UNIT -1

Why data mining?

Evolution of Database Technology

1960s and earlier:

Data Collection and Database Creation

– Primitive file processing

Evolution of Database Technology

1970s - early 1980s:

Data Base Management Systems

- Hieratical and network database systems
- Relational database Systems
- Query languages: SQL
- Transactions, concurrency control and recovery.
- On-line transaction processing (OLTP)

Evolution of Database Technology

Mid -1980s - present:

– Advanced data models

Extended relational, object-relational

– Advanced application-oriented DBMS

spatial, scientific, engineering, temporal, multimedia, active, stream and sensor, knowledge-based

Evolution of Database Technology

Late 1980s-present

- Advanced Data Analysis
  - Data warehouse and OLAP
  - Data mining and knowledge discovery
  - Advanced data mining appliations
  - Data mining and socity

1990s-present:

- XML-based database systems
- Integration with information retrieval
- Data and information integreation

Evolution of Database Technology

Present – future:

- New generation of integrated data and information system.

What Is Data Mining?

What Is Data Mining?

Data mining refers to extracting or mining knowledge from large amounts of data.

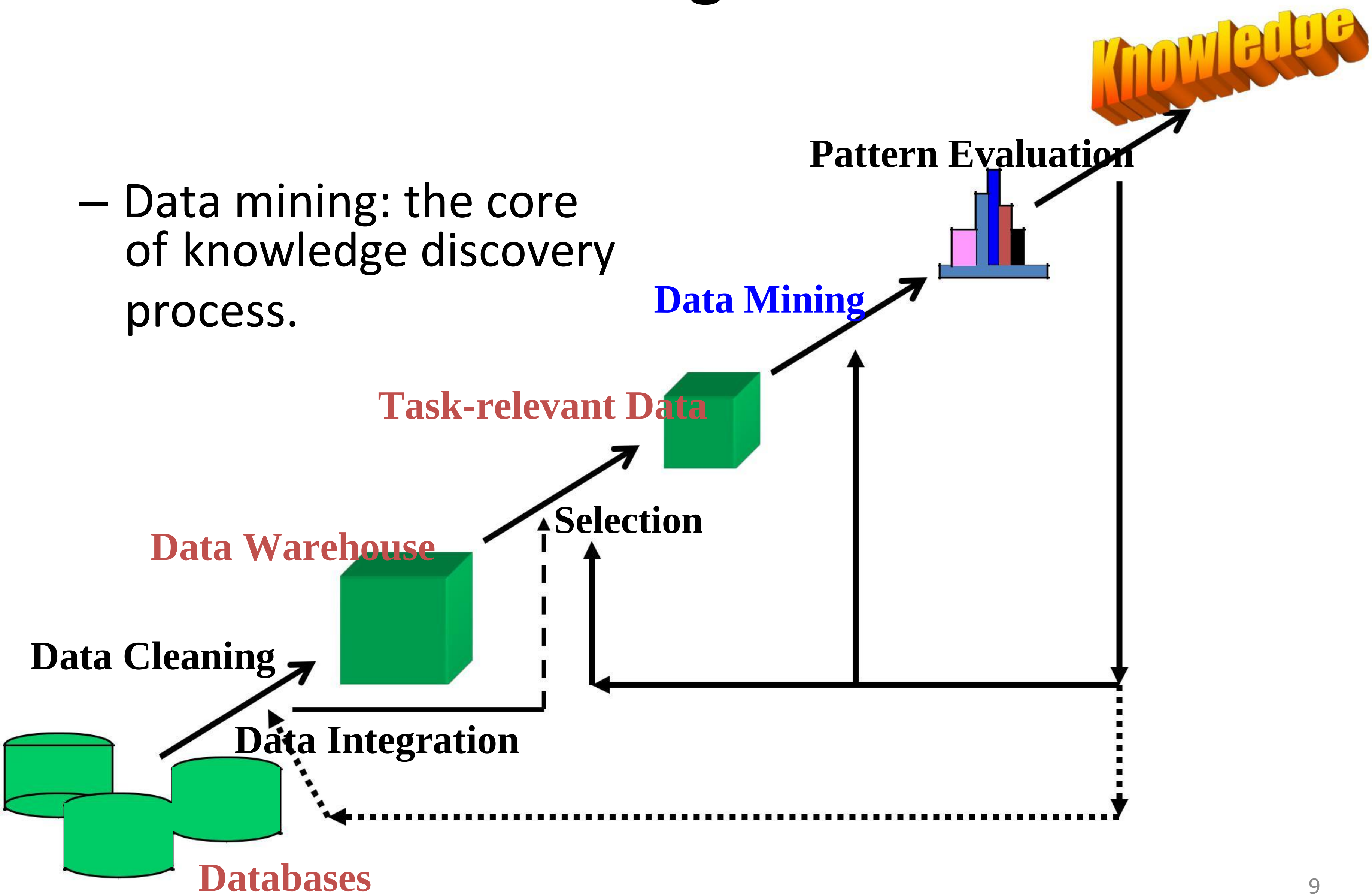
Mining of gold from rocks or sand

Knowledge mining from data, knowledge extraction, data/pattern analysis, data archeology, and data dreding.

Knowledge Discovery from data, or KDD

Data Mining: A KDD Process

– Data mining: the core of knowledge discovery process.



Steps of a KDD Process

Data cleaning

Data integration

Data selection

Data transformation

Data mining

Pattern evaluation

Knowledge presentaion

Steps of a KDD Process

Learning the application domain:
- relevant prior knowledge and goals of application

Creating a target data set: data selection

Data cleaning and preprocessing

Data reduction and transformation:
- Find useful features, dimensionality/variable reduction, invariant representation.

Steps of a KDD Process

Choosing functions of data mining

– summarization, classification, regression, association, clustering.

Choosing the mining algorithms

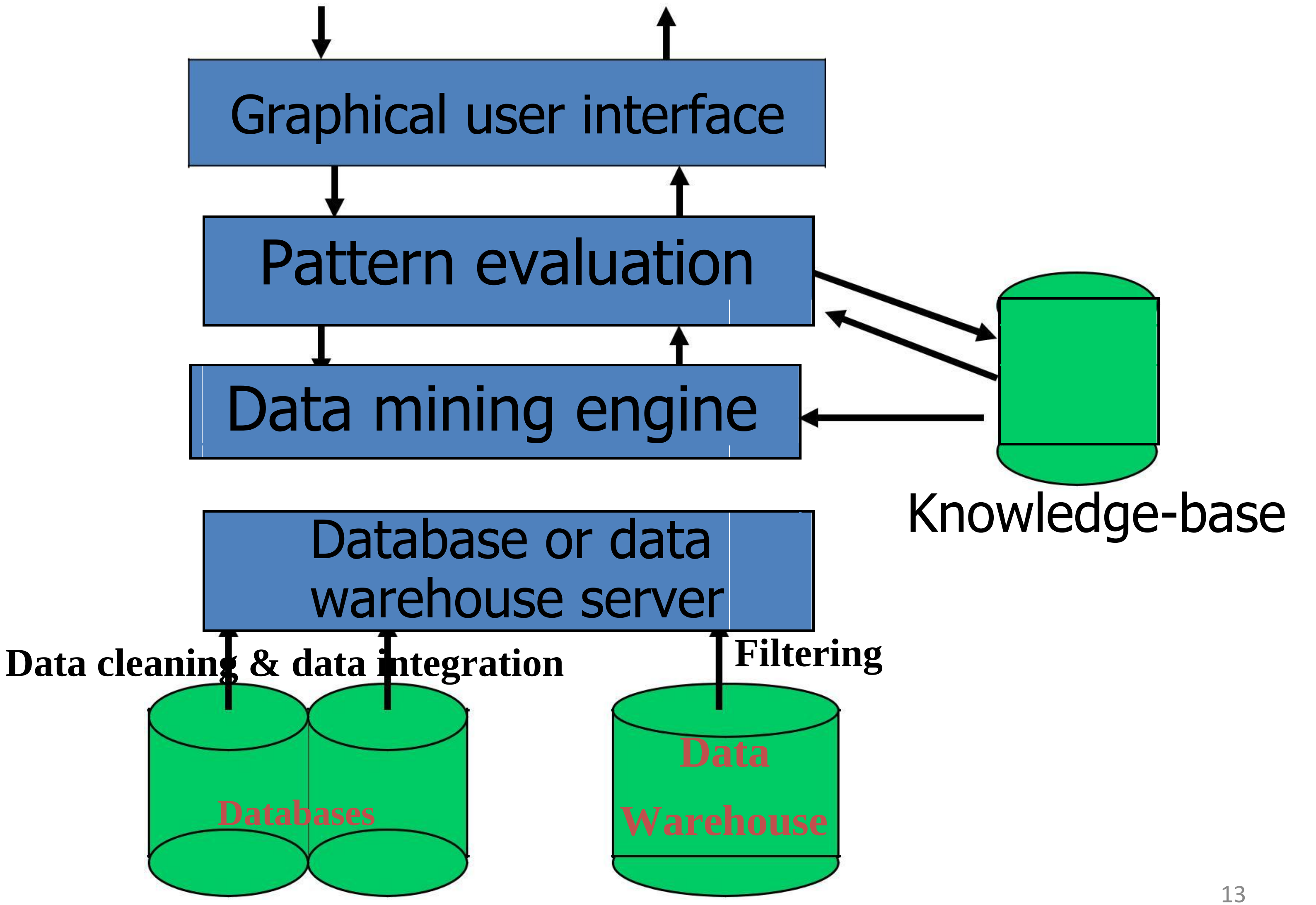
Data mining: search for patterns of interest

Pattern evaluation and knowledge presentation

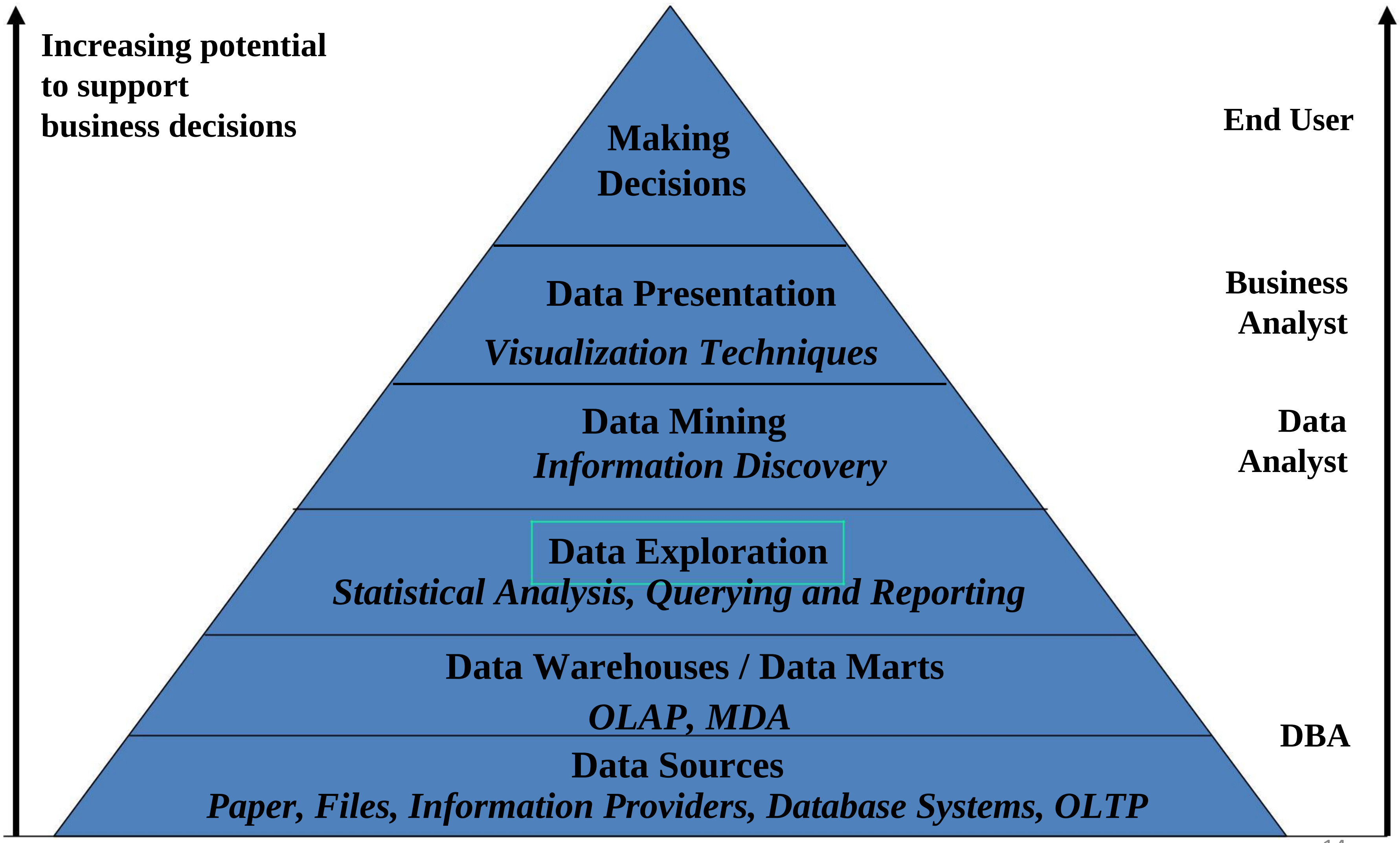
– visualization, transformation, removing redundant patterns, etc.

Use of discovered knowledge

Architecture of a Typical Data Mining System



Data Mining and Business Intelligence



Data Mining: On What Kind of Data?

Data Mining: On What Kind of Data?

Relational databases

Data warehouses

Transactional databases

Data Mining: On What Kind of Data?

Advanced DB and information repositories

- Object-oriented and object-relational databases
- Spatial databases
- Time-series data and temporal data
- Text databases and multimedia databases
- Heterogeneous and legacy databases
- WWW

Data mining primitives: What defines a data mining task?

Why Data Mining Primitives and Languages?

Finding all the patterns autonomously in a database? — unrealistic because the patterns could be too many but uninteresting

Data mining should be an interactive process

- User directs what to be mined

Users must be provided with a set of primitives to be used to communicate with the data mining system

Incorporating these primitives in a data mining query language

- More flexible user interaction
- Foundation for design of graphical user interface
- Standardization of data mining industry and practice

What Defines a Data Mining Task ?

Task-relevant data

Type of knowledge to be mined

Background knowledge

Pattern interestingness measurements

Visualization of discovered patterns

Task-Relevant Data (Minable View)

Database or data warehouse name

Database tables or data warehouse cubes

Condition for data selection

Relevant attributes or dimensions

Data grouping criteria

Types of knowledge to be mined

Characterization

Discrimination

Association

Classification/prediction

Clustering

Outlier analysis

Other data mining tasks

Background Knowledge: Concept Hierarchies

Ā □ Ā □

schema hierarchy

– street < city < province_or_state < country

Ā □ Ā □

set-grouping hierarchy

– {20-39} = young, {40-59} = middle_aged

Ā □ Ā □

operation-derived hierarchy

– email address: login-name < department < university < country

Ā □ Ā □

ule-based hierarchy

– low_profit_margin (X) <= price(X, P1) and cost (X, P2) and (P1 - P2) < $50

Measurements of Pattern Interestingness

- Simplicity

  association rule length, decision tree size

- Certainty

  confidence, P(A|B) = n(A and B)/ n (B), classification reliability or accuracy, certainty factor, rule strength, rule quality, discriminating weight

- Utility

  potential usefulness, support (association), noise threshold (description)

- Novelty

  not previously known, surprising (used to remove redundant rules, Canada vs. Vancouver rule implication support ratio

Visualization of Discovered Patterns

Different backgrounds/usages may require different forms of representation

– rules, tables, cross tabs, pie/bar chart

Concept hierarchy is also important

– Discovered knowledge might be more understandable when represented at high level of abstraction

– Interactive drill up/down, pivoting, slicing and dicing provide different perspective to data

Different kinds of knowledge require different representation: association, classification, clustering

A data mining query language

A Data Mining Query Language (DMQL)

Motivation

– A DMQL can provide the ability to support ad-hoc and interactive data mining

– By providing a standardized language like SQL

to achieve a similar effect like that SQL has on relational database

Foundation for system development and evolution

Facilitate information exchange, technology transfer, commercialization and wide acceptance

Design

– DMQL is designed with the primitives

Syntax for DMQL

Syntax for specification of

- task-relevant data
- the kind of knowledge to be mined
- concept hierarchy specification
- interestingness measure
- pattern presentation and visualization

  — a DMQL query

Syntax for task-relevant data specification

use database database_name, or use data warehouse data_warehouse_name

from relation(s)/cube(s) [where condition]

in relevance to att_or_dim_list

order by order_list

group by grouping_list

having condition

Syntax for specifying the kind of knowledge to be mined

Characterization

Mine_Knowledge_Specification ::=
mine characteristics [as pattern_name]
analyze measure(s)

Discrimination

Mine_Knowledge_Specification ::= mine
comparison [as pattern_name] for
target_class where target_condition
{versus contrast_class_i where contrast_condition_i}
analyze measure(s)

- Association

Mine_Knowledge_Specification ::=
mine associations [as pattern_name]

Syntax for specifying the kind of knowledge to be mined

- Classification

Mine_Knowledge_Specification ::=
mine classification [as pattern_name]
analyze classifying_attribute_or_dimension

Prediction

Mine_Knowledge_Specification ::=
mine prediction [as pattern_name]
analyze prediction_attribute_or_dimension
{set {attribute_or_dimension_i= value_i}}

Syntax for concept hierarchy specification

- To specify what concept hierarchies to use

  use hierarchy **<hierarchy>** for **<attribute_or_dimension>**

- use different syntax to define different type of hierarchies
  - schema hierarchies

    define hierarchy **time_hierarchy** on **date** as **[date,month quarter,year]**

  - set-grouping hierarchies

    define hierarchy **age_hierarchy** for **age** on **customer** as

    level1: {young, middle_aged, senior} < level0: all

    level2: {20, ..., 39} < level1: young

    level2: {40, ..., 59} < level1: middle_aged

    level2: {60, ..., 89} < level1: senior

Syntax for concept hierarchy specification

– operation-derived hierarchies

define hierarchy age_hierarchy for age on customer as

{age_category(1), ..., age_category(5)} := cluster(default, age, 5) < all(age)

Syntax for concept hierarchy specification

– rule-based hierarchies

define hierarchy profit_margin_hierarchy on item as
level_1: low_profit_margin < level_0: all
if (price - cost)< $50
level_1: medium-profit_margin < level_0: all
if ((price - cost) > $50) and ((price - cost) <= $250))
level_1: high_profit_margin < level_0: all
if (price - cost) > $250

Syntax for interestingness measure specification

Interestingness measures and thresholds can be specified by the user with the statement:

with <interest_measure_name> threshold = threshold_value

- **Example:**

with support threshold = 0.05

with confidence threshold = 0.7

Syntax for pattern presentation and visualization specification

- syntax which allows users to specify the display of discovered patterns in one or more forms

  display as <**result_form**>

- To facilitate interactive viewing at different concept level, the following syntax is defined:

  Multilevel_Manipulation ::= roll up on attribute_or_dimension
  | drill down on attribute_or_dimension
  | add attribute_or_dimension | drop attribute_or_dimension

The full specification of a DMQL query

use database AllElectronics_db

use hierarchy location_hierarchy for B.address

mine characteristics as customerPurchasing

analyze count%

in relevance to C.age, I.type, I.place_made

from customer C, item I, purchases P, items_sold S, works_at W, branch

where I.item_ID = S.item_ID and S.trans_ID = P.trans_ID

and P.cust_ID = C.cust_ID and P.method_paid = ``AmEx''

and P.empl_ID = W.empl_ID and W.branch_ID = B.branch_ID and B.address = ``Canada'' and I.price >= 100

with noise threshold = 0.05 display as table

Other Data Mining Languages & Standardization Efforts

Association rule language specifications

- MSQL (Imielinski & Virmani'99)
- MineRule (Meo Psaila and Ceri'96)
- Query flocks based on Datalog syntax (Tsur et al'98)

OLEDB for DM (Microsoft'2000)

- Based on OLE, OLE DB, OLE DB for OLAP
- Integrating DBMS, data warehouse and data mining

CRISP-DM (CRoss-Industry Standard Process for Data Mining)

- Providing a platform and process structure for effective data mining
- Emphasizing on deploying data mining technology to solve business problems

Design graphical user interfaces based on a data mining query language

Designing Graphical User Interfaces based on a data mining query language

What tasks should be considered in the design GUIs based on a data mining query language?

- Data collection and data mining query composition
- Presentation of discovered patterns
- Hierarchy specification and manipulation
- Manipulation of data mining primitives
- Interactive multilevel mining
- Other miscellaneous information

Architecture of data mining systems

Data Mining System Architectures

Coupling data mining system with DB/DW system

- No coupling—flat file processing,
- Loose coupling

  Fetching data from DB/DW
- Semi-tight coupling—enhanced DM performance

  Provide efficient implement a few data mining primitives in a DB/DW system- sorting, indexing, aggregation, histogram analysis, multiway join, precomputation of some stat functions

Data Mining System Architectures

Tight coupling—A uniform information processing environment

- DM is smoothly integrated into a DB/DW system, mining query is optimized based on mining query, indexing, query processing methods

Data Mining Functionalities

Data Mining Functionalities

Concept description: Characterization and discrimination

- Data can be associated with classes or concepts
- Ex. AllElectronics store classes of items for sale include computer and printers.
- Description of class or concept called class/concept description.
- Data characterization
- Data discrimination

Data Mining Functionalities

Mining Frequent Patterns, Associations, and Correlations

Frequent patters- patterns occurs frequently

Item sets, subsequences and substructures

Frequent item set

Sequential patterns

Structured patterns

Data Mining Functionalities

Association Analysis

- Multi-dimensional vs. single-dimensional association
- age(X, “20..29”) ^ income(X, “20..29K”) => buys(X, “PC”) [support = 2%, confidence = 60%]
- contains(T, “computer”) => contains(x, “software”) [support=1%, confidence=75%]

Data Mining Functionalities

Classification and Prediction

– Finding models (functions) that describe and distinguish data classes or concepts for predict the class whose label is unknown

– E.g., classify countries based on climate, or classify cars based on gas mileage

– Models: decision-tree, classification rules (if-then), neural network

– Prediction: Predict some unknown or missing numerical values

Data Mining Functionalities

Cluster analysis

– Analyze class-labeled data objects, clustering analyze data objects without consulting a known class label.

– Clustering based on the principle: maximizing the intra-class similarity and minimizing the interclass similarity

Data Mining Functionalities

Outlier analysis

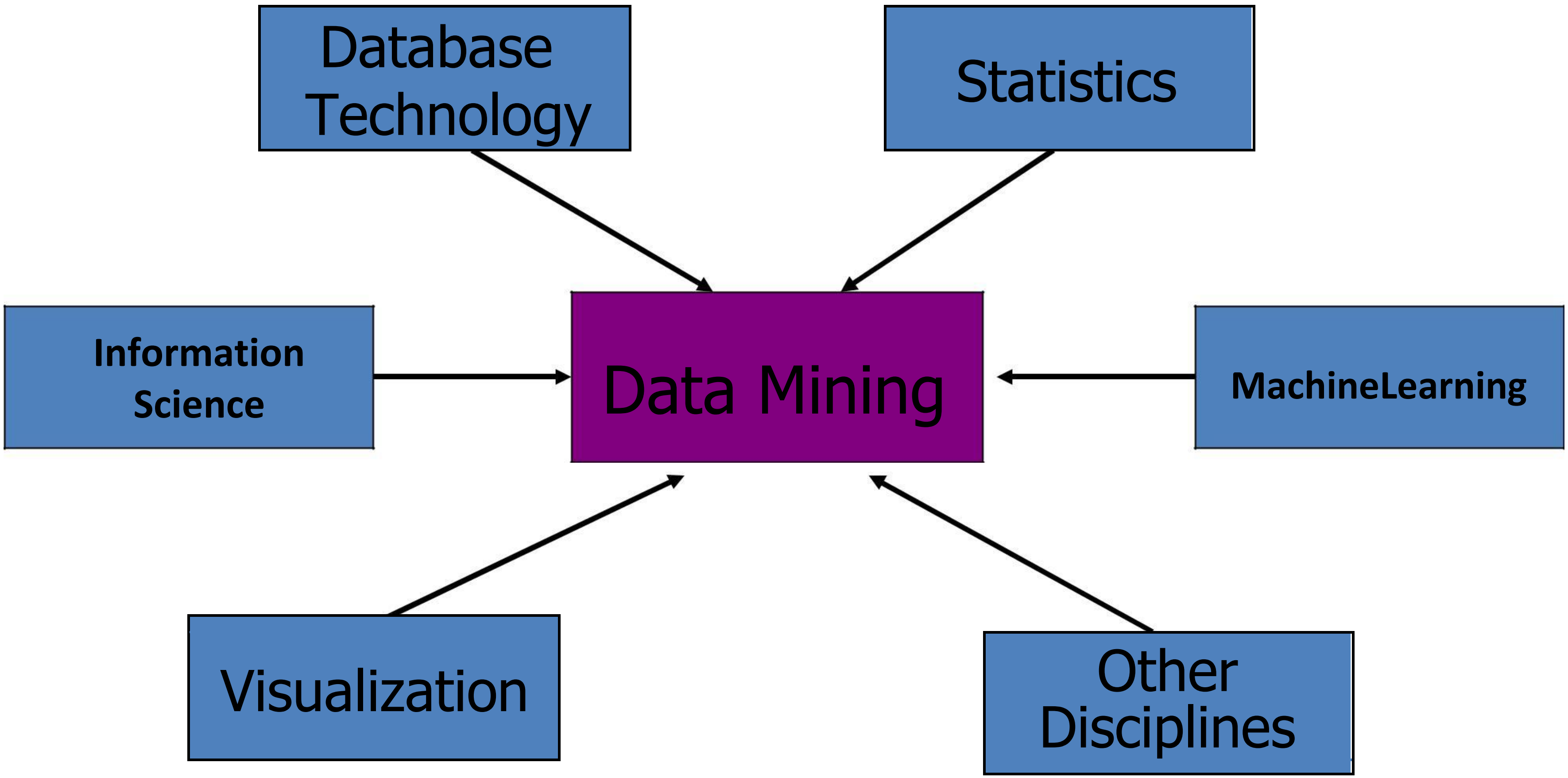
- Outlier: a data object that does not comply with the general behavior of the model of the data
- It can be considered as noise or exception but is quite useful in fraud detection, rare events analysis

Trend and evolution analysis

- Trend and deviation: regression analysis
- Sequential pattern mining, periodicity analysis
- Similarity-based analysis

Data Mining: Classification Schemes

Data Mining: Confluence of Multiple Disciplines



Data Mining: Classification Schemes

General functionality

– Descriptive data mining

– Predictive data mining

Data mining various criteria's:

– Kinds of databases to be mined

– Kinds of knowledge to be discovered

– Kinds of techniques utilized

– Kinds of applications adapted

Data Mining: Classification Schemes

Databases to be mined

– Relational, transactional, object-oriented, object-relational, active, spatial, time-series, text, multi-media, heterogeneous, legacy, WWW, etc.

Knowledge to be mined

– Characterization, discrimination, association, classification, clustering, trend, deviation and outlier analysis, etc.

– Multiple/integrated functions and mining at multiple levels

analysis, Web mining, Weblog analysis, etc.

Data Mining: Classification Schemes

Techniques utilized

– Database-oriented, data warehouse (OLAP), machine learning, statistics, visualization, neural network, etc.

Applications adapted

– Retail, telecommunication, banking, fraud analysis, DNA mining, stock market

Major Issues in Data Mining

Major Issues in Data Mining

Mining methodology and user interaction issues

– Mining different kinds of knowledge in databases

– Interactive mining of knowledge at multiple levels of abstraction

– Incorporation of background knowledge

– Data mining query languages and ad-hoc data mining

– Expression and visualization of data mining results

– Handling noise and incomplete data

– Pattern evaluation: the interestingness problem

Major Issues in Data Mining

Performance issues

- Efficiency and scalability of data mining algorithms
- Parallel, distributed and incremental mining methods

Major Issues in Data Mining

Issues relating to the diversity of data types

– Handling relational and complex types of data

– Mining information from heterogeneous databases and global information systems (WWW)

Why preprocess the data?

Lecture-13 Why Data Preprocessing?

