CS4.405 : Data Analytics-I

P. Krishna Reddy

IIIT Hyderabad
E-mail: pkreddy@iiit.ac.in
http://www.iiit.net/~pkreddy

Prof. P. Krishna Reddy
Data Science and Analytics Center, IIIT Hyderabad, India
E-mail: pkreddy@iiit.ac.in; http://www.iiit.ac.in/~pkreddy
Phone: 040-66531322; +91-9849329324

Research Areas

•Data mining/ Web Mining/ Bio-mining

Diverse frequent patterns, Coverage Patterns, Periodic frequent patterns, Rare knowledge extraction, Classification/prediction, Clustering.

Internet monetization, Search engine advertisement, Banner advertisement placemen, Recommendation systems, Link-based community extraction, E-commerce, Information extraction

Protein function prediction.

•Database Systems/ Systems Building

Schema summarization, Transaction synchronization, Database recovery, Data replication, Speculative transaction management, Memory efficient mining, User interfaces, Weather-based decision support systems.

•IT for Agriculture

Personalized agro-advice advisory, Knowledge transfer, Systems for bridging lab to land gap, Virtual crop labs for agriculture education, Location-specific content development, Agriculture extension, Agriculture planning tools for a farmer.

•IT for law:

Finding similar judgements

Systems Built

•eSagu: An IT-based Personalized Agro-Advisory System
•eAquaSagu: An IT-based Personalized Aqua-Advisory System
•eAgromet: An IT-based agro-meteorological advisory system
•Village-level eSagu: A Scalable and Location-Specific Agro-advisory System

Key Ideas Developed

•Coverage patterns, Diverse Patterns
•Dense-bipartite graph based web communities
•Rare frequent /knowledge patterns.
•Periodic frequent patterns
•Weather condition, Coupled weather condition

•Deadlock prevention using data flow graphs.
•Replica synchronization using data flow graphs
•Backup commit protocol
•Speculative locking protocol for regular and read-only transactions
•Temporality-based user interface design
•Query by object based user interface design

•Personalized agro-advisory (eSagu and eAgromet)
•Post-graduate Diploma in applied agriculture and IT (PGDAAIT)
•Virtual crop labs.

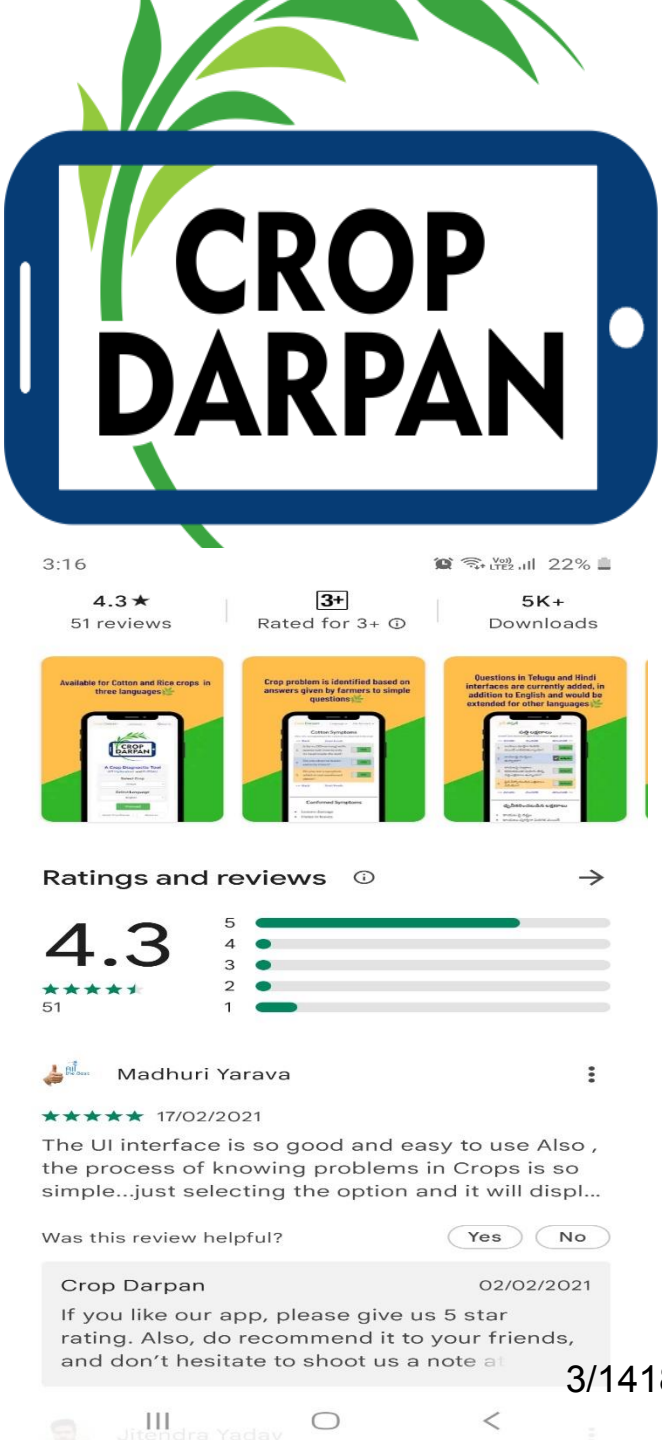
Future Research

•Diverse frequent patterns based recommendation and search systems.
•Coverage patterns based banner/search engine advertisement management
•Periodic frequent pattern based knowledge extraction system
•Efficient knowledge transfer systems
•Coverage pattern based subgraph discovery and visual mining
•Building query by object interface for non-SQL users
•Speculation-based synchronization for big data
•Protein function prediction
•Scalable decision support system for agriculture
•Building of resource planning tool for farmers
•Virtual crop labs for enhancing practical agricultural education
•Decision support system for law

Farm diagnostic tool

Install Crop Darpan App in your smart phone

- For desktop, the system is available at www.cropdarpan.com
- Crop Darpan App is available (on Andoid OS) at play store. Download and give feedback. https://play.google.com/store/apps/details?id=in.iiit.cropdarpan
- App is live on Apple App Store. https://apps.apple.com/us/app/crop-darpan/id1556486922#?platform=iphone



Introduction: Presentation Outline

- **Why Data Analytics (or data mining)?**
  - **Courses you have Completed**
  - Content of human mind
  - Sample data mining problems
- Data mining definition and KDD process
- Multidimensional view of data mining
- Data mining functionalities
- Course outline
- Summary and History of data mining society

Courses You have Completed

- Digital Logic Design
  - Able to carry out arithmetic and logic operations
- Operating systems
  - Able to store the file and retrieve the file
- Programming language
  - Able to write a program, given the requirement
- Algorithms, data structures
  - Efficient computation given the requirement: sorting and searching
- Database systems
  - Able to store data and retrieve data based on SQL

- Questions
  - What will you do with 10,000 or 100,000 rows result?
  - Can we get some new insights? If yes, how?

Introduction: Presentation Outline

- **Why Data Analytics (or data mining)?**
  - Courses you have Completed
  - **Content of human mind**
  - Sample data mining problems
- Data mining definition and KDD process
- Multidimensional view of data mining
- Data mining functionalities
- Course outline
- Summary and History of data mining society

Data, Information, Knowledge, and Wisdom by Gene Bellinger, Durval Castro, Anthony Mills

- According to Russell Ackoff, content of human mind can be classified into five categories: Data, Information, Knowledge, Understanding and wisdom

- Data: Symbols
  - **Data represents a fact or a statement of event without relation to other things.**
  - Data is raw. It simply exists and has no significance beyond its existence (in and of itself). It can exist in any form, usable or not. It does not have meaning of itself. In computer parlance, a spreadsheet generally starts out by holding data.
  - Ex: It is raining.

Content of Human Mind

- **Information:** Data that are processed to be useful; provides answer to "who", "what", "where", and "when" questions.
  - Information is data that has been given meaning by way of relational connection. This "meaning" can be useful, but does not have to be.
  - Information embodies **the understanding** of a relationship of some sort, possibly cause and effect.
  - Example The temperature dropped 15 degrees and then it started raining.
  - In computer parlance, a relational database makes information from the data stored within it.

Content of Human Mind

- Knowledge: application of data and information; answers “how” questions.
  - Knowledge represents a pattern that connects and generally providing a high level of predictability as what is described or what will happen next.
  - Knowledge is the appropriate collection of information, such that it's intent is to be useful. Knowledge is a deterministic process.
  - When someone "memorizes" information (as less-aspiring test-bound students often do), then they have amassed knowledge. This knowledge has useful meaning to them, but it does not provide for, in and of itself, an integration such as would infer further knowledge.

Content of Human Mind

- Knowledge: application of data and information; answers "how" questions.
  - …
  - For example, elementary school children memorize, or amass knowledge of, the "times table". They can tell you that "2 x 2 = 4" because they have amassed that knowledge (it being included in the times table). But when asked what is "1267 x 300", they can not respond correctly because that entry is not in their times table.
  - To correctly answer such a question requires a true cognitive and analytical ability that is only encompassed in the next level... understanding.
  - **Ex: If the humidity is very high and the temperature drops suddenly, the atmosphere is often unlikely to be able to hold the moisture, so it rains.**
  - In computer parlance, most of the applications we use (modeling, simulation, **data mining** etc.) some type of stored knowledge.

Content of Human Mind

- Understanding: appreciation of why
  - Understanding is an interpolative and probabilistic process. It is cognitive and analytical.
  - It is the process by which one can take knowledge and synthesize new knowledge from previously held knowledge.
  - The difference between understanding and knowledge is between "learning" and "memorizing".
  - People who have understanding can undertake useful actions because they can synthesize new knowledge, or in some cases, at least new information, from what is previously known (and understood).
  - **That is, understanding can build upon currently held information, knowledge and understanding itself.**
  - In computer parlance, AI systems possess understanding in the sense that they are able to synthesize new knowledge from previously stored information and knowledge.

Content of human mind

- **Wisdom: evaluated understanding**
- Wisdom embodies more of an understanding of fundamental principles embodied within the knowledge that are essentially the basis for the knowledge being what it is. Wisdom is essentially systemic.
  - Ex: It rains because it rains. And this encompasses an understanding of all the interactions that happen between raining, evaporation, air currents, temperature gradients, changes, raining.
  - Wisdom is an extrapolative and non-deterministic, non-probabilistic process.
  - It calls upon all the previous levels of consciousness, and specifically upon special types of human programming (moral, ethical codes, etc.).
  - **It beckons to give us understanding about which there has previously been no understanding, and in doing so, goes far beyond understanding itself.**

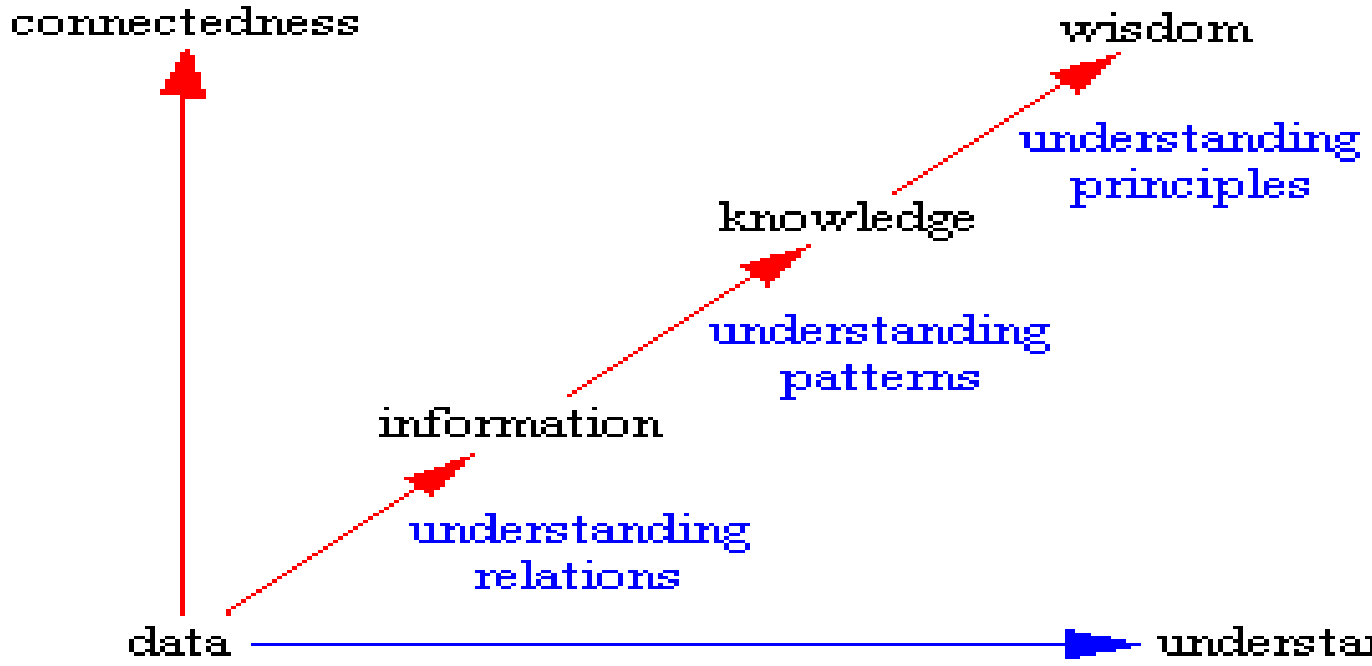
Content of human mind

- **Wisdom: evaluated understanding**
  - It is the essence of philosophical probing.
  - Unlike the previous four levels, it asks questions to which there is no (easily-achievable) answer, and in some cases, to which there can be no humanly-known answer period.
  - Wisdom is therefore, the process by which we also discern, or judge, between right and wrong, good and bad.

  - I personally believe that computers do not have, and will never have the ability to posses wisdom.
  - Wisdom is a uniquely human state, or as I see it, wisdom requires one to have a soul, for it resides as much in the heart as in the mind. And a soul is something machines will never possess (or perhaps I should reword that to ay, a soul is something that, in general, will never possess a machine).

Content of human mind

- Understanding that supports the transition from each stage to the next.
- Understanding is not a separate level of its own.
- Important: Ability to connect



Examples

- Example 1
  - Abugt dbesbt regtc uatn s uitrzt.
  - ubtxte pstye ysote anet sser extess
  - bxtedstes bet3 ibtes otesb tapbesct ehracts
- There is no foundation for you to connect with the pattern. If you know or **understand** the translation, these are Newton's 3 laws of motion

- Example 2
  - I have a box.
  - The box is 3' wide, 3' deep, and 6' high.
  - The box is very heavy.
  - The box has a door on the front of it.
  - When I open the box it has food in it.
  - It is colder inside the box than it is outside.
  - You usually find the box in the kitchen.
  - There is a smaller compartment inside the box with ice in it.
  - When you open the door the light comes on.
  - When you move this box you usually find lots of dirt underneath it.
  - Junk has a real habit of collecting on top of this box.
- At some point in the sequence, you connected **(understand)** with the pattern and understood it was a description of a refrigerator. From that point on each statement only added confirmation to **your understanding**.
- But, if you lived in a society that had never seen a refrigerator you might still be scratching your head as to what the sequence of statements referred to.

Introduction: Presentation Outline

- **Why Data Analytics (or data mining)?**
  - Courses you have Completed
  - Content of human mind
  - **Sample data mining problems**
- Data mining definition and KDD process
- Multidimensional view of data mining
- Data mining functionalities
- Course outline
- Summary and History of data mining society

Sample data mining problem # 1

I manage a supermarket (restaurant, video store, book store), and my cash register (or website) pumps transactions into my DB.

- Can you help me visualize my sales?
- Can you profile my customers?
- Tell me something interesting …
- I do not know statistics, and I do not want to hire statisticians.

Sample data mining problem #2

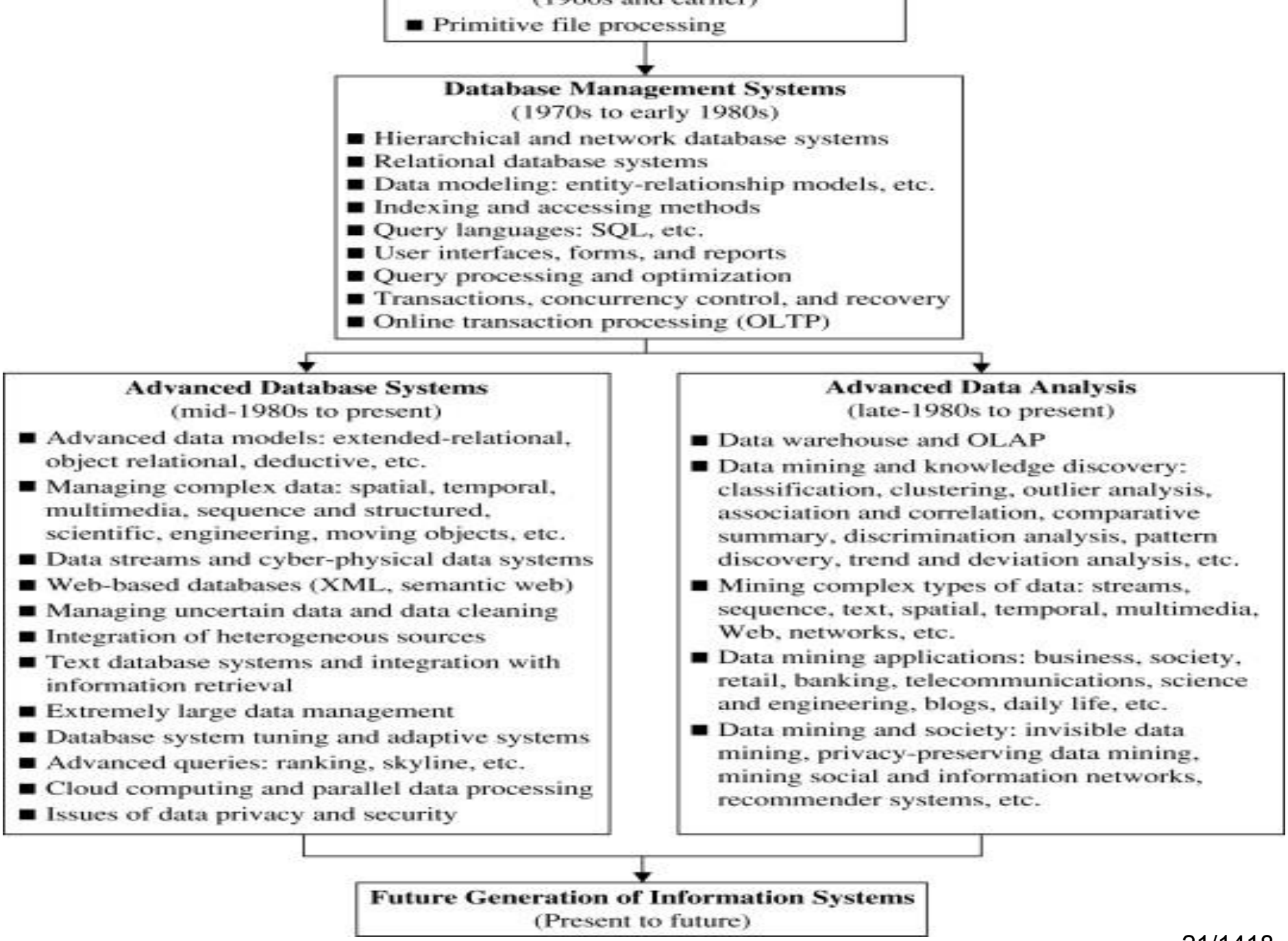
- I am an astronomer and I have sky survey 3 tera bytes of data, 2 billion objects.
  - Can you help to recognize the objects ?
  - Most of my data is beyond my reach.
  - Can you find new/unusual items in my data ?
  - Can you help me with basic manipulation, so I can focus on basic science ?
  - I know my data and statistics, but that is not enough …

Issue of Knowledge

- 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

Evolution of Sciences

- Before 1600, **empirical science**
- 1600-1950s, **theoretical science**
  - Each discipline has grown a *theoretical* component. Theoretical models often motivate experiments and generalize our understanding.
- 1950s-1990s, **computational science**
  - Over the last 50 years, most disciplines have grown a third, *computational* branch (e.g. empirical, theoretical, and computational ecology, or physics, or linguistics.)
  - Computational Science traditionally meant simulation. It grew out of our inability to find closed-form solutions for complex mathematical models.
- 1990-now, **data science**
  - The flood of data from new scientific instruments and simulations
  - The ability to economically store and manage petabytes of data online
  - The Internet and computing Grid that makes all these archives universally accessible
  - Scientific info. management, acquisition, organization, query, and visualization tasks scale almost linearly with data volumes. Data mining is a major new challenge!
- Jim Gray and Alex Szalay, *The World Wide Telescope: An Archetype for Online Science*, Comm. ACM, 45(11): 50-54, Nov. 2002


(1960s and earlier)
■ Primitive file processing
Database Management Systems
(1970s to early 1980s)
■ Hierarchical and network database systems
■ Relational database systems
■ Data modeling: entity-relationship models, etc.
■ Indexing and accessing methods
■ Query languages: SQL, etc.
■ User interfaces, forms, and reports
■ Query processing and optimization
■ Transactions, concurrency control, and recovery
■ Online transaction processing (OLTP)
Advanced Database Systems
(mid-1980s to present)
■ Advanced data models: extended-relational, object relational, deductive, etc.
■ Managing complex data: spatial, temporal, multimedia, sequence and structured, scientific, engineering, moving objects, etc.
■ Data streams and cyber-physical data systems
■ Web-based databases (XML, semantic web)
■ Managing uncertain data and data cleaning
■ Integration of heterogeneous sources
■ Text database systems and integration with information retrieval
■ Extremely large data management
■ Database system tuning and adaptive systems
■ Advanced queries: ranking, skyline, etc.
■ Cloud computing and parallel data processing
■ Issues of data privacy and security
Advanced Data Analysis
(late-1980s to present)
■ Data warehouse and OLAP
■ Data mining and knowledge discovery: classification, clustering, outlier analysis, association and correlation, comparative summary, discrimination analysis, pattern discovery, trend and deviation analysis, etc.
■ Mining complex types of data: streams, sequence, text, spatial, temporal, multimedia, Web, networks, etc.
■ Data mining applications: business, society, retail, banking, telecommunications, science and engineering, blogs, daily life, etc.
■ Data mining and society: invisible data mining, privacy-preserving data mining, mining social and information networks, recommender systems, etc.
Future Generation of Information Systems
(Present to future)

Introduction: Presentation Outline

- Why Data Analytics (or data mining)?
  - Courses you have Completed
  - Content of human mind
  - Sample data mining problems
- **Data mining definition and KDD process**
- Multidimensional view of data mining
- Data mining functionalities
- Course outline
- Summary and History of data mining society

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

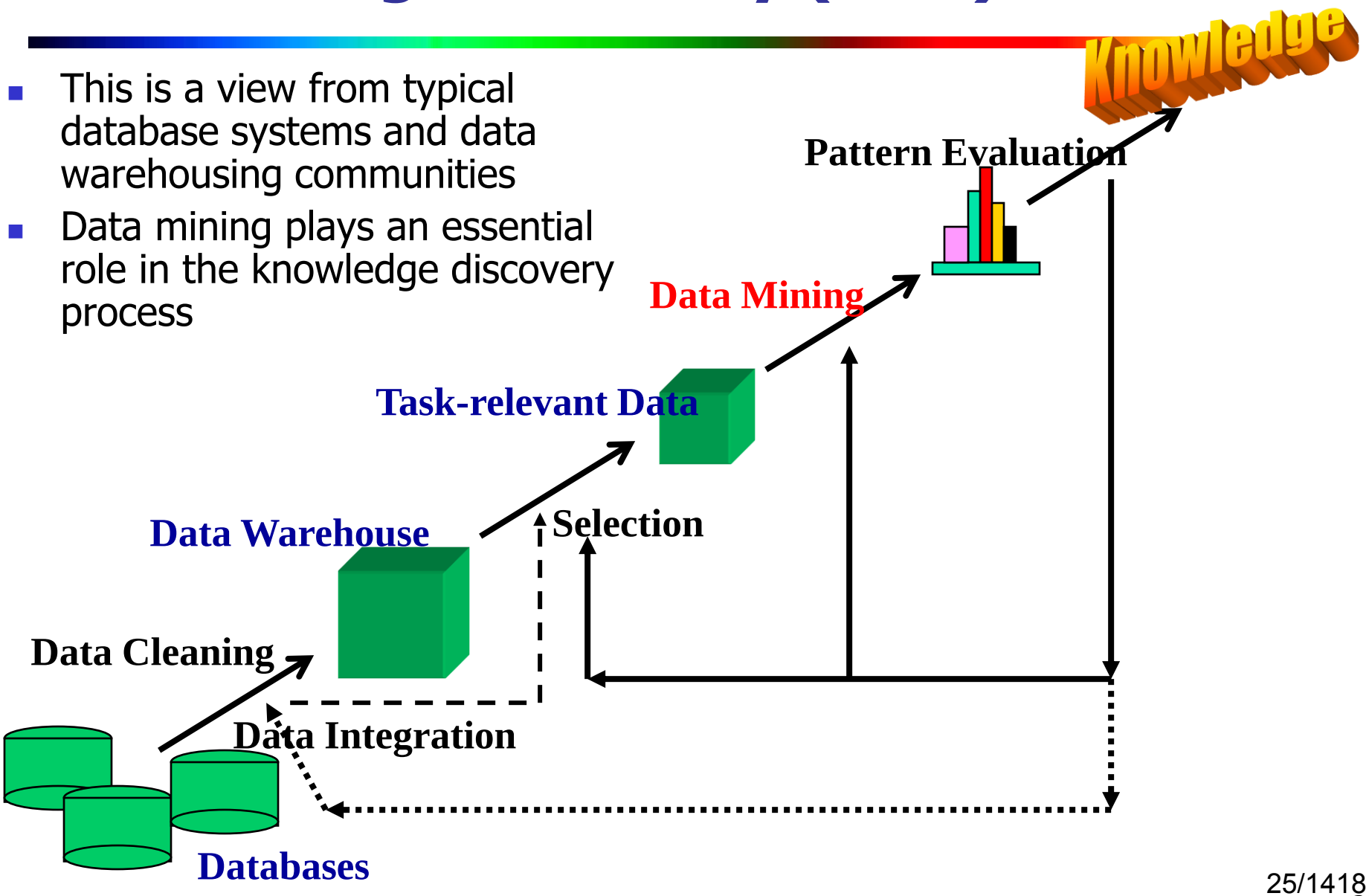
What is (not) Data Mining?

- **What is not Data Mining?**
  - Look up phone number in phone directory
  - Query a Web search engine for information about "Amazon"

- **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 (e.g. Amazon rainforest, Amazon.com,)

Knowledge Discovery (KDD) Process

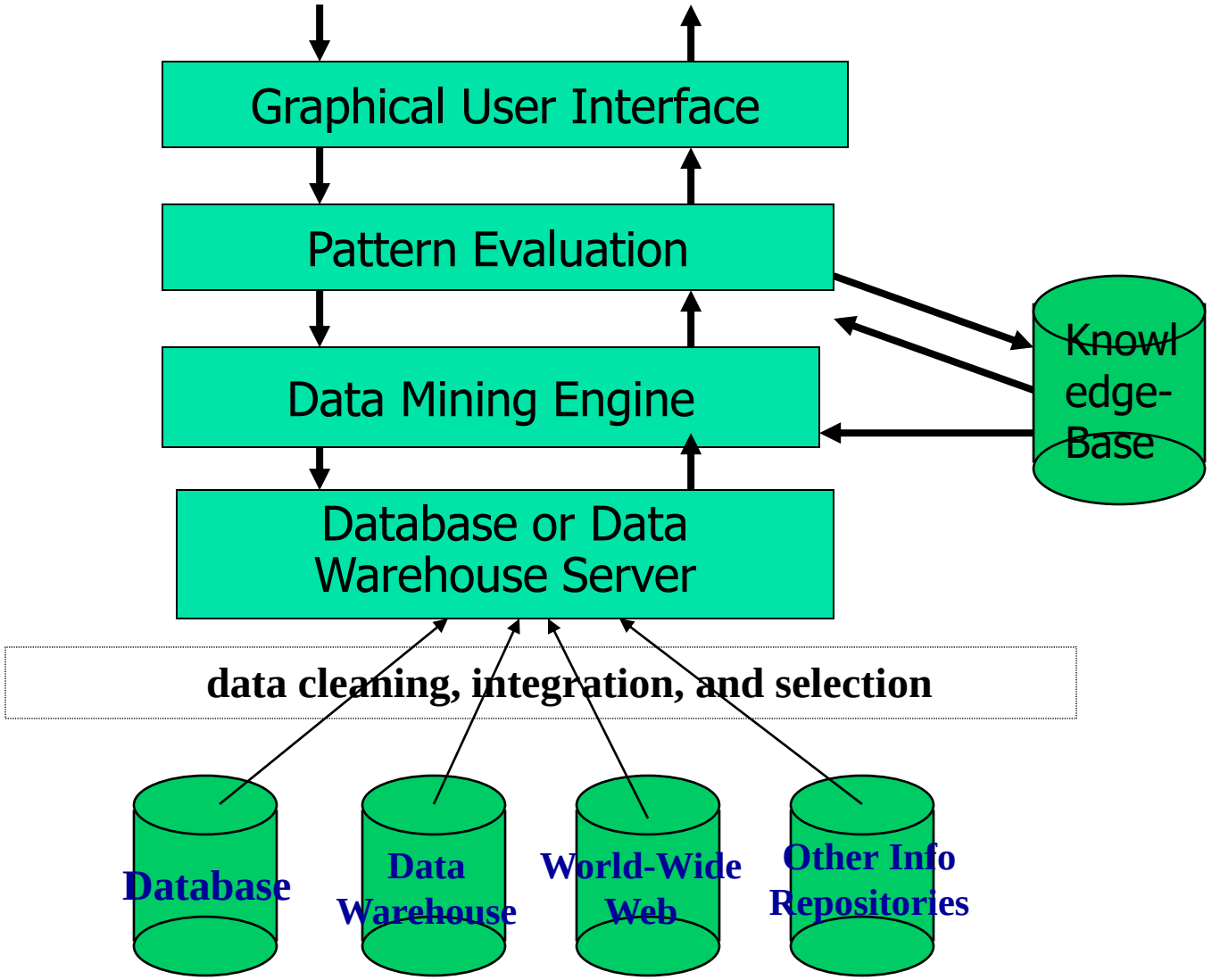
- This is a view from typical database systems and data warehousing communities
- Data mining plays an essential role in the knowledge discovery process



Steps of a KDD Process

- Learning the application domain:
  - relevant prior knowledge and goals of application
- Data cleaning: to remove noise and inconsistent data
- Data integration: Multiple data sources can be combined
- Creating a target data set: data selection
- Data cleaning and preprocessing: (may take 60% of effort!)
- Data reduction and transformation:
  - Find useful features, dimensionality/variable reduction, invariant representation.
- Choosing functions of data mining
  - summarization, association, classification, clustering.
- Choosing the mining algorithm(s)
- Data mining: search for patterns of interest
- Pattern evaluation and knowledge presentation
  - visualization, transformation, removing redundant patterns, etc.
- Use of discovered knowledge

Architecture: Typical Data Mining System



Components of data mining system

- **Database, Data warehouse, World Wide Web or other information Repository**
  - Data cleaning and data integration techniques are performed on this data
- **Database and data warehouse server:** Responsible for fetching the relevant data, based on the user's data mining request.
- **Knowledge-base:** Domain knowledge which is used to guide the data mining process.
  - Attribute levels, semantics, user beliefs, pattern interestingness, thrsholds, meta data
- **Data mining engine:** Set of functional modules for tasks such as characterization, summarization, association, classification, clustering, outlier extraction
- **Pattern evaluation:** Employees interestingness measures
  - Put the evaluation pattern as much deep as you can so that one can optimize.
- **User interface:** communication between users and the data mining system.

Data Mining in Business Intelligence

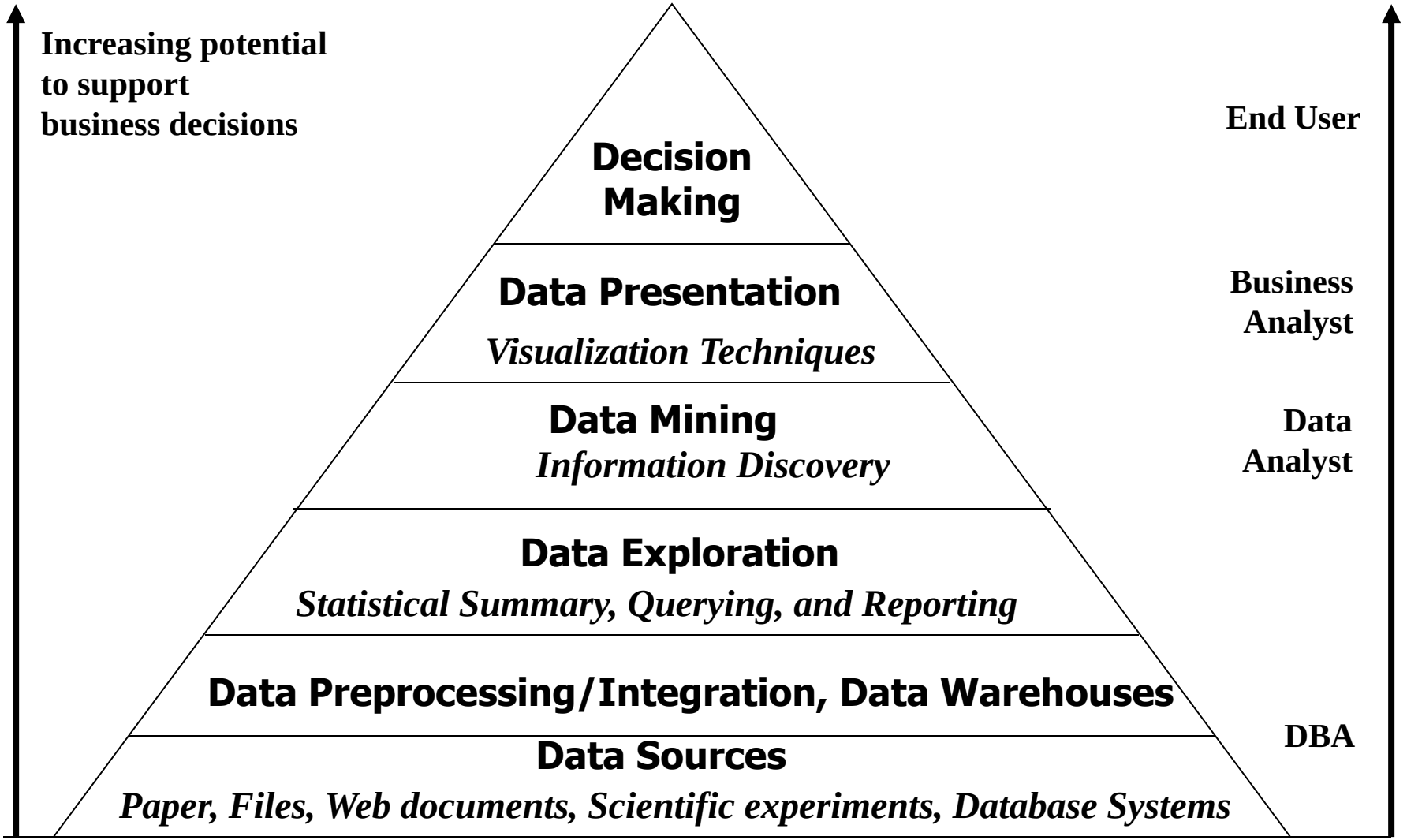
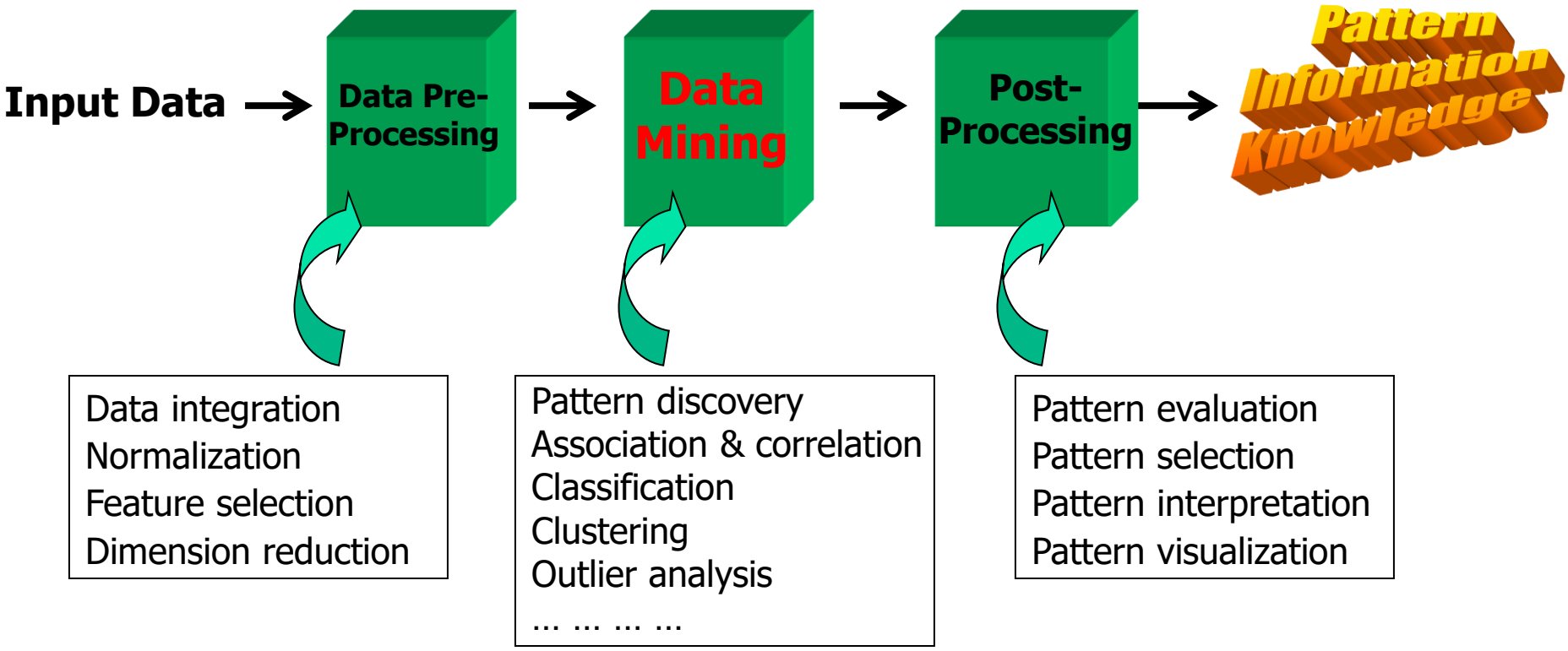




Fig. 3. Data-to-insight-to-decision whole-of-life analytics.

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

Introduction: Presentation Outline

- Why Data Analytics (or data mining)?
  - Courses you have Completed
  - Content of human mind
  - Sample data mining problems
- Data mining definition and KDD process
- **Multidimensional view of data mining**
- Data mining functionalities
- Course outline
- Summary and History of data mining society

Multi-Dimensional View of Data Mining

- **Data to be mined**
  - Database data (extended-relational, object-oriented, heterogeneous, legacy), data warehouse, transactional data, stream, spatiotemporal, time-series, sequence, text and web, multi-media, graphs & social and information networks
- **Knowledge to be mined (or: Data mining functions)**
  - Characterization, discrimination, association, classification, clustering, trend/deviation, outlier analysis, etc.
  - Descriptive vs. predictive data mining
  - Multiple/integrated functions and mining at multiple levels
- **Techniques utilized**
  - Data-intensive, data warehouse (OLAP), machine learning, statistics, pattern recognition, visualization, high-performance, etc.
- **Applications adapted**
  - Retail, telecommunication, banking, fraud analysis, bio-data mining, stock market analysis, text mining, Web mining, etc.

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

Introduction: Presentation Outline

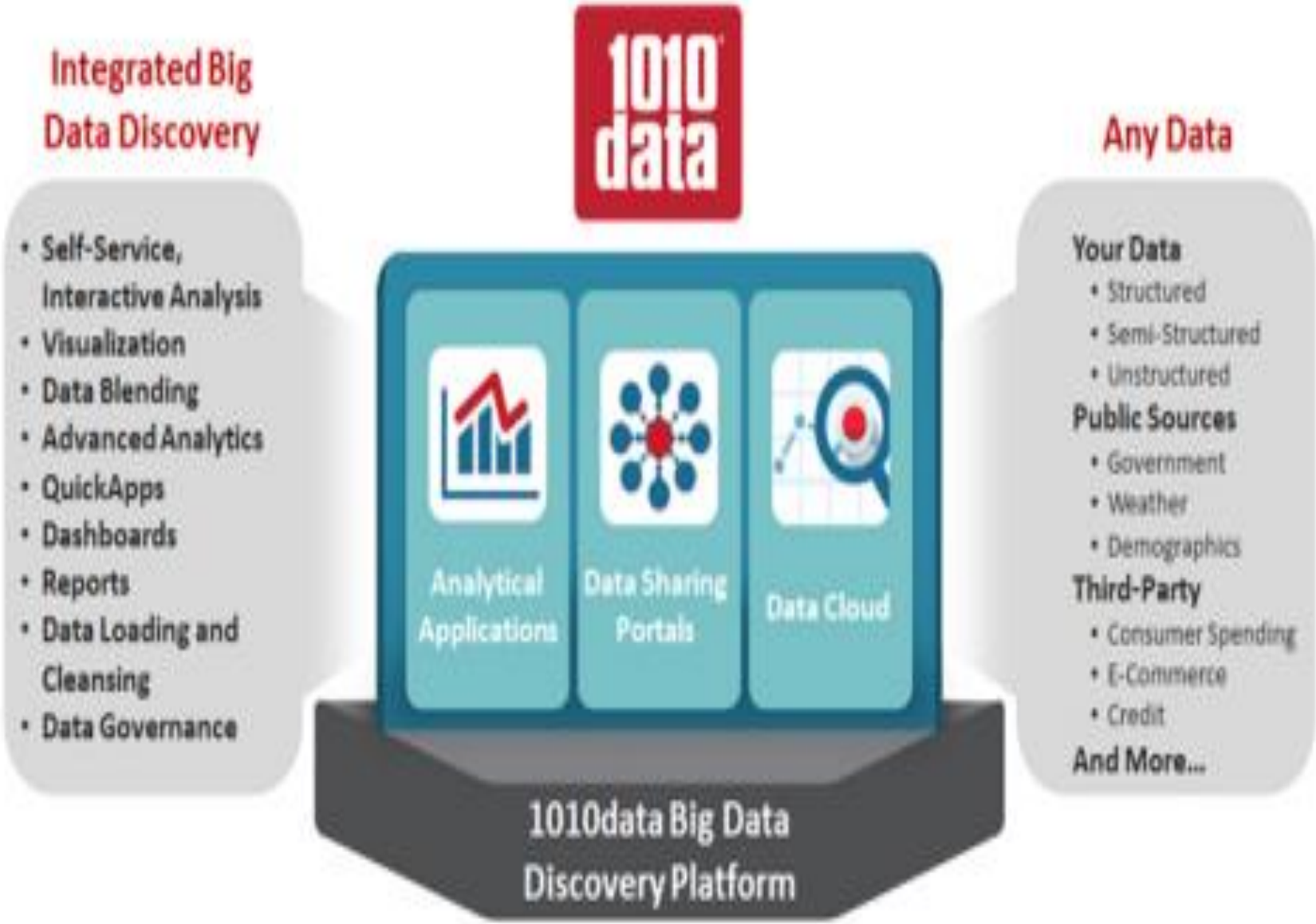
- Why Data Analytics (or data mining)?
  - Courses you have Completed
  - Content of human mind
  - Sample data mining problems
- Data mining definition and KDD process
- Multidimensional view of data mining
- **Data mining functionalities**
- Course outline
- Summary and History of data mining society

Data Mining Functionalities

- Generalization/Summarization: characterization and discrimination
- Pattern mining, Association mining, correlation
- Classification
- Clustering
- Outlier analysis
- Sequential, trend and evolution analysis
- Structure and network analysis

16 Top Big Data Analytics Platforms

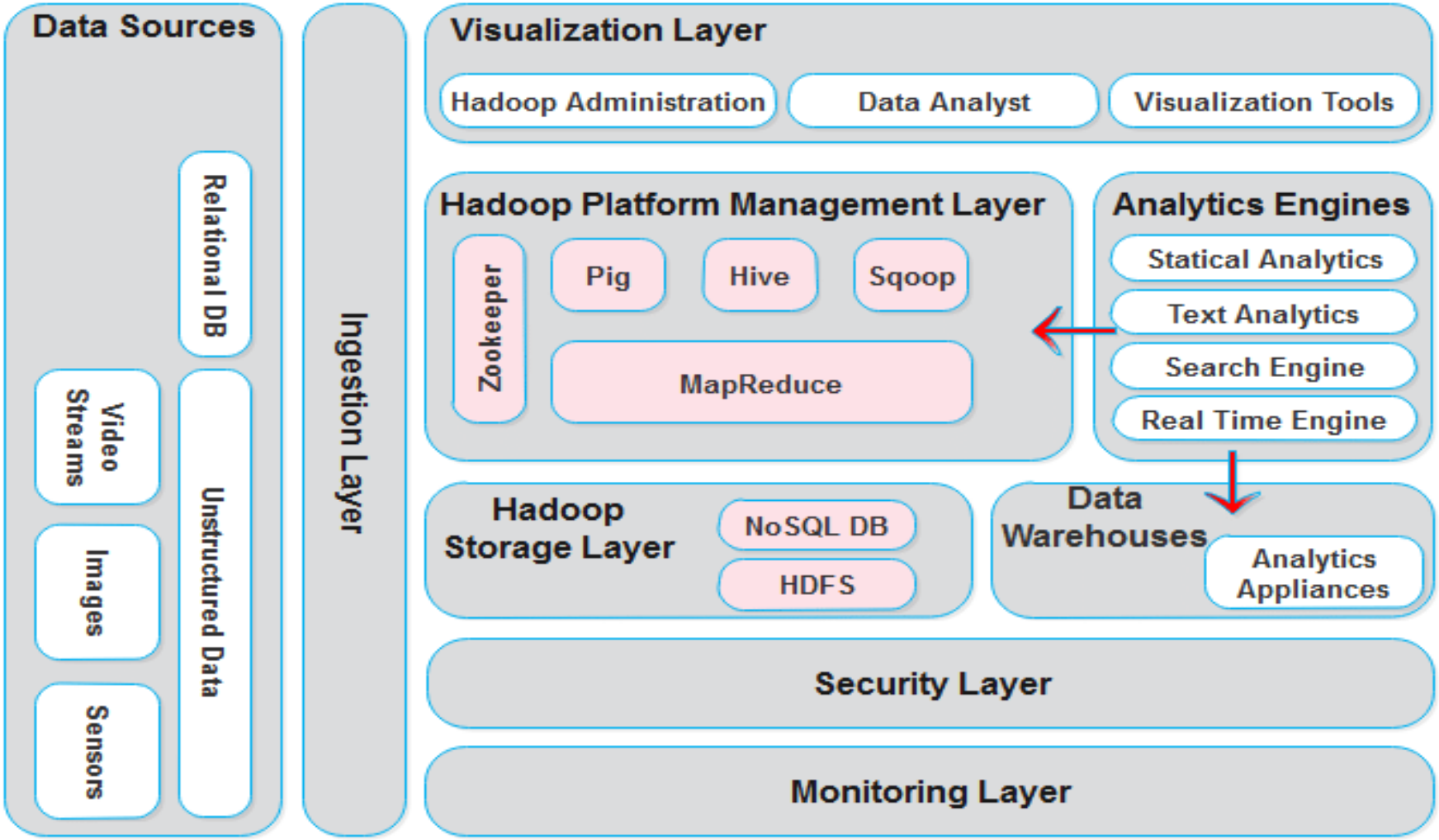



Integrated Big Data Discovery
• Self-Service, Interactive Analysis
• Visualization
• Data Blending
• Advanced Analytics
• QuickApps
• Dashboards
• Reports
• Data Loading and Cleansing
• Data Governance
1010 data
Analytical Applications
Data Sharing Portals
Data Cloud
1010data Big Data Discovery Platform
Any Data
Your Data
• Structured
• Semi-Structured
• Unstructured
Public Sources
• Government
• Weather
• Demographics
Third-Party
• Consumer Spending
• E-Commerce
• Credit
And More...

Top 10 Open Source Big Data Tools



Hadoop Big Data Architecture



Course Outline
(CS4.405: Data Analytics-I)

Prerequisite Course / Knowledge:

(i) Data and Applications, *or equivalent courses that cover Data modelling, normalization, SQL*

(ii) First courses on programming, data-structures and algorithms

(iii) Basics of Python language, to be able to use relevant libraries and toolkits for data analytics

Objective:

- In a computerized and networked society, vast amount of data is being collected every day in multiple domains.
- We are drowning in data, but starving for knowledge or actionable insights.
- Data mining or data analytics constitute a collection of concepts and algorithms, which are being developed to answer "how" questions by extracting interesting and useful knowledge of from large data.
- Data analytics based platforms are being operated in multiple domains to extract valuable and actionable insights from the data to improve the business performance.
- **The objective** of the course is to learn the important concepts and algorithms related to data analytics and data mining functionalities such as summarization, pattern mining, classification, clustering.
- **We will also briefly discuss the related research trends.**

Course Outcomes (COs)

- After completing the course successfully, the students are able to
  - CO-1. describe the concepts of data summarization, data warehousing, pattern mining, classification and clustering approaches
  - CO-2. perform the task of data summarization, pattern mining, classification and clustering based on the requirement.
  - CO-3. prescribe a single or a combination of data summarization, pattern mining, classification and clustering approaches for the problem scenario of a business/ organization.
  - CO-4. construct the improved data analytics methods for the existing services.
  - CO-5. formulate new data mining problems for creating new services and design the corresponding solutions

Detailed Syllabus

- Unit1:Introduction, data summarization through characterization, discrimination and data cube techniques (9 hours)
- Unit 2: Concepts and algorithms for mining patterns and associations (10 hours)
- Unit 3: Concepts and algorithms related to classification and regression (10 hours)
- Unit 4: Concepts and algorithms for clustering the data (10 hours)
- Unit 5: Outlier analysis and future trends. (3 hours)

Course Code: CS4.405: Data Analytics-I

Class Schedule (MONSOON 2024)

class	Date	Topic	Lab
1	1/8/2024 (THU)	Introduction to Data Mining	
2	5/8/2024 (MON)	Overview of data mining concepts, Knowing your data	Lab 1
3	8/8/2024 (THU)	Data preprocessing	
4	12/8/2024 (MON)	Attribute Oriented Induction	
5	16/8/2024 (FRI)	Data warehousing and OLAP technology	
6	19/8/2024 (MON)	DATA CUBE COMPUTATION ALGORITHMS 1. On the computation of multidimensional aggregates (Agarwal et al.'96) 2. Multiway Array Aggregation for Full Cube Computation (Zhao et al SIGMOD'97)	Lab 2
7	22/8/2024 (THU)	DATA CUBE COMPUTATION ALGORITHMS 3.BUC: Computing Iceberg Cubes from the Apex Cuboid Downward (Beyer et al SIGMOD'99) 4.H-Cubing: Exploring an H-Tree Structure (Han et al SIGMOD'01)	
8	26/8/2024 (MON)	Quiz I	
9	29/8/2024 (THU)	Overview of other data cube algorithms: Star Cubing, Shell Fragments, Sampling Cubes, Ranking Cubes, Prediction Cubes, Discovery driven Cubes	
10	02/9/2024 (MON)	Introduction to Association mining	Lab 3
11	05/9/2024 (THU)	Apriori Algorithm and extensions	
12	09/9/2024 (MON)	FP growth algorithm	
13	12/9/2024 (THU)	ECLAT algorithm and pattern Evaluation	
14	19/9/2024 (THU)	Advances in Pattern Mining: Overview of Multi-level mining, Quantitative association rules, Utility mining, Diverse Patterns, Coverage Patterns	
		Midterm Exam	
15	26/9/2024 (THU)	Basic concepts	Lab 4
16	30/9/2024 (MON)	Decision Tree Induction	
17	3/10/2024 (THU)	Bayes Classification Methods	
18	7/10/2024 (MON)	Rule-Based Classification	
19	10/10/2024 (THU)	Model Evaluation and Selection, Techniques to Improve Accuracy	
20	14/10/2024 (MON)	Overview of Other classification methods: Bayesian belief network (probabilistic networks), Support Vector Machine (SVM), Pattern-based classification, lazy learners (KNN, case-based reasoning), genetic algorithms, rough set and fuzzy set approaches, Multiclass classification, Semi-supervised classification, Active learning, Transfer learning	
21	17/10/2024 (THU)	Quiz 2	
22	21/10/2024 (MON)	Introduction to clustering	
23	24/10/2024 (THU)	Partitioning methods	Lab 5
24	28/10/2024 (MON)	Hierarchical methods	
25	07/11/2024 (THU)	Chameleon algorithm and evaluation of clustering results	
26	11/11/2024 (TUE)	Overview: Advanced cluster analysis and outlier mining	
27	14/11/2024 (FRI)	Overview of deep learning	
28	18/11/2024(MON)	Data mining trends	

Lab

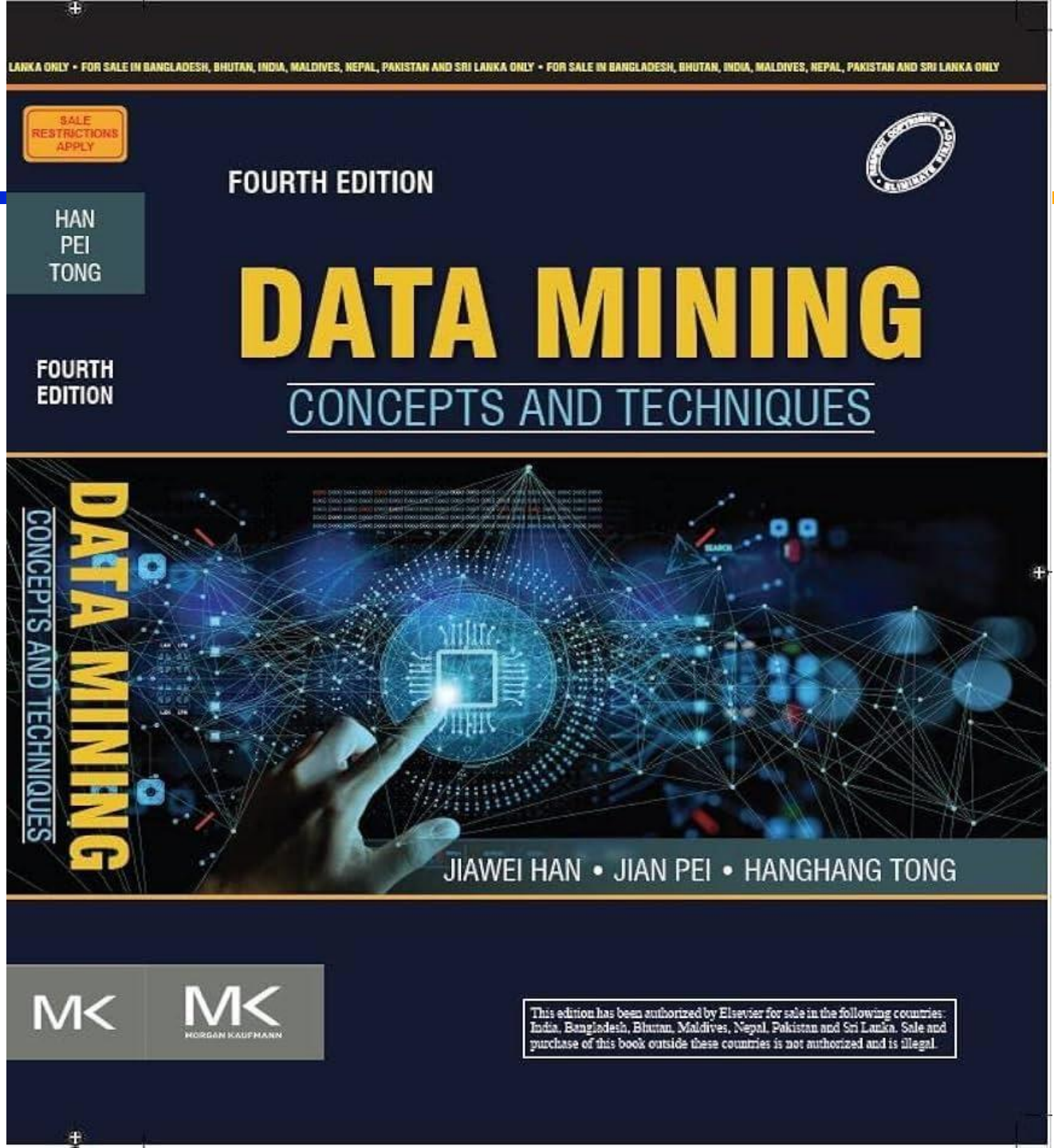
- Five mini projects related to the above syllabus will be done by students in the laboratory

Reference Books and materials:

1. **Book: Jiawei Han, Jian Pei, Hanghang Tong, Data Mining: Concepts and Techniques, Fourth edition, 2022, Elseiver Inc.**

2. Book: Pang-Nong Tan, Michael Steinbach and Vipin Kumar, Introduction to Data Mining, 2006, Pearson Education.

1. Research Papers: About 25 research papers from the proceeding of the conferences and journals related to data summarization, data warehousing, pattern mining, classification, clustering, outlier detection.


LANKA ONLY • FOR SALE IN BANGLADESH, BHUTAN, INDIA, MALDIVES, NEPAL, PAKISTAN AND SRI LANKA ONLY • FOR SALE IN BANGLADESH, BHUTAN, INDIA, MALDIVES, NEPAL, PAKISTAN AND SRI LANKA ONLY
SALE RESTRICTIONS APPLY
FOURTH EDITION
HAN
PEI
TONG
DATA MINING
CONCEPTS AND TECHNIQUES
FOURTH EDITION
DATA MINING
CONCEPTS AND TECHNIQUES
JIAWEI HAN • JIAN PEI • HANGHANG TONG
MK
MK
MORGAN KAUFMANN
This edition has been authorized by Elsevier for sale in the following countries: India, Bangladesh, Bhutan, Maldives, Nepal, Pakistan and Sri Lanka. Sale and purchase of this book outside these countries is not authorized and is illegal.

Assessment methods and weightages

- Two Classroom tests: 10 marks (5+5)
- Mid Semester Examination in theory: 20 marks,
- End Semester Examination in Theory: 40 marks,
- Assessment of five mini projects in the Laboratory: 30 marks
- **New Idea**
  - **Based on the course contents, if you produce a new idea of publishable quality by doing additional work, additional weightage will be given.**

About new idea

- You can give two or three new ideas as per the below format (follow this
format for each idea).

  ****

  1. Title of the problem:
  2. Problem statement (about 50 words)
  3. Background (< 200 words)
  4. What is the state of art and gaps (<200 words)
  5. Description of the proposed solution (<500 words)
  6. Explain how the proposed solution will be different over existing approaches (<200 words)
  7. Explain how you will conduct the experiments and the corresponding performance metrics (<200 words)
  8. References (<10 references)

  ****

Lab Assignments

- You will be given the lab assignments in advance
  - The instructor will provide you with adequate background information to do the assignment
  - Some installations will be necessary. Install well in advance and if you encounter any problems with any installation, you can contact your TA
- Policies for lab
  - You should try to solve any programming issues by yourself first, remember that troubleshooting is how you would actually be learning a lot
  - Every time you encounter any minor programming problem, if you ask the TA or instructor, you will lose out on an excellent learning experience
  - Only if you are unable to solve the problem after putting in a reasonable amount of effort, you should contact your TA or your instructor
  - You should also look online for solutions to your problems, but do not copy code from anywhere
- Grading criteria
  - The overall quality of your lab assignment submission in terms of correctness, quality of code, system design etc.

Plagiarism

- This course has a zero-tolerance policy w.r.t. plagiarism
- Any instance of plagiarism will result in serious penalties (e.g., an F grade for the entire course, among other penalties)
- Forget about doing any kind of plagiarism

Deadlines

- Strict deadlines: You will not be able to submit after the deadline has already passed.

Accessing the Course Materials

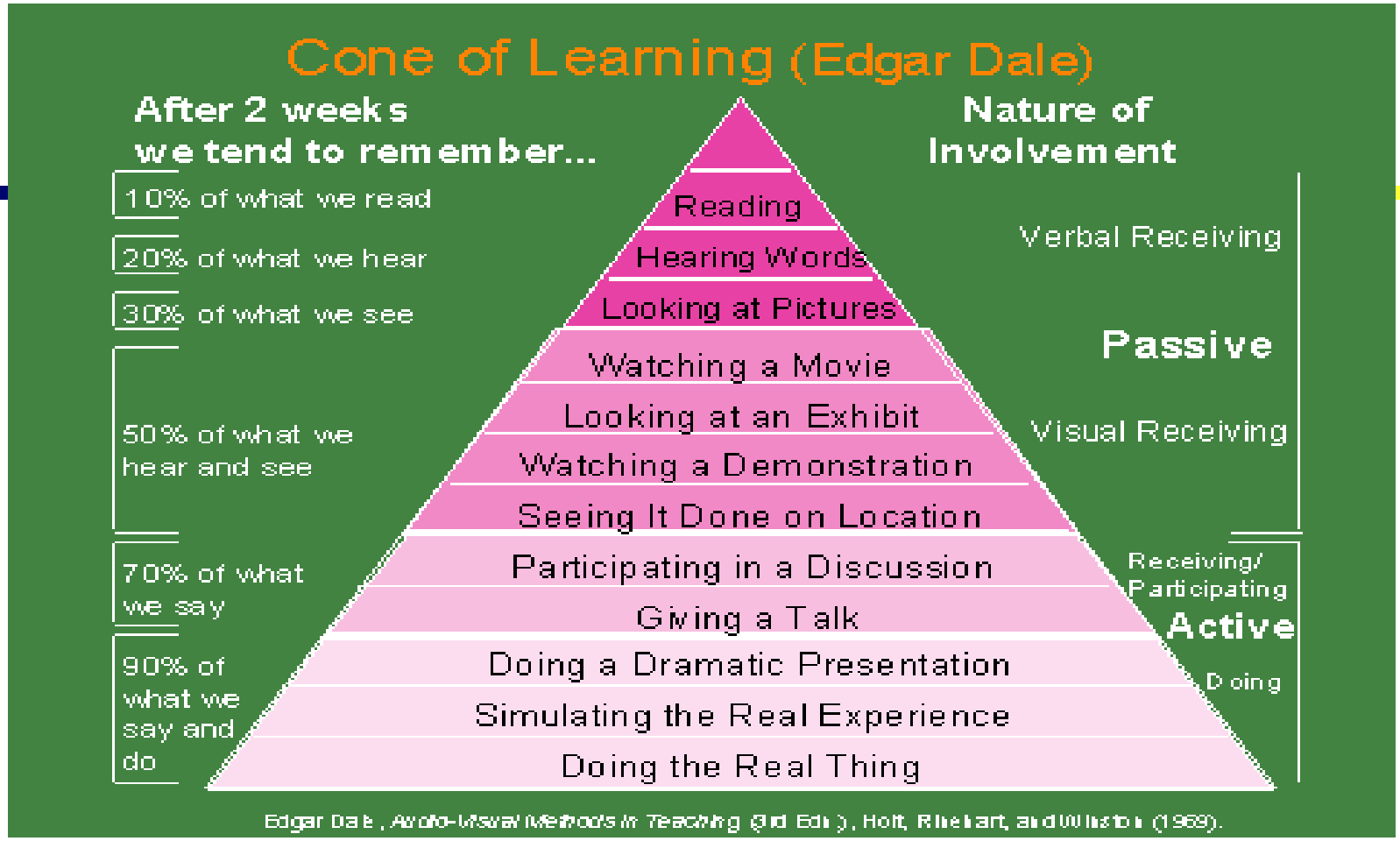
- The presentations, the materials, lab assignments are available on the course portal
  - Pls. access the course portal regularly
- All students should procure the book, as soon as possible

Asking Questions

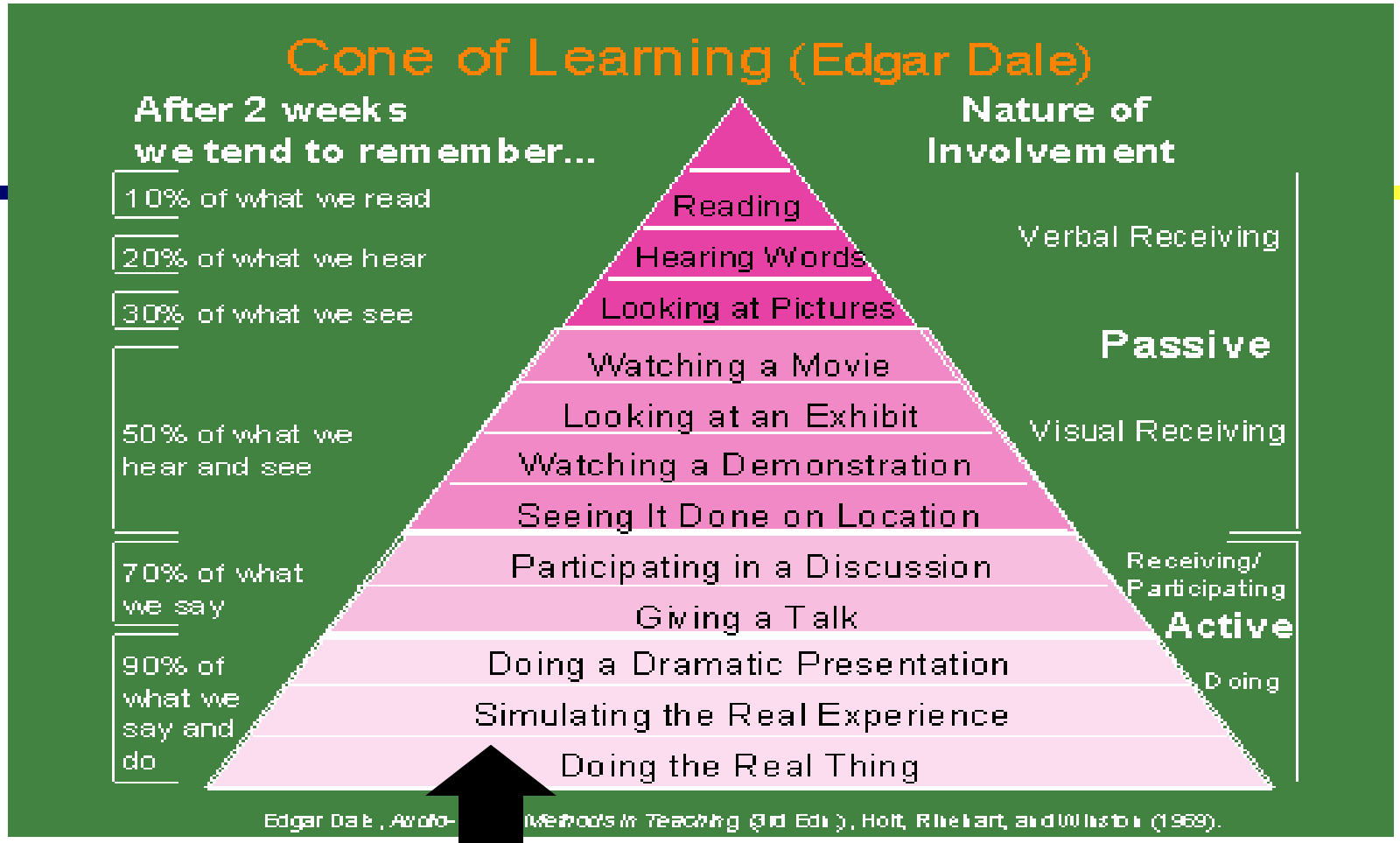
- Theory and lab related questions can be asked through the course portal
  - It is better, because all other students can gain the related knowledge
  - Try to ask as many subject/lab questions as possible.
  - Do not hesitate to ask silly/simple questions regarding lab or theory
    - And try to get the problem resolved as soon as possible

Maximize the benefit from the course

- In the course, we are going to study about an important technology
  - Data scientist or data analyst
- Focus on a thorough understanding of the concepts, not on memorizing
  - Try to understand every sentence of the book



Source: http://www.cals.ncsu.edu/agexed/sae/ppt1/sld012.htm



Bottomline: Do the assignments sincerely because it will facilitate you in **INTERNALIZING** the ideas/techniques you learnt in this course.

Source: http://www.cals.ncsu.edu/agexed/sae/ppt1/sld012.htm

Presentation Outline

- Why Data Analytics (or data mining)?
  - Courses you have Completed
  - Content of human mind
  - Sample data mining problems
- Data mining definition and KDD process
- Multidimensional view of data mining
- Overview of data mining functionalities
- Issues in data mining
- Course outline
- **Summary and History of data mining society**

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.**

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

Conferences and Journals on Data Mining

- KDD Conferences
  - ACM SIGKDD Int. Conf. on Knowledge Discovery in Databases and Data Mining (KDD)
  - SIAM Data Mining Conf. (SDM)
  - (IEEE) Int. Conf. on Data Mining (ICDM)
  - European Conf. on Machine Learning and Principles and practices of Knowledge Discovery and Data Mining (ECML-PKDD)
  - Pacific-Asia Conf. on Knowledge Discovery and Data Mining (PAKDD)
  - Int. Conf. on Web Search and Data Mining (WSDM)
- Other related conferences
  - DB conferences: ACM SIGMOD, VLDB, ICDE, EDBT, ICDT, …
  - Web and IR conferences: WWW, SIGIR, WSDM
  - ML conferences: ICML, NIPS
  - PR conferences: CVPR,
- Journals
  - Data Mining and Knowledge Discovery (DAMI or DMKD)
  - IEEE Trans. On Knowledge and Data Eng. (TKDE)
  - KDD Explorations
  - ACM Trans. on KDD

Where to Find References? DBLP, CiteSeer, Google

- Data mining and KDD (SIGKDD: CDROM)
  - Conferences: ACM-SIGKDD, IEEE-ICDM, SIAM-DM, PKDD, PAKDD, etc.
  - Journal: Data Mining and Knowledge Discovery, KDD Explorations, ACM TKDD
- Database systems (SIGMOD: ACM SIGMOD Anthology—CD ROM)
  - Conferences: ACM-SIGMOD, ACM-PODS, VLDB, IEEE-ICDE, EDBT, ICDT, DASFAA
  - Journals: IEEE-TKDE, ACM-TODS/TOIS, JIIS, J. ACM, VLDB J., Info. Sys., etc.
- AI & Machine Learning
  - Conferences: Machine learning (ML), AAAI, IJCAI, COLT (Learning Theory), CVPR, NIPS, etc.
  - Journals: Machine Learning, Artificial Intelligence, Knowledge and Information Systems, IEEE-PAMI, etc.
- Web and IR
  - Conferences: SIGIR, WWW, CIKM, etc.
  - Journals: WWW: Internet and Web Information Systems,
- Statistics
  - Conferences: Joint Stat. Meeting, etc.
  - Journals: Annals of statistics, etc.
- Visualization
  - Conference proceedings: CHI, ACM-SIGGraph, etc.
  - Journals: IEEE Trans. visualization and computer graphics, etc.

OTHER SLIDES

Data Mining Function: (1) Generalization

- 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, e.g., dry vs. wet region

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, …

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

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

Time and Ordering: Sequential Pattern, Trend and Evolution Analysis

- Sequence, trend and evolution analysis
  - Trend, time-series, and deviation analysis: e.g., regression and value prediction
  - Sequential pattern mining
    - e.g., first buy digital camera, then buy large SD memory cards
  - Periodicity analysis
  - Motifs and biological sequence analysis
    - Approximate and consecutive motifs
  - Similarity-based analysis
- Mining data streams
  - Ordered, time-varying, potentially infinite, data streams

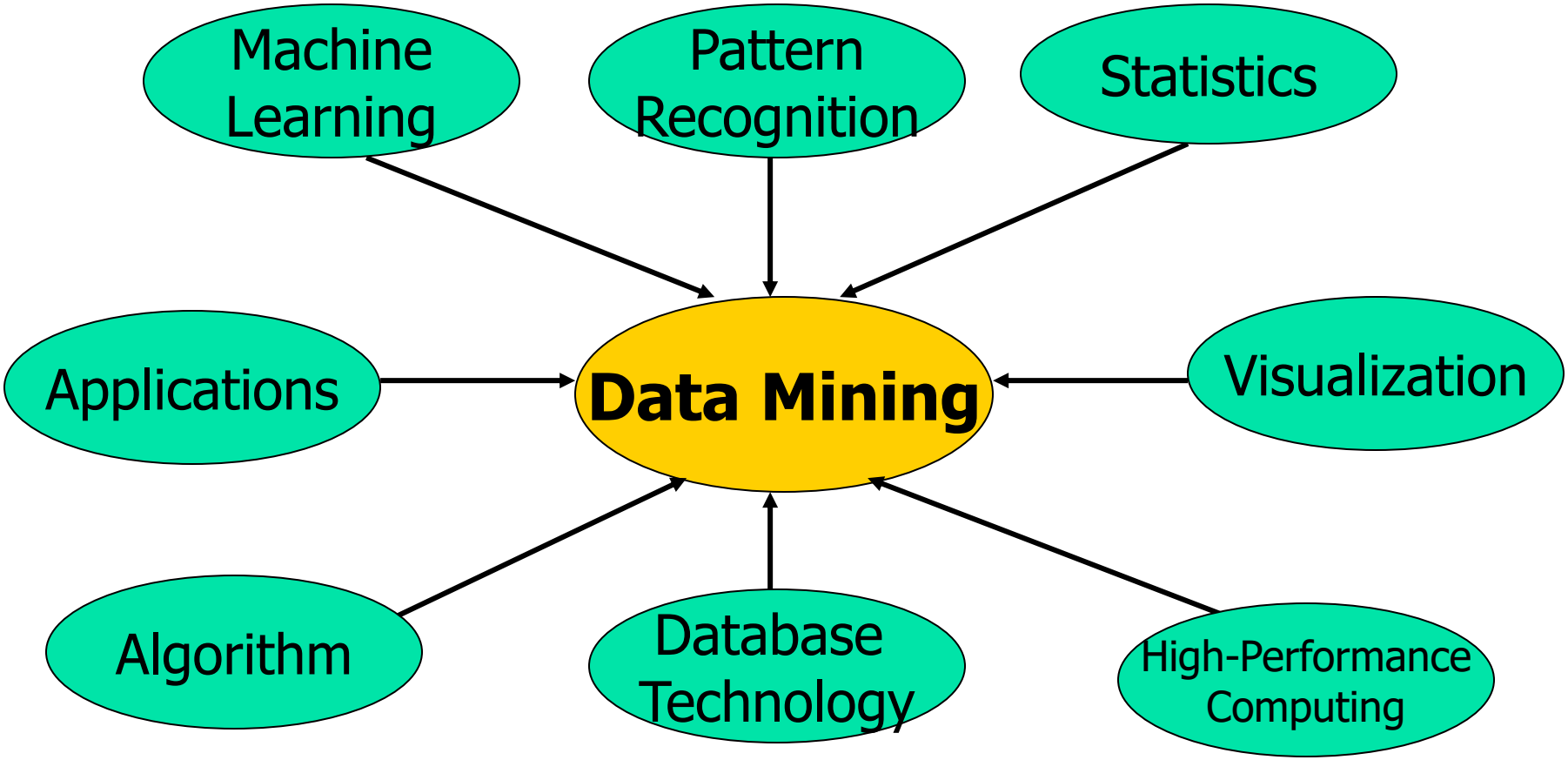
Structure and Network Analysis

- Graph mining
  - Finding frequent subgraphs (e.g., chemical compounds), trees (XML), substructures (web fragments)
- Information network analysis
  - Social networks: actors (objects, nodes) and relationships (edges)
    - e.g., author networks in CS, terrorist networks
  - Multiple heterogeneous networks
    - A person could be multiple information networks: friends, family, classmates, …
  - Links carry a lot of semantic information: Link mining
- Web mining
  - Web is a big information network: from PageRank to Google
  - Analysis of Web information networks
    - Web community discovery, opinion mining, usage mining, …

Evaluation of Knowledge

- Are all mined knowledge interesting?
  - One can mine tremendous amount of "patterns" and knowledge
  - Some may fit only certain dimension space (time, location, …)
  - Some may not be representative, may be transient, …
- Evaluation of mined knowledge → directly mine only interesting knowledge?
  - Descriptive vs. predictive
  - Coverage
  - Typicality vs. novelty
  - Accuracy
  - Timeliness
  - …

Data Mining: Confluence of Multiple Disciplines



Why Confluence of Multiple Disciplines?

