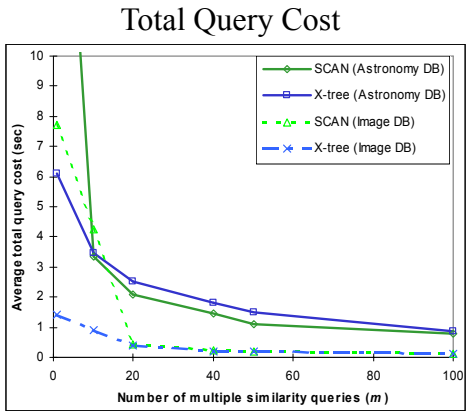


Image DB



Total Query Cost

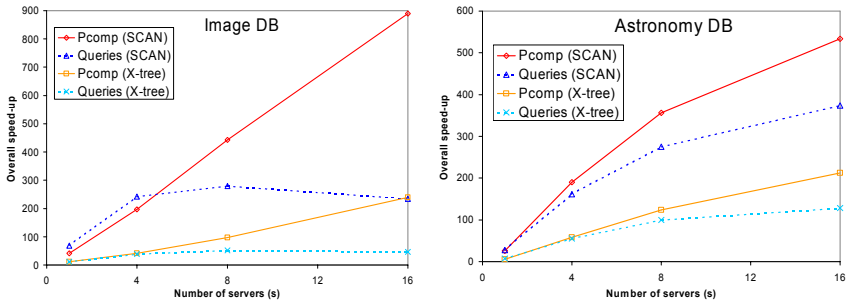


Further Improvements

- Parallelization: use s servers in a shared nothing environment to perform queries in parallel
  - Data is distributed among s servers
    - local parts of the data are s times smaller
    - small communication overhead
    - Local answer sets are s times smaller (on the average)
    - ➔ increase the number of queries m proportionally – However, the initialization overhead is $O(m^2)$ in the number of reference points m **!**
- Use pre-computed reference point along the principal axes instead of the distances between the queries to avoid distance calculations
  - Only a very small number of reference points necessary to avoid a huge amount of distance calculations

Evaluation of the Combined Effects

Overall Speed-up w.r.t. the number of servers s



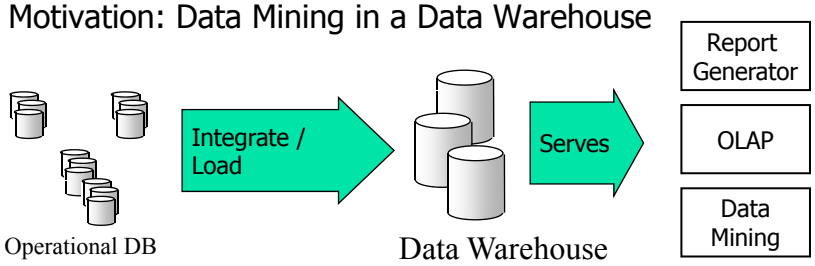
Number of simultaneous queries = number of servers * 100

Chapter 3: Clustering

- Introduction to clustering
- Expectation Maximization: a statistical approach
- Partitioning Methods
  - K-Means
  - K-Medoid
  - Choice of parameters: Initialization, Silhouette coefficient
- Density-based Methods: DBSCAN
- Hierarchical Methods
  - Density-based hierarchical clustering: OPTICS
  - Agglomerative Hierarchical Clustering: Single-Link + Variants
- Scaling Up Clustering Algorithms
  - BIRCH, Data Bubbles, Index-based Clustering, GRID Clustering
- Sets of Similarity Queries (Similarity Join)
- Advanced Clustering Topics
  - Incremental Clustering, Generalized DBSCAN, Outlier Detection
- Subspace Clustering

Incremental Clustering

- Motivation: Data Mining in a Data Warehouse

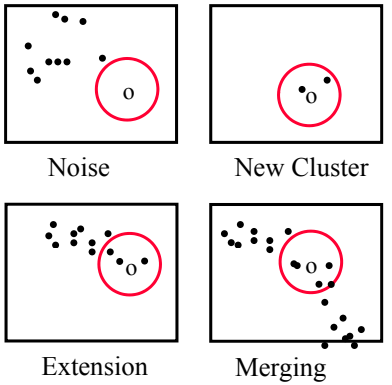


- Updates are collected and periodically inserted into the Data Warehouse
- all patterns derived from the Data Warehouse have to be updated

→ incremental Data Mining algorithms

Incremental DBSCAN

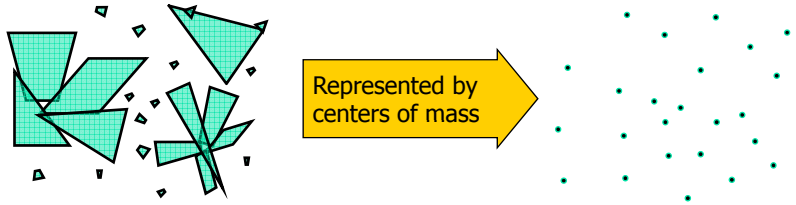
- In a density-based clustering, only a certain neighborhood of objects is affected when inserting/deleting some objects
  - In general not necessary to re-cluster the whole database
- Example: *Insertion*
  - Only check the affected neighborhood of an inserted object o
  - Only a few range queries necessary
  - Keep track of the current clustering via virtual cluster-ids



Generalized Density-Based Clustering

- Motivation: Clustering a set of *spatially extended objects*
  - Example: polygonal data, e.g., from a 2D GIS application
  - Preliminary approach: transform polygons to selected points

⇒ poor representation



- Take into account the area or other non-spatial attributes
- Use "natural" notions of connectedness (e.g. intersects or meets)

Density-Connected Sets

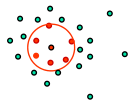
Basic idea of a density-based cluster :

"ε-neighborhood contains at least ***MinPts*** points***"***

"distance ≤ ε"

NPred(*o*,*p*)
reflexive, symmetric
for pairs of objects

Generalized Neighborhood
$N_{NPred}(o) = \{p \mid NPred(o, p)\}$



Generalization

"$| N_\varepsilon | \geq MinPts$"

MinWeight(N)
arbitrary predicate
for sets of objects

Generalized Minimum Cardinality
$MinWeight(N_{NPred}(o))$

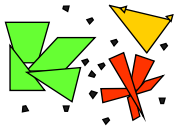
"NPred-neighborhood has at least ***MinWeight"***

➡ Algorithm follows the same schema as DBSCAN

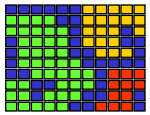
Examples of Density-Connected Sets

- Density-Based Clusters:
  - *NPred(o,p):* "distance(o,p) $\leq \varepsilon$"
  - *MinWeight(N):* card*(N)* ≥ *MinPts*

- Clustering Polygons:
  - *NPred(o,p):* "o intersects p"
  - *MinWeight(N):* sum of areas ≥ *MinArea*



- Simple Forms of Region Growing:
  - *NPred:* "pixels o,p adjacent and of same color"
  - *MinWeight(N):* TRUE

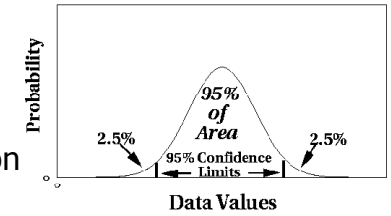


Outlier Detection

- Hawkins Definition of an Outlier
  - An object that deviates so much from the rest of the data set as to arouse suspicion that it was generated by a different mechanism
- Problem
  - Find top *n* outlier points
- Applications:
  - Credit card fraud detection
  - Telecom fraud detection
  - Customer segmentation
  - Medical analysis

Outlier Discovery: Statistical Approaches

- Assume an underlying model that generates data set (e.g. normal distribution)
- Discordance tests depending on
  - data distribution
  - distribution parameter (e.g., mean, variance)
  - number of expected outliers
- Limitations
  - most tests are for single attributes only
  - In many cases, data distribution may not be known

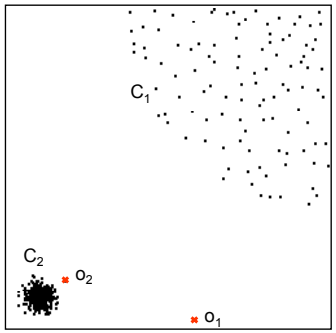


Outlier Discovery: Distance-Based Approach

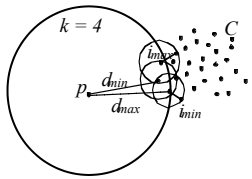
- Introduced to overcome the main limitations imposed by statistical methods
  - Multi-dimensional analysis without knowing data distribution.
- Distance-based outlier: A *DB(pct, d_min)*-outlier is an object *o* in a dataset *DB* such that at least a fraction *pct* of the objects in *DB* lies at a distance greater than *d_min* from *o*
- Algorithms for mining distance-based outliers
  - Index-based algorithm
  - Nested-loop algorithm
  - Cell-based algorithm

Outlier Discovery: Local Outlier Factors

- o_1: clear *DB*(*pct*, *dmin*)-outlier for
  - *Pct:* greater than 99%
  - *dmin*: very large
- o_2: Not well captured
  - either: not an *DB*(*pct*, *dmin*)-outlier
  - or: many points in C_1 are outliers, too!



- Local outlier factor of points *p:*
  - Basic Idea: Look at the *k*-nearest neighbor distance of *p* relative to the *k*-nearest neighbor distances of these *k* neighbors of *p*

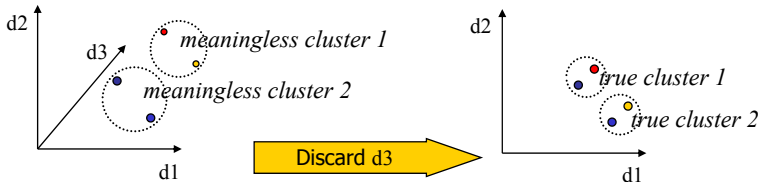


Chapter 3: Clustering

- Introduction to clustering
- Expectation Maximization: a statistical approach
- Partitioning Methods
  - K-Means
  - K-Medoid
  - Choice of parameters: Initialization, Silhouette coefficient
- Density-based Methods: DBSCAN
- Hierarchical Methods
  - Density-based hierarchical clustering: OPTICS
  - Agglomerative Hierarchical Clustering: Single-Link + Variants
- Scaling Up Clustering Algorithms
  - BIRCH, Data Bubbles, Index-based Clustering, GRID Clustering
- Sets of Similarity Queries (Similarity Join)
- Advanced Clustering Topics
  - Incremental Clustering, Generalized DBSCAN, Outlier Detection
- Subspace Clustering

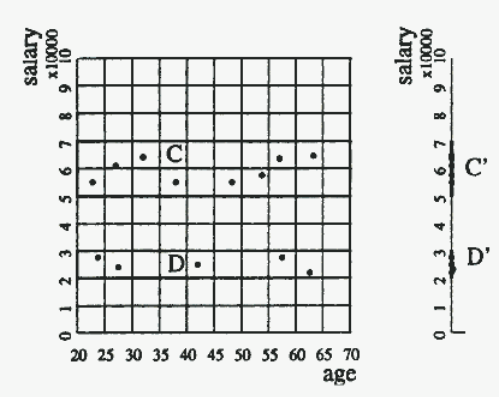
Subspace Clustering

- Problem: selection of a relevant subset of dimensions for a given clustering task.
  - Irrelevant dimensions can obscure an otherwise good result



  - Too many dimensions reduce the interpretability of a result

Subspace Clustering – Motivation



Imagine a cluster to be characterized by a 1D subspace, e.g., „salary"

Subspace Clustering: Major Approaches

- *Subspace clustering:* determine subspaces of the original space that allow a better clustering than the original points
- *Fascicles F(k, t):* subsets of records that have *k* attributes such that
  - the values in each of the *k* attributes vary by at most *t*
    - range *t* for numeric attributes
    - number of different values *t* for categorical attributes
  - The number of records in the fascicle exceeds some threshold
  - *k* is maximal

Subspace Clustering: CLIQUE

CLIQUE [Agrawal, Gehrke, Gunopulos & Raghavan 1998]

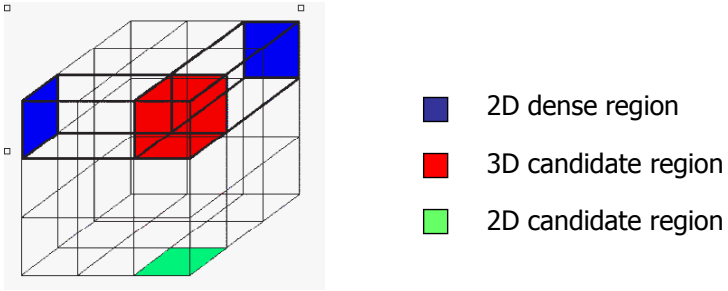
1. Identification of subspaces with clusters
2. Identification of clusters
3. Generation of cluster descriptions
4. Cluster: „dense region" in the data space

- Density threshold τ
  - Region *r* is dense if *r* contains more than τ points
- Grid based approach
  - Each dimension is partitioned into ξ intervals
  - Cluster is union of connected dense regions

Subspace Clustering

- Identification of subspaces with clusters
- Task: Discover dense base regions
- Naive approach
  - Compute histograms for all subsets of dimensions
  - Inefficient for high dimensional data, i.e. $O(2^d)$ for *d* dimensions
- Greedy algorithm (Bottom-Up)
  - Start with empty set of dimensions
  - Iteratively include a new dimension
- Justification of algorithm: monotonicity of density
  - If a region *r* is dense in a *k*-dimensional space *s*, then any projection of *r* into a $(k-1)$-dimensional subspace of *s* is dense.

Subspace Clustering – Example



- Runtime complexity of greedy algorithm: $O(\xi^k + n \cdot k)$
  - for *n* data objects and *k* being the highest number of dimensions of any dense region
- Heuristics to reduce the number of candidate regions
  - Principle of „minimum description length"

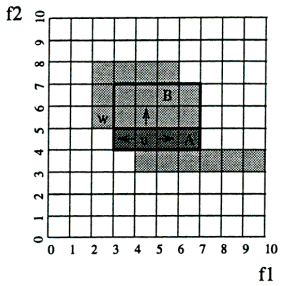
Subspace Clustering – Identifying Clusters

- Task: find maximal sets of connected dense regions
- Given all dense regions within a single *k*-dimensional subspace
- Search space is a graph
  - Vertices are dense regions
  - Edges are boundaries or dimensions the regions have in common
- Depth first search applicable
- Runtime complexity:
  - Assume dense regions to reside in main memory (e.g., in a hash tree)
  - For each dense region, $2k$ neighbors have to be checked

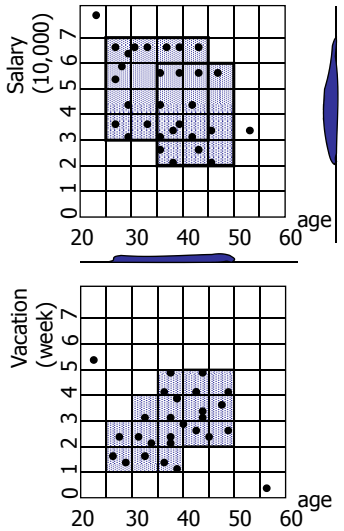
  → $2kn$ accesses to the data structure

Subspace Clustering – Generation of Cluster Descriptions

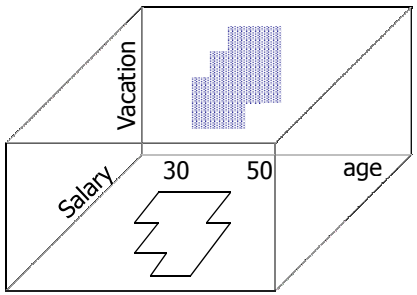
- Problem
  - Given a cluster, i.e., a set of connected dense regions
  - Desired result: set of hyperrectangles that optimally cover the cluster region
- Standard methods?
  - Problem is NP hard
  - Too inefficient for high dimensions *d*
- Heuristic method
  - Cover the cluster by maximal regions
  - Remove redundant regions



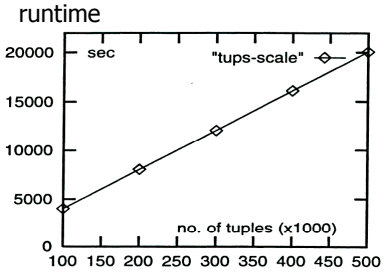
Subspace Clustering – Example

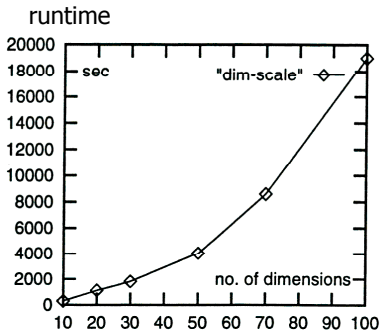


$\tau = 3$



Subspace Clustering – Experiments





- Cf. Runtime complexity of CLIQUE: linear in *n*, superlinear in *d*

Subspace Clustering – Discussion (CLIQUE)

+ automatic detection of subspaces with clusters
+ automatic detection of clusters at all
+ no assumptions about the distribution of data
+ insensitivity to the order of the data
+ good scalability wrt. the number *n* of data objects

– accuracy of result depends on parameter ξ
– heuristics needed that restricts search on all subsets of dimensions

→ Possibly, not all subspaces with clusters are found

Recent Developments in Subspace Clustering

- ENCLUS: based on CLIQUE → Entropy as a density criterion
  - Advantage: Correlation of dimensions can be evaluated
  - Limitations: Entropy is computationally complex
- MAFIA: based on CLIQUE → adaptive grid of variable size in different dimensions
  - Advantage: can be parallelized (pMAFIA)
  - Limitations: only axis-parallel subspace
- RIS (Ranking Interesting Subspaces): no Clustering, searches and evaluates subspaces which can then be used by „traditional" clustering algorithms
  - Calculates for any database object o those subspaces for which o is still a core object
  - Integrates the dimensionality and, possibly, closeness to the edges of the data space into the evaluation
  - Prunes sub- and superspaces which are redundant

Recent Developments in Subspace Clustering

- PROCLUS → based on CLARANS
  - For each of k random medoids, calculate its best dimensions
    - correlation with its neighboring points in these dimensions
  - assign points to their cluster
  - If clustering is improved, iterate; else the algorithm terminates
  - Advantage: linear in both the number of objects and dimensions
  - Limitations: may converge only locally minimum, fixed number of medoids, only convex clusters, axis-parallel projections only
- Monte Carlo Projective Clustering
  - Approximates optimal projective cluster
  - For a set of random points, determine those dimensions where the distance to a pivot *p* is less than a threshold ω in this dimension
  - The cluster is the set of points which are in a 2ω hyperbox around *p*
  - Iterate, keeping clusters of more than *MinPts* elements
  - Return only the best subspace cluster
  - Advantage: good results for highdimensional spaces
  - Limitations: only axis-parallel clusters

Recent Developments in Subspace Clustering

- δ-Clusters: data coherence: similar, not necessarily identical values
  - Uses a matrix of objects/attributes
  - Actions: add/remove object/attribute to/from cluster
  - In each iteration, determine possible actions for each cluster
    - perform action with best improvement over all clusters
  - Different heuristics used for the ordering of actions
  - Advantage: general model for coherence search in data
  - Limitations: computationally complex; only additive or multiplicative coherence (i.e. no negative or other correlations)
- 4C: based on DBSCAN and PCA to find coherences in the data
  - Density is extended to include correlation using local covariance matrices
  - Algorithm similar to DBSCAN
  - Advantage: integrates coherence and density
  - Limitations: dimensionality of the correlation has to be specified, not hierarchical, only linear dependencies

References: Subspace Clustering

- ENCLUS: C. Cheng, A. Fu, Y. Zhang: Entropy-Based Subspace Clustering for Numerical Data, Proc. ACM SIGKDD 1999, 84-93
- pMAFIA: H. Nagesh, S. Goil, A. Choudhary: A Scalable Parallel Subspace Clustering Algorithm for Massive Data Sets. ICPP 2000: 477-
- RIS: K. Kailing, H.-P. Kriegel, P. Kröger, S. Wanka: Ranking Interesting Subspaces for Clustering High Dimensional Data, PKDD 2003, 241-252
- PROCLUS: C. Aggarwal, J. Wolf, P. Yu, C. Procopiuc, J. Park: Fast algorithms for projected clustering, Proc. ACM SIGMOD 1999, 61-72
- Monte Carlo: C. Procopiuc, M. Jones, P. Agarwal, T. Murali: A Monte Carlo algorithm for fast projective clustering, Proc. ACM SIGMOD 2002, 418-427
- δ-Clusters: J. Yang, W. Wang, H. Wang, P. Yu: Delta-Clusters: Capturing Subspace Correlation in a Large Data Set, Proc. ICDE 2002, 517-528
- 4C: C. Böhm, K. Kailing, P. Kröger, A. Zimek, Computing Clusters of Correlation Connected objects, Proc. ACM SIGMOD 2004, 455-466

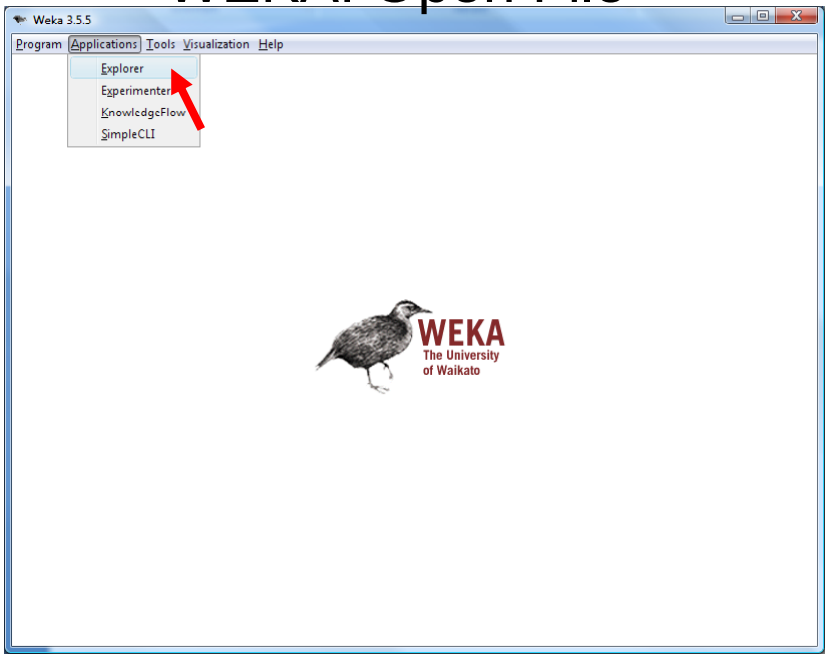
References: Clustering

- Ankerst M., Breunig M., Kriegel H.-P., Sander J. 1999, „OPTICS: Ordering Points To Identify the Clustering Structure", *Proc. ACM SIGMOD Int. Conf. on Management of Data (SIGMOD'99)*. ACM Press, New York, NY, pp. 49—60.
- Braunmueller B., Ester M., Kriegel H.-P., and Sander J., "Efficiently Supporting Multiple Similarity Queries for Mining in Metric Databases", Proc. 16th Int. Conf. on Data Engineering (*ICDE'00*). IEEE Computer Society Press, Los Alamitos, CA, pp. 256-267, Feb 2000.
- Breunig M., Kriegel H.-P., Kröger P., and Sander J., "Data Bubbles: Quality Preserving Performance Boosting for Hierarchical Clustering", In Proc. ACM SIGMOD Int. Conf. on Management of Data (*SIGMOD'01*), ACM Press, New York, NY, May 2001.
- Ester M., Kriegel H.-P., Sander J., Xu X. 1996, „A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise", *Proc. 2nd Int. Conf. on Knowledge Discovery and Data Mining (KDD'96)*. AAAI Press, Menlo Park, CA, pp. 226—231.
- Ester M., Kriegel H.-P., Sander J., Wimmer M. Xu X. 1998, „Incremental Clustering for Mining in a Data Warehousing Environment", *Proc. 24th Int. Conf. on Very Large Databases (VLDB'98)*, Morgan Kaufmann Publishers, San Francisco, CA, pp. 323—333.
- Fayyad U., Reina C., Bradley P. S. 1998, „Initialization of Iterative Refinement Clustering Algorithms", *Proc. 4th Int. Conf. on Knowledge Discovery and Data Mining (KDD'98)*, AAAI Press, Menlo Park, CA, pp. 194—198.

References: Clustering

- Forgy E. W. 1965, „Cluster analysis of multivariate data: Efficiency vs. interpretability of classification (abstract)", *Biometrics*, Vol. 21, pp. 768—769.
- Jain A. K., Dubes R. C., 1988, *Algorithms for Clustering Data*, Prentice-Hall.
- Kaufman L., Rousseeuw P. J. 1990, *Finding Groups in Data: An Introduction to Cluster Analysis*, John Wiley & Sons.
- MacQueen, J. 1967, „Some Methods for Classification and Analysis of Multivariate Observations", *5th Berkeley Symp. Math. Statist. Prob.*, Volume 1, pp. 281—297.
- Murtagh F. 1983, „A Survey of Recent Advances in Hierarchical Clustering Algorithms", *The Computer Journal*, Vol 26. No. 4, pp. 354—359.
- Ng R. T., Han J., 1994, „Efficient and Effective Clustering Methods for Spatial Data Mining", *Proc. 20th Int. Conf. on Very Large Data Bases (VLDB'94)*, Morgan Kaufmann Publishers, San Francisco, California, pp. 144—155.
- Rohlf F. J. 1973, „Hierarchical clustering using the minimum spanning tree", *The Computer Journal, Vol* 16, No. 1, pp. 93—95.
- Sander J., Ester M., Kriegel H.-P., Xu X. 1998, „Density-Based Clustering in Spatial Databases: The Algorithm GDBSCAN and its Applications", *Data Mining and Knowledge Discovery, An International Journal*, Kluwer Academic Publishers, Norwell, MA, Vol. 2, No. 2., 1998, pp. 169—194.
- Zhang T., Ramakrishnan R., Linvy M. 1996, „BIRCH: An Efficient Data Clustering Method for Very Large Databases", *Proc. ACM SIGMOD Int. Conf. on Management of Data (SIGMOD'96)*, ACM Press, New York, pp. 103—114.

WEKA: Open File



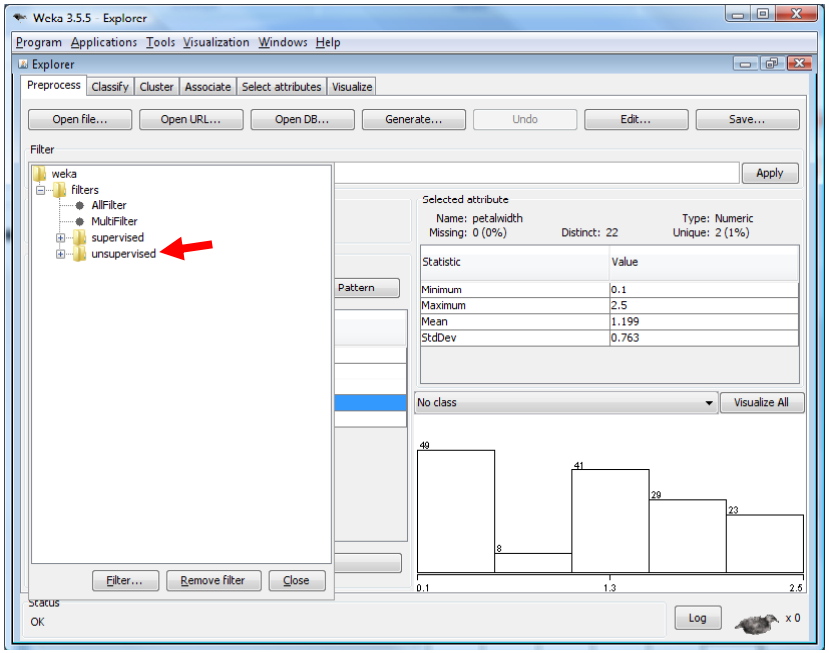
Exercise 4) WEKA

a) How large is each bin for each attribute?

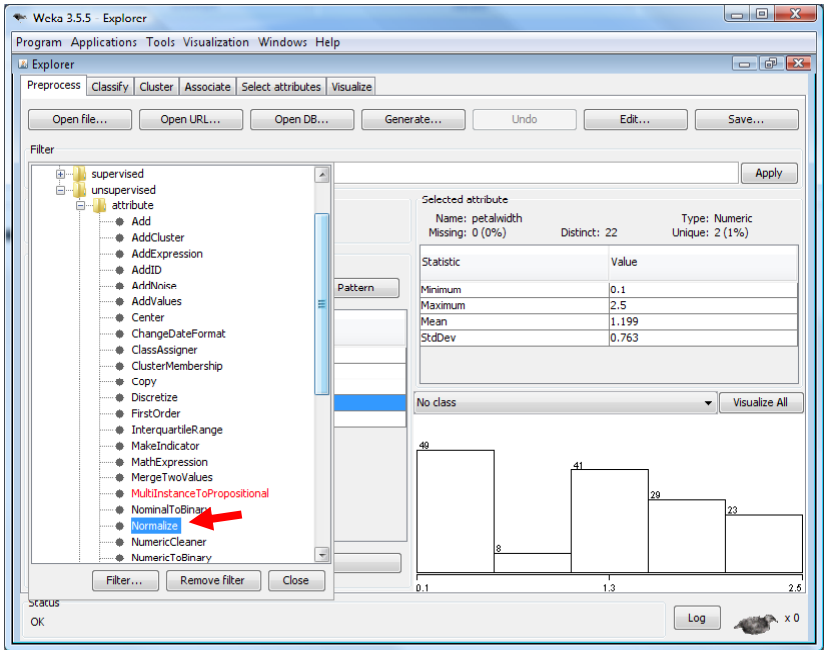
b) What is the mean value and standard deviation for each attribute?

c) Normalize the data using the attribute filtering function of WEKA.

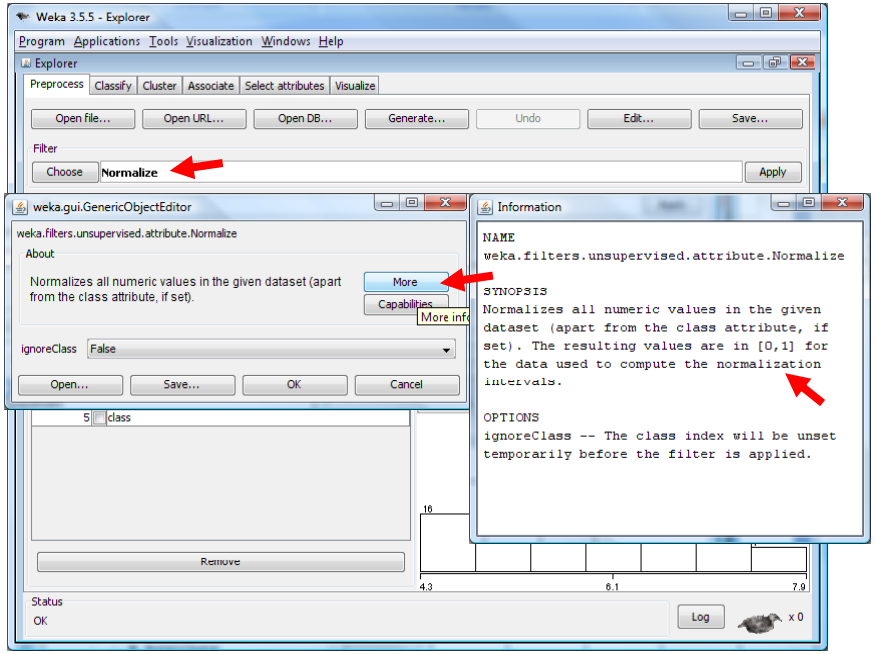
WEKA: Statistics



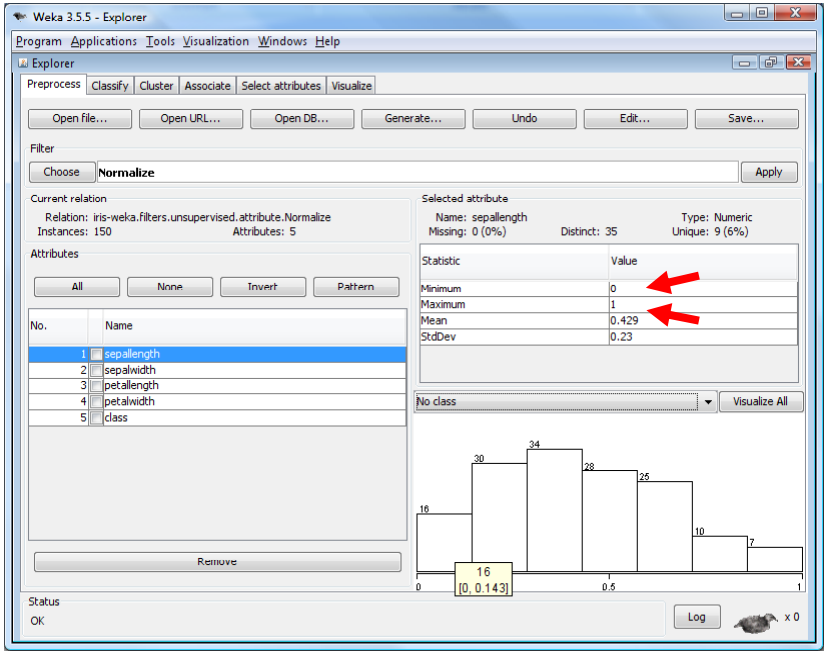
WEKA: Statistics



WEKA: Statistics



WEKA: Statistics

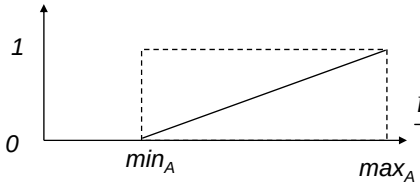


Data Transformation: min-max Normalization

- min=-max normalization with: min=0 and max=1

$$v' = \frac{v - min_A}{max_A - min_A}$$

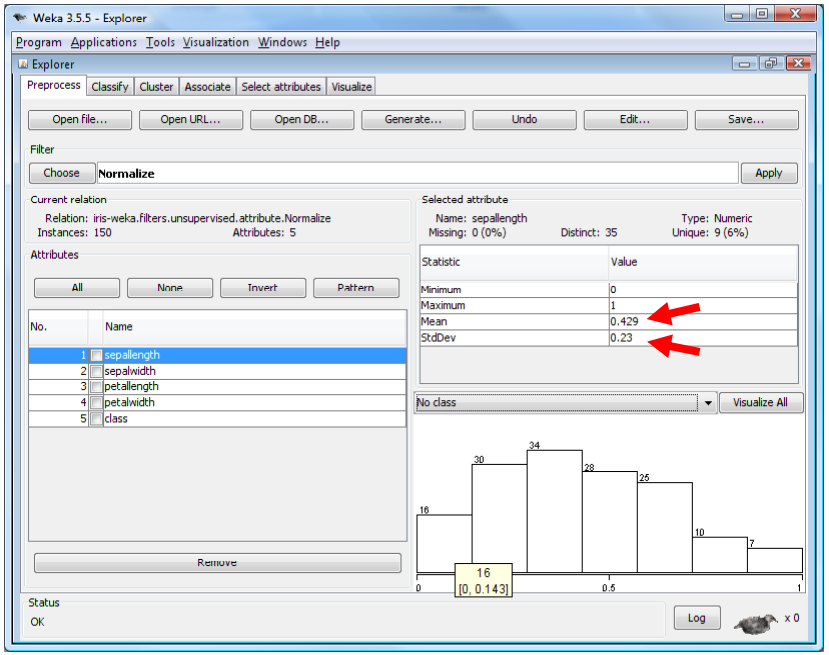
- transforms data linearly to a new range
  - range outliers may be detected afterwards as well



slope is:

$$\frac{new\_max_A - new\_min_A}{max_A - min_A}$$

WEKA: Statistics



Exercise 4) WEKA

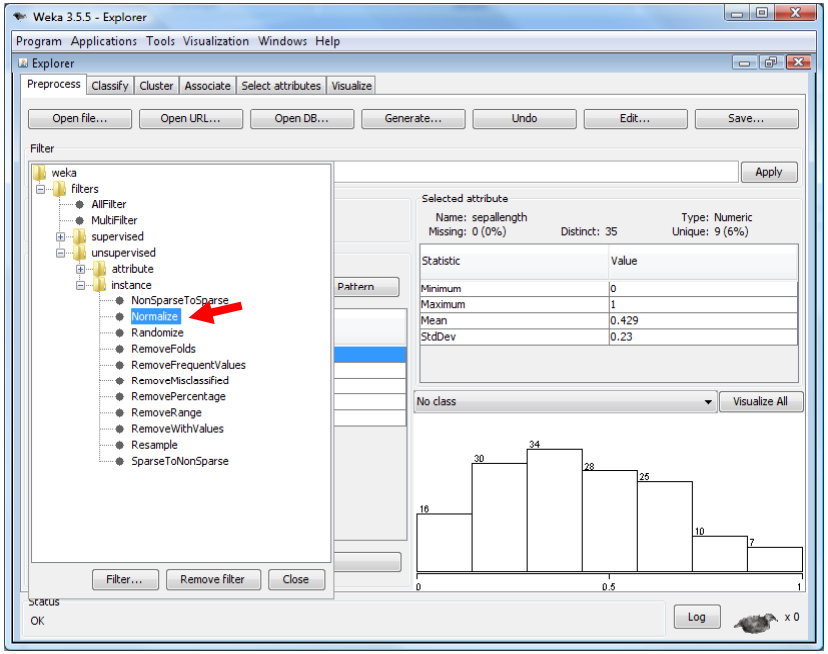
a) How large is each bin for each attribute?

b) What is the mean value and standard deviation for each attribute?