Data in the real world is:

- 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
- Completeness
- Consistency
- Timeliness
- Believability
- Value added
- Interpretability
- Accessibility

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

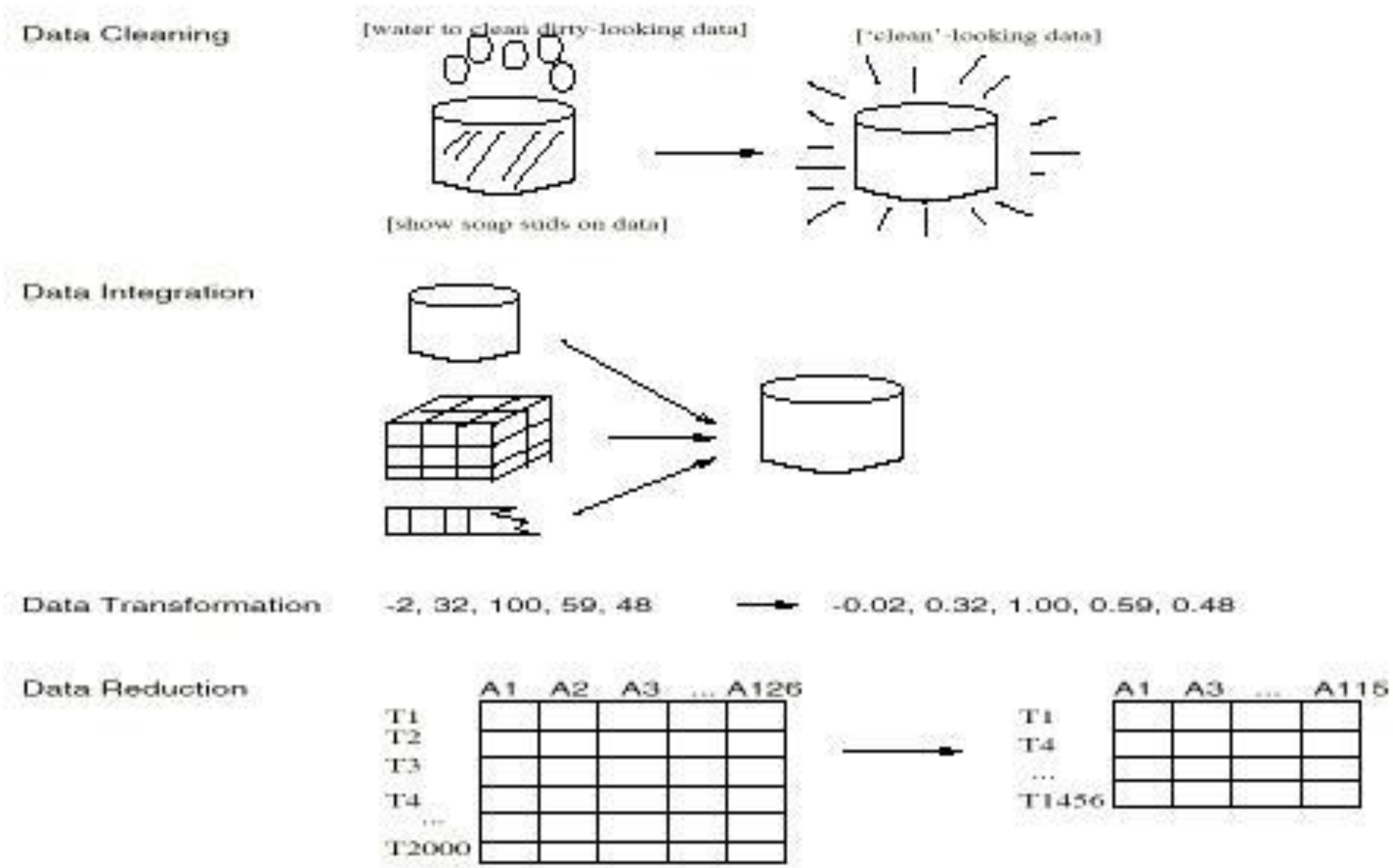
Data reduction

– 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

Forms of data preprocessing



Data cleaning

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

Fill in the missing value manually

Use a global constant to fill in the missing value: ex. “unknown”

How to Handle Missing Data?

Use the attribute mean to fill in the missing value

Use the attribute mean for all samples belonging to the same class to fill in the missing value

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 (equal-frequency) bins

– then one can smooth by bin means, smooth by bin median, smooth by bin boundaries

Clustering

– detect and remove outliers

Regression

– smooth by fitting the data to a regression functions – linear regression

Simple Discretization Methods: Binning

Equal-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
- But outliers may dominate presentation
- Skewed data is not handled well.

Equal-depth (frequency) partitioning:

- It divides the range into *N* intervals, each containing approximately same number of samples
- Good data scaling
- Managing categorical attributes can be tricky.

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

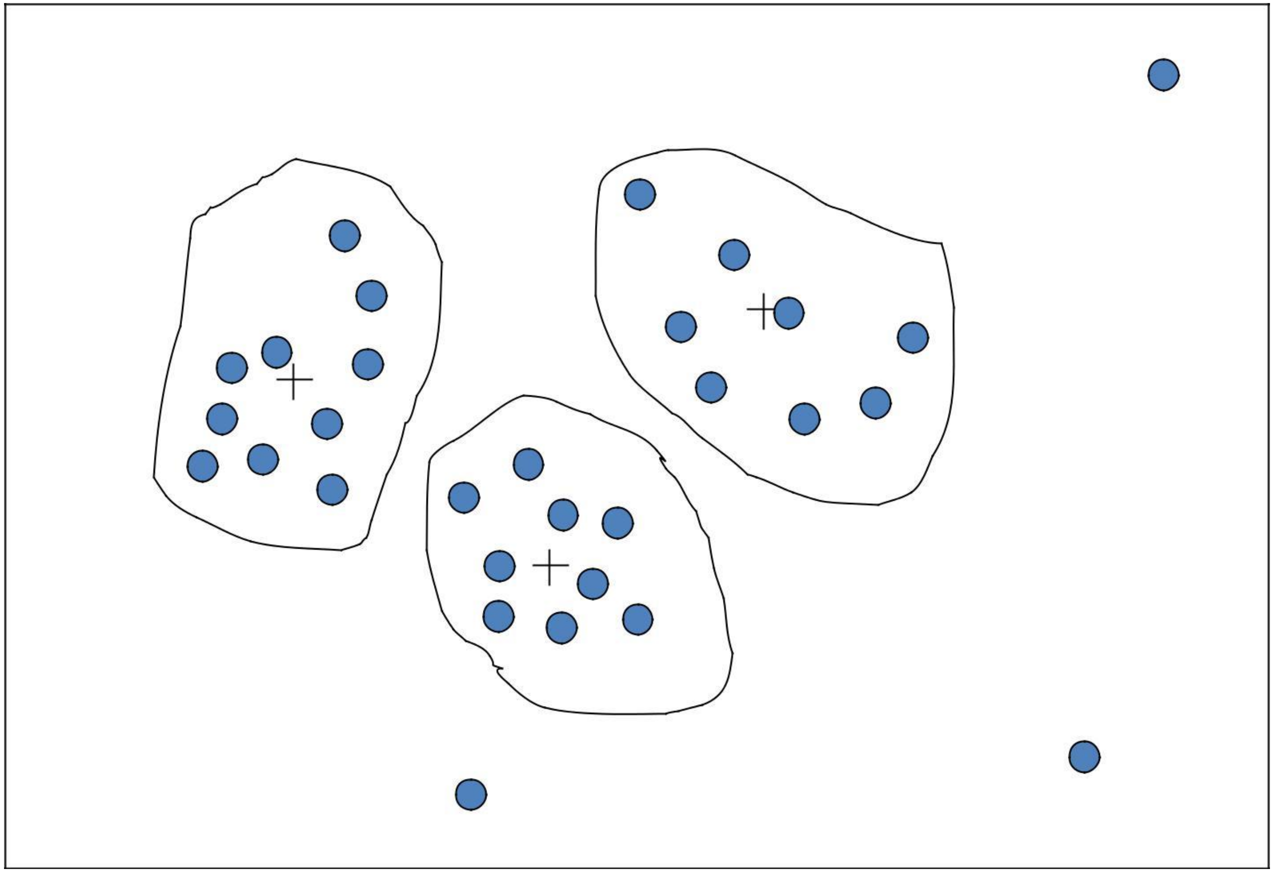
Smoothing by bin boundaries:

Bin 1: 4, 4, 4, 15

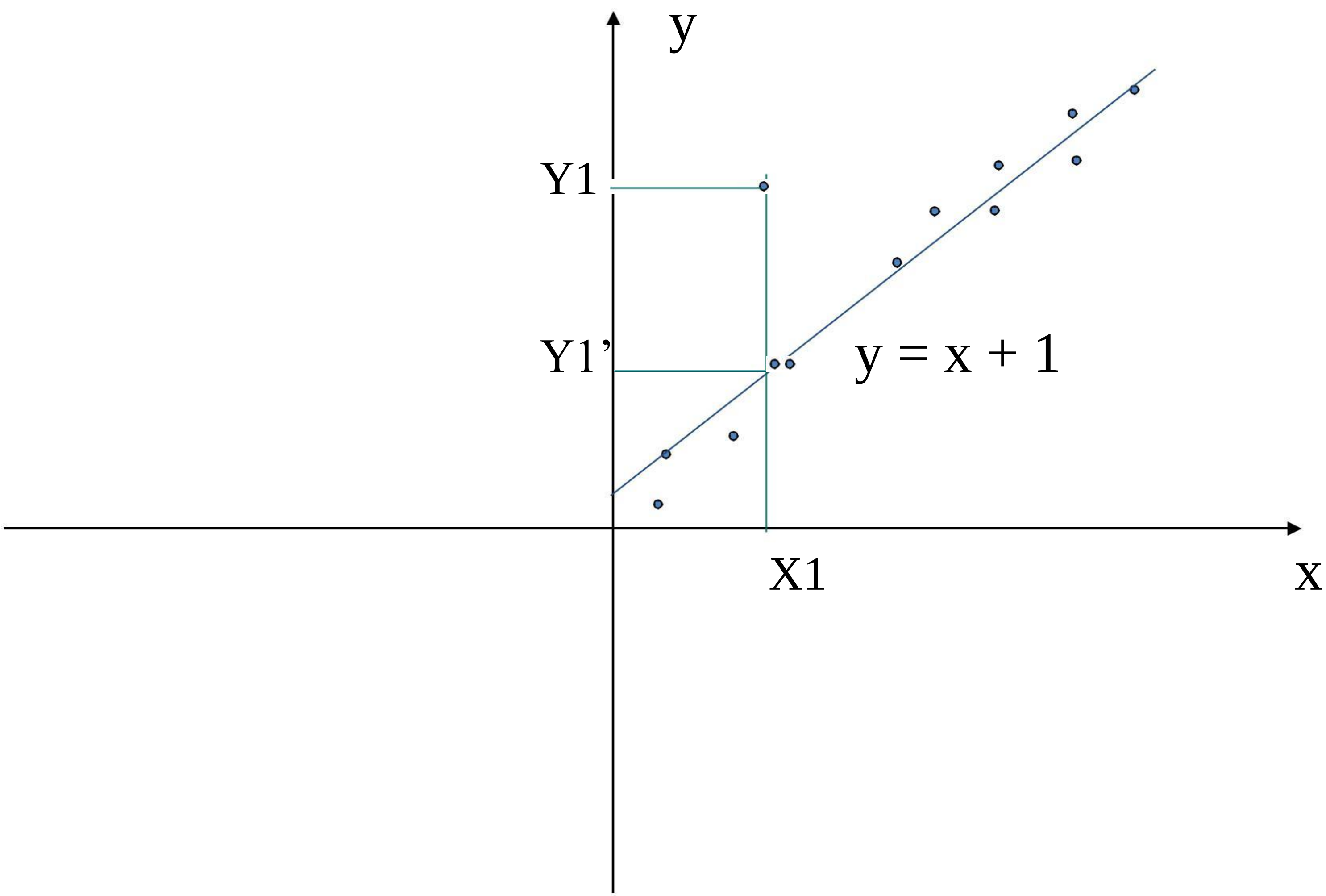
Bin 2: 21, 21, 25, 25

Bin 3: 26, 26, 26, 34

Cluster Analysis



Regression



Data integration and transformation

Data Integration

Data integration:

– combines data from multiple 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 integration of multiple databases

- The same attribute may have different names in different databases
- One attribute may be a “derived” attribute in another table, e.g., annual revenue

Handling Redundant Data in Data Integration

Redundant data may be able to be detected by correlation 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

Data Transformation

Normalization: scaled to fall within a small, specified range

- min-max normalization
- z-score normalization
- normalization by decimal scaling

Attribute/feature construction

- New attributes constructed from the given ones

Data Transformation: Normalization

- min-max normalization

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

- z-score normalization

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

- normalization by decimal scaling

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

Data reduction

Data Reduction

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 reduction strategies

- Data cube aggregation
- Attribute subset selection
- 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 (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 # of patterns in the patterns, easier to understand

Heuristic methods

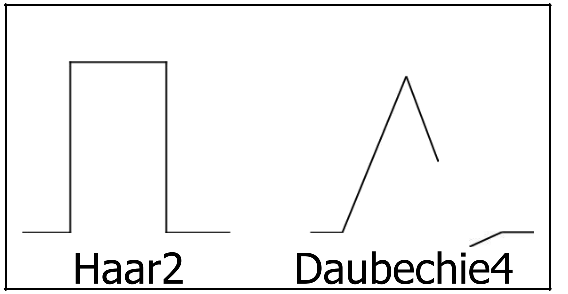
– step-wise forward selection

– step-wise backward elimination

– combining forward selection and backward elimination

– decision-tree induction

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

Given *N* data vectors from *k*-dimensions, find $c <= k$ orthogonal vectors that can be best used to represent data

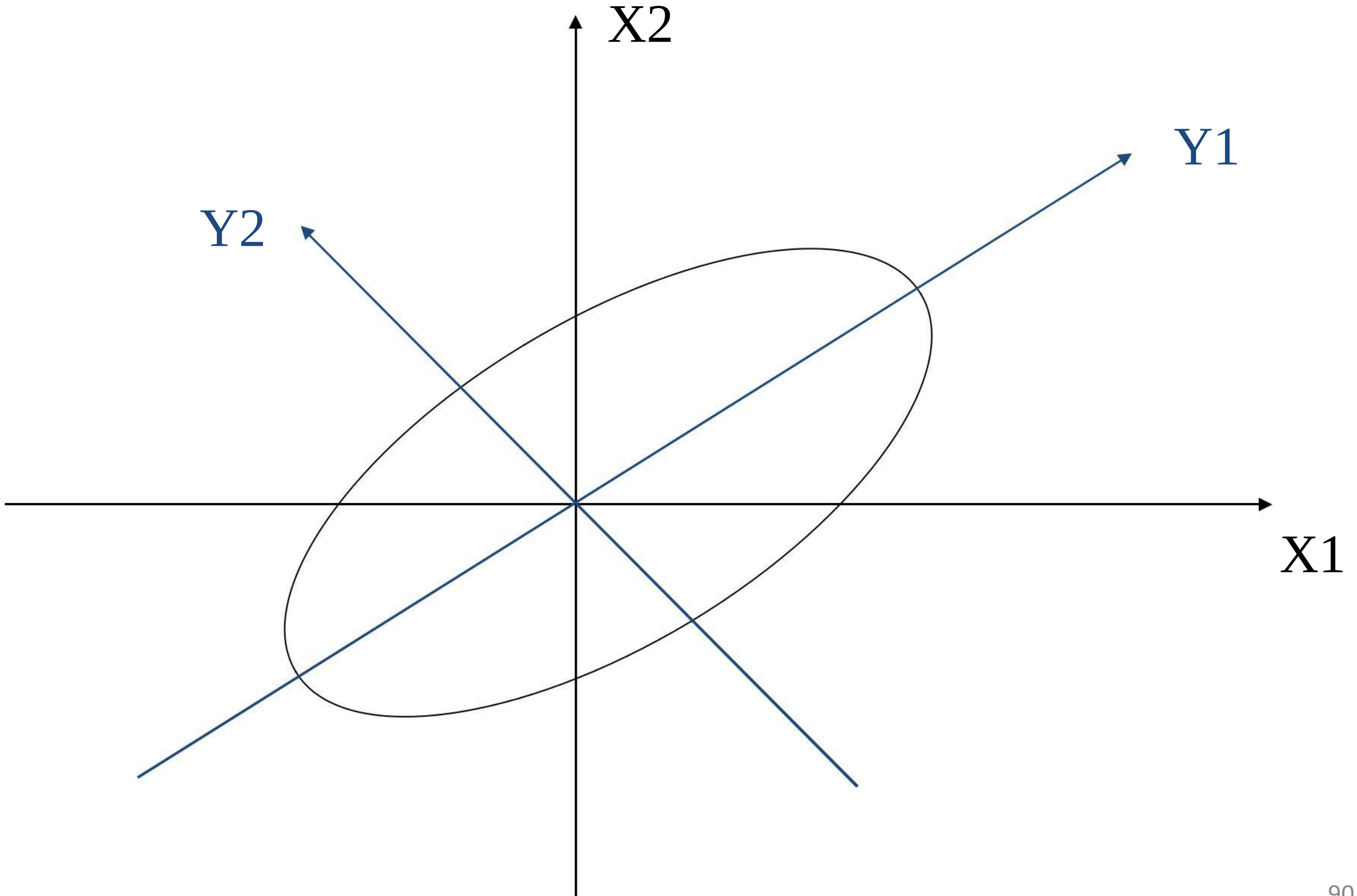
- The original data set is reduced to one consisting of N data vectors on c principal components (reduced dimensions)

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

Principal Component Analysis



Attribute subset selection

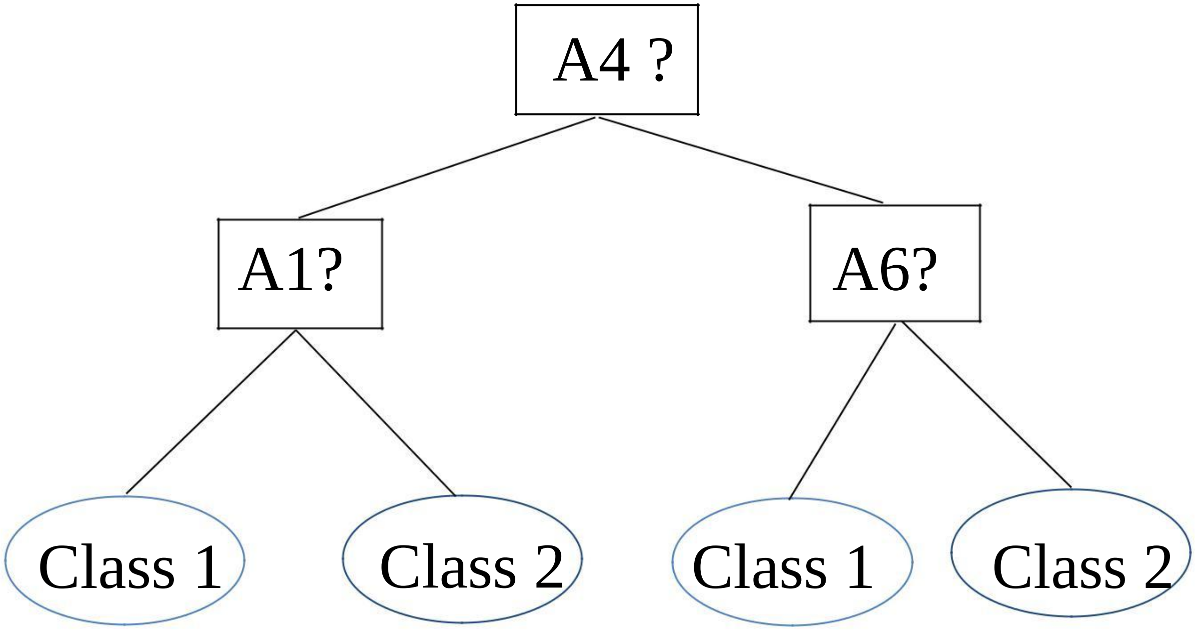
Attribute subset selection reduces the data set size by removing irrelevent or redundant attributes.

Goal is find min set of attributes

Uses basic heuristic methods of attribute selection

Example of Decision Tree Induction

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



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

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

Regression and Log-Linear Models

Linear regression: Data are modeled to fit a straight line

– Often uses the least-square method to fit the line

Multiple regression: allows a response variable Y to be modeled as a linear function of multidimensional feature vector

Log-linear model: approximates discrete multidimensional probability distributions

Regress Analysis and Log-Linear Models

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 $Y_1, Y_2, \ldots, X_1, X_2, \ldots$.

Multiple regression: $Y = b0 + b1\ X1 + b2\ X2.$

- 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.
- Probability: $p(a, b, c, d) = \alpha_{ab}\ \beta_{ac} \chi_{ad}\ \delta_{bcd}$

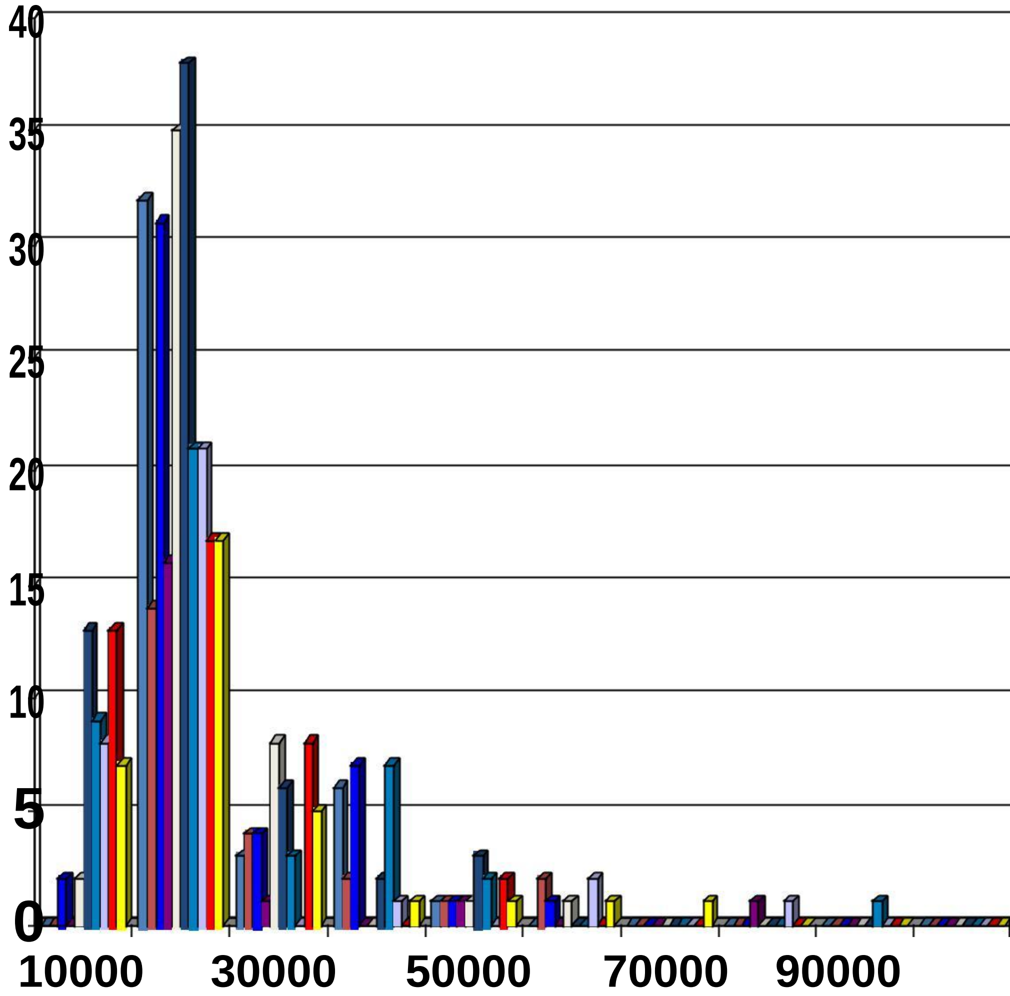
Histograms

A popular data reduction technique

Divide data into buckets and store average (sum) for each bucket

Can be constructed optimally in one dimension using dynamic programming

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.

Sampling

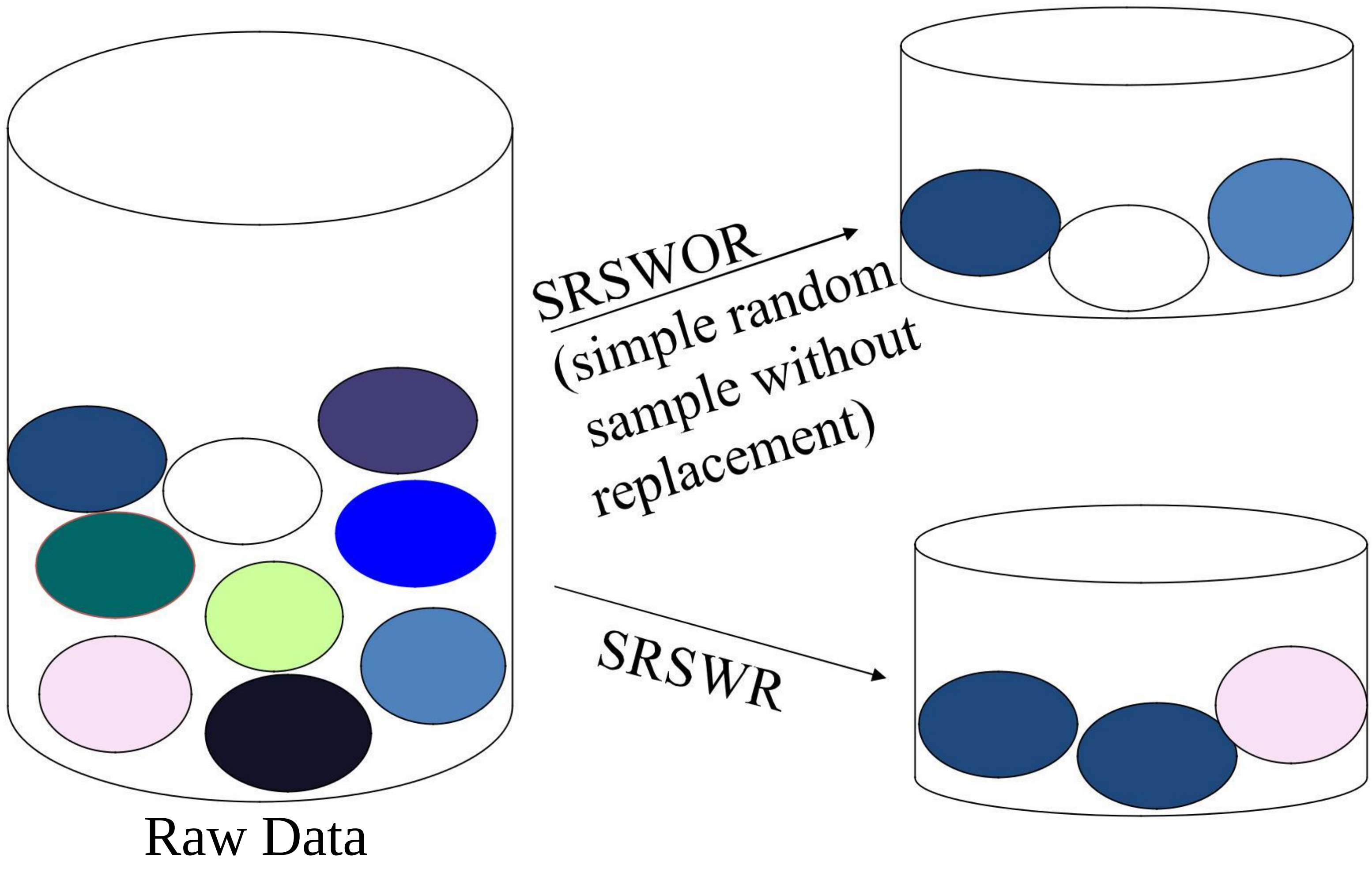
Allows a large data set to be represented by a much smaller of the data.

Let a large data set D, contains N tuples.