- Tremendous amount of data
  - Algorithms must be highly scalable to handle such as tera-bytes of data
- High-dimensionality of data
  - Micro-array may have tens of thousands of dimensions
- High complexity of data
  - Data streams and sensor data
  - Time-series data, temporal data, sequence data
  - Structure data, graphs, social networks and multi-linked data
  - Heterogeneous databases and legacy databases
  - Spatial, spatiotemporal, multimedia, text and Web data
  - Software programs, scientific simulations
- New and sophisticated applications

Applications of Data Mining

- 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
- Data mining and software engineering (e.g., IEEE Computer, Aug. 2009 issue)
- From major dedicated data mining systems/tools (e.g., SAS, MS SQL-Server Analysis Manager, Oracle Data Mining Tools) to invisible data mining

Introduction: Presentation Outline

- Why Data Analytics (or data mining)?
  - Courses you have Completed
  - Content of human mind
  - Sample data mining problems
- Data mining definition and KDD process
- Multidimensional view of data mining
- Overview of data mining functionalities
- **Issues in data mining**
- Course outline
- Summary and History of data mining society

Major Issues in Data Mining (1)

- Mining Methodology
  - Mining various and new kinds of knowledge
  - Mining knowledge in multi-dimensional space
  - Data mining: An interdisciplinary effort
  - Boosting the power of discovery in a networked environment
  - Handling noise, uncertainty, and incompleteness of data
  - Pattern evaluation and pattern- or constraint-guided mining
- User Interaction
  - Interactive mining
  - Incorporation of background knowledge
  - Presentation and visualization of data mining results

Major Issues in Data Mining (2)

- Efficiency and Scalability
  - Efficiency and scalability of data mining algorithms
  - Parallel, distributed, stream, and incremental mining methods
- Diversity of data types
  - Handling complex types of data
  - Mining dynamic, networked, and global data repositories
- Data mining and society
  - Social impacts of data mining
  - Privacy-preserving data mining
  - Invisible data mining

Introduction: Presentation Outline

- Why Data Analytics (or data mining)?
  - Courses you have Completed
  - Content of human mind
  - Sample data mining problems
- Data mining definition and KDD process
- Multidimensional view of data mining
- Overview of data mining functionalities
- Issues in data mining
- **Course outline**
- Summary and History of data mining society

Data Mining Concepts and Data Statistics

P. Krishna Reddy
IIIT Hyderabad

Documents Uploaded

- Data, Information, Knowledge, and Wisdom by Gene Bellinger, Durval Castro, Anthony Mills
- From Data to Wisdom: A Note by Russell Ackoff
- Data Science— A Systematic Treatment by M. TAMER ÖZSU, Communications of ACM, 2023
- Theoretical frameworks for data mining, Heikki Mannila, 2000.
- Data Science: A Comprehensive Overview, LONGBING CAO, ACM Computing Surveys, 2017

Reference

- Chapter 1 and Chapter 2 of the text book.

Detailed Syllabus

- **Unit1: Introduction, data and data preprocessing,** data summarization through characterization, discrimination and data cube techniques (9 hours)
- Unit 2: Concepts and algorithms for mining patterns and associations (10 hours)
- Unit 3: Concepts and algorithms related to classification and regression (10 hours)
- Unit 4: Concepts and algorithms for clustering the data (10 hours)
- Unit 5: Outlier analysis and future trends. (3 hours)

Presentation Outline

- **Data mining functionalities**
- Research Issues in Data mining
- Sources of data
- Data Objects and Attribute Types
- About Big Data
- Basic Statistical Descriptions of Data
- Data Visualization
- Case study
- Summary

Data Mining Functionalities

- Summarization
- Pattern mining, Association mining, correlation
- Classification
- Clustering
- Outlier analysis
- Sequential, trend and evolution analysis
- Structure and network analysis

Data Mining Function (1): Summarization

- Input: Large data
- Output: summarize/characterize an interested set of data and compare it with the contrasting sets at some high levels.
  - Data characterization: Summarizing the data of the class under study.
  - Data discrimination: Comparison of target class with one or a set of target class.
- Multidimensional concept description: Characterization and discrimination
  - Generalize, summarize, and contrast data characteristics, e.g., dry vs. wet region
- Output can be represented with pie charts, bar charts, curves, multidimensional data cubes, and multidimensional tables, including crosstabs

Concept description: Characterization and discrimination

- Generalize, summarize, and contrast data characteristics, e.g., dry vs. wet regions.
- Example

Initial Relation

Name	Gender	Major	Birth-Place	Birth_date	Residence	Phone #	GPA
Jim Woodman	M	CS	Vancouver,BC, Canada	8-12-76	3511 Main St., Richmond	687-4598	3.67
Scott Lachance	M	CS	Montreal, Que, Canada	28-7-75	345 1st Ave., Richmond	253-9106	3.70
Laura Lee	F	Physics	Seattle, WA, USA	25-8-70	125 Austin Ave., Burnaby	420-5232	3.83
…	…	…	…	…	…	…	…
Removed	Retained	Sci,Eng, Bus	Country	Age range	City	Removed	Excl, VG,..

Prime Generalized Relation

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

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

Tabulation of Data

- Tabulation facilitates the presentation of large information into concise way under different titles and sub-titles.

Table 1. Students

Class	Rural				Urban			
	Male		Female		Male		Female	
	Hostellers	Day Scholars	Hostellers	Day scholars	Hostellers	Day Scholars	Hostellers	Day scholars
Intermediate								
Graduate								
Post Graduate								

Summarization techniques

- Statistical measures
- Attribute oriented induction
- Data cube based OLAP methods
  - Multidimensional summarization

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?

Approaches to mine Association Rules

- Apriori approach
- FP Tree approach
- Hash-based itemset counting: A *k*-itemset whose corresponding hashing bucket count is below the threshold cannot be frequent
- Transaction reduction: A transaction that does not contain any frequent k-itemset is useless in subsequent scans
- Partitioning: Any itemset that is potentially frequent in DB must be frequent in at least one of the partitions of DB
- Sampling: mining on a subset of given data, lower support threshold + a method to determine the completeness
- Dynamic itemset counting: add new candidate itemsets only when all of their subsets are estimated to be frequent

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 applications:
  - Credit card fraud detection, direct marketing, classifying stars, diseases, web-pages, …

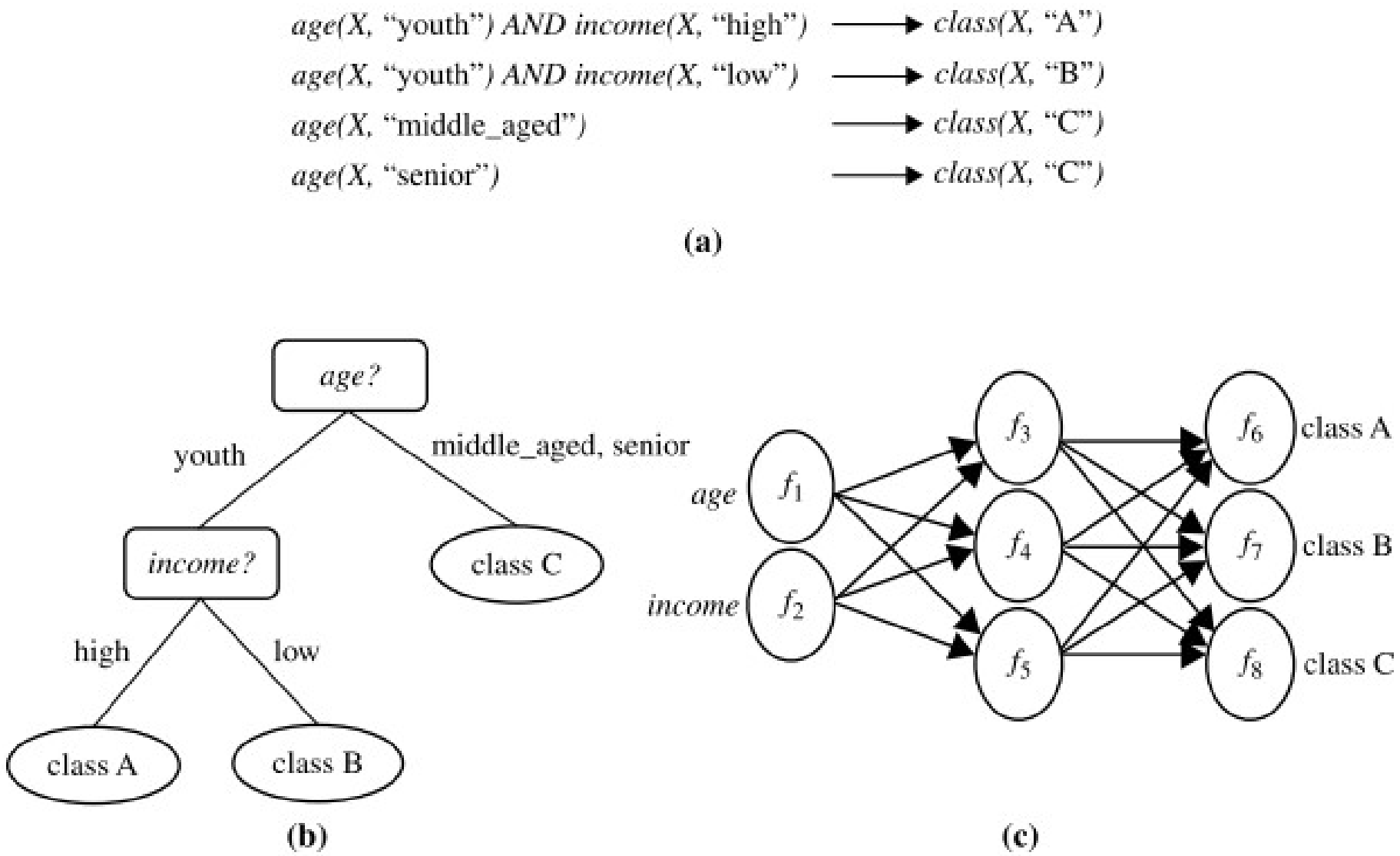


FIGURE 1.2

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

Approaches to classification

- Decision tree
- Bayesian Classification
- Neural networks
- Association based classification
- k-nearest neighbor classifier
- Ensemble classification

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

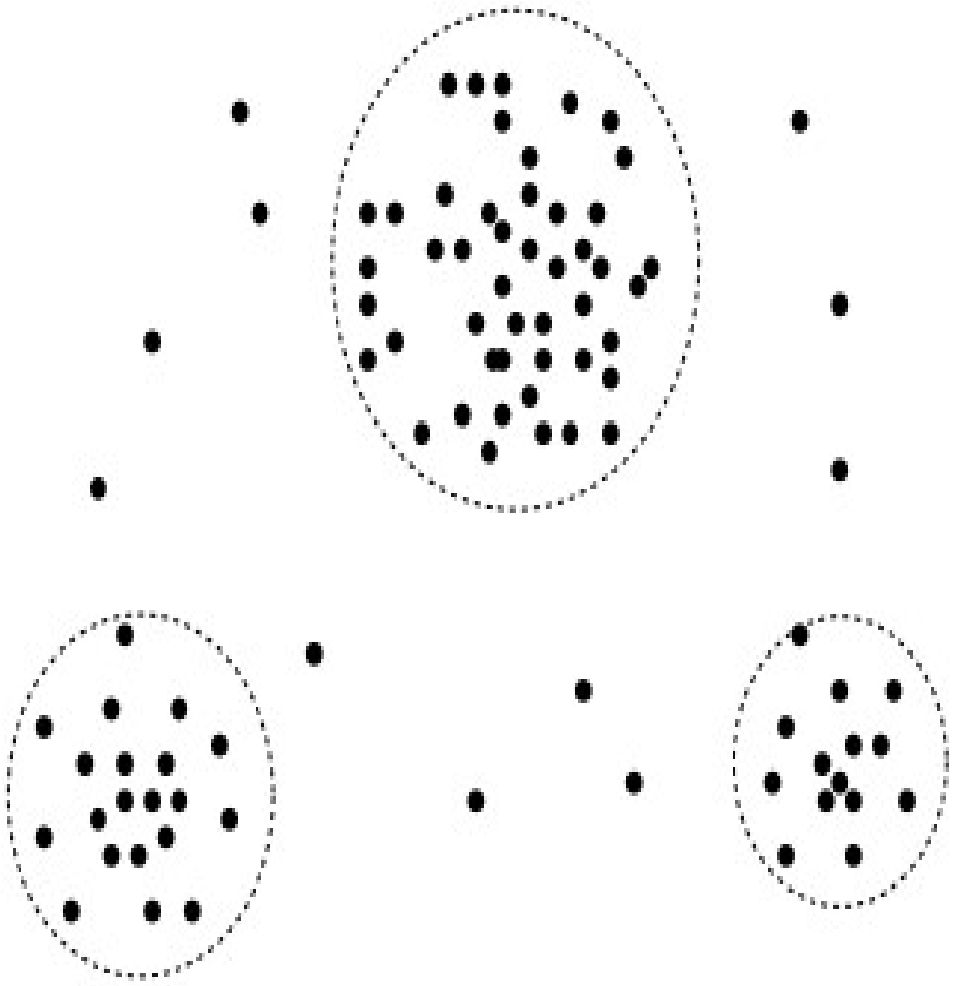


FIGURE 1.3

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

Major Clustering Approaches

- Partitioning algorithms: Construct various partitions and then evaluate them by some criterion
- Hierarchy algorithms: Create a hierarchical decomposition of the set of data (or objects) using some criterion
- Density-based: based on connectivity and density functions
- Grid-based: based on a multiple-level granularity structure
- Model-based: A model is hypothesized for each of the clusters and the idea is to find the best fit of that model to each other

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

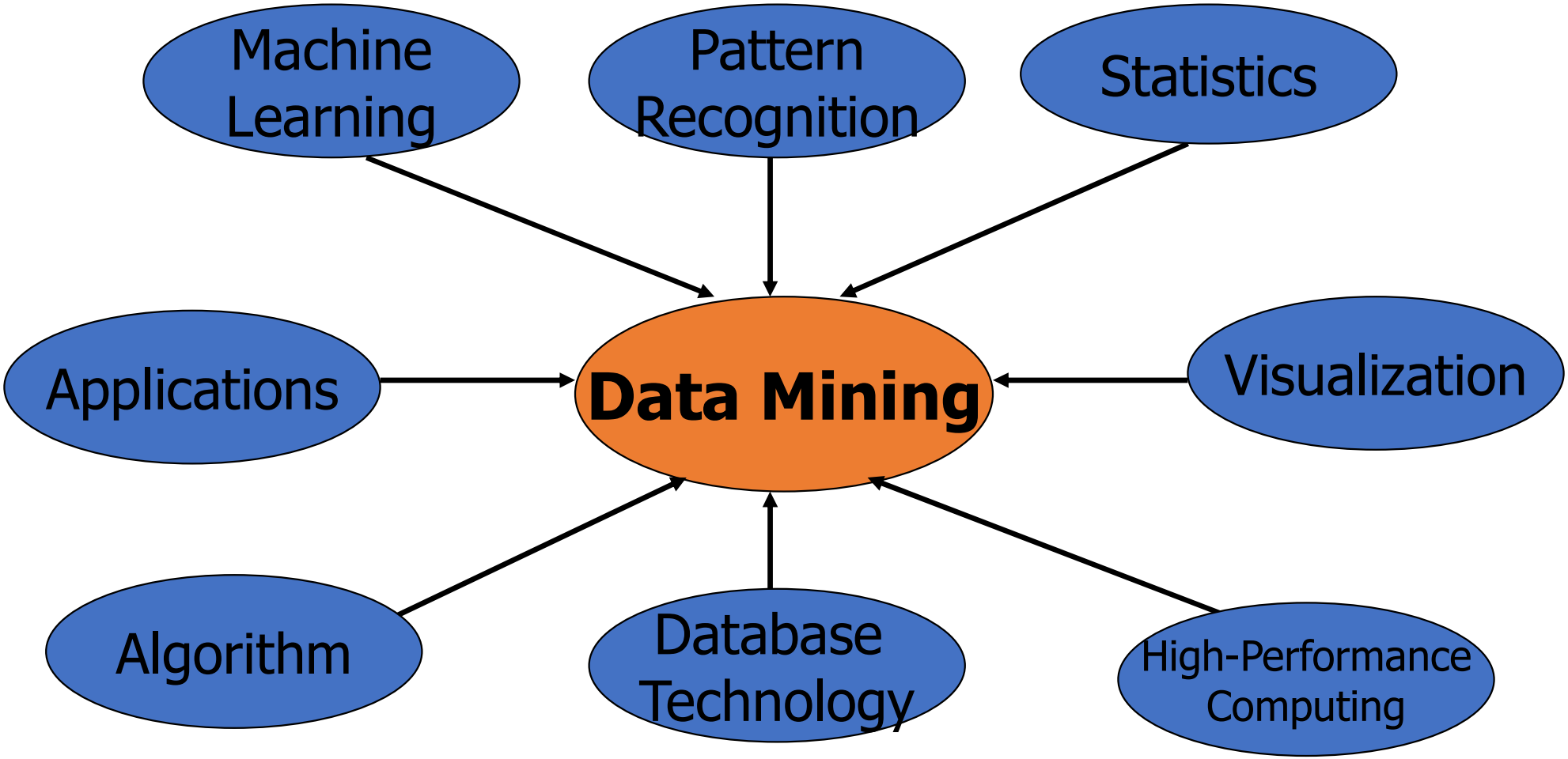
Time and Ordering: Sequential Pattern, Trend and Evolution Analysis

- Sequence, trend and evolution analysis
  - Trend, time-series, and deviation analysis: e.g., regression and value prediction
  - Sequential pattern mining
    - e.g., first buy digital camera, then buy large SD memory cards
  - Periodicity analysis
  - Motifs and biological sequence analysis
    - Approximate and consecutive motifs
  - Similarity-based analysis
- Mining data streams
  - Ordered, time-varying, potentially infinite, data streams

Structure and Network Analysis

- Graph mining
  - Finding frequent subgraphs (e.g., chemical compounds), trees (XML), substructures (web fragments)
- Information network analysis
  - Social networks: actors (objects, nodes) and relationships (edges)
    - e.g., author networks in CS, terrorist networks
  - Multiple heterogeneous networks
    - A person could be multiple information networks: friends, family, classmates, …
  - Links carry a lot of semantic information: Link mining
- Web mining
  - Web is a big information network: from PageRank to Google
  - Analysis of Web information networks
    - Web community discovery, opinion mining, usage mining, …

Data Mining: Confluence of Multiple Disciplines



Presentation Outline

- Data mining functionalities
- **Research Issues in Data mining**
- Sources of data
- Data Objects and Attribute Types
- About Big Data
- Basic Statistical Descriptions of Data
- Data Visualization
- Case study
- Summary

About Big Data

How much data?

- Google processes 20 PB a day (2008)
- Facebook has 2.5 PB of user data + 15 TB/day (4/2009)
- eBay has 6.5 PB of user data + 50 TB/day (5/2009)
- CERN's Large Hydron Collider (LHC) generates 15 PB a year

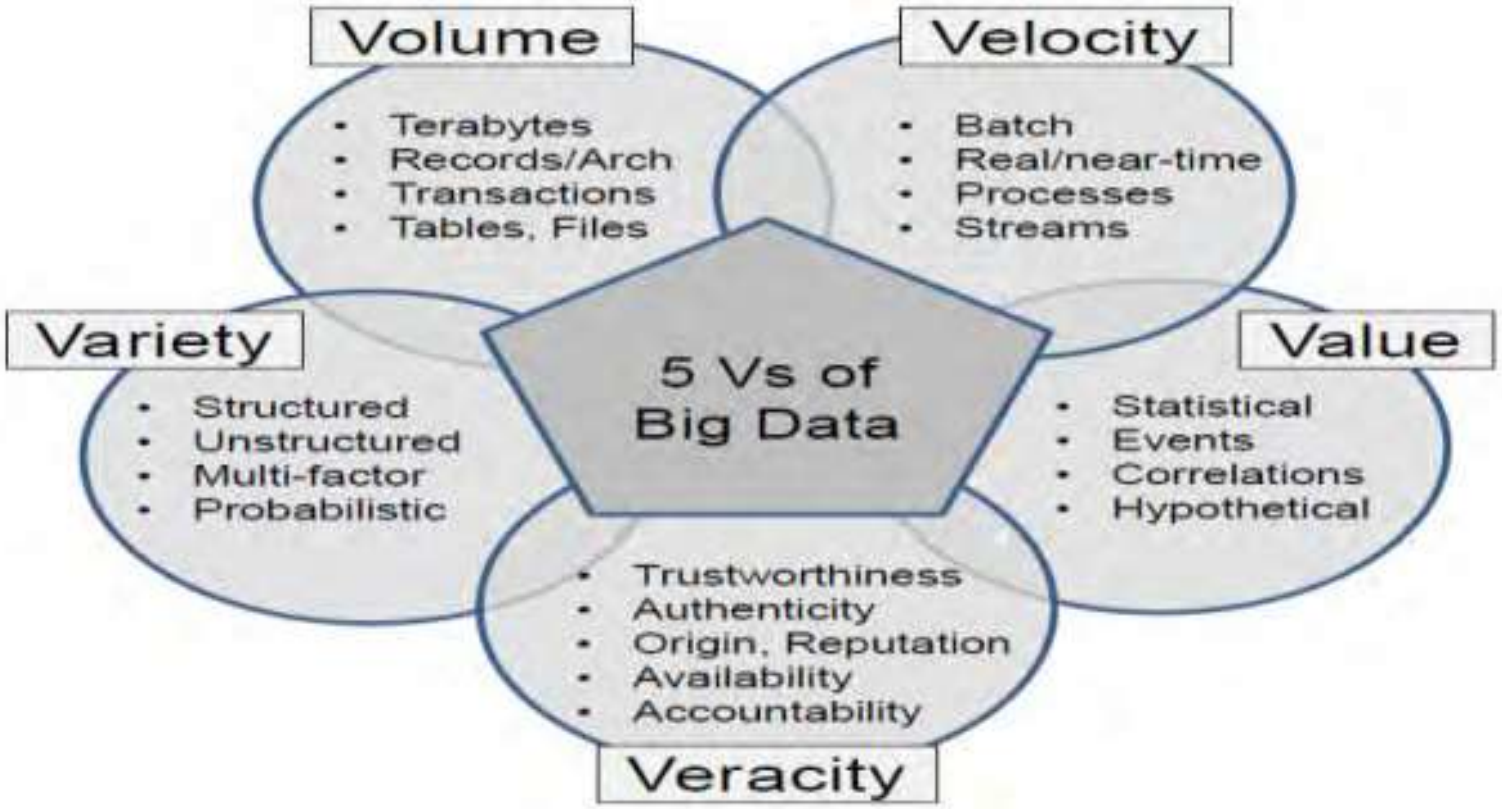


Big Data Definition

- No single standard definition…

"***Big Data***" is data whose scale, diversity, and complexity require new architecture, techniques, algorithms, and analytics to manage it and extract value and hidden knowledge from it…

Characteristics of Big data



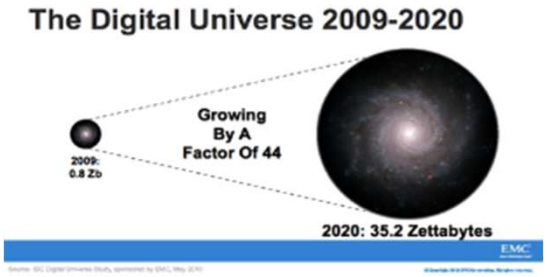
Volume

- The costs of computing, storage, and connectivity resources are plunging, and new technologies like scanners, smartphones, ubiquitous video, and other data-collectors mean we are awash in volumes of data that dwarf what was available even five to 10 years ago.

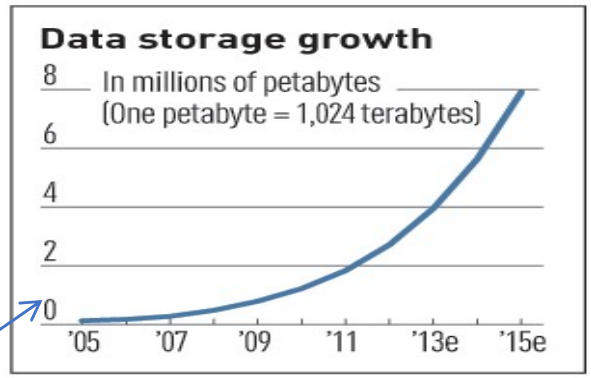
- We capture every mouse click, phone call, text message, Web search, transaction, and more. As the volume of data grows, we can learn more – but only if we uncover meaningful relationships and patterns.

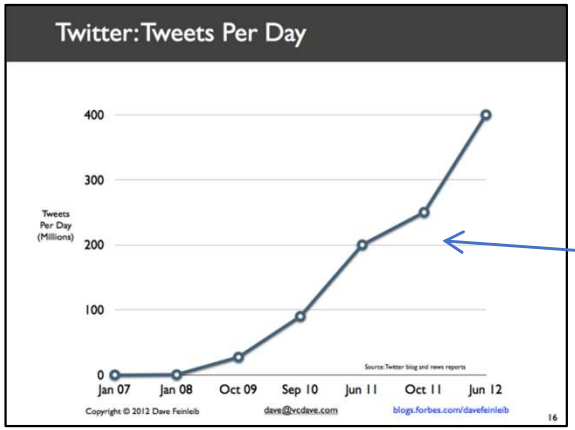
Volume..

- **Data Volume**
  - 44x increase from 2009 2020
  - From 0.8 zettabytes to 35zb
- Data volume is increasing exponentially





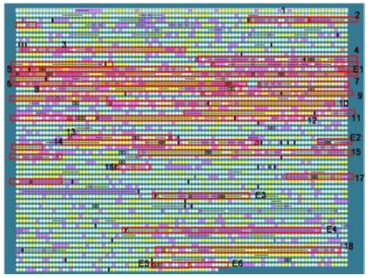


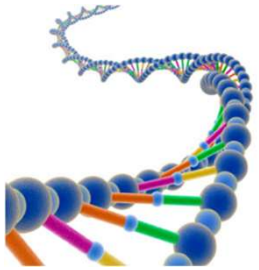


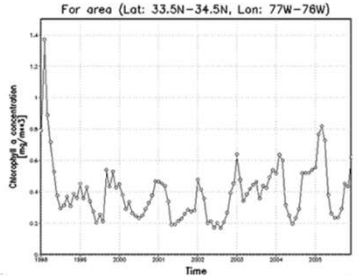
Exponential increase in collected/generated data

Variety

- Various formats, types, and structures
- Text, numerical, images, audio, video, sequences, time series, social media data, multi-dim arrays, etc…
- Static data vs. streaming data
- A single application can be generating/collecting many types of data

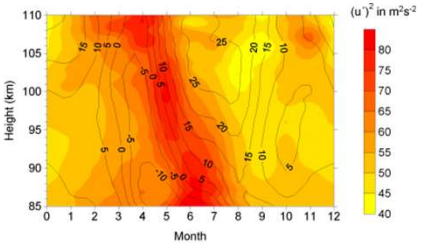












To extract knowledge➔ all these types of data need to linked together

Velocity

- Data is begin generated fast and need to be processed fast
- Online Data Analytics
- Late decisions ➔ missing opportunities



- **Examples**
  - **E-Promotions:** Based on your current location, your purchase history, what you like ➔ send promotions right now for store next to you
  - **Healthcare monitoring:** sensors monitoring your activities and body ➔ any abnormal measurements require immediate reaction

Veracity

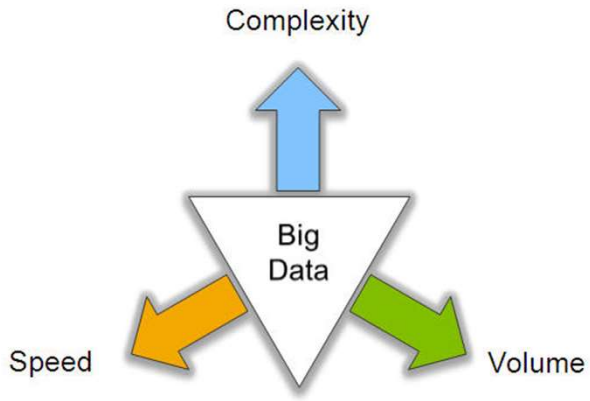
- Veracity indicates the inherent trustworthiness of data.
- The uncertainty about the consistency or completeness of data and other ambiguities can become major obstacles.
- As a result, basic principles as data quality, data cleansing, master data management, and data governance remain critical disciplines when working with Big Data.

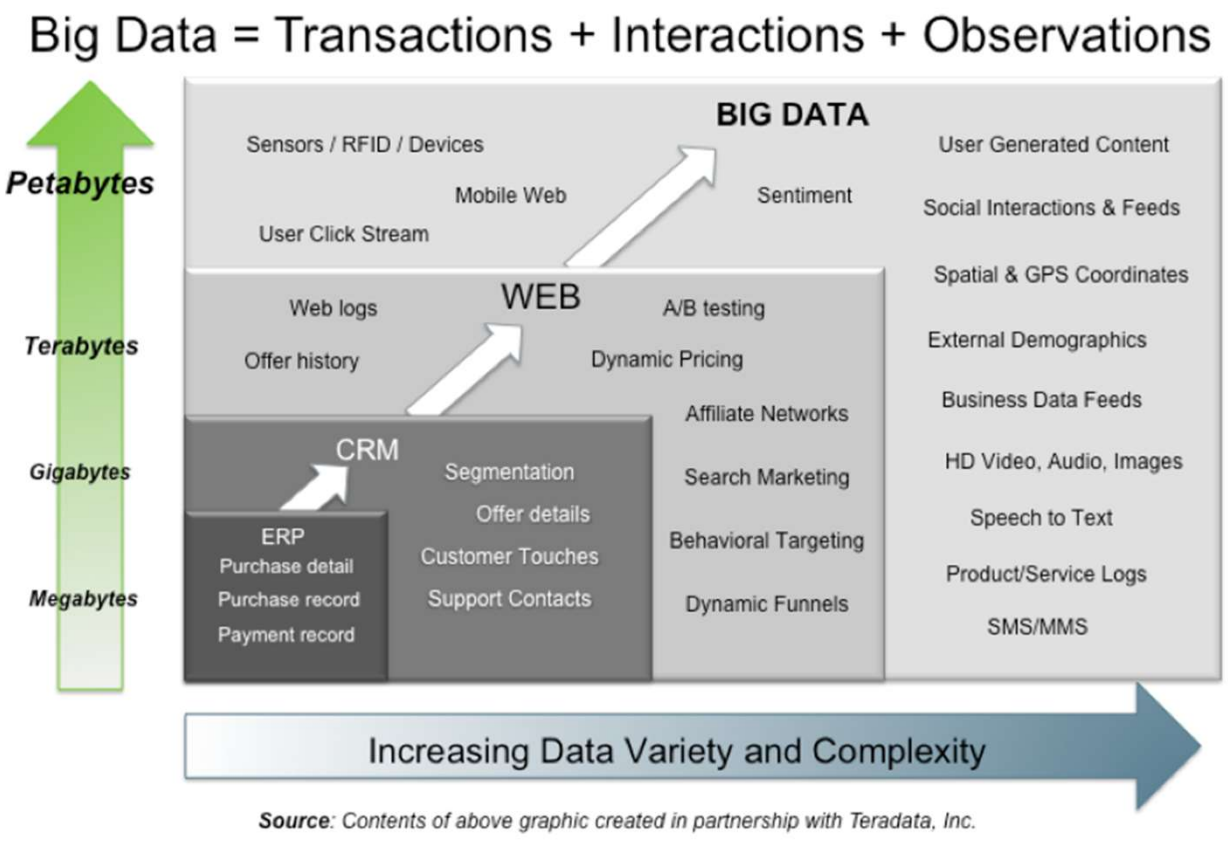
Value

- Access to big data is no good, unless it turned into a good value.
- Return on investment

Big Data: 3V's

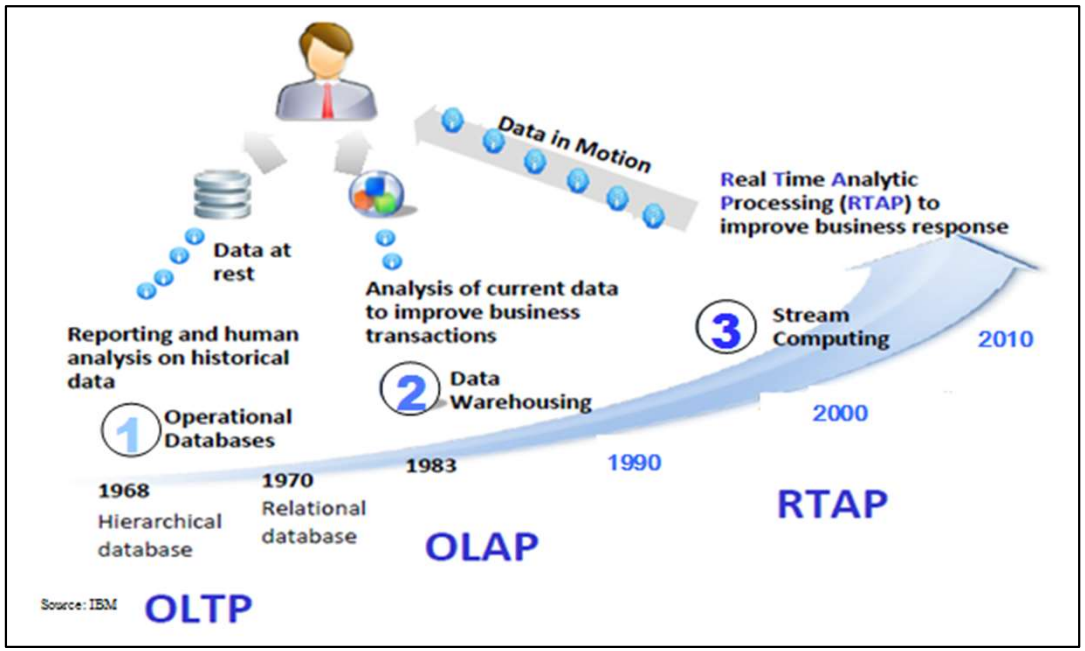






Source: Contents of above graphic created in partnership with Teradata, Inc.

Harnessing Big Data



- **OLTP:** Online Transaction Processing (DBMSs)
- **OLAP:** Online Analytical Processing (Data Warehousing)
- **RTAP:** Real-Time Analytics Processing (Big Data Architecture & technology)

Who's Generating Big Data



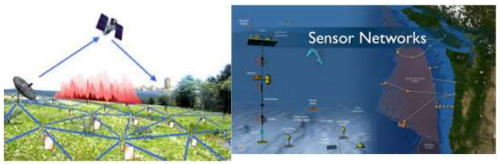
Social media and networks
(all of us are generating data)



Scientific instruments
(collecting all sorts of data)



Mobile devices
(tracking all objects all the time)



Sensor technology and networks
(measuring all kinds of data)

- The progress and innovation is no longer hindered by the ability to collect data
- But, by the ability to manage, analyze, summarize, visualize, and discover knowledge from the collected data in a timely manner and in a scalable fashion

The Model Has Changed…

- **The Model of Generating/Consuming Data has Changed**

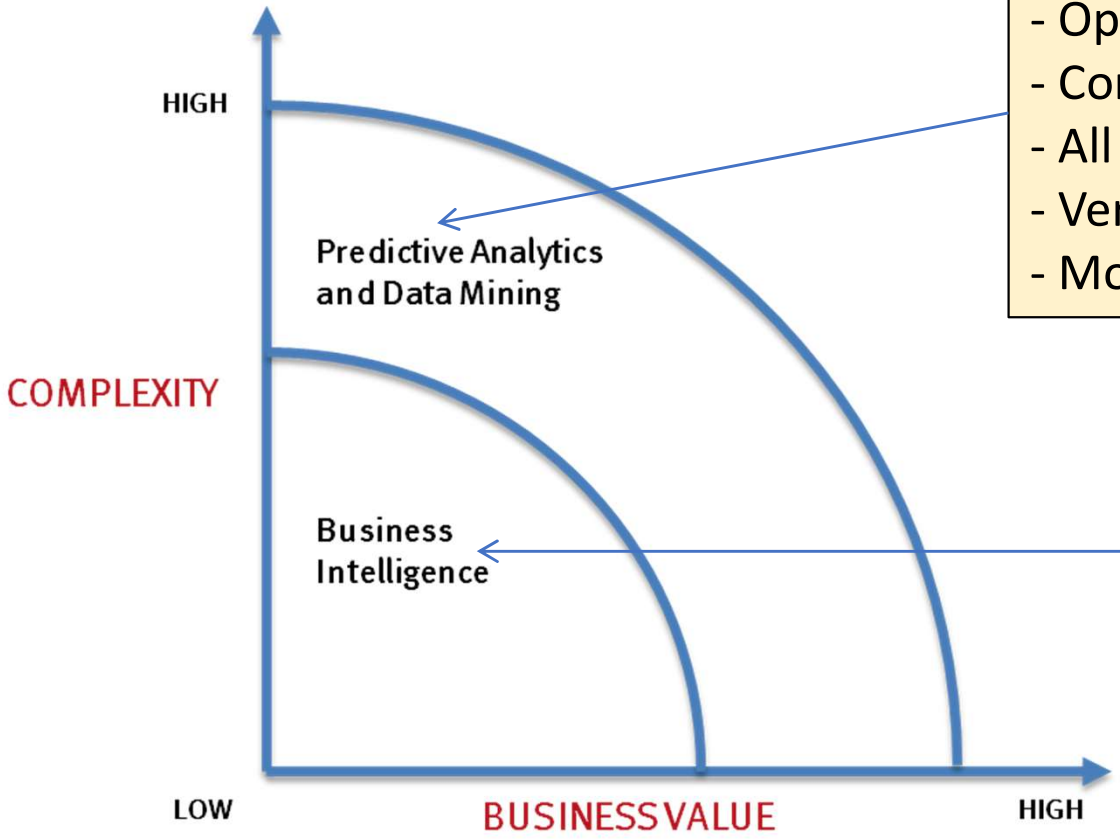
Old Model: Few companies are generating data, all others are consuming data



New Model: all of us are generating data, and all of us are consuming data



What's driving Big Data

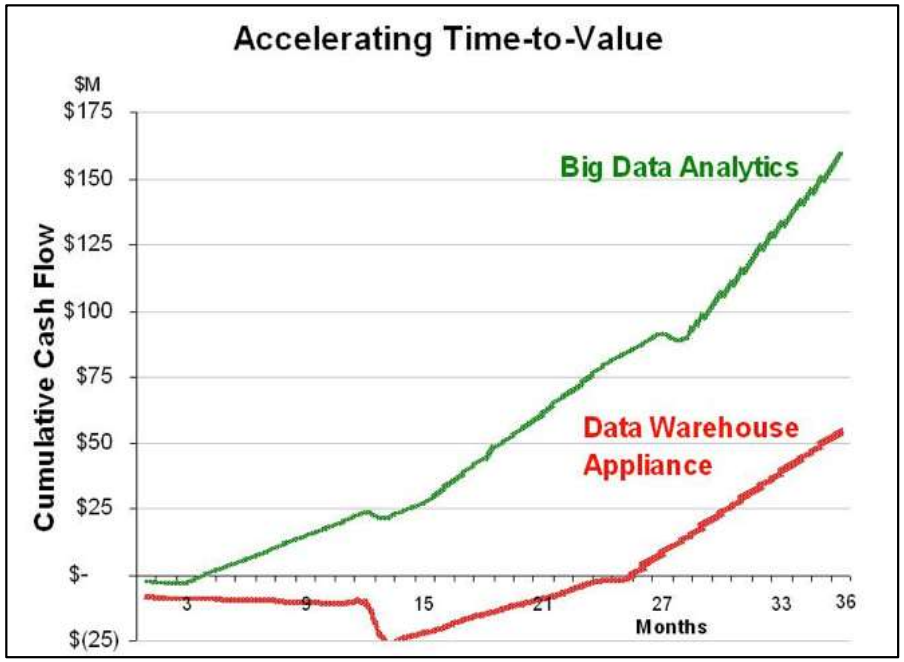


- Optimizations and predictive analytics
- Complex statistical analysis
- All types of data, and many sources
- Very large datasets
- More of a real-time

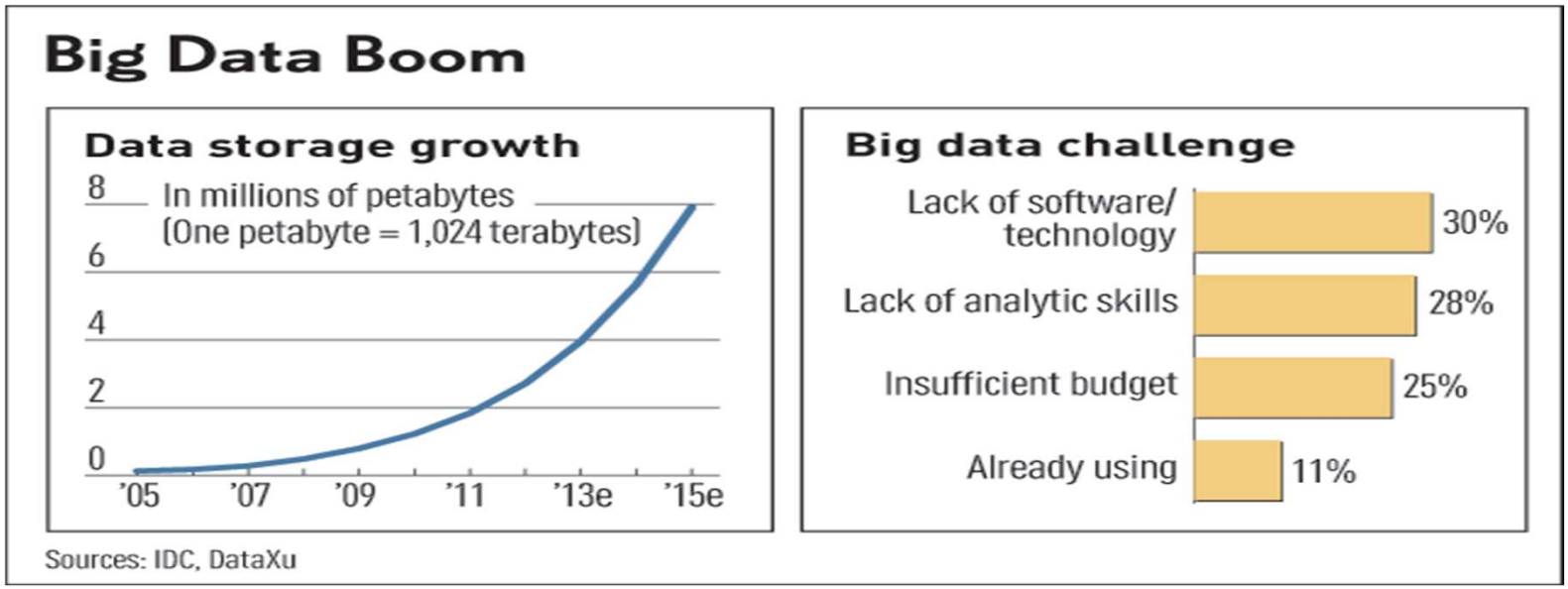
- Ad-hoc querying and reporting
- Data mining techniques
- Structured data, typical sources
- Small to mid-size datasets

Value of Big Data Analytics

- Big data is more real-time in nature than traditional DW applications
- Traditional DW architectures (e.g. Exadata, Teradata) are not well-suited for big data apps
- Shared nothing, massively parallel processing, scale out architectures are well-suited for big data apps

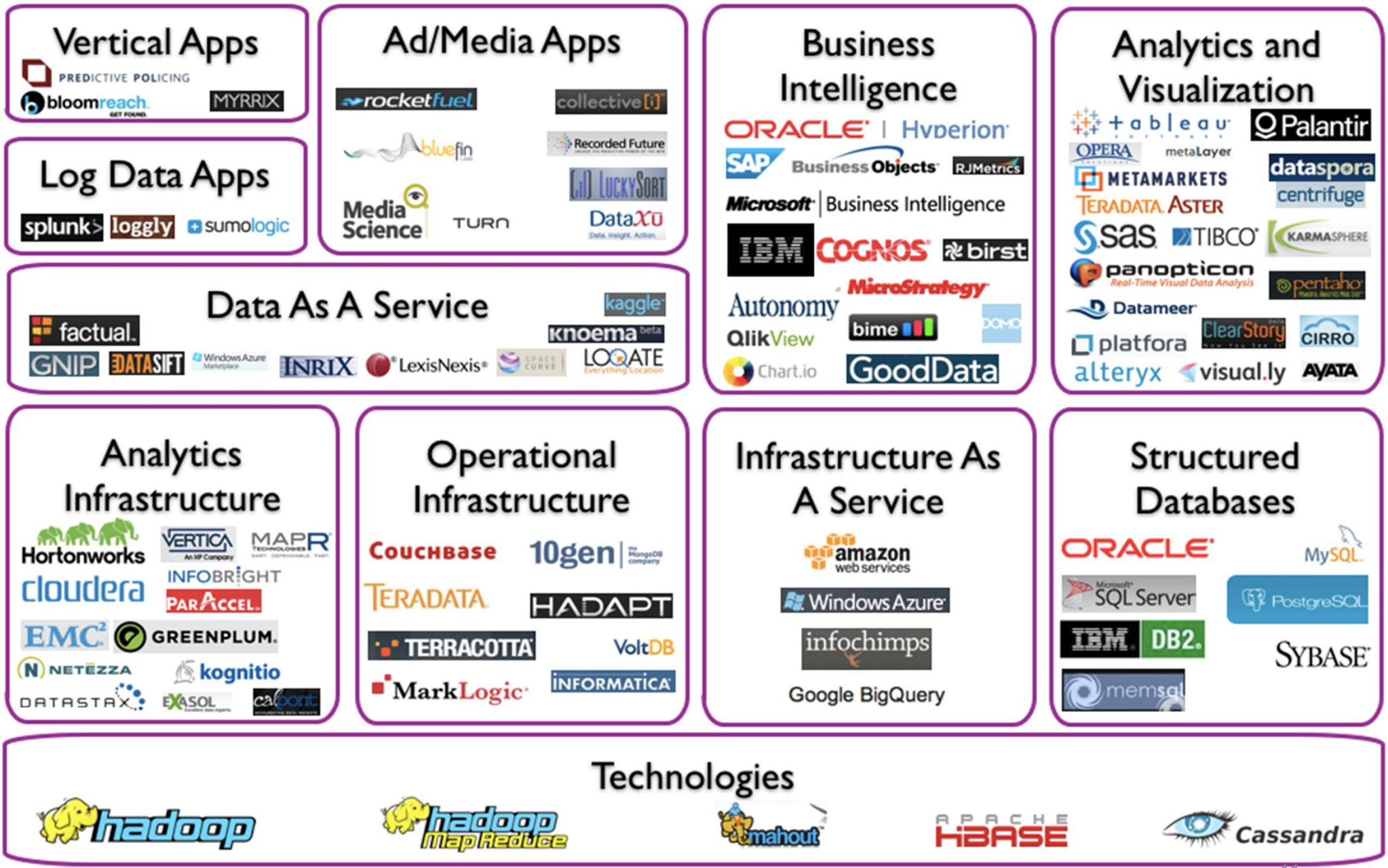


Challenges in Handling Big Data



- **The Bottleneck is in technology**
  - New architecture, algorithms, techniques are needed
- **Also in technical skills**
  - Experts in using the new technology and dealing with big data

Big Data Landscape

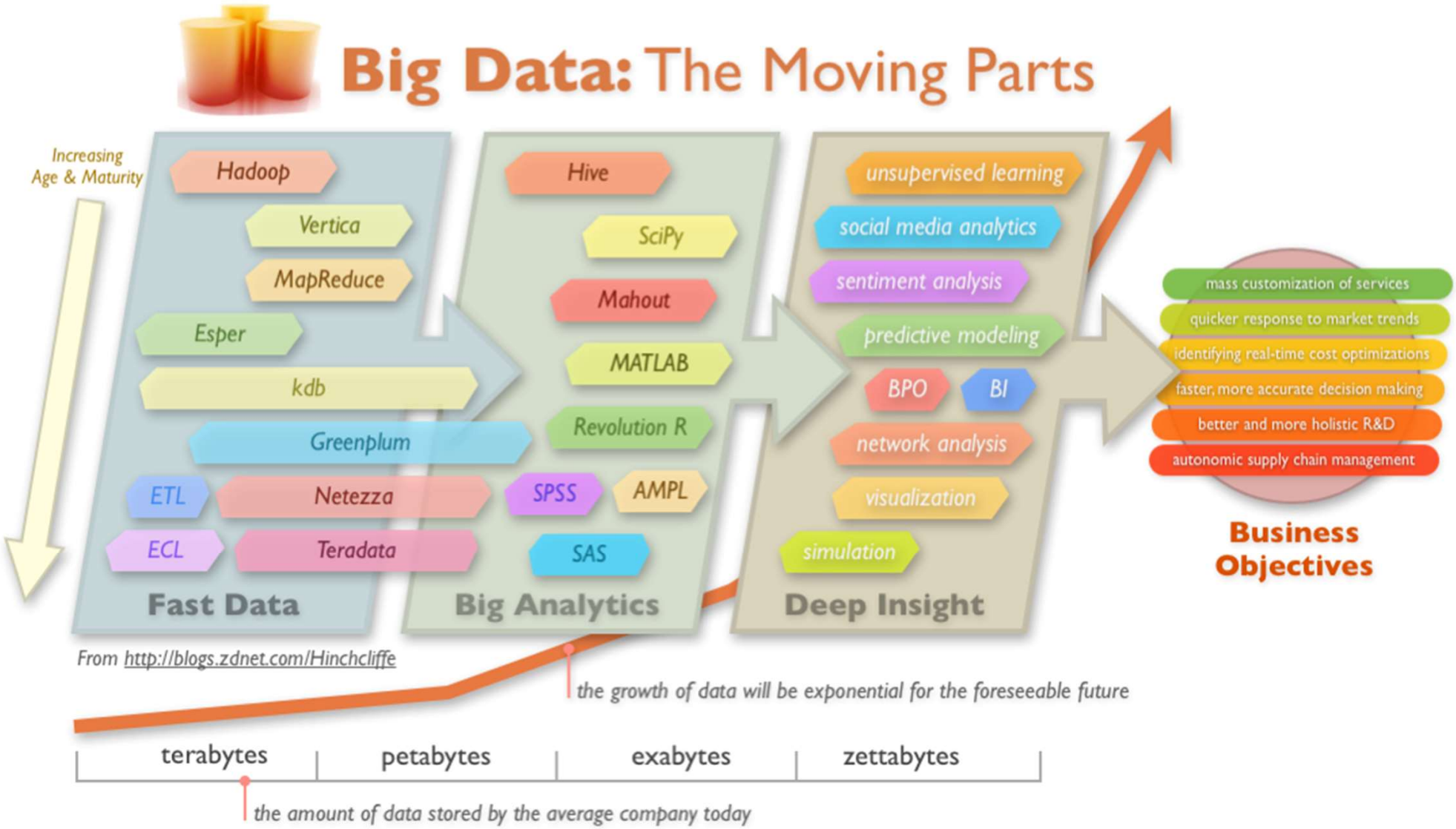


Copyright © 2012 Dave Feinleib

dave@vcdave.com

blogs.forbes.com/davefeinleib

Big Data Technology



About Big Data: Conclusion

- Beginning of big data economy.
- Big data and data science will bring a major social change.
- Companies which fail to exploit big data runs the risk of left behind.
- Should be exploited for sustainable development and equitable society

Evaluation of Knowledge

- Are all mined knowledge interesting?
  - One can mine tremendous amount of “patterns” and knowledge
  - Some may fit only certain dimension space (time, location, …)
  - Some may not be representative, may be transient, …
- Evaluation of mined knowledge → directly mine only interesting knowledge?
  - Descriptive vs. predictive
  - Coverage
  - Typicality vs. novelty
  - Accuracy
  - Timeliness
  - …

Applications of Data Mining

- 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
- Data mining and software engineering (e.g., IEEE Computer, Aug. 2009 issue)
- From major dedicated data mining systems/tools (e.g., SAS, MS SQL-Server Analysis Manager, Oracle Data Mining Tools) to invisible data mining

Why Confluence of Multiple Disciplines?

- Tremendous amount of data
  - Algorithms must be highly scalable to handle such as tera-bytes of data
- High-dimensionality of data
  - Micro-array may have tens of thousands of dimensions
- High complexity of data
  - Data streams and sensor data
  - Time-series data, temporal data, sequence data
  - Structure data, graphs, social networks and multi-linked data
  - Heterogeneous databases and legacy databases
  - Spatial, spatiotemporal, multimedia, text and Web data
  - Software programs, scientific simulations
- New and sophisticated applications

Major Issues in Data Mining (1)

- Mining Methodology
  - Mining various and new kinds of knowledge
  - Mining knowledge in multi-dimensional space
  - Data mining: An interdisciplinary effort
  - Boosting the power of discovery in a networked environment
  - Handling noise, uncertainty, and incompleteness of data
  - Pattern evaluation and pattern- or constraint-guided mining
- User Interaction
  - Interactive mining
  - Incorporation of background knowledge
  - Presentation and visualization of data mining results

Major Issues in Data Mining (2)

- Efficiency and Scalability
  - Efficiency and scalability of data mining algorithms
  - Parallel, distributed, stream, and incremental mining methods
- Diversity of data types
  - Handling complex types of data
  - Mining dynamic, networked, and global data repositories
- Data mining and society
  - Social impacts of data mining
  - Privacy-preserving data mining
  - Invisible data mining

Important Characteristics of Structured Data

- Dimensionality
  - Curse of dimensionality
- Sparsity
  - Only presence counts
  - Bag-of-words
- Resolution
  - Patterns depend on the scale
- Distribution
  - Centrality and dispersion

Multi-Dimensional View of Data Mining

- **Data to be mined**
  - Database data (extended-relational, object-oriented, heterogeneous, legacy), data warehouse, transactional data, stream, spatiotemporal, time-series, sequence, text and web, multi-media, graphs & social and information networks
- **Knowledge to be mined (or: Data mining functions)**
  - summarization association, classification, clustering, trend/deviation, outlier analysis, etc.
  - Descriptive vs. predictive data mining
  - Multiple/integrated functions and mining at multiple levels
- **Techniques utilized**
  - Data-intensive, data warehouse (OLAP), machine learning, statistics, pattern recognition, visualization, high-performance, etc.
- **Applications adapted**
  - Retail, telecommunication, banking, fraud analysis, bio-data mining, stock market analysis, text mining, Web mining, etc.

Research Issues in Data Mining

- Data mining (knowledge discovery in databases):
  - Extraction of interesting (non-trivial, implicit, previously unknown and potentially useful) information or patterns from data in large databases
- So far, data mining means
  - Summarization, Association rules, clustering, classification
- Research Issues
  - Finding new patterns
    - Market basket data, Complex data, stream data
  - Improving the performance of existing algorithms
    - Scalable algorithms
      - Data, features or dimensions
    - Complex data
  - Visualizing the patterns

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

Presentation Outline

- Data mining functionalities
- Research Issues in Data mining
- **Sources of data**
- Data Objects and Attribute Types
- About Big Data
- Basic Statistical Descriptions of Data
- Data Visualization
- Case study
- Summary

Sources of data

- Primary data
  - Primary data or raw data is **a type of information that is obtained directly from first-hand sources through experiments, surveys, or observations**.

- Examples of primary data
  - Autobiographies and memoirs.
  - Diaries, personal letters, and correspondence.
  - Interviews, surveys, and fieldwork.
  - Internet communications on email, blogs, listservs, and newsgroups.
  - Photographs, drawings, and posters.
  - Works of art and literature.

- Secondary data
  - Secondary data means **data collected by someone else earlier**.

- Examples of secondary data
  - Tax records and social security data.
  - Census data.
  - Electoral statistics.
  - Health records.
  - Books, journals, or other print media.
  - Social media monitoring, internet searches, and other online data.
  - Sales figures or other reports from third-party companies.

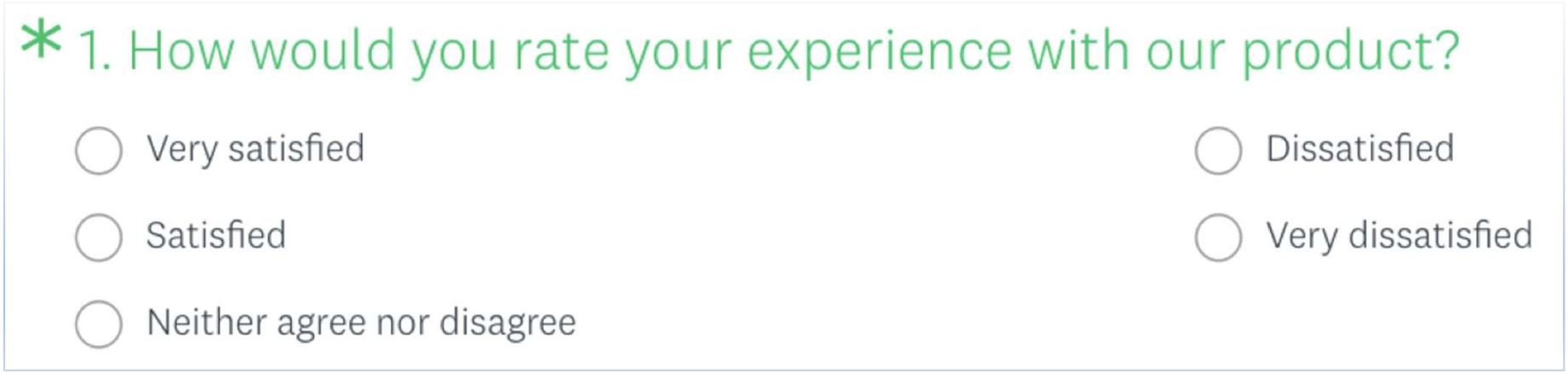
Example of primary data collection

- Quantitative methods
  - Quantitative research is the process of collecting and analyzing numerical data. It can be used to find patterns and averages, make predictions, test causal relationships, and generalize results to wider populations.
- Qualitative methods
  - Examples
  - One-on-one interviews.
    - Interviews are one of the most common qualitative data-collection methods, and they're a great approach when you need to gather highly personalized information
  - Open-ended surveys and questionnaires.
  - Focused groups
  - Observation
  - Case studies

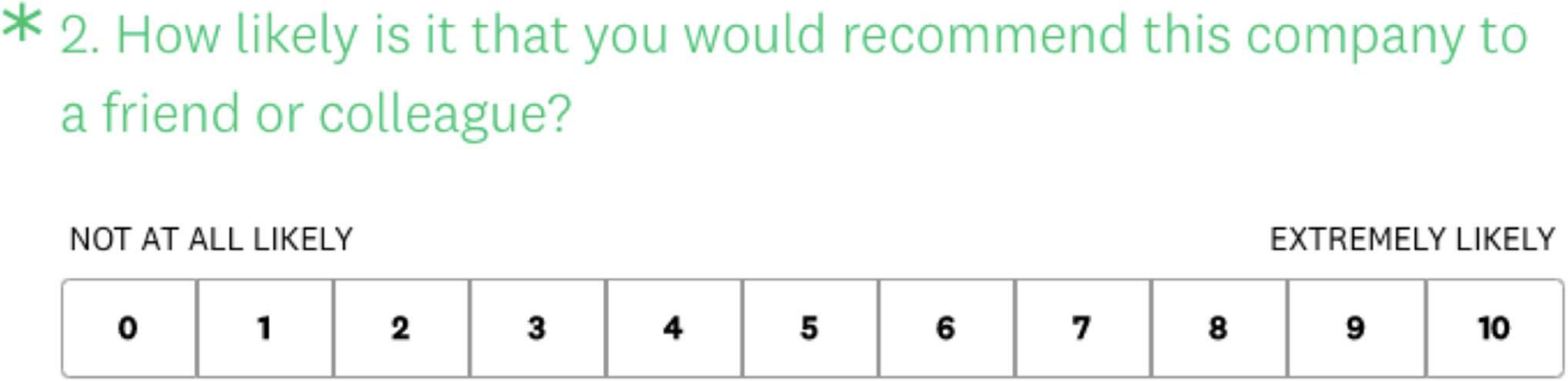
A few sample questions

- Next few slides contain a few sample survey questions

Single-answer multiple choice question.



Rating scales questions



Source: https://www.surveymonkey.com/mp/survey-question-types/

Likert scales: Do you agree or disagree

8. I'm satisfied with the investment my organization makes in education:

- ◯ Strongly agree
- ◯ Agree
- ◯ Neither agree nor disagree
- ◯ Disagree
- ◯ Strongly disagree

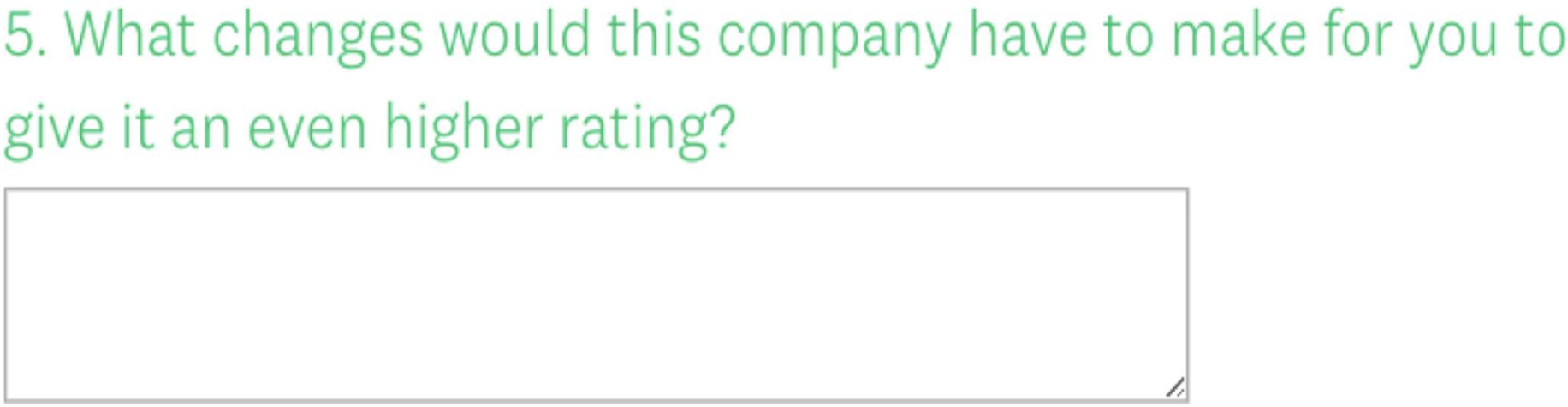
Dropdown questions

3. What's your age?



Open-ended questions

- Open-ended survey questions require respondents to type their answer into a comment box and don't provide specific pre-set answer options. Responses are then viewed individually or by text analysis tools.



Demographic questions

14. Which of the following best describes your current relationship status?

- ◯ Married
- ◯ Widowed
- ◯ Divorced
- ◯ Separated
- ◯ In a domestic partnership or civil union
- ◯ Single, but cohabiting with a significant other
- ◯ Single, never married

Ranking questions

5. Rank the following shows in order of preference—1 being your favorite and 5 being your least favorite.

- ⠿ [ ] The office
- ⠿ [ ] Parks and Recreation
- ⠿ [ ] Arrested Development
- ⠿ [ ] Orange is the New Black
- ⠿ [ ] New Girl

Image choice questions

7. Now that you've reviewed the logos, please pick your favorite.





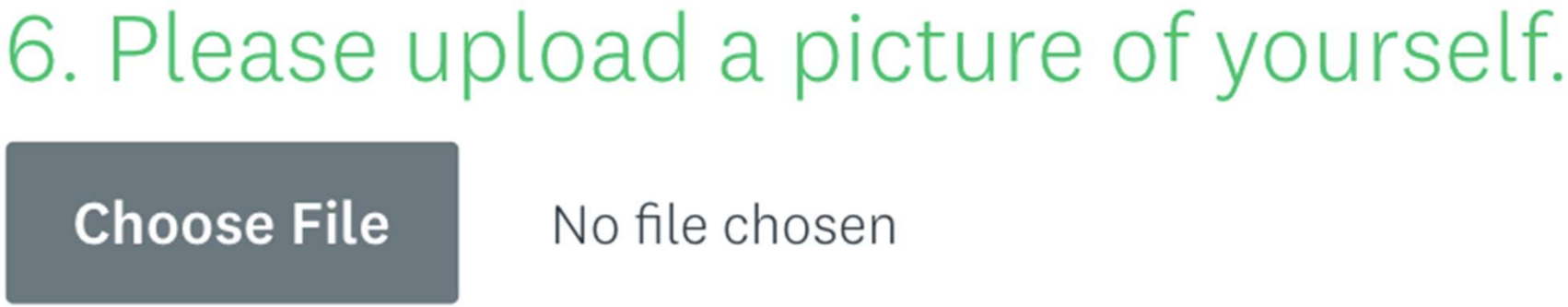


Click map questions

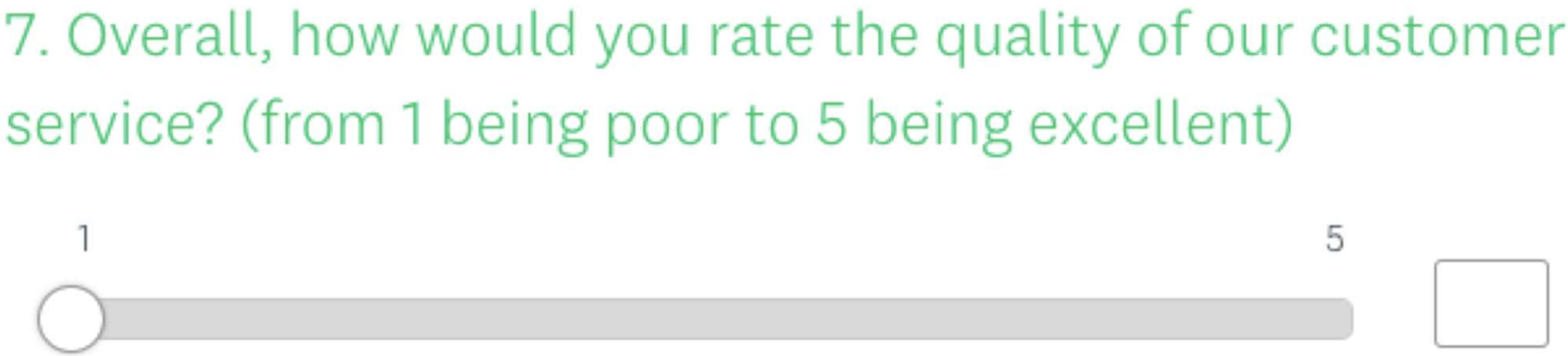
Click the part of the packaging that is the most appealing to you.



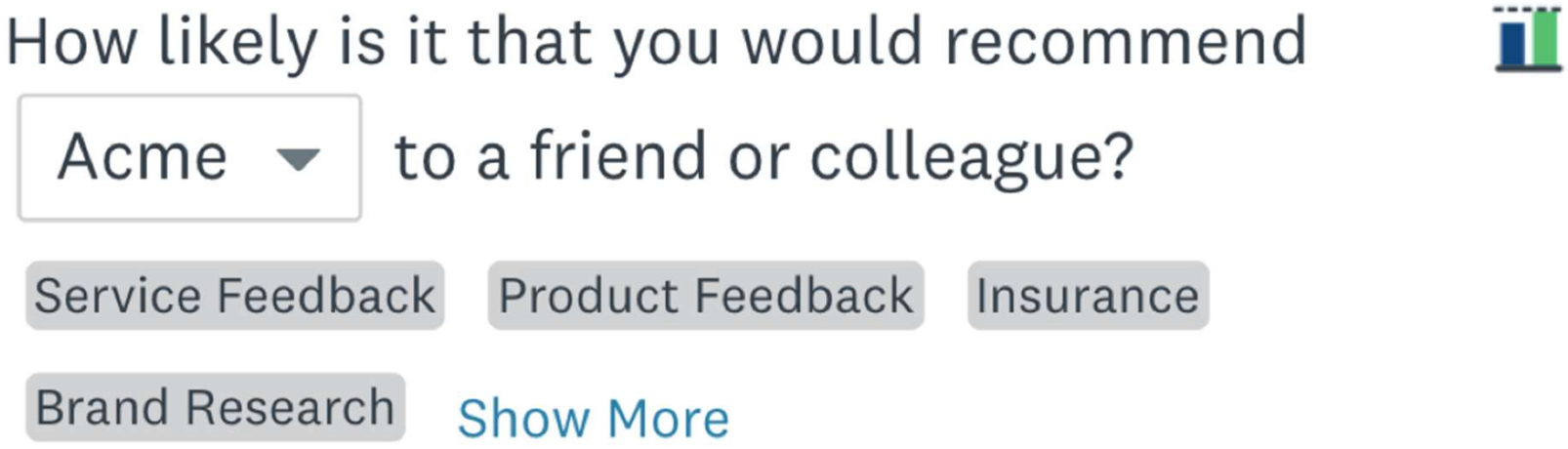
File upload questions: Uploading of resume or image



Slider questions



Benchmarkable questions



Form Number 2802 3739

Census of India 2021 Houselisting and Housing Census Schedule

Confidential when filled | Page No. | SIDE- A

ion ulars | State/UT | District | Tahsil/Taluk/ P.S./Dev. Block Circle/Mandal | Town/ Village | Ward Code No. (only for Town) | Houselisting Block No.

Fill if the census house is used wholly or partly as a residence (If '1' or '2' in col. 7)

Fill columns 14 to 34 for normal households. Put dash (-) in case of institutional households and non-residential census houses

Amenities available to the household | Assets possessed by the household

Building number (Municipal / local authority / census number)	Census house number	Predominant material of floor, wall and roof of the census house (Give code from the respective lists below) Floor	Wall	Roof	Ascertain use of Census house (Write the actual use and then choose the appropriate code from the list below and record the same in the box at the left hand side of the column) Code No.	Actual use	If '1' or '2' in col. 7, condition of this census house: Good-1 / Livable- 2 / Dilapidated-3	Household number (Give separate serial number to each household and write '999' for every institutional household)	Total number of persons normally residing in this household	Name of the head of the household	Do not fill columns 12 and 13 for institutional households: Sex: Male-1/ Female-2/ Transgender person-3	If SC* or ST* or Other: SC -1/ ST - 2 / Other - 3	Ownership status of this house: Owned-1/ Rented but has own house elsewhere-2/ Rented and doesn't own any house-3/ Any other-4	Number of dwelling rooms # exclusively in possession of this household (Record 0,1,2,3..)	Number of married couple(s) living in this household (Record 0, 1, 2, 3....)	Main source of drinking water (Give code from the list below)	Availability of drinking water source: S Within premises-1/ Near the premises-2/ Away-3	Main source of lighting: Electricity-1/ Kerosene-2 Solar-3/ Other oil-4/ Any other-5/ No lighting-6	Access to latrine: Yes: Exclusively for household use only -1/ Shared with other household-2/ Public latrine-3/ No: Open-4	If '1' or '2' in col. 20, then type of latrine (Give code from the list below)	Waste water outlet connected to: Closed drainage-1/Open drainage-2/ No drainage-3	Bathing facility available within the premises: Yes: Bathroom-1/ Enclosure without roof-2/ No-3	Availability of kitchen and LPG/PNG Connection: (Give code from the list below)	Main fuel used for cooking: (If '1' to '6' in col. 24) (Give code from the list below)	Radio/ Transistor: Yes: Traditional radio set-1/ On mobile/smartphone-2/ On any other device-3 /No-4	Television: Yes: Doordarshan free dish-1/ Other DTH/Dish-2/ Cable connection-3/ Any other-4/No -5	Access to internet: Yes: On laptop/Computer-1/ On mobile/smartphone-2/ On any other device-3/ No-4	Laptop/ Computer: Yes: 1/ No-2	Telephone and Mobile phone/Smartphone (Give code from the list below)	Bicycle and Scooter/ Motorcycle/ Moped (Give code from list below)	Car/ Jeep/ Van: Yes: 1/ No-2	Main cereal consumed in the household: Rice-1/ Wheat-2/ Jowar-3/ Bajra-4/ Maize-5/ Any other-6	Mobile Number (For Census related communications only)
2	3	4	5	6	7		8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34					

End of 2nd Lecture

Presentation Outline

- Data mining functionalities
- Research Issues in Data mining
- Sources of data
- **Data Objects and Attribute Types**
- Basic Statistical Descriptions of Data
- Data Visualization
- Case study
- Summary

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, records, vector, patten, event, case, observation or entity*
- Data objects are described by **attributes**.
- Database rows -> data objects; columns ->attributes.

