Data Mining: Introduction

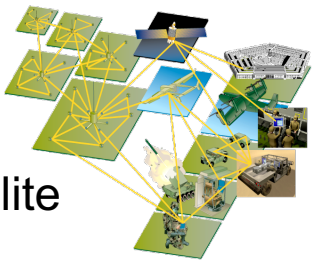
Introduction to Data Mining

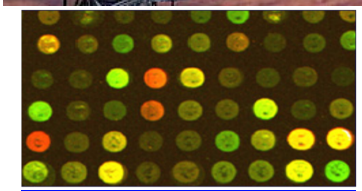
Why Mine Data? Commercial Viewpoint

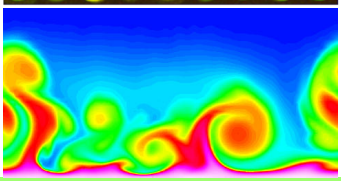
- Lots of data is being collected and warehoused
  - Web data, e-commerce
  - purchases at department/ grocery stores
  - Bank/Credit Card transactions

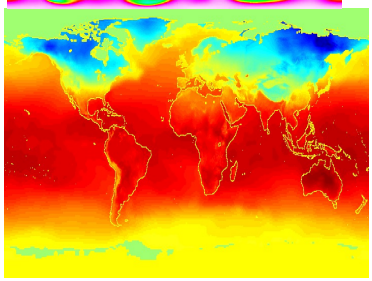
- Computers have become cheaper and more powerful
- Competitive Pressure is Strong
  - Provide better, customized services for an *edge* (e.g. in Customer Relationship Management)

Why Mine Data? Scientific Viewpoint





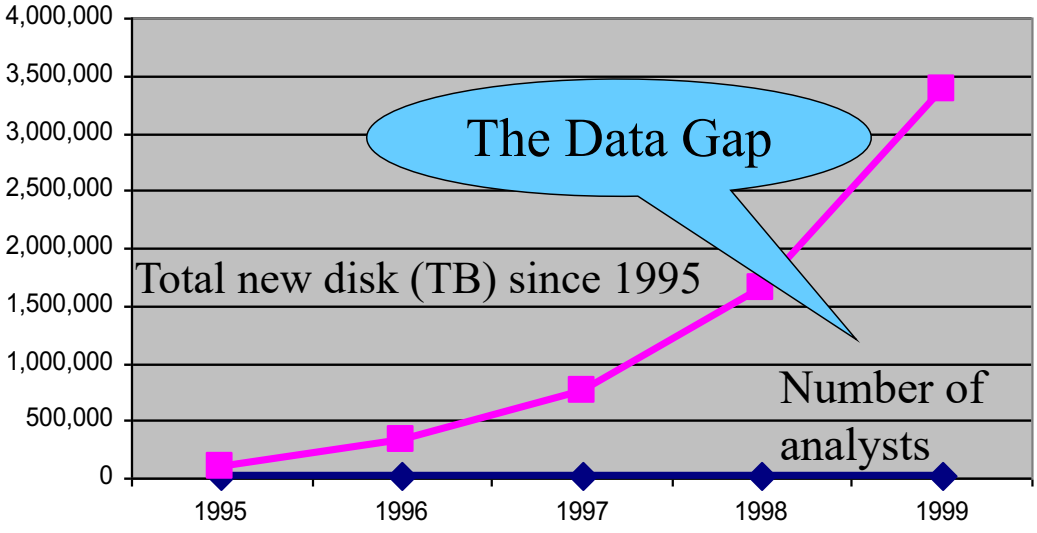




- Data collected and stored at enormous speeds (GB/hour)
  - remote sensors on a satellite
  - telescopes scanning the skies
  - microarrays generating gene expression data
  - scientific simulations generating terabytes of data
- Traditional techniques infeasible for raw data
- Data mining may help scientists
  - in classifying and segmenting data
  - in Hypothesis Formation

Mining Large Data Sets - Motivation

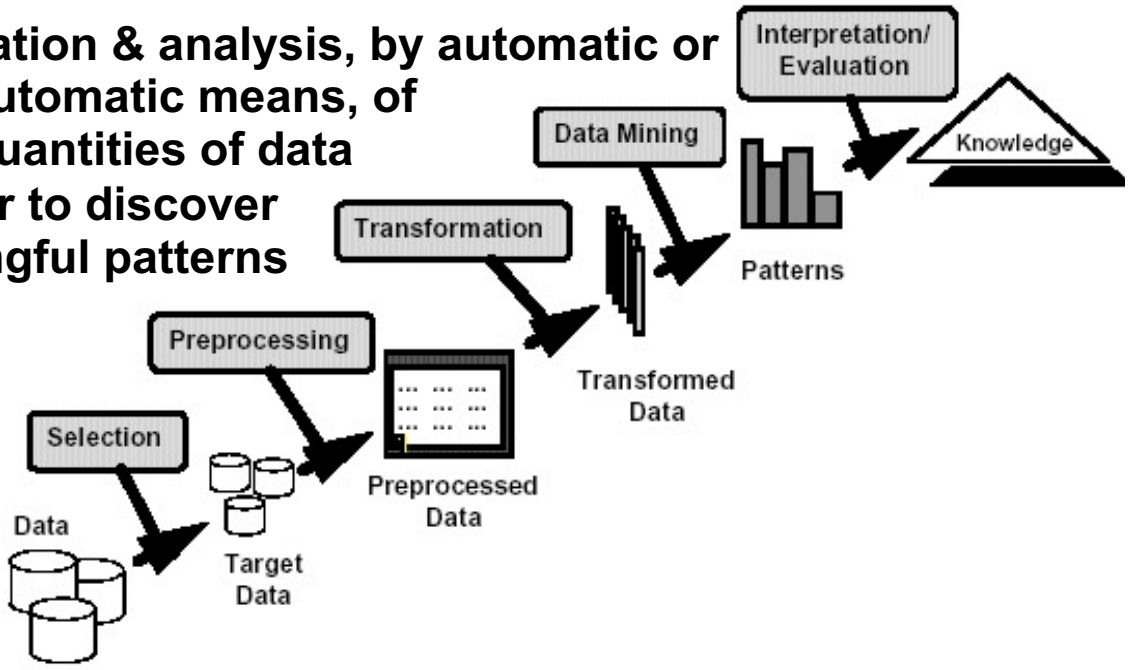
- **There is often information "hidden" in the data that is not readily evident**
- **Human analysts may take weeks to discover useful information**
- **Much of the data is never analyzed at all**



From: R. Grossman, C. Kamath, V. Kumar, "Data Mining for Scientific and Engineering Applications"

What is Data Mining?

- **Many Definitions**
  - **Non-trivial extraction of implicit, previously unknown and potentially useful information from data**
  - **Exploration & analysis, by automatic or semi-automatic means, of large quantities of data in order to discover meaningful patterns**



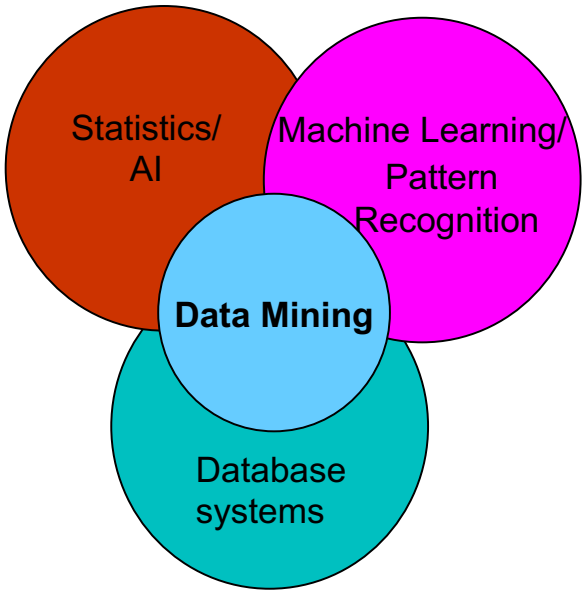
What is (not) Data Mining?

- **What is not Data Mining?**
  - Look up phone number in phone directory
  - Simple querying; For example, a database query “SELECT * FROM table” is just a database query and it displays information from the table but actually, this is not hidden information. So it is a simple query and not data mining.

- **What is Data Mining?**
  - Certain names are more prevalent in certain US locations (O’Brien, O’Rurke, O’Reilly… in Boston area)
  - Group together similar documents returned by search engine according to their context

Origins of Data Mining

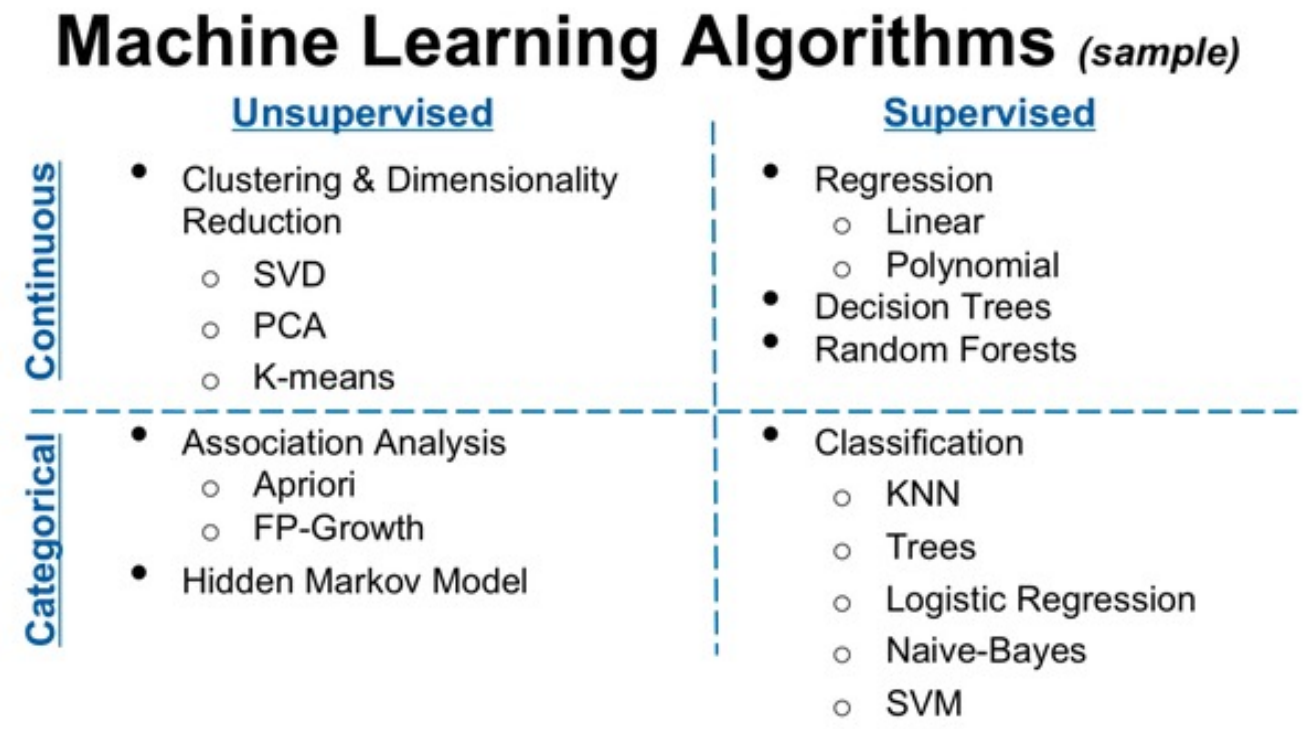
- **Draws ideas from machine learning/AI, pattern recognition, statistics, and database systems**
- **Traditional Techniques may be unsuitable due to**
  - **Enormity of data**
  - **High dimensionality of data**
  - **Heterogeneous, distributed nature of data**



Data Mining Tasks

- Prediction Methods
  - Use some variables to predict unknown or future values of other variables.
- Description Methods
  - Find human-interpretable patterns that describe the data.

From [Fayyad, et.al.] Advances in Knowledge Discovery and Data Mining, 1996



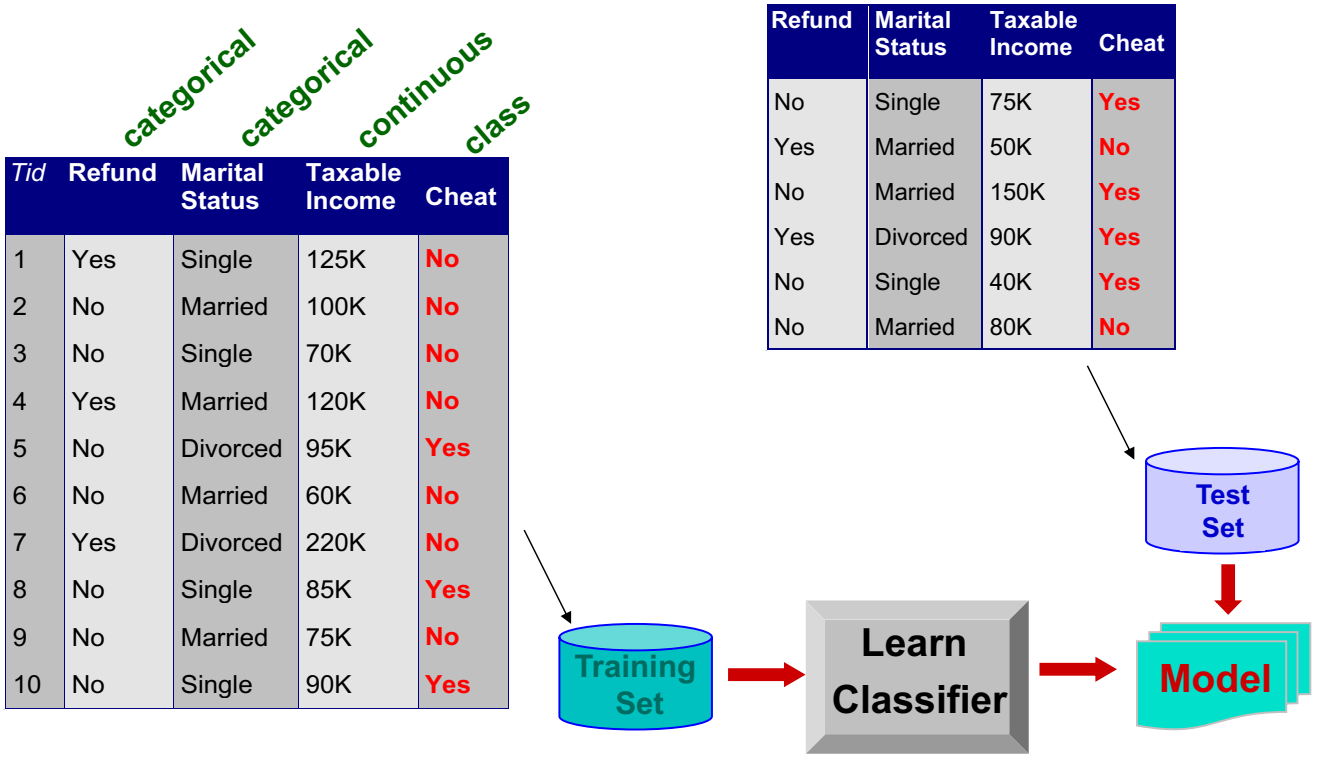
Data Mining Tasks...

- Classification [Predictive]
- Regression [Predictive]
- Clustering [Descriptive]
- Association Rule Discovery [Descriptive]
- Deviation/Anomaly Detection [Predictive]

Classification: Definition

- Given a collection of records (*training set*)
  - Each record contains a set of *attributes*, one of the attributes is the *class*.
- Find a *model* for class attribute as a function of the values of other attributes.
- Goal: previously unseen records should be assigned a class as accurately as possible.
  - A *test set* is used to determine the accuracy of the model. Usually, the given data set is divided into training and test sets, with training set used to build the model and test set used to validate it.

Classification Example



New Data:: No, Married, 60k, ?

Classification: Application 1

- Direct Marketing
  - Goal: Reduce cost of mailing by *targeting* a set of consumers likely to buy a new cell-phone product.
  - Approach:
    - Use the data for a similar product introduced before.
    - We know which customers decided to buy and which decided otherwise. This *{buy, don't buy}* decision forms the *class attribute*.
    - Collect various demographic, lifestyle, and company-interaction related information about all such customers.
      - Type of business, where they stay, how much they earn, etc.
    - Use this information as input attributes to learn a classifier model.

From [Berry & Linoff] Data Mining Techniques, 1997

Classification: Application 2

- Fraud Detection
  - Goal: Predict fraudulent cases in credit card transactions.
  - Approach:
    - Use credit card transactions and the information on its account-holder as attributes.
      - When does a customer buy, what does he buy, how often he pays on time, etc
    - Label past transactions as fraud or fair transactions. This forms the class attribute.
    - Learn a model for the class of the transactions.
    - Use this model to detect fraud by observing credit card transactions on an account.

Classification: Application 3

- Customer Attrition/Churn:
  - Goal: To predict whether a customer is likely to be lost to a competitor.
  - Approach:
    - Use detailed record of transactions with each of the past and present customers, to find attributes.
      - How often the customer calls, where he calls, what time-of-the day he calls most, his financial status, marital status, etc.
    - Label the customers as loyal or disloyal.
    - Find a model for loyalty.

From [Berry & Linoff] Data Mining Techniques, 1997

Classification: Application 4

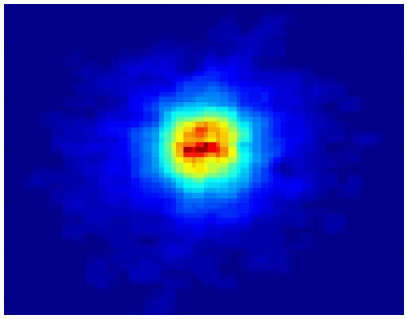
- Sky Survey Cataloging
  - Goal: To predict class (star or galaxy) of sky objects, especially visually faint ones, based on the telescopic survey images (from Palomar Observatory).
    - 3000 images with 23,040 x 23,040 pixels per image.
  - Approach:
    - Segment the image.
    - Measure image attributes (features) - 40 of them per object.
    - Model the class based on these features.
    - Success Story: Could find 16 new high red-shift quasars, some of the farthest objects that are difficult to find!

From [Fayyad, et.al.] Advances in Knowledge Discovery and Data Mining, 1996

Classifying Galaxies

Courtesy: http://aps.umn.edu

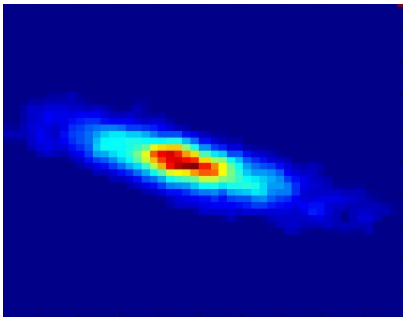
Early



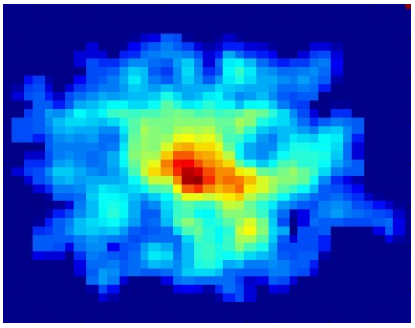
Class:
- **Stages of Formation**

Attributes:
- **Image features,**
- **Characteristics of light waves received, etc.**

Intermediate



Late



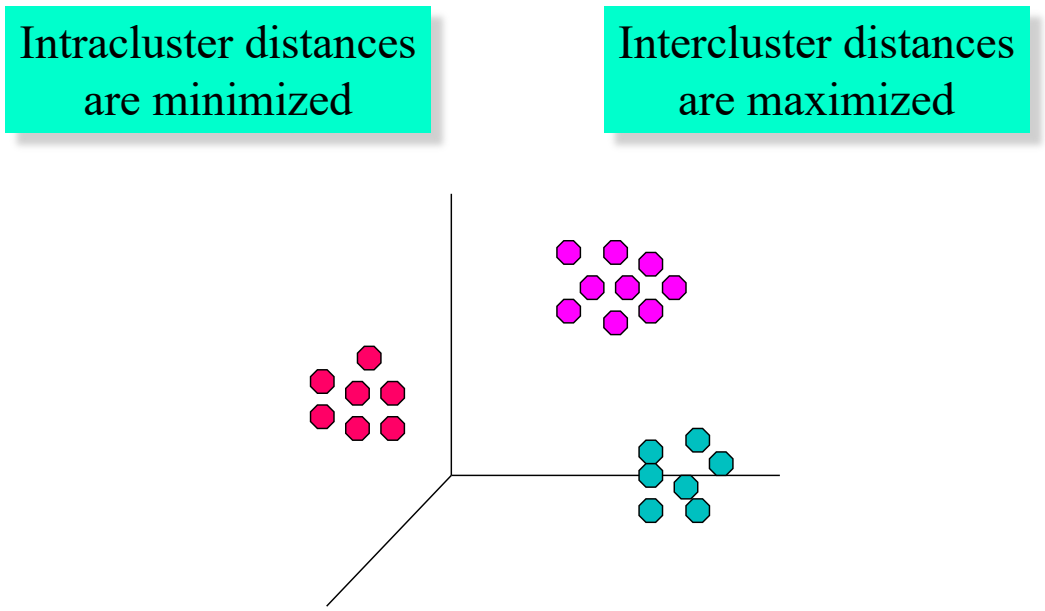
Data Size:
- **72 million stars, 20 million galaxies**
- **Object Catalog: 9 GB**
- **Image Database: 150 GB**

Clustering Definition

- Given a set of data points, each having a set of attributes, and a similarity measure among them, find clusters such that
  - Data points in one cluster are more similar to one another.
  - Data points in separate clusters are less similar to one another.
- Similarity Measures:
  - Euclidean Distance if attributes are continuous.
  - Other Problem-specific Measures.

Illustrating Clustering

- Euclidean Distance Based Clustering in 3-D space.



Clustering: Application 1

- Market Segmentation:
  - Goal: subdivide a market into distinct subsets of customers where any subset may conceivably be selected as a market target to be reached with a distinct marketing mix.
  - Approach:
    - Collect different attributes of customers based on their geographical and lifestyle related information.
    - Find clusters of similar customers.
    - Measure the clustering quality by observing buying patterns of customers in same cluster vs. those from different clusters.

Clustering: Application 2

- Document Clustering:
  - Goal: To find groups of documents that are similar to each other based on the important terms appearing in them.
  - Approach: To identify frequently occurring terms in each document. Form a similarity measure based on the frequencies of different terms. Use it to cluster.
  - Gain: Information Retrieval can utilize the clusters to relate a new document or search term to clustered documents.

Illustrating Document Clustering

- Clustering Points: 3204 Articles of Los Angeles Times.
- Similarity Measure: How many words are common in these documents (after some word filtering).

Category	*Total Articles*	*Correctly Placed*
Financial	555	364
Foreign	341	260
National	273	36
Metro	943	746
Sports	738	573
Entertainment	354	278

Association Rule Discovery: Definition

- Given a set of records each of which contain some number of items from a given collection;
  - Produce dependency rules which will predict occurrence of an item based on occurrences of other items.

TID	*Items*
1	Bread, Coke, Milk
2	Beer, Bread
3	Beer, Coke, Diaper, Milk
4	Beer, Bread, Diaper, Milk
5	Coke, Diaper, Milk

Rules Discovered:
{Milk} --> {Coke}
{Diaper, Milk} --> {Beer}

Association Rule Discovery: Application 1

- Marketing and Sales Promotion:
  - Let the rule discovered be

    {Bagels, … } --> {Potato Chips}
  - Potato Chips as consequent => Can be used to determine what should be done to boost its sales.
  - Bagels in the antecedent => Can be used to see which products would be affected if the store discontinues selling bagels.
  - Bagels in antecedent *and* Potato chips in consequent => Can be used to see what products should be sold with Bagels to promote sale of Potato chips!

Association Rule Discovery: Application 2

- Supermarket shelf management.
  - Goal: To identify items that are bought together by sufficiently many customers.
  - Approach: Process the point-of-sale data collected with barcode scanners to find dependencies among items.
  - A classic rule --
    - If a customer buys diaper and milk, then he is very likely to buy beer.
    - So, don't be surprised if you find beer stacked next to diapers!

Association Rule Discovery: Application 3

- Inventory Management:
  - Goal: A consumer appliance repair company wants to anticipate the nature of repairs on its consumer products and keep the service vehicles equipped with right parts to reduce on number of visits to consumer households.
  - Approach: Process the data on tools and parts required in previous repairs at different consumer locations and discover the co-occurrence patterns.

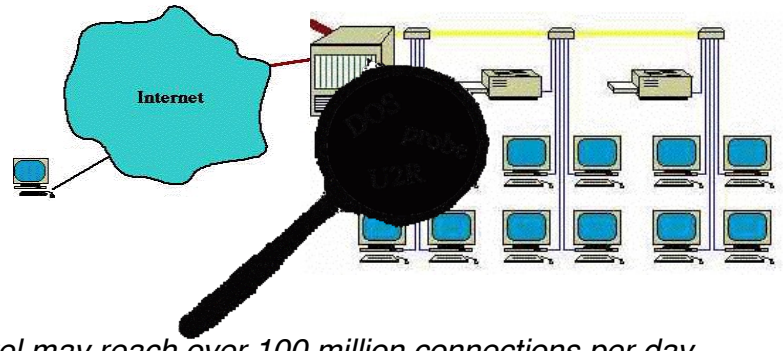
Regression

- Predict a value of a given continuous valued variable based on the values of other variables, assuming a linear or nonlinear model of dependency.
- Greatly studied in statistics, neural network fields.
- Examples:
  - Predicting sales amounts of new product based on advetising expenditure.
  - Predicting wind velocities as a function of temperature, humidity, air pressure, etc.
  - Time series prediction of stock market indices.

Deviation/Anomaly Detection

- Detect significant deviations from normal behavior
- Applications:
  - Credit Card Fraud Detection

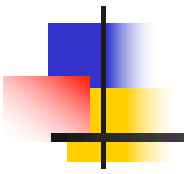
  - Network Intrusion Detection



Typical network traffic at University level may reach over 100 million connections per day

Challenges of Data Mining

- Scalability
- Dimensionality
- Complex and Heterogeneous Data
- Data Quality
- Data Ownership and Distribution
- Privacy Preservation
- Streaming Data

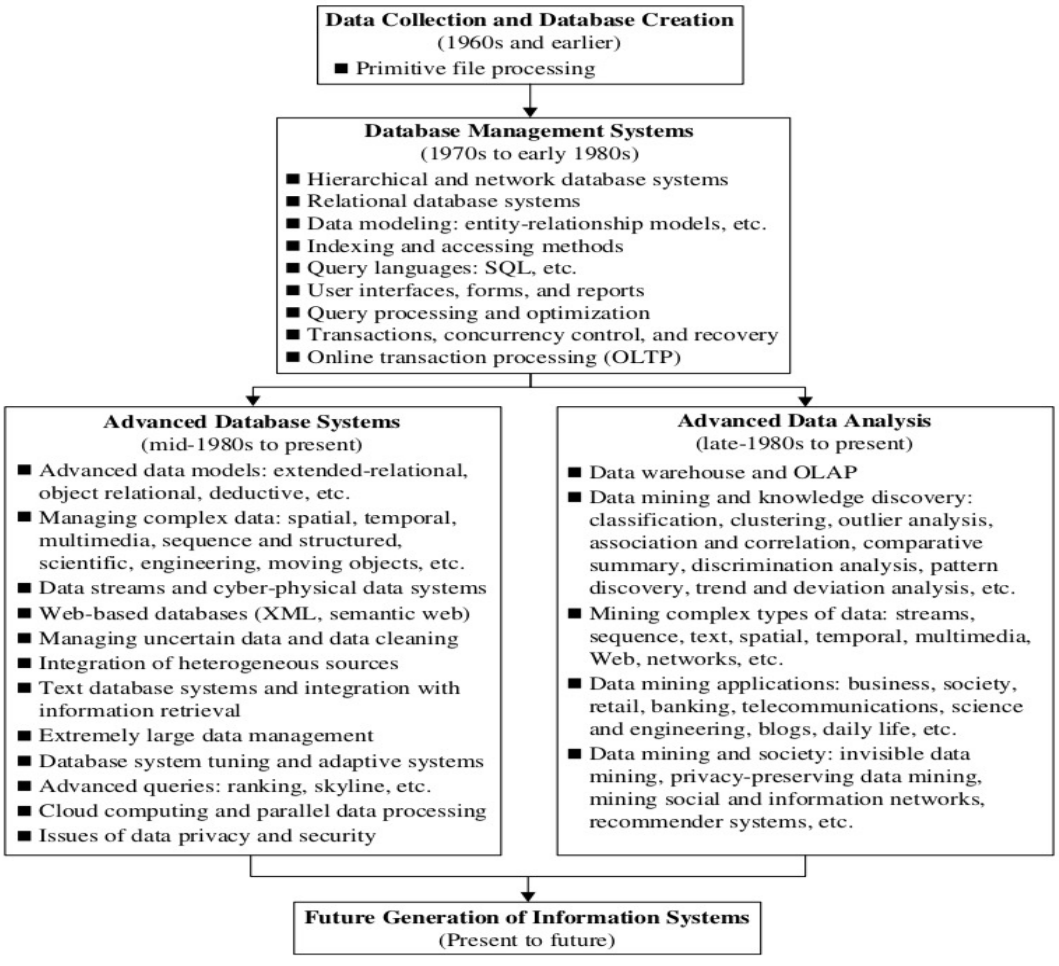


Introduction

1.1 Data Mining Origin
1.2 Data Mining & Data Warehousing basics

Why Data Mining?

- The Explosive Growth of Data: from terabytes to petabytes
  - Data collection and data availability
    - Automated data collection tools, database systems, Web, computerized society
  - Major sources of abundant data
    - Business: Web, e-commerce, transactions, stocks, …
    - Science: Remote sensing, bioinformatics, scientific simulation, …
    - Society and everyone: news, digital cameras, YouTube
- We are drowning in data, but starving for knowledge!
- "Necessity is the mother of invention"—Data mining—Automated analysis of massive data sets



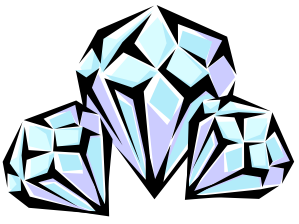
The evolution of database system technology.

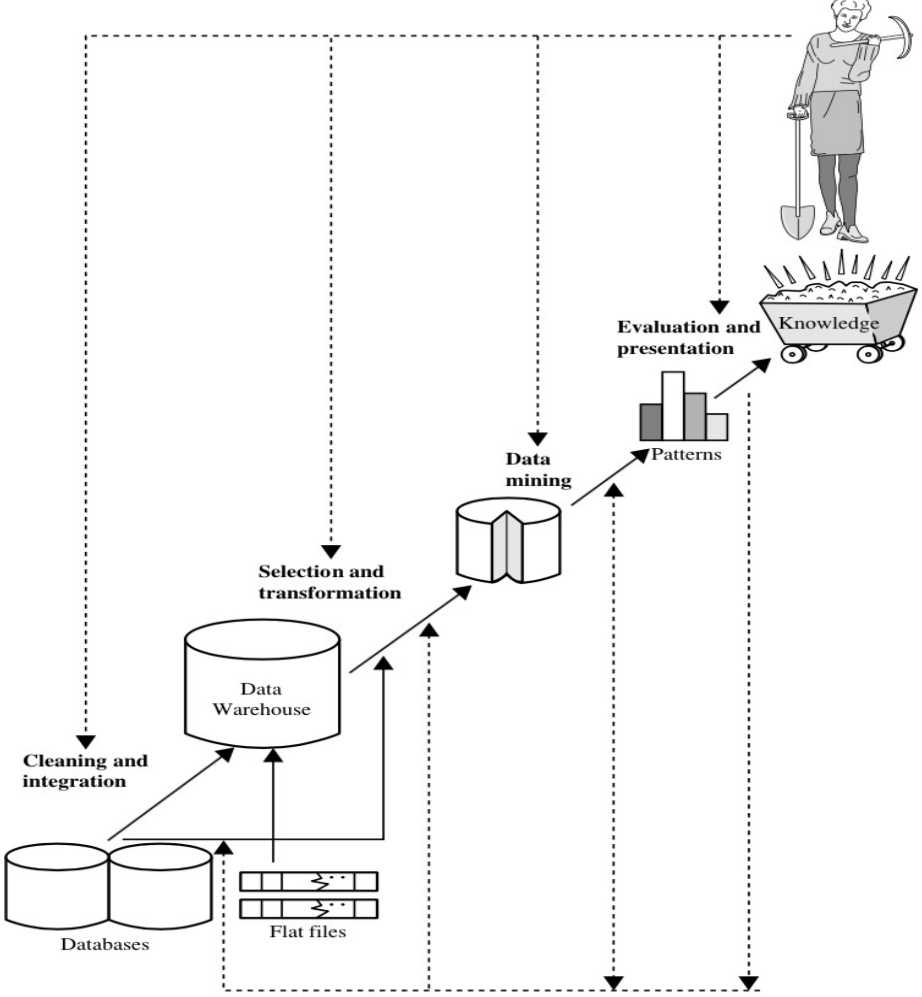
Evolution of Database Technology

- 1960s:
  - Data collection, database creation, IMS and network DBMS
- 1970s:
  - Relational data model, relational DBMS implementation
- 1980s:
  - RDBMS, advanced data models (extended-relational, OO, deductive, etc.)
  - Application-oriented DBMS (spatial, scientific, engineering, etc.)
- 1990s:
  - Data mining, data warehousing, multimedia databases, and Web databases
- 2000s
  - Stream data management and mining
  - Data mining and its applications
  - Web technology (XML, data integration) and global information systems

What Is Data Mining?

- Data mining (knowledge discovery from data)
  - Extraction of interesting (non-trivial, implicit, previously unknown and potentially useful) patterns or knowledge from huge amount of data
  - Data mining: a misnomer?
- Alternative names
  - Knowledge discovery (mining) in databases (KDD), knowledge extraction, data/pattern analysis, data archeology, data dredging, information harvesting, business intelligence, etc.
- Watch out: Is everything "data mining"?
  - Simple search and query processing
  - (Deductive) expert systems

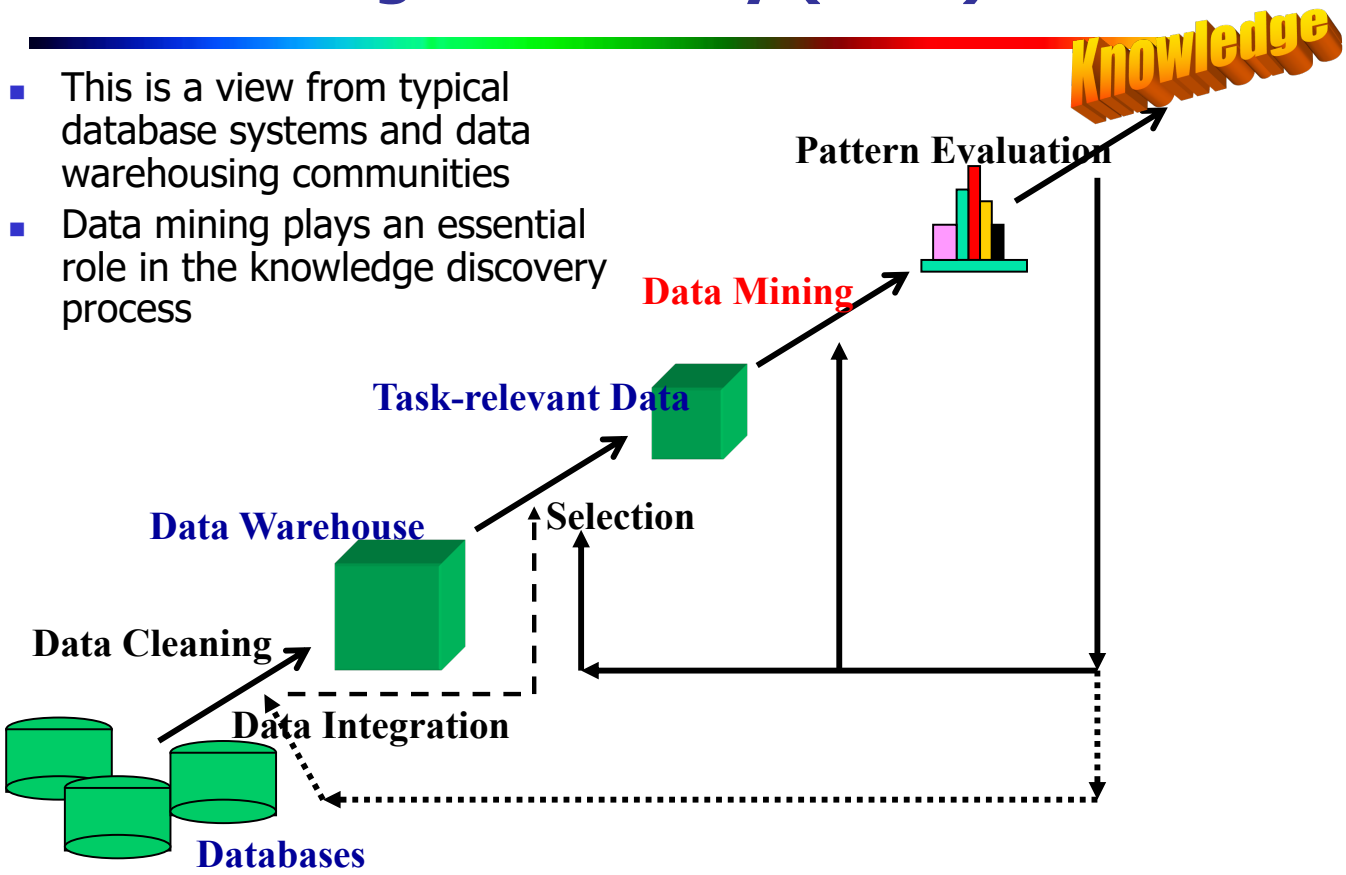




Data mining as a step in the process of knowledge discovery.

Knowledge Discovery (KDD) Process

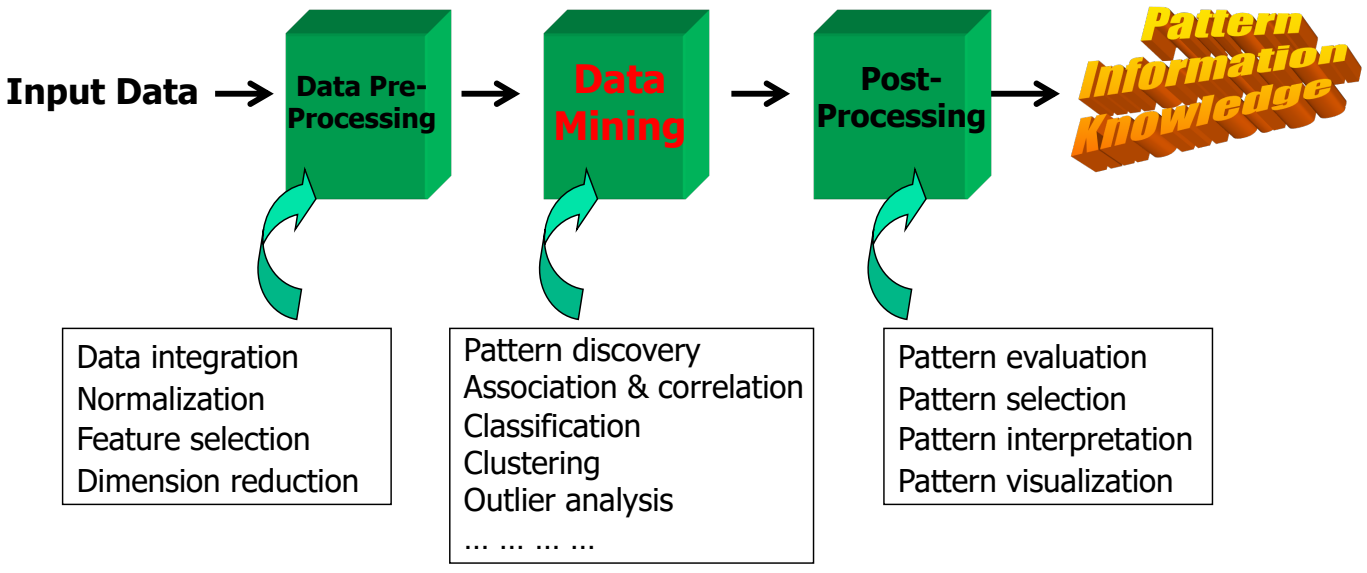
- This is a view from typical database systems and data warehousing communities
- Data mining plays an essential role in the knowledge discovery process



Example: A Web Mining Framework

- Web mining usually involves
  - Data cleaning
  - Data integration from multiple sources
  - Warehousing the data
  - Data cube construction
  - Data selection for data mining
  - Data mining
  - Presentation of the mining results
  - Patterns and knowledge to be used or stored into knowledge-base

KDD Process: A Typical View from ML and Statistics



- This is a view from typical machine learning and statistics communities

Example: Medical Data Mining

- Health care & medical data mining – often adopted such a view in statistics and machine learning
- Preprocessing of the data (including feature extraction and dimension reduction)
- Classification or/and clustering processes
- Post-processing for presentation

Data Mining: On What Kinds of Data?

- Database-oriented data sets and applications
  - Relational database, data warehouse, transactional database
- Advanced data sets and advanced applications
  - Data streams and sensor data
  - Time-series data, temporal data, sequence data (incl. bio-sequences)
  - Structure data, graphs, social networks and multi-linked data
  - Object-relational databases
  - Heterogeneous databases and legacy databases
  - Spatial data and spatiotemporal data
  - Multimedia database
  - Text databases
  - The World-Wide Web

Data Mining Function:
1. Concept Description (class description): Characterization and Discrimination

- Information integration and data warehouse construction
  - Data cleaning, transformation, integration, and multidimensional data model
- Data cube technology
  - Scalable methods for computing (i.e., materializing) multidimensional aggregates
  - OLAP (online analytical processing)
- Multidimensional concept description: Characterization and discrimination
  - Generalize, summarize, and contrast data characteristics

Data Mining Function:
2. Association and Correlation Analysis

- Frequent patterns (or frequent itemsets)
  - What items are frequently purchased together in your Walmart?
- Association, correlation vs. causality
  - A typical association rule
    - Diaper → Beer [0.5%, 75%] (support, confidence)
  - Are strongly associated items also strongly correlated?
- How to mine such patterns and rules efficiently in large datasets?
- How to use such patterns for classification, clustering, and other applications?

Data Mining Function: 3. Classification

- Classification and label prediction
  - Construct models (functions) based on some training examples
  - Describe and distinguish classes or concepts for future prediction
    - E.g., classify countries based on (climate), or classify cars based on (gas mileage)
  - Predict some unknown class labels
- Typical methods
  - Decision trees, naïve Bayesian classification, support vector machines, neural networks, rule-based classification, pattern-based classification, logistic regression, …
- Typical applications:
  - Credit card fraud detection, direct marketing, classifying stars, diseases, web-pages, …

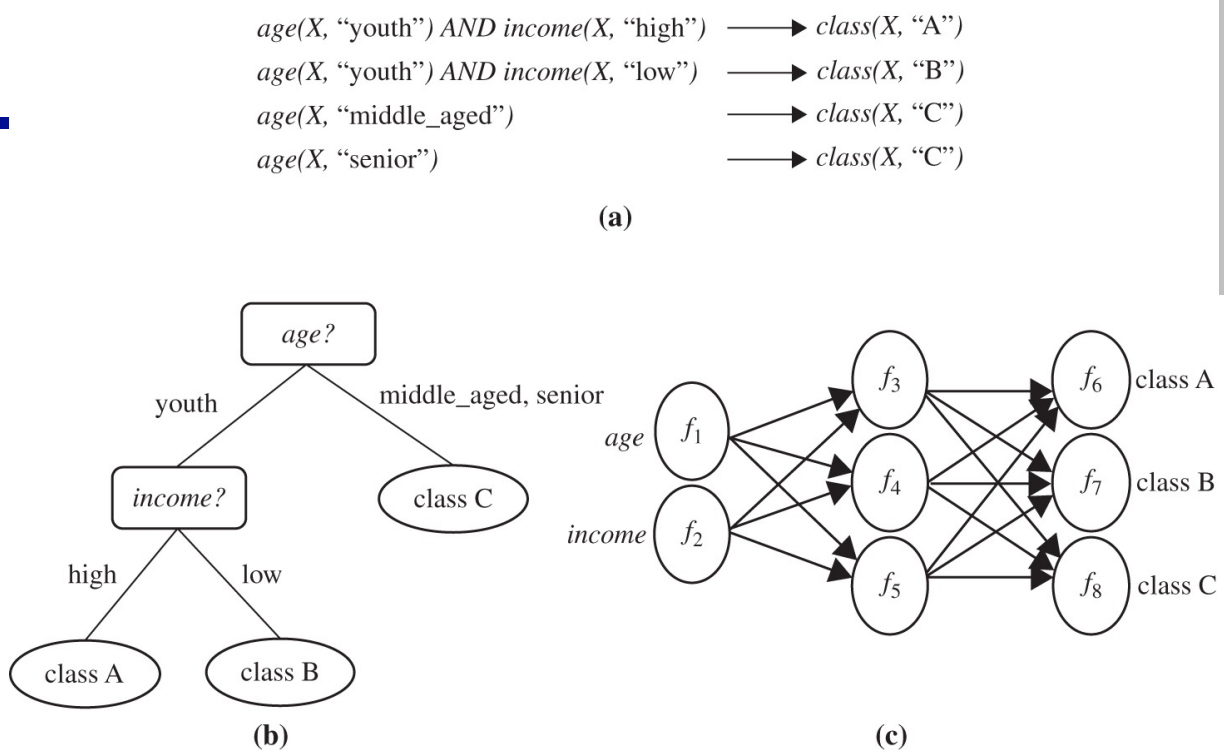


Figure 1.9 A classification model can be represented in various forms: (a) IF-THEN rules, (b) a decision tree, or (c) a neural network.

Data Mining Function: 4. Cluster Analysis

- Unsupervised learning (i.e., Class label is unknown)
- Group data to form new categories (i.e., clusters), e.g., cluster houses to find distribution patterns
- Principle: Maximizing intra-class similarity & minimizing interclass similarity
- Many methods and applications

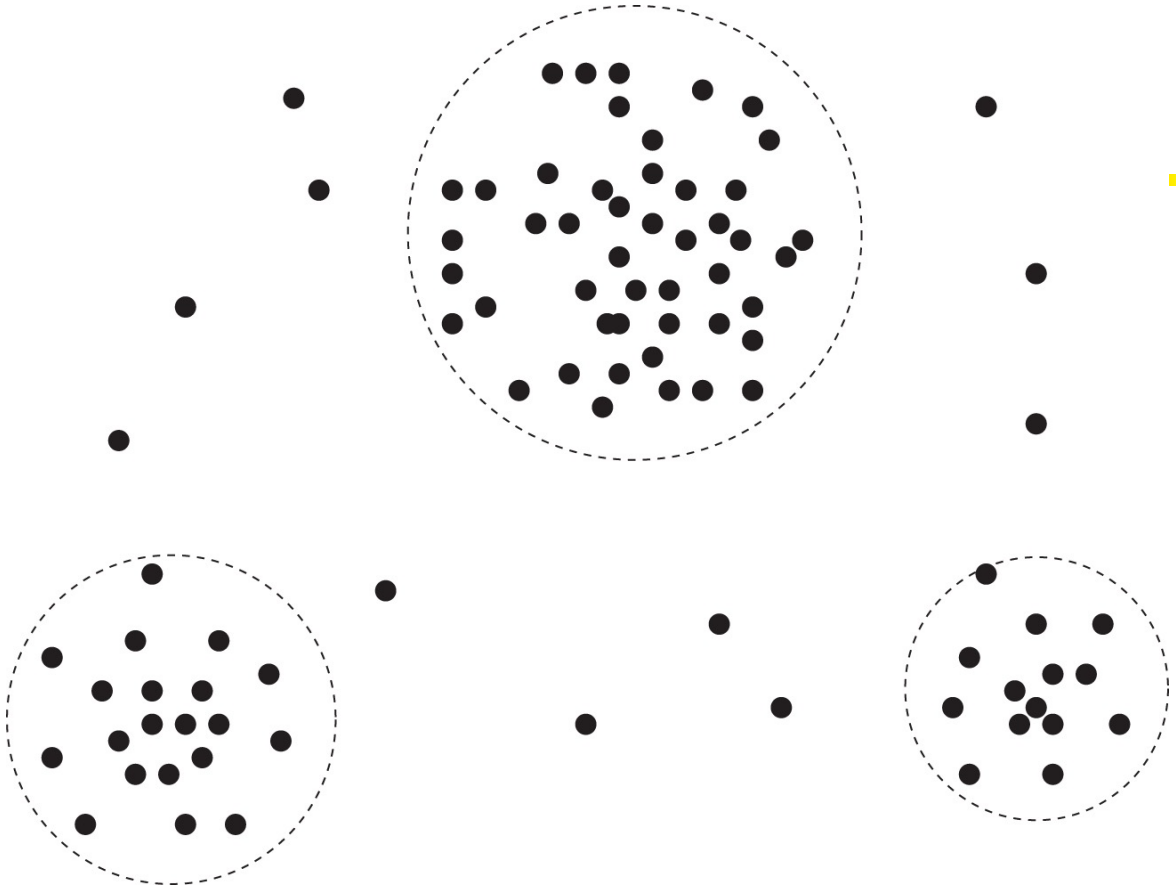


Figure 1.10 A 2-D plot of customer data with respect to customer locations in a city, showing three data clusters.

Data Mining Function: 5. Outlier Analysis

- Outlier analysis
  - Outlier: A data object that does not comply with the general behavior of the data
  - Noise or exception? — One person's garbage could be another person's treasure
  - Methods: by product of clustering or regression analysis, …
  - Useful in fraud detection, rare events analysis

What is Data Mining?

- "The process of discovering meaningful patterns and trends often previously unknown by using some mathematical algorithm on huge amount of stored data"
- "Extraction of interesting, non-trivial, implicit, previously unknown and potentially useful information or patterns from data in large database."
- Data mining is basically concerned with the analysis of data and the use of software techniques for finding patterns and regularities in sets of data.

Data Mining Tasks

- Prediction Methods
  - Use some variables to predict unknown or future values of other variables.
  - Typical questions answered by predictive models are:
    - Who is likely to respond to next product?
    - Which customers are likely to leave in the next six months?
- Description Methods
  - Find human-interpretable patterns that describe the data.
  - Typical questions answered by descriptive data mining are:
    - What is in the data?
    - What doesn't look like?
    - Are there any unusual patterns?
    - What does the data suggest for customer segmentation?

- Functionalities of **descriptive data mining** are: Clustering, Summarization, Visualization, and Association.

- **Predictive data mining** (Eg: Classification, Regression) **are** never 100% accurate. The performance of a model on past data is not predicting the known outcomes.
  Suitable for unknown data set.