Methods to reduce data set D:

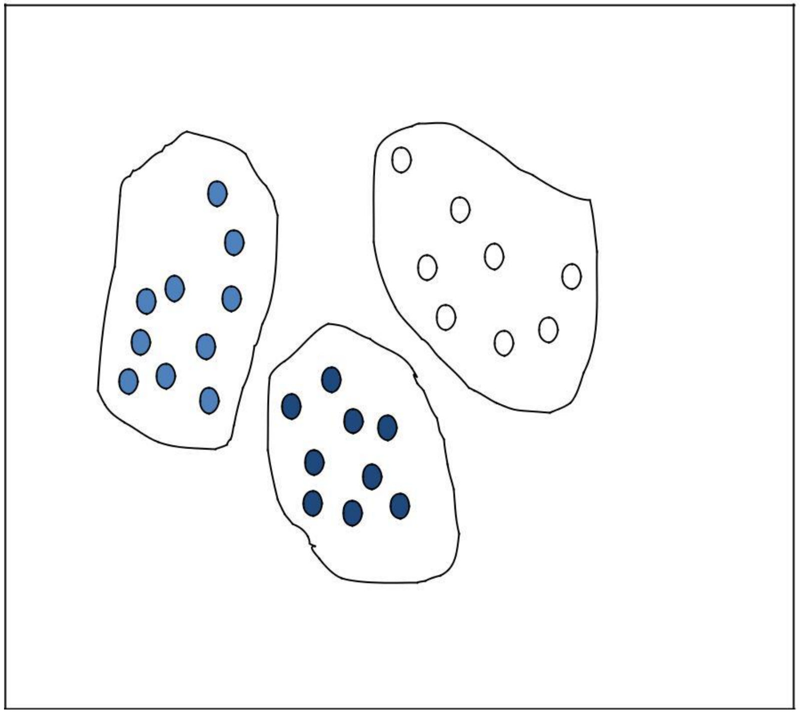
- Simple random sample without replacement (SRSWOR)
- Simple random sample with replacement (SRSWR)
- Cluster sample
- Stright sample

Sampling

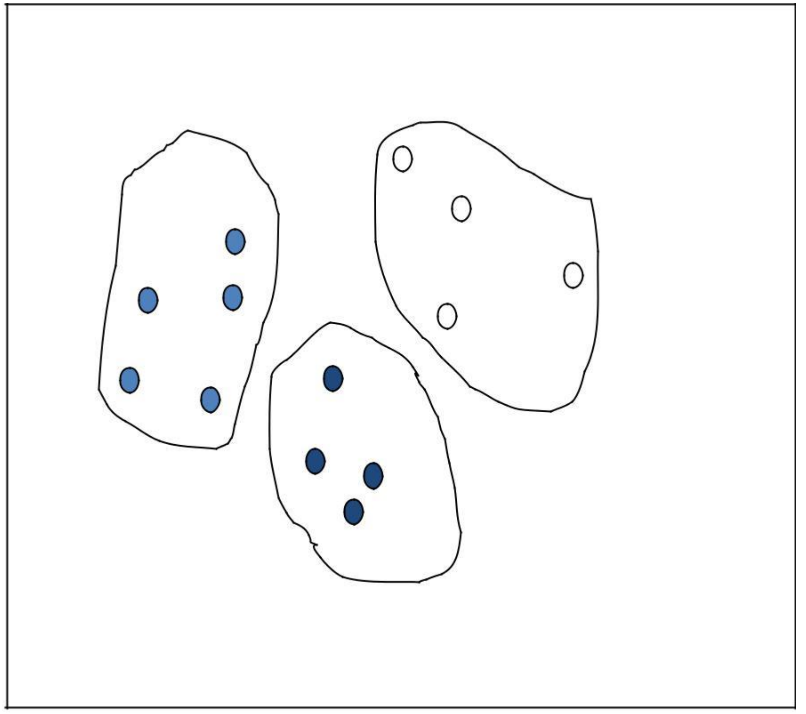

SRSWOR
(simple random
sample without
replacement)
SRSWR
Raw Data

Sampling

Raw Data



Cluster/Stratified Sample



Discretization and concept hierarchy generation

Discretization

Three types of attributes:

– 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

Histogram analysis

Clustering analysis

Entropy-based discretization

Discretization by intuitive partitioning

Entropy-Based Discretization

Given a set of samples S, if S is partitioned into two intervals S1 and S2 using boundary T, the entropy after partitioning is

$$E(S,T) = \frac{|S_1|}{|S|} Ent(S_1) + \frac{|S_2|}{|S|} Ent(S_2)$$

- The boundary that minimizes the entropy function over all possible boundaries is selected as a binary discretization.
- The process is recursively applied to partitions obtained until some stopping criterion is met, e.g., $Ent(S) - E(T, S) > \delta$
- Experiments show that it may reduce data size and improve classification accuracy

Discretization by intuitive partitioning

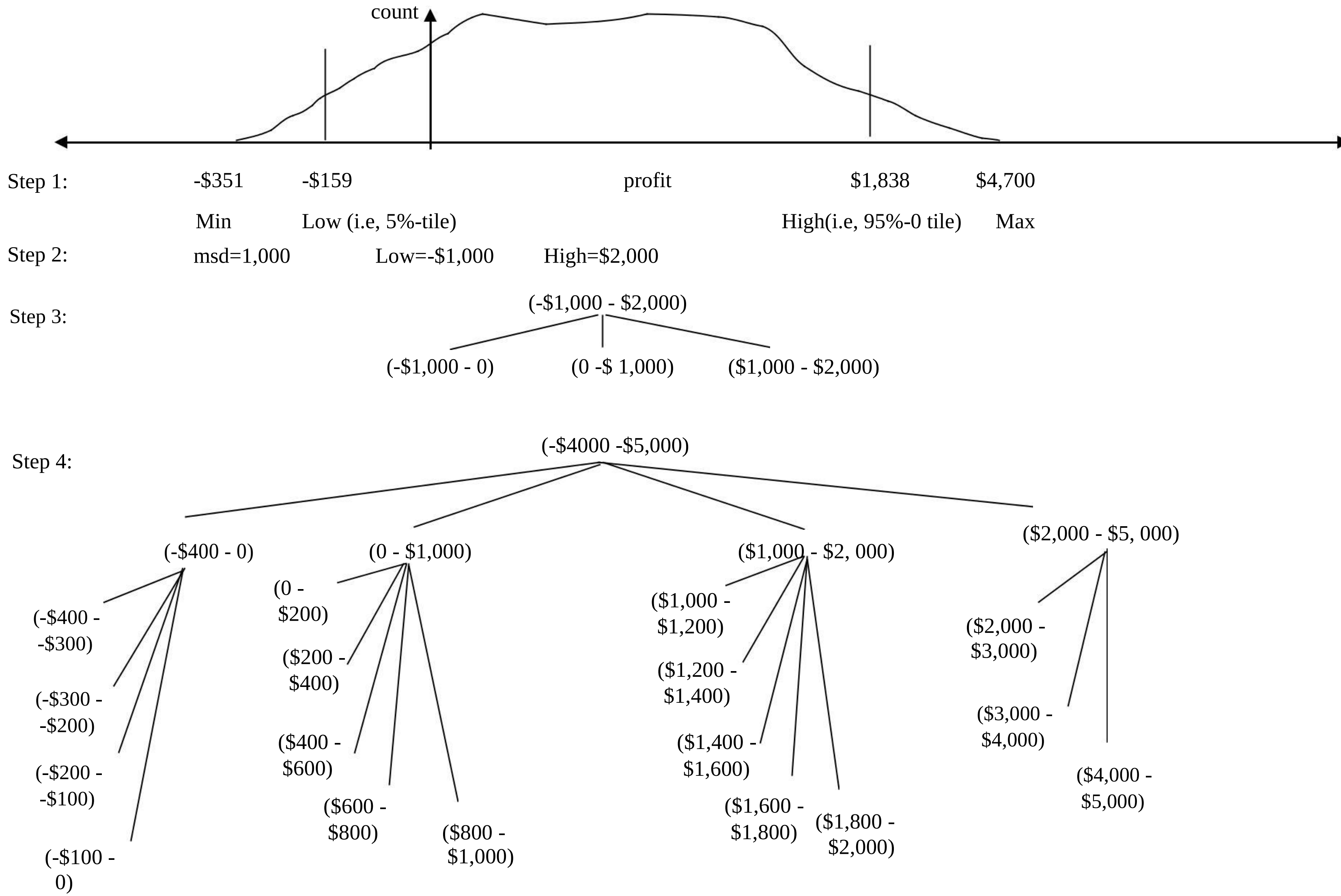
- 3-4-5 rule can be used to segment numeric data into relatively uniform, “natural” intervals.

If an interval covers 3, 6, 7 or 9 distinct values at the most significant digit, partition the range into 3 equal-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



Concept hierarchy generation for categorical data

Specification of a partial ordering of attributes explicitly at the schema level by users or experts

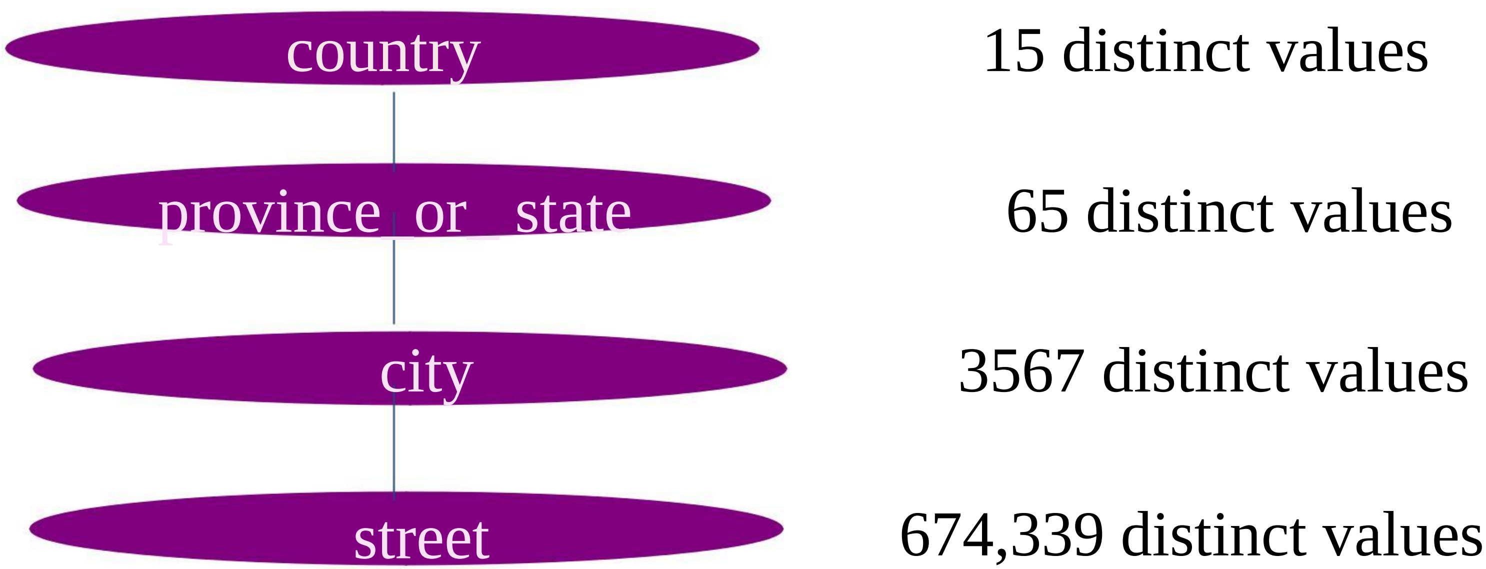
Specification of a portion of a hierarchy by explicit data grouping

Specification of a set of attributes, but not of their partial ordering

Specification of only a partial set of attributes

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.



UNIT -2

What is Data Warehouse?

What is Data Warehouse?

Defined in many different ways

- A decision support database that is maintained separately from the organization’s operational database
- 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 warehousing:

- The process of constructing and using data warehouses

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
- But the key of operational data may or may not contain "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. Operational DBMS

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 + subject
- View: current, local vs. evolutionary, integrated
- Access patterns: update vs. read-only but complex queries

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

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

Why Separate Data Warehouse?

High performance for both systems

– DBMS— tuned for OLTP: access methods, indexing, concurrency control, recovery

– Warehouse—tuned for OLAP: complex OLAP queries, multidimensional view, consolidation.

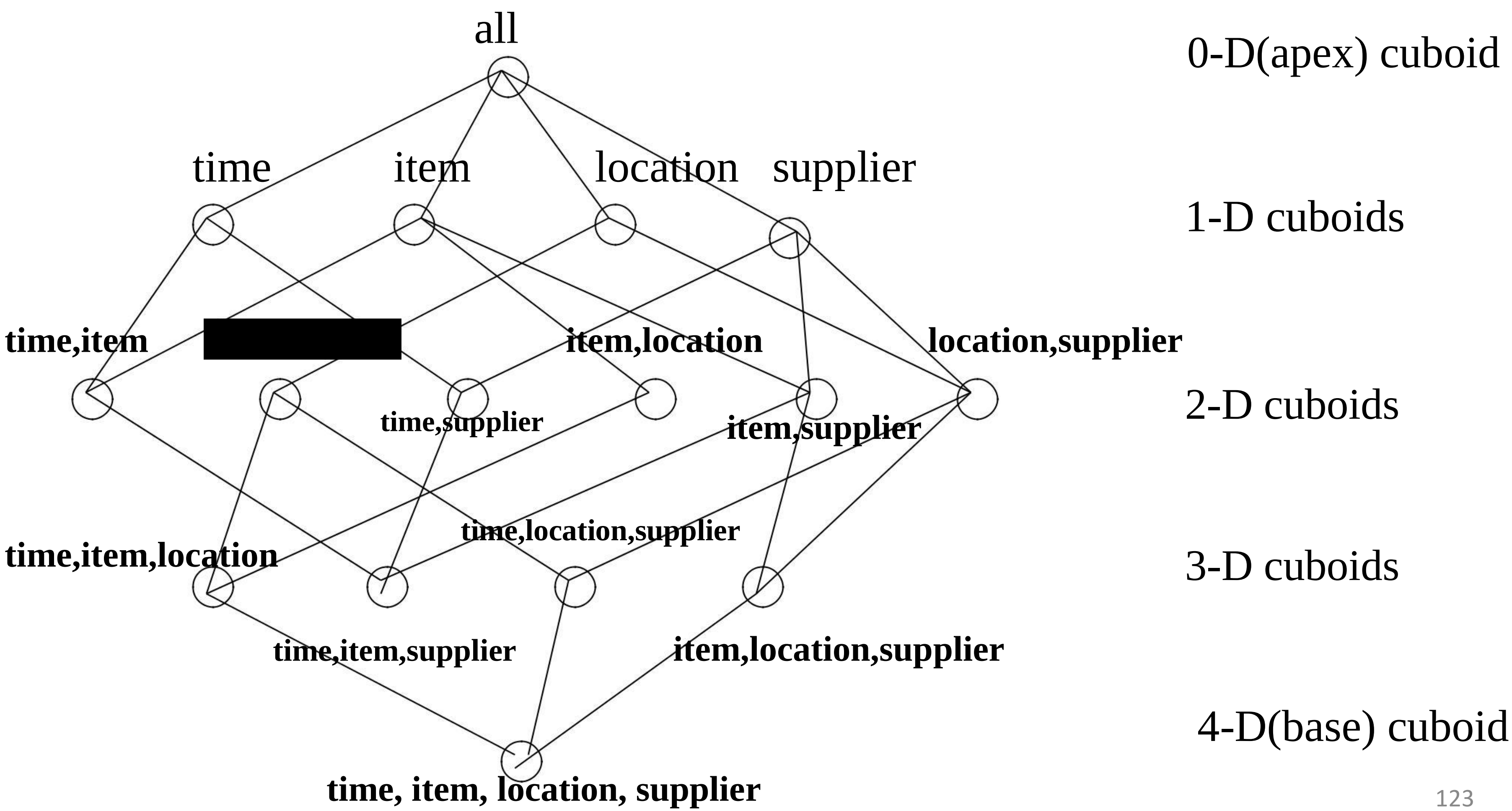
Why Separate Data Warehouse?

Different functions and different data:

- missing data: Decision support requires historical data which operational DBs do not typically maintain
- data consolidation: DS requires consolidation (aggregation, summarization) of data from heterogeneous sources
- data quality: different sources typically use inconsistent data representations, codes and formats which have to be reconciled

A multi-dimensional data model

Cube: A Lattice of Cuboids

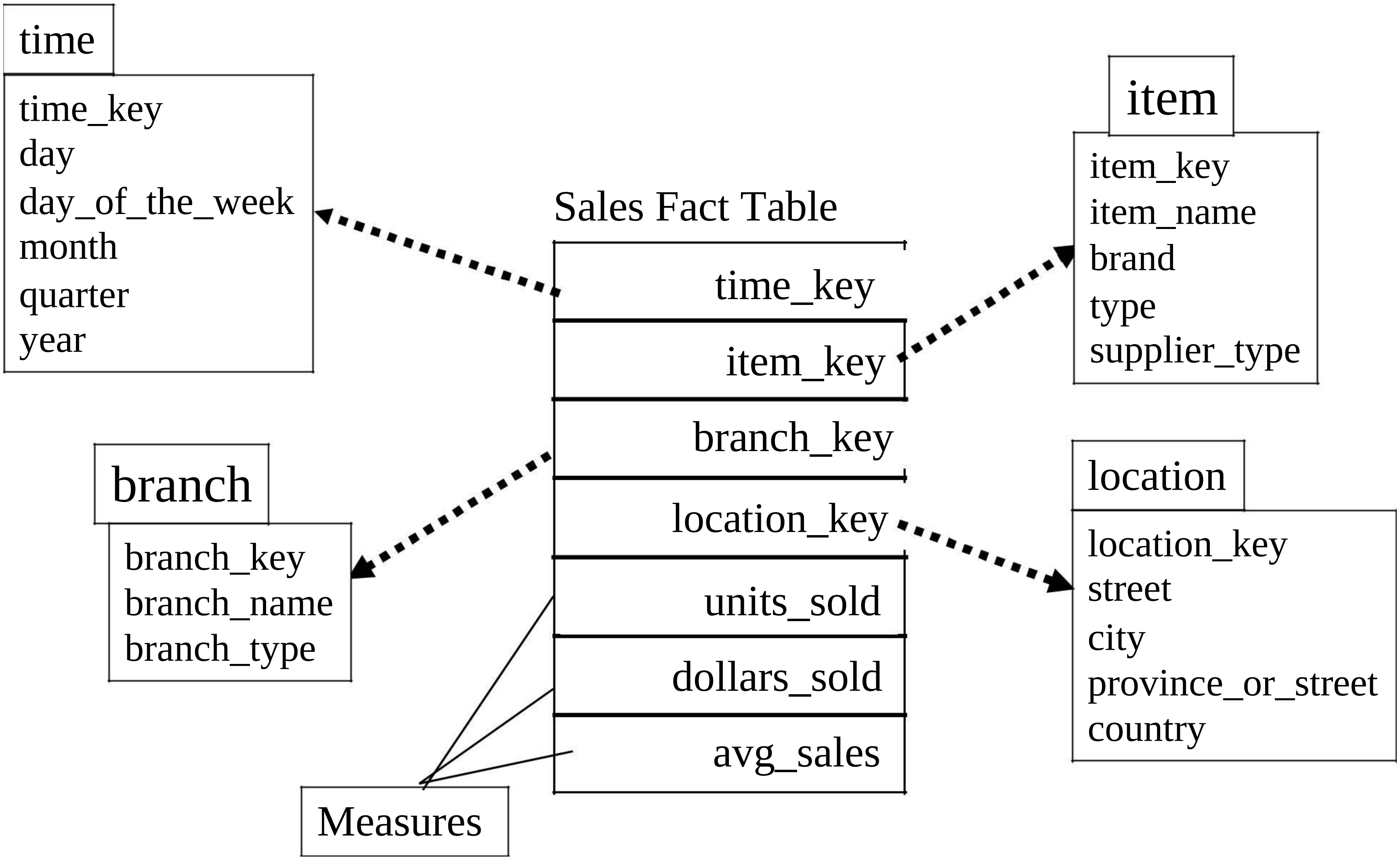


Conceptual Modeling of Data Warehouses

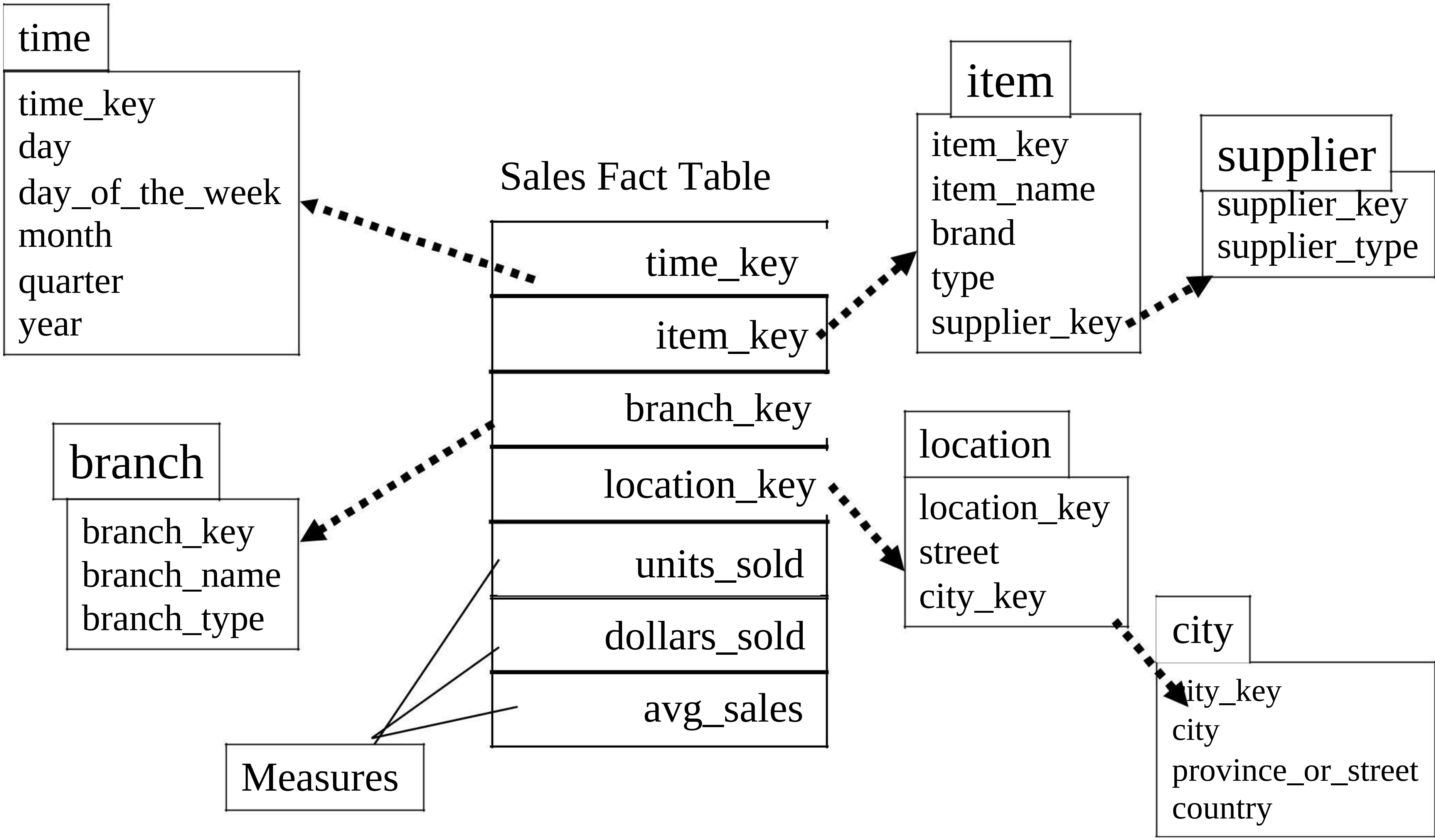
Modeling data warehouses: dimensions & measures

- 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, viewed as a collection of stars, therefore called galaxy schema or fact constellation

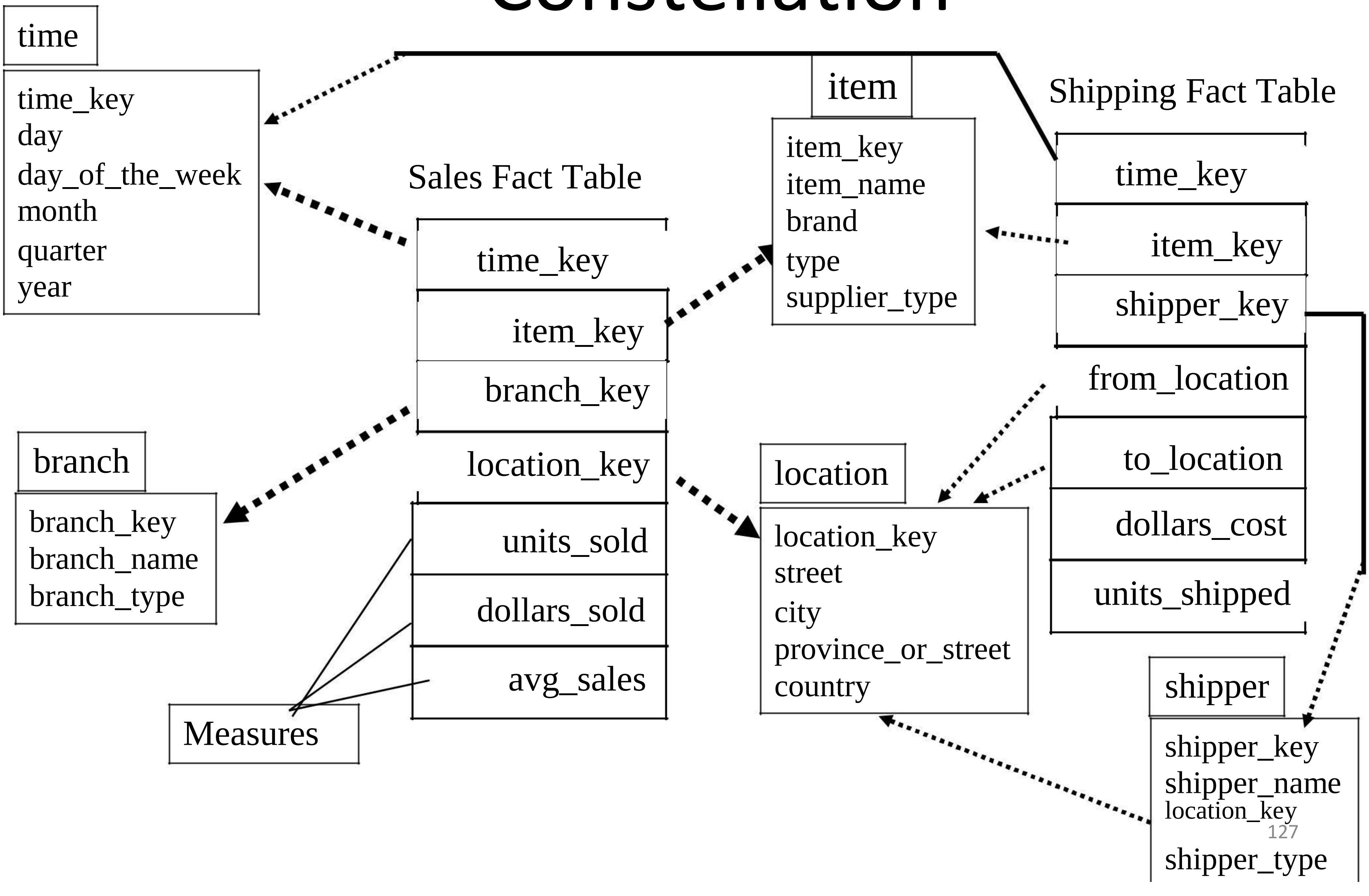
Example of Star Schema



Example of Snowflake Schema



Example of Fact Constellation



A Data Mining Query Language, DMQL: Language Primitives

- Cube Definition (Fact Table)

  define cube <cube_name> [<dimension_list>]: <measure_list>

- Dimension Definition (Dimension Table)

  define dimension <dimension_name> as (<attribute_or_subdimension_list>)

  Special Case (Shared Dimension Tables)

  – First time as “cube definition”

  – define dimension <dimension_name> as <dimension_name_first_time> in cube <cube_name_first_time>

Defining a Star Schema in DMQL

define cube sales_star [time, item, branch, location]:

dollars_sold = sum(sales_in_dollars), avg_sales = avg(sales_in_dollars), units_sold = count(*)

define dimension time as (time_key, day, day_of_week, month, quarter, year)

define dimension item as (item_key, item_name, brand, type, supplier_type)

define dimension branch as (branch_key, branch_name, branch_type)

define dimension location as (location_key, street, city, province_or_state, country)

Defining a Snowflake Schema in DMQL

define cube sales_snowflake [time, item, branch, location]:

dollars_sold = sum(sales_in_dollars), avg_sales = avg(sales_in_dollars), units_sold = count(*)

define dimension time as (time_key, day, day_of_week, month, quarter, year)

define dimension item as (item_key, item_name, brand, type, supplier(supplier_key, supplier_type))

Defining a Snowflake Schema in DMQL

define dimension branch as (branch_key, branch_name, branch_type)

define dimension location as (location_key, street, city(city_key, province_or_state, country))

Defining a Fact Constellation in DMQL

define cube sales [time, item, branch, location]:

dollars_sold = sum(sales_in_dollars), avg_sales = avg(sales_in_dollars), units_sold = count(*)

define dimension time as (time_key, day, day_of_week, month, quarter, year)

define dimension item as (item_key, item_name, brand, type, supplier_type)

define dimension branch as (branch_key, branch_name, branch_type)

define dimension location as (location_key, street, city, province_or_state, country)

Defining a Fact Constellation in DMQL

define cube shipping [time, item, shipper, from_location, to_location]:

dollar_cost = sum(cost_in_dollars), unit_shipped = count(*)

define dimension time as time in cube sales

define dimension item as item in cube sales

define dimension shipper as (shipper_key, shipper_name, location as location in cube sales, shipper_type)

define dimension from_location as location in cube sales

define dimension to_location as location in cube sales

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.

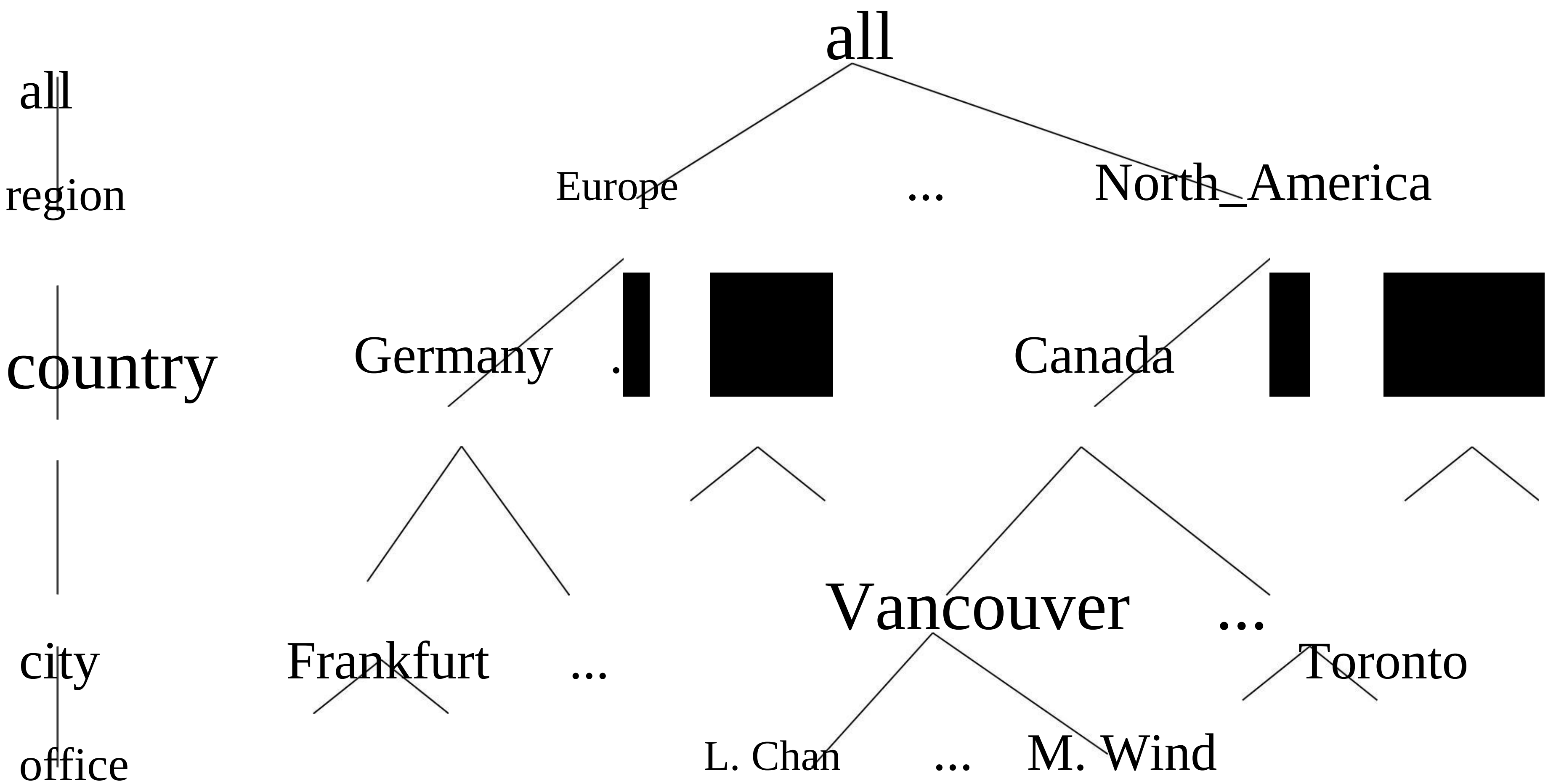
E.g., avg(), min_N(), standard_deviation().

Measures: Three Categories

holistic: if there is no constant bound on the storage size needed to describe a sub aggregate.

E.g., median(), mode(), rank().