Types of Data Sets

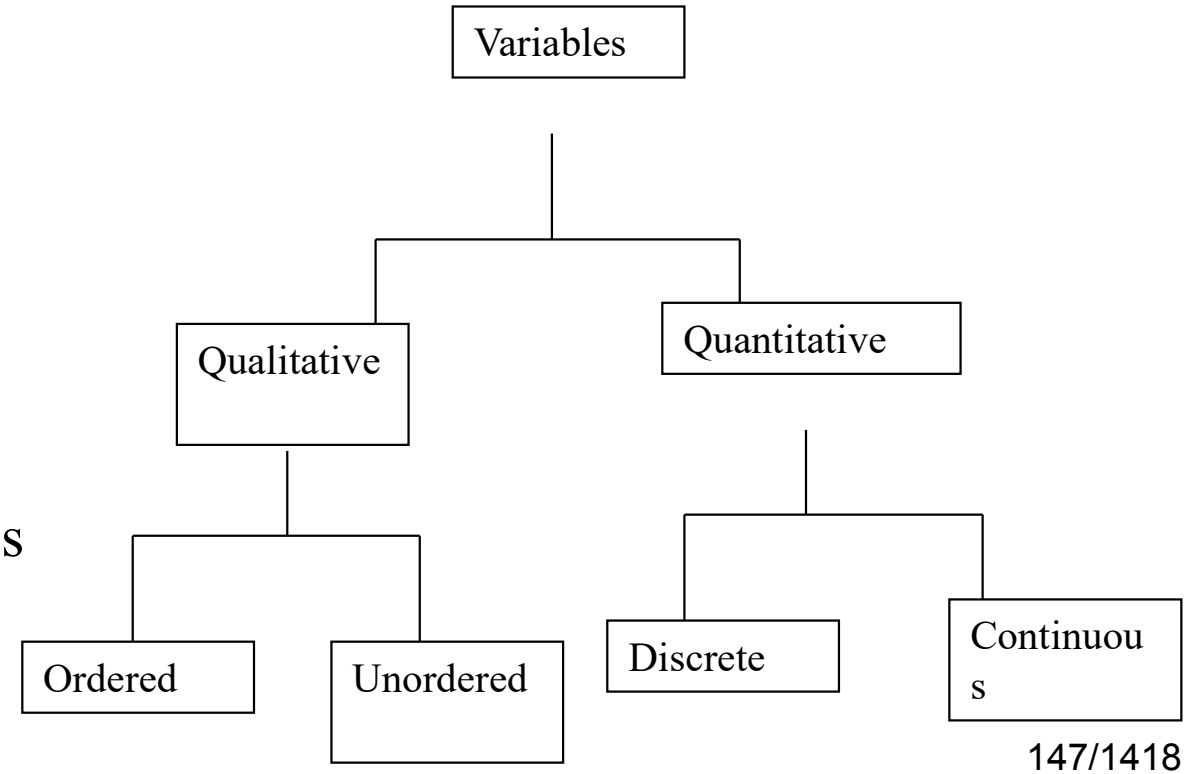
- Record
  - Relational records
  - Data matrix, e.g., numerical matrix, crosstabs
  - Document data: text documents: term-frequency vector
  - Transaction data
- Graph and network
  - World Wide Web
  - Social or information networks
  - Molecular Structures
- Ordered
  - Video data: sequence of images
  - Temporal data: time-series
  - Sequential Data: transaction sequences
  - Genetic sequence data
- Spatial, image and multimedia:
  - Spatial data: maps
  - Image data:
  - Video data:

	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

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

Attributes

- **Attribute (or dimensions, features, variables):**
  - a data field, representing a characteristic or feature of a data object.
    - *E.g., customer _ID, name, address*
- **A measurement scale:**
  - Associates a numerical or symbolic value with an attribute of an object.
- Type of attribute: Type of measurement of scale
- Attribute Types:
  - Qualitative/categorical attributes
    - Nominal attributes
    - Binary attributes
    - Ordinal attributes
  - Quantitative attributes or numeric attributes
    - Interval-scaled attributes
    - Ratio-scaled attributes



Qualitative Attributes

- **Nominal attributes :** categories, states, or "names of things"
  - *Hair_color = {auburn, black, blond, brown, grey, red, white}*
  - marital status, occupation, ID numbers, zip codes
  - **No order**
  - **Mean/median can not be calculated. Mode is the option.**
- **Binary attributes:** Nominal attribute with only 2 states (0 and 1)
  - Symmetric binary attribute: both outcomes are equally important
    - e.g., gender
  - Asymmetric binary attribute: outcomes not equally important.
    - e.g., medical test (positive vs. negative)
    - Convention: assign 1 to most important outcome (e.g., HIV positive)

- **Ordinal**
  - Values have a meaningful order (ranking), but the magnitude between successive values is unknown.
  - *Example*
    - *Size = {small, medium, large},* grades, army rankings
    - Customer satisfaction: 0: very satisfied, 1: some dissatisfied, 2: neutral, 3: satisfied, and 4: very satisfied
    - Central tendency: Mode or median is the measure
  - New value=f(old value)

Quantitative Numeric Attribute Types

- Quantity (integer or real-valued)
- **Interval**
  - Measured on a scale of **equal-sized units**
  - Values have order
    - E.g., *temperature in C˚or F˚, calendar dates*
  - No true zero-point. New value= a*old value+b
- **Ratio**
  - 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., the *temperature in Kelvin, length, counts, monetary quantities*
    - *New value= a*old-value*

Discrete vs. Continuous Attributes

- **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

Presentation Outline

- Data mining functionalities
- Research Issues in Data mining
- Sources of data
- Data Objects and Attribute Types
- **Basic Statistical Descriptions of Data**
- Data Visualization
- Summary

Basic Statistical Descriptions of Data

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

Basic Statistical Descriptions of Data…

- Measures of Central Tendency (middle or center of the distribution)
  - Mean, Median, Mode, Geometric Mean, Harmonic Mean
- Variability measures or measuring the dispersion. How the data is spread out?
  - Range, Quartile Deviation, Mean Deviation, Standard Deviation, Coefficient of Variation
- Skewness
- Constructing a boxplot

Measures of Central Tendency

- A group of data is represented with a single number
- It brings very important information from it

Arithmetic Mean

- Arithmetic mean: Sum of observations divided by its number.

$$\overline{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

  Note: *n* is sample size and *N* is population size.

  - Weighted arithmetic mean:

$$\mu = \frac{\sum x}{N}$$

  - Trimmed mean: chopping extreme values

- It should be computable.
  - Mean from categorical data is not calculable

$$\overline{x} = \frac{\sum_{i=1}^{n} w_i x_i}{\sum_{i=1}^{n} w_i}$$

- Advantages: Can carry out algebraic manipulations.
- Demerits:
  - Gives more weight to extreme items (outliers).

Median

- Median: Middle value if odd number of values, or average of the middle two values otherwise
- Merits
  - It can be calculated even if extreme classes are not defined.
- Property
  - Sum of absolute deviations from is least when it is taken from median.
- Demerits
  - Not based on all observations
  - Not widely used in practice

Mode

- Mode: It is the value which occurs most frequently.
- Merits
  - It can be easily located
  - It can be calculated when extreme classes are not defined
  - It is used widely in the business.
  - It can be calculated easily except if maximum frequency occurs more than once.
- Demerits
  - Not based on the all observations
  - It is not having algebraic properties.
  - It is not stable
    - Different class intervals results into different mode value.
- The relationship between mean, median and mode
  - Mean - mode=3(mean-median)

Selecting among the mean, median and mode

- Common mistake: specifying the wrong index of central tendency
  - It is common to specify the mean
- If data is categorical, if yes use “mode”.
  - If no
    - If the total is of interest, use “mean”
    - If no
      - If the distribution is skewed, use “median”
      - Otherwise, use “mean”

- Type of micro processor → use mode
- Total CPU time → mean

- Skew: ratio of minimum and maximum is large, the data is skewed.

Summarizing the Variability

- It is a measure that can give the widespread or scattering of observations among themselves or from a center point.
  - Range, Quartile deviation, Mean deviation, and standard deviation (variance)
- Range:
  - The difference between the highest and lowest values in a series of observations.
  - Problem: it depends on two extreme values.
- Quartile deviation
  - Quartile deviation (Q.D)=$(Q_3-Q_1)/2$, where Q1 is first quartile, Q3 is third quartile. The first and third quartiles are called lower and upper quartiles, respectively.
  - First quartile: The value of the variate below which one-fourth of the values lie and above which three-fourths of the values lie, when the values are arranged in ascending order of magnitude.
  - Third quartile: The value of the variate below which three-fourths of the values lie and above which the remaining one-fourth of the values lie, when the values are arranged in ascending order of magnitude.
  - Merits:
    - The presence of abnormal values does not affect quartile deviation.

Mean deviation, Standard Deviation, Variance

- Mean Deviation: The mean deviation is the mean of the absolute values of the deviations taken from the average.

- Standard deviation: It is defined as the square root of the mean of the squares of the deviations taken from the arithmetic mean.

  - σ =square root($1/n \sum (x1 - X)^{2)}$ where X is arithmetic mean

- Variance: Square of standard deviation.

Coefficient of Variation

- CV is the percentage ratio of S.D to mean.
- CV= (SD/Mean)*100

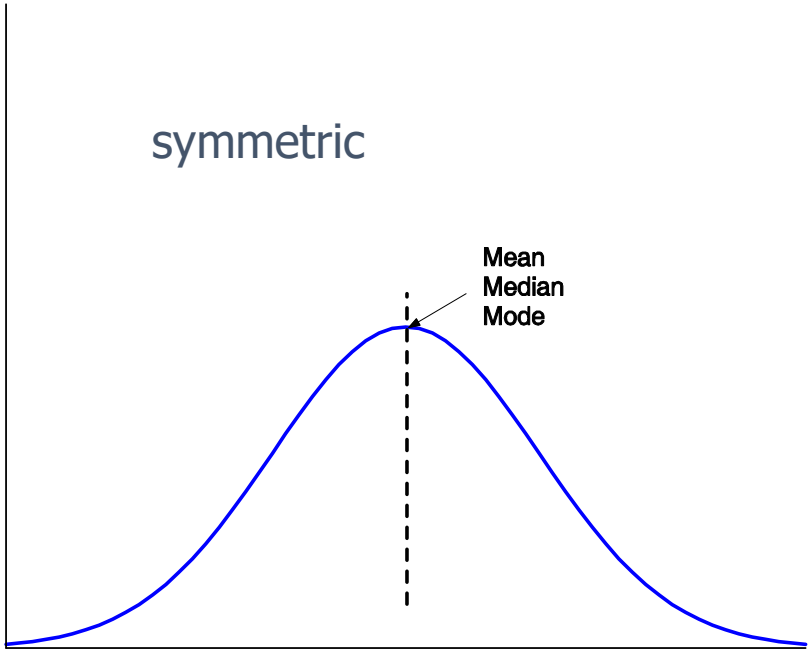
- For a player scores: if CV is high he/she is inconsistent,

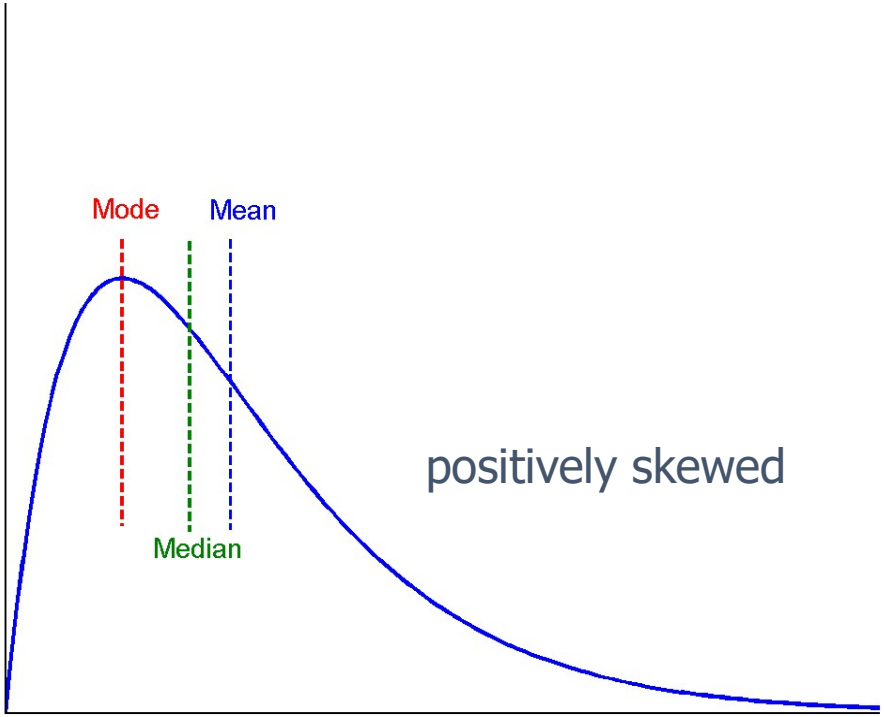
Statistical Population

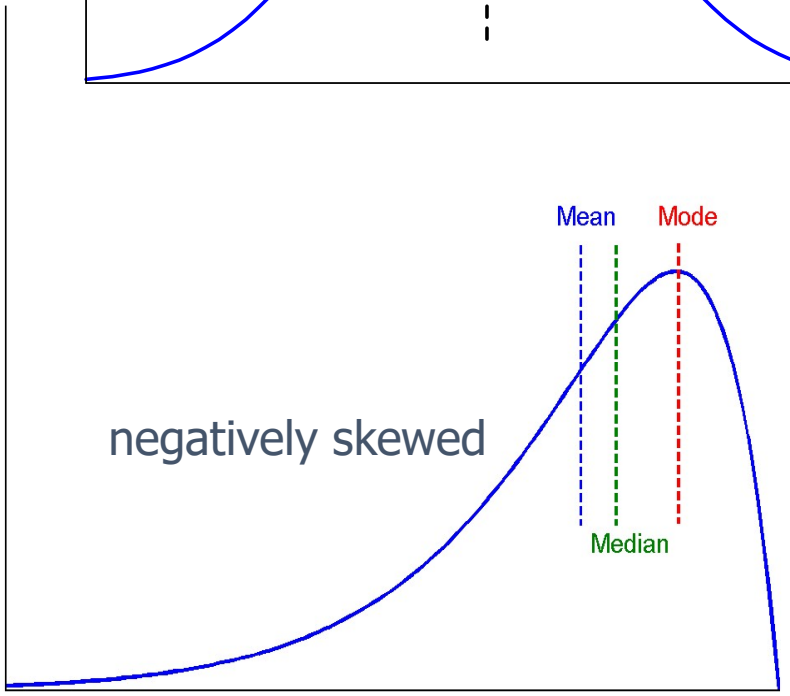
- Population: All the values/objects
- Sample: A part of population.
  - A sample should be a representative of population
  - Should be taken in a random manner.
  - Population is specified with parameters
  - Sample is specified with statistics
- Population error: Difference between the sample mean and population mean.

Symmetric vs. Skewed Data

- Even if two measures mean and standard deviations are same for the distributions, still the shape of two curves may differ.







Determining the Distribution of Data

- The simplest way is to plot the histogram of data
  - Requires dividing the range into a number of sub-ranges called cells or buckets.
- The count of observations that fall into each cell are determined.
- The counts are normalized to frequencies by dividing by the total number of observations. The cell frequencies are plotted as a column chart
- Key problem in determining the cell size.
  - Small cells lead to very few observations per cell and large variations in the number of observations.
  - Large cells result in fewer variations, but the details of observations are completely lost.
  - Guideline: if a cell has fewer than five observations, the cell size should be increased, or a variable cell histogram should be used.

Measures of Skewness

- Pearson's Coefficient of skewness
  - (Mean-mode)/SD
- Quartile coefficient of skewness
  - ((Q3-Q2)-(Q2-Q1))/(Q3-Q1)

Selecting Index of Dispersion

- If the variable is bounded, use “range”.
- If there are no natural bounds, and the distribution is symmetric use either SD, variance or CV.
- If the distribution is non-symmetric, percentiles are best choices.

Box Plot
or The 5 Number Summary

- A **box plot** or **boxplot** (also known as a **box-and-whisker diagram** or **plot**) is a convenient way of graphically depicting groups of numerical data through their five-number summaries:
  - the smallest observation (sample minimum),
  - lower quartile (Q1),
  - median (Q2),
  - upper quartile (Q3), and
  - largest observation (sample maximum).
  - A boxplot may also indicate which observations, if any, might be considered outliers.
    - usually, a value higher/lower than 1.5 x IQR

Constructing a box and whisker plot

- Step 1 - Find the median.
- Remember, the median is the middle value in a data set.

18, 27, 34, 52, 54, 59, 61, 68, 78, 82, 85, 87, 91, 93, 100

68 is the median of this data set.

Constructing a box and whisker plot

- Step 2 – Find the lower quartile.
- The lower quartile is the median of the data set to the left of 68.

(18, 27, 34, 52, 54, 59, 61,) 68, 78, 82, 85, 87, 91, 93, 100

52 is the lower quartile

Constructing a box and whisker plot

- Step 3 – Find the upper quartile.
- The upper quartile is the median of the data set to the right of 68.

18, 27, 34, 52, 54, 59, 61, 68, (78, 82, 85, 87, 91, 93, 100)

87 is the upper quartile

Constructing a box and whisker plot

- Step 4 – Find the maximum and minimum values in the set.
- The maximum is the greatest value in the data set.
- The minimum is the least value in the data set.

18, 27, 34, 52, 54, 59, 61, 68, 78, 82, 85, 87, 91, 93, 100

18 is the minimum and 100 is the maximum.

Constructing a box and whisker plot

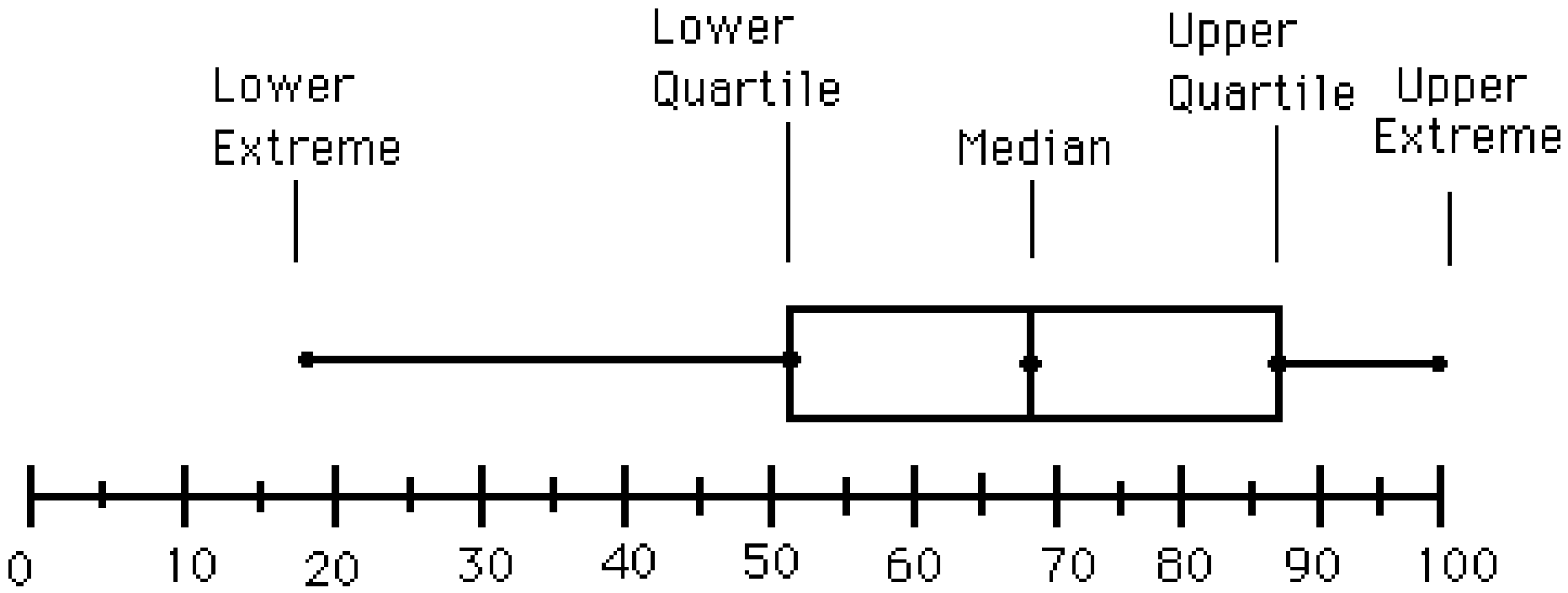
- Step 5 – Find the inter-quartile range (IQR).
- The inter-quartile (IQR) range is the difference between the upper and lower quartiles.
  - Upper Quartile = 87
  - Lower Quartile = 52
  - 87 – 52 = 35
  - 35 = IQR

The 5 Number Summary

- Organize the 5 number summary
  - Median – 68
  - Lower Quartile – 52
  - Upper Quartile – 87
  - Max – 100
  - Min – 18

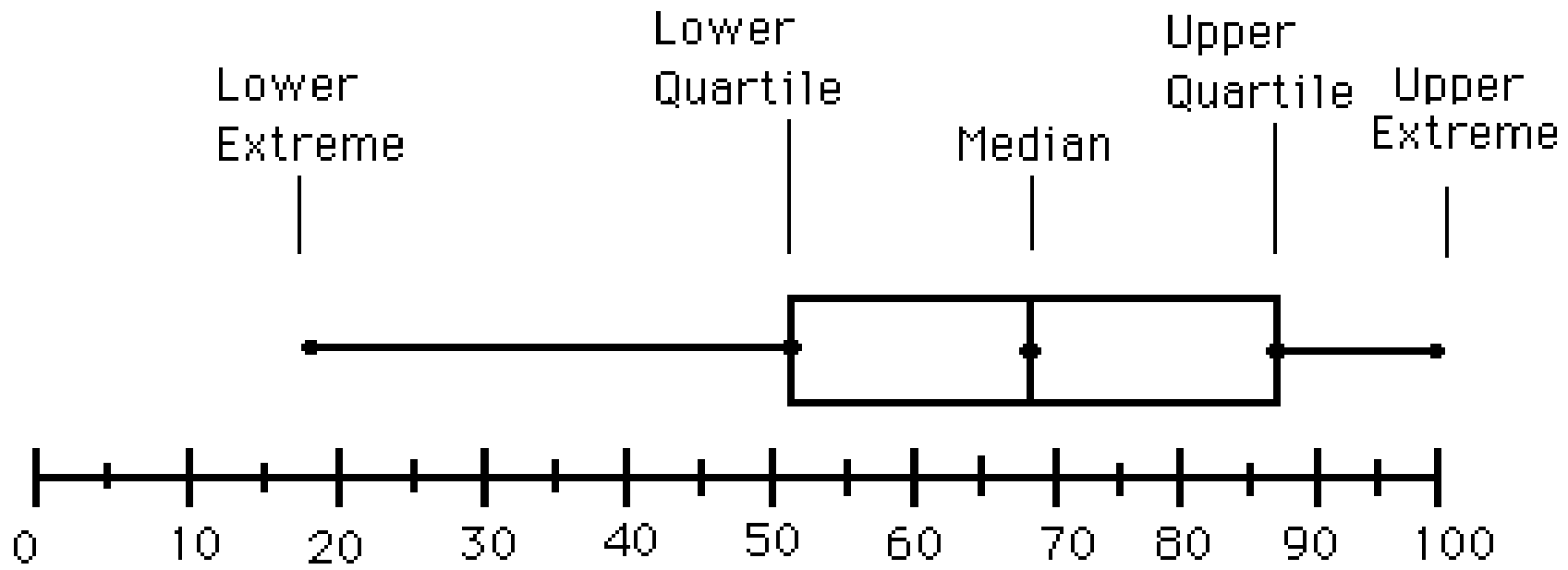
Graphing The Data

- Notice, the Box includes the lower quartile, median, and upper quartile.
- The Whiskers extend from the Box to the max and min.

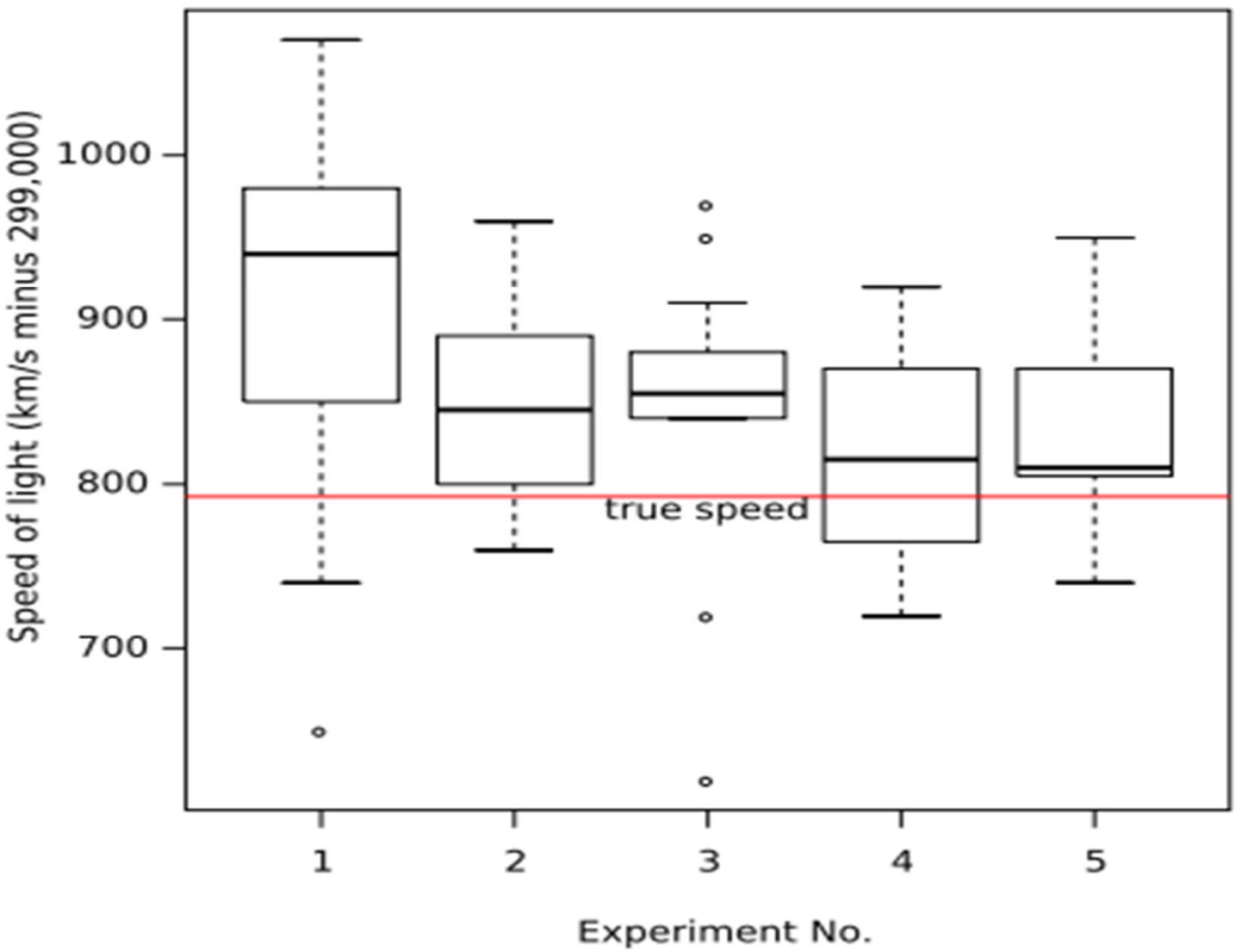


Analyzing The Graph

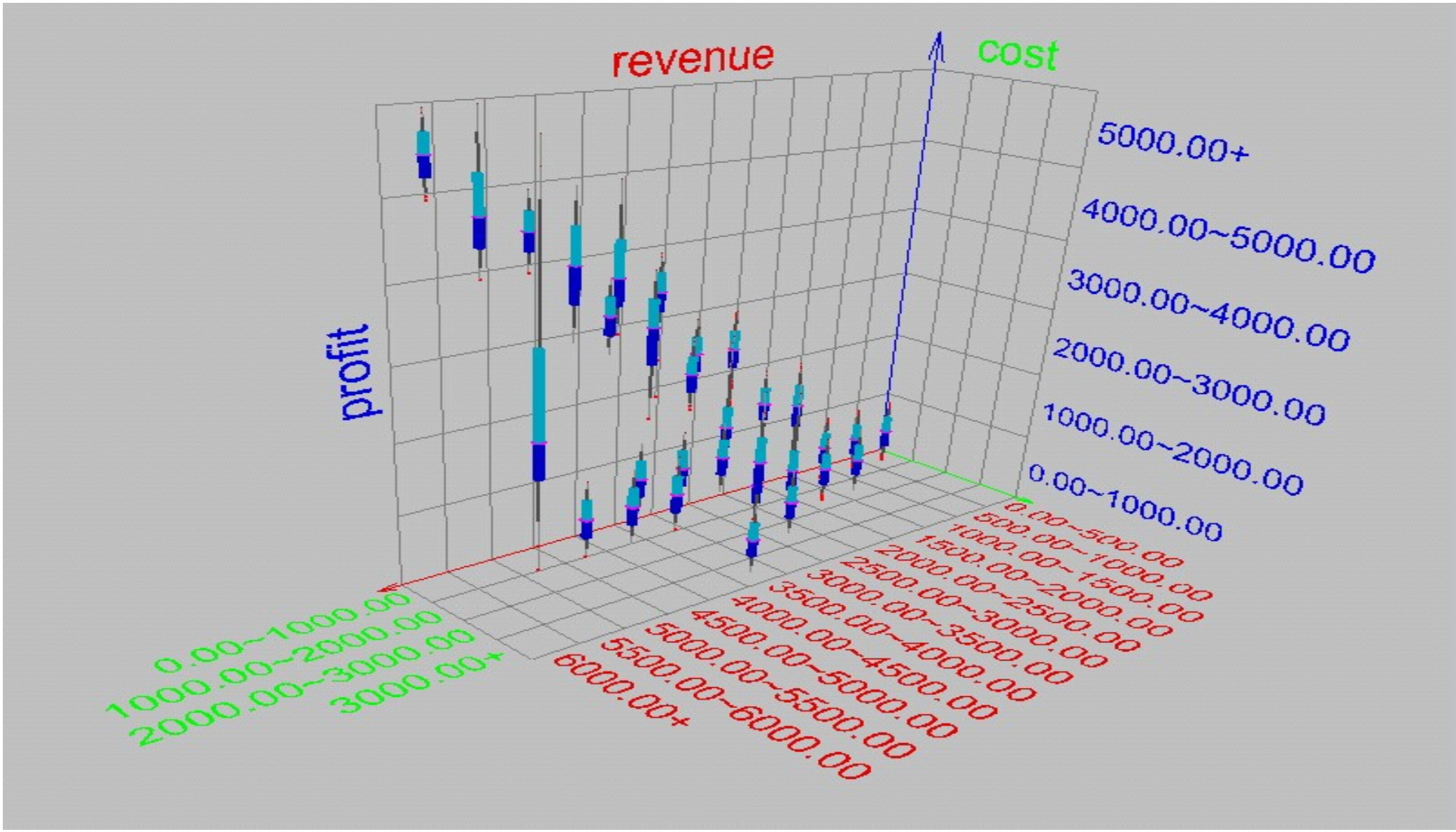
- The data values found inside the box represent the middle half (50%) of the data.
- The line segment inside the box represents the median



Example Box Plot



Visualization of Data Dispersion: 3-D Boxplots

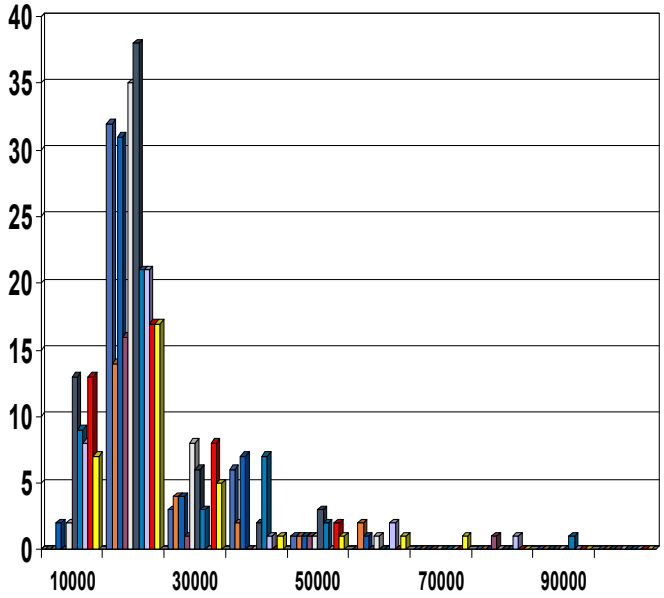


Graphic Displays of Basic Statistical Descriptions

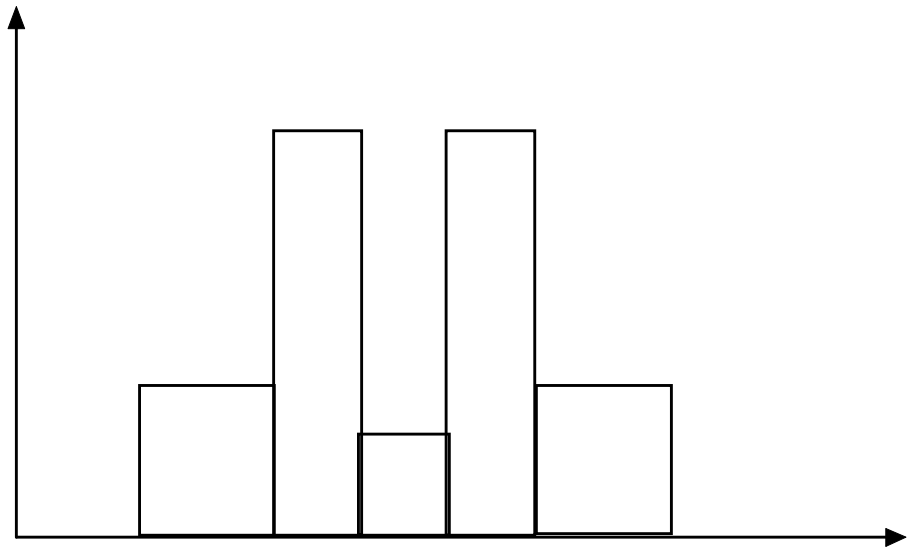
- **Boxplot**: graphic display of five-number summary
- **Histogram**: x-axis are values, y-axis repres. frequencies
- **Quantile plot**: each value x_i is paired with f_i indicating that approximately 100 f_i % of data are $\leq x_i$
- **Quantile-quantile (q-q) plot**: graphs the quantiles of one univariant distribution against the corresponding quantiles of another
- **Scatter plot**: each pair of values is a pair of coordinates and plotted as points in the plane

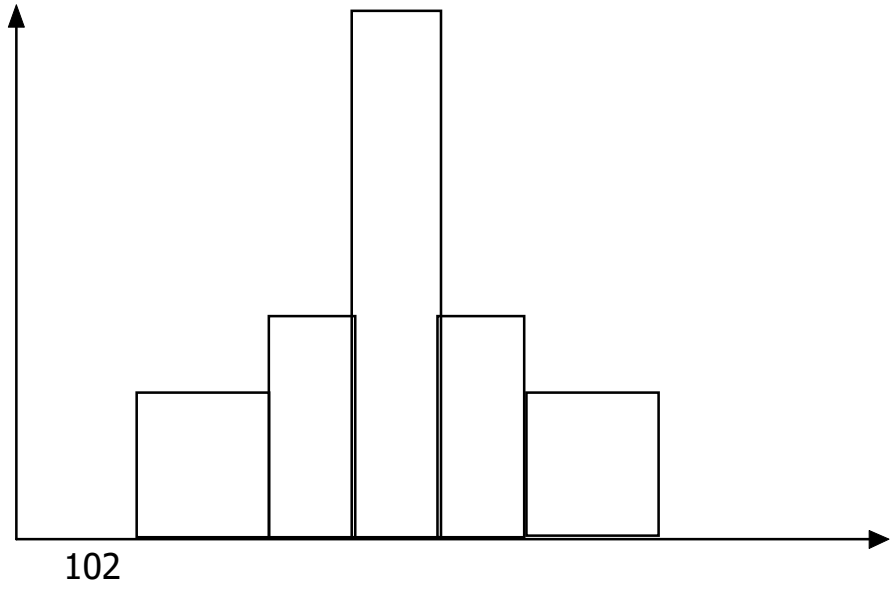
Histogram Analysis

- Histogram: Graph display of tabulated frequencies, shown as bars
- It shows what proportion of cases fall into each of several categories
- Differs from a bar chart in that it is the *area* of the bar that denotes the value, not the height as in bar charts, a crucial distinction when the categories are not of uniform width
- The categories are usually specified as non-overlapping intervals of some variable. The categories (bars) must be adjacent



Histograms Often Tell More than Boxplots

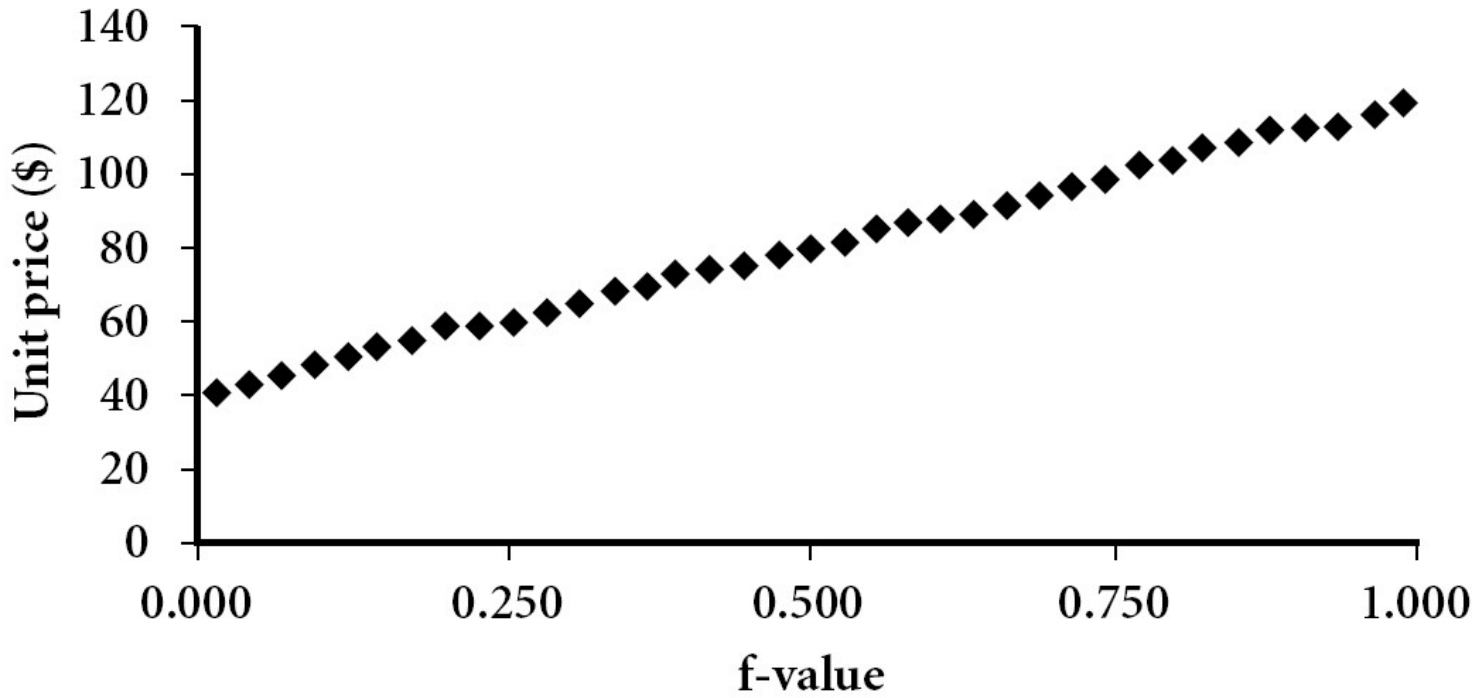




- The two histograms shown in the left may have the same boxplot representation
  - The same values for: min, Q1, median, Q3, max
- But they have rather different data distributions

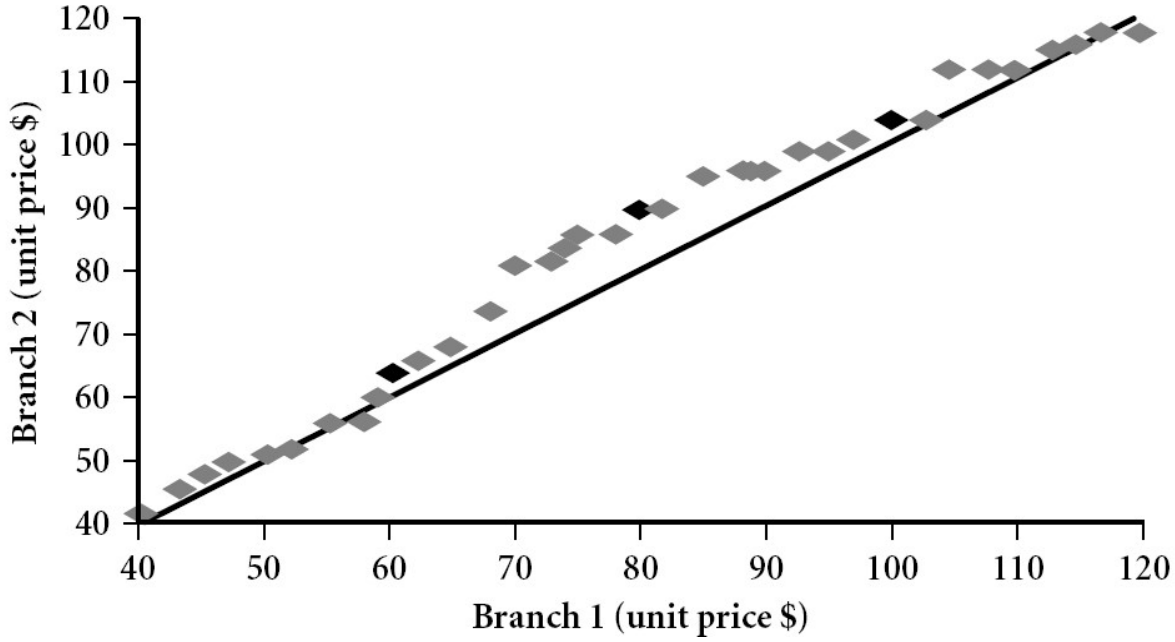
Quantile Plot

- Displays all of the data (allowing the user to assess both the overall behavior and unusual occurrences)
- Plots quantile information
  - For a data x_i data sorted in increasing order, f_i indicates that approximately 100 f_i% of the data are below or equal to the value x_i



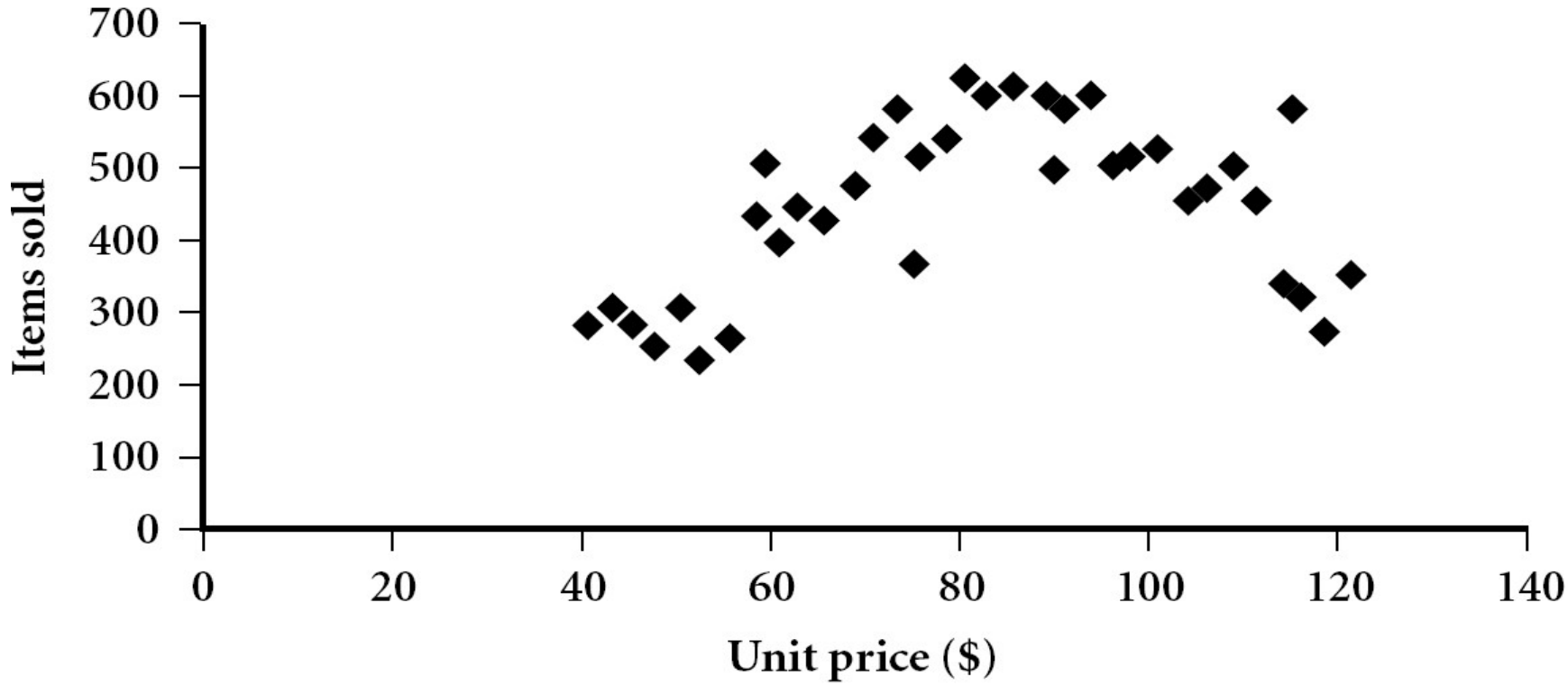
Quantile-Quantile (Q-Q) Plot

- Graphs the quantiles of one univariate distribution against the corresponding quantiles of another
- View: Is there is a shift in going from one distribution to another?
- Example shows unit price of items sold at Branch 1 vs. Branch 2 for each quantile. Unit prices of items sold at Branch 1 tend to be lower than those at Branch 2.



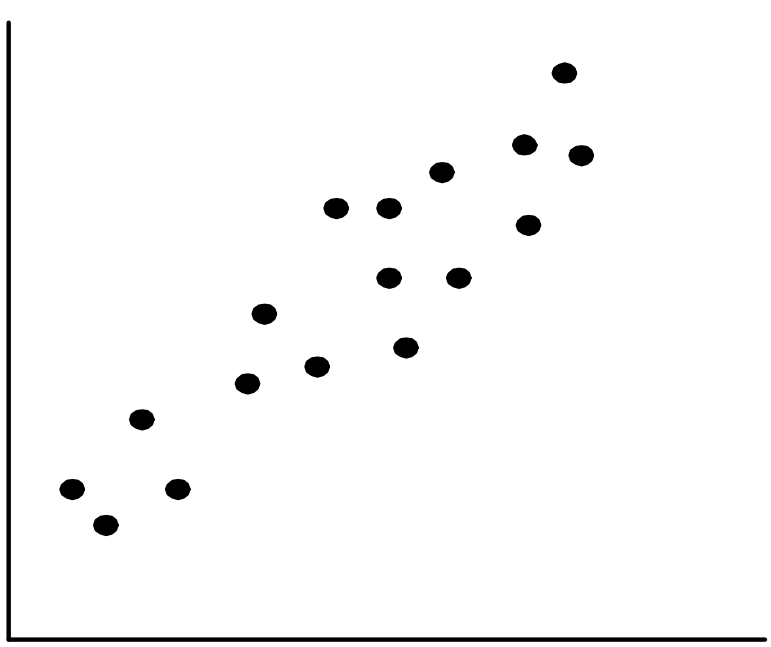
Scatter plot

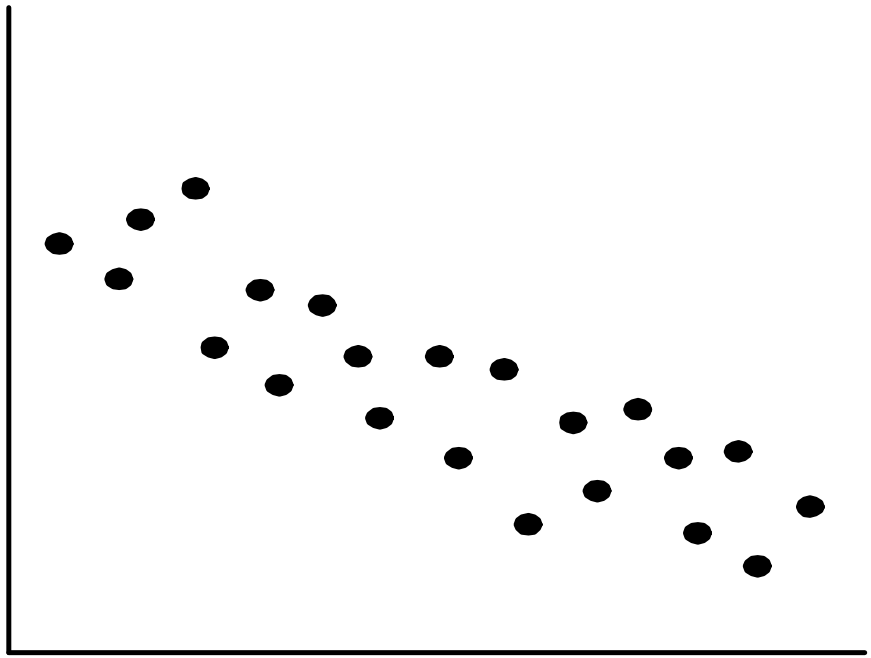
- Provides a first look at bivariate data to see clusters of points, outliers, etc
- Each pair of values is treated as a pair of coordinates and plotted as points in the plane



105

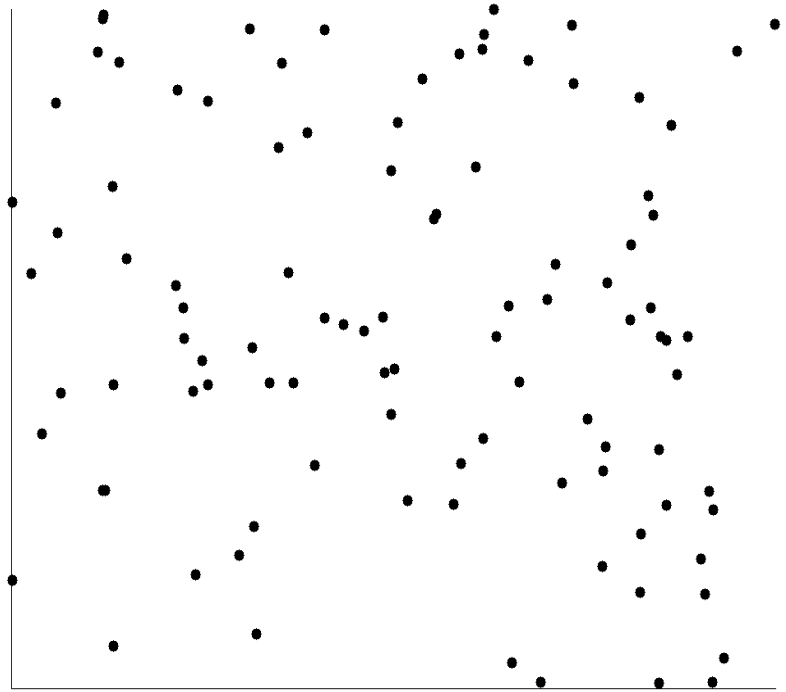
Positively and Negatively Correlated Data

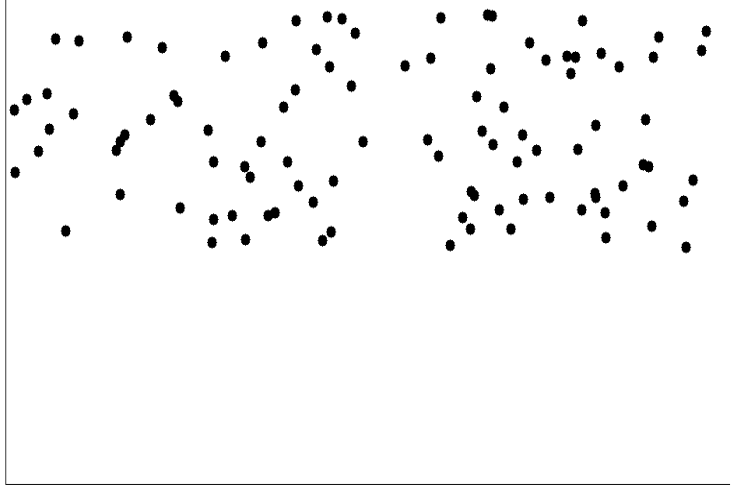


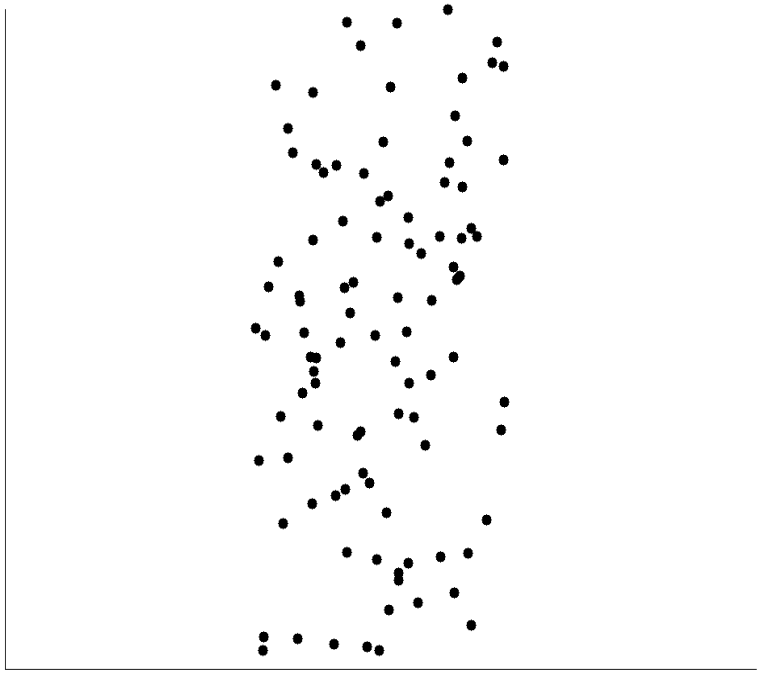


- The left half fragment is positively correlated
- The right half is negative correlated

Uncorrelated Data







Presentation Outline

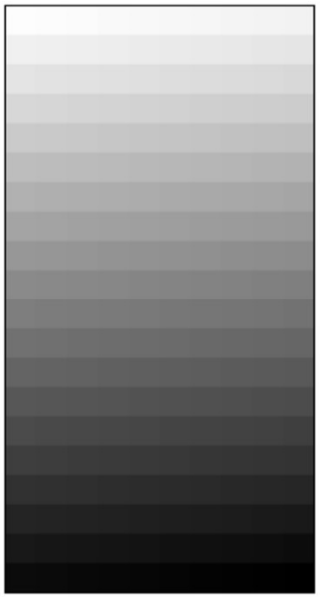
- Data mining functionalities
- Research Issues in Data mining
- Sources of data
- Data Objects and Attribute Types
- Basic Statistical Descriptions of Data
- **Data Visualization**
- Case study
- Summary

Data Visualization

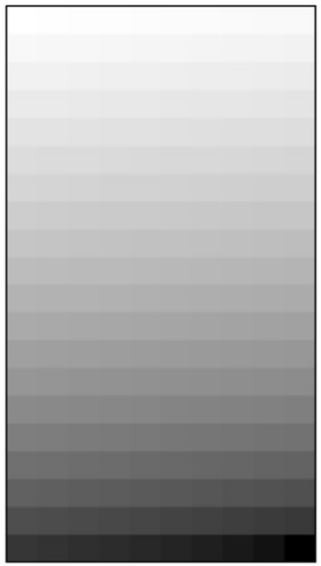
- Why data visualization?
  - Gain insight into an information space by mapping data onto graphical primitives
  - Provide qualitative overview of large data sets
  - Search for patterns, trends, structure, irregularities, relationships among data
  - Help find interesting regions and suitable parameters for further quantitative analysis
  - Provide a visual proof of computer representations derived
- Categorization of visualization methods:
  - Pixel-oriented visualization techniques
  - Geometric projection visualization techniques
  - Icon-based visualization techniques
  - Hierarchical visualization techniques
  - Visualizing complex data and relations

Pixel-Oriented Visualization Techniques

- For a data set of m dimensions, create m windows on the screen, one for each dimension
- The m dimension values of a record are mapped to m pixels at the corresponding positions in the windows
- The colors of the pixels reflect the corresponding values
- Example: Income ascending order of customer information table



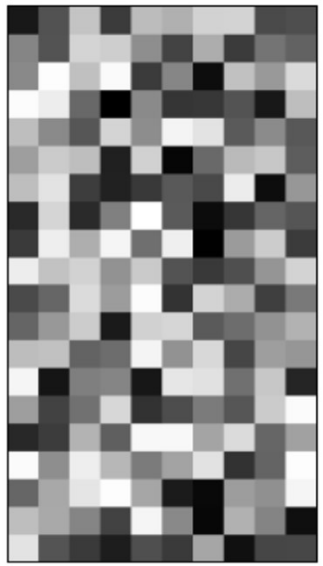
(a) Income



(b) Credit Limit



(c) transaction volume



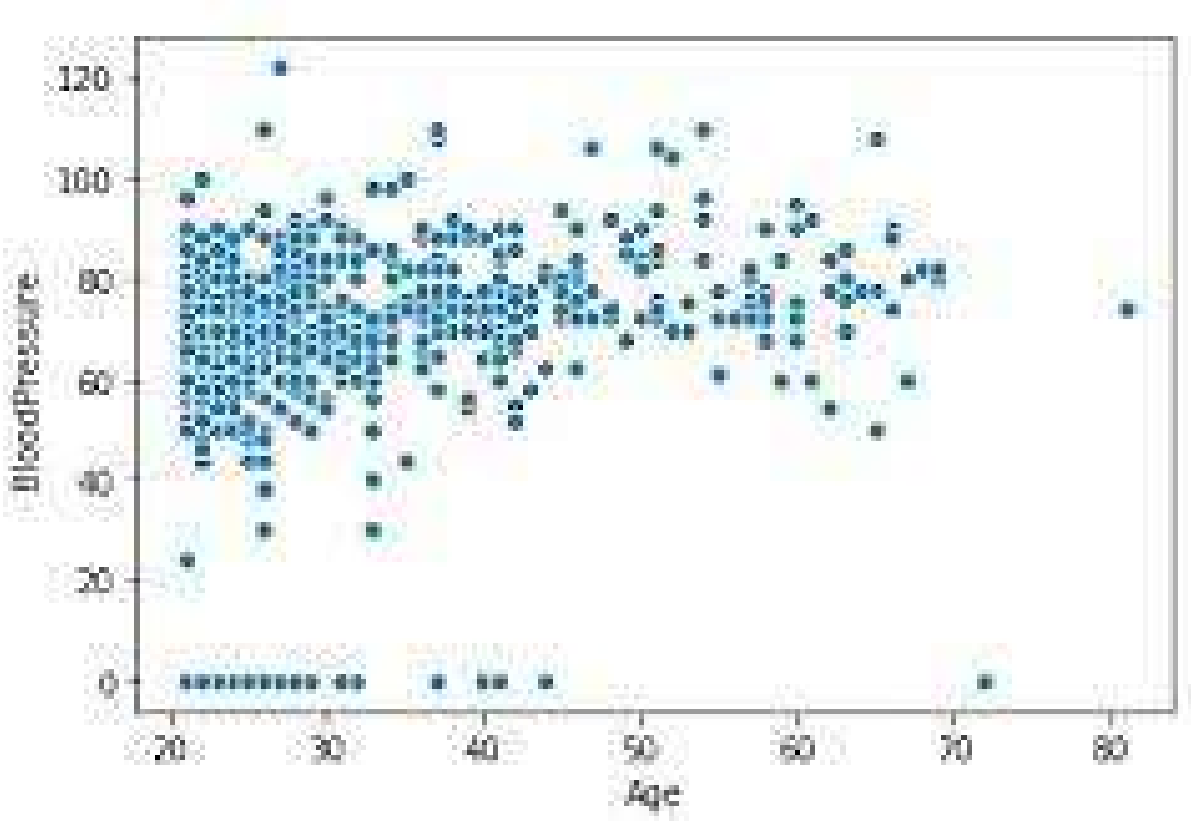
(d) age

Geometric Projection Visualization Techniques

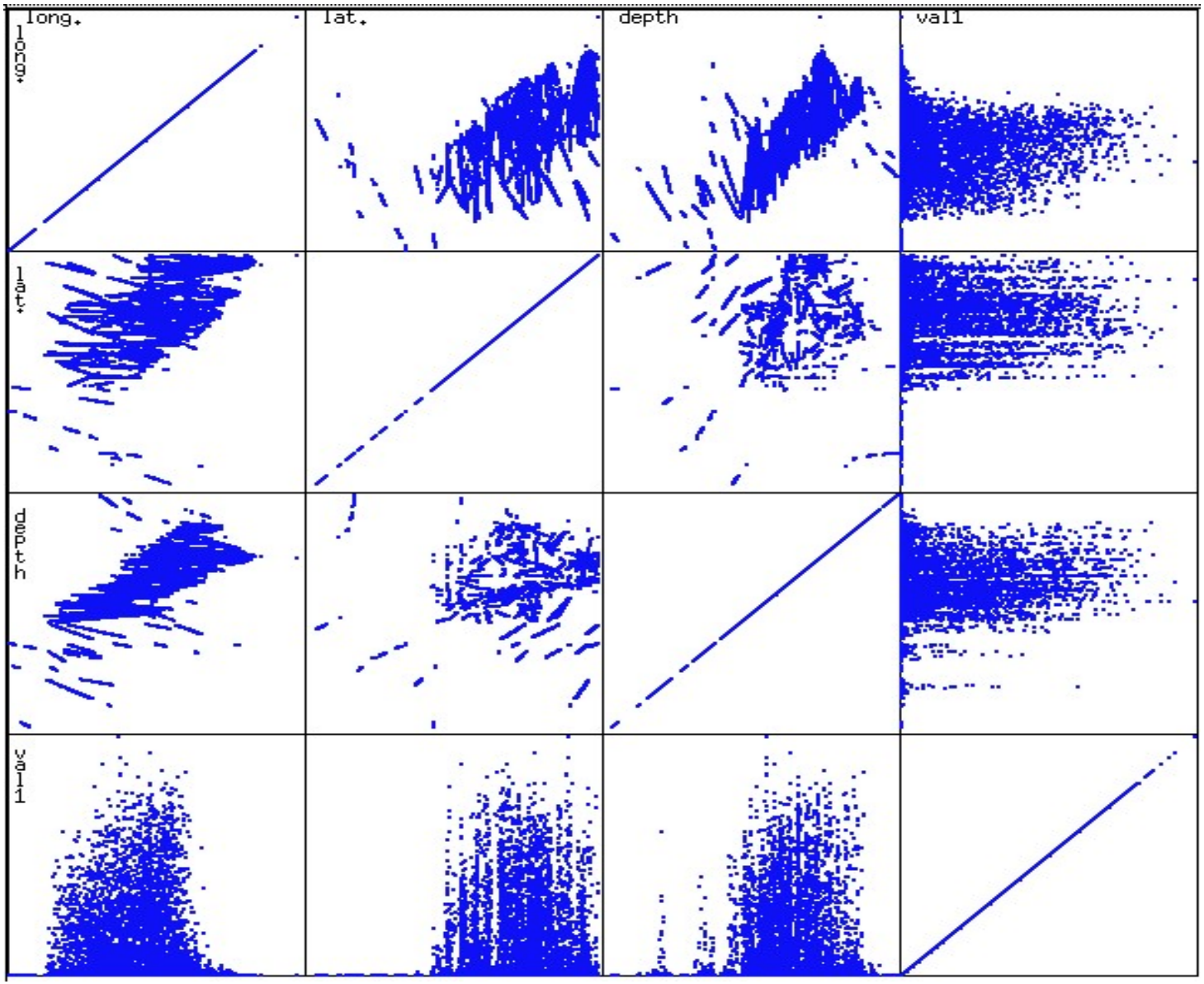
- Visualization of geometric transformations and projections of the data
- Methods
  - Scatterplot and scatterplot matrices
  - Landscapes
  - Parallel coordinates

Scatter plot

A scatter plot displays 2D data points using Cartesian coordinates.