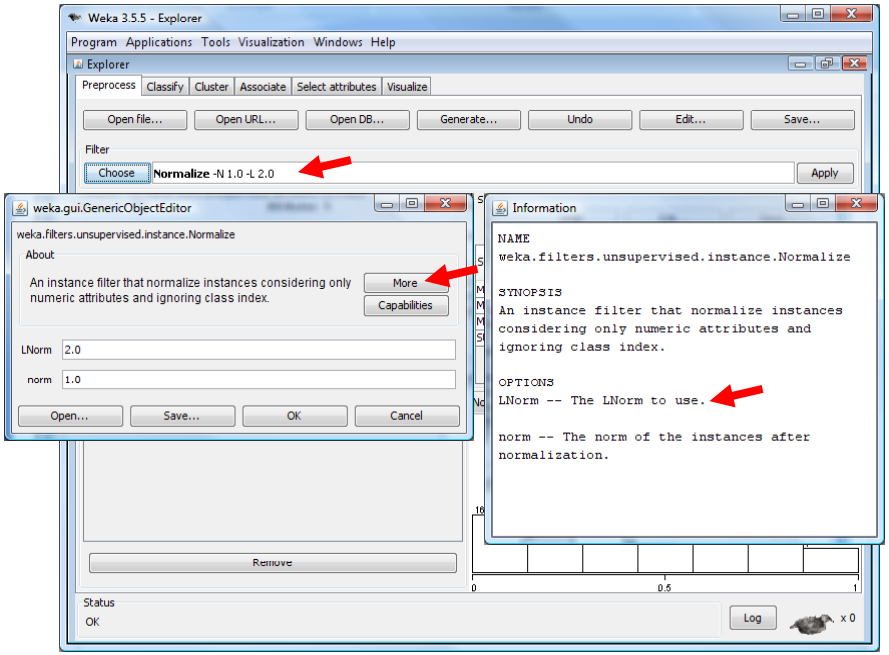
c) Normalize the data using the attribute filtering function of WEKA.

d) Normalize the data using the instance filtering function of WEKA.
What is the normalization method?

WEKA: Statistics



WEKA: Statistics



Definition: Norm

- Examples for norms in $\mathbb{R}^d$ for $x = (x_1,\ldots,x_d)$ are:
  - L_p-Norm:

$$p \geq 1 \qquad \|x\|_p = \left(\sum_{1 \leq i \leq d} |x_i|^p \right)^{1/p} \qquad dist_p(u,v) = \left(\sum_{1 \leq i \leq d} |u_i - v_i|^p \right)^{1/p}$$

  - p = 1, Manhattan norm:

$$\|x\|_1 = \sum_{1 \leq i \leq d} |x_i| \qquad dist_1(u,v) = \sum_{1 \leq i \leq d} |u_i - v_i|$$

  - p = 2, eucledian Norm:

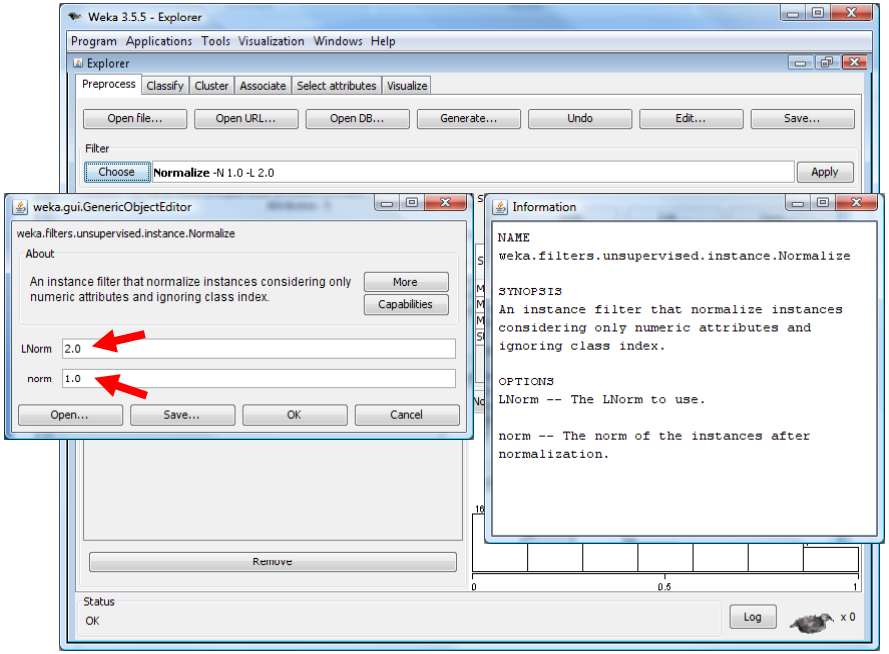
$$\|x\|_2 = \sqrt{\sum_{1 \leq i \leq d} |x_i|^2} = \sqrt{x^T \cdot x} \qquad dist_2(u,v) = \sqrt{\sum_{1 \leq i \leq d} |u_i - v_i|^2}$$

  - $p \to \infty$, Maximumsnorm:

$$\|x\|_\infty = \max \{ |x_i|, 1 \leq i \leq d \}$$

$$dist_\infty(u,v) = \max \{ |u_i - v_i|, 1 \leq i \leq d \}$$

WEKA: Statistics



Text distances

- “Unlike classification or prediction, which analyzes data objects with class labels, clustering analyzes data objects without consulting a known class label. The class labels are not in the data because they are not known.”
- “Classification can be used for prediction of class labels of data objects. However, in many applications, prediction of missing or not known data values rather than class labels is performed to fit data objects into a schema.”
- “Sun Salutation, a ritual performed in the early morning, combines seven different postures. The sun, the life generator, is invoked by this Yogic exercise, an easy way to keep fit.”

- Stopwords = { a, an, are, be, because, by, can, for, however, in, into, is, keep, many, not, of, or, rather, than, the, they, this, to, unlike, used, way, which, with, without }

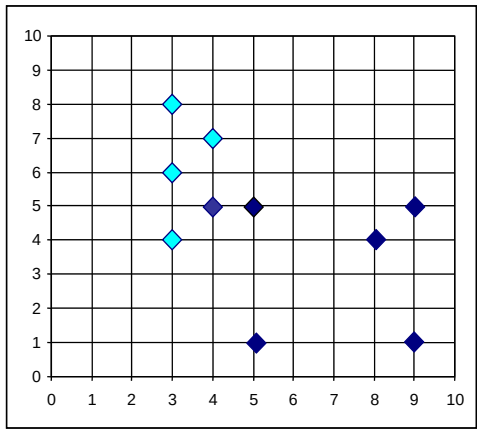
Attributes	Text 1	Text 2	Text 3	Log(1)	Log(2)	Log(3)	Len(1)	Len(2)	Log(3)
analyzes	2			0.301	0	0	0.0906	0	0
applications		1		0	0	0	0	0	0
class	3	2		0.4771	0.301	0	0.2276	0.0906	0
classification	1	1		0	0	0	0	0	0
clustering	1			0	0	0	0	0	0
combines			1	0	0	0	0	0	0
consulting	1			0	0	0	0	0	0
data	3	3		0.4771	0.4771	0	0.2276	0.2276	0
different			1	0	0	0	0	0	0
early			1	0	0	0	0	0	0
easy			1	0	0	0	0	0	0
exercise			1	0	0	0	0	0	0
fit		1	1	0	0	0	0	0	0
generator			1	0	0	0	0	0	0
invoked			1	0	0	0	0	0	0
known	2	1		0.301	0	0	0.0906	0	0
labels	3	2		0.4771	0.301	0	0.2276	0.0906	0
life			1	0	0	0	0	0	0
missing		1		0	0	0	0	0	0
morning			1	0	0	0	0	0	0
objects	2	2		0.301	0.301	0	0.0906	0.0906	0
performed		1	1	0	0	0	0	0	0
postures			1	0	0	0	0	0	0
prediction	1	2		0	0.301	0	0	0.0906	0
ritual			1	0	0	0	0	0	0
salutation			1	0	0	0	0	0	0
schema		1		0	0	0	0	0	0
seven			1	0	0	0	0	0	0
sun			2	0	0	0.301	0	0	0
values		1		0	0	0	0	0	0
Yogic			1	0	0	0	**0.9771**	**0.7682**	**0.301**

Attributes	Log(1)	Log(2)	Log(3)	Len(1)	Len(2)	Log(3)
analyzes	0.301	0	0	0.0906	0	0
applications	0	0	0	0	0	0
class	0.4771	0.301	0	0.2276	0.0906	0
classification	0	0	0	0	0	0
clustering	0	0	0	0	0	0
combines	0	0	0	0	0	0
consulting	0	0	0	0	0	0
data	0.4771	0.4771	0	0.2276	0.2276	0
different	0	0	0	0	0	0
early	0	0	0	0	0	0
easy	0	0	0	0	0	0
exercise	0	0	0	0	0	0
fit	0	0	0	0	0	0
generator	0	0	0	0	0	0
invoked	0	0	0	0	0	0
known	0.301	0	0	0.0906	0	0
labels	0.4771	0.301	0	0.2276	0.0906	0
life	0	0	0	0	0	0
missing	0	0	0	0	0	0
morning	0	0	0	0	0	0
objects	0.301	0.301	0	0.0906	0.0906	0
performed	0	0	0	0	0	0
postures	0	0	0	0	0	0
prediction	0	0.301	0	0	0.0906	0
ritual	0	0	0	0	0	0
salutation	0	0	0	0	0	0
schema	0	0	0	0	0	0
seven	0	0	0	0	0	0
sun	0	0	0.301	0	0	0.0906
values	0	0	0	0	0	0
Yogic	0	0	0	0	0	0
				0.9771	**0.7682**	**0.301**

$$dist(x, y) = 1 - \frac{\langle x, y \rangle}{|x| \cdot |y|}$$

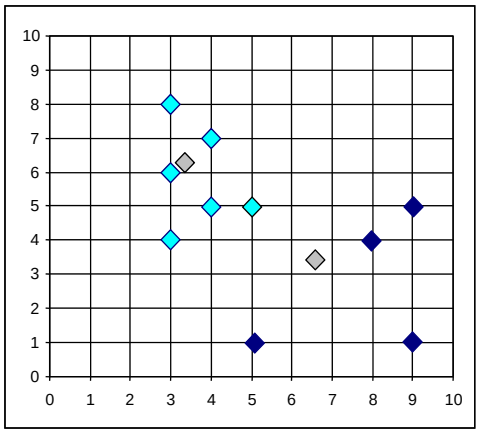
Initial partition:

Centroids:

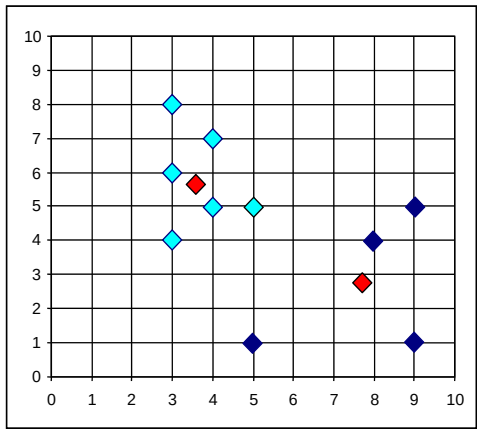


partition after first step:

Centroids:



partition after second step: (final)



Exercise 3-4) WEKA

Remove the class attribute using the preprocessing dialog.

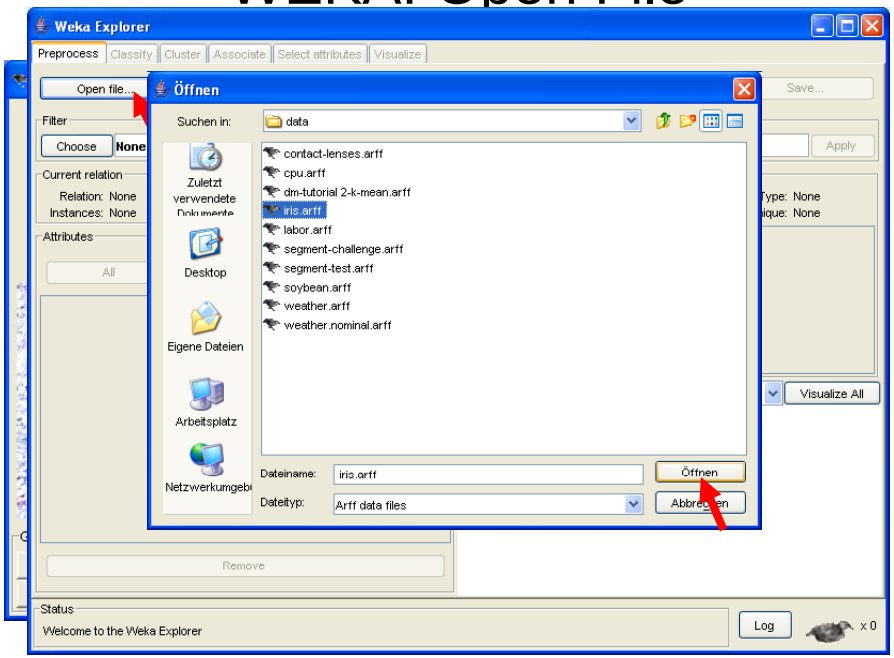
Go to the clustering dialog.

Cluster the iris dataset using the k-Means Clustering algorithm with k=5.

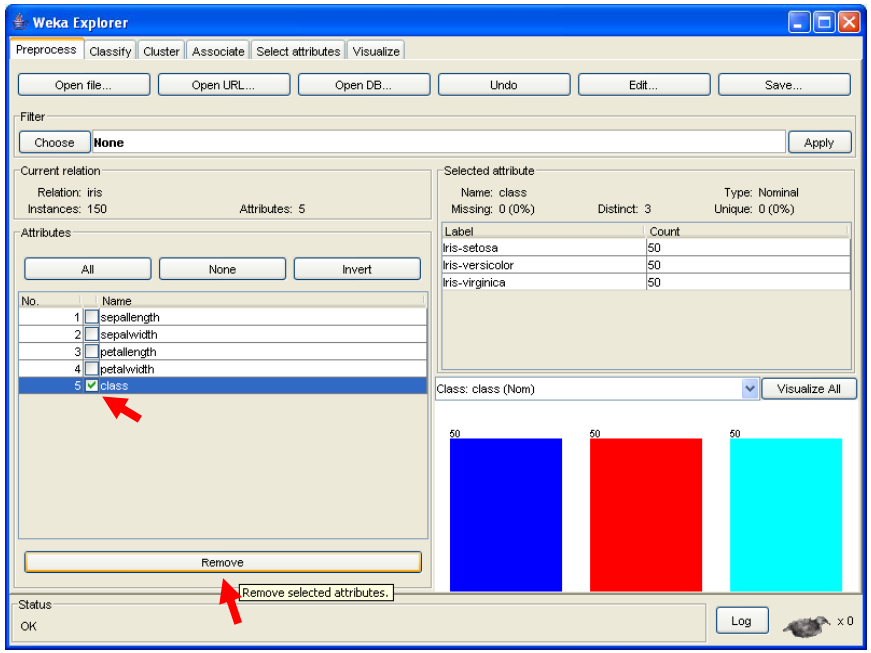
Hand in the result given by WEKA (Cluster mean and standard deviation).

Visualize the cluster mean values and standard deviation for
- *sepallength* versus *sepalwidth*
- *petallength* versus *petalwidth*

WEKA: Open File



WEKA: Statistics



Exercise 3-4) WEKA

Remove the class attribute using the preprocessing dialog.

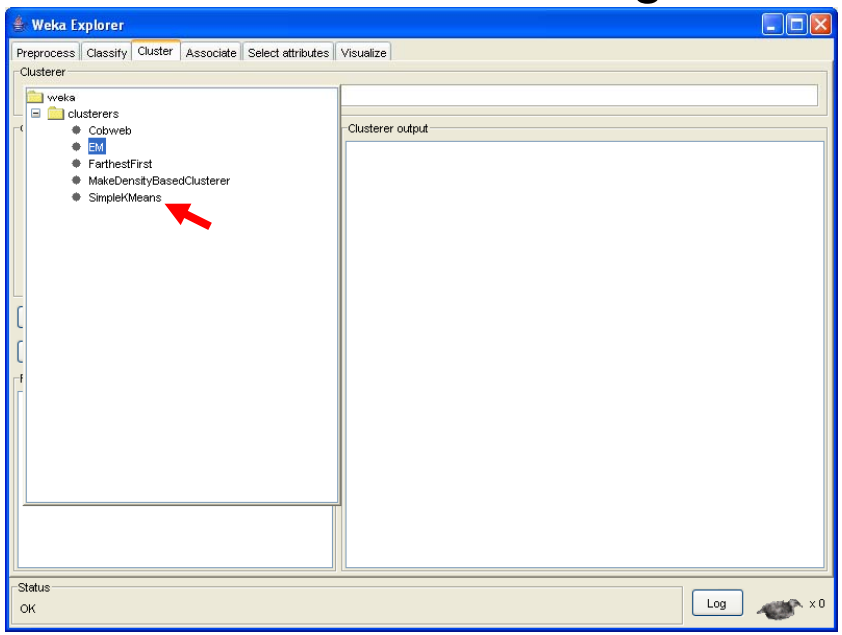
Go to the clustering dialog.

Cluster the iris dataset using the k-Means Clustering algorithm with k=5.

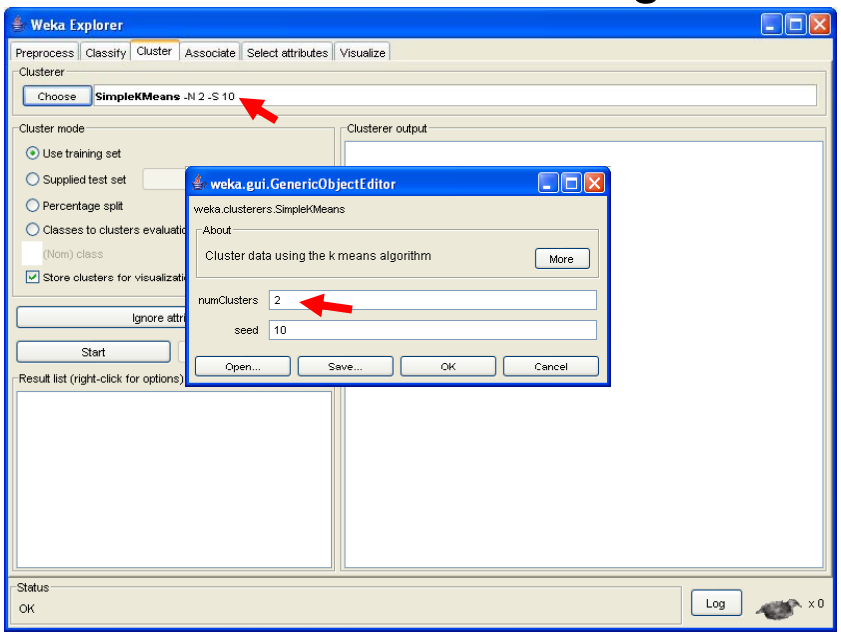
Hand in the result given by WEKA (Cluster mean and standard deviation).

Visualize the cluster mean values and standard deviation for
- *sepallength* versus *sepalwidth*
- *petallength* versus *petalwidth*

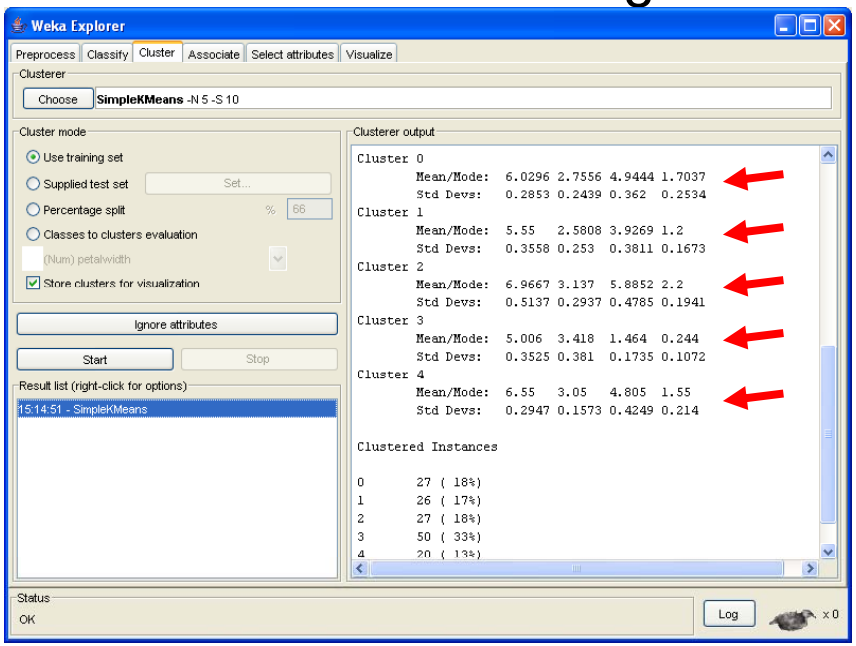
WEKA: Clustering



WEKA: Clustering



WEKA: Clustering



Exercise 3-4) WEKA

How large is each bin for each attribute?

What is the mean value and standard deviation for each attribute?

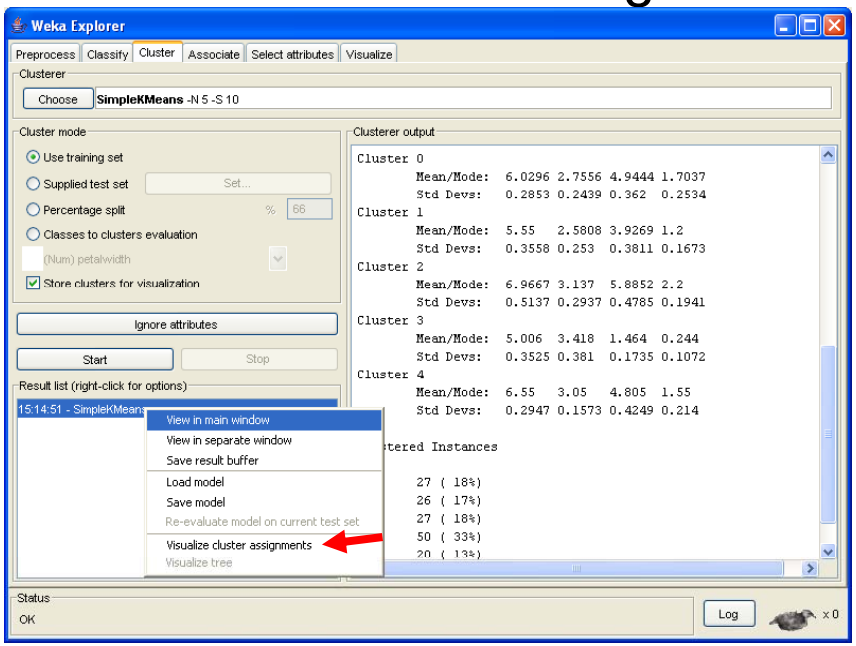
Remove the class attribute using the preprocessing dialog.
Go to the clustering dialog.

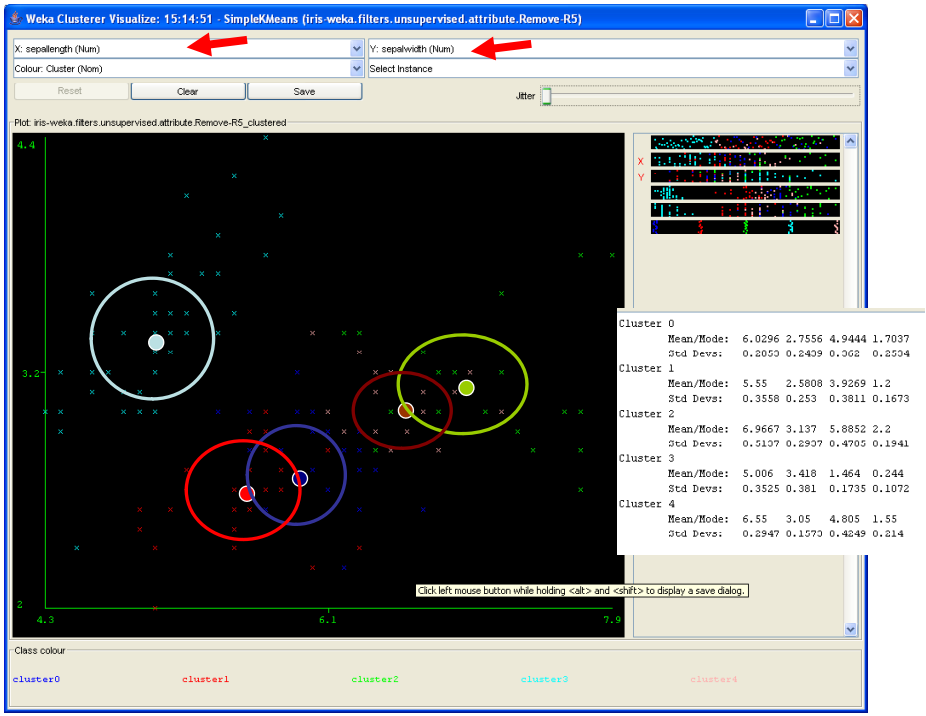
Cluster the iris dataset using the k-Means Clustering algorithm with k=5.

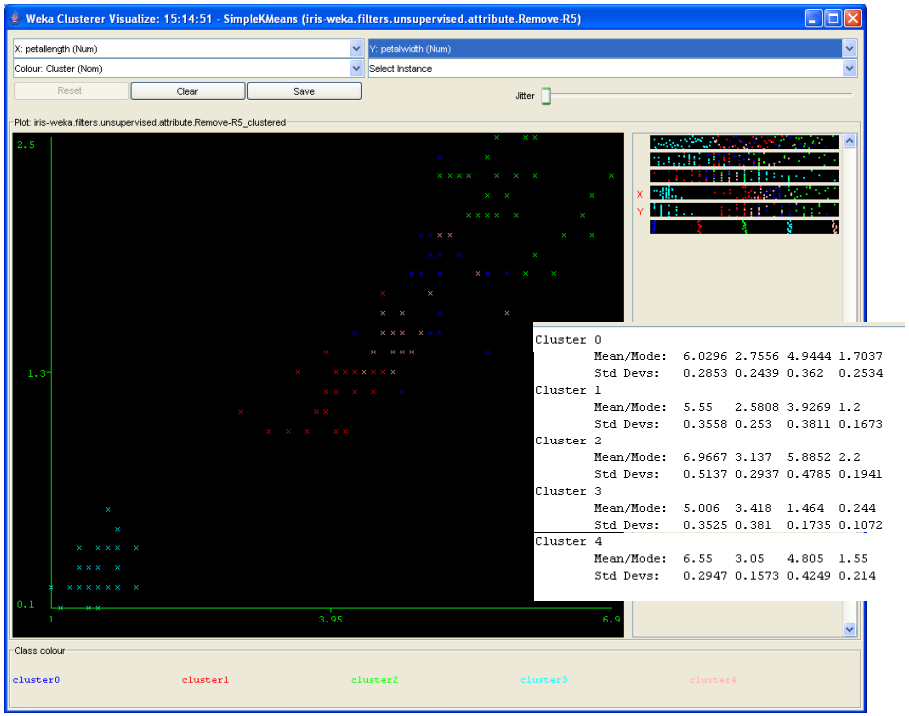
Hand in the result given by WEKA (Cluster mean and standard deviation).

Visualize the cluster mean values and standard deviation for
- *sepallength* versus *sepalwidth*
- *petallength* versus *petalwidth*

WEKA: Clustering







Exercise 3.4 c) WEKA

- Create an “arff”-file containing the datapoints from exercise 3.

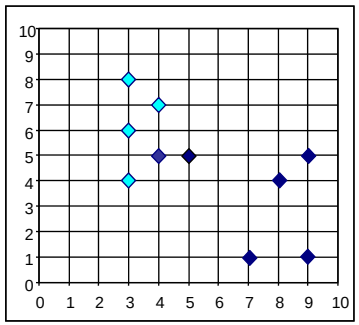
- Cluster the data file using WEKA with k=2 and k=3 clusters.
- Hand in the result given by WEKA (Cluster mean and standard deviation).

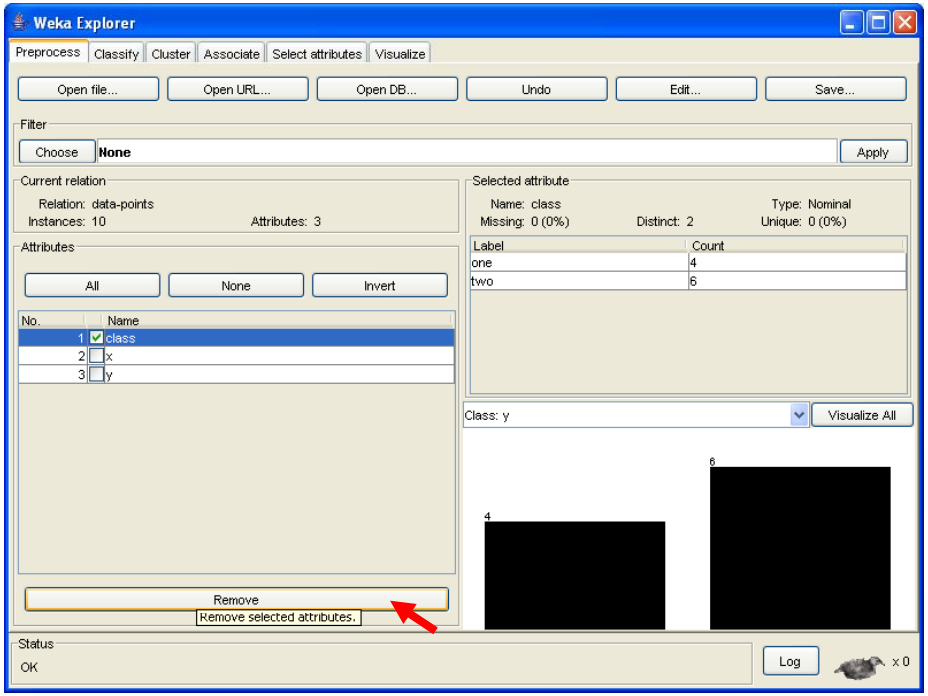
Data File

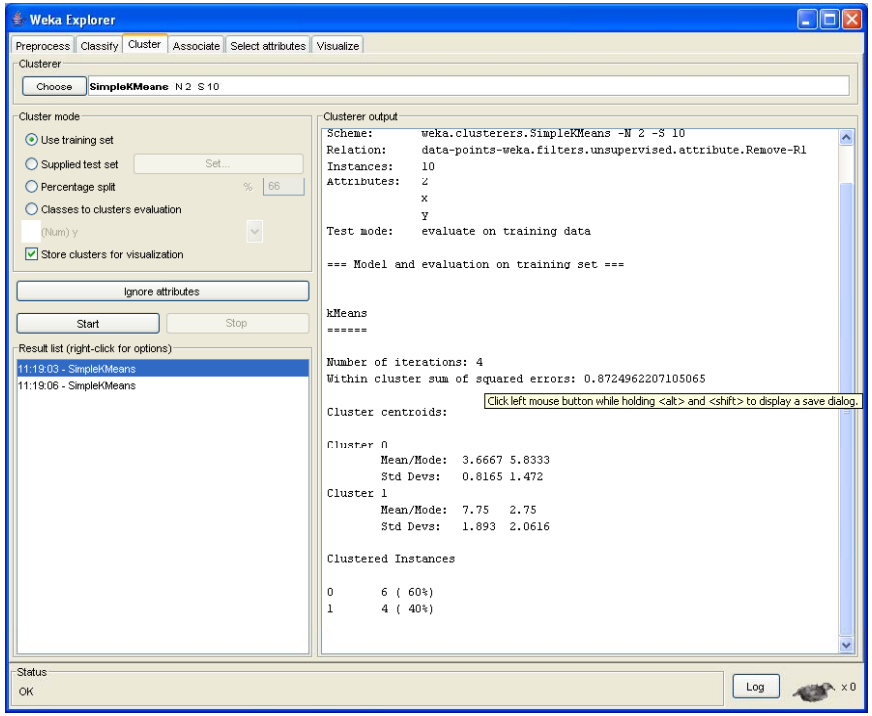
```
% Data points for data mining tutorial
%           Exercise: Cluster the data file with k-mean
%                     clustering using k=2 and k=3 clusters.

@relation data-points

@attribute class       {one, two}
@attribute x           numeric
@attribute y           numeric

@data
% 10 instances
one,3,8
one,3,6
one,3,4
one,4,7
two,4,5
two,5,1
two,5,5
two,8,4
two,9,1
two,9,5
```