Data Mining Vs. Query Tools

- i. **SQL** can find normal queries from the database such as **what is an average turnover?** Whereas **data minin**g tools find interesting patterns and facts such as **what are the important trends in sells?**
- ii. **Data mining** is much more **faster** than **SQL in trend and pattern analysis** since it uses algorithm like machine learning, genetic algorithm.
- iii. If **we know exactly what we are looking for**, we **use SQL** not **if we know only vaguely** what we are looking for we **use data mining**.
- iv. **Hybrid information can't be easily** be **traced using SQL**.

Applications of Data Mining

- E-Commerce
- Bank Loan Decision
- Retail
- Digital Payment
- Web page analysis: from web page classification, clustering to PageRank & HITS algorithms
- Collaborative analysis & recommender systems
- Basket data analysis to targeted marketing
- Biological and medical data analysis: classification, cluster analysis (microarray data analysis), biological sequence analysis, biological network analysis
- From major dedicated data mining systems/tools (e.g., SAS, MS SQL-Server Analysis Manager, Oracle Data Mining Tools) to invisible data mining

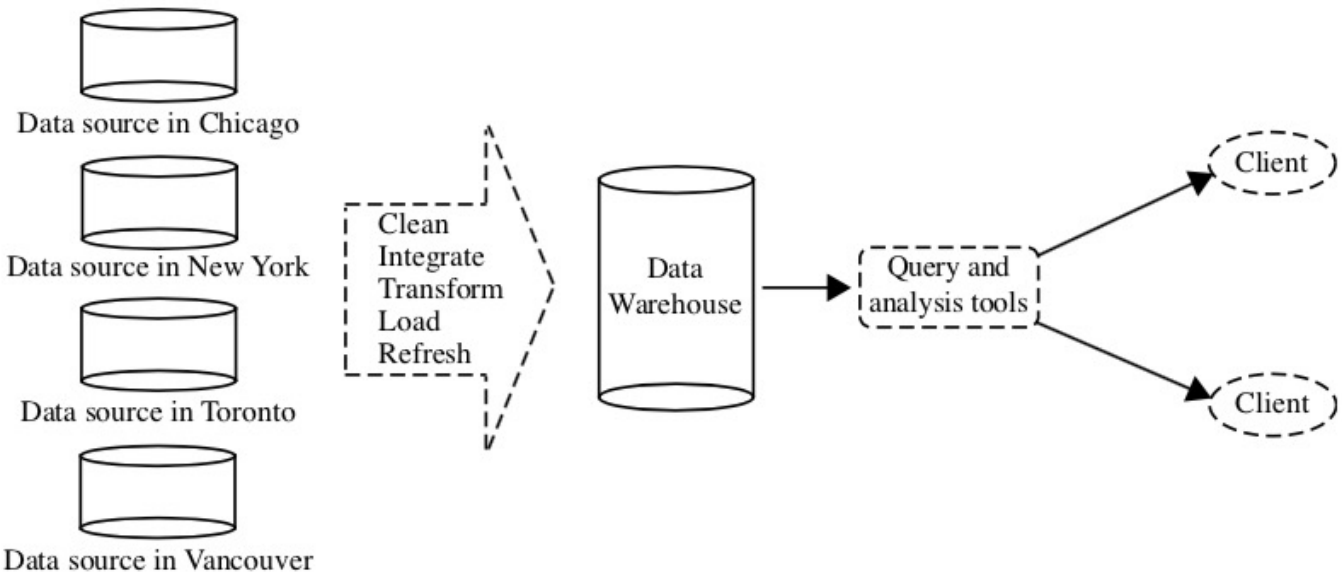
A Brief History of Data Mining Society

- 1989 IJCAI Workshop on Knowledge Discovery in Databases
  - Knowledge Discovery in Databases (G. Piatetsky-Shapiro and W. Frawley, 1991)
- 1991-1994 Workshops on Knowledge Discovery in Databases
  - Advances in Knowledge Discovery and Data Mining (U. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, 1996)
- 1995-1998 International Conferences on Knowledge Discovery in Databases and Data Mining (KDD'95-98)
  - Journal of Data Mining and Knowledge Discovery (1997)
- ACM SIGKDD conferences since 1998 and SIGKDD Explorations
- More conferences on data mining
  - PAKDD (1997), PKDD (1997), SIAM-Data Mining (2001), (IEEE) ICDM (2001), etc.
- ACM Transactions on KDD starting in 2007

Summary

- Data mining: Discovering interesting patterns and knowledge from massive amount of data
- A natural evolution of database technology, in great demand, with wide applications
- A KDD process includes data cleaning, data integration, data selection, transformation, data mining, pattern evaluation, and knowledge presentation
- Mining can be performed in a variety of data
- Data mining functionalities: characterization, discrimination, association, classification, clustering, outlier and trend analysis, etc.
- Data mining technologies and applications
- Major issues in data mining

What is a Data Warehouse?



Typical framework of a data warehouse for *AllElectronics*.

What is a Data Warehouse?

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

- 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*

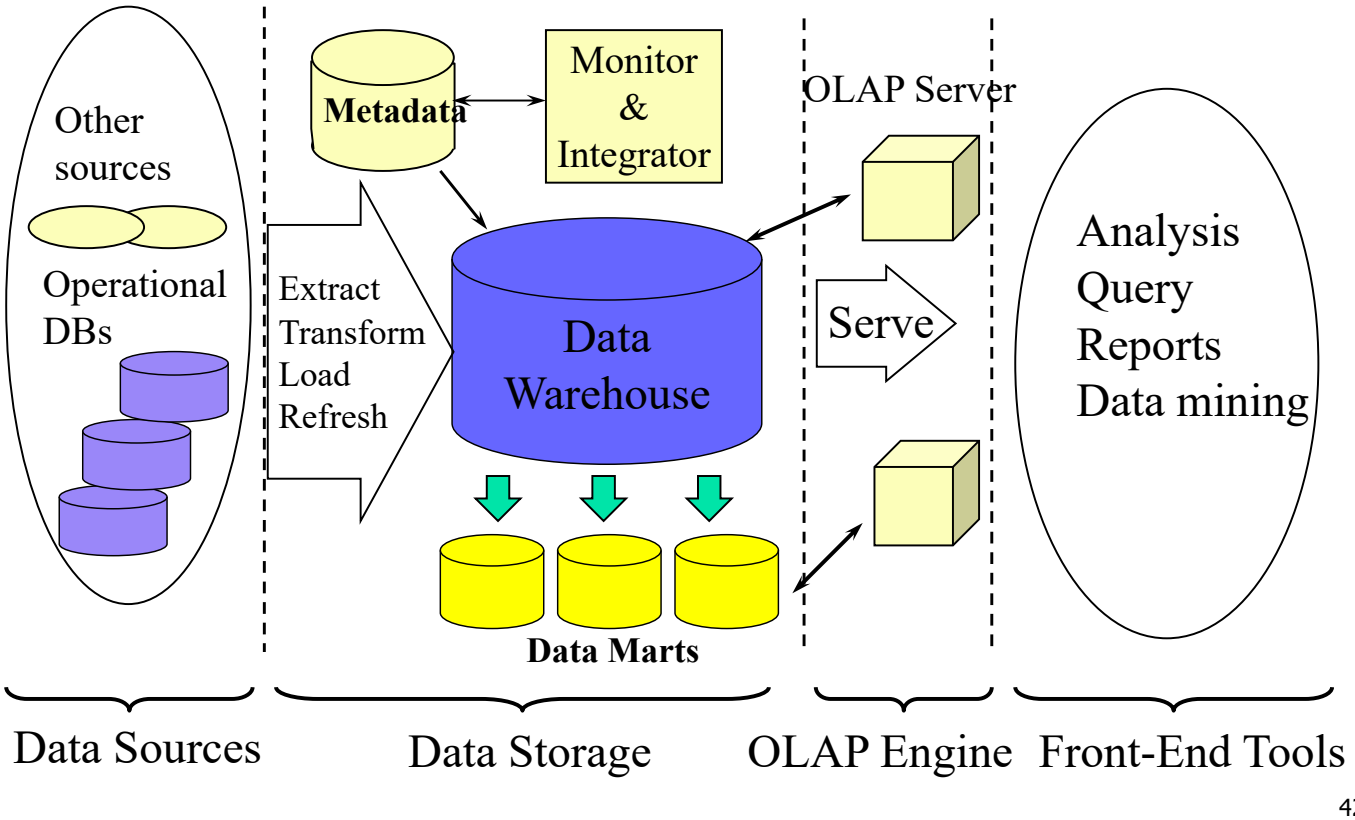
OLTP vs. OLAP

	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 a 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
- 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

Data Warehouse: A Multi-Tiered Architecture



Three Data Warehouse Models

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

Data Mart

- Data Mart is a subset of the information content of a data warehouse that is stored in its own database.
- Data mart may or may not be sourced from an enterprise data warehouse i.e. it could have been directly populated from source data.
- Data mart can improve query performance simply by reducing the volume of data that needs to be scanned to satisfy the query.
- Data marts are created along functional level to reduce the likelihood of queries requiring data outside the mart.
- Data marts may help in multiple queries or tools to access data by creating their own internal database structures.
- Eg: Departmental Store, Banking System.

Metadata Repository

- **Meta data** is the data defining warehouse objects.

It stores:

- **Description of the structure of the data 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 defn, data ownership, charging policies

Extraction, Transformation, and Loading (ETL)

- **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 indicies and partitions
- **Refresh**
  - propagate the updates from the data sources to the warehouse

DATA WAREHOUSHING -- PROCESS MANAGERS

- Process managers are responsible for maintaining the flow of data both into and out of the data warehouse.
- There are three different types of process managers:
  - Load manager
  - Warehouse manager
  - Query manager

Load Manager

- Load manager performs the operations required to extract and load the data into the database. The size and complexity of a load manager varies between specific solutions from one data warehouse to another.
- Load Manager Architecture
  - The load manager does performs the following functions:
    - Extract data from the source system.
    - Fast load the extracted data into temporary data store.
    - Perform simple transformations into structure similar to the one in the data warehouse

Warehouse Manager

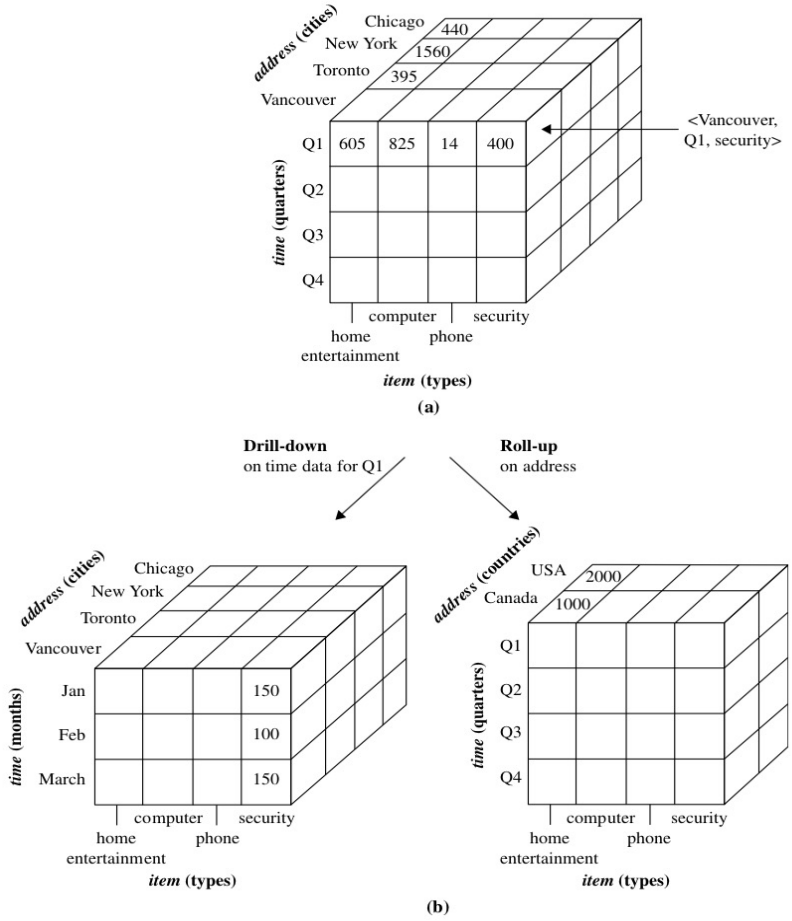
- The warehouse manager is responsible for the warehouse management process.
- It consists of a third-party system software, C programs, and shell scripts.
- The size and complexity of a warehouse manager varies between specific solutions.
- A warehouse manager includes the following:
  - The controlling process
  - Stored procedures or C with SQL
  - Backup/Recovery tool
  - SQL scripts

- **Functions of Warehouse Manager**
- Analyzes the data to perform consistency and referential integrity checks.
- Creates indexes, business views, partition views against the base data.
- Generates new aggregations and updates the existing aggregations.
- Generates normalizations.
- Transforms and merges the source data of the temporary store into the published data warehouse.
- Backs up the data in the data warehouse.
- Archives the data that has reached the end of its captured life.

Query Manager

- The query manager is responsible for directing the queries to suitable tables.
- By directing the queries to appropriate tables, it speeds up the query request and response process.
- In addition, the query manager is responsible for scheduling the execution of the queries posted by the user.
- A query manager includes the following components:
  - Query redirection via C tool or RDBMS
  - Stored procedures
  - Query management tool
  - Query scheduling via C tool or RDBMS
  - Query scheduling via third-party software

- **Functions of Query Manager:**
- It presents the data to the user in a form they understand.
- It schedules the execution of the queries posted by the end-user.
- It stores query profiles to allow the warehouse manager to determine which indexes and aggregations are appropriate.



A multidimensional data cube, commonly used for data warehousing, (a) showing summarized data for *AllElectronics* and (b) showing summarized data resulting from drill-down and roll-up operations on the cube in (a). For improved readability, only some of the cube cell values are shown.

2. Data Pre-Processing (6 hours)

- 2.1. Data Types and Attributes
- 2.2. Data Pre-processing
- 2.3. OLAP
- 2.4 Characteristics of OLAP Systems
- 2.5 Multidimensional View and Data cube
- 2.6 Data Cube Implementation
- 2.7 Data Cube Operations
- 2.8 Guidelines for OLAP Implementation

2.1 Data Types and Attributes

Types of Data Sets

- Record
  - Relational records
  - Data matrix, e.g., numerical matrix, crosstabs
  - Document data: text documents: term-frequency vector
  - Transaction data
- Graph and network
  - World Wide Web
  - Social or information networks
  - Molecular Structures

	team	coach	play	ball	score	game	win	lost	timeout	season
Document 1	3	0	5	0	2	6	0	2	0	2
Document 2	0	7	0	2	1	0	0	3	0	0
Document 3	0	1	0	0	1	2	2	0	3	0

Types of Data Sets

- Ordered
  - Video data: sequence of images
  - Temporal data: time-series
  - Sequential Data: transaction sequences
  - Genetic sequence data
- Spatial, image and multimedia:
  - Spatial data: maps
  - Image data:
  - Video data:

TID	*Items*
1	Bread, Coke, Milk
2	Beer, Bread
3	Beer, Coke, Diaper, Milk
4	Beer, Bread, Diaper, Milk
5	Coke, Diaper, Milk

Data Objects

- Data sets are made up of data objects.
- A **data object** represents an entity.
- Examples:
  - sales database: customers, store items, sales
  - medical database: patients, treatments
  - university database: students, professors, courses
- Also called *samples , examples, instances, data points, objects, tuples*.
- Data objects are described by **attributes**.
- Database rows -> data objects; columns ->attributes.

What is an Attribute?

- Entity/DataObjects/Instances/Examples/DataPoints will have certain **attributes**.
- An attribute is a property or characteristic of an object. Examples: eye color of a person, temperature, *customer_ID, name, address etc.*
- Attribute is also known as variable, field, characteristic, or feature
- A collection of attributes describe an object. Object is also known as record, point, case, sample, entity, or instance.
- Attribute values are numbers or symbols assigned to an attribute

What is an Attribute?

- Same attribute can be mapped to different attribute values. Example: height can be measured in feet or meters.
- Different attributes can be mapped to the same set of values. Example: Attribute values for ID and age are integers but properties of attribute values can be different. ID has no limit but age has a maximum and minimum value.
- Also Known as:
  - Features (Machine Learning)
  - Dimensions (Data Mining)
  - Variables (Statistcs)

Types of Attributes: Approach 1

- Nominal/categorical
- Binary
- Ordinal
- Numeric: quantitative
  - Interval-scaled
  - Ratio-scaled

Types of Attributes: Approach 1

- **Nominal/categorical:** categories, states, or "names of things"
  - Values assumed as linguistic and has no any order
  - *Eg: Hair_color = {auburn, black, blond, brown, grey, red, white}*
  - Eg: marital status, occupation, ID numbers, zip codes
  - Only operator = make sense
- **Binary:** Nominal attribute with only 2 states (0 and 1)
  - Symmetric binary: both outcomes equally important
    - e.g., gender
  - Asymmetric binary: outcomes not equally important.
    - e.g., medical test (positive vs. negative)
    - Convention: assign 1 to most important outcome (e.g., HIV positive)

Types of Attributes: Approach 1

- **Ordinal:** May be like nominal but with some orders semantically.
  - Values have a meaningful order (ranking) but magnitude between successive values is not known.
  - Eg: months, medals, ranks
  - *Eg: size = {small, medium, large},* grades, army rankings
  - Operator: = , < , >
- **Numeric:** Also known as continuous attributes because they have values like 2.5,0.32 etc
  - Two types: **Interval** and **Ratio**

Numeric

- Quantity (integer or real-valued)
- Operator: =, <, >, +, -, *, /
- **Interval :** (Like arithmetic series)
  - Measured on a scale of **equal-sized units**
  - Values have order
    - E.g., *temperature in C˚or F˚, calendar dates, height, profit*
  - No true zero-point
- **Ratio:** (Like geometric series)
  - Inherent **zero-point**
  - We can speak of values as being an order of magnitude larger than the unit of measurement (10 K˚ is twice as high as 5 K˚).
    - e.g., *temperature in Kelvin, length, counts, monetary quantities*

Types of Attributes: Approach 2

- **Discrete Attribute**
  - Has only a finite or countably infinite set of values
    - E.g., zip codes, profession, or the set of words in a collection of documents
  - Sometimes, represented as integer variables
  - Note: Binary attributes are a special case of discrete attributes
- **Continuous Attribute**
  - Has real numbers as attribute values
    - E.g., temperature, height, or weight
  - Practically, real values can only be measured and represented using a finite number of digits
  - Continuous attributes are typically represented as floating-point variables

Types of Attributes: Approach 3

- **Character**:
  - values are represented in forms of character or set of characters (string).
- **Number**:
  - values are represented in forms of number.
  - Number may be in form of whole number, decimal number.

2.2 Data Pre-processing

Data Quality: Why Preprocess the Data?

- Measures for data quality: A multidimensional view
  - Accuracy: correct or wrong, accurate or not
  - Completeness: not recorded, unavailable, …
  - Consistency: some modified but some not, dangling,.
  - Timeliness: timely update?
  - Believability: how trustable the data are correct?
  - Interpretability: how easily the data can be understood?

Major Tasks in Data Preprocessing

1. **Data cleaning**
   1. Fill in missing values, smooth noisy data, identify or remove outliers, and resolve inconsistencies
2. **Data integration**
   1. Integration of multiple databases, data cubes, or files
3. **Data reduction**
   1. Dimensionality reduction(Wavelet Transf^{n},PCA,Attr Subset Se
   2. Numerosity reduction (Parametric [Using model], Non-Parametric)
   3. Data compression
4. **Data transformation and data discretization**
   1. Normalization
   2. Concept hierarchy generation

I. Data Cleaning

- Data in the Real World Is Dirty: Lots of potentially incorrect data, e.g., instrument faulty, human or computer error, transmission error
- incomplete: lacking attribute values, lacking certain attributes of interest, or containing only aggregate data
  - e.g., *Occupation*=" " (missing data)
- noisy: containing noise, errors, or outliers
  - e.g., *Salary*="−10" (an error)
- inconsistent: containing discrepancies in codes or names, e.g.,
  - *Age*="42", *Birthday*="03/07/2010"
  - Was rating "1, 2, 3", now rating "A, B, C"
- Intentional (e.g., *disguised missing* data)
  - Jan. 1 as everyone's birthday?

Incomplete (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 (when doing classification)—not effective when the % of missing values per attribute varies considerably
- Fill in the missing value manually: tedious + infeasible?
- Fill in it automatically with
  - a global constant : e.g., "unknown", a new class?!
  - the attribute mean
  - the attribute mean for all samples belonging to the same class: smarter
  - the most probable 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 be due to
  - faulty data collection instruments
  - data entry problems
  - data transmission problems
  - technology limitation
  - inconsistency in naming convention
- Other data problems which require data cleaning
  - duplicate records
  - incomplete data
  - inconsistent data

How to Handle Noisy Data?

- Binning
  - first sort data and partition into (equal-frequency) bins
  - then one can smooth by bin means, smooth by bin median, smooth by bin boundaries, etc.
- Regression
  - smooth by fitting the data into regression functions
- Clustering
  - detect and remove outliers
- Combined computer and human inspection
  - detect suspicious values and check by human (e.g., deal with possible outliers)

2. Data Integration

- **Data integration**:
  - Combines data from multiple sources into a coherent store
- Schema integration: e.g., A.cust-id ≡ B.cust-#
  - Integrate metadata from different sources
- Entity identification problem:
  - Identify real world entities from multiple data sources, e.g., Bill Clinton = William Clinton
- 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

Correlation Analysis (Nominal Data)

- **X^2 (chi-square) test**

$$\chi^2 = \sum \frac{(Observed - Expected)^2}{Expected}$$

- The larger the X^2 value, the more likely the variables are related
- The cells that contribute the most to the X^2 value are those whose actual count is very different from the expected count
- **Correlation does not imply causality**
  - # of hospitals and # of car-theft in a city are correlated
  - Both are causally linked to the third variable: population

Chi-Square Calculation: An Example

	Play chess	Not play chess	Sum (row)
Like science fiction	250(90)	200(360)	450
Not like science fiction	50(210)	1000(840)	1050
Sum(col.)	300	1200	1500

- X^2 (chi-square) calculation (numbers in parenthesis are expected counts calculated based on the data distribution in the two categories)

$$\chi^2 = \frac{(250-90)^2}{90} + \frac{(50-210)^2}{210} + \frac{(200-360)^2}{360} + \frac{(1000-840)^2}{840} = 507.93$$

- It shows that like_science_fiction and play_chess are correlated in the group

3. Data Reduction Strategies

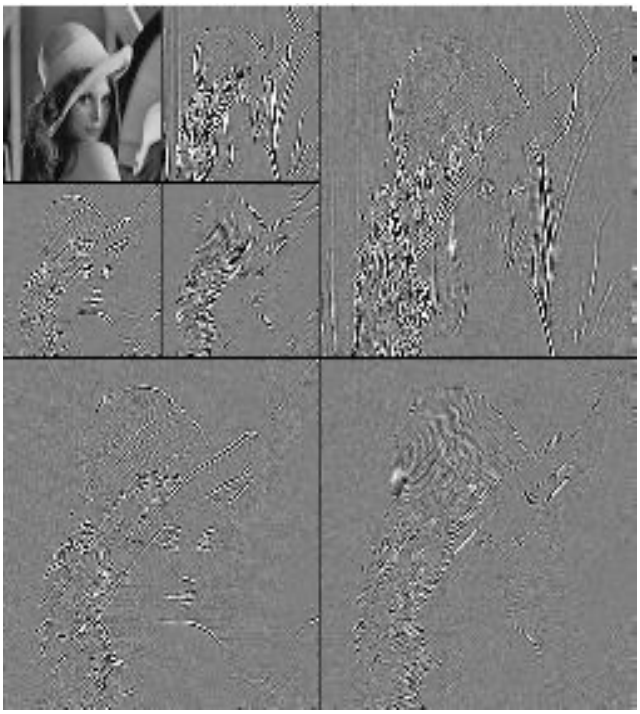
- **Data reduction**: Obtain a reduced representation of the data set that is much smaller in volume but yet produces the same (or almost the same) analytical results
- Why data reduction? — A database/data warehouse may store terabytes of data. Complex data analysis may take a very long time to run on the complete data set.
- Data reduction strategies
  a) Dimensionality reduction, e.g., remove unimportant attributes
     - Wavelet transforms
     - Principal Components Analysis (PCA)
     - Feature subset selection, feature creation
  b) Numerosity reduction (some simply call it: Data Reduction)
     - Parametric (Using model): Regression and Log-Linear Models
     - Non-Parametirc: Histograms, clustering, sampling
  c) Data compression

Data Reduction :a) Dimensionality Reduction

- **Curse/harm of dimensionality**
  - When dimensionality increases, data becomes increasingly sparse
  - Density and distance between points, which is critical to clustering, outlier analysis, becomes less meaningful
  - The possible combinations of subspaces will grow exponentially
- **Dimensionality reduction**
  - Avoid the curse of dimensionality
  - Help eliminate irrelevant features and reduce noise
  - Reduce time and space required in data mining
  - Allow easier visualization
- **Dimensionality reduction techniques**
  - Wavelet transforms
  - Principal Component Analysis
  - Supervised and nonlinear techniques (e.g., feature selection)

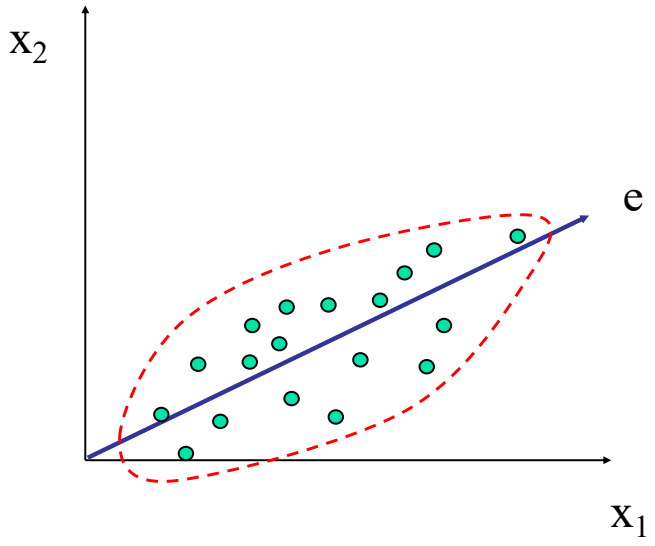
What Is Wavelet Transform?

- Decomposes a signal into different frequency subbands
  - Applicable to n-dimensional signals
- Data are transformed to preserve relative distance between objects at different levels of resolution
- Allow natural clusters to become more distinguishable
- Used for image compression



Principal Component Analysis (PCA)

- Find a projection that captures the largest amount of variation in data
- The original data are projected onto a much smaller space, resulting in dimensionality reduction. We find the eigenvectors of the covariance matrix, and these eigenvectors define the new space



Principal Component Analysis (Steps)

- Given *N* data vectors from *n*-dimensions, find $k \leq n$ orthogonal vectors (*principal components*) that can be best used to represent data
  - Normalize input data: Each attribute falls within the same range
  - Compute *k* orthonormal (unit) vectors, i.e., *principal components*
  - Each input data (vector) is a linear combination of the *k* principal component vectors
  - The principal components are sorted in order of decreasing "significance" or strength
  - Since the components are sorted, the size of the data can be reduced by eliminating the *weak components*, i.e., those with low variance (i.e., using the strongest principal components, it is possible to reconstruct a good approximation of the original data)
- Works for numeric data only

Attribute Subset Selection

- Another way to reduce dimensionality of data
- Redundant attributes
  - Duplicate much or all of the information contained in one or more other attributes
  - E.g., purchase price of a product and the amount of sales tax paid
- Irrelevant attributes
  - Contain no information that is useful for the data mining task at hand
  - E.g., students' ID is often irrelevant to the task of predicting students' GPA

Attribute Creation (Feature Generation)

- Create new attributes (features) that can capture the important information in a data set more effectively than the original ones
- Three general methodologies
  - Attribute extraction
    - Domain-specific
  - Mapping data to new space (see: data reduction)
    - E.g., Fourier transformation, wavelet transformation, manifold approaches (not covered)
  - Attribute construction
    - Combining features (see: discriminative frequent patterns in Chapter 7)
    - Data discretization

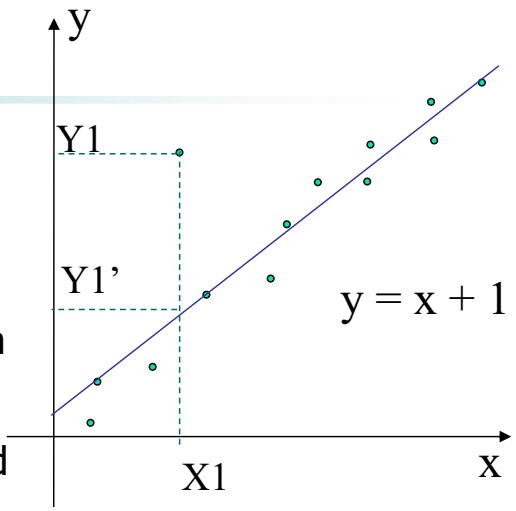
Data Reduction :b) Numerosity Reduction

- Reduce data volume by choosing alternative, *smaller forms* of data representation
- **Parametric methods** (e.g., regression)
  - Assume the data fits some model, estimate model parameters, store only the parameters, and discard the data (except possible outliers)
  - Ex.: 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, …

Parametric Data Reduction: Regression and Log-Linear Models

- **Linear regression**
  - Data 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

Regression Analysis



- Regression analysis: A collective name for techniques for the modeling and analysis of numerical data consisting of values of a ***dependent variable*** (also called ***response variable*** or *measurement*) and of one or more *independent variables* (aka. ***explanatory variables*** or ***predictors***)
- The parameters are estimated so as to give a "**best fit**" of the data
- Most commonly the best fit is evaluated by using the ***least squares method***, but other criteria have also been used
- Used for prediction (including forecasting of time-series data), inference, hypothesis testing, and modeling of causal relationships

Clustering

- Partition data set into clusters based on similarity, and store cluster representation (e.g., centroid and diameter) 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
- Cluster analysis will be studied in depth in Chapter 10

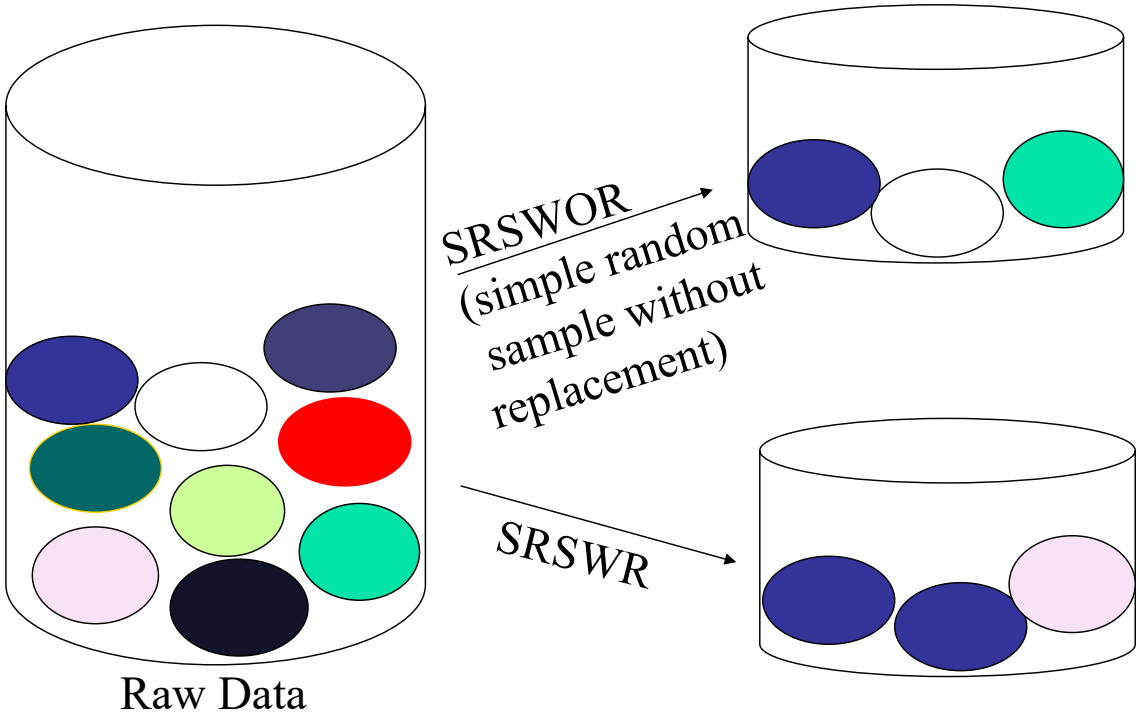
Sampling

- Sampling: obtaining a small sample *s* to represent the whole data set *N*
- Allow a mining algorithm to run in complexity that is potentially sub-linear to the size of the data
- Key principle: Choose a representative subset of the data
  - Simple random sampling may have very poor performance in the presence of skew
  - Develop adaptive sampling methods, e.g., stratified sampling:
- Note: Sampling may not reduce database I/Os (page at a time)

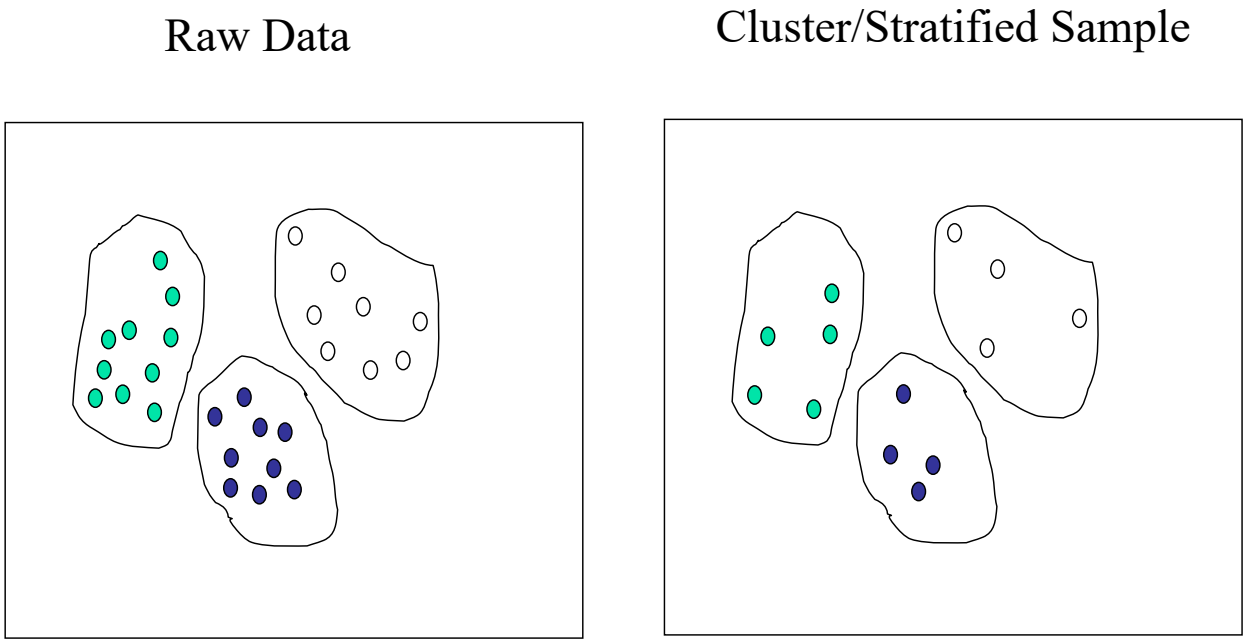
Types of Sampling

- **Simple random sampling**
  - There is an equal probability of selecting any particular item
- **Sampling without replacement**
  - Once an object is selected, it is removed from the population
- **Sampling with replacement**
  - A selected object is not removed from the population
- **Stratified sampling:**
  - If *D* is divided into mutually disjoint parts called *strata,* a stratified sample of *D* is generated by obtaining an SRS at each stratum.

Sampling: With or without Replacement



Sampling: Cluster or Stratified Sampling



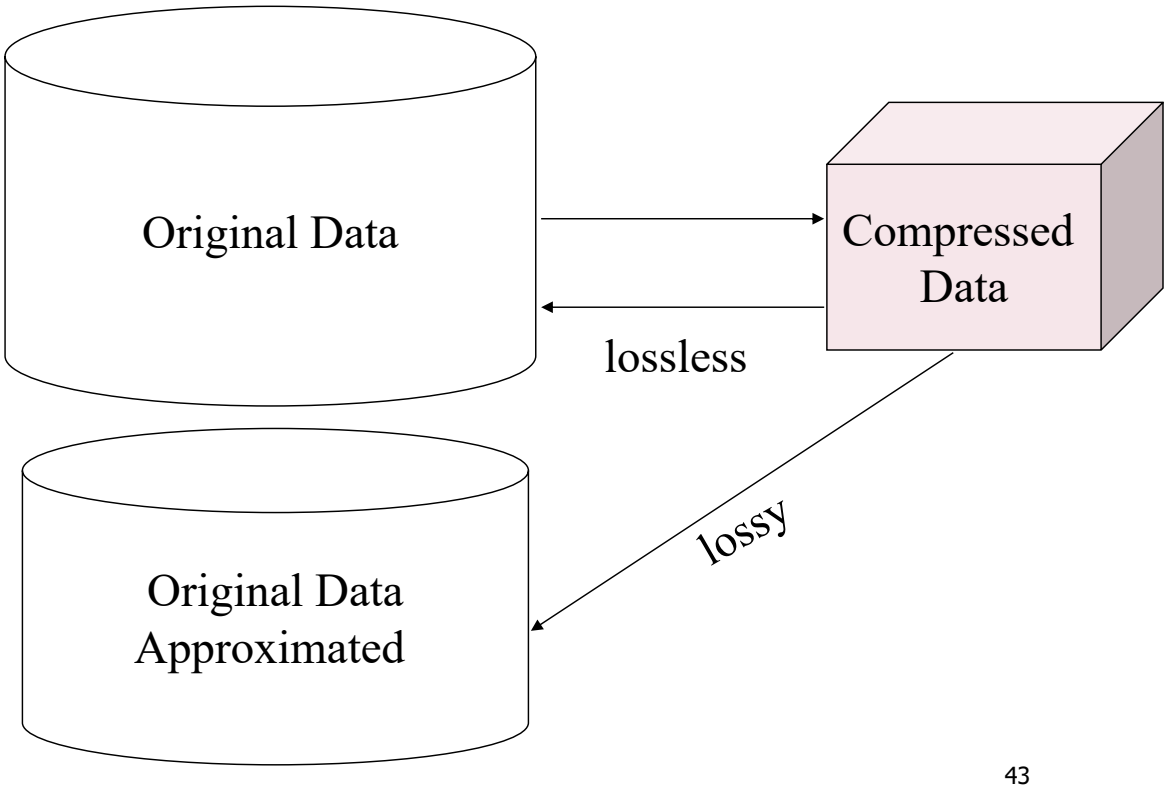
Data Cube Aggregation

- The lowest level of a data cube (base cuboid)
  - 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

Data Reduction :c) Data Compression

- String compression
  - There are extensive theories and well-tuned algorithms
  - Typically lossless, but only limited manipulation is possible without expansion
- Audio/video compression
  - Typically lossy compression, with progressive refinement
  - Sometimes small fragments of signal can be reconstructed without reconstructing the whole
- Dimensionality and numerosity reduction may also be considered as forms of data compression

Data Compression



4. Data Transformation

- A function that maps the entire set of values of a given attribute to a new set of replacement values s.t. each old value can be identified with one of the new values
- Methods
  - Smoothing: Remove noise from data
  - Attribute/feature construction
    - New attributes constructed from the given ones
  - Aggregation: Summarization, data cube construction
  - Normalization: Scaled to fall within a smaller, specified range
    - min-max normalization
    - z-score normalization
    - normalization by decimal scaling
  - Discretization: Concept hierarchy climbing

Normalization

- **Min-max normalization**: to [new_min$_A$, new_max$_A$]

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

  - Ex. Let income range \$12,000 to \$98,000 normalized to [0.0, 1.0]. Then \$73,000 is mapped to $\frac{73{,}600 - 12{,}000}{98{,}000 - 12{,}000}(1.0 - 0) + 0 = 0.716$

- **Z-score normalization** (μ: mean, σ: standard deviation):

$$v' = \frac{v - \mu_A}{\sigma_A}$$

  - Ex. Let μ = 54,000, σ = 16,000. Then $\frac{73{,}600 - 54{,}000}{16{,}000} = 1.225$

- **Normalization by decimal scaling**

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

Discretization

- Three types of attributes
  - Nominal—values from an unordered set, e.g., color, profession
  - Ordinal—values from an ordered set, e.g., military or academic rank
  - Numeric—real numbers, e.g., integer or real numbers
- Discretization: Divide the range of a continuous attribute into intervals
  - Interval labels can then be used to replace actual data values
  - Reduce data size by discretization
  - Supervised vs. unsupervised
  - Split (top-down) vs. merge (bottom-up)
  - Discretization can be performed recursively on an attribute
  - Prepare for further analysis, e.g., classification

Data Discretization Methods

- Typical methods: All the methods can be applied recursively
  - Binning
    - Top-down split, unsupervised
  - Histogram analysis
    - Top-down split, unsupervised
  - Clustering analysis (unsupervised, top-down split or bottom-up merge)
  - Decision-tree analysis (supervised, top-down split)
  - Correlation (e.g., χ^2) analysis (unsupervised, bottom-up merge)

Simple Discretization: Binning

- Equal-width (distance) partitioning
  - 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
  - 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

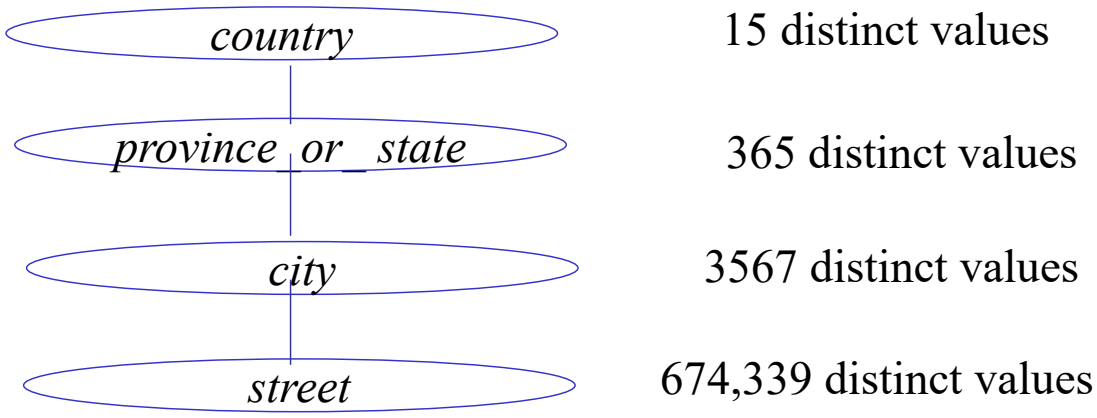
* Partition into equal-frequency (**equi-depth**) bins:
  - Bin 1: 4, 8, 9, 15
  - Bin 2: 21, 21, 24, 25
  - Bin 3: 26, 28, 29, 34
* Smoothing by **bin means**:
  - Bin 1: 9, 9, 9, 9
  - Bin 2: 23, 23, 23, 23
  - Bin 3: 29, 29, 29, 29
* Smoothing by **bin boundaries**:
  - Bin 1: 4, 4, 4, 15
  - Bin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

Concept Hierarchy Generation

- **Concept hierarchy** organizes concepts (i.e., attribute values) hierarchically and is usually associated with each dimension in a data warehouse
- Concept hierarchies facilitate **drilling and rolling** in data warehouses to view data in multiple granularity
- Concept hierarchy formation: Recursively reduce the data by collecting and replacing low level concepts (such as numeric values for *age*) by higher level concepts (such as *youth, adult*, or *senior*)
- Concept hierarchies can be explicitly specified by domain experts and/or data warehouse designers
- Concept hierarchy can be automatically formed for both numeric and nominal data. For numeric data, use discretization methods shown.

Automatic Concept Hierarchy Generation

- Some hierarchies can be automatically generated based on the analysis of the number of distinct values per attribute in the data set
  - The attribute with the most distinct values is placed at the lowest level of the hierarchy



Summary

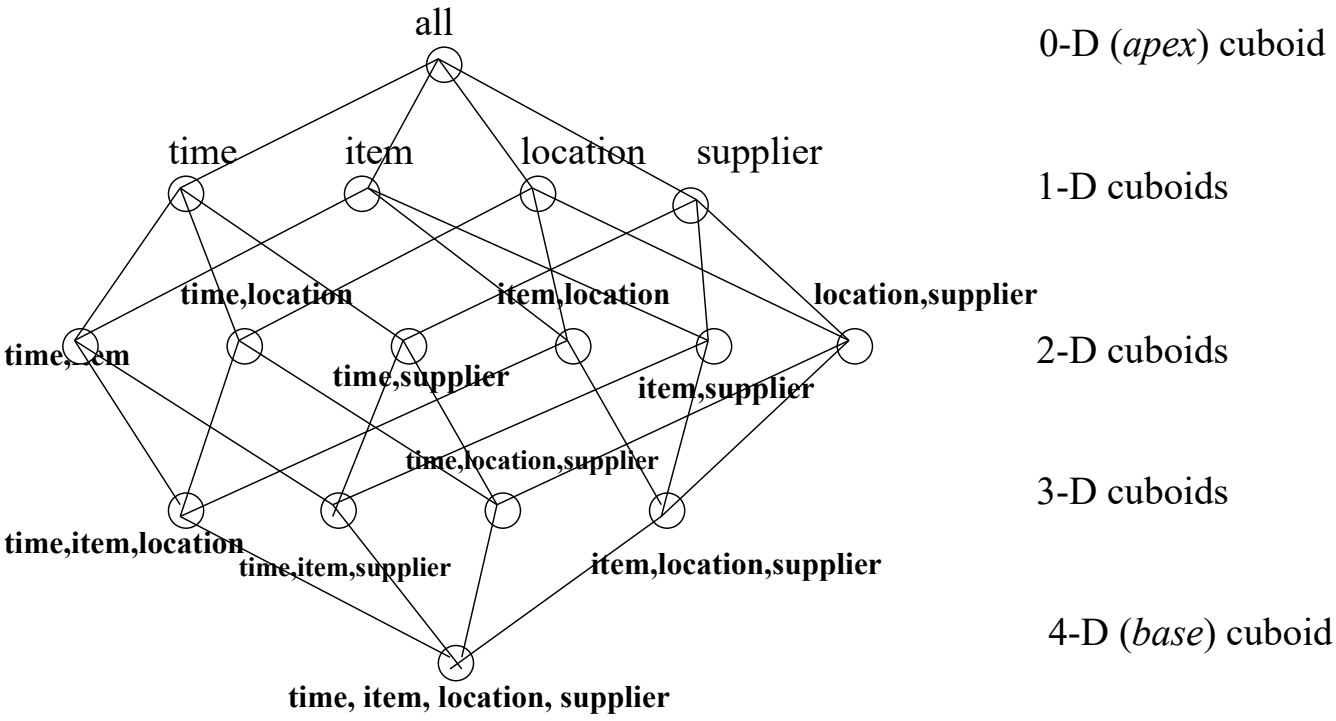
- **Data quality**: accuracy, completeness, consistency, timeliness, believability, interpretability
- **Data cleaning**: e.g. missing/noisy values, outliers
- **Data integration** from multiple sources:
  - Entity identification problem
  - Remove redundancies
  - Detect inconsistencies
- **Data reduction**
  - Dimensionality reduction
  - Numerosity reduction
  - Data compression
- **Data transformation and data discretization**
  - Normalization
  - Concept hierarchy generation

2.3 OLAP & Multidimensional Data Analysis

From Tables and Spreadsheets to Data Cubes

- A **data warehouse** is based on a multidimensional data model which views data in the form of a data cube
- A data cube, such as sales, allows data to be modeled and viewed in multiple dimensions
  - **Dimension tables**, such as item (item_name, brand, type), or time(day, week, month, quarter, year) or location(USA,Canada)
  - **Fact table** contains **measures** (such as dollars_sold) and keys to each of the related dimension tables
- In data warehousing literature, an n-D base cube is called a base cuboid. The top most 0-D cuboid, which holds the highest-level of summarization, is called the apex cuboid. The lattice of cuboids forms a data cube.