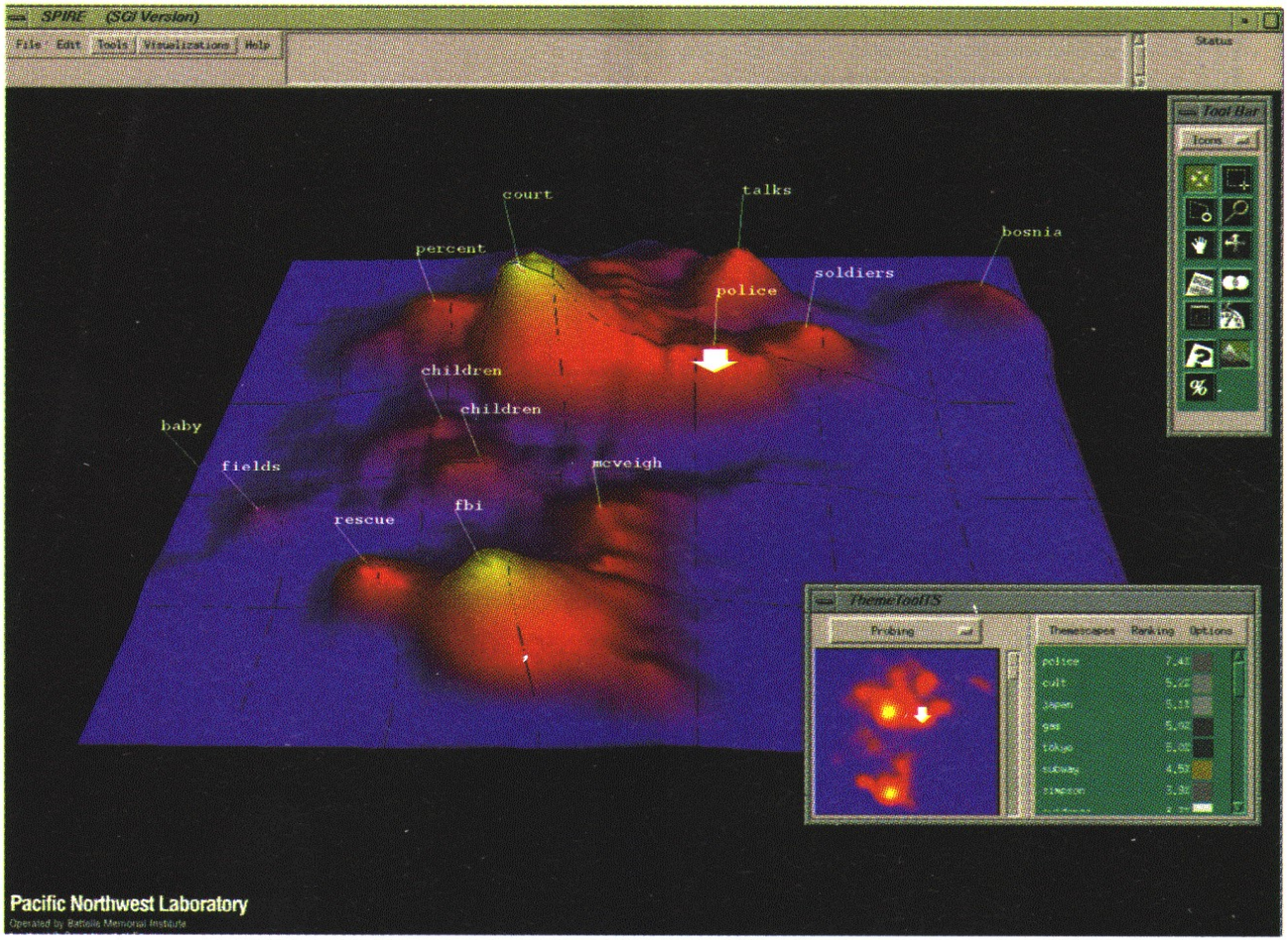
Scatterplot Matrices



Used by permission of M. Ward, Worcester Polytechnic Institute

Matrix of scatterplots (x-y-diagrams) of the k-dim. data [total of (k2/2-k) scatterplots]

Landscapes



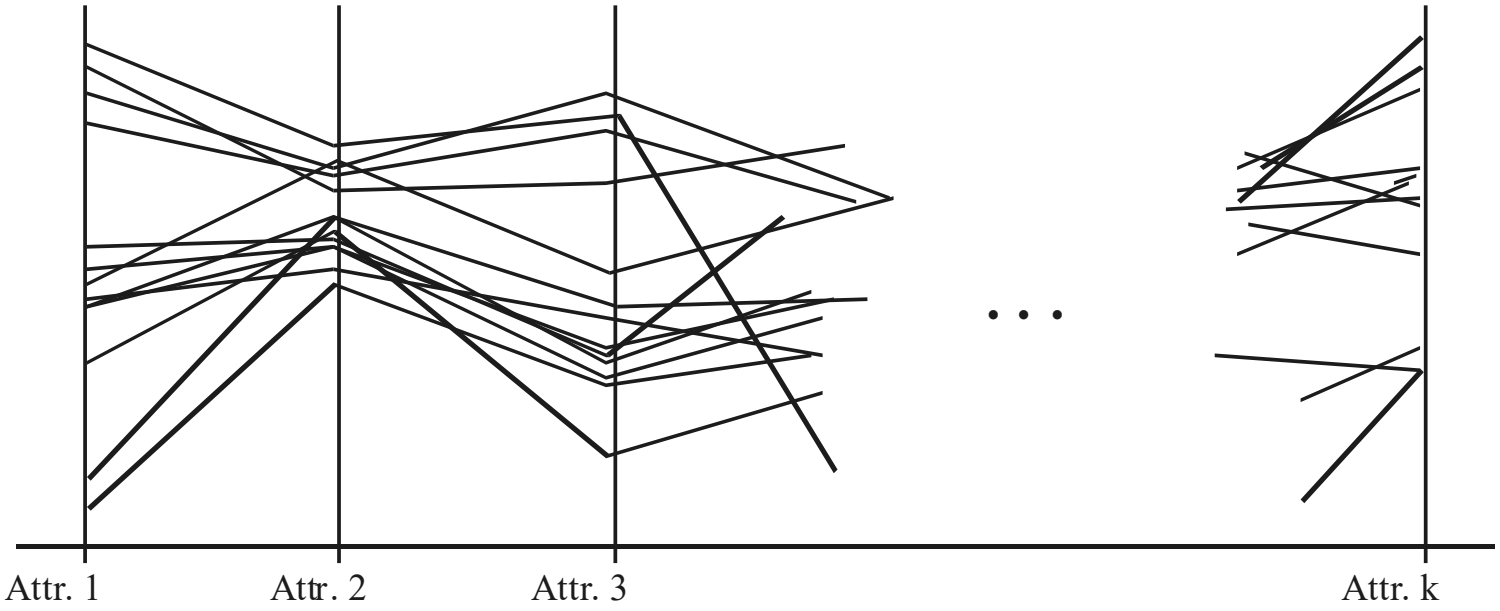
Used by permission of B. Wright, Visible Decisions Inc.

news articles visualized as a landscape

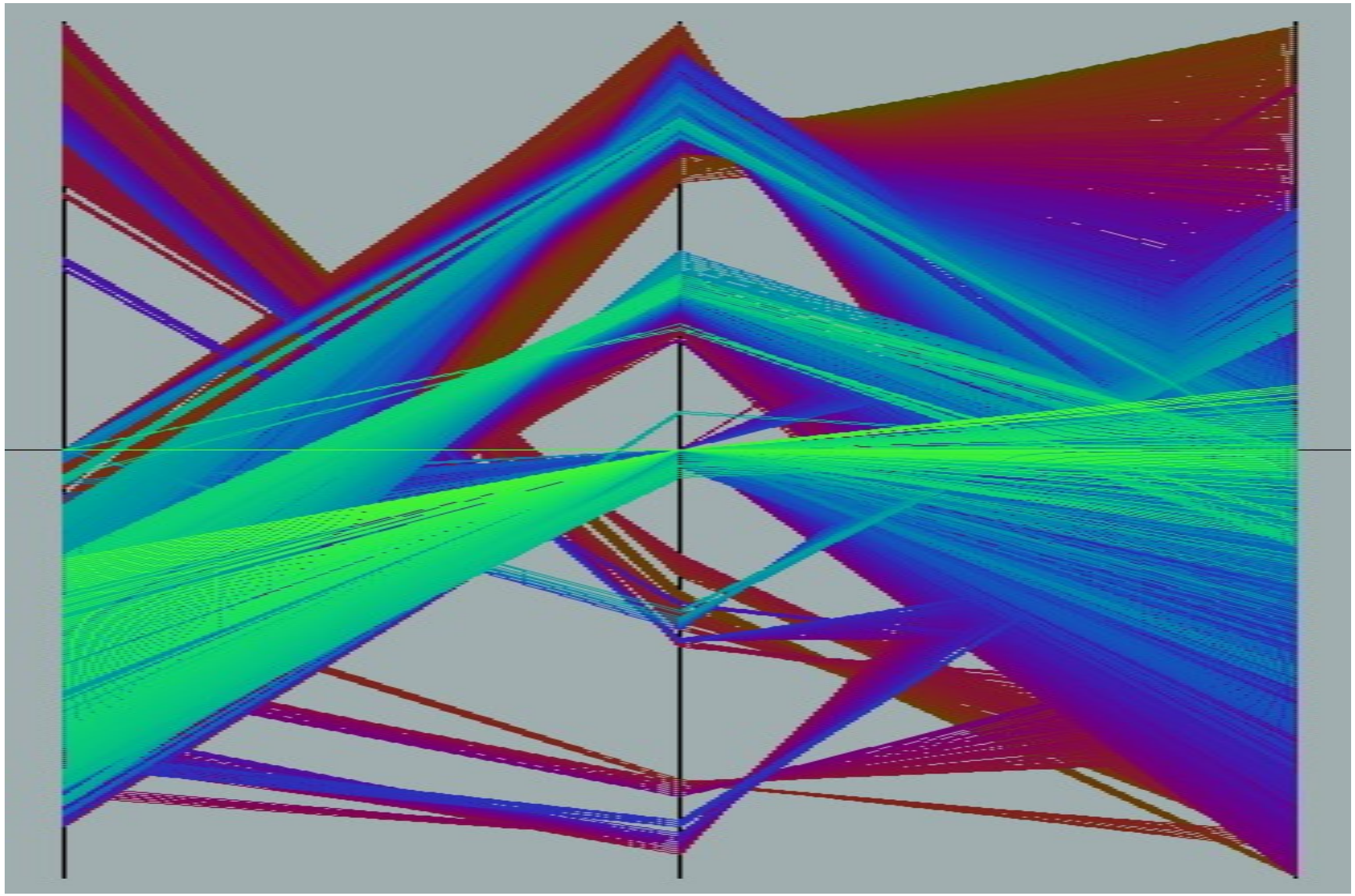
- Visualization of the data as perspective landscape
- The data needs to be transformed into a (possibly artificial) 2D spatial representation which preserves the characteristics of the data

Parallel Coordinates

- In equidistant axes which are parallel to one of the screen axes and correspond to the attributes
- The axes are scaled to the [minimum, maximum]: range of the corresponding attribute
- Every data item corresponds to a polygonal line which intersects each of the axes at the point which corresponds to the value for the attribute



Parallel Coordinates of a Data Set

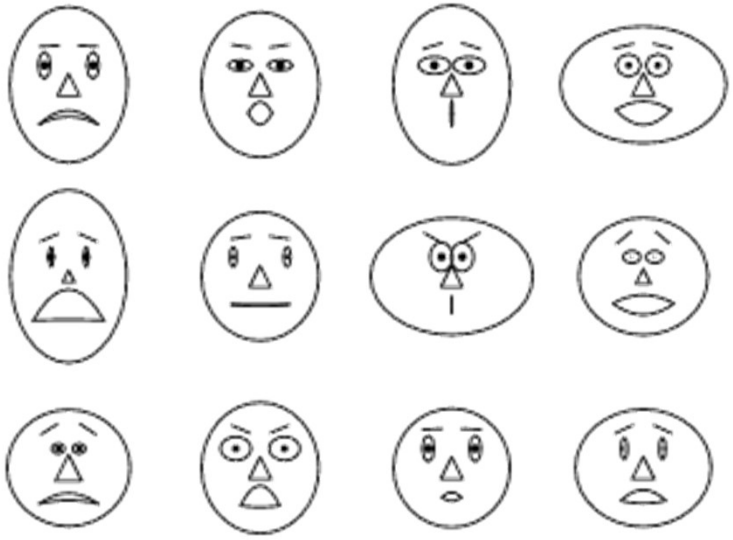


Icon-Based Visualization Techniques

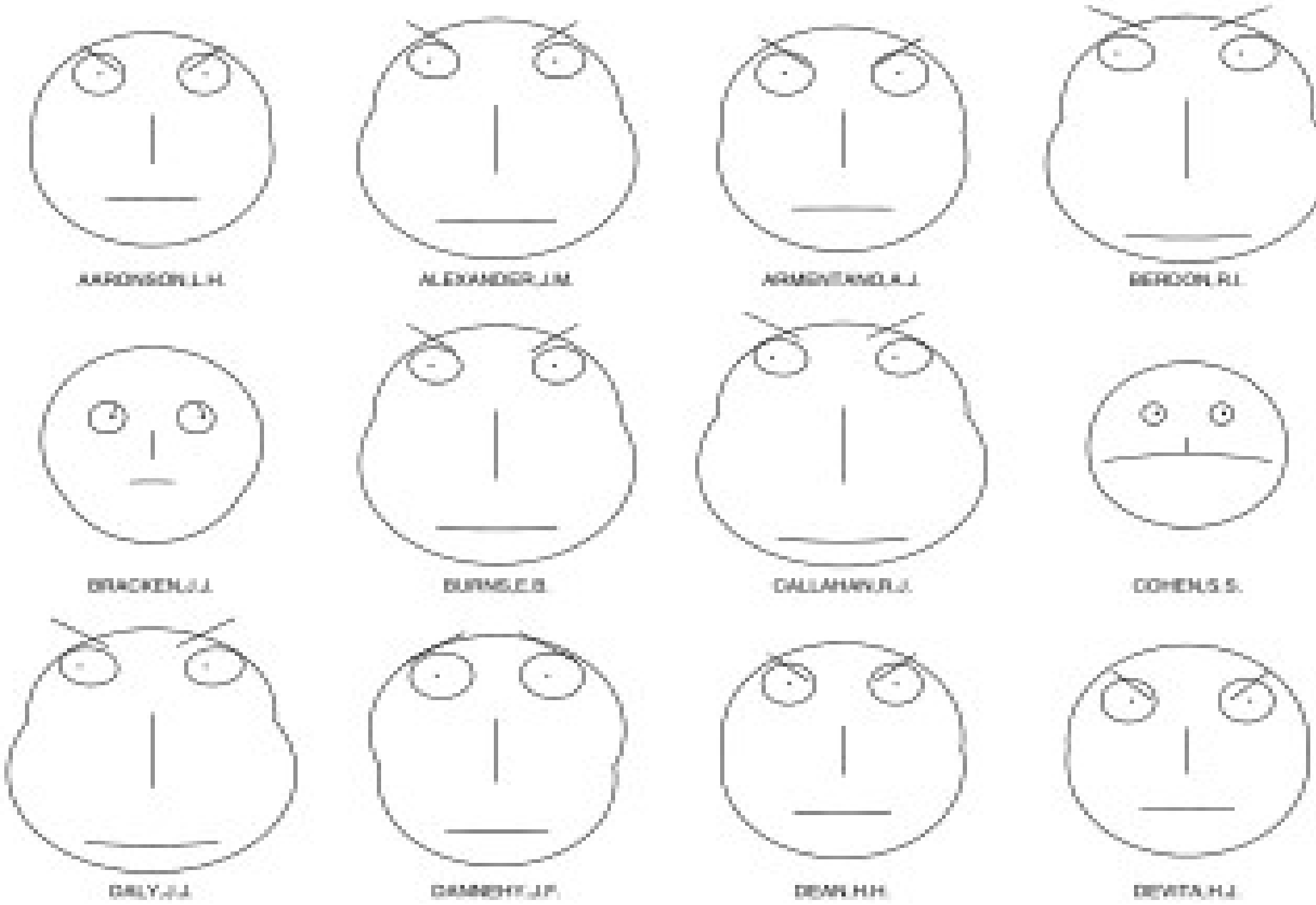
- Visualization of the data values as features of icons
- Typical visualization methods
  - Chernoff Faces
  - Stick Figures
- General techniques
  - Shape coding: Use shape to represent certain information encoding
  - Color icons: Use color icons to encode more information
  - Tile bars: Use small icons to represent the relevant feature vectors in document retrieval

Chernoff Faces

- A way to display variables on a two-dimensional surface, e.g., let x be eyebrow slant, y be eye size, z be nose length, etc.
- The figure shows faces produced using 10 characteristics--head eccentricity, eye size, eye spacing, eye eccentricity, pupil size, eyebrow slant, nose size, mouth shape, mouth size, and mouth opening): Each assigned one of 10 possible values, generated using *Mathematica* (S. Dickson)

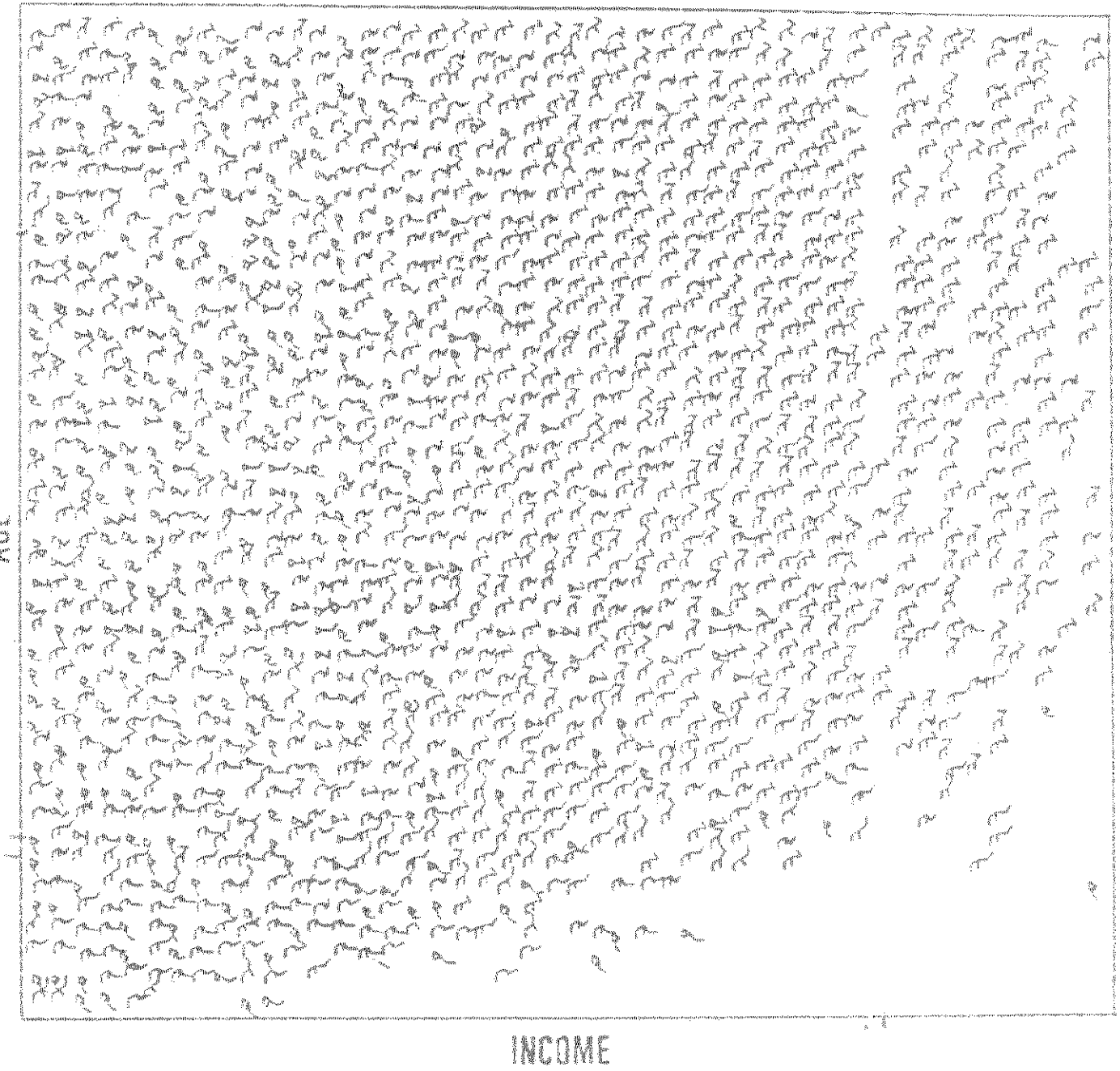


- REFERENCE: Gonick, L. and Smith, W. *The Cartoon Guide to Statistics.* New York: Harper Perennial, p. 212, 1993
- Weisstein, Eric W. "Chernoff Face." From *MathWorld*--A Wolfram Web Resource. mathworld.wolfram.com/ChernoffFace.html



This example shows Chernoff faces for lawyers' ratings of twelve judges

Stick Figure



used by permission of G. Grinstein, University of Massachusettes at Lowell

A census data figure showing age, income, gender, education, etc.

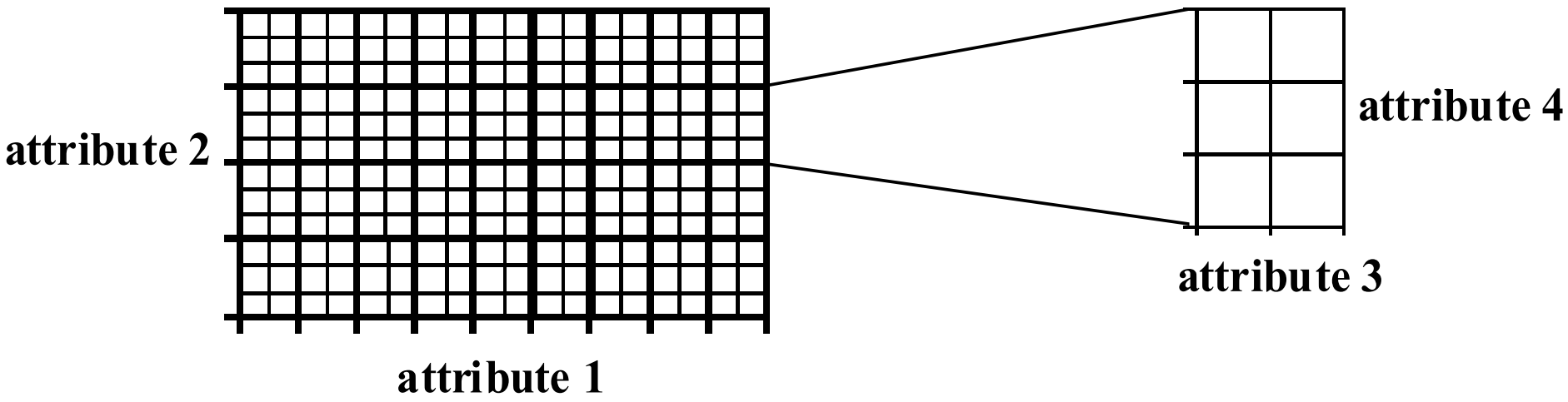
A 5-piece stick figure (1 body and 4 limbs w. different angle/length)

Two attributes mapped to axes, remaining attributes mapped to angle or length of limbs". Look at texture pattern

Hierarchical Visualization Techniques

- Visualization of the data using a hierarchical partitioning into subspaces
  - Dimensionality stacking
  - Tag cloud

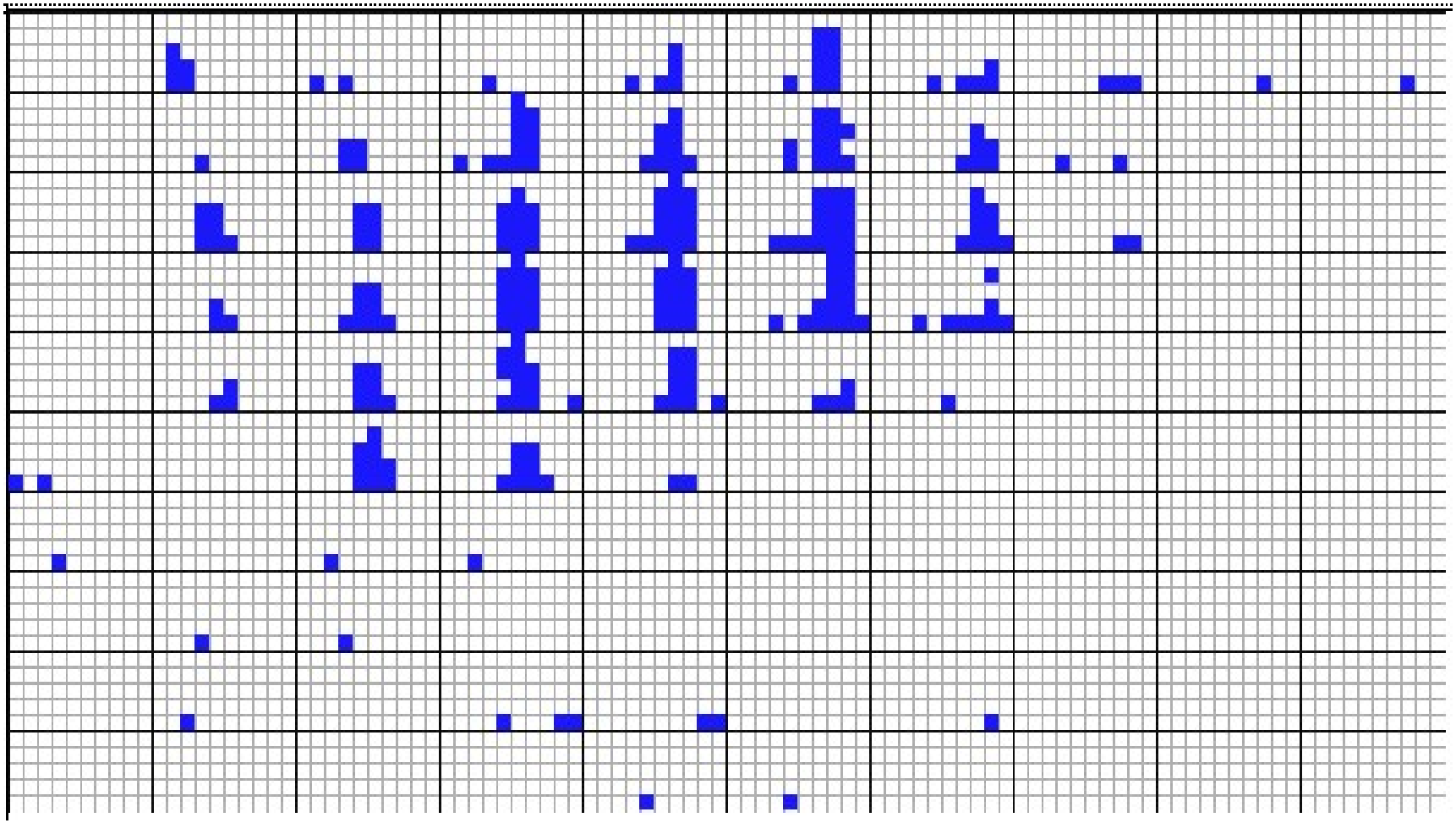
Dimensional Stacking



- Partitioning of the n-dimensional attribute space in 2-D subspaces, which are 'stacked' into each other
- Partitioning of the attribute value ranges into classes. The important attributes should be used on the outer levels.
- Adequate for data with ordinal attributes of low cardinality
- But, difficult to display more than nine dimensions
- Important to map dimensions appropriately

Dimensional Stacking

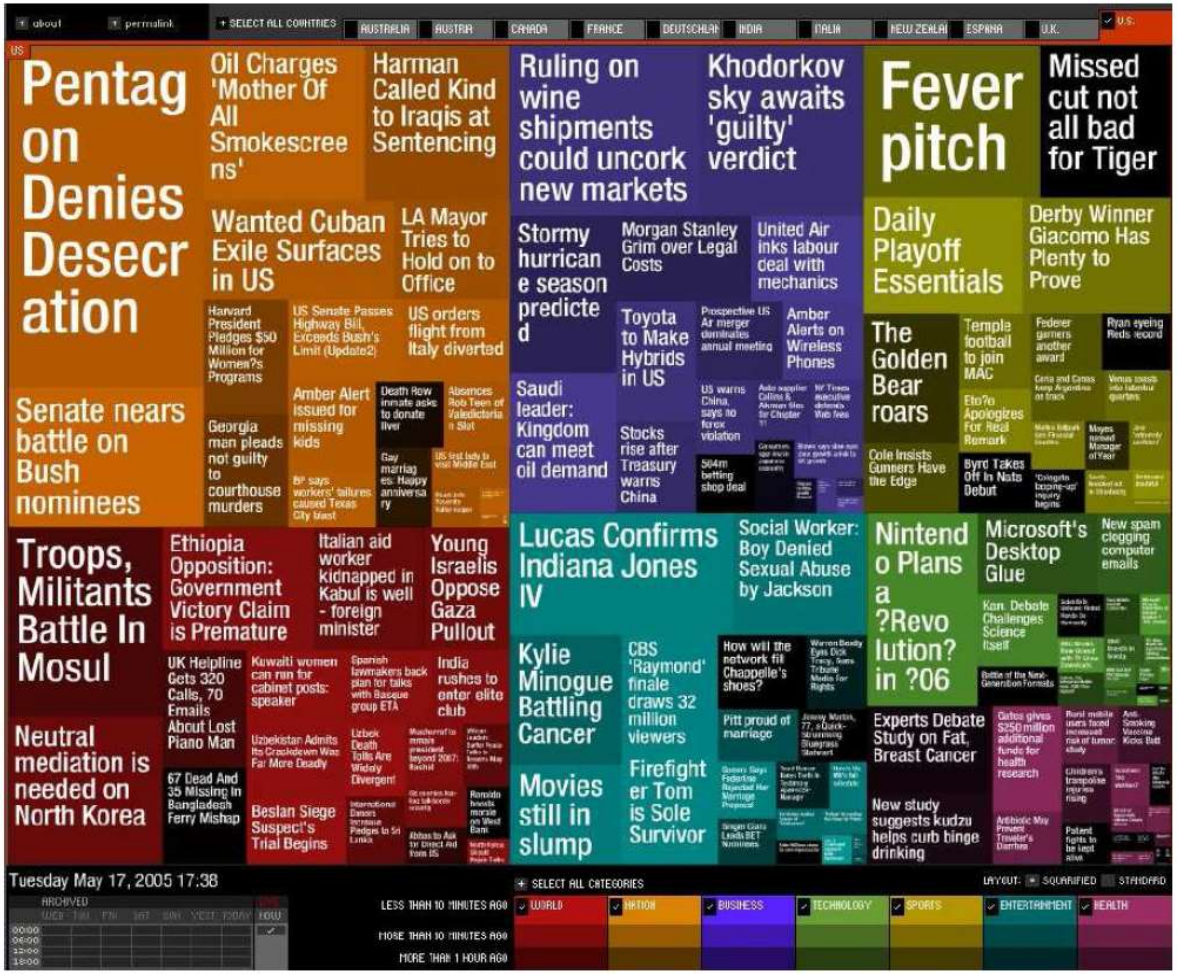
Used by permission of M. Ward, Worcester Polytechnic Institute



Visualization of oil mining data with longitude and latitude mapped to the outer x-, y-axes and ore grade and depth mapped to the inner x-, y-axes

Visualizing Complex Data and Relations

- Visualizing non-numerical data: text and social networks
- Tag cloud: visualizing user-generated tags
  - The importance of tag is represented by font size/color
- Besides text data, there are also methods to visualize relationships, such as visualizing social networks



Newsmap: Google News Stories in 2005

Presentation Outline

- Data mining functionalities
- Research Issues in Data mining
- Sources of data
- Data Objects and Attribute Types
- Basic Statistical Descriptions of Data
- Data Visualization
- **Case study**
- Summary

Case Study: Analysis of paper submissions to DASFAA-2022 conference

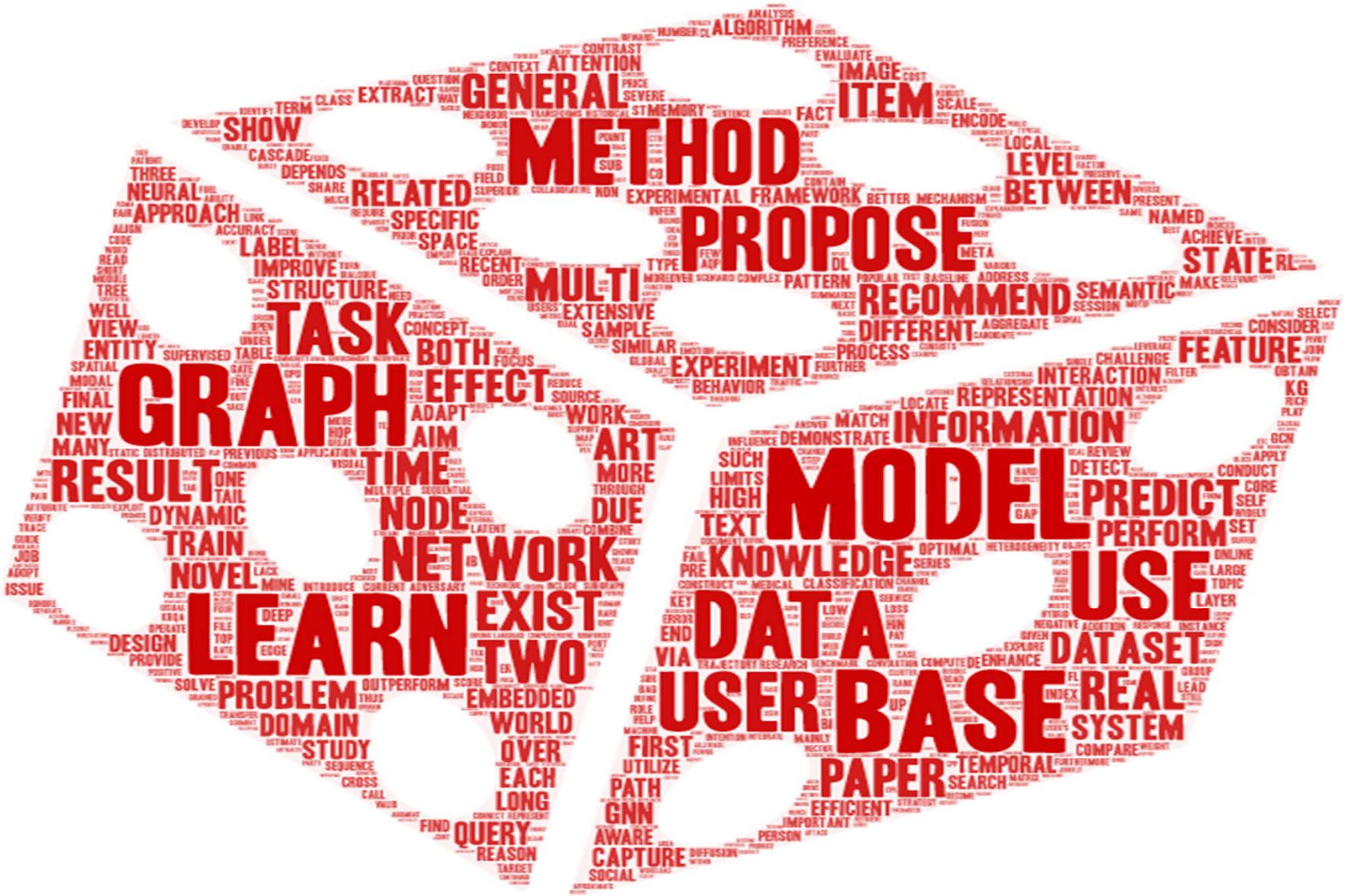


The 27th International Conference on Database Systems for Advanced Applications (DASFAA-2022), April 11-14, 2022, Hyderabad, India.

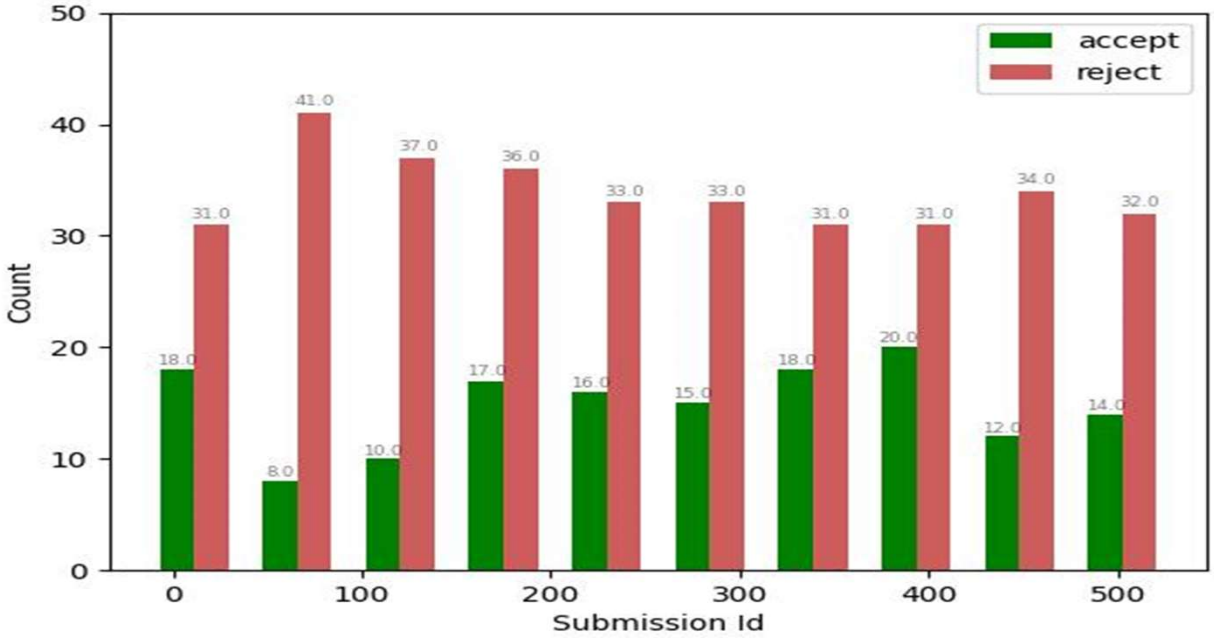


- Number of valid submissions (after desk-reject): 400
  - Yes, this is the number; no rounding off was done!
- Program committee
  - Reviewers: 205
  - Meta-Reviewers: 41
- Accept
  - Full: 72 (18%)
  - Short: 76 (19%)

Tag Cloud from Accepted Abstracts

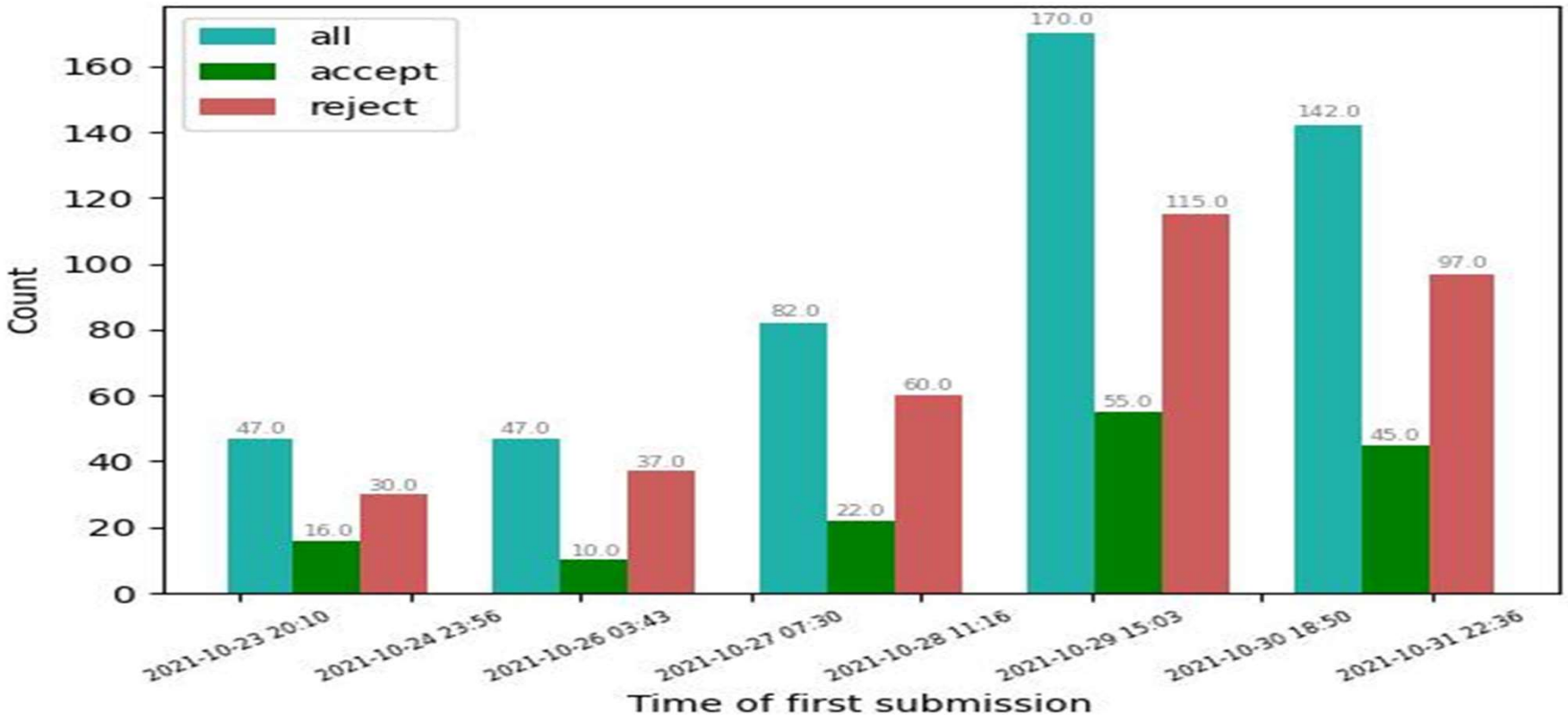


Does submission id matter?



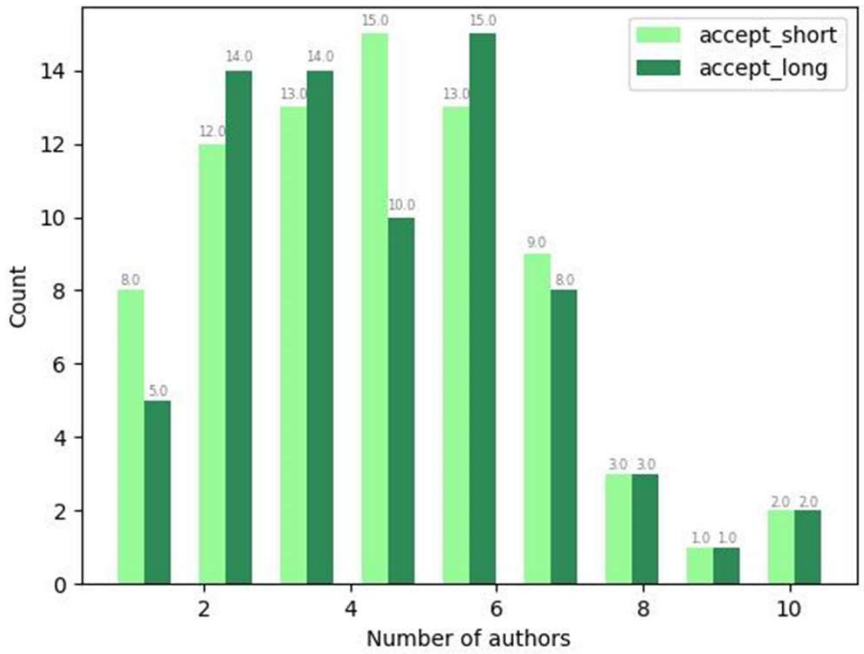
- Is there a sweet spot around 80 percentile?
- How will one know when to hit the 80 percentile?

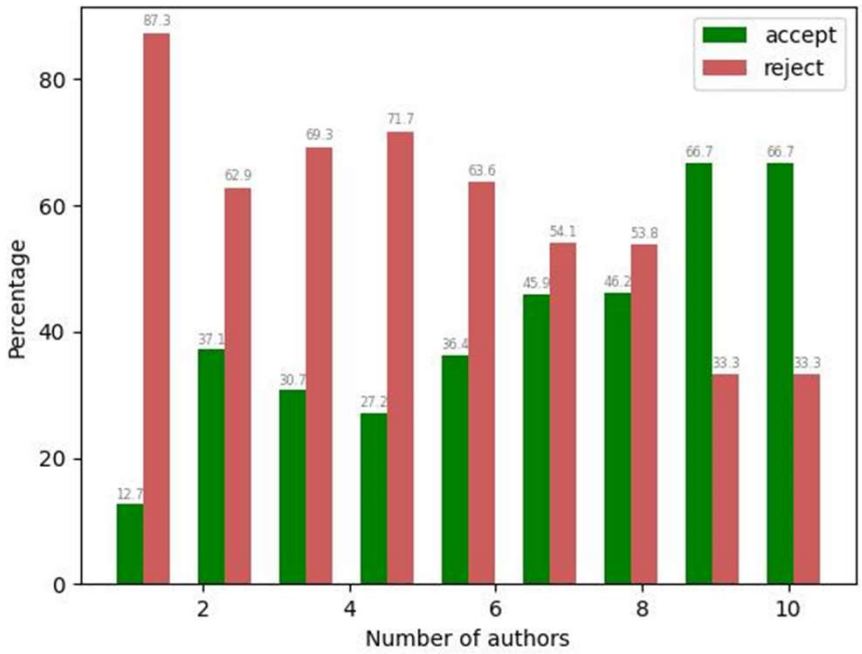
Time of first submission



- 1.5 to 2 days before the deadline!

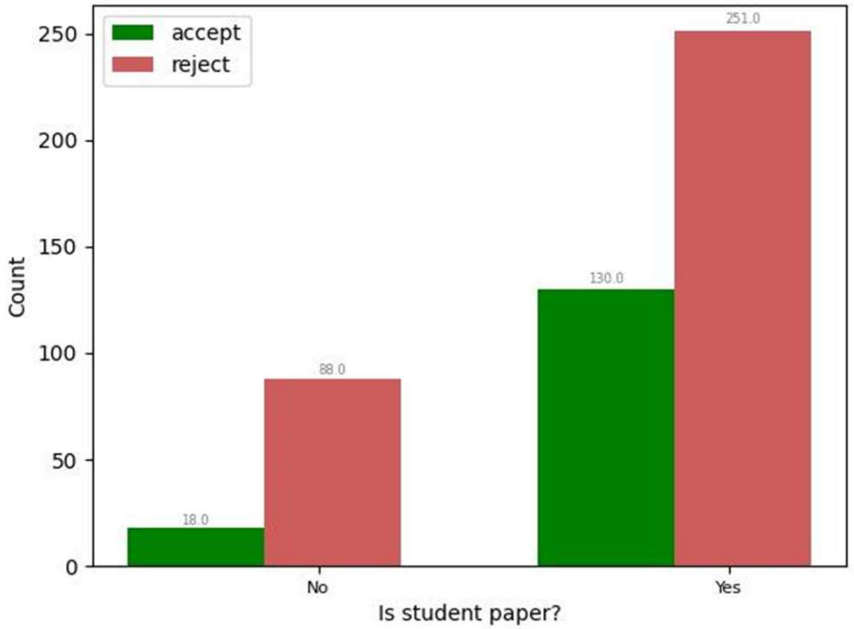
Does number of authors matter?

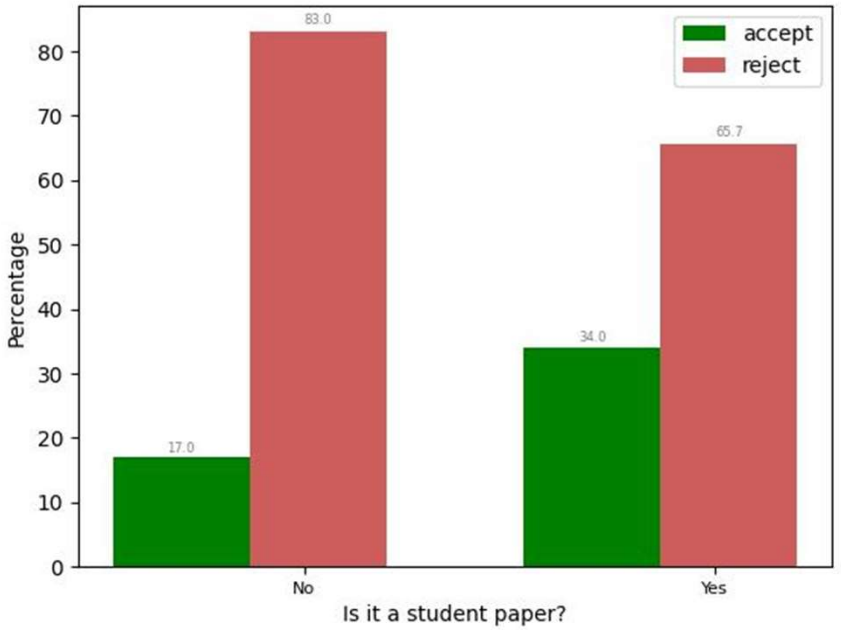




- The more the merrier – collaborations and multiple eyes matter!

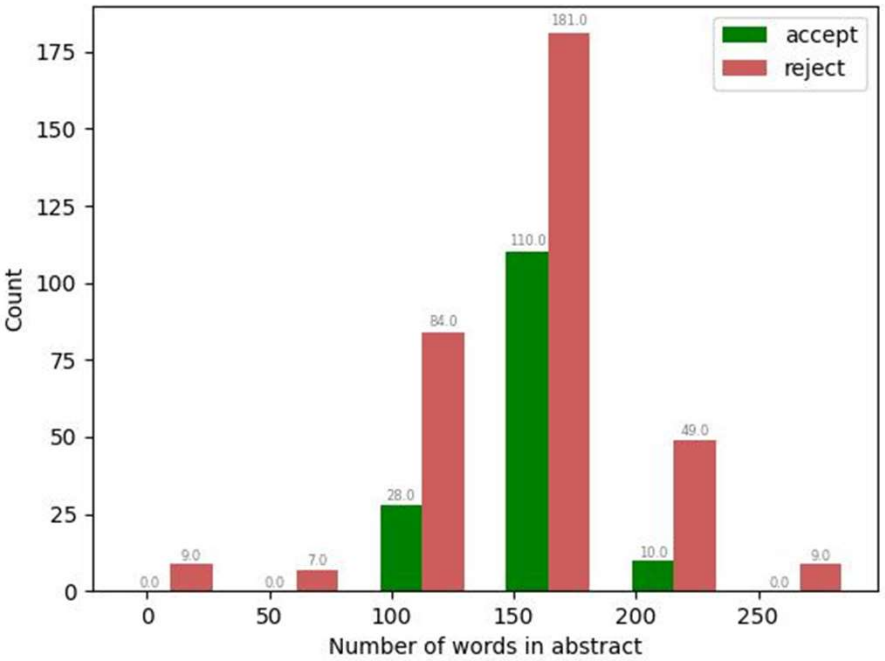
Student paper – first author is a student

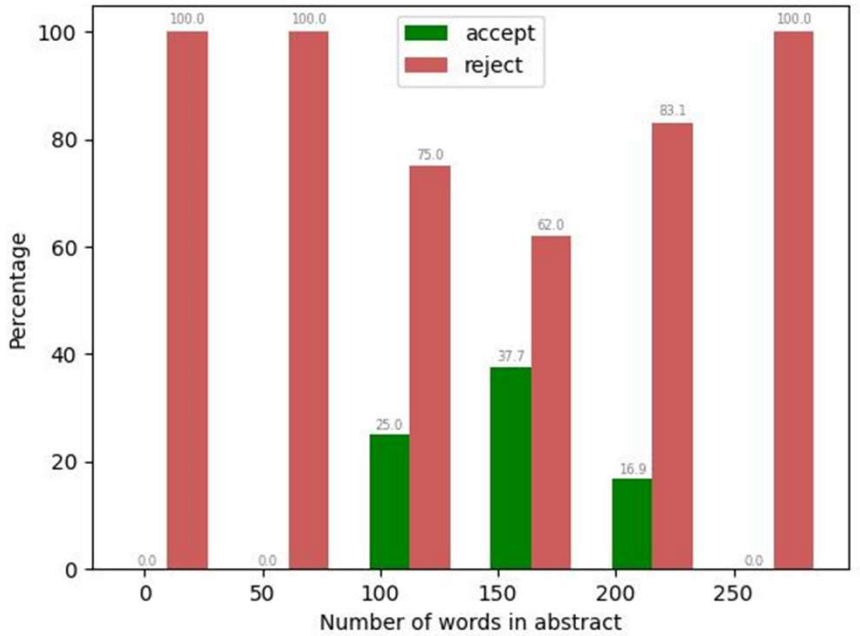




- Significant jump in acceptance percentage when a student is the first author – why?

Number of words in the abstract





- Too long or too short an abstract is not good

Other factors

- Other factors tried but no significant visual difference
- Number of words in the title
- Number of characters in the title
- ...
- ...

Summary

- Data attribute types: nominal, binary, ordinal, interval-scaled, ratio-scaled
- Many types of data sets, e.g., numerical, text, graph, Web, image.
- Gain insight into the data by:
  - Basic statistical data description: central tendency, dispersion, graphical displays
  - Data visualization: map data onto graphical primitives
  - Measure data similarity (not discussed)
- Above steps are the beginning of data preprocessing.
- Many methods have been developed but still an active area of research.

References

- W. Cleveland, Visualizing Data, Hobart Press, 1993
- T. Dasu and T. Johnson. Exploratory Data Mining and Data Cleaning. John Wiley, 2003
- U. Fayyad, G. Grinstein, and A. Wierse. Information Visualization in Data Mining and Knowledge Discovery, Morgan Kaufmann, 2001
- L. Kaufman and P. J. Rousseeuw. Finding Groups in Data: an Introduction to Cluster Analysis. John Wiley & Sons, 1990.
- H. V. Jagadish, et al., Special Issue on Data Reduction Techniques. Bulletin of the Tech. Committee on Data Eng., 20(4), Dec. 1997
- D. A. Keim. Information visualization and visual data mining, IEEE trans. on Visualization and Computer Graphics, 8(1), 2002
- D. Pyle. Data Preparation for Data Mining. Morgan Kaufmann, 1999
- S. Santini and R. Jain," Similarity measures", IEEE Trans. on Pattern Analysis and Machine Intelligence, 21(9), 1999
- E. R. Tufte. The Visual Display of Quantitative Information, 2nd ed., Graphics Press, 2001
- C. Yu , et al., Visual data mining of multimedia data for social and behavioral studies, Information Visualization, 8(1), 2009

Dissimilarity Computation and Data Preprocessing

P. Krishna Reddy

IIIT Hyderabad

Lecture #4

Detailed Syllabus

- **Introduction (1.5 hour): Definition, KDD framework, Issues in data mining.**
- **Data summarization (7.5 hrs): Data Types, Preprocessing,** Characterization, Discrimination, data warehousing techniques (Multidimensional data model, Data warehousing architecture, Data cube computation and OLAP technology)
- Concepts and algorithms for mining patterns and associations (9 hours) (Frequent item-set generation, A priori and FP-growth algorithm, Evaluation of Association patterns) and preprocessing
- Concepts and algorithms related to classification and regression (9hrs) (Overview, Decision tree induction, Over-fitting and under-fitting, Scalable decision tree algorithms, Bayesian Classification, Regression-based Prediction methods (9 hours)
- Concepts and algorithms for clustering the data (9 hours) (Overview, Types of Data, K-means, Aglomerative clustering, Clustering algorithms (DBSCAN, BIRCH, CURE, ROCK, CHAMELEON)).
- Outlier analysis and future trends (graph mining, spatio-temporal mining). (3 hours)

Outline

- Proximity computation measures

Proximity: Similarity or Dissimilarity

- So far we have analyzed the values of single attribute
  - Central tendency, dispersion, and spread of some attribute X.
- Objects are described by multiple attributes.
- **Similarity**
  - Numerical measure of how alike two data objects are
    - How to compare two persons or two objects?
  - 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

Dissimilarity/Similarity metric

- 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
- sim(i,j)= 1-d(i,j)

Metric

- A distance that satisfies the following properties is a metric
  - Non-negativity: $d(i, j) > 0$ if $i \neq j$, and $d(i, i) = 0$ (Positive definiteness)
    - Distance is non-negative number
  - Identify: $d(i,j) >= 0$
    - The distance of an object to itself is 0
  - $d(i, j) = d(j, i)$ (Symmetry)
    - Distance is a symmetric function
  - $d(i, j) \leq d(i, k) + d(k, j)$ (Triangle Inequality)

Data Matrix and Dissimilarity Matrix

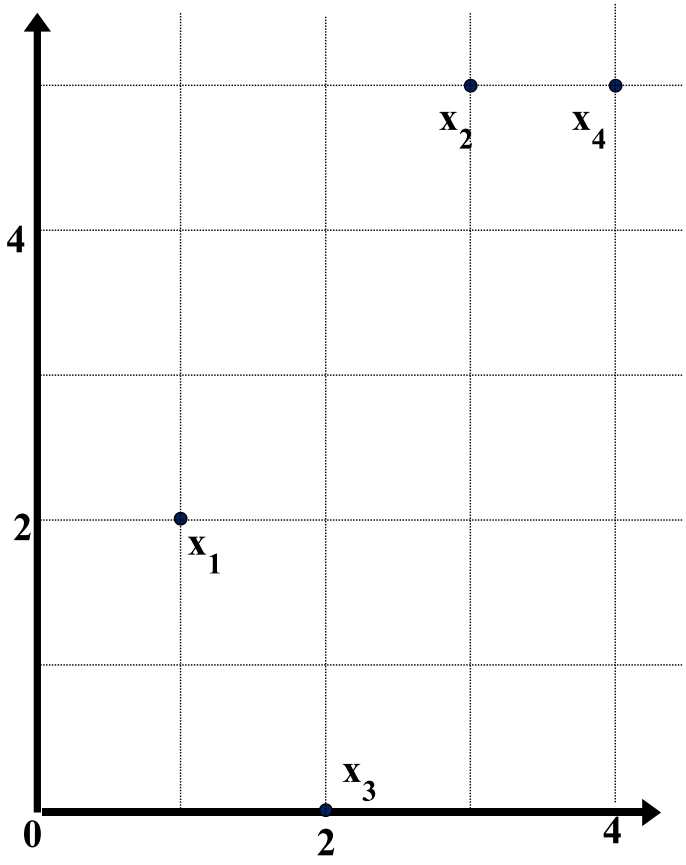
- Data matrix
  - n data points with p dimensions

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

- Dissimilarity matrix
  - n data points, but registers only the distance
  - A triangular matrix
  - Single mode
- Most clustering and nearest neighbor algorithms operate on dissimilarity matrix

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

Example:
Data Matrix and Dissimilarity Matrix



Data Matrix

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

Dissimilarity Matrix

(with Euclidean Distance)

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

Types of Attributes

- Nominal attributes
- Binary attributes
- Numeric data
- Ordinal data
- Mixed data

Proximity Measure for Nominal Attributes

- Can take 2 or more states
  - Example: Map colour (red, yellow, blue, green (generalization of a binary attribute))
- Let i and j are objects with p variables
- Method 1: Simple matching
  - *m*: # of matches, *p*: total # of variables
  - The dissimilarity

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

- Method 2: Use a large number of binary attributes
  - creating a new binary attribute for each of the *M* nominal states

Proximity Measure for Binary Attributes

- A contingency table for binary data
  - q: # of attributes equal to 1 in i and j.
  - r: # of attributes equal to 1 in i but equal to 0 in j
  - s :# of attributes equal to 0 in i but equal to 1 in j
  - t: # of attributes that equal to 0 for both i and j
- Distance measure for symmetric binary variables:
- Distance measure for asymmetric binary variables:
- Jaccard coefficient (*similarity* measure for *asymmetric* binary variables): 1-d(i,j)

Object *i* \ Object *j*	1	0	sum
1	q	r	$q+r$
0	s	t	$s+t$
sum	$q+s$	$r+t$	p

$$d(i,j) = \frac{r+s}{q+r+s+t}$$

$$d(i,j) = \frac{r+s}{q+r+s}$$

$$sim_{Jaccard}(i,j) = \frac{q}{q+r+s}$$

- Note: Jaccard coefficient is the same as "coherence":

$$coherence(i,j) = \frac{sup(i,j)}{sup(i)+sup(j)-sup(i,j)} = \frac{q}{(q+r)+(q+s)-q}$$

Example: Dissimilarity between Asymmetric Binary Variables

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

- Gender is a symmetric attribute (not counted in)
- The remaining attributes are asymmetric binary
- Let the values Y and P be 1, and the value N be 0
- Distance: $d(i, j) = \frac{r + s}{q + r + s}$

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

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

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

Mary

Jack	1	0	$\sum_{row}$
1	2	0	2
0	1	3	4
$\sum_{col}$	3	3	6

Jim

Jack	1	0	$\sum_{row}$
1	1	1	2
0	1	3	4
$\sum_{col}$	2	4	6

Mary

Jim	1	0	$\sum_{row}$
1	1	1	2
0	2	2	4
$\sum_{col}$	3	3	6

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)

Special Cases of Minkowski Distance

- h = 1: Manhattan (city block, L_1 norm) distance
  - E.g., the Hamming distance: the number of bits that are different between two binary vectors

$$d(i,j) = |x_{i_1} - x_{j_1}| + |x_{i_2} - x_{j_2}| + \ldots + |x_{i_p} - x_{j_p}|$$

- h = 2: (L_2 norm) Euclidean distance

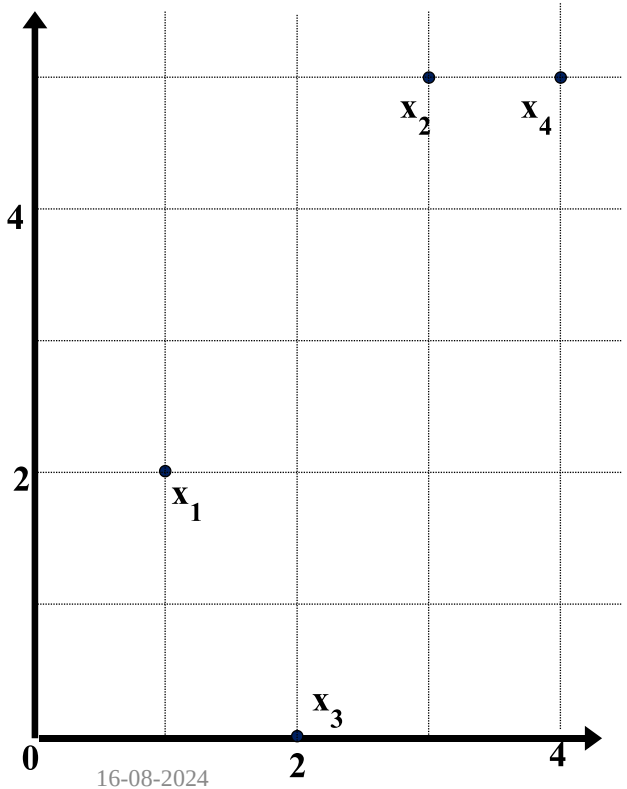
$$d(i,j) = \sqrt{(|x_{i_1} - x_{j_1}|^2 + |x_{i_2} - x_{j_2}|^2 + \ldots + |x_{i_p} - x_{j_p}|^2)}$$

- $h \to \infty$. "supremum" (L_{max} norm, L_∞ norm) distance.
  - This is the maximum difference between any component (attribute) of the vectors

$$d(i,j) = \lim_{h \to \infty} \left(\sum_{f=1}^{p} |x_{if} - x_{jf}|^h \right)^{\frac{1}{h}} = \max_{f}^{p} |x_{if} - x_{jf}|$$

Example: Minkowski Distance

point	attribute 1	attribute 2
x1	1	2
x2	3	5
x3	2	0
x4	4	5



Dissimilarity Matrices

Manhattan (L_1)

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

Euclidean (L_2)

L2	x1	x2	x3	x4
x1	0			
x2	3.61	0		
x3	2.24	5.1	0	
x4	4.24	1	5.39	0

Supremum

L_∞	x1	x2	x3	x4
x1	0			
x2	3	0		
x3	2	5	0	
x4	3	1	5	0

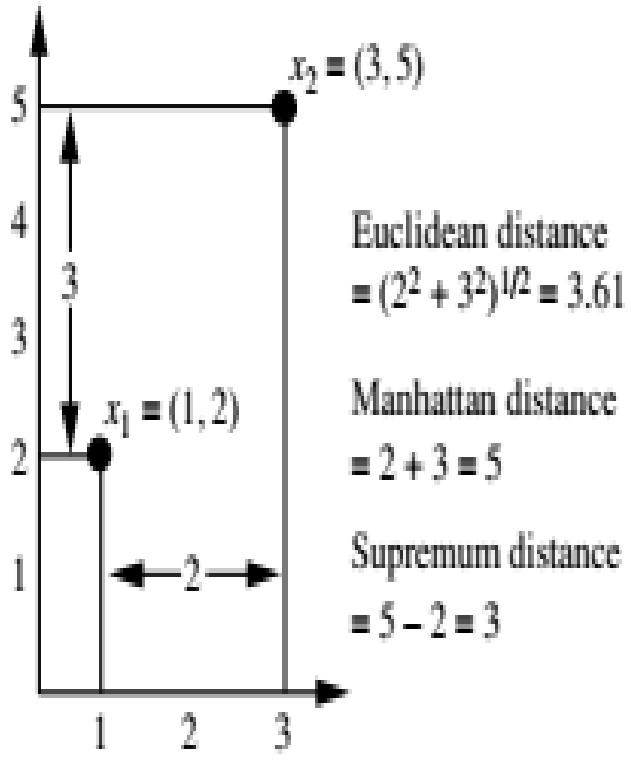


Figure 2.23 Euclidean, Manhattan, and supremum distances between two objects.

Standardizing Numeric Data

- Z-score: $z = \frac{x - \mu}{\sigma}$
  - x: raw score to be standardized, μ: mean of the population, σ: standard deviation
  - the distance between the raw score and the population mean in units of the standard deviation
  - negative when the raw score is below the mean, "+" when above
- An alternative way: Calculate the mean absolute deviation

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

where

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

  - standardized measure (*z-score*): $z_{if} = \frac{x_{if} - m_f}{s_f}$
- Using mean absolute deviation is more robust than using standard deviation

Ordinal Variables

- An ordinal variable can be discrete or continuous
- Order is important, e.g., rank
- Can be treated like interval-scaled
  - replace x_{if} by their rank $r_{if} \in \{1, \ldots, M_f\}$
  - map the range of each variable onto [0, 1] by replacing *i*-th object in the *f*-th variable by

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

  - compute the dissimilarity using methods for interval-scaled variables

Attributes of Mixed Type

- A dataset may contain all attribute types
  - Nominal, symmetric binary, asymmetric binary, numeric, and ordinal
- One may use a weighted formula to combine their effects:

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

  - If f is numeric: Use the normalized distance
  - If f is binary or nominal: $d_{ij}^{(f)} = 0$ if $x_{if} = x_{jf}$; or $d_{ij}^{(f)} = 1$ otherwise
  - If f is ordinal
    - Compute ranks z_{if} (where $z_{if} = \frac{r_{if} - 1}{M_f - 1}$)
    - Treat z_{if} as interval-scaled

Cosine Similarity

- A **document** can be represented by thousands of attributes, each recording the *frequency* of a particular word (such as keywords) or phrase in the document.

Document	*team*	*coach*	*hockey*	*baseball*	*soccer*	*penalty*	*score*	*win*	*loss*	*season*
Document1	5	0	3	0	2	0	0	2	0	0
Document2	3	0	2	0	1	1	0	1	0	1
Document3	0	7	0	2	1	0	0	3	0	0
Document4	0	1	0	0	1	2	2	0	3	0

- Other vector objects: gene features in micro-arrays, …
- Applications: information retrieval, biologic taxonomy, gene feature mapping, ...
- Cosine measure: If d_1 and d_2 are two vectors (e.g., term-frequency vectors), then

  $\cos(d_1, d_2) = (d_1 \bullet d_2) / \|d_1\| \, \|d_2\|$,

  where $\bullet$ indicates vector dot product, $\|d\|$: the length of vector d

Example: Cosine Similarity

- $\cos(d_1, d_2) = (d_1 \bullet d_2) / \|d_1\| \|d_2\|$,
  where $\bullet$ indicates vector dot product, $\|d\|$: the length of vector d

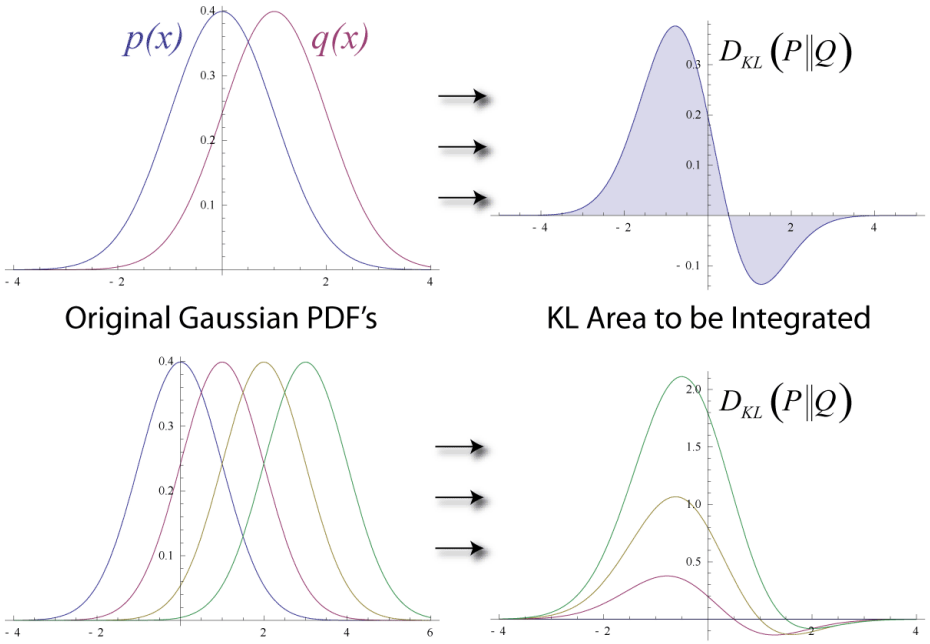
- Ex: Find the **similarity** between documents 1 and 2.

$d_1 = (5, 0, 3, 0, 2, 0, 0, 2, 0, 0)$
$d_2 = (3, 0, 2, 0, 1, 1, 0, 1, 0, 1)$

$d_1 \bullet d_2 = 5*3+0*0+3*2+0*0+2*1+0*1+0*1+2*1+0*0+0*1 = 25$
$\|d_1\| = (5*5+0*0+3*3+0*0+2*2+0*0+0*0+2*2+0*0+0*0)^{0.5} = (42)^{0.5} = 6.481$
$\|d_2\| = (3*3+0*0+2*2+0*0+1*1+1*1+0*0+1*1+0*0+1*1)^{0.5} = (17)^{0.5} = 4.12$
$\cos(d_1, d_2) = 0.94$

KL Divergence: Comparing Two Probability Distributions

- *The Kullback-Leibler (KL) divergence:* Measure the difference between two probability distributions over the same variable *x*
  - From information theory, closely related to *relative entropy*, *information divergence*, and *information for discrimination*
- $D_{KL}(p(x) \parallel q(x))$: divergence of $q(x)$ from $p(x)$, measuring the information lost when $q(x)$ is used to approximate $p(x)$



$$D_{KL}(p(x)||q(x)) = \sum_{x \in X} p(x) \ln \frac{p(x)}{q(x)}$$

Discrete form

$$D_{KL}(p(x)||q(x)) = \int_{-\infty}^{\infty} p(x) \ln \frac{p(x)}{q(x)} dx$$

Continuous form

Ack.: Wikipedia entry: *The Kullback-Leibler (KL) divergence*

More on KL Divergence

$$D_{KL}(p(x)||q(x)) = \sum_{x \in X} p(x) \ln \frac{p(x)}{q(x)}$$

- The KL divergence measures the expected number of extra bits required to code samples from $p(x)$ ("true" distribution) when using a code based on $q(x)$, which represents a theory, model, description, or approximation of $p(x)$
- The KL divergence is not a distance measure, not a metric: asymmetric, not satisfy triangular inequality ($D_{KL}(P\|Q)$ does not equal $D_{KL}(Q\|P)$)
- In applications, P typically represents the "true" distribution of data, observations, or a precisely calculated theoretical distribution, while Q typically represents a theory, model, description, or approximation of P.
- The Kullback–Leibler divergence from Q to P, denoted $D_{KL}(P\|Q)$, is a measure of the information gained when one revises one's beliefs from the prior probability distribution Q to the posterior probability distribution P. In other words, it is the amount of information lost when Q is used to approximate P.
- The KL divergence is sometimes also called the information gain achieved if P is used instead of Q. It is also called the relative entropy of P with respect to Q.

KDD process
Data preprocessing is a major step

- Learning the application domain:
  - relevant prior knowledge and goals of application
- Creating a target data set: data selection
- **Preprocessing: (may take 60% of effort!)**
- **Data reduction and transformation:**
  - **Find useful features, dimensionality/variable reduction, invariant representation.**
- Choosing functions of data mining
  - summarization, classification, association, clustering.
- Choosing the mining algorithm(s)
- Data mining: search for patterns of interest
- Pattern evaluation and knowledge presentation
  - visualization, transformation, removing redundant patterns, etc.
- Use of discovered knowledge

Why Data Preprocessing?

- Data in the real world is dirty
  - incomplete: lacking attribute values, lacking certain attributes of interest, or containing only aggregate data
    - e.g., occupation=“ ”
  - noisy: containing errors or outliers
    - e.g., Salary=“-10”
  - inconsistent: containing discrepancies in codes or names
    - e.g., Age=“42” Birthday=“03/07/1997”
    - e.g., Was rating “1,2,3”, now rating “A, B, C”
    - e.g., discrepancy between duplicate records

Why Is Data Dirty?

- Incomplete data may come from
  - "Not applicable" data value when collected
  - Different considerations between the time when the data was collected and when it is analyzed.
  - Human/hardware/software problems
- Noisy data (incorrect values) may come from
  - Faulty data collection instruments
  - Human or computer error at data entry
  - Errors in data transmission
- Inconsistent data may come from
  - Different data sources
  - Functional dependency violation (e.g., modify some linked data)
- Duplicate records also need data cleaning

Why Is Data Preprocessing Important?

- No quality data, no quality mining results!
  - Quality decisions must be based on quality data
    - e.g., duplicate or missing data may cause incorrect or even misleading statistics.
  - Data warehouse needs consistent integration of quality data
- Data extraction, cleaning, and transformation comprises the majority of the work of building a data warehouse

Data Preprocessing

- Data Preprocessing: An Overview
  - **Data Quality**
  - Major tasks in Data Preprocessing
- Data Cleaning
- Data Integration
- Data Transformation
- Dimensionality Reduction
- Summary

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?

Example

- Consider AllElectronics company
  - You want to analyze <item, price, units_sold>
  - Observations
    - Do not have values for several attributes
    - You want to know whether the item was advertised or not, but you find that it was not recorded
    - There are errors, unusual values, data inconsistencies
  - It means
    - Data is incomplete (lacking required attributes)
    - Inaccurate or noisy (containing errors or deviate from the expected values)
    - Inconsistent (containing discrepancies in the department codes used to categorize items)
      - *Age*="42", *Birthday*="03/07/2010"
      - Was rating "1, 2, 3", now rating "A, B, C"
      - discrepancy between duplicate records

- accuracy, completeness and consistency are common-place properties of real world databases
- Reasons
  - Inaccuracy
    - Data collection instruments may be faulty
    - Human or computer errors in data entry
    - Users purposefully submit incorrect data intentionally for mandatory fields
      - By choosing January 1 by default
    - Data transmission errors
    - Buffer size limitation
    - Naming conventions for data codes and different formats

- Reasons for incomplete data
  - Attributes of interest may not be available
    - They were not considered important at the time of entry.
    - Relevant data may not be recorded due to misunderstanding or because of equipment malfunctions
    - Data may be deleted
    - Missing values of the attributes may not be inferred.
- Data quality depends on the intended use of data
  - If the 80% of addresses are correct
    - It is OK for marketing manager
    - But, may not be Ok for sales manager

Timeliness, Believability, Interpretability

- Timeliness: Consider a giving monthly bonuses to top sales representatives
  - If the data is not updated in a monthly fashion has a negative impact on the data quality
- Believability
  - How much the data is trused by users
  - Several errors many lead to lack of trust
- Interpretability
  - How easy the data are understood
  - For example, the data may use several codes; difficult to interpret.

Data Preprocessing

- Data Preprocessing: An Overview
  - Data Quality
  - **Major tasks in Data Preprocessing**
- Data Cleaning
- Data Integration
- Data Transformation
- Dimensionality Reduction
- Summary

Major Tasks in Data Preprocessing

- **Data cleaning**
  - Fill in missing values, smooth noisy data, identify or remove outliers, and resolve inconsistencies
  - Apply data cleaning techniques
- **Data integration**
  - Integration of multiple databases, data cubes, or files
  - Examples:
  - 1. customer identification may be referred as customer-id in one database and cust_id in another database
  - 2. first name may be Bill on one database and William in another database
  - How to create a common database by integrating Facebook and LinkedIn users?

Major Tasks in Data Preprocessing

- **Data reduction**
  - Question: Data set is very huge. How to reduce the data set size without jeopardizing the data mining results?
  - Dimensionality reduction
    - Compressed representation
  - Numerosity reduction
    - Regression or log-linear models
  - Data compression
- **Data transformation and data discretization**
  - Better results if the data is transformed, i.e., normalized
  - Normalization: Example age and salary: transform all age and salary values between 0 and 1.
  - Discretization: mining data at different abstraction levels
  - Concept hierarchy generation

Data Preprocessing

- Data Preprocessing: An Overview
  - Data Quality
  - Major tasks in Data Preprocessing
- **Data Cleaning**
- Data Integration
- Data Transformation
- Dimensionality Reduction
- Summary

How to Handle Missing Data?

- Method 1: Ignore the tuple:
  - usually done when class label is missing (when doing classification)—not effective when the % of missing values per attribute varies considerably
- Method 2: Fill in the missing value manually: tedious + infeasible?
- Method 3: Use a global constant to fill the missing value
  - a global constant : e.g., "unknown", a new class?!
- Method 4: fill with the attribute mean/ median/mode
- Method 5: fill with the attribute mean/ median/mode belong to the same class: Smarter
- Method 6: the most probable value: inference-based such as Bayesian formula or decision tree-
  - Popular strategy

How to Handle 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?