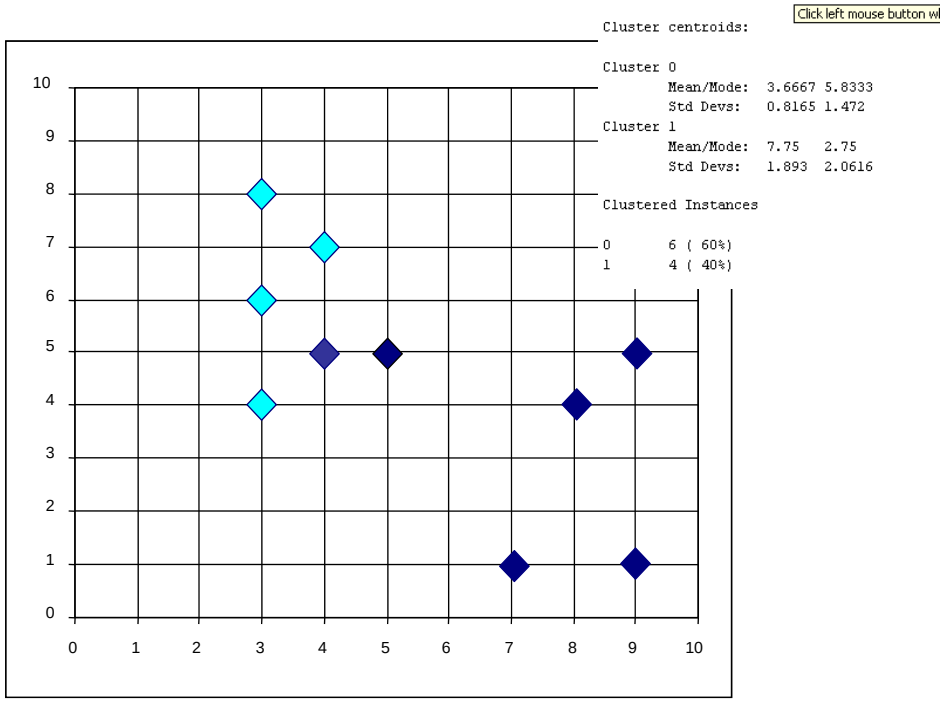
nominal attribute

numeric attribute

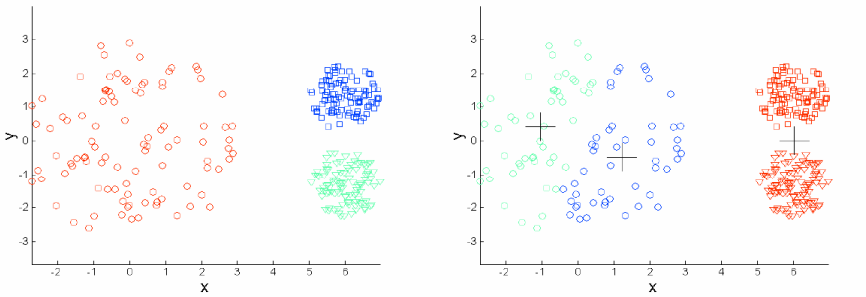








Limitations of K-means: Differing Density



Original Points

K-means (3 Clusters)

© for the original version:
- Jörg Sander and Martin Ester
- Jiawei Han and Micheline Kamber

Data Mining Algorithms

Lecture Course with Tutorials
Summer 2005

Chapter 4: Classification

Chapter 4: Classification

Introduction: Classification Problem

- Given
  - a d-dimensional data space D with attributes a_i, $i = 1, \ldots, d$
  - a set $C = \{c_1, \ldots, c_k\}$ of k distinct class labels c_j, $j = 1, \ldots, k$
  - a set $O \subseteq D$ of objects, $o = (o_1, \ldots, o_d)$, with known class labels from C
- Searched
  - class label for objects from $D - O$, i.e. for objects with unknown class
  - a classifier $K: D \rightarrow C$
- Demarcation from clustering
  - Classification: classes are known in advance (a priori)
  - Clustering: classes are determined
- Related problem: prediction
  - Determine the value of a numerical attribute
  - Method: e.g., regression analysis

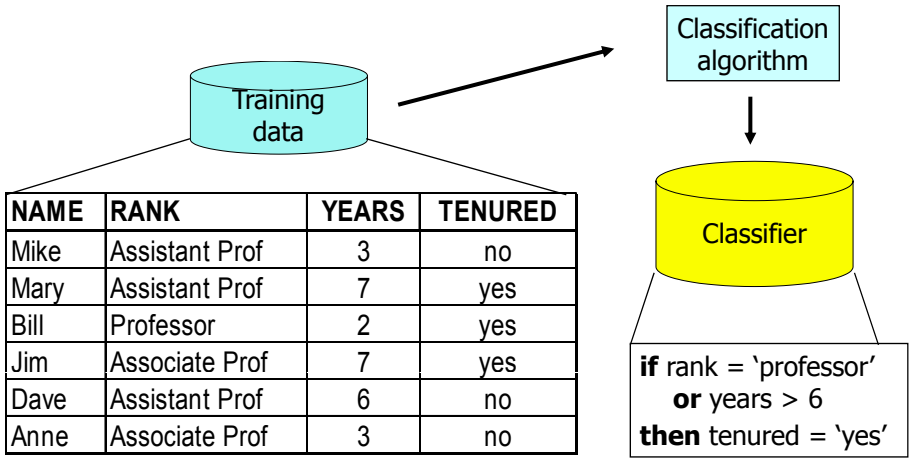
Introduction: Example

ID	age	car type	risk
1	23	family	high
2	17	sportive	high
3	43	sportive	high
4	68	family	low
5	32	truck	low

- Simple Classifier

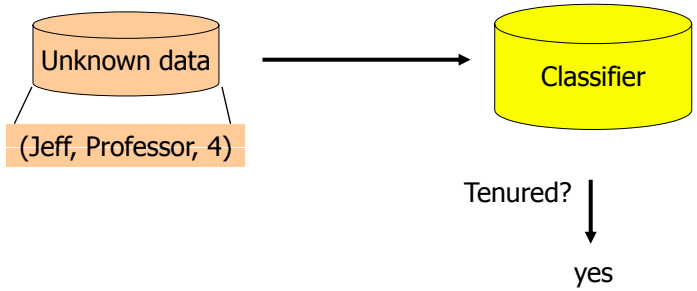
```
if age > 50 then risk = low;
if age ≤ 50 and car type = truck then risk = low;
if age ≤ 50 and car type ≠ truck then risk = high.
```

Classification Process: Model Construction (= training phase)



NAME	RANK	YEARS	TENURED
Mike	Assistant Prof	3	no
Mary	Assistant Prof	7	yes
Bill	Professor	2	yes
Jim	Associate Prof	7	yes
Dave	Assistant Prof	6	no
Anne	Associate Prof	3	no

Classification Process: Application of the Model (test phase, application phase)



- Goal is sometimes not to classify unknown data but to get a better understanding of the data

Evaluation of Classifiers – Accuracy

- *Classification Accuracy*
  - Predict the class label for each object from a data set *o* (= test set)
  - Determine the fraction of correctly predicted class labels:

$$classification\ accuracy = \frac{count(\text{correctly predicted class label})}{count(o)}$$

  - Classification error = 1 – classification accuracy

Evaluation of Classifiers – Notions

- *Overfitting*
  - Classifier is optimized to training data
  - May yield worse results for entire data set
  - Potential reasons
    - bad quality of training data (noise, missing values, wrong values)
    - different statistical characteristics of training data and test data
- *Train-and-Test*
  Decomposition of data set o into two partitions
  - *Training data* to train the classifier
    - construction of the model by using information about the class labels
  - *Test data* to evaluate the classifier
    - temporarily hide class labels, predict them anew and compare results with original class labels

Evaluation of Classifiers – Cross Validation

- Train-and-Test is not applicable if the set of objects for which the class label is known is very small
- *m*-fold *Cross Validation*
  - Decompose data set evenly into *m* subsets of (nearly) the same size.
  - Iteratively use *m* – 1 partitions as training data and the remaining single partition as test data.
  - Combine the *m* classification accuracy values to an overall classification accuracy, and combine the *m* generated models to an overall model for the data.

Evaluation of Classifiers – Leave-One-Out

- If data set is very small
- *Leave-one-out* is, in some sense, a degenerate variant of cross validation
  - For each of the objects *o* in the data set *D*:
    - Use set *D* – *o* as training set
    - Use the singleton set {*o*} as test set
    - Predict the class label of object *o*
  - Compute classification accuracy by dividing the number of correct predictions through the database size |D|
- Particularly well applicable to nearest-neighbor classifiers

Quality Measures for Classifiers

- Classification accuracy
- Compactness of the model
  - decision tree size; number of decision rules
- Interpretability of the model
  - Insights and understanding provided by the model
- Efficiency
  - Time to generate the model (training time)
  - Time to apply the model (prediction time)
- Scalability for large databases
  - Efficiency in disk-resident databases
- Robustness
  - Robust against noise or missing values

Quality Measures for Classifiers

- Let *K* be a classifier, $TR \subseteq O$ a training set, and $TE \subseteq O$ a test set. Let C(o) denote the correct class label of an object $o \in O$.
- *Classification Accuracy* of *K* on *TE:*

$$G_{TE}(K) = \frac{|\{o \in TE, K(o) = C(o)\}|}{|TE|}$$

- True classification error

$$F_{TE}(K) = \frac{|\{o \in TE, K(o) \neq C(o)\}|}{|TE|}$$

- Apparent classification error

$$F_{TR}(K) = \frac{|\{o \in TR, K(o) \neq C(o)\}|}{|TR|}$$

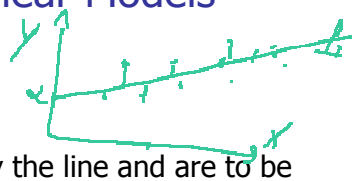
What Is Prediction?

- Prediction is *similar* to classification
  - First, construct a model
  - Second, use model to predict unknown value
    - Major method for prediction is regression
      - Linear and multiple regression
      - Non-linear regression
- Prediction is *different* from classification
  - Classification refers to predict categorical class label
  - Prediction models continuous-valued functions

Predictive Modeling in Databases

- Predictive modeling: Predict data values or construct generalized linear models based on the database data.
  - Predict value ranges or category distributions
  - Predict individual values (by regression techniques)
- Method outline:
  - Minimal generalization
  - Attribute relevance analysis
  - Generalized linear model construction
  - Prediction
- Determine the major factors which influence the prediction
  - Data relevance analysis: uncertainty measurement, entropy analysis, expert judgement, etc.
- Multi-level prediction: drill-down and roll-up analysis

Regress Analysis and Log-Linear Models in Prediction



- *Linear regression:* $Y = \alpha + \beta X$
  - Two parameters, α and β specify the line and are to be estimated by using the data at hand.
  - using the least squares criterion to the known values of Y_1, Y_2, …, X_1, X_2, …
- *Multiple regression:* $Y = b_0 + b_1 X_1 + b_2 X_2$
  - Many nonlinear functions can be transformed into the above.
- *Log-linear models:*
  - The multi-way table of joint probabilities is approximated by a product of lower-order tables.
  - Predict values of cells in a cube by using margin values.
  - Applicable to categorial data only.
  - Probability: $p(a, b, c, d) = \alpha_{ab} \beta_{ac} \chi_{ad} \delta_{bcd}$

Chapter 4: Classification

- Introduction
  - Classification problem, evaluation of classifiers
- Bayesian Classifiers
  - Optimal Bayes classifier, naive Bayes classifier, applications
- Nearest Neighbor Classifier
  - Basic notions, choice of parameters, applications
- Decision Tree Classifiers
  - Basic notions, split strategies, overfitting, pruning of decision trees
- Support vector machines

Bayesian Classification: Why?

- *Probabilistic learning*
  - Calculate explicit probabilities for hypothesis, among the most practical approaches to certain types of learning problems
- *Incremental*
  - Each training example can incrementally increase/decrease the probability that a hypothesis is correct. Prior knowledge can be combined with observed data.
- *Probabilistic prediction*
  - Predict multiple hypotheses, weighted by their probabilities
- *Standard*
  - Even when Bayesian methods are computationally intractable, they can provide a standard of optimal decision making against which other methods can be measured

Bayes Classifiers – Motivation

- Given
  - an object *o* and two class labels, *positive* and *negative*
  - three independent hypotheses h_1, h_2, h_3
  - the a-posteriori probabilities of the hypotheses for a given *o:*
    - $P(h_1 \mid o) = 0.4$
    - $P(h_2 \mid o) = 0.3$
    - $P(h_3 \mid o) = 0.3$
  - the a-posteriori probabilities of the classes for a given hypothesis:
    - $P(\textit{negative} \mid h_1) = 0$, $P(\textit{positive} \mid h_1) = 1$
    - $P(\textit{negative} \mid h_2) = 1$, $P(\textit{positive} \mid h_2) = 0$
    - $P(\textit{negative} \mid h_3) = 1$, $P(\textit{positive} \mid h_3) = 0$
- Result: Object *o* belongs
  - to class *positive* with a probability of 0.4 (=1*0.4+0*0.3+0*0.3)
  - to class *negative* with a probability of 0.6 (=0*0.4+1*0.3+1*0.3)

Optimal Bayes Classifier (1)

- Let $H = \{h_1, \ldots h_l\}$ a set of *independent* hypotheses
- Let *o* be a query object to be classified
- The optimal Bayes classifier assigns the following class label to object *o:*

$$\arg\max_{c_j \in C} \left\{ \sum_{h_i \in H} P(c_j \mid h_i) \cdot P(h_i \mid o) \right\}$$

- See example from above: object *o* is assigned to class *negative*

- Optimal Bayes classifier
  - Among the classifiers using the same a-priori knowledge, there is no classifier that yields a better classification accuracy.

Optimal Bayes Classifier (2)

Bayes´ rule: $p(c_j \mid o) \cdot p(o) = p(o \mid c_j) \cdot p(c_j)$

- The assumption that in any case, exactly one of the hypothesis h_i holds leads to a *simplified decision rule:*

$$\arg\max_{c_j \in C} \{ p(c_j \mid o) \}$$

- Since, typically, the values of $P(c_j \mid o)$ are not known, the rule is transformed by using *Bayes´ theorem:*

$$\arg\max_{c_j \in C} \{ p(c_j \mid o) \} = \arg\max_{c_j \in C} \left\{ \frac{p(o \mid c_j) \cdot p(c_j)}{p(o)} \right\} = \arg\max_{c_j \in C} \{ p(o \mid c_j) \cdot p(c_j) \}$$

- Final decision rule for the *optimal Bayes classifier* (called *Maximum Likelihood* classifier)

$$c_{max} = \arg\max_{c_j \in C} \{ P(o \mid c_j) \cdot P(c_j) \}$$

Naïve Bayes Classifier (1)

- Estimate the values of p(c_j) by using the observed frequency of the individual class labels c_j
- How to estimate the values of p(o | c_j)?
- Assumptions of the naïve Bayes classifier
  - Objects are given as d-dim. vectors, o = (o_1, …, o_d)
  - For any given class c_j, the attribute values o_i are *conditionally independent*, i.e.

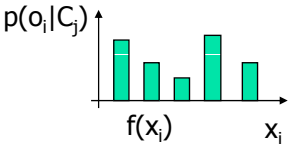
$$p(o_1,\ldots,o_d \mid c_j) = \prod_{i=1}^{d} p(o_i \mid c_j)$$

- Decision rule for the *naïve Bayes classifier*

$$\arg\max_{c_j \in C} \left\{ p(c_j) \cdot \prod_{i=1}^{d} p(o_i \mid c_j) \right\}$$

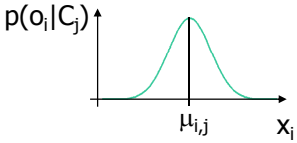
Naïve Bayesian Classifier (2)

- Independency assumption: $p(o_1,\ldots,o_d \mid c_j) = \prod_{i=1}^{d} p(o_i \mid c_j)$
- If i-th attribute is categorical:
  p(o_i|C) is estimated as the relative frequency of samples having value x_i as i-th attribute in class C



- If i-th attribute is continuous:
  p(o_i|C) is estimated through e.g.:
  - Gaussian density function determine ($\mu_{i,j}$, $\sigma_{i,j}$)

$$\Rightarrow p(o_i \mid C_j) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2}\left(\frac{o_i - \mu_{i,j}}{\sigma_{i,j}}\right)^2}$$



- Computationally easy in both cases

Bayesian Classifier

- Assuming dimensions of o =(o_1…o_d) are not independent
- Assume multivariate normal distribution (=Gaussian)

$$P(o \mid C_j) = \frac{1}{(2\pi)^{d/2} \mid \Sigma_j \mid^{1/2}} e^{-\frac{1}{2}(o-\mu_j)\Sigma_j^{-1}(o-\mu_j)^T}$$

with

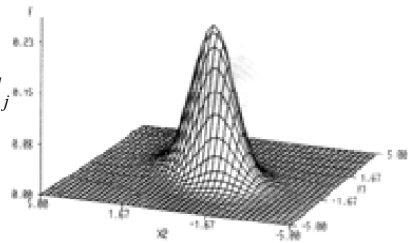
μ_j mean vector of class C_j

N_j is number of objects of class C_j

Σ_j is the d x d covariance matrix

$$\Sigma_j = \sum_{i=1}^{N_j} (o_i - \mu_j) \bullet (o_i - \mu_j)^T$$

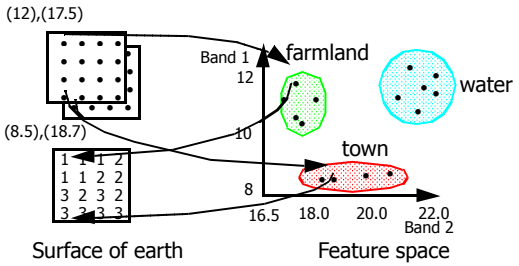
(outer product)



$|\Sigma_j|$ is the determinant of Σ_j *and* Σ_j^{-1} is the inverse of Σ_j

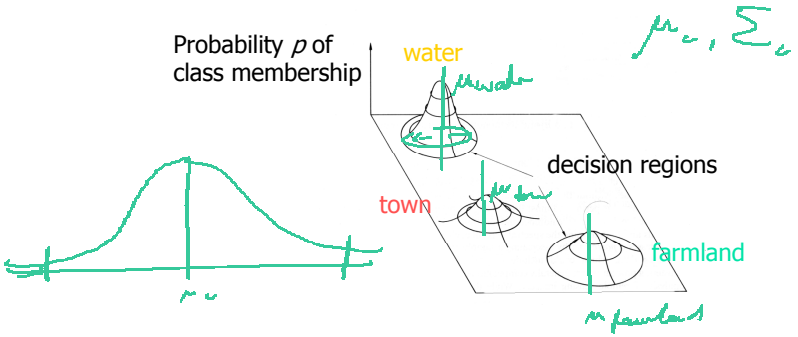
Example: Interpretation of Raster Images

- Scenario: automated interpretation of raster images
  - Take d images from a certain region (d frequency bands)
  - Represent each pixel by d gray values: (o_1, …, o_d)
- Basic assumption: different surface properties of the earth („landuse") follow a characteristic reflection and emission pattern



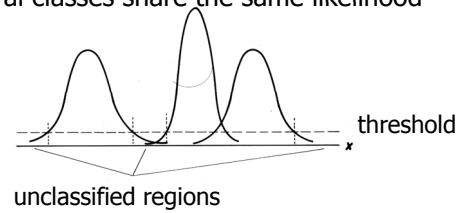
Interpretation of Raster Images

- Application of the optimal Bayes classifier
  - Estimation of the p(o | c) without assumption of conditional independence
  - Assumption of d-dimensional normal (= Gaussian) distributions for the gray value vectors of a class



Interpretation of Raster Images

- Method: Estimate the following measures from training data
  - μ_j: d-dimensional mean vector of all feature vectors of class c_j
  - Σ_j: d x d Covariance matrix of class c_j
- Problems with the decision rule
  - if likelihood of respective class is very low
  - if several classes share the same likelihood



Example: Classification of Text

- Applications [Craven et al. 1999], [Chakrabarti, Dom & Indyk 1998]
  - email filtering (spam, categories)
  - classification of web sites
- Vocabulary $T = \{t_1, \ldots, t_d\}$ of relevant terms
- Representation of a text document $o = (o_1, \ldots, o_d)$
  - o_i indicates the frequency how often term t_i occurs in document o
- Method
  - Selection of relevant terms
  - Computation of term frequencies
  - Classification of new documents

Classifying Text: Selection of Terms

- Reduction of occurring terms to basic representatives
  - Stemming
  - Depending on the language of the texts
- Single-word of multi-word terms?
- Elimination of stop words (and, or, is, …)
- Further reduction of the number of terms
- Still up to 100,000 terms to handle

Classifying Text: Reduction of the Number of Terms

- Optimal approach
  - There are $O(2^{NumberOfTerms})$ many subsets of terms
  - Optimal subset cannot be determined efficiently
- Greedy approach
  - Evaluate the separatability of each term individually
  - Descendingly sort the terms according to that measure
  - Choose the first d terms as attributes

Classifying Text: Classification of New Documents (1)

- Application of the naive Bayes classifier
  - Problem: frequencies of the different terms are not independent from each other but are, typically, correlated
- Important task:
  - Estimate the conditional probabilities $p(o_i \mid c_j)$ from the training documents
- Consider the generation of a document o from class c which has n terms by a Bernoulli experiment
  - Assume a coin that has a face for each of m terms t_i
  - Throw this m-sided coin n times

Classifying Text: Classification of New Documents (2)

- Probability that coin shows face t_i
  - $f(t_i, c)$ is the relative frequency of term t_i in class c
- Potential problem
  - Term t_i does not occur in any training document of class c_j
  - Term t_i occurs in a document o to be classified
  - Within that document o, other important (characteristic) terms of class c_j occur
- Goal: avoid $P(o_i \mid c_j) = 0$
- Smoothing of relative frequencies

Classifying Text: Experiments

- Experimental setup [Craven et al. 1999]
  - Training set: 4,127 web pages of computer science dept's
  - Classes: department, faculty, staff, student, research project, course, other
  - 4-fold cross validation: Three universities for training, fourth university for test
- Summary of results
  - Classification accuracies of 70% to 80% for most classes
  - Classification accuracy of 9% for class staff but 80% correct in superclass person
  - Poor classification accuracy for class other due to high variance of the documents in that class

Bayesian Classifiers – Discussion

+ Optimality
  → golden standard for comparison with competing classifiers
+ High classification accuracy for many applications
+ Incremental computation
  → classifier can be adopted to new training objects
  (store *count*, *sum*, *square-sum* to derive *mean*, *variance* etc.)
+ Incorporation of expert knowledge about the application

– Limited applicability
  → often, required conditional probabilities are not available
– Lack of efficient computation
  → in case of a high number of attributes
  → particularly for Bayesian belief networks

The independence hypothesis…

- … makes computation possible
- … yields optimal classifiers when satisfied
- … but is seldom satisfied in practice, as attributes (variables) are often correlated.
- Attempts to overcome this limitation:
  - Bayesian networks, that combine Bayesian reasoning with causal relationships between attributes
  - Decision trees, that reason on one attribute at the time, considering most important attributes first

Chapter 4: Classification

- Introduction
  - Classification problem, evaluation of classifiers
- Bayesian Classifiers
  - Optimal Bayes classifier, naive Bayes classifier, applications
- Nearest Neighbor Classifier
  - Basic notions, choice of parameters, applications
- Decision Tree Classifiers
  - Basic notions, split strategies, overfitting, pruning of decision trees
- Support vector machines

Supervised vs. Unsupervised Learning

- Supervised learning (classification)
  - Supervision: The training data (observations, measurements, etc.) are accompanied by labels indicating the class of the observations
  - New data is classified based on the training set
- Unsupervised learning (clustering)
  - The class labels of training data is unknown
  - Given a set of measurements, observations, etc. with the aim of establishing the existence of classes or clusters in the data

Instance-Based Methods

- Instance-based learning:
  - Store training examples and delay the processing ("lazy evaluation") until a new instance must be classified
- Typical approaches
  - *k*-nearest neighbor approach
    - Instances represented as points in a Euclidean space or, more general, as points in a *metric* space.
  - Locally weighted regression
    - Constructs local approximation
  - Case-based reasoning
    - Uses symbolic representations and knowledge-based inference

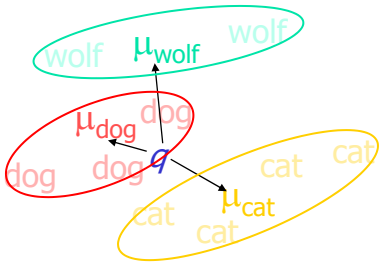
Nearest Neighbor Classifiers

- Motivation: Problems with Bayes classifiers
  - Assumption of normal (Gaussian) distribution of the vectors of a class requires the estimation of the parameters μ_i and Σ_i
  - Estimation of μ_i requires significantly less training data than the estimation of Σ_i
- Objective
  - Classifier that requires no more information than mean values μ_i of each class c_i
  - Or even less than mean values but only the training points

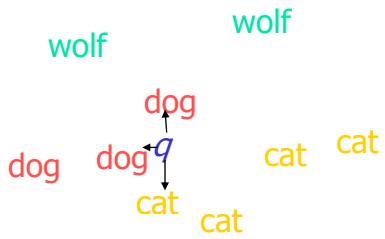
  → Nearest neighbor classifier

Nearest Neighbor Classifiers: Example

Variant 1
use *mean values* μ_j



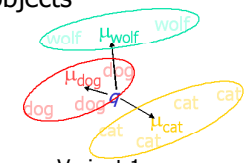
Variant 2
use *individual objects*



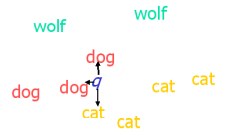
- Classifier decides that query object *q* is a dog
- Instance-based learning
- Related to case-based reasoning

Nearest Neighbor Classifiers: Basics

- Fundamental procedure
  - Use attribute vectors $o = (o_1, \ldots, o_d)$ as training objects
  - Variant 1:
    - Determine mean vector μ_i for each class c_j (in training phase)
    - Assign query object to the class c_j of the nearest mean vector μ_i
  - Variant 2:
    - Assign query object to the class c_j of the closest training object
  - Generalizations:
    - Use more than one representative per class (Var. 1)
    - Consider $k > 1$ neighbors for the class assignment decision (Var. 2)
    - Use weights for the classes of the k nearest neighbors



Variant 1

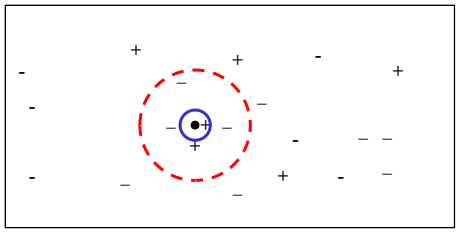


Variant 2

Nearest Neighbor Classifiers: Notions

- Distance function
  - Defines the (dis-)similarity for pairs of objects
- Number *k* of neighbors to be considered
- Decision set
  - Set of *k* nearest neighboring objects to be used in the decision rule
- Decision rule
  - Given the class labels of the objects from the decision set, how to determine the class label to be assigned to the query object?

Nearest Neighbor Classifiers: Example



Classes + and –

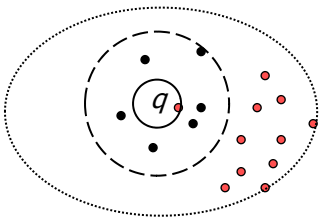
decision set for $k = 1$

decision set for $k = 5$

- Using unit weights (i.e., no weights) for the decision set
  - Simply called *"majority criterion"*
  - rule $k = 1$ yields class „+", rule $k = 5$ yields class „–"
- Using the reciprocal square of the distances as weights
  - Both rules, $k = 1$ and $k = 5$, yield class „+"
- Using a-priori probability (=frequency) of classes as weights
  - Both rules, $k = 1$ and $k = 5$, yield class „+"

Nearest Neighbor Classifiers: Parameters

- Problem of choosing an appropriate value for parameter *k*
  - *k* too small: high sensitivity against outliers
  - *k* too large: decision set contains many objects from other classes
  - Empirically, $1 << k < 10$ yields a high classification accuracy in many cases



decision set for $k = 1$

decision set for $k = 7$

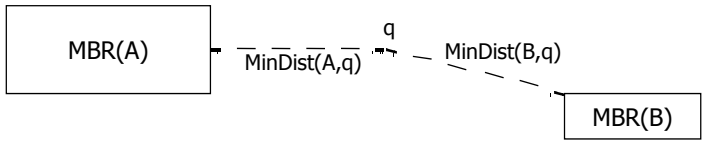
decision set for $k = 17$

Nearest Neighbor Classifiers: Decision Rules

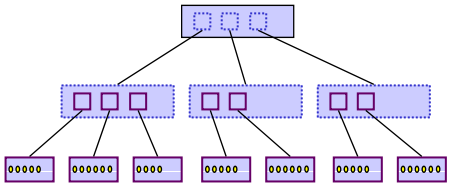
- Standard rule
  - Choose majority class in the decision set, i.e. the class with the most representatives in the decision set
- Weighted decision rules
  - Use weights for the classes in the decision set
    - Use distance to the query object: $1/d(o,q)^2$
    - Use frequency of classes in the training set, i.e. the *a-priori probability* of the class
- Example
  - Class a: 95%, class b: 5%
  - Decision set = {a, a, a, a, b, b, b}
  - Standard rule yields class a
  - Weighted rule yields class b

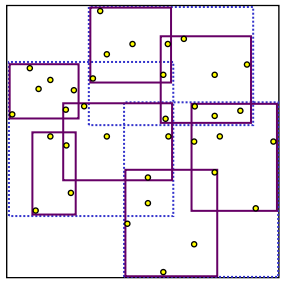
NN Classifiers: Index Support

- Assume a balanced indexing structure to be given
  - Examples: R-tree, X-tree, M-tree, …
- Nearest Neighbor Search
  - Query point *q*
  - Partition list
    - Minimum bounding rectangles (MBRs) for which the corresponding subtrees have still to be processed
  - NN: nearest neighbor of *q* in the data pages read up to now



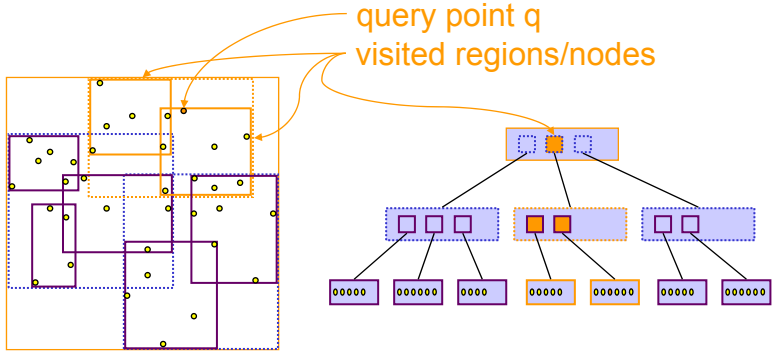
NN Classifiers: Index Support with R-Trees





- properties of the R-Tree
  - leaf nodes contain data points, inner nodes contain MBRs (Minimum Bounding Rectangles) and pointers
  - all leaves have the same distance (= path length) to the root node
  - each node contains at most M entries
  - each node (exception: root node) contains at least m ($\leq M/2$) entries

NN Classifiers: Index Support with R-Trees (2)



- Start with root node
- Update "partition list" (sorted by *mindist*) with nodes, that still need to be visited (prune nodes, that have larger *mindist* than distance from q to NN candidate)
- Visit next node in "partition list" (lowest *mindist*)
- When visiting leaf node: update possible NN candidate (if any point in node is closer than previously found NN candidate)
- This is a "best first search" algorithm which runs in between O(log n) and O(n)

Example: Classification of Stars

- Analysis of astronomical data

Removal of noise

Image segmentation

Feature extraction

Automatic *Classification* of star type

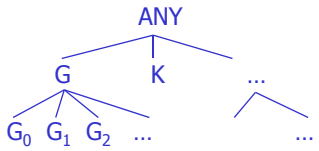
Manual analysis of interesting star types

- Classification of star types with a NN classifier
- Use the Hipparcos catalogue as training set

Classification of Stars: Training Data

- Use *Hipparcos Catalogue* [ESA 1998] to „train" the classifier

- Contains around 118,000 stars
- 78 attributes (brightness, distance from earth, color spectrum, …)
- Class label attribute: spectral type (= attribute H76)
  - Examples
    - H76: G0
    - H76: G7.2
    - H76: KIII/IV



- Values of the spectral type are vague
- Hierarchy of classes
  - Use the first level of the class hierarchy

Classification of Stars: Training Data

Distribution of classes in Hipparcos Catalogue

Class	#Instances	fraction of instances	
K	32,036	27.0	frequent classes
F	25,607	21.7	
G	22,701	19.3	
A	18,704	15.8	
B	10,421	8.8	
M	4,862	4.1	
O	265	0.22	rare classes
C	165	0.14	
R	89	0.07	
W	75	0.06	
N	63	0.05	
S	25	0.02	
D	27	0.02	

Classification of Stars: Experiments

Experimental Evaluation [Poschenrieder 1998]

- Distance function
  - using 6 attributes (color, brightness, distance from earth)
  - using 5 attributes (color, brightness)
  - Result: best classification accuracy obtained for 6 attributes

- Number k of neighbors
  - Result: best classification accuracy obtained for $k = 15$

- Decision Rule
  - weighted by distance
  - weighted by class frequency
  - Result: best classification accuracy obtained by using distance-based weights but not by using frequency-based weights

Classification of Stars: Experiments

class	incorrectly classified	correctly classified	classification accuracy
K	408	2338	85.1%
F	350	2110	85.8%
G	784	1405	64.2%
A	312	975	75.8%
B	308	241	43.9%
M	88	349	79.9%
C	4	5	55.6%
R	5	0	0%
W	4	0	0%
O	9	0	0%
N	4	1	20%
D	3	0	0%
S	1	0	0%
Total	2461	7529	75.3%

- High accuracy for frequent classes, poor accuracy for rare classes
- Most of the rare classes have less than $k / 2 = 8$ instances

NN Classification: Discussion

+ applicability: training data required only
+ high classification accuracy in many applications
+ easy incremental adaptation to new training objects
+ useful also for prediction
+ robust to noisy data by averaging *k*-nearest neighbors

– naïve implementation is inefficient
  - requires k-nearest neighbor query processing
  - support by database techniques may help to reduce from O(n) to O(log n) for n training objects
– does not produce explicit knowledge about classes
  - But provides some explanation information
– Curse of dimensionality: distance between neighbors could be dominated by irrelevant attributes
  - To overcome it, stretch axes or eliminate least relevant attributes

Remarks on Lazy vs. Eager Learning

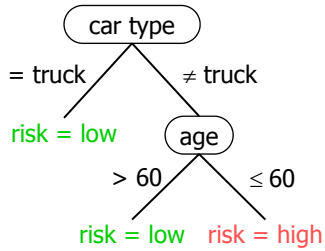
- Instance-based learning: lazy evaluation
- Decision-tree and Bayesian classification: eager evaluation
- Key differences
  - Lazy method may consider query instance *xq* when deciding how to generalize beyond the training data *D*
  - Eager method cannot since they have already chosen global approximation when seeing the query
- Efficiency
  - Lazy - less time training but more time predicting
- Accuracy
  - Lazy method effectively uses a richer hypothesis space since it uses many local linear functions to form its implicit global approximation to the target function
  - Eager: must commit to a single hypothesis that covers the entire instance space

Chapter 4: Classification

- Introduction
  - Classification problem, evaluation of classifiers
- Bayesian Classifiers
  - Optimal Bayes classifier, naive Bayes classifier, applications
- Nearest Neighbor Classifier
  - Basic notions, choice of parameters, applications
- Decision Tree Classifiers
  - Basic notions, split strategies, overfitting, pruning of decision trees
- Support vector machines

Decision Tree Classifiers: Motivation

ID	age	car type	risk
1	23	family	high
2	17	sportive	high
3	43	sportive	high
4	68	family	low
5	32	truck	low



- Decision trees represent explicit knowledge
- Decision trees are intuitive to most users

Decision Tree Classifiers: Basics

- Decision tree
  - A flow-chart-like tree structure
  - Internal node denotes a test on an attribute
  - Branch represents an outcome of the test
  - Leaf nodes represent class labels or class distribution
- Decision tree generation consists of two phases
  - Tree construction
    - At start, all the training examples are at the root
    - Partition examples recursively based on selected attributes
  - Tree pruning
    - Identify and remove branches that reflect noise or outliers
- Use of decision tree: Classifying an unknown sample
  - Traverse the tree and test the attribute values of the sample against the decision tree
  - Assign the class label of the respective leaf to the query object

Algorithm for Decision Tree Induction

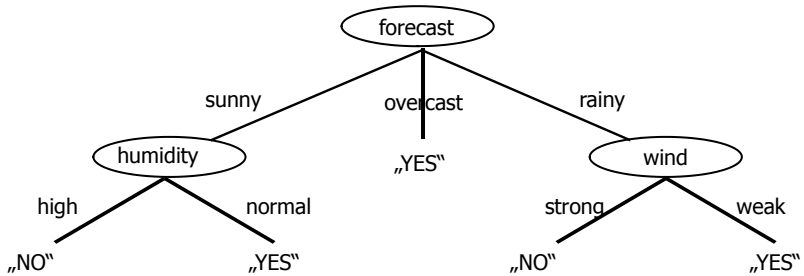
- Basic algorithm (a greedy algorithm)
  - Tree is created in a top-down recursive divide-and-conquer manner
  - Attributes may be categorical or continuous-valued
  - At start, all the training examples are assigned to the root node
  - Recursively partition the examples at each node and push them down to the new nodes
    - Select test attributes and determine split points or split sets for the respective values on the basis of a heuristic or statistical measure (*split strategy,* e.g., information gain)
- Conditions for stopping partitioning
  - All samples for a given node belong to the same class
  - There are no remaining attributes for further partitioning – majority voting is employed for classifying the leaf
  - There are no samples left