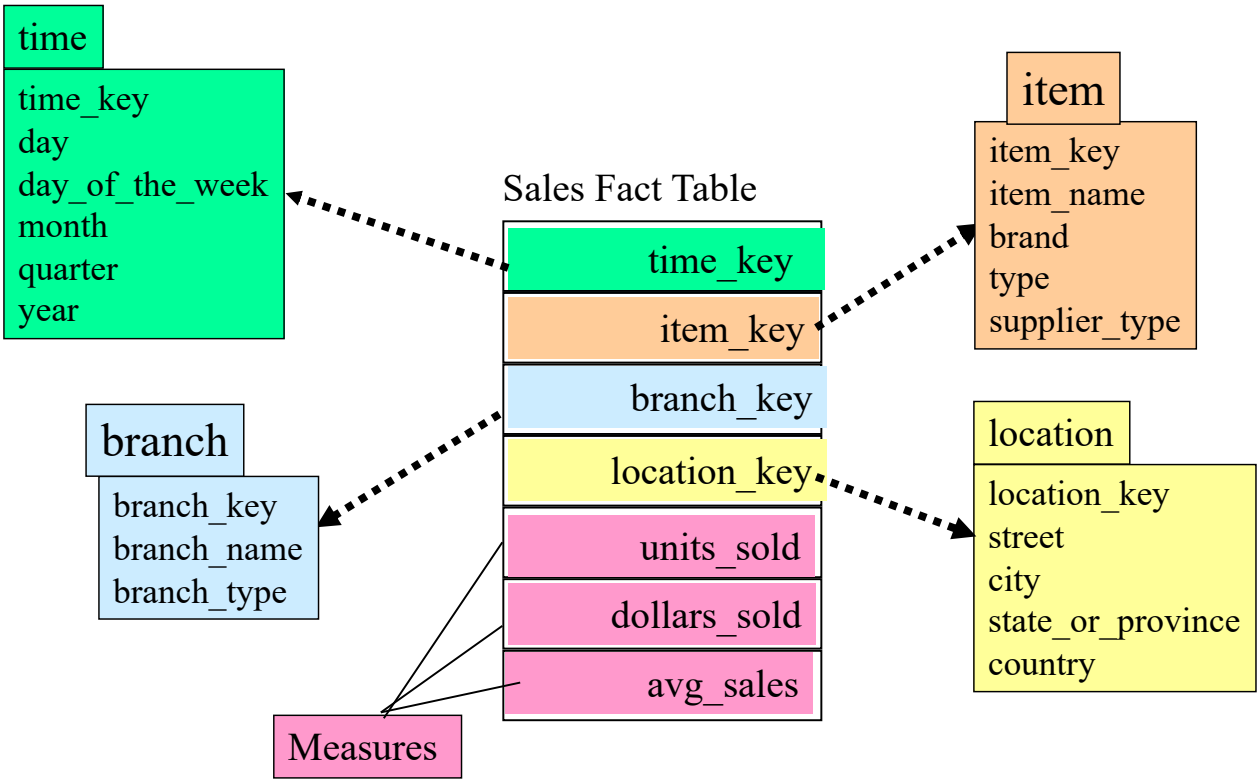
Cube: A Lattice of Cuboids



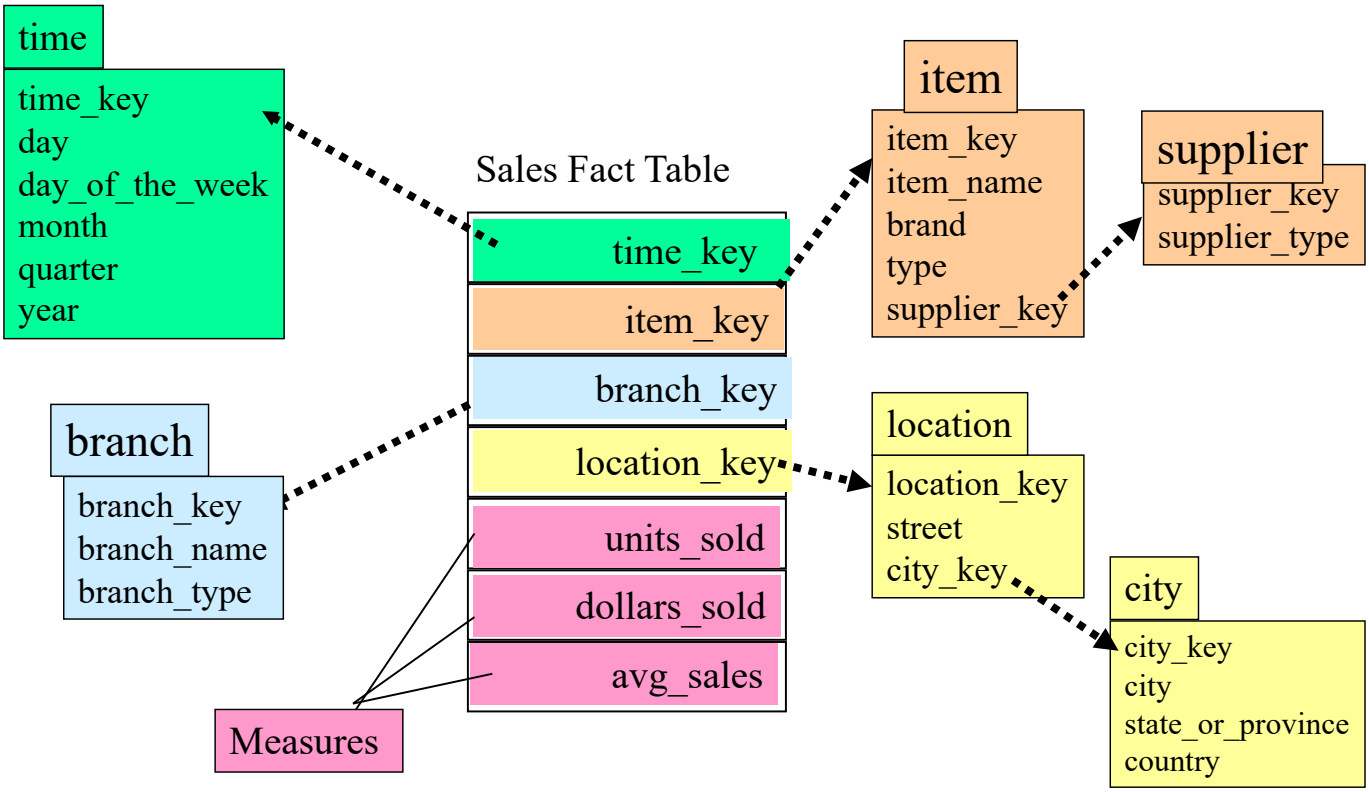
Conceptual Modeling of Data Warehouses

- 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

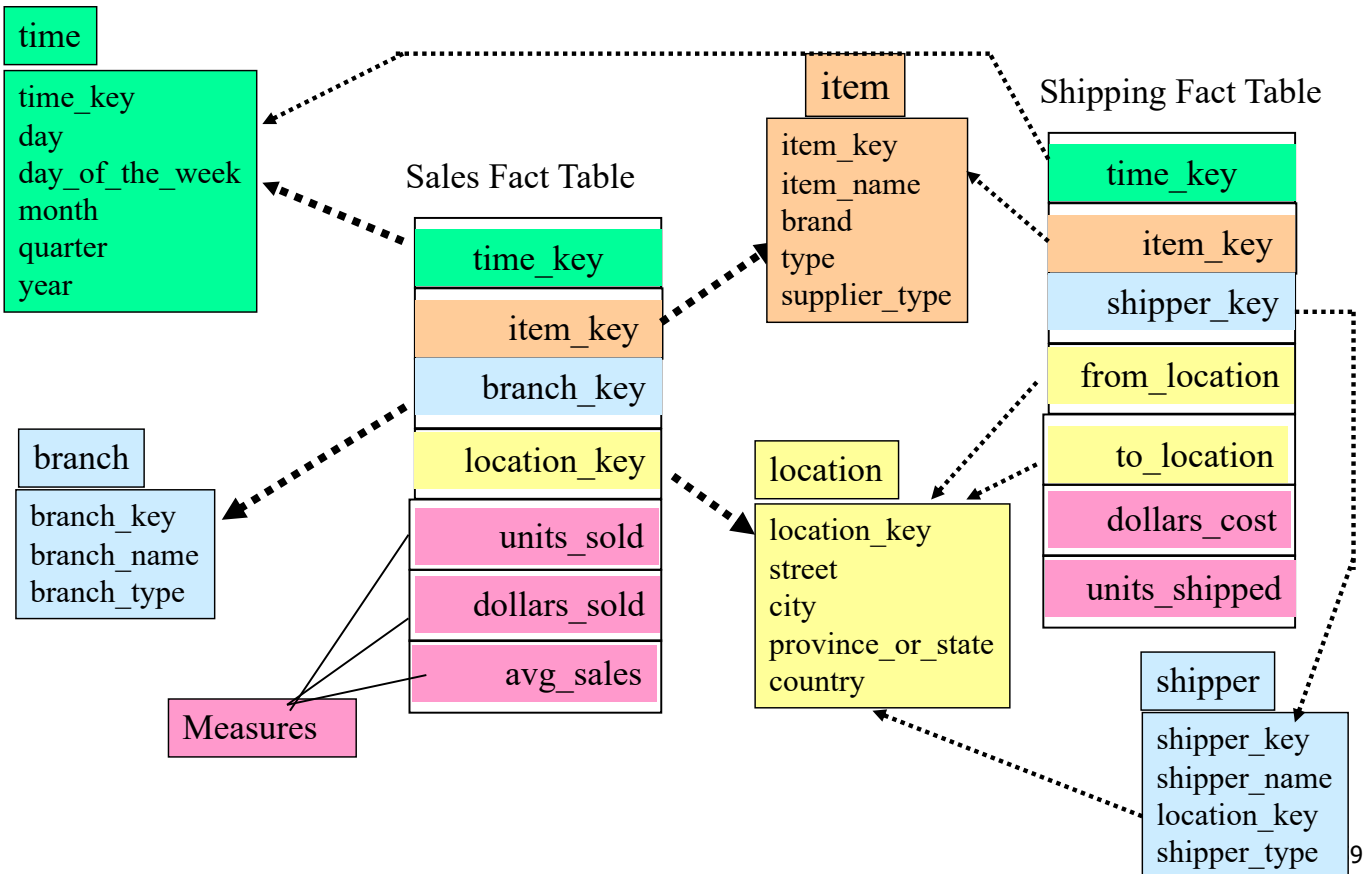
Example of Star Schema



Example of Snowflake Schema

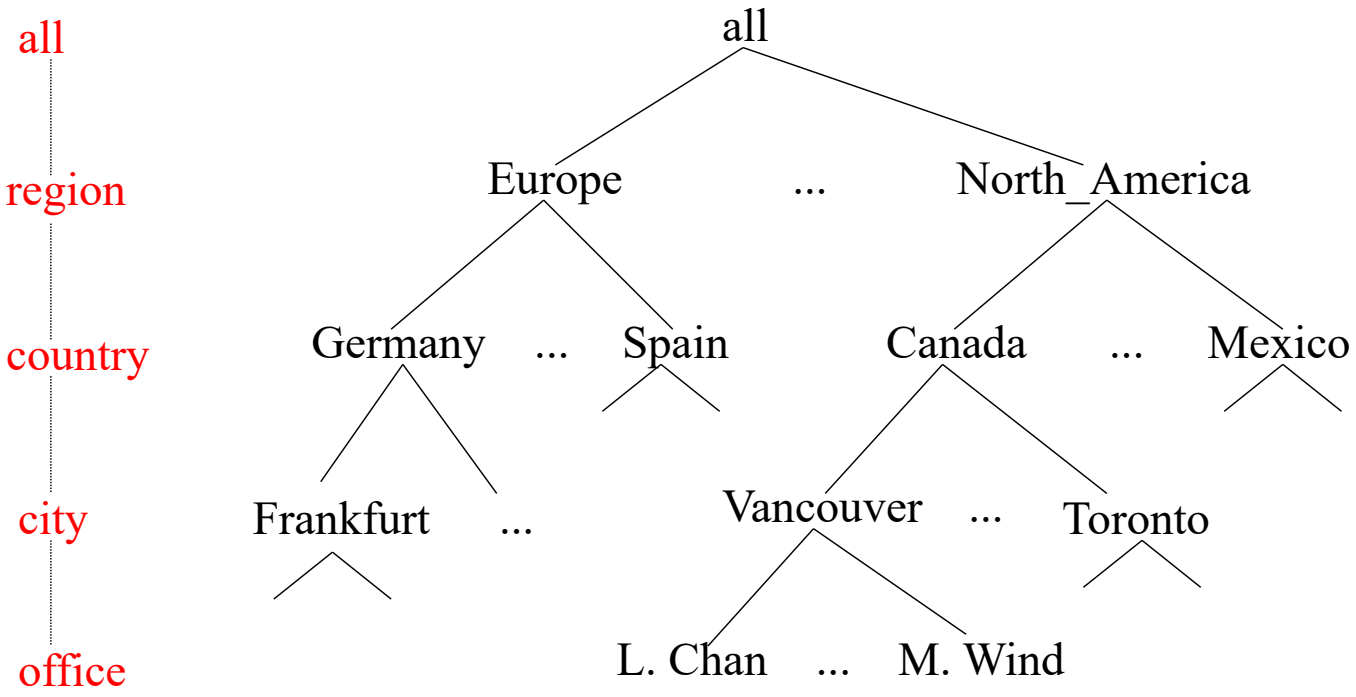


Example of Fact Constellation



9

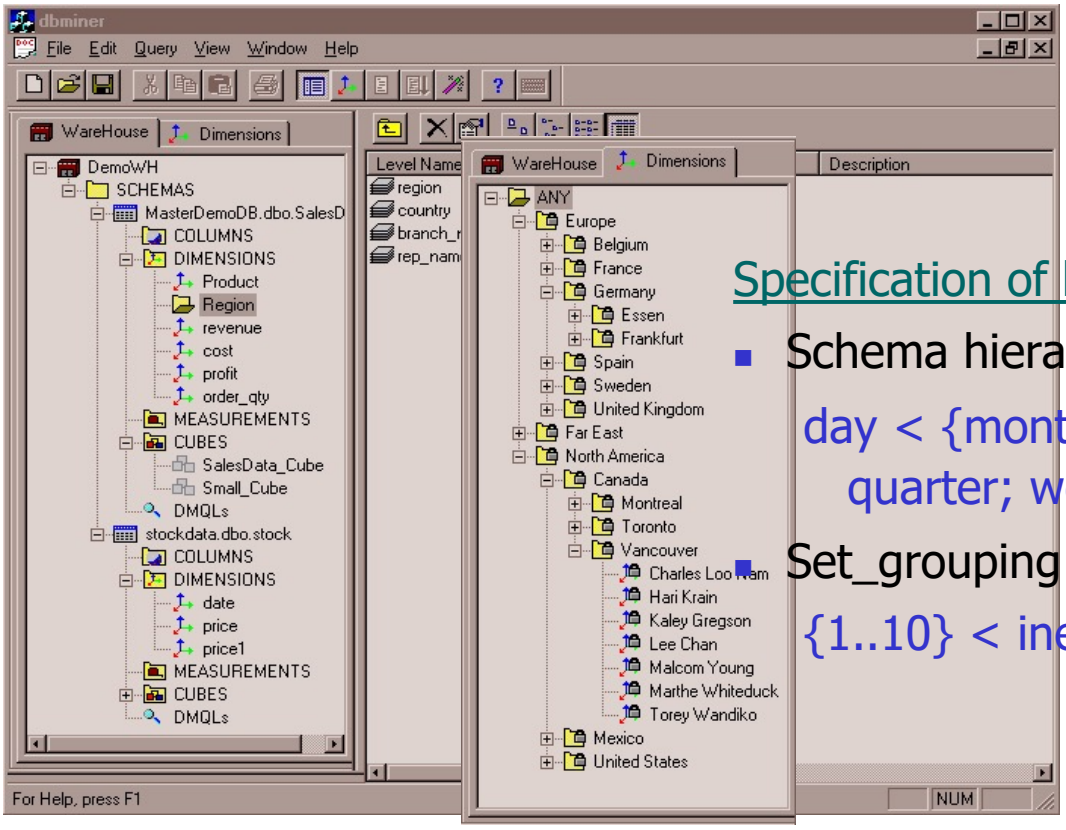
A Concept Hierarchy: Dimension (location)



Data Cube 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()
- Holistic: if there is no constant bound on the storage size needed to describe a subaggregate.
  - E.g., median(), mode(), rank()

View of Warehouses and Hierarchies



Specification of hierarchies

- Schema hierarchy

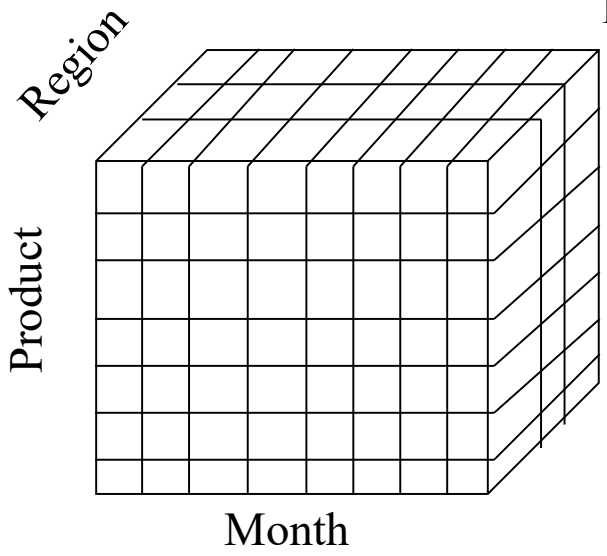
  day < {month < quarter; week} < year

- Set_grouping hierarchy

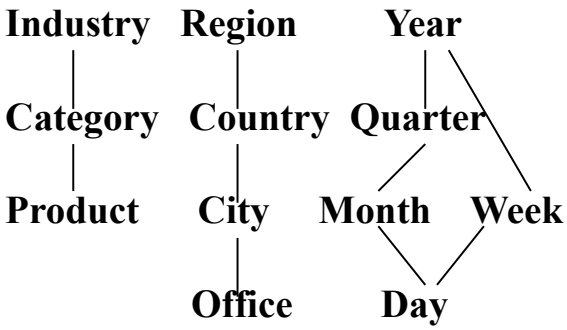
  {1..10} < inexpensive

Multidimensional Data

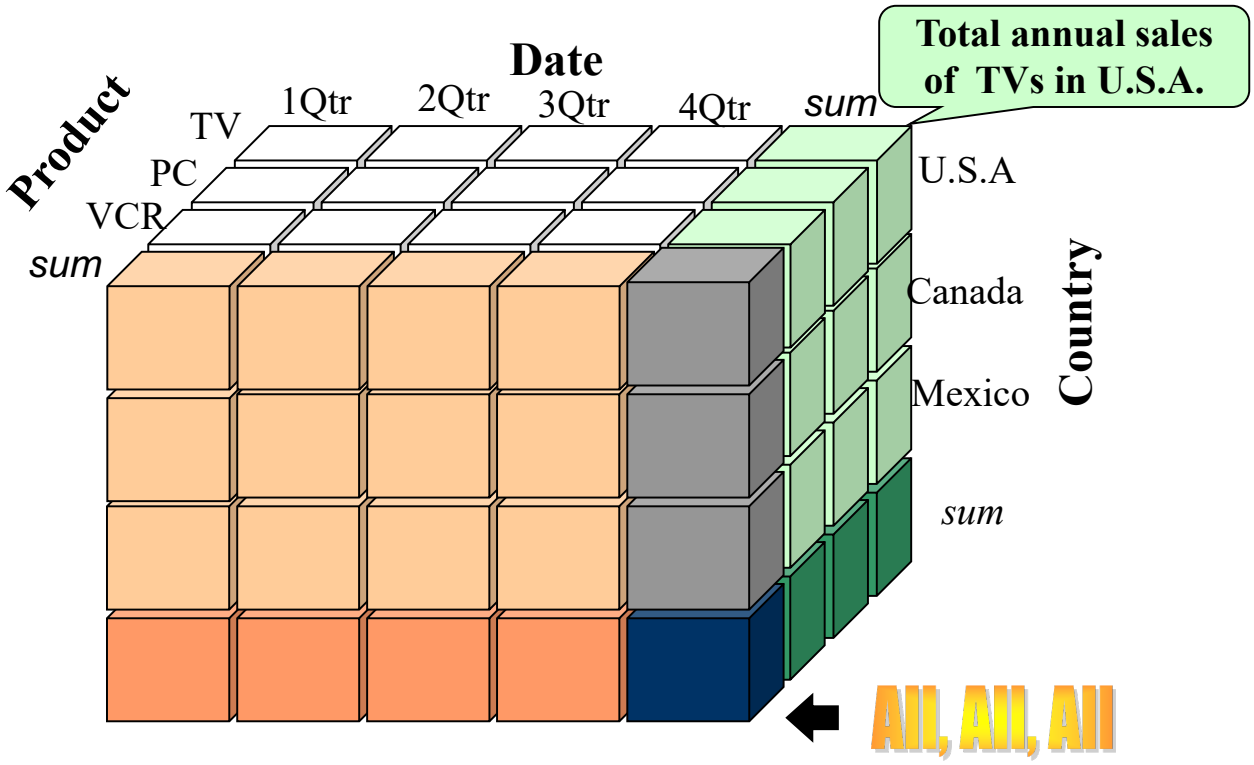
- Sales volume as a function of product, month, and region



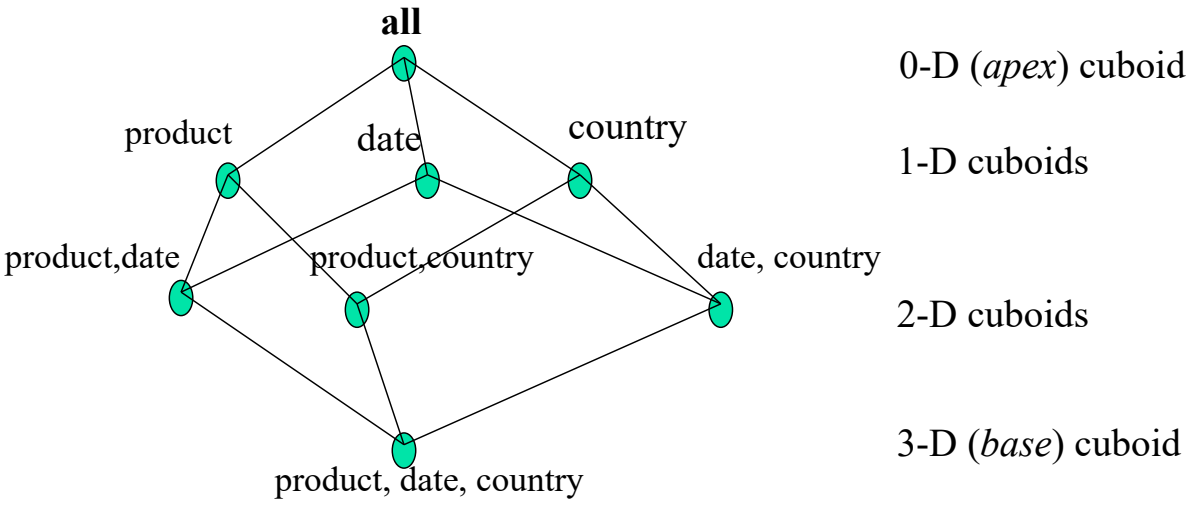
Dimensions: *Product, Location, Time*
Hierarchical summarization paths



A Sample Data Cube



Cuboids Corresponding to the Cube



Typical OLAP Operations

- Roll up (drill-up): summarize data
  - *by climbing up hierarchy or by dimension reduction*
- Drill down (roll down): reverse of roll-up
  - *from higher level summary to lower level summary or detailed data, or introducing new dimensions*
- Slice and dice: *project and select*
- Pivot (rotate):
  - *reorient the cube, visualization, 3D to series of 2D planes*

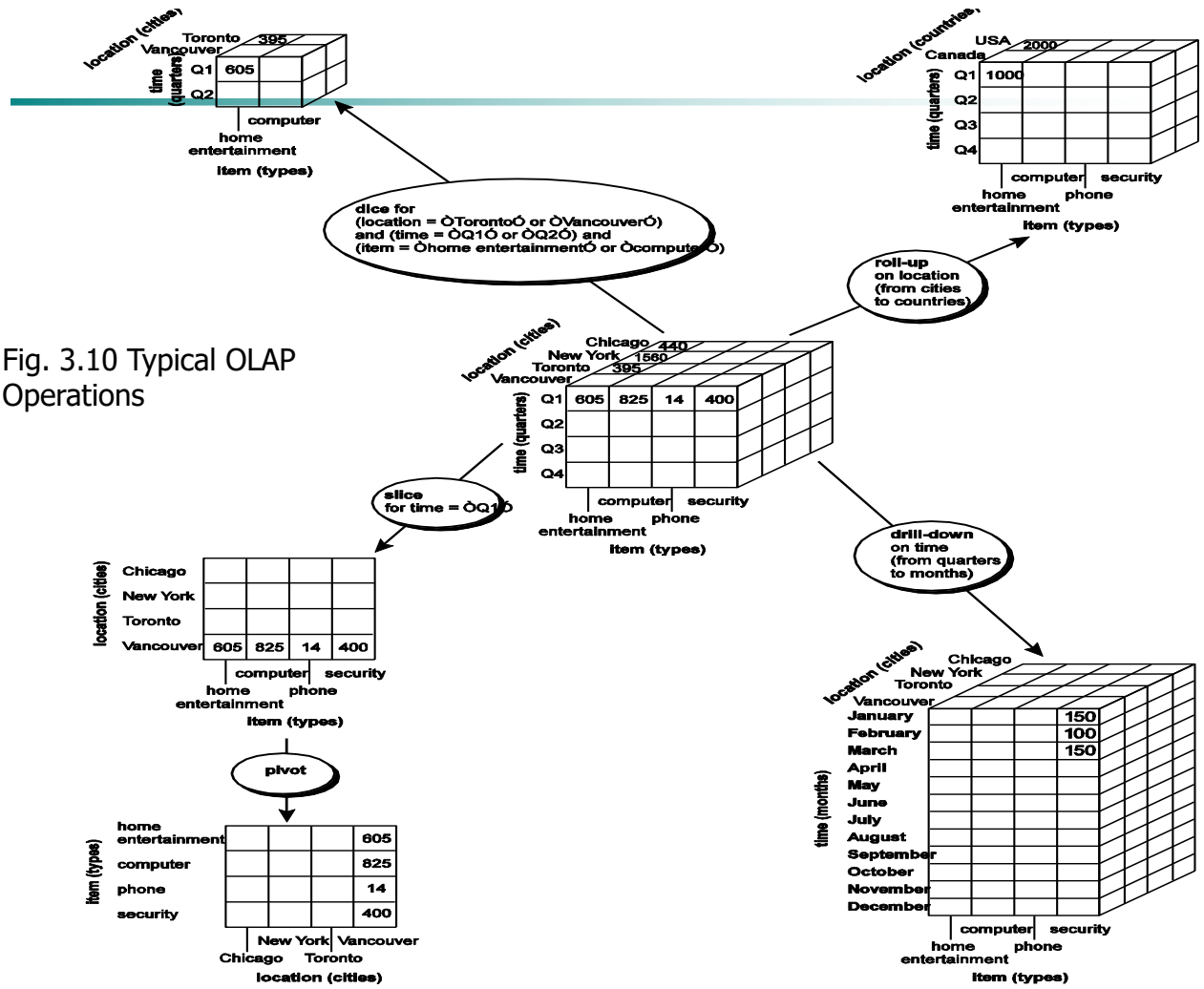


Fig. 3.10 Typical OLAP Operations

Summary

- Data warehousing: A multi-dimensional model of a data warehouse
  - A data cube consists of *dimensions* & *measures*
  - Star schema, snowflake schema, fact constellations
  - OLAP operations: drilling, rolling, slicing, dicing and pivoting
- Data Warehouse Architecture, Design, and Usage
  - Multi-tiered architecture
  - Business analysis design framework
  - Information processing, analytical processing, data mining, OLAM (Online Analytical Mining)
- Implementation: Efficient computation of data cubes
  - Partial vs. full vs. no materialization
  - Indexing OALP data: Bitmap index and join index
  - OLAP query processing
  - OLAP servers: ROLAP, MOLAP, HOLAP
- Data generalization: Attribute-oriented induction

OLAP

What is OLAP (Online Analytical Processing)?

OLAP stands for **On-Line Analytical Processing**. OLAP is a classification of software technology which authorizes analysts, managers, and executives to gain insight into information through fast, consistent, interactive access in a wide variety of possible views of data that has been transformed from raw information to reflect the real dimensionality of the enterprise as understood by the clients.

OLAP implement the multidimensional analysis of business information and support the capability for complex estimations, trend analysis, and sophisticated data modeling. It is rapidly enhancing the essential foundation for Intelligent Solutions containing Business Performance Management, Planning, Budgeting, Forecasting, Financial Documenting, Analysis, Simulation-Models, Knowledge Discovery, and Data Warehouses Reporting. OLAP enables end-clients to perform ad hoc analysis of record in multiple dimensions, providing the insight and understanding they require for better decision making.

Characteristics of OLAP

1. Multidimensional conceptual view: OLAP systems let business users have a dimensional and logical view of the data in the data warehouse. It helps in carrying slice and dice operations.

2. Multi-User Support: Since the OLAP techniques are shared, the OLAP operation should provide normal database operations, containing retrieval, update, adequacy control, integrity, and security.

3. Accessibility: OLAP acts as a mediator between data warehouses and front-end. The OLAP operations should be sitting between data sources (e.g., data warehouses) and an OLAP front-end.

4. Storing OLAP results: OLAP results are kept separate from data sources.

5. Uniform documenting performance: Increasing the number of dimensions or database size should not significantly degrade the reporting performance of the OLAP system.

6. OLAP provides for distinguishing between zero values and missing values so that aggregates are computed correctly.

7. OLAP system should ignore all missing values and compute correct aggregate values.

8. OLAP facilitate interactive query and complex analysis for the users.

9. OLAP allows users to drill down for greater details or roll up for aggregations of metrics along a single business dimension or across multiple dimension.

10. OLAP provides the ability to perform intricate calculations and comparisons.

11. OLAP presents results in a number of meaningful ways, including charts and graphs.

Guidelines for OLAP Implementation

OLAP was introduced by **Dr.E.F.Codd** in 1993 and he presented 12 rules regarding OLAP:

1. Multidimensional Conceptual View:
Multidimensional data model is provided that is intuitively analytical and easy to use. A multidimensional data model decides how the users perceive business problems.

2.Transparency:
It makes the technology, underlying data repository, computing architecture, and the diverse nature of source data totally transparent to users.

3.Accessibility:
Access should provided only to the data that is actually needed to perform the specific analysis, presenting a single, coherent and consistent view to the users.

4.Consistent Reporting Performance:
Users should not experience any significant degradation in reporting performance as the number of dimensions or the size of the database increases. It also ensures users must perceive consistent run time, response time or machine utilization every time a given query is run.

5. **Client/Server Architecture:**
It conforms the system to the principles of client/server architecture for optimum performance, flexibility, adaptability, and interoperability.

6. Generic Dimensionality:
It should be ensured that very data dimension is equivalent in both structure and operational capabilities. Have one logical structure for all dimensions.

7. Dynamic Sparse Matrix Handling:
Adaption should be of the physical schema to the specific analytical model being created and loaded that optimizes sparse matrix handling.

8. Multi-user Support:
Support should be provided for end users to work concurrently with either the same analytical model or to create different models from the same data.

9. **Unrestricted Cross-dimensional Operations:**
System should have abilities to recognize dimensional and automatically perform roll-up and drill-down operations within a dimension or across dimensions.

10. Intuitive Data Manipulation:
Consolidation path reorientation, drill-down, and roll-up and other manipulations to be accomplished intuitively should be enabled and directly via point and click actions.

11. Flexible Reporting:
Business user is provided capabilities to arrange columns, rows, and cells in manner that gives the facility of easy manipulation, analysis and synthesis of information.

12.Unlimited Dimensions and Aggregation Levels:
There should be at least fifteen or twenty data dimensions within a common analytical model.

Classification and Prediction: Review of Basic Concepts

Supervised vs. Unsupervised Learning

- **Supervised learning (classification)**
  - We know the class labels and the number of classes
  - 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)**
  - We do not know the class labels and may not know the number of classes
  - 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

Prediction Problems: Classification vs. Numeric Prediction(Regression)

- **Classification**
  - predicts categorical class labels (discrete or nominal)
  - 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
- **Numeric Prediction (Regression)**
  - models continuous-valued functions, i.e., predicts unknown or missing values
- **Typical applications**
  - Credit/loan approval:
  - Medical diagnosis: if a tumor is cancerous or benign
  - Fraud detection: if a transaction is fraudulent
  - Web page categorization: which category it is

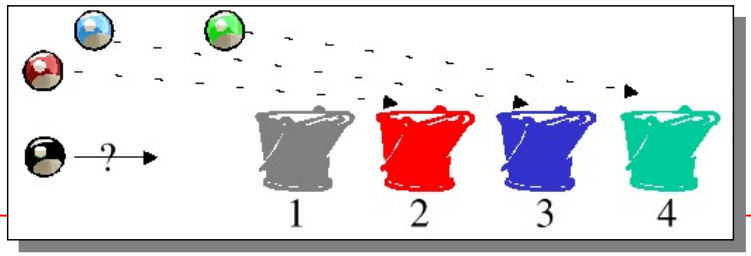
What Is Prediction/Estimation?

- **(Numerical) prediction is similar to classification**
  - construct a model
  - use model to predict continuous or ordered value for a given input
- **Major method for prediction: regression**
  - model the relationship between one or more *independent* or **predictor** variables and a *dependent* or **response** variable
- **Regression analysis**
  - Linear and multiple regression
  - Non-linear regression
  - Other regression methods: generalized linear model, Poisson regression, log-linear models, regression trees
- **Prediction is different from classification**
  - Classification refers to predict categorical class label
  - Prediction models continuous-valued functions

What Is Classification?

- The goal of data classification is to organize and categorize data in distinct classes
  - A model is first created based on the data distribution
  - The model is then used to classify new data
  - Given the model, a class can be predicted for new data

- Classification = prediction for discrete and nominal values (e.g., class/category labels)
  - Also called "Categorization"



Prediction, Clustering, Classification

- **What is Prediction/Estimation?**
  - The goal of prediction is to forecast or deduce the value of an attribute based on values of other attributes
  - A model is first created based on the data distribution
  - The model is then used to predict future or unknown values
  - Most common approach: regression analysis

Example of Classification Learning

- **X (Feature Space): <size, color, shape>**
  - size ∈ {small, medium, large}
  - color ∈ {red, blue, green}
  - shape ∈ {square, circle, triangle}

X = *instance language* or *instance* or *feature space*.

- ***C* = {positive, negative}** C = fixed set of class labels
- ***D*:**

Example	Size	Color	Shape	Category/Class
1	small	red	circle	positive
2	large	red	circle	positive
3	small	red	triangle	negative
4	large	blue	circle	negative

D = set of training examples

- **Hypotheses? circle → positive? red → positive?**

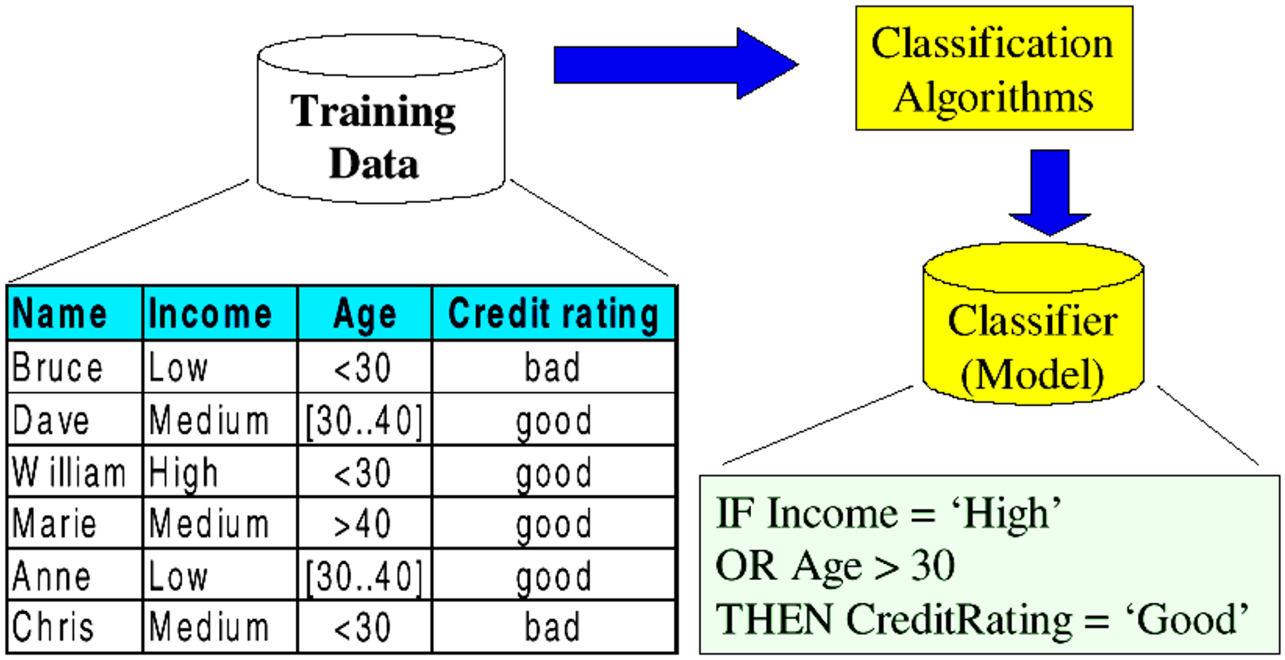
General Learning Issues(All Predictive Modeling Tasks)

- **Many hypotheses can be consistent with the training data**
- **Bias: Any criteria other than consistency with the training data that is used to select a hypothesis**
- **Classification accuracy (% of instances classified correctly)**
  - Measured on independent test data
- **Efficiency Issues:**
  - Training time (efficiency of training algorithm)
  - Testing time (efficiency of subsequent classification)
- **Generalization**
  - Hypotheses must generalize to correctly classify instances not in training data
  - Simply memorizing training examples is a consistent hypothesis that does not generalize

Classification: 3 Step Process

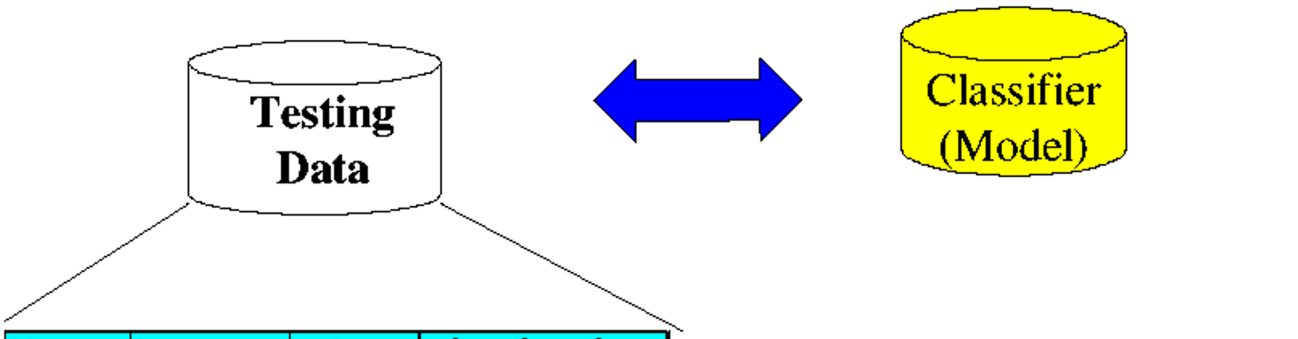
- **1. Model construction (Learning):**
  - Each record (instance, example) is assumed to belong to a predefined class, as determined by one of the attributes
    - This attribute is call the target attribute
    - The values of the target attribute are the **class labels**
  - The set of all instances used for learning the model is called **training set**
  - The model may be represented in many forms: **decision trees, probabilities, neural networks, ….**
- **2. Model Evaluation (Accuracy):**
  - Estimate accuracy rate of the model based on a **test set**
  - The known labels of test instances are compared with the predicts class from model
  - Test set is independent of training set otherwise over-fitting will occur
- **3. Model Use (Classification):**
  - The model is used to classify unseen instances (i.e., to predict the class labels for new unclassified instances)
  - Predict the value of an actual attribute

Model Construction



Name	Income	Age	Credit rating
Bruce	Low	<30	bad
Dave	Medium	[30..40]	good
William	High	<30	good
Marie	Medium	>40	good
Anne	Low	[30..40]	good
Chris	Medium	<30	bad

Model Evaluation

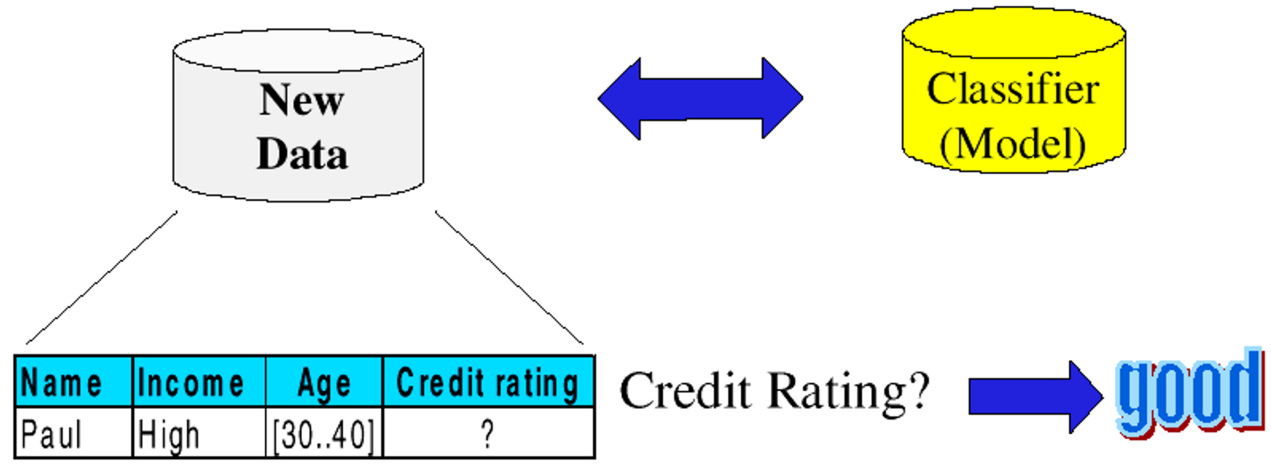


Name	Income	Age	Credit rating
Tom	Medium	<30	bad
Jane	High	<30	bad
Wei	High	>40	good
Hua	Medium	[30..40]	good

How accurate is the model?

IF Income = 'High'
OR Age > 30
THEN CreditRating = 'Good'

Model Use: Classification



Name	Income	Age	Credit rating
Paul	High	[30..40]	?

3.2 Decision Tree Classifier

Classification: 3 Step Process

1. **Model construction (Learning):**

   Each record (instance, example) is assumed to belong to a predefined class, as determined by one of the attributes

   This attribute is called the target attribute

   The values of the target attribute are the **class labels**

   The set of all instances used for learning the model is called **training set**

2. **Model Evaluation (Accuracy):**

   Estimate accuracy rate of the model based on a **test set**

   The known labels of test instances are compared with the predicts class from model

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

3. **Model Use (Classification):**

   The model is used to classify unseen instances (i.e., to predict the class labels for new unclassified instances)

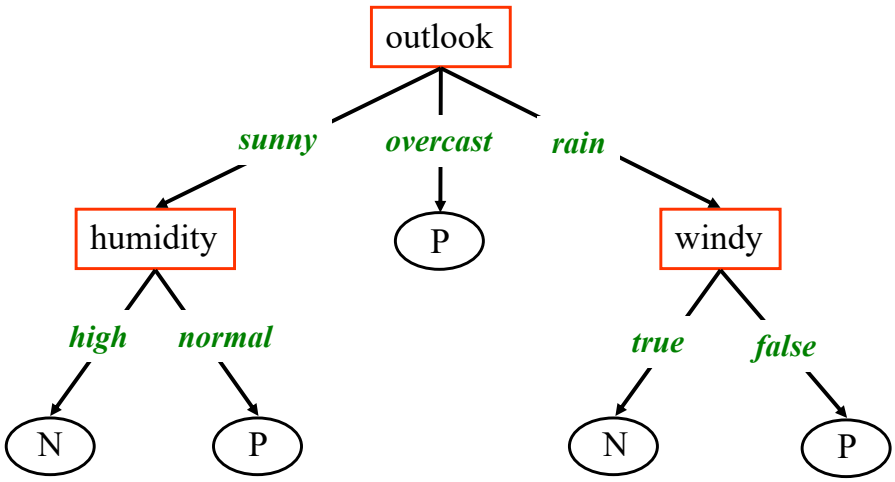
   Predict the value of an actual attribute

Classification Methods

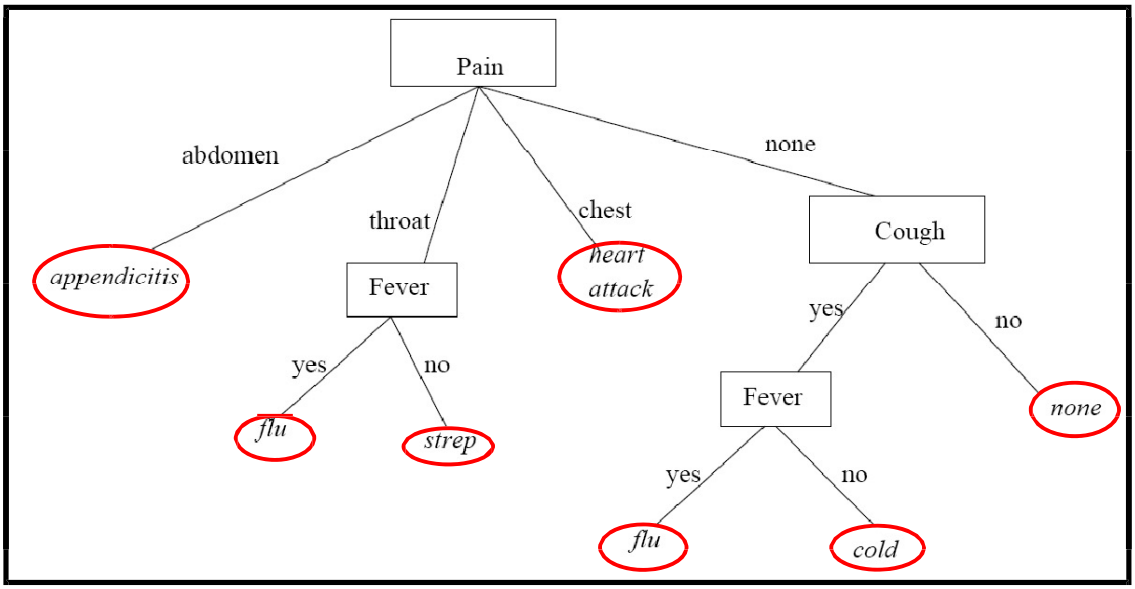
- Decision Tree Induction
- Bayesian Classification
- K-Nearest Neighbor
- Neural Networks
- If Then Rule
- Association-Based Classification
- Genetic Algorithms
- Many More ….
- Also Ensemble Methods

Decision Trees

- A decision tree is a flow-chart-like tree structure
  - Internal node denotes a test on an attribute (feature)
  - Branch represents an outcome of the test
  - Leaf node represents class label or class label distribution



Decision Tree Classific ation Example



Decision Tree Learning Overview

• Decision Tree learning is one of the most widely used and practical methods for inductive inference over supervised data.

• A decision tree represents a procedure for classifying categorical data based on their attributes.

• It is also efficient for processing large amount of data, so is often used in data mining application.

• The construction of decision tree does not require any domain knowledge or parameter setting, and therefore appropriate for exploratory knowledge discovery.

• Their representation of acquired knowledge in tree form is intuitive and easy to assimilate by humans

Decision node attribute Selection:

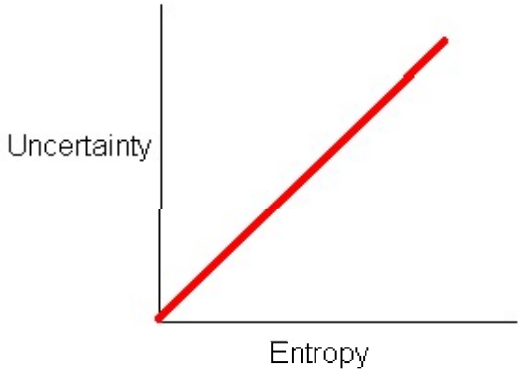
- Hunt's Algorithm (Random node selection)
- Based on Entropy Calculation : (select maximum as node) **ID3 (Gain)**, C4.5(Gain Ratio)
- Based on Gini-Index : (select minimum as node) SLIQ,SPRINT,CART

What is Entropy?

- The entropy is a measure of the uncertainty associated with a random variable
- As uncertainty and or randomness increases for a result set so does the entropy
- Values range from 0 – 1 to represent the entropy of information

For given data partition D:

$$Entropy\,(D) \equiv \sum_{i=1} - p_i \log_2(p_i)$$



$$Entropy\,(D) \equiv \sum_{i=1} - p_i \log_2(p_i)$$

Gain (Information Gain)

- Information gain is used as an attribute selection measure
- Pick the attribute that has the highest **Gain**

$$\boldsymbol{Gain(D, A) = Entropy(D) - \sum_{j=1}^{v} \frac{|D_j|}{|D|} Entropy(D_j)}$$

- ***D*: given data partition**
- ***A*: attribute**

v : Suppose we were partition the tuples in ***D*** on some attribute ***A*** having ***v*** distinct values ***D*** is split into v partition or subsets, {D_1, D2, …Dj}, where ***Dj*** contains those tupes in ***D*** that have outcome a_j of ***A***.

RID	*age*	*income*	*student*	*credit_rating*	*Class: buys_computer*
1	youth	high	no	fair	no
2	youth	high	no	excellent	no
3	middle_aged	high	no	fair	yes
4	senior	medium	no	fair	yes
5	senior	low	yes	fair	yes
6	senior	low	yes	excellent	no
7	middle_aged	low	yes	excellent	yes
8	youth	medium	no	fair	no
9	youth	low	yes	fair	yes
10	senior	medium	yes	fair	yes
11	youth	medium	yes	excellent	yes
12	middle_aged	medium	no	excellent	yes
13	middle_aged	high	yes	fair	yes
14	senior	medium	no	excellent	no

Table 6.1 Class-labeled training tuples from AllElectronics customer database.

- Class P: *buys_computer* = "yes"
- Class N: *buys_computer* = "no"

$$Entropy(D) = -\frac{9}{14}\log_2(\frac{9}{14}) - \frac{5}{14}\log_2(\frac{5}{14}) = 0.940$$

- Compute the expected information requirement for each attribute: start with the attribute *age*

$$Gain(age, D)$$

$$= Entropy(D) - \sum_{v \in \{Youth, Middle\ aged, Senior\}} \frac{|S_v|}{|S|} Entropy(S_v)$$

$$= Entropy(D) - \frac{5}{14} Entropy(S_{youth}) - \frac{4}{14} Entropy(S_{middle\_aged}) - \frac{5}{14} Entropy(S_{senior})$$

$$= 0.246$$

$$Gain(age, D) = \mathbf{0.246}$$

$$Gain(income, D) = 0.029$$

$$Gain(student, D) = 0.151$$

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

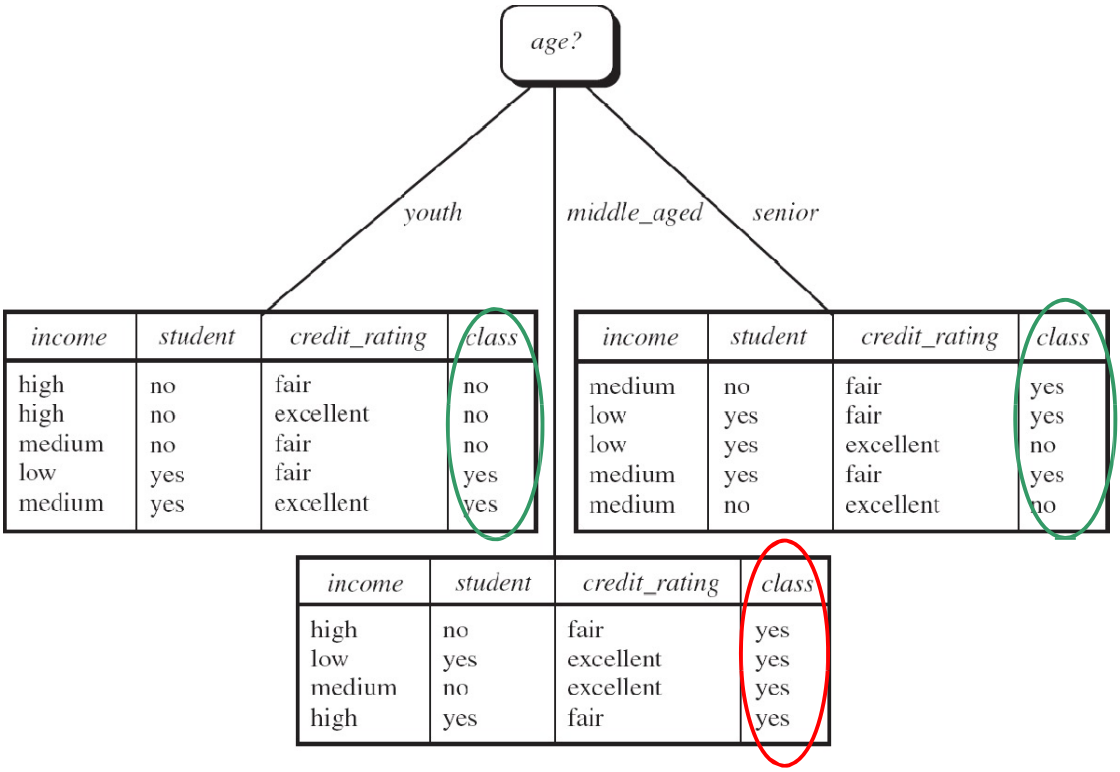


Figure The attribute *age* has the highest information gain and therefore becomes the splitting attribute at the root node of the decision tree. Branches are grown for each outcome of age. The tuples are shown partitioned accordingly.