A Concept Hierarchy: Dimension (location)

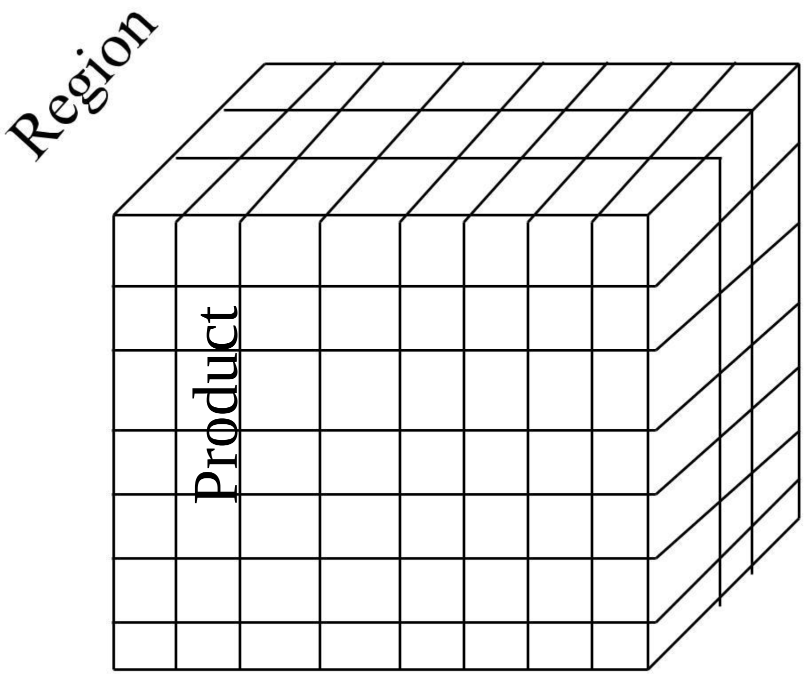


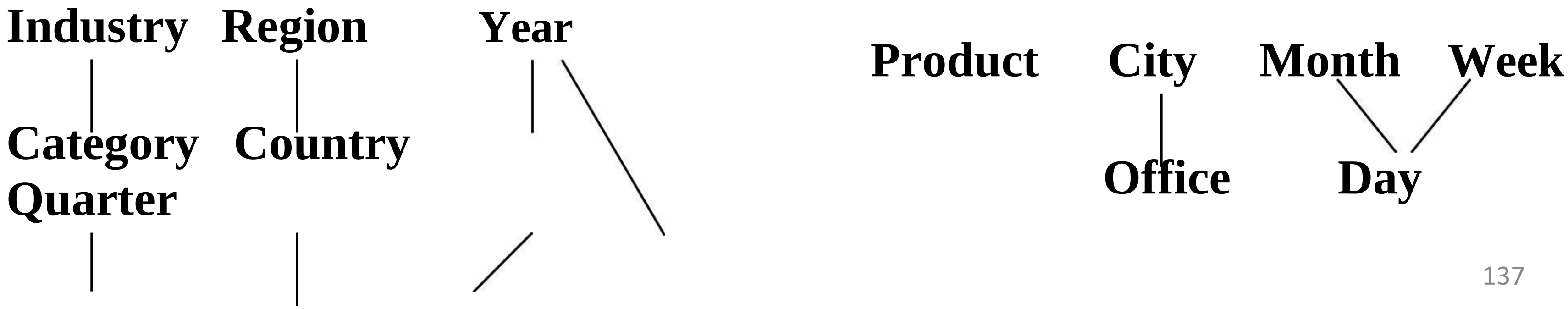
Multidimensional Data

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

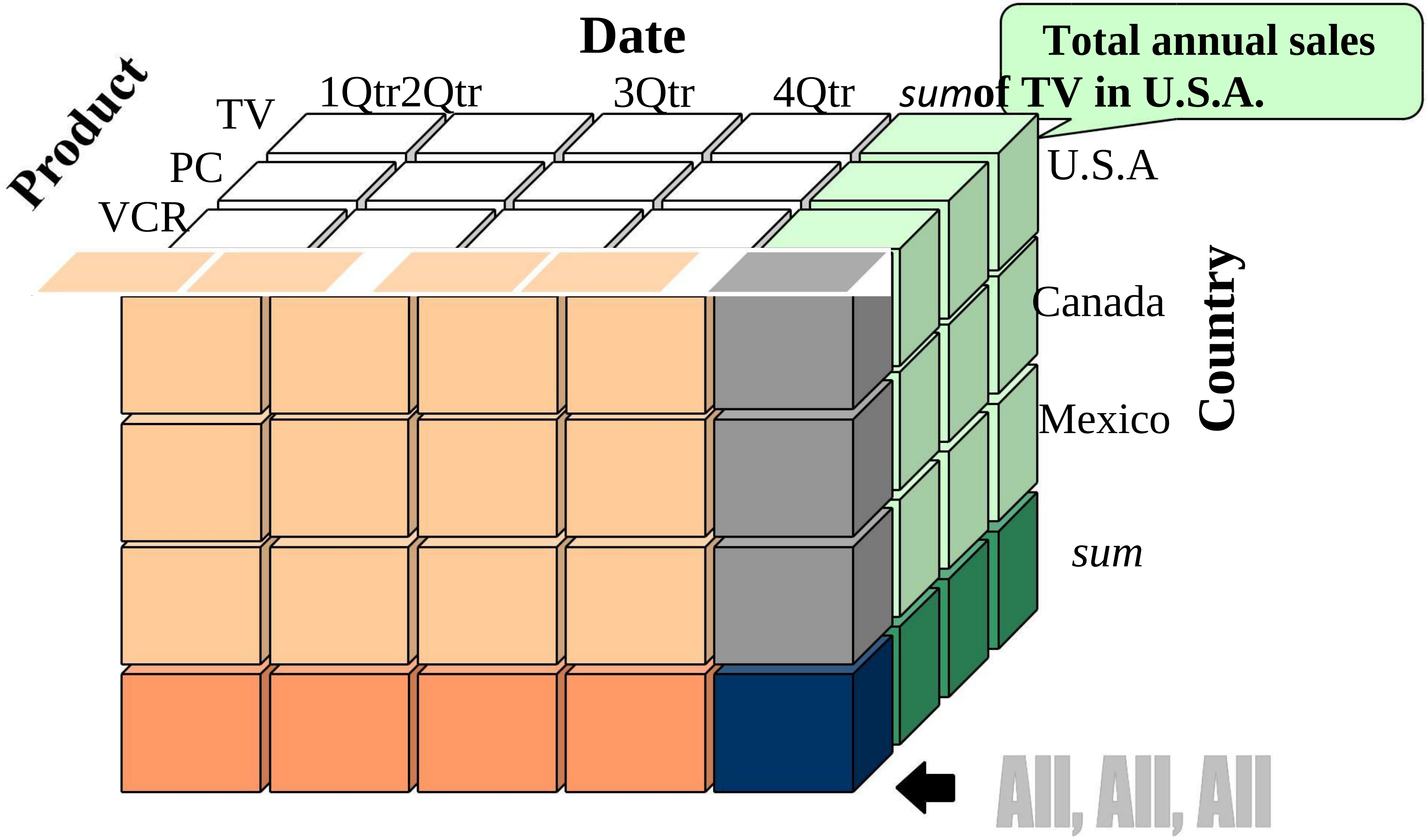
Dimensions: Product, Location, Time
Hierarchical summarization paths

Month

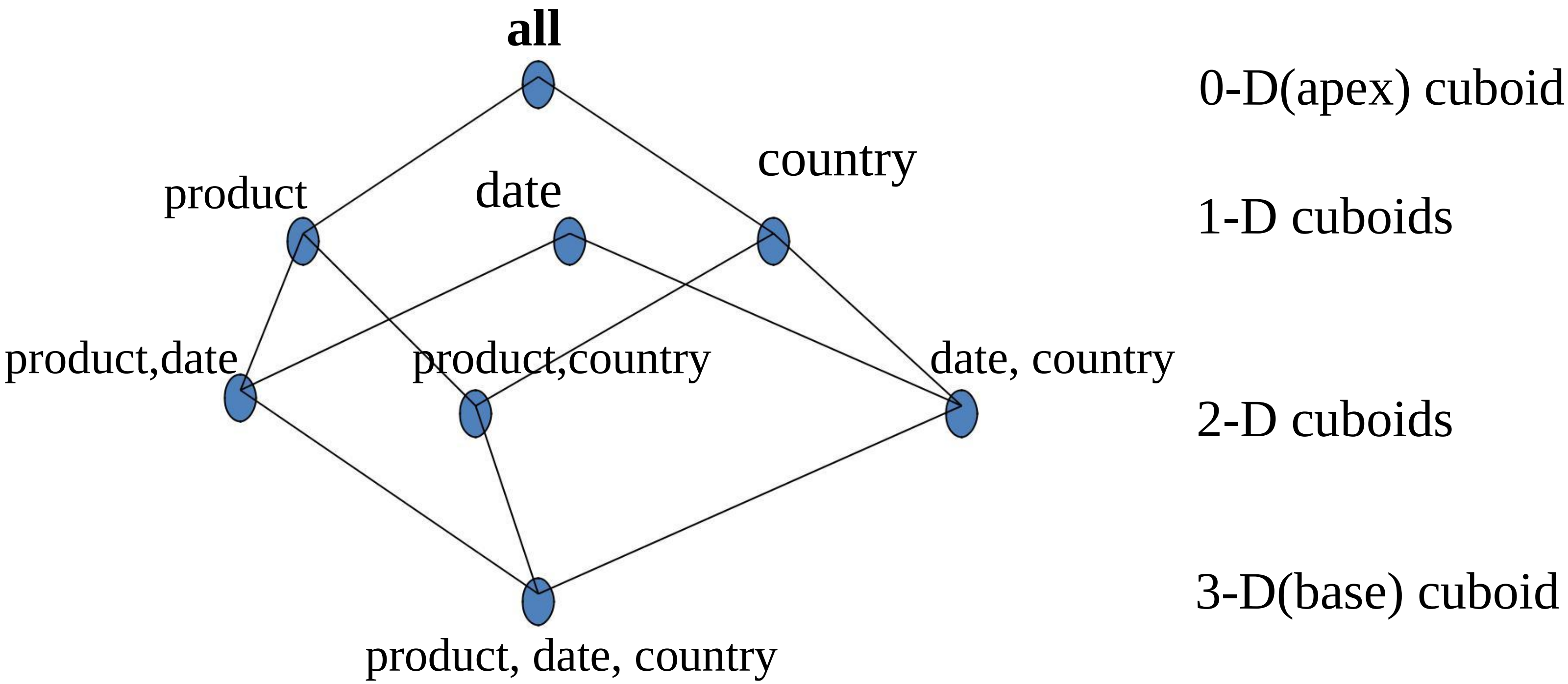



Industry
Region
Year
Category
Country
Quarter
Product
City
Month
Week
Office
Day

A Sample Data Cube



Cuboids Corresponding to the Cube



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:

– *project and select*

OLAP Operations

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)*

Data warehouse architecture

Steps for the Design and Construction of Data Warehouse

The design of a data warehouse: a business analysis framework

The process of data warehouse design

A three-tier data ware house architecture

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

Design of a Data Warehouse: A Business Analysis Framework

– Data warehouse view

consists of fact tables and dimension tables

– Data source view

exposes the information being captured, stored, and managed by operational systems

– Business query view

sees the perspectives

Data Warehouse Design Process

Top-down, bottom-up approaches or a combination of both

- <u>Top-down</u>: Starts with overall design and planning (mature)
- <u>Bottom-up</u>: Starts with experiments and prototypes (rapid)

From software engineering point of view

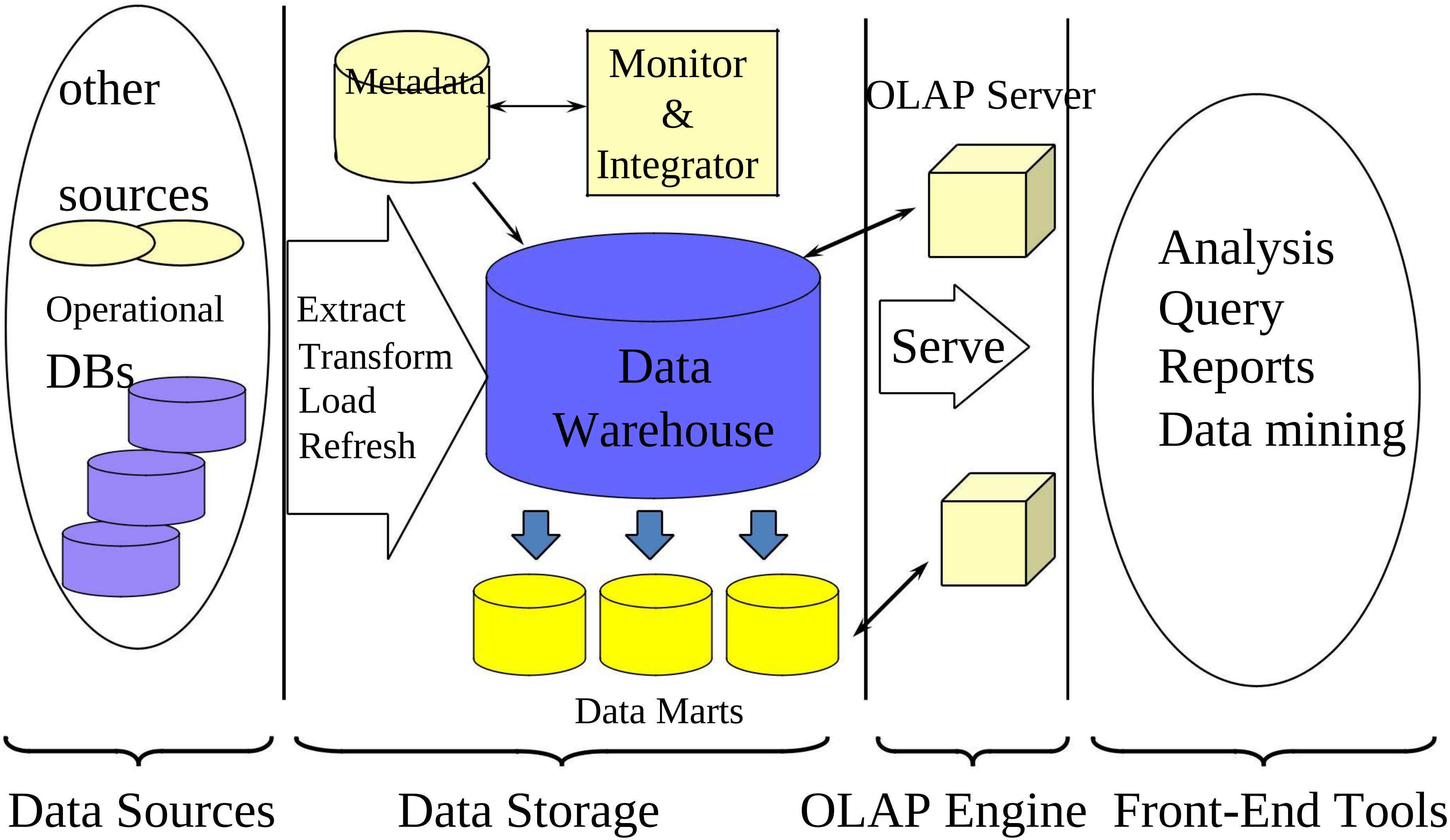
- <u>Waterfal</u>l: structured and systematic analysis at each step before proceeding to the next
- <u>Spiral</u>: rapid generation of increasingly functional systems, short turn around time, quick turn around

Data Warehouse Design Process

Typical data warehouse design process

- Choose a business process to model, e.g., orders, invoices, etc.
- Choose the <u>*grain*</u> (*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



Metadata Repository

Meta data is the data defining warehouse objects. It has the following kinds

- Description of the structure of the warehouse

  schema, view, dimensions, hierarchies, derived data defn, data mart locations and contents

- Operational meta-data

  data lineage (history of migrated data and transformation path), currency of data (active, archived, or purged), monitoring information (warehouse usage statistics, error reports, audit trails)

- The algorithms used for summarization
- The mapping from operational environment to the data warehouse
- Data related to system performance

  warehouse schema, view and derived data definitions

- Business data

  business terms and definitions, ownership of data, charging policies

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

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

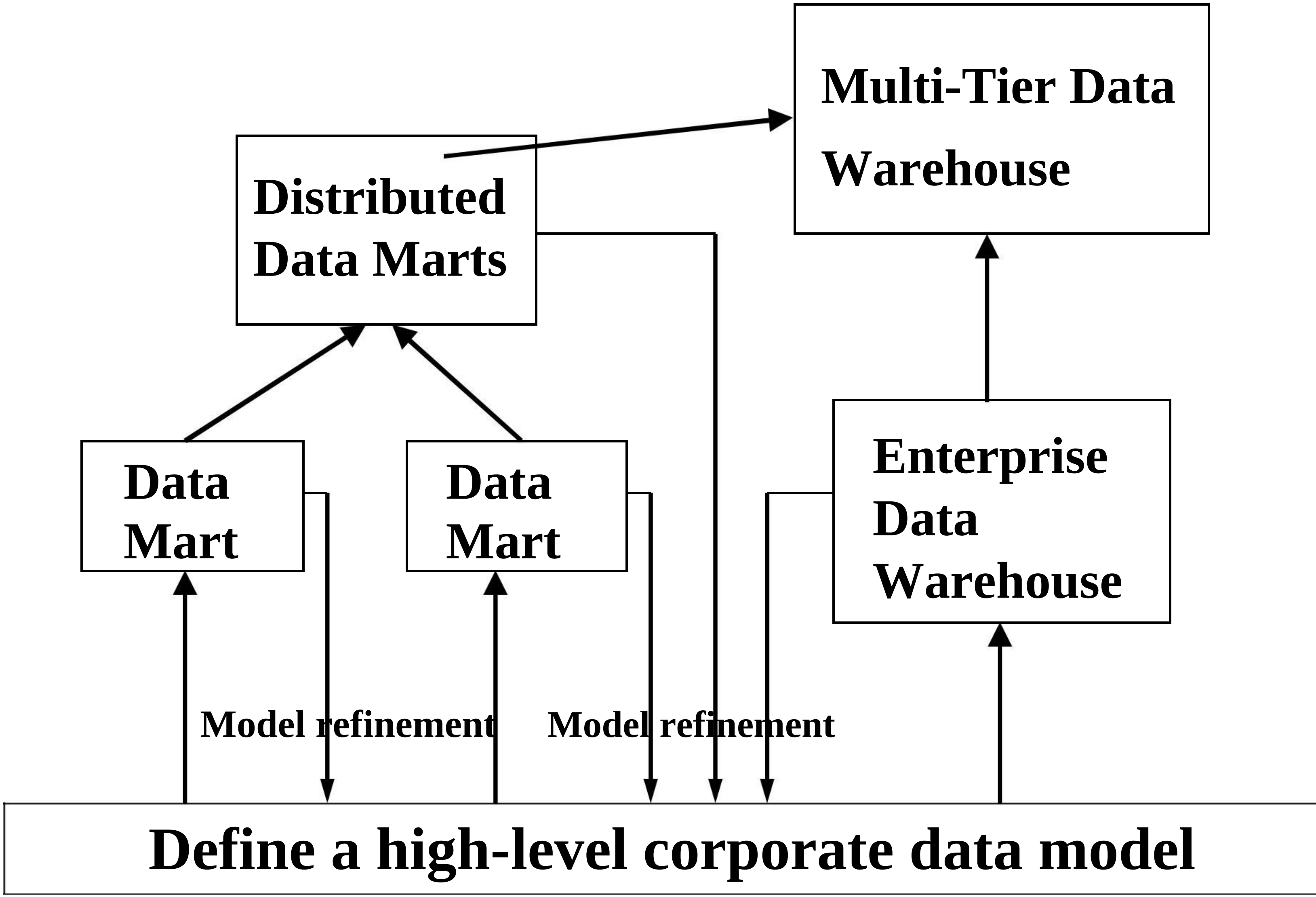
Independent vs. dependent (directly from warehouse) data mart

Virtual warehouse

– A set of views over operational databases

– Only some of the possible summary views may be materialized

Data Warehouse Development: A Recommended Approach



Types of OLAP Servers

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

Types of OLAP Servers

Hybrid OLAP (HOLAP)

– User flexibility, e.g., low level: relational, high-level: array

Specialized SQL servers

– specialized support for SQL queries over star/snowflake schemas

Data warehouse implementation

Efficient Data Cube Computation

Data cube can be viewed as a lattice of cuboids

- The bottom-most cuboid is the base cuboid
- The top-most cuboid (apex) contains only one cell
- How many cuboids in an n-dimensional cube with L levels?

$$T = \prod_{i=1}^{n} (L_i + 1)$$

Materialization of data cube

- Materialize every (cuboid) (full materialization), none (no materialization), or some (partial materialization)
- Selection of which cuboids to materialize
  - Based on size, sharing, access frequency, etc.

Cube Operation

- Cube definition and computation in DMQL

  define cube sales[item, city, year]: sum(sales_in_dollars)

  compute cube sales

  Transform it into a SQL-like language (with a new operator cube by, introduced by Gray et al.'96)

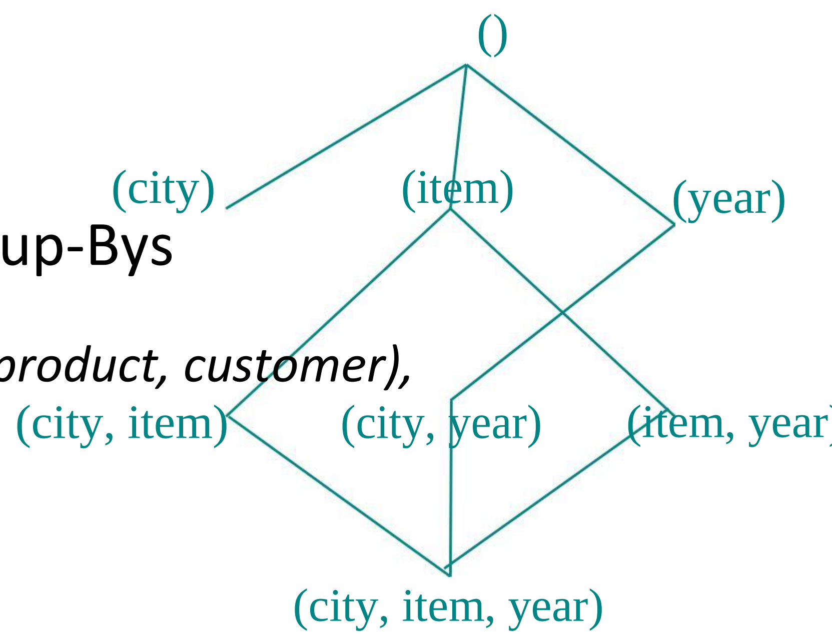
  SELECT item, city, year, SUM (amount)

  FROM SALES

  CUBE BY item, city, year

- Need compute the following Group-Bys

  (date, product, customer),
  (date,product),(date, customer), (product, customer),
  (date), (product), (customer)
  ()



Cube Computation: ROLAP-Based Method

Efficient cube computation methods

- ROLAP-based cubing algorithms (Agarwal et al'96)
- Array-based cubing algorithm (Zhao et al'97)
- Bottom-up computation method (Bayer & Ramarkrishnan'99)

ROLAP-based cubing algorithms

- Sorting, hashing, and grouping operations are applied to the dimension attributes in order to reorder and cluster related tuples
- Grouping is performed on some sub aggregates as a "partial grouping step"
- Aggregates may be computed from previously computed aggregates, rather than from the base fact table

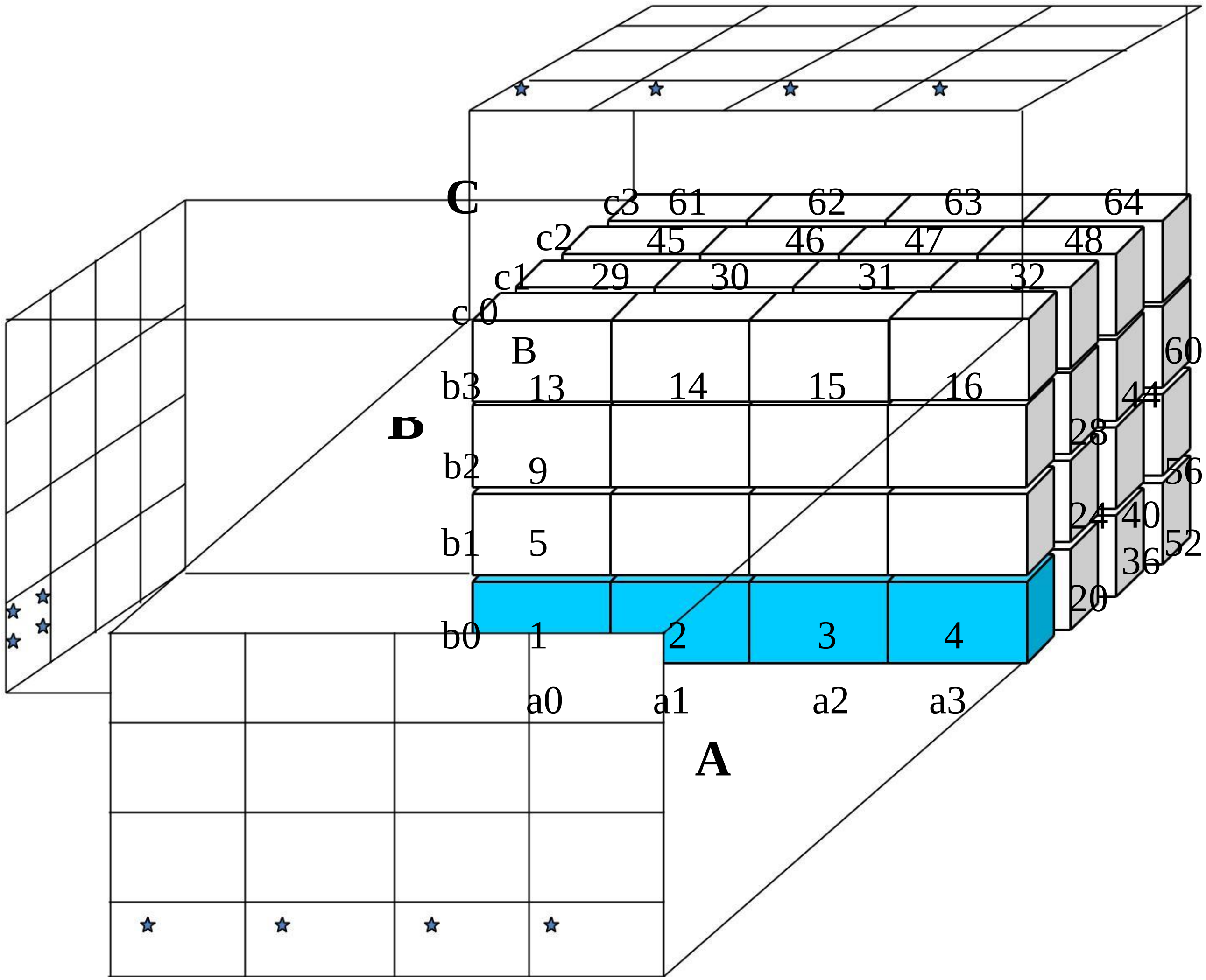
Multi-way Array Aggregation for Cube Computation

Partition arrays into chunks (a small sub cube which fits in memory).

Compressed sparse array addressing: (chunk_id, offset)

Compute aggregates in “multi way” by visiting cube cells in the order which minimizes the # of times to visit each cell, and reduces memory access and storage cost.

Multi-way Array Aggregation for Cube Computation



Multi-Way Array Aggregation for Cube Computation

Method: the planes should be sorted and computed according to their size in ascending order.

- Idea: keep the smallest plane in the main memory, fetch and compute only one chunk at a time for the largest plane

Limitation of the method: computing well only for a small number of dimensions

- If there are a large number of dimensions, "bottom-up computation" and iceberg cube computation methods can be explored

Indexing OLAP Data: Bitmap Index

- Index on a particular column
- Each value in the column has a bit vector: bit-op is fast
- The length of the bit vector: # of records in the base table
- The *i*-th bit is set if the *i*-th row of the base table has the value for the indexed column
- not suitable for high cardinality domains

Base table

Cust	Region	Type
C1	Asia	Retail
C2	Europe	Dealer
C3	Asia	Dealer
C4	America	Retail
C5	Europe	Dealer

Index on Region

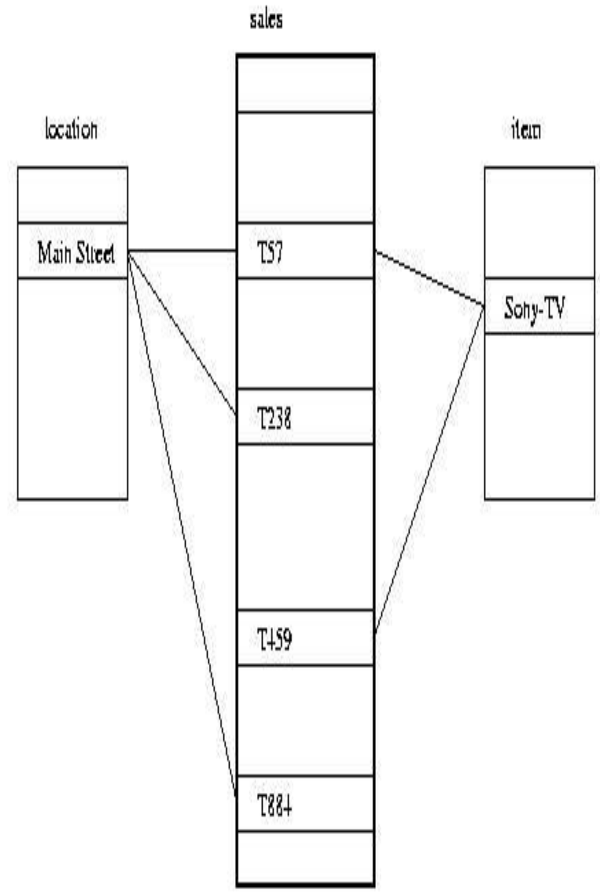
RecID	Asia	Europe	America
1	1	0	0
2	0	1	0
3	1	0	0
4	0	0	1
5	0	1	0

Index on Type

RecID	Retail	Dealer
1	1	0
2	0	1
3	0	1
4	1	0
5	0	1

Indexing OLAP Data: Join Indices

- Join index: JI(R-id, S-id) where R (R-id, …) S (S-id, …)
- Traditional indices map the values to a list of record ids
  - It materializes relational join in JI file and speeds up relational join — a rather costly operation
- In data warehouses, join index relates the values of the dimensions of a start schema to rows in the fact table.
  - E.g. fact table: *Sales* and two dimensions *city* and *product*
    - A join index on *city* maintains for each distinct city a list of R-IDs of the tuples recording the Sales in the city
  - Join indices can span multiple dimensions



Efficient Processing OLAP Queries

Determine which operations should be performed on the available cuboids:

- transform drill, roll, etc. into corresponding SQL and/or OLAP operations, e.g, dice = selection + projection

Determine to which materialized cuboid(s) the relevant operations should be applied.