Decision Tree Classifiers: Training Data for „playing_tennis"

- Query: How about playing tennis today?
- Training data set:

day	forecast	temperature	humidity	wind	tennis decision
1	sunny	hot	high	weak	no
2	sunny	hot	high	strong	no
3	overcast	hot	high	weak	yes
4	rainy	mild	high	weak	yes
5	rainy	cool	normal	weak	yes
6	rainy	cool	normal	strong	no
7	sunny	hot	normal	weak	yes

Decision Tree Classifiers: A Decision Tree for „playing_tennis"

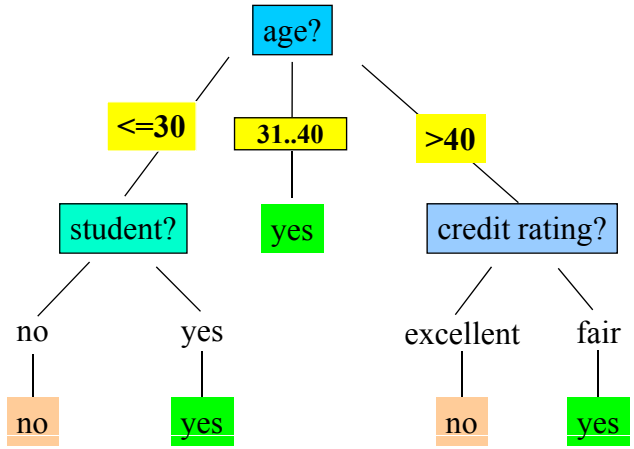


Example: Training Dataset for "buys_computer"

This follows an example from Quinlan's ID3

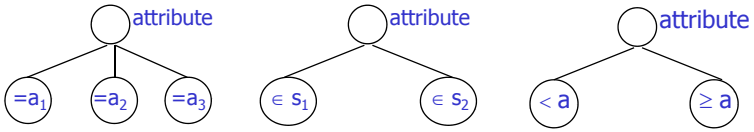
age	income	student	credit_rating	
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

Output: A Decision Tree for "buys_computer"



Split Strategies: Types of Splits

- Categorical attributes
  - split criteria based on equality „*attribute* = *a*" or
  - based on subset relationships „*attribute* ∈ *set*"
  - many possible choices (subsets)



- Numerical attributes
  - split criteria of the form „*attribute* < *a*"
  - many possible choices for the split point

Split Strategies: Quality of Splits

- Given
  - a set T of training objects
  - a (disjoint, complete) partitioning $T_1, T_2, \ldots, T_m$ of T
  - the relative frequencies p_i of class c_i in T
- Searched
  - a measure for the heterogeneity of a set S of training objects with respect to the class membership
  - a split of T into partitions $T_1, T_2, \ldots, T_m$ such that the heterogeneity is minimized
- Proposals: Information gain, Gini index

Split Strategies: Attribute Selection Measures

- Information gain (ID3/C4.5)
  - All attributes are assumed to be categorical
  - Can be modified for continuous-valued attributes
- Gini index (IBM IntelligentMiner)
  - All attributes are assumed continuous-valued
  - Assume there exist several possible split values for each attribute
  - May need other tools, such as clustering, to get the possible split values
  - Can be modified for categorical attributes

Split Strategies: Information Gain

- used in ID3 / C4.5
- Entropy
  - minimum number of bits to encode a message that contains the class label of a random training object
  - the *entropy* of a set T of training objects is defined as follows:

$$entropy(T) = \sum_{i=1}^{k} p_i \cdot \log_2 p_i$$

for k classes c_i with frequencies p_i

  - entropy(T) = 0 if $p_i = 1$ for any class c_i
  - entropy (T) = 1 if there are $k = 2$ classes with $p_i = ½$ for each i

- Let A be the attribute that induced the partitioning $T_1, T_2, \ldots, T_m$ of T. The *information gain* of attribute A wrt. T is defined as follows:

$$information\ gain(T, A) = entropy(T) - \sum_{i=1}^{m} \frac{|T_i|}{|T|} \cdot entropy(T_i)$$

Split Strategies: Gini Index

- Used in IBM's IntelligentMiner
- The *Gini index* for a set T of training objects is defined as follows

$$gini(T) = 1 - \sum_{j=1}^{k} p_j^2$$

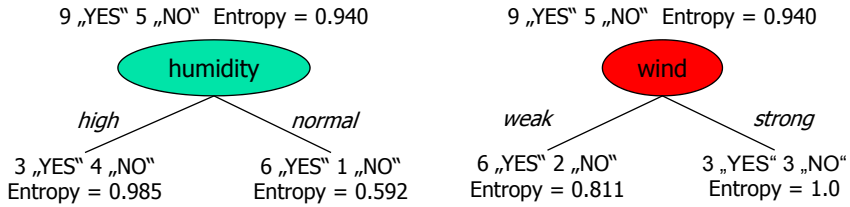
for k classes c_i with frequencies p_i

  - small value of Gini index ⇔ low heterogeneity
  - large value of Gini index ⇔ high heterogeneity

- Let A be the attribute that induced the partitioning $T_1, T_2, \ldots, T_m$ of T. The *Gini index* of attribute A wrt. T is defined as follows:

$$gini_A(T) = \sum_{i=1}^{m} \frac{|T_i|}{|T|} \cdot gini(T_i)$$

Split Strategies: Example



$$information\ gain(T, humidity) = 0.94 - \frac{7}{14} \cdot 0.985 - \frac{7}{14} \cdot 0.592 = 0.151$$

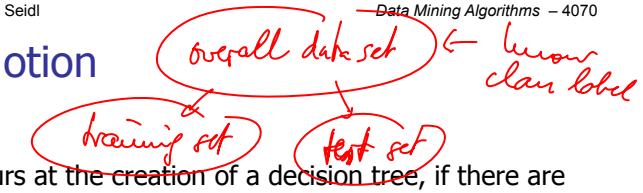
$$information\ gain(T, wind) = 0.94 - \frac{8}{14} \cdot 0.811 - \frac{6}{14} \cdot 1.0 = 0.048$$

Result: humidity yields the highest information gain

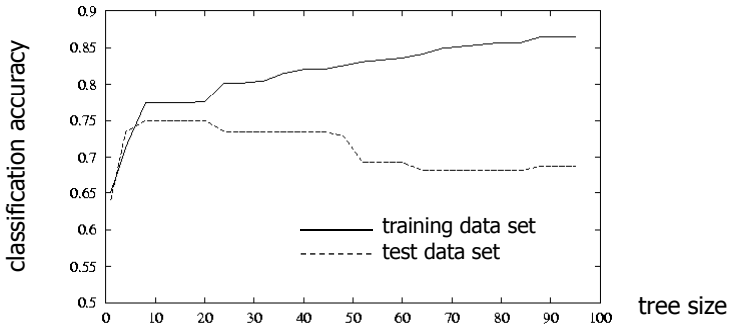
Avoid Overfitting in Classification

- The generated tree may overfit the training data
  - Too many branches, some may reflect anomalies due to noise or outliers
  - Result is in poor accuracy for unseen samples
- Two approaches to avoid overfitting
  - Prepruning: Halt tree construction early—do not split a node if this would result in the goodness measure falling below a threshold
    - Difficult to choose an appropriate threshold
  - Postpruning: Remove branches from a "fully grown" tree—get a sequence of progressively pruned trees
    - Use a set of data different from the training data to decide which is the "best pruned tree"

Overfitting: Notion



- *Overfitting* occurs at the creation of a decision tree, if there are two trees E and E' for which the following holds:
  - on the training set, E has a smaller error rate than E'
  - on the overall data set, E' has a smaller error rate than E



Overfitting: Avoidance

- Removal of noisy and **erraneous** training data
  - in particular, remove contradicting training data
- Choice of an appropriate size of the training set
  - not too small, not too large
- Choice of an appropriate value for minimum support
  - *minimum support:* minimum number of data objects a leaf node contains
  - in general, *minimum support* >> 1
- Choice of an appropriate value for minimum confidence
  - *minimum confidence:* minimum fraction of the majority class in a leaf node
  - typically, minimum confidence << 100%
  - leaf nodes can errors or noise in data records absorb
- Post pruning of the decision tree
  - pruning of overspecialized branches

Pruning of Decision Trees: Approach

Error-Reducing Pruning [Mitchell 1997]

- Decompose classified data into training set and test set
- Creation of a decision tree E for the training set
- Pruning of E by using the test set T
  - determine the subtree of E whose pruning reduces the classification error on T the most
  - remove that subtree
  - finished if no such subtree exists

- only applicable if a sufficient number of classified data is available

Pruning of Decision Trees: Approach

Minimal Cost Complexity Pruning
[Breiman, Friedman, Olshen & Stone 1984]

- Does not require a separate test set
  - applicable to small training sets as well
- Pruning of the decision tree by using the training set
  - classification error is no appropriate quality measure
- New quality measure for decision trees:
  - trade-off of classification error and tree size
  - weighted sum of classification error and tree size
- General observation
  - the smaller decision trees yield the better generalization

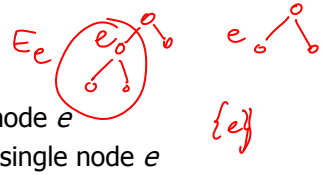
Pruning of Decision Trees: Notions

- Size $|E|$ of a decision tree *E:* number of leaf nodes
- Cost-complexity quality measure of *E* with respect to training set *T* and complexity parameter $\alpha \geq 0$:

$$CC_T(E,\alpha) = F_T(E) + \alpha \cdot |E|$$

- For the *smallest minimal subtree* $E(\alpha)$ of *E* wrt. α, it is true that:
  - (1) there is no subtree of *E* with a smaller cost complexity
  - (2) if $E(\alpha)$ and *B* both fulfill (1), then is $E(\alpha)$ a subtree of *B*
- $\alpha = 0$: $E(\alpha) = E$ i.e., only error does matter
- $\alpha = \infty$: $E(\alpha)$ = root node of *E* i.e., only tree size does matter
- $0 < \alpha < \infty$: $E(\alpha)$ is a proper substructure of E, i.e. more than the root node or the root node

Pruning of Decision Trees: Notions (2)



- Let E_e denote the subtree with root node *e* and $\{e\}$ the tree that consists of the single node *e*
- Relationship of E_e and $\{e\}$:
  - for small values of α: $CC_{Te}(E_e, \alpha) < CC_{Te}(\{e\}, \alpha)$
  - for large values of α: $CC_{Te}(E_e, \alpha) > CC_{Te}(\{e\}, \alpha)$
- Critical value of α for *e:*
  - α_{crit}: $CC_{Te}(E_e, \alpha_{crit}) = CC_{Te}(\{e\}, \alpha_{crit})$
  - for $\alpha \geq \alpha_{crit}$, it's worth to prune the tree at node *e*
- *weakest link:* node with minimal value of α_{crit}

Pruning of Decision Trees: Method

- Start with a complete tree *E*
- Iteratively remove the weakest link from the current tree
- If there are several weakest links, remove them all in the same step
- Result: sequence of pruned trees
  - $E(\alpha_1) > E(\alpha_2) > \ldots > E(\alpha_m)$ ... = {root}
  - where $\alpha_1 < \alpha_2 < \ldots < \alpha_m$
- Selection of the best $E(\alpha_i)$
  - Estimate the classification error on the overall data set by an l-fold cross validation on the training set

Pruning of Decision Trees: Example

i	|E_i|	observed error	estimated error	actual error
1	71	0,00	0,46	0,42
2	63	0,00	0,45	0,40
3	58	0,04	0,43	0,39
4	40	0,10	0,38	0,32
5	34	0,12	0,38	0,32
6	19	0,20	0,32	0,31
7	10	0,29	0,31	0,30
8	9	0,32	0,39	0,34
9	7	0,41	0,47	0,47
10	. . .	. . .	. . .	. . .

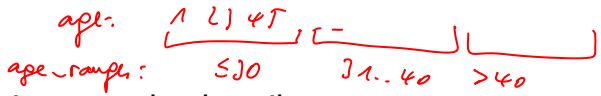
E_7 yields the smallest estimated error and the lowest actual classification error

Extracting Classification Rules from Trees

- Represent the knowledge in the form of IF-THEN rules
- One rule is created for each path from the root to a leaf
- Each attribute-value pair along a path forms a conjunction
- The leaf node holds the class prediction
- Rules are easier for humans to understand
- Example

IF *age* = '<=30' AND *student* = '*no*' THEN *buys_computer* = '*no*'
IF *age* = '<=30' AND *student* = '*yes*' THEN *buys_computer* = '*yes*'
IF *age* = '31…40' THEN *buys_computer* = '*yes*'
IF *age* = '>40' AND *credit_rating* = '*excellent*' THEN *buys_computer* = '*yes*'
IF *age* = '>40' AND *credit_rating* = '*fair*' THEN *buys_computer* = '*no*'

Enhancements to basic decision tree induction



- Allow for continuous-valued attributes
  - Dynamically define new discrete-valued attributes that partition the continuous attribute value into a discrete set of intervals
- Handle missing attribute values
  - Assign the most common value of the attribute
  - Assign probability to each of the possible values
- Attribute construction
  - Create new attributes based on existing ones that are sparsely represented
  - This reduces fragmentation, repetition, and replication

Classification in Large Databases

- Classification—a classical problem extensively studied by statisticians and machine learning researchers
- Scalability: Classifying data sets with millions of examples and hundreds of attributes with reasonable speed
- Why decision tree induction in data mining?
  - relatively faster learning speed (than other classification methods)
  - convertible to simple and easy to understand classification rules
  - can use SQL queries for accessing databases
  - comparable classification accuracy with other methods

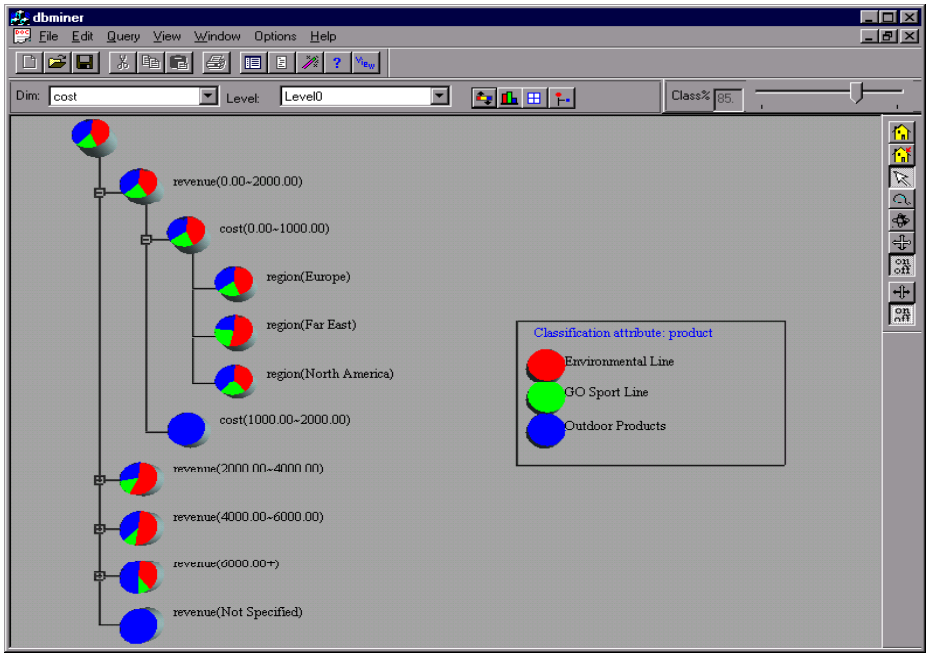
Scalable Decision Tree Induction Methods in Data Mining Studies

- SLIQ (EDBT'96 — Mehta et al.)
  - builds an index for each attribute and only class list and the current attribute list reside in memory
- SPRINT (VLDB'96 — J. Shafer et al.)
  - constructs an attribute list data structure
- PUBLIC (VLDB'98 — Rastogi & Shim)
  - integrates tree splitting and tree pruning: stop growing the tree earlier
- RainForest (VLDB'98 — Gehrke, Ramakrishnan & Ganti)
  - separates the scalability aspects from the criteria that determine the quality of the tree
  - builds an AVC-list (attribute, value, class label)

Data Cube-Based Decision-Tree Induction

- Integration of generalization with decision-tree induction (Kamber et al. '97).
- Classification at primitive concept levels
  - E.g., precise temperature, humidity, outlook, etc.
  - Low-level concepts, scattered classes, bushy classification-trees
  - Semantic interpretation problems.
- Cube-based multi-level classification
  - Relevance analysis at multi-levels.
  - Information-gain analysis with dimension + level.

Presentation of Classification Results



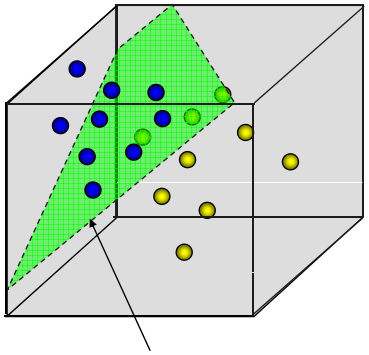
Chapter 4: Classification

- Introduction
  - Classification problem, evaluation of classifiers
- Bayesian Classifiers
  - Optimal Bayes classifier, naive Bayes classifier, applications
- Nearest Neighbor Classifier
  - Basic notions, choice of parameters, applications
- Decision Tree Classifiers
  - Basic notions, split strategies, overfitting, pruning of decision trees
- Support vector machines

© and acknowledgements: Prof. Dr. Hans-Peter Kriegel and Matthias Schubert (LMU Munich) and Dr. Thorsten Joachims (U Dortmund and Cornell U)

Support Vector Machines (SVM)

Motivation: Linear Separation



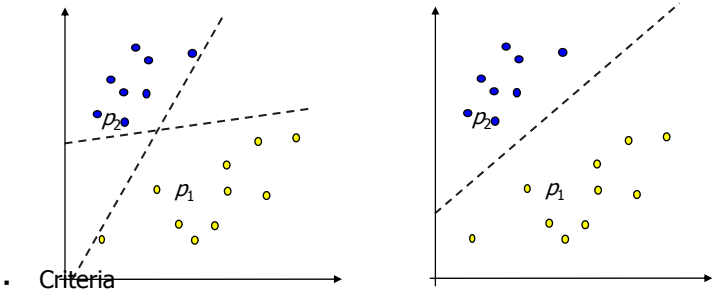
- Vectors in $\mathcal{R}^d$ represent objects
- Objects belong to exactly one of two respective classes
- For the sake of simpler formulas, the used class labels are:
  $y = -1$ and $y = +1$
- Classification by linear separation: determine hyperplane which separates both vector sets with a „maximal stability"
- Assign unknown elements to the halfspace in which they reside

Support Vector Machines

- Problems of linear separation
  - Definition and efficient determination of the maximum stable hyperplane
  - Classes are not always linearly separable
  - Computation of selected hyperplanes is very expensive
  - Restriction to two classes
  - ...
- Approach to solve these problems
  - Support Vector Machines (SVMs) [Vapnik 1979, 1995]

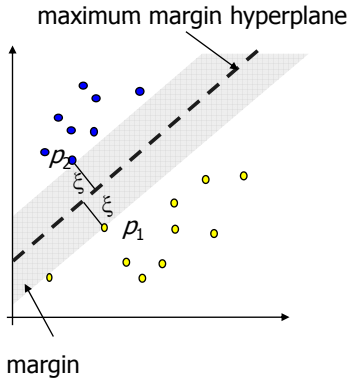
Maximum Margin Hyperplane

- Observation: There is no unique hyperplane to separate p_1 from p_2
- Question: which hyperplane separates the classes best?



- Criteria
  - Stability at insertion
  - Distance to the objects of both classes

Support Vector Machines: Principle



- Basic idea: Linear separation with the Maximum Margin Hyperplane (MMH)
  - Distance to points from any of the two sets is maximal, i.e. at least ξ
  - Minimal probability that the separating hyperplane has to be moved due to an insertion
  - Best generalization behaviour
- MMH is „maximally stable"
- MMH only depends on points p_i whose distance to the hyperplane exactly is ξ
  - p_i is called a *support vector*

Maximum Margin Hyperplane

- Recall some algebraic notions for feature space *FS*
  - Inner product of two vectors $\mathbf{x}, \mathbf{y} \in FS$: $\langle \mathbf{x}, \mathbf{y} \rangle$
    - e.g., canonical scalar product: $\langle \mathbf{x}, \mathbf{y} \rangle = \sum_{i=1}^{d} (\mathbf{x}_i \cdot \mathbf{y}_i)$
  - Hyperplane $H(\mathbf{w},b)$ with normal vector $\mathbf{w}$ and value b:
    $$\mathbf{x} \in H(\mathbf{w}, b) \quad \Leftrightarrow \quad \langle \mathbf{w}, \mathbf{x} \rangle + b = 0$$
  - The normal vector $\mathbf{w}$ may be normalized to $\mathbf{w}^0$:
    $$\mathbf{w}^0 = \frac{1}{\sqrt{\langle \mathbf{w}, \mathbf{w} \rangle}} \cdot \mathbf{w}, \quad \text{then } \langle \mathbf{w}^0, \mathbf{w}^0 \rangle = 1$$
  - Distance of a vector x to the hyperplane $H(\mathbf{w}^0,b)$:
    $$dist(\mathbf{x}, H(\mathbf{w}^0, b)) = \left| \langle \mathbf{w}^0, \mathbf{x} \rangle + b \right|$$

Computation of the Maximum Margin Hyperplane

Two assumptions for classifying x_i (class 1: $y_i = +1$, class 2: $y_i = -1$):

1) The classification is accurate (no error)

$$\left.\begin{array}{lll} y_i = -1 & \Rightarrow & \langle \mathbf{w}, \mathbf{x}_i \rangle + b < 0 \\ y_i = +1 & \Rightarrow & \langle \mathbf{w}, \mathbf{x}_i \rangle + b > 0 \end{array}\right\} \quad \Leftrightarrow \quad y_i \cdot (\langle \mathbf{w}, \mathbf{x}_i \rangle + b) > 0$$

2) The margin is maximal

- Let ξ denote the minimum distance of any training object x_i to the hyperplane $H(w,b)$: $\xi = \min_{\mathbf{x}_i \in TR} \left| \langle \mathbf{w}^0, \mathbf{x}_i \rangle + b \right|$
- Then: Maximize ξ subject to $\forall i \in [1..n]$: $y_i \cdot (\langle \mathbf{w}^0, \mathbf{x}_i \rangle + b) \geq \xi$

Maximum Margin Hyperplane

- Maximize ξ subject to $\forall i \in [1..n]$: $y_i \cdot (\langle \mathbf{w}^0, \mathbf{x}_i \rangle + b) \geq \xi$
- Scaling $\mathbf{w}^0$ by $1/\xi$, i.e. $\mathbf{w} = \mathbf{w}^0 / \xi$ yields the rephrased condition

  $$\forall i \in [1..n]: \; y_i \cdot (\langle \mathbf{w}, \mathbf{x}_i \rangle + b') \geq 1$$

- *Maximizing* ξ corresponds to *minimizing* $\langle \mathbf{w}, \mathbf{w} \rangle = \langle \mathbf{w}^0, \mathbf{w}^0 \rangle / \xi^2$:

Primary optimization problem:

Find a vector $\mathbf{w}$ and value b that minimize $\langle \mathbf{w}, \mathbf{w} \rangle$

subject to $\forall i \in [1..n]$: $y_i \cdot (\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1$

Dual Optimization Problem

- For computational purposes, transform the primary optimization problem into a dual one by using Lagrange multipliers

Dual optimization problem: Find parameters α_i that

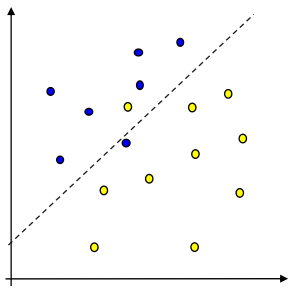
minimize $L(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \cdot \alpha_j \cdot y_i \cdot y_j \cdot \langle \mathbf{x}_i \cdot \mathbf{x}_j \rangle$

subject to $\sum_{i=1}^{n} \alpha_i \cdot y_i = 0$ and $0 \leq \alpha_i$

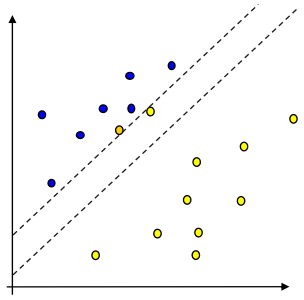
- For the solution, use algorithms from optimization theory
- Up to now only linearly separable data
- If data is not linearly separable: Soft Margin Optimization

Soft Margin Optimization

- Problem of Maximum Margin Optimization: How to treat non-linearly separable data?
  - Two typical problems:



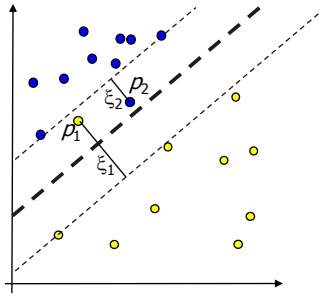
data points are not separable



complete separation is not optimal

- Trade-off between training error and size of margin

Soft Margin Optimization



- Additionally regard the number of training errors when optimizing:
  - ξ_i is the distance from p_i to the margin (often called slack variable)
  - C controls the influence of single training vectors

Primary optimization problem with soft margin:

Find an $H(\mathbf{w},b)$ that minimizes $\frac{1}{2}\langle \mathbf{w},\mathbf{w}\rangle + C\cdot\sum_{i=1}^{n}\xi_i$

subject to $\forall i \in [1..n]$: $y_i\cdot(\langle \mathbf{w},\mathbf{x}_i\rangle + b) \geq 1-\xi_i$ and $\xi_i \geq 0$

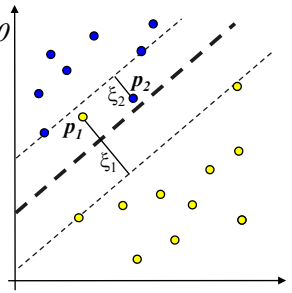
Soft Margin Optimization

Dual optimization problem with Lagrange multipliers:

Dual OP: Maximize $L(\alpha) = \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\cdot\alpha_j\cdot y_i\cdot y_j\cdot\langle \mathbf{x}_i\cdot\mathbf{x}_j\rangle$

subject to $\sum_{i=1}^{n}\alpha_i\cdot y_i = 0$ and $0 \leq \alpha_i \leq C$

$0 < \alpha_i < C$: p_i is a support vector with $\xi_i = 0$
$\alpha_i = C$: p_i is a support vector with $\xi_i > 0$
$\alpha_i = 0$: p_i is no support vector

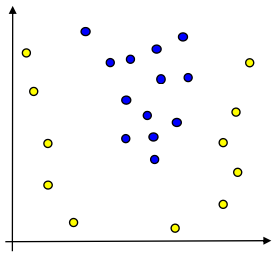


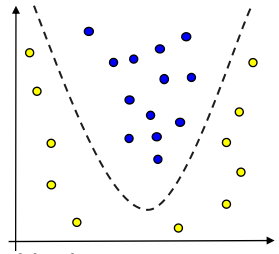
Decision rule:

$$h(\mathbf{x}) = sign\left(\sum_{\mathbf{x}_i\in SV}\alpha_i\cdot y_i\cdot\langle \mathbf{x}_i,\mathbf{x}\rangle + b\right)$$

Kernel Machines: Non-Linearly Separable Data Sets

- Problem: For real data sets, a linear separation with a high classification accuracy often is not possible
- Idea: Transform the data non-linearly into a new space, and try to separate the data in the new space linearly (extension of the hypotheses space)





Example for a quadratically separable data set

Kernel Machines: Extension of the Hypotheses Space

- Principle

input space $\Rightarrow \phi$ extended feature space

  - Try to separate in the extended feature space linearly

- Example

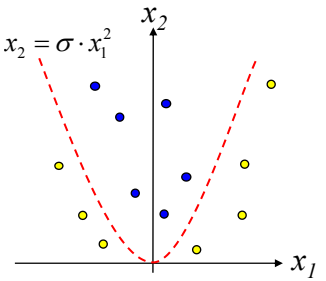
$(x, y, z) \Rightarrow \phi$ $(x, y, z, x^2, xy, xz, y^2, yz, z^2)$

  - Here: a hyperplane in the extended feature space is a polynomial of degree 2 in the input space

Kernel Machines: Example

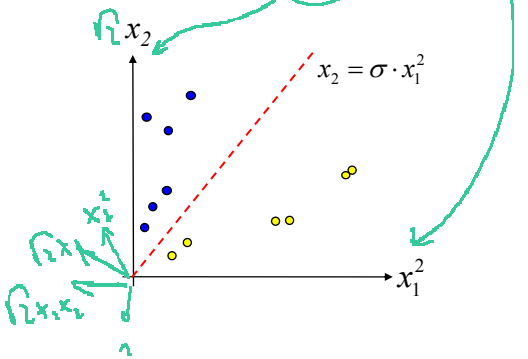
Input space (2 attributes):

$$\mathbf{x} = (x_1, x_2)$$



Extended space (6 attributes):

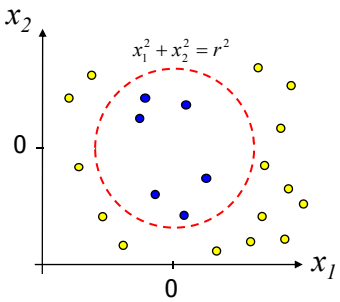
$$\phi(\mathbf{x}) = (x_1^2, x_2^2, \sqrt{2} \cdot x_1, \sqrt{2} \cdot x_2, \sqrt{2} \cdot x_1 \cdot x_2, 1)$$



Kernel Machines: Example (2)

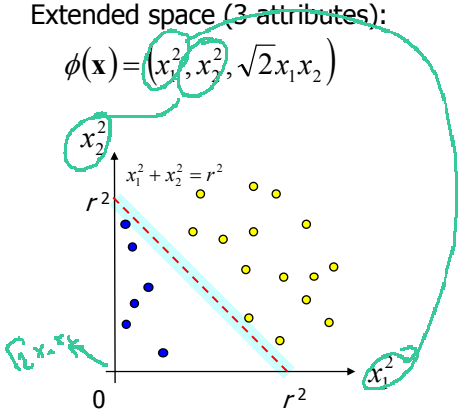
Input space (2 attributes):

$$\mathbf{x} = (x_1, x_2)$$



Extended space (3 attributes):

$$\phi(\mathbf{x}) = (x_1^2, x_2^2, \sqrt{2}x_1x_2)$$



Kernel Machines

- Introduction of a kernel corresponds to a feature transformation

$$\phi(\mathbf{x}): FS_{old} \longrightarrow FS_{new}$$

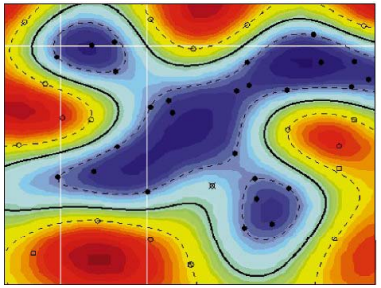
Dual optimization problem:

Maximize $L(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \cdot \alpha_j \cdot y_i \cdot y_j \cdot \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle$

subject to $\sum_{i=1}^{n} \alpha_i \cdot y_i = 0$ and $0 \leq \alpha_i \leq C$

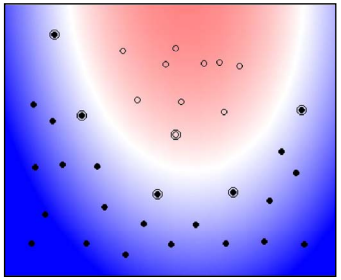
- Feature transform ϕ only affects the scalar product of training vectors
- Kernel K is a function: $K_\phi(\mathbf{x}_i, \mathbf{x}_j) = \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle$

Kernel Machines: Examples



Radial basis kernel

$$K(\mathbf{x},\mathbf{y}) = \exp\left(-\gamma \cdot |\mathbf{x}-\mathbf{y}|^2\right)$$



Polynomial kernel (degree 2)

$$K(\mathbf{x},\mathbf{y}) = \left(\langle \mathbf{x},\mathbf{y}\rangle + 1\right)^d$$

Support Vector Machines: Discussion

+ generate classifiers with a high classification accuracy
+ relatively weak tendency to overfitting (generalization theory)
+ efficient classification of new objects
+ compact models

– training times may be long (appropriate feature space may be very high-dimensional)
– expensive implementation
– resulting models rarely provide an intuition

Chapter 4 – Conclusions

- Classification is an extensively studied problem (mainly in statistics, machine learning & neural networks)
- Classification is probably one of the most widely used data mining techniques with a lot of extensions
- Scalability is still an important issue for database applications: thus combining classification with database techniques should be a promising topic
- Research directions: classification of non-relational data, e.g., text, spatial, multimedia, etc.
- Example: kNN-classifiers rely on distances but to not require vector representations of data

References (I)

- C. Apte and S. Weiss. Data mining with decision trees and decision rules. Future Generation Computer Systems, 13, 1997.
- L. Breiman, J. Friedman, R. Olshen, and C. Stone. Classification and Regression Trees. Wadsworth International Group, 1984.
- P. K. Chan and S. J. Stolfo. Learning arbiter and combiner trees from partitioned data for scaling machine learning. In Proc. 1st Int. Conf. Knowledge Discovery and Data Mining (KDD'95), pages 39-44, Montreal, Canada, August 1995.
- U. M. Fayyad. Branching on attribute values in decision tree generation. In Proc. 1994 AAAI Conf., pages 601-606, AAAI Press, 1994.
- J. Gehrke, R. Ramakrishnan, and V. Ganti. Rainforest: A framework for fast decision tree construction of large datasets. In Proc. 1998 Int. Conf. Very Large Data Bases, pages 416-427, New York, NY, August 1998.
- T. Joachims: Learning to Classify Text using Support Vector Machines. Kluwer, 2002.
- M. Kamber, L. Winstone, W. Gong, S. Cheng, and J. Han. Generalization and decision tree induction: Efficient classification in data mining. In Proc. 1997 Int. Workshop Research Issues on Data Engineering (RIDE'97), pages 111-120, Birmingham, England, April 1997.

References (II)

- J. Magidson. The Chaid approach to segmentation modeling: Chi-squared automatic interaction detection. In R. P. Bagozzi, editor, Advanced Methods of Marketing Research, pages 118-159. Blackwell Business, Cambridge Massechusetts, 1994.
- M. Mehta, R. Agrawal, and J. Rissanen. SLIQ : A fast scalable classifier for data mining. In Proc. 1996 Int. Conf. Extending Database Technology (EDBT'96), Avignon, France, March 1996.
- S. K. Murthy, Automatic Construction of Decision Trees from Data: A Multi-Diciplinary Survey, Data Mining and Knowledge Discovery 2(4): 345-389, 1998
- J. R. Quinlan. Bagging, boosting, and c4.5. In Proc. 13th Natl. Conf. on Artificial Intelligence (AAAI'96), 725-730, Portland, OR, Aug. 1996.
- R. Rastogi and K. Shim. Public: A decision tree classifer that integrates building and pruning. In Proc. 1998 Int. Conf. Very Large Data Bases, 404-415, New York, NY, August 1998.
- J. Shafer, R. Agrawal, and M. Mehta. SPRINT: A scalable parallel classifier for data mining. In Proc. 1996 Int. Conf. Very Large Data Bases, 544-555, Bombay, India, Sept. 1996.
- S. M. Weiss and C. A. Kulikowski. Computer Systems that Learn: Classification and Prediction Methods from Statistics, Neural Nets, Machine Learning, and Expert Systems. Morgan Kaufman, 1991.