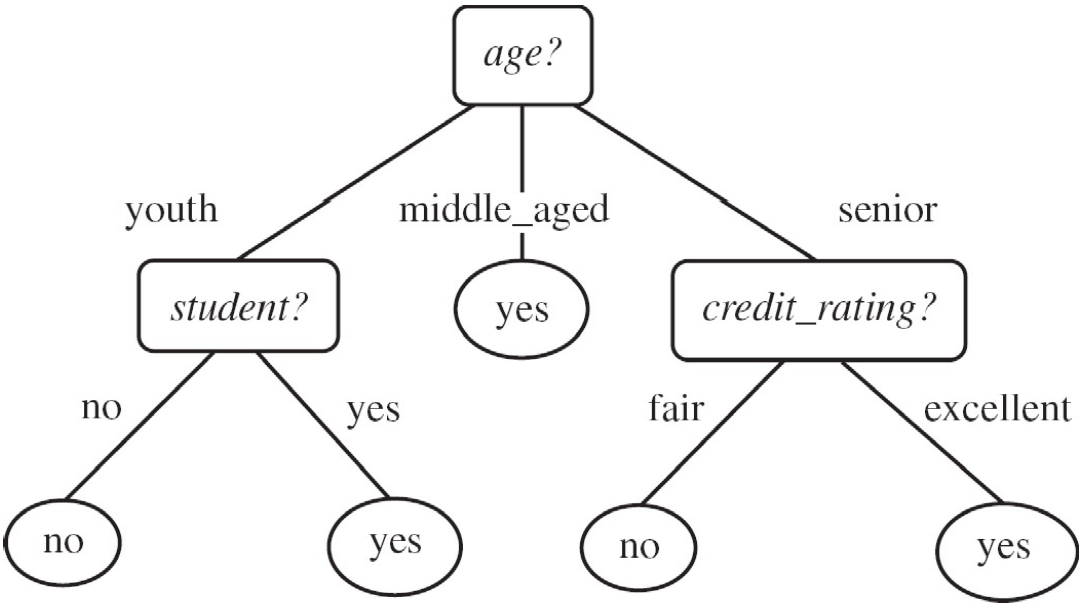


Figure A decision tree for the concept *buys_computer*, indicating whether a customer at *AllElectronics* is likely to purchase a computer. Each internal (nonleaf) node represents a test on an attribute. Each leaf node represents a class (either *buys_computer* = *yes* or *buy_computers* = *no*.

Exercise

Construct a decision tree to classify "golf play."

Weather and Possibility of Golf Play				
Weather	Temperature	Humidity	Wind	Golf Play
fine	hot	high	none	no
fine	hot	high	few	no
cloud	hot	high	none	yes
rain	warm	high	none	yes
rain	cold	midiam	none	yes
rain	cold	midiam	few	no
cloud	cold	midiam	few	yes
fine	warm	high	none	no
fine	cold	midiam	none	yes
rain	warm	midiam	none	yes
fine	warm	midiam	few	yes
cloud	warm	high	few	yes
cloud	hot	midiam	none	yes
rain	warm	high	few	no

Instance Language for Classification

- **Example: "is it a good day to play golf?"**
  - a set of attributes and their possible values:

outlook	sunny, overcast, rain
temperature	cool, mild, hot
humidity	high, normal
windy	true, false

A particular *instance* in the training set might be:

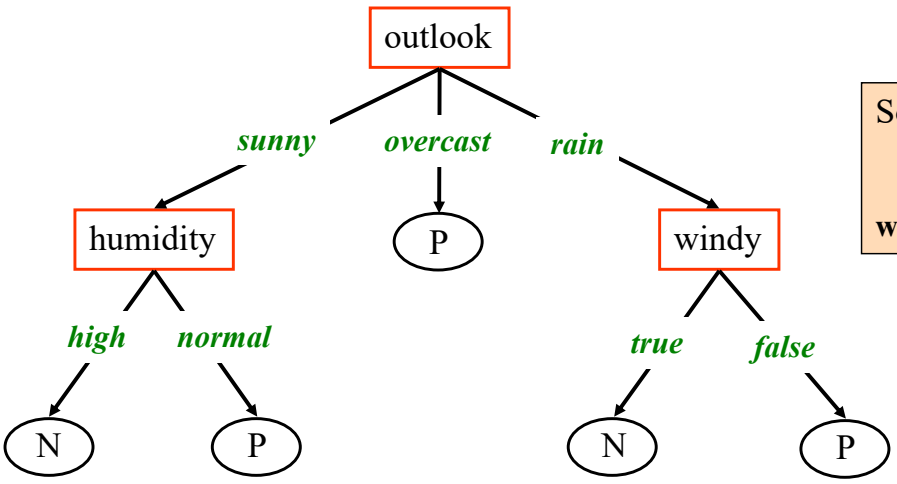
<overcast, hot, normal, false>: play

In this case, the target class is a binary attribute, so each instance represents a positive or a negative example.

Outlook	Tempreature	Humidity	Windy	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

Using Decision Trees for Classification

- **Examples can be classified as follows**
  - 1. look at the example's value for the feature specified
  - 2. move along the edge labeled with this value
  - 3. if you reach a leaf, return the label of the leaf
  - 4. otherwise, repeat from step 1
- **Example (a decision tree to decide whether to go on a picnic):**

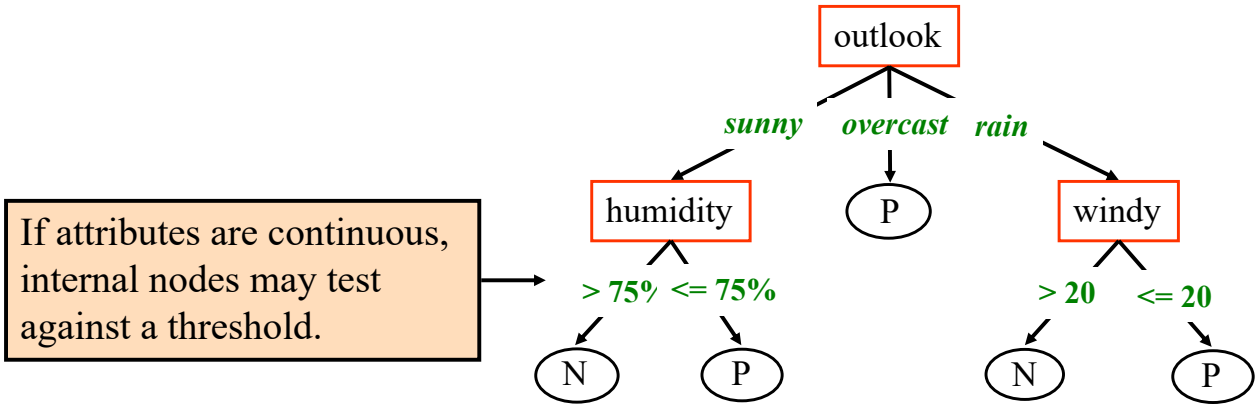


So a new instance:

<rainy, hot, normal, true>: ?

will be classified as "noplay"

Decision Trees and Decision Rules



If attributes are continuous, internal nodes may test against a threshold.

Each path in the tree represents a decision rule:

Rule1:
If (outlook="sunny") AND (humidity<=0.75)
Then (play="yes")

Rule2:
If (outlook="rainy") AND (wind>20)
Then (play="no")

Rule3:
If (outlook="overcast")
Then (play="yes")

• • •

Decision Tree Algorithm

- **Hunt's Algorithm**
- **ID3, J48, C4.5 (Based on Entropy Calculation)**
- **SLIQ,SPRINT,CART (Based on Gini-Index)**

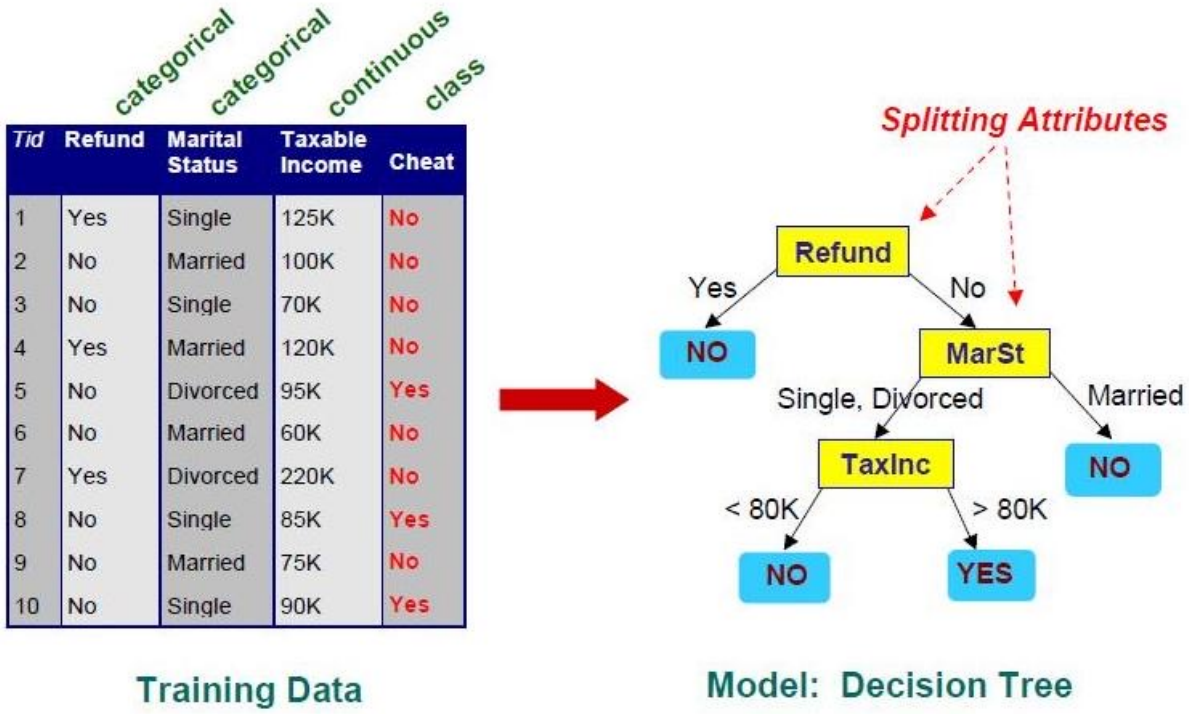
- **Decision Tree Algorithm**
  - **Hunt's** Algorithm
  - **ID3**, J48, C4.5 (Based on Entropy Calculation)
  - SLIQ,SPRINT,CART (Based on Gini-Index)

Hunt's Algorithm

- Hunt's algorithm grows a decision tree in a recursive fashion by partitioning the training data into successively into subsets.
- Let Dt be the set of training data that reach a node 't'. The general recursive procedure is defined as:
  - If Dt contains records that belong the same class yt, then t is a leaf node labeled as yt.
  - If Dt is an empty set, then t is a leaf node labeled by the default class, yd
  - If Dt contains records that belong to more than one class, use an attribute test to split the data into smaller subsets.

- - It recursively applies the procedure to each subset until all the records in the subset belong to the same class.
- - The Hunt's algorithm assumes that each combination of attribute sets has a unique class label during the procedure.
- - If all the records associated with Dt have identical attribute values except for the class label, then it is not possible to split these records any future. In this case, the node is declared a leaf node with the same class label as the majority class of training records associated with this node.

Example:



Tree Induction:

Tree induction is based on Greedy Strategy i.e. split the records based on an attribute test that optimize certain criterion.

Issues:

1.How to split the record?

2.How to specify the attribute test condition?

•- Depends on attribute types and number of ways to split the record i.e. 2-ways split /multi- way split.

•- Depends upon attribute types. (Nominal, Ordinal, Continuous)

3. **When to stop splitting?**

- When all records are belongs to the same class or all records have similar attributes.

4. **How to determine the best split?**

•- Nodes with homogenous class distribution are preferred.

•- Measure the node impurity.

- Gain / Gain Ratio : Select maximum
- Gini-Index : Select minimum

Entropy

Entropy H(S)is a measure of the amount of uncertainty in the dataset (S)(i.e. entropy characterizes the dataset (S)).

$$H(S) = -\sum_{x \in X} p(x) \log_2 p(x)$$

where,

S = The current dataset for which entropy is being calculated (changes every iteration of the ID3 algorithm)

X = Set of classes in S

P(x) - The probability of each set S

- When H(S) = 0, the set S is perfectly classified (i.e. all elements in S ar of the same class).
- In ID3, entropy is calculated for each remaining attribute. The attribute with
the smallest entropy is used to split the set S on this iteration.
- The higher the entropy, the higher the potential to improve the classification here.

Information Gain

Information gain is the measure of the difference in entropy from before to after the set S is split on an attribute A.

In other words, how much uncertainty in dataset (S) was reduced after splitting dataset S on attribute A.

$$IG(A, S) = H(S) - \sum_{t \in T} p(t) H(t)$$

Where,

•H(S) - Entropy of dataset S

•T - The subsets created from splitting dataset S by attribute A.

•P(t) - The probability of class t

•H(t) - Entropy of subset t

In ID3, information gain can be calculated (instead of entropy) for each attribute.

The attribute with the largest information gain is used to split the set on particular iteration.

ID3 Algorithm

- - The ID3 algorithm begins with the original dataset as the root node.
- - On each iteration of the algorithm, it iterates through every unused attribute of the
- dataset and calculates the entropy (or information gain) of that attribute.
- - It then selects the attribute which has the smallest entropy (or largest information gain)
- value.
- - The dataset is then split by the selected attribute to produce subsets of the data.
- - The algorithm continues to recur on each subset, considering only attributes never
- selected before. Recursion on a subset may stop in one of these cases:

ID3 Algorithm

i. **Every element in the subset belongs to the same class , then the node is turned into a leaf and labeled with the class of the examples.**

ii. **If the examples do not belong to the same class :**

   i. Calculate entropy and hence information gain to select the best node to split data.

   ii. Partition the data into subset.

iii. **Recursively repeat until all data are correctly classified**

Throughout the algorithm, the decision tree is constructed with each non-terminal node representing the selected attribute on which the data was split, and terminal nodes representing the class label of the final subset of this branch.

Trees Construction Algorithm (ID3)

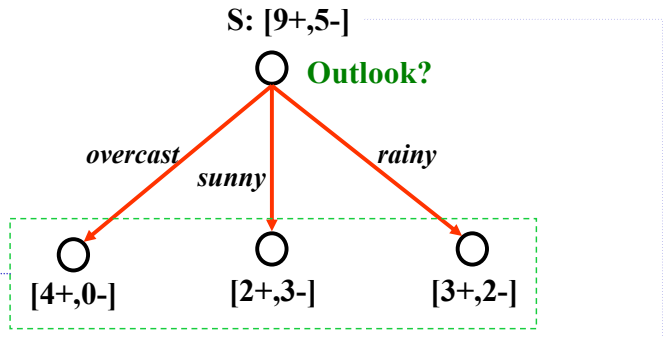
- **Decision Tree Learning Method (ID3)**
  - Input: a set of training instances ***S***, a set of features ***F***
  - 1. If every element of ***S*** has a class value "yes", return "yes"; if every element of ***S*** has class value "no", return "no"
  - 2. Otherwise, choose the **best feature** ***f*** from ***F*** (if there are no features remaining, then return failure);
  - 3. Extend tree from ***f*** by adding a new branch for each attribute value of ***f***
    - 3.1. Set ***F'*** = ***F*** – {***f***},
  - 4. Distribute training instances to leaf nodes (so each leaf node ***n*** represents the subset of examples S_n of ***S*** with the corresponding attribute value
  - 5. Repeat steps 1-5 for each leaf node ***n*** with S_n as the new set of training instances and ***F'*** as the new set of attributes

Note: ID3 algorithm only deals with categorical attributes, but can be extended(as in C4.5) to handle continuous attributes

Attribute Selection - Example

- **The "Golf" example: what attribute should we choose as the root?**

Day	outlook	temp	humidity	wind	play
D1	sunny	hot	high	weak	*No*
D2	sunny	hot	high	strong	*No*
D3	overcast	hot	high	weak	*Yes*
D4	rain	mild	high	weak	*Yes*
D5	rain	cool	normal	weak	*Yes*
D6	rain	cool	normal	strong	*No*
D7	overcast	cool	normal	strong	*Yes*
D8	sunny	mild	high	weak	*No*
D9	sunny	cool	normal	weak	*Yes*
D10	rain	mild	normal	weak	*Yes*
D11	sunny	mild	normal	strong	*Yes*
D12	overcast	mild	high	strong	*Yes*
D13	overcast	hot	normal	weak	*Yes*
D14	rain	mild	high	strong	*No*



H(9,5) = -(9/14).log(9/14) - (5/14).log(5/14)
= 0.94

H(4,0) = -(4/4).log(4/4) - (0/4).log(0/4)
= 0

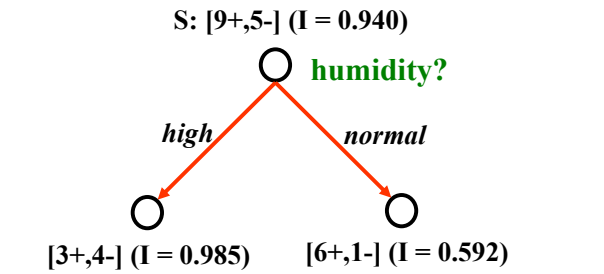
H(2,3) = -(2/5).log(2/5) - (3/5).log(3/5)
= 0.97

H(3,2) = -(3/5).log(3/5) - (2/5).log(2/5)
= 0.97

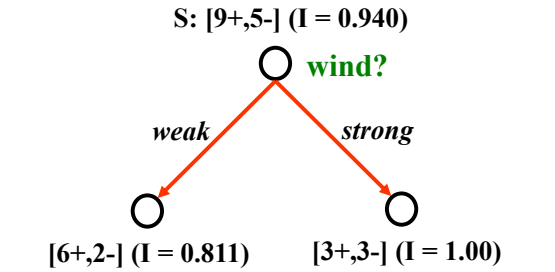
Gain(outlook) = .94 - (4/14)*0
- (5/14)*.97
- (5/14)*.97
= .24

Attribute Selection - Example (Cont.)

Day	outlook	temp	humidity	wind	play
D1	sunny	hot	high	weak	No
D2	sunny	hot	high	strong	No
D3	overcast	hot	high	weak	Yes
D4	rain	mild	high	weak	Yes
D5	rain	cool	normal	weak	Yes
D6	rain	cool	normal	strong	No
D7	overcast	cool	normal	strong	Yes
D8	sunny	mild	high	weak	No
D9	sunny	cool	normal	weak	Yes
D10	rain	mild	normal	weak	Yes
D11	sunny	mild	normal	strong	Yes
D12	overcast	mild	high	strong	Yes
D13	overcast	hot	normal	weak	Yes
D14	rain	mild	high	strong	No



Gain(humidity) = .940 - (7/14)*.985 - (7/14)*.592 = .151

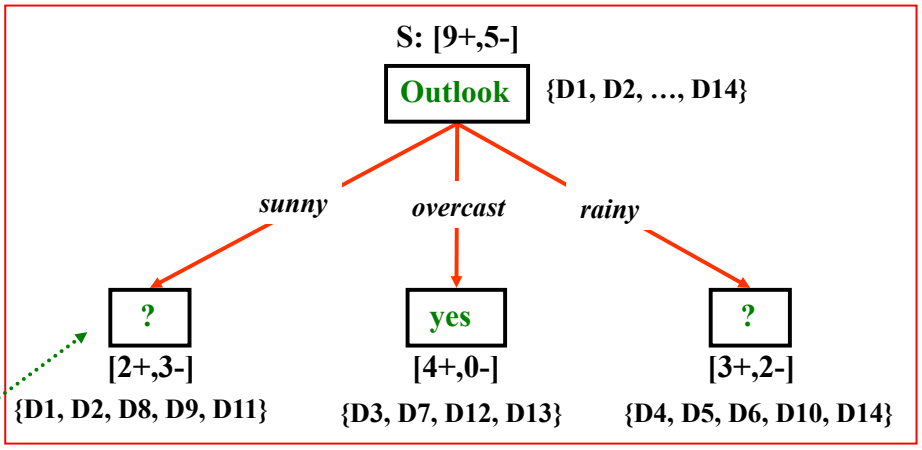


Gain(wind) = .940 - (8/14)*.811 - (8/14)*1.0 = .048

So, classifying examples by humidity provides more information gain than by wind. Similarly, we must find the information gain for "temp". In this case, however, you can verify that outlook has largest information gain, so it'll be selected as root

Attribute Selection - Example (Cont.)

- Partially learned decision tree



- which attribute should be tested here?

S_{sunny} = {D1, D2, D8, D9, D11}

Gain(S_{sunny}, humidity) = .970 - (3/5)*0.0 - (2/5)*0.0 = .970

Gain(S_{sunny}, temp) = .970 - (2/5)*0.0 - (2/5)*1.0 - (1/5)*0.0 = .570

Gain(S_{sunny}, wind) = .970 - (2/5)*1.0 - (3/5)*.918 = .019

Other Attribute Selection Measures

- **Gain ratio: (similar to Gain: maximum is selected)**
  - Information Gain measure tends to be biased in favor attributes with a large number of values
  - Gain Ratio normalizes the Information Gain with respect to the total entropy of all splits based on values of an attribute
  - Used by C4.5 (the successor of ID3)
  - But, tends to prefer unbalanced splits (one partition much smaller than others)
- **Gini index: (minimum is selected)**
  - A measure of ***impurity*** (based on relative frequencies of classes in a set of instances)
    - The attribute that provides the smallest Gini index (or the largest reduction in impurity due to the split) is chosen to split the node
  - Possible Problems:
    - Biased towards multivalued attributes; similar to Info. Gain.
    - Has difficulty when # of classes is large

Gain Ratio for Attribute Selection (C4.5)

- **Information gain measure is biased towards attributes with a large number of values**
- **C4.5 (a successor of ID3) uses gain ratio to overcome the problem (normalization to information gain)**

$$SplitInfo_A(D) = -\sum_{j=1}^{v} \frac{|D_j|}{|D|} \times \log_2\left(\frac{|D_j|}{|D|}\right)$$

  - GainRatio(A) = Gain(A)/SplitInfo(A)
- **Ex.**

$$SplitInfo_{income}(D) = -\frac{4}{14} \times \log_2\left(\frac{4}{14}\right) - \frac{6}{14} \times \log_2\left(\frac{6}{14}\right) - \frac{4}{14} \times \log_2\left(\frac{4}{14}\right) = 1.557$$

  - gain_ratio(income) = 0.029/1.557 = 0.019
- **The attribute with the maximum gain ratio is selected as the splitting attribute**

Gini Index (CART, IBM IntelligentMiner)

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

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

  where p_j is the relative frequency of class *j* in *D*

- If a data set *D* is split on A into two subsets D_1 and D_2, the *gini* index *gini*(*D*) is defined as

$$gini_A(D) = \frac{|D_1|}{|D|} gini(D_1) + \frac{|D_2|}{|D|} gini(D_2)$$

- Reduction in Impurity:

$$\Delta gini(A) = gini(D) - gini_A(D)$$

- The attribute provides the smallest $gini_{split}(D)$ (or the largest reduction in impurity) is chosen to split the node (*need to enumerate all the possible splitting points for each attribute*)

Computation of Gini Index

- Ex. D has 9 tuples in buys_computer = "yes" and 5 in "no"

$$gini(D) = 1 - \left(\frac{9}{14}\right)^2 - \left(\frac{5}{14}\right)^2 = 0.459$$

- Suppose the attribute income partitions D into 10 in D_1: {low, medium} and 4 in D_2

$$gini_{income \in \{low,medium\}}(D) = \left(\frac{10}{14}\right) Gini(D_1) + \left(\frac{4}{14}\right) Gini(D_1)$$

$$= \frac{10}{14}\left(1 - \left(\frac{7}{10}\right)^2 - \left(\frac{3}{10}\right)^2\right) + \frac{4}{14}\left(1 - \left(\frac{2}{4}\right)^2 - \left(\frac{2}{4}\right)^2\right)$$

$$= 0.443$$

$$= Gini_{income \in \{high\}}(D).$$

  $Gini_{\{low,high\}}$ is 0.458; $Gini_{\{medium,high\}}$ is 0.450. Thus, split on the {low,medium} and {high} since it has the **lowest Gini index**

- All attributes are assumed continuous-valued
- May need other tools, e.g., clustering, to get the possible split values
- Can be modified for categorical attributes

Comparing Attribute Selection Measures

- **The three measures, in general, return good results but**
  - **Information gain**:
    - biased towards multivalued attributes
  - **Gain ratio**:
    - tends to prefer unbalanced splits in which one partition is much smaller than the others
  - **Gini index**:
    - biased to multivalued attributes
    - has difficulty when # of classes is large
    - tends to favor tests that result in equal-sized partitions and purity in both partitions

Advantages of Decision Tree Classifier

1.– Inexpensive to construct
2.– Extremely fast at classifying unknown records
3.– Easy to interpret for small-sized trees
4.– Accuracy is comparable to other classification techniques for many simple data sets

Overfitting and Tree Pruning

- **Overfitting: An induced tree may overfit the training data**
  - Too many branches, some may reflect anomalies due to noise or outliers
  - Some splits or leaf nodes may be the result of decision based on very few instances, resulting in poor accuracy for unseen instances
  - 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 error rate going above a pre-specified threshold
    - Difficult to choose an appropriate threshold
  - Postpruning: *Remove branches* from a "fully grown" tree
    - Get a sequence of progressively pruned trees
    - Use a test data different from the training data to measure error rates
    - Select the "best pruned tree"

3. Classification (12 hours)

3.1 Basics and Algorithms

3.2 Decision Tree Classifier [humanoriented]

3.3 Rule Based Classifier

3.4 Nearest Neighbor Classifier

3.5 Bayesian Classifier

3.6 Artificial Neural Network Classifier

3.7 Issues : Overfitting, Validation, Model Comparison

3.3 Rule Based Classifier

•It classifies records by using a collection of "If ..... Then....." rules.
•A rule base classifier uses a set of "If ..... Then....." rules for classification.
•The 'If' part or left hand side of a rule is known as the rule **antecedent** or precondition where as the 'Then" part or right hand side is the rule **consequent**.
• In the rule antecedent, the condition consists of one or more attribute tests. (Eg: age=youth and student=yes -> loan=No)
•If the condition in a rule antecedent holds true for a given tuple, the rule **antecedent is satisfied** and that the rule **covers the tuple**.
•**Coverage** of a rule is the fraction of records that **satisfy the antecedent** of a rule.
Coverage (R) = **Ncovers / D**
Where,
Ncovers = number of tuples/record covered by rule R (i.e. number of tuples/record that can be classified by the rule; Simply tuples which satisfy antecedent).
D = number of tuples in total data set.

Accuracy of a rule is fraction of records that satisfy both the antecedent and consequent of a rule.
Accuracy = **Ncorrect / Ncovers**
Where,
Ncorrect = Number of records that are correctly classified by the rule
Ncovers = Number of record that can be cover/classified by the rule

How does Rule-Based Classifier work?

•- If a rule is satisfied by a tuple, the rule is said to be triggered. Triggering doesn't always mean firing because there may be more than one rules that can be satisfied.
•Three different cases occur for classification.
•Case-I: If only one rule is satisfied
•Case-II: If more than one rules are satisfied
•Case-III: If no rule is satisfied

•3 different cases occur for classification.

•Case-I: If only one rule is satisfied

- When any instances is covered by only one rule then the rule fires by returning the class prediction for the tuple defined by the rule.

Case-II: If more than one rules are satisfied

•If more than one rules are triggered, we need a conflict resolution strategy to find which rule is fired.

•- Rule ordering or rule ranking or rule priority can be set in case of rules conflict. A rule ordering may be class-based or rule-based.

•- Rule-based ordering: Individual rules are ranked based on their quality.

•Class-based ordering: Rules that belong to the same class appear together

•- When rule-based ordering is used, the rule set is known as a decision list.

•Case-III: If no rule is satisfied

- If any instance not triggered by any rule, use default class for classification. Mostly most frequent class is assigned as default class.

Using IF-THEN Rules for Classification

- Represent the knowledge in the form of IF-THEN rules

  R: IF *age* = youth AND *student* = yes THEN *buys_computer* = yes

  - Rule antecedent/precondition vs. rule consequent
- Assessment of a rule: *coverage* and *accuracy*
  - n_{covers} = # of tuples covered by R
  - $n_{correct}$ = # of tuples correctly classified by R

  coverage(R) = n_{covers} /|D| /* D: training data set */

  accuracy(R) = $n_{correct}$ / n_{covers}
- If more than one rule are triggered, need **conflict resolution**
  - Size ordering: assign highest priority to triggering rules that has the "toughest" requirement (i.e., with the *most attribute tests*)
  - Class-based ordering: decreasing order of *prevalence or misclassification cost per class*
  - Rule-based ordering (**decision list**): rules are organized into one long priority list, according to some measure of rule quality or by experts

Example:

S.No.	Name	Blood Type	Give Birth	Can fly	Live in water	Class
1	Lemur	Warm	Yes	No	No	?
2	Turtle	Cold	No	No	Sometimes	?
3	Shark	Cold	Yes	No	Yes	?

Rule base

R1: (Give Birth = No) ^ (Can fly = Yes) => Birds
R2: (Give Birth = No) ^ (Live in Water = Yes) => Fishes
R3: (Give Birth = Yes) ^ (Blood Type = Warm) => Mammals
R4: (Give Birth = No) ^ (Can fly = No) => Reptiles
R5: (Live in Water = Sometimes) => Amphibians

- In above example, R1 and R2 don't have any coverage. R3, R4 & R5 have coverage.

Characteristics of Rule-Based Classifier

Mutually exclusive Rules

a. Classifier contains mutually exclusive rules if all the rules are independent of each other.
b. Every record is covered by at most one rule. (then only it become mutually exclusive)
c. Rules are no longer mutually exclusive if a record may triggered by more than one rule.
d. To make mutually exclusive we apply rule ordering.

Exhaustive Rules

a. Classifier has exhaustive coverage if it accounts for every possible combination of attribute values (every possible rule).
b. Each record is covered by at least one rule.
c. Rules are no longer exhaustive if a record may not trigger any rules.
d. To make rules exhaustive use default class.

Building Classification Rules

- Two approaches are used to build classification rules.

A. Direct Method

- Extract rules **directly from data**. It is an inductive and sequential approach.

Sequential Covering

1. Start from an empty rule
2. Grow a rule using the Learn-One-Rule function
3. Remove training records covered by the rule
4. Repeat Step (2) and (3) until stopping criterion is met

Aspects of Sequential Covering

- Rule Growing
- Instance Elimination
- Rule Evaluation
- Stopping Criterion
- Rule Pruning

i. Rule Growing

a. CN2 Algorithm:

– Start from an empty conjunct: {}
– Add conjuncts that minimizes the entropy measure: {A}, {A,B}, ...
– Determine the rule consequent by taking majority class of instances covered by the rule

b. RIPPER Algorithm:

– Start from an empty rule: {} => class
– Add conjuncts that maximize FOIL's information gain measure:
R0: {} => class (initial rule)
R1: {A} => class (rule after adding conjunct)
Gain (R0, R1) = t [log (p1/(p1+n1)) – log (p0/(p0 + n0))]
•Where, t: number of positive instances covered by both R0 and R1
p0: number of positive instances covered by R0
n0: number of negative instances covered by R0
p1: number of positive instances covered by R1
n1: number of negative instances covered by R1

ii. Instance Elimination

- We need to eliminate instances otherwise, the next rule is identical to previous rule.
- We remove positive instances to ensure that the next rule is different.
- We remove negative instances to prevent underestimating accuracy of rule

iii.Rule Evaluation

Accuracy = n_c/n

n: Number of instances
n_c: Nimber of instances covered by rule

iv. Stopping Criterion and Rule Pruning Stopping criterion

– Compute the gain

– If gain is not significant, discard the new rule.

v. Rule Pruning

– Similar to post-pruning of decision trees.

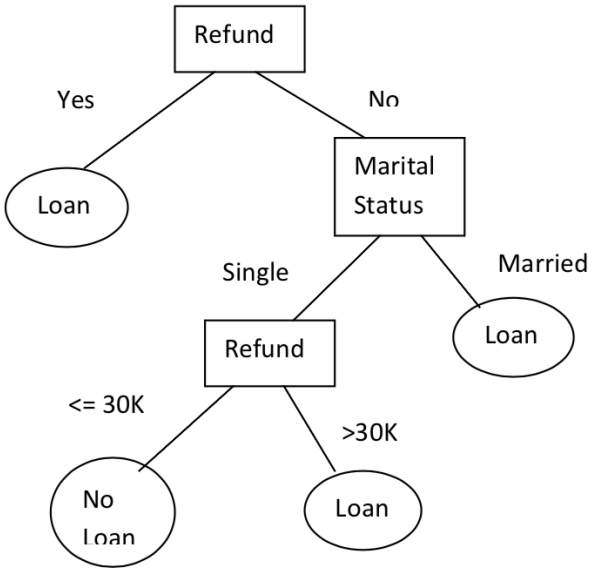
– Reduced Error Pruning:

- Remove one of the conjuncts in the rule
- Compare error rate on validation set before and after pruning
- If error improves, prune the conjunct

B. Indirect Method:

<- Extract rules from other classification models (e.g. decision trees, neural networks, etc).

Eg; Rule Extraction from Decision Tree



Rules:

R1: (Refund = Yes) => Loan

R2: (Refund = No) ^ (Marital Status = Married) => Loan

Rule simplification

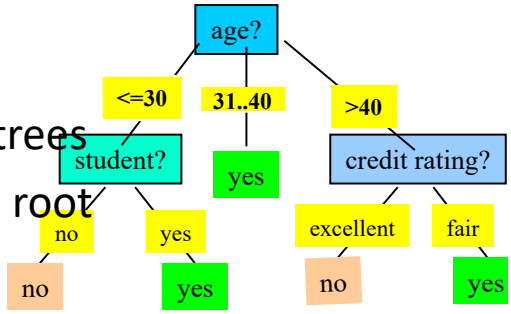
Complex rules can be simplified. In above example R2 can be simplified as:

r2: (Marital Status = Married) => Loan

Advantages of Rule-Based Classifiers

- As highly expressive as decision trees
- Easy to interpret
- Easy to generate
- Can classify new instances rapidly
- Performance comparable to decision trees

Rule Extraction from a Decision Tree



- Rules are *easier to understand* than large trees
- 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 holds the class prediction
- Rules are mutually exclusive and exhaustive

- Example: Rule extraction from our *buys_computer* decision-tree

IF *age* = young AND *student* = *no* THEN *buys_computer* = *no*

IF *age* = young AND *student* = *yes* THEN *buys_computer* = *yes*

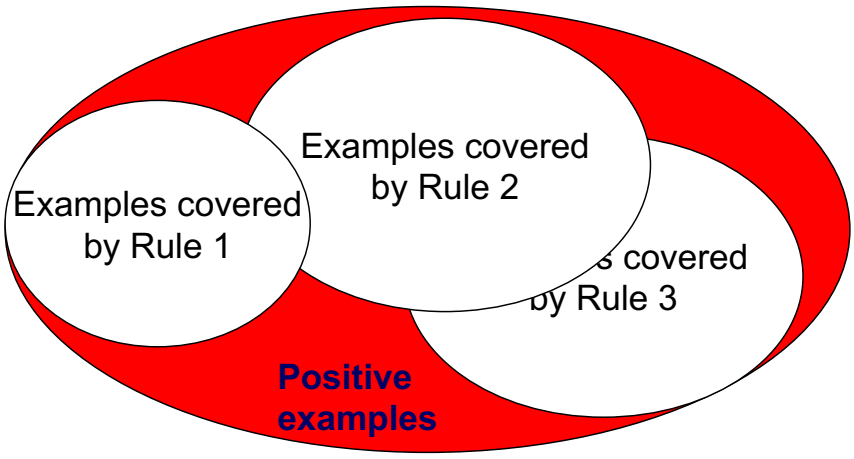
IF *age* = mid-age THEN *buys_computer* = *yes*

IF *age* = old AND *credit_rating* = *excellent* THEN *buys_computer* = *no*

IF *age* = old AND *credit_rating* = *fair* THEN *buys_computer* = *yes*

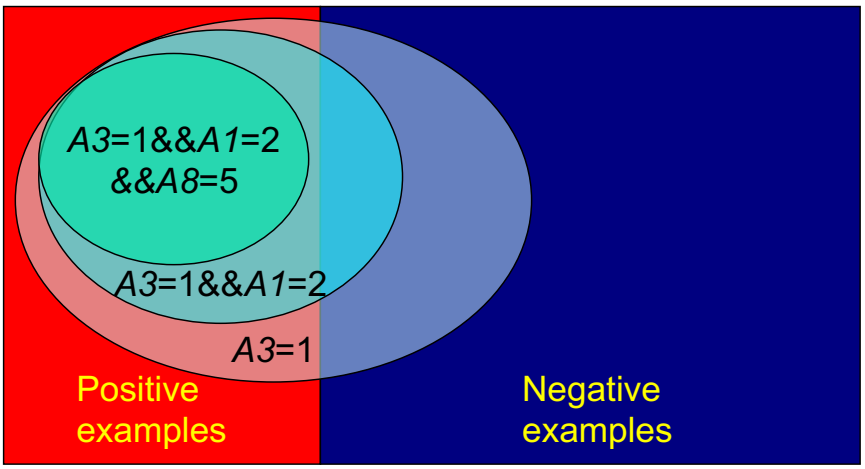
Sequential Covering Algorithm

while (enough target tuples left)
    generate a rule
    remove positive target tuples satisfying this rule



Rule Generation

- To generate a rule
  while(true)
    find the best predicate *p*
    if foil-gain(*p*) > threshold **then** add *p* to current rule
    else break



How to Learn-One-Rule?

- Start with the *most general rule* possible: condition = empty
- *Adding new attributes* by adopting a greedy depth-first strategy
  - Picks the one that most improves the rule quality
- Rule-Quality measures: consider both coverage and accuracy
  - Foil-gain (in FOIL & RIPPER): assesses info_gain by extending condition

$$FOIL\_Gain = pos' \times (\log_2 \frac{pos'}{pos'+neg'} - \log_2 \frac{pos}{pos+neg})$$

    - favors rules that have high accuracy and cover many positive tuples
- Rule pruning based on an independent set of test tuples

$$FOIL\_Prune(R) = \frac{pos - neg}{pos + neg}$$

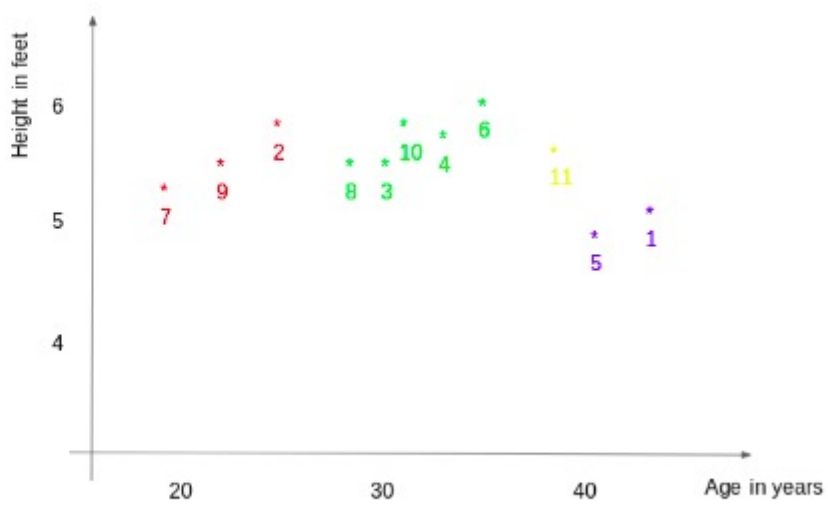
Pos/neg are # of positive/negative tuples covered by R.

If *FOIL_Prune* is higher for the pruned version of R, prune R

3.4 Nearest Neighbor Classifier

Consider the height and age for 11 people. On the basis of given features ('Age' and 'Height'), the table can be represented in a graphical format as shown below:

ID	Age	Height	Weight
1	45	5	77
2	26	5.11	47
3	30	5.6	55
4	34	5.9	59
5	40	4.8	72
6	36	5.8	60
7	19	5.3	40
8	28	5.8	60
9	23	5.5	45
10	32	5.6	58
11	38	5.5	?



SOLUTION:

ID	Height	Age	Weight
1	5	45	77
5	4.8	40	72
6	5.8	36	60

the weight for ID#11 would be = (77+72+60)/3 = 69.66 kg.

Instance Based Classifier

- It Store the training records and use training records to predict the class label of unseen cases.
- i. Rote-learner
- Memorizes entire training data and performs classification only if attributes of record match one of the training examples exactly
- ii. Nearest neighbor –
- Uses k "closest" points (nearest neighbors) for performing classification. K-closet neighbor of a record 'X' are data points that have the K-smallest distance of 'X'.
- Classification based on learning by analogy i.e. by comparing a given test tuple with training tuple that are similar to it.
- Training tuples are described by n-attributes.
- When given an unknown tuple, a k-nearest- neighbor classifier searches the pattern space for the k-training tuples that are closest to the unknown tuple.

- Nearest neighbor classifier requires 3 things:
  a) Set of stored records (because implicit method)
  b) Distance matric to compute distance between records. For distance calculation any standard approach can be used such as Euclidean distance.
  c) The value of 'K', the number of nearest neighbor to retrieve.

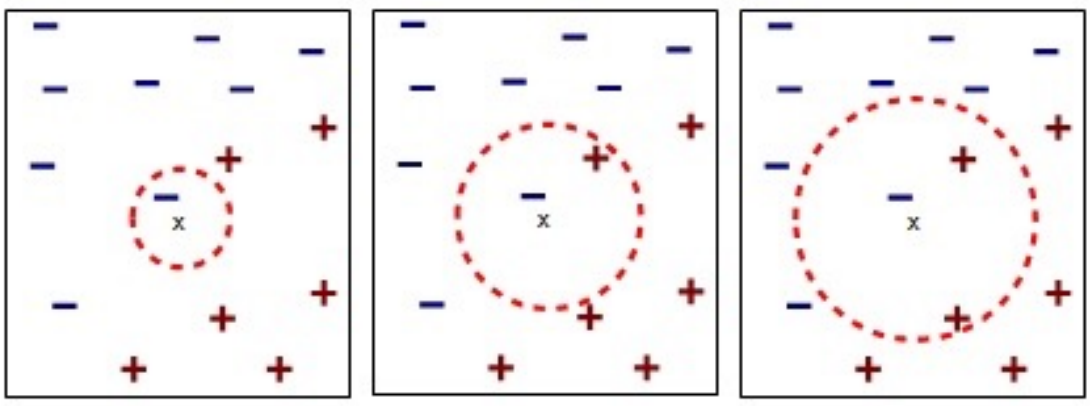
Algorithm/steps

-To classify the unknown records

-Compute distance to other training records. (NOTE: No test records)

-Identify the k-nearest neighbor.

-Use class label nearest neighbors to determine the class label of unknown record. In case of conflict, use majority vote for classification.

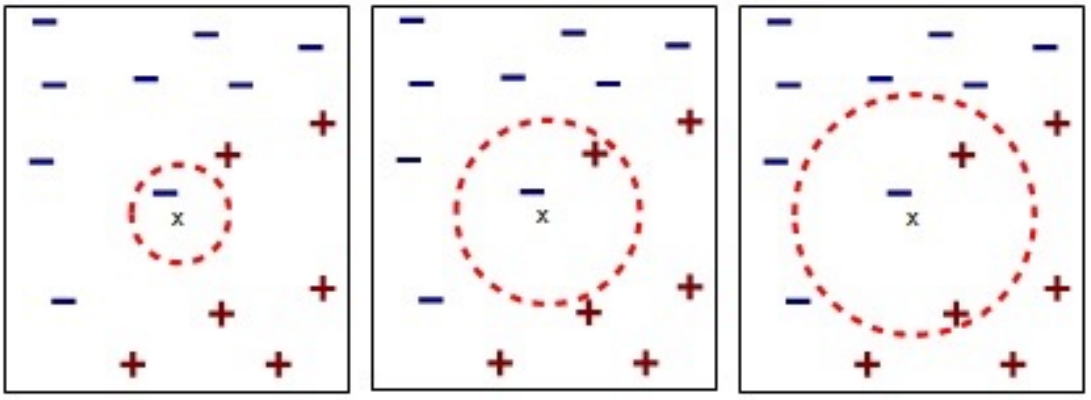


(a) 1-nearest neighbor (b) 2-nearest neighbor (c) 3-nearest neighbor

Issues of classification using k-nearest neighbor classification

i. Choosing the value of K

- One of challenge in classification is to choose the appropriate value of K. If K is too small, it is sensitive to noise points. If K is too large, neighbor may include points from other classes.
- With the change of value of K, the classification result may vary.



(a) 1-nearest neighbor (b) 2-nearest neighbor (c) 3-nearest neighbor

ii. Scaling Issue

- Attribute may have to be scaled to prevent distance measure from being dominated by one of attributes.
- Eg. Height varies from 1.5m to 1.8m, Weight vary from 20kg to 80kg etc.

iii. Distance computing for non-numeric data.