Exploring indexing structures and compressed vs. dense array structures in MOLAP

From data warehousing to data mining

Data Warehouse Usage

Three kinds of data warehouse applications

- Information processing

  supports querying, basic statistical analysis, and reporting using crosstabs, tables, charts and graphs

- Analytical processing

  multidimensional analysis of data warehouse data

  supports basic OLAP operations, slice-dice, drilling, pivoting

- Data mining

  knowledge discovery from hidden patterns

  supports associations, constructing analytical models, performing classification and prediction, and presenting the mining results using visualization tools.

Differences among the three tasks

From On-Line Analytical Processing to On Line Analytical Mining (OLAM)

Why online analytical mining?

- High quality of data in data warehouses

  DW contains integrated, consistent, cleaned data

- Available information processing structure surrounding data warehouses

  ODBC, OLEDB, Web accessing, service facilities, reporting and OLAP tools

- OLAP-based exploratory data analysis

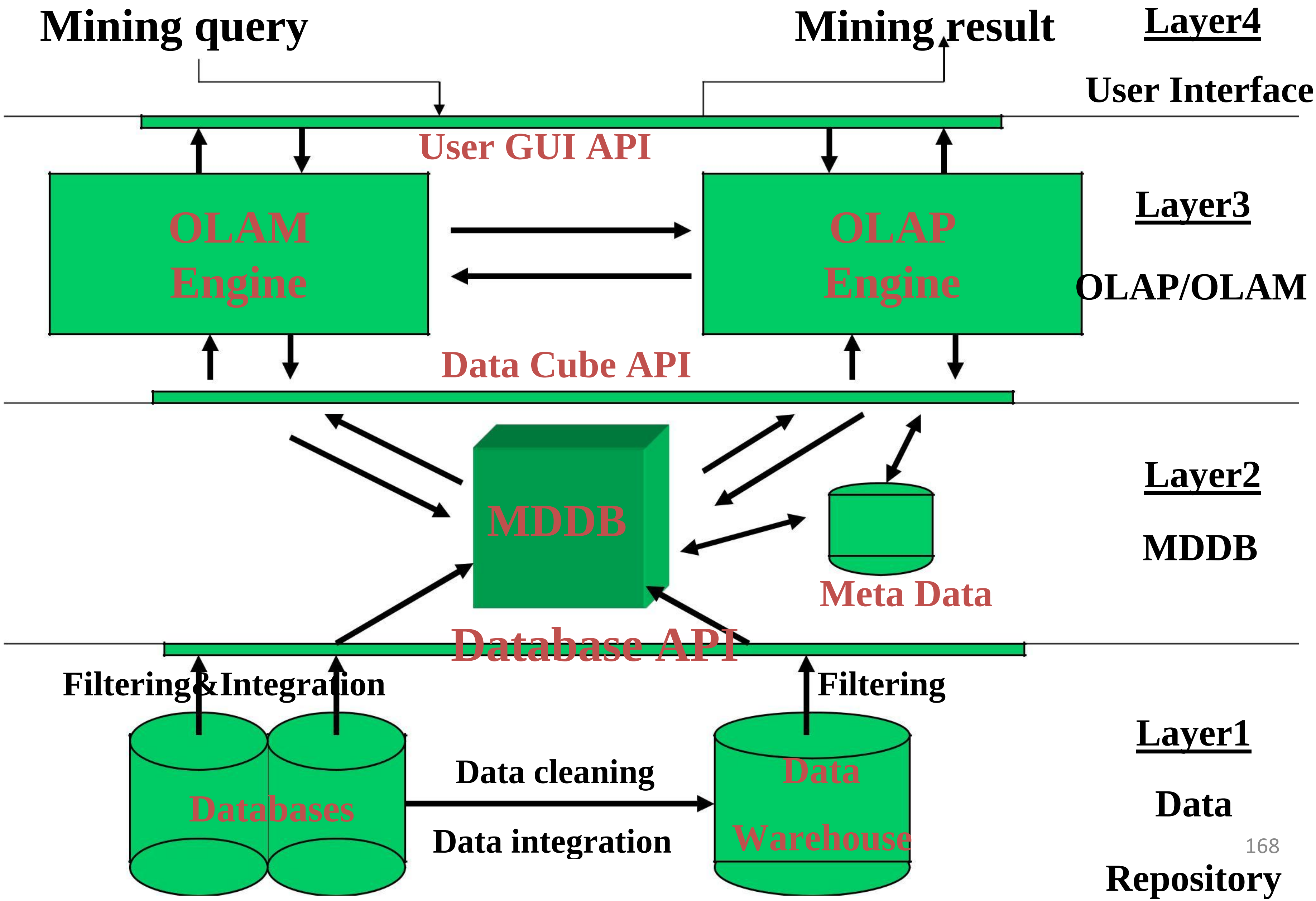
  mining with drilling, dicing, pivoting, etc.

- On-line selection of data mining functions

  integration and swapping of multiple mining functions, algorithms, and tasks.

Architecture of OLAM

An OLAM Architecture



UNIT - 3

Mining Frequent Patterns , Associations and Correlations

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.

Applications

- Basket data analysis, cross-marketing, catalog design, loss-leader analysis, clustering, classification, etc.

Association Mining

- Rule form

  prediction (Boolean variables) $=>$ prediction (Boolean variables) [support, confidence]

  – Computer => antivirus_software [support =2%, confidence = 60%]
  – buys (x, “computer”) $\rightarrow$ buys (x, “antivirus_software”) [0.5%, 60%]

Association Rule: Basic Concepts

Given a database of transactions 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

Find frequent patterns

Example for frequent itemset mining is market basket analysis.

Association rule performance measures

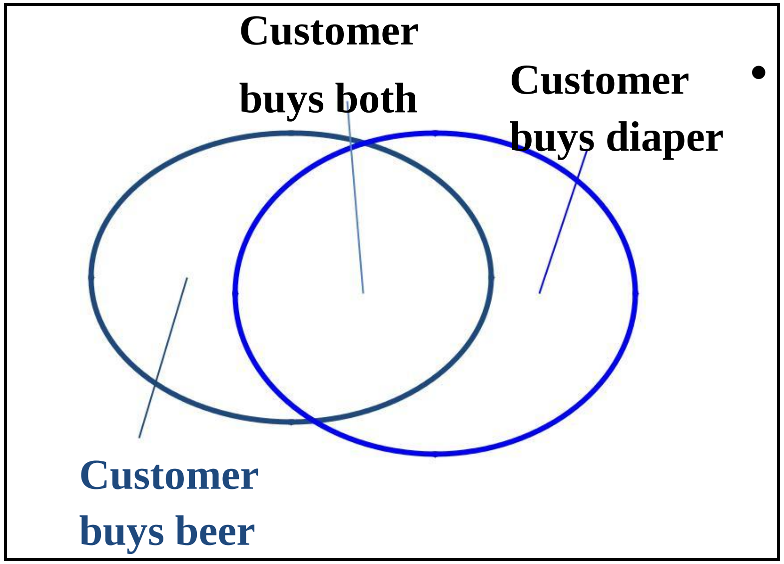
Confidence

Support

Minimum support threshold

Minimum confidence threshold

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%, we have

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

Martket Basket Analysis

Shopping baskets

Each item has a Boolean variable representing the presence or absence of that item.

Each basket can be represented by a Boolean vector of values assigned to these variables.

Identify patterns from Boolean vector

Patterns can be represented by association rules.

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%]
- 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

Mining single-dimensional Boolean association rules from transactional databases

Apriori Algorithm

Single dimensional, single-level, Boolean frequent item sets

Finding frequent item sets using candidate generation

Generating association rules from frequent item sets

Mining Association Rules—An 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%

The Apriori principle:

Any subset of a frequent itemset must be frequent

Mining Frequent Itemsets: the Key Step

Find the *frequent itemsets*: the sets of items that have minimum support

- A subset of a frequent itemset must also be a frequent itemset

  i.e., if {*AB*} is a frequent itemset, both {*A*} and {*B*} should be a frequent itemset

- Iteratively find frequent itemsets with cardinality from 1 to *k (k*-itemset*)*

Use the frequent itemsets to generate association rules.

The Apriori Algorithm

Join Step

– C_k is generated by joining L_{k-1} with itself

Prune Step

– Any (k-1)-itemset that is not frequent cannot be a subset of a frequent k-itemset

The Apriori Algorithm

- <u>Pseudo-code</u>:

C_k: Candidate itemset of size k
L_k : frequent itemset of size k

L_1 = {frequent items};
for (k = 1; L_k !=$\emptyset$; k++) **do begin**
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 $\cup_k L_k$;

The Apriori Algorithm — Example

Database D

TID	Items
100	1 3 4
200	2 3 5
300	1 2 3 5
400	2 5

Scan D →

C_1

itemset	sup.
{1}	2
{2}	3
{3}	3
{4}	1
{5}	3

→

L_1

itemset	sup.
{1}	2
{2}	3
{3}	3
{5}	3

C_2

itemset
{1 2}
{1 3}
{1 5}
{2 3}
{2 5}
{3 5}

Scan D ←

C_2

itemset	sup
{1 2}	1
{1 3}	2
{1 5}	1
{2 3}	2
{2 5}	3
{3 5}	2

←

L_2

itemset	sup
{1 3}	2
{2 3}	2
{2 5}	3
{3 5}	2

C_3

itemset
{2 3 5}

Scan D →

L_3

itemset	sup
{2 3 5}	2

How to Generate Candidates?

Suppose the items in L_{k-1} are listed in an order

Step 1: self-joining L_{k-1}

insert into C_k

select p.item$_1$, p.item$_2$, …, p.item$_{k-1}$,

q.item$_{k-1}$ from L_{k-1} p, L_{k-1} q

where p.item$_1$=q.item$_1$, …, p.item$_{k-2}$=q.item$_{k-2}$, p.item$_{k-1}$ < q.item$_{k-1}$

- Step 2: pruning

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***

How to Count Supports of Candidates?

Why 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 node of hash-tree contains a list of itemsets and counts

– Interior node contains a hash table

– Subset function: finds all the candidates contained in a transaction

Example of Generating Candidates

L_3={*abc, abd, acd, ace, bcd*}

Self-joining: $L_3 * L_3$

- *abcd* from *abc* and *abd*
- *acde* from *acd* and *ace*

Pruning:

- *acde* is removed because *ade* is not in L_3

C_4={*abcd*}

Methods to Improve Apriori's Efficiency

Hash-based itemset counting

– A *k*-itemset whose corresponding hashing bucket count is below the threshold cannot be frequent

Transaction reduction

– A transaction that does not contain any frequent k-itemset is useless in subsequent scans

Partitioning

– Any itemset that is potentially frequent in DB must be frequent in at least one of the partitions of DB

Methods to Improve Apriori's Efficiency

Sampling

– mining on a subset of given data, lower support threshold + a method to determine the completeness

Dynamic itemset counting

– add new candidate itemsets only when all of their subsets are estimated to be frequent

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

Mining multilevel association rules from transactional databases

Mining various kinds of association rules

Mining Multilevel association rules

– Concepts at different levels

Mining Multidimensional association rules

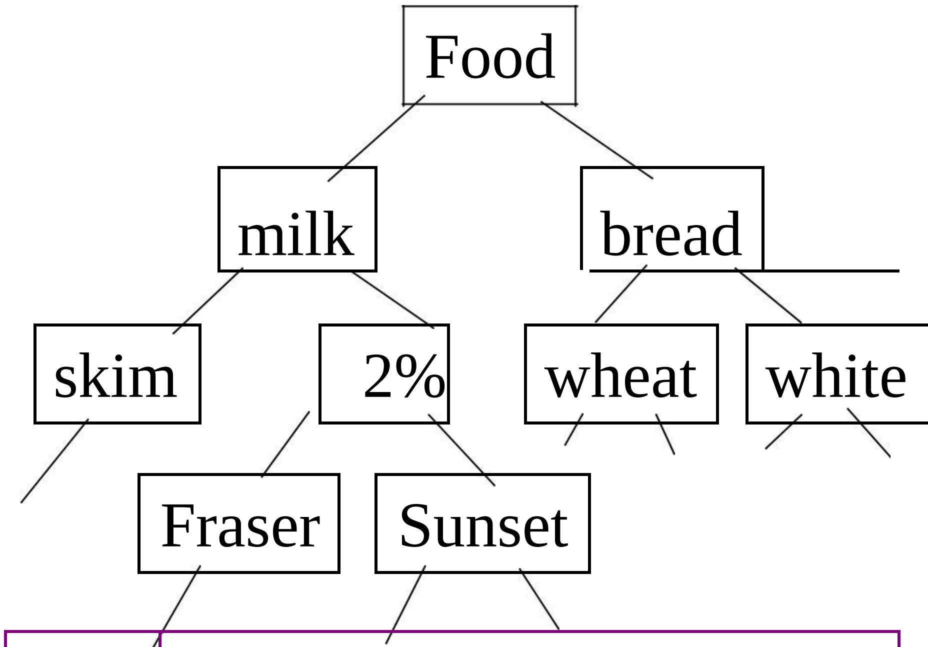
– More than one dimensional

Mining Quantitative association rules

– Numeric attributes

Multiple-Level Association Rules

- Items often form hierarchy.
- Items at the lower level are expected to have lower support.
- Rules regarding itemsets at appropriate levels could be quite useful.
- Transaction database can be encoded based on dimensions and levels
- We can explore shared multi-level mining



TID	Items
T1	{111, 121, 211, 221}
T2	{111, 211, 222, 323}
T3	{112, 122, 221, 411}
T4	{111, 121}
T5	{111, 122, 211, 221, 413}

Multi-level Association

Uniform Support- the same minimum support for all levels

- **+** One minimum support threshold. No need to examine itemsets containing any item whose ancestors do not have minimum support.
- – Lower level items do not occur as frequently. If support threshold

  too high $\Rightarrow$ miss low level associations

  too low $\Rightarrow$ generate too many high level associations

Multi-level Association

Reduced Support- reduced minimum support at lower levels

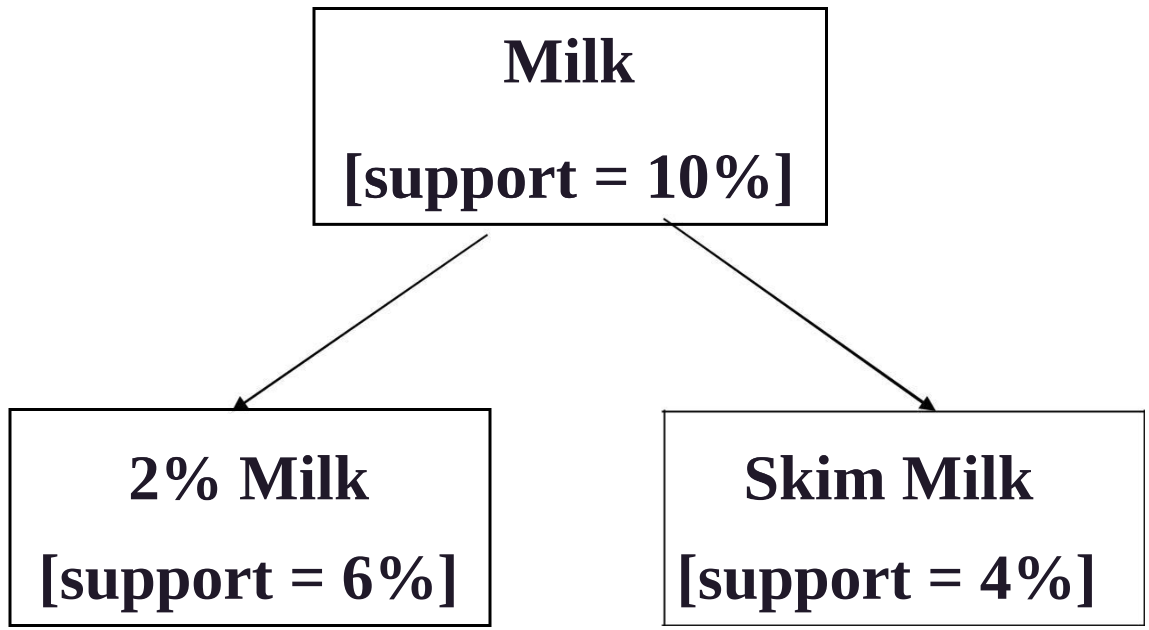
- There are 4 search strategies:
  - Level-by-level independent
  - Level-cross filtering by k-itemset
  - Level-cross filtering by single item
  - Controlled level-cross filtering by single item

Uniform Support

Multi-level mining with uniform support

Level 1
min_sup = 5%

Level 2
min_sup = 5%



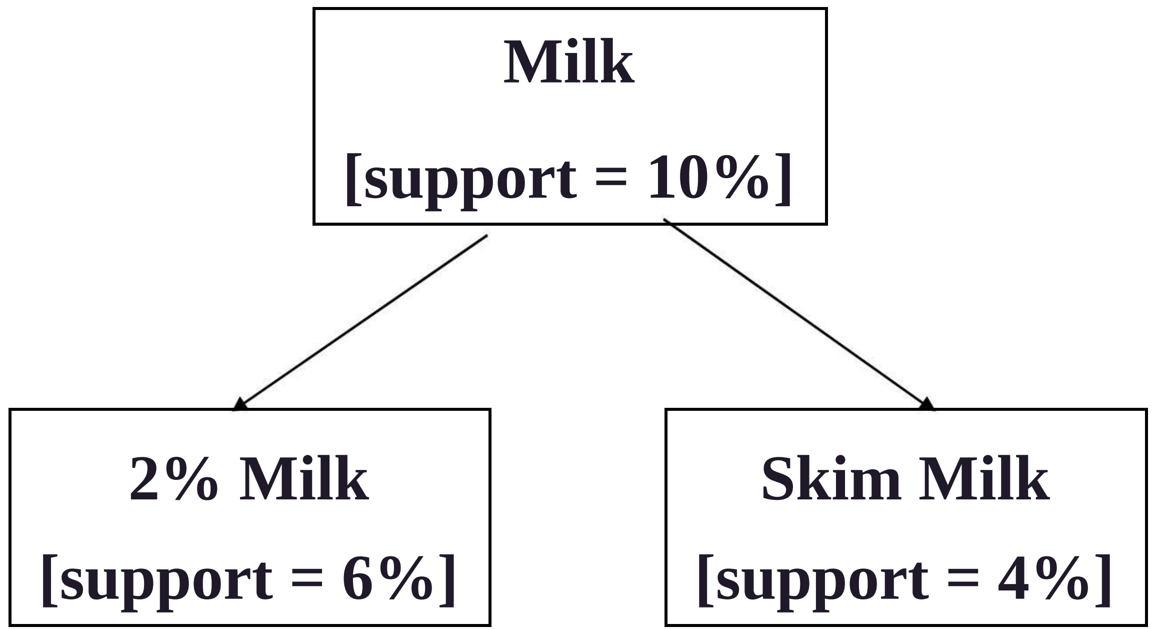
[Back]

Reduced Support

Multi-level mining with reduced support

Level 1
min_sup = 5%

Level 2
min_sup = 3%



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%]
- 2% 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.

Mining multidimensional association rules from transactional databases and data warehouse

Multi-Dimensional Association

- Single-dimensional rules

  buys(X, “milk”) $\Rightarrow$ buys(X, “bread”)

- Multi-dimensional rules
  - Inter-dimension association rules -no repeated predicates

    age(X,”19-25”) $\wedge$ occupation(X,“student”) $\Rightarrow$ buys(X,“coke”)
  - hybrid-dimension association rules -repeated predicates

    age(X,”19-25”) $\wedge$ buys(X, “popcorn”) $\Rightarrow$ buys(X, “coke”)

Multi-Dimensional Association

Categorical Attributes

– finite number of possible values, no ordering among values

Quantitative Attributes

– numeric, implicit ordering among values

Techniques for Mining MD 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.

Quantitative association rules

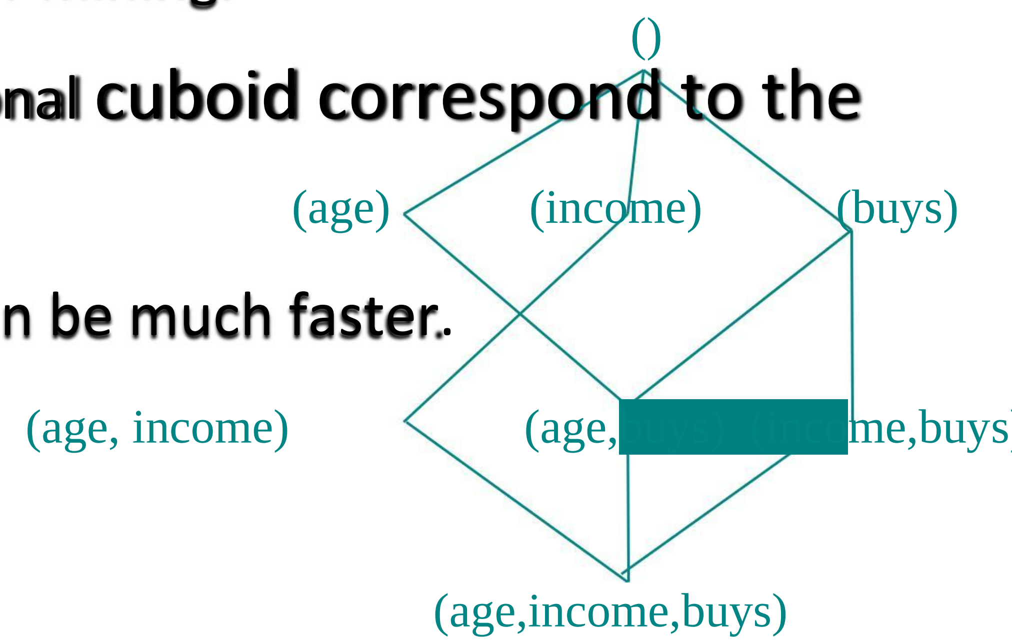
– Quantitative attributes are dynamically discretized into "bins"based on the distribution of the data.

Distance-based association rules

– This is a dynamic discretization process that considers the distance between data points.

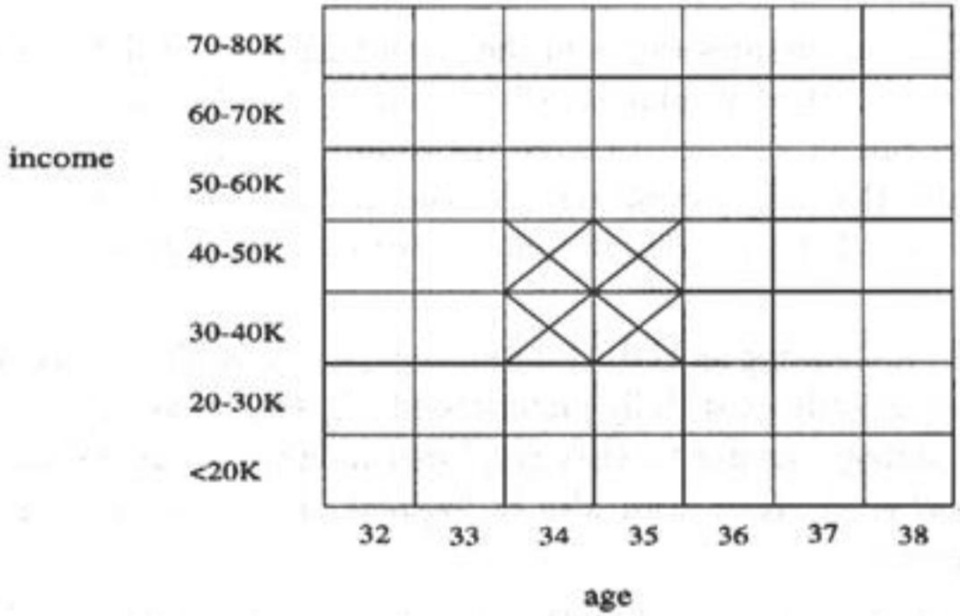
Static Discretization of Quantitative Attributes

- Discretized prior to mining using concept hierarchy.
- Numeric values are replaced by ranges.
- In relational database, finding all frequent k-predicate sets will require k or k+1 table scans.
- Data cube is well suited for mining.
- The cells of an n-dimensional cuboid correspond to the predicate sets.
- Mining from data cubescan be much faster.



Quantitative Association Rules

- **Numeric attributes are *dynamically* discretized**
  - Such that the confidence or compactness of the rules mined is maximized.
- **2-D quantitative association rules:** $A_{quan1} \wedge A_{quan2} \Rightarrow A_{cat}$
- **Cluster "adjacent"**
  association rules to form general rules using a 2-D grid.



- **Example:**

age(X,"30-34") $\wedge$ income(X,"24K - 48K")

$\Rightarrow$ buys(X,"high resolution TV")

From association mining
to correlation analysis

Interestingness Measurements

Objective measures

– Two popular measurements

support

confidence

Subjective measures

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

- Among 5000 students
  - 3000 play basketball
  - 3750 eat cereal
  - 2000 both play basket ball and eat cereal
- *play basketball* ⇒ *eat cereal* [40%, 66.7%] is misleading because the overall percentage of students eating cereal is 75% which is higher than 66.7%.
- *play basketball* ⇒ *not eat cereal* [20%, 33.3%] is far more accurate, although with lower support and confidence

	basketball	not basketball	sum(row)
cereal	2000	1750	3750
not cereal	1000	250	1250
sum(col.)	3000	2000	5000

Criticism to Support and Confidence

Example

- X and Y: positively correlated,
- X and Z, negatively related
- support and confidence of X=>Z dominates

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

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

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

Other Interestingness Measures: Interest

- Interest (correlation, lift) $$\frac{P(A \wedge B)}{P(A)P(B)}$$
  - taking both P(A) and P(B) in consideration
  - 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

Constraint-based association mining

Constraint-Based Mining

Interactive, exploratory mining

kinds of constraints

- Knowledge type constraint- classification, association, etc.
- Data constraint: SQL-like queries
- Dimension/level constraints
- Rule constraint
- Interestingness constraints

Rule Constraints in Association Mining

Two kind of rule constraints:

- Rule form constraints: meta-rule guided mining.

  P(x, y) ^ Q(x, w) $\rightarrow$ takes(x, “database systems”).

- Rule (content) constraint: constraint-based query optimization (Ng, et al., SIGMOD’98).

  sum(LHS) < 100 ^ min(LHS) > 20 ^ count(LHS) > 3 ^ sum(RHS) > 1000

1-variable vs. 2-variable constraints

- 1-var: A constraint confining only one side (L/R) of the rule, e.g., as shown above.
- 2-var: A constraint confining both sides (L and R).

  sum(LHS) < min(RHS) ^ max(RHS) < 5* sum(LHS)

Constrain-Based Association Query

Database: (1) trans (TID, Itemset), (2) itemInfo (Item, Type, Price)

A constrained asso. query (CAQ) is in the form of $\{(S_1, S_2)/C\}$,

- where C is a set of constraints on S_1, S_2 including frequency constraint

A classification of (single-variable) constraints:

- Class constraint: $S \subset A$. *e.g.* $S \subset Item$
- Domain constraint:
  - $S \theta v$, $\theta \in \{=, \neq, <, \leq, >, \geq\}$. *e.g.* $S.Price < 100$
  - $v \theta S$, θ is $\in$ or $\notin$. *e.g.* $snacks \notin S.Type$
  - $V \theta S$, or $S \theta V$, $\theta \in \{\subseteq, \subset, \not\subset, =, \neq\}$
    - *e.g.* $\{snacks, sodas\} \subseteq S.Type$
- Aggregation constraint: $agg(S)\ \theta\ v$, where *agg* is in {*min, max, sum, count, avg*}, and $\theta \in \{=, \neq, <, \leq, >, \geq\}$.
  - *e.g.* $count(S_1.Type) = 1$, $avg(S_2.Price) < 100$

Constrained Association Query Optimization Problem

Given a CAQ = { (S_1, S_2) / C }, the algorithm should be :

- sound: It only finds frequent sets that satisfy the given constraints C
- complete: All frequent sets satisfy the given constraints C are found

A naïve solution:

- Apply Apriori for finding all frequent sets, and then to test them for constraint satisfaction one by one.

Our approach:

- Comprehensive analysis of the properties of constraints and try to push them as deeply as possible inside the frequent set computation.