- Method 1: Binning: consult neighbourhood
  - 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.
- Method 2: Regression
  - smooth by fitting the data into regression functions (will discuss later)
- Method 3: Clustering
  - detect and remove outliers
- Combined computer and human inspection
  - detect suspicious values and check by human (e.g., deal with possible outliers)

Simple Discretization Methods: Binning

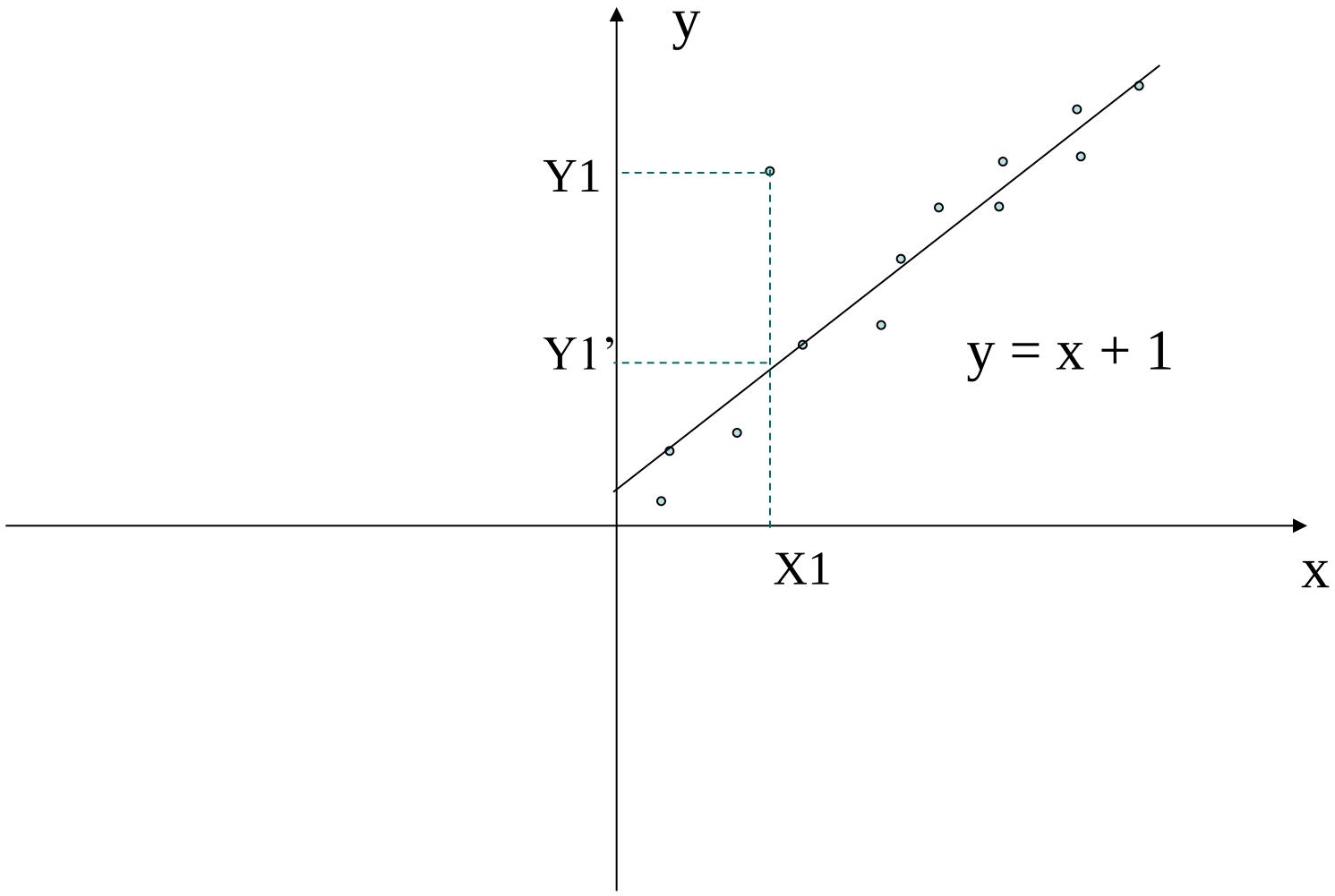
- Equal-width (distance) partitioning
  - Divides the range into *N* intervals of equal size: uniform grid
  - if *A* and *B* are the lowest and highest values of the attribute, the width of intervals will be: $W = (B - A)/N$.
  - The most straightforward, but outliers may dominate presentation
  - Skewed 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

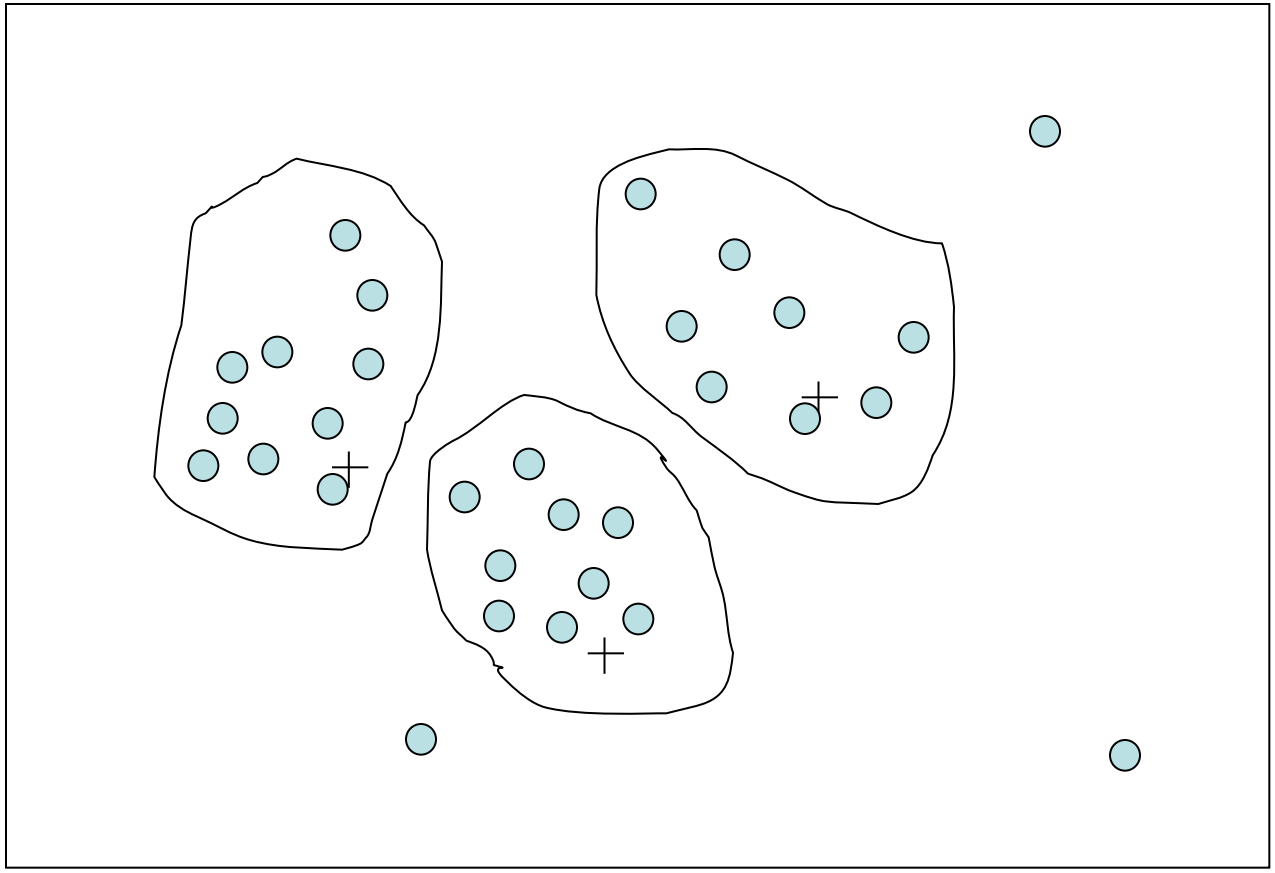
- Sorted data for price (in dollars): 4, 8, 9, 15, 21, 21, 24, 25, 26, 28, 29, 34

* 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

Regression



Cluster Analysis



Lecture #5

Data Cleaning as a Process

- Step 1: Data discrepancy detection
- Step 2: Data transformation.

Data Cleaning as a Process
Step 1: data discrepancy detection

- Reasons
  - Poorly designed data entry forms
  - Deliberate errors
  - Data decay (outdated addresses)
  - Inconsistent data codes
  - The same attribute may have different names
- Use the following techniques
- Use metadata (e.g., domain, range, dependency, distribution)
  - Use any knowledge you have about the data
    - Acceptable values of each attribute, range of attribute
    - Write your script
      - Identify anomalies and trends using basic statistics
- Check Field overloading
    - Developers squeeze new attribute definitions to unused portions of already used attributes

Data Cleaning as a Process
Step 1: data discrepancy detection (contd.)

- Check the following rules
  - Uniqueness rule
    - Each value of the attribute is different from other value
  - Consecutive rule
    - There is no missing value between minimum and maximum
      - Example check numbers
  - Null rule
    - Use of blanks, question marks, special characters or strings
    - Example: License number is kept blank by non-drivers
- Use commercial tools
  - Data scrubbing: use simple domain knowledge (e.g., postal code, spell-check) to detect errors and make corrections
  - Data auditing: by analyzing data to discover rules and relationship to detect violators (e.g., correlation and clustering to find outliers)

Data Cleaning as a Process
Step 2: data transformation

- Data migration
  - Data migration tools: allow transformations to be specified
    - Example: replacing gender with sex
  - ETL (Extraction/Transformation/Loading) tools: allow users to specify transformations through a graphical user interface
- Integration of the two processes (discrepancy detection and transformation)
  - Iterative and interactive (e.g., Potter's Wheels tool)
  - https://control.cs.berkeley.edu/abc/
- Important aspect
  - Update the meta-data whenever you modify data

Data Preprocessing

- Data Preprocessing: An Overview
  - Data Quality
  - Major tasks in Data Preprocessing
- Data Cleaning
- **Data Integration**
- Data Transformation
- Dimensionality Reduction
- Summary

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
- *Sample research paper: Guoliang Li: Human-in-the-loop Data Integration. Proc. VLDB Endow. 10(12): 2006-2017 (2017)*

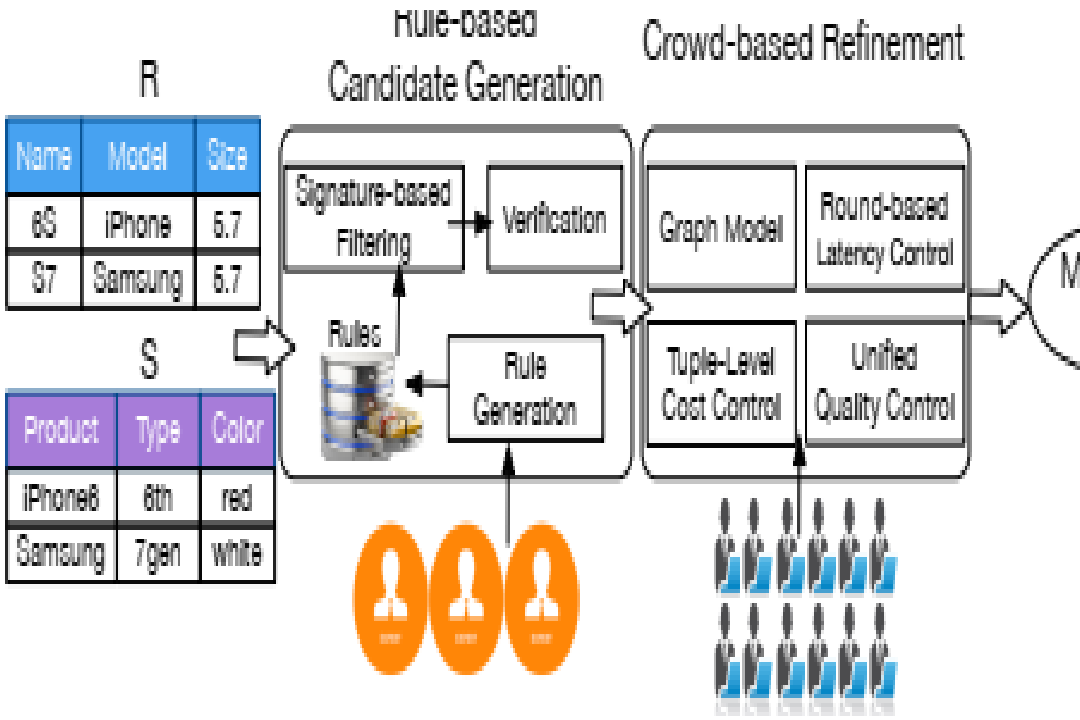


Figure 1: Hybrid Human-Machine Framework.

Data Integration: Entity identification problem:

- Identify real world entities from multiple data sources,
  - e.g., Bill Clinton = William Clinton
- Structure of the data
  - Functional dependencies and referential integrity constraints should match
  - Example: in one data, discount may be on entire order and in another data discount may be on line item
- 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

Data integration: Handling Redundancy in Data Integration

- Redundant data occur often when integration of multiple databases
  - *Object identification*: The same attribute or object may have different names in different databases
  - *Derivable data:* One attribute may be a "derived" attribute in another table, e.g., annual revenue
- Redundant attributes may be able to be detected by *correlation analysis* and *covariance analysis*
- Careful integration of the data from multiple sources may help reduce/avoid redundancies and inconsistencies and improve mining speed and quality

Correlation Analysis (Nominal Data)

- Suppose *A* has *c* distinct values, namely *a*1,*a*2,..,*ac*. *B* has *r* distinct values, namely *b*1,*b*2,…,*br*.
- The data tuples described by *A* and *B* can be shown as a **contingency table**
- Let *(Ai,Bj)* denote the joint event that attribute *A* takes on value *ai* and attribute *B* takes on value *bj*, that is, where *(A = ai, B = bj)*.

- The $\chi 2$ value (also known as the *Pearson* $\chi 2$ *statistic*) is computed as

$$\chi^2 = \sum_{i=1}^{c} \sum_{j=1}^{r} \frac{(o_{ij} - e_{ij})^2}{e_{ij}}, \qquad (3.1)$$

- Where oij is the observed frequency of the joint event (Ai, Bj) and eij is the expected frequency of the joint event (Ai,Bj)

$$e_{ij} = \frac{count(A = a_i) \times count(B = b_j)}{n}, \qquad (3.2)$$

Correlation Analysis (Nominal Data)

- 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

Correlation Analysis (Numeric Data)

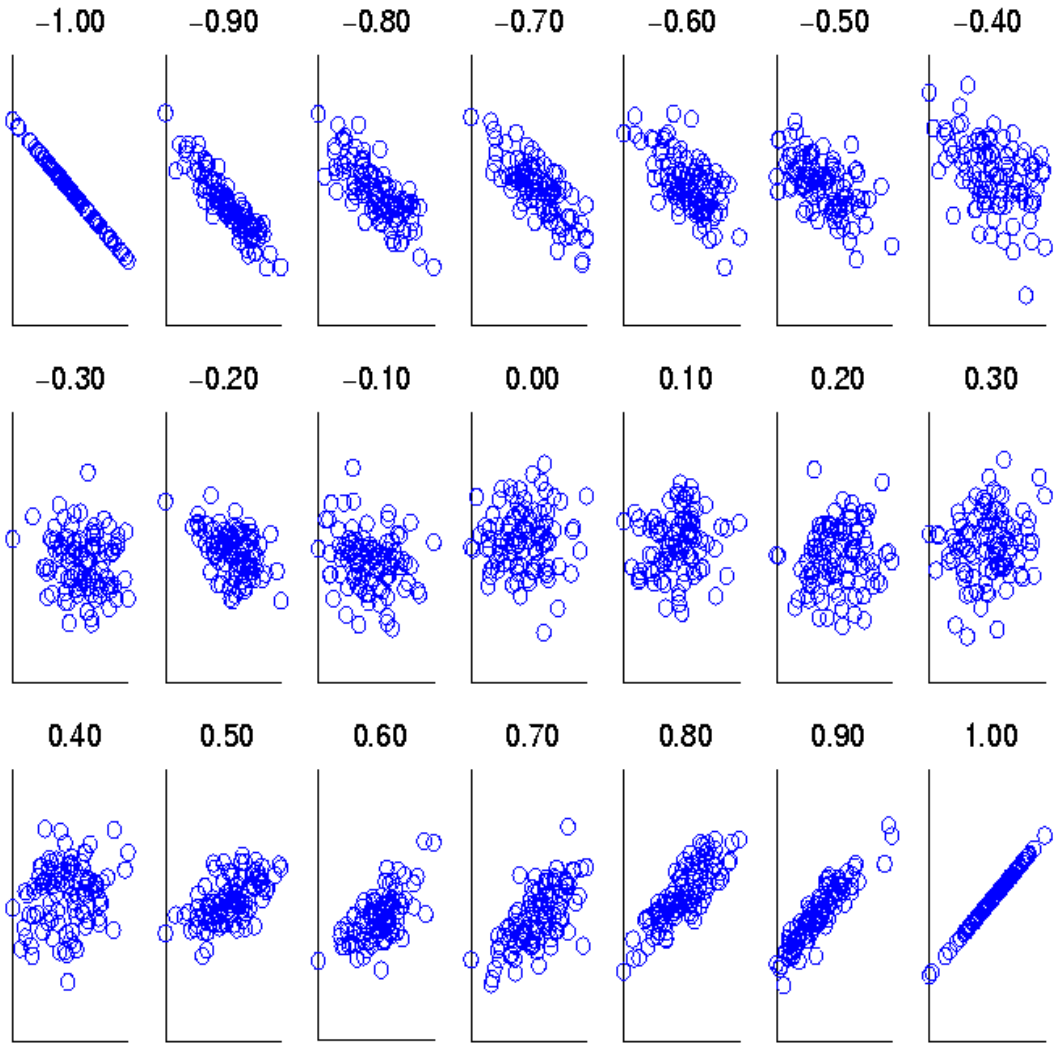
- Correlation coefficient (also called Pearson's product moment coefficient)

$$r_{A,B} = \frac{\sum_{i=1}^{n}(a_i - \bar{A})(b_i - \bar{B})}{n\sigma_A\sigma_B} = \frac{\sum_{i=1}^{n}(a_i b_i) - n\bar{A}\bar{B}}{n\sigma_A\sigma_B},$$

where n is the number of tuples, $\bar{A}$ and $\bar{B}$ are the respective means of A and B, σ_A and σ_B are the respective standard deviation of A and B, and $\Sigma(a_i b_i)$ is the sum of the AB cross-product.

- If $r_{A,B} > 0$, A and B are positively correlated (A's values increase as B's). The higher, the stronger correlation.
- $r_{A,B} = 0$: independent; $r_{AB} < 0$: negatively correlated

Visually Evaluating Correlation



Scatter plots showing the similarity from –1 to 1.

Covariance (Numeric Data)

- Covariance is similar to correlation

$$Cov(A, B) = E((A - \bar{A})(B - \bar{B})) = \frac{\sum_{i=1}^{n}(a_i - \bar{A})(b_i - \bar{B})}{n}$$

Correlation coefficient: $r_{A,B} = \frac{Cov(A, B)}{\sigma_A \sigma_B}$

where n is the number of tuples, $\bar{A}$ and $\bar{B}$ are the respective mean or **expected values** of A and B, σ_A and σ_B are the respective standard deviation of A and B.

- **Positive covariance**: If $Cov_{A,B} > 0$, then A and B both tend to be larger than their expected values.
- **Negative covariance**: If $Cov_{A,B} < 0$ then if A is larger than its expected value, B is likely to be smaller than its expected value.
- **Independence**: $Cov_{A,B} = 0$ but the converse is not true:
  - Some pairs of random variables may have a covariance of 0 but are not independent. Only under some additional assumptions (e.g., the data follow multivariate normal distributions) does a covariance of 0 imply independence

Co-Variance: An Example

$$Cov(A,B) = E((A-\bar{A})(B-\bar{B})) = \frac{\sum_{i=1}^{n}(a_i-\bar{A})(b_i-\bar{B})}{n}$$

- It can be simplified in computation as

$$Cov(A,B) = E(A \cdot B) - \bar{A}\bar{B}$$

- Suppose two stocks A and B have the following values in one week: (2, 5), (3, 8), (5, 10), (4, 11), (6, 14).
- Question: If the stocks are affected by the same industry trends, will their prices rise or fall together?
  - E(A) = (2 + 3 + 5 + 4 + 6)/ 5 = 20/5 = 4
  - E(B) = (5 + 8 + 10 + 11 + 14) /5 = 48/5 = 9.6
  - Cov(A,B) = (2×5+3×8+5×10+4×11+6×14)/5 − 4 × 9.6 = 4
- Thus, A and B rise together since Cov(A, B) > 0.

Data Integration: Tuple duplication

- Duplication should also be detected at tuple level
  - Denormalized tables
  - Same purchaser name is appearing in different addresses

Data Preprocessing

- Data Preprocessing: An Overview
  - Data Quality
  - Major tasks in Data Preprocessing
- Data Cleaning
- Data Integration
- **Data Transformation**
- Dimensionality Reduction
- Summary

Data Transformation

- In *data transformation*, the data are transformed or consolidated into forms appropriate for mining.
- Through appropriate data transformation, the resulting mining process may be more efficient, and the patterns found may be easier to understand.
- Approaches
  - Normalization
  - Discretization
  - Data compression
  - Sampling

Normalization

- The measurement unit used can affect the data analysis.
  - For example, changing measurement units from meters to inches for *height* or from kilograms to pounds for weight may lead to very different results.
  - Salary attribute may dominate over age attribute
- To help avoid dependence on the choice of measurement units, the data should be *normalized* or *standardized*.
- Normalizing the data attempts to give all attributes an equal weight.

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**

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

Discretization

- Data discretization is a common data transformation technique where the raw values of a numeric attribute (e.g., *age*) are replaced by interval labels (e.g., 0–10, 11–20, etc.) or conceptual labels (e.g., *youth*, *adult*, *senior*).
  - The labels, in turn, can be recursively organized into higher-level concepts, resulting in a *concept hierarchy* for the numeric attribute.

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

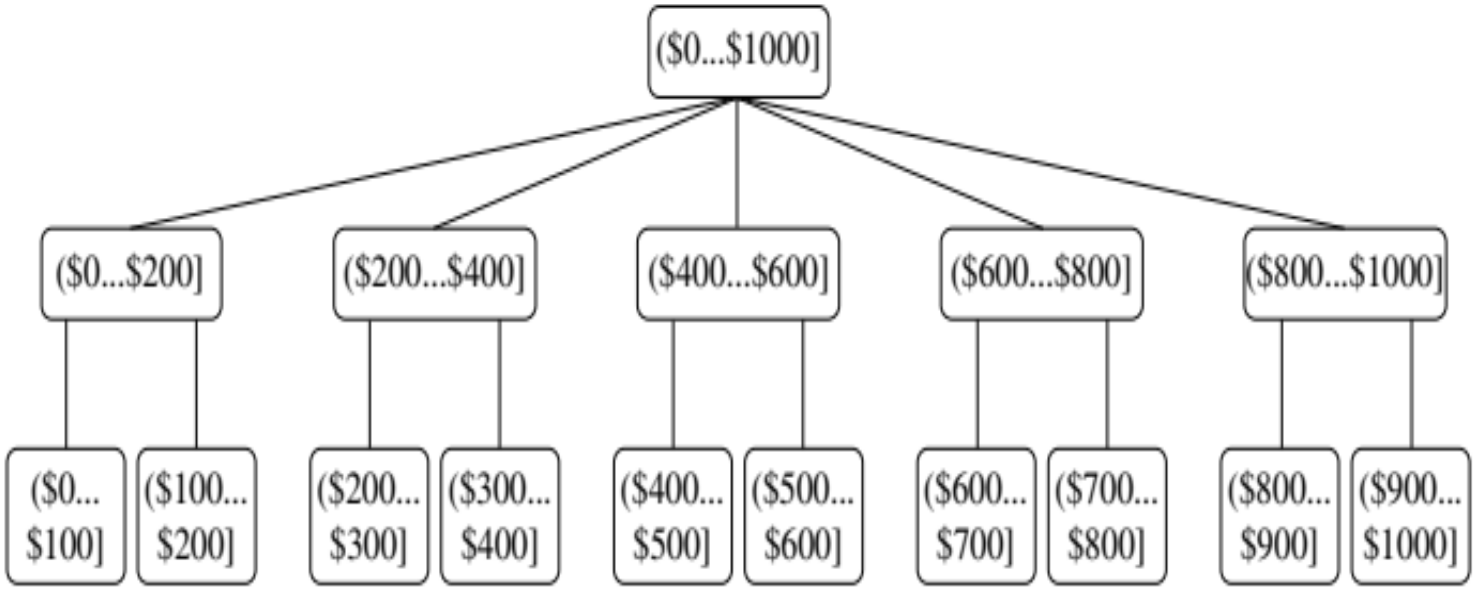


Figure 3.12 A concept hierarchy for the attribute *price*, where an interval ($X...$Y] denotes the range from $X (exclusive) to $Y (inclusive).

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

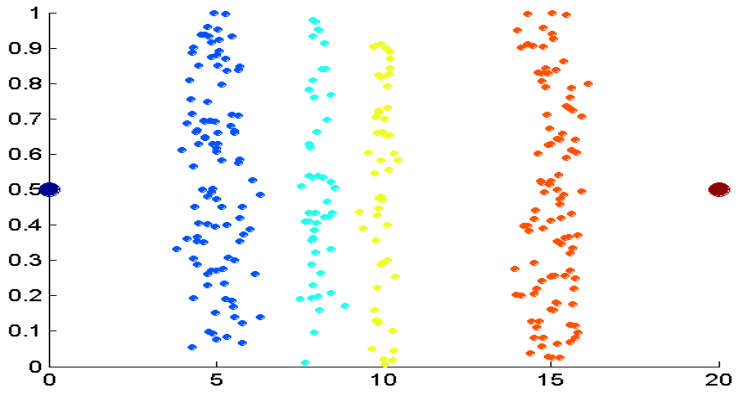
- Equal-width (distance) partitioning
  - Divides the range into *N* intervals of equal size: uniform grid
  - if *A* and *B* are the lowest and highest values of the attribute, the width of intervals will be: $W = (B - A)/N$.
  - The most straightforward, but outliers may dominate the presentation
  - Skewed data is not handled well
- Equal-depth (frequency) partitioning
  - Divide the range into *N* intervals, each containing approximately same number of samples
  - Good data scaling
  - Managing categorical attributes can be tricky

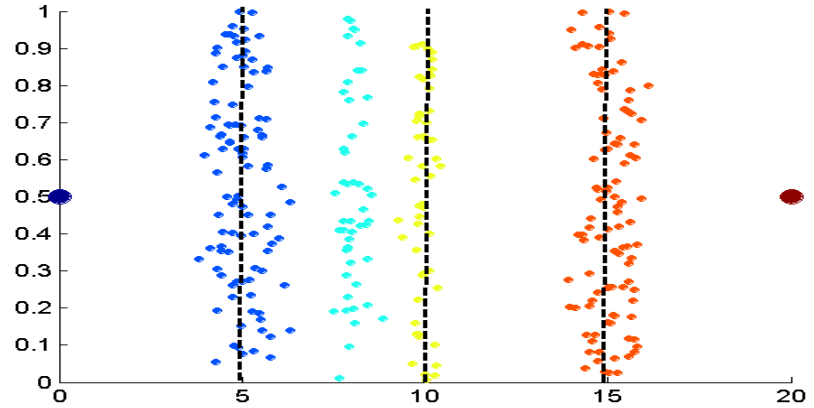
Binning Methods for Data Smoothing

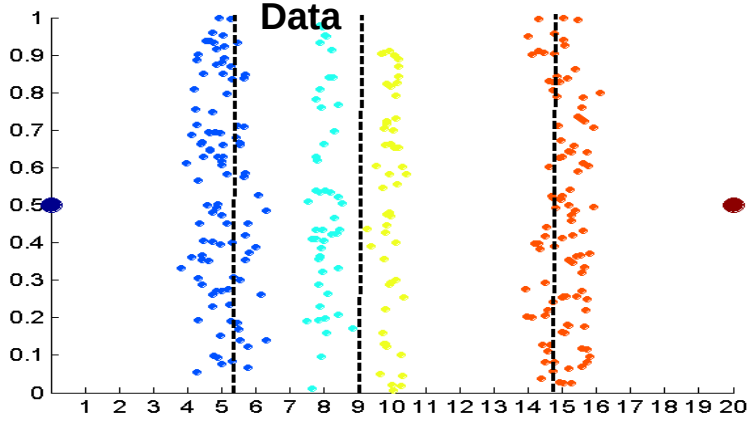
- Sorted data for price (in dollars): 4, 8, 9, 15, 21, 21, 24, 25, 26, 28, 29, 34

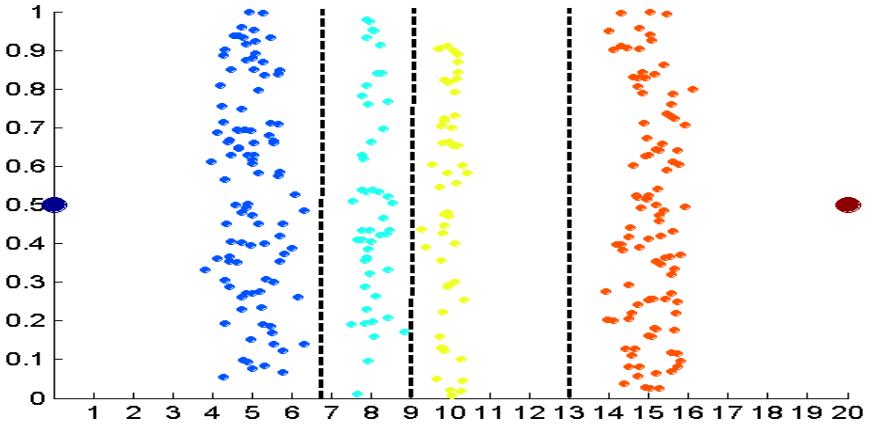
* Partition into equal-frequency (**equi-depth**) bins:
  - Bin 1: 4, 8, 9, 15
  - Bin 2: 21, 21, 24, 25
  - Bin 3: 26, 28, 29, 34
* Smoothing by **bin means**:
  - Bin 1: 9, 9, 9, 9
  - Bin 2: 23, 23, 23, 23
  - Bin 3: 29, 29, 29, 29
* Smoothing by **bin boundaries**:
  - Bin 1: 4, 4, 4, 15
  - Bin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

Discretization Without Using Class Labels (Binning vs. Clustering)









Equal frequency (binning)

K-means clustering leads to better results

Discretization by Classification & Correlation Analysis

- Classification (e.g., decision tree analysis)
  - Supervised: Given class labels, e.g., cancerous vs. benign
  - Using *entropy* to determine split point (discretization point)
  - Top-down, recursive split
  - Details to be covered in Chapter 7
- Correlation analysis (e.g., Chi-merge: χ^2-based discretization)
  - Supervised: use class information
  - Bottom-up merge: find the best neighboring intervals (those having similar distributions of classes, i.e., low χ^2 values) to merge
  - Merge performed recursively, until a predefined stopping condition

Concept Hierarchy Generation

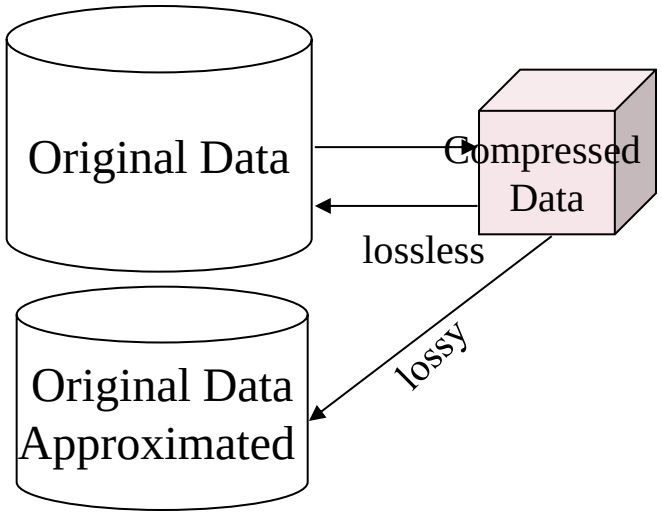
- **Concept hierarchy** organizes concepts (i.e., attribute values) hierarchically and is usually associated with each dimension in a data warehouse
- Concept hierarchies facilitate drilling and rolling in data warehouses to view data in multiple granularity
- Concept hierarchy formation: Recursively 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.

Concept Hierarchy Generation for Nominal Data

- Specification of a partial/total ordering of attributes explicitly at the schema level by users or experts
  - *street* < *city* < *state* < *country*
- Specification of a hierarchy for a set of values by explicit data grouping
  - {Urbana, Champaign, Chicago} < Illinois
- Specification of only a partial set of attributes
  - E.g., only *street* < *city*, not others
- Automatic generation of hierarchies (or attribute levels) by the analysis of the number of distinct values
  - E.g., for a set of attributes: {*street*, *city*, *state*, *country*}

Data Compression

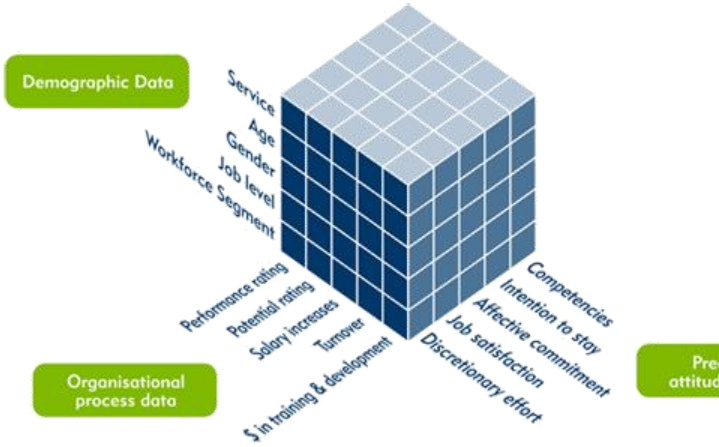
- String compression
  - There are extensive theories and well-tuned algorithms
  - Typically lossless, but only limited manipulation is possible without expansion
- Audio/video compression
  - Typically lossy compression, with progressive refinement
  - Sometimes, small fragments of the signal can be reconstructed without reconstructing the whole
- Time sequence is not audio
  - Typically short and vary slowly with time
- Data reduction and dimensionality reduction may also be considered as forms of data compression



Lossy vs. lossless compression

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



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
  - Exceptions, e.g., weekday, month, quarter, year

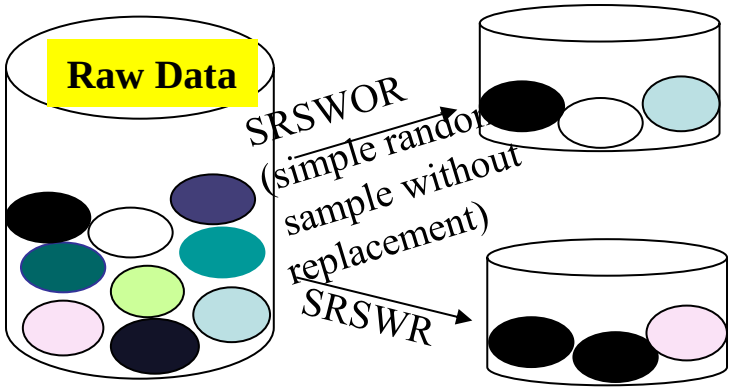


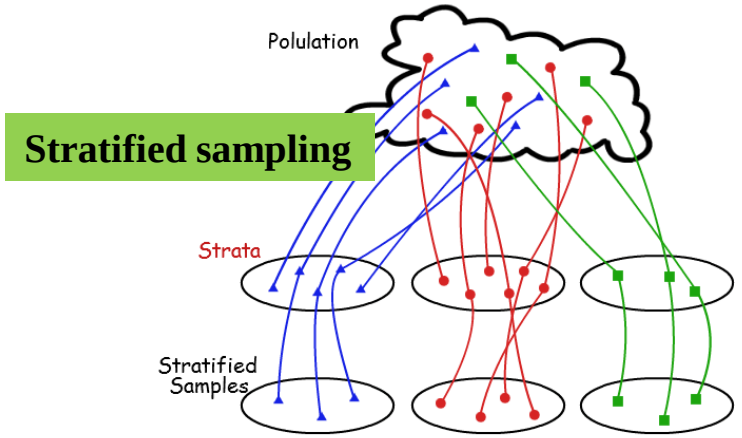
Sampling

- 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:** 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**
  - Partition (or cluster) the data set, and draw samples from each partition (proportionally, i.e., approximately the same percentage of the data)





Data Preprocessing

- Data Preprocessing: An Overview
  - Data Quality
  - Major tasks in Data Preprocessing
- Data Cleaning
- Data Integration
- Data Transformation
- **Dimensionality Reduction**
- Summary

Data Reduction Strategies

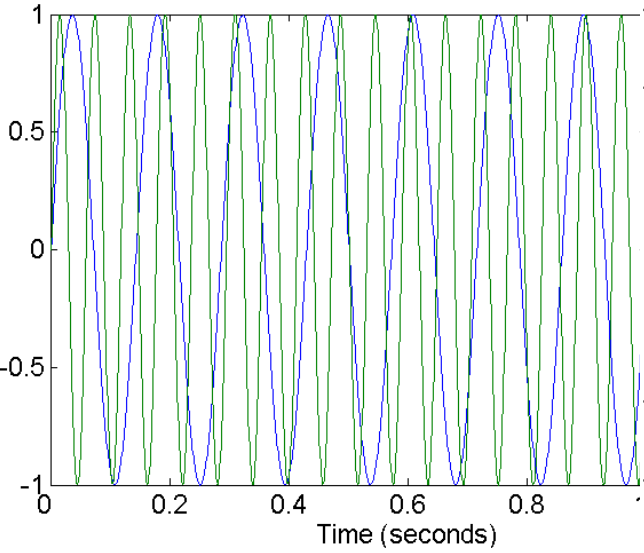
- **Data reduction**: Obtain a reduced representation of the data set that is much smaller in volume but yet produces the same (or almost the same) analytical results
- Why data reduction? — A database/data warehouse may store terabytes of data. Complex data analysis may take a very long time to run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

Data Reduction 1: Dimensionality Reduction

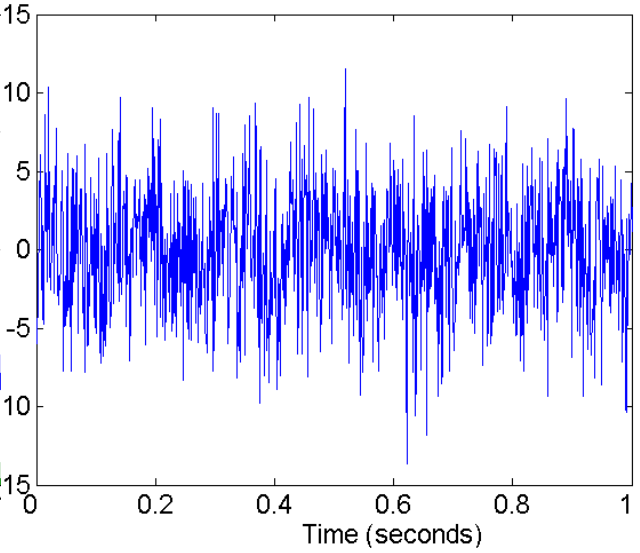
- **Curse of dimensionality**
  - When dimensionality increases, data becomes increasingly sparse
  - Density and distance between points, which is critical to clustering, outlier analysis, becomes less meaningful
  - The possible combinations of subspaces will grow exponentially
- **Dimensionality reduction**
  - 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)

Mapping Data to a New Space

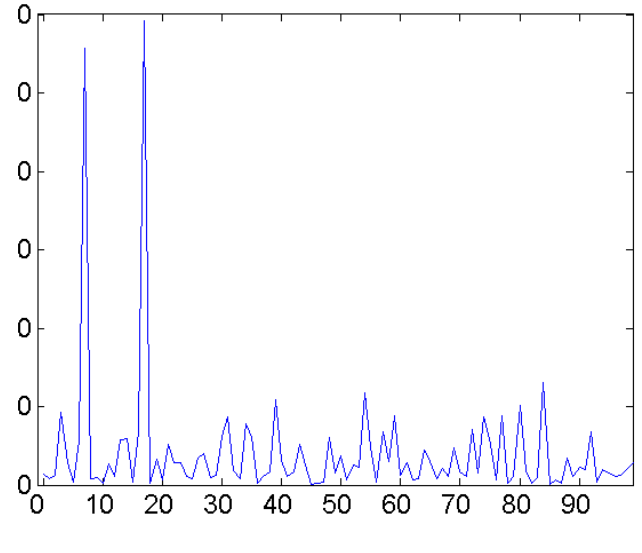
- **Fourier transform**
- **Wavelet transform**



Two Sine Waves



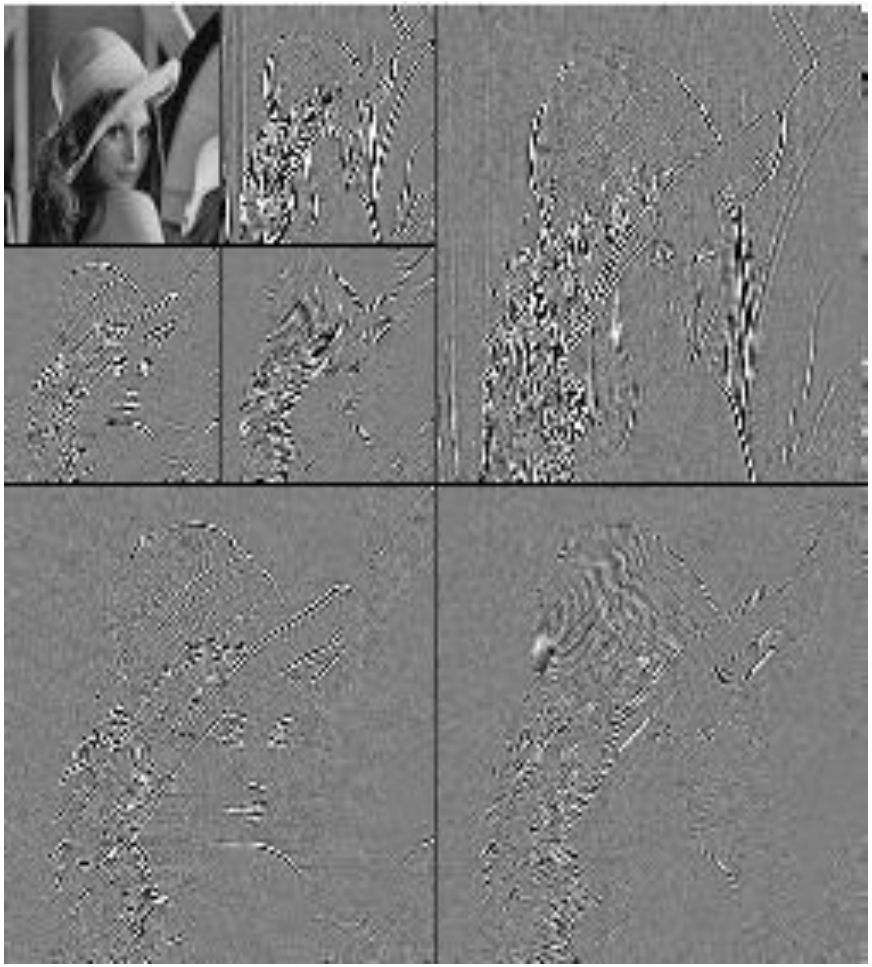
Two Sine Waves + Noise



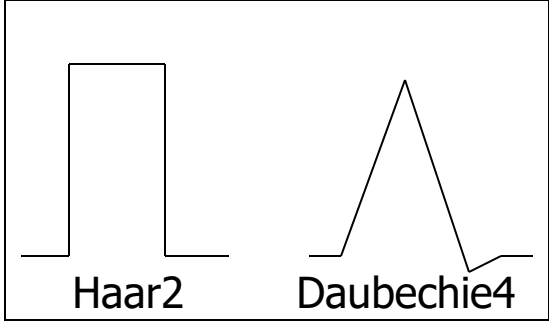
Frequency

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



Wavelet Transformation



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

Wavelet Decomposition

- Wavelets: A math tool for space-efficient hierarchical decomposition of functions
- S = [2, 2, 0, 2, 3, 5, 4, 4] can be transformed to $S_{\wedge}$ = [$2^{3}/_{4}$, $-1^{1}/_{4}$, $^{1}/_{2}$, 0, 0, -1, -1, 0]
- Compression: many small detail coefficients can be replaced by 0's, and only the significant coefficients are retained

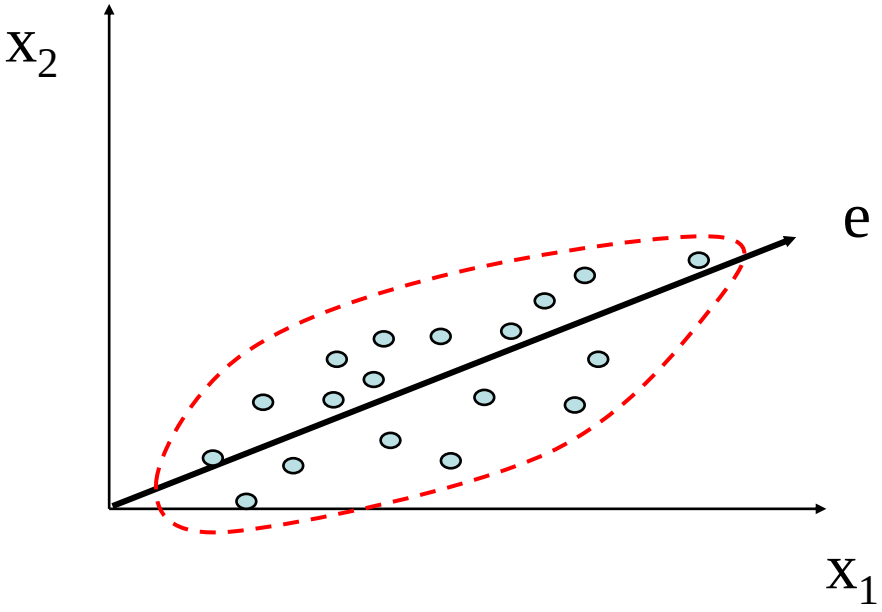
Resolution	Averages	Detail Coefficients
8	[2, 2, 0, 2, 3, 5, 4, 4]	
4	[2, 1, 4, 4]	[0, −1, −1, 0]
2	$[1\frac{1}{2}, 4]$	$[\frac{1}{2}, 0]$
1	$[2\frac{3}{4}]$	$[-1\frac{1}{4}]$

Why Wavelet Transform?

- Use hat-shape filters
  - Emphasize region where points cluster
  - Suppress weaker information in their boundaries
- Effective removal of outliers
  - Insensitive to noise, insensitive to input order
- Multi-resolution
  - Detect arbitrary shaped clusters at different scales
- Efficient
  - Complexity O(N)
- Only applicable to low dimensional data

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
- Reference: https://www.turing.com/kb/guide-to-principal-component-analysis

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

Heuristic Search in Attribute Selection

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

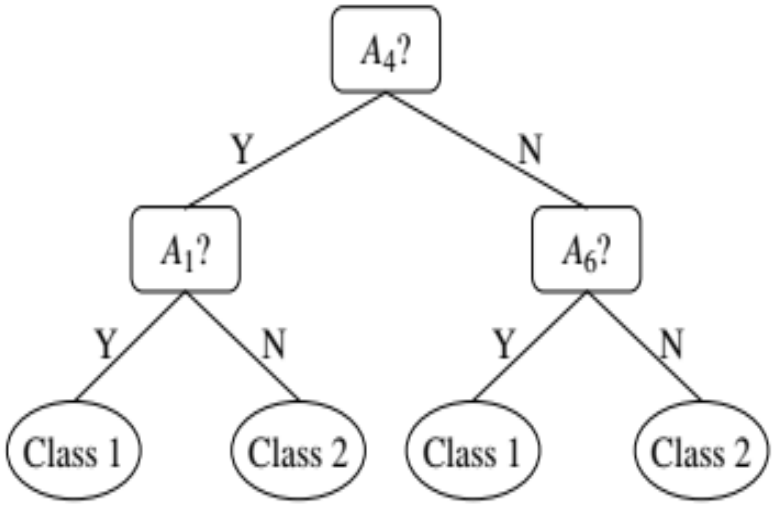
Forward selection	Backward elimination	Decision tree induction
Initial attribute set: $\{A_1, A_2, A_3, A_4, A_5, A_6\}$ Initial reduced set: $\{\}$ => $\{A_1\}$ => $\{A_1, A_4\}$ => Reduced attribute set: $\{A_1, A_4, A_6\}$	Initial attribute set: $\{A_1, A_2, A_3, A_4, A_5, A_6\}$ => $\{A_1, A_3, A_4, A_5, A_6\}$ => $\{A_1, A_4, A_5, A_6\}$ => Reduced attribute set: $\{A_1, A_4, A_6\}$	Initial attribute set: $\{A_1, A_2, A_3, A_4, A_5, A_6\}$  => Reduced attribute set: $\{A_1, A_4, A_6\}$

Figure 3.6 Greedy (heuristic) methods for attribute subset selection.

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

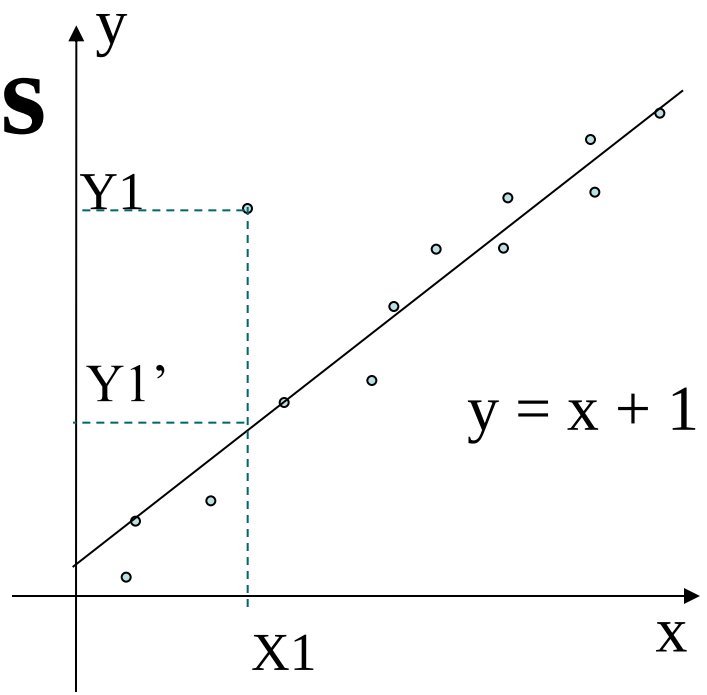
Data Reduction 2: Numerosity Reduction

- 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, …

Numerosity 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

Regress Analysis and Log-Linear Models

- Linear regression: $Y = w\,X + b$
  - Two regression coefficients, w and b, specify the line and are to be estimated by using the data at hand
  - Using the least squares criterion to the known values of Y_1, Y_2, ..., X_1, X_2, ...
- Multiple regression: $Y = b_0 + b_1\,X_1 + b_2\,X_2$
  - Many nonlinear functions can be transformed into the above
- Log-linear models:
  - Approximate discrete multidimensional probability distributions
  - Estimate the probability of each point (tuple) in a multi-dimensional space for a set of discretized attributes, based on a smaller subset of dimensional combinations
  - Useful for dimensionality reduction and data smoothing

Histogram Analysis

- Divide data into buckets and store average (sum) for each bucket
- Partitioning rules:
  - Equal-width: equal bucket range
  - Equal-frequency (or equal-depth)

Example 3.3 **Histograms.** The following data are a list of *AllElectronics* prices for commonly sold items (rounded to the nearest dollar). The numbers have been sorted: 1, 1, 5, 5, 5, 5, 5, 8, 8, 10, 10, 10, 10, 12, 14, 14, 14, 15, 15, 15, 15, 15, 15, 18, 18, 18, 18, 18, 18, 18, 18, 20, 20, 20, 20, 20, 20, 20, 21, 21, 21, 21, 25, 25, 25, 25, 25, 28, 28, 30, 30, 30.

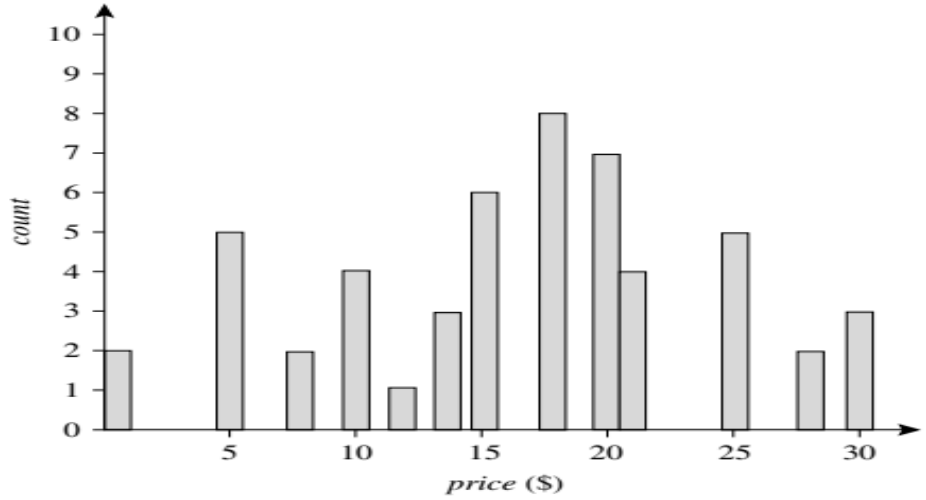


Figure 3.7 A histogram for *price* using singleton buckets—each bucket represents one price–value/frequency pair.

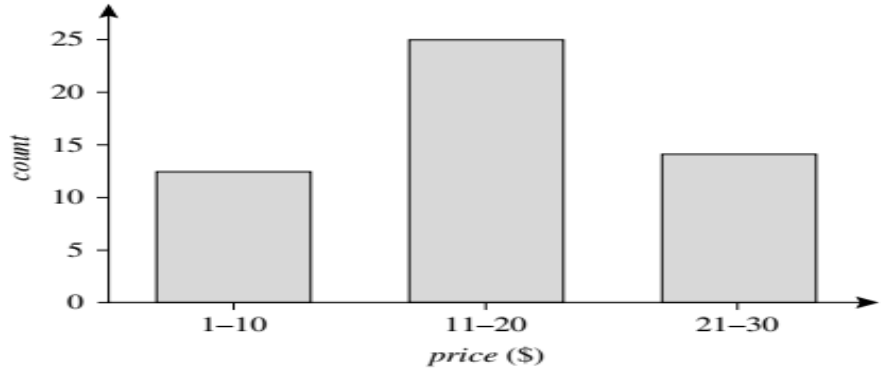
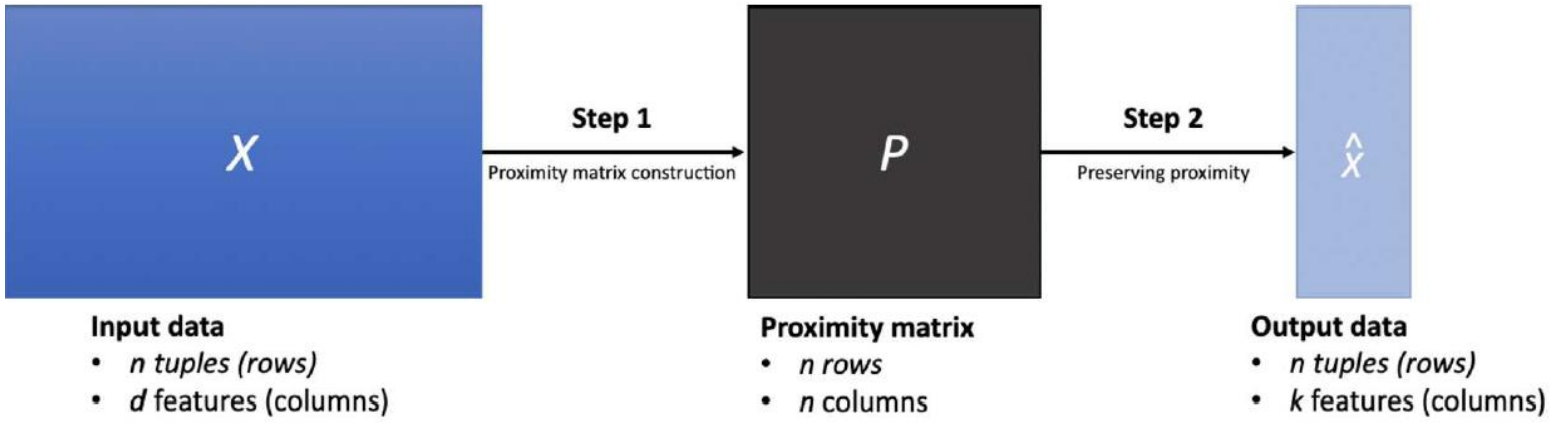


Figure 3.8 An equal-width histogram for *price*, where values are aggregated so that each bucket has a uniform width of $10.

Nonlinear Dimensionality Reduction Methods

- PCA is a linear method for dimensionality reduction
  - Each principal component is a linear combination of the original input attributes
  - It works well if the input data approximately follows a Gaussian distribution or forms a few linearly separable clusters
- When the input data is **linearly inseparable**, we need to construct a proximity matrix (P) and learn a new matrix with *k* features (*k* << *d*) that preserves the proximity



Nonlinear Dimensionality Reduction (I): Kernel PCA (KPCA)

- Use a kernel function $\kappa(\cdot)$ to construct the kernel matrix: $P(i, j) = \kappa(x_i, x_j)$, and learn the best low-dimensional representations so that the estimated proximity matrix $P\hat{}$ is as close as possible to the kernel matrix P
- This can be obtained by using top-*k* eigenvectors and eigenvalues of the kernel matrix P
- Typical kernel functions:
  - (1) polynomial kernel: $\kappa(x_i, x_j) = (1 + x_i \cdot x_j)^p$
  - (2) radial basis function (RBF): $\kappa(x_i, x_j) = e^{\frac{-\|x_i - x_j\|^2}{2\sigma^2}}$
  - If we choose a linear kernel: $\kappa(x_i, x_j) = x_i \cdot x_j$, KPCA degenerates to the standard PCA
- Major formulas of Kernel PCA vs. SNE (Stochastic neighborhood embedding)

	Step 1: Proximity Construction	Step 2: Preserving Proximity
KPCA	$P(i, j) = \kappa(x_i, x_j)$	$\min \sum_{i,j=1}^{n} (P(i, j) - \hat{P}(i, j))^2 = \|P - \hat{P}\|_{fro}^2$
SNE	$P(i, j) = \frac{e^{-d_{ij}^2}}{\sum_{l=1, l \neq i}^{n} e^{-d_{il}^2}}$	$\min \sum_{i=1}^{n} \mathrm{KL}(P_i \| \hat{P}_i)$

Nonlinear Dimensionality Reduction (II): SNE

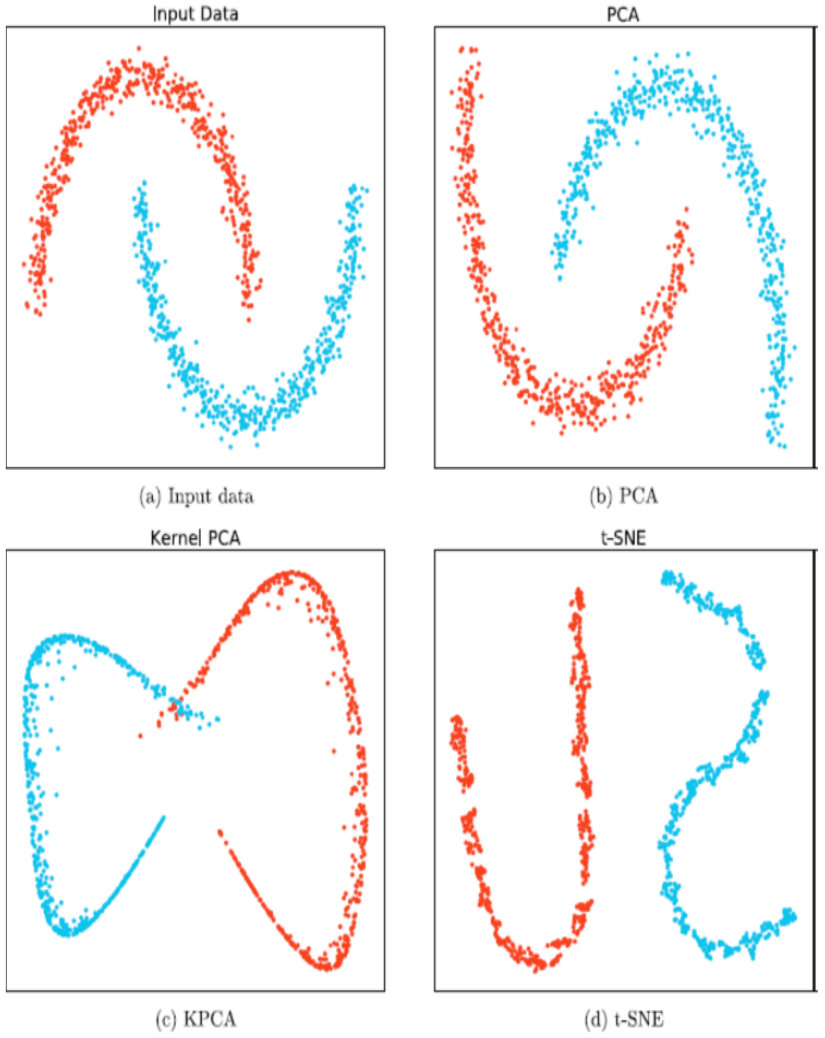
- SNE (Stochastic neighborhood embedding)
  - Construct a proximity matrix P using the formula: $P(i,j) = \frac{e^{-d_{ij}^2}}{\sum_{l=1,l\neq i}^{n} e^{-d_{il}^2}}$ where $d_{ij}^2 = \frac{\|x_i - x_j\|^2}{2\sigma^2}$
    - rep. the probability that x_j is the neighbor of x_i
  - Suppose we have learned the low-dimensional representations $x_i\hat{}$, we can compute anther estimated proximity matrix in the similar way: $P\hat{}\,(i,j)$
  - We want to make the estimated proximity matrix P^ to be as close as possible to P
  - That is, we want to minimize the overall K-L divergence, that is,

$$\hat{x}_i = \arg\min_{\hat{x}_i,(i=1,\ldots n)} \sum_{i=1}^{n} D_{KL}(P_i||\hat{P}_i)$$

	Step 1: Proximity Construction	Step 2: Preserving Proximity
KPCA	$P(i,j) = \kappa(x_i, x_j)$	$\min \sum_{i,j=1}^{n} (P(i,j) - \hat{P}(i,j))^2 = \|P - \hat{P}\|_{fro}^2$
SNE	$P(i,j) = \frac{e^{-d_{ij}^2}}{\sum_{l=1,l\neq i}^{n} e^{-d_{il}^2}}$	$\min \sum_{i=1}^{n} \mathrm{KL}(P_i\|\|\hat{P}_i)$

Example: Comparison on Nonlinear Data Points: Linear vs. Nonlinear Dimensional Reduction Methods

- Visualization: An example of linear vs. nonlinear dimensionality reduction methods
  - Given a collection of input data in 2-D space (Fig. (a)): Red and blue data points are not linearly separable
  - PCA transformation can not make it linearly separable
  - KPCA can make the points linearly separable
  - t-SNE (t-distributional NSE) can make them linearly separable



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

References

- Refer to Bibliographic notes

- **Book: Jiawei Han, Jian Pei, Hanghang Tong, Data Mining: Concepts and Techniques, Fourth edition, 2022, Elseiver Inc.**

ATTRIBUTE-ORIENTED INDUCTION IN RELATIONAL DATABASES
by

Yandong Cai

Presented by

P. Krishna Reddy
IIIT Hyderabad

- Reference 1: Jiawei Han, Yandong Cai, Nick Cercone: Knowledge Discovery in Databases: An Attribute-Oriented Approach. VLDB 1992: 547-559

- Reference 2: Attribute Oriented Induction in Relational Databases, MS thesis, 1989, Simon Fraser University, https://summit.sfu.ca/item/4542

Outline

- Introduction
- Learning from examples
- Concepts of learning from databases
- Attribute oriented induction in relational databases
- Variations of the learning algorithms
- Discussion
- Implementation and experiments
- Conclusions and future research

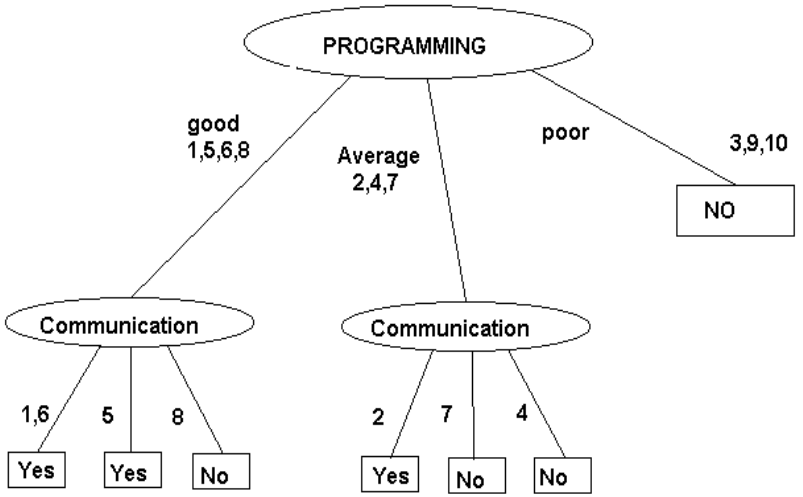
Introduction

- It is advantageous to learn characteristics of data in relational databases.
- By learning from databases, knowledge rules can be extracted from the large amount of data and interesting relationships among data can be discovered automatically.
- Relational databases store a large amount of information in a structured and organized manner.
- Each tuple in the database can be viewed as a typed logical formula in the conjunctive normal form.
- Such uniformity facilitates the application of well developed database implementation techniques and the development of efficient learning algorithms in large databases

Paradigm: Learning from Examples

- The paradigm of "learning from examples"
  - Learning by generalizing specific facts or observations has been adopted in many existing induction learning algorithms
  - **Example:** A recruitment agency which specialises in recruiting trainee programmers wants to develop a decision tree to filter out unsuitable applicants.
  - The agency we three criteria to make this decision: programming ability(Prog.), communication ability*Com.) & presentation(Pres.). Suppose the Agency provide the examples in the following Table:

NO	Prog.	Com	Pres	Decision
1.	good	ok	neat	yes
2.	average	ok	fair	yes
3.	poor	liar	scruffy	no
4.	average	liar	fair	no
5.	good	boring	scruffy	yes
6.	good	ok	scruffy	yes
7.	average	boring	scruffy	no
8.	good	liar	scruffy	no
9.	poor	boring	fair	no
10.	poor	boring	scruffy	no



Motivation

- The facts (large) in the database can be viewed as examples for learning.
  - The paradigm "learning from examples" should be a strategy for learning
- However, (limitations)
  - Current learning systems do not integrate well with relational database systems
    - Due to large number of possible combinations in such testing, the tuple-oriented approach is quite inefficient to perform learning from large databases
    - For example, if there are 100 training examples (tuples), each tuple has 5 attributes, and each attribute value can be one of the three concepts on three different generalization levels, such coverage testing will be invoked up to 100 * (5*5*5) = 12500 times in the worst case.
  - Moreover, most existing algorithms do not take the features and implementation techniques provided by database systems
    - They do not have negative examples

Introduction

- Following the learning from examples paradigm, the proposed attribute-oriented concept tree ascending technique, integrates database operations with the learning process.
- There are two types of knowledge rules
  - characteristic rules and classification rules
- Two algorithms are proposed
  - LCHR algorithm for Learning CHaracteristic Rules
  - LCLR algorithm for Learning CLassification Rules.
- These two algorithms can learn both conjunctive rules and restricted forms of disjunctive rules, and learning can be performed with databases containing exceptions and noisy data using database statistics.
- The algorithms show that attribute-oriented induction substantially reduces the complexity of the database learning processes

Outline

- Introduction
- **Learning from examples (Related Work)**
- Concepts of learning from databases
- Attribute oriented induction in relational databases
- Variations of the learning algorithms
- Discussion
- Implementation and experiments
- Conclusions and future research

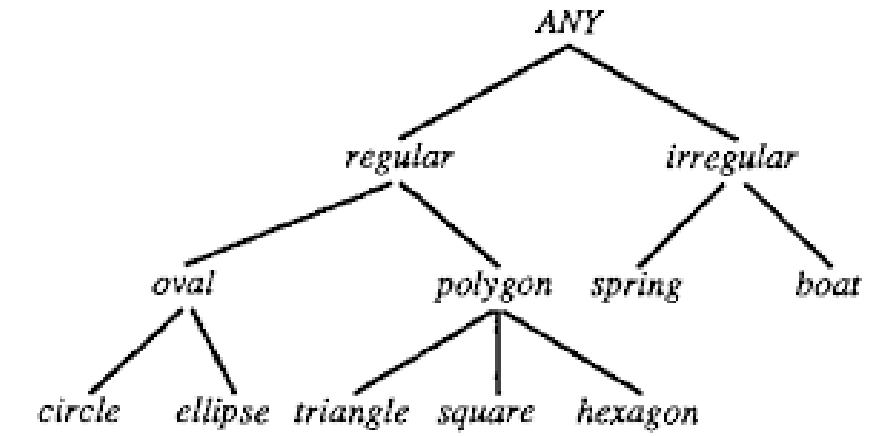
Learning from Examples: An AI approach

- Learning from examples can be viewed as a reasoning process from specific instances to general concepts.
- Learning from examples can be characterized by a tuple <P, N, C, A>, [GeN87]
  - where P is a set of positive examples of a concept,
  - N is a set of negative examples of a concept,
  - C is the conceptual bias which consists of a set of concepts to be used in defining learning rules and results,
  - and A is the logical bias which captures particular logic forms

- Approach 1 [Mic83]:
  - The training examples are classified in advance by the teacher into two disjoint sets, the positive example set and the negative example set. The learning task is to generalize these low-level concepts to general rules.
- Approach 2 [GeN87]:
  - There could be numerous inductive conclusions derived fiom a set of training examples. For instance, the concept "red" can be generalized in several ways: "red or black", "dark color", "warm color", etc.
  - To cope with this multiplicity of possibilities, it is necessary to use some additional information, problem background knowledge, to constrain the space of possible inductive conclusions and locate the most desired one(s).
  - The conceptual bias and the logical bias provide the desired concepts and logic forms which serve as this kind of background knowledge.
  - These biases restrict the candidates to formulas with a particular vocabulary and logic forms. Only those concepts which can be written in terms of this fixed vocabulary and logic forms are considered in the learning process.

- Depending on the structure of the attribute domains, we can distinguish among three basic types of attributes [Mic83]:
  - Nominal attributes:
    - Consists of independent symbols or names
    - No structure is assumed to relate the values in the domain.
    - Example: computer ID
  - Numeric attributes:
    - The value set of such attributes is a totally odered set. For example, weight, salary and gpa are numeric attributes.
  - Structured attributes:
    - The value set of such attributes has a tree structure which forms a generalization hierarchy.
    - A parent node in such a structure represents a more general concept than the concepts represented by its children nodes.

- The domain of structured attributes is defined by the problem background knowledge.
  - For example, the attribute shape could be a structured attribute whose domain is a tree structure with a set of leaves:
  - {triangle, circle, ellipse, hexagon, square, boat, spring), and
  - the non-leaf nodes are defined by rules:
    {circle, ellipse) c oval
    {triangle, square, hexagon) ⊂ polygon
    {oval, polygon] c regular
    {spring, boat) c irregular



About generalization rules

- The following generalization rules are particularly useful in learning systems [CoF83, Mic831]
  - If the concept F(v) holds for v when v is a constant a, or a constant b, and so on, then these concepts can be generalized into a statement that F(v) holds for every value of v.
  - F(a) AND F(b) .... GENERALIZED TO (|<) F(v) .
- Dropping conditions
  - red (v) AND apple (v) |< apple(v)
- Adding options
  - red(v) |< red(v) OR blue(v).
- Turning conjunction into disjunction.
  - red AND circle |< red OR circle
- Climbing a generalization tree
  - By ascending the generalization tree, the lower level concept is substituted for by the higher level concept.

Control Strategies in Learning from Examples

- Induction methods can be
  - bottom-up or top-down or combined
- In the data-driven methods, the presentation of the training examples drives the search.
  - These methods process the input examples one at a time, gradually generalizing the current set of concepts until a final conjunctive generalization is computed.
- In the model-driven methods, an a priori model is used to constrain the search.
  - These methods search a set of possible generalizations in an attempt to find a few "best" hypotheses that satisfy certain requirements.

- Data-driven techniques generally have the advantage of supporting incremental learning.
  - The learning process can start not only from the specific training examples, but also from the rules which have been discovered.
  - The learning systems are capable of updating the existing hypotheses to account for each new example.
- In contrast, the model-driven methods, which test and reject hypotheses based on an examination of the whole body of data, are difficult to be used in incremental learning situations.
  - When new training examples become available, model-driven methods must either backtrack or restart the learning process from the very beginning.
- Model-driven methods tend to have good noise immunity
- Noise impacts data driven methods.

- Several systems are being built (refer research paper)
- Example
  - If the weather is cloudy or raining, and the temperature is 40 to 60 degrees, the means of transportation should be a car.
  - [Transport = car] <= [Weather-type = cloudy V rain] A [Temp = 40..60]
- There are metrics to measure the strength of each rule (refer research paper)

Knowledge Discovery in Large Databases and Knowledges-Base

- There are efforts to build systems [KMK89]
- Suffers from computational inefficiency
- Just use database for retrieval purpose
- Follow tuple oriented approach.

Outline

- Introduction
- Learning from examples (Related Work)
- **Concepts of learning from databases**
- Attribute oriented induction in relational databases
- Variations of the learning algorithms
- Discussion
- Implementation and experiments
- Conclusions and future research

Primitives of learning from databases

- Primitives of Learning from Databases are characterized by a triple <D, C, A>, where
  - D represents the set of data in the database relevant to a specific learning task,
  - C represents a set of "concept biases" (generalization hierarchy, etc.) useful for defining particular concepts, and
  - A is a language used to phrase definition

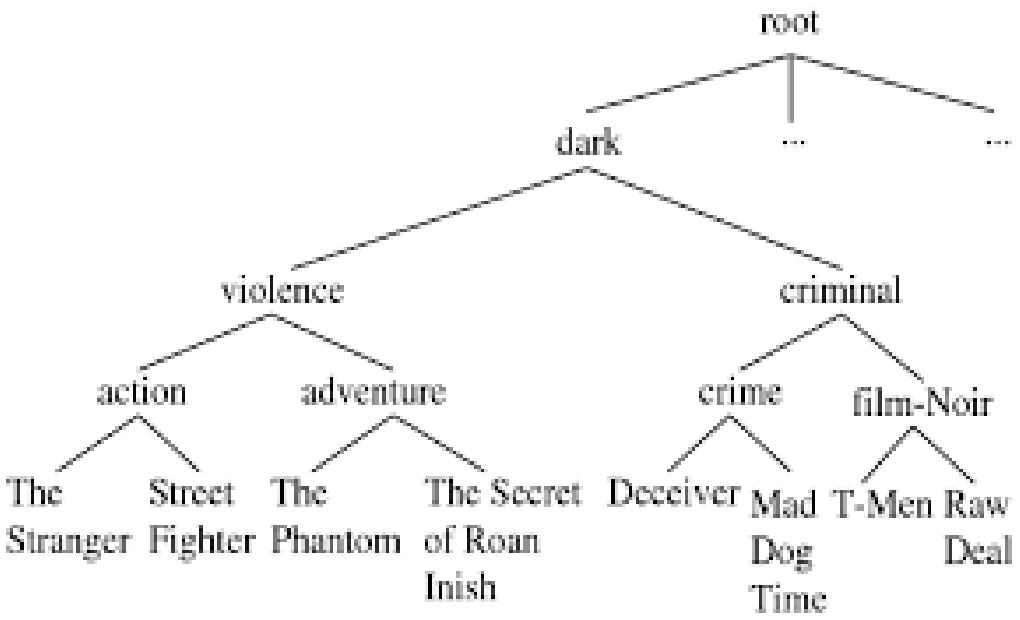
Data Relevant to Learning Task

- Characterize the specific learning task
  - For example, to characterize the features of graduate students, only the data relevant to graduate students are useful in the learning process.
- Data (D) should be obtained by performing relational operations, such as selection, projection and join.
- We assume that appropriate **preprocessing steps** are performed

- Most learning from examples algorithms partition the set of examples into positive and negative sets.
  - The positive examples are used to generalize the learning concepts, and the negative examples are used to specialize the learning concepts
- However, all of the data stored in the database which characterize the features of a property are positive data.
- However, most database applications assume that all of the information about a property is stored in the database.
  - For example, to distinguish graduate students from undergraduate students, the properties which belong to undergraduate students can be viewed as negative data.

Conceptual Bias Useful for Defining Concepts

- Assumption:
  - The conceptual bias is given for each attribute which is represented as a concept tree.
  - Such a concept tree is specified using an IS-A hierarchy and stored in a relation table, the concept hierarchy table.



Language Used to Phrase Definition

- We employ First-order predicate calculus

Name	Category	Major	Birth_Place	GPA
Jackson	senior	computing	Vancouver	3.5

can be viewed as representing the logic formula:

$$\exists\, t\ (\,(Name\,(t) = \text{Jackson}) \wedge (Category\,(t) = \text{senior}) \wedge (Major\,(t) = \text{computing})$$
$$\wedge\ (Birth_Place\,(t) = \text{Vancouver}) \wedge (GPA\,(t) = 3.5)\,).$$

- The intermediate and final learning results can also be represented using relational tables.
  - Definition: A generalized relation is a relation obtained by substituting the specific concept(s) by the general concept(s) in some attribute(s)
- The maximum number of tuples in a final generalized relation, can be specified by users as a threshold value of the learning process.