- Use Distance as 0 for the same data and maximum possible distance for different data.

iv. Missing values

- Use maximum possible distance

v. Knn are lazy learners

- It does not build model explicitly
- Classifying unknown records are relatively expensive

Disadvantages:

- Poor accuracy when data have noise and irrelevant attributes.
- Slow when classifying test tuples.(NOTE: fast in tree classifier)
- Classifying unknown records are relatively expensive

Distance and Similarity Measures

Distance or Similarity Measures

- **Many data mining and analytics tasks involve the comparison of objects and determining in terms of their similarities (or dissimilarities)**
  - Clustering
  - Nearest-neighbor search, classification, and prediction
  - Characterization and discrimination
  - Automatic categorization
  - Correlation analysis
- **Many of todays real-world applications rely on the computation similarities or distances among objects**
  - Personalization
  - Recommender systems
  - Document categorization
  - Information retrieval
  - Target marketing

Similarity and Dissimilarity

- **Similarity**
  - Numerical measure of how alike two data objects are
  - Value is higher when objects are more alike
  - Often falls in the range [0,1]
- **Dissimilarity (e.g., distance)**
  - Numerical measure of how different two data objects are
  - Lower when objects are more alike
  - Minimum dissimilarity is often 0
  - Upper limit varies
- **Proximity refers to a similarity or dissimilarity**

Distance or Similarity Measures

- **Measuring Distance**
  - In order to group similar items, we need a way to measure the distance between objects (e.g., records)
  - Often requires the representation of objects as "feature vectors"

An Employee DB

ID	Gender	Age	Salary
1	F	27	19,000
2	M	51	64,000
3	M	52	100,000
4	F	33	55,000
5	M	45	45,000

Feature vector corresponding to Employee 2: <M, 51, 64000.0>

Term Frequencies for Documents

	T1	T2	T3	T4	T5	T6
Doc1	0	4	0	0	0	2
Doc2	3	1	4	3	1	2
Doc3	3	0	0	0	3	0
Doc4	0	1	0	3	0	0
Doc5	2	2	2	3	1	4

Feature vector corresponding to Document 4: <0, 1, 0, 3, 0, 0>

Distance or Similarity Measures

- **Properties of Distance Measures:**
  - for all objects A and B, $dist(A, B) \geq 0$, and $dist(A, B) = dist(B, A)$
  - for any object A, $dist(A, A) = 0$
  - $dist(A, C) \leq dist(A, B) + dist(B, C)$

- **Representation of objects as vectors:**
  - Each data object (item) can be viewed as an n-dimensional vector, where the dimensions are the attributes (features) in the data
  - Example (employee DB): Emp. ID 2 = <M, 51, 64000>
  - Example (Documents): DOC2 = <3, 1, 4, 3, 1, 2>
  - The vector representation allows us to compute distance or similarity between pairs of items using standard vector operations, e.g.,
    - Cosine of the angle between vectors
    - Manhattan distance
    - Euclidean distance
    - Hamming Distance

Data Matrix and Distance Matrix

- **Data matrix**
  - Conceptual representation of a table
    - Cols = features; rows = data objects
  - *n* data points with *p* dimensions
  - Each row in the matrix is the vector representation of a data object

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

- **Distance (or Similarity) Matrix**
  - *n* data points, but indicates only the pairwise distance (or similarity)
  - A triangular matrix
  - Symmetric

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

Common Distance Measures for Numeric Data

- **Consider two vectors**
  - Rows in the data matrix

$$X = \langle x_1, x_2, \cdots, x_n \rangle \qquad Y = \langle y_1, y_2, \cdots, y_n \rangle$$

- **Common Distance Measures:**
  - Manhattan distance:

$$dist(X,Y) = |x_1 - y_1| + |x_2 - y_2| + \cdots + |x_n - y_n|$$

  - Euclidean distance:

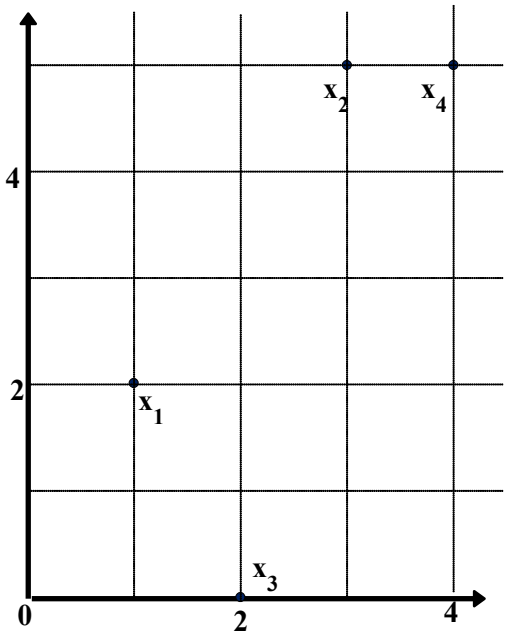
$$dist(X,Y) = \sqrt{(x_1 - y_1)^2 + \cdots + (x_n - y_n)^2}$$

  - Distance can be defined as a dual of a similarity measure

$$dist(X,Y) = 1 - sim(X,Y)$$

$$sim(X,Y) = \frac{\sum_i (x_i \times y_i)}{\sqrt{\sum_i x_i^2 \times \sum_i y_i^2}}$$

Example: Data Matrix and Distance Matrix



Data Matrix

point	attribute1	attribute2
x1	1	2
x2	3	5
x3	2	0
x4	4	5

Distance Matrix (Manhattan)

	x1	*x2*	*x3*	*x4*
x1	0			
x2	5	0		
x3	3	6	0	
x4	6	1	7	0

Distance Matrix (Euclidean)

	x1	*x2*	*x3*	*x4*
x1	0			
x2	3.61	0		
x3	2.24	5.1	0	
x4	4.24	1	5.39	0

Distance on Numeric Data: Minkowski Distance

- **Minkowski distance: A popular distance measure**

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

  - 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 h is the order (the distance so defined is also called L-*h* norm)
- **Note that Euclidean and Manhattan distances are special cases**
  - $h = 1$: (L_1 norm) Manhattan distance

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

  - $h = 2$: (L_2 norm) Euclidean distance

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

Vector-Based Similarity Measures

- **In some situations, distance measures provide a skewed view of data**
  - E.g., when the data is very sparse and 0's in the vectors are not significant
  - In such cases, typically vector-based similarity measures are used
  - Most common measure: Cosine similarity
  - the cosine similarity is:

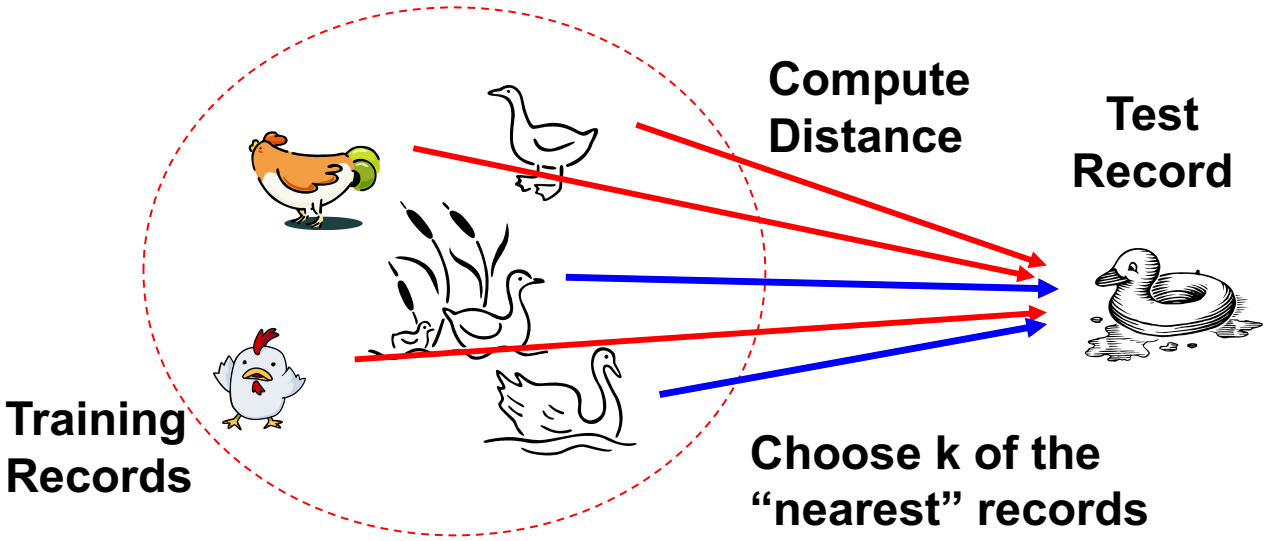
$$sim(X,Y) = \frac{X \bullet Y}{\|X\| \times \|y\|} = \frac{\sum_i (x_i \times y_i)}{\sqrt{\sum_i x_i^2} \times \sqrt{\sum_i y_i^2}}$$

Distance-Based Classification

- **Basic Idea: classify new instances based on their similarity to or distance from instances we have seen before**
  - also called "instance-based learning"
- **Simplest form of MBR: *Rote Learning***
  - learning by memorization
  - save all previously encountered instance; given a new instance, find one from the memorized set that most closely "resembles" the new one; assign new instance to the same class as the "nearest neighbor"
  - more general methods try to find *k* nearest neighbors rather than just one
  - but, how do we define "resembles?"
- **MBR is "lazy"**
  - defers all of the real work until new instance is obtained; no attempt is made to learn a generalized model from the training set
  - less data preprocessing and model evaluation, but more work has to be done at classification time

Nearest Neighbor Classifiers

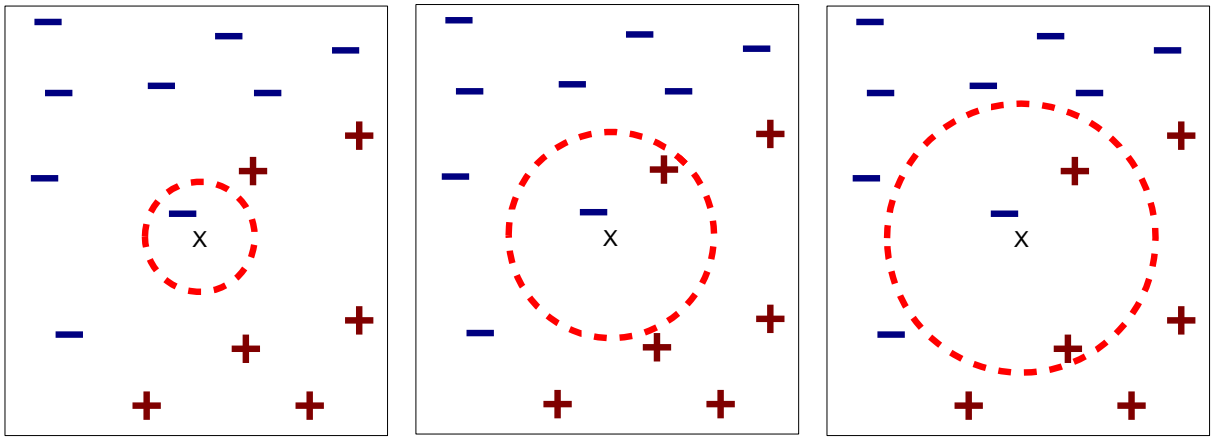
- **Basic idea:**
  - If it walks like a duck, quacks like a duck, then it's probably a duck



K-Nearest-Neighbor Strategy

- **Given object *x*, find the *k* most similar objects to *x***
  - The *k* nearest neighbors
  - Variety of distance or similarity measures can be used to identify and rank neighbors
  - Note that this requires comparison between *x* and all objects in the database
- **Classification:**
  - Find the class label for each of the *k* neighbor
  - Use a voting or weighted voting approach to determine the majority class among the neighbors (a combination function)
    - Weighted voting means the closest neighbors count more
  - Assign the majority class label to *x*
- **Prediction:**
  - Identify the value of the target attribute for the *k* neighbors
  - Return the weighted average as the predicted value of the target attribute for *x*

K-Nearest-Neighbor Strategy



(a) 1-nearest neighbor (b) 2-nearest neighbor (c) 3-nearest neighbor

K-nearest neighbors of a record x are data points that have the k smallest distance to x

Combination Functions

- **Voting: the "democracy" approach**
  - poll the neighbors for the answer and use the majority vote
  - the number of neighbors (*k*) is often taken to be odd in order to avoid ties
    - works when the number of classes is two
    - if there are more than two classes, take k to be the number of classes plus 1

- **Impact of *k* on predictions**
  - in general different values of *k* affect the outcome of classification
  - we can associate a confidence level with predictions (this can be the % of neighbors that are in agreement)
  - problem is that no single category may get a majority vote
  - if there is strong variations in results for different choices of *k*, this an indication that the training set is not large enough

Voting Approach - Example

Will a new customer respond to solicitation?

ID	Gender	Age	Salary	Respond?
1	F	27	19,000	no
2	M	51	64,000	yes
3	M	52	105,000	yes
4	F	33	55,000	yes
5	M	45	45,000	no
new	F	45	100,000	?

Using the voting method without confidence

	Neighbors	Answers	k =1	k = 2	k = 3	k = 4	k = 5
D_man	4,3,5,2,1	Y,Y,N,Y,N	yes	yes	yes	yes	yes
D_euclid	4,1,5,2,3	Y,N,N,Y,Y	yes	?	no	?	yes

Using the voting method with a confidence

	k =1	k = 2	k = 3	k = 4	k = 5
D_man	yes, 100%	yes, 100%	yes, 67%	yes, 75%	yes, 60%
D_euclid	yes, 100%	yes, 50%	no, 67%	yes, 50%	yes, 60%

KNN for Document Categorization

	T1	T2	T3	T4	T5	T6	T7	T8	Cat
DOC1	2	0	4	3	0	1	0	2	Cat1
DOC2	0	2	4	0	2	3	0	0	Cat1
DOC3	4	0	1	3	0	1	0	1	Cat2
DOC4	0	1	0	2	0	0	1	0	Cat1
DOC5	0	0	2	0	0	4	0	0	Cat1
DOC6	1	1	0	2	0	1	1	3	Cat2
DOC7	2	1	3	4	0	2	0	2	Cat2
DOC8	3	1	0	4	1	0	2	1	?

KNN for Document Categorization

Using Cosine Similarity to find K=3 neighbors:

	T1	T2	T3	T4	T5	T6	T7	T8	Norm	Sim(D8,Di)
DOC1	2	0	4	3	0	1	0	2	5.83	0.61
DOC2	0	2	4	0	2	3	0	0	5.74	0.12
DOC3	4	0	1	3	0	1	0	1	5.29	0.84
DOC4	0	1	0	2	0	0	1	0	2.45	0.79
DOC5	0	0	2	0	0	4	0	0	4.47	0.00
DOC6	1	1	0	2	0	1	1	3	4.12	0.73
DOC7	2	1	3	4	0	2	0	2	6.16	0.72
DOC8	3	1	0	4	1	0	2	1	5.66	

E.g.: Sim(D8,D7) = (D8 • D7) / (Norm(D8).Norm(D7)
= (3x2+1x1+0x3+4x4+1x0+0x2+2x0+1x2) / (5.66 x 6.16)
= 25 / 34.87 = 0.72

KNN for Document Categorization

	T1	T2	T3	T4	T5	T6	T7	T8	Cat	Sim(D8,Di)
DOC1	2	0	4	3	0	1	0	2	**Cat1**	0.61
DOC2	0	2	4	0	2	3	0	0	**Cat1**	0.12
DOC3	4	0	1	3	0	1	0	1	**Cat2**	0.84
DOC4	0	1	0	2	0	0	1	0	**Cat1**	0.79
DOC5	0	0	2	0	0	4	0	0	**Cat1**	0.00
DOC6	1	1	0	2	0	1	1	3	**Cat2**	0.73
DOC7	2	1	3	4	0	2	0	2	**Cat2**	0.72
DOC8	3	1	0	4	1	0	2	1	5.66	

- **Simple voting:**
  - Cat for DOC 8 = Cat2 with confidence 2/3 = 0.67
- **Weighted voting:**
  - Cat for DOC 8 = Cat2
  - Confidence: (0.84 + 0.73) / (0.84 + 0.79 + 0.73) = 0.66

Combination Functions

- **Weighted Voting: not so "democratic"**
  - similar to voting, but the vote some neighbors counts more
  - "shareholder democracy?"
  - question is which neighbor's vote counts more?

Advantage of weighted voting

-> introduces enough variation to prevent ties in most cases

-> helps distinguish between competing neighbors

- **How can weights be obtained?**
  - Distance-based
    - closer neighbors get higher weights
    - "value" of the vote is the inverse of the distance (may need to add a small constant)
    - the weighted sum for each class gives the combined score for that class
    - to compute confidence, need to take weighted average
  - Heuristic
    - weight for each neighbor is based on domain-specific characteristics of that neighbor

3.5 Bayesian Classifier

Classification: 3 Step Process

- **1. Model construction (Learning):**
  - Each record (instance, example) is assumed to belong to a predefined class, as determined by one of the attributes
    - This attribute is called the target attribute
    - The values of the target attribute are the **class labels**
  - The set of all instances used for learning the model is called **training set**
- **2. Model Evaluation (Accuracy):**
  - Estimate accuracy rate of the model based on a **test set**
  - The known labels of test instances are compared with the predicts class from model
  - Test set is independent of training set otherwise over-fitting will occur
- **3. Model Use (Classification):**
  - The model is used to classify unseen instances (i.e., to predict the class labels for new unclassified instances)
  - Predict the value of an actual attribute

- Bayesian classification is based on Baye's Theorem.
- It is a statistical classifier that predicts class membership probabilities such as the
probability that a given tuple belongs to a particular class.
Baye's Law
$P(A/B) = P(B/A)\ P(A)/\ P(B)$

1. Bayesian Belief Networks (Graphical Method)

•- Bayesian Belief Network specifies joint conditional probability distributions.

•- Bayesian Networks and Probabilistic Network are known as belief network.

•- It allows class conditional independencies to be defined between subsets of variables.

•- It provides a graphical model of causal relationship on which learning can be performed.

- It represents a set of random variables and their conditional dependencies via a directed acyclic graph

2. Naïve Bayesian Classifier

•- The Naive Bayes Classifier technique is based on the so-called Bayesian theorem and is particularly suited when the dimensionality of the inputs is high.

•- It simplifies the computational complexity.

•- Naïve Bayesian Classifier assumes that the effect of an attribute value on a given class is independent of the value of other attributes i.e. class conditional independence.

Bayesian Classification: Why?

- **A statistical classifier: performs *probabilistic prediction, i.e.,* predicts class membership probabilities**
- **Foundation: Based on Bayes' Theorem.**
- **Performance: A simple Bayesian classifier, *naïve Bayesian classifier*, has comparable performance with decision tree and selected neural network classifiers**
- **Incremental: Each training example can incrementally increase/decrease the probability that a hypothesis is correct — prior knowledge can be combined with observed data**
- **Standard: Even when Bayesian methods are computationally intractable, they can provide a standard of optimal decision making against which other methods can be measured**

Bayesian Theorem: Basics

- **Let X be a data sample ("*evidence*"): class label is unknown**
- **Let H be a *hypothesis* that X belongs to class C**
- **Classification is to determine P(H|X), (*posteriori probability),* the probability that the hypothesis holds given the observed data sample X**
- **P(H) (*prior probability*), the initial probability**
  - E.g., **X** will buy computer, regardless of age, income, …
- **P(X): probability that sample data is observed**
- **P(X|H) (likelyhood), the probability of observing the sample X, given that the hypothesis holds**
  - E.g., Given that **X** will buy computer, the prob. that X is 31..40, medium income

Bayesian Theorem

- **Given training data X, *posteriori probability of a hypothesis* H, P(H|X), follows the Bayes theorem**

$$P(H|\mathbf{X}) = \frac{P(\mathbf{X}|H)P(H)}{P(\mathbf{X})} = P(\mathbf{X}|H) \times P(H) / P(\mathbf{X})$$

- **Informally, this can be written as**

  posteriori = likelihood x prior/evidence

- **Predicts X belongs to C_2 iff the probability $P(C_i|X)$ is the highest among all the $P(C_k|X)$ for all the *k* classes**
- **Practical difficulty: require initial knowledge of many probabilities, significant computational cost**

Towards Naïve Bayesian Classifier

- **Let D be a training set of tuples and their associated class labels, and each tuple is represented by an n-D attribute vector X = (x_1, x_2, …, x_n)**
- **Suppose there are *m* classes C_1, C_2, …, C_m.**
- **Classification is to derive the maximum posteriori, i.e., the maximal $P(C_i|X)$**
- **This can be derived from Bayes' theorem**

$$P(C_i|\mathbf{X}) = \frac{P(\mathbf{X}|C_i)P(C_i)}{P(\mathbf{X})}$$

- **Since P(X) is constant for all classes, only**

needs to be maximized $$P(C_i|\mathbf{X}) = P(\mathbf{X}|C_i)P(C_i)$$

Derivation of Naïve Bayes Classifier

- A simplified assumption: attributes are conditionally independent (i.e., no dependence relation between attributes):

$$P(\mathbf{X}|C_i) = \prod_{k=1}^{n} P(x_k|C_i) = P(x_1|C_i) \times P(x_2|C_i) \times \ldots \times P(x_n|C_i)$$

- This greatly reduces the computation cost: Only counts the class distribution
- If A_k is categorical,

$$P(x_k|C_i) = \frac{\text{\# of tuples in } C_i \text{ having value } x_k \text{ for } A_k}{\text{\# of tuples of } C_i \text{ in D i.e. } |C_{i,D}|}$$

- If A_k is continous-valued,

$P(x_k|C_i)$ is usually computed based on Gaussian distribution with a mean μ and standard deviation σ

$$g(x, \mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

and $P(x_k|C_i)$ is $P(\mathbf{X}|C_i) = g(x_k, \mu_{C_i}, \sigma_{C_i})$

Naïve Bayesian Classifier: Training Dataset

Class:
C1:buys_computer = 'yes'
C2:buys_computer = 'no'

Data sample
X = (age <=30,
Income = medium,
Student = yes
Credit_rating = Fair)

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

Naïve Bayesian Classifier: An Example

- **$P(C_i)$: P(buys_computer = "yes") = 9/14 = 0.643**
  P(buys_computer = "no") = 5/14= 0.357
- **Compute $P(X|C_i)$ for each class**
  P(age = "<=30" | buys_computer = "yes") = 2/9 = 0.222
  P(age = "<= 30" | buys_computer = "no") = 3/5 = 0.6
  P(income = "medium" | buys_computer = "yes") = 4/9 = 0.444
  P(income = "medium" | buys_computer = "no") = 2/5 = 0.4
  P(student = "yes" | buys_computer = "yes) = 6/9 = 0.667
  P(student = "yes" | buys_computer = "no") = 1/5 = 0.2
  P(credit_rating = "fair" | buys_computer = "yes") = 6/9 = 0.667
  P(credit_rating = "fair" | buys_computer = "no") = 2/5 = 0.4
- **X = (age <= 30 , income = medium, student = yes, credit_rating = fair)**

$P(X|C_i)$: P(X|buys_computer = "yes") = 0.222 x 0.444 x 0.667 x 0.667 = 0.044
P(X|buys_computer = "no") = 0.6 x 0.4 x 0.2 x 0.4 = 0.019
$P(X|C_i)*P(C_i)$: P(X|buys_computer = "yes") * P(buys_computer = "yes") = 0.028
P(X|buys_computer = "no") * P(buys_computer = "no") = 0.007

Therefore, X belongs to class ("buys_computer = yes")

Avoiding the Zero-Probability Problem

- **Naïve Bayesian prediction requires each conditional prob. be non-zero. Otherwise, the predicted prob. will be zero**

$$P(X \mid C_i) = \prod_{k=1}^{n} P(x_k \mid C_i)$$

- **Ex. Suppose a dataset with 1000 tuples, income=low (0), income= medium (990), and income = high (10)**
- **Use Laplacian correction (or Laplacian estimator)**
  - *Adding 1 to each case*
    Prob(income = low) = 1/1003
    Prob(income = medium) = 991/1003
    Prob(income = high) = 11/1003
  - The "corrected" prob. estimates are close to their "uncorrected" counterparts

Naïve Bayesian Classifier: Comments

- **Advantages**
  - Easy to implement
  - Good results obtained in most of the cases
- **Disadvantages**
  - Assumption: class conditional independence, therefore loss of accuracy
  - Practically, dependencies exist among variables
    - E.g., hospitals: patients: Profile: age, family history, etc.

      Symptoms: fever, cough etc., Disease: lung cancer, diabetes, etc.
    - Dependencies among these cannot be modeled by Naïve Bayesian Classifier
- **How to deal with these dependencies? Bayesian Belief Networks (Chapter 9)**

3. Classification (12 hours)

3.1 Basics and Algorithms

3.2 Decision Tree Classifier [humanoriented]

3.3 Rule Based Classifier

3.4 Nearest Neighbor Classifier

3.5 Bayesian Classifier

3.6 Artificial Neural Network Classifier

3.7 Issues : Overfitting, Validation, Model Comparison

3.6 Artificial Neural Network Classifier

- It is set of connected i/o units in which each connection has a weight associated with it.
- During the learning phase the network learns by adjusting the weights so as to be able to predict the correct class label of i/p labels.
- It also referred as connectionist learning due to connection between units.
- It has long training time and poor interpretability but has tolerance to noisy data.
- It can classify pattern on which they have not been trained.
- Well suited for continuous valued i/ps.
- It has parallel topology and processing

- Before training the network topology must:

I. Specifying number of i/p nodes/units: Depends upon number of independent variable in data set.

II. Number of hidden layers: Generally only layer is considered in most of the problem. Two layers can be designed for complex problem. Number of nodes in the hidden layer can be adjusted iteratively.

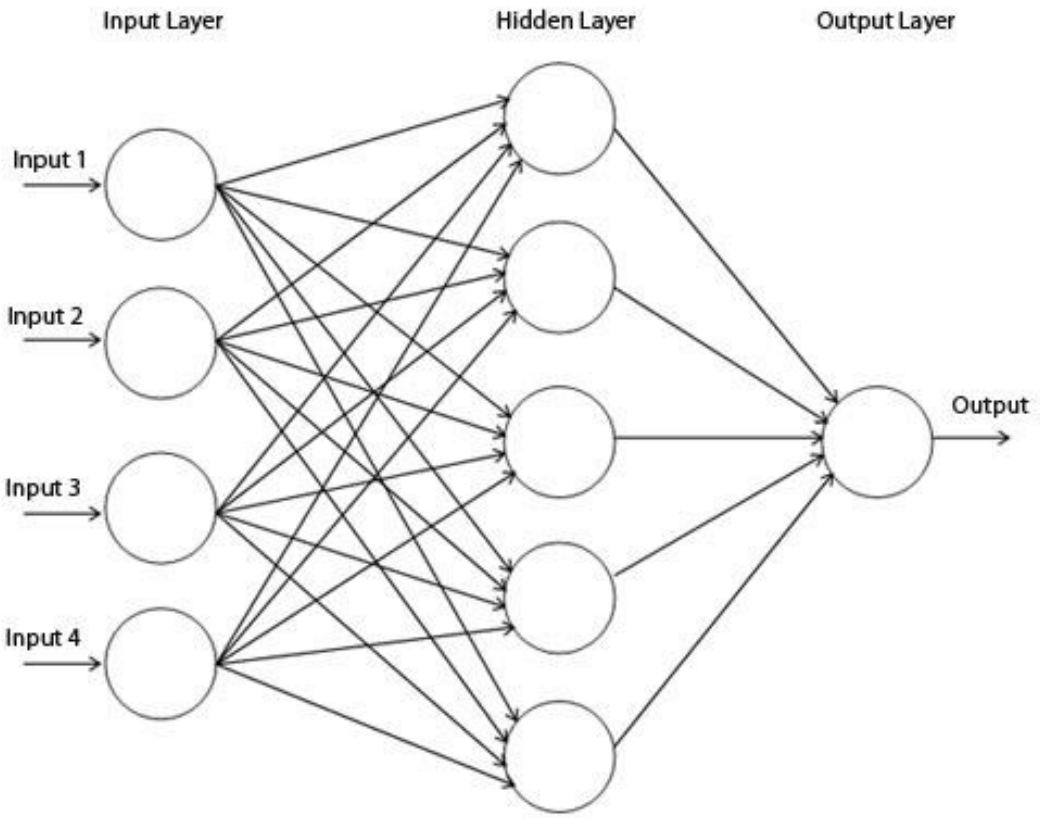
III. Number of output nodes/units: Depends upon number of class labels of the data set.

IV. Learning rate: Can be adjusted iteratively.

V. Learning algorithm: Any appropriate learning algorithm can be selected during training phase.

VI. Bias value: Can be adjusted iteratively.

- During training the connection weights must be adjusted to fit i/p values with the o/p values.



Back propagation algorithm

- **Step 1: Initialization:** Set all the weights and thresholds levels of the network to random numbers uniformly distributed inside a small range.
- **Step 2: Activation:** Activate the back propagation neural network by applying i/ps and desired o/ps.

i. Calculate the actual o/ps of the neurons in the hidden layers.

ii. Calculate the actual o/ps of the neurons in the o/p layers.

- **Step 3: Weight training:**

i. Updates weights in the back propagation network by propagating backwards the errors associated with the o/p neurons.

ii. Calculate error gradient of o/p layer and hence of neurons in the hidden layer.

- **Step 4: Iteration:** Increase iteration by repeating steps 2&3 until selected error criteria is satisfied.

3. Classification (12 hours)

3.1 Basics and Algorithms

3.2 Decision Tree Classifier [humanoriented]

3.3 Rule Based Classifier

3.4 Nearest Neighbor Classifier

3.5 Bayesian Classifier

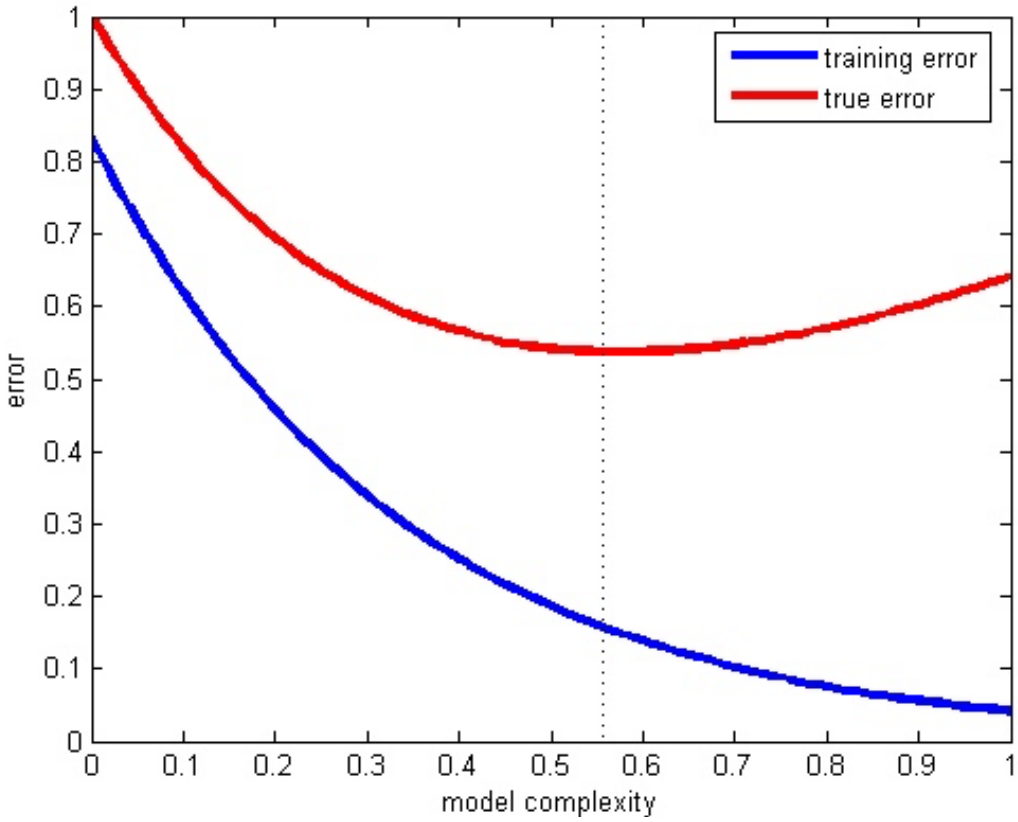
3.6 Artificial Neural Network Classifier

3.7 Issues : Overfitting, Validation, Model Comparison

3.7 Issues : Overfitting, Validation, Model Comparison

Overfitting

- Overfitting occurs when a statistical model describes random error or noise instead of the underlying relationship.
- Overfitting generally occurs when a model is excessively complex, such as having too many parameters relative to the number of observations.
- A model which has been overfit will generally have poor predictive performance.
- Overfitting depends not only on the number of parameters and data but also the conformability of the model structure.
- In order to avoid overfitting, it is necessary to use additional techniques (e.g. crossvalidation, pruning (Pre or Post), model comparison.



- Reason
  - Noise in training data.
  - Incomplete training data.
  - Flaw in assumed theory.

Validation

- Validation techniques are motivated by two fundamental problems in pattern recognition:
  - model selection and
  - performance estimation
- Validation Approaches:
  - One approach is to use the entire training data to select our classifier and estimate the error rate, but the final model will normally overfit the training data.
  - A much better approach is to split the training data into disjoint subsets cross validation (The Holdout Method)

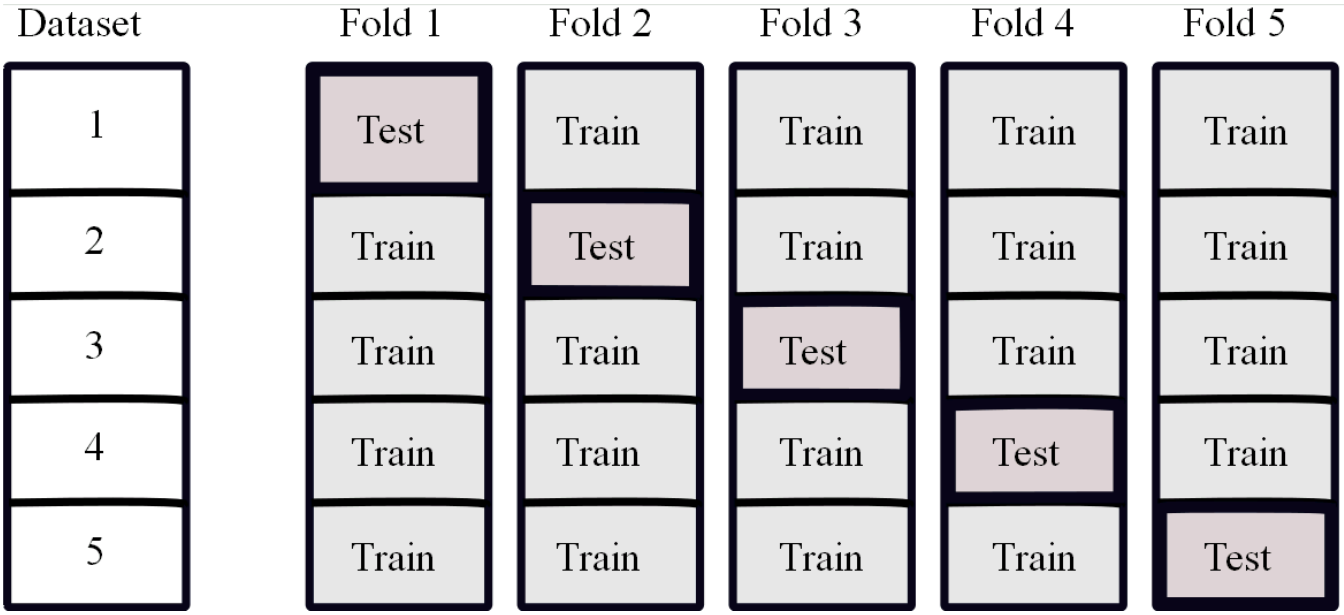
Cross Validation (The holdout method)

- Data set divided into two groups.
  - Training set: used to train the classifier and
  - Test set: used to estimate the error rate of the trained classifier
- Total number of examples = Training Set +Test Set
- Approach1: Random Sub sampling
  - Random Sub sampling performs K data splits of the dataset
  - Each split randomly selects (fixed) no. examples without replacement
  - For each data split we retrain the classifier from scratch with the training examples and estimate error with the test examples

- Approach2: K-Fold Cross-Validation
  - K-Fold Cross validation is similar to Random Sub sampling.
  - Create a K-fold partition of the dataset, For each of K experiments, use K-1 folds for training and the remaining one for testing.
  - The advantage of K-Fold Cross validation is that all the examples in the dataset are eventually used for both training and testing.
  - The true error is estimated as the average error rate

- Approach3: Leave-one-out Cross-Validation
  - Leave-one-out is the degenerate case of K-Fold Cross Validation, where K is chosen as the total number of examples where one sample is left out at each experiment.

Example – 5 Fold Cross Validation



Model Comparison

- Models can be evaluated based on the output using different method :
  - i. Confusion Matrix
  - ii. ROC Analysis
  - iii. Others such as: Gain and Lift Charts, K-S Charts

i. Confusion Matrix (Contigency Table):

- A confusion matrix contains information about actual and predicted classifications done by classifier.
- Performance of such system is commonly evaluated using data in the matrix.
- It is also known as a contingency table or an error matrix, is a specific table layout that allows visualization of the performance of an algorithm.
- Each column of the matrix represents the instances in a predicted class, while each row represents the instances in an actual class.

Classifier Evaluation Metrics: Confusion Matrix

Confusion Matrix:

Actual class\Predicted class	Predicted C_1	Predicted $\neg C_1$
Actual C_1	**True Positives (TP)**	**False Negatives (FN)**
Actual $\neg C_1$	**False Positives (FP)**	**True Negatives (TN)**

Example of Confusion Matrix:

Actual class\Predicted class	buy_computer = yes	buy_computer = no	Total
buy_computer = yes	**6954**	**46**	7000
buy_computer = no	**412**	**2588**	3000
Total	7366	2634	10000

- Given *m* classes, an entry, $\boldsymbol{CM_{i,j}}$ in a **confusion matrix** indicates # of tuples in class *i* that were labeled by the classifier as class *j*
- May have extra rows/columns to provide totals

Classifier Evaluation Metrics: Accuracy, Error Rate, Sensitivity and Specificity

A\P	C	¬C	
C	**TP**	**FN**	**P**
¬C	**FP**	**TN**	**N**
	P'	**N'**	**All**

- **Classifier Accuracy,** or recognition rate: percentage of test set tuples that are correctly classified

  Accuracy = (TP + TN)/All

- **Error rate:** *1 – accuracy*, or

  Error rate = (FP + FN)/All

- **Class Imbalance Problem**:
  - One class may be *rare*, e.g. fraud, or HIV-positive
  - Significant *majority of the negative class* and minority of the positive class
- **Sensitivity**: True Positive recognition rate
  - **Sensitivity = TP/P**
- **Specificity**: True Negative recognition rate
  - **Specificity = TN/N**
- **FPR = 1- TNR(specificity)**

Classifier Evaluation Metrics: Precision and Recall, and F-measures

- **Precision**: exactness – what % of tuples that the classifier labeled as positive are actually positive

$$precision = \frac{TP}{TP + FP}$$

- **Recall:** completeness – what % of positive tuples did the classifier label as positive?
- Perfect score is 1.0

$$recall = \frac{TP}{TP + FN}$$

- Inverse relationship between precision & recall
- ***F* measure (*F_1* or *F*-score)**: harmonic mean of precision and recall,

$$F = \frac{2 \times precision \times recall}{precision + recall}$$

- ***F_β*:** weighted measure of precision and recall
  - assigns ß times as much weight to recall as to precision

$$F_\beta = \frac{(1 + \beta^2) \times precision \times recall}{\beta^2 \times precision + recall}$$

Classifier Evaluation Metrics: Example

Actual Class\Predicted class	cancer = yes	cancer = no	Total	Recognition(%)
cancer = yes	**90**	**210**	300	30.00 (*sensitivity*
cancer = no	**140**	**9560**	9700	98.56 (*specificity)*
Total	230	9770	10000	96.40 (*accuracy*)

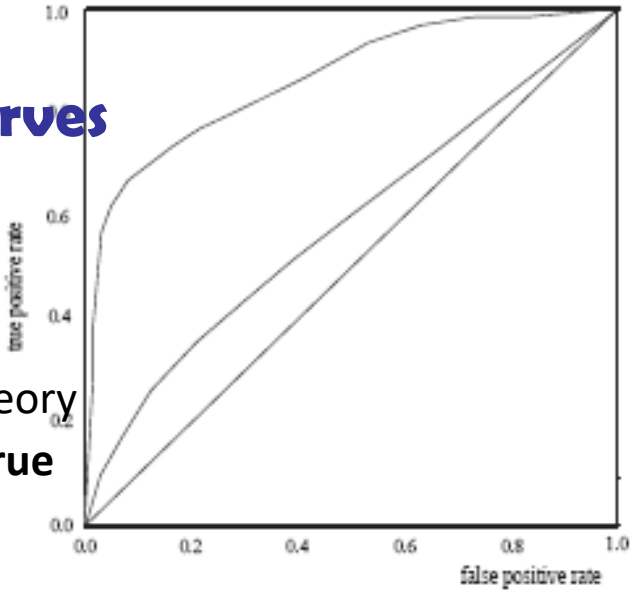
- *Precision* = 90/230 = 39.13% *Recall* = 90/300 = 30.00%

ii. ROC Analysis

- Receiver Operating Characteristic (ROC), or ROC curve, is a graphical plot that illustrates the performance of a binary classifier system as its discrimination threshold is varied.
- The curve is created by **plotting** the **true positive rate against the false positive rate** at various threshold settings.
- The ROC curve plots sensitivity (TPR) versus FPR
- ROC analysis provides tools to select possibly optimal models and to discard suboptimal ones independently from (and prior to specifying) the cost context or the class distribution.

- ROC analysis is related in a **direct and natural way to cost/benefit analysis** of diagnostic decision making.

Model Selection: ROC Curves



- **ROC** (Receiver Operating Characteristics) curves: for visual comparison of classification models
- Originated from signal detection theory
- Shows the **trade-off between** the **true positive rate and the false positive rate**
- The area under the ROC curve is a **measure of the accuracy of the model**
- **Rank the test tuples in decreasing order**: the one that is most likely to belong to the positive class appears at the top of the list
- The **closer to the diagonal line** (i.e., the closer the area is to 0.5), the **less accurate is the model**
- Vertical axis represents the true positive rate
- Horizontal axis rep. the false positive rate
- A model with perfect accuracy will have an area of 1.0

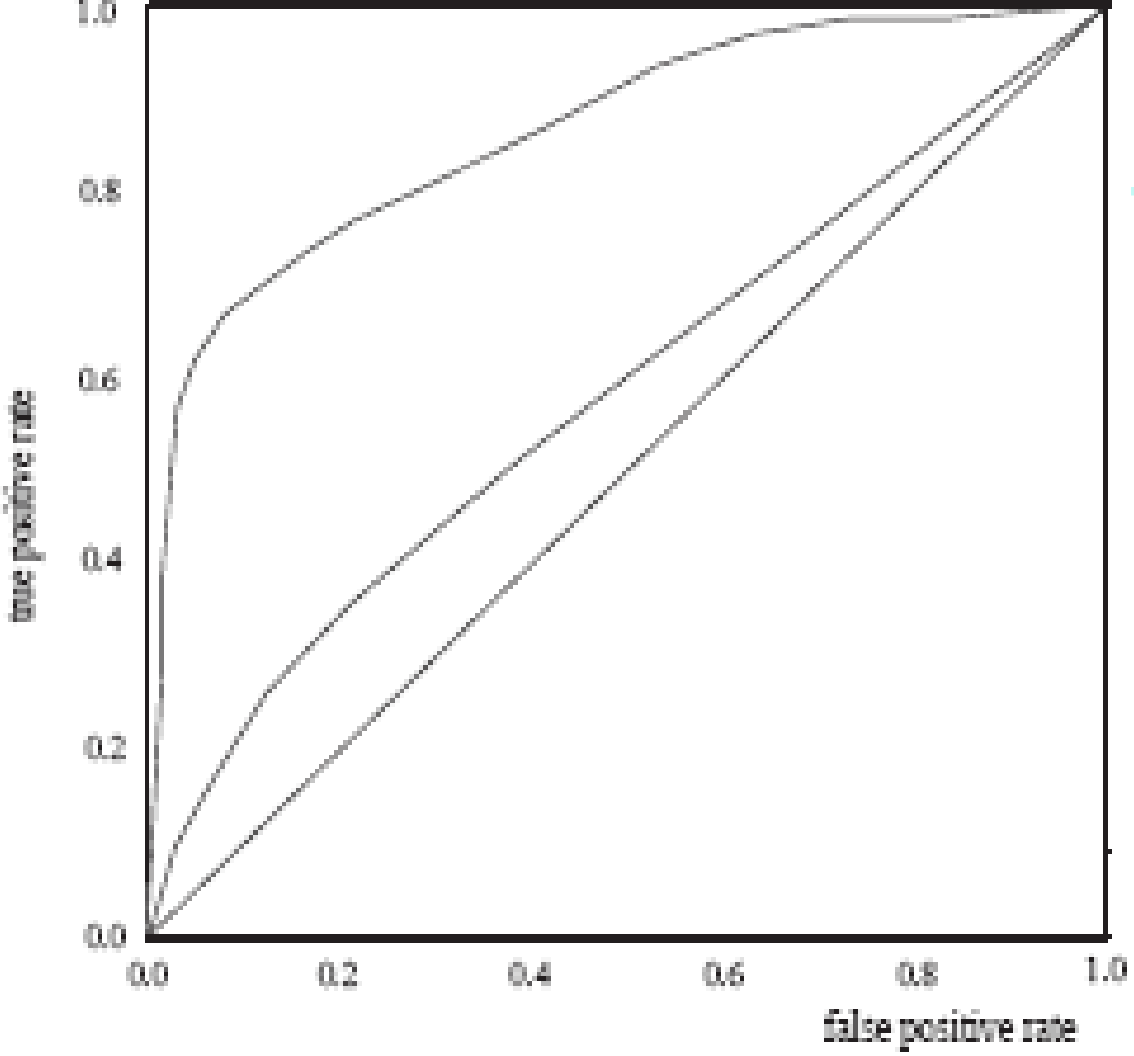


Figure shows the ROC curves of two classification models. The diagonal line representing random guessing is also shown. Thus, the closer the ROC curve of a model is to the diagonal line, the less accurate the model.

If the model is really good, initially we are more likely to encounter true positives as we move down the ranked list.

Thus, the curve moves steeply up from zero. Later, as we start to encounter fewer and fewer true positives, and more and more false positives, the curve eases off and becomes more horizontal.

To assess the accuracy of a model, we can measure the area under the curve. Several software packages are able to perform such calculation.

The closer the area is to 0.5, the less accurate the corresponding model is. A model with perfect accuracy will have an area of 1.0.

Issues Affecting Model Selection

- **Accuracy**
  - classifier accuracy: predicting class label
- **Speed**
  - time to construct the model (training time)
  - time to use the model (classification/prediction time)
- **Robustness**: handling noise and missing values
- **Scalability**: efficiency in disk-resident databases
- **Interpretability**
  - understanding and insight provided by the model
- Other measures, e.g., goodness of rules, such as decision tree size or compactness of classification rules

Summary (I)

- Classification is a form of data analysis that extracts models describing important data classes.
- Effective and scalable methods have been developed for decision tree induction, Naive Bayesian classification, rule-based classification, and many other classification methods.
- Evaluation metrics include: accuracy, sensitivity, specificity, precision, recall, *F* measure, and F_{β} measure.
- Stratified k-fold cross-validation is recommended for accuracy estimation

Summary (II)

- Significance tests and ROC curves are useful for model selection.
- There have been numerous comparisons of the different classification methods; the matter remains a research topic
- No single method has been found to be superior over all others for all data sets
- Issues such as accuracy, training time, robustness, scalability, and interpretability must be considered and can involve trade-offs, further complicating the quest for an overall superior method

Association Rule Discovery : *Apriori* Principle

Market Basket Analysis

- **Goal of MBA is to find associations (affinities) among groups of items occurring in a transactional database**
  - has roots in analysis of point-of-sale data, as in supermarkets
  - but, has found applications in many other areas

- **Association Rule Discovery**
  - most common type of MBA technique
  - Find all rules that *associate* the presence of one set of items with that of another set of items.
  - Example: *98% of people who purchase tires and auto accessories also get automotive services done*
  - We are interested in rules that are
    - non-trivial (possibly unexpected)
    - actionable
    - easily explainable

Format of Association Rules

- **Typical Rule form:**
  - Body ==> Head
  - Body and Head can be represented as sets of items (in transaction data) or as conjunction of predicates (in relational data)
  - Support and Confidence
    - Usually reported along with the rules
    - Metrics that indicate the strength of the item associations
- **Examples:**
  - {diaper, milk} ==> {beer} [support: 0.5%, confidence: 78%]
  - buys(x, "bread") /\ buys(x, “eggs") ==> buys(x, "milk") [sup: 0.6%, conf: 65%]
  - major(x, "CS") /\ takes(x, "DB") ==> grade(x, "A") [1%, 75%]
  - age(X,30-45) /\ income(X, 50K-75K) ==> owns(X, SUV)
  - age=“30-45”, income=“50K-75K” ==> car=“SUV”

Association Rules – Basic Concepts

Let **D** be database of transactions

– e.g.:

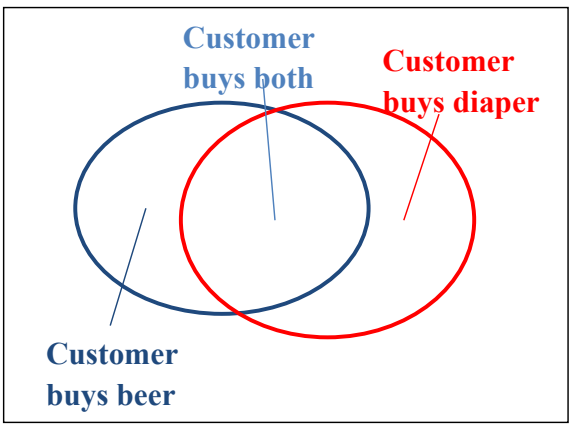
Transaction ID	Items
1000	A, B, C
2000	A, B
3000	A, D
4000	B, E, F

- Let ***I*** be the set of items that appear in the database, e.g., ***I={A,B,C,D,E,F}***
  - Each transaction ***t*** is a subset of ***I***
- A ***rule*** is an implication among ***itemsets*** ***X*** and ***Y***, of the form by **X → Y**, where **X⊂*I***, **Y⊂*I***, and **X∩Y=∅**
  – e.g.: **{B,C} → {A}** is a rule

Association Rules – Basic Concepts

- **Itemset**
  – A set of one or more items
    - E.g.: {Milk, Bread, Diaper}
  – k-itemset
    - An itemset that contains k items
- **Support count (σ)**
  – Frequency of occurrence of an itemset (number of transactions it appears)
  – E.g. **σ({Milk, Bread,Diaper}) = 2**
- **Support(s)**
  – Fraction of the transactions in which an itemset appears
  – E.g. **s({Milk, Bread, Diaper}) = 2/5**
- **Frequent Itemset**
  – An itemset whose support is greater than or equal to a ***minsup*** threshold

TID	*Items*
1	**Bread, Milk**
2	**Bread, Diaper, Beer, Eggs**
3	**Milk, Diaper, Beer, Coke**
4	**Bread, Milk, Diaper, Beer**
5	**Bread, Milk, Diaper, Coke**



Association Rules – Basic Concepts

- **Association Rule**
  - **X → Y**, where **X** and **Y** are non-overlapping itemsets
  - **{Milk, Diaper} → {Beer}**

TID	Items
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

- **Rule Evaluation Metrics**
  - **Support (s)**
    - Fraction of transactions that contain both **X** and **Y**
    - **i.e., support of the itemset X ∪ Y**
  - **Confidence (c)**
    - Measures how often items in **Y** appear in transactions that contain **X**

Example:

$$\{\text{Milk}, \text{Diaper}\} \rightarrow \text{Beer}$$

$$s = \frac{\sigma(\text{Milk}, \text{Diaper}, \text{Beer})}{|\text{D}|} = \frac{2}{5} = 0.4$$

$$c = \frac{\sigma(\text{Milk}, \text{Diaper}, \text{Beer})}{\sigma(\text{Milk}, \text{Diaper})} = \frac{2}{3} = 0.67$$

Association Rules – Basic Concepts

Another interpretation of support and confidence for ***X → Y***

– Support is the probability that a transaction contains **{X ∪ Y}** **or Pr(X /\ Y)**

support(X → Y) = support(X ∪ Y) = σ(X ∪ Y) / |D|

– Confidence is the ***conditional probability*** that a transaction will contains **Y** given that it contains X **or** **Pr(Y | X)**

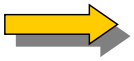
confidence(X → Y) = σ(X ∪ Y) / σ(X)
= support(X ∪ Y) / support(X)

Support and Confidence - Example

$$\text{support}(X \rightarrow Y) = \text{support}(X \cup Y) = \sigma(X \cup Y) / |D|$$

$$\text{confidence}(X \rightarrow Y) = \sigma(X \cup Y) / \sigma(X) = \text{support}(X \cup Y) / \text{support}(X)$$

Transaction ID	Items Bought
1001	A, B, C
1002	A, C
1003	A, D
1004	B, E, F
1005	A, D, F



Itemset {A, C} has a support of 2/5 = 40%

Rule {A} ==> {C} has confidence of 50%

Rule {C} ==> {A} has confidence of 100%

Support for {A, C, E} ?
Support for {A, D, F} ?

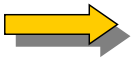
Confidence for {A, D} ==> {F} ?
Confidence for {A} ==> {D, F} ?

Lift (Improvement)

- High confidence rules are not necessarily useful
  - What if confidence of {A, B} → {C} is less than Pr({C})?
  - Lift gives the predictive power of a rule compared to random chance:

$$lift(X \rightarrow Y) = \frac{\Pr(Y|X)}{\Pr(Y)} = \frac{confidence(X \rightarrow Y)}{support(Y)} = \frac{support(X \cup Y)}{support(X).support(Y)}$$

Transaction ID	Items Bought
1001	A, B, C
1002	A, C
1003	A, D
1004	B, E, F
1005	A, D, F



Itemset {A} has a support of 4/5
Rule {C} ==> {A} has confidence of 2/2

Lift = 5/4 = 1.25

Itemset {A} has a support of 4/5
Rule {B} ==> {A} has confidence of 1/2

Lift = 5/8 = 0.625

Steps in Association Rule Discovery

1. **Find the *frequent* itemsets**
   (item sets are the sets of items that have minimum support)

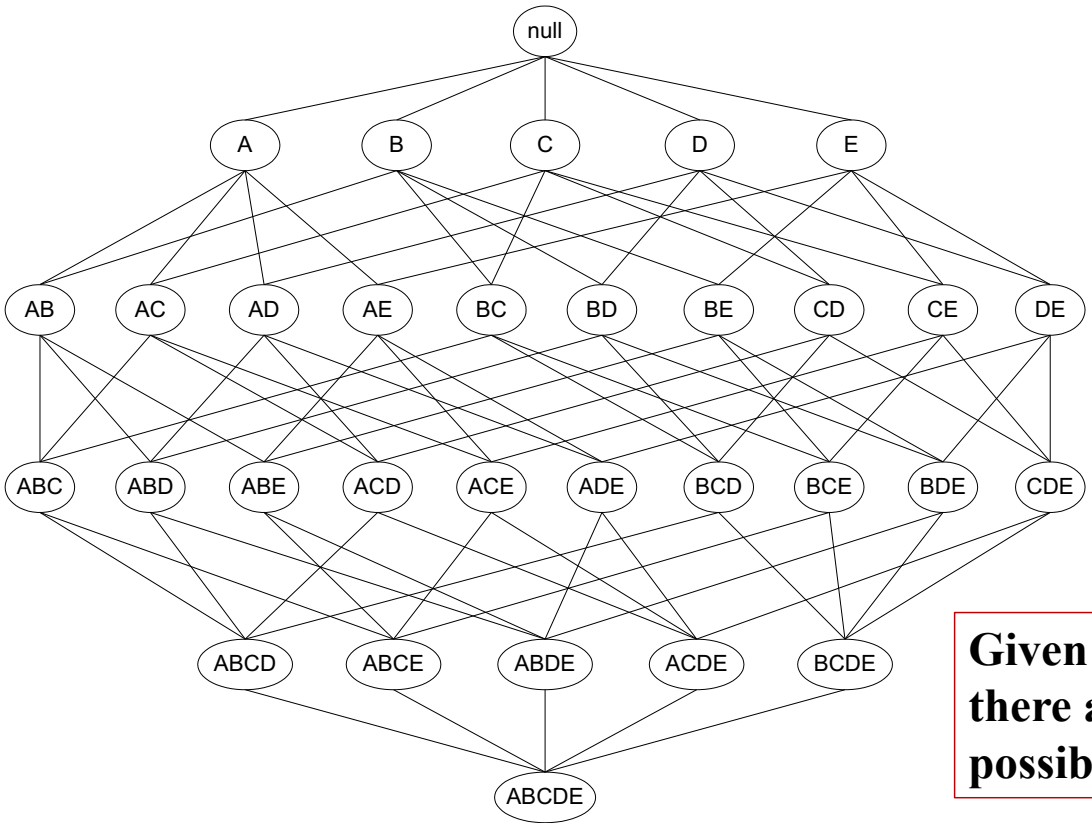
2. **Use the frequent itemsets to generate association rules**

Brute Force Algorithm:

- **List all possible itemsets and compute their support**
- **Generate all rules from frequent itemset**
- **Prune rules that fail the *minconf* threshold**

Would this work?!