Anti-monotone and Monotone Constraints

A constraint C_a is anti-monotone iff. for any pattern S not satisfying C_a, none of the super-patterns of S can satisfy C_a

A constraint C_m is monotone iff. for any pattern S satisfying C_m, every super-pattern of S also satisfies it

Succinct Constraint

A subset of item I_s is a succinct set, if it can be expressed as $\sigma_p(I)$ for some selection predicate p, where σ is a selection operator

$SP \subseteq 2^I$ is a succinct power set, if there is a fixed number of succinct set $I_1, \ldots, I_k \subseteq I$, s.t. SP can be expressed in terms of the strict power sets of $I_1, \ldots, I_k$ using union and minus

A constraint C_s is succinct provided $SAT_{C_s}(I)$ is a succinct power set

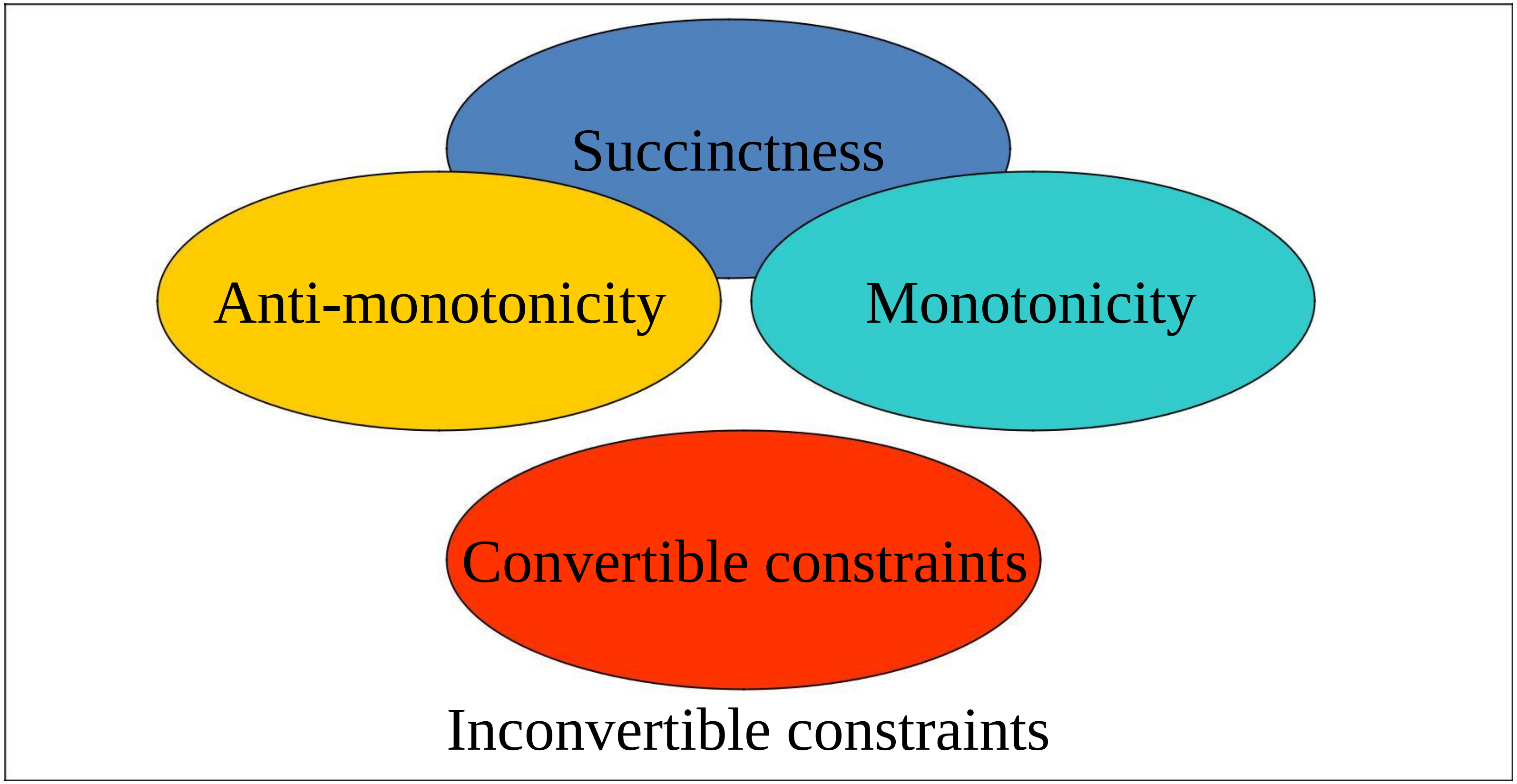
Convertible Constraint

Suppose all items in patterns are listed in a total order R

A constraint C is convertible anti-monotone iff a pattern S satisfying the constraint implies that each suffix of S w.r.t. R also satisfies C

A constraint C is convertible monotone iff a pattern S satisfying the constraint implies that each pattern of which S is a suffix w.r.t. R also satisfies C

Relationships Among Categories of Constraints



Property of Constraints: Anti-Monotone

Anti-monotonicity: *If a set S violates the constraint, any superset of S violates the constraint.*

Examples:

- *sum(S.Price)* $\leq$ *v* is anti-monotone
- *sum(S.Price)* $\geq$ *v* is not anti-monotone
- *sum(S.Price)* = *v* is partly anti-monotone

Application:

- Push "*sum(S.price)* $\leq$ 1000" deeply into iterative frequent set computation.

Characterization of Anti-Monotonicity Constraints

$S\ \theta\ v,\ \theta \in \{=, \leq, \geq\}$	yes
$v \in S$	no
$S \supseteq V$	no
$S \subseteq V$	yes
$S = V$	partly
$\min(S) \leq v$	no
$\min(S) \geq v$	yes
$\min(S) = v$	partly
$\max(S) \leq v$	yes
$\max(S) \geq v$	no
$\max(S) = v$	partly
$\text{count}(S) \leq v$	yes
$\text{count}(S) \geq v$	no
$\text{count}(S) = v$	partly
$\text{sum}(S) \leq v$	yes
$\text{sum}(S) \geq v$	no
$\text{sum}(S) = v$	partly
$\text{avg}(S)\ \theta\ v,\ \theta \in \{=, \leq, \geq\}$	convertible
(frequent constraint)	(yes)

Example of Convertible Constraints: Avg(S) θ V

Let R be the value descending order over the set of items

- E.g. I={9, 8, 6, 4, 3, 1}

Avg(S) $\geq$ v is convertible monotone w.r.t. R

- If S is a suffix of S_1, avg(S_1) $\geq$ avg(S)
  - {8, 4, 3} is a suffix of {9, 8, 4, 3}
  - avg({9, 8, 4, 3})=6 $\geq$ avg({8, 4, 3})=5
- If S satisfies avg(S) $\geq$v, so does S_1
  - {8, 4, 3} satisfies constraint avg(S) $\geq$ 4, so does {9, 8, 4, 3}

Property of Constraints: Succinctness

Succinctness:

- For any set S_1 and S_2 satisfying C, $S_1 \cup S_2$ satisfies C
- Given A_1 is the sets of size 1 satisfying C, then any set S satisfying C are based on A_1 , i.e., it contains a subset belongs to A_1 ,

Example :

- *sum(S.Price)* $\geq$ *v* is not succinct
- *min(S.Price)* $\leq$ *v* is succinct

Optimization:

- If C is succinct, then C is pre-counting prunable. The satisfaction of the constraint alone is not affected by the iterative support counting.

Characterization of Constraints by Succinctness

S θ v, θ ∈ { =, ≤, ≥ }	Yes
v ∈ S	yes
S ⊇ V	yes
S ⊆ V	yes
S = V	yes
min(S) ≤ v	yes
min(S) ≥ v	yes
min(S) = v	yes
max(S) ≤ v	yes
max(S) ≥ v	yes
max(S) = v	yes
count(S) ≤ v	weakly
count(S) ≥ v	weakly
count(S) = v	weakly
sum(S) ≤ v	no
sum(S) ≥ v	no
sum(S) = v	no
avg(S) θ v, θ ∈ { =, ≤, ≥ }	no
(frequent constraint)	(no)

UNIT – 4

Classification and Prediction

What is Classification & Prediction

Classification:

– predicts categorical class labels
– classifies data (constructs a model) based on the training set and the values (class labels) in a classifying attribute and uses it in classifying new data

Prediction:

– models continuous-valued functions
– predicts unknown or missing values

Applications

– credit approval
– target marketing
– medical diagnosis
– treatment effectiveness analysis

Classification—A Two-Step Process

Learning step- describing a set of predetermined classes

- Each tuple/sample is assumed to belong to a predefined class, as determined by the class label attribute
- The set of tuples used for model construction: training set
- The model is represented as classification rules, decision trees, or mathematical formulae

Classification- for classifying future or unknown objects

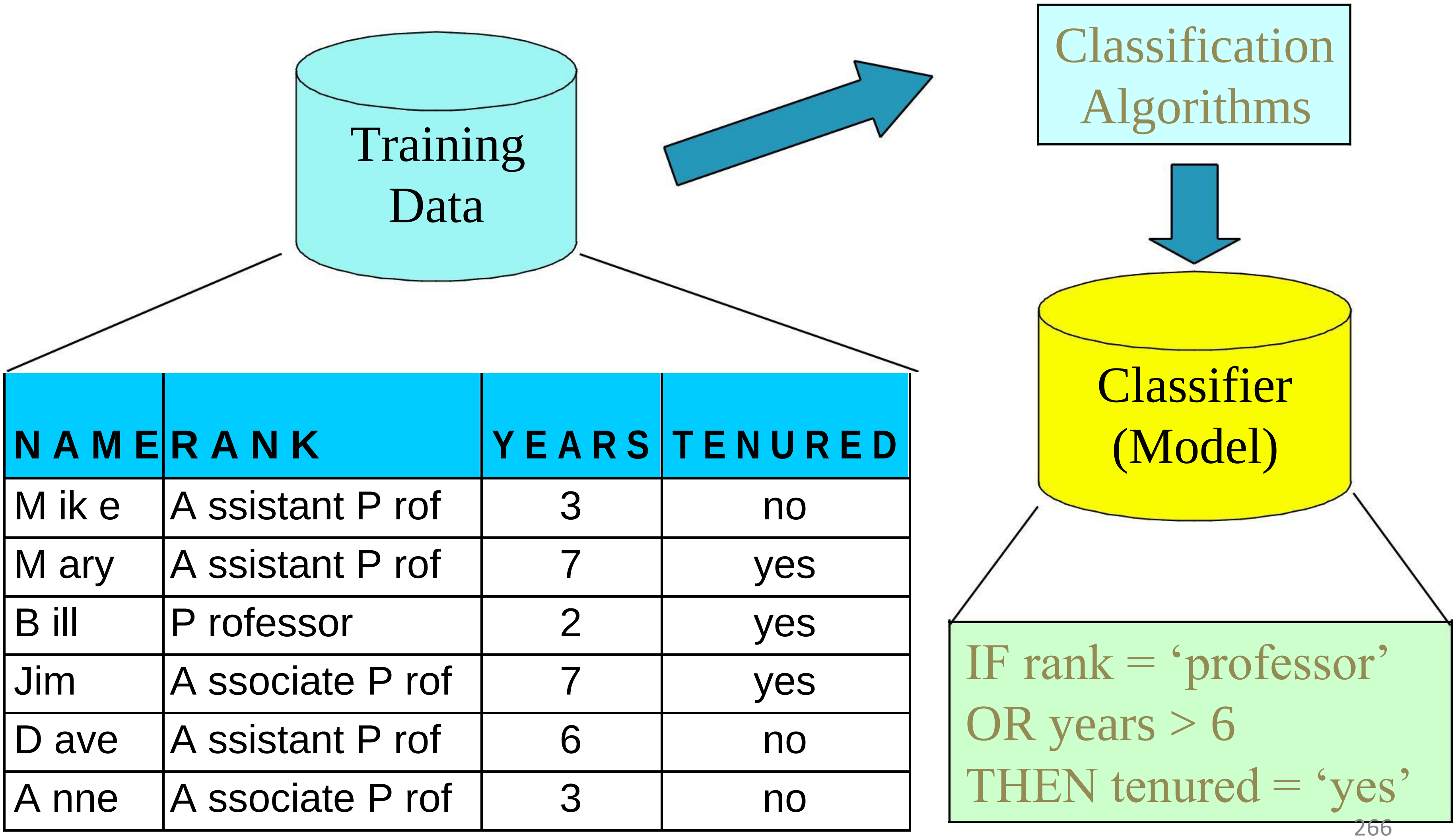
- Estimate accuracy of the model

  The known label of test sample is compared with the classified result from the model

  Accuracy rate is the percentage of test set samples that are correctly classified by the model

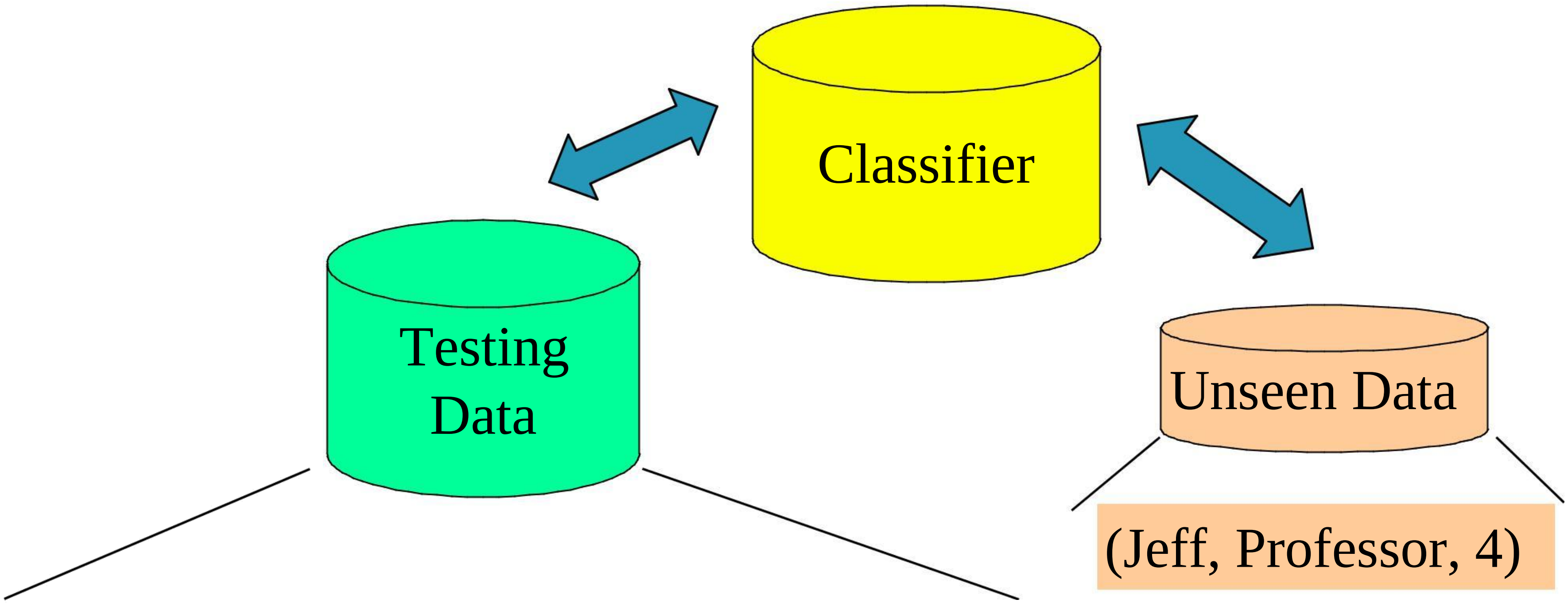
  Test set is independent of training set, otherwise over-fitting will occur

Classification Process : Model Construction

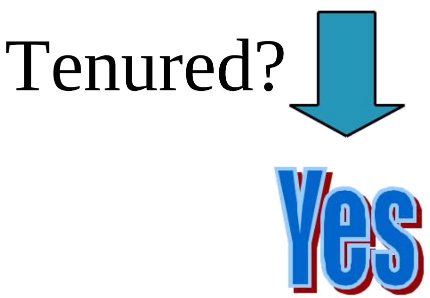


NAME	RANK	YEARS	TENURED
Mike	Assistant Prof	3	no
Mary	Assistant Prof	7	yes
Bill	Professor	2	yes
Jim	Associate Prof	7	yes
Dave	Assistant Prof	6	no
Anne	Associate Prof	3	no

Classification Process : Use the Model in Prediction



NAME	RANK	YEARS	TENURED
Tom	Assistant Prof	2	no
Merlisa	Associate Prof	7	no
George	Professor	5	yes
Joseph	Assistant Prof	7	yes



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

Issues regarding classification and prediction

Issues regarding classification and prediction - Preparing the data for classification and prediction

Data cleaning

– Preprocess data in order to reduce noise and handle missing values

Relevance analysis (feature selection)

– Remove the irrelevant or redundant attributes

Data transformation

– Generalize and/or normalize data

Issues regarding classification and prediction Comparing Classification Methods

- Accuracy
- Speed and scalability
  - time to construct the model
  - time to use the model
- Robustness
  - handling noise and missing values
- Scalability
  - efficiency in disk-resident databases
- Interpretability:
  - understanding and insight provded by the model
- interpretability
  - decision tree size
  - compactness of classification rules

Classification by decision tree induction

Classification by Decision Tree Induction

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

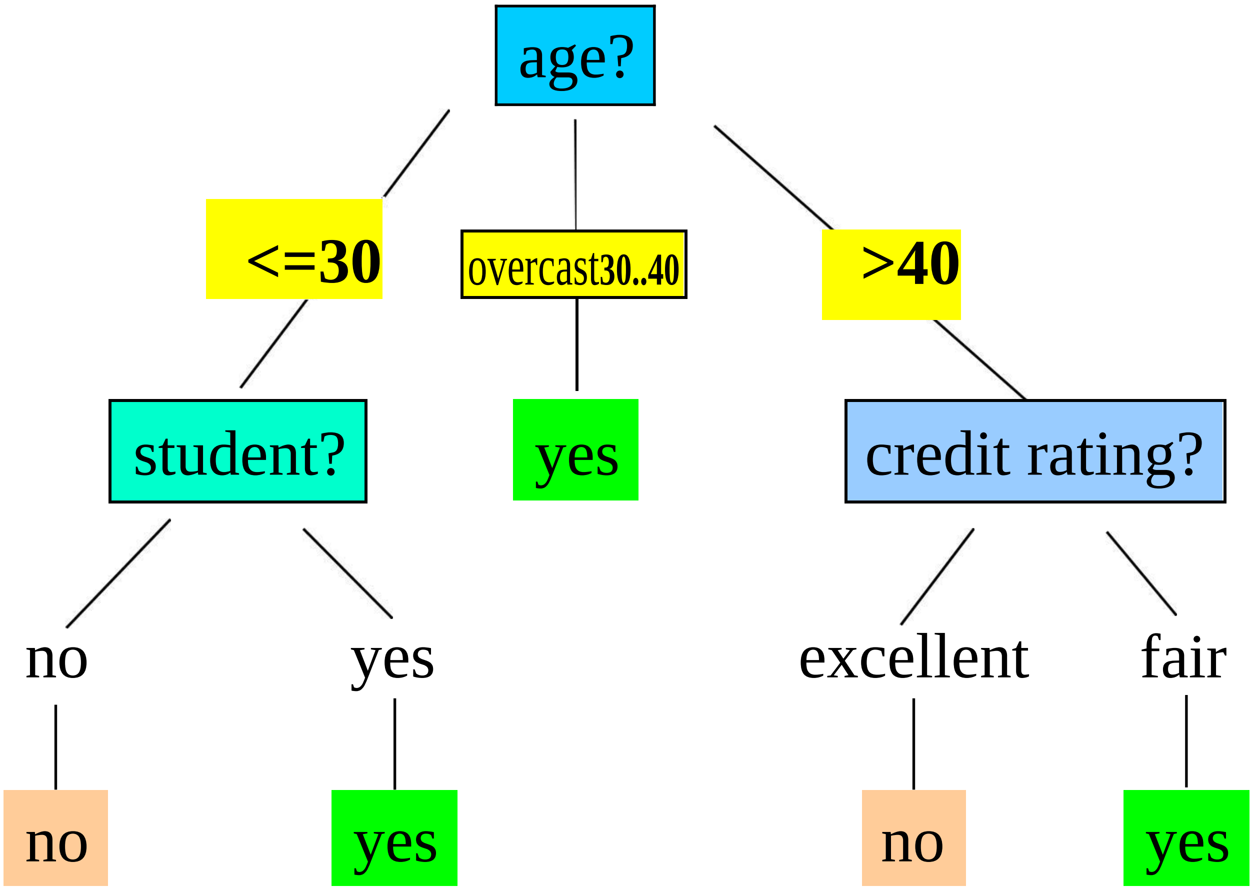
- Test the attribute values of the sample against the decision tree

Training Dataset

This follows an example from Quinlan's ID3

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

Output: A Decision Tree for "*buys_computer*"



Algorithm for Decision Tree Induction

Basic algorithm (a greedy algorithm)

- Tree is constructed in a top-down recursive divide-and-conquer manner
- At start, all the training examples are at the root
- Attributes are categorical (if continuous-valued, they are discretized in advance)
- Examples are partitioned recursively based on selected attributes
- Test attributes are selected on the basis of a heuristic or statistical measure (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

Attribute Selection Measure

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

Information Gain (ID3/C4.5)

Select the attribute with the highest information gain

Assume there are two classes, *P* and *N*

- Let the set of examples *S* contain *p* elements of class *P* and *n* elements of class *N*
- The amount of information, needed to decide if an arbitrary example in *S* belongs to *P* or *N* is defined as

$$I(p,n) = -\frac{p}{p+n}\log_2\frac{p}{p+n} - \frac{n}{p+n}\log_2\frac{n}{p+n}$$

Information Gain in Decision Tree Induction

Assume that using attribute A a set *S* will be partitioned into sets {$S_1, S_2, \ldots, S_v$}

– If S_i contains p_i examples of *P* and n_i examples of *N*, the entropy, or the expected information needed to classify objects in all subtrees S_i is

$$E(A) = \sum_{i=1}^{v} \frac{p_i + n_i}{p + n} I(p_i, n_i)$$

- The encoding information that would be gained by branching on *A*

$$Gain(A) = I(p, n) - E(A)$$

Attribute Selection by Information Gain Computation

Class P: buys_computer = "yes"

Class N: buys_computer = "no"

I(p, n) = I(9, 5) =0.940

Compute the entropy for *age*:

age	p_i	n_i	$I(p_i, n_i)$
<=30	2	3	0.971
30…40	4	0	0
>40	3	2	0.971

$$E(age) = \frac{5}{14} I(2,3) + \frac{4}{14} I(4,0) + \frac{5}{14} I(3,2) = 0.69$$

Hence

$$Gain(age) = I(p, n) - E(age)$$

Similarly

$$Gain(income) = 0.029$$

$$Gain(student) = 0.151$$

$$Gain(credit\_rating) = 0.048$$

Gini Index (IBM IntelligentMiner)

- If a data set *T* contains examples from *n* classes, gini index, *gini*(*T*) is defined as

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

where p_j is the relative frequency of class *j* in *T.*

If a data set *T* is split into two subsets T_1 and T_2 with sizes N_1 and N_2 respectively, the *gini* index of the split data contains examples from *n* classes, the *gini* index *gini*(*T*) is defined as

$$gini_{split}(T) = \frac{N_1}{N} gini(T_1) + \frac{N_2}{N} gini(T_2)$$

The attribute provides the smallest $gini_{split}(T)$ is chosen to split the node (*need to enumerate all possible splitting points for each attribute*).

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*"

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"

Approaches to Determine the Final Tree Size

Separate training and testing sets

Use cross validation, 10-fold cross validation

Use all the data for training

– apply a statistical test (chi-square) to estimate whether expanding or pruning a node may improve the entire distribution

Use minimum description length (MDL) principle:

– halting growth of the tree when the encoding is minimized

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)

Bayesian Classification

Bayesian Classification

Statical classifiers

Based on Baye's theorem

Naïve Bayesian classification

Class conditional independence

Bayesian belief netwoks

Bayesian Classification

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

Baye's Theorem

Let X be a data tuple and H be hypothesis, such that X belongs to a specific class C.

Posterior probability of a hypothesis h on X, P(h|X) follows the Baye's theorem

$$P(H \mid X) = \frac{P(X \mid H)P(H)}{P(X)}$$

Naïve Bayes Classifier

Let D be a training data set of tuples and associated class labels

$X = (x1, x2, x3,\ldots xn)$ and $m = C1, C2, C3,..Cm$

Bayes theorem:

$$P(Ci|X) = P(X|Ci)\cdot P(Ci) / P(X)$$

Naïve Baye's predicts that X belongs to class Ci if and only if

$$P(Ci/X) > P(Cj/X) \text{ for } 1<=j<=m,\ i!=j$$

Naïve Bayes Classifier

P(X) is constant for all classes

$P(C1)=P(C2)=.....=P(Cn)$

$P(X|Ci)\cdot P(Ci)$ is to be maximize

$P(Ci)=|Ci,d|/|D|$

Naïve Bayesian Classification

Naïve assumption: attribute independence

$$P(x_1,\ldots,x_k|C) = P(x_1|C)\cdot\ldots\cdot P(x_k|C)$$

If attribute is categorical:

$P(x_i|C)$ is estimated as the relative freq of samples having value x_i as i-th attribute in class C

If attribute is continuous:

$P(x_i|C)$ is estimated thru a Gaussian density function

Computationally easy in both cases

Play-tennis example: estimating $P(x_i|C)$

Outlook	Temperature	Humidity	W indy	Class
sunny	Hot	high	false	N
sunny	Hot	high	true	N
overcast	Hot	high	false	P
rain	Mild	high	false	P
rain	Cool	normal	false	P
rain	Cool	normal	true	N
overcast	Cool	normal	true	P
sunny	Mild	high	false	N
sunny	Cool	normal	false	P
rain	Mild	normal	false	P
sunny	Mild	normal	true	P
overcast	Mild	high	true	P
overcast	Hot	normal	false	P
rain	Mild	high	true	N

P(p) = 9/14	
P(n) = 5/14	

outlook	
P(sunny|p) = 2/9	P(sunny|n) = 3/5
P(overcast|p) = 4/9	P(overcast|n) = 0
P(rain|p) = 3/9	P(rain|n) = 2/5
temperature	
P(hot|p) = 2/9	P(hot|n) = 2/5
P(mild|p) = 4/9	P(mild|n) = 2/5
P(cool|p) = 3/9	P(cool|n) = 1/5
humidity	
P(high|p) = 3/9	P(high|n) = 4/5
P(normal|p) = 6/9	P(normal|n) = 2/5
windy	
P(true|p) = 3/9	P(true|n) = 3/5
P(false|p) = 6/9	P(false|n) = 2/5

Play-tennis example: classifying X

An unseen sample X = <rain, hot, high, false>

$P(X|p) \cdot P(p) =$
$P(rain|p) \cdot P(hot|p) \cdot P(high|p) \cdot P(false|p) \cdot P(p)$
$= 3/9 \cdot 2/9 \cdot 3/9 \cdot 6/9 \cdot 9/14 = 0.010582$

$P(X|n) \cdot P(n) =$
$P(rain|n) \cdot P(hot|n) \cdot P(high|n) \cdot P(false|n) \cdot P(n)$
$= 2/5 \cdot 2/5 \cdot 4/5 \cdot 2/5 \cdot 5/14 = 0.018286$

Sample X is classified in class n (donʻt play)

How effective are Bayesian classifiers?

makes computation possible

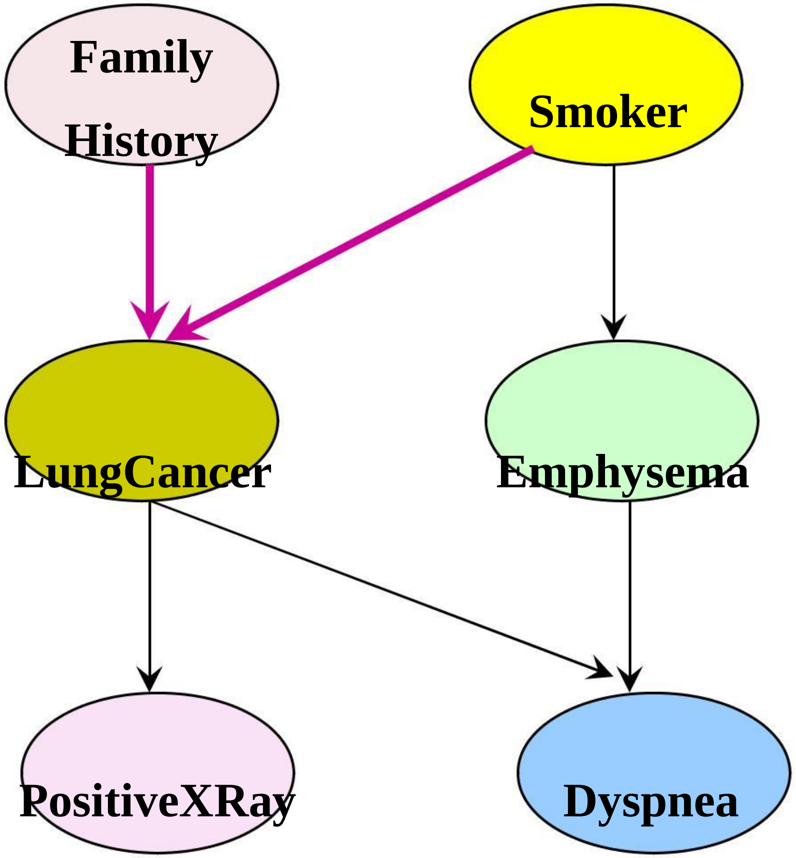
optimal classifiers when satisfied

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

Bayesian Belief Networks (I)



	(FH, S)	(FH, ~S)	(~FH, S)	(~FH, ~S)
LC	0.8	0.5	0.7	0.1
~LC	0.2	0.5	0.3	0.9

The conditional probability table for the variable LungCancer

Bayesian Belief Networks

Bayesian Belief Networks

Bayesian belief network allows a *subset* of the variables conditionally independent

A graphical model of causal relationships

Several cases of learning Bayesian belief networks

- Given both network structure and all the variables: easy
- Given network structure but only some variables
- When the network structure is not known in advance

The *k*-Nearest Neighbor Algorithm

- All instances correspond to points in the n-D space.
- The nearest neighbor are defined in terms of Euclidean distance.
- The target function could be discrete- or real-valued.
- For discrete-valued, the *k*-NN returns the most common value among the k training examples nearest to x_q.
- Vonoroi diagram: the decision surface induced by

1-NN for a typical set of training examples.

\+

Discussion on the *k*-NN Algorithm

The k-NN algorithm for continuous-valued target functions

– Calculate the mean values of the *k* nearest neighbors

Distance-weighted nearest neighbor algorithm

– Weight the contribution of each of the k neighbors according to their distance to the query point x_q
giving greater weight to closer neighbors

– Similarly, for real-valued target functions

- Robust to noisy data by averaging k-nearest neighbors ≡

$w \cdot \frac{1}{\quad}$

Curse of dimensionality: distance between neighbors could be dominated by irrelevant attributes.

– To overcome it, axes stretch or elimination of the least relevant attributes.