Two Types of Rules

- Characteristic rules and Classification rules can be learned
- Characteristic rules
  - Definition: A characteristic rule is an assertion which characterizes the concepts satisfied by all of the data stored in the database.
    - Example: For example, the symptoms of a specific disease can be summarized as a characteristic rule.
- Classification rules
  - Definition: A classification rule is an assertion which discriminates the concepts of one class from other classes.
    - Distinguish one disease from others, a classification rule should summarize the symptoms that discriminate this disease from others.
- A characteristic rule provides generalized concepts about a property which can help people recognize the common features of the data in a class.
- The classification rule gives a discriminant criterion which can be used to predict the class membership of new data.

- The characteristic rules only concern the characteristics of the data.
  - Positive examples alone are enough to furnish the learning task.
- For learning classification rules, the negative examples must be incorporated into the learning process to derive the concepts which have the discriminant property
  - For example, the data about students may be classified into graduate students and undergraduate students based on the value of the attribute "Category".
- About target class and contrasting class.
  - Definition: **A target class** is a class in which the data are tuples in the database consistent with the learning concepts.
  - Definition: **A contrasting class** is a class in which the data do not belong to the target class.
    - For instance, to distinguish graduate students from undergraduate students, the class of graduate students is the target class, and the class of undergraduate students is the contrasting class

Example

A relation Student in a sample university database

Name	Category	Major	Birth_Place	GPA
Anderson	M.A.	history	Vancouver	3.5
Bach	junior	math	Calgary	3.7
Carey	junior	liberal arts	Edmonton	2.6
Fraser	M.S.	physics	Ottawa	3.9
Gupta	Ph.D.	math	Bombay	3.3
Hart	sophomore	chemistry	Richmond	2.7
Jackson	senior	computing	Victoria	3.5
Liu	Ph.D.	biology	Shanghai	3.4
Meyer	sophomore	music	Burnaby	3.0
Monk	Ph.D.	computing	Victoria	3.8
Wang	M.S.	statistics	Nanjing	3.2
Wise	freshman	literature	Toronto	3.9

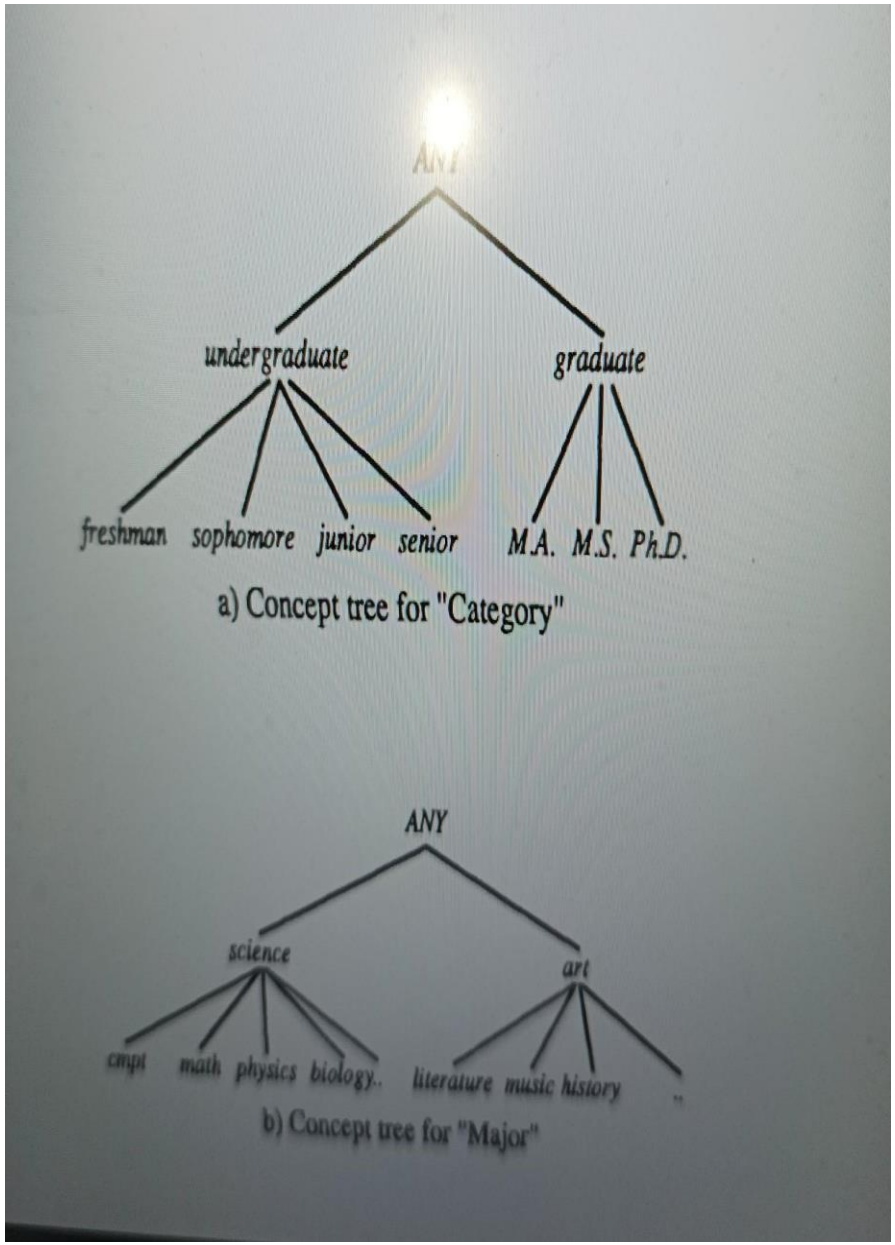
Learning Concept	Major	Birth_Place	GPA	Mark
graduate	art	B.C.	excellent	
	science	Ontario	excellent	
	science	B.C.	excellent	*
	science	India	good	
	science	China	good	
undergraduate	science	Alberta	excellent	
	art	Alberta	average	
	science	B.C.	average	
	science	B.C.	excellent	*
	art	B.C.	average	
	art	Ontario	excellent	

Table 4.4. A generalized relation

Concept Hierarchy C

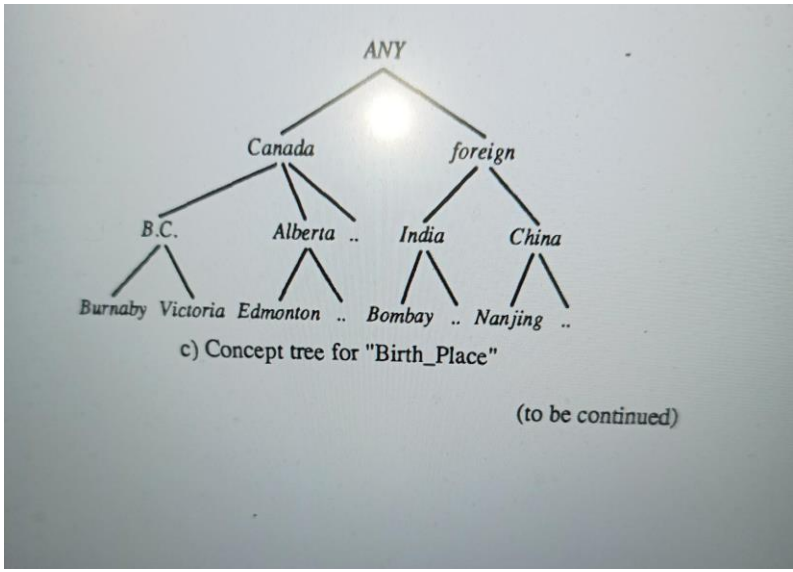
- Relationship between the values in category and "graduates" should be consulted in order to extract the relevant set of data.
- We then examine C, the conceptual bias. Consider the following C, where A c B indicates that B is a"generalization" of A.

- { computing, math, biology, chemistry, statistics,physics } c science
  (music, history, liberal arts, literature) c art
  { freshman, sophomore, junior, senior) c undergraduate
  M.S., M.A., Ph.D.) c graduate
  (Bumaby, Richmond, Vancouver,
  Victoria) c British Columbia
  { Calgary, Edmonton) c Alberta
  { Ottawa, Toronto) c Ontario
  (Bombay } c India
  [Shanghai, Nanjing) c China
  (China, India) c Foreign
  [British Columbia, Alberta, Ontario) c Canada
  { 2.0 - 2.9) c average
  [3.0 -3.4) c good
  (3.5 - 4.0 } c excellent



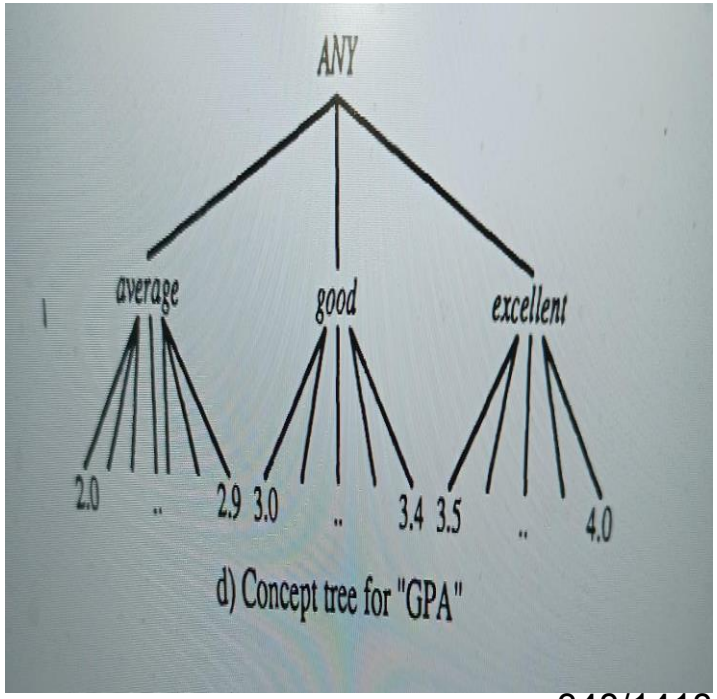
a) Concept tree for "Category"

b) Concept tree for "Major"



c) Concept tree for "Birth_Place"

(to be continued)



d) Concept tree for "GPA"

The set of data relevant to "graduates

Name	Category	Major	Birth_Place	GPA
Anderson	M.A.	history	Vancouver	3.5
Fraser	M.S.	physics	Ottawa	3.9
Gupta	Ph.D.	math	Bombay	3.3
Liu	Ph.D.	biology	Shanghai	3.4
Monk	Ph.D.	computing	Victoria	3.8
Wang	M.S.	statistics	Nanjing	3.2

Major	Birth_Place	GPA
ANY	Canada	excellent
science	Foreign	good

Table 3.3. The learned rule in relation table form

Outline

- Introduction
- Learning from examples (Related Work)
- Concepts of learning from databases
- **Attribute oriented induction in relational databases**
- Variations of the learning algorithms
- Discussion
- Implementation and experiments
- Conclusions and future research

Two algorithms: the LCHR algorithm and the LCLR algorithm

- Learning Characteristic Rules
  - Generalization Strategy 1 (Attribute Removal)
    - If there is a large set of distinct values for an attribute but there is no higher level concept provided for the attribute, the attribute should be removed during generalization.
    - Example: Name is the key of the relation and represents a large set of values so it should be removed.
    - Generalize to ALL or NULL
  - Generalization Strategy 2. (Generalization on the smallest decomposable components)
    - Generalization should be performed on the smallest decomposable components of a data relation.
    - Least commitment principle
      - Commitment to minimally generalized concepts
      - By generalizing lease decomposable attributes, we may discover interesting information about that components.

Two algorithms: the LCHR algorithm and the LCLR algorithm

- Learning Characteristic Rules
  - Generalization Strategy 3. (Ascending the concept tree)
    - If there are many distinct values for an attribute and there exists a higher level concept in the concept tree for the attribute, each value in the attribute of the relation should be substituted by a higher level concept in the learning process
  - Generalization Strategy 4. (Threshold control)
    - `If` the number of distinct values in a resulting relation is larger than the specified threshold value, further generalization on this attribute should be performed

Algorithm 4. LCHR -Learning characteristic rules from relational databases.

- Input: (i) a relational database,(ii) a concept hierarchy table,(iii) the learning task, and (iv) the threshold value (T).
- Output: A characteristic rule for the target class learned from the database
- Step 1. Select the task-relevant data relation P.
- Step 2. Perform attribute-oriented induction, which is described by the following procedure
  - Generalization is performed on each attribute of P.
- Step 3: Simplify the generalized relation
  - If only one attribute of several tuples contains distinct values, the several tuples can be reduced into one by taking the distinct values of that attribute as a set.
- Step 4: Transform the final relation into logic formulas

Learning Classification Rule

- Similar to the LCHR algorithm, the LCLR algorithms also applies the attribute-oriented induction technique.

- The difference is that in the extraction of classification rules, the facts which support the target class serve as positive examples, while the facts which support the other classes serve as negative examples.

- Since the learning task is to discover the concepts that have discriminant properties**, the portion of facts in the target class that overlaps with other classes should be detected and removed from the description of classification rules**

Overlapping rules

- Overlapping tuples
  Definition: A set of overlapping tuples is a set of tuples which are shared by different classes.

- Generalization Strategy 5. (Handling overlapping tuples)
  - If there are overlapping tuples in both target and contrasting classes, these tuples should be marked and eliminated from the final generalized relation

LCLR -Learning classification rules from relational database

- Input: (i) a relational database,(ii) a concept hierarchy table,(iii) the learning task, and (iv) the threshold value (T).
- Output: A classification rule for the target class learned from the database
- Step 1. Select the task-relevant data of the target class and the contrasting class to form relation P and cluster the data by classes.
- Step 2. Perform attribute-oriented induction, which is described by the following procedure
  - Generalization is performed on each attribute Ai of each class.
  - Perform intersection of both classes and mark the overlapping tuples;
- Step 3: Remove the overlapping tuples
- Step 4: Transform the final relation into logic formula

Outline

- Introduction
- Learning from examples (Related Work)
- Concepts of learning from databases
- Attribute oriented induction in relational databases
- **Variations of the learning algorithms**
- **Discussion**
- **Implementation and experiments**
- **Conclusions and future research**

Variations of the algorithms

- Adjusting Thresholds for Different Learning Results
  - The output vary based on the thresholds
- Dealing with Different Kinds of Concept Hierarchies
  - Balanced and imbalanced concept hierarchies
- Generalization of Concepts in the Hierarchies with Lattice
- Incorporating Quantitative Information
  - Addition of attribute “votes”, how many tuples have contributed to this rule?
- Handling Noisy Data and Exceptional Cases

DISCUSSION

- Limitations
  - Assumption:
    - availability of concept hierarchy data
    - Applicable on only formatted data

Experiments

- Refer research paper

Conclusion and Future work

- Conclusion
  - Our analysis of the algorithms demonstrates that the attribute-oriented induction approach substantially reduces the complexity of the database learning process.
- Future Work
  - Applications of Knowledge Rules Discovered from Relational Database
    - Discovery of knowledge rules for knowledge-base systems and expert systems
    - Processing of queries which involve abstract concepts.
    - Semantic query optimization using the learned rules
    - Construction of an Interactive Learning System
  - Discovery of Concept Hierarchies

Data Warehousing and OLAP Technology
Class #6

- Chapter 3 of the textbook

Detailed Syllabus

- Introduction (1.5 hour): Definition, KDD framework, Issues in data mining.
- Data summarization (7.5 hrs): Data Types, Preprocessing, Characterization, Discrimination, **data warehousing techniques (Data warehousing technology,** Data cube computation**)**
- Concepts and algorithms for mining patterns and associations (9 hours) (Frequent item-set generation, A priori and FP-growth algorithm, Evaluation of Association patterns) and preprocessing
- Concepts and algorithms related to classification and regression (9hrs) (Overview, Decision tree induction, Over-fitting and under-fitting, Scalable decision tree algorithms, Bayesian Classification, Regression-based Prediction methods (9 hours)
- Concepts and algorithms for clustering the data (9 hours) (Overview, Types of Data, K-means, Aglomerative clustering, Clustering algorithms (DBSCAN, BIRCH, CURE, ROCK, CHAMELEON)).
- Outlier analysis and future trends (graph mining, spatio-temporal mining). (3 hours)

- Data warehousing is the framework to generalize huge multi-dimensional databases

Data Warehousing and OLAP Technology

- Data Warehouse: Basic Concepts
- Data Warehouse Modeling: Data Cube and OLAP
- Data Warehouse Design and Usage
- Data Warehouse Implementation
- Data Generalization by Attribute-Oriented Induction
- Summary

What is 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*

Data Warehouse vs. Heterogeneous DBMS

- Traditional heterogeneous DB integration: A query driven approach
  - Build wrappers/mediators on top of heterogeneous databases
  - When a query is posed to a client site, a meta-dictionary is used to translate the query into queries appropriate for individual heterogeneous sites involved, and the results are integrated into a global answer set
  - Complex information filtering, compete for resources
- Data warehouse: update-driven, high performance
  - Information from heterogeneous sources is integrated in advance and stored in warehouses for direct query and analysis

Data Warehouse vs. Operational DBMS

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

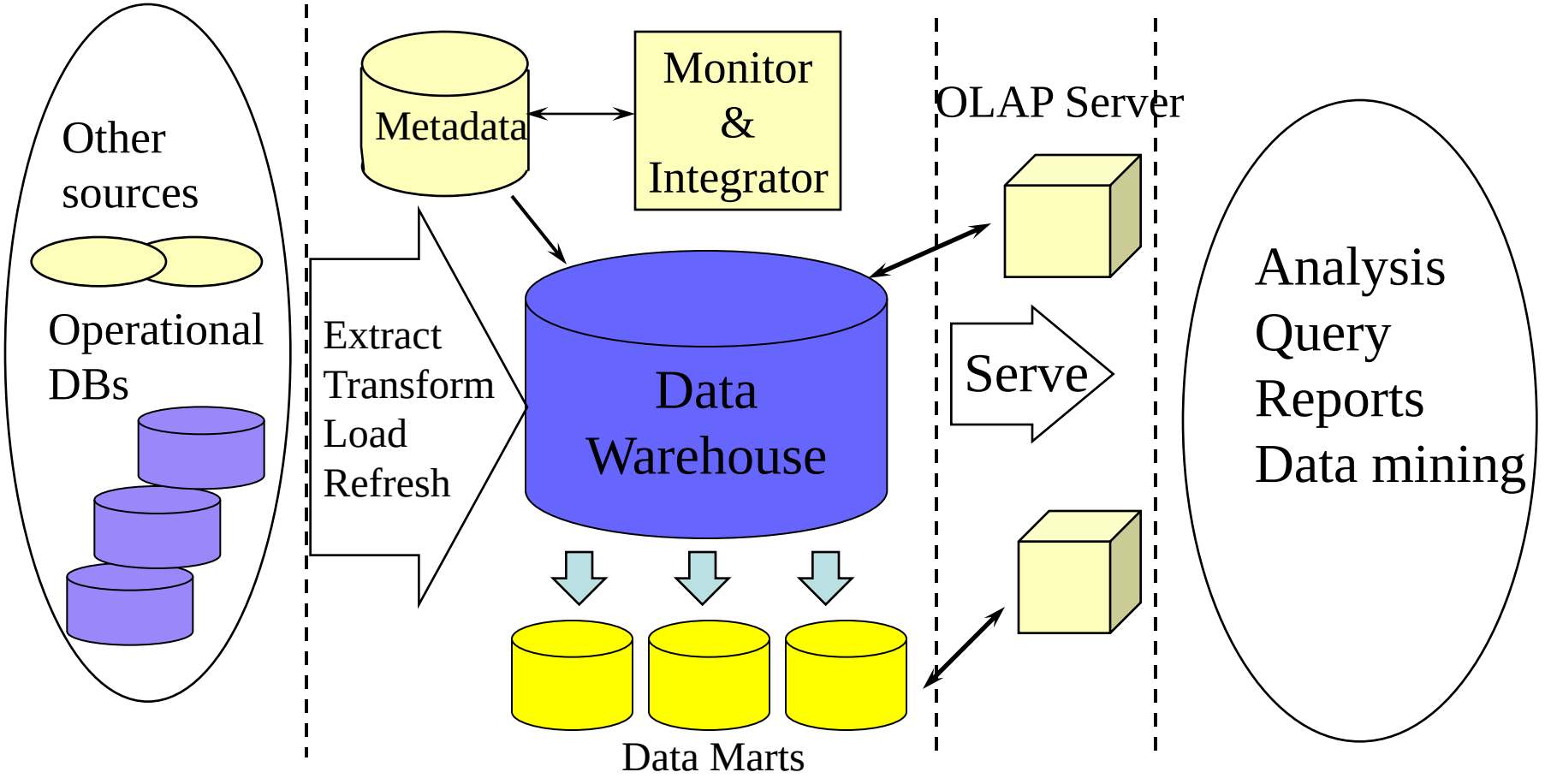
OLTP vs. OLAP

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

Why Separate Data Warehouse?

- High performance for both systems
  - DBMS— tuned for OLTP: access methods, indexing, concurrency control, recovery
  - Warehouse—tuned for OLAP: complex OLAP queries, multidimensional view, consolidation
- 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
- Note: There are more and more systems which perform OLAP analysis directly on relational databases

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

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

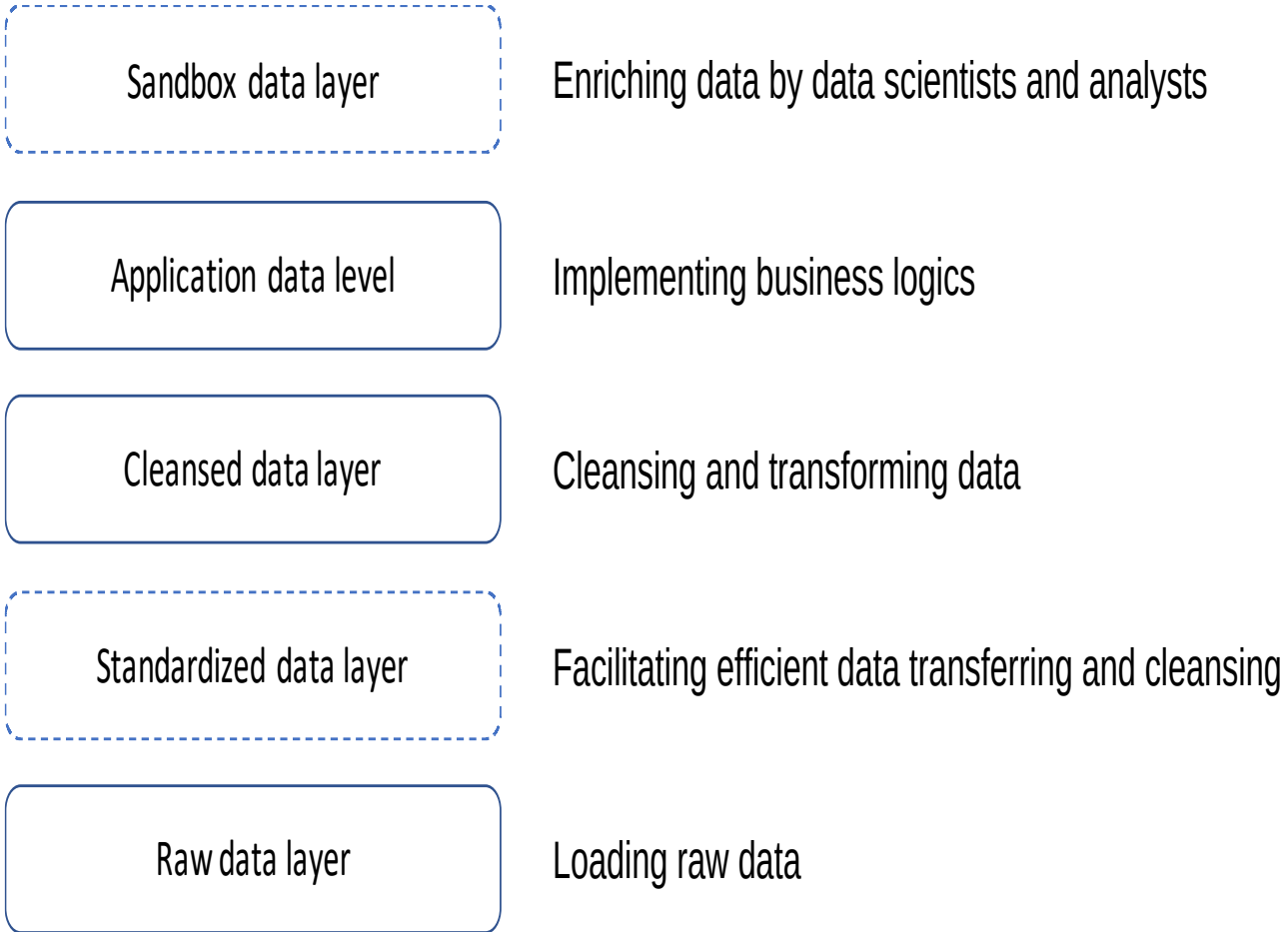
Metadata Repository

- **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 definitions, ownership of data, charging policies

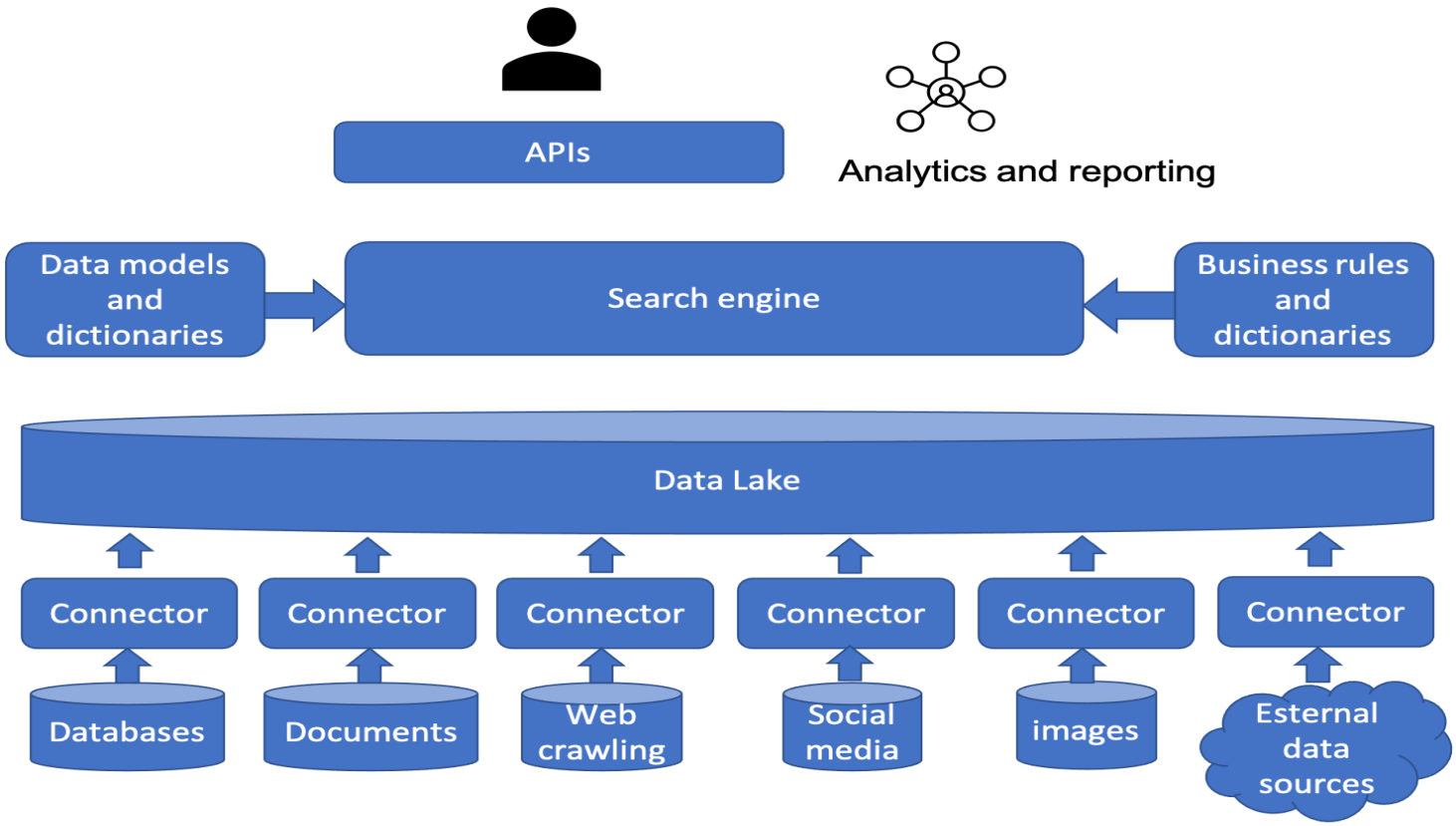
Data Lake

- A data lake is a centralized repository storing all structured and unstructured data at any scale in an organization
- Data is stored as is,
  - without having first to structure the data, and run different types of analytics—from dashboards and visualizations to big data processing, real-time analytics, and machine learning to guide better decision
- https://azure.microsoft.com/en-in/resources/cloud-computing-dictionary/what-is-a-data-lake

Layers of Storage



Conceptual Architecture



Two Purposes and Ecosystems

- Analytic use cases: decision support based on data
  - Data warehouses
  - Structured data, SQL/Python
- Operational use cases: data intelligence in applications
  - Data lakes
  - Raw data, Java/Scala, Python, R, and SQL
- Potential convergence of data warehouses and data lakes

Data Lakehouse

- A data lakehouse is an open standards-based storage solution that is multifaceted in nature.
  - It can address the needs of data scientists and engineers who conduct deep data analysis and processing and traditional data warehouse professionals who curate and publish data for business intelligence and reporting purposes.
  - The beauty of the lakehouse is that each workload can seamlessly operate on top of the data lake without having to duplicate the data into another structurally predefined database.
  - This ensures that everyone is working on the most up-to-date data while also reducing redundancies.

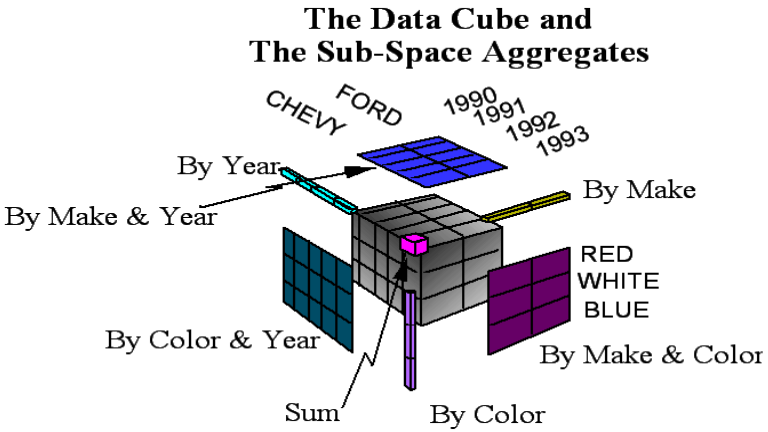
Data Warehousing and OLAP Technology: An Overview

- Data Warehouse: Basic Concepts
- **Data Warehouse Modeling: Data Cube and OLAP**
- Data Warehouse Design and Usage
- Data Warehouse Implementation
- Data Generalization by Attribute-Oriented Induction
- Summary

CUBE Operator

- Problems with GROUP BY
  - GROUP BY cannot directly construct:
    - Histograms
    - Roll-Up Reports
    - Cross-Tabs
- CUBE Operator
  - Generalize GROUP BY and Roll-Up and Cross-Tabs!!



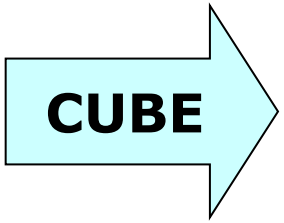


CUBE Operator
Think of the N-dimensional Cube

- N-dimensional Aggregate [sum(), max(),...]
  - Fits relational model exactly**:**
    - **a_1, a_2, ..., a_N, f()**
- Super-aggregate over *N-1* Dimensional sub-cubes
    - **ALL, a_2, ..., a_N , f()**
    - **a_3 , ALL, a_3, ..., a_N , f()**
    - **...**
    - **a_1, a_2, ..., ALL, f()**
  - This is the *N-1* Dimensional cross-tab**.**
- Super-aggregate over *N-2* Dimensional sub-cubes
    - **ALL, ALL, a_3, ..., a_N , f()**
    - **...**
    - **a_1, a_2 ,...., ALL, ALL, f()**
- …

CUBE Operator
An Example

SALES			
Model	Year	Color	Sales
Chevy	1990	red	5
Chevy	1990	white	87
Chevy	1990	blue	62
Chevy	1991	red	54
Chevy	1991	white	95
Chevy	1991	blue	49
Chevy	1992	red	31
Chevy	1992	white	54
Chevy	1992	blue	71
Ford	1990	red	64
Ford	1990	white	62
Ford	1990	blue	63
Ford	1991	red	52
Ford	1991	white	9
Ford	1991	blue	55
Ford	1992	red	27
Ford	1992	white	62
Ford	1992	blue	39



DATA CUBE			
Model	Year	Color	Sales
ALL	ALL	ALL	942
chevy	ALL	ALL	510
ford	ALL	ALL	432
ALL	1990	ALL	343
ALL	1991	ALL	314
ALL	1992	ALL	285
ALL	ALL	red	165
ALL	ALL	white	273
ALL	ALL	blue	339
chevy	1990	ALL	154
chevy	1991	ALL	199
chevy	1992	ALL	157
ford	1990	ALL	189
ford	1991	ALL	116
ford	1992	ALL	128
chevy	ALL	red	91
chevy	ALL	white	236
chevy	ALL	blue	183
ford	ALL	red	144
ford	ALL	white	133
ford	ALL	blue	156
ALL	1990	red	69
ALL	1990	white	149
ALL	1990	blue	125
ALL	1991	red	107
ALL	1991	white	104
ALL	1991	blue	104
ALL	1992	red	59
ALL	1992	white	116
ALL	1992	blue	110

- Think of ALL as a token representing the set
  - **{red, white, blue}**
  - **{1990, 1991, 1992}**
  - **{Chevy, Ford}**

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)
  - 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.

Table 4.2 2-D View of Sales Data for *AllElectronics* According to *time* and *item*

	location = "Vancouver"			
	item (type)			
time (quarter)	home entertainment	computer	phone	security
Q1	605	825	14	400
Q2	680	952	31	512
Q3	812	1023	30	501
Q4	927	1038	38	580

Note: The sales are from branches located in the city of Vancouver. The measure displayed is *dollars_sold* (in thousands).

Table 4.3 3-D View of Sales Data for *AllElectronics* According to *time*, *item*, and *location*

	location = "Chicago"				**location** = "New York"				**location** = "Toronto"				**location** = "Vancouver"			
	item				**item**				**item**				**item**			
time	home ent.	comp.	phone	sec.	home ent.	comp.	phone	sec.	home ent.	comp.	phone	sec.	home ent.	comp.	phone	sec.
Q1	854	882	89	623	1087	968	38	872	818	746	43	591	605	825	14	400
Q2	943	890	64	698	1130	1024	41	925	894	769	52	682	680	952	31	512
Q3	1032	924	59	789	1034	1048	45	1002	940	795	58	728	812	1023	30	501
Q4	1129	992	63	870	1142	1091	54	984	978	864	59	784	927	1038	38	580

Note: The measure displayed is *dollars_sold* (in thousands).

in this way, we may display any n-dimensional data as a series of $(n-1)$-dimensional

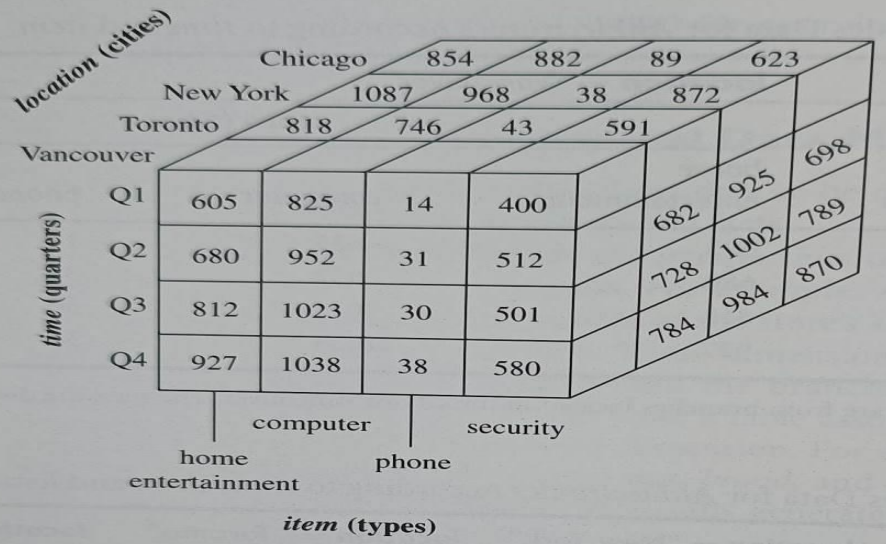


Figure 4.3 A 3-D data cube representation of the data in Table 4.3, according to *time*, *item*, and *location*. The measure displayed is *dollars_sold* (in thousands).

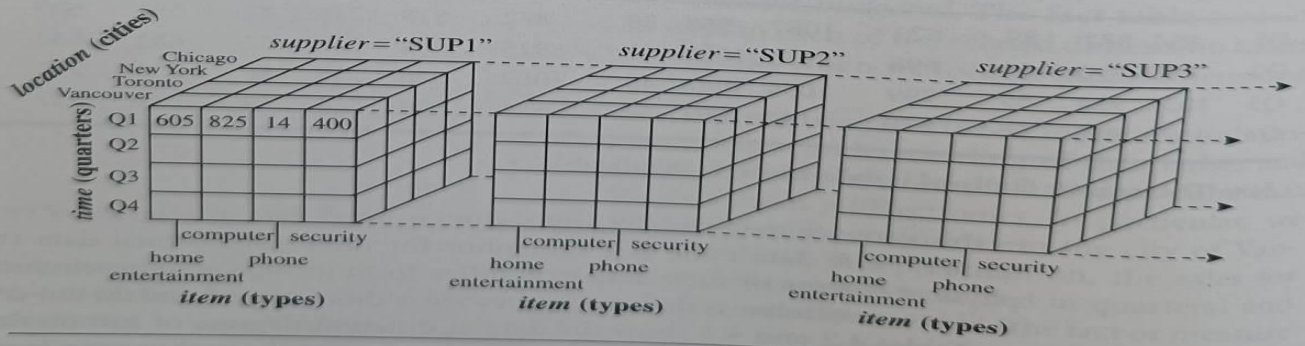
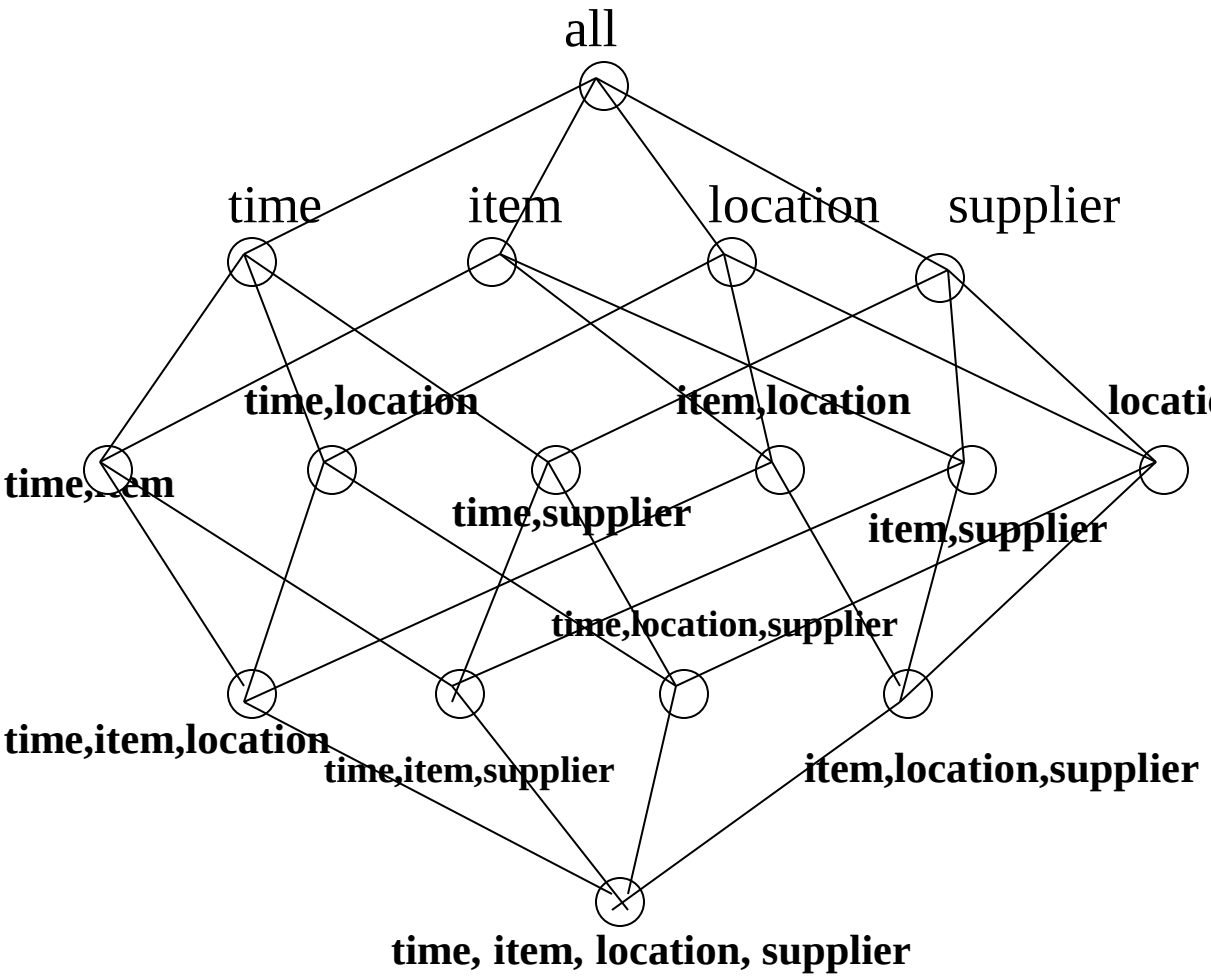


Figure 4.4 A 4-D data cube representation of sales data, according to *time*, *item*, *location*, and *supplier*. The measure displayed is *dollars_sold* (in thousands). For improved readability, only some of the cube values are shown.

and *location*, summarized for all suppliers. The 0-D cuboid, which

Cube: A Lattice of Cuboids



0-D(apex) cuboid

1-D cuboids

2-D cuboids

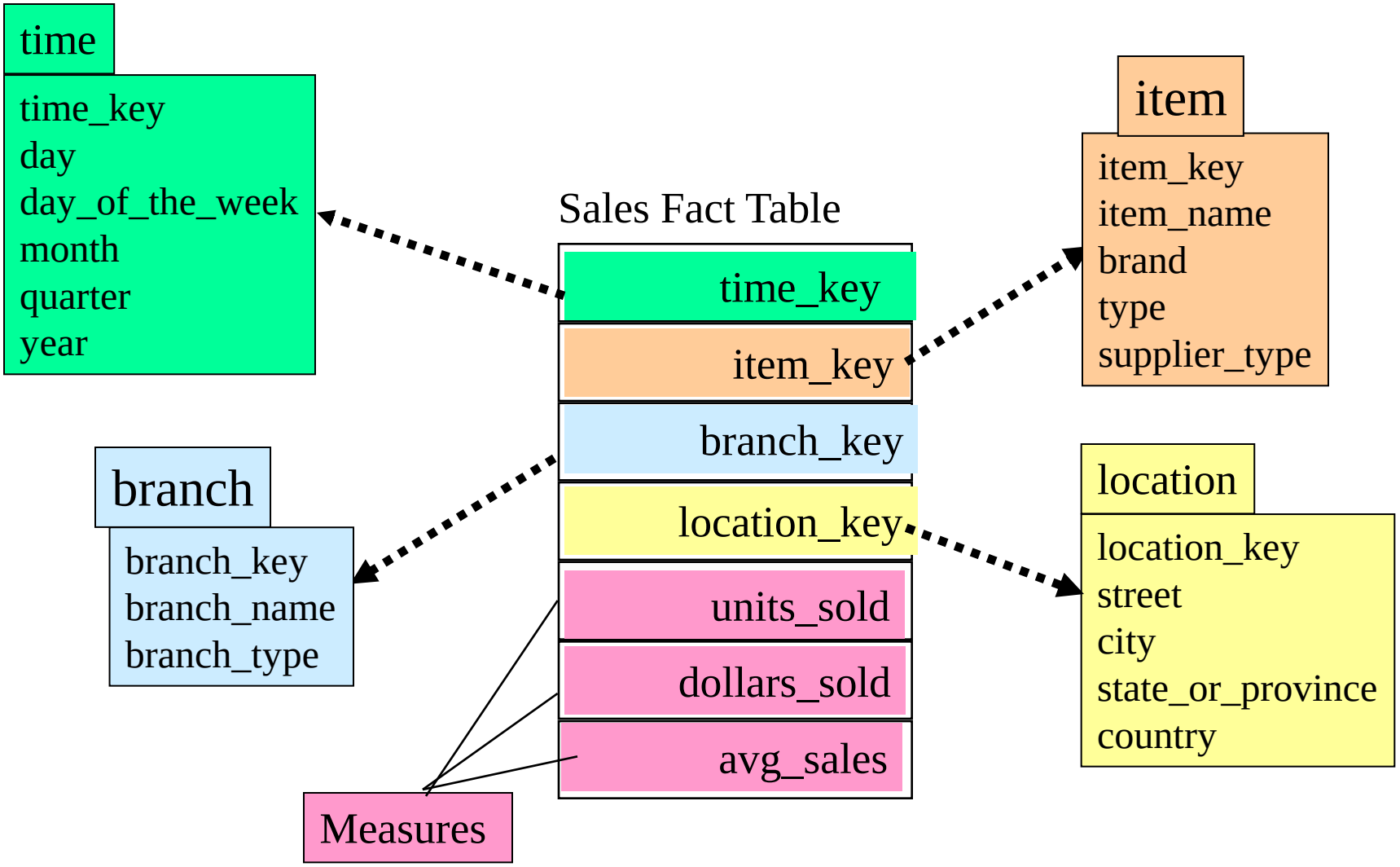
3-D cuboids

4-D(base) cuboid

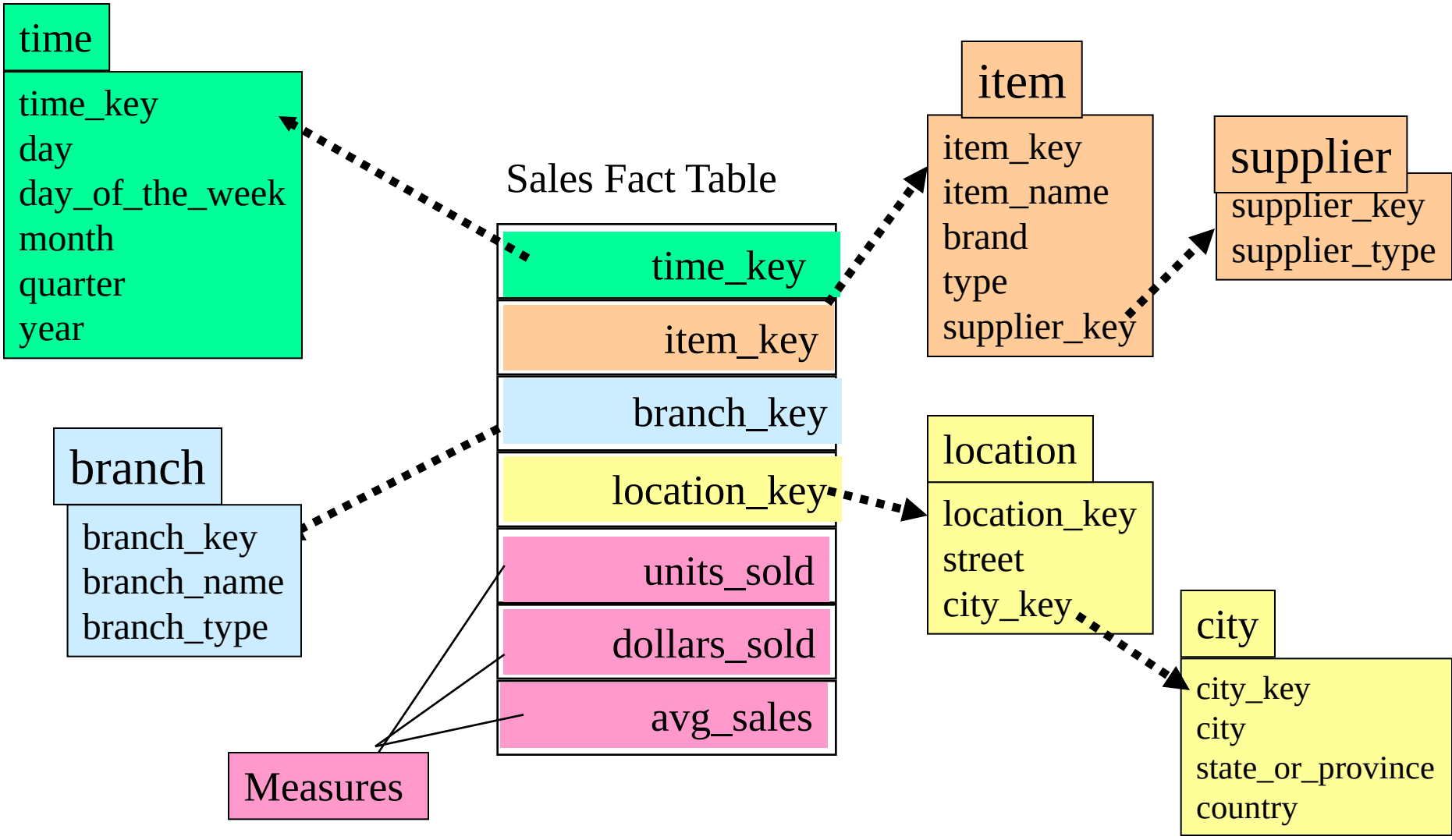
Conceptual Modeling of Data Warehouses

- 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

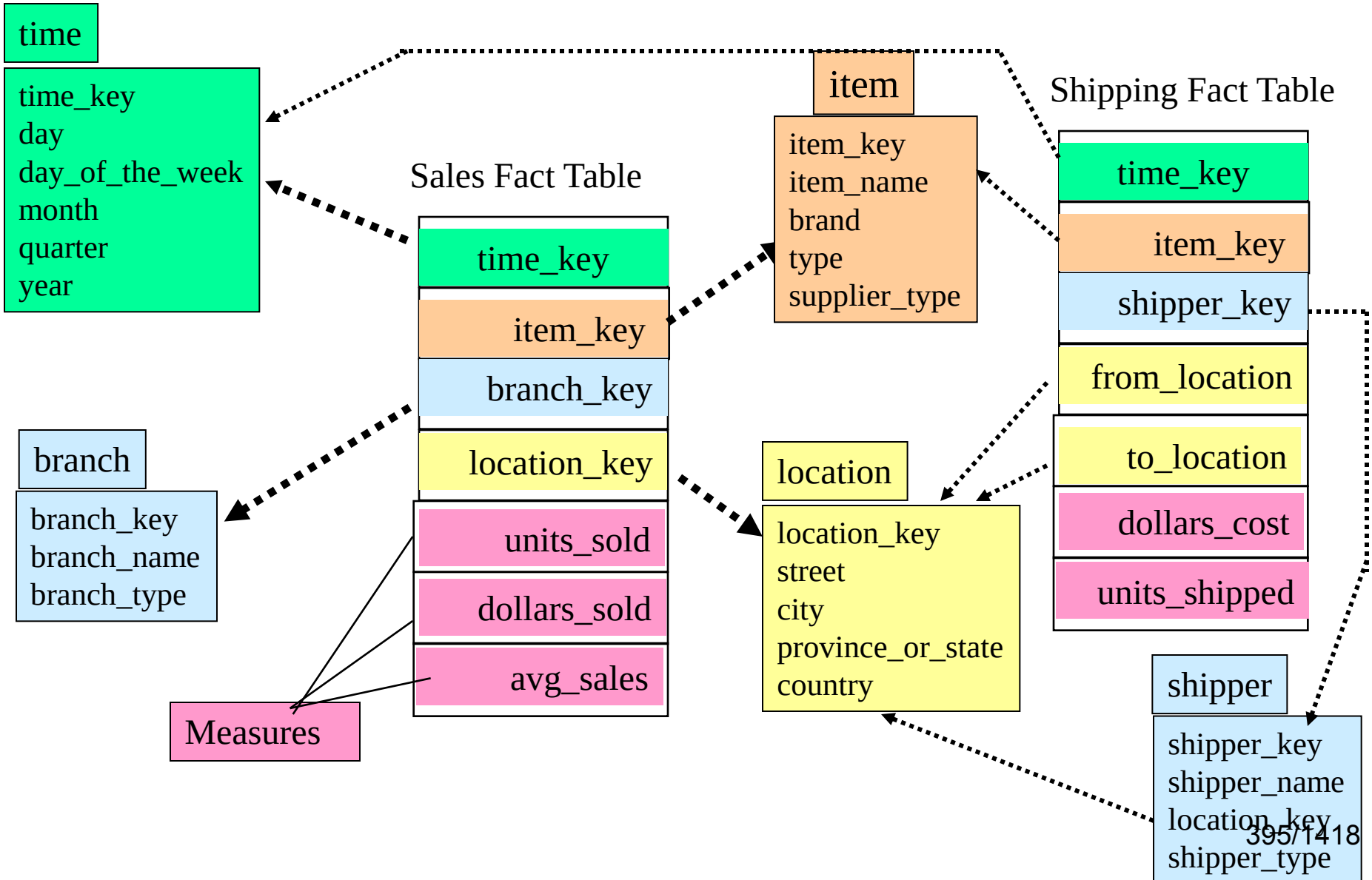
Example of Star Schema



Example of Snowflake Schema



Example of Fact Constellation



Cube Definition Syntax (BNF) in Data Mining Query Language (DMQL)

- Cube Definition (Fact Table)

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

- Dimension Definition (Dimension Table)

  define dimension <dimension_name> as
  (<attribute_or_subdimension_list>)

- Special Case (Shared Dimension Tables)
  - First time as “cube definition”
  - define dimension <dimension_name> as <dimension_name_first_time> in cube <cube_name_first_time>

Defining Star Schema in DMQL

define cube sales_star [time, item, branch, location]:
dollars_sold = sum(sales_in_dollars), avg_sales =
avg(sales_in_dollars), units_sold = count(*)

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

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

define dimension branch as (branch_key, branch_name, branch_type)

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

Defining Snowflake Schema in DMQL

define cube sales_snowflake [time, item, branch, location]:
dollars_sold = sum(sales_in_dollars), avg_sales = avg(sales_in_dollars),
units_sold = count(*)

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

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

define dimension branch as (branch_key, branch_name, branch_type)

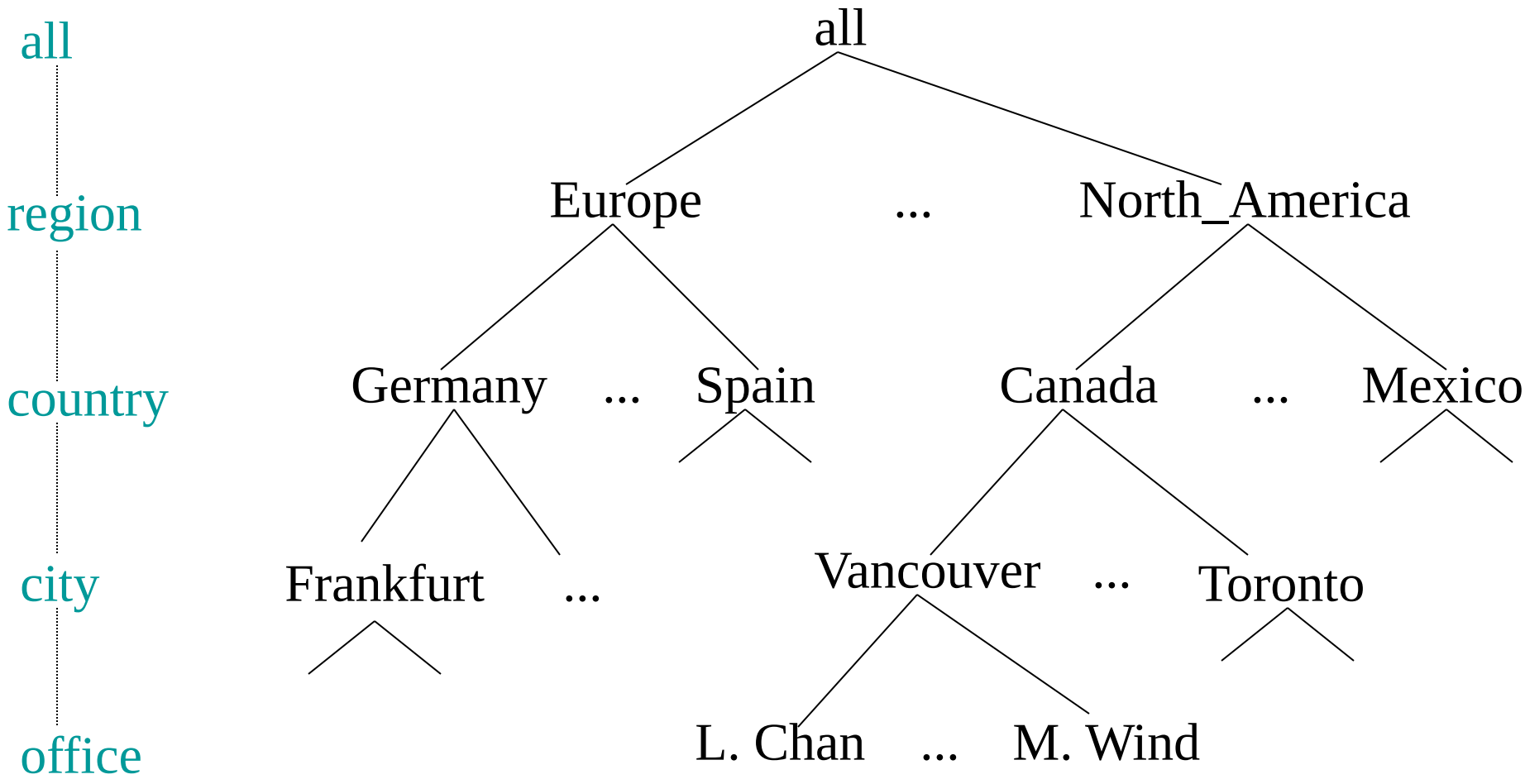
define dimension location as (location_key, street, city(city_key,
province_or_state, country))

Defining Fact Constellation in DMQL

```
define cube sales [time, item, branch, location]:
        dollars_sold = sum(sales_in_dollars), avg_sales = avg(sales_in_dollars), units_sold = count(*)
define dimension time as (time_key, day, day_of_week, month, quarter, year)
define dimension item as (item_key, item_name, brand, type, supplier_type)
define dimension branch as (branch_key, branch_name, branch_type)
define dimension location as (location_key, street, city, province_or_state, country)
define cube shipping [time, item, shipper, from_location, to_location]:
        dollar_cost = sum(cost_in_dollars), unit_shipped = count(*)
define dimension time as time in cube sales
define dimension item as item in cube sales
define dimension shipper as (shipper_key, shipper_name, location as location in cube sales,
    shipper_type)
define dimension from_location as location in cube sales
define dimension to_location as location in cube sales
```

A Concept Hierarchy: Dimension (location)

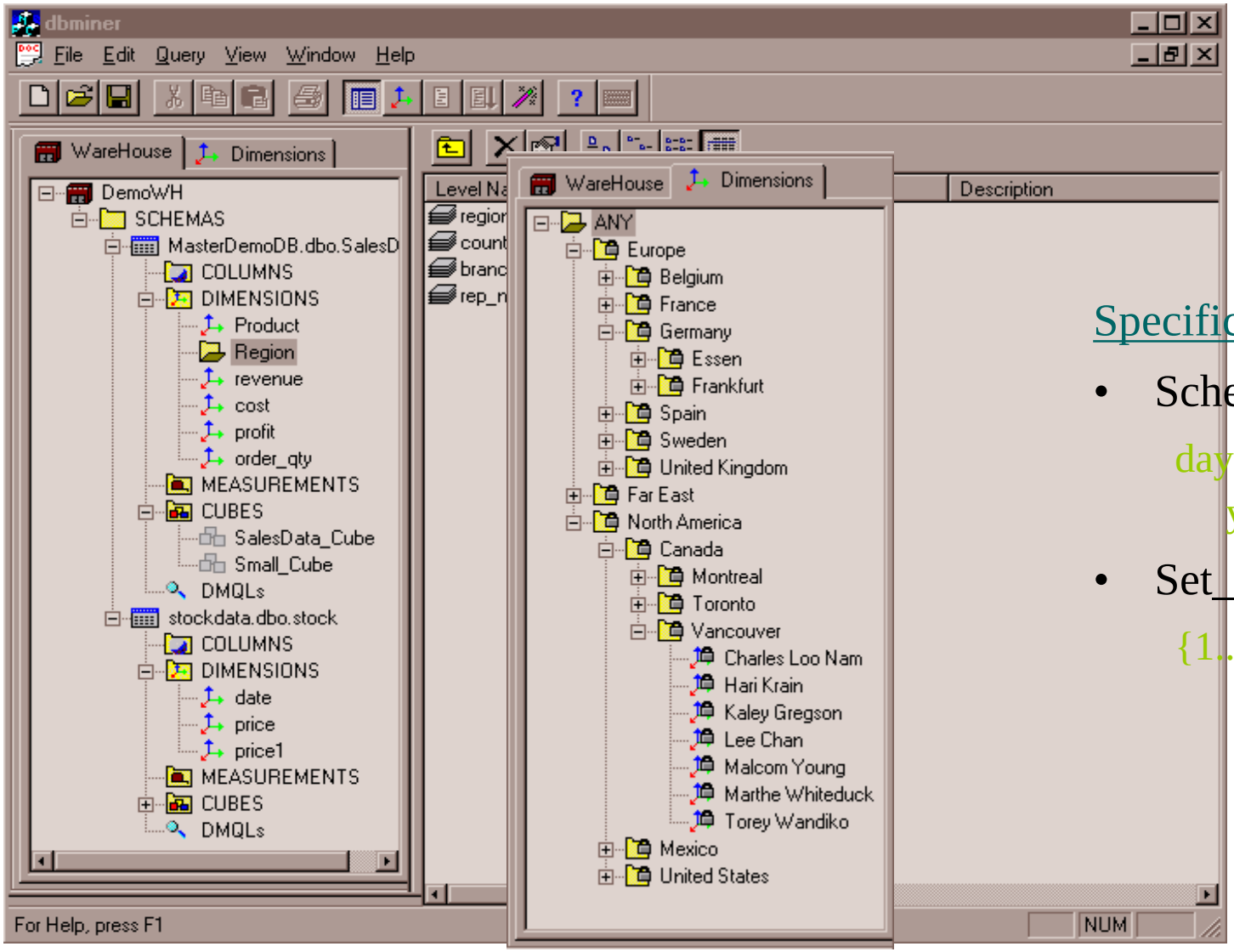
Definition: A concept hierarchy defines a sequence of mappings from a set of low-level concepts to higher-level, more general concepts.