How many itemsets are there?



Given *n* items, there are 2^n possible itemsets

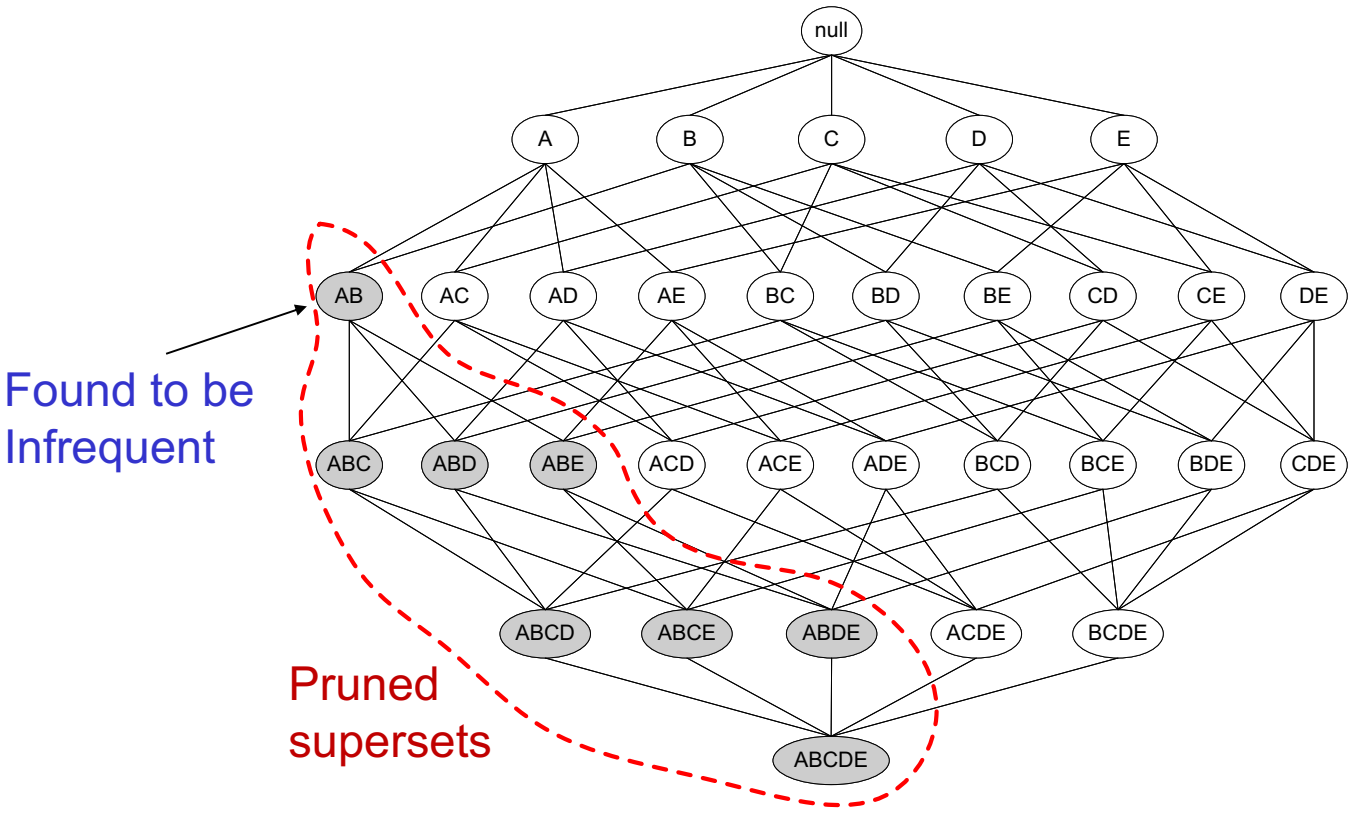
Solution: The Apriroi Principle

- Support is "downward closed"
  - If an itemset is frequent (has enough support), then all of its subsets must also be frequent
    - if {AB} is a frequent itemset, both {A} and {B} are frequent itemsets
  - This is due to the ***anti-monotone*** property of support

$$\forall X, Y : (X \subseteq Y) \Rightarrow s(X) \geq s(Y)$$

- **Corollary:** if an itemset doesn't satisfy minimum support, none of its supersets will either
  - this is essential for pruning search space)

The Apriori Principle



Support-Based Pruning

Item	Count
Bread	4
Coke	2
Milk	4
Beer	3
Diaper	4
Eggs	1

Items (1-itemsets)

minsup = 3/5

Itemset	Count
{Bread,Milk}	3
{Bread,Beer}	2
{Bread,Diaper}	3
{Milk,Beer}	2
{Milk,Diaper}	3
{Beer,Diaper}	3

Pairs (2-itemsets)

(No need to generate candidates involving Coke or Eggs)

Triplets (3-itemsets)

Itemset	Count
{Bread,Milk,Diaper}	3

Apriori Algorithm

C_k : Candidate itemset of size k

L_k : Frequent itemset of size k

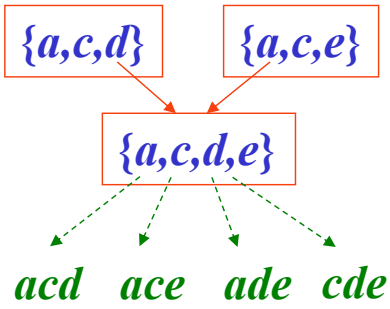
$L_1 = \{\text{frequent items}\};$
for $(k = 1; L_k \text{ !=}\emptyset; k\text{++})$ **do begin**
  C_{k+1} = candidates generated from L_k;
  for each transaction t in database **do**
    increment the count of all candidates in C_{k+1} that are contained in t
  L_{k+1} = candidates in C_{k+1} with min_support
  end
return $\cup_k L_k$;

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

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*}



Apriori Algorithm - An Example

Assume minimum support = 2

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

Note: {1,2,3}{1,2,5} and {1,3,5} not in C_3

Apriori Algorithm - An Example

L_2

item set	sup
{1 3}	2
{2 3}	2
{2 5}	3
{3 5}	2

L_3

itemset	sup
{2 3 5}	2

The final "frequent" item sets are those remaining in L2 and L3. However, {2,3}, {2,5}, and {3,5} are all contained in the larger item set {2, 3, 5}. Thus, the final group of item sets reported by Apriori are {1,3} and {2,3,5}. These are the only item sets from which we will generate association rules.

Generating Association Rules from Frequent Itemsets

- **Only strong association rules are generated**
- **Frequent itemsets satisfy minimum support threshold**
- **Strong rules are those that satisfy minimum confidence threshold**
- ***confidence*($A \rightarrow B$) = Pr($B \mid A$) =** $\dfrac{support(A \cup B)}{support(A)}$

For each frequent itemset, **f**, generate all non-empty subsets of **f**
For every non-empty subset **s** of **f** **do**
  if support(**f**)/support(**s**) ≥ min_confidence **then**
    output rule **s** ==> **(f-s)**
end

Generating Association Rules
(Example Continued)

- Item sets: {1,3} and {2,3,5}
- Recall that confidence of a rule LHS → RHS is Support of itemset (i.e. LHS ∪ RHS) divided by support of LHS.

Candidate rules for {1,3}		Candidate rules for {2,3,5}			
Rule	Conf.	Rule	Conf.	Rule	Conf.
{1}→{3}	2/2 = 1.0	{2,3}→{5}	2/2 = 1.00	{2}→{5}	3/3 = 1.00
{3}→{1}	2/3 = 0.67	{2,5}→{3}	2/3 = 0.67	{2}→{3}	2/3 = 0.67
		{3,5}→{2}	2/2 = 1.00	{3}→{2}	2/3 = 0.67
		{2}→{3,5}	2/3 = 0.67	{3}→{5}	2/3 = 0.67
		{3}→{2,5}	2/3 = 0.67	{5}→{2}	3/3 = 1.00
		{5}→{2,3}	2/3 = 0.67	{5}→{3}	2/3 = 0.67

Assuming a min. confidence of 75%, the final set of rules reported by Apriori are: {1}→{3}, {2,3}→{5}, {3,5}→{2}, {5}→{2} and {2}→{5}

Frequent Patterns Without Candidate Generation

- **Bottlenecks of the Apriori approach**
  - Breadth-first (i.e., level-wise) search
  - Candidate generation and test (Often generates a huge number of candidates)
- **The FPGrowth Approach (J. Han, J. Pei, Y. Yin, 2000)**
  - Depth-first search; avoids explicit candidate generation
- **Basic Idea: Grow long patterns from short ones using locally frequent items only**
  - "abc" is a frequent pattern; get all transactions having "abc"
  - "d" is a local frequent item in DB|abc → abcd is a frequent pattern
- **Approach:**
  - Use a compressed representation of the database using an FP-tree
  - Once an FP-tree has been constructed, it uses a recursive divide-and-conquer approach to mine the frequent itemsets

Reducing Number of Comparisons

Candidate counting:

•- Scan the database of transactions to determine the support of each candidate itemset.

•- To reduce the number of comparisons, store the candidates in a hash structure

•- Instead of matching each transaction against every candidate, match it against candidates contained in the hashed buckets

Hash Table

•- A hash table (hash map) is a data structure used to implement an associative array, a structure that can map keys to values.

•- A hash table uses a hash function to compute an index into an array of buckets or slots, from which the correct value can be found.

•- Max leaf size: max number of itemsets stored in a leaf node, if number of candidate itemsets exceeds max leaf size, split the node.

Factors Affecting Complexity

- **Choice of minimum support threshold: Lowering support threshold results in more frequent itemsets.This may increase number of candidates and max length of frequent itemsets.**
- **Dimensionality (number of items) of the data set: More space is needed to store support count of each item. If number of frequent items increases, both computation and I/O costs may also increase.**
- **Size of database: Since Apriori makes multiple passes, run time of algorithm may increase with number of transactions.**
- **Average transaction width: Transaction width increases with denser data sets. This may increase max length of frequent itemsets and traversals of hash tree (number of subsets in a transaction increases with its width)**

4.3 FP-Growth, FP-Tree

Scalable Frequent Itemset Mining Methods

1. **Apriori**: A Candidate Generation-and-Test Approach
   - Also need to Improving the Efficiency of Apriori
2. **FPGrowth**: A Frequent Pattern-Growth Approach

Apriori vs FPGrowth

- Bottlenecks of the Apriori approach
  - Breadth-first (i.e., level-wise) search
  - Candidate generation and test
    - Often generates a huge number of candidates
- The FPGrowth Approach (J. Han, J. Pei, and Y. Yin, SIGMOD' 00)
  - Depth-first search
  - Avoid explicit candidate generation
- Major philosophy: Grow long patterns from short ones using local frequent items only
  - "abc" is a frequent pattern
  - Get all transactions having "abc", i.e., project DB on abc: DB|abc
  - "d" is a local frequent item in DB|abc → abcd is a frequent pattern

Comparative Result

Table 4.3: Comparative table

Parameters	Apriori Algorithm	FP-growth Algorithm
Technique	Use Apriori property and join and prune property	It constructs conditional frequent pattern tree and conditional pattern base from database which satisfy minimum support.
Memory utilization	Due to large number of candidate are generated so require large memory space.	Due to compact structure and no candidate generation require less memory.
No. of scans	Multiple scans for generating candidate sets.	Scan the DB only twice and twice only.
Time	Execution time is more as time is wasted in producing candidates every time.	Execution time is small than Apriori algorithm.

Frequent Pattern (FP) Growth Method

•- Mining frequent itemsets without candidate generation.
•- It is a divide and conquers strategy.
•- It compress the database representing frequent items into a frequent –pattern tree (FP- Tree), which retains the itemsets association information.
•- Divides the compressed database into a set of conditional databases, each associated with one frequent item or pattern fragment and then mines each such database separately.
•- FP-Growth method transforms the problem of finding long frequent patterns to searching for shorter ones recursively and then concatenating the suffix.
•- It uses least frequent items as suffix .

Adv: Reduce search cost, has good selectivity, faster than apriori.

Disadv: When the database is large, it is sometimes unrealistic to construct a main memory based FP-tree.

Frequent Pattern (FP) Growth Algorithm has 2 steps:

Step 1: Build FP-Tree (FP-Tree algorithm)

•- Create root node of tree, labeled with null.
•- Scan the transactional database.
•- The items in each transaction are processed in sorted order (Descending) and branch is created for each transaction.

•***Step2: Extract Frequent Itemset (Conditional FP-Tree algorithm)***

•- Start from each frequent length pattern as an initial suffix pattern.
•- Construct conditional pattern base. (Pattern base is a sub database which consists of the set of prefix paths in the FP-tree co-occurring with suffix pattern.
•- Construct its FP-tree and perform mining recursively on such a tree

What is FP Growth?

- FP Growth Stands for frequent pattern growth
- It is a scalable technique for mining frequent pattern in a database

FP Growth

- FP growth improves Apriority to a big extent
- Frequent Item set Mining is possible without candidate generation
- Only “two scan” to the database is needed

BUT HOW?

FP Growth

- Simply a two step procedure
  - Step 1: Build a compact data structure called the FP-tree
    - Built using 2 passes over the data-set.
  - Step 2: Extracts frequent item sets directly from the FP-tree

FP Growth

- Now Lets Consider the following transaction table

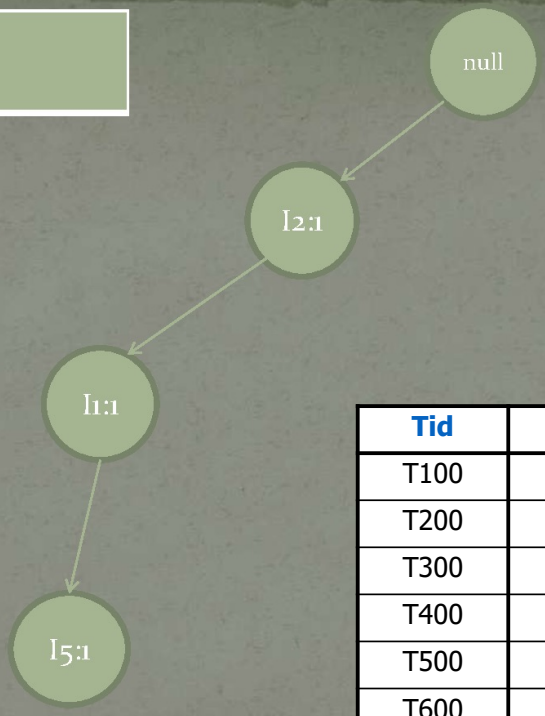
Tid	Items
T100	I1,I2,I5
T200	I2,I4
T300	I2,I3
T400	I1,I2,I4
T500	I1,I3
T600	I2,I3
T700	I1,I3
T800	I1,I2,I3,I5
T900	I1,I2,I3

FP Growth

- Now we will build a FP tree of that database
- Item sets are considered in order of their descending value of support count.

Calculate Support Count (Descending order):
I2:7
I1:6
I3:6
I4:2
I5:2

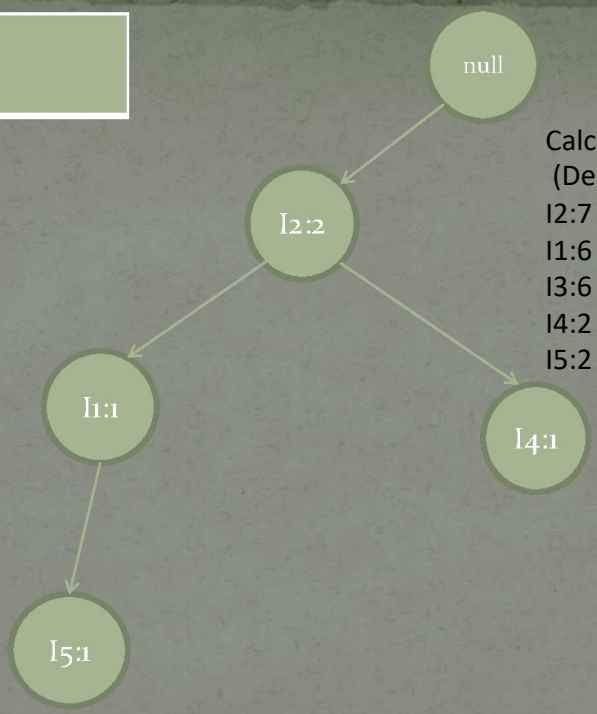
For Transaction:
I2,I1,I5



Calculate Support Count
(Descending order):
I2:7
I1:6
I3:6
I4:2
I5:2

Tid	Items
T100	I1,I2,I5
T200	I2,I4
T300	I2,I3
T400	I1,I2,I4
T500	I1,I3
T600	I2,I3
T700	I1,I3
T800	I1,I2,I3,I5
T900	I1,I2,I3

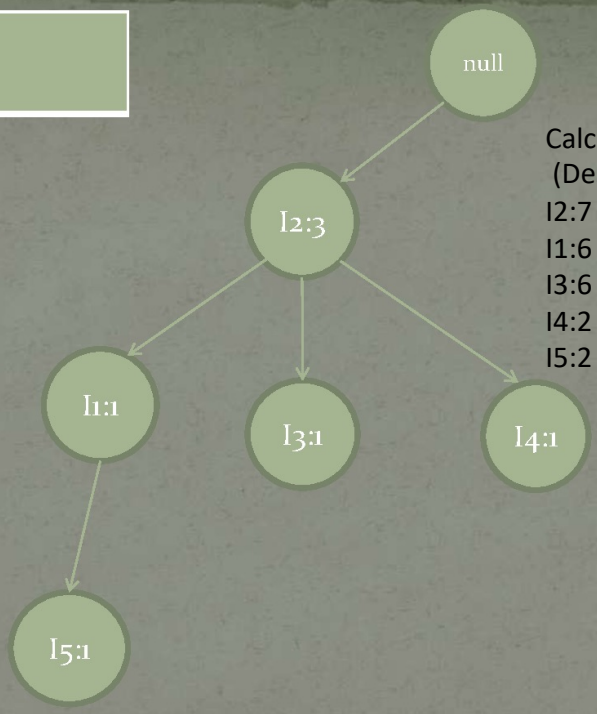
For Transaction:
I2,I4



Calculate Support Count
(Descending order):
I2:7
I1:6
I3:6
I4:2
I5:2

Tid	Items
T100	I1,I2,I5
T200	I2,I4
T300	I2,I3
T400	I1,I2,I4
T500	I1,I3
T600	I2,I3
T700	I1,I3
T800	I1,I2,I3,I5
T900	I1,I2,I3

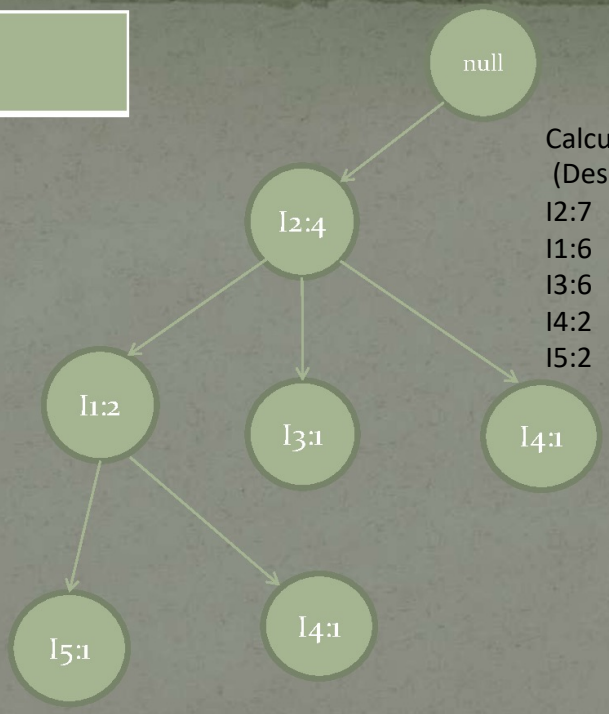
For Transaction:
I2,I3



Calculate Support Count
(Descending order):
I2:7
I1:6
I3:6
I4:2
I5:2

Tid	Items
T100	I1,I2,I5
T200	I2,I4
T300	I2,I3
T400	I1,I2,I4
T500	I1,I3
T600	I2,I3
T700	I1,I3
T800	I1,I2,I3,I5
T900	I1,I2,I3

For Transaction:
I2,I1,I4

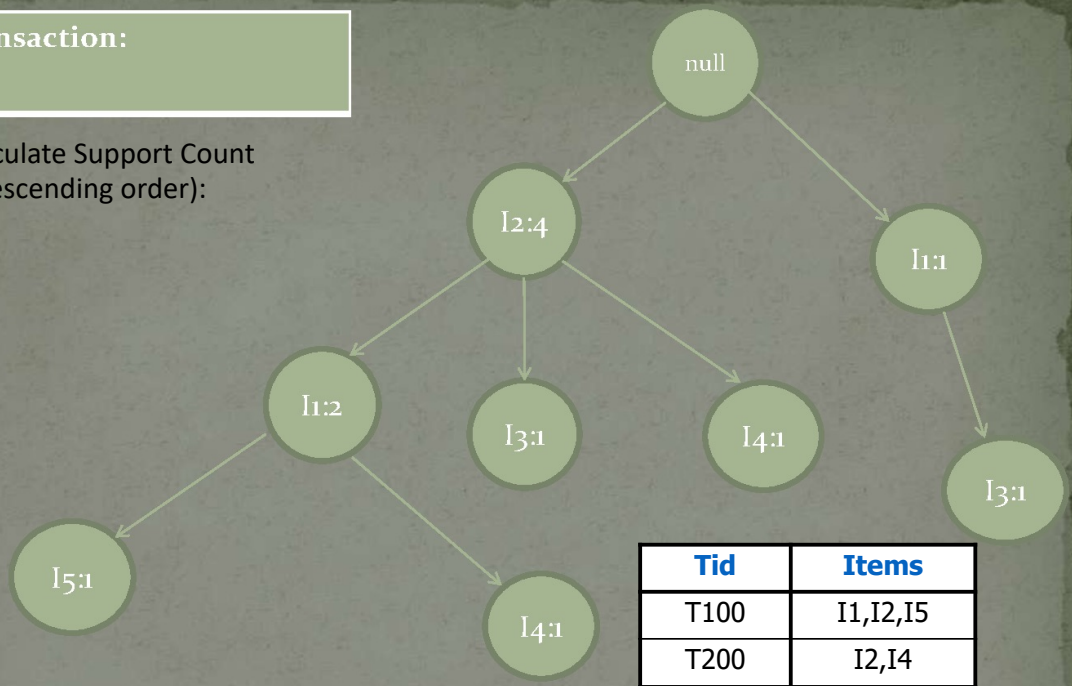


Calculate Support Count
(Descending order):
I2:7
I1:6
I3:6
I4:2
I5:2

Tid	Items
T100	I1,I2,I5
T200	I2,I4
T300	I2,I3
T400	I1,I2,I4
T500	I1,I3
T600	I2,I3
T700	I1,I3
T800	I1,I2,I3,I5
T900	I1,I2,I3

For Transaction:
I1,I3

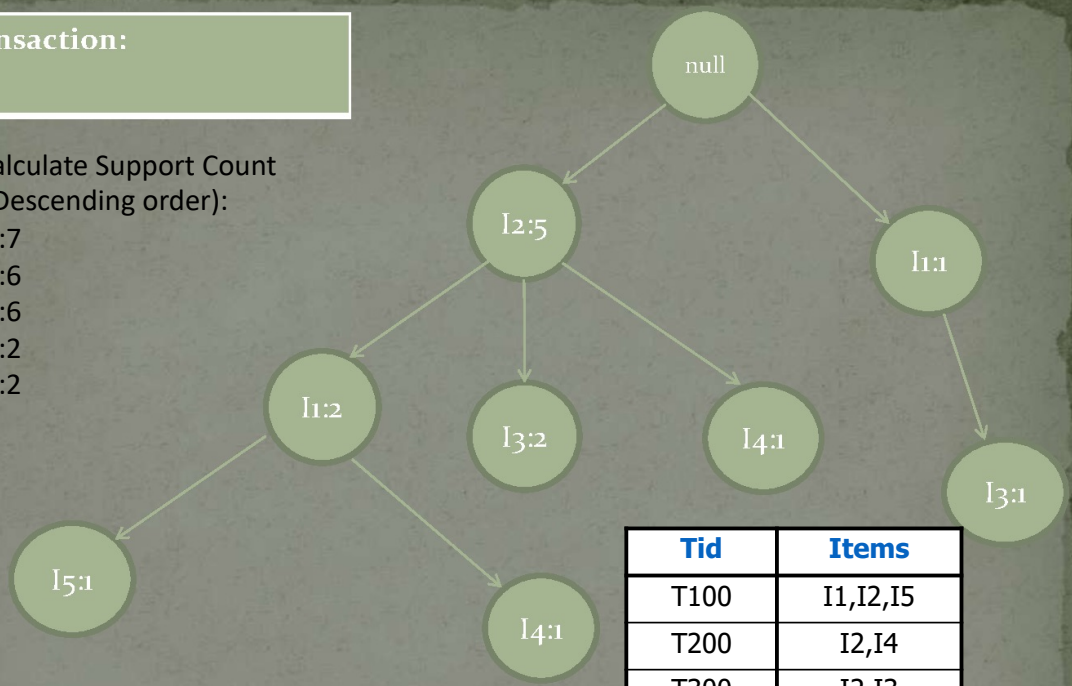
Calculate Support Count
(Descending order):
I2:7
I1:6
I3:6
I4:2
I5:2



Tid	Items
T100	I1,I2,I5
T200	I2,I4
T300	I2,I3
T400	I1,I2,I4
T500	I1,I3
T600	I2,I3

For Transaction:
I2,I3

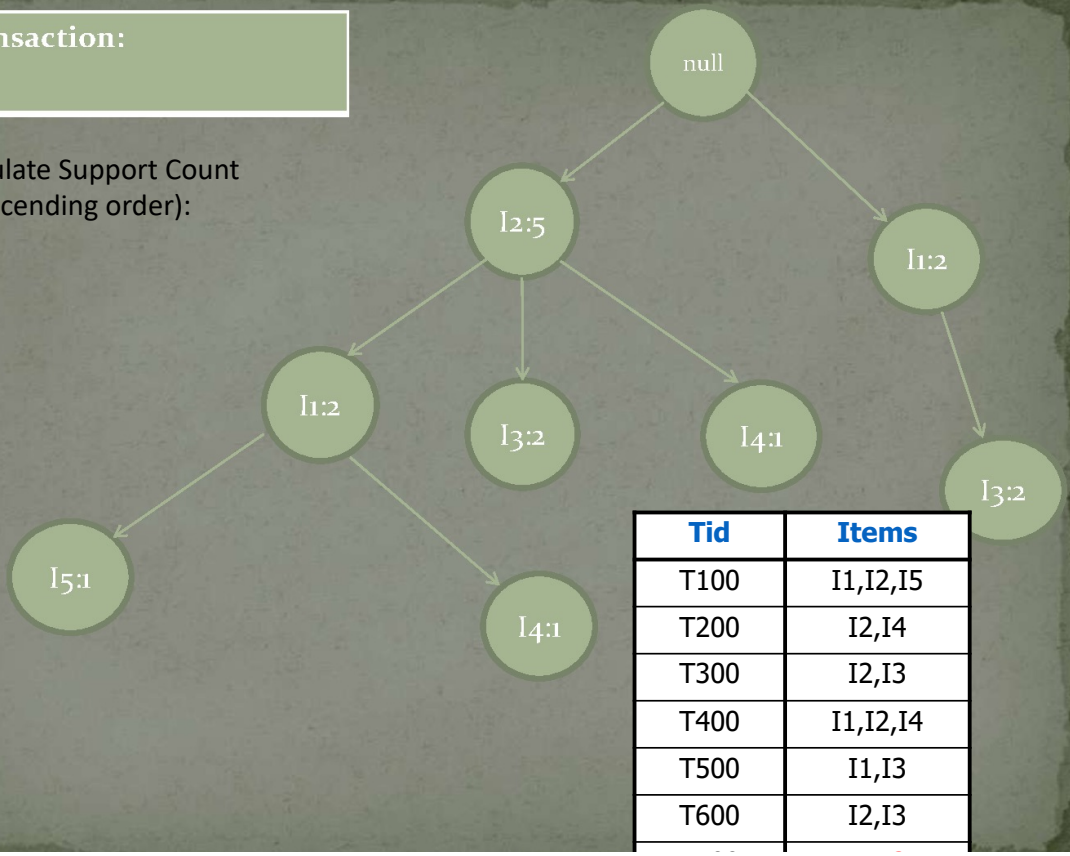
Calculate Support Count
(Descending order):
I2:7
I1:6
I3:6
I4:2
I5:2



Tid	Items
T100	I1,I2,I5
T200	I2,I4
T300	I2,I3
T400	I1,I2,I4
T500	I1,I3
T600	I2,I3
T700	I1,I3

For Transaction:
I1,I3

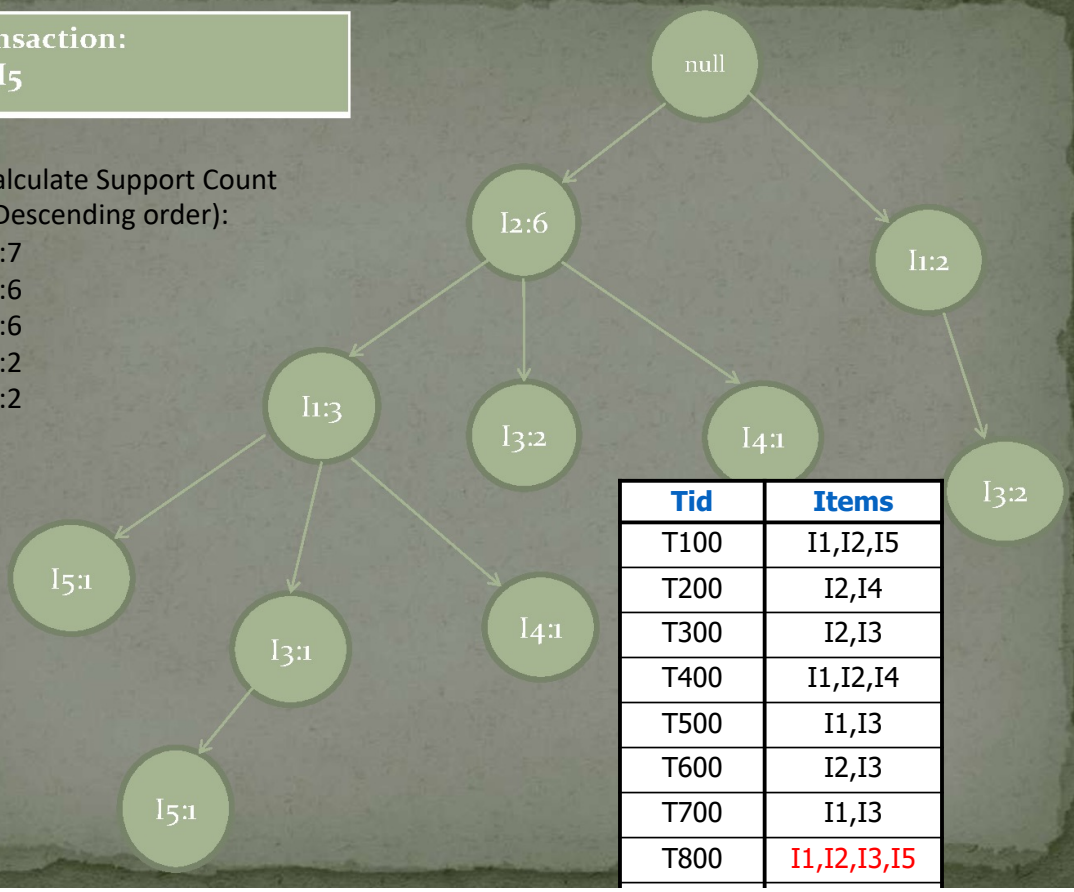
Calculate Support Count
(Descending order):
I2:7
I1:6
I3:6
I4:2
I5:2



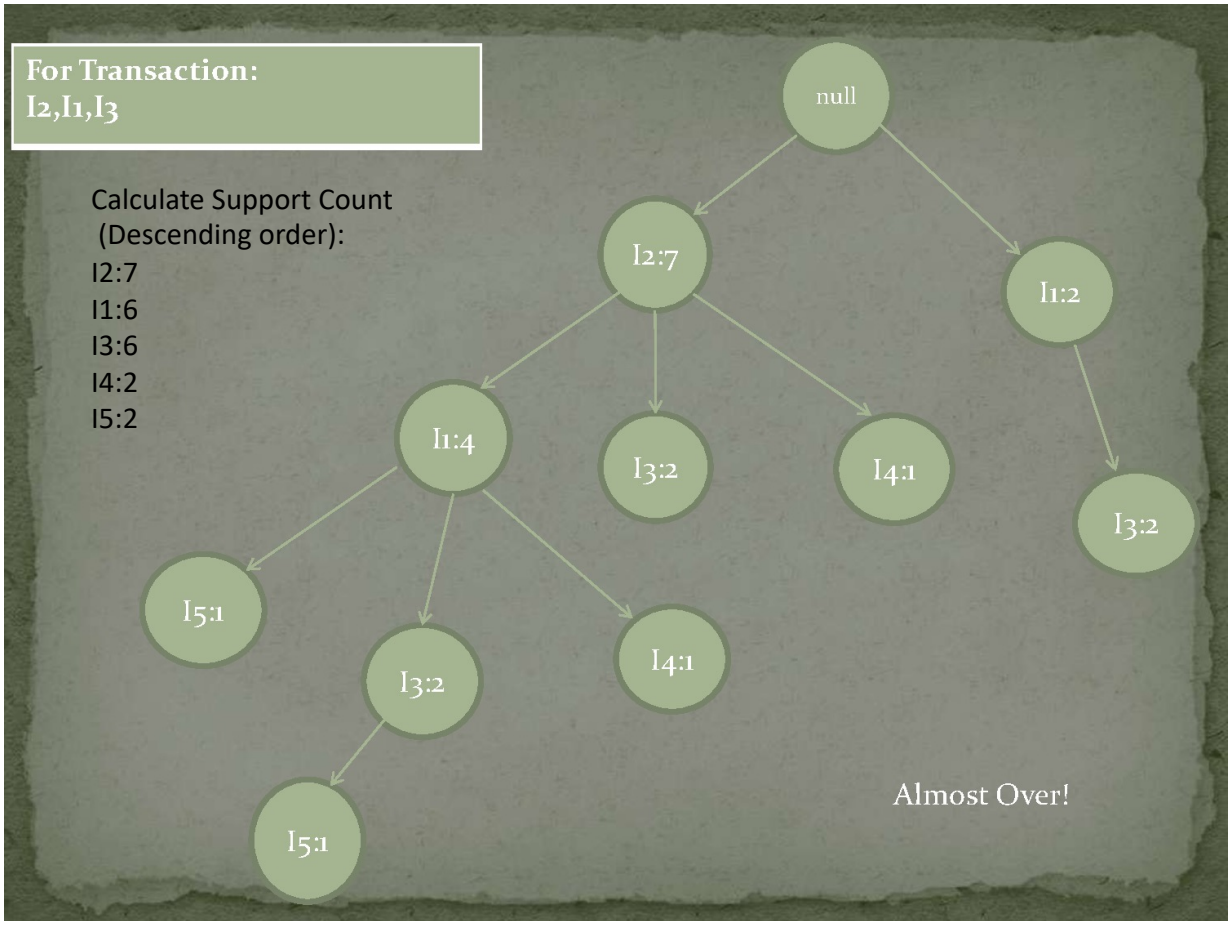
Tid	Items
T100	I1,I2,I5
T200	I2,I4
T300	I2,I3
T400	I1,I2,I4
T500	I1,I3
T600	I2,I3

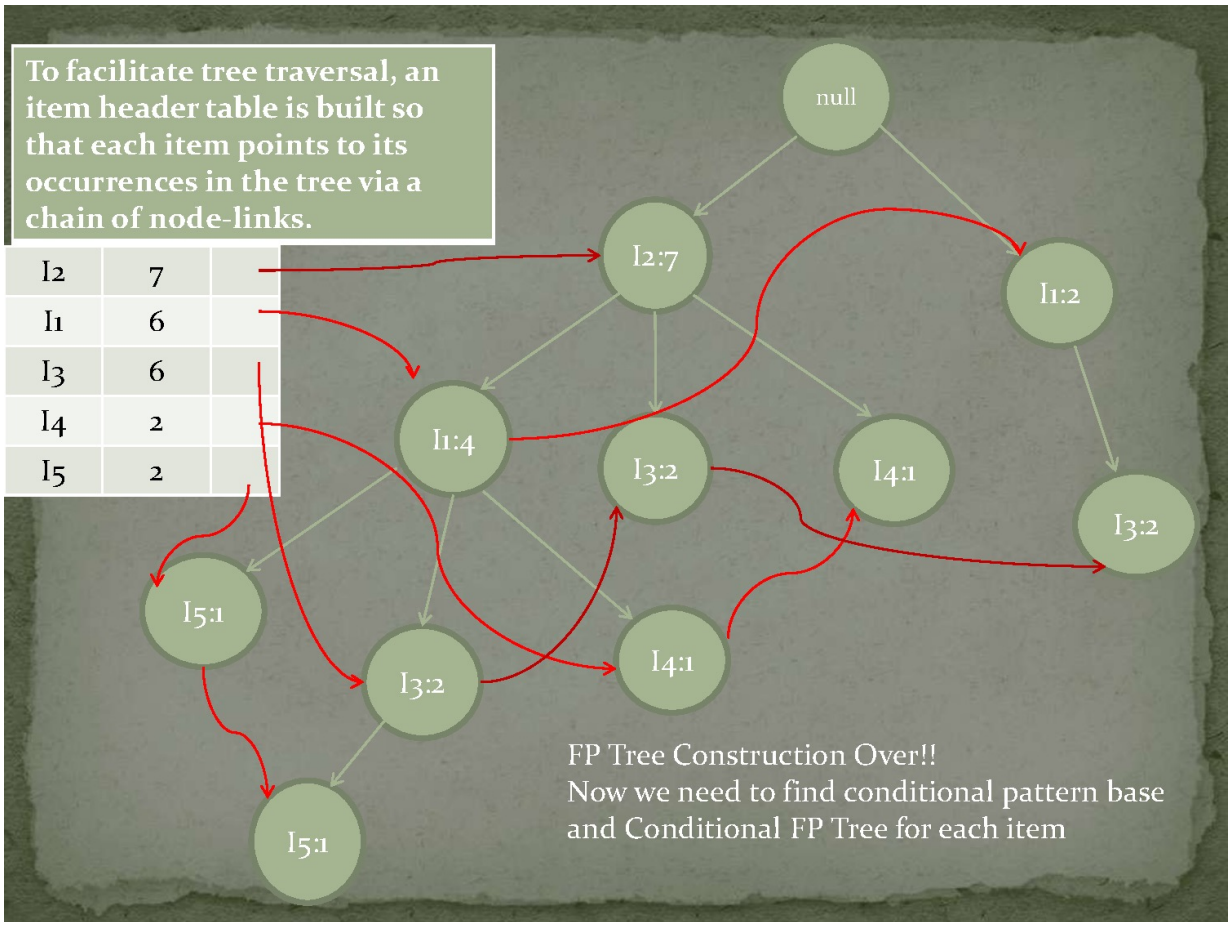
For Transaction:
I2,I1,I3,I5

Calculate Support Count
(Descending order):
I2:7
I1:6
I3:6
I4:2
I5:2



Tid	Items
T100	I1,I2,I5
T200	I2,I4
T300	I2,I3
T400	I1,I2,I4
T500	I1,I3
T600	I2,I3
T700	I1,I3
T800	I1,I2,I3,I5


For Transaction:
I2,I1,I3
Calculate Support Count
(Descending order):
I2:7
I1:6
I3:6
I4:2
I5:2
null
I2:7
I1:2
I1:4
I3:2
I4:1
I3:2
I5:1
I3:2
I4:1
I5:1
Almost Over!


To facilitate tree traversal, an item header table is built so that each item points to its occurrences in the tree via a chain of node-links.
I2 7
I1 6
I3 6
I4 2
I5 2
null
I2:7
I1:2
I1:4
I3:2
I4:1
I3:2
I5:1
I3:2
I4:1
I5:1
FP Tree Construction Over!!
Now we need to find conditional pattern base and Conditional FP Tree for each item

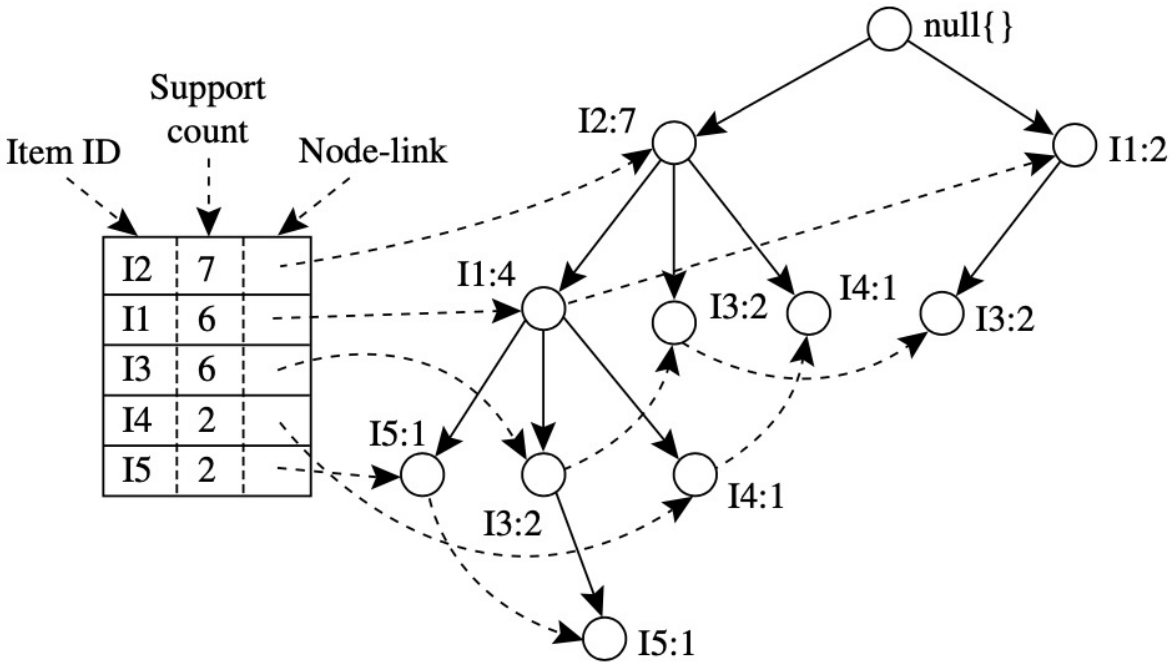
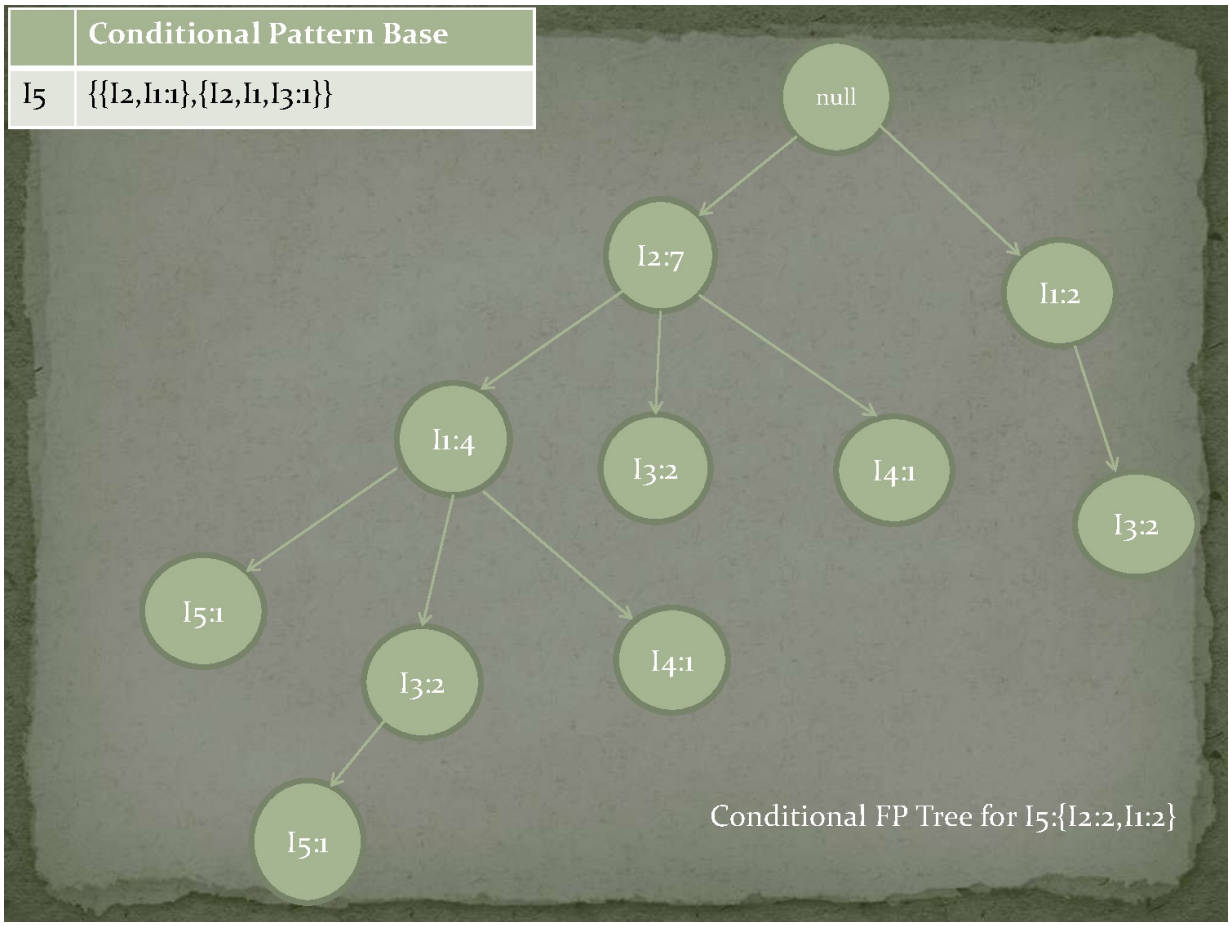
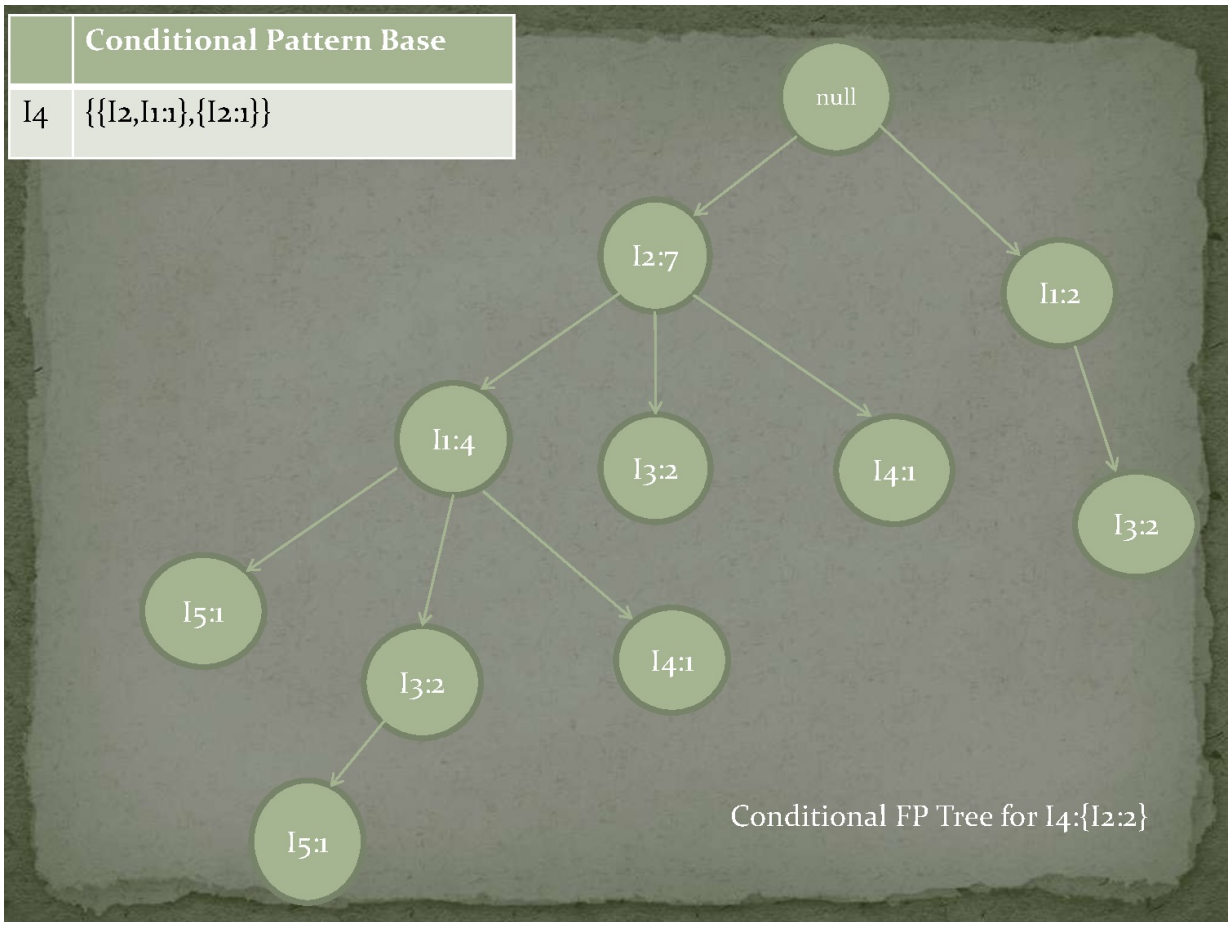


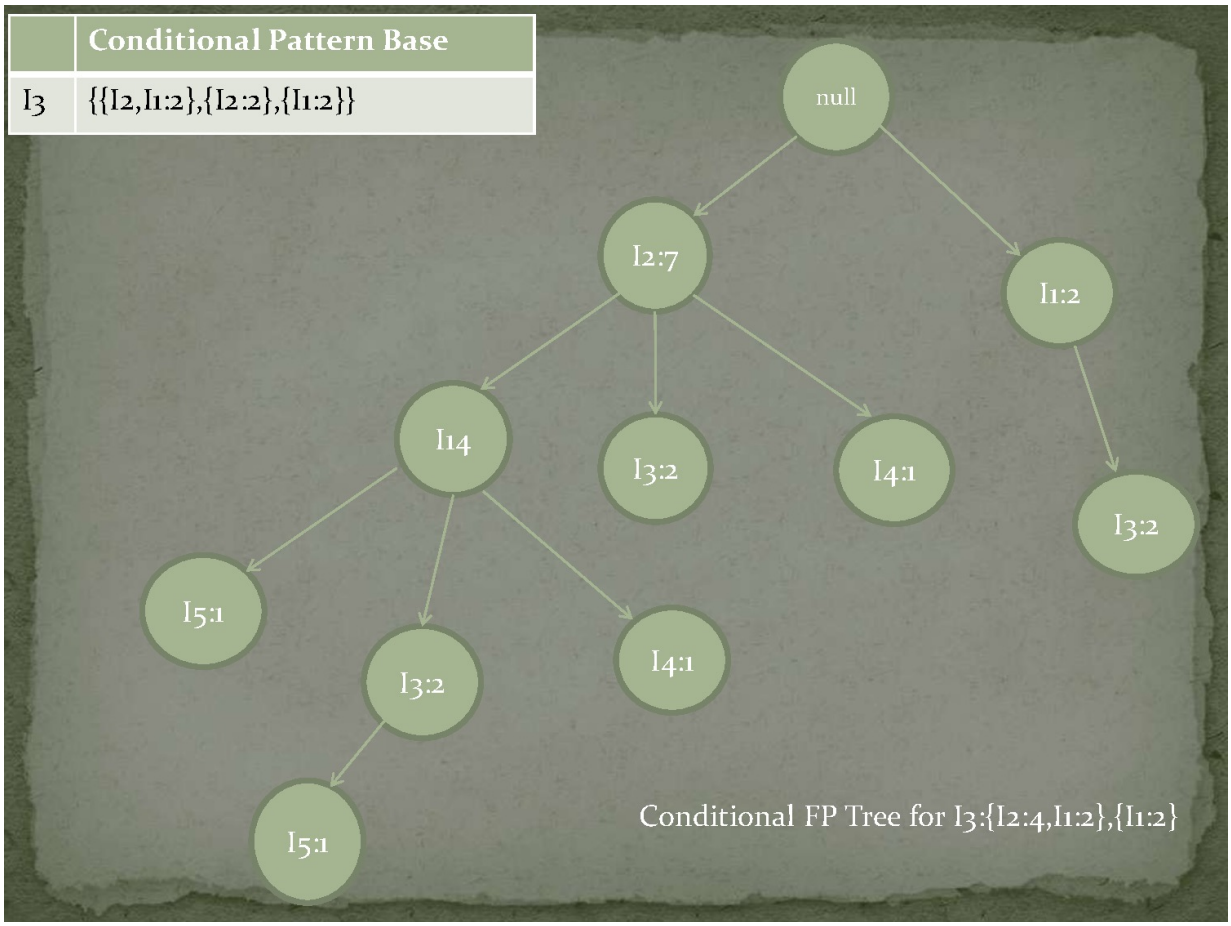
Figure 6.7 An FP-tree registers compressed, frequent pattern information.