Data Mining Algorithms

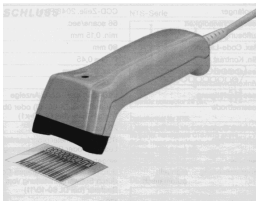
Lecture Course with Tutorials
Summer 2007

Chapter 5: Mining Association Rules

Chapter 5: Mining Association Rules

Example: Basket Data Analysis

- Transaction database
  - {butter, bread, milk, sugar}
  - {butter, flour, milk, sugar}
  - {butter, eggs, milk, salt}
  - {eggs}
  - {butter, flour, milk, salt, sugar}



- Question of interest:
  - Which items are bought together frequently?
- Applications
  - Improved store layout
  - Cross marketing
  - Focused attached mailings / add-on sales

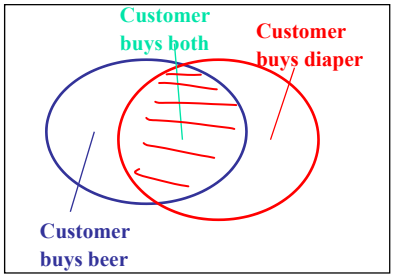
What Is Association Mining?

- Association rule mining
  - Finding frequent patterns, associations, correlations, or causal structures among sets of items or objects in transaction databases, relational databases, and other information repositories.
  - Rule form: "Body ⇒ Head [support, confidence]"
- Applications
  - Basket data analysis, cross-marketing, catalog design, loss-leader analysis, clustering, classification, etc.
- Examples
  - buys(x, "diapers") ⇒ buys(x, "beers") [0.5%, 60%]
  - major(x, "CS") ^ takes(x, "DB") ⇒ grade(x, "A") [1%, 75%]

Association Rule: Basic Concepts

- Given: (1) database of transactions, (2) each transaction is a list of items (purchased by a customer in a visit)
- Find: all rules that correlate the presence of one set of items with that of another set of items
  - E.g., *98% of people who purchase tires and auto accessories also get automotive services done*
- Applications
  - ** ⇒ Maintenance Agreement* (What the store should do to boost Maintenance Agreement sales)
  - *Home Electronics ⇒ ** (What other products should the store stocks up?)
  - Attached mailing in direct marketing

Rule Measures: Support and Confidence



Find all the rules $X \& Y \Rightarrow Z$ with minimum confidence and support

- support, *s*, probability that a transaction contains {X, Y, Z}
- confidence, *c*, conditional probability that a transaction having {X, Y} also contains Z

Transaction ID	Items Bought
2000	A,B,C
1000	A,C
4000	A,D
5000	B,E,F

Let minimum support 50%, and minimum confidence 50%, then we have

- $A \Rightarrow C$ (50%, 66.6%)
- $C \Rightarrow A$ (50%, 100%)

Association Rule Mining: A Road Map

- *Boolean vs. quantitative associations* (Based on the types of values handled)
  - buys(x, "SQLServer") ^ buys(x, "DMBook") → buys(x, "DBMiner") [0.2%, 60%]
    - Short notation: SQLServer, DMBook ⇒ DBMiner [0.2%, 60%]
  - age(x, "30..39") ^ income(x, "42..48K") → buys(x, "PC") [1%, 75%]
- *Single dimension vs. multiple dimensional associations*
- *Single level vs. multiple-level analysis*
  - What *brands* of beers are associated with what *brands* of diapers?
- *Various extensions*
  - Correlation, causality analysis
    - Association does not necessarily imply correlation or causality
  - Maxpatterns and closed itemsets
  - Constraints enforced
    - E.g., small sales (sum < 100) trigger big buys (sum > 1,000)?

Chapter 5: Mining Association Rules

- Introduction
  - Transaction databases, market basket data analysis
- Simple Association Rules
  - Basic notions, apriori algorithm, hash trees, FP-tree, interestingness
- Hierarchical Association Rules
  - Motivation, notions, algorithms, interestingness
- Extensions and Summary

Simple Association Rules: Basic Notions

- *Items* $I = \{i_1, ..., i_m\}$: a set of literals (denoting items)
- *Itemset* X: Set of items $X \subseteq I$
- *Database* D: Set of *transactions* T, each transaction is a set of items $T \subseteq I$
- Transaction T *contains* an itemset X: $X \subseteq T$
- The items in transactions and itemsets are sorted lexicographically:
  - itemset $X = (x_1, x_2, ..., x_k)$, where $x_1 \leq x_2 \leq ... \leq x_k$
- *Length* of an itemset: number of elements in the itemset
- *k-itemset:* itemset of length k
- *Support* of an itemset X: fraction of transactions in D that contain X:
  - support(X) = count ($\{T, X \subseteq T\}$) / count (D)
- *Association rule:* Rule $X \Rightarrow Y$, where $X \subseteq I$, $Y \subseteq I$ and $X \cap Y = \emptyset$

Simple Association Rules: Basics (2)

- *Support s* of an association rule $X \Rightarrow Y$ in D
  - Support of itemsets covering $X \cup Y$ in D
  - Support ($X \Rightarrow Y$, D) = count ($\{T, X \cup Y \subseteq T\}$) / count ($D$)
- *Confidence c* of an association rule $X \Rightarrow Y$ in D
  - Fraction of transactions which contain the itemset Y from the subset of transactions from D which contain the itemset X
  - Conf ($X \Rightarrow Y$, D) = count ($\{T, X \cup Y \subseteq T\}$) / count ($\{T, X \subseteq T\}$)
  - Conf ($X \Rightarrow Y$, D) = support ($X \cup Y$) / support (X)
- Task of mining association rules
  - Given a database D, determine all association rules having a support $\geq$ *minSup* and confidence $\geq$ *minConf*

Mining Association Rules—Example

Transaction ID	Items Bought
2000	A,B,C
1000	A,C
4000	A,D
5000	B,E,F

Min. support 50%
Min. confidence 50%

Frequent Itemset	Support
{A}	75%
{B}	50%
{C}	50%
{A,C}	50%

For rule $A \Rightarrow C$:

support = support($\{A, C\}$) = 50%

confidence = support($\{A, C\}$) / support($\{A\}$) = 66.6%

Mining Association Rules: Key Steps

- Step 1: Find *frequent itemsets*, i.e. sets of items that have at least a minimum support
  - A subset of a frequent itemset must also be a frequent itemset
    - i.e., if $\{A,B\}$ is a frequent itemset, both $\{A\}$ and $\{B\}$ must be frequent itemsets
  - Iteratively find frequent itemsets with cardinality from 1 to k
- Step 2: Use the *frequent itemsets* $\{A,B,...,Z\}$ to generate association rules, e.g. $A,B,... \Rightarrow Z$ or $A,Z,... \Rightarrow B,C$
  - n frequent items yield $2^n - 2$ association rules

Mining Frequent Itemsets: Basic Idea

- Naïve Algorithm
  - count the frequency of for all possible subsets of I in the database
  - *too expensive* since there are 2^m such itemsets for |I| = m items
- The ***Apriori*** principle (monotonicity):
  Any subset of a frequent itemset must be frequent
- Method based on the apriori principle
  - First count the 1-itemsets, then the 2-itemsets, then the 3-itemsets, and so on
  - When counting (k+1)-itemsets, only consider those (k+1)-itemsets where all subsets of length k have been determined as frequent in the previous step

The Apriori Algorithm

variable C_k: candidate itemsets of size k
variable L_k: frequent itemsets of size k

L_1 = {frequent items}
for (k = 1; L_k !=∅; k++) **do begin**
  // JOIN STEP: join L_k with itself to produce C_{k+1}
  // PRUNE STEP: discard (k+1)-itemsets from C_{k+1} that contain non-frequent k-itemsets as subsets
  C_{k+1} = candidates generated from L_k

  for each transaction t in database do
    Increment the count of all candidates in C_{k+1} that are contained in t
  L_{k+1} = candidates in C_{k+1} with min_support
  end
return $\cup_k L_k$

Generating Candidates (Join Step)

- Requirements for candidate k-itemsets C_k
  - Must contain all frequent k-itemsets (superset property $C_k \supseteq L_k$)
  - Significantly smaller than the set of all k-subsets
  - Suppose the items are sorted by any order (e.g., lexicograph.)
- Step 1: Joining
  - Consider frequent ($k-1$)-itemsets p and q
  - p and q are joined if they share the same first $k-2$ items

insert into C_k
select $p.i_1, p.i_2, \ldots, p.i_{k-2}, p.i_{k-1}, q.i_{k-1}$
from L_{k-1} p, L_{k-1} q
where $p.i_1 = q.i_1, \ldots, p.i_{k-2} = q.i_{k-2}, p.i_{k-1} < q.i_{k-1}$

$p \in L_{k-1}$ (1, 2, 3)
(1, 2, 3, 4) $\in C_k$
$q \in L_{k-1}$ (1, 2, 4)

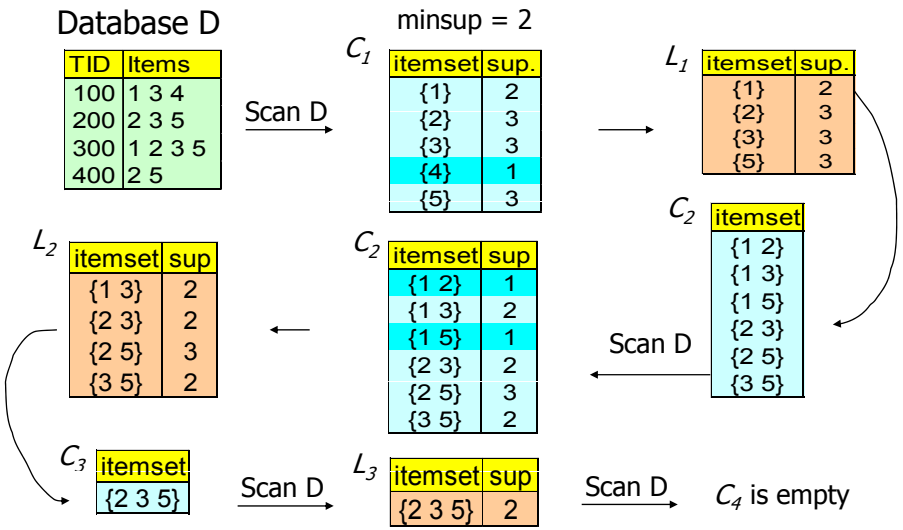
Generating Candidates (Prune Step)

- Step 2: Pruning
  - Remove candidate k-itemsets which contain a non-frequent ($k-1$)-subset s, i.e., $s \notin L_{k-1}$
  - **forall** *itemsets* c ***in*** C_k **do**
    forall *(k–1)-subsets* s ***of*** c **do**
      if *(s is not in L_{k-1})* **then delete** c **from** C_k
- Example 1
  - L_3 = {(1 2 3), (1 2 4), (1 3 4), (1 3 5), (2 3 4)}
  - Candidates after the join step: {(1 2 3 4), (1 3 4 5)}
  - In the pruning step: delete (1 3 4 5) because (3 4 5) $\notin L_3$, i.e., (3 4 5) is not a frequent 3-itemset; also (1 4 5) $\notin L_3$
  - C_4 = {(1 2 3 4)}

Generating Candidates – Example 2

- $L_3 = \{abc, abd, acd, ace, bcd\}$
- Self-joining: $L_3 * L_3$
  - *abcd* from *abc* and *abd*
  - *acde* from *acd* and *ace*
- Pruning:
  - *abcd* is ok: abc, abd, acd, bcd are in L_3
  - *acde* is removed because *ade* (and *cde*) is not in L_3
- $C_4 = \{abcd\}$

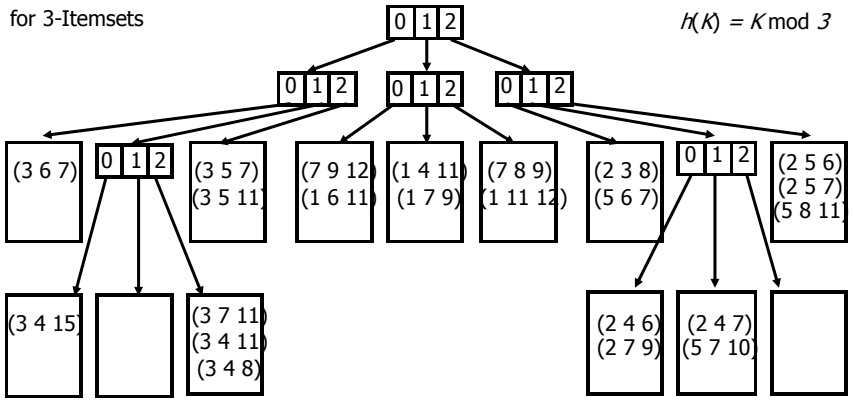
Generating Candidates – Full Example



How to Count Supports of Candidates?

- Why is counting supports of candidates a problem?
  - The total number of candidates can be very huge
  - One transaction may contain many candidates
- Method
  - Candidate itemsets are stored in a *hash-tree*
  - *Leaf* nodes of hash-tree contain lists of itemsets and their support (i.e., counts)
  - *Interior* nodes contain hash tables
  - *Subset function* finds all the candidates contained in a transaction

Hash-Tree – Example



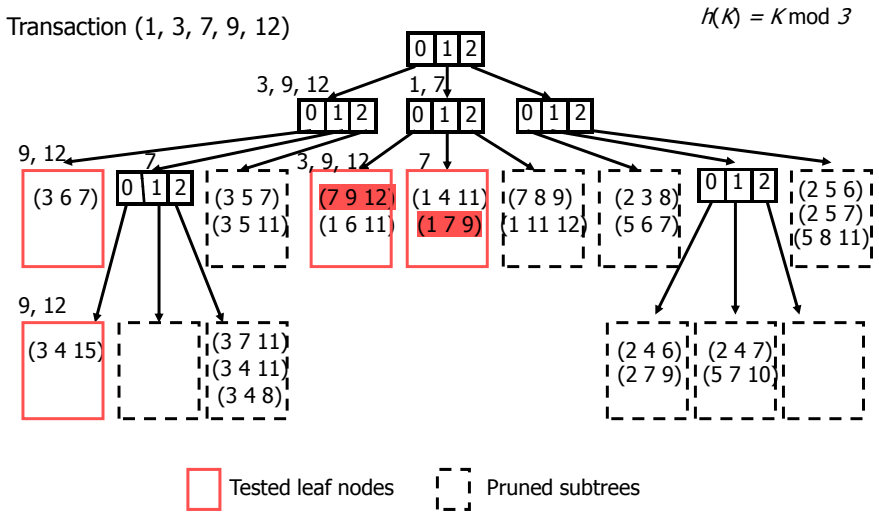
Hash-Tree – Construction

- Searching for an itemset
  - Start at the root
  - At level *d:* apply the hash function *h* to the *d*-th item in the itemset
- Insertion of an itemset
  - search for the corresponding leaf node, and insert the itemset into that leaf
  - if an overflow occurs:
    - Transform the leaf node into an internal node
    - Distribute the entries to the new leaf nodes according to the hash function

Hash-Tree – Counting

- Search all candidate itemsets contained in a transaction $T = (t_1\ t_2\ \ldots\ t_n)$
- At the root
  - Determine the hash values for each item $t_1\ t_2\ \ldots\ t_{n-k}$ in T
  - Continue the search in the resulting child nodes
- At an internal node at level d (reached after hashing of item t_i)
  - Determine the hash values and continue the search for each item t_j with $j > i$ and $j \leq n-k+i$
- At a leaf node
  - Check whether the itemsets in the leaf node are contained in transaction T

Counting Itemsets in a Transaction using a Hash-Tree – Example



Methods to Improve Apriori's Efficiency (1)

- Hash-based itemset counting [Park, Chen & Yu '95]
  - Manage support counts by using a hash table (not a hash tree), i.e. several k-itemsets share the same hashing bucket in table H_k.
  - A k-itemset whose corresponding hashing bucket count is below the threshold cannot be frequent
  - More efficient access to candidates, but less accurate counting
- Transaction reduction [Agrawal & Srikant '94]
  - A transaction that does not contain any frequent k-itemset is useless in subsequent scans
  - Remove these transactions from the database
  - More efficient database scans (read accesses) but several write accesses to the database

Methods to Improve Apriori's Efficiency (2)

- Partitioning [Savasere, Omiecinski & Navathe '95]
  - Any itemset that is potentially frequent in DB must be relatively frequent in at least one of the partitions of DB (*minsup* / #partitions)
  - Process the database partition by partition in main memory
  - More efficient for partitions but expensive combination of partial results
- Sampling [Toivonen '96]
  - Mining on a subset of given data, lower support threshold + a method to determine the completeness
  - Apply the apriori algorithm to a subset of the database
  - Count support of the resulting frequent itemsets on the entire database
  - Potentially, find new candidates and count them on the database
- Dynamic itemset counting
  - add new candidate itemsets only when all of their subsets are estimated to be frequent

Generating Rules from Frequent Itemsets

- For each frequent itemset X
  - For each subset A of X, form a rule $A \Rightarrow (X - A)$
  - Delete those rules that do not have minimum confidence
- Computation of the confidence of a rule $A \Rightarrow (X - A)$

$$confidence(A \Rightarrow (X - A)) = \frac{support(X)}{support(A)}$$

- Store the frequent itemsets and their support in a hash table in main memory → no additional database access

itemset	support
{A}	2
{B}	4
{C}	5
{A, B}	3
{A, C}	2
{B, C}	4
{A, B, C}	2

- Example: X = {A, B, C}, minConf=60%
  - conf (A ⇒ B, C) = 1; conf (B, C ⇒ A) = 1/2
  - conf (B ⇒ A, C) = 1/2; conf (A, C ⇒ B) = 1
  - conf (C ⇒ A, B) = 2/5; conf (A, B ⇒ C) = 2/3

Is Apriori Fast Enough? — Performance Bottlenecks

- The core of the Apriori algorithm:
  - Use frequent ($k-1$)-itemsets to generate candidate frequent k-itemsets
  - Use database scan and pattern matching to collect counts for the candidate itemsets
- The bottleneck of *Apriori*: candidate generation
  - Huge candidate sets:
    - 10^4 frequent 1-itemsets will generate 10^7 candidate 2-itemsets
    - To discover a frequent pattern of size 100, e.g., $\{a_1, a_2, \ldots, a_{100}\}$, one needs to generate $2^{100} \approx 10^{30}$ candidates.
  - Multiple scans of database:
    - Needs n+1 scans, n is the length of the longest pattern

Mining Frequent Patterns Without Candidate Generation

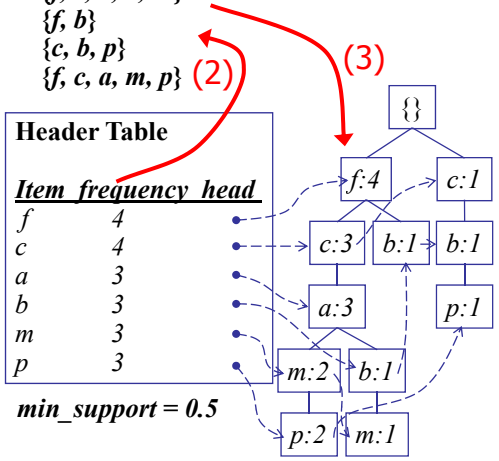
- Compress a large database into a compact, Frequent-Pattern tree (FP-tree) structure
  - highly condensed, but complete for frequent pattern mining
  - avoid costly database scans
- Develop an efficient, FP-tree-based frequent pattern mining method
  - A divide-and-conquer methodology: decompose mining tasks into smaller ones
  - Avoid candidate generation: sub-database test only!

Construct FP-tree from a Transaction DB

TID	Items bought	(ordered) frequent items
100	{f, a, c, d, g, i, m, p}	{f, c, a, m, p}
200	{a, b, c, f, l, m, o}	{f, c, a, b, m}
300	{b, f, h, j, o}	{f, b}
400	{b, c, k, s, p}	{c, b, p}
500	{a, f, c, e, l, p, m, n}	{f, c, a, m, p}

Steps:

1. Scan DB once, find frequent 1-itemsets (single items)
2. Order frequent items in frequency descending order
3. Scan DB again, construct FP-tree starting with most frequent item per transaction



Benefits of the FP-tree Structure

- Completeness:
  - never breaks a long pattern of any transaction
  - preserves complete information for frequent pattern mining
- Compactness
  - reduce irrelevant information—infrequent items are gone
  - frequency descending ordering: more frequent items are more likely to be shared
  - never be larger than the original database (if not count node-links and counts)
  - Experiments demonstrate compression ratios over 100

Mining Frequent Patterns Using FP-tree

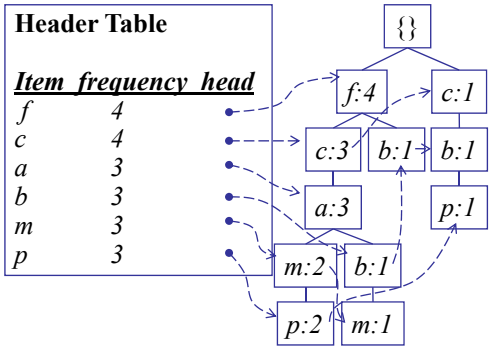
- General idea (divide-and-conquer)
  - Recursively grow frequent pattern path using the FP-tree
- Method
  - For each item, construct its conditional pattern-base (prefix paths), and then its conditional FP-tree
  - Repeat the process on each newly created conditional FP-tree
  - Until the resulting FP-tree is empty, or it contains only one path (single path will generate all the combinations of its sub-paths, each of which is a frequent pattern)

Major Steps to Mine FP-tree

1) Construct conditional pattern base for each node in the FP-tree
2) Construct conditional FP-tree from each conditional pattern-base
3) Recursively mine conditional FP-trees and grow frequent patterns obtained so far
   - If the conditional FP-tree contains a single path, simply enumerate all the patterns

Step 1: From FP-tree to Conditional Pattern Base

- Starting at the frequent header table in the FP-tree
- Traverse the FP-tree by following the link of each frequent item
- Accumulate all of transformed prefix paths of that item to form a conditional pattern base



***Conditional* pattern bases**

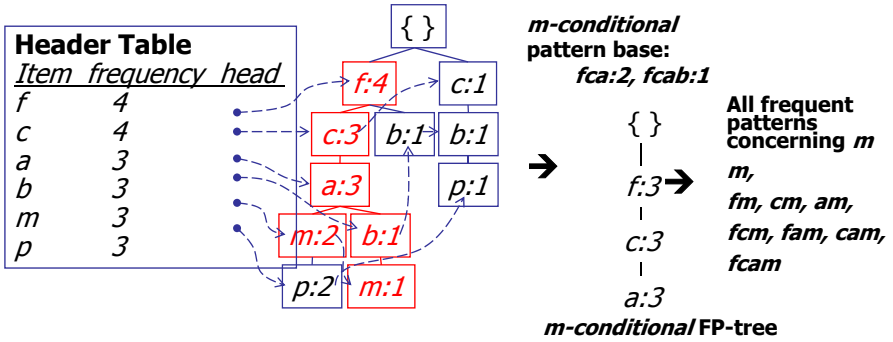
item	*cond. pattern base*
c	*f:3*
a	*fc:3*
b	*fca:1, f:1, c:1*
m	*fca:2, fcab:1*
p	*fcam:2, cb:1*

Properties of FP-tree for Conditional Pattern Base Construction

- Node-link property
  - For any frequent item a_i, all the possible frequent patterns that contain a_i can be obtained by following a_i's node-links, starting from a_i's head in the FP-tree header
- Prefix path property
  - To calculate the frequent patterns for a node a_i in a path P, only the prefix sub-path of a_i in P need to be accumulated, and its frequency count should carry the same count as node a_i.

Step 2: Construct Conditional FP-tree

- For each pattern-base
  - Accumulate the count for each item in the base
  - Construct the FP-tree for the frequent items of the pattern base

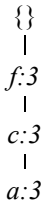


Mining Frequent Patterns by Creating Conditional Pattern-Bases

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

Step 3: Recursively mine the conditional FP-tree

***m*-conditional FP-tree:**



Conditional pattern base of "am": **(fc:3)**

***am*-conditional FP-tree**

Conditional pattern base of "cm": **(f:3)**

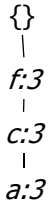
***cm*-conditional FP-tree**

Conditional pattern base of "cam": **(f:3)**

***cam*-conditional FP-tree**

Single FP-tree Path Generation

- Suppose an FP-tree T has a single path P
- The complete set of frequent pattern of T can be generated by enumeration of all the combinations of the sub-paths of P



➔

All frequent patterns concerning *m*

m,
fm, cm, am,
fcm, fam, cam,
fcam

***m*-conditional FP-tree**

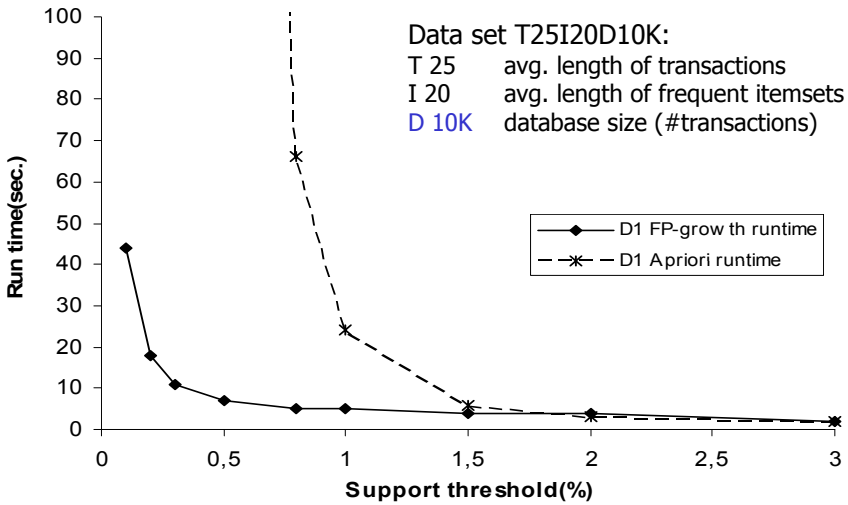
Principles of Frequent Pattern Growth

- Pattern growth property
  - Let α be a frequent itemset in DB, B be α's conditional pattern base, and β be an itemset in B. Then $\alpha \cup \beta$ is a frequent itemset in DB iff β is frequent in B.
- "*abcdef*" is a frequent pattern, if and only if
  - "*abcde*" is a frequent pattern, and
  - "*f*" is frequent in the set of transactions containing "*abcde*"

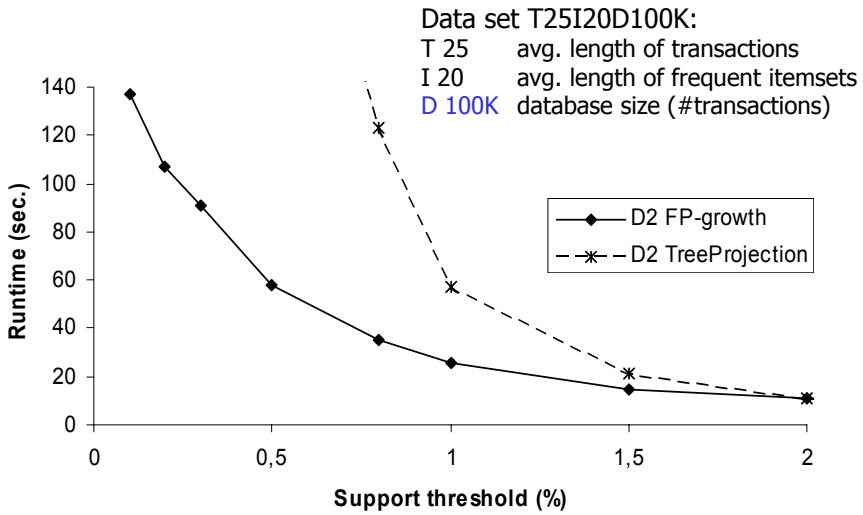
Why Is *Frequent Pattern Growth* Fast?

- Performance study in [Han, Pei&Yin '00] shows
  - FP-growth is an order of magnitude faster than Apriori, and is also faster than tree-projection
- Reasoning
  - No candidate generation, no candidate test
  - Use compact data structure
  - Eliminate repeated database scan
  - Basic operation is counting and FP-tree building

FP-growth vs. Apriori: Scalability With the Support Threshold



FP-growth vs. Tree-Projection: Scalability with Support Threshold



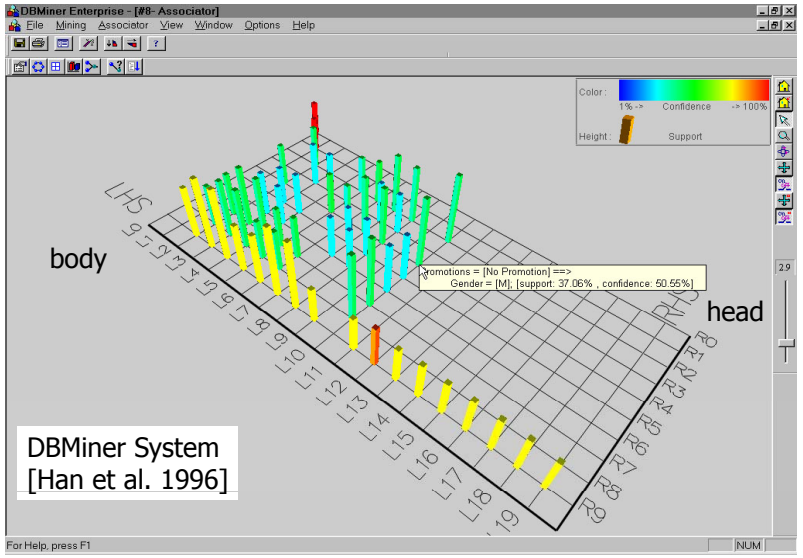
Presentation of Association Rules: Table Form

	Body	Implies	Head	Supp (%)	Conf (%)
1	cost(x) = '0.00~1000.00'	==>	revenue(x) = '0.00~500.00'	28.45	40.4
2	cost(x) = '0.00~1000.00'	==>	revenue(x) = '500.00~1000.00'	20.46	29.05
3	cost(x) = '0.00~1000.00'	==>	order_qty(x) = '0.00~100.00'	59.17	84.04
4	cost(x) = '0.00~1000.00'	==>	revenue(x) = '1000.00~1500.00'	10.45	14.84
5	cost(x) = '0.00~1000.00'	==>	region(x) = 'United States'	22.56	32.04
6	cost(x) = '1000.00~2000.00'	==>	order_qty(x) = '0.00~100.00'	12.91	69.34
7	order_qty(x) = '0.00~100.00'	==>	revenue(x) = '0.00~500.00'	28.45	34.54
8	order_qty(x) = '0.00~100.00'	==>	cost(x) = '1000.00~2000.00'	12.91	15.67
9	order_qty(x) = '0.00~100.00'	==>	region(x) = 'United States'	25.9	31.45
10	order_qty(x) = '0.00~100.00'	==>	cost(x) = '0.00~1000.00'	59.17	71.86
11	order_qty(x) = '0.00~100.00'	==>	product_line(x) = 'Tents'	13.52	16.42
12	order_qty(x) = '0.00~100.00'	==>	revenue(x) = '500.00~1000.00'	19.67	23.88
13	product_line(x) = 'Tents'	==>	order_qty(x) = '0.00~100.00'	13.52	98.72
14	region(x) = 'United States'	==>	order_qty(x) = '0.00~100.00'	25.9	81.94
15	region(x) = 'United States'	==>	cost(x) = '0.00~1000.00'	22.56	71.39
16	revenue(x) = '0.00~500.00'	==>	cost(x) = '0.00~1000.00'	28.45	100
17	revenue(x) = '0.00~500.00'	==>	order_qty(x) = '0.00~100.00'	28.45	100
18	revenue(x) = '1000.00~1500.00'	==>	cost(x) = '0.00~1000.00'	10.45	96.75
19	revenue(x) = '500.00~1000.00'	==>	cost(x) = '0.00~1000.00'	20.46	100
20	revenue(x) = '500.00~1000.00'	==>	order_qty(x) = '0.00~100.00'	19.67	96.14
21					
22					
23	cost(x) = '0.00~1000.00'	==>	revenue(x) = '0.00~500.00' AND order_qty(x) = '0.00~100.00'	28.45	40.4
24	cost(x) = '0.00~1000.00'	==>	revenue(x) = '0.00~500.00' AND order_qty(x) = '0.00~100.00'	28.45	40.4
25	cost(x) = '0.00~1000.00'	==>	revenue(x) = '500.00~1000.00' AND order_qty(x) = '0.00~100.00'	19.67	27.93
26	cost(x) = '0.00~1000.00'	==>	revenue(x) = '500.00~1000.00' AND order_qty(x) = '0.00~100.00'	19.67	27.93
27	cost(x) = '0.00~1000.00' AND order_qty(x) = '0.00~100.00'	==>	revenue(x) = '500.00~1000.00'	19.67	33.23

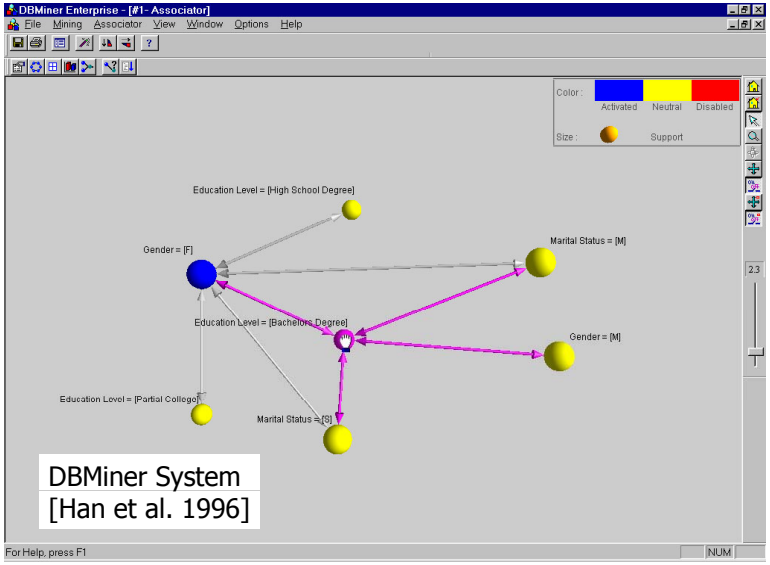
Sheet1

DBMiner System
[Han et al. 1996]

Presentation of Association Rules: Chart Form



Presentation of Association Rules: Rule Graph



DBMiner System
[Han et al. 1996]

Iceberg Queries

- Iceberg query: Compute aggregates over one attribute or a set of attributes only for those whose aggregate values is above certain threshold
- Example:

```
select P.custID, P.itemID, sum(P.qty)
from purchase P
group by P.custID, P.itemID
having sum(P.qty) >= 10
```

- Compute iceberg queries efficiently by Apriori:
  - First compute lower dimensions
  - Then compute higher dimensions only when all the lower ones are above the threshold

Interestingness Measurements

- *Objective* measures
  - Two popular measurements:
  - support and
  - confidence
- *Subjective* measures [Silberschatz & Tuzhilin, KDD95]
  - A rule (pattern) is interesting if it is
  - unexpected (surprising to the user) and/or
  - actionable (the user can do something with it)

Criticism to Support and Confidence

- Example 1 [Aggarwal & Yu, PODS98]
  - Among 5000 students
    - 3000 play basketball (=60%)
    - 3750 eat cereal (=75%)
    - 2000 both play basket ball and eat cereal (=40%)
  - Rule *play basketball* $\Rightarrow$ *eat cereal* [40%, 66.7%] is misleading because the overall percentage of students eating cereal is 75% which is higher than 66.7%
  - Rule *play basketball* $\Rightarrow$ *not eat cereal* [20%, 33.3%] is far more accurate, although with lower support and confidence
  - Observation: *play basketball* and *eat cereal* are *negatively correlated*