Measures of Data Cube: Three Categories

- A measure in data cube is a numeric function which can be computed at each point in data space.
- 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()
    - Avg() can be computed by sum()/count() with two arguments, both sum() and count() are distributed aggregate functions.
- Holistic: if there is no constant bound on the storage size needed to describe a sub-aggregate.
    - E.g., median(), mode(), rank()

View of Warehouses and Hierarchies



Specification of hierarchies

- Schema hierarchy

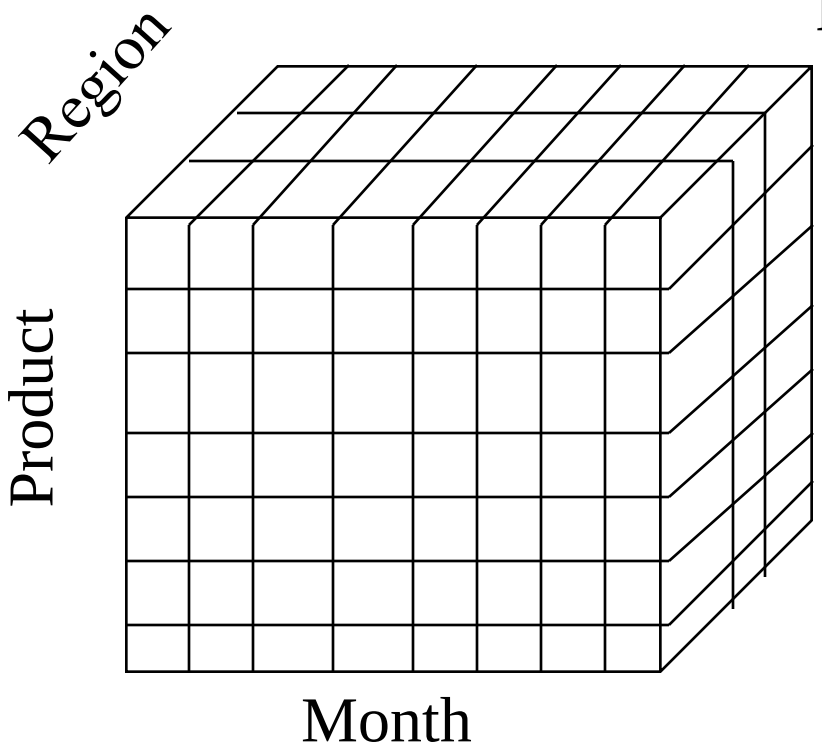
  day < {month < quarter; w
  year

- Set_grouping hierarchy

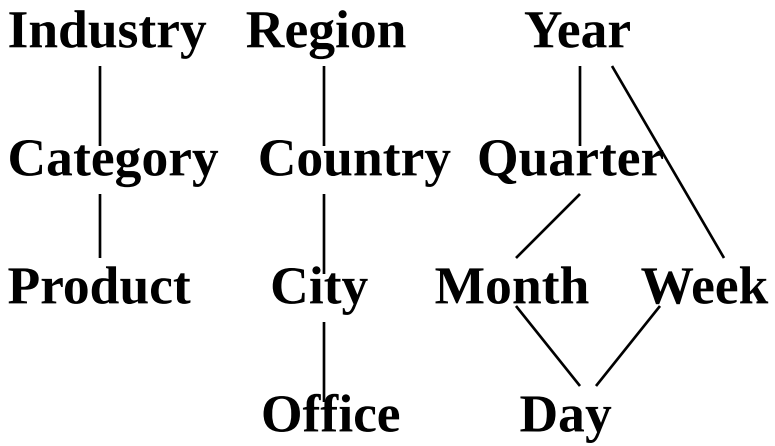
  {1..10} < inexpensive

Multidimensional Data

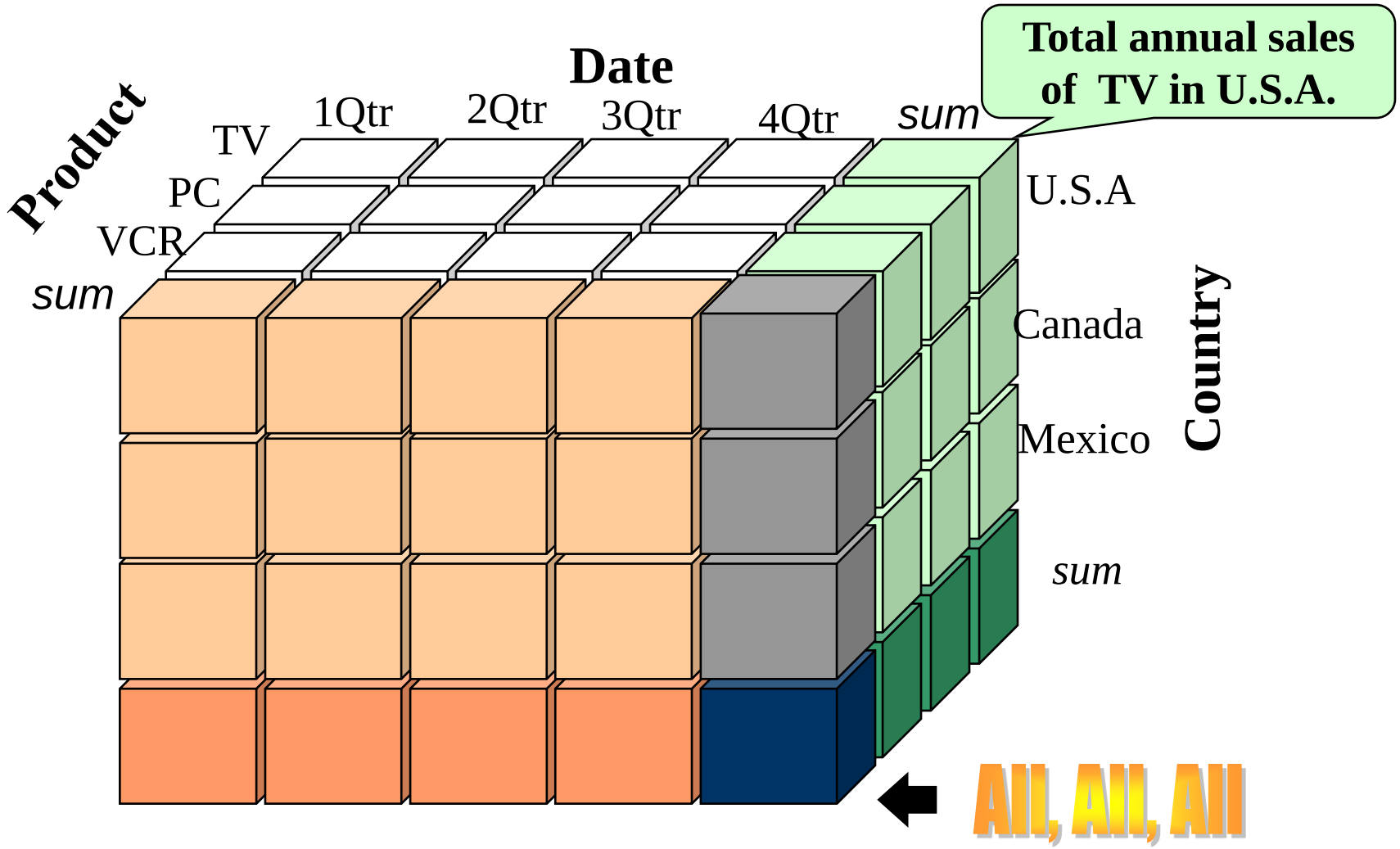
- Sales volume as a function of product, month, and region



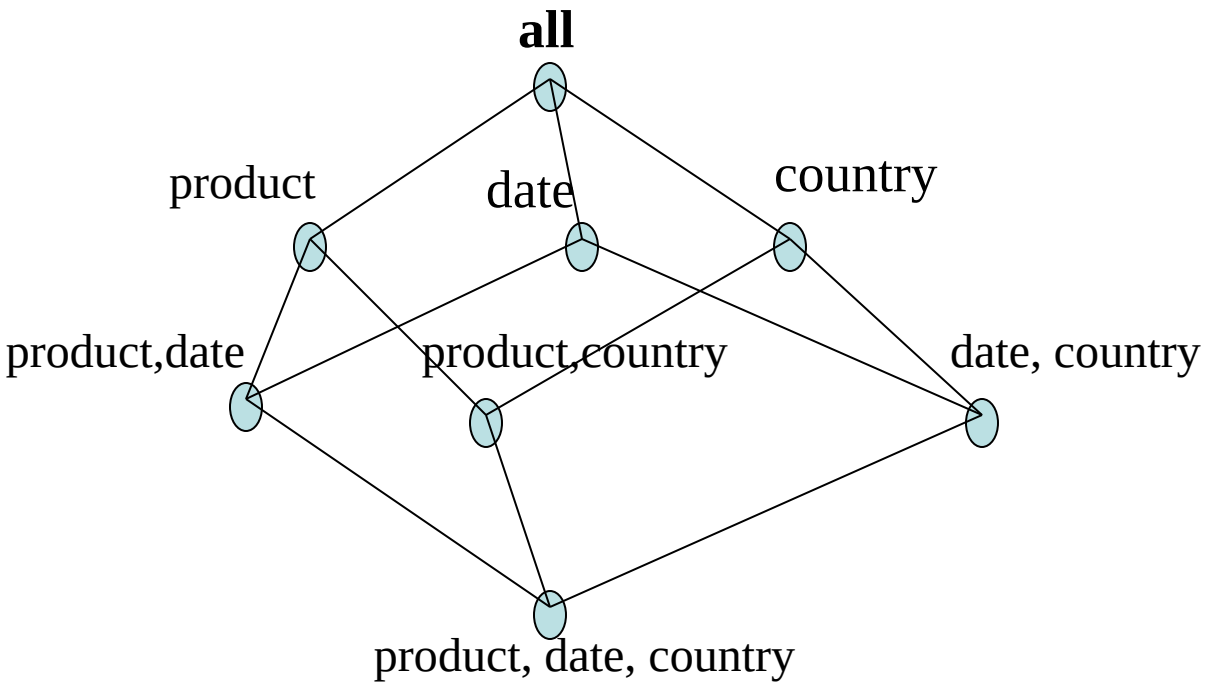
Dimensions: Product, Location, Time
Hierarchical summarization paths



A Sample Data Cube



Cuboids Corresponding to the Cube



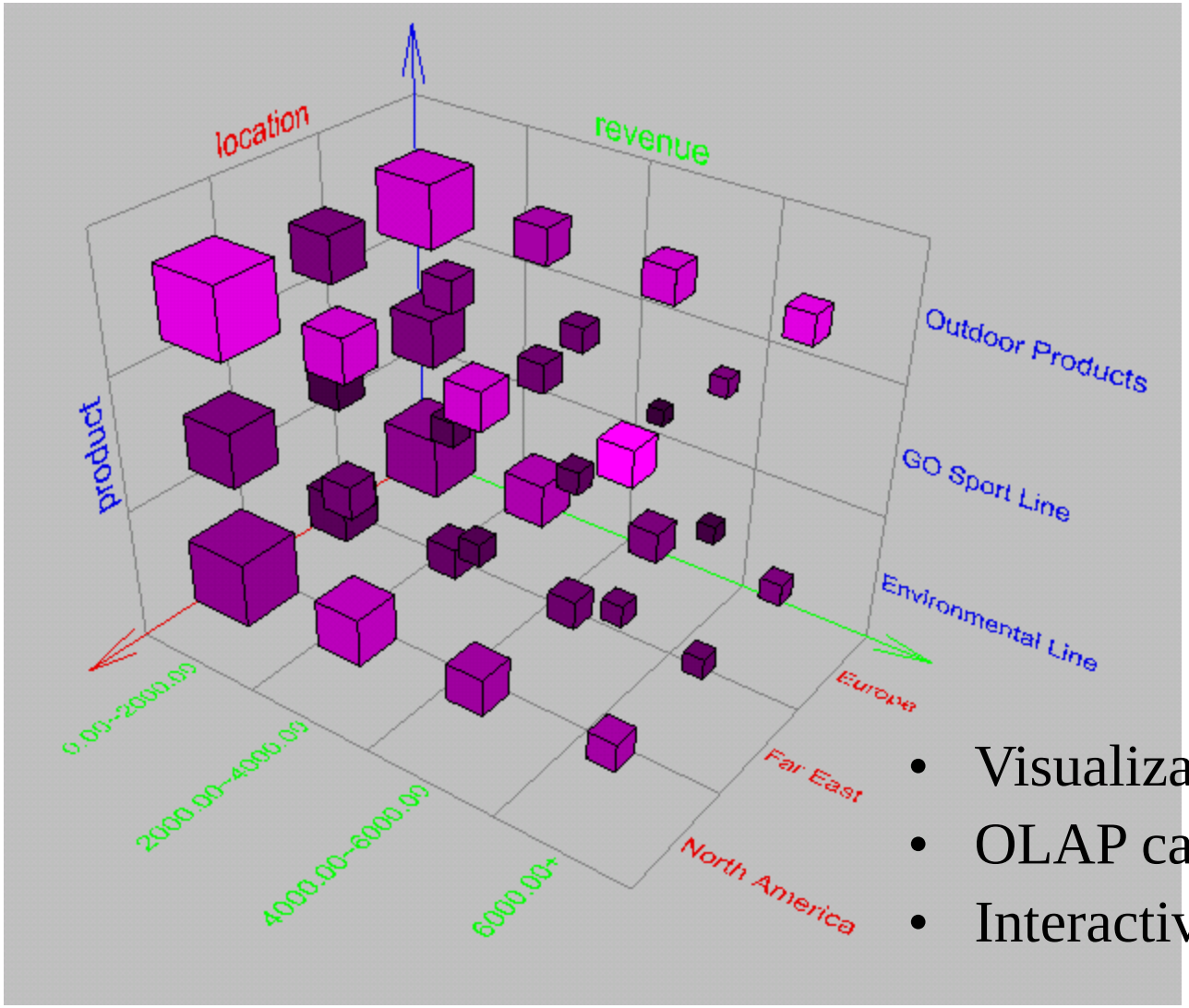
0-D(apex) cuboid

1-D cuboids

2-D cuboids

3-D(base) cuboid

Browsing a Data Cube



- Visualization
- OLAP capabilities
- Interactive manipulation

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
- Other operations
  - drill across: involving (across) more than one fact table
  - drill through: through the bottom level of the cube to its back-end relational tables (using SQL)

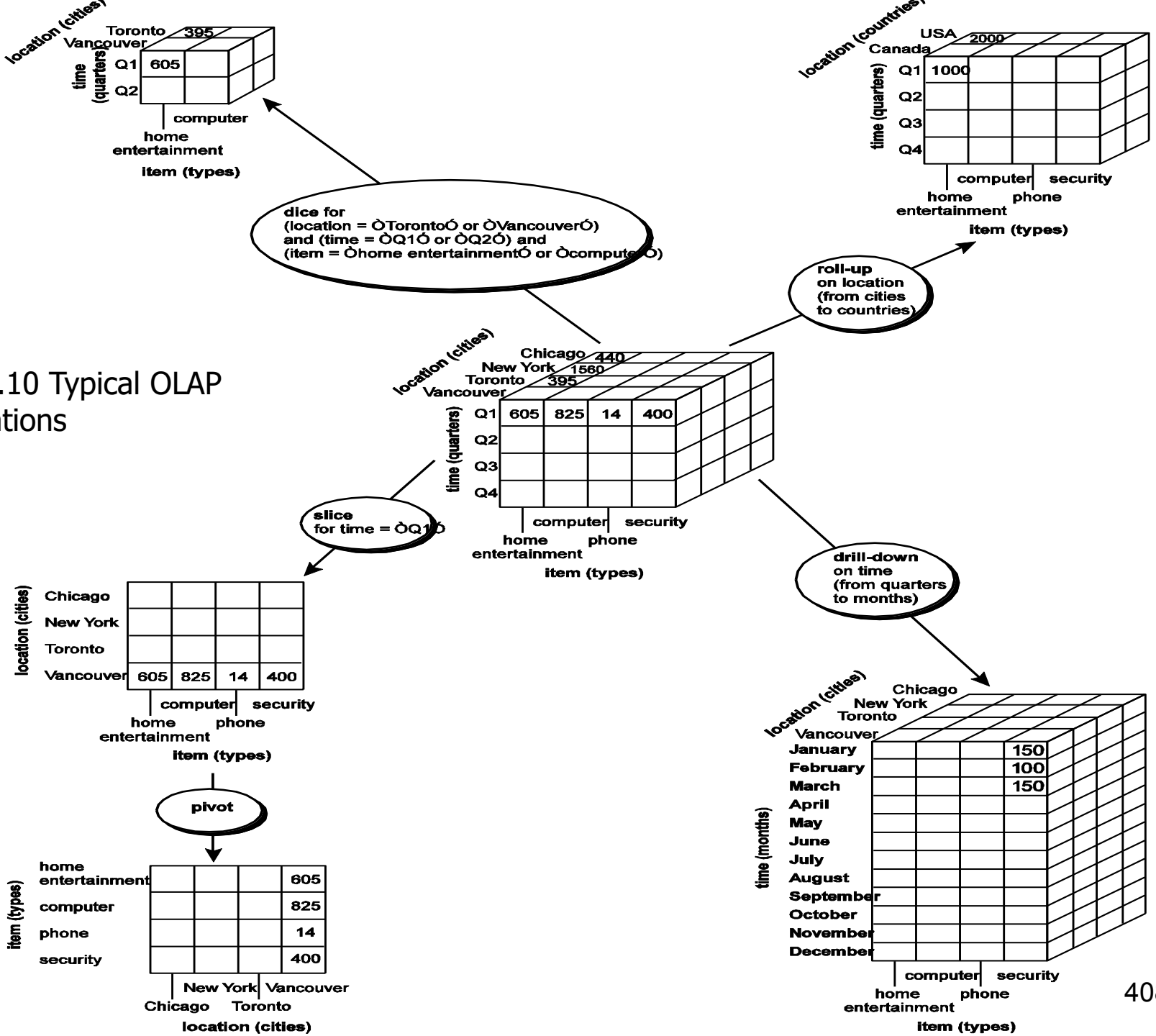
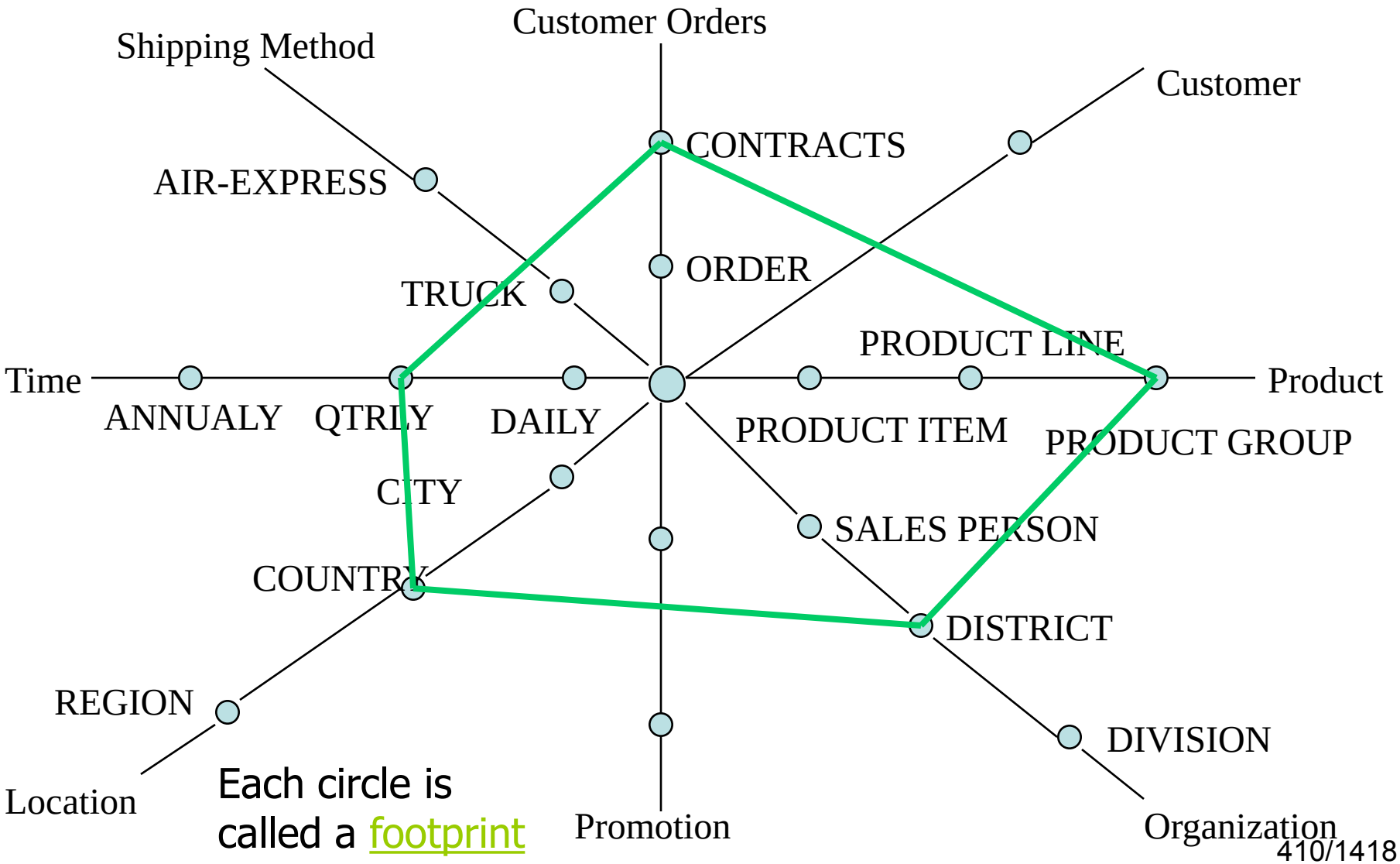


Fig. 3.10 Typical OLAP Operations

A Star-Net Query Model

- A Star-Net model consists of radial lines emanating from a central point, where each line represents a concept hierarchy of a dimension.
- Each abstraction level or a circle in the hierarchy is called **footprint**.
- It represents the granularities available for use by OLAP operations such as drill-down and roll-up.

A Star-Net Query Model



Each circle is called a footprint

Data Warehousing and OLAP Technology: An Overview

- Data Warehouse: Basic Concepts
- Data Warehouse Modeling: Data Cube and OLAP
- **Data Warehouse Design and Usage**
- Data Warehouse Implementation
- Data Generalization by Attribute-Oriented Induction
- Summary

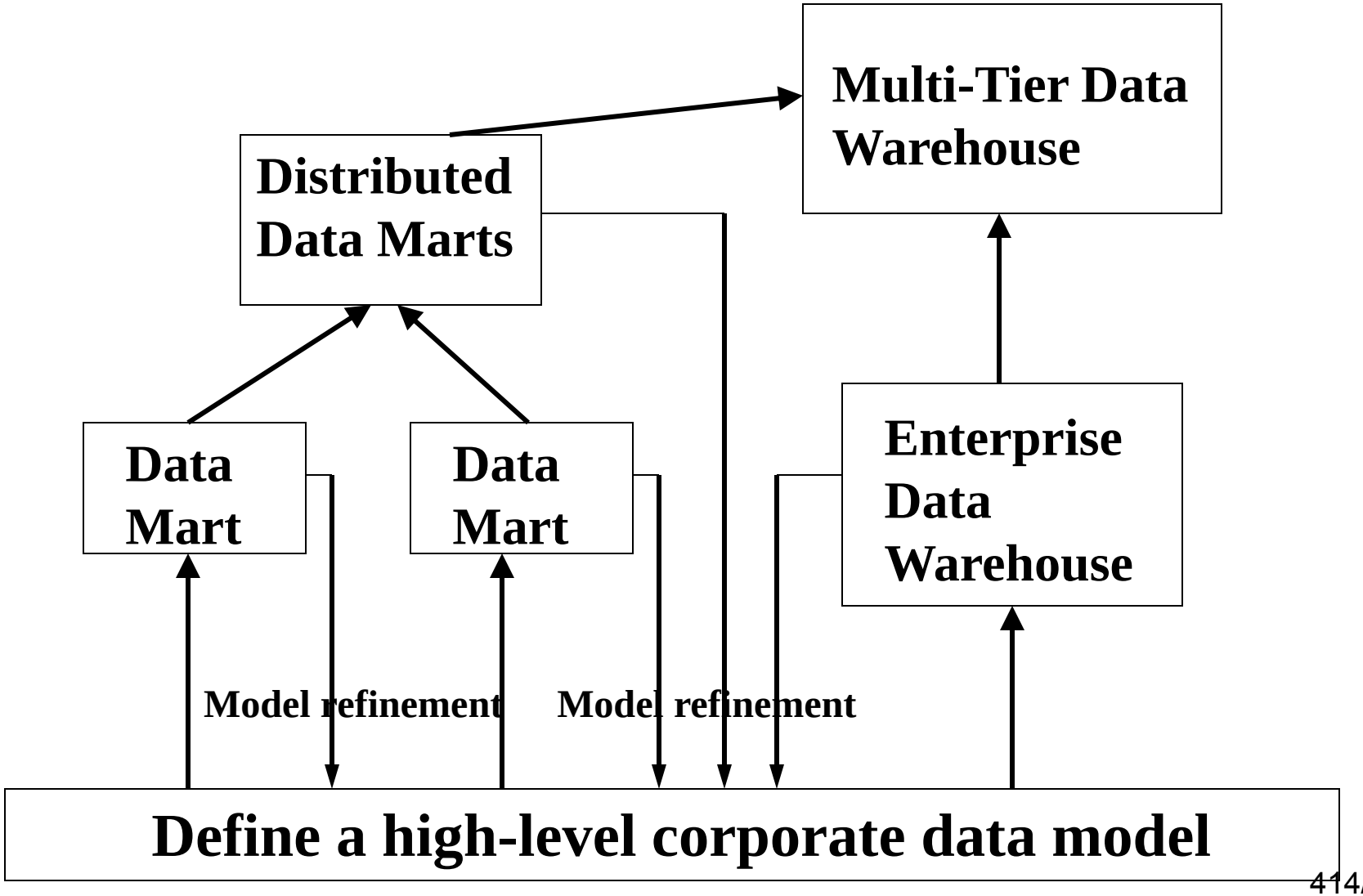
Design of Data Warehouse: A Business Analysis Framework

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

Data Warehouse Design Process

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

Data Warehouse Development: A Recommended Approach



Data Warehouse Usage

- Three kinds of data warehouse applications
  - Information processing
    - supports querying, basic statistical analysis, and reporting using crosstabs, tables, charts and graphs
  - Analytical processing
    - multidimensional analysis of data warehouse data
    - supports basic OLAP operations, slice-dice, drilling, pivoting
  - Data mining
    - knowledge discovery from hidden patterns
    - supports associations, constructing analytical models, performing classification and prediction, and presenting the mining results using visualization tools

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

- Why online analytical mining?
  - High quality of data in data warehouses
    - DW contains integrated, consistent, cleaned data
  - Available information processing structure surrounding data warehouses
    - ODBC (Open Database Connectivity), OLEDB (**Object Linking and Embedding, Database)**, Web accessing, service facilities, reporting and OLAP tools
  - OLAP-based exploratory data analysis
    - Mining with drilling, dicing, pivoting, etc.
  - On-line selection of data mining functions
    - Integration and swapping of multiple mining functions, algorithms, and tasks

Data Warehousing and OLAP Technology: An Overview

- Data Warehouse: Basic Concepts
- Data Warehouse Modeling: Data Cube and OLAP
- Data Warehouse Design and Usage
- **Data Warehouse Implementation**
- Data Generalization by Attribute-Oriented Induction
- Summary

Efficient Data Cube Computation

- Data cube can be viewed as a lattice of cuboids
  - The bottom-most cuboid is the base cuboid
  - The top-most cuboid (apex) contains only one cell
  - How many cuboids in an n-dimensional cube with L levels?

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

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

Cube Operation

- Cube definition and computation in DMQL

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

  compute cube sales

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

  SELECT item, city, year, SUM (amount)

  FROM SALES

  CUBE BY item, city, year

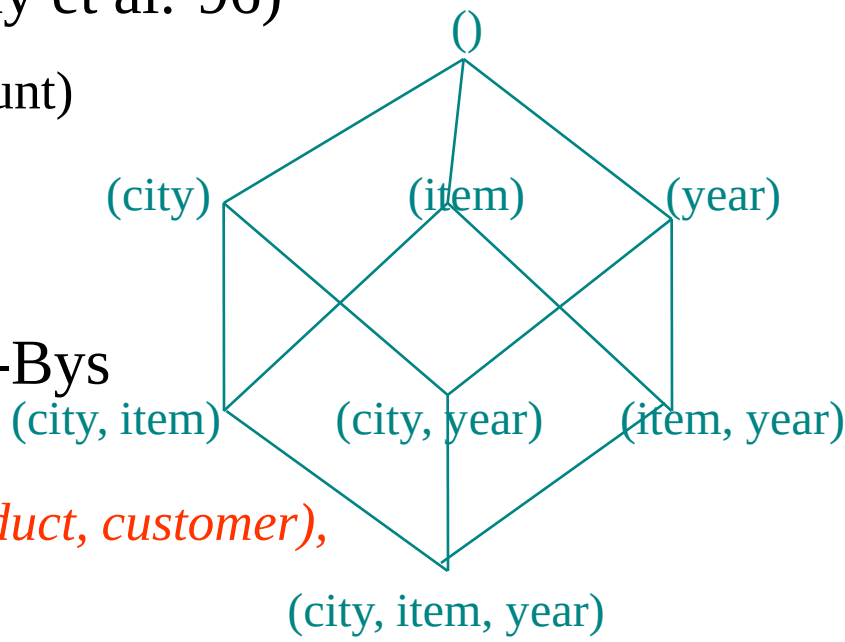
- Need compute the following Group-Bys

  (date, product, customer),

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

  (date), (product), (customer)

  ()



Bitmap indexes

- Assume that records have permanent numbers
- A bit-map index is a collection of bit vectors of length n, one for each value may appear in the field F.
- The vector for value v has 1 in position i if the i'th record has v in field F, and it has 0 there if not.
- Example for F and G fields:
  - (30, foo), (30,bar), (40, baz), (50, foo), (40, bar), (30, baz)
  - Bit index for F, each of 6 bits. For 30, it is 110001, for 40, it is 001010, and for 50, it is 000100.
  - Bit index for G also have three vectors. For "foo" it is, 100100, for "bar" it is 010010, and for "baz" it is 001001.

Bit map indexes: Partial match

- Bit maps allow answering of partial match queries quickly and efficiently.
- Example:
  - Movie(title, year, length, studioName)
  - SELECT title
  - FROM Movie
  - WHERE studioName= ‘Disney’ and Year=1965;
- If we have bitmap for studioName and year, then intersection or AND operation will give the result.
- Bit vectors do not occupy much space.

Bitmap indexes: range queries

- Example: consider the gold jewelry data of twelve points
  - 1 (25,60), 2(45,60), 3(50,75), 4(50,100), 5(50,120), 6(70, 110), 7(85,140), 8(30,260), 9(25,400), 10(45,350), 11(50,275), 12(60,260)
- Age has seven different values
  - 25(100000001000), 30(000000010000), 45(010000000100), 50(001110000010), 60(000000000001), 70(000001000000)
  - 85(000000100000)
- Salary has 10 different values
  - 60(000000000000), 75(001000000000), 100(000100000000)
  - 110(000001000000), 120(000010000000), 140(000000100000)
  - 260(000000010001), 275(000000000010), 350(000000000100)
  - 400(000000001000),

Example continued

- Find the jewelry buyers with an age range 45-55 and salary in the range 100-200
- Find the bit vectors of for the age values in the range and take OR
  - 010000000100 (for 45) and 001110000010 (for 50)
  - Result: 011110000110
- Find the bit vectors of salaries between 100 and 200 thousand.
  - There are four: 100,110,120, and 140, their bitwaise OR is 000111100000
- Take AND of both bit vectors
  - 000110000000
  - Find two records (50,100) and (50,120) are in the range.

Indexing OLAP Data: Bitmap Index

- Index on a particular column
- Each value in the column has a bit vector: bit-op is fast
- The length of the bit vector: # of records in the base table
- The *i*-th bit is set if the *i*-th row of the base table has the value for the indexed column
- not suitable for high cardinality domains
  - A bit compression technique, Word-Aligned Hybrid (WAH), makes it work for high cardinality domain as well [Wu, et al. TODS'06]

Base table

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

Index on Region

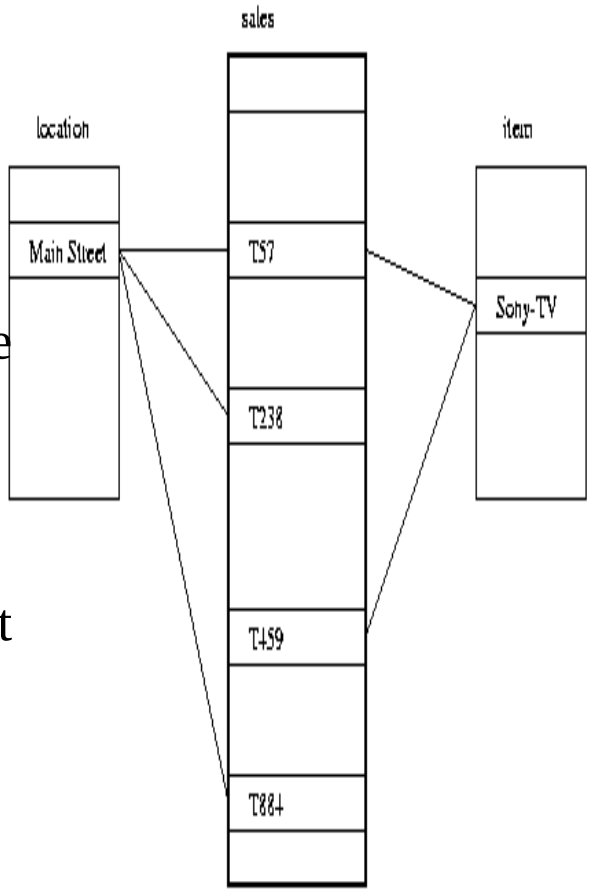
RecID	Asia	Europe	America
1	1	0	0
2	0	1	0
3	1	0	0
4	0	0	1
5	0	1	0

Index on Type

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

Indexing OLAP Data: Join Indices

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



Example of the join index

Fact: **store_sales**

Row id	...	ss_item_sk
1		1
2		1
3		2
4		5
5		3
6		1
7		2
8		4
9		6
10		5

Dimension: **item**

Row id	i_item_sk	i_item_name	...	i_item_color
1	1	TV		Black
2	2	Car		Yellow
3	3	IPhone		Black
4	4	Laptop		Green
5	5	Shirt		Black
6	6	Lamp		Brown

Index over store_sales.ss_item_sk:

Value	Row ids
1	1,2,6
2	3,7
3	5
4	8
5	4,10
6	9

Join index over item.i_item_color join key:
(store_sale.ss_item_sk=item.i_item_sk)

Value	Item sk	Row ids
Black	1,3,5 →	1,2,4,5,6,10
Yellow	2 →	3,7
Green	4 →	8
Brown	6 →	9

Efficient Processing OLAP Queries

- **Determine which operations** should be performed on the available cuboids
  - Transform drill, roll, etc. into corresponding SQL and/or OLAP operations, e.g., dice = selection + projection
- **Determine which materialized cuboid(s)** should be selected for OLAP op.
  - Let the query to be processed be on {*brand, province_or_state*} with the condition "*year = 2004*", and there are 4 materialized cuboids available:

    1) {*year, item_name, city*}

    2) {*year, brand, country*}

    3) {*year, brand, province_or_state*}

    4) {*item_name, province_or_state*} where *year = 2004*

    Which should be selected to process the query?
- Explore indexing structures and compressed vs. dense array structs in MOLAP

OLAP Server Architectures

- Relational OLAP (ROLAP)
  - Use relational or extended-relational DBMS to store and manage warehouse data and OLAP middle ware
  - Include optimization of DBMS backend, implementation of aggregation navigation logic, and additional tools and services
  - Greater scalability
- Multidimensional OLAP (MOLAP)
  - Sparse array-based multidimensional storage engine
  - Fast indexing to pre-computed summarized data
- Hybrid OLAP (HOLAP) (e.g., Microsoft SQLServer)
  - Flexibility, e.g., low level: relational, high-level: array
- Specialized SQL servers (e.g., Redbricks)
  - Specialized support for SQL queries over star/snowflake schemas

Data Warehousing and OLAP Technology: An Overview

- Data Warehouse: Basic Concepts
- Data Warehouse Modeling: Data Cube and OLAP
- Data Warehouse Design and Usage
- Data Warehouse Implementation
- **Data Generalization by Attribute-Oriented Induction**
- Summary

Attribute-Oriented Induction

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

Attribute-Oriented Induction: An Example

Example: Describe general characteristics of graduate students in the University database

- Step 1. Fetch relevant set of data using an SQL statement, e.g.,

  Select * (i.e., name, gender, major, birth_place, birth_date, residence, phone#, gpa)

  from student

  where student_status in {“Msc”, “MBA”, “PhD” }

- Step 2. Perform attribute-oriented induction
- Step 3. Present results in generalized relation, cross-tab, or rule forms

Class Characterization: An Example

Initial Relation

Name	Gender	Major	Birth-Place	Birth_date	Residence	Phone #	GPA
Jim Woodman	M	CS	Vancouver,BC, Canada	8-12-76	3511 Main St., Richmond	687-4598	3.67
Scott Lachance	M	CS	Montreal, Que, Canada	28-7-75	345 1st Ave., Richmond	253-9106	3.70
Laura Lee	F	Physics	Seattle, WA, USA	25-8-70	125 Austin Ave., Burnaby	420-5232	3.83
…	…	…	…	…	…	…	…
Removed	Retained	Sci,Eng, Bus	Country	Age range	City	Removed	Excl, VG,..

Prime Generalized Relation

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

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

Basic Principles of Attribute-Oriented Induction

- Data focusing: task-relevant data, including dimensions, and the result is the *initial relation*
- Attribute-removal: remove attribute *A* if there is a large set of distinct values for *A* but (1) there is no generalization operator on *A*, or (2) *A*'s higher level concepts are expressed in terms of other attributes
- Attribute-generalization: If there is a large set of distinct values for *A*, and there exists a set of generalization operators on *A*, then select an operator and generalize *A*
- Attribute-threshold control: typical 2-8, specified/default
- Generalized relation threshold control: control the final relation/rule size

Attribute-Oriented Induction: Basic Algorithm

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

Presentation of Generalized Results

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

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

Mining Class Comparisons

- Comparison: Comparing two or more classes
- Method:
  - Partition the set of relevant data into the target class and the contrasting class(es)
  - Generalize both classes to the same high level concepts
  - Compare tuples with the same high level descriptions
  - Present for every tuple its description and two measures
    - support - distribution within single class
    - comparison - distribution between classes
  - Highlight the tuples with strong discriminant features
- Relevance Analysis:
  - Find attributes (features) which best distinguish different classes

Concept Description vs. Cube-Based OLAP

- **Similarity**:
  - Data generalization
  - Presentation of data summarization at multiple levels of abstraction
  - Interactive drilling, pivoting, slicing and dicing
- **Differences**:
  - OLAP has systematic preprocessing, query independent, and can drill down to rather low level
  - AOI has automated desired level allocation, and may perform dimension relevance analysis/ranking when there are many relevant dimensions
  - AOI works on the data which are not in relational forms

Data Warehousing and OLAP Technology: An Overview

- Data Warehouse: Basic Concepts
- Data Warehouse Modeling: Data Cube and OLAP
- Data Warehouse Design and Usage
- Data Warehouse Implementation
- Data Generalization by Attribute-Oriented Induction
- **Summary**

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

References

- Refer to the Bibliographic notes of Chapter 3

- **Book: Jiawei Han, Jian Pei, Hanghang Tong, Data Mining: Concepts and Techniques, Fourth edition, 2022, Elseiver Inc.**

Surplus Slides

Compression of Bitmap Indices

- Bitmap indexes must be compressed to reduce I/O costs and minimize CPU usage—majority of the bits are 0's
- Two compression schemes:
  - Byte-aligned Bitmap Code (BBC)
  - Word-Aligned Hybrid (WAH) code
- Time and space required to operate on compressed bitmap is proportional to the total size of the bitmap
- Optimal on attributes of low cardinality as well as those of high cardinality.
- WAH out performs BBC by about a factor of two

Chapter 4: Data Cube Technology

P. Krishna Reddy, IIIT Hyderabad

Detailed Syllabus

- Introduction (1.5 hour): Definition, KDD framework, Issues in data mining.
- Data summarization (7.5 hrs): Data Types, Preprocessing, Characterization, Discrimination, data warehousing techniques (Data warehousing technology, **Data cube computation)**
- Concepts and algorithms for mining patterns and associations (9 hours) (Frequent item-set generation, A priori and FP-growth algorithm, Evaluation of Association patterns) and preprocessing
- Concepts and algorithms related to classification and regression (9hrs) (Overview, Decision tree induction, Over-fitting and under-fitting, Scalable decision tree algorithms, Bayesian Classification, Regression-based Prediction methods (9 hours)
- Concepts and algorithms for clustering the data (9 hours) (Overview, Types of Data, K-means, Aglomerative clustering, Clustering algorithms (DBSCAN, BIRCH, CURE, ROCK, CHAMELEON)).
- Outlier analysis and future trends (graph mining, spatio-temporal mining). (3 hours)

About OLAP and OLAM

- OLAP: Online Analytical Processing (OLAP)
  - Interactive analysis of multidimensional data at varied granularity levels
  - OLAP tools employ data cube and multidimensional data model to provide flexible access to summarized data
    - Drill-down (to see more specialized data) or roll-up (to see total sales)
- OLAM: exploratory multidimensional data mining or online analytical data mining
  - Searches for interesting patterns by exploring the data in multidimensional space.
  - Users can focus on a subset of dimensions by varying abstraction levels to find classification models, clusters, predictive rules and outliers.

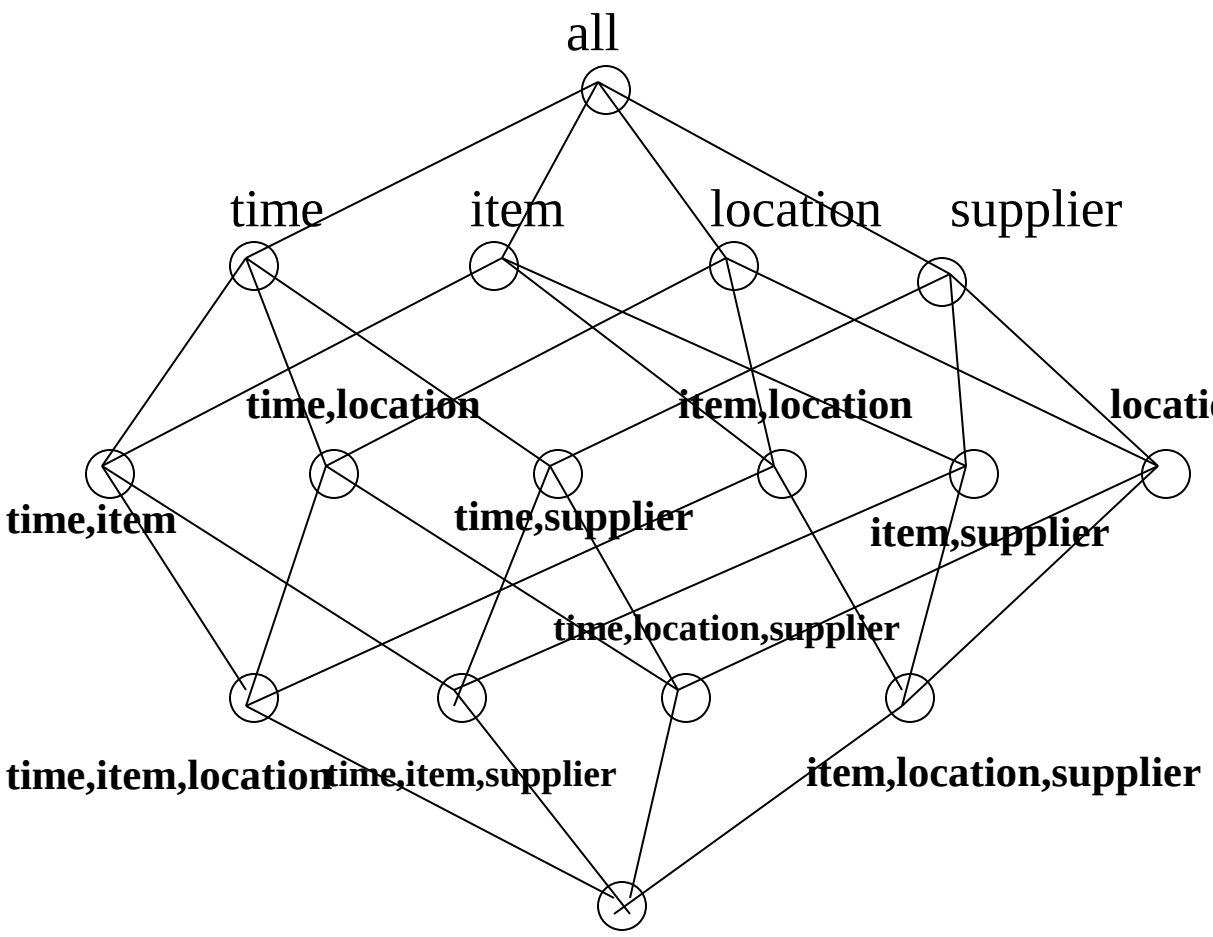
Outline

- **Data Cube Computation: Preliminary Concepts**
- Computing full/iceberg cubes: four methodologies
  - **1.**General heuristics: On the computation of multidimensional aggregates (Agarwal et al.'96)
  - 2.Bottom-Up: Multiway Array Aggregation for Full Cube Computation (Zhao et al SIGMOD'97)
  - **3**. Top-down: BUC: Computing Iceberg Cubes from the Apex Cuboid Downward (Beyer et al SIGMOD'99)
  - 4**.** Top-down: H-Cubing: Exploring an H-Tree Structure (Han et al SIGMOD'01)
- Overview of Other methodologies
    - 5.Star-cubing: Computing Iceberg Cubes Using a Dynamic Star-tree Structure (Xin et al VLDB'03)
    - 6.Precomputing Shell Fragments for Fast High-Dimensional OLAP (Li, et al. VLDB'04)
- Overview of Data Cubes for Advanced Applications
  - 7. Sampling Cubes: OLAP on Sampling Data (Li et al SIGMOD 08)
  - 8. Ranking Cubes: Efficient Computation of Ranking Queries (Xin et al VLDB06)
- Overview of Knowledge Discovery with Data Cubes
  - **9**. Prediction Cubes: Data Mining in Multi-Dimensional Cube Space (Chen et al VLDB'05)
  - 10**.** Complex Aggregation at Multiple Granularity: Multi-feature Cubes (Ross et al EDBT 1998)
  - 11**.** Discovery-Driven Data Cubes (Sarawagi et al EDBT 1998)
- Summary

Cube Materialization: Full Cube, Iceberg Cube, Closed Cube and Shell Cube

- Base Cell: A cell in the base cuboid.
- Aggregate cell: A cell from non-based cuboid is an aggregate.
- An aggregate cell aggregates over one or more dimensions.
  - Example: Let a=(a1,a2,...,an, measure) be a cell from one of the cuboids making-up a data cube. We say, "a" is a multi dimensional cell (from m-dimensional cuboid) if m (m<=n) values among {a1,a2,...,an} are not "*"
  - If m=n, then "a" is a base cell, otherwise it is an aggregate cell.
- Ancestor/descendant
  - 1-D cell is an ancestor of 2-D cell. 2-D cell is an ancestor of 3-D cell.

Data Cube: A Lattice of Cuboids



0-D(apex) cuboid

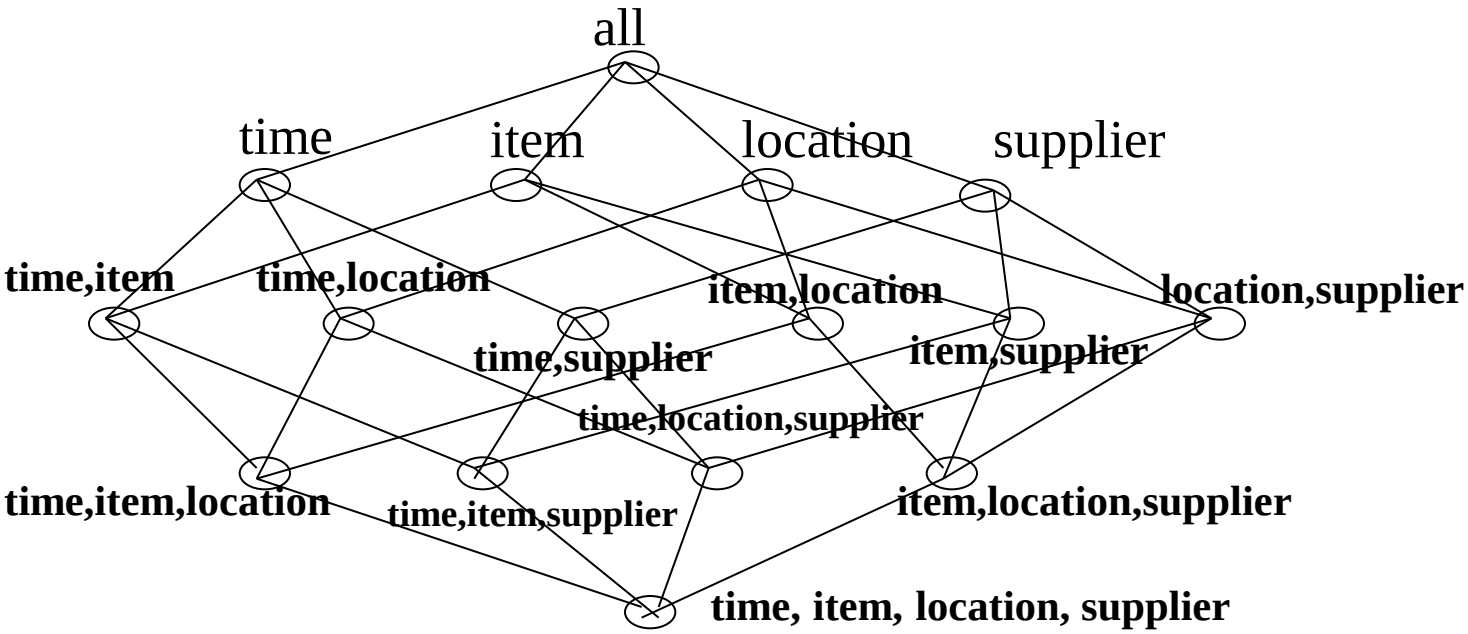
1-D cuboids

2-D cuboids

3-D cuboids

4-D(base) cuboid

Data Cube: A Lattice of Cuboids



0-D(apex) cuboid

1-D cuboids

2-D cuboids

3-D cuboids

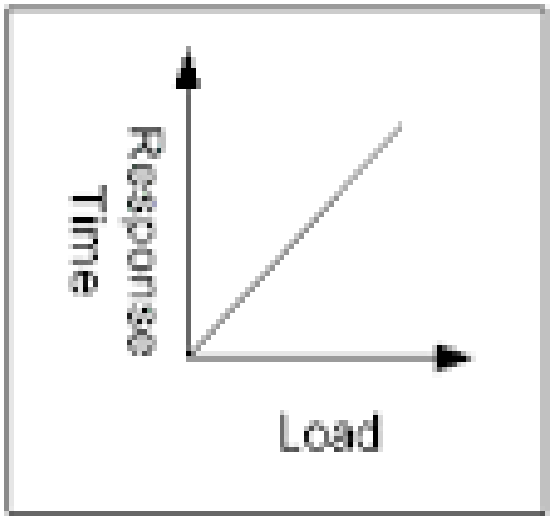
4-D(base) cuboid

- Base vs. aggregate cells; ancestor vs. descendant cells; parent vs. child cells
  1. (9/15, milk, Urbana, Dairy_land)
  2. (9/15, milk, Urbana, *)
  3. (*, milk, Urbana, *)
  4. (*, milk, Urbana, *)
  5. (*, milk, Chicago, *)
  6. (*, milk, *, *)

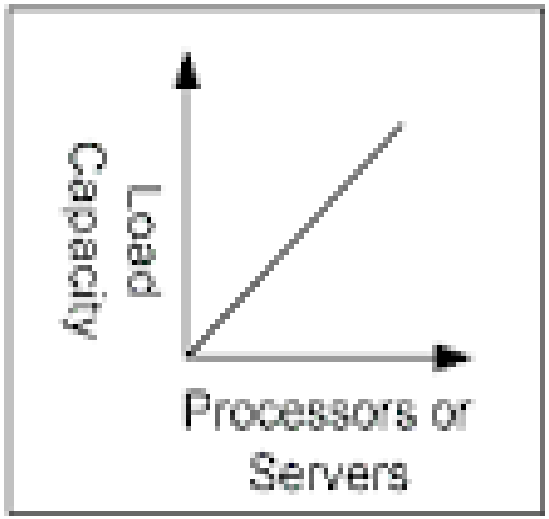
About the computation of data cube

- **Full cube**: all the cells of all cuboids for a given data cube
- Computation of full cuboid is of exponential complexity.
  - A data cube of n dimensions contains 2^n cuboids
  - More cuboids, if we consider concept hierarchy
  - The size of each cuboid depends on the cardinality of the dimension.
  - Precomputation of each cube requires huge and excessive amount of memory.
- It is important to explore the scalable methods.
  - memory consumed
  - Size of the computed data
  - time required for computation

About scalability



Linear Scalability Relative to Load



Linear Scalability Relative to System Resources (e.g., hardware)

Issue: Developing scalable algorithms

About Iceberg cubes

- Several cubes may not be interesting
  - For may cuboids, the aggregate might be zero or of less value.
- Sparse cuboid: the product of cardinalities of dimensions is large relative to the number of non-zero valued tuples.
- Sparse cube: if the cube contains many sparse cuboids.
- So, it is better to materialize only those cells above some minimum threshold.
- Such partially materialized cubes are known as iceberg cubes.
- By materializing only few cells in the data cube, the result is seen as a tip of iceberg.

Cube Materialization: Full Cube vs. Iceberg Cube



- Full cube vs. iceberg cube

```
compute cube sales iceberg as
select month, city, customer group, count(*)
from salesInfo
cube by month, city, customer group
having count(*) >= min support
```

iceberg condition

- Computing *only* the cuboid cells whose measure satisfies the iceberg condition
- Only a small portion of cells may be "above the water'' in a sparse cube
- Avoid explosive growth: A cube with 100 dimensions
  - 2 base cells: (a1, a2, …., a100), (b1, b2, …, b100)
  - How many aggregate cells if "having count >= 1"?
  - What about "having count >= 2"?

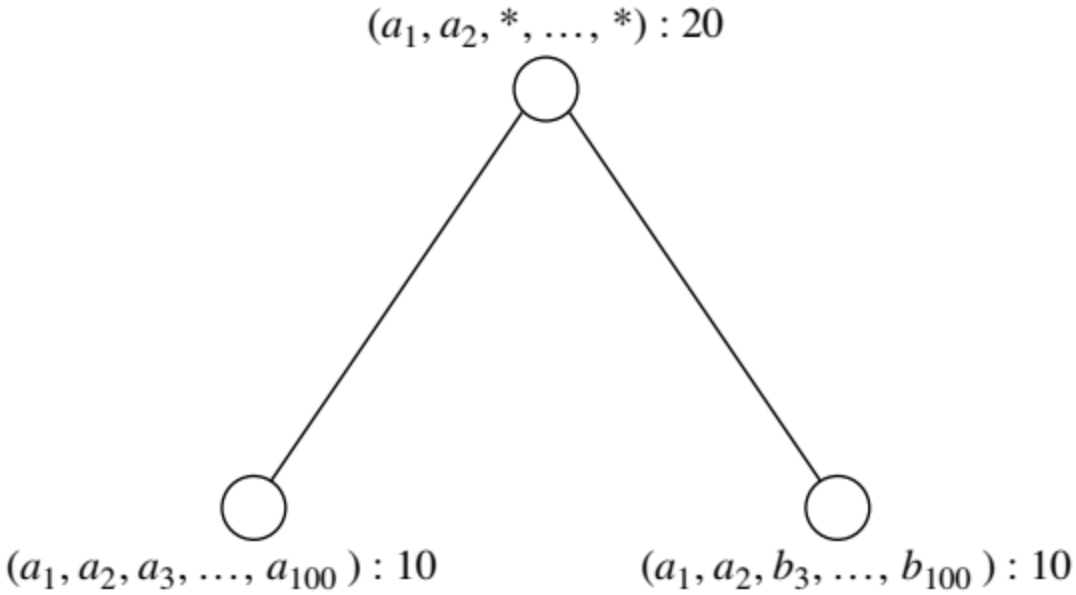
Iceberg Cube, Closed Cube & Cube Shell

- Is iceberg cube good enough?
  - 2 base cells: {(a_1, a_2, a_3 . . . , a_{100}):10, (a_1, a_2, b_3, . . . , b_{100}):10}
  - How many cells will the iceberg cube have if having count(*) >= 10?
- Close cube:
  - Closed cell c: if there exists no cell d, s.t. d is a descendant of c, and d has the same measure value as c.
  - Closed cube: a cube consisting of only closed cells
  - What is the closed cube of the above base cuboid? Hint: only 3 cells
- Cube Shell
  - Precompute only the cuboids involving a small # of dimensions, e.g., 3
  - More dimension combinations will need to be computed on the fly



For (A_1, A_2, … A_{10}), how many combinations to compute?

Three closed cells forming the lattice of a closed cube.



Example problem

6. When computing a cube of high dimensionality, we encounter the inherent *curse of dimensionality* problem: there exists a huge number of subsets of combinations of dimensions.

   (a) Suppose that there are only two base cells, $\{(a_1, a_2, a_3, \ldots, a_{100}), (a_1, a_2, b_3, \ldots, b_{100})\}$, in a 100-dimensional base cuboid. Compute the number of nonempty aggregate cells. Comment on the storage space and time required to compute these cells.

   (b) Suppose we are to compute an iceberg cube from the above. If the minimum support count in the iceberg condition is two, how many aggregate cells will there be in the iceberg cube? Show the cells.

   (c) Introducing iceberg cubes will lessen the burden of computing trivial aggregate cells in a data cube. However, even with iceberg cubes, we could still end up having to compute a large number of trivial uninteresting cells (i.e., with small counts). Suppose that a database has 20 tuples that map to (or cover) the two following base cells in a 100-dimensional base cuboid, each with a cell count of 10: $\{(a_1, a_2, a_3, \ldots, a_{100}) : 10, (a_1, a_2, b_3, \ldots, b_{100}) : 10\}$.

      i. Let the minimum support be 10. How many distinct aggregate cells will there be like the following: $\{(a_1, a_2, a_3, a_4, \ldots, a_{99}, *) : 10, \ldots, (a_1, a_2, *, a_4, \ldots, a_{99}, a_{100}) : 10, \ldots, (a_1, a_2, a_3, *, \ldots, *, *) : 10\}$?

      ii. If we ignore all the aggregate cells that can be obtained by replacing some constants with $*$'s while keeping the same measure value, how many distinct cells are left? What are the cells?

Answer:

(a) Suppose that there are only two base cells, $\{(a_1, a_2, a_3, \ldots, a_{100}), (a_1, a_2, b_3, \ldots, b_{100})\}$, in a 100-dimensional base cuboid. Compute the number of nonempty aggregate cells. Comment on the storage space and time required to compute these cells.

Each base cell generates $2^{100}-1$ aggregate cells. (We subtract 1 because, for example, $(a_1, a_2, a_3, \ldots, a_{100})$ is not an aggregate cell.) Thus, the two base cells generate $2 \times (2^{100}-1) = 2^{101}-2$ aggregate cells, however, four of these cells are counted twice. These four cells are: $(a_1, a_2, *, \ldots, *)$, $(a_1, *, \ldots, *)$, $(*, a_2, *, \ldots, *)$, and $(*, *, \ldots, *)$. Therefore, the total number of cells generated is $2^{101} - 6$.

(b) Suppose we are to compute an iceberg cube from the above. If the minimum support count in the iceberg condition is two, how many aggregate cells will there be in the iceberg cube? Show the cells.

They are 4: $\{(a_1, a_2, *, \ldots, *), (a_1, *, *, \ldots, *), (*, a_2, *, \ldots, *), (*, *, *, \ldots, *)\}$.

(c) Introducing iceberg cubes will lessen the burden of computing trivial aggregate cells in a data cube. However, even with iceberg cubes, we could still end up having to compute a large number of trivial uninteresting cells (i.e., with small counts). Suppose that a database has 20 tuples that map to (or cover) the two following base cells in a 100-dimensional base cuboid, each with a cell count of 10: $\{(a_1, a_2, a_3, \ldots, a_{100}) : 10, (a_1, a_2, b_3, \ldots, b_{100}) : 10\}$.

i. Let the minimum support be 10. How many distinct aggregate cells will there be like the following: $\{(a_1, a_2, a_3, a_4, \ldots, a_{99}, *) : 10, \ldots, (a_1, a_2, *, a_4, \ldots, a_{99}, a_{100}) : 10, \ldots, (a_1, a_2, a_3, *, \ldots, *, *) : 10\}$?
There will be $2^{101} - 6$, as shown above.

ii. If we ignore all the aggregate cells that can be obtained by replacing some constants by $*$'s while keeping the same measure value, how many distinct cells are left? What are the cells?
There are only three distinct cells left: $\{(a_1, a_2, a_3, \ldots, a_{100}) : 10, (a_1, a_2, b_3, \ldots, b_{100}) : 10, (a_1, a_2, *, \ldots, *) : 20\}$.

Outline

- Data Cube Computation: Preliminary Concepts
- Computing full/iceberg cubes: four methodologies
  - **1.General heuristics: On the computation of multidimensional aggregates (Agarwal et al.'96)**
  - 2.Bottom-Up: Multiway Array Aggregation for Full Cube Computation (Zhao et al SIGMOD'97)
  - **3**. Top-down: BUC: Computing Iceberg Cubes from the Apex Cuboid Downward (Beyer et al SIGMOD'99)
  - 4**.** Top-down: H-Cubing: Exploring an H-Tree Structure (Han et al SIGMOD'01)
- Overview of Other methodologies
  - 5.Star-cubing: Computing Iceberg Cubes Using a Dynamic Star-tree Structure (Xin et al VLDB'03)
  - 6.Precomputing Shell Fragments for Fast High-Dimensional OLAP (Li, et al. VLDB'04)
- Overview of Data Cubes for Advanced Applications
  - 7. Sampling Cubes: OLAP on Sampling Data (Li et al SIGMOD 08)
  - 8. Ranking Cubes: Efficient Computation of Ranking Queries (Xin et al VLDB06)
- Overview of Knowledge Discovery with Data Cubes
  - **9**. Prediction Cubes: Data Mining in Multi-Dimensional Cube Space (Chen et al VLDB'05)
  - 10**.** Complex Aggregation at Multiple Granularity: Multi-feature Cubes (Ross et al EDBT 1998)
  - 11**.** Discovery-Driven Data Cubes (Sarawagi et al EDBT 1998)
- Summary

On the Computation of Multidimensional Aggregates

Sameet Agarwal Rakesh Agrawal Prasad M. Deshpande Ashish Gupta
Jeffrey F. Naughton Raghu Ramakrishnan Sunita Sarawagi

Proceedings of the 22nd VLDB Conference
Mumbai(Bombay), India, 1996

Optimizations Possible

- **Smallest-parent:** Computing a group-by from the smallest previously computed group-by. For instance, AB can be computed from ABC, ABD or ABCD. ABC or ABD are clearly better choices for computing AB.
- **Cache-results:** Caching (in memory) the results of a group-by from which other group-bys are computed to reduce disk I/O. For instance, for the cube in Figure 1, having computed ABC, we compute AB
- **Amortize-scans:** This optimization aims at amortizing disk reads by computing as many group-bys as possible, together in memory. For instance, if the group-by ABCD is stored on disk, we could reduce disk read costs if all of ABC, ACD, ABD and BCD were computed in one scan of ABCD.
- **Share-sorts:** This optimization is specific to the sort-based algorithms and aims at sharing sorting cost across multiple group-bys.
- **Share-partitions:** When the hash table is too large to fit in memory, data is partitioned and aggregation is done for each partition that fits in memory. We can save on partitioning cost by sharing this cost across multiple groupbys

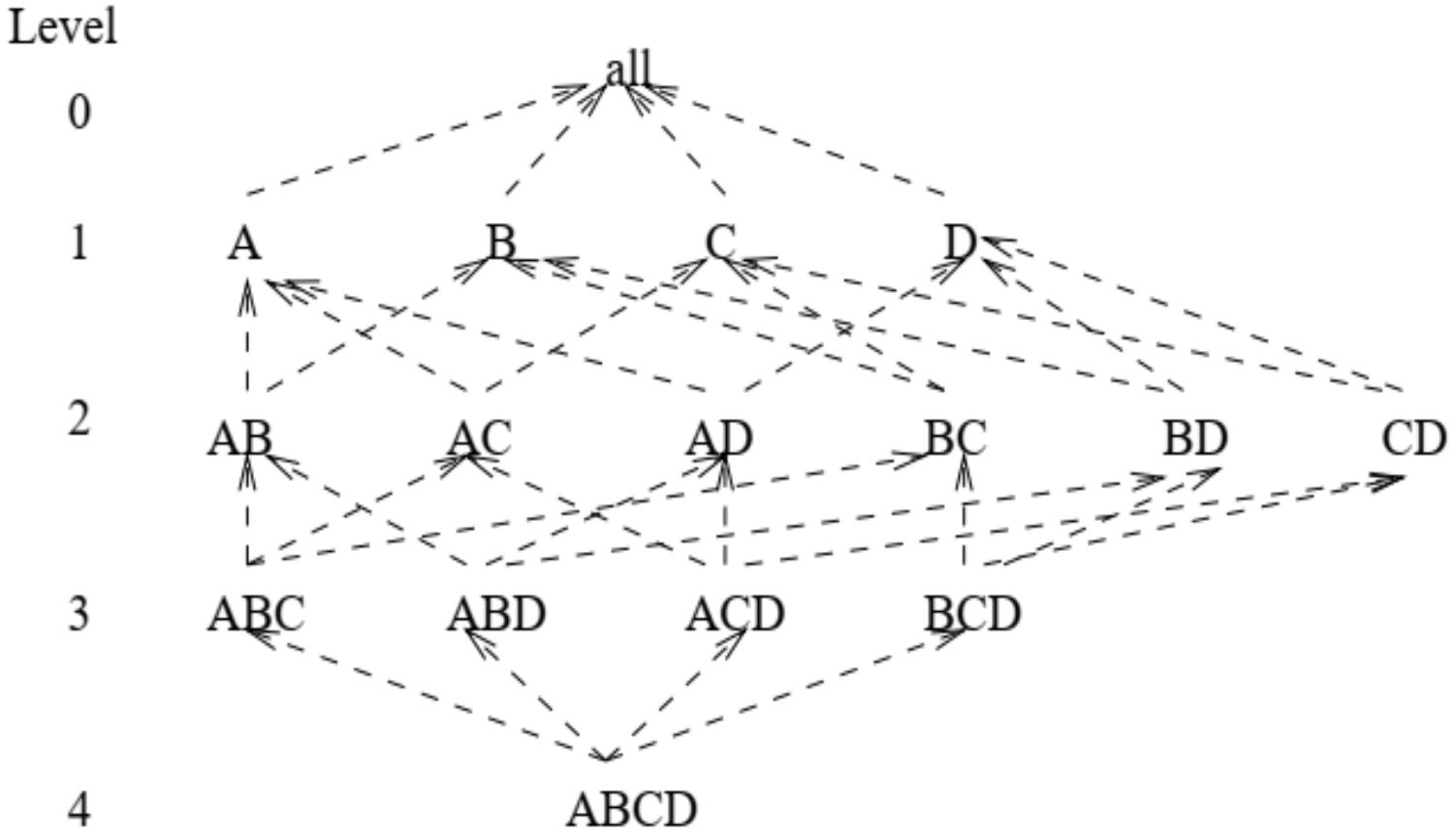


Figure 1: A search lattice for the cube operator

Algorithm PipeSort

- Employs the optimizations **cache-results and amortize-scans** to reduce disk scan cost by executing multiple group-bys in a pipelined fashion.
- Example
  - For instance, consider the example of using data sorted in the order ABCD to compute prefixes ABCD, ABC, AB and A.
  - Instead of computing each of these group-bys separately, we can compute them in a pipelined fashion.
    - Sort the raw data in the attribute order ABCD, we scan the sorted data to compute group-by ABCD.
      - Every time a tuple of ABCD is computed, it is propagated up the pipeline to compute ABC; every time a tuple of ABC is computed, it is propagated up to compute AB, and so on.
    - Thus, each pipeline is a list of group-bys all of which are computed in a single scan of the sort input stream. During the course of execution of a pipeline we need to keep only one tuple per group-by in the pipeline in memory.

Algorithm

- For each group-by we have estimate of the number of distinct values.
- Input is search lattice (Figure 1):
- Output: subgraph of the search lattice where each group-by is connected to a single parent from which it will be computed and is associated with attribute order in which it will be sorted.
- Generate plan incrementally.



(a) Transformed search lattice

(b) Minimum cost matching

Figure 2: Computing level 1 group-bys from level 2 group-bys in a 3 attribute cube

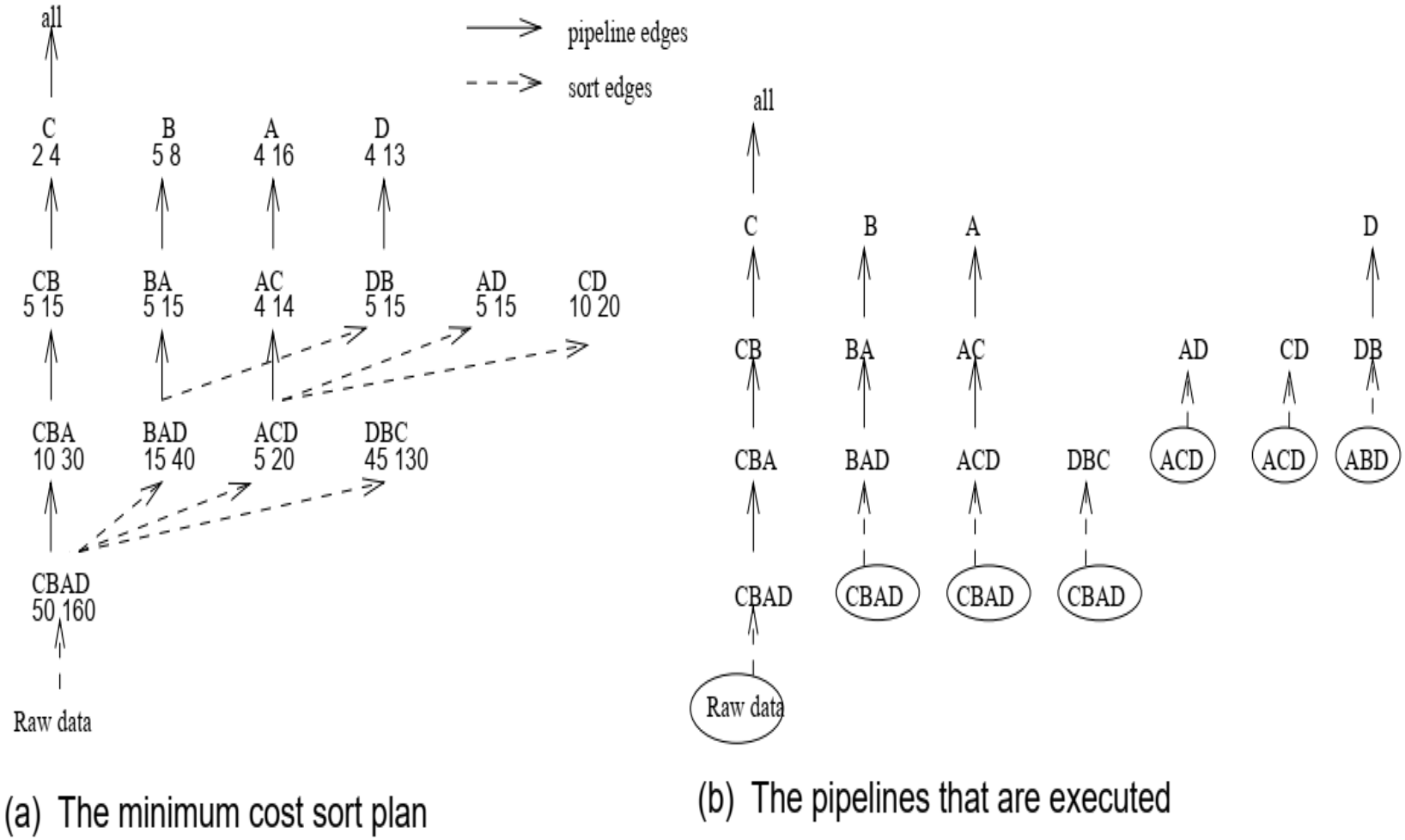


Figure 3: Sort-based method for computing a four attribute cube

Algorithm PipeHash

- The input is the search lattice.
- **Careful memory allocation**
  - **Optimizations: cache results and amortize scans**
- The PipeHash algorithm chooses for each group-by, the parent group-by with the smallest estimated total size.
- A minimum spanning tree (MST) is formed
  - where each vertex is a group-by and an edge from group-by a to b shows that a is the smallest parent of b.
- Approach: Divide the MST into smaller subtrees each of which can be computed in one scan of the group-by at the root of the MST such that the cost of scanning (from disk) the root group-by is minimized.
  - The problem is NP-complete
  - Heuristic: Optimizations cache-results and amortize-scans are favored by choosing as large a subtree of the MST as possible so that we can use the method above to compute together the group-bys in the subtree.

In the figure, we show the MST for a four attribute search lattice (the size of each group-by is indicated below the group-by).

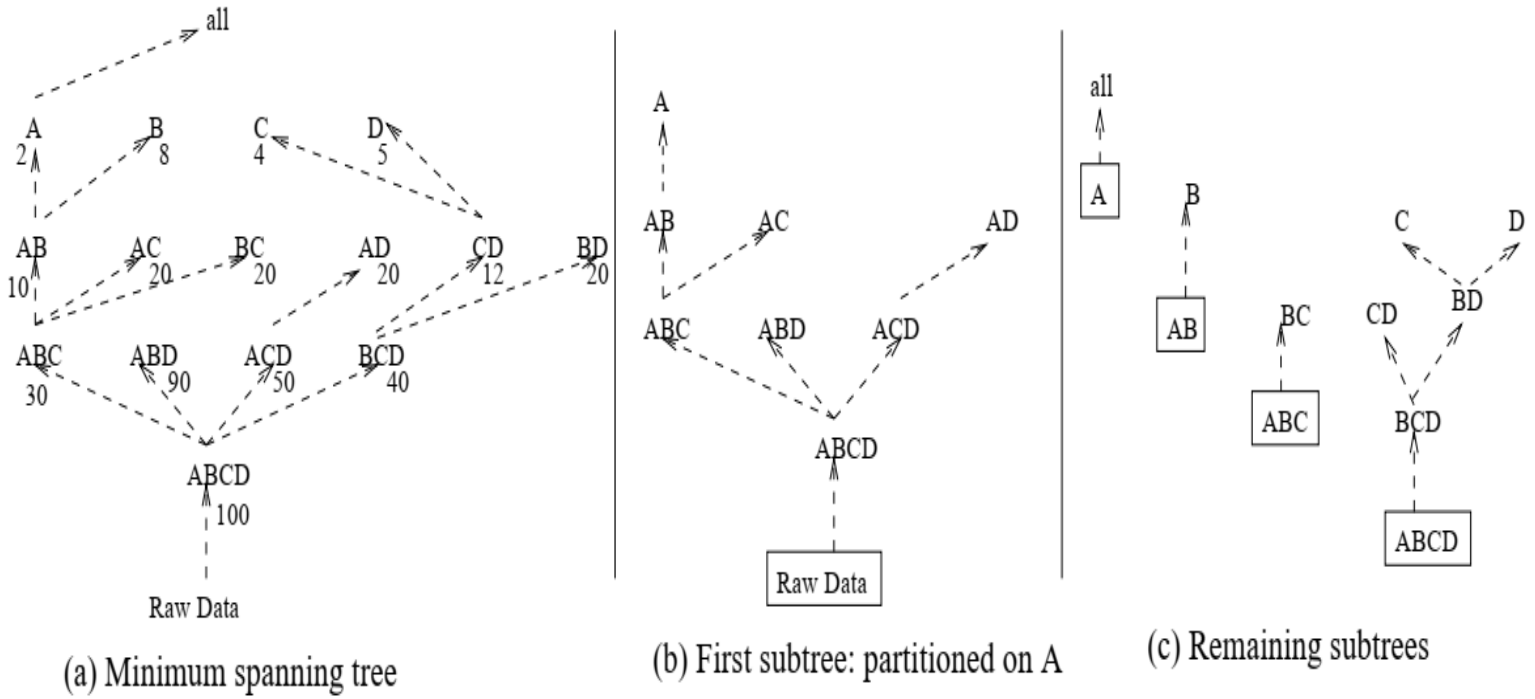


Figure 4: PipeHash on a four attribute group-by

Hierarchy of Group-By Queries (Parent Method)

- Consider the computation of the CUBE with {A; B ; C; D}.
  - The cuboid {A; C} can be computed from the cuboid {A; B ; C} or the cuboid {A; C; D}
  - In general, a cuboid on attribute set X can be computed from a cuboid on attribute set {Y} iff X subset of Y.
- Optimization
  - Choose Y to be as small as possible.
  - Heuristic: computing a cuboid with k-1 attributes from a cuboid with k attributes
  - Example: it is better to compute sum of sales by (product) using sum of sales by (product, customer) rather than sum of sales by (product, year, customer).
- We can view this hierarchy as a DAG, where the nodes are cuboids and there is an edge from a k attribute cuboid to a k-1 attribute cuboid iff the k-1 attribute set is a subset of the k attribute set.
- The DAG captures the consider-computing-from" relationship between the cuboids. The DAG for the CUBE on {A; B ; C; D} is shown in Figure 7.

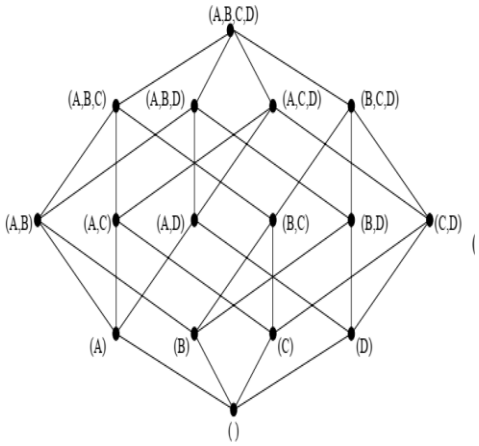


Figure 7: Sort orders enforced on the cuboids

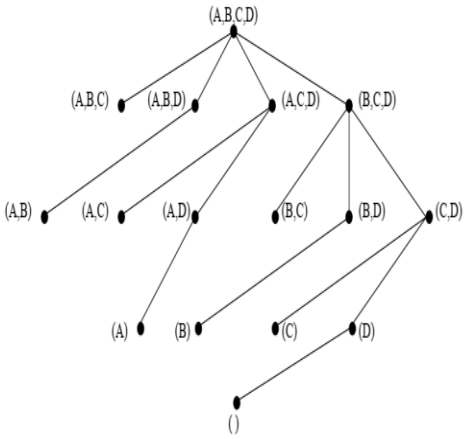


Figure 8: Cuboid Tree obtained from the cuboid DAG

Overlap Method

- As in the Parent method, the Overlap method computes each cuboid from one of its parents in the cuboid tree.
- Main Concept**: partially matching sort orders.**
- It tries to do better than the Parent method by overlapping the computation of different cuboids and using partially matching sort orders. This can significantly reduce the number of I/Os required.
- Sorting combined with Overlap seems to be a good option due to the following observations which help in reducing the number of sorting steps.
  - Cuboids can be computed from a sorted cuboid in sorted order.
  - An existing sort order on a cuboid can be used while computing other cuboids from it. For example, consider a cuboid X = {A; B ; D} to be computed from Y = {A; B ; C; D}.
  - Let Y be sorted in ABCD order which is not the same as ABD order needed to compute X. But Y need not be resorted to compute X. The existing order on Y can be used.

Main concept of overlap method: Sorted Runs

Sorted Runs :

Consider a cuboid on j attributes $\{A_1, A_2, \ldots, A_j\}$. We use $(A_1, A_2, \ldots, A_j)$ to denote the cuboid sorted on the attributes A_1, A_2, ..., A_j in that order. Consider the cuboid $S = (A_1, A_2, \ldots, A_{l-1}, A_{l+1}, \ldots, A_j)$ computed using $B = (A_1, A_2, \ldots, A_j)$. A *sorted run* R of S in B is defined as follows: $R = \Pi_{A_1, A_2, \ldots, A_{l-1}, A_{l+1}, \ldots, A_j}(Q)$ where Q is a **maximal** sequence of tuples τ of B such that for each tuple in Q, the first l columns have the same value. Informally a sorted run of S in B is a maximal run of tuples in B whose ordering is consistent with their ordering in the sort order associated with S.

For example, consider $B = [(a,1,2),(a,1,3),(a,2,2),(b,1,3),(b,3,2),(c,3,1)]$. Let S be the cuboid on the first and third attribute. i.e., $S = [(a,2),(a,3),(b,3),(b,2),(c,1)]$. The sorted runs for S are $[(a,2),(a,3)]$, $[(a,2)]$, $[(b,3)]$, $[(b,2)]$ and $[(c,1)]$.

Partitions :

B and S have a common prefix of $A_1, A_2, \ldots, A_{l-1}$. A *partition* of the cuboid S in B is a union of sorted runs such that the first $l-1$ columns (the common prefix) of all the tuples of the sorted runs have the same value. In the above example, the partitions for S in B will be $[(a,2),(a,3)]$, $[(b,2),(b,3)]$ and $[(c,1)]$.

This definition implies that all tuples of one partition are either less or greater than all tuples of any other partition. Tuples from different partitions will not merge for aggregation. Thus partition becomes a unit of computation and each partition can be computed independently of the others.

Outline of the Overlap Method

- The available memory may be insufficient for large CUBEs. We could try to reduce the amount of memory needed to compute a particular cuboid.
- Since partition can be a unit of computation, while computing a cuboid from another sorted cuboid, we just need memory sufficient to hold a partition of the cuboid.
- Basic idea
  - As soon as a partition is completed, the tuples can be pipelined into the computation of descendant cuboids, or written out to disk; the same memory can then be used to start computation of the next partition.
  - This is a significant reduction since for most cuboids the partition size is much less than the size of the cuboid.
- Example:
  - While computing (A; B ; C) and (A; B ; D) from (A; B ; C; D) the partition size for (A; B ; C) is 1 tuple (since (A; B ; C) sort order matches (A; B ; C; D) sort order) whereas the partition size for (A; B ; D) is bounded by the number of distinct values of D. So for computing these we just need space sufficient to hold a partition.
- So, with the overlap method, computation of many cuboids can be overlapped in the available memory effectively reducing the number of scans.

Summary

- Pipeline: Reduce scanning
- PipeHash: Create smallest size parent
- Parent method: Compute higher level cuboids from smallest cubiods
- Overlap: Create sorted orders

Outline

- Data Cube Computation: Preliminary Concepts
- Computing full/iceberg cubes: four methodologies
  - 1.General heuristics: On the computation of multidimensional aggregates (Agarwal et al.'96)
  - **2.Bottom-Up: Multiway Array Aggregation for Full Cube Computation (Zhao et al SIGMOD'97)**
  - **3**. Top-down: BUC: Computing Iceberg Cubes from the Apex Cuboid Downward (Beyer et al SIGMOD'99)
  - 4**.** Top-down: H-Cubing: Exploring an H-Tree Structure (Han et al SIGMOD'01)
- Overview of Other methodologies
  - 5.Star-cubing: Computing Iceberg Cubes Using a Dynamic Star-tree Structure (Xin et al VLDB'03)
  - 6.Precomputing Shell Fragments for Fast High-Dimensional OLAP (Li, et al. VLDB'04)
- Overview of Data Cubes for Advanced Applications
  - 7. Sampling Cubes: OLAP on Sampling Data (Li et al SIGMOD 08)
  - 8. Ranking Cubes: Efficient Computation of Ranking Queries (Xin et al VLDB06)
- Overview of Knowledge Discovery with Data Cubes
  - **9**. Prediction Cubes: Data Mining in Multi-Dimensional Cube Space (Chen et al VLDB'05)
  - 10**.** Complex Aggregation at Multiple Granularity: Multi-feature Cubes (Ross et al EDBT 1998)
  - 11**.** Discovery-Driven Data Cubes (Sarawagi et al EDBT 1998)
- Summary

An Array-Based Algorithm for Simultaneous Multidimensional Aggregates *

Yihong Zhao
Computer Sciences Department
University of Wisconsin-Madison
zhao@cs.wisc.edu

Prasad M. Deshpande
Computer Sciences Department
University of Wisconsin-Madison
pmd@cs.wisc.edu

Jeffrey F. Naughton
Computer Sciences Department
University of Wisconsin-Madison
naughton@cs.wisc.edu

SIGMOD 97

Introduction

- Several efficient algorithms for Relational OnLine Analytical Processing (ROLAP) have been developed to compute the Cube.
- This paper: how to compute the Cube for Multidimensional OLAP (MOLAP) systems
- MOLAP systems present a different sort of challenge in computing the cube than do ROLAP systems.
- Reason: the data structures to store data are different

ROLAP systems

- ROLAP systems store in relational tables
  - A “cell” in a logically multidimensional space is represented as a tuple, with some attributes that identify the location of the tuple in the multidimensional space, and other attributes that contain the data value corresponding to that data cell.
  - Example:
    - A cell of the array might be represented by the tuple (shoes, WestTown, 3-July-96, $34. 00).
    - Computing the cube over such a table requires a generalization of standard relational aggregation operators.
- In prior work, three main ideas have been used to make ROLAP computation efficient:
  - Using some sort of grouping operation on the dimension attributes to bring together related tuples (e.g., sorting or hashing),
  - Using the grouping performed on behalf of one of the sub-aggregates as a partial grouping to speed the computation another sub-aggregate, and
  - To compute an aggregate from another aggregate, rather than from larger table.