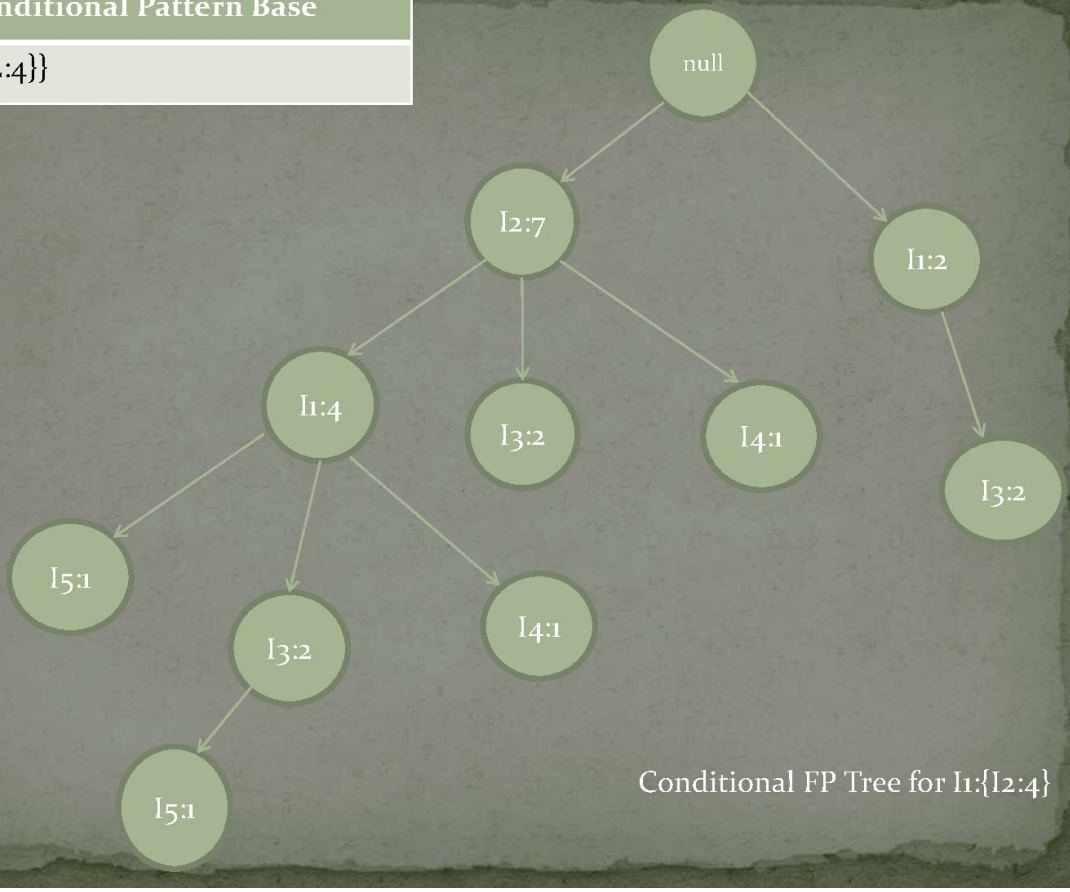
	Conditional Pattern Base
I4	{{I2,I1:1},{I2:1}}



	Conditional Pattern Base
I3	{{I2,I1:2},{I2:2},{I1:2}}



	Conditional Pattern Base
I1	{{I2:4}}



Conditional FP Tree for I1:{I2:4}

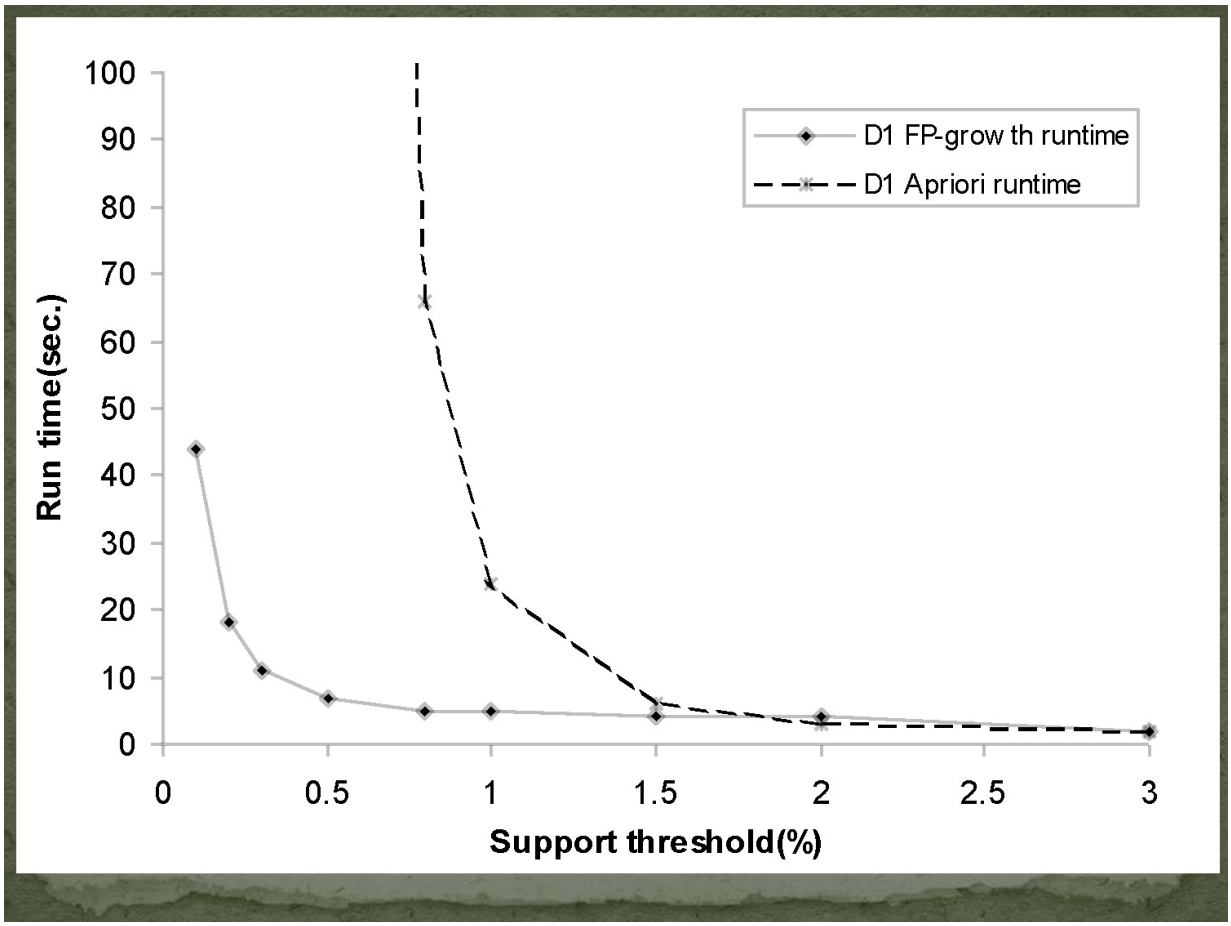
Frequent Patters Generated

	Frequent Pattern Generated
I5	{I2, I5: 2}, {I1, I5: 2}, {I2, I1, I5: 2}
I4	{I2, I4: 2}
I3	{I2, I3: 4}, {I1, I3: 4}, {I2, I1, I3: 2}
I1	{I2, I1: 4}

Summary of problem solution (FROM BOOK)
Write in this way in exam::

Mining the FP-Tree by Creating Conditional (Sub-)Pattern Bases

Item	*Conditional Pattern Base*	*Conditional FP-tree*	*Frequent Patterns Generated*
I5	{{I2, I1: 1}, {I2, I1, I3: 1}}	⟨I2: 2, I1: 2⟩	{I2, I5: 2}, {I1, I5: 2}, {I2, I1, I5: 2}
I4	{{I2, I1: 1}, {I2: 1}}	⟨I2: 2⟩	{I2, I4: 2}
I3	{{I2, I1: 2}, {I2: 2}, {I1: 2}}	⟨I2: 4, I1: 2⟩, ⟨I1: 2⟩	{I2, I3: 4}, {I1, I3: 4}, {I2, I1, I3: 2}
I1	{{I2: 4}}	⟨I2: 4⟩	{I2, I1: 4}



Benefits of the FP-tree Structure

- Completeness
  - Preserve complete information for frequent pattern mining
  - Never break a long pattern of any transaction
- Compactness
  - Reduce irrelevant info—infrequent items are gone
  - Items in frequency descending order: the more frequently occurring, the more likely to be shared
  - Never be larger than the original database (not count node-links and the *count* field)

Advantages of the Pattern Growth Approach

- Divide-and-conquer:
  - Decompose both the mining task and DB according to the frequent patterns obtained so far
  - Lead to focused search of smaller databases
- Other factors
  - No candidate generation, no candidate test
  - Compressed database: FP-tree structure
  - No repeated scan of entire database
  - Basic ops: counting local freq items and building sub FP-tree, no pattern search and matching
- A good open-source implementation and refinement of FPGrowth
  - FPGrowth+ (Grahne and J. Zhu, FIMI'03)

Q: What is the most significant advantage of FP-Tree? Why FP-Tree is complete in relevance to frequent pattern mining?

- Efficiency, the most significant advantage of the *FP-tree* is that it requires two scans to the underlying database (and only two scans) to construct the FP-tree. This efficiency is further apparent in database with prolific and long patterns or for mining frequent patterns with low support threshold.
- As each transaction in the database is mapped to one path in the FP-Tree, therefore, the frequent item-set information in each transaction is completely stored in the FP-Tree. Besides, one path in the FP-Tree may represent frequent item-sets in multiple transactions without ambiguity since the path representing every transaction must start from the root of each item prefix sub-tree.

4.4 Handling Categorical Attributes

Categorical data

- Categorical data is a statistical data type consisting of categorical variables, used for observed data whose value is one of a fixed number of nominal categories.
- More specifically, categorical data may derive from either or both of observations made of qualitative data, where the observations are summarized as counts or cross tabulations, or of quantitative data.
- Observations might be directly observed counts of events happening or they might counts of values that occur within given intervals.
- Often, purely categorical data are summarized in the form of a contingency table.
- However, particularly when considering data analysis, it is common to use the term "categorical data" to apply to data sets that, while containing some categorical variables, may also contain non-categorical variables.

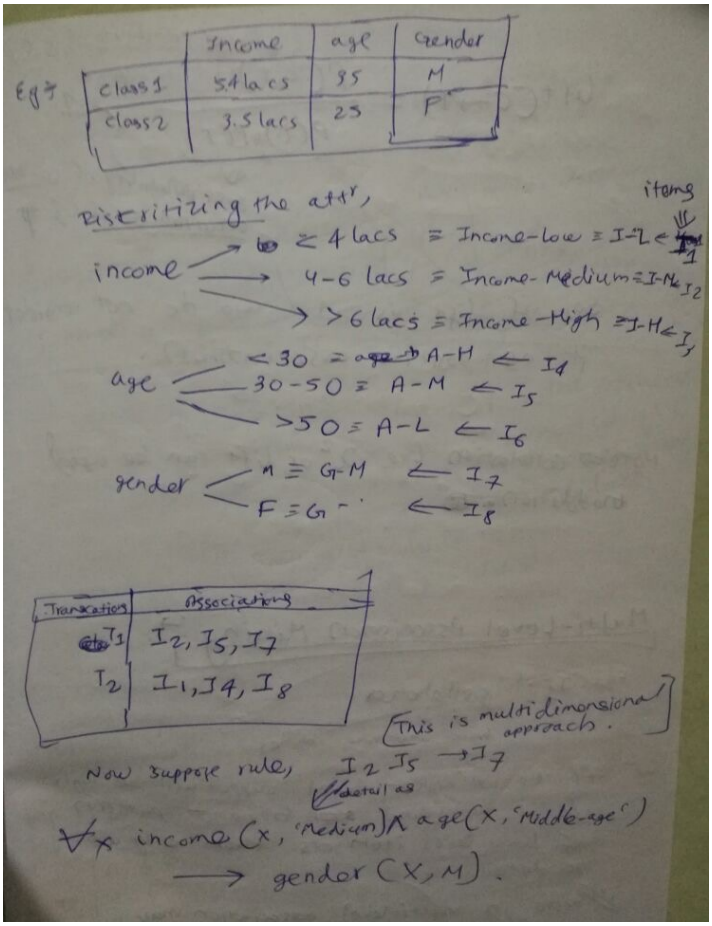
Potential Issues

- **What if attribute has many possible values:**
  - Example: attribute country has more than 200 possible values. Many of the attribute values may have very low support.
  - Potential solution: Aggregate the low-support attributes values.
- **What if distribution of attribute values is highly skewed:**
  - Example: 95% of the visitors have Buy = No. Most of the items will be associated with (Buy=No) item
  - Potential solution: drop the highly frequent items

Handling Categorical (Multidimensional Association) Rules

- Example we see till now are for unidimentional association rules.
- For categorical multidimentional we discritize quantity and convert into (attribute,values) pairs and model them as items of transaction representing an attributes.

Example:



Handling Categorical Attributes

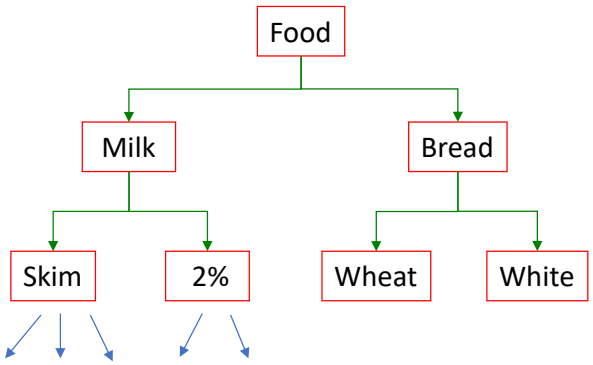
- Transform categorical attribute into asymmetric binary variables. i.e If the outcomes of a binary variable are not equally important.
- Introduce a new “item” for each distinct attribute- value pair.

Multilevel Association mining

- Advance mining
- Consider sub-level / specialization

Extensions: Multiple-Level Association Rules

- Items often form a 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



Associations in Recommender Systems

5. Cluster Analysis (9 hours)

5.1 Basics and Algorithms

5.2 K-means Clustering

5.3 Hierarchical Clustering

5.4 DBSCAN Clustering

5.5 Issues : Evaluation, Scalability, Comparison

5.1 Basics and Algorithms

What is Cluster Analysis?

- Cluster: A collection of data objects
  - similar (or related) to one another within the same group
  - dissimilar (or unrelated) to the objects in other groups
- Cluster analysis (or *clustering*, *data segmentation, …*)
  - Finding similarities between data according to the characteristics found in the data and grouping similar data objects into clusters
- Unsupervised learning: no predefined classes (i.e., *learning by observations* vs. learning by examples: supervised)
- Typical applications
  - As a stand-alone tool to get insight into data distribution
  - As a preprocessing step for other algorithms

- Cluster analysis is a multivariate method which aims to classify a sample of subjects (or objects) on the basis of a set of measured variables into a number of different groups such that similar subjects are placed in the same group.
- An example where this might be used is in the field of psychiatry, where the characterization of patients on the basis of clusters of symptoms can be useful in the identification of an appropriate form of therapy.
- In marketing, it may be useful to identify distinct groups of potential customers so that, for example, advertising can be appropriately targeted.

WARNING ABOUT CLUSTER ANALYSIS

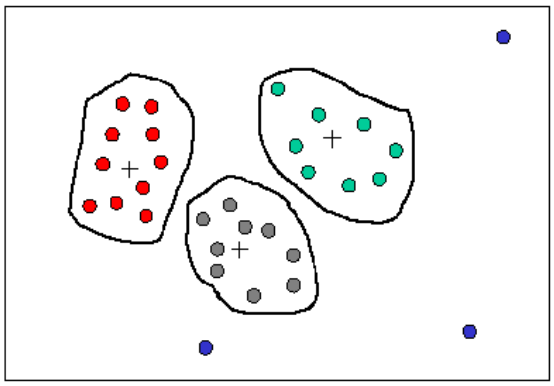
- Cluster analysis has no mechanism for differentiating between relevant and irrelevant variables.
- Therefore the choice of variables included in a cluster analysis must be underpinned by conceptual considerations.
- This is very important because the clusters formed can be very dependent on the variables included.

What is Clustering in Data Mining?

Clustering is a process of partitioning a set of data (or objects) in a set of meaningful sub-classes, called **clusters**

Helps users understand the natural grouping or structure in a data set

- Cluster:
  - a collection of data objects that are "similar" to one another and thus can be treated collectively as one group
  - but as a collection, they are sufficiently different from other groups



Applications of Cluster Analysis

- Data reduction
  - Summarization: Preprocessing for regression, PCA, classification, and association analysis
  - Compression: Image processing: vector quantization
- Hypothesis generation and testing
- Prediction based on groups
  - Cluster & find characteristics/patterns for each group
- Finding K-nearest Neighbors
  - Localizing search to one or a small number of clusters
- Outlier detection: Outliers are often viewed as those “far away” from any cluster

7

Clustering for Data Understanding and Applications

- Biology: taxonomy of living things: kingdom, phylum, class, order, family, genus and species
- Information retrieval: document clustering
- Land use: Identification of areas of similar land use in an earth observation database
- Marketing: Help marketers discover distinct groups in their customer bases, and then use this knowledge to develop targeted marketing programs
- 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
- Climate: understanding earth climate, find patterns of atmospheric and ocean
- Economic Science: market resarch

Clustering as a Preprocessing Tool (Utility)

- Summarization:
  - Preprocessing for regression, PCA, classification, and association analysis
- Compression:
  - Image processing: vector quantization
- Finding K-nearest Neighbors
  - Localizing search to one or a small number of clusters
- Outlier detection
  - Outliers are often viewed as those “far away” from any cluster

Basic Steps to Develop a Clustering Task

- Feature selection / Preprocessing
  - Select info concerning the task of interest
  - Minimal information redundancy
  - May need to do normalization/standardization
- Distance/Similarity measure
  - Similarity of two feature vectors
- Clustering criterion
  - Expressed via a cost function or some rules
- Clustering algorithms
  - Choice of algorithms
- Validation of the results
- Interpretation of the results with applications

Distance or Similarity Measures

$X = \langle x_1, x_2, \cdots, x_n \rangle$ $Y = \langle y_1, y_2, \cdots, y_n \rangle$

- Common Distance Measures:
  - Manhattan distance:

$$dist(X,Y) = |x_1 - y_1| + |x_2 - y_2| + \cdots + |x_n - y_n|$$

  - Euclidean distance:

$$dist(X,Y) = \sqrt{(x_1 - y_1)^2 + \cdots + (x_n - y_n)^2}$$

  - Cosine similarity:

$$dist(X,Y) = 1 - sim(X,Y)$$

$$sim(X,Y) = \frac{\sum_i (x_i \times y_i)}{\sqrt{\sum_i x_i^2 \times \sum_i y_i^2}}$$

Quality: What Is Good Clustering?

- A good clustering method will produce high quality clusters
  - high intra-class similarity: cohesive within clusters
  - low inter-class similarity: distinctive between clusters
- The quality of a clustering method depends on
  - the similarity measure used
  - its implementation, and
  - Its ability to discover some or all of the hidden patterns

Approaches to cluster analysis

- There are a number of different methods that can be used to carry out a cluster analysis; these methods can be classified as follows:
  - Non-hierarchical methods
    - Partitioning approach (**k-means**)
    - Density-based approach. (**DBSCAN**)
  - Hierarchical methods
    – **Agglomerative** methods, in which subjects start in their own separate cluster.
    – **Divisive** methods, in which all subjects start in the same cluster and the above strategy is applied in reverse until every subject is in a separate cluster.

Major Clustering Approaches

- Partitioning approach:
  - Construct various partitions and then evaluate them by some criterion, e.g., minimizing the sum of square errors
  - Typical methods: k-means, k-medoids, CLARANS
- Density-based approach:
  - Based on connectivity and density functions
  - Typical methods: DBSCAN, OPTICS, DenClue
- Hierarchical approach:
  - Create a hierarchical decomposition of the set of data (or objects) using some criterion
  - Typical methods: Diana, Agnes, BIRCH, CAMELEON
- Model-based:
  - A model is hypothesized for each of the clusters and tries to find the best fit of that model to each other
  - Typical methods: EM, SOM, COBWEB
- Grid-based approach:
  - based on a multiple-level granularity structure
  - Typical methods: STING, WaveCluster, CLIQUE

Measure the Quality of Clustering

- Dissimilarity/Similarity metric
  - Similarity is expressed in terms of a distance function, typically metric: $d(i, j)$
  - The definitions of distance functions are usually rather different for interval-scaled, boolean, categorical, ordinal ratio, and vector variables
  - Weights should be associated with different variables based on applications and data semantics
- Quality of clustering:
  - There is usually a separate "quality" function that measures the "goodness" of a cluster.
  - It is hard to define "similar enough" or "good enough"
    - The answer is typically highly subjective

Considerations for Cluster Analysis

- Partitioning criteria
  - Single level vs. hierarchical partitioning (often, multi-level hierarchical partitioning is desirable)
- Separation of clusters
  - Exclusive (e.g., one customer belongs to only one region) vs. non-exclusive (e.g., one document may belong to more than one class)
- Similarity measure
  - Distance-based (e.g., Euclidian, road network, vector) vs. connectivity-based (e.g., density or contiguity)
- Clustering space
  - Full space (often when low dimensional) vs. subspaces (often in high-dimensional clustering)

Requirements and Challenges

- Scalability
  - Clustering all the data instead of only on samples
- Ability to deal with different types of attributes
  - Numerical, binary, categorical, ordinal, linked, and mixture of these
- Constraint-based clustering
  - User may give inputs on constraints
  - Use domain knowledge to determine input parameters
- Interpretability and usability
- Others
  - Discovery of clusters with arbitrary shape
  - Ability to deal with noisy data
  - Incremental clustering and insensitivity to input order
  - High dimensionality

5.2 K-means Clustering

Partitioning Algorithms: Basic Concept

- Partitioning method: Partitioning a database ***D*** of ***n*** objects into a set of ***k*** clusters, such that the sum of squared distances is minimized (where c_i is the centroid or medoid of cluster C_i)

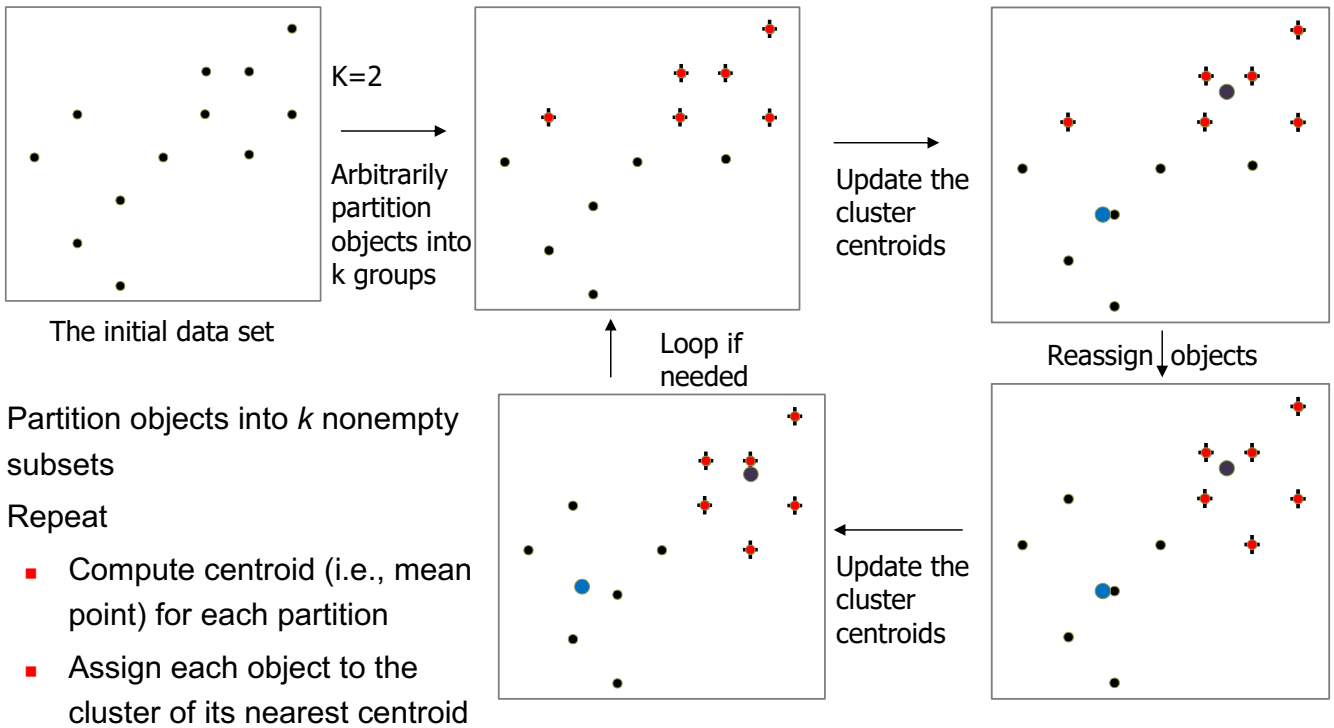
$$E = \Sigma_{i=1}^{k} \Sigma_{p \in C_i} (p - c_i)^2$$

- Given *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* (MacQueen'67, Lloyd'57/'82): Each cluster is represented by the center of the cluster
  - *k-medoids* or PAM (Partition around medoids) (Kaufman & Rousseeuw'87): 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 four steps:
  - Partition objects into *k* nonempty subsets
  - Compute seed points as the centroids of the clusters of the current partitioning (the centroid is the center, i.e., *mean point*, of the cluster)
  - Assign each object to the cluster with the nearest seed point
  - Go back to Step 2, stop when the assignment does not change

An Example of *K-Means* Clustering



- Partition objects into *k* nonempty subsets
- Repeat
  - Compute centroid (i.e., mean point) for each partition
  - Assign each object to the cluster of its nearest centroid
- Until no change

Exercise 1. K-means clustering

Use the k-means algorithm and Euclidean distance to cluster the following 8 examples into 3 clusters:

A1=(2,10), A2=(2,5), A3=(8,4), A4=(5,8), A5=(7,5), A6=(6,4), A7=(1,2), A8=(4,9).

The distance matrix based on the Euclidean distance is given below:

	A1	A2	A3	A4	A5	A6	A7	A8
A1	0	$\sqrt{25}$	$\sqrt{36}$	$\sqrt{13}$	$\sqrt{50}$	$\sqrt{52}$	$\sqrt{65}$	$\sqrt{5}$
A2		0	$\sqrt{37}$	$\sqrt{18}$	$\sqrt{25}$	$\sqrt{17}$	$\sqrt{10}$	$\sqrt{20}$
A3			0	$\sqrt{25}$	$\sqrt{2}$	$\sqrt{2}$	$\sqrt{53}$	$\sqrt{41}$
A4				0	$\sqrt{13}$	$\sqrt{17}$	$\sqrt{52}$	$\sqrt{2}$
A5					0	$\sqrt{2}$	$\sqrt{45}$	$\sqrt{25}$
A6						0	$\sqrt{29}$	$\sqrt{29}$
A7							0	$\sqrt{58}$
A8								0

Suppose that the initial seeds (centers of each cluster) are A1, A4 and A7. Run the k-means algorithm for
1 epoch only. At the end of this epoch show:
a) The new clusters (i.e. the examples belonging to each cluster)
b) The centers of the new clusters

Solution:

a)

d(a,b) denotes the Eucledian distance between a and b. It is obtained directly from the distance matrix or calculated as follows:

d(a,b)=sqrt((xb-xa)2+(yb-ya)2))

seed1=A1=(2,10), seed2=A4=(5,8), seed3=A7=(1,2)

epoch1 – start:

A1:
d(A1, seed1)=0 as A1 is seed1
d(A1, seed2)= $\sqrt{13}$ >0
d(A1, seed3)= $\sqrt{65}$ >0
➔A1 ∈ cluster1

A2:
d(A2,seed1)= $\sqrt{25}$ = 5
d(A2, seed2)= $\sqrt{18}$ = 4.24
d(A2, seed3)= $\sqrt{10}$ = 3.16 ← smaller
➔ A2 ∈ cluster3

A3:
d(A3, seed1)= $\sqrt{36}$ = 6
d(A3, seed2)= $\sqrt{25}$ = 5 ← smaller
d(A3, seed3)= $\sqrt{53}$ = 7.28
➔ A3 ∈ cluster2

A4:
d(A4, seed1)= $\sqrt{13}$
d(A4, seed2)=0 as A4 is seed2
d(A4, seed3)= $\sqrt{52}$ >0
➔ A4 ∈ cluster2

A5:
d(A5, seed1)= $\sqrt{50}$ = 7.07

A6:
d(A6, seed1)= $\sqrt{52}$ = 7.21

d(A5, seed2)= $\sqrt{13}$ = 3.60 ← smaller
d(A5, seed3)= $\sqrt{45}$ = 6.70
➔ A5 ∈ cluster2

d(A6, seed2)= $\sqrt{17}$ = 4.12 ← smaller
d(A6, seed3)= $\sqrt{29}$ = 5.38
➔ A6 ∈ cluster2

A7:
d(A7, seed1)= $\sqrt{65}$ >0
d(A7, seed2)= $\sqrt{52}$ >0
d(A7, seed3)=0 as A7 is seed3
➔ A7 ∈ cluster3

A8:
d(A8, seed1)= $\sqrt{5}$
d(A8, seed2)= $\sqrt{2}$ ← smaller
d(A8, seed3)= $\sqrt{58}$
➔ A8 ∈ cluster2

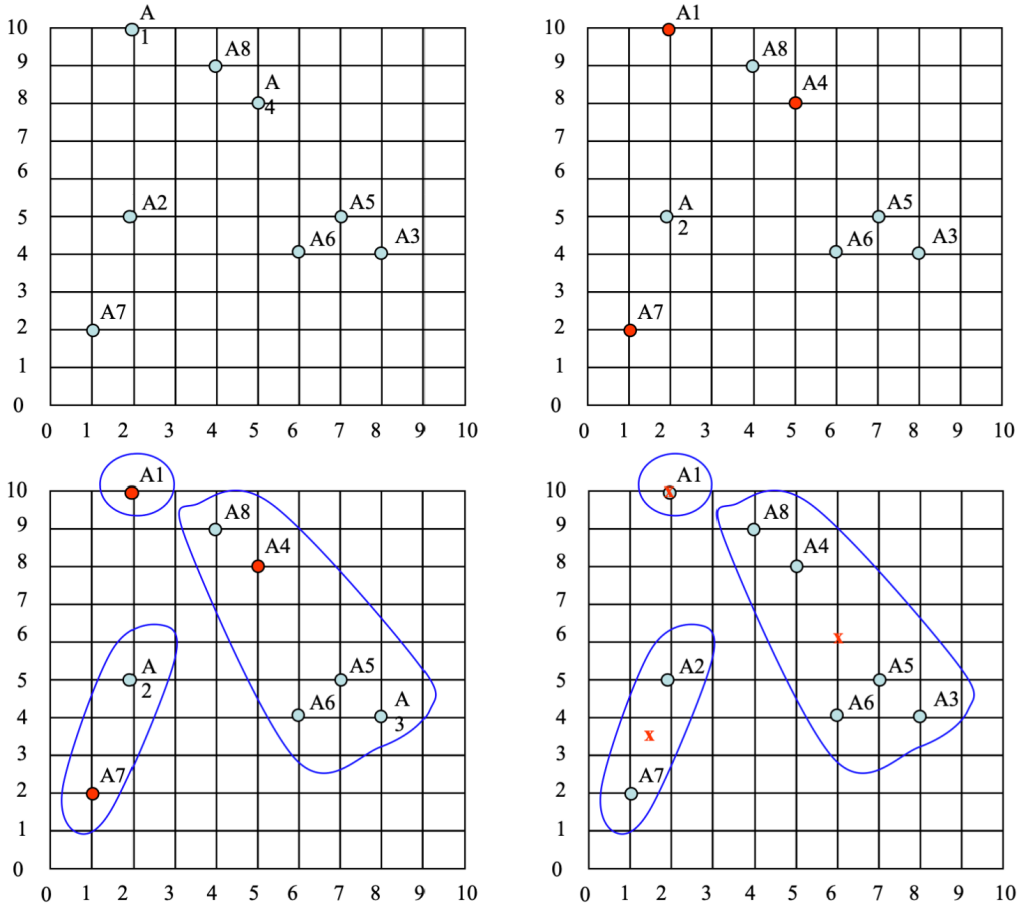
end of epoch1

new clusters: 1: {A1}, 2: {A3, A4, A5, A6, A8}, 3: {A2, A7}

new clusters: 1: {A1}, 2: {A3, A4, A5, A6, A8}, 3: {A2, A7}

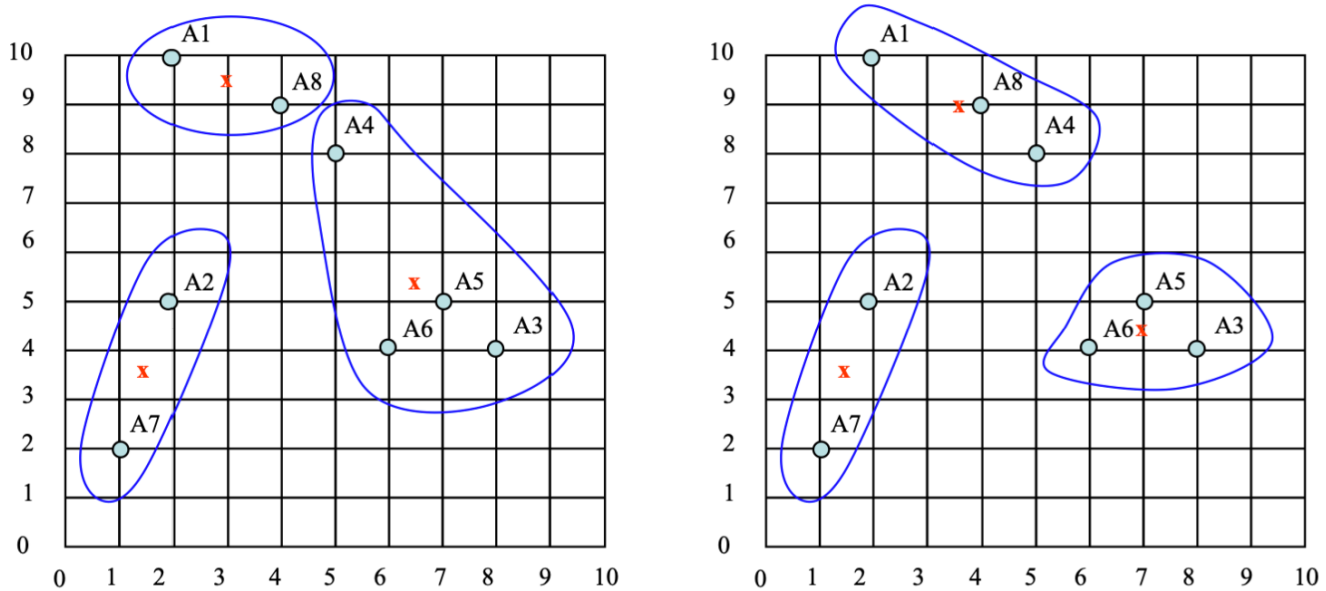
b) centers of the new clusters:
C1= (2, 10), C2= ((8+5+7+6+4)/5, (4+8+5+4+9)/5) = (6, 6),
C3= ((2+1)/2, (5+2)/2) = (1.5, 3.5)

c) Draw a 10 by 10 space with all the 8 points and show the clusters after the first epoch and the new centroids.



d) How many more iterations are needed to converge? Draw the result for each epoch.

d)
We would need two more epochs. After the 2nd epoch the results would be:
1: {A1, A8}, 2: {A3, A4, A5, A6}, 3: {A2, A7}
with centers C1=(3, 9.5), C2=(6.5, 5.25) and C3=(1.5, 3.5).
After the 3rd epoch, the results would be:
1: {A1, A4, A8}, 2: {A3, A5, A6}, 3: {A2, A7}
with centers C1=(3.66, 9), C2=(7, 4.33) and C3=(1.5, 3.5).



Example

As a simple illustration of a k-means algorithm, consider the following data set consisting of the scores of two variables on each of seven individuals. Find 2 clusters from this.

Subject	A	B
1	1.0	1.0
2	1.5	2.0
3	3.0	4.0
4	5.0	7.0
5	3.5	5.0
6	4.5	5.0
7	3.5	4.5

K-Means Example: Document Clustering

Initial arbitrary assignment (k=3):
C1 = {D1,D2},
C2 = {D3,D4},
C3 = {D5,D6}

	T1	T2	T3	T4	T5
D1	0	3	3	0	2
D2	4	1	0	1	2
D3	0	4	0	0	2
D4	0	3	0	3	3
D5	0	1	3	0	1
D6	2	2	0	0	4
D7	1	0	3	2	0
D8	3	1	0	0	2
Cluster Centroids →	4/2	5/2	5/2		

Now compute the similarity (or distance) of each item to each cluster, resulting a cluster-document similarity matrix (here we use dot product as the similarity measure).

	D1	D2	D3	D4	D5	D6	D7	D8
C1	29/2	29/2	24/2	27/2	17/2	32/2	15/2	24/2
C2	31/2	20/2	38/2	45/2	12/2	34/2	6/2	17/2
C3	28/2	21/2	22/2	24/2	17/2	30/2	11/2	19/2

Example (Continued)

	D1	D2	D3	D4	D5	D6	D7	D8
C1	29/2	29/2	24/2	27/2	17/2	32/2	15/2	24/2
C2	31/2	20/2	38/2	45/2	12/2	34/2	6/2	17/2
C3	28/2	21/2	22/2	24/2	17/2	30/2	11/2	19/2

For each document, reallocate the document to the cluster to which it has the highest similarity (shown in red in the above table). After the reallocation we have the following new clusters. Note that the previously unassigned D7 and D8 have been assigned, and that D1 and D6 have been reallocated from their original assignment.

C1 = {D2,D7,D8}, C2 = {D1,D3,D4,D6}, C3 = {D5}

This is the end of first iteration (i.e., the first reallocation). Next, we repeat the process for another reallocation…

Example (Continued)

Now compute new cluster centroids using the original document-term matrix

C1 = {D2,D7,D8}, C2 = {D1,D3,D4,D6}, C3 = {D5}

	T1	T2	T3	T4	T5
D1	0	3	3	0	2
D2	4	1	0	1	2
D3	0	4	0	0	2
D4	0	3	0	3	3
D5	0	1	3	0	1
D6	2	2	0	0	4
D7	1	0	3	2	0
D8	3	1	0	0	2
C1	8/3	2/3	3/3	3/3	4/3
C2	2/4	12/4	3/4	3/4	11/4
C3	0/1	1/1	3/1	0/1	1/1

This will lead to a new cluster-doc similarity matrix similar to previous slide. Again, the items are reallocated to clusters with highest similarity.

	D1	D2	D3	D4	D5	D6	D7	D8
C1	7.67	**15.01**	5.34	9.00	5.00	**12.00**	7.67	**11.34**
C2	**16.75**	11.25	**17.50**	**19.50**	8.00	6.68	4.25	10.00
C3	14.00	3.00	6.00	6.00	**11.00**	9.34	**9.00**	3.00

New assignment → **C1 = {D2,D6,D8}, C2 = {D1,D3,D4}, C3 = {D5,D7}**

Note: This process is now repeated with new clusters. However, the next iteration in this example Will show no change to the clusters, thus terminating the algorithm.

K-Means Algorithm

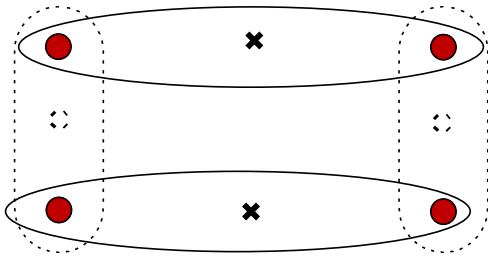
- Strength of the *k-means*:
  - *Relatively efficient*: $O(tkn)$, where n is # of objects, k is # of clusters, and t is # of iterations. Normally, $k, t << n$
  - Often terminates at a *local optimum*
- Weakness of the *k-means*:
  - Applicable only when *mean* is defined; what about categorical data?
  - Need to specify k, the *number* of clusters, in advance
  - Unable to handle noisy data and *outliers*
- *Variations of K-Means usually differ in:*
  - Selection of the initial k means
  - Distance or similarity measures used
  - Strategies to calculate cluster means

Comments on the *K-Means* Method

- Strength: *Efficient*: *O*(*tkn*), where *n* is # objects, *k* is # clusters, and *t* is # iterations. Normally, *k*, *t* << *n*.
  - Comparing: PAM: $O(k(n-k)^2)$, CLARA: $O(ks^2 + k(n-k))$
- Comment: Often terminates at a *local optimal*.
- Weakness
  - Applicable only to objects in a continuous n-dimensional space
    - Using the k-modes method for categorical data
    - In comparison, k-medoids can be applied to a wide range of data
  - Need to specify *k,* the *number* of clusters, in advance (there are ways to automatically determine the best k (see Hastie et al., 2009)
  - Sensitive to noisy data and *outliers*
  - Not suitable to discover clusters with *non-convex shapes*

Variations of the *K-Means* Method

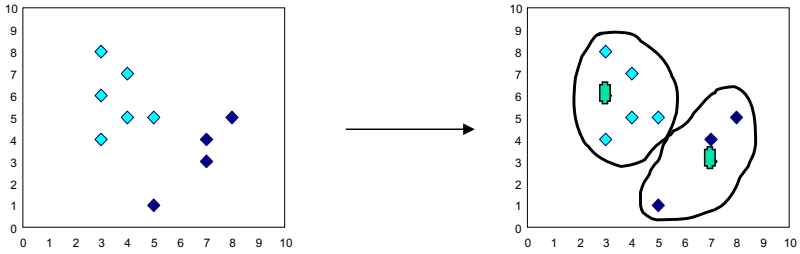
- Most of the variants of the *k-means* which differ in
  - Selection of the initial *k* means
  - Dissimilarity calculations
  - Strategies to calculate cluster means



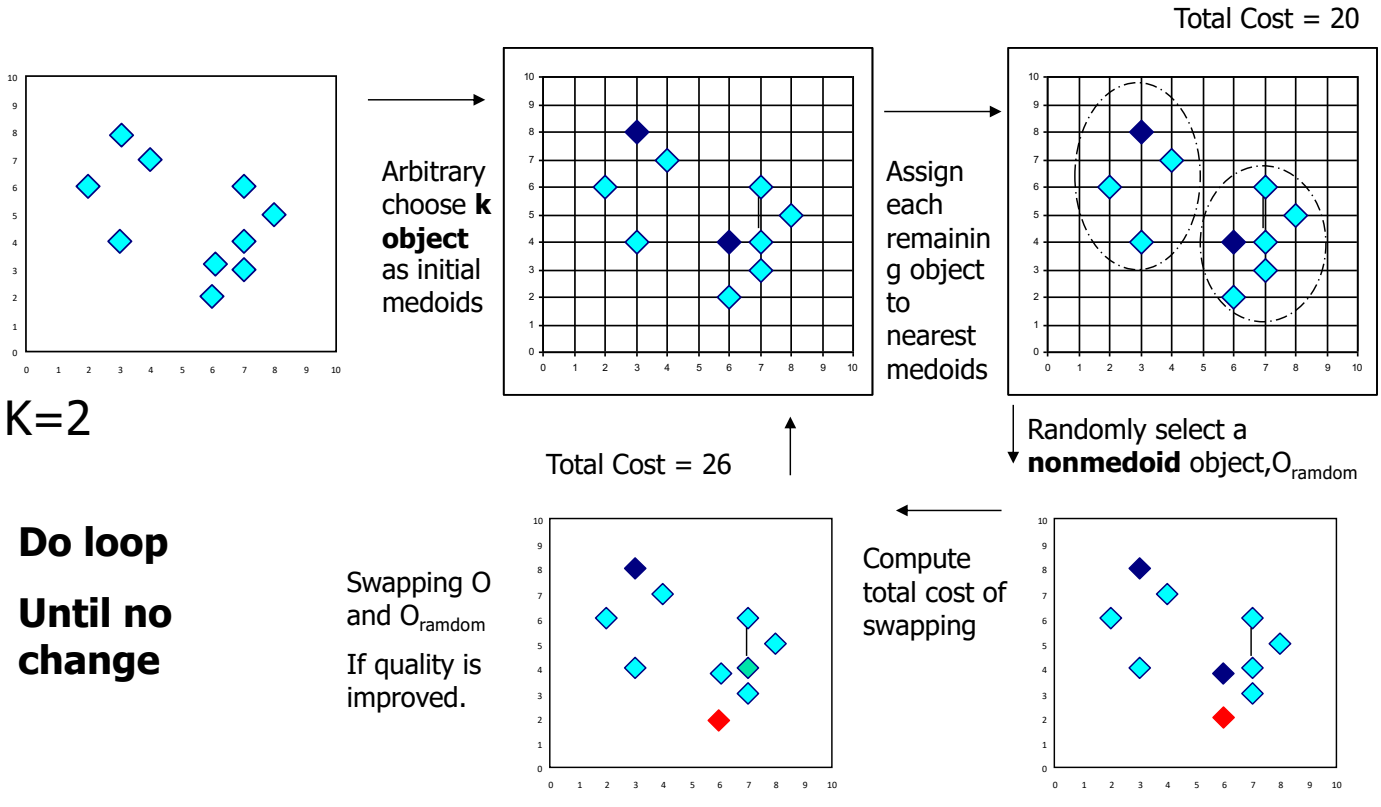
- Handling categorical data: *k-modes*
  - Replacing means of clusters with modes
  - Using new dissimilarity measures to deal with categorical objects
  - Using a frequency-based method to update modes of clusters
  - A mixture of categorical and numerical data: *k-prototype* method

What Is the Problem of the K-Means Method?