Interestingness of Association Rules

- Goal: Delete misleading association rules
- Condition for a rule A ⇒ B

$$\frac{P(A \cup B)}{P(A)} > P(B) + d$$ for a suitable threshold d > 0

- Measure for the interestingness of a rule

$$\frac{P(A \cup B)}{P(A)} - P(B)$$

  - The larger the value, the more interesting the relation between A and B, expressed by the rule.
- Other measures: correlation between A and B → $\frac{P(A \cup B)}{P(A)P(B)}$

Criticism to Support and Confidence: Correlation of Itemsets

- Example 2

X	1	1	1	1	0	0	0	0
Y	1	1	0	0	0	0	0	0
Z	0	1	1	1	1	1	1	1

Rule	Support	Confidence
X=>Y	25%	50%
X=>Z	37.50%	75%

  - X and Y: positively correlated
  - X and Z: negatively related
  - support and confidence of X=>Z dominates
- We need a measure of dependent or correlated events

$$corr_{A,B} = \frac{P(A \cup B)}{P(A)P(B)}$$

- P(B|A)/P(B) is also called the lift of rule A => B

Other Interestingness Measures: Interest

- *Interest* (*correlation, lift*): $\frac{P(A \cup B)}{P(A)P(B)}$
  - taking both P(A) and P(B) in consideration
  - Correlation equals 1, i.e. P(A ∪ B) = P(B) · P(A), if A and B are independent events
  - A and B negatively correlated, if the value is less than 1; otherwise A and B positively correlated

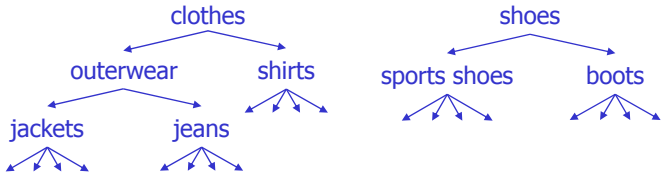
X	1	1	1	1	0	0	0	0
Y	1	1	0	0	0	0	0	0
Z	0	1	1	1	1	1	1	1

Itemset	Support	Interest
X,Y	25%	2
X,Z	37.50%	0.9
Y,Z	12.50%	0.57

Chapter 5: Mining Association Rules

Hierarchical Association Rules: Motivation

- Problem of association rules in plain itemsets
  - *High minsup:* apriori finds only few rules
  - *Low minsup:* apriori finds unmanagably many rules
- Exploit item taxonomies (generalizations, *is-a* hierarchies) which exist in many applications



- Task: find association rules between generalized items
- Support for sets of item types (e.g., product groups) is higher than support for sets of individual items

Hierarchical Association Rules: Motivating Example

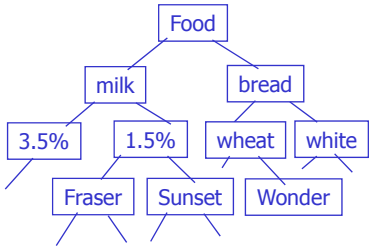
- Examples

jeans	⇒ boots	}	Support < minsup
jackets	⇒ boots		
outerwear	⇒ boots		Support > minsup

- Characteristics
  - Support("outerwear ⇒ boots") is not necessarily equal to the sum support("jackets ⇒ boots") + support("jeans ⇒ boots")
  - If the support of rule "outerwear ⇒ boots" exceeds minsup, then the support of rule "clothes ⇒ boots" does, too

Mining Multi-Level Associations

- Example generalization hierarchy:



- A top_down, progressive deepening approach:
  - First find high-level strong rules:
    - *milk* → *bread* [20%, 60%].
  - Then find their lower-level "weaker" rules:
    - 1.5% *milk* → *wheat bread* [6%, 50%].
- Variations at mining multiple-level association rules.
  - Level-crossed association rules:
    - 1.5 % *milk* → *Wonder wheat bread*
  - Association rules with multiple, alternative hierarchies:
    - 1.5 % *milk* → *Wonder bread*

Hierarchical Association Rules: Basic Notions

- [Srikant & Agrawal 1995]
- Let $U = \{i_1, ..., i_m\}$ be a universe of literals called items (basic items as well as generalized items)
- Let *h* be a directed acyclic graph defined as follows:
  - The universe of literals *U* forms the set of vertices in *h*
  - A pair *(i, j)* forms an edge in *h* if *i* is a *generalization* of *j*
    - *i* is called *parent* or *direct ancestor* of *j*
    - *j* is called a *child* or a *direct descendant* of *i*
- x' is an *ancestor* of x and, thus, x is a *descendant* of x' wrt. *h,* if there is a path from x' to x in *h*
- A set of items z' is called an *ancestor* of a set of items z if at least one item in z' is an ancestor of an item in z

Hierarchical Association Rules: Basic Notions (2)

- Let D be a set of transaction T with $T \subseteq U$
  - Typically, transactions T in D only contain items from the leaves of graph h
- A transaction T *supports an item* $i \in U$ if i or any descendant of i is contained in T
- A transaction T *supports a set* $X \subseteq U$ of items if T supports each item in X

- Support of a set $X \subseteq U$ of items in D:
  - Percentage of transactions in D that contain X as a subset

Hierarchical Association Rules: Basic Notions (3)

- Hierarchical association rule
  - $X \Rightarrow Y$ with $X \subseteq U$, $Y \subseteq U$, $X \cap Y = \emptyset$
  - No item in Y is ancestor of an item in X wrt. H (i.e., avoid rules $X \Rightarrow$ ancestor(X) where always conf. = 100%)

- Support of a hierarchical association rule $X \Rightarrow Y$ in D:
  - Support of the set $X \cup Y$ in D

- Confidence of a hierarchical association rule $X \Rightarrow Y$ in D:
  - Percentage of transactions that support Y among the subset of transactions that support X

Hierarchical Association Rules: Example

transaction id	items
1	shirt
2	jacket, boots
3	jeans, boots
4	sports shoes
5	sports shoes
6	jacket

- Support of {clothes}: 4 of 6 = 67%
- Support of {clothes, boots}: 2 of 6 = 33%

- „shoes $\Rightarrow$ clothes“: support 33%, confidence 50%
- „boots $\Rightarrow$ clothes“: support 33%, confidence 100%

Determination of Frequent Itemsets: Basic Algorithm for Hierarchical Rules

- Idea: Extend the transactions in the database by all the ancestors of the items contained

- Method:
  - For all transactions t in the database
    - Create an empty new transaction t'
    - For each item i in t, insert i and all its ancestors wrt. h in t'
    - Avoid inserting duplicates
  - Based on the new transactions t‘, find frequent itemsets for simple association rules (e.g., by using the apriori algorithm)

Determination of Frequent Itemsets: Optimization of Basic Algorithm

- Precomputation of ancestors
  - Additional data structure that holds the association of each item to the list of its ancestors: item $\rightarrow$ list of successors
  - supports a more efficient access to the ancestors of an item
- Filtering of new ancestors
  - Add only ancestors to a transaction which occur in an element of the candidate set C_k of the current iteration
  - Example
    - C_k = {{clothes, shoes}}
    - Substitute „jacketABC" by „clothes"

Determination of Frequent Itemsets: Optimization of Basic Algorithm (2)

- Algorithm *Cumulate:* Exclude redundant itemsets
  - Let X be a *k*-itemset, *i* an item and *i'* an ancestor of *i*
  - X = {*i*, *i'*, …}
  - Support of X – {*i'*} = support of X
  - When generating candidates, X can be excluded
  - *k*-itemsets that contain an item *i* and an ancestor *i'* of *i* as well are not counted

Multi-level Association: Redundancy Filtering

- Some rules may be redundant due to "ancestor" relationships between items.
- Example
  - milk $\Rightarrow$ wheat bread [support = 8%, confidence = 70%]
  - 1.5% milk $\Rightarrow$ wheat bread [support = 2%, confidence = 72%]
- We say the first rule is an ancestor of the second rule.
- A rule is redundant if its support is close to the "expected" value, based on the rule's ancestor.

Multi-Level Mining: Progressive Deepening

- A top-down, progressive deepening approach:
  - First mine high-level frequent items:
    - milk (15%), bread (10%)
  - Then mine their lower-level "weaker" frequent itemsets:
    - 1.5% milk (5%), wheat bread (4%)
- Different min_support threshold across multi-levels lead to different algorithms:
  - If adopting the same min_support across multi-levels
    - toss t if any of t's ancestors is infrequent.
  - If adopting reduced min_support at lower levels
    - then examine only those descendents whose ancestor's support is frequent/non-negligible.

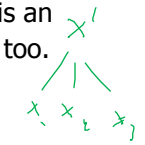
Progressive Refinement of Data Mining Quality

- Why progressive refinement?
  - Mining operator can be expensive or cheap, fine or rough
  - Trade speed with quality: step-by-step refinement.
- Superset coverage property:
  - Preserve all the positive answers—allow a false positive test but not a false negative test.
- Two- or multi-step mining:
  - First apply rough/cheap operator (superset coverage)
  - Then apply expensive algorithm on a substantially reduced candidate set (Koperski & Han, SSD'95).

Determination of Frequent Itemsets: Stratification

- Alternative to basic algorithm (i.e., to apriori algorithm)
- *Stratification:* build layers from the sets of itemsets

- Basic observation
  - If itemset X´ does not have minimum support, and X´ is an ancestor of X, then X does not have minimum support, too.



- Method
  - For a given *k,* do not count all *k*-itemsets simultaneously
  - Instead, count the more general itemsets first, and count the more specialized itemsets only when required

Determination of Frequent Itemsets: Stratification (2)

- Example
  - C_k = {{clothes, shoes}, {outerwear, shoes}, {jackets, shoes}}
  - First, count the support for {clothes, shoes}
  - Only if support exceeds minsup, count the support for {outerwear, shoes}

- Notions
  - *Depth* of an itemset
    - For itemsets X from a candidate set C_k without direct ancestors in C_k: depth(X) = 0
    - For all other itemsets X in C_k: depth(X) = 1 + max {depth(X'), X'$\in C_k$ is a parent of X}
  - (C_k^n): set of itemsets of depth n from C_k, $0 \le n \le$ maxdepth t

Determination of Frequent Itemsets: Algorithm Stratify

- Method
  - Count the itemsets from C_k^0
  - Remove all descendants of elements from (C_k^0) that do not have minimum support
    - Count the remaining elements in (C_k^1)
    - ...

- Trade-off between number of itemsets for which support is counted simultaneously and number of database scans

- If $|C_k^n|$ is small, then count candidates of depth (n, n+1, ..., t) at once

Determination of Frequent Itemsets: Stratification – Problems

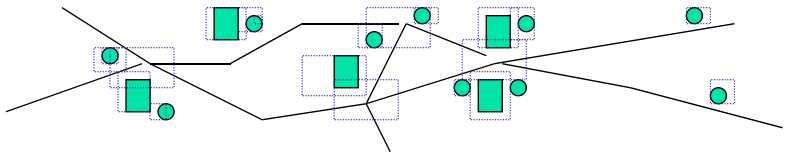
- Problem of algorithm Stratify
  - If many itemsets with small depth share the minimum support, only few itemsets of a higher depth are excluded
- Improvements of algorithm Stratify
  - Estimate the support of all itemsets in C_k by using a sample
  - Let C_k` be the set of all itemsets for which the sample suggests that all or at least all their ancestors in C_k share the minimum support
  - Determine the actual support of the itemsets in C_k` by a single database scan
  - Remove all descendants of elements in C_k` that have a support below the minimum support from the set C_k`` = C_k – C_k`
  - Determine the support of the remaining itemsets in C_k`` in a second database scan

Determination of Frequent Itemsets: Stratification – Experiments

- Test data
  - Supermarket data
    - 548,000 items; item hierarchy with 4 levels; 1.5M transactions
  - Department store data
    - 228,000 items; item hierarchy with 7 levels; 570,000 transactions
- Results
  - Optimizations of algorithms cumulate and stratify can be combined
  - *cumulate* optimizations yield a strong efficiency improvement
  - *Stratification* yields a small additional benefit only

Progressive Refinement Mining of Spatial Association Rules

- Hierarchy of spatial relationship:
  - "g_close_to": near_by, touch, intersect, contain, etc.
  - First search for rough relationship and then refine it.
- Two-step mining of spatial association:
  - Step 1: rough spatial computation (as a filter)
    - Using MBR or R-tree for rough estimation.
  - Step2: Detailed spatial algorithm (as refinement)
    - Apply only to those objects which have passed the rough spatial association test (no less than *min_support*)



Interestingness of Hierarchical Association Rules – Notions

clothes ⇒ shoes
:
jackets ⇒ shoes

- Rule X´ ⇒ Y´ is an *ancestor* of rule X ⇒ Y if:
  - Itemset X´ is an ancestor of itemset X or itemset Y´ is an ancestor of itemset Y
- Rule X´ ⇒ Y´ is a *direct ancestor* of rule X ⇒ Y in a set of rules if:
  - Rule X´ ⇒ Y` is an ancestor of rule X ⇒ Y, and
  - There is no rule X`` ⇒ Y`` such that X`` ⇒ Y`` is an ancestor of X ⇒ Y and X´ ⇒ Y´ is an ancestor of X`` ⇒ Y``
- A hierarchical association rule X ⇒ Y is called *R-interesting* if:
  - There are no direct ancestors of X ⇒ Y or
  - Actual support is larger than *R* times the expected support or
  - Actual confidence is larger than *R* times the expected confidence

Interestingness of Hierarchical Association Rules – Example

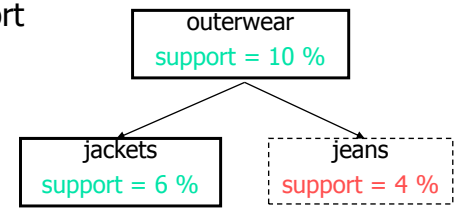
- Example
  - Let $R = 2$

Item	Support
clothes	20
outerwear	10
jackets	4

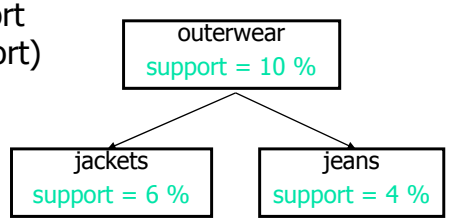
No.	rule	support	*R*-interesting?
1	clothes ⇒ shoes	10	yes: no ancestors
2	outerwear ⇒ shoes	9	yes: Support >> *R* * expected support (wrt. rule 1)
3	jackets ⇒ shoes	4	no: Support < *R* * expected support (wrt. rule 2)

Hierarchical Association Rules – How to Choose Minimum Support?

- Uniform Support



- Reduced Support (Variable Support)



Chapter 5: Mining Association Rules

- Introduction
  - Transaction databases, market basket data analysis
- Simple Association Rules
  - Basic notions, apriori algorithm, hash trees, FP-tree, interestingness
- Hierarchical Association Rules
  - Motivation, notions, algorithms, interestingness
- Extensions and Summary

Multi-Dimensional Association: Concepts

- Single-dimensional rules:
  - buys(X, "milk") ⇒ buys(X, "bread")
- Multi-dimensional rules: ≥ 2 dimensions or predicates
  - Inter-dimension association rules (*no repeated predicates*)
    - age(X,"19-25") ∧ occupation(X,"student") ⇒ buys(X,"coke")
  - hybrid-dimension association rules (*repeated predicates*)
    - age(X,"19-25") ∧ buys(X, "popcorn") ⇒ buys(X, "coke")
- Categorical Attributes
  - finite number of possible values, no ordering among values
- Quantitative Attributes
  - numeric, implicit ordering among values

Techniques for Mining Multi-Dimensional Associations

- Search for frequent *k*-predicate set:
  - Example: {age, occupation, buys} is a 3-predicate set.
  - Techniques can be categorized by how age are treated.

1. Using static discretization of quantitative attributes
   - Quantitative attributes are statically discretized by using predefined concept hierarchies.
2. Quantitative association rules
   - Quantitative attributes are dynamically discretized into "bins"based on the distribution of the data.
3. Distance-based association rules
   - This is a dynamic discretization process that considers the distance between data points.

Why Is the Big Pie Still There?

- From association to correlation and causal structure analysis
  - Association does not necessarily imply correlation or causal relationships
- From intra-transaction association to inter-transaction associations
  - E.g., break the barriers of transactions (Lu, et al. TOIS'99).
- From association analysis to classification and clustering analysis
  - E.g, clustering association rules
- Constraint-based mining of associations

Summary

- Association rule mining
  - probably the most significant contribution from the database community in KDD
  - A large number of papers have been published
- Many interesting issues have been explored
- An interesting research direction
  - Association analysis in other types of data: spatial data, multimedia data, time series data, etc.

References

- R. Agarwal, C. Aggarwal, and V. V. V. Prasad. A tree projection algorithm for generation of frequent itemsets. In Journal of Parallel and Distributed Computing (Special Issue on High Performance Data Mining), 2000.
- R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. SIGMOD'93, 207-216, Washington, D.C.
- R. Agrawal and R. Srikant. Fast algorithms for mining association rules. VLDB'94 487-499, Santiago, Chile.
- R. Agrawal and R. Srikant. Mining sequential patterns. ICDE'95, 3-14, Taipei, Taiwan.
- R. J. Bayardo. Efficiently mining long patterns from databases. SIGMOD'98, 85-93, Seattle, Washington.
- S. Brin, R. Motwani, and C. Silverstein. Beyond market basket: Generalizing association rules to correlations. SIGMOD'97, 265-276, Tucson, Arizona.
- S. Brin, R. Motwani, J. D. Ullman, and S. Tsur. Dynamic itemset counting and implication rules for market basket analysis. SIGMOD'97, 255-264, Tucson, Arizona, May 1997.
- K. Beyer and R. Ramakrishnan. Bottom-up computation of sparse and iceberg cubes. SIGMOD'99, 359-370, Philadelphia, PA, June 1999.
- D.W. Cheung, J. Han, V. Ng, and C.Y. Wong. Maintenance of discovered association rules in large databases: An incremental updating technique. ICDE'96, 106-114, New Orleans, LA.
- M. Fang, N. Shivakumar, H. Garcia-Molina, R. Motwani, and J. D. Ullman. Computing iceberg queries efficiently. VLDB'98, 299-310, New York, NY, Aug. 1998.

References (2)

- G. Grahne, L. Lakshmanan, and X. Wang. Efficient mining of constrained correlated sets. ICDE'00, 512-521, San Diego, CA, Feb. 2000.
- Y. Fu and J. Han. Meta-rule-guided mining of association rules in relational databases. KDOOD'95, 39-46, Singapore, Dec. 1995.
- T. Fukuda, Y. Morimoto, S. Morishita, and T. Tokuyama. Data mining using two-dimensional optimized association rules: Scheme, algorithms, and visualization. SIGMOD'96, 13-23, Montreal, Canada.
- E.-H. Han, G. Karypis, and V. Kumar. Scalable parallel data mining for association rules. SIGMOD'97, 277-288, Tucson, Arizona.
- J. Han, G. Dong, and Y. Yin. Efficient mining of partial periodic patterns in time series database. ICDE'99, Sydney, Australia.
- J. Han and Y. Fu. Discovery of multiple-level association rules from large databases. VLDB'95, 420-431, Zurich, Switzerland.
- J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. SIGMOD'00, 1-12, Dallas, TX, May 2000.
- T. Imielinski and H. Mannila. A database perspective on knowledge discovery. Communications of ACM, 39:58-64, 1996.
- M. Kamber, J. Han, and J. Y. Chiang. Metarule-guided mining of multi-dimensional association rules using data cubes. KDD'97, 207-210, Newport Beach, California.
- M. Klemettinen, H. Mannila, P. Ronkainen, H. Toivonen, and A.I. Verkamo. Finding interesting rules from large sets of discovered association rules. CIKM'94, 401-408, Gaithersburg, Maryland.

References (3)

- F. Korn, A. Labrinidis, Y. Kotidis, and C. Faloutsos. Ratio rules: A new paradigm for fast, quantifiable data mining. VLDB'98, 582-593, New York, NY.
- B. Lent, A. Swami, and J. Widom. Clustering association rules. ICDE'97, 220-231, Birmingham, England.
- H. Lu, J. Han, and L. Feng. Stock movement and n-dimensional inter-transaction association rules. SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery (DMKD'98), 12:1-12:7, Seattle, Washington.
- H. Mannila, H. Toivonen, and A. I. Verkamo. Efficient algorithms for discovering association rules. KDD'94, 181-192, Seattle, WA, July 1994.
- H. Mannila, H Toivonen, and A. I. Verkamo. Discovery of frequent episodes in event sequences. Data Mining and Knowledge Discovery, 1:259-289, 1997.
- R. Meo, G. Psaila, and S. Ceri. A new SQL-like operator for mining association rules. VLDB'96, 122-133, Bombay, India.
- R.J. Miller and Y. Yang. Association rules over interval data. SIGMOD'97, 452-461, Tucson, Arizona.
- R. Ng, L. V. S. Lakshmanan, J. Han, and A. Pang. Exploratory mining and pruning optimizations of constrained associations rules. SIGMOD'98, 13-24, Seattle, Washington.
- N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal. Discovering frequent closed itemsets for association rules. ICDT'99, 398-416, Jerusalem, Israel, Jan. 1999.

References (4)

- J.S. Park, M.S. Chen, and P.S. Yu. An effective hash-based algorithm for mining association rules. SIGMOD'95, 175-186, San Jose, CA, May 1995.
- J. Pei, J. Han, and R. Mao. CLOSET: An Efficient Algorithm for Mining Frequent Closed Itemsets. DMKD'00, Dallas, TX, 11-20, May 2000.
- J. Pei and J. Han. Can We Push More Constraints into Frequent Pattern Mining? KDD'00. Boston, MA. Aug. 2000.
- G. Piatetsky-Shapiro. Discovery, analysis, and presentation of strong rules. In G. Piatetsky-Shapiro and W. J. Frawley, editors, Knowledge Discovery in Databases, 229-238. AAAI/MIT Press, 1991.
- B. Ozden, S. Ramaswamy, and A. Silberschatz. Cyclic association rules. ICDE'98, 412-421, Orlando, FL.
- J.S. Park, M.S. Chen, and P.S. Yu. An effective hash-based algorithm for mining association rules. SIGMOD'95, 175-186, San Jose, CA.
- S. Ramaswamy, S. Mahajan, and A. Silberschatz. On the discovery of interesting patterns in association rules. VLDB'98, 368-379, New York, NY..
- S. Sarawagi, S. Thomas, and R. Agrawal. Integrating association rule mining with relational database systems: Alternatives and implications. SIGMOD'98, 343-354, Seattle, WA.
- A. Savasere, E. Omiecinski, and S. Navathe. An efficient algorithm for mining association rules in large databases. VLDB'95, 432-443, Zurich, Switzerland.
- A. Savasere, E. Omiecinski, and S. Navathe. Mining for strong negative associations in a large database of customer transactions. ICDE'98, 494-502, Orlando, FL, Feb. 1998.

References (5)

- C. Silverstein, S. Brin, R. Motwani, and J. Ullman. Scalable techniques for mining causal structures. VLDB'98, 594-605, New York, NY.
- R. Srikant and R. Agrawal. Mining generalized association rules. VLDB'95, 407-419, Zurich, Switzerland, Sept. 1995.
- R. Srikant and R. Agrawal. Mining quantitative association rules in large relational tables. SIGMOD'96, 1-12, Montreal, Canada.
- R. Srikant, Q. Vu, and R. Agrawal. Mining association rules with item constraints. KDD'97, 67-73, Newport Beach, California.
- H. Toivonen. Sampling large databases for association rules. VLDB'96, 134-145, Bombay, India, Sept. 1996.
- D. Tsur, J. D. Ullman, S. Abitboul, C. Clifton, R. Motwani, and S. Nestorov. Query flocks: A generalization of association-rule mining. SIGMOD'98, 1-12, Seattle, Washington.
- K. Yoda, T. Fukuda, Y. Morimoto, S. Morishita, and T. Tokuyama. Computing optimized rectilinear regions for association rules. KDD'97, 96-103, Newport Beach, CA, Aug. 1997.
- M. J. Zaki, S. Parthasarathy, M. Ogihara, and W. Li. Parallel algorithm for discovery of association rules. Data Mining and Knowledge Discovery, 1:343-374, 1997.
- M. Zaki. Generating Non-Redundant Association Rules. KDD'00. Boston, MA. Aug. 2000.
- O. R. Zaiane, J. Han, and H. Zhu. Mining Recurrent Items in Multimedia with Progressive Resolution Refinement. ICDE'00, 461-470, San Diego, CA, Feb. 2000.

© for the original version:
Jiawei Han and Micheline Kamber
http://www.cs.sfu.ca/~han/dmbook

Data Mining Algorithms

Lecture Course with Tutorials
Summer 2007

Chapter 6: Concept Description and Generalization

Chapter 6: Concept Description and Generalization

- What is concept description?
- Descriptive statistical measures in large databases
- Data generalization and summarization-based characterization
- Analytical characterization: Analysis of attribute relevance
- Mining class comparisons: Discriminating between different classes
- Summary

What is Concept Description?

- Descriptive vs. predictive data mining
  - Descriptive mining: describes concepts or task-relevant data sets in concise, summarative, informative, discriminative forms
  - Predictive mining: Based on data and analysis, constructs models for the database, and predicts the trend and properties of unknown data
- Concept description:
  - Characterization: provides a concise and succinct summarization of the given collection of data
  - Comparison (Discrimination): provides descriptions comparing two or more collections of data

Concept Description vs. OLAP

- OLAP
  - restricted to a small number of dimension and measure types
  - user-controlled process
- Concept description
  - can handle complex data types of the attributes and their aggregations
  - a more automated process

Concept Description vs. Learning-from-example Paradigm

- Difference in philosophies and basic assumptions
  - Positive and negative samples in learning-from-example: positive used for generalization, negative - for specialization
  - Positive samples only in data mining: hence generalization-based, to drill-down backtrack the generalization to a previous state
- Difference in methods of generalizations
  - Machine learning generalizes on a tuple by tuple basis
  - Data mining generalizes on an attribute by attribute basis (see *attribute-oriented induction*)

Chapter 6: Concept Description and Generalization

- What is concept description?
- Descriptive statistical measures in large databases
- Data generalization and summarization-based characterization
- Analytical characterization: Analysis of attribute relevance
- Mining class comparisons: Discriminating between different classes
- Summary

Mining Data Dispersion Characteristics

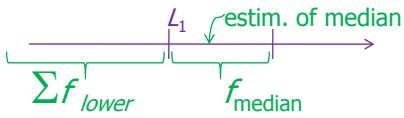
- *Motivation*
  - To better understand the data: central tendency, variation and spread
- *Data dispersion characteristics*
  - median, max, min, quantiles, outliers, variance, etc.
- *Numerical dimensions* correspond to sorted intervals
  - Data dispersion: analyzed with multiple granularities of precision
  - Boxplot or quantile analysis on sorted intervals
- *Dispersion analysis on computed measures*
  - Folding measures into numerical dimensions
  - Boxplot or quantile analysis on the transformed cube

Measuring the Central Tendency (1)

- *Mean* — (weighted) arithmetic mean $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \qquad \bar{x}_w = \frac{\sum_{i=1}^{n} w_i x_i}{\sum_{i=1}^{n} w_i}$
- *Median* — a holistic measure
  - Middle value if odd number of values, or average of the middle two values otherwise
  - Estimate the median for grouped data by interpolation:

$$median \approx L_1 + \left(\frac{n/2 - \left(\sum f\right)_{lower}}{f_{median}} \right) \cdot c$$

L_1 — lowest value of the class containing the median
n — overall number of data values
$\sum f_{lower}$ — sum of the frequencies of all classes that are lower than the median
f_{median} — frequency of the median class
c — size of the median class interval



Measuring the Central Tendency (2)

- *Mode*
  - Value that occurs most frequently in the data
  - Well suited for categorical (i.e., non-numeric) data
  - Unimodal, bimodal, trimodal, …: there are 1, 2, 3, … modes in the data (multimodal in general), cf. mixture models
  - There is no mode if each data value occurs only once
  - Empirical formula for unimodal frequency curves that are moderately skewed:

$$mean - mode \approx 3 \cdot (mean - median)$$

- *Midrange*
  - Average of the largest and the smallest values in a data set:

$$(max - min) / 2$$

Measuring the Dispersion of Data

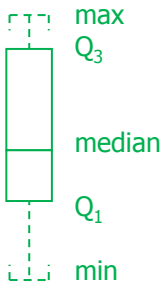
- Quartiles, outliers and boxplots
  - Quartiles: Q_1 (25th percentile), Q_3 (75th percentile)
  - Inter-quartile range: IQR = $Q_3 - Q_1$
  - Five number summary: min, Q_1, median, Q_3, max
  - Boxplot (next slide): ends of the box are the quartiles, median is marked, whiskers (Barthaare, Backenbart); plot outlier individually
  - Outlier: usually, values that are more than 1.5 x IQR below Q_1 or above Q_3

- Variance and standard deviation
  - Variance (algebraic, scalable computation): $\sigma^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2$
  - Standard deviation σ: square root of variance

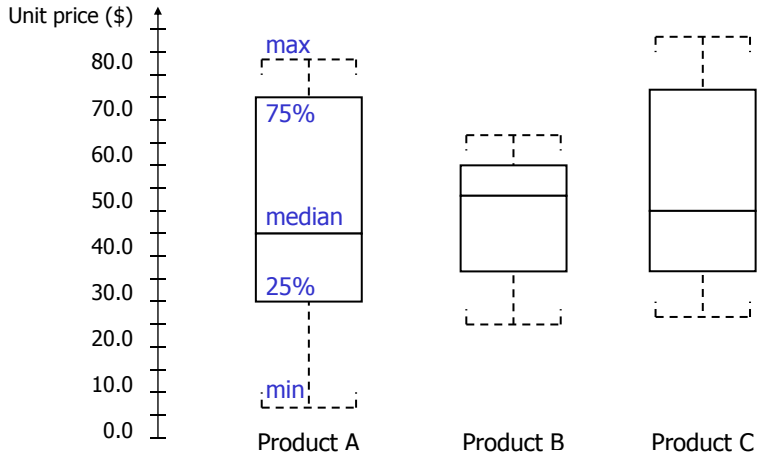
Boxplot Analysis

- Five-number summary of a distribution
  - Minimum, Q1, Median, Q3, Maximum
  - = 0%, 25%, 50%, 75%, 100%-quantiles ("25-percentile", etc.)

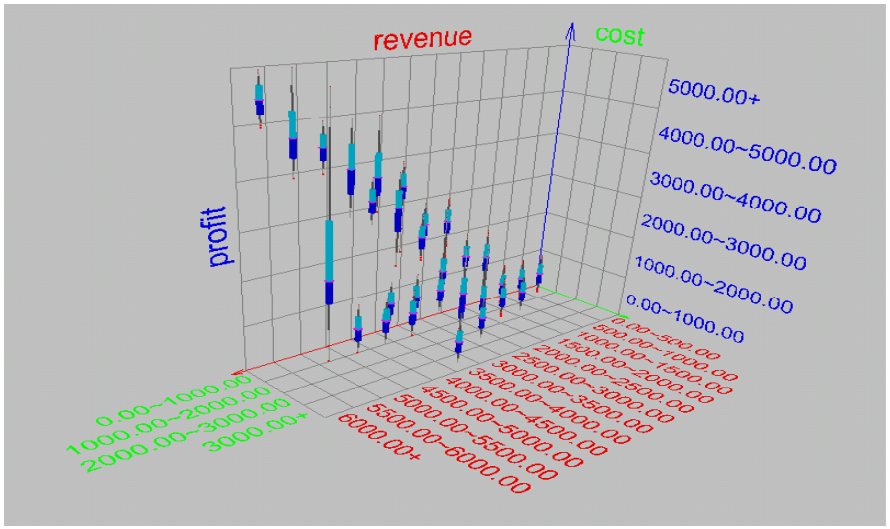
- Boxplot
  - Data is represented with a box
  - The ends of the box are at the first and third quartiles, i.e., the height of the box is IQR
  - The median is marked by a line within the box
  - Whiskers: two lines outside the box extend to Minimum and Maximum



Boxplot Examples



Visualization of Data Dispersion: Boxplot Analysis



Mining Descriptive Statistical Measures in Large Databases

- Variance

alternatives: $\frac{1}{n-1}$, $\frac{1}{n}$

$$\sigma^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2 = \frac{1}{n-1}\left[\sum x_i^2 - \frac{1}{n}\left(\sum x_i\right)^2\right]$$

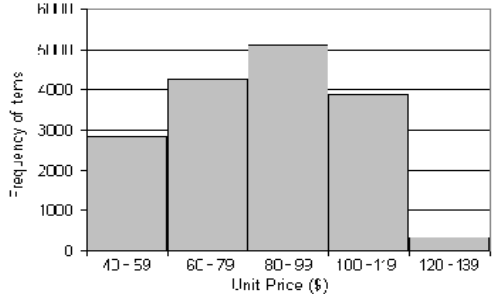
May be computed in a single pass!

Requires two passes but is numerically much more stable

- Standard deviation: the square root of the variance
  - Measures the spread around the mean
  - It is zero if and only if all the values are equal
  - Both the deviation and the variance are algebraic

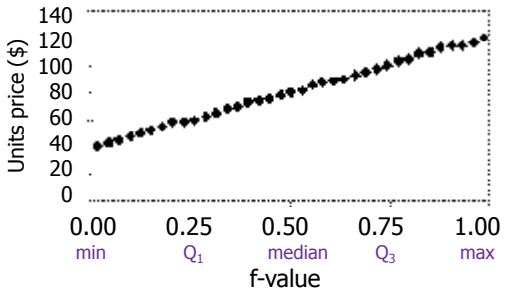
Histogram Analysis

- Graph displays of basic statistical class descriptions
  - Frequency histograms
    - A univariate graphical method
    - Consists of a set of rectangles that reflect the counts (frequencies) of the classes present in the given data



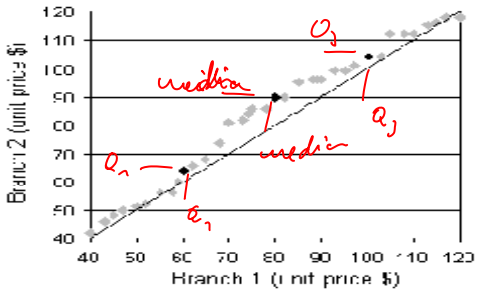
Quantile Plot

- Displays all of the data (allowing the user to assess both the overall behavior and unusual occurrences)
- Plots quantile information
  - The q-quantile x_q indicates the value x_q for which the fraction q of all data is less than or equal to x_q (called percentile if q is a percentage); e.g., median = 50%-quantile or 50th percentile.



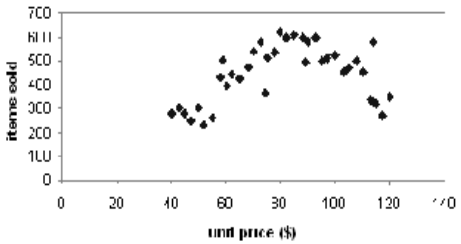
Quantile-Quantile (Q-Q) Plot

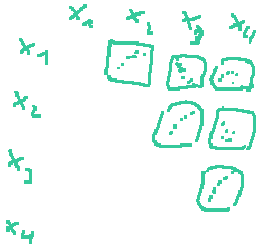
- Graphs the quantiles of one univariate distribution against the corresponding quantiles of another
- Allows the user to view whether there is a shift in going from one distribution to another



Scatter plot

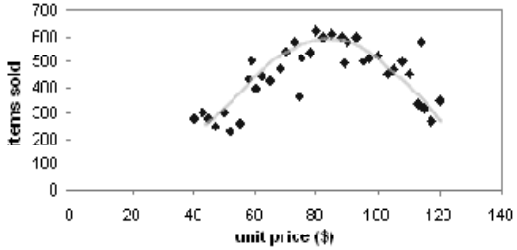
- Provides a first look at bivariate data to see clusters of points, outliers, etc
- Each pair of values is treated as a pair of coordinates and plotted as points in the plane





Loess Curve (local regression)

- Adds a smooth curve to a scatter plot in order to provide better perception of the pattern of dependence
- Loess curve is fitted by setting two parameters: a smoothing parameter, and the degree of the polynomials that are fitted by the regression

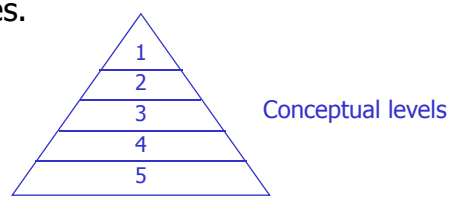


Chapter 6: Concept Description and Generalization

- What is concept description?
- Descriptive statistical measures in large databases
- Data generalization and summarization-based characterization
- Analytical characterization: Analysis of attribute relevance
- Mining class comparisons: Discriminating between different classes
- Summary

Data Generalization and Summarization-based Characterization

- Data generalization
  - A process which abstracts a large set of task-relevant data in a database from low conceptual levels to higher ones.