UNIT – 5

Cluster Analysis

General Applications of Clustering

Pattern Recognition

Spatial Data Analysis

– create thematic maps in GIS by clustering feature spaces

– detect spatial clusters and explain them in spatial data mining

Image Processing

Economic Science (market research)

WWW

– Document classification

– Cluster Weblog data to discover groups of similar access patterns

Examples of Clustering Applications

Marketing: Help marketers discover distinct groups in their customer bases, and then use this knowledge to develop targeted marketing programs

Land use: Identification of areas of similar land use in an earth observation database

Insurance: Identifying groups of motor insurance policy holders with a high average claim cost

City-planning: Identifying groups of houses according to their house type, value, and geographical location

Earth-quake studies: Observed earth quake epicenters should be clustered along continent faults

What Is Good Clustering?

A good clustering method will produce high quality clusters with

- high intra-class similarity
- low inter-class similarity

The quality of a clustering result depends on both the similarity measure used by the method and its implementation.

The quality of a clustering method is also measured by its ability to discover some or all of the hidden patterns.

Requirements of Clustering in Data Mining

Scalability

Ability to deal with different types of attributes

Discovery of clusters with arbitrary shape

Minimal requirements for domain knowledge to determine input parameters

Able to deal with noise and outliers

Insensitive to order of input records

High dimensionality

Incorporation of user-specified constraints

Interpretability and usability

Types of Data in Cluster Analysis

Data Structures

- Data matrix
  - (two modes)

$$\begin{bmatrix} x_{11} & \dots & x_{1f} & \dots & x_{1p} \\ \dots & \dots & \dots & \dots & \dots \\ x_{i1} & \dots & x_{if} & \dots & x_{ip} \\ \dots & \dots & \dots & \dots & \dots \\ x_{n1} & \dots & x_{nf} & \dots & x_{np} \end{bmatrix}$$

- Dissimilarity matrix
  - (one mode)

$$\begin{bmatrix} 0 & & & & \\ d(2,1) & 0 & & & \\ d(3,1) & d(3,2) & 0 & & \\ : & : & : & & \\ d(n,1) & d(n,2) & \dots & \dots & 0 \end{bmatrix}$$

Measure the Quality of Clustering

- Dissimilarity/Similarity metric: Similarity is expressed in terms of a distance function, which is typically metric: $d(i, j)$

There is a separate "quality" function that measures the "goodness" of a cluster.

The definitions of distance functions are usually very different for interval-scaled, boolean, categorical, ordinal and ratio variables.

Weights should be associated with different variables based on applications and data semantics.

It is hard to define "similar enough" or "good enough"

  - the answer is typically highly subjective.

Type of data in clustering analysis

Interval-scaled variables

Binary variables

Categorical, Ordinal, and Ratio Scaled variables

Variables of mixed types

Interval-valued variables

Standardize data

– Calculate the mean absolute deviation:

$$s_f = \frac{1}{n}(|x_{1f} - m_f| + |x_{2f} - m_f| + ... + |x_{nf} - m_f|)$$

where $m_f = \frac{1}{n}(x_{1f} + x_{2f} + ... + x_{nf})$.

– Calculate the standardized measurement (*z-score*)

$$z_{if} = \frac{x_{if} - m_f}{s_f}$$

Using mean absolute deviation is more robust than using standard deviation

Similarity and Dissimilarity Objects

Distances are normally used to measure the similarity or dissimilarity between two data objects

Some popular ones include: *Minkowski distance*:

$$d(i,j) = \sqrt[q]{(|x_{i1} - x_{j1}|^q + |x_{i2} - x_{j2}|^q + \ldots + |x_{ip} - x_{jp}|^q)}$$

where $i = (x_{i1}, x_{i2}, \ldots, x_{ip})$ and $j = (x_{j1}, x_{j2}, \ldots, x_{jp})$ are two p-dimensional data objects, and q is a positive integer

- If $q = 1$, d is Manhattan distance

$$d(i,j) = |x_{i1} - x_{j1}| + |x_{i2} - x_{j2}| + \ldots + |x_{ip} - x_{jp}|$$

Similarity and Dissimilarity Objects

- *If q = 2, d* is Euclidean distance:

$$d(i,j) = \sqrt{(|x_{i1} - x_{j1}|^2 + |x_{i2} - x_{j2}|^2 + ... + |x_{ip} - x_{jp}|^2)}$$

  – Properties

    $d(i,j) \geq 0$

    $d(i,i) = 0$

    $d(i,j) = d(j,i)$

    $d(i,j) \leq d(i,k) + d(k,j)$

Also one can use weighted distance, parametric Pearson product moment correlation, or other disimilarity measures.

Binary Variables

- A contingency table for binary data

Object *j*

	1	0	*sum*
1	a	b	$a+b$
Object *i* 0	c	d	$c+d$
sum	$a+c$	$b+d$	p

Simple matching coefficient (invariant, if the binary variable is *symmetric*):

$$d(i, j) = \frac{b+c}{a+b+c+d}$$

- Jaccard coefficient (noninvariant if the binary variable is *asymmetric*):

$$d(i, j) = \frac{b+c}{a+b+c}$$

Dissimilarity between Binary Variables

- Example

Name	Gender	Fever	Cough	Test-1	Test-2	Test-3	Test-4
Jack	M	Y	N	P	N	N	N
Mary	F	Y	N	P	N	P	N
Jim	M	Y	P	N	N	N	N

  - gender is a symmetric attribute
  - the remaining attributes are asymmetric binary
  - let the values Y and P be set to 1, and the value N be set to 0

$$d(jack, mary) = \frac{0+1}{2+0+1} = 0.33$$

$$d(jack, jim) = \frac{1+1}{1+1+1} = 0.67$$

$$d(jim, mary) = \frac{1+2}{1+1+2} = 0.75$$

Categorical Variables

A generalization of the binary variable in that it can take more than 2 states, e.g., red, yellow, blue, green

Method 1: Simple matching

– *M* is no of matches, *p* is total no of variables

$$d(i, j) = \frac{p - m}{p}$$

- Method 2: use a large number of binary variables
  - creating a new binary variable for each of the *M* nominal states

Ordinal Variables

An ordinal variable can be discrete or continuous

order is important, e.g., rank

Can be treated like interval-scaled

– replacing x_{if} by their rank $r_{if} \in \{1,...,M_f\}$

– map the range of each variable onto [0, 1] by replacing *i*-th object in the *f*-th variable by

$$z_{if} = \frac{r_{if} - 1}{M_f - 1}$$

– compute the dissimilarity using methods for interval-scaled variables

Ratio-Scaled Variables

Ratio-scaled variable: a positive measurement on a nonlinear scale, approximately at exponential scale, such as Ae^{Bt} or Ae^{-Bt}

Methods:

- treat them like interval-scaled variables
- apply logarithmic transformation

$$y_{if} = log(x_{if})$$

- treat them as continuous ordinal data treat their rank as interval-scaled.

Variables of Mixed Types

A database may contain all the six types of variables

– symmetric binary, asymmetric binary, nominal, ordinal, interval and ratio.

One may use a weighted formula to combine their effects.

$$d(i, j) = \frac{\sum_{f=1}^{p} \delta_{ij}^{(f)} d_{ij}^{(f)}}{\sum_{f=1}^{p} \delta_{ij}^{(f)}}$$

– f is binary or nominal:

$d_{ij}^{(f)} = 0$ if $x_{if} = x_{jf}$, or $d_{ij}^{(f)} = 1$ o.w.

– f is interval-based: use the normalized distance

– f is ordinal or ratio-scaled

- compute ranks r_{if} and
- and treat z_{if} as interval-scaled

$$z_{if} = \frac{r_{if} - 1}{M_f - 1}$$

A Categorization of Major Clustering Methods

Major Clustering Approaches

Partitioning algorithms

– Construct various partitions and then evaluate them by some criterion

Hierarchy algorithms

– Create a hierarchical decomposition of the set of data (or objects) using some criterion

Density-based

– based on connectivity and density functions

Grid-based

– based on a multiple-level granularity structure

Model-based

– A model is hypothesized for each of the clusters and the idea is to find the best fit of that model to each other

Partitioning Methods

Partitioning Algorithms: Basic Concept

Partitioning method: Construct a partition of a database ***D*** of ***n*** objects into a set of ***k*** clusters

Given a *k*, find a partition of *k clusters* that optimizes the chosen partitioning criterion

- Global optimal: exhaustively enumerate all partitions
- Heuristic methods: *k-means* and *k-medoids* algorithms
- *k-means* - Each cluster is represented by the center of the cluster
- *k-medoids* or PAM (Partition around medoids) - Each cluster is represented by one of the objects in the cluster

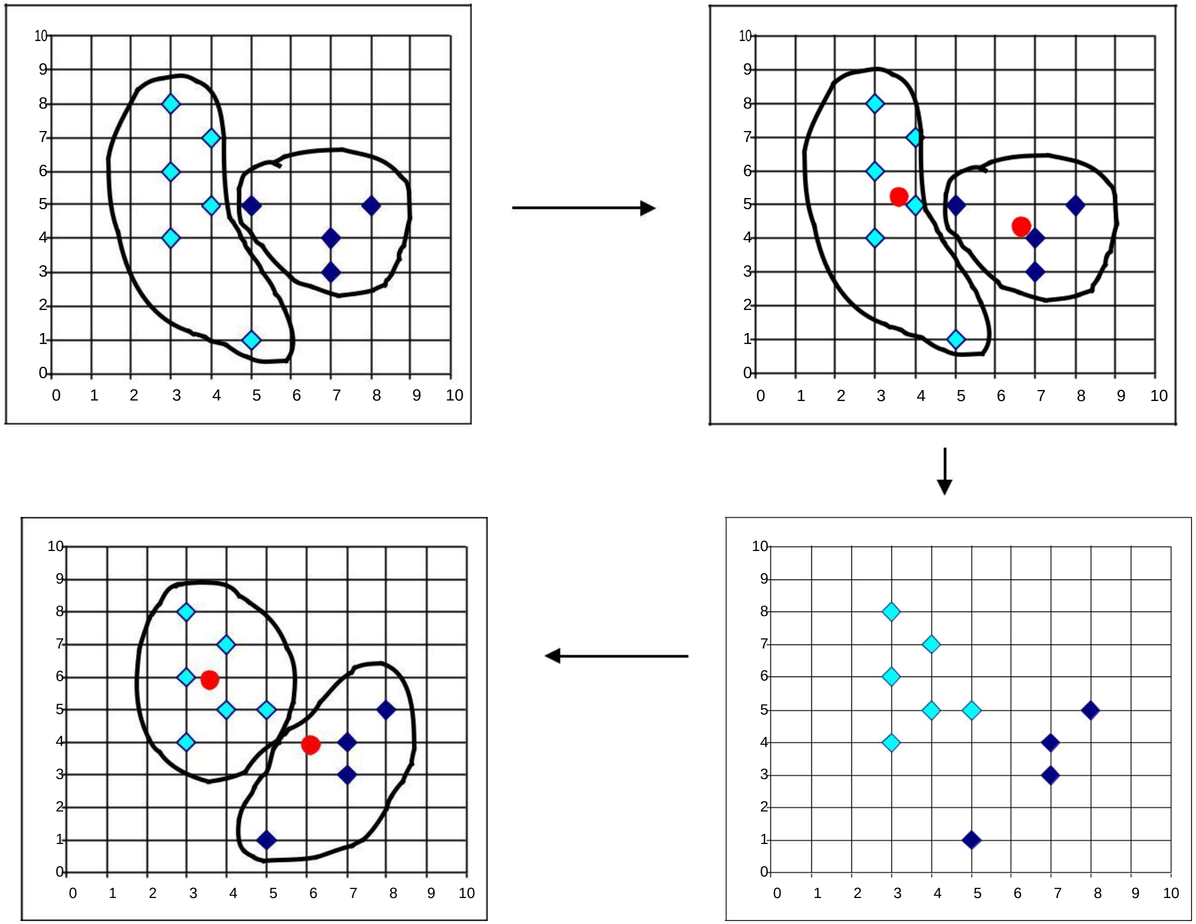
The *K-Means* Clustering Method

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

- Partition objects into *k* nonempty subsets
- Compute seed points as the centroids of the clusters of the current partition. The centroid is the center (mean point) of the cluster.
- Assign each object to the cluster with the nearest seed point.
- Go back to Step 2, stop when no more new assignment.

The *K-Means* Clustering Method

- Example



the *K-Means* Method

- <u>Strength</u>
  - *Relatively efficient*: *O*(*tkn*), where *n* is no of objects, *k* is no of clusters, and *t* is no of iterations. Normally, *k*,

    $t << n$.
  - Often terminates at a *local optimum*. The *global optimum* may be found using techniques such as: *deterministic annealing* and *genetic algorithms*

<u>Weakness</u>

  - Applicable only when *mean* is defined, then what about categorical data?
  - Need to specify *k,* the *number* of clusters, in advance
  - Unable to handle noisy data and *outliers*
  - Not suitable to discover clusters with *non-convex shapes*

Variations of the *K-Means* Method

A few variants of the *k-means* which differ in

- Selection of the initial *k* means
- Dissimilarity calculations
- Strategies to calculate cluster means

Handling categorical data: *k-modes*

- Replacing means of clusters with modes
- Using new dissimilarity measures to deal with categorical objects
- Using a frequency-based method to update modes of clusters
- A mixture of categorical and numerical data: *k-prototype* method

The *K-Medoids* Clustering Method

Find *representative* objects, called medoids, in clusters

PAM (Partitioning Around Medoids, 1987)

– starts from an initial set of medoids and iteratively replaces one of the medoids by one of the non-medoids if it improves the total distance of the resulting clustering

– *PAM* works effectively for small data sets, but does not scale well for large data sets

CLARA (Kaufmann & Rousseeuw, 1990)

CLARANS (Ng & Han, 1994): Randomized sampling

Focusing + spatial data structure (Ester et al., 1995)

PAM (Partitioning Around Medoids) (1987)

PAM (Kaufman and Rousseeuw, 1987), built in Splus

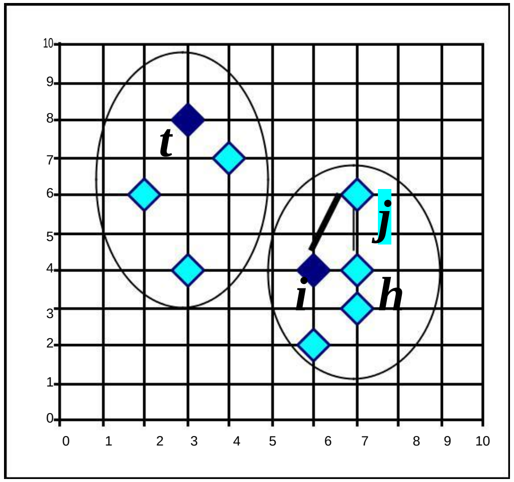
Use real object to represent the cluster

- Select ***k*** representative objects arbitrarily
- For each pair of non-selected object ***h*** and selected object ***i***, calculate the total swapping cost TC_{ih}
- For each pair of ***i*** and ***h***,

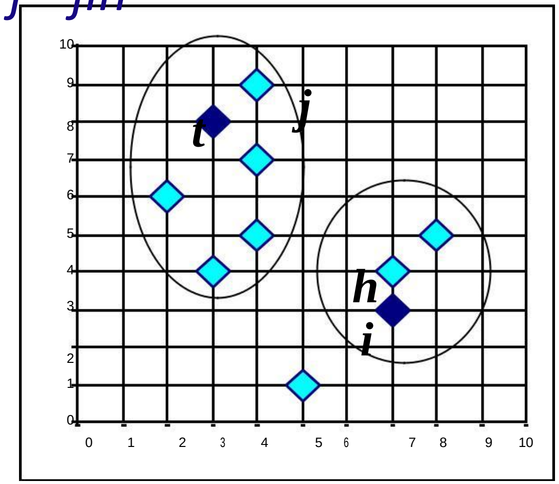
  If $TC_{ih} < 0$, ***i*** is replaced by ***h***

  Then assign each non-selected object to the most similar representative object
- repeat steps 2-3 until there is no change

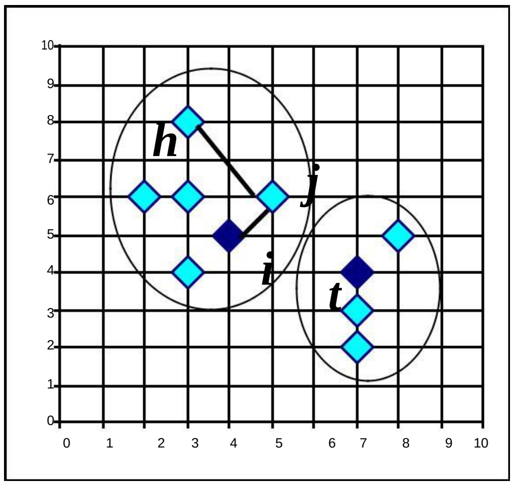
PAM Clustering: Total swapping cost $TC_{ih} = \sum_j C_{jih}$



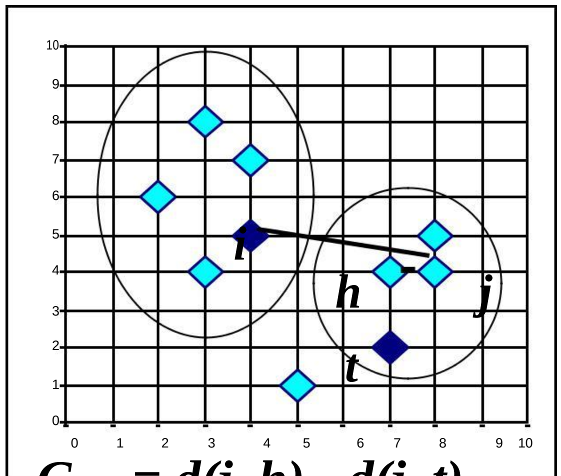
$C_{jih} = d(j, h) - d(j, i)$



$C_{jih} = 0$



$C_{jih} = d(j, t) - d(j, i)$



$C_{jih} = d(j, h) - d(j, t)$

CLARA (Clustering Large Applications) (1990)

CLARA (Kaufmann and Rousseeuw in 1990)

– Built in statistical analysis packages, such as S+

It draws *multiple samples* of the data set, applies *PAM* on each sample, and gives the best clustering as the output

<u>Strength</u>: deals with larger data sets than *PAM*

<u>Weakness:</u>

– Efficiency depends on the sample size

– A good clustering based on samples will not necessarily represent a good clustering of the whole data set if the sample is biased

CLARANS (“Randomized” CLARA) *(1994)*

CLARANS (A Clustering Algorithm based on Randomized Search)
CLARANS draws sample of neighbors dynamically

The clustering process can be presented as searching a graph where every node is a potential solution, that is, a set of *k* medoids

If the local optimum is found, *CLARANS* starts with new randomly selected node in search for a new local optimum

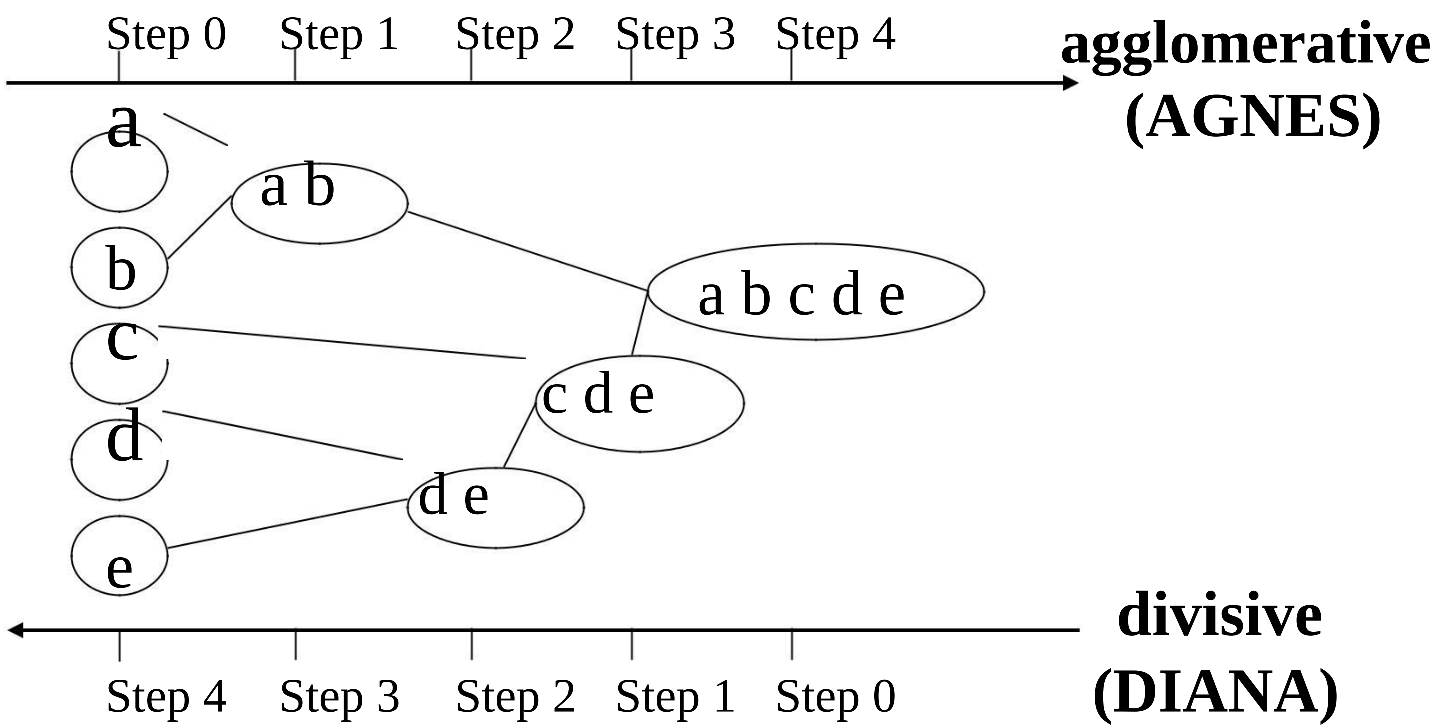
It is more efficient and scalable than both *PAM* and *CLARA*

Focusing techniques and spatial access structures may further improve its performance

Hierarchical Methods

Hierarchical Clustering

Use distance matrix as clustering criteria. This method does not require the number of clusters ***k*** as an input, but needs a termination condition



AGNES (Agglomerative Nesting)

Introduced in Kaufmann and Rousseeuw (1990)

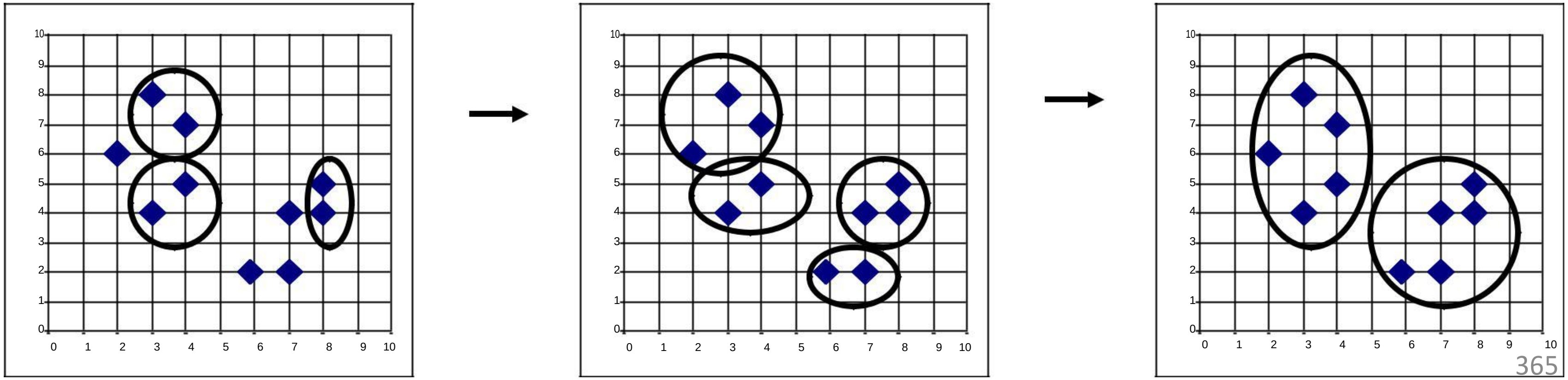
Implemented in statistical analysis packages, e.g., Splus

Use the Single-Link method and the dissimilarity matrix.

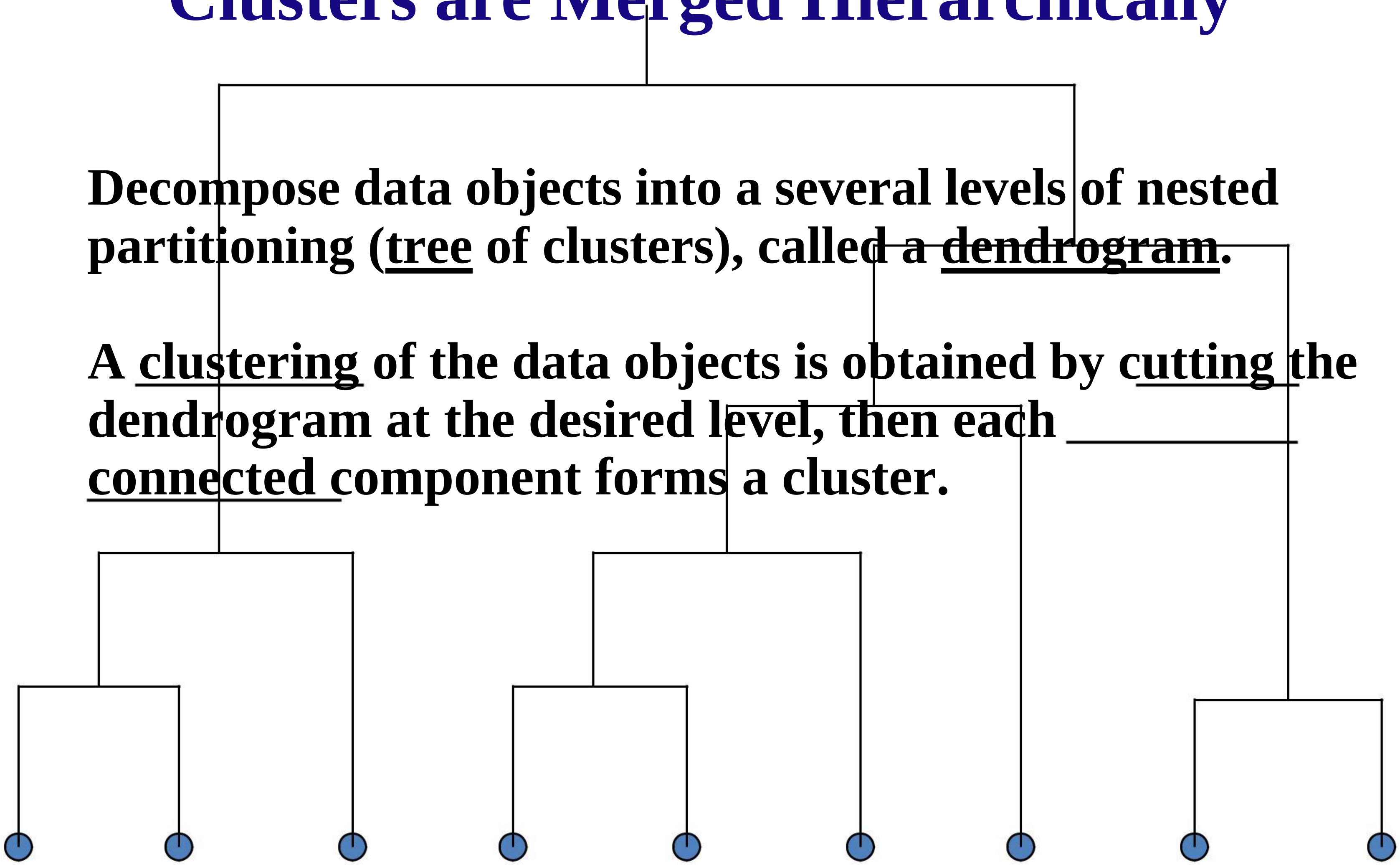
Merge nodes that have the least dissimilarity

Go on in a non-descending fashion

Eventually all nodes belong to the same cluster



A *Dendrogram* Shows How the Clusters are Merged Hierarchically



Decompose data objects into a several levels of nested partitioning (<u>tree</u> of clusters), called a <u>dendrogram</u>.

A <u>clustering</u> of the data objects is obtained by <u>cutting</u> the dendrogram at the desired level, then each <u>connected</u> component forms a cluster.