- The k-means algorithm is sensitive to outliers !
  - Since an object with an extremely large value may substantially distort the distribution of the data
- K-Medoids: Instead of taking the **mean** value of the object in a cluster as a reference point, **medoids** can be used, which is the **most centrally located** object in a cluster



PAM: A Typical K-Medoids Algorithm



The K-Medoid Clustering Method

- *K-Medoids* Clustering: Find *representative* objects (medoids) in clusters
  - *PAM* (Partitioning Around Medoids, Kaufmann & Rousseeuw 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 (due to the computational complexity)
- Efficiency improvement on PAM
  - *CLARA* (Kaufmann & Rousseeuw, 1990): PAM on samples
  - *CLARANS* (Ng & Han, 1994): Randomized re-sampling

A Disk Version of *k*-means

- *k*-means can be implemented with data on disk
  - In each iteration, it scans the database once
  - The centroids are computed incrementally
- It can be used to cluster large datasets that do not fit in main memory
- We need to control the number of iterations
  - In practice, a limited is set (< 50)
- There are better algorithms that scale up for large data sets, e.g., BIRCH

BIRCH

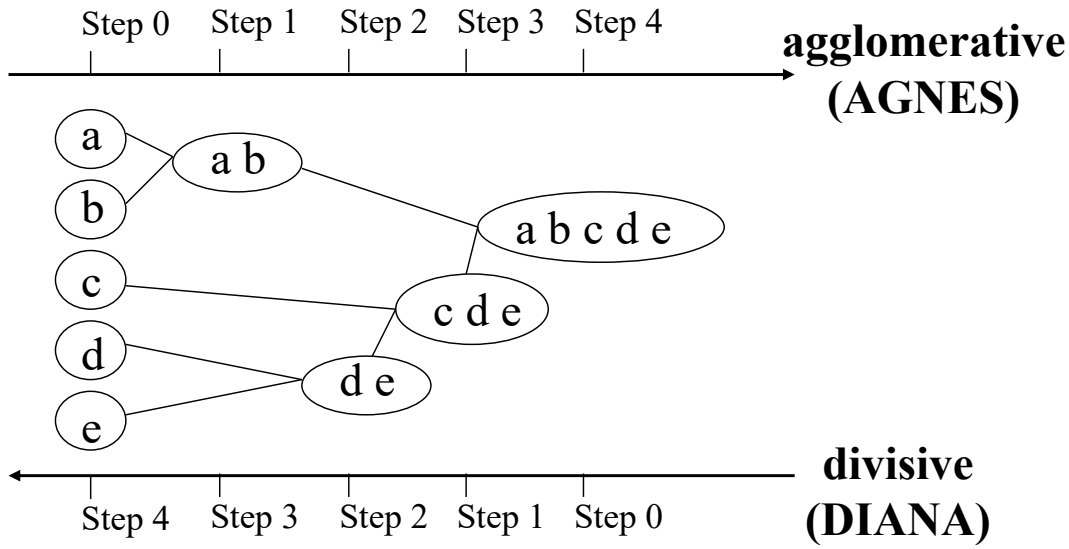
- Designed for very large data sets
  - Time and memory are limited
  - Incremental and dynamic clustering of incoming objects
  - Only one scan of data is necessary
  - Does not need the whole data set in advance
- Two key phases:
  - Scans the database to build an in-memory tree
  - Applies clustering algorithm to cluster the leaf nodes

43

5.3 Hierarchical Clustering

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

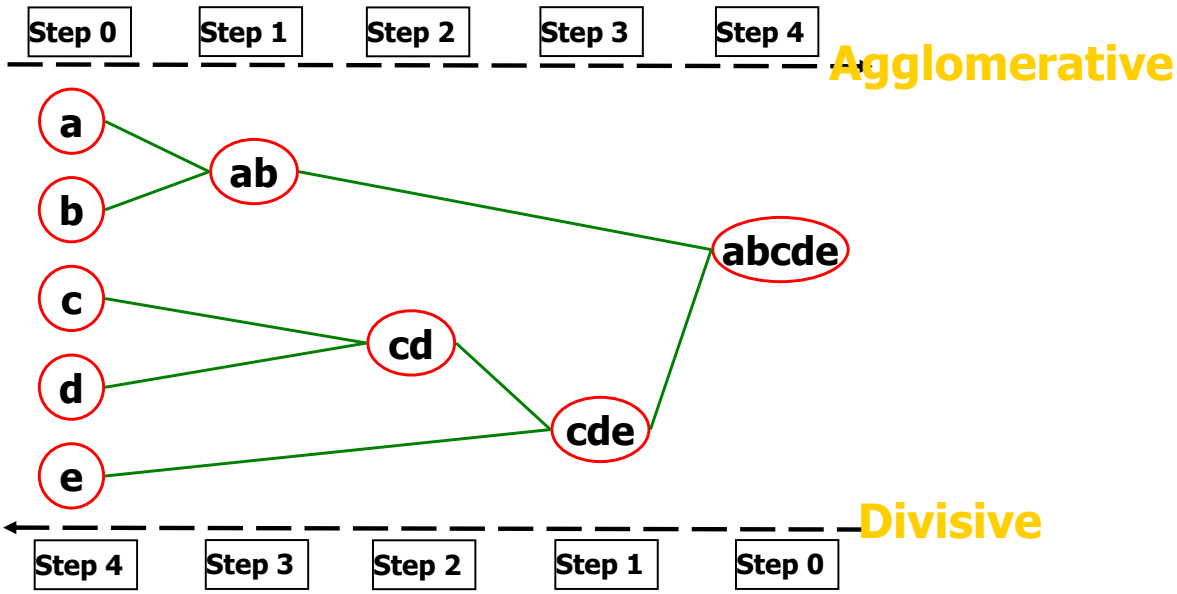


Hierarchical Clustering Algorithms

- Two main types of hierarchical clustering
  - **Agglomerative:**
    - Start with the points as individual clusters
    - At each step, merge the closest pair of clusters until only one cluster (or **k** clusters) left
  - **Divisive:**
    - Start with one, all-inclusive cluster
    - At each step, split a cluster until each cluster contains a point (or there are **k** clusters)
- Traditional hierarchical algorithms use a similarity or distance matrix
  - Merge or split one cluster at a time

Hierarchical Clustering Algorithms

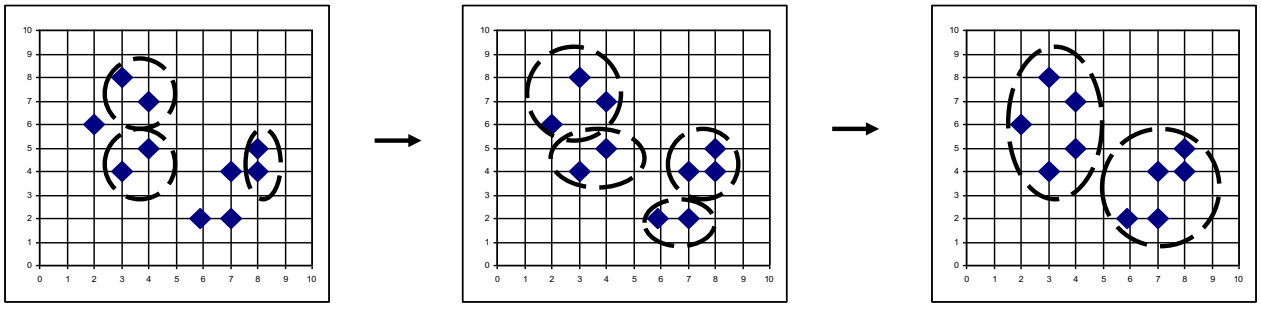
- Use dist / sim matrix as clustering criteria
  - does not require the no. of clusters as input, but needs a termination condition



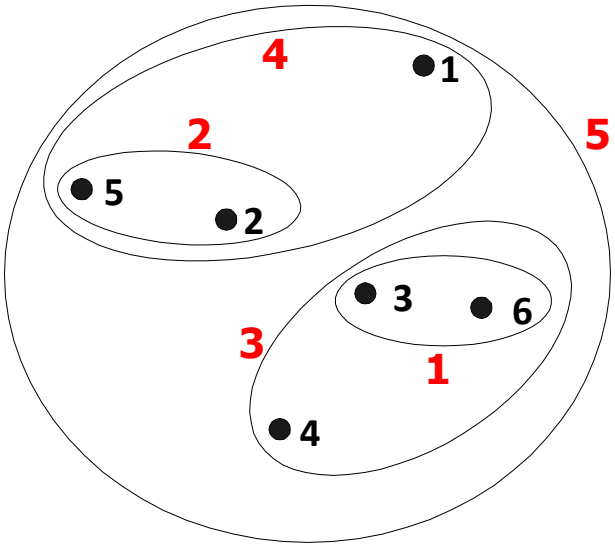
47

AGNES (Agglomerative Nesting)

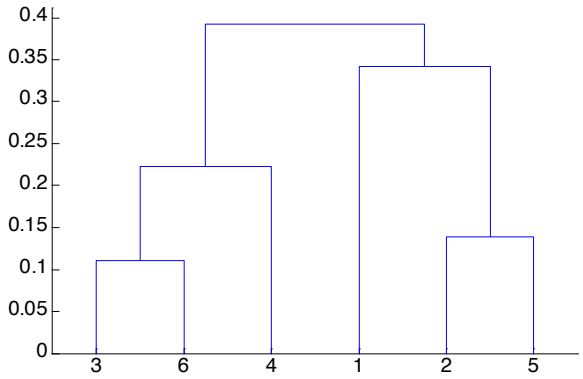
- Introduced in Kaufmann and Rousseeuw (1990)
- Implemented in statistical packages, e.g., Splus
- Use the **single-link** method and the dissimilarity matrix
- Merge nodes that have the least dissimilarity
- Go on in a non-descending fashion
- Eventually all nodes belong to the same cluster



Hierarchical Agglomerative Clustering
:: Example



Nested Clusters

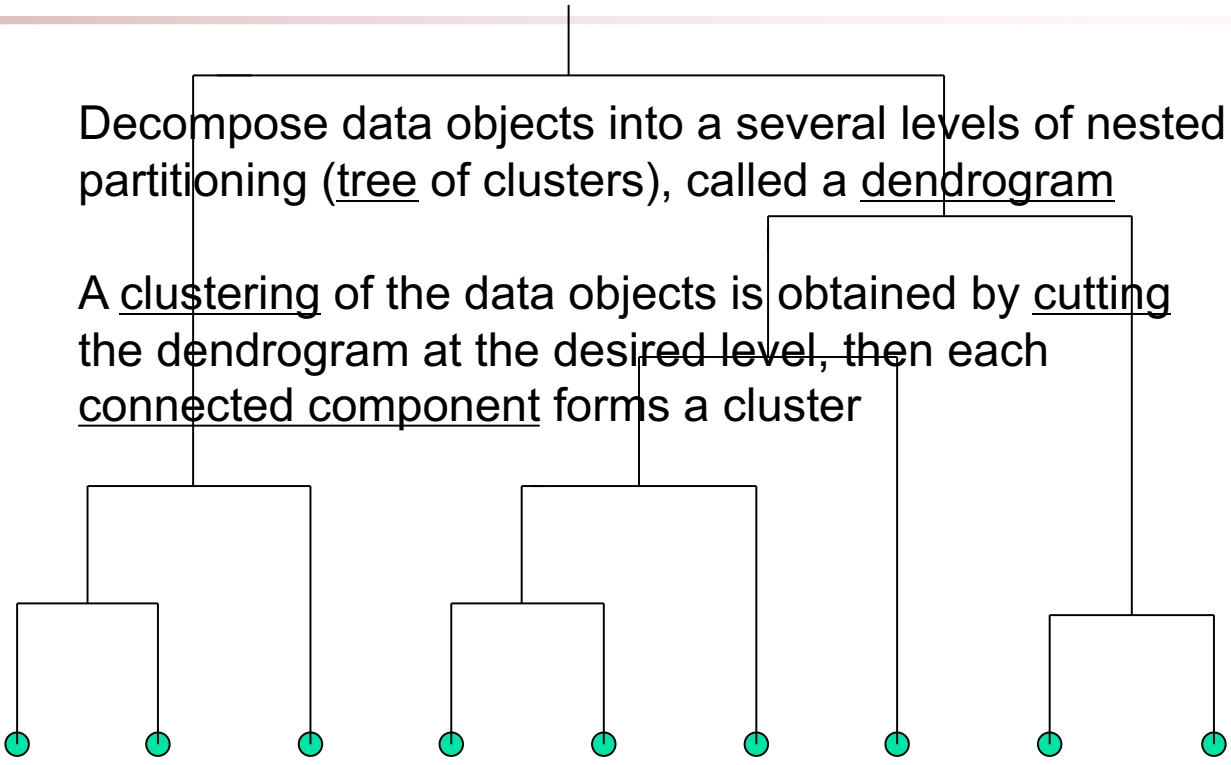


Dendrogram

Dendrogram: Shows How Clusters are Merged

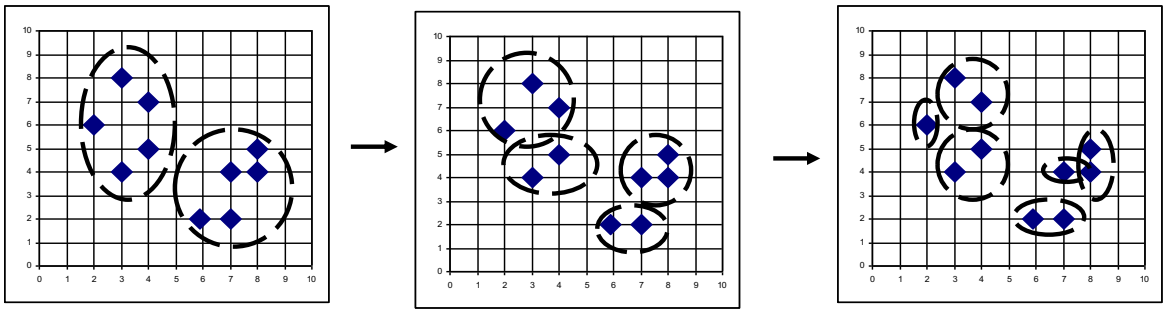
Decompose data objects into a several levels of nested partitioning (tree of clusters), called a dendrogram

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

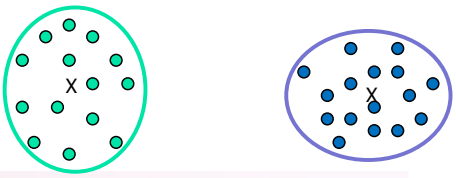


DIANA (Divisive Analysis)

- 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



Distance between Clusters



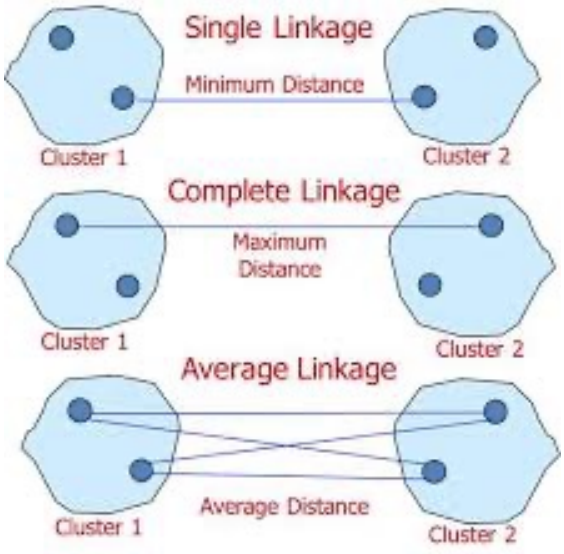
- Single link: smallest distance between an element in one cluster and an element in the other, i.e., $dist(K_i, K_j) = min(t_{ip}, t_{jq})$
- Complete link: largest distance between an element in one cluster and an element in the other, i.e., $dist(K_i, K_j) = max(t_{ip}, t_{jq})$
- Average: avg distance between an element in one cluster and an element in the other, i.e., $dist(K_i, K_j) = avg(t_{ip}, t_{jq})$
- Centroid: distance between the centroids of two clusters, i.e., $dist(K_i, K_j) = dist(C_i, C_j)$
- Medoid: distance between the medoids of two clusters, i.e., $dist(K_i, K_j) = dist(M_i, M_j)$
  - Medoid: a chosen, centrally located object in the cluster

Distance Between Two Clusters

NOTE: These are distance between clusters not point

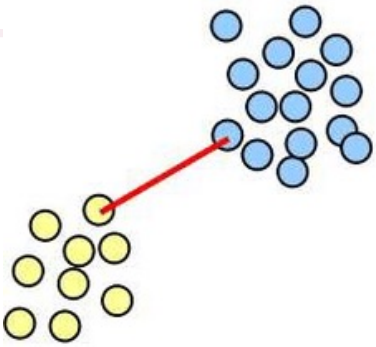
- The basic procedure varies based on the method used to determine inter-cluster distances or similarities

- Different methods results in different variants of the algorithm
  - Single link
  - Complete link
  - Average link
  - Ward's method
  - Etc.



Single Link Method

- The distance between two clusters is the distance between two closest data points in the two clusters, one data point from each cluster
- It can find arbitrarily shaped clusters, but
  - It may cause the undesirable "chain effect" due to noisy points



single-link

Two natural clusters are split into two

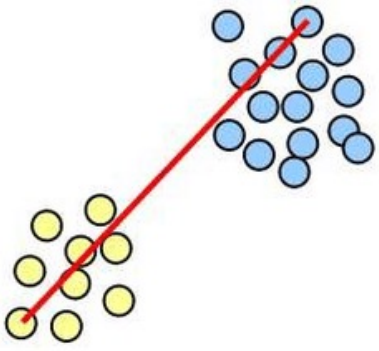
Distance between two clusters

- **Single-link distance** between clusters C_i and C_j is the ***minimum distance*** between any object in C_i and any object in C_j
  - The distance is defined by the two **most similar** objects

$$D_{sl}(C_i, C_j) = \min_{x,y} \{ d(x, y) \mid x \in C_i, y \in C_j \}$$

Complete Link Method

- The distance between two clusters is the distance of two furthest data points in the two clusters
- It is sensitive to outliers because they are far away



complete-link

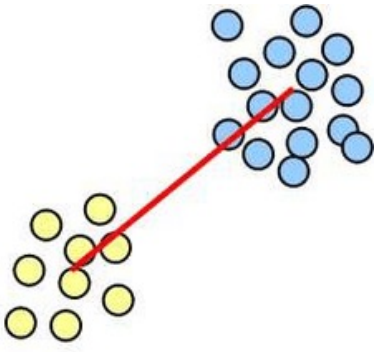
Distance between two clusters

- **Complete-link distance** between clusters C_i and C_j is the ***maximum distance*** between any object in C_i and any object in C_j
  - The distance is defined by the two **least similar** objects

$$D_{cl}(C_i, C_j) = \max_{x,y} \{ d(x, y) \mid x \in C_i, y \in C_j \}$$

Average link and centroid methods

- Average link: A compromise between
  - the sensitivity of complete-link clustering to outliers and
  - the tendency of single-link clustering to form long chains that do not correspond to the intuitive notion of clusters as compact, spherical objects
  - In this method, the distance between two clusters is the average distance of all pair-wise distances between the data points in two clusters.



average-link

Extensions to Hierarchical Clustering

- Major weakness of agglomerative clustering methods
  - Can never undo what was done previously
  - Do not scale well: time complexity of at least $O(n^2)$, where n is the number of total objects
- Integration of hierarchical & distance-based clustering
  - BIRCH (1996): uses CF-tree and incrementally adjusts the quality of sub-clusters
  - CHAMELEON (1999): hierarchical clustering using dynamic modeling

5.4 DBSCAN Clustering

Density-Based Methods

- Partitioning and hierarchical methods are designed to find spherical-shaped clusters.
- They have difficulty finding clusters of arbitrary shape such as the "S" shape and oval clusters
- To find clusters of arbitrary shape, alternatively, we can model clusters as dense regions in the data space, separated by sparse regions.
- This is the main strategy behind *density-based clustering methods*, which can discover clusters of nonspherical shape.

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 interesting studies:
  - DBSCAN: Ester, et al. (KDD'96)
  - OPTICS: Ankerst, et al (SIGMOD'99).
  - DENCLUE: Hinneburg & D. Keim (KDD'98)
  - CLIQUE: Agrawal, et al. (SIGMOD'98) (more grid-based)

- *"How can we find dense regions in density-based clustering?"*
  - The *density* of an object ***o*** can be measured by the number of objects close to ***o***
- ***core objects*** : objects that have dense neighborhoods
- *"How does* **DBSCAN** *quantify the neighborhood of an object?"*
  - A user-specified para- meter ε > 0 is used to specify the radius of a neighborhood we consider for every object.
  - The **ε-neighborhood** of an object ***o*** is the space within a radius ε centered at ***o***.

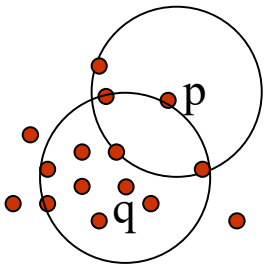
- Given a set, *D*, of objects, we can identify all core objects with respect to the given parameters, ε and *MinPts*.
- The clustering task is therein reduced to using core objects and their neighborhoods to form dense regions, where the dense regions are clusters.
- For a core object ***q*** and an object ***p***, we say that ***p*** is **directly density-reachable** from ***q*** (with respect to ε and *MinPts*) if ***p*** is within the ε-neighborhood of ***q***.
- Clearly, an object ***p*** is directly density-reachable from another object ***q*** if and only if ***q*** is a core object and ***p*** is in the ε-neighborhood of ***q***.
- Using the directly density-reachable relation, a core object can "bring" all objects from its ε-neighborhood into a dense region.

- To connect core objects as well as their neighbors in a dense region, **DBSCAN** uses the notion of density-connectedness.
- Two objects ***p*1**,***p*2** ∈ *D* are **density-connected** with respect to ε and *MinPts* if there is an object ***q*** ∈ *D* such that both ***p*1** and ***p*2** are density- reachable from ***q*** with respect to ε and *MinPts*.
- Unlike density-reachability, density- connectedness is an equivalence relation.
- It is easy to show that, for objects ***o*1**, ***o*2**, and ***o*3**, if ***o*1** and ***o*2** are density-connected, and ***o*2** and ***o*3** are density-connected, then so are ***o*1** and ***o*3**.

Density-Based Clustering: Basic Concepts

- 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* w.r.t. *Eps*, *MinPts* if
  - *p* belongs to $N_{Eps}(q)$
  - core point condition:

$$|N_{Eps}(q)| \geq MinPts$$

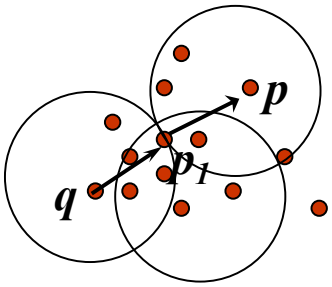


MinPts = 5

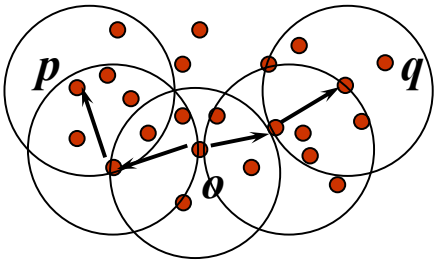
Eps = 1 cm

Density-Reachable and Density-Connected

- Density-reachable:
  - A point *p* is density-reachable from a point *q* w.r.t. *Eps*, *MinPts* if there is a chain of points p_1, …, p_n, p_1 = *q*, p_n = *p* such that p_{i+1} is directly density-reachable from p_i

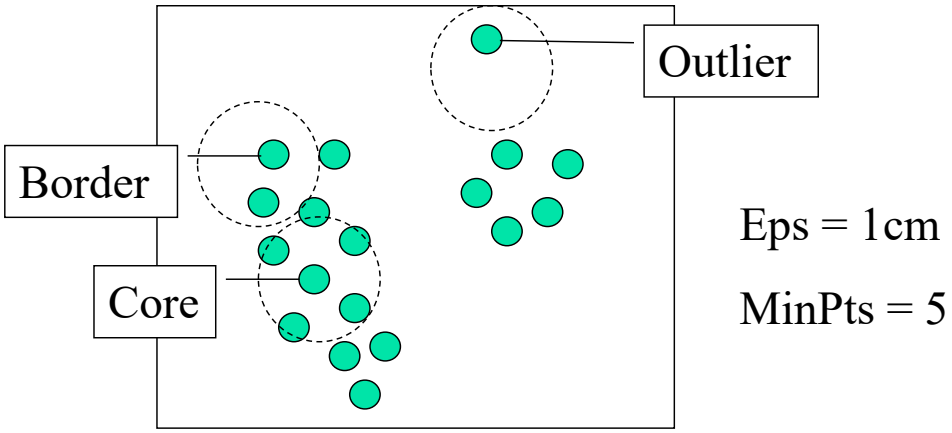


- Density-connected
  - A point *p* is density-connected to a point *q* w.r.t. *Eps*, *MinPts* if there is a point *o* such that both, *p* and *q* are density-reachable from *o* w.r.t. *Eps* and *MinPts*



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



DBSCAN: The Algorithm

- Arbitrary select a point *p*
- Retrieve all points density-reachable from *p* w.r.t. *Eps* and *MinPts*
- If *p* is a core point, a cluster is formed
- If *p* is a border point, no points are density-reachable from *p* and DBSCAN visits the next point of the database
- Continue the process until all of the points have been processed

5.5 Issues : Evaluation, Scalability, Comparison

Determine the Number of Clusters

- Empirical method
  - # of clusters ≈√n/2 for a dataset of n points
- Elbow method
  - Use the turning point in the curve of sum of within cluster variance w.r.t the # of clusters
- Cross validation method
  - Divide a given data set into *m* parts
  - Use *m* – 1 parts to obtain a clustering model
  - Use the remaining part to test the quality of the clustering
    - E.g., For each point in the test set, find the closest centroid, and use the sum of squared distance between all points in the test set and the closest centroids to measure how well the model fits the test set
  - For any k > 0, repeat it *m* times, compare the overall quality measure w.r.t. different *k's*, and find # of clusters that fits the data the best

Measuring Clustering Quality

- Two methods: extrinsic vs. intrinsic
- Extrinsic: supervised, i.e., the ground truth is available
  - Compare a clustering against the ground truth using certain clustering quality measure
  - Ex. BCubed precision and recall metrics
- Intrinsic: unsupervised, i.e., the ground truth is unavailable
  - Evaluate the goodness of a clustering by considering how well the clusters are separated, and how compact the clusters are
  - Ex. Silhouette coefficient

Measuring Clustering Quality: Extrinsic Methods

- Clustering quality measure: $Q(C, C_g)$, for a clustering C given the ground truth C_g.
- Q is good if it satisfies the following **4** essential criteria
  - Cluster homogeneity: the purer, the better
  - Cluster completeness: should assign objects belong to the same category in the ground truth to the same cluster
  - Rag bag: putting a heterogeneous object into a pure cluster should be penalized more than putting it into a *rag bag* (i.e., "miscellaneous" or "other" category)
  - Small cluster preservation: splitting a small category into pieces is more harmful than splitting a large category into pieces

Summary

- Cluster analysis groups objects based on their similarity and has wide applications
- Measure of similarity can be computed for various types of data
- Clustering algorithms can be categorized into partitioning methods, hierarchical methods, density-based methods, grid-based methods, and model-based methods
- K-means and K-medoids algorithms are popular partitioning-based clustering algorithms
- Birch and Chameleon are interesting hierarchical clustering algorithms, and there are also probabilistic hierarchical clustering algorithms
- DBSCAN, OPTICS, and DENCLU are interesting density-based algorithms
- STING and CLIQUE are grid-based methods, where CLIQUE is also a subspace clustering algorithm
- Quality of clustering results can be evaluated in various ways

Unit 6: Information Privacy and Data Mining

- 6.1 Basic principles to Protect Information Privacy
- 6.2 Uses and Misuses of Data Mining
- 6.3 Primary Aims of data Mining
- 6.4 Pitfalls of Data Mining

6.1 Basic principles to Protect Information Privacy

The **General Data Protection Regulation (GDPR)** comes into effect in May 2018 and replaces all EU directive (95/46/EC). The new regulation strengthens local European legislation for Data Protection and aligns regulators under one authority.

We can split GDPR into six privacy principles:

1. **Lawfulness, fairness and transparency:**

Transparency: Tell the subject what data processing will be done.
Fair: What is processed must match up with how it has been described
Lawful: Processing must meet the tests described in GDPR

2. **Purpose limitations**
3. **Data minimisation**
4. **Accuracy**
5. **Storage limitations**
6. **Integrity and confidentiality**

1. Lawfulness, fairness and transparency:

2. Purpose limitations

Personal data can only be obtained for “specified, explicit and legitimate purposes”.

Data can only be used for a specific processing purpose that the subject has been made aware of and no other, without further consent.

3. Data minimization

Data collected on a subject should be “adequate, relevant and limited to what is necessary in relation to the purposes for which they are processed”.
i.e. No more than the minimum amount of data should be kept for specific processing.

4. Accuracy

Data must be “accurate and where necessary kept up to date”
Baselining ensures good protection and protection against identity theft. Data holders should build rectification processes into data management / archiving activities for subject data.

5. Storage limitations

Regulator expects personal data is “kept in a form which permits identification of data subjects for no longer than necessary”.
i.e. Data no longer required should be removed.

6. Integrity and confidentiality

Requires processors to handle data “in a manner [ensuring] appropriate security of the personal data including protection against unlawful processing or accidental loss, destruction or damage”.

6.2 Uses and Misuses of Data Mining

Uses of data mining

1. It is helpful to predict future trends:

Most of the working nature of the data mining systems carries on all the informational factors of the elements and their structure.

One of the common benefits that can be derived with these data mining systems is that they can be helpful while predicting future trends. And that is quite possible with the help of technology and behavioral changes adopted by the people.

2. It signifies customer habits:

For example, while working in the marketing industry one can understand all the matters of customer behaviour and their habits. And that is possible with the help of data mining systems.

As these data mining systems handle all the information acquiring techniques. It is helpful in keeping the track of customer habits and their behavior.

3. Helps in decision making:

There are some people who make use of these data mining techniques to help them with some kind of decision making.

Nowadays, all the information about anything can be determined easily with the help of technology and similarly, with the help of such technology one can make a precise decision about something unknown and unexpected.

Uses (Continued)

4. Increase company revenue:

As it has been explained earlier that data mining is a process wherein which it involves some sort of technology to acquire

some information about anything possible. And this type of technology makes things easier for their profit earning ratio.

As people can collect information about the marketed products online, which eventually reduces the cost of the product and their services.

5. It depends upon market-based analysis:

Data mining process is a system where in which all the information has been gathered on the basis of market information.

Nowadays, technology plays a crucial role in everything and that casualty can be seen in these data mining systems. Therefore, all the information collected through these data mining is basically from marketing analysis.

6. Quick fraud detection:

Most parts of the data mining process is basically from information gathered with the help of marketing analysis. With the help of such marketing analysis, one can also find out those fraudulent acts and products available in the market.

6.2 Uses and Misuses of Data Mining

Misuse/Disadvantages of data mining

- **Privacy Issues**
  - Because of privacy issues, people are afraid of their personal information is collected and used in an unethical way that potentially causing them a lot of troubles.
- **Security issues**
  - **T**here have been a lot of cases that hackers accessed and stole big data of customers from the big corporation such as Ford Motor Credit Company, Sony… with so much personal and financial information available, the credit card stolen and identity theft become a big problem.
- **Misuse of information/inaccurate information**
  - Information is collected through data mining intended for the ethical purposes can be misused. This information may be exploited by unethical people or businesses to take benefits of vulnerable people or discriminate against a group of people.
  - In addition, data mining technique is not perfectly accurate. Therefore, if inaccurate information is used for decision-making, it will cause serious consequence.

6.3 Primary Aims of data Mining

- The main purpose of data mining process is to discover those records of information and summarize it in a simpler format for the purpose of others.
- **Goals of Data Mining**

1. Prediction: Determine how certain attributes will behave in the future. For example, how much sales volume a store will generate in a given period.

2. Identification: Identify patterns in data. For example, newly wed couples tend to spend more money buying furnitures.

3. Classification: Partition data into classes. For example, customers can be classified into different categories with different behavior in shopping.

4. Optimization: Optimize the use of limited resources such as time, space, money or materials. For example, how to best use advertising to maximize profits (sales).

6.4 Pitfalls of Data Mining

1. Overfitting and cross-validated data

2. Fallacy of incomplete evidence (The art of "cherry picking"):

Cherry picking is selecting results to fit a specific claim and excluding those that don't. This is a common practice often seen in public debates and politics where two sides can both present data that back their position.

3. Perverse Incentive (The cobra effect):

It is the act of setting an incentive that accidentally produces the opposite result to the one intended. It is said that the term comes from an infamous program created by the British government to reduce cobras population in India. The British announced that they would pay anyone who brings them a dead cobra. Indians took up cobra breedering to profit from the program. The government discovered these farms and ended the buyback program. This caused the closure of the breeders, resulting in the liberation of all the cobras. In the end, this translated to a huge increase in the total population of cobras, the opposite of what the British government wanted.

Eg: Recommender system recommends based on history purchase but not works while trying to buy new type of product

4. The gambler's fallacy and the regression fallacy:

The gambler's fallacy is the act of mistakenly believing that, because something recently has happened more frequently, it is now less likely to happen (and vice-versa).

This assumes that if something happens that's unusually good or bad, it will revert to average over time. This fallacy is often used to "find" a causal explanation for outliers generated by a process in which natural fluctuations exists.

5. Quantitative fallacy (The McNamara fallacy):

The fallacy involves making a decision based solely on quantitative observations or metrics, and ignoring all others. The reason given is often that these other observations cannot be proven.

Chapter- 7 Advanced Application

A. Web Mining

Web mining is the application of data mining techniques to extract knowledge from Web data, i.e. Web Content, Web Structure and Web Usage data.

Web Mining Taxonomy

Web Mining can be broadly divided into three distinct categories, according to the kinds of data to be mined.

a. Web Content Mining:

- Web Content Mining is the process of extracting useful information from the contents of Web documents.
- Content data corresponds to the collection of facts a Web page was designed to convey to the users.
- May consist of text, images, audio, video, or structured records such as lists and tables.
- Web content has been the most widely researched. Issues addressed in text mining are, topic discovery, extracting association patterns, clustering of web documents and classification of Web Pages.

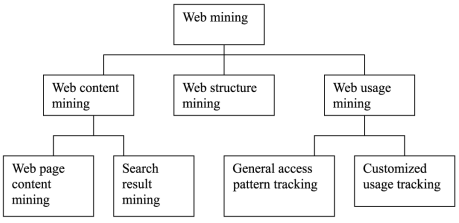
b. Web Structure Mining:

- The structure of a typical Web graph consists of Web pages as nodes, and hyperlinks as edges connecting related pages.
- Web Structure Mining is the process of discovering structure information from the Web. This can be further divided into two kinds based on the kind of structure information used.
    - Hyperlinks: A Hyperlink is a structural unit that connects a location in a Web page to different location, either within the same Web page or on a different Web page. A hyperlink that connects to a different part of the same page is called an Intra-Document Hyperlink, and a hyperlink that connects two different pages is called an Inter-Document Hyperlink.
    - Document Structure: In addition, the content within a Web page can also be organized in a tree-structured format, based on the various HTML and XML tags within the page. Mining efforts here have focused on automatically extracting document object model structures out of documents.

c. Web Usage Mining:

- Web Usage Mining is the application of data mining techniques to discover interesting usage patterns from Web data, in order to understand and better serve the needs of Web-based applications.

- Usage data captures the identity or origin of Web users along with their browsing behavior at a Web site.
- Web usage mining itself can be classified further depending on the kind of usage data considered:
    - Web Server Data: The user logs are collected by Web server. Typical data includes IP address, page reference and access time.
    - Application Server Data: Commercial application servers such as Web logic Story Server have significant features to enable E-commerce applications to be built on top of them with little effort. A key feature is the ability to track various kinds of business events and log them in application server logs.
    - Application Level Data: New kinds of events can be defined in an application, and logging can be turned on for them - generating histories of these specially defined events. It must be noted however that many end applications require a combination of one or more of the techniques applied in the above the categories.



Challenges:

i. Too huge for effective data warehousing and data mining.
ii. Too complex and heterogeneous.
iii. Growing and changing rapidly
iv. Broad diversity of user communities.
v. Only small portion of the information on the web is truly relevant or useful.

The Page Rank Algorithm

The original Page Rank algorithm was described by Lawrence Page and Sergey Brin in several publications. It is given by

$$PR(A) = (1-d) + d\,(PR(T1)/C(T1) + \ldots + PR(Tn)/C(Tn))$$

where
PR(A) is the Page Rank of page A,
PR(Ti) is the Page Rank of pages Ti which link to page A,
C(Ti) is the number of outbound links on page Ti and
d is a damping factor which can be set between 0 and 1.

- Page Rank does not rank web sites as a whole, but is determined for each page individually. Further, the Page Rank of page A is recursively defined by the Page Ranks of those pages which link to page A.
- The Page Rank of pages Ti which link to page A does not influence the PageRank of page A uniformly. Within the Page Rank algorithm, the Page Rank of a page T is always weighted by the number of outbound links C(T) on page T. This means that the more outbound links a page T has, the less will page A benefit from a link to it on page T.
- The weighted Page Rank of pages Ti is then added up. The outcome of this is that an additional inbound link for page A will always increase page A's Page Rank.
- Finally, the sum of the weighted Page Ranks of all pages Ti is multiplied with a damping factor d which can be set between 0 and 1. Thereby, the extend of PageRank benefit for a page by another page linking to it is reduced.

A Different Notation of the PageRank Algorithm

Lawrence Page and Sergey Brin have published two different versions of their Page Rank algorithm in different papers. In the second version of the algorithm, the Page Rank of page A is given as

PR(A) = (1-d) / N + d (PR(T1)/C(T1) + ... + PR(Tn)/C(Tn))

Where N is the total number of all pages on the web. The second version of the algorithm, indeed, does not differ fundamentally from the first one.

The Characteristics of Page Rank

The characteristics of Page Rank shall be illustrated by a small example.

We regard a small web consisting of three pages A, B and C, whereby page A links to the pages B and C, page B links to page C and page C links to page A. According to Page and Brin, the damping factor d is usually set to 0.85, but to keep the calculation simple we set it to 0.5. The exact value of the damping factor d admittedly has effects on Page Rank, but it does not influence the fundamental principles of Page Rank. So, we get the following equations for the Page Rank calculation:
PR(A) = 0.5 + 0.5 PR(C)

PR(B) = 0.5 + 0.5 (PR(A) / 2)
PR(C) = 0.5 + 0.5 (PR(A) / 2 + PR(B))

These equations can easily be solved. We get the following Page Rank values for the single pages:

PR(A) = 14/13 = 1.07692308
PR(B) = 10/13 = 0.76923077
PR(C) = 15/13 = 1.15384615

It is obvious that the sum of all pages' Page Ranks is 3 and thus equals the total number of web pages. As shown above this is not a specific result for our simple example. For our simple three-page example it is easy to solve the according equation system to determine Page Rank values. In practice, the web consists of billions of documents and it is not possible to find a solution by inspection.

The Iterative Computation of Page Rank

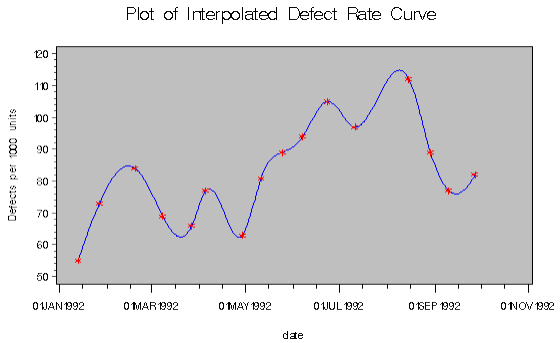
Because of the size of the actual web, the Google search engine uses an approximate, iterative computation of Page Rank values. Each page is assigned an initial starting value and the Page Ranks of all pages are then calculated in several computation circles based on the equations determined by the Page Rank algorithm. The iterative calculation shall again be illustrated by our three-page example, whereby each page is assigned a starting Page Rank value of 1.

Iteration	PR(A)	PR(B)	PR(C)
0	1	1	1
1	1	0.75	1.125
2	1.0625	0.765625	1.1484375
3	1.07421875	0.76855469	1.15283203
4	1.07641602	0.76910400	1.15365601
5	1.07682800	0.76920700	1.15381050
6	1.07690525	0.76922631	1.15383947
7	1.07691973	0.76922993	1.15384490
8	1.07692245	0.76923061	1.15384592
9	1.07692296	0.76923074	1.15384611
10	1.07692305	0.76923076	1.15384615
11	1.07692307	0.76923077	1.15384615
12	1.07692308	0.76923077	1.15384615

We see that we get a good approximation of the real Page Rank values after only a few iterations.

B. Time Series Data Mining

- Consists of sequences of values or events obtained over repeated measurement of time at equal time interval in most of the time.
- Used in application such as stock prediction, economic analysis etc.
- In general, there are two goals in time series analysis.
    i. Modeling Time Series: Generating the time series with underlying mechanism.
    ii. Forecasting Time Series: Predict the future values of the time series variables.



Major components for trend analysis in time series data

i. **Trend or Long term Movements**: Indicates the general direction in which a time series is moving over long or short interval of time through trend curve or trend line.
ii. **Cyclic Movement or Cyclic Variations**: Long term oscillations about a trend curve or line which may or may not be periodic.
iii. **Seasonal Movements or Variations**: These are systematic or calendar related. Eg. Sudden rise in sales of sweets in Tihar.
iv. **Irregular or Random Movements:** Series due to random or chance events. Eg. Price rise in crisis of supply.

Approaches for time series data analysis:

- Regression analysis is commonly used for find trend in time series data.
- Seasonal Index is used for analysis to adjust the reative values of a variable during the time series.
- Autocorrelation analysis is applied between i^{th} element of the series and the $(i-k)^{th}$ element to detect seasonal patterns. Where K is referred to as the log.
- Calculating the moving average of order n is the common method for determining trend.
  Eg:
  Original Data: 3 7 2 0 4 5 9 7 2
  Moving average of order3: (3 + 7 + 2)/3 = 4, 3 2 3 6 7 6
  Weighted (1, 4, 1) average: ((1*3 +4*7 +1*2)/(1+4 +1))= 5.5, 2.5 1 3.5 5.5 8 6.5
- Free hand method is used to draw approximate curve or line to fit a set of data based on user's judgment.
- Least square method is used to fit best curve.

C. Object/ Image/ Multimedia Mining:

- Multimedia database system stores and manages a large collection of multimedia data such as audio, video, images, graphics, speech, text etc.
- Image/multimedia mining deals with extraction of implicit knowledge, data relationship or other patterns not explicitly stored in images/multimedia
- The fundamental challenges in images mining is to determine the low-level pixel representation contained in an image or image sequence and cane be effectively and efficiently processed to identify high level spatial objects and relationships.
- Typical image/multimedia processing involves preprocessing, transformations and feature extraction mining, evaluation and interpretation of the knowledge.
- Different data mining techniques can be used such as association rules, clustering.

Unit 10 Capacity Planning

- 10.1 Calculating storage requirement, CPU requirements

- The data warehouse (& mining) environment - **like all** other computer environments - **requires hardware resources**.
- Given the **volume of data** and the **type of processing** that goes against the data, the data warehouse is **capable** of consuming large amounts of **resources**.
- For organizations that want to be in a **proactive stance** - where hardware resource utilization is not a surprise and the response time of a **system is anticipated ahead** of building the system, **capacity planning for the data warehouse environment is a very important exercise**.
- There are several aspects to the data warehouse environment that make capacity planning for the data warehouse a unique exercise:
  - The first factor is that the workload for the data warehouse environment is very variable.
  - A second factor making capacity planning for the data warehouse a risky business is that the data warehouse normally entails much more data than was ever encountered in the operational environment.
  - A third factor making capacity planning for the data warehouse environment a nontraditional exercise is that the data warehouse environment and the operational environments do not mix under the stress of a workload of any size at all.

Consider the patterns of hardware utilization as shown by Figure 1.

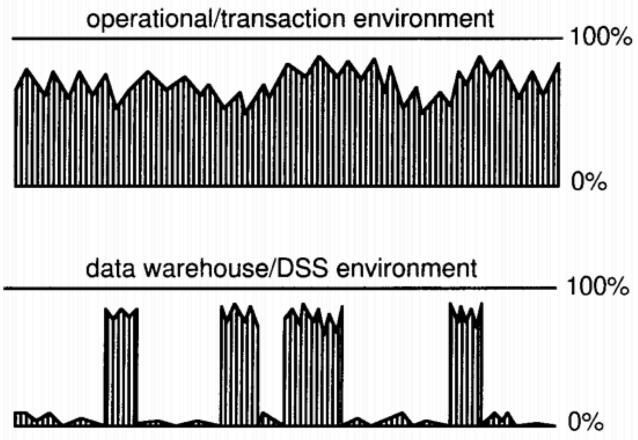


Figure 1: The fundamentally different patterns of hardware utilization between the data warehouse environment and the operational environment.

In the operational environment: uses hardware in a static fashion

In the data warehouse: hardware is used in a binary fashion

- Trying to mix the different patterns of hardware leads to some basic difficulties.

Figure 2 shows what happens when the two types of patterns of utilization are mixed in the same machine at the same time.

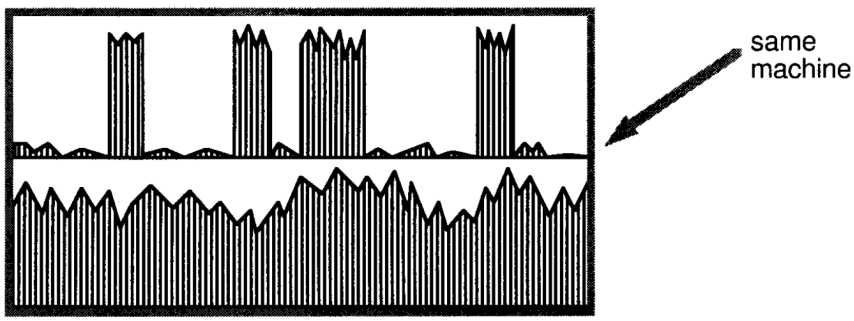


Figure 2: Trying to mix the two fundamentally different patterns of execution in the same machine at the same time leads to some very basic conflicts.

The patterns are simply incompatible.
Either you get **good response time** and a **low rate of machine utilization** (at which point the financial manager is unhappy), **or** you get **high machine utilization** and **poor response time** (at which point the user is unhappy.)

- The **need to split the two environments** is important to the data **warehouse capacity planner** because the capacity planner needs to be aware of circumstances in which the patterns of access are mixed.
- In other words, when doing capacity planning, **there is a need to separate the two environments**.

CALCULATING DISK STORAGE

- The **calculations for space** are almost always done **exclusively for the current detailed data** in the data warehouse.
- The reason why the other levels of data are not included in this analysis is that:
  - they consume much less storage than the current detailed level of data, and
  - they are much harder to identify. Therefore, the considerations of capacity planning for disk storage center around the current detailed level.

The calculations for disk storage are very straightforward. Figure 6 shows the elements of calculation.

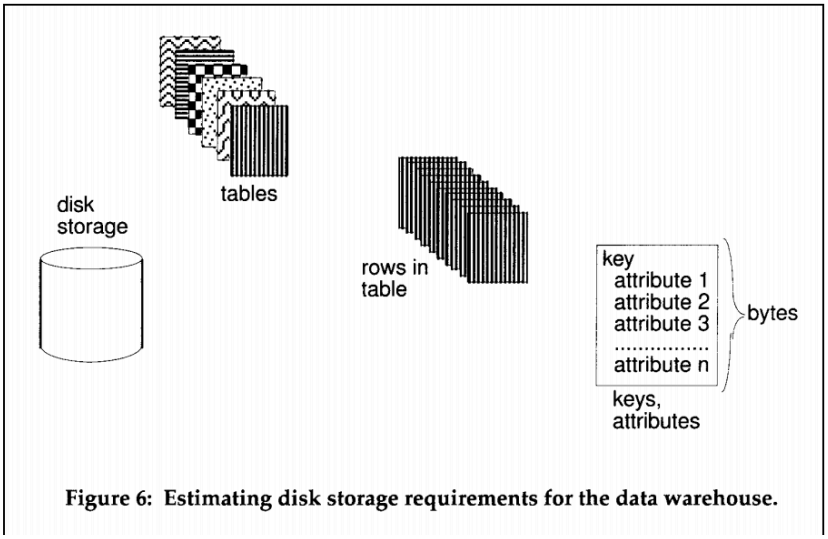


Figure 6: Estimating disk storage requirements for the data warehouse.

PROCESSOR REQUIREMENTS

- In order to make sense of the estimation of the processor requirements for the data warehouse, the work passing through the data warehouse processor must be divided into one of three categories - background processing, predictable DSS processing, and unpredictable DSS processing.

Figure 7 shows these three categories

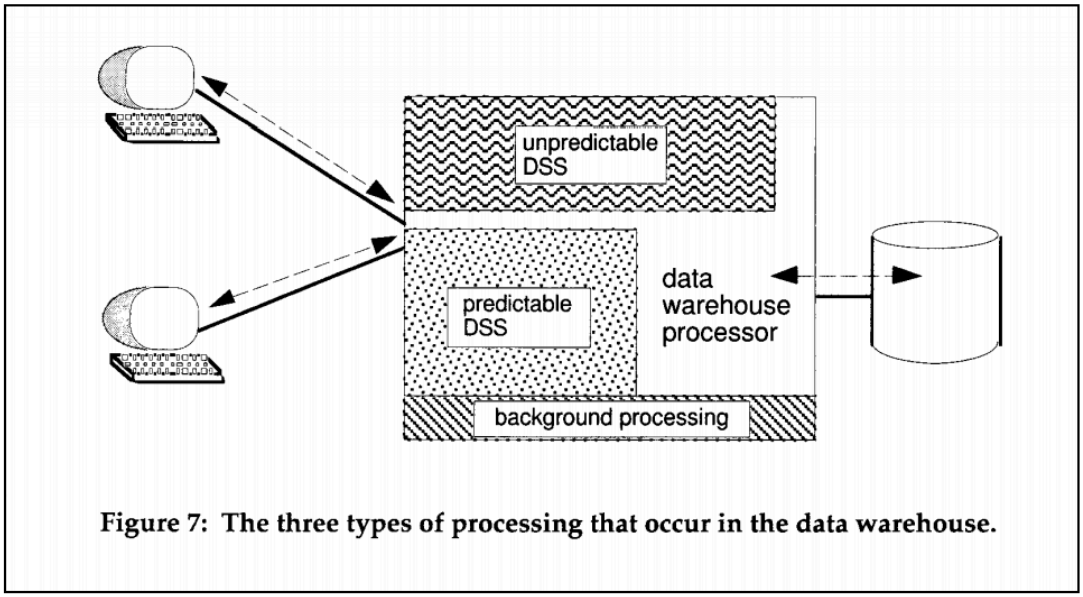


Figure 7: The three types of processing that occur in the data warehouse.

The parameters of interest for the data warehouse designer (for both the background processing and the predictable DSS processing) are:

- the number of times the process will be run,
- the number of I/Os the process will use,
- whether there is an arrival peak to the processing,
- the expected response time.

These metrics can be arrived at by examining the pattern of calls made to the dbms and the interaction with data managed under the dbms.

SUMMARY

- Capacity planning is important for the data warehouse environment as it was (and still is!) for the operational environment. Capacity planning in the data warehouse environment centers around planning disk storage and processing resources.
- There is an important but indirect relationship between data and processing power - the more data there is, the more the processing power required.
- Processing in the data warehouse environment must be physically separated from processing in the operational environment.
- Disk storage capacity is a function of the level of detail stored, the length of time the data is kept, and the number of occurrences of data to be stored.
- Processor capacity is a function of the workload passing through the environment. The important features of the processing environment are the characteristics of the workload, which can be described as consisting of background processing, predictable DSS processing, and unpredictable DSS processing. The profile of the transactions are merged together to produce a definitive picture of the resources needed for the data warehouse environment.