  - Approaches:
    - Data cube approach (OLAP approach)
    - Attribute-oriented induction approach

Characterization: Data Cube Approach (without using AO-Induction)

- Perform computations and store results in data cubes
- Strength
  - An efficient implementation of data generalization
  - Computation of various kinds of measures
    - e.g., count( ), sum( ), average( ), max( )
  - Generalization and specialization can be performed on a data cube by *roll-up* and *drill-down*
- Limitations
  - handle only dimensions of *simple nonnumeric data* and measures of *simple aggregated numeric values*.
  - Lack of intelligent analysis, can't tell which dimensions should be used and what levels should the generalization reach

Attribute-Oriented Induction

- Proposed in 1989 (KDD '89 workshop)
- Not confined to categorical data nor particular measures.
- How is it done?
  - Collect the task-relevant data (*initial relation*) using a relational database query.
  - Perform generalization by either *attribute removal* or *attribute generalization*.
  - Apply aggregation by merging identical, generalized tuples and accumulating their respective counts.
  - Interactive presentation with users.

Attribute-Oriented Induction: Basic Principles

- Data focusing: task-relevant data, including dimensions, and the result is the *initial relation*.
- Attribute-removal: remove attribute *A* if
  1) there is a large set of distinct values for *A* but there is no generalization operator (concept hierarchy) on *A*, or
  2) *A*'s higher level concepts are expressed in terms of other attributes (e.g. *street* is covered by *city*, *province_or_state*, *country*).
- Attribute-generalization: if there is a large set of distinct values for *A*, and there exists a set of generalization operators (i.e., a concept hierarchy) on *A*, then select an operator and generalize *A*.

Attribute Generalization Control

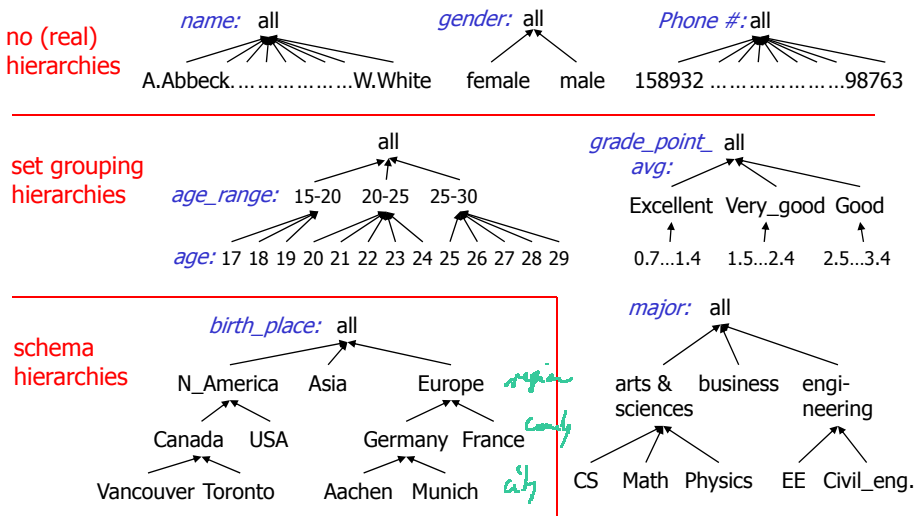
- Problem: How many distinct values for an attribute?
  - overgeneralization (values are too high-level) or
  - undergeneralization (level not sufficiently high)
  - both yield rules of poor usefulness.

- Two common approaches
  - Attribute-threshold control: default or user-specified, typically 2-8 values
  - Generalized relation threshold control: control the size of the final relation/rule, e.g., 10-30

Attribute-Oriented Induction: Basic Algorithm

- InitialRelation: Query processing of task-relevant data, deriving the *initial working relation*.
- PrepareForGeneralization (PreGen): Based on the analysis of the number of distinct values in each attribute, determine generalization plan for each attribute: removal? or generalization to which level of abstraction in the concept hierarchy?
- PrimeGeneralization (PrimeGen): Based on the prepared plan, perform generalization to the right level to derive a "prime generalized relation", accumulating the counts.
- Presentation: User interaction: (1) adjust levels by drilling, (2) pivoting, (3) mapping into rules, cross tabs, visualization presentations.

Example: Given Concept Hierarchies



Example: Initial Relation

Name	Gender	Major	Birth-Place	Birth_date	Residence	Phone #	GPA
Jim Woodman	M	CS	Vancouver,BC, Canada	8-12-81	3511 Main St., Richmond	687-4598	3.67
Scott Lachance	M	CS	Montreal, Que, Canada	28-7-80	345 1st Ave., Richmond	253-9106	3.70
Laura Lee	F	Physics	Seattle, WA, USA	25-8-75	125 Austin Ave., Burnaby	420-5232	3.83
…	…	…	…	…	…	…	…
Removed	Retained	Sci,Eng, Bus	Country	Age range	City	Removed	Excel, VG,..

- *Name:* large number of distinct values, no hierarchy—removed.
- *Gender:* only two distinct values—retained.
- *Major:* many values, hierarchy exists—generalized to Sci.,Eng.,Bus.
- *Birth_place:* many values, hierarchy—generalized, e.g., to country.
- *Birth_date:* many values—generalized to age (or age_range).
- *Residence:* many streets and numbers—generalized to city.
- *Phone number:* many values, no hierarchy—removed.
- *Grade_point_avg* (GPA): hierarchy exists—generalized to good….
- *Count*: additional attribute to aggregate base tuples

Attribute-oriented Induction-Implementation(1)

- Input parameters
  - *DB* — a relational database
  - *DMQuery* — a data mining query
  - *Attributes* — a list of attributes a_i
  - *Generalization*(a_i) — a set of concept hierarchies or generalization operators on attributes a_i
  - *AttributeGeneralizationThreshold*(a_i) — attribute generalization thresholds for each a_i
- Output parameters
  - *P* — a prime generalized relation
- Procedure InitialRelation (*DMQuery, DB*)
  - Fetch the task-relevant data into *W*, the working relation.

Attribute-oriented Induction-Implementation(2)

- Procedure PrepareForGeneralization *(W)*
  - Scan working relation *W* and collect the distinct values for each attribute a_i
    - If *W* is very large, a sample of *W* may be used instead.
  - For each attribute a_i:
    - Determine whether a_i should be removed
    - If not, compute the minimum desired level L_i based on the attribute threshold, and determine the mapping-pairs *(v, v')* for distinct values *v* of a_i in *A* and the corresponding generalized values *v'* at level L_i.

Attribute-oriented Induction-Implementation(3)

- Procedure PrimeGeneralization *(W)*
  - Derive the prime_generalized_relation *P* by replacing each value *v* in *W* by the corresponding *v'* in the mapping while maintaining *count* (and other aggregate values, if any).
  - Depending on the number of distinct values at the prime relation level, *P* can be coded as a sorted relation or as a multidimensional array.
- Variation of the algorithm [Han, Cai & Cercone 1993]
  - Rather than performing PrepareForGeneralization and PrimeGeneralization in a sequence, the prime relation *P* can be updated for each attribute selection step. Then, the order of attribute selection may control the process.

Order of Attribute Selection

Strategies to select the next attribute for generalization

- Aiming at minimal degree of generalization
  - Choose attribute that reduces the number of tuples the most.
  - Useful heuristics: choose attribute a_i with highest number m_i of distinct values.
- Aiming at similar degree of generalization for all attributes
  - Choose the attribute currently having the least degree of generalization
- User-controlled
  - Domain experts may specify appropriate priorities for the selection of attributes

Presentation of Generalized Results

- *Generalized relation*
  - Relations where some or all attributes are generalized, with counts or other aggregation values accumulated.
- *Cross tabulation*
  - Mapping results into cross tabulation form.
- *Visualization techniques*
  - Pie charts, bar charts, curves, cubes, and other visual forms.
- *Quantitative characteristic rules*
  - Mapping generalized result into characteristic rules with quantitative information associated with it, e.g.,

$$grad(x) \wedge male(x) \Rightarrow$$
$$birth\_region(x) = \text{"}Canada\text{"}[t:53\%] \vee$$
$$birth\_region(x) = \text{"}foreign\text{"}[t:47\%].$$

t: typicality weight (see next slide)

Quantitative Characteristic Rules

- *Typicality weight (t_weight)* of the disjuncts in a rule
  - $\{q_1, \ldots, q_n\}$: generalized tuples that represent the target class
  - *t_weight:* fraction of tuples representing the target class in initial relation covered by a single generalized tuple q_a
  - definition:

$$t\_weight(q_a) = \frac{\text{count}(q_a)}{\sum_{i=1}^{n} \text{count}(q_i)}$$

  - range is [0…1]
- Form of a *Quantitative Characteristic Rule:* (cf. crosstab)

$$\forall X, \text{target\_class}(X) \Rightarrow$$
$$\text{condition}_1(X)[t:w_1] \vee \ldots \vee \text{condition}_m(X)[t:w_m]$$

  - Disjunction represents a *necessary* condition of the target class
  - *Not sufficient*: a tuple that meets the conditions could belong to another class

Presentation—Generalized Relation

Example: A generalized relation for the sales in 2002

Location	Item	Sales (in million $)	Count (in thousands)
Asia	TV	15	300
Europe	TV	12	250
North_America	TV	28	450
Asia	Computer	120	1000
Europe	Computer	150	1200
North_America	computer	200	1800

Presentation—Crosstab

Example: A crosstab for the sales in 2002

Location \ item	TV		Computer		*Both_items*	
	sales	count	sales	count	sales	count
Asia	15	300	120	1000	135	1300
Europe	12	250	150	1200	162	1450
North_America	28	450	200	1800	228	2250
all_regions	45	1000	470	4000	525	5000

count corresponds to *t_weight*

Example: Generalized Relation

Initial Relation

Name	Gender	Major	Birth-Place	Birth_date	Residence	Phone #	GPA
Jim Woodman	M	CS	Vancouver, BC, Canada	8-12-81	3511 Main St., Richmond	687-4598	3.67
Scott Lachance	M	CS	Montreal, Que, Canada	28-7-80	345 1st Ave., Richmond	253-9106	3.70
Laura Lee	F	Physics	Seattle, WA, USA	25-8-75	125 Austin Ave., Burnaby	420-5232	3.83
…	…	…	…	…	…	…	…
Removed	Retained	Sci,Eng, Bus	Country	Age range	City	Removed	Excl, VG,..

Prime Generalized Relation

Gender	Major	Birth_region	Age_range	Residence	GPA	Count
M	Science	Canada	20-25	Richmond	Very-good	16
F	Science	Foreign	25-30	Burnaby	Excellent	22
…	…	…	…	…	…	…

Crosstab for Generalized Relation

Gender \ Birth_Region	Canada	Foreign	Total
M	16	14	30
F	10	22	32
Total	26	36	62

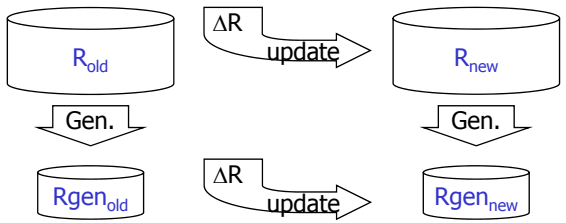
AOI: Implementation by Cube Technology

Alternative Approaches:

- Construct a data cube on-the-fly for the given data mining query
  - Facilitate efficient drill-down analysis
  - May increase the response time
  - A balanced solution: precomputation of "subprime" relation
- Use a predefined & precomputed data cube
  - Construct a data cube beforehand
  - Facilitate not only the attribute-oriented induction, but also attribute relevance analysis, dicing, slicing, roll-up and drill-down
  - Cost of cube computation and the nontrivial storage overhead

Incremental Generalization

- Database update: $R_{new} = R_{old} \cup \Delta R$
- Incremental Mining: update generalized relations (small) directly without applying generalization to R_{new} (large)



- Requirements
  - Efficiency: significantly faster update of generalized relations
  - Correctness: update(generalize(R)) = generalize(update(R))

Incremental Generalization—Algorithms

- Insertion of new tuples
  - Generalize ΔR to the same level of abstraction as in the generalized relation R_{gen} to derive ΔR_{gen}
  - Union $R_{gen} \cup \Delta R_{gen}$, i.e., merge counts and other statistical information to produce a new relation R_{gen}'
  - If relation threshold (i.e., # tuples) is exceeded, further apply the attribute-oriented generalization to R_{gen}'
- Deletion of tuples
  - Generalize ΔR to the same level of abstraction in the generalized relation R_{gen} to derive ΔR_{gen}
  - Remove ΔR_{gen} from R_{gen}, i.e., decrease counts and maintain other statistical information to produce R_{gen}'
  - Relation threshold is ok but overgeneralization may occur

Incremental Generalization—Problem of Overgeneralization

- *Overgeneralization:* incremental deletion yields a higher level of abstraction than generalizing the updated base relation.
- Example

Name	Gender	Age
Ann	Female	47
Ben	Male	45
Clara	Female	48
Dan	Male	62

Delete (Ben,male,45)

Gen.

Gender	Age_Range	Count
Female	Mid_aged	2
Male	Senior	1

≠!

Gen.

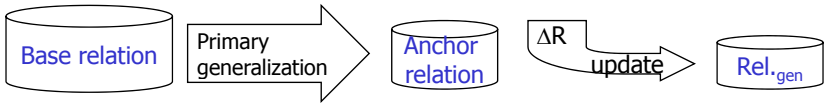
Age_range	Count
Mid_aged	3
Senior	1

Delete (mid_aged,1)

Age_Range	Count
Mid_aged	2
Senior	1

Incremental Generalization—Anchor Relation

- Approach to prevent from overgeneralization
  - Use an intermediate *anchor relation*, i.e., an relation that is generalized to an level of abstraction below the desired final level of abstraction.
- Procedure
  - Generalize updates ΔR to the level of the anchor relation
  - Apply the generalized updates ΔR_{gen} to the anchor relation
  - Generalize the updated anchor relation to the final level



Chapter 6: Concept Description and Generalization

- What is concept description?
- Descriptive statistical measures in large databases
- Data generalization and summarization-based characterization
- Analytical characterization: Analysis of attribute relevance
- Mining class comparisons: Discriminating between different classes
- Summary

Characterization vs. OLAP

- Shared concepts:
  - Presentation of data summarization at multiple levels of abstraction.
  - Interactive drilling, pivoting, slicing and dicing.
- Differences:
  - Automated desired level allocation.
  - Dimension relevance analysis and ranking when there are many relevant dimensions.
  - Sophisticated typing on dimensions and measures.
  - Analytical characterization: data dispersion analysis.

Streuung

Attribute Relevance Analysis

- Why?—Support for specifying generalization parameters
  - Which dimensions should be included?
  - How high level of generalization?
  - Automatic vs. interactive
  - Reduce number of attributes
    → easy to understand patterns / rules
- What?—Purpose of the method
  - statistical method for preprocessing data
    - filter out irrelevant or weakly relevant attributes
    - retain or rank the relevant attributes
  - relevance related to dimensions and levels
  - analytical characterization, analytical *comparison*

Attribute relevance analysis (cont'd)

- How?—Steps of the algorithm:
  - Data Collection
  - Analytical Generalization
    - Use information gain analysis (e.g., entropy or other measures) to identify highly relevant dimensions and levels.
  - Relevance Analysis
    - Sort and select the most relevant dimensions and levels.
  - Attribute-oriented Induction for class description
    - On selected dimension/level

Relevance Measures

- Quantitative relevance measure determines the classifying power of an attribute within a set of data.

- Competing methods
  - information gain (ID3) — discussed here
  - gain ratio (C4.5)
  - gini index (IBM Intelligent Miner)
  - χ^2 contingency table statistics
  - uncertainty coefficient

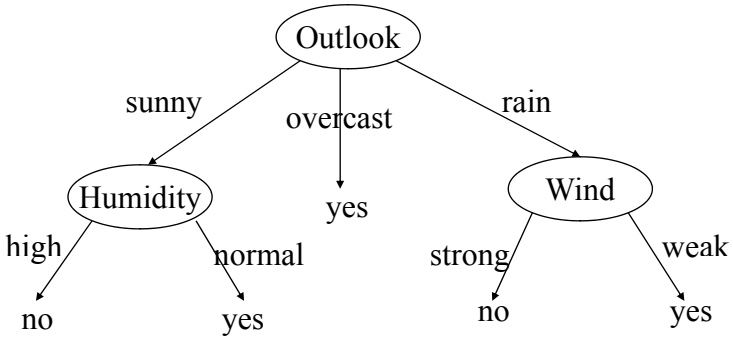
Information-Theoretic Approach

- Decision tree
  - each internal node tests an attribute
  - each branch corresponds to attribute value
  - each leaf node assigns a classification
- ID3 algorithm
  - build decision tree based on training objects with known class labels to classify testing objects
  - rank attributes with information gain measure
  - minimal height
    - the least number of tests to classify an object

Top-Down Induction of Decision Tree

Attributes = {Outlook, Temperature, Humidity, Wind}

PlayTennis = {yes, no}



Entropy and Information Gain

- S contains s_i tuples of class C_i for $i = \{1, \ldots, m\}$
- Information measures info required to classify any arbitrary tuple

$$I(s_1, s_2, \ldots, s_m) = -\sum_{i=1}^{m} \frac{s_i}{s} \log_2 \frac{s_i}{s}$$

- Entropy of attribute A with values $\{a_1, a_2, \ldots, a_v\}$

$$E(A) = \sum_{j=1}^{v} \frac{s_{1j} + \ldots + s_{mj}}{s} I(s_{1j}, \ldots, s_{mj})$$

- Information gained by branching on attribute A

$$\text{Gain}(A) = I(s_1, s_2, \ldots, s_m) - E(A)$$

Example: Analytical Characterization

- Task
  - Mine general characteristics describing graduate students using analytical characterization
- Given
  - attributes *name, gender, major, birth_place, birth_date, phone#, gpa*
  - *generalization*(a_i) = concept hierarchies on a_i
  - U_i = attribute analytical thresholds for a_i
  - R = attribute relevance threshold
  - T_i = attribute generalization thresholds for a_i

Example: Analytical Characterization (2)

- Step 1: Data collection
  - target class: graduate student
  - contrasting class: undergraduate student
- Step 2: Analytical generalization using thresholds U_i
  - attribute removal
    - remove *name* and *phone#*
  - attribute generalization
    - generalize *major, birth_place, birth_date, gpa*
    - accumulate counts
  - candidate relation
    - *gender, major, birth_country, age_range, gpa*

Example: Analytical characterization (3)

gender	major	birth_country	age_range	gpa	count
M	Science	Canada	20-25	Very_good	16
F	Science	Foreign	25-30	Excellent	22
M	Engineering	Foreign	25-30	Excellent	18
F	Science	Foreign	25-30	Excellent	25
M	Science	Canada	20-25	Excellent	21
F	Engineering	Canada	20-25	Excellent	18

Candidate relation for Target class: Graduate students (Σ=120)

gender	major	birth_country	age_range	gpa	count
M	**Science**	Foreign	<20	Very_good	18
F	Business	Canada	<20	Fair	20
M	Business	Canada	<20	Fair	22
F	Science	Canada	20-25	Fair	24
M	Engineering	Foreign	20-25	Very_good	22
F	Engineering	Canada	<20	Excellent	24

Candidate relation for Contrasting class: Undergraduate students (Σ=130)

Example: Analytical Characterization (4)

- Step 3: Relevance analysis
  - Calculate expected info required to classify an arbitrary tuple

$$I(s_1, s_2) = I(120,130) = -\frac{120}{250}\log_2\frac{120}{250} - \frac{130}{250}\log_2\frac{130}{250} = 0.9988$$

  - Calculate entropy of each attribute: e.g. *major*

For *major="Science"*: $s_{11}=84$ $s_{21}=42$ $I(s_{11}, s_{21})=0.9183$
For *major="Engineering"*: $s_{12}=36$ $s_{22}=46$ $I(s_{12}, s_{22})=0.9892$
For *major="Business"*: $s_{13}=0$ $s_{23}=42$ $I(s_{13}, s_{23})=0$

Number of grad students in "Science"

Number of undergrad students in "Science"

Example: Analytical Characterization (5)

- Calculate expected info required to classify a given sample if *S* is partitioned according to the attribute

$$E(\text{major}) = \frac{126}{250} I(s_{11}, s_{21}) + \frac{82}{250} I(s_{12}, s_{22}) + \frac{42}{250} I(s_{13}, s_{23}) = 0.7873$$

- Calculate information gain for each attribute

$$\text{Gain}(\text{major}) = I(s_1, s_2) - E(\text{major}) = 0.2115$$

  - Information gain for all attributes

Gain(gender) = 0.0003
Gain(birth_country) = 0.0407
Gain(major) = 0.2115
Gain(gpa) = 0.4490
Gain(age_range) = 0.5971

Example: Analytical Characterization (6)

- Step 4a: Derive initial working relation W_0
  - Use attribute relevance threshold *R*, e.g., *R* = 0.1
  - remove irrelevant/weakly relevant attributes (*gain* < *R*) from candidate relation, i.e., drop *gender*, *birth_country*
  - remove contrasting class candidate relation

major	age_range	gpa	count
Science	20-25	Very_good	16
Science	25-30	Excellent	47
Science	20-25	Excellent	21
Engineering	20-25	Excellent	18
Engineering	25-30	Excellent	18

Initial target class working relation W_0: Graduate students

- Step 4b: Perform attribute-oriented induction using thresholds T_i

Chapter 6: Concept Description and Generalization

- What is concept description?
- Descriptive statistical measures in large databases
- Data generalization and summarization-based characterization
- Analytical characterization: Analysis of attribute relevance
- Mining class comparisons: Discriminating between different classes
- Summary

Mining Class Comparisons

- Comparison
  - Comparing two or more classes.
- Relevance Analysis
  - Find attributes (features) which best distinguish different classes.
- Method
  - Partition the set of relevant data into the target class and the contrasting class(es)
  - Analyze the attribute's relevances
  - Generalize both classes to the same high level concepts
  - Compare tuples with the same high level descriptions
  - Present the results and highlight the tuples with strong discriminant features

Example: Analytical comparison (2)

- Task
  - Compare graduate and undergraduate students using discriminant rule.
- Given
  - attributes *name, gender, major, birth_place, birth_date, residence, phone#, gpa*
  - *generalization*(a_i) = concept hierarchies on attributes a_i
  - U_i = attribute analytical thresholds for attributes a_i
  - R = attribute relevance threshold
  - T_i = attribute generalization thresholds for attributes a_i

Example: Analytical comparison (3)

- Step1: Data collection
  - target and contrasting classes
- Step 2: Attribute relevance analysis
  - remove attributes *name, gender, major, phone#*
- Step 3: Synchronous generalization
  - controlled by user-specified dimension thresholds
  - prime target and contrasting class(es) relations/cuboids

Example: Analytical comparison (4)

birth_country	age_range	Gpa	count%
Canada	20-25	Good	5.53%
Canada	25-30	Good	2.32%
Canada	over_30	Very_good	5.86%
...	...	...	...
Other	over_30	Excellent	4.68%

Prime generalized relation for the target class: Graduate students

birth_country	age_range	Gpa	count%
Canada	15-20	Fair	5.53%
Canada	15-20	Good	4.53%
...	...	...	...
Canada	25-30	Good	5.02%
...	...	...	...
Other	over_30	Excellent	0.68%

Prime generalized relation for the contrasting class: Undergraduate students

Example: Analytical comparison (5)

- Step 4: Compare tuples; drill down, roll up and other OLAP operations on target and contrasting classes to adjust levels of abstractions of resulting description.
- Step 5: Presentation
  - as generalized relations, crosstabs, bar charts, pie charts, or rules
  - contrasting measures to reflect comparison between target and contrasting classes
    - e.g. count%

Quantitative Discriminant Rules

- C_j = target class
- q_a = a generalized tuple covers some tuples of class
  - but can also cover some tuples of any contrasting class C_i
- *Discrimination weight (d_weight)*
  - m classes C_i
  - definition: $$d\_weight\ (q_a, C_j) = \frac{\text{count}\ (q_a \in C_j)}{\sum_{i=1}^{m} \text{count}\ (q_a \in C_i)}$$ target class / contrasting classes
  - range: [0, 1]
  - high d_weight: q_a primarily represents a target class concept
  - low d_weight: q_a is primarily derived from contrasting classes
- Form of a *Quantitative Discriminant Rule:*

$$\forall X, \text{target\_class}(X) \Leftarrow \text{condition}\ (X)\ \ [d : \text{d\_weight}\ ]$$

Example: Quantitative Discriminant Rule

Status	Birth_country	Age_range	Gpa	Count
Graduate	Canada	25-30	Good	90
Undergraduate	Canada	25-30	Good	210

Count distribution between graduate and undergraduate students for a generalized tuple

- Quantitative discriminant rule

$\forall X$, graduate_student(X) $\Leftarrow$ birth_country(X) = 'Canada' $\wedge$
age_range(X) = '25-30' $\wedge$
gpa(X) = 'good' [d: 30%]

  - *d_weight* = 90/(90+210) = 30%
  - Rule is *sufficient* (but not *necessary*):
    - if X fulfills the condition, the probability that X is a graduate student is 30%, but not vice versa, i.e., there are other grad studs, too.

Discriminant weight vs. Typicality weight

- *Quantitative characteristic rules*
  $$grad(x) \wedge male(x) \Rightarrow birth\_region(x) = \text{"Canada"}[t:53\%] \vee birth\_region(x) = \text{"foreign"}[t:47\%]$$
  - *necessary* condition of target class
  - *t_weight:* fraction of tuples representing the target class in initial relation covered by a single generalized tuple q_a
    $$t\_weight(q_a) = \frac{\text{count}(q_a)}{\sum_{i=1}^{n} \text{count}(q_i)}$$
  - Example: percentage of all male grad students born in Canada is 53%, percentage of male grad students born in foreign countries is 47%
- *Quantitative Discriminant Rules*
  $$\forall X, graduate\_student(X) \Leftarrow birth\_country(X) = \text{'Canada'} \wedge age\_range(X) = \text{'25-30'} \wedge gpa(X) = \text{'good'}\ [d:30\%]$$
  - Rule is *sufficient*
  - *d_weight:* a generalized tuple covers some tuples of target class, but can also cover some tuples of contrasting class
    $$d\_weight(q_a, C_j) = \frac{\text{count}(q_a \in C_j)}{\sum_{i=1}^{m} \text{count}(q_a \in C_i)}$$
  - Example: those born in Canada, between 25 and 30 years old, and "good" gpa have a 30% probability of being a grad student

Class Description

- Quantitative *characteristic* rule (*necessary*)
  $$\forall X, target\_class(X) \Rightarrow condition_1(X)[t:w_1] \vee \ldots \vee condition_m(X)[t:w_m]$$
- Quantitative *discriminant* rule (*sufficient*)
  $$\forall X, target\_class(X) \Leftarrow condition_1(X)[d:w'_1] \vee \ldots \vee condition_m(X)[d:w'_m]$$
- Quantitative *description* rule (*necessary and sufficient*)
  $$\forall X, target\_class(X) \Leftrightarrow condition_1(X)[t:w_1, d:w'_1] \vee \ldots \vee condition_m(X)[t:w_m, d:w'_m]$$

Disjunctive Normal Form: Each condition in the disjunction ($\vee$) may be a conjunction ($\wedge$) with no more disjunctions inside

Example: Quantitative Description Rule

Location/item		TV			Computer			Both_items	
	Count	*t-wt*	*d-wt*	*Count*	*t-wt*	*d-wt*	*Count*	*t-wt*	*d-wt*
Europe	80	25%	40%	240	75%	30%	320	100%	32%
N_Am	120	17.65%	60%	560	82.35%	70%	680	100%	68%
Both_ regions	200	20%	100%	800	80%	100%	1000	100%	100%

Crosstab showing associated *t-weight*, *d-weight* values and total number (*count,* in thousands) of TVs and computers sold at AllElectronics in 1998

- Quantitative description rule for target class *Europe*

$$\forall X, \text{Europe}(X) \Leftrightarrow (\text{item}(X) = \text{"TV"})[t:25\%, d:40\%] \vee (\text{item}(X) = \text{"computer"})[t:75\%, d:30\%]$$

Chapter 6: Concept Description and Generalization

- What is concept description?
- Descriptive statistical measures in large databases
- Data generalization and summarization-based characterization
- Analytical characterization: Analysis of attribute relevance
- Mining class comparisons: Discriminating between different classes
- Summary

Summary

- Concept description: characterization and discrimination
- OLAP-based vs. attribute-oriented induction (AOI)
- Efficient implementation of AOI
- Analytical characterization and comparison
- Descriptive statistical measures in large databases
  - d_weight, t_weight

References

- Y. Cai, N. Cercone, and J. Han. Attribute-oriented induction in relational databases. In G. Piatetsky-Shapiro and W. J. Frawley, editors, Knowledge Discovery in Databases, pages 213-228. AAAI/MIT Press, 1991.
- S. Chaudhuri and U. Dayal. An overview of data warehousing and OLAP technology. ACM SIGMOD Record, 26:65-74, 1997
- C. Carter and H. Hamilton. Efficient attribute-oriented generalization for knowledge discovery from large databases. IEEE Trans. Knowledge and Data Engineering, 10:193-208, 1998.
- W. Cleveland. Visualizing Data. Hobart Press, Summit NJ, 1993.
- J. L. Devore. Probability and Statistics for Engineering and the Science, 4th ed. Duxbury Press, 1995.
- T. G. Dietterich and R. S. Michalski. A comparative review of selected methods for learning from examples. In Michalski et al., editor, Machine Learning: An Artificial Intelligence Approach, Vol. 1, pages 41-82. Morgan Kaufmann, 1983.
- M. Ester, R. Wittmann. Incremental Generalization for Mining in a Data Warehousing Environment. Proc. Int. Conf. on Extending Database Technology, pp.135-149, 1998.
- J. Gray, S. Chaudhuri, A. Bosworth, A. Layman, D. Reichart, M. Venkatrao, F. Pellow, and H. Pirahesh. Data cube: A relational aggregation operator generalizing group-by, cross-tab and sub-totals. Data Mining and Knowledge Discovery, 1:29-54, 1997.
- J. Han, Y. Cai, and N. Cercone. Data-driven discovery of quantitative rules in relational databases. IEEE Trans. Knowledge and Data Engineering, 5:29-40, 1993.

References (cont.)

- J. Han and Y. Fu. Exploration of the power of attribute-oriented induction in data mining. In U.M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors, Advances in Knowledge Discovery and Data Mining, pages 399-421. AAAI/MIT Press, 1996.
- R. A. Johnson and D. A. Wichern. Applied Multivariate Statistical Analysis, 3rd ed. Prentice Hall, 1992.
- E. Knorr and R. Ng. Algorithms for mining distance-based outliers in large datasets. VLDB'98, New York, NY, Aug. 1998.
- H. Liu and H. Motoda. Feature Selection for Knowledge Discovery and Data Mining. Kluwer Academic Publishers, 1998.
- R. S. Michalski. A theory and methodology of inductive learning. In Michalski et al., editor, Machine Learning: An Artificial Intelligence Approach, Vol. 1, Morgan Kaufmann, 1983.
- T. M. Mitchell. Version spaces: A candidate elimination approach to rule learning. IJCAI'97, Cambridge, MA.
- T. M. Mitchell. Generalization as search. Artificial Intelligence, 18:203-226, 1982.
- T. M. Mitchell. Machine Learning. McGraw Hill, 1997.
- J. R. Quinlan. Induction of decision trees. Machine Learning, 1:81-106, 1986.
- D. Subramanian and J. Feigenbaum. Factorization in experiment generation. AAAI'86, Philadelphia, PA, Aug. 1986.