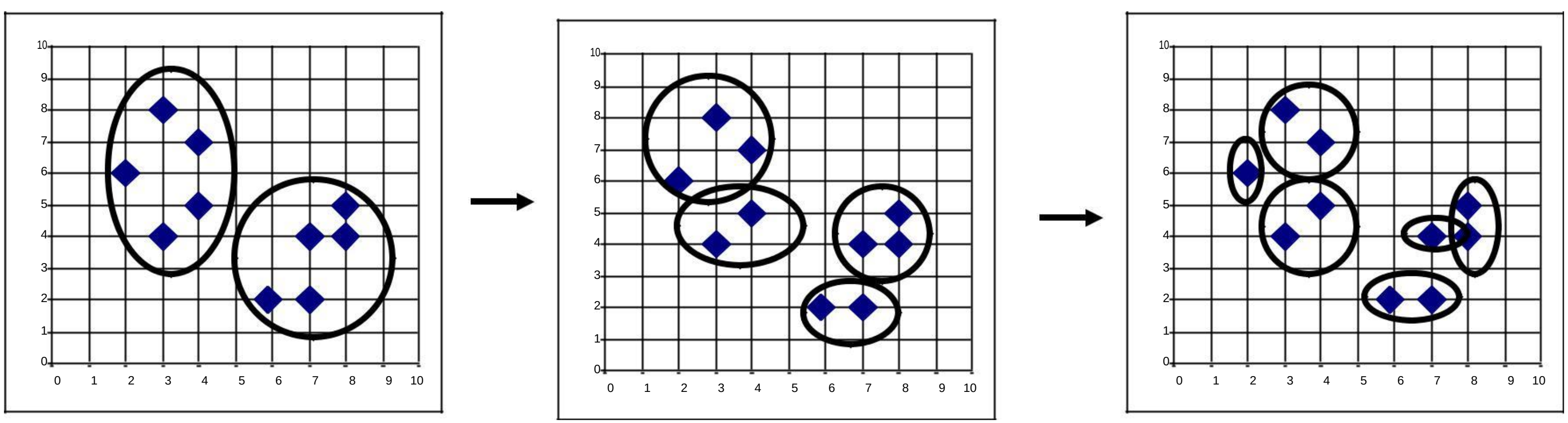
DIANA (Divisive Analysis)

Introduced in Kaufmann and Rousseeuw (1990)

Implemented in statistical analysis packages, e.g., Splus

Inverse order of AGNES

Eventually each node forms a cluster on its own



More on Hierarchical Clustering Methods

Major weakness of agglomerative clustering methods

- do not scale well: time complexity of at least $O(n^2)$, where n is the number of total objects
- can never undo what was done previously

Integration of hierarchical with distance-based clustering

- BIRCH (1996): uses CF-tree and incrementally adjusts the quality of sub-clusters
- CURE (1998): selects well-scattered points from the cluster and then shrinks them towards the center of the cluster by a specified fraction
- CHAMELEON (1999): hierarchical clustering using dynamic modeling

BIRCH (1996)

Birch: Balanced Iterative Reducing and Clustering using Hierarchies, 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 Feature Vector

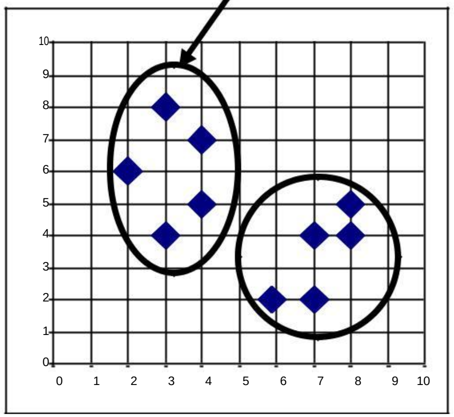
Clustering Feature: $CF = (N, \overrightarrow{LS}, SS)$

N: **Number of data points**

$LS: \sum_{i=1}^{N} \overrightarrow{X_i}$

$SS: \sum_{i=1}^{N} \overrightarrow{X_i}^2$

CF = (5, (16,30),(54,190))

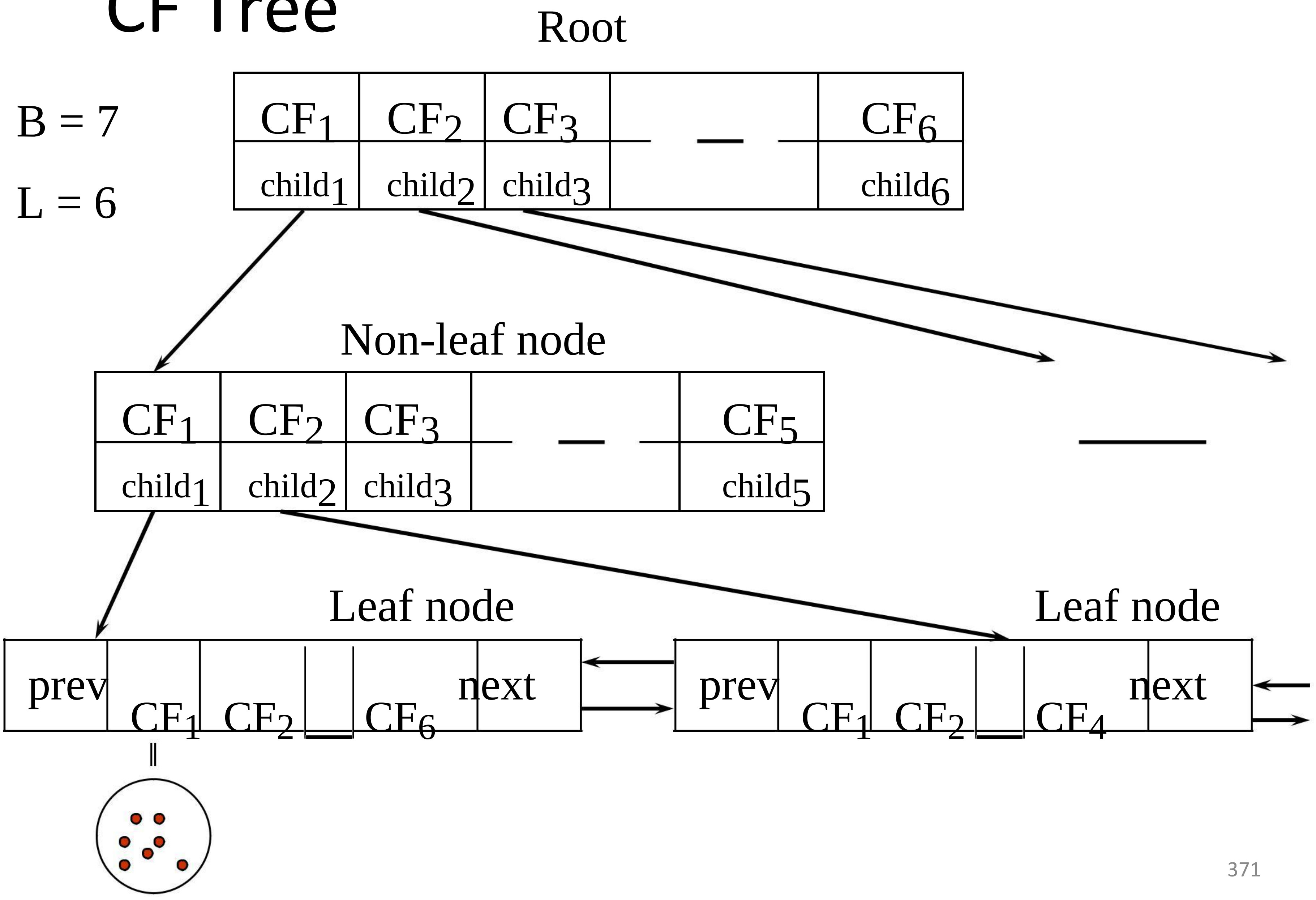


(3,4)
(2,6)
(4,5)
(4,7)
(3,8)

CF Tree

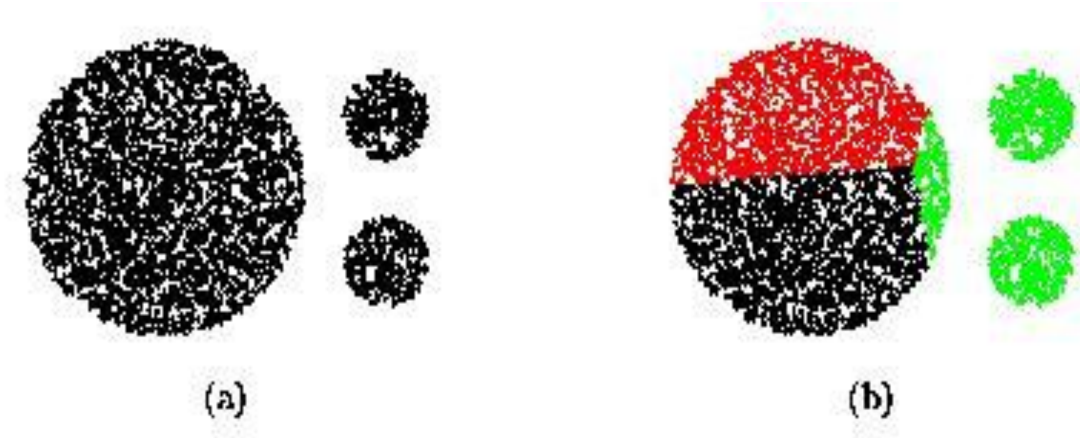
B = 7

L = 6



Root
CF1 CF2 CF3 CF6
child1 child2 child3 child6
Non-leaf node
CF1 CF2 CF3 CF5
child1 child2 child3 child5
Leaf node
prev CF1 CF2 CF6 next
Leaf node
prev CF1 CF2 CF4 next

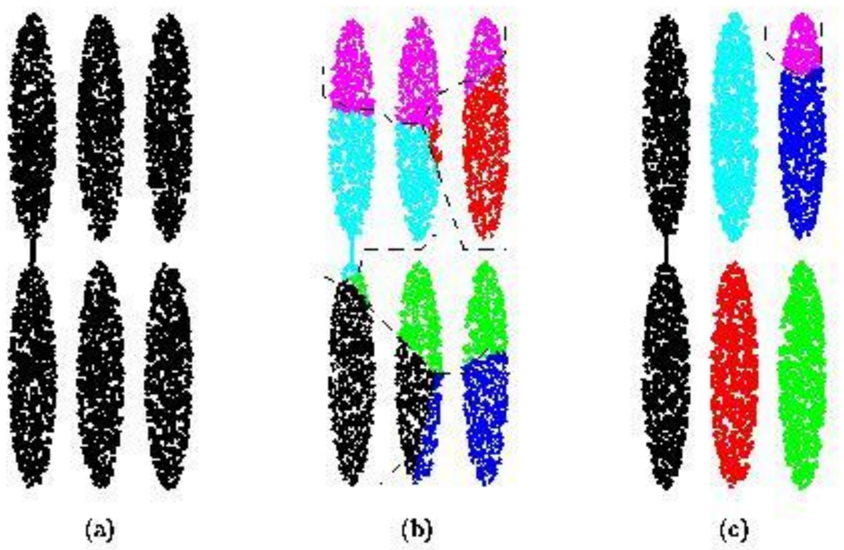
CURE (Clustering Using REpresentatives)



CURE: proposed by Guha, Rastogi & Shim, 1998

- Stops the creation of a cluster hierarchy if a level consists of *k* clusters
- Uses multiple representative points to evaluate the distance between clusters, adjusts well to arbitrary shaped clusters and avoids single-link effect

Drawbacks of Distance-Based Method



Drawbacks of square-error based clustering method

- Consider only one point as representative of a cluster
- Good only for convex shaped, similar size and density, and if *k* can be reasonably estimated

Cure: The Algorithm

– Draw random sample *s*.

– Partition sample to *p* partitions with size *s/p*

– Partially cluster partitions into *s/pq* clusters

– Eliminate outliers

  By random sampling

  If a cluster grows too slow, eliminate it.

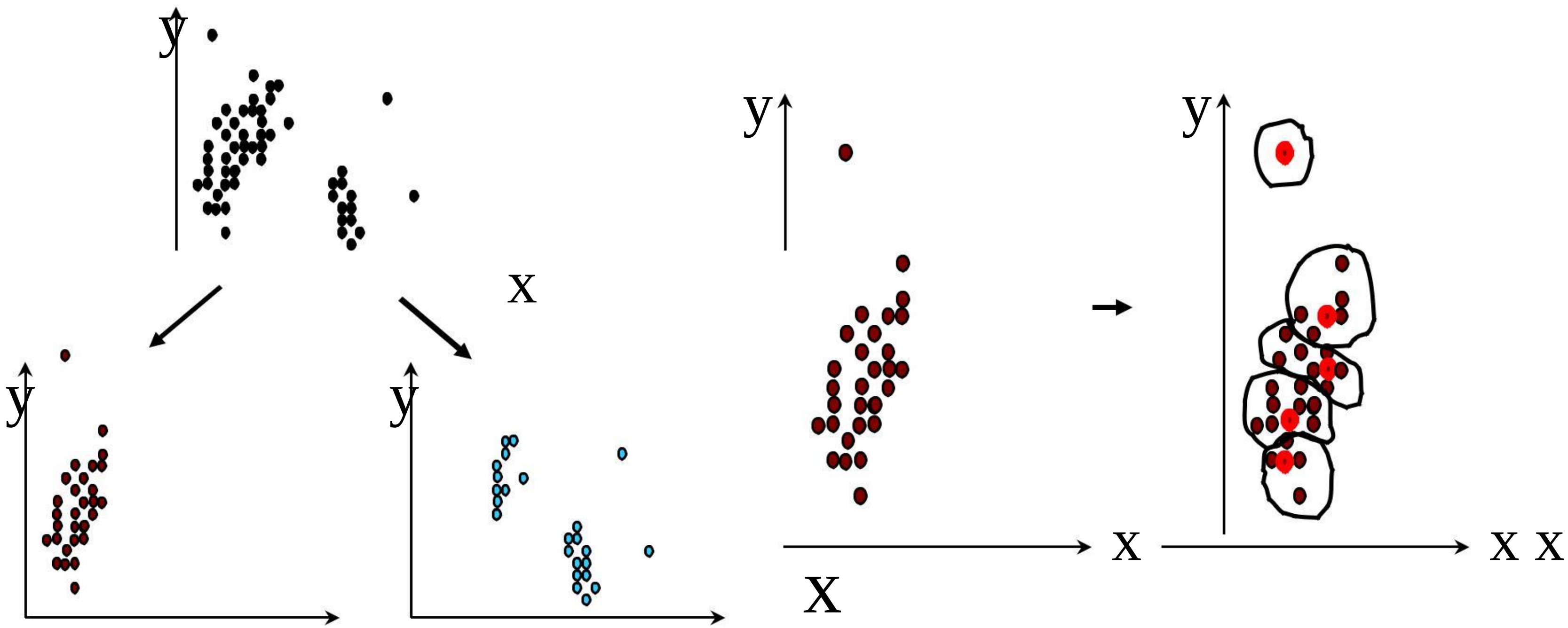
– Cluster partial clusters.

– Label data in disk

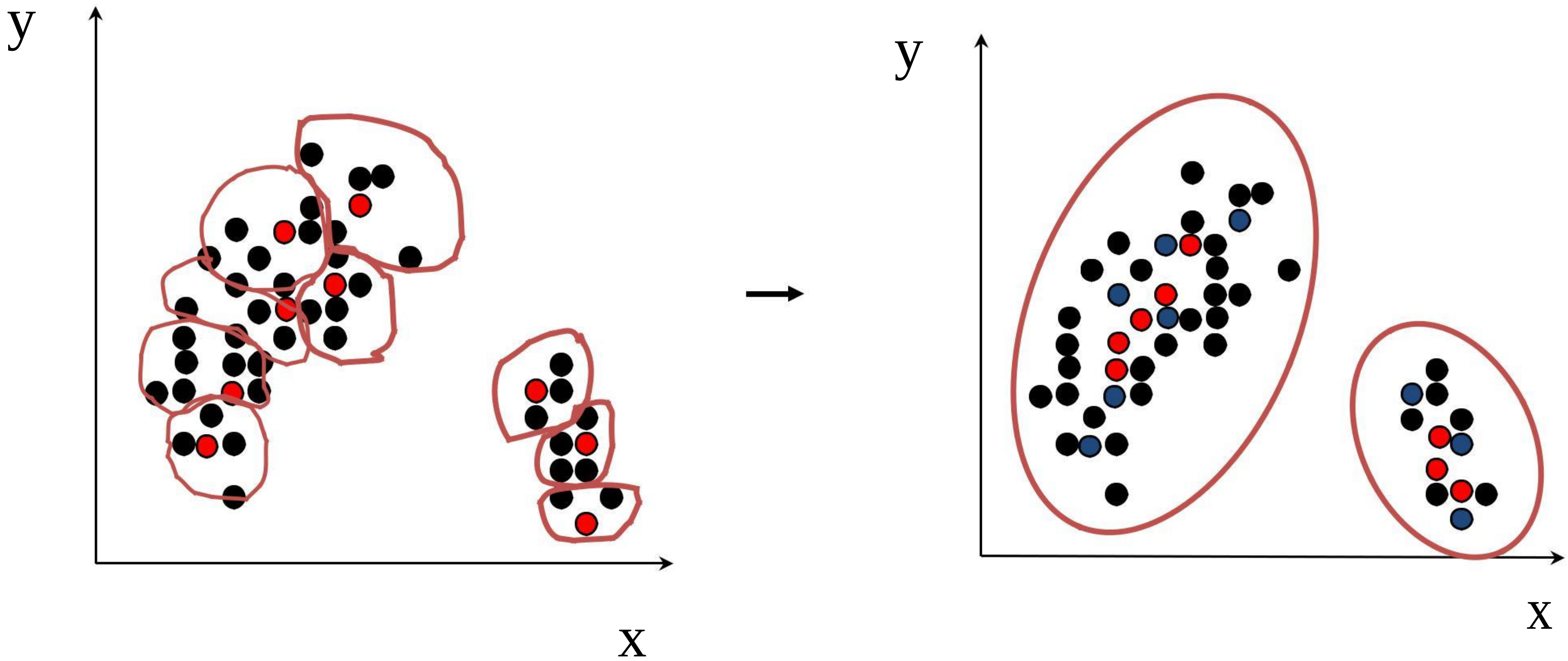
Data Partitioning and Clustering

- s = 50
- p = 2
- s/p = 25

- s/pq = 5



Cure: Shrinking Representative Points



Shrink the multiple representative points towards the gravity center by a fraction of α.

Multiple representatives capture the shape of the cluster

Clustering Categorical Data: ROCK

ROCK: Robust Clustering using linKs,

by S. Guha, R. Rastogi, K. Shim (ICDE'99).

- Use links to measure similarity/proximity
- Not distance based
- Computational complexity: $O(n^2 + nm_m m_a + n^2 \log n)$

Basic ideas:

- Similarity function and neighbors: $Sim(T_1, T_2) = \frac{|T_1 \cap T_2|}{|T_1 \cup T_2|}$

Let $T_1 = \{1,2,3\}$, $T_2=\{3,4,5\}$

$$Sim(T1, T2) = \frac{|\{3\}|}{|\{1,2,3,4,5\}|} = \frac{1}{5} = 0.2$$

Rock: Algorithm

Links: The number of common neighbours for the two points.

{1,2,3}, {1,2,4}, {1,2,5}, {1,3,4}, {1,3,5}
{1,4,5}, {2,3,4}, {2,3,5}, {2,4,5}, {3,4,5}

{1,2,3} ⟵ 3 ⟶ {1,2,4}

Algorithm

- Draw random sample
- Cluster with links
- Label data in disk

CHAMELEON

CHAMELEON: hierarchical clustering using dynamic modeling, by G. Karypis, E.H. Han and V. Kumar'99

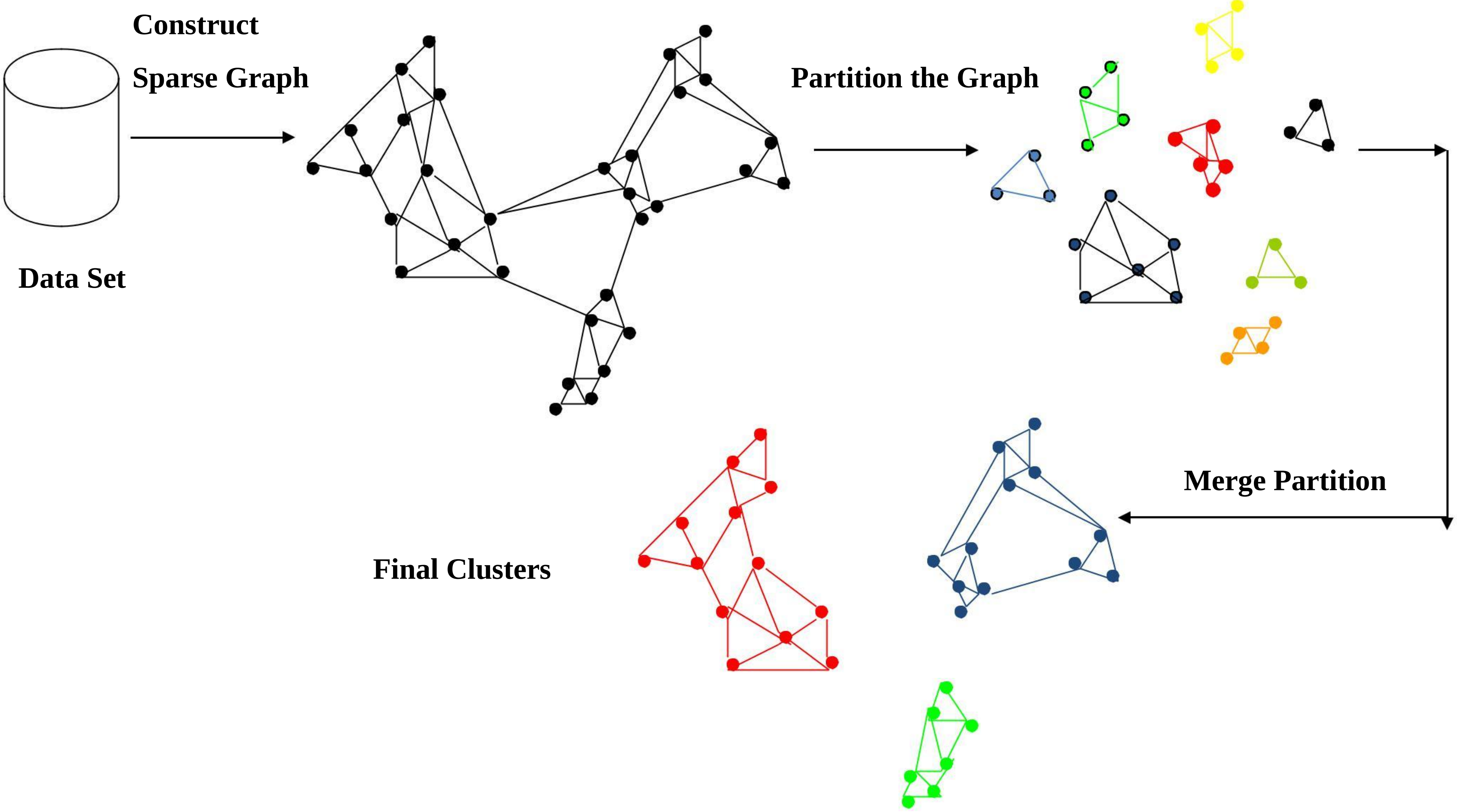
Measures the similarity based on a dynamic model

- Two clusters are merged only if the *interconnectivity* and *closeness (proximity)* between two clusters are high *relative to* the internal interconnectivity of the clusters and closeness of items within the clusters

A two phase algorithm

- 1. Use a graph partitioning algorithm: cluster objects into a large number of relatively small sub-clusters
- 2. Use an agglomerative hierarchical clustering algorithm: find the genuine clusters by repeatedly combining these sub-clusters

Overall Framework of CHAMELEON



Density-Based Methods

Density-Based Clustering Methods

Clustering based on density (local cluster criterion), such as density-connected points

Major features:

- Discover clusters of arbitrary shape
- Handle noise
- One scan
- Need density parameters as termination condition

Several methods

- DBSCAN
- OPTICS
- DENCLUE
- CLIQUE

Density-Based Clustering

Two parameters*:*

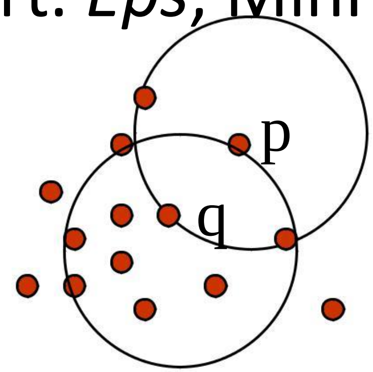
- *Eps*: Maximum radius of the neighbourhood
- *MinPts*: Minimum number of points in an Eps-neighbourhood of that point

$N_{Eps}(p)$: {q belongs to D | dist(p,q) <= Eps}

Directly density-reachable**:** A point *p* is directly density-reachable from a point *q* wrt. *Eps*, MinPts if

- 1) *p* belongs to $N_{Eps}(q)$
- 2) core point condition:

$|N_{Eps}(q)| >= MinPts$



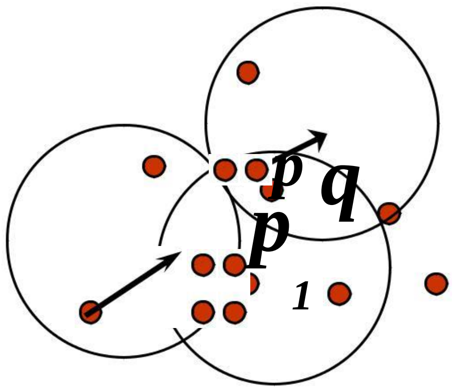
MinPts = 5

Eps = 1 cm

Density-Based Clustering

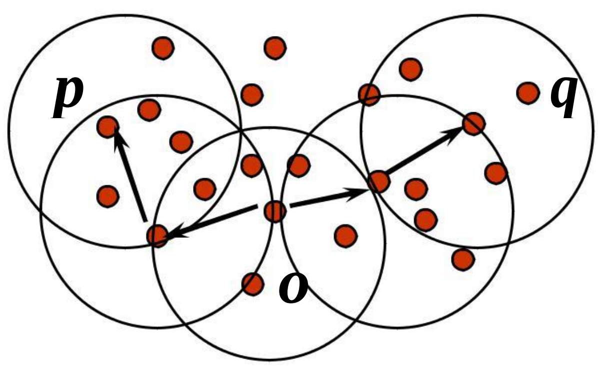
Density-reachable:

– A point p is density-reachable from a point q wrt. *Eps*, *MinPts* if there is a chain of points $p_1, \ldots, p_n$, $p_1 = q$, $p_n = p$ such that p_{i+1} is directly density-reachable from p_i



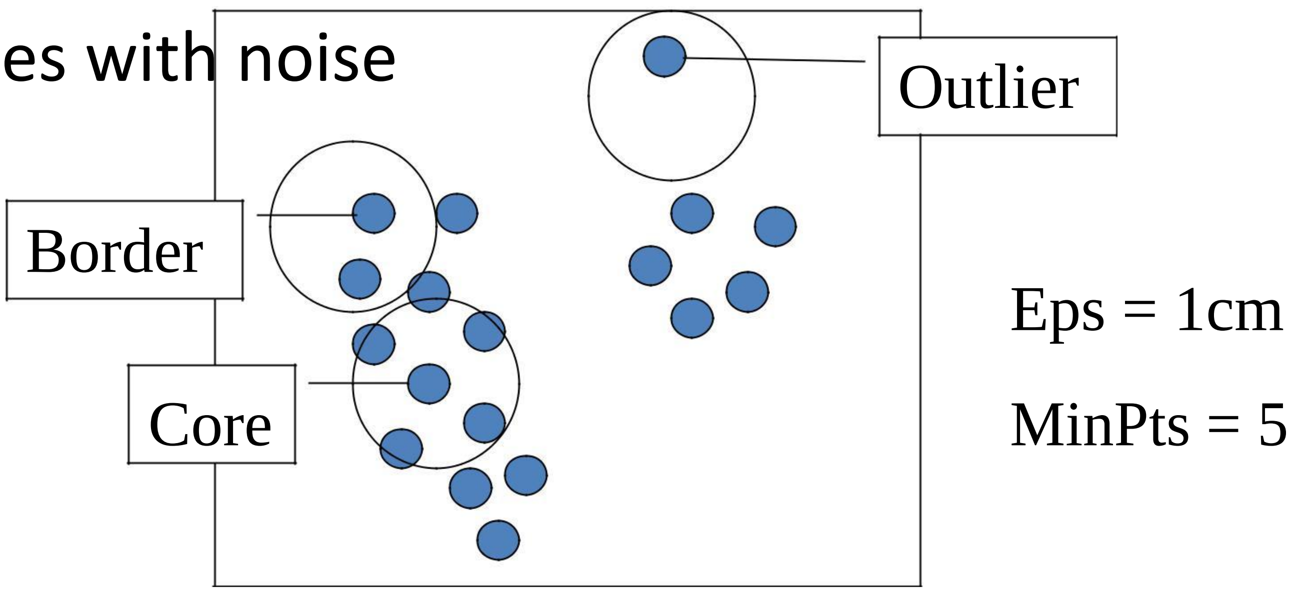
- Density-connected

– A point p is density-connected to a point q wrt. *Eps*, *MinPts* if there is a point o such that both, p and q are density-reachable from o wrt. *Eps* and *MinPts*.



DBSCAN: Density Based Spatial Clustering of Applications with Noise

- Relies on a *density-based* notion of cluster: A *cluster* is defined as a maximal set of density-connected points
- Discovers clusters of arbitrary shape in spatial databases with noise



Outlier Analysis

What Is Outlier Discovery?

What are outliers?

- The set of objects are considerably dissimilar from the remainder of the data
- Example: Sports: Michael Jordon, Wayne Gretzky, ...

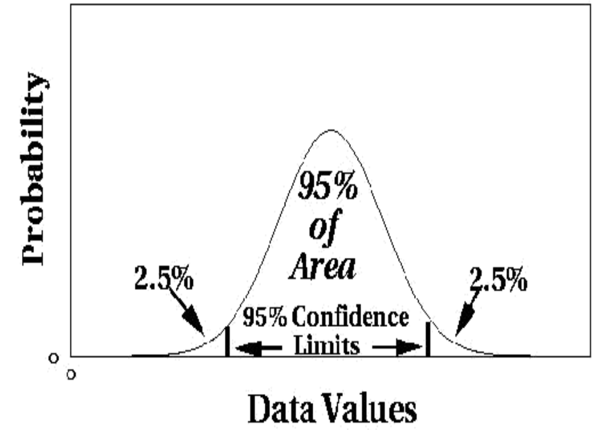
Problem

- Find top n outlier points

Applications:

- Credit card fraud detection
- Telecom fraud detection
- Customer segmentation
- Medical analysis

Outlier Discovery: Statistical Approaches



- Assume a model underlying distribution that generates data set (e.g. normal distribution)
- Use discordancy tests depending on
  - data distribution
  - distribution parameter (e.g., mean, variance)
  - number of expected outliers
- Drawbacks
  - most tests are for single attribute
  - In many cases, data distribution may not be known

Outlier Discovery: Distance-Based Approach

- Introduced to counter the main limitations imposed by statistical methods
  - We need multi-dimensional analysis without knowing data distribution.
- Distance-based outlier: A DB(p, D)-outlier is an object O in a dataset T such that at least a fraction p of the objects in T lies at a distance greater than D from O
- Algorithms for mining distance-based outliers
  - Index-based algorithm
  - Nested-loop algorithm
  - Cell-based algorithm

Outlier Discovery: Deviation-Based Approach

Identifies outliers by examining the main characteristics of objects in a group

Objects that “deviate” from this description are considered outliers

sequential exception technique

- simulates the way in which humans can distinguish unusual objects from among a series of supposedly like objects

OLAP data cube technique

- uses data cubes to identify regions of anomalies in largemultidimensionaldata.