© for the original version:
- Jiawei Han and Micheline Kamber
- Jörg Sander and Martin Ester

Data Mining Algorithms

Lecture Course with Tutorials
Summer 2007

Chapter 7: Mining Complex Types of Data

Chapter 7. Mining Complex Types of Data

- Multidimensional analysis and descriptive mining of complex data objects
- Mining text databases
- Mining the World-Wide Web
- Summary

Mining Complex Data Objects: *Generalization* of Structured Data

- Set-valued attribute
  - Generalization of each value in the set into its corresponding higher-level concepts
  - Derivation of the general behavior of the set, such as the number of elements in the set, the types or value ranges in the set, or the weighted average for numerical data
  - E.g., *hobby* = {*tennis, hockey, chess, violin, nintendo_games*} generalizes to {*sports, music, video_games*}
- List-valued or a sequence-valued attribute
  - Same as set-valued attributes except that the order of the elements in the sequence should be observed in the generalization

Generalizing Spatial and Multimedia Data

- Spatial data:
  - Generalize detailed geographic points into clustered regions, such as business, residential, industrial, or agricultural areas, according to land usage
  - Require the merge of a set of geographic areas by spatial operations
- Image data:
  - Extracted by aggregation and/or approximation
  - Size, color, shape, texture, orientation, and relative positions and structures of the contained objects or regions in the image
- Music data:
  - Summarize its melody: based on the approximate patterns that repeatedly occur in the segment
  - Summarized its style: based on its tone, tempo, or the major musical instruments played

Chapter 7. Mining Complex Types of Data

- Multidimensional analysis and descriptive mining of complex data objects
- Mining text databases
- Mining the World-Wide Web
- Summary

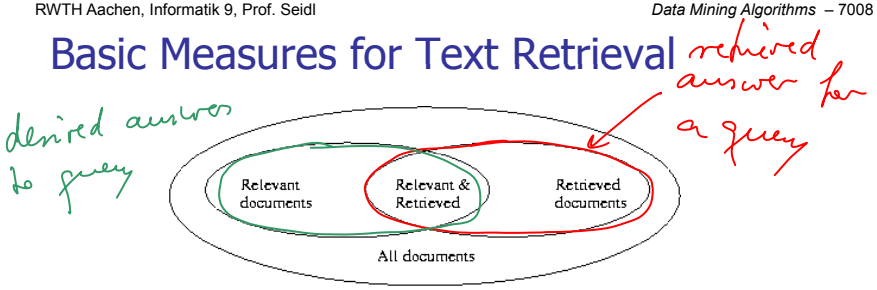
Text Databases and IR

- Text databases (document databases)
  - Large collections of documents from various sources: news articles, research papers, books, digital libraries, e-mail messages, and Web pages, library database, etc.
  - Data stored is usually *semi-structured (Bsp. XML)*
  - Traditional information retrieval techniques become inadequate for the increasingly vast amounts of text data
- Information retrieval
  - A field developed in parallel with database systems
  - Information is organized into (a large number of) documents
  - Information retrieval problem: locating relevant documents based on user input, such as keywords or example documents

Information Retrieval

- Typical IR systems
  - Online library catalogs
  - Online document management systems
- Information retrieval vs. database systems
  - Some DB problems are not present in IR, e.g., update, transaction management, complex objects
  - Some IR problems are not addressed well in DBMS, e.g., unstructured documents, approximate search using keywords and relevance

Basic Measures for Text Retrieval



- Precision: the percentage of retrieved documents that are in fact relevant to the query (i.e., "correct" responses)

$$precision = \frac{|\{Relevant\} \cap \{Retrieved\}|}{|\{Retrieved\}|}$$

- Recall: the percentage of documents that are relevant to the query and were, in fact, retrieved

$$recall = \frac{|\{Relevant\} \cap \{Retrieved\}|}{|\{Relevant\}|}$$

Keyword-Based Retrieval

- A document is represented by a string, which can be identified by a set of keywords
- Queries may use expressions of keywords
  - E.g., car *and* repair shop, tea *or* coffee, DBMS *but not* Oracle
  - Queries and retrieval should consider synonyms, e.g., repair and maintenance
- Major difficulties of the model
  - Synonymy: A keyword *T* does not appear anywhere in the document, even though the document is closely related to *T*, e.g., data mining
  - Polysemy: The same keyword may mean different things in different contexts, e.g., mining

Similarity-Based Retrieval in Text Databases

- Finds similar documents based on a set of common keywords
- Answer should be based on the degree of relevance based on the nearness of the keywords, relative frequency of the keywords, etc.
- Basic techniques
  - Stop list
    - Set of words that are deemed "irrelevant", even though they may appear frequently
    - E.g., a, the, of, for, with, etc.
    - Stop lists may vary when document set varies
  - Word stem
    - Several words are small syntactic variants of each other since they share a common word stem
    - E.g., drug, drugs, drugged

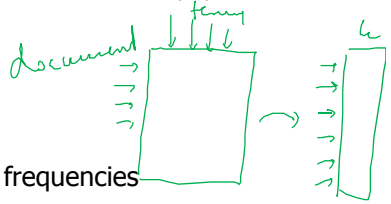
Similarity-Based Retrieval in Text Databases (2)

- A term frequency table
  - Each entry *frequent_table(i, j)* = # of occurrences of the word t_i in document d_j
  - Usually, the *ratio* instead of the absolute number of occurrences is used
- Similarity metrics: measure the closeness of a document to a query (a set of keywords)
  - Relative term occurrences
  - Cosine distance:

$$\langle v_1, v_2 \rangle = |v_1| \cdot |v_2| \cdot \cos(v_1, v_2)$$

$$similarity(v_1, v_2) = \frac{\langle v_1, v_2 \rangle}{|v_1||v_2|}$$

Latent Semantic Indexing

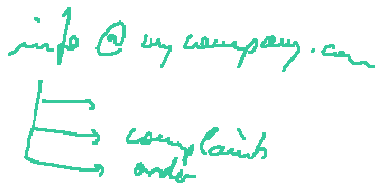


- Basic idea
  - Similar documents have similar word frequencies
  - Difficulty: the size of the term frequency matrix is very large
  - Use a singular value decomposition (SVD = PCA = KLT) technique to reduce the size of frequency table *(reduction of dimensionality)*
  - Retain the *K* most significant rows of the frequency table
- Method
  - Create a term frequency matrix, *freq_matrix*
  - SVD construction: Compute the singular valued decomposition of *freq_matrix F* by splitting it into 3 matrices, $F = U \cdot S \cdot V$, $V \cdot V^t = Id$
  - Vector identification: For each document *d*, replace its original document vector by a new excluding the eliminated terms
  - Index creation: Store the set of all vectors, indexed by one of a number of techniques (such as TV-tree)

Other Text Retrieval Indexing Techniques

- Inverted index
  - Maintains two hash- or B+-tree indexed tables:
    - a posting represents the occurrence of a term *T* in a document *d* i.e. a posting is a link object in the *m:n*-relationship term-documents
    - document_table: a set of document records <doc_id, postings_list>
    - term_table: a set of term records, <term, postings_list>
    - In SQL: *index on (doc_id, term)* and *index on (term, doc_id)*
  - Answer query: Find all docs associated with one or a set of terms
  - Advantage: easy to implement
  - Disadvantage: do not handle well synonymy and polysemy, and posting lists could be too long (storage could be very large)
- Signature file
  - Associate a signature with each document
  - A signature is a representation of an ordered list of terms that describe the document
  - Order is obtained by frequency analysis, stemming and stop lists

Types of Text Data Mining



- Keyword-based association analysis
- Automatic document classification
- Similarity detection
  - Cluster documents by a common author
  - Cluster documents containing information from a common source
- Link analysis: unusual correlation between entities
- Sequence analysis: predicting a recurring event
- Anomaly detection: find information that violates usual patterns
- Hypertext analysis
  - Patterns in anchors/links
    - Anchor text correlations with linked objects

Keyword-based association analysis

- Collect sets of keywords or terms that occur frequently together and then find the association or correlation relationships among them
- First preprocess the text data by parsing, stemming, removing stop words, etc.
- Then evoke association mining algorithms
  - Consider each document as a transaction
  - View a set of keywords in the document as a set of items in the transaction
- Term level association mining
  - No need for human effort in tagging documents
  - The number of meaningless results and the execution time is greatly reduced

Automatic document classification

- Motivation
  - Automatic classification for the tremendous number of on-line text documents (Web pages, emails, etc.)
- A classification problem
  - Training set: Human experts generate a training data set
  - Classification: The computer system discovers the classification rules
  - Application: The discovered rules can be applied to classify new/unknown documents
- Text document classification differs from the classification of relational data
  - Document databases are not structured according to attribute-value pairs

Association-Based Document Classification

- Extract keywords and terms by information retrieval and simple association analysis techniques
- Obtain concept hierarchies of keywords and terms using
  - Available term classes, such as WordNet
  - Expert knowledge
  - Some keyword classification systems
- Classify documents in the training set into class hierarchies
- Apply term association mining method to discover sets of associated terms
- Use the terms to maximally distinguish one class of documents from others
- Derive a set of association rules associated with each document class
- Order the classification rules based on their occurrence frequency and discriminative power
- Used the rules to classify new documents

Document Clustering

- Automatically group related documents based on their contents
- Require no training sets or predetermined taxonomies, generate a taxonomy at runtime
- Major steps
  - Preprocessing
    - Remove stop words, stem, feature extraction, lexical analysis, …
  - Hierarchical clustering
    - Compute similarities applying clustering algorithms, …
  - Slicing
    - Fan out controls, flatten the tree to configurable number of levels, …

Chapter 7. Mining Complex Types of Data

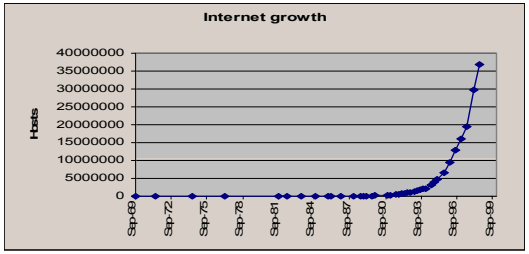
- Multidimensional analysis and descriptive mining of complex data objects
- Mining text databases
- Mining the World-Wide Web
- Summary

Mining the World-Wide Web

- The WWW is huge, widely distributed, global information service center for
  - Information services: news, advertisements, consumer information, financial management, education, government, e-commerce, etc. (*content*)
  - Hyper-link information (*structure*)
  - Access and usage information (*usage*)
- WWW provides rich sources for data mining
- Challenges
  - Too huge for effective data warehousing and data mining
  - Too complex and heterogeneous: no standards and structure

Mining the World-Wide Web

- Growing and changing very rapidly



- Broad diversity of user communities
- Only a small portion of the information on the Web is truly relevant or useful
  - 99% of the Web information is useless to 99% of Web users
  - How can we find high-quality Web pages on a specified topic?

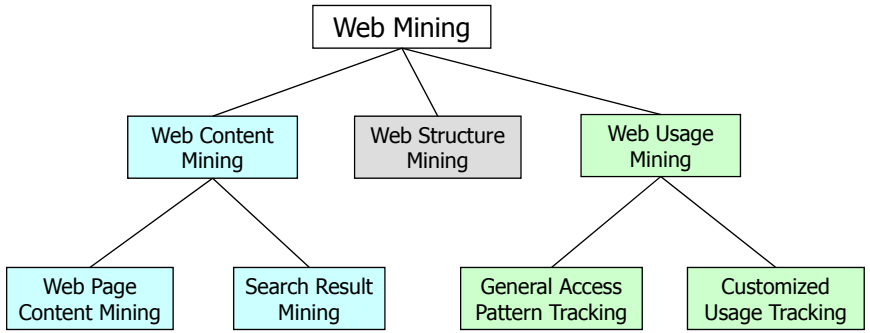
Web search engines

- Index-based: search the Web, index Web pages, and build and store huge keyword-based indices
- Help locate sets of Web pages containing certain keywords
- Deficiencies
  - A topic of any breadth may easily contain hundreds of thousands of documents
  - Many documents that are highly relevant to a topic may not contain keywords defining them (polysemy)

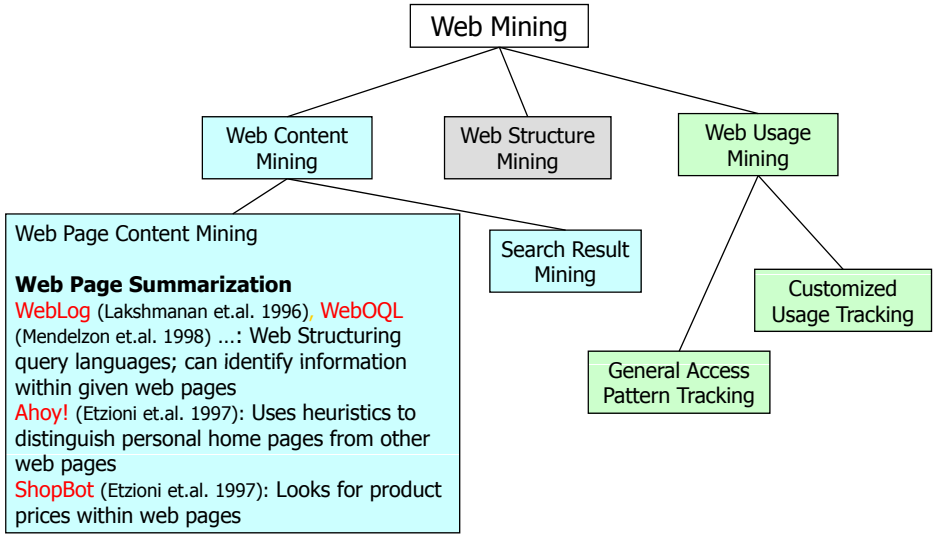
Web Mining: A more challenging task

- Searches for
  - Web access patterns
  - Web structures
  - Regularity and dynamics of Web contents
- Problems
  - The "abundance" (Überfluss) problem
  - Limited coverage of the Web: hidden Web sources, majority of data in DBMS
  - Limited query interface based on keyword-oriented search
  - Limited customization to individual users

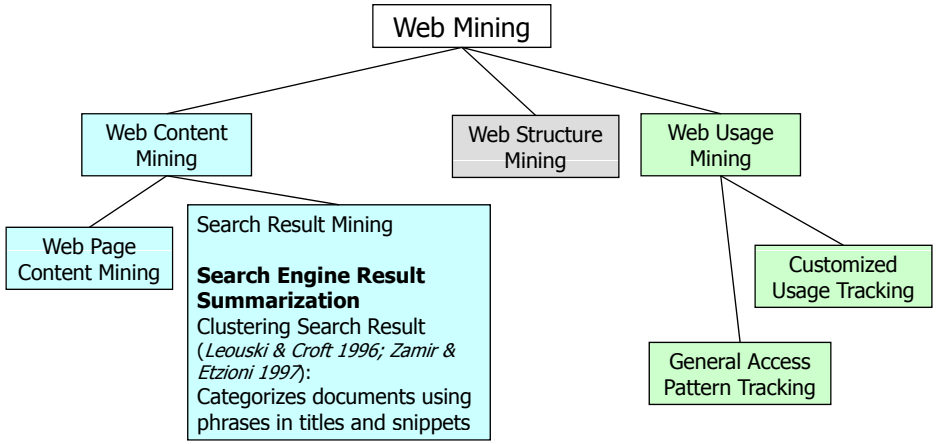
Web Mining Taxonomy



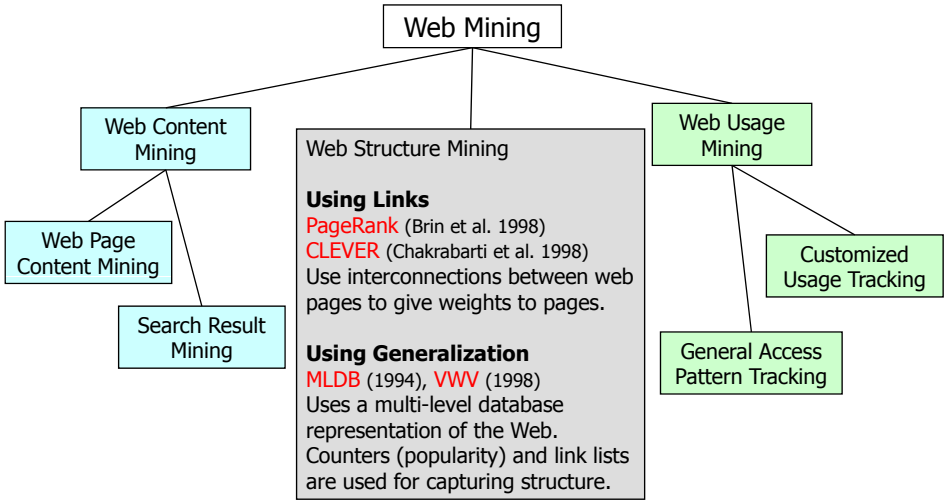
Mining the World-Wide Web



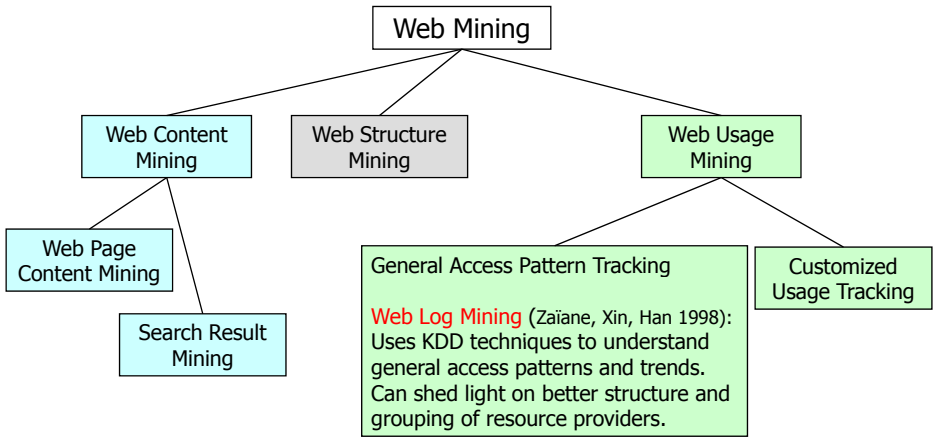
Mining the World-Wide Web



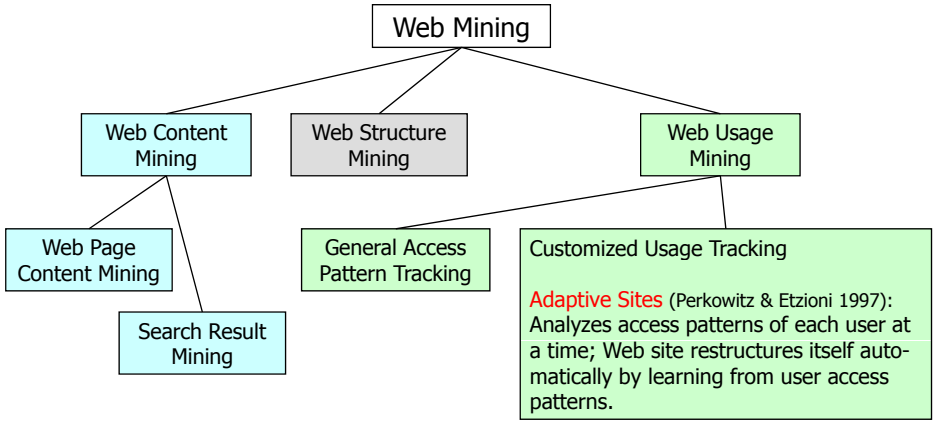
Mining the World-Wide Web



Mining the World-Wide Web



Mining the World-Wide Web



Mining the Web's Link Structures

- Finding authoritative Web pages
  - Retrieving pages that are not only relevant, but also of high quality, or authoritative on the topic
- Hyperlinks can infer the notion of authority
  - The Web consists not only of pages, but also of hyperlinks pointing from one page to another
  - These hyperlinks contain an enormous amount of latent human annotation
  - A hyperlink pointing to another Web page can be considered as the author's endorsement (Bestätigung) of the other page

Mining the Web's Link Structures

- Problems with the Web linkage structure
  - Not every hyperlink represents an endorsement
    - Other purposes are for navigation or for paid advertisements
    - If the majority of hyperlinks are for endorsement, the collective opinion will still dominate
  - One authority will seldom have its Web page point to its rival authorities in the same field
  - Authoritative pages are seldom particularly descriptive
- Hub
  - Set of Web pages that provides collections of links to authorities

HITS (Hyperlink-Induced Topic Search)

- Explore interactions between hubs and authoritative pages
- Use an index-based search engine to form the root set
  - Many of these pages are presumably relevant to the search topic
  - Some of them should contain links to most of the prominent authorities
- Expand the root set into a base set
  - Include all of the pages that the root-set pages link to, and all of the pages that link to a page in the root set, up to a designated size cutoff
- Apply weight-propagation
  - An iterative process that determines numerical estimates of hub and authority weights

Systems Based on HITS

- Output a short list of the pages with large hub weights, and the pages with large authority weights for the given search topic
- Systems based on the HITS algorithm
  - Clever, Google: achieve better quality search results than those generated by term-index engines such as AltaVista and those created by human ontologists such as Yahoo!
- Difficulties from ignoring textual contexts
  - Drifting: when hubs contain multiple topics
  - Topic hijacking: when many pages from a single Web site point to the same single popular site

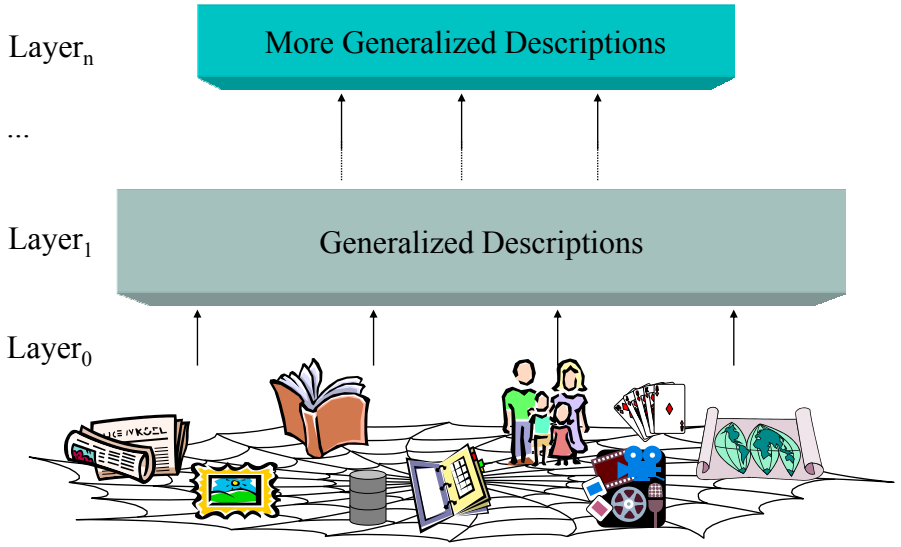
Automatic Classification of Web Documents

- Assign a class label to each document from a set of predefined topic categories
- Based on a set of examples of preclassified documents
- Example
  - Use Yahoo!'s taxonomy and its associated documents as training and test sets
  - Derive a Web document classification scheme
  - Use the scheme classify new Web documents by assigning categories from the same taxonomy
- Keyword-based document classification methods
- Statistical models

Multilayered Web Information Base

- $Layer_0$: the Web itself
- $Layer_1$: the Web page descriptor layer
  - Contains descriptive information for pages on the Web
  - An abstraction of $Layer_0$: substantially smaller but still rich enough to preserve most of the interesting, general information
  - Organized into dozens of semistructured classes
    - *document, person, organization, ads, directory, sales, software, game, stocks, library_catalog, geographic_data, scientific_data*, etc.
- $Layer_2$ and up: various Web directory services constructed on top of $Layer_1$
  - provide multidimensional, application-specific services

Multiple Layered Web Architecture



Mining the World-Wide Web

Layer-0: Primitive data

Layer-1: dozen database relations representing types of objects (metadata)

document, organization, person, software, game, map, image,…

• **document**(file_addr, authors, title, publication, publication_date, abstract, language, table_of_contents, category_description, keywords, index, multimedia_attached, num_pages, format, first_paragraphs, size_doc, timestamp, access_frequency, links_out,...)

• **person**(last_name, first_name, home_page_addr, position, picture_attached, phone, e-mail, office_address, education, research_interests, publications, size_of_home_page, timestamp, access_frequency, ...)

• **image**(image_addr, author, title, publication_date, category_description, keywords, size, width, height, duration, format, parent_pages, colour_histogram, Colour_layout, Texture_layout, Movement_vector, localisation_vector, timestamp, access_frequency, ...)

Mining the World-Wide Web

Layer-2: simplification of layer-1

•**doc_brief**(file_addr, authors, title, publication, publication_date, abstract, language, category_description, key_words, major_index, num_pages, format, size_doc, access_frequency, links_out)

•**person_brief** (last_name, first_name, publications,affiliation, e-mail, research_interests, size_home_page, access_frequency)

Layer-3: generalization of layer-2

•**cs_doc**(file_addr, authors, title, publication, publication_date, abstract, language, category_description, keywords, num_pages, form, size_doc, links_out)

•**doc_summary**(affiliation, field, publication_year, count, first_author_list, file_addr_list)

•**doc_author_brief**(file_addr, authors, affiliation, title, publication, pub_date, category_description, keywords, num_pages, format, size_doc, links_out)

•**person_summary**(affiliation, research_interest, year, num_publications, count)

XML and Web Mining

- XML can help to extract the correct descriptors
  - Standardization would greatly facilitate information extraction

    <NAME> eXtensible Markup Language**</NAME>**
    <RECOM>World-Wide Web Consortium**</RECOM>**
    <SINCE>1998**</SINCE>**
    <VERSION>1.0**</VERSION>**
    <DESC>Meta language that facilitates more meaningful and precise declarations of document content**</DESC>**
    <HOW>Definition of new tags and DTDs**</HOW>**

  - Potential problem
    - XML can help solve heterogeneity for vertical applications, but the freedom to define tags can make horizontal applications on the Web more heterogeneous

Benefits of Multi-Layer Meta-Web

- Benefits:
  - Multi-dimensional Web info summary analysis
  - Approximate and intelligent query answering
  - Web high-level query answering (WebSQL, WebML)
  - Web content and structure mining
  - Observing the dynamics/evolution of the Web
- Is it realistic to construct such a meta-Web?
  - Benefits even if it is partially constructed
  - Benefits may justify the cost of tool development, standardization and partial restructuring

Web Usage Mining

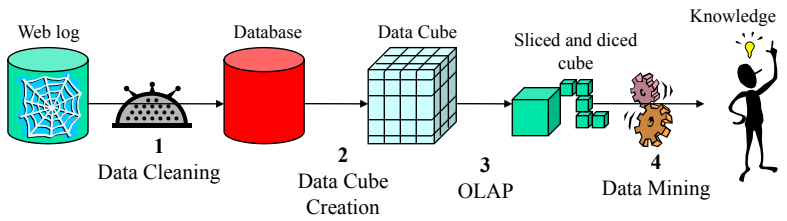
- Mining Web log records to discover user access patterns of Web pages
- Applications
  - Target potential customers for electronic commerce
  - Enhance the quality and delivery of Internet information services to the end user
  - Improve Web server system performance
  - Identify potential prime advertisement locations
- Web logs provide rich information about Web dynamics
  - Typical Web log entry includes the URL requested, the IP address from which the request originated, and a timestamp

Techniques for Web usage mining

- Construct multidimensional view on the Weblog database
  - Perform multidimensional OLAP analysis to find the top *N* users, top *N* accessed Web pages, most frequently accessed time periods, etc.
- Perform data mining on Weblog records
  - Find association patterns, sequential patterns, and trends of Web accessing
  - May need additional information,e.g., user browsing sequences of the Web pages in the Web server buffer
- Conduct studies to
  - Analyze system performance, improve system design by Web caching, Web page prefetching, and Web page swapping

Mining the World-Wide Web

- Design of a Web Log Miner
  - Web log is filtered to generate a relational database
  - A data cube is generated form database
  - OLAP is used to drill-down and roll-up in the cube
  - OLAM is used for mining interesting knowledge



Chapter 7. Mining Complex Types of Data

- Multidimensional analysis and descriptive mining of complex data objects
- Mining text databases
- Mining the World-Wide Web
- Summary

Summary

- Mining complex types of data include object data, spatial data, multimedia data, time-series data, text data, and Web data
- Text mining goes beyond keyword-based and similarity-based information retrieval and discovers knowledge from semi-structured data using methods like keyword-based association and document classification
- Web mining includes mining Web link structures to identify authoritative Web pages, the automatic classification of Web documents, building a multilayered Web information base, and Weblog mining

References (1)

- R. Agrawal, C. Faloutsos, and A. Swami. Efficient similarity search in sequence databases. In Proc. 4th Int. Conf. Foundations of Data Organization and Algorithms, Chicago, Oct. 1993.
- R. Agrawal, K.-I. Lin, H.S. Sawhney, and K. Shim. Fast similarity search in the presence of noise, scaling, and translation in time-series databases. VLDB'95, Zurich, Switzerland, Sept. 1995.
- G. Arocena and A. O. Mendelzon. WebOQL : Restructuring documents, databases, and webs. ICDE'98, Orlando, FL, Feb. 1998.
- R. Agrawal, G. Psaila, E. L. Wimmers, and M. Zait. Querying shapes of histories. VLDB'95, Zurich, Switzerland, Sept. 1995.
- R. Agrawal and R. Srikant. Mining sequential patterns. ICDE'95, Taipei, Taiwan, Mar. 1995.
- S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. WWW'98, Brisbane, Australia, 1998.
- C. Bettini, X. Sean Wang, and S. Jajodia. Mining temporal relationships with multiple granularities in time sequences. Data Engineering Bulletin, 21:32-38, 1998.
- R. Baeza-Yates and B. Ribeiro-Neto. Modern Information Retrieval. Addison-Wesley, 1999.
- S. Chakrabarti, B. E. Dom, and P. Indyk. Enhanced hypertext classification using hyper-links. SIGMOD'98, Seattle, WA, June 1998.
- S. Chakrabarti, B. E. Dom, S. R. Kumar, P. Raghavan, S. Rajagopalan, A. Tomkins, D. Gibson, and J. M. Kleinberg. Mining the web's link structure. COMPUTER, 32:60-67, 1999.

References (2)

- J. Chen, D. DeWitt, F. Tian, and Y. Wang. NiagraCQ: A scalable continuous query system for internet databases. SIGMOD'00, Dallas, TX, May 2000.
- C. Chatfield. The Analysis of Time Series: An Introduction, 3rd ed. Chapman and Hall, 1984.
- S. Chakrabarti. Data mining for hypertex: A tutorial survey. SIGKDD Explorations, 1:1-11, 2000.
- S. Deerwester, S. Dumais, G. Furnas, T. Landauer, and R. Harshman. Indexing by latent semantic analysis. J. American Society for Information Science, 41:391-407, 1990.
- M. Ester, A. Frommelt, H.-P. Kriegel, and J. Sander. Algorithms for claracterization and trend detection in spatial databases. KDD'98, New York, NY, Aug. 1998.
- M.J. Egenhofer. Spatial Query Languages. UMI Research Press, University of Maine, Portland, Maine, 1989.
- M. Ester, H.-P. Kriegel, and J. Sander. Spatial data mining: A database approach. SSD'97, Berlin, Germany, July 1997.
- C. Faloutsos. Access methods for text. ACM Comput. Surv., 17:49-74, 1985.
- U. M. Fayyad, S. G. Djorgovski, and N. Weir. Automating the analysis and cataloging of sky surveys. In U.M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors, Advances in Knowledge Discovery and Data Mining, AAAI/MIT Press, 1996.
- R. Feldman and H. Hirsh. Finding associations in collectionds of text. In R. S. Michalski, I. Bratko, and M. Kubat, editors, "Machine Learning and Data Mining: Methods and Applications", John Wiley Sons, 1998.

References (3)

- C. Faloutsos and K.-I. Lin. FastMap: A fast algorithm for indexing, data-mining and visualization of traditional and multimedia datasets. SIGMOD'95, San Jose, CA, May 1995.
- D. Florescu, A. Y. Levy, and A. O. Mendelzon. Database techniques for the world-wide web: A survey. SIGMOD Record, 27:59-74, 1998.
- U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy (eds.). Advances in Knowledge Discovery and Data Mining. AAAI/MIT Press, 1996.
- C. Faloutsos, M. Ranganathan, and Y. Manolopoulos. Fast subsequence matching in time-series databases. SIGMOD'94, Minneapolis, Minnesota, May 1994.
- M. Flickner, H. Sawhney, W. Niblack, J. Ashley, B. Dom, Q. Huang, M. Gorkani, J. Hafner, D. Lee, D. Petkovic, S. Steele, and P. Yanker. Query by image and video content: The QBIC system. IEEE Computer, 28:23-32, 1995.
- S. Guha, R. Rastogi, and K. Shim. Rock: A robust clustering algorithm for categorical attributes. ICDE'99, Sydney, Australia, Mar. 1999.
- R. H. Gueting. An introduction to spatial database systems. The VLDB Journal, 3:357-400, 1994.
- J. Han, G. Dong, and Y. Yin. Efficient mining of partial periodic patterns in time series database. ICDE'99, Sydney, Australia, Apr. 1999.
- J. Han, K. Koperski, and N. Stefanovic. GeoMiner: A system prototype for spatial data mining. SIGMOD'97, Tucson, Arizona, May 1997.

References (4)

- J. Han, S. Nishio, H. Kawano, and W. Wang. Generalization-based data mining in object-oriented databases using an object-cube model. Data and Knowledge Engineering, 25:55-97, 1998.
- J. Han, J. Pei, B. Mortazavi-Asl, Q. Chen, U. Dayal, and M.-C. Hsu. Freespan: Frequent pattern-projected sequential pattern mining. KDD'00, Boston, MA, Aug. 2000.
- J. Han, N. Stefanovic, and K. Koperski. Selective materialization: An efficient method for spatial data cube construction. PAKDD'98. Melbourne, Australia, Apr. 1998.
- J. Han, Q. Yang, and E. Kim. Plan mining by divide-and-conquer. DMKD'99, Philadelphia, PA, May 1999.
- K. Koperski and J. Han. Discovery of spatial association rules in geographic information databases. SSD'95, Portland, Maine, Aug. 1995.
- J. M. Kleinberg. Authoritative sources in a hyperlinked environment. Journal of ACM, 46:604-632, 1999.
- E. Knorr and R. Ng. Finding aggregate proximity relationships and commonalities in spatial data mining. IEEE Trans. Knowledge and Data Engineering, 8:884-897, 1996.
- J. M. Kleinberg and A. Tomkins. Application of linear algebra in information retrieval and hypertext analysis. PODS'99. Philadelphia, PA, May 1999.
- H. Lu, J. Han, and L. Feng. Stock movement and n-dimensional inter-transaction association rules. DMKD'98, Seattle, WA, June 1998.

References (5)

- W. Lu, J. Han, and B. C. Ooi. Knowledge discovery in large spatial databases. In Proc. Far East Workshop Geographic Information Systems, Singapore, June 1993.
- D. J. Maguire, M. Goodchild, and D. W. Rhind. Geographical Information Systems: Principles and Applications. Longman, London, 1992.
- H. Miller and J. Han. Geographic Data Mining and Knowledge Discovery. Taylor and Francis, 2000.
- A. O. Mendelzon, G. A. Mihaila, and T. Milo. Querying the world-wide web. Int. Journal of Digital Libraries, 1:54-67, 1997.
- H. Mannila, H Toivonen, and A. I. Verkamo. Discovery of frequent episodes in event sequences. Data Mining and Knowledge Discovery, 1:259-289, 1997.
- A. Natsev, R. Rastogi, and K. Shim. Walrus: A similarity retrieval algorithm for image databases. SIGMOD'99, Philadelphia, PA, June 1999.
- B. Ozden, S. Ramaswamy, and A. Silberschatz. Cyclic association rules. ICDE'98, Orlando, FL, Feb. 1998.
- M. Perkowitz and O. Etzioni. Adaptive web sites: Conceptual cluster mining. IJCAI'99, Stockholm, Sweden, 1999.
- P. Raghavan. Information retrieval algorithms: A survey. In Proc. 1997 ACM-SIAM Symp. Discrete Algorithms, New Orleans, Louisiana, 1997.

References (6)

- D. Rafiei and A. Mendelzon. Similarity-based queries for time series data. SIGMOD'97, Tucson, Arizona, May 1997.
- G. Salton. Automatic Text Processing. Addison-Wesley, 1989.
- J. Srivastava, R. Cooley, M. Deshpande, and P. N. Tan. Web usage mining: Discovery and applications of usage patterns from web data. SIGKDD Explorations, 1:12-23, 2000.
- P. Stolorz and C. Dean. Quakefinder: A scalable data mining system for detecting earthquakes from space. KDD'96, Portland, Oregon, Aug. 1996.
- G. Salton and M. McGill. Introduction to Modern Information Retrieval. McGraw-Hill, 1983.
- V. S. Subrahmanian. Principles of Multimedia Database Systems. Morgan Kaufmann, 1998.
- C. J. van Rijsbergen. Information Retrieval. Butterworth, 1990.
- K. Wang, S. Zhou, and S. C. Liew. Building hierarchical classifiers using class proximity. VLDB'99, Edinburgh, UK, Sept. 1999.
- B.-K. Yi, H. V. Jagadish, and C. Faloutsos. Efficient retrieval of similar time sequences under time warping. ICDE'98, Orlando, FL, Feb. 1998.
- C. T. Yu and W. Meng. Principles of Database Query Processing for Advanced Applications. Morgan Kaufmann, 1997.

References (7)

- B.-K. Yi, N. Sidiropoulos, T. Johnson, H. V. Jagadish, C. Faloutsos, and A. Biliris. Online data mining for co-evolving time sequences. ICDE'00, San Diego, CA, Feb. 2000.
- C. Zaniolo, S. Ceri, C. Faloutsos, R. T. Snodgrass, C. S. Subrahmanian, and R. Zicari. Advanced Database Systems. Morgan Kaufmann, 1997.
- O. R. Za"iane and J. Han. Resource and knowledge discovery in global information systems: A preliminary design and experiment. KDD'95, Montreal, Canada, Aug. 1995.
- O. R. Za"iane and J. Han. WebML : Querying the world-wide web for resources and knowledge. WIDM'98, Bethesda, Maryland, Nov. 1998.
- O. R. Za"iane, J. Han, Z. N. Li, J. Y. Chiang, and S. Chee. MultiMedia-Miner: A system prototype for multimedia data mining. SIGMOD'98, Seattle, WA, June 1998.
- O. R. Za"iane, J. Han, and H. Zhu. Mining recurrent items in multimedia with progressive resolution refinement. ICDE'00, San Diego, CA, Feb. 2000.
- M. J. Zaki, N. Lesh, and M. Ogihara. PLANMINE: Sequence mining for plan failures. KDD'98, New York, NY, Aug. 1998.
- X. Zhou, D. Truffet, and J. Han. Efficient polygon amalgamation methods for spatial OLAP and spatial data mining. SSD'99. Hong Kong, July 1999.
- O. R. Za"iane, M. Xin, and J. Han. Discovering Webaccess patterns and trends by applying OLAP and data mining technology on Web logs. ADL'98, Santa Barbara, CA, Apr. 1998.