MOLAP systems

- Store data in arrays
  - Example: For storing the tuple (shoes, WestToun, 3-JuIY-1996, $34.00), a MOLAP system would just store the data value $34. 00;
  - The position within the sparse array would encode the fact that this is a sales volume for shoes in the West Town store on July 3, 1996.
- For computing the cube on data stored in arrays, one can once again use the ROLAP trick of computing one aggregate from another.
- However, none of the other techniques of ROLAP cube can be applied.
  - No equivalent of "reordering to bring together related tuples" based upon dimension values.
  - There is no concept of using an order generated by one subaggregate in the computation of another;
- Should visit those values in the right order so that the computation is efficient.
- Basic idea
  - is to simultaneously compute spatially-delimited partial aggregates so that a cell does not have to be revisited for each sub-aggregate.
  - one must "chunk" them into small memory-sized pieces, and perform some sort of "compression" to avoid wasting space on cells that contain no valid data

- In this paper, we present a MOLAP algorithm incorporating of these ideas.
- The algorithm succeeds in overlapping the computation of multiple subaggregates, and makes good use of available main memory.
- It contains several mathematical proofs and theorems.

- One can always use our MOLAP algorithm in a relational system by the following three-step procedure:
  - Scan the table, and load it into an array.
  - Compute the cube on the resulting array
  - Dump the resulting cubed array into tables.
- Surprise:
  - that this three-step approach was actually faster than the direct approach of cubing the table.

Array Storage Issues

- There are three main issues to resolve.
- First,
  - it is highly likely in a multidimensional application that the array itself is far too large to fit in memory.
  - In this case, the array must be split up into “chunks”, each of which is small enough to fit comfortably in memory.
- Second,
  - even with this ‘chunking”, it is likely that many of the cells in the array are empty, meaning that there is no data for that combination of coordinates.
  - To efficiently store this sort of data we need to compress these chunks.
- Third,
  - in many cases an array may need to be loaded from data that is not in array format (e.g., from a relational table or from an external load file.)

1. Chunking Arrays

- Arrays should be stored in chunks, bot how?
- Standard programming language technique of storing the array in a row major or column major order is not very efficient.
  - Consider a row major representation of a two-dimensional array, with dimensions Store and Date, where Store forms the row and Date forms the column.
  - Accessing the array in the row order (order of ***Stores)*** is efficient with this representation, since each disk page that we read will contain several Stores. However, accessing in the order of columns ***(Date)*** is inefficient.
  - If the Store dimension is big, each disk page read will only contain data for one ***Date. T***o get data for the next ***Date*** will require another disk access;
    - In fact there will be one disk access for each ***Date*** required.
  - The simple row major layout creates an asymmetry among the dimensions, favouring one over the other. This is because data is accessed from disk in units of pages
- Solution: Chunking is a way to divide an n-dimensional array into small sizen-dimensional chunks and store each chunk as one object on disk. Each array chunk has `n' dimensions and will correspond to the blocking size on the disk.

Compressing Sparse Arrays

- For dense chunks,
  - Greater than 40% of the array cells have a valid value,
  - we do not compress the array, simply storing all cells of the array as is but assigning a null value to invalid array cells.
  - We will have uniform array chunks.
- Sparse chunks
  - Storing the data without compression is a waste of storage, as most of the cells do not contain valid data.
  - In this case we use what we call “chunk-offset compression”.
  - In chunk-offset compression, for each valid array entry, we store a pair, (offsetInChunk, data). The offsetlnChunk integer can be computed as follows: consider the chunk as a normal (uncompressed) array. Each cell c in the chunk is defined by a set of indices
  - For example, if we are working with a three-dimensional chunk, a given cell will have an “address” (i,j,k) in the chunk.

Loading Arrays from Table

- The table size and number of chunks are known.
- If the memory is less than the table size, create partitions of the main memory size.
- Next, read the partitions and create chunks consists of hash buckets.

A Basic Array Cubing Algorithm

- Use minimum memory, no overlap
- First consider how to compute a group-by from a simple non-chunked array.
  - Consider three dimensional array A, B, and C. and we want to compute the aggregate AB?
  - This can be seen as projecting onto the AB plane; logically, this can be done by sweeping a plane through the C dimension, aggregating as we go, until the whole array has been swept.
- Consider that ***ABC*** array is stored in a number of chunks.
  - Now instead of sweeping an entire plane of size |***A***|*\|***B***|. where IA| and IBI are the sizes of the ***A*** and B dimensions,
    - we do it on a chunk by chunk basis.
  - With this, each chunk can be read by once
- This generalization of this algorithm to higher dimensions is straight-forward; instead of sweeping planes through arrays, in higher dimensions, say ***k*** dimensional arrays, one sweeps k-1 dimensional arrays.

Computing all aggregates of an array

- If the array has dimensions ABC, we need to compute AB, BC, AC, and A, B, C, as well as the overall total aggregate.
- Naive approach
  - compute all of these aggregates from the initial ABC array.
- It is far more efficient to compute A from AB than computing A from ABC.
  - This idea has been explored in the ROLAP cube computation literature.
  - the aggregates to be computed can be viewed as a lattice, with ***ABC as*** the root
  - *ABC* has children AB, BC, AC and AC had children A and C and so forth.
  - To compute the cube efficiently we embed a tree in this lattice, and compute each aggregate from its parent in this tree.
- For ROLAP cube computations this is a difficult question, since the sizes of the tables corresponding to the nodes in the lattice are not known until they are computed, so heuristics must be used.
- For our chunk-based array algorithm
  - We know about the storage requirement
  - The "minimum size spanning tree" can be denied for the lattice.
  - For each node n in the lattice, its parent in the minimum size spanning tree is the node n' which has the minimum size. The "n" can be computed.

Example

- The array ***ABC' is*** a 16 x 16 x 16 array with 4 x 4 x 4 array chunks laid out in the dimension order ABC.
- The chunks are numbered from 1 to 64.
- For example, to computing the BC group-by, we read in the chunk number order from 1 to 64, aggregate each four ABC chunks to a BC chunk, output the BC chunk to disk, and reuse the memory for the next BC chunk

- **So far we have not used the hierarchy of aggregates**
- **Th above basic algorithm will compute *AB* from *ABC*, then will re-scan *ABC* to compute *AC*, then will scan it a third time to compute *BC*.**

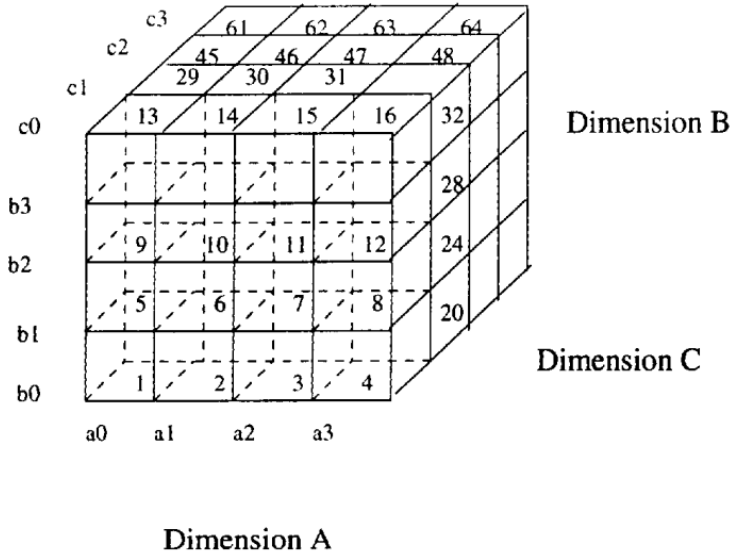


Figure 1: 3 D array

Multi-Way Array Aggregation

- Determine the dimensional order
  - When computing multiple group-bys simultaneously, the total memory required depends critically on the order in which the input array is scanned.
  - Basic idea: exploit a special logical order called "dimension order".
- Naïve algorithm:
  - We can formulate a general rule to determine what chunks of each group-by of the cube need to stay in memory in order to avoid rescanning a chunk of the input array.

- Example:
  - The array chunks are read in the dimension order *ABC,* i.e.. from chunk 1 to chunk 64. Suppose chunk 1 is read in.
  - For group-by *AB,* this chunk is aggregated along the C dimension to get a chunk of *AB.*
  - Similarly for *AC* and *BC,* this chunk is aggregated along *B* and *A* dimensions respectively.
  - Thus the first chunk's *AB* group by is aggregated to the chunk (a_0, b_0) of *AB;* the first chunk's *AC'* is aggregated to the chunk (a_0, c_0) of *AC;* the first chunk's *BC* is aggregated to the chunk (b_0, c_0) of *BC.*
  - As we read in new chunks, we aggregate the chunk's AB, *AC* and *BC* group-by to the corresponding chunks of group-bys *A B,* AC and *BC.*

Multi-way aggregation

- To compute each chunk of ***AB, AC,* and *BC*** group-by, we may naively allocate memory to each chunk of those group-bys in memory.
- Basic idea:
  - However, **we can exploit the order** in which each chunk is brought in memory to reduce the memory required by all group-by (AB,AC,BC, A,B,C)
- Example:
  - Read the chunks in dimension order (A, ***B, C)*** layout, which is a linear order from chunk 1 to chunk 64.
  - For the chunk 1 to chunk 4, complete the aggregation for the chunk (b_0, c_0) of *BC* after *aggre*gating each chunk's *BC* group-by to the chunk (b_0, c_0) of *BC*.
  - Once the (b_0, c_0) chunk is completed, we write out the chunk and reassign the chunk memory to the chunk (b_1, c_0) which is computed from the next 4 chunks of *ABC i.e.* the chunk 4 to chunk 8. **So we allot only one chunk of *BC* in memory to compute the entire *BC* group-by.** Similarly, we can carry out group by for AC group-by and AB group-by
- Notice that we generate each ***BC*** chunk in the dimension order ***(B, C).***
- **Before we write each *BC* chunk to disk,** we use the *BC* chunks to compute the chunks of *B* or C as if we read in each *BC* chunk in the dimension order *(B, C).*

Multi-Way Array Aggregation for Cube Computation (Method Summary)

- Method: the chunks should be sorted and computed according to their size in ascending order
  - Idea: keep the smallest plane in the main memory, fetch and compute only one chunk at a time for the largest plane
- Limitation of the method: computing well only for a small number of dimensions
  - If there are a large number of dimensions, "top-down" computation and iceberg cube computation methods can be explored

Outline

- Data Cube Computation: Preliminary Concepts
- Computing full/iceberg cubes: four methodologies
  - 1.General heuristics: On the computation of multidimensional aggregates (Agarwal et al.'96)
  - 2.Bottom-Up: Multiway Array Aggregation for Full Cube Computation (Zhao et al SIGMOD'97)
  - **3. Top-down: BUC: Computing Iceberg Cubes from the Apex Cuboid Downward (Beyer et al SIGMOD'99)**
  - 4**.** Top-down: H-Cubing: Exploring an H-Tree Structure (Han et al SIGMOD'01)
- Overview of Other methodologies
  - 5.Star-cubing: Computing Iceberg Cubes Using a Dynamic Star-tree Structure (Xin et al VLDB'03)
  - 6.Precomputing Shell Fragments for Fast High-Dimensional OLAP (Li, et al. VLDB'04)
- Overview of Data Cubes for Advanced Applications
  - 7. Sampling Cubes: OLAP on Sampling Data (Li et al SIGMOD 08)
  - 8. Ranking Cubes: Efficient Computation of Ranking Queries (Xin et al VLDB06)
- Overview of Knowledge Discovery with Data Cubes
  - **9**. Prediction Cubes: Data Mining in Multi-Dimensional Cube Space (Chen et al VLDB'05)
  - 10**.** Complex Aggregation at Multiple Granularity: Multi-feature Cubes (Ross et al EDBT 1998)
  - 11**.** Discovery-Driven Data Cubes (Sarawagi et al EDBT 1998)
- Summary

Bottom-Up Computation of Sparse and Iceberg CUBEs

Kevin Beyer
Computer Sciences Department
University of Wisconsin – Madison
beyer@cs.wisc.edu

Raghu Ramakrishnan
Computer Sciences Department
University of Wisconsin – Madison
raghu@cs.wisc.edu

Presentation made by

P. Krishna Reddy, IIIT Hyderabad

Outline

- Introduction
- Motivation
- The Iceberg-CUBE Problem
- Previous CUBE Algorithms
- Algorithm Bottom-Up Cube
- Performance Analysis
- Conclusions

Introduction

- The basic CUBE problem computes all aggregates as efficiently as possible.
- The chief difficulty is that
  - The CUBE problem is exponential in the number of dimensions: for d dimensions, 2^d group-bys are computed.
  - The size of each group-by depends upon the cardinality of its dimensions.
  - If every store sold every product, then the (Product, Store) group-by would have |Product| x |Store| result tuples.
  - However, as the number of dimensions or the cardinalities increase, the product of the cardinalities grossly exceeds the (fixed) size of the input relation for many of the group-bys.
  - Even in our small example, if the data comes from a large department store it is highly unlikely that a given customer purchased even 5% of of the products, or shopped in more than 1% of the stores.

Contributions

- Introduced a variant of CUBE problem called Iceberg-CUBE
- We present a simple and efficient algorithm, called Bottom Up CUBE (BUC), for the Iceberg-CUBE problem
- We present a simple and efficient algorithm, called BUC, for the Iceberg-CUBE problem

Motivation

- A full CUBE on a real nine-dimensional dataset containing weather conditions at various weather stations on land for September 1985.
  - The dataset had 1,015,367 tuples (~39MB).
  - The CUBE on this dataset produces 210,343,580 tuples (~8GB)-more than 200 times the input size!
- Two alternatives to improve the performance
  - Compute a subset of CUBEs
  - Iceberg CUBE: proposed approach as a new alternative

The Iceberg-CUBE Problem

•CUBE

SELECT A,B,C,COUNT(•),SUM(X)
FROM R
CUBE BY A,B,C

•Iceberg CUBE

SELECT A,B,C,COUNT(•),SUM(X)
FROM R
CUBE BY A,B,C
HAVING COUNT(•) > N

•An Iceberg query computes a single group-by and eliminates all tuples with an aggregate value below some threshold. For example:

```
SELECT  A,B,C,COUNT(*)
FROM    R
GROUP BY A,B,C
HAVING COUNT(*) >= N
```

Previous Approaches

- Pipesort
  - Converts a CUBE lattice into processing tree
- PipeHash
  - Estimate
  - Compute by starting from smallest parent
- Overlap
  - Sorting order of Prefix attributes is shared
  - Overlap as much sorting as possible by computing a group-by from a parent with the maximum sort-order overlap.
- ArrayCube
  - it uses in-memory arrays to store the partitions and to avoid sorting.
- PartitionedCube and MemoryCube
  - PartitionedCube partitions the data on some attribute into memory-sized units
  - and MemoryCube computes the CUBE on each in-memory partition.

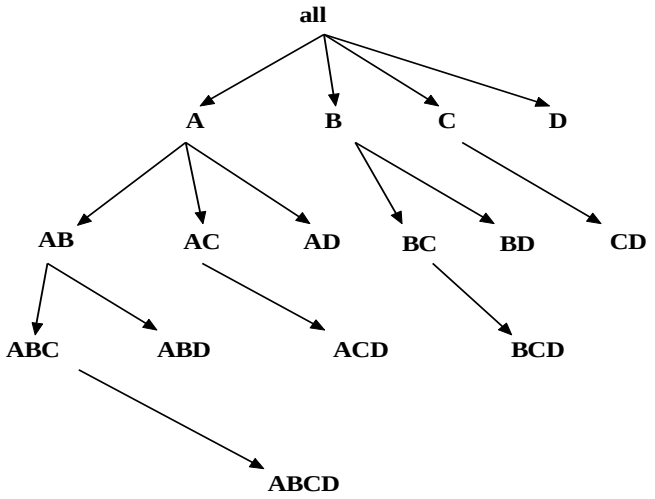
Algorithm Bottom-Up Cube (BUC)

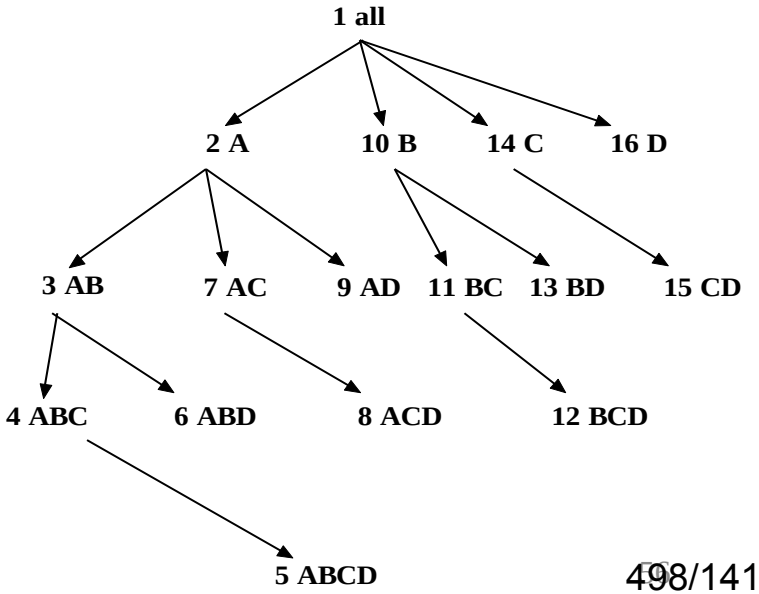
- BUC is proposed for sparse CUBE and Iceberg-CUBE computation.
- Basic idea
  - The idea in BUC is to combine the 1/0 efficiency of PartitionedCube/ MemoryCube, but to take advantage of minimum support pruning like Apriori.
- To achieve pruning, BUC proceeds from the bottom of the lattice (i.e., the smallest / most aggregated group-bys), and works its way up towards the larger, less aggregated group bys.
- All of the previous algorithms compute in the opposite direction. Since parent group-bys are used to compute child group-bys, the algorithms cannot avoid computing the parents.

Bottom-Up Computation (BUC)

- Bottom-up cube computation

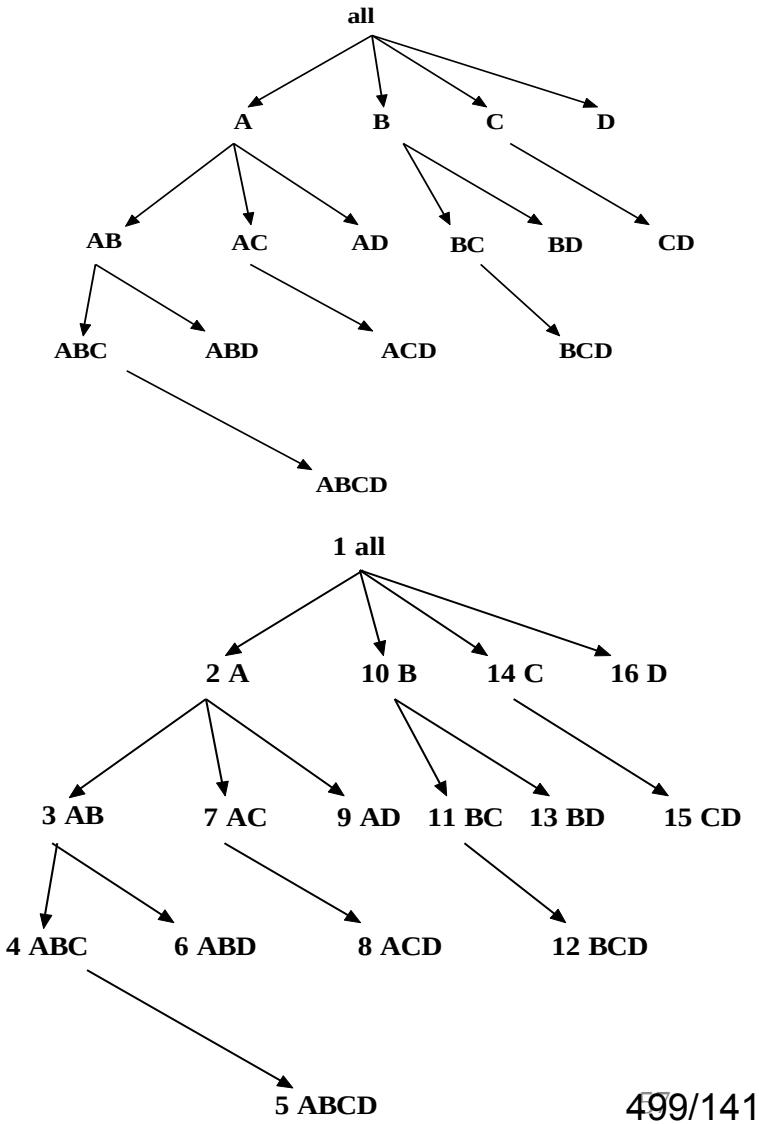
  (Note: top-down in our view!)
- Divides dimensions into partitions and facilitates iceberg pruning
  - If *minsup* = 1 $\Rightarrow$ compute full CUBE!
- No simultaneous aggregation





Iceberg CUBE with BUC

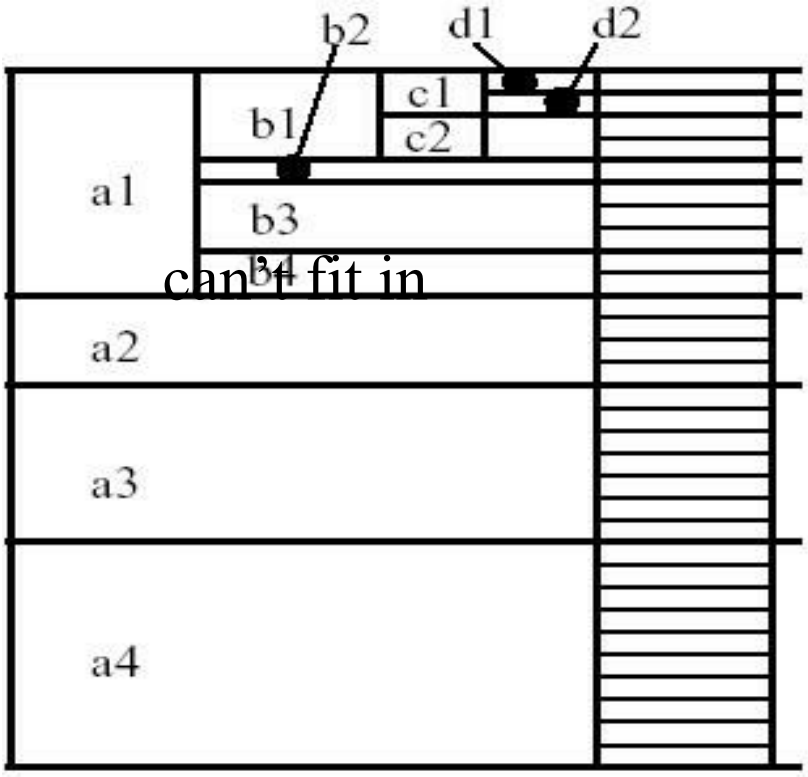
- **If a partition does not satisfy *min_sup*, its descendants can be pruned**
- No simultaneous aggregation
- When a small partition is found, instead of writing for all of the group-bys, BUC simply skips the partition and does not consider any of the partition's ancestors.
- The pruning is correct because the partition sizes are always decreasing when BUC recurses, and therefore none of the ancestors can have minimum support.



Additional Pruning Functions

- For any two sets (of tuples, in our case) S and *T* such that *S is a subset of T*, a function *f* is **monotonically decreasing** if *f(S) <= f(T)*.
  - Counting
- Average is not monotonic
- Consider HAVING COUNT(*) < X.
  - In this case, the user wants the groups with little aggregation, which occur towards the top of the lattice.
  - BUC can not help

BUC: Partitioning



- Usually, entire data set can't fit in main memory
- Sort *distinct* values
  - partition into blocks that fit
- Continue processing
- Optimizations
  - Partitioning
    - External Sorting, Hashing, Counting Sort
  - Ordering dimensions to encourage pruning
    - Cardinality, Skew, Correlation
  - Collapsing duplicates
    - Can't do holistic aggregates anymore!

- **Partitioning**
  - When 'input' does not fit in main memory, the data must be partitioned to disk. This can be done with hash-partitioning, followed by a partitioning within each bucket (which hopefully now fits in main memory), or external-sort can be used.
  - When performing an external partitioning, the aggregation step can be combined with the partitioning.
- **Dimension Ordering**
  - The performance of BUC is sensitive to the ordering of the dimensions.
  - The goal of BUC is to prune as early as possible; i.e., BUG wants to find partitions that do not meet minimum support
  - For best performance, the most **discriminating** dimensions should be used first
  - **Cardinality**
    - **Higher the cardinality and the smaller the partitions**
  - **Skew**
    - **Skewed dimensions will have less cardinality and so should be considered later**
  - **Correlation**
    - **If the dimension is correlated with earlier dimension, it decreases pruning.**

Experimental Results

- See the research paper
- Results on real data
  - Weather data
  - Mail order data

Conclusion

- Major contribution
  - Sharing of partition costs
  - Like multiway, it can not share the computation of aggregates between parent and child group-bys.
  - Because, computation cuboid AB does not help ABC.

Topics till this slide are for QUIZ 1

Outline

- Data Cube Computation: Preliminary Concepts
- Computing full/iceberg cubes: four methodologies
  - 1.General heuristics: On the computation of multidimensional aggregates (Agarwal et al.'96)
  - 2.Bottom-Up: Multiway Array Aggregation for Full Cube Computation (Zhao et al SIGMOD'97)
  - 3. Top-down: BUC: Computing Iceberg Cubes from the Apex Cuboid Downward (Beyer et al SIGMOD'99)
  - **4. Top-down: H-Cubing: Exploring an H-Tree Structure (Han et al SIGMOD'01)**
- Overview of Other methodologies
  - 5.Star-cubing: Computing Iceberg Cubes Using a Dynamic Star-tree Structure (Xin et al VLDB'03)
  - 6.Precomputing Shell Fragments for Fast High-Dimensional OLAP (Li, et al. VLDB'04)
- Overview of Data Cubes for Advanced Applications
  - 7. Sampling Cubes: OLAP on Sampling Data (Li et al SIGMOD 08)
  - 8. Ranking Cubes: Efficient Computation of Ranking Queries (Xin et al VLDB06)
- Overview of Knowledge Discovery with Data Cubes
  - **9**. Prediction Cubes: Data Mining in Multi-Dimensional Cube Space (Chen et al VLDB'05)
  - 10**.** Complex Aggregation at Multiple Granularity: Multi-feature Cubes (Ross et al EDBT 1998)
  - 11**.** Discovery-Driven Data Cubes (Sarawagi et al EDBT 1998)
- Summary

Efficient Computation of Iceberg Cubes with Complex Measures *

Jiawei Han† Jian Pei† Guozhu Dong‡ Ke Wang†

† School of Computing Science, Simon Fraser University, B.C., Canada, {han, peijian, wangk}@cs.sfu.ca

‡ Department of Computer Science, Wright State University, Dayton, OH, U.S.A., gdong@cs.wright.edu

SIGMOD 2001

Presented by P. Krishna Reddy
IIIT Hyderabad

Outline

- Introduction
- Iceberg cube with the average measures
- Exploration of Weaker, Anti-monotonic Conditions
- Extension of Apriori and BUC for Iceberg Cube with Average
- Top-k H-Cubing: Top-k Cubing Using a Hyper-Tree Structure
- Performance Analysis
- Conclusions

Introduction

- Materializing iceberg cubes is one of existing method.
  - where an iceberg cube is a subset of a cube containing only those cells whose measure, such as count, satisfies certain constraints, such as minimal support threshold
- Advantage of Iceberg cube
  - Most of the cells at the low level cuboids are likely to contain trivial aggregate values and may not pass certain threshold, and therefore, do not need to be computed in an iceberg cube.
  - This not only saves processing time and disk space but also makes the analysis focused only on interesting data

Introduction

- Previous studies on efficient computation iceberg cubes have been conned to with simple measures, such as count and sum, by exploring the anti-monotonic property of such icebergs.
  - For example, if the count of a cell c in a cuboid C is no higher than v, then the count of any of c's descendant cells in the lower level cuboids can never be higher than v, and thus can be pruned by the Apriori-like method.
- Unfortunately, not all the measures have such antimonotonic property.
  - For example, even if the average value in a cell c of a cuboid C is no higher than v, the average value of some of c's descendant cells in the lower level cuboids may still be higher than v.
- **We study how to efficiently compute iceberg cubes with non-anti-monotonic measures**

Example queries

Example 1 Suppose a sales database has four dimensions: *time*, *location*, *customer*, and *product*, and two measures: *price* and *cost* (note: *profit* = *price* − *cost*). The following queries require the computation of iceberg cubes with such *complex* measures.

Q_1: Find groups of sales which contain at least 50 items and whose average item price is at least \$800, grouped by month, city, and/or customer groups.

Q_2: Find groups of sales which contain at least 200 items and whose *total profit*^{2} is more than \$6000, grouped by month, city, and/or customer groups.

Q_3: For sales grouped by month, city, and/or customer groups, containing at least 20 items, with an average item price of no less than \$800, find those customer groups on which one can make at least 10% more profit than the average of all the customers. □

Contributions

- It develops a mapping which transforms some non-antimonotonic testing conditions to somewhat weaker but anti-monotonic testing conditions.
  - For example, test on average can be mapped to an anti-monotonic, top-k average test. Mappings for several other measures are worked out as well.
- It extends two previously studied methods, Apriori and BUC, to Top-k Apriori and Top-k BUC, for computing iceberg cubes with the average measure.
- To further improve the performance for computing iceberg cubes, a hypertree structure, called H-tree, is designed, and a new iceberg cubing method, called Top-k H-Cubing, is developed.
- The method explores several efficient processing techniques, including
  - tree-based data compression
  - dynamic link adjustment, and
  - quantitative information merge

Outline

- Introduction
- **Iceberg cube with the average measures**
- Exploration of Weaker, Anti-monotonic Conditions
- Extension of Apriori and BUC for Iceberg Cube with Average
- Top-k H-Cubing: Top-k Cubing Using a Hyper-Tree Structure
- Performance Analysis
- Conclusions

Example 2 (Iceberg cubes on average) Consider a *Sales_Info* table, given in Table 1, which registers sales related to month, day, city, customer group, product, cost, and price.

Mon.	Day	City	Cust_group	Product	Cost	Price
Jan	10	Toronto	Edu.	HP Printer	500	485
Jan	15	Toronto	Household	Sony TV	800	1200
Jan	20	Toronto	Edu.	Canon Camera	1160	1280
Feb	20	Montreal	Busi.	IBM Laptop	1500	2500
Mar.	4	Vancouver	Edu.	Seagate HD	540	520
...	...	...	...	...	...	...

Table 1: A *Sales_Info* table.

An iceberg cube, *Sales_Iceberg*, which computes Q_1 is presented as follows.

```
CREATE CUBE Sales_Iceberg AS
SELECT month, city, customer-group, AVG(price),
       COUNT(*)
FROM Sales_Info
CUBEBY month, city, customer-group
HAVING AVG(price) >= 800 AND COUNT(*) >= 50
```

- Sales Iceberg is the restriction of the data cube
  - Sales Iceberg excludes all the cells of the latter whose average price is less than $800 or whose count is less than 50.
- It is also different from its corresponding Iceberg query,
  - Iceberg CUBE query contains the qualified cells of all the possible group-bys of the three dimensions
  - Sales_Iceberg contains only the qualified cells with the three dimensions grouped together

Definition

- Given an iceberg cube ICube, the (whole) data cube formed by the same specification of ICube without the HAVING clause is called the background cube of ICube, and is denoted as B(ICube).

$\mathcal{B}(Sales_Iceberg)$, the *background cube* of the iceberg cube, *Sales_Iceberg*, of Example 2 is defined as,

```
CREATE CUBE Sales_Cube AS
SELECT month, city, customer-group, AVG(price),
       COUNT(*)
FROM Sales_Info
CUBEBY month, city, customer-group
```
□

Definition 2 An iceberg cube, *ICube*, is **anti-monotonic** if and only if for each cell c in $\mathcal{B}(ICube)$, if c *violates* the constraint specified by *ICube*'s `HAVING` clause, so does every descendant of c. □

Example 4 Given the sales table in Example 2, *Count_Iceberg*, shown below, is *anti-monotonic*.

```
CREATE CUBE Count_Iceberg AS
SELECT month, city, customer-group, COUNT(*)
FROM Sales_Info
CUBEBY month, city, customer-group
HAVING COUNT(*) >= 100
```

- If a cell c count is less than 100, then every descendant of c will violate the constraint since the count of each subcube of c must be no larger than that of c.
- Sales Iceberg in Example 2 is, however, not antimonotonic.
  - For example, even when the average price of all the items sold in March is less than $800, e.g., (March; *; *; 600; 1800), the average price for a subset containing only the sales to business people, e.g., (March;* ; Busi:; 1300; 360), may still satisfy the constraint specified in the HAVING clause.

Outline

- Introduction
- Iceberg cube with the average measures
- **Exploration of Weaker, Anti-monotonic Conditions**
- Extension of Apriori and BUC for Iceberg Cube with Average
- Top-k H-Cubing: Top-k Cubing Using a Hyper-Tree Structure
- Performance Analysis
- Conclusions

- **Question:** As iceberg cube involves the non-anti-monotonic measure average, it does not have the Apriori property. Can we find a weaker but anti-monotonic auxiliary condition that may help us compute iceberg cubes efficiently?
- **Solution:** Yes, Top-k average: An anti-monotonic condition for testing average

3.1 Top-k average: An anti-monotonic condition for testing average

Let us examine the following iceberg cube $AvgI$, a generalization of Example 2, defined on a relational table T with i dimensions and one measure M.

CREATE CUBE $AvgI$ AS
SELECT $A_1, A_2, \ldots, A_m$, AVG(M), COUNT($*$)
FROM T
CUBEBY $A_1, A_2, \ldots, A_m$
HAVING AVG(M) >= v AND COUNT($*$) >= k

Definition 3 A cell c is said to have n base cells if it covers n nonempty descendant base cells. The **top-k average** of c, denoted as $avg^k(c)$, is the average value of the *top-k base cells* of c (i.e., the first k cells when all the base cells in c are sorted in value-descending order) if $k \leq n$; or $-\infty$ if $k > n$.[3] □

Lemma 3.1 (Top-k Apriori) *Let c be an m-d cell which fails to satisfy $avg^k(c) \geq v$ in cube $\mathcal{B}(AvgI)$. If a cell c' is a descendant of c, then c' cannot satisfy $avg^k(c') \geq v$ in cube $\mathcal{B}(AvgI)$.* □

This lemma stimulates us to explore the utilization of the auxiliary condition $avg^k(c) \geq v$ as a looser bound for computing iceberg cubes with the `HAVING` clause "$avg(c) \geq v$ `AND` $count(c) \geq k$". The effectiveness of the search space pruning by top-k average is demonstrated in our performance study in section 6.

Optimization: Binning technique for top-k average

- Issue: How to keep track of top-k average values? If k is large, it is substantial.
- Solution: Binning technique can be used to reduce
  - the cost of storage and
  - computation of top-k average.
- Two ideas: Large value collapsing and small value binning
- Large value collapsing
  - For any measure value v' which is no less than v (i.e., v' >= v) in $avg^k(c)$ >= v, where v' is called a large value, there is no need to store it explicitly.
  - Instead, it is sufficient to store only two measures: (1) count, the number of large values, and (2) sum, the sum of all large values.
- Small value binning:
  - If the large values registered can make $avg^k(c)$ >= v, there is no need to store small ones (a value v' is small if v' < v).
  - Otherwise, we can set up a small set of bins and register two measures, count and sum, for each bin.
- The large-value group can be considered as a special bin, bin_1.
- Let the upper value boundary of bin_i be $\max(bin_i)$ and the lower one be $\min(bin_i)$.
  - For all (1 <= i < j), we have $\min(bin_i) > \max(bin_j)$.
- To make binning more effective, denser bins are used for the region closer to v, and sparser bins for the region relatively far away from v.
  - For example, suppose v >= 0, one can set up the ranges of five bins.
    - range(bin[1]) = [v; ∞),
    - range(bin[2]) = [0:95v; v),
    - range(bin[3]) = [0:85v; 0:95v),
    - range(bin[4]) = [0:70v; 0:85v), and
    - range(bin[5]) = [0:50v; 0:70v).
- Notice since we have count and sum of all the cells, that for the remaining range [- ∞; 0:50v) can be derived easilv.

The set of bins for a cell c can be used to judge whether $avg^k(c) \geq v$ is false as follows. Let m be the smallest number such that the sum of counts of the upper m bins is no less than k, i.e., $count_m = \Sigma_{i=1}^{m} count(bin_i) \geq k$. We approximate $avg^k(c)$ using,

$avg'^k(c) = (\Sigma_{i=1}^{m-1} sum(bin_i) + max(bin_m) \times n_k)/k$,

where $n_k = k - \Sigma_{i=1}^{m-1} count(bin_i)$.

Lemma 3.2 *$avg^k(c) \leq avg'^k(c)$. Consequently, if $avg'^k(c) < v$, then no descendant of c can satisfy the* Having-*condition in AvgI.* □

- We denote three pieces of information sum, count, and top-k bins as quant-info, which often need to be accumulated with each cell for efficient computation of average iceberg cubes.

Outline

- Introduction
- Iceberg cube with the average measures
- Exploration of Weaker, Anti-monotonic Conditions
- **Extension of Apriori and BUC for Iceberg Cube with Average**
- Top-k H-Cubing: Top-k Cubing Using a Hyper-Tree Structure
- Performance Analysis
- Conclusions

Top-k Apriori

- Scan DB once to accumulate quant-info (i.e., count, sum, and top-k bin measures) for the 0-d cell c0 of the 0-d cuboid.
- Output the 0-d cuboid,
  - $R_0 = \{c_0 \ / \ \text{count}(c_0) >= 50 \ \wedge \ \text{avg}(c_0) = \frac{\text{sum}(c_0)}{\text{count}(c_0)} >= 800\}$, and
  - Keep the 0-d live set, $L_0 = \{c_0 \ / \ avg^{l50}(c_0) >= 800\}$. If L_0 =Φ;, the computation terminates.
- Otherwise, compute 1-d cells as follows.
  - All the 1-d cells are candidate cells, i.e., forming the candidate set C_1, such as (Jan; ; ; : : :), (Feb; ; ; : : :), . . . , (; T oronto; ; : : :), (; Vancouver; ; : : :), ...
  - Then scan DB, accumulate quant-info for each c_1 in C_1, output R_1, and keep the live set L_1:
    - $R_1 = \{c_1 \ / \ c_1 \ \varepsilon \ C_1 \ \wedge \ \text{count}(c_1) >= 50 \ \wedge \ \text{avg}(c_1) = \frac{\text{sum}(c_1)}{\text{count}(c_1)} >= 800\}$
    - $L_1 = \{c_1 \ / \ c_1 \ \varepsilon \ C_1 \wedge avg^{l50}(c_1) >= 800\}$
- This process continues level-by-level, until the live set L_k or the candidate set C_k for some k is empty
- Top-k Apriori computes average iceberg cubes by exploring candidate generation and level-wise computation.
- However, it still involves costly processing:
  - (1) it takes m scans of DB where m is the maximum number of dimensions containing nonempty candidate set, and
  - (2) it may generate a huge number of candidate sets.

Top-k BUC

- An efficient iceberg cube computation method BottomUp Cube (BUC) builds the cube from lower number of dimension combinations to higher ones.
- It explores the dimension ordering by putting the most discriminating dimensions rst and then recursively partitioning DB according to the ordering.
- At each step of recursive partition,
  - one can push in the iceberg constraint, such as min count, to remove those that cannot satisfy it.
  - This can be applied to computing iceberg cubes with the average measure.
    - For example, for computing AvgI, one can use $avg^k(c) > v$ to test the partitions generated: any partition that cannot pass the test will not need to be considered further

Outline

- Introduction
- Iceberg cube with the average measures
- Exploration of Weaker, Anti-monotonic Conditions
- Extension of Apriori and BUC for Iceberg Cube with Average
- **Top-k H-Cubing: Top-k Cubing Using a Hyper-Tree Structure**
- Performance Analysis
- Conclusions

Top-k H-Cubing: Top-k Cubing Using a Hyper-Tree Structure

- Example: a tree structure HT structure
- Tree HT has a root node null. and dimensions are in cardinality-ascending order, i.e. R: G, M, C.
- A header table is created, in which each entry records the quant-info for an attribute-value pair.
  - The rst tuple, t1 = (Edu:; J an; T oronto; 485), is inserted into HT , with three nodes, Edu:, Jan and Toronto inserted in sequence to form the rst branch, and quant-info in the leaf(Toronto).
  - Also, price 485 is used to update quant-info for Edu:, Jan and Toronto in the header table.
  - Similarly, t2 = (Household; Jan; Toronto; 1200), is inserted. Since the two leaf nodes have the same label, they are linked by a side-link.
  - Since t3 = (Edu:; Jan; Toronto; 1280) has the same attribute values as t1, t3 shares the path as t1, with quant-info in the leaf and header updated.
- The remaining tuples can be inserted similarly (Fig. 2, next slide).
- The tree so formed is called an H-tree. Its construction requires only one scan of the database.

Mon.	Day	City	Cust_group	Product	Cost	Price
Jan	10	Toronto	Edu.	HP Printer	500	485
Jan	15	Toronto	Household	Sony TV	800	1200
Jan	20	Toronto	Edu.	Canon Camera	1160	1280
Feb	20	Montreal	Busi.	IBM Laptop	1500	2500
Mar.	4	Vancouver	Edu.	Seagate HD	540	520
...	...	...	...	...	...	...

Table 1: A *Sales_Info* table.

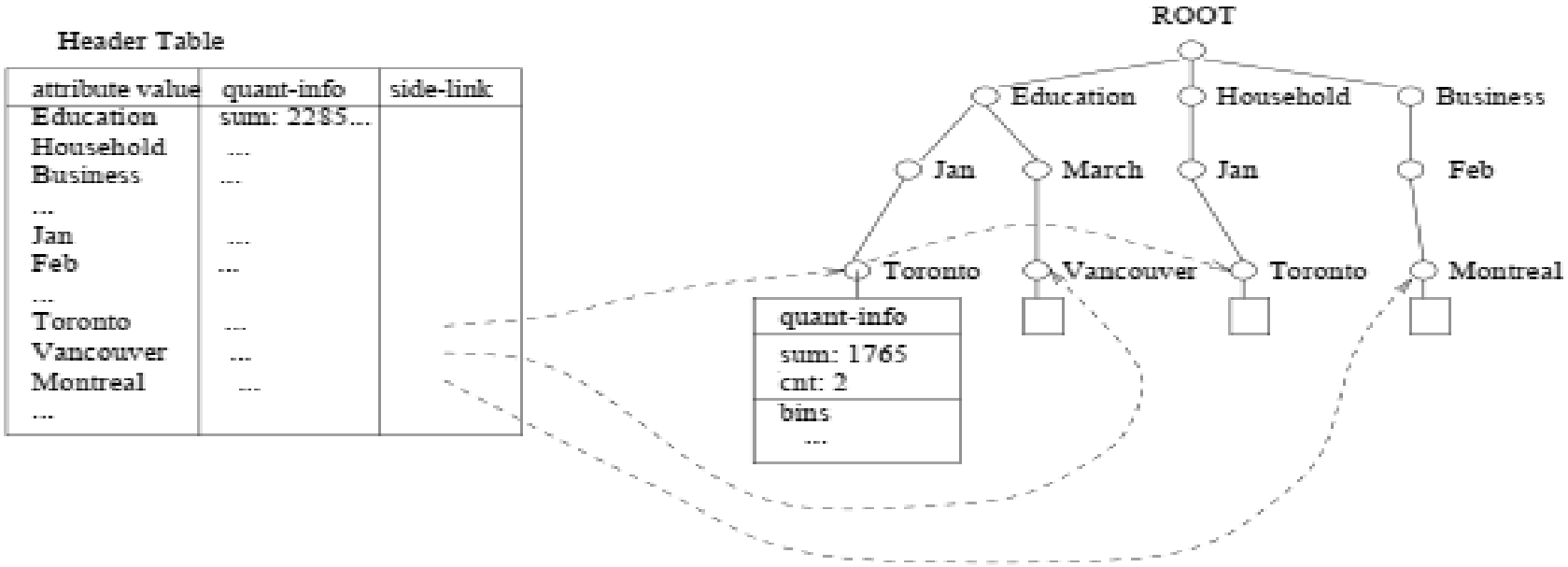


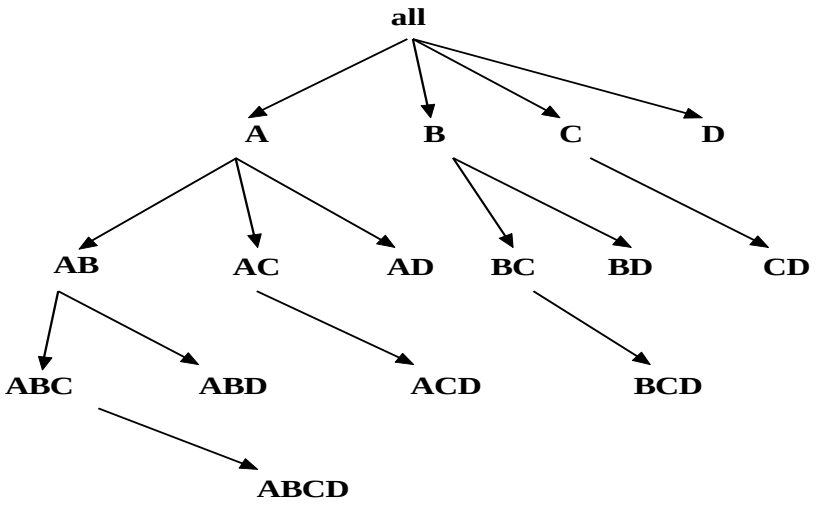
Figure 2: An H-tree.

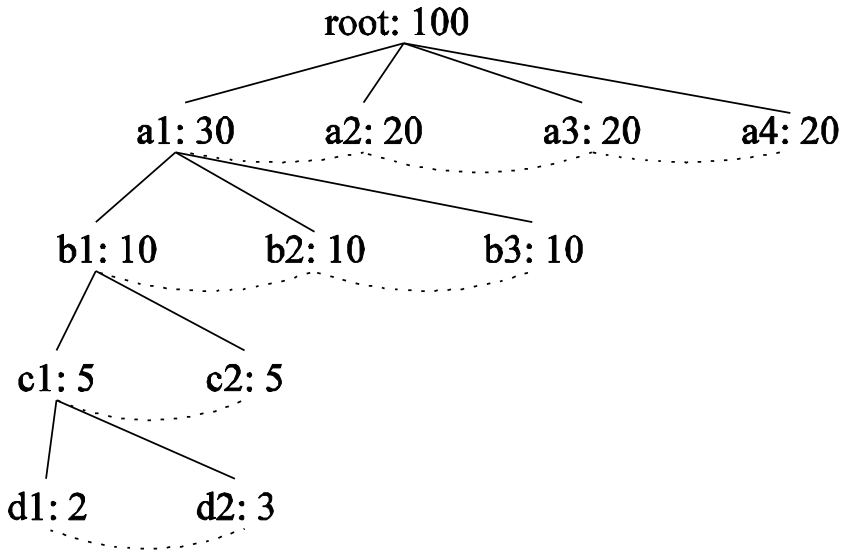
Properties of H-tree

- The H-tree HT has the following properties.
- 1. (Construction cost) The H-tree can be constructed by scanning the database only once.
- 2. (Completeness) The H-tree and its header table H contain the complete information needed for computing iceberg cube AvgI.
- 3. (Compactness) Let there be n tuples in table T and m attributes involved in AvgI. The number of nodes in H-tree cannot exceed n*m + 1.

H-Cubing: Using H-Tree Structure

- Bottom-up computation
- Exploring an H-tree structure
- If the current computation of an H-tree cannot pass min_sup, do not proceed further (pruning)
- No simultaneous aggregation





Top-k H-Cubing: Computing iceberg cubes using H-tree

- Step 1. Compute cells involving dimension C.
- The quant-info in the H-tree tells whether a cell in the form of (; ; c), where c is a city, passes the top-k average and average tests.
- For example, the entry Toronto in the header table contains $avg^k(c)$, $avg^k(price)$ and avg(price) for (*; *; Toronto).
- If avg(price) passes the average price threshold, output the cell.
- If $avg^k(price)$ passes it, the descendants of the cell (*; *;Toronto) should be examined as shown below.
  - The sub-graph of HT containing only the paths related to Toronto, denoted as $HT_{Toronto}$, is an H-tree for sub-cube (*;*; Toronto). $HT_{Toronto}$ is sufficient to compute the iceberg sub-cube w.r.t. Toronto.
  - Traverse the side-link Toronto in the header table H and
    - (1) make a copy of quant-info in every leaf-node labeled Toronto to its parent node in the tree,
    - (2) build a new header table $H_{Toronto}$, which collects quant-info for every attribute-value w.r.t. Toronto, and
    - (3) link all the parent nodes of the leaf-nodes labeled Toronto. (Fig. 3 is the updated tree)
  - Based on the header table $H_{Toronto}$, output all the cells of the form (*; m; Toronto) or (g; *; Toronto), which pass the average price test, where m ε M and g ε G. Also, explore recursively the descendants of the cells of the form (*; m; Toronto) or (g; *; Toronto) which pass the top-k average test.

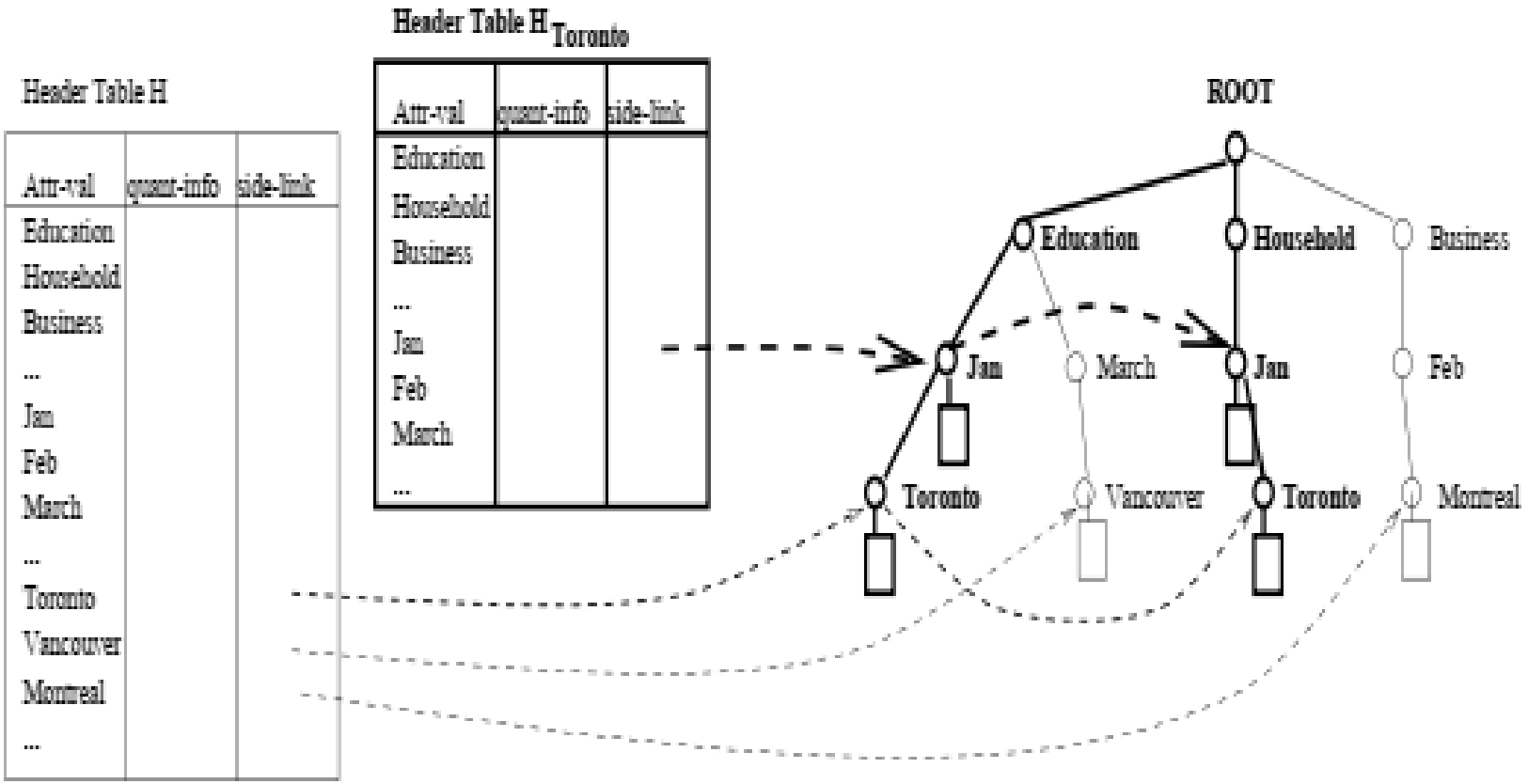


Figure 3: Updated H-tree for computing descendants of $(*, *, Toronto)$.

Step 2. Compute cells involving dimension M but no C

- 1. Roll-up quant-info to dimension M.
  - Every leaf node in H-tree merges its quant-info into that of its parent node. All nodes labeled by a common month, should be linked by side-links and also linked to the corresponding row in the header table H.
- 2. Compute cells involving M but no C. This is similar to Step 1 demonstrated before.

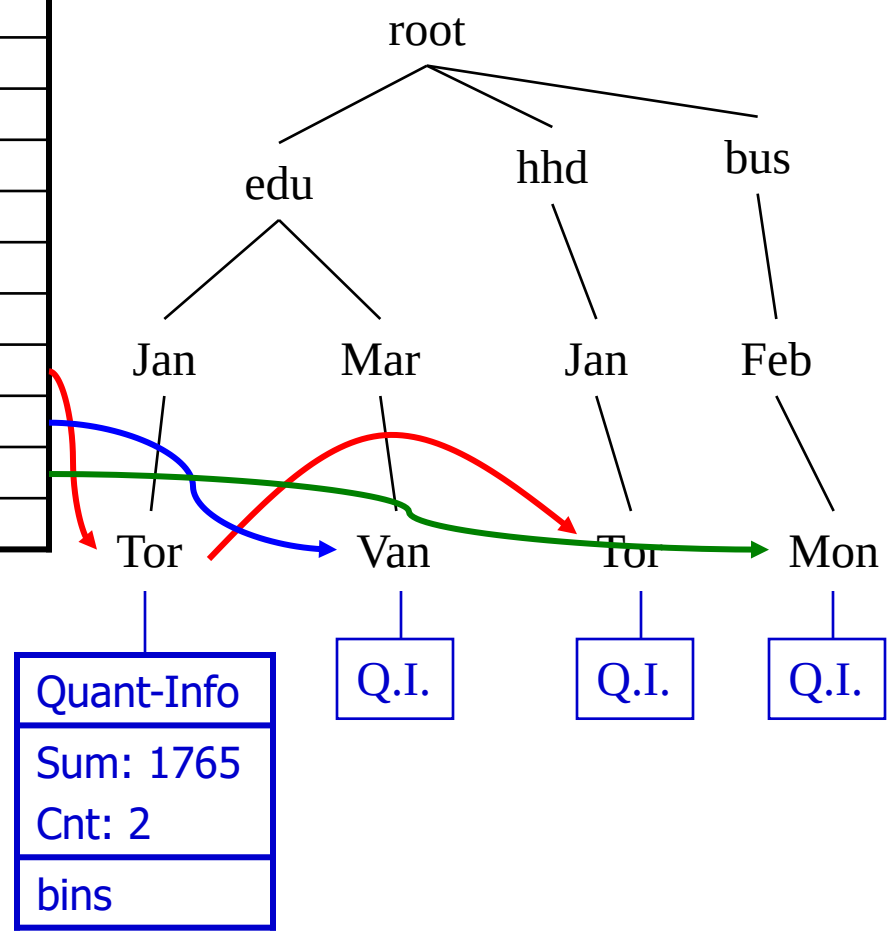
Reduction of Database Scans

- When the H-tree for a partition can not fit in memory, we turn to H-tree –based mining.
- We can use a a dual projection method, which requires less memory and disk space but scans DB and each partitioned database only once.

H-tree: A Prefix Hyper-tree

Header table

Attr. Val.	Quant-Info	Side-link
Edu	Sum:2285 …	
Hhd	…	
Bus	…	
…	…	
Jan	…	
Feb	…	
…	…	
Tor	**…**	
Van	**…**	
Mon	**…**	
…	…	



Month	City	Cust_grp	Prod	Cost	Price
Jan	Tor	Edu	Printer	500	485
Jan	Tor	Hhd	TV	800	1200
Jan	Tor	Edu	Camera	1160	1280
Feb	Mon	Bus	Laptop	1500	2500
Mar	Van	Edu	HD	540	520
…	…	…	…	…	…

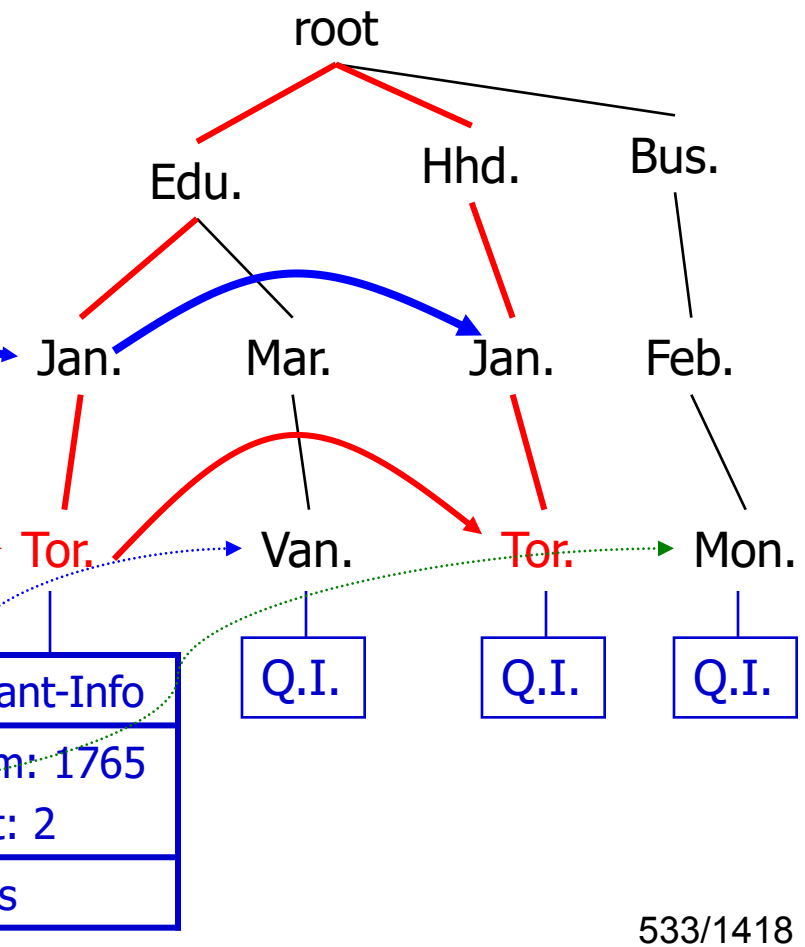
Computing Cells Involving "City"

Header Table H_{Tor}

Attr. Val.	Q.I.	Side-link
Edu	…	
Hhd	…	
Bus	…	
…	…	
Jan	**…**	
Feb	…	
…	…	

From (*, *, Tor) to (*, Jan, Tor)

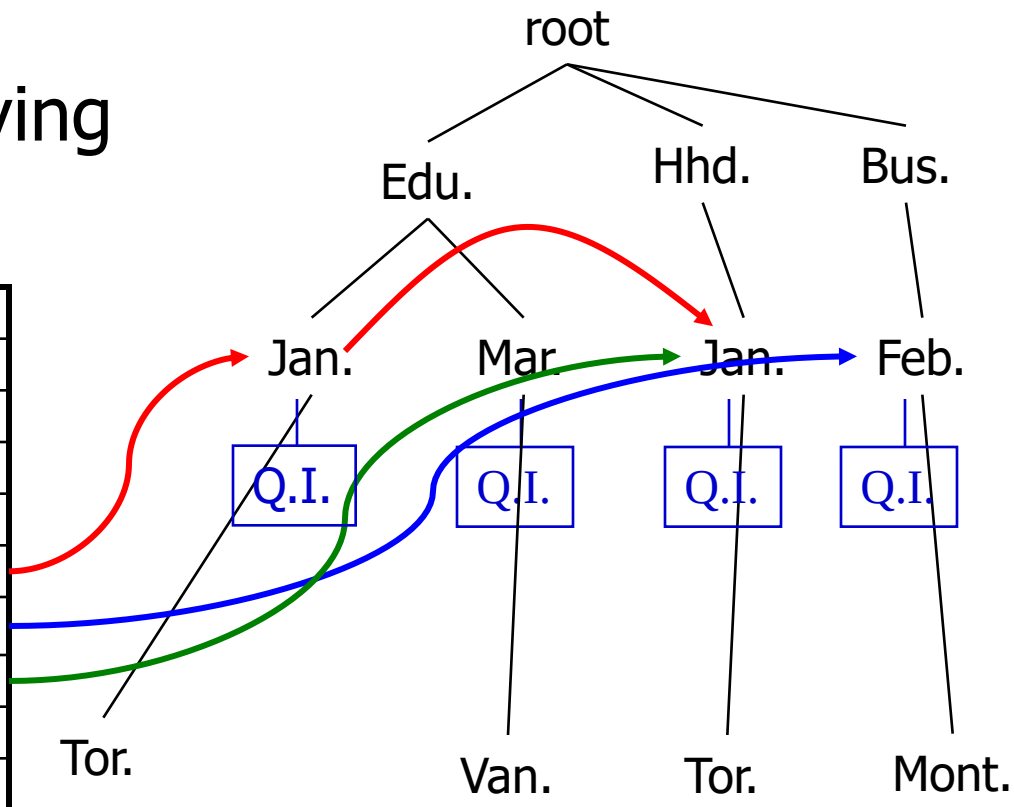
Attr. Val.	Quant-Info	Side-link
Edu	Sum:2285 …	
Hhd	…	
Bus	…	
…	…	
Jan	…	
Feb	…	
…	…	
Tor	**…**	
Van	…	
Mon	…	
…	…	



Computing Cells Involving Month But No City

1. Roll up quant-info
2. Compute cells involving month but no city

Attr. Val.	Quant-Info	Side-link
Edu.	Sum:2285 …	
Hhd.	…	
Bus.	…	
…	…	
Jan.	**…**	
Feb.	**…**	
Mar.	**…**	
…	…	
Tor.	…	
Van.	…	
Mont.	…	
…	…	

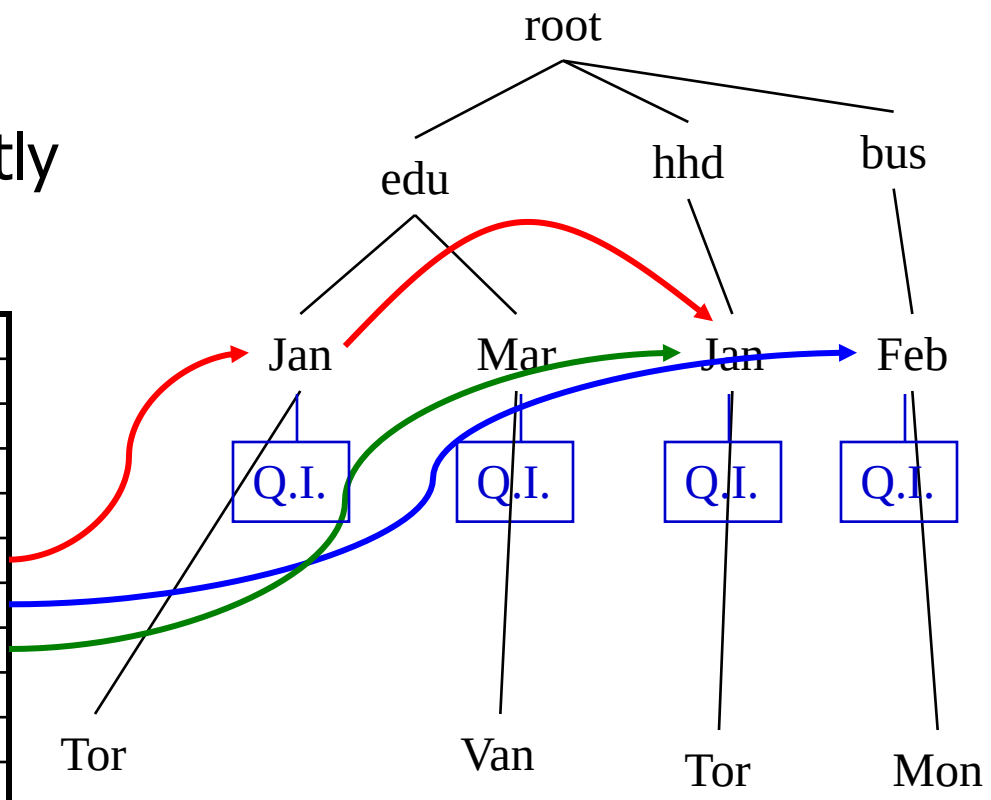


Top-k OK mark: if Q.I. in a child passes top-k avg threshold, so does its parents. No binning is needed!

Computing Cells Involving Only Cust_grp

Check header table directly

Attr. Val.	Quant-Info	Side-link
Edu	Sum:2285 …	
Hhd	…	
Bus	…	
…	…	
Jan	…	
Feb	…	
Mar	…	
…	…	
Tor	…	
Van	…	
Mon	…	
…	…	



Performance Analysis

- See the paper

Discussion

- The proposed approach can be extended to other measures, in addition to average.

Summary

- Studies issues on efficient computation of iceberg cubes with some popularly encountered complex measures.
- It contributes to iceberg cube computation in two aspects:
  - A methodology is developed that derives a weaker but antimonotonic condition for testing and pruning search space
  - Instead of simple extension of two previously studied methods, Apriori and BUC, to Top-k Apriori and Top-k BUC, an interesting hypertree structure, called H-tree, is designed and a new iceberg cubing method, is developed.
- Top-k H-Cubing performs better than the existing method.
- Remark
  - Interesting progress has been made for efficient computation of iceberg cubes with some complex measures.
  - Efficient computation of iceberg cubes with some other complex measures is still an open problem.
  - Moreover, the application of the H-tree structure and its computation method to other OLAP and data mining tasks may deserve further attention.

Outline

- Data Cube Computation: Preliminary Concepts
- Computing full/iceberg cubes: four methodologies
  - 1.General heuristics: On the computation of multidimensional aggregates (Agarwal et al.'96)
  - 2.Bottom-Up: Multiway Array Aggregation for Full Cube Computation (Zhao et al SIGMOD'97)
  - 3. Top-down: BUC: Computing Iceberg Cubes from the Apex Cuboid Downward (Beyer et al SIGMOD'99)
  - 4. Top-down: H-Cubing: Exploring an H-Tree Structure (Han et al SIGMOD'01)
- Overview of Other methodologies
    - **5.Star-cubing: Computing Iceberg Cubes Using a Dynamic Star-tree Structure (Xin et al VLDB'03)**
    - 6.Precomputing Shell Fragments for Fast High-Dimensional OLAP (Li, et al. VLDB'04)
- Overview of Data Cubes for Advanced Applications
  - 7. Sampling Cubes: OLAP on Sampling Data (Li et al SIGMOD 08)
  - 8. Ranking Cubes: Efficient Computation of Ranking Queries (Xin et al VLDB06)
- Overview of Knowledge Discovery with Data Cubes
  - **9**. Prediction Cubes: Data Mining in Multi-Dimensional Cube Space (Chen et al VLDB'05)
  - 10**.** Complex Aggregation at Multiple Granularity: Multi-feature Cubes (Ross et al EDBT 1998)
  - 11**.** Discovery-Driven Data Cubes (Sarawagi et al EDBT 1998)
- Summary

Star-Cubing: Computing Iceberg Cubes by Top-Down and Bottom-Up Integration*

Dong Xin Jiawei Han Xiaolei Li Benjamin W. Wah

University of Illinois at Urbana-Champaign, Urbana, IL 61801, USA

VLDB 2003

Presented by

P. Krishna Reddy, IIIT Hyderabad

Outline

- Introduction
- Overview of cubing algorithms
- Start-Cubing: An Integration of top-down and bottom-up approach
- Performance Analysis
- Conclusion

Introduction

- Data cube computation is one of the most essential but expensive operations in data warehousing.
- Previous studies have developed two major approaches, top-down vs. bottom-up, for efficient cube computation.
  - MultiWay Array Cube (called MultiWay) algorithm aggregates simultaneously on multiple dimensions;
    - however, it cannot take advantage of Apriori pruning when computing iceberg cubes.
  - BUC and H-Cubing, computes the iceberg cube bottom-up and facilitates Apriori pruning.
    - BUC explores fast sorting and partitioning techniques; whereas
    - H-Cubing explores a data structure, H-Tree, for shared computation.
- In this paper, we present a new method, StarCubing, that integrates the strengths of the previous three algorithms and performs aggregations on multiple dimensions simultaneously.
- It utilizes a star-tree structure, extends the simultaneous aggregation methods, and enables the pruning of the group-by's that do not satisfy the iceberg condition.

Outline

- Introduction
- **Overview of cubing algorithms**
- Start-Cubing: An Integration of top-down and bottom-up approach
- Performance Analysis
- Conclusion

MultiWay

- MultiWay is an array-based top-down cubing algorithm.
- It uses a compressed sparse array structure to load the base cuboid and compute the cube.
- In order to save memory usage, the array structure is partitioned into chunks. It is unnecessary to keep all the chunks in memory since only parts of the group-by arrays are needed at any time.
- By carefully arranging the chunk computation order, multiple cuboids can be computed simultaneously in one pass
- Taking ABCD as the base cuboid, the results of computing cuboid ACD can be used to compute AD, which in turn can be used to compute A.
- This shared computation makes MultiWay perform aggregations simultaneously on multiple dimensions, which leads to the computation order
- Intermediate aggregate values can be re-used for the computation of successive descendant cuboids.

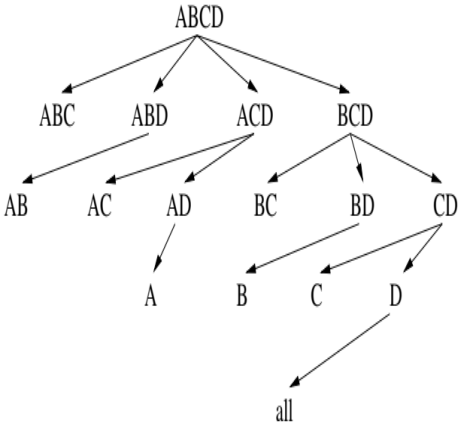


Figure 1: Top-Down Computation

About Multyway

- +ves
  - Good for moderate number of dimensions
- -ves
  - If the dimensionality is high and the data is too sparse, the method becomes infeasible because the arrays and intermediate results become too large to fit in memory.
  - The top-down algorithm cannot take advantage of Apriori pruning during iceberg cubing, i.e., the iceberg condition can only be used after the whole cube is computed.
    - This is because the successive computation does not have the anti-monotonic property .
      - If a cell in ABD does not satisfy min sup, one cannot assert that its children cell" in the cuboid AB does not satisfy min sup either since a cell in AB is likely to contain more base cells than that in ABD.

BUC

- BUC employs a bottom-up computation by starting at the apex cuboid
- BUC starts by reading the first dimension and partitioning it based on its distinct values. Then for each partition in the first dimension, it recursively computes the remaining dimensions.

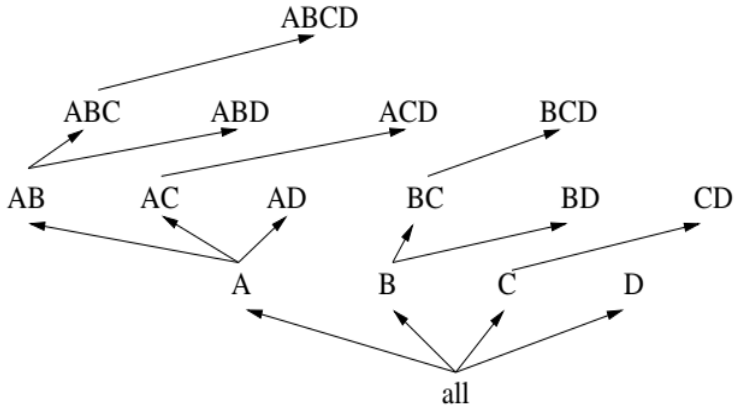


Figure 2: **Bottom-Up Computation**

About BUC

- +ves
  - Apriori pruning is used to reduce unnecessary computation based on the anti-monotonic property, which was not possible in the top-down computation.
    - For example, if the count of a cell c in a cuboid A is smaller than min sup, then the count of any descendant cells of c (with more dimensions, e.g., AC, ACD) can never be higher than min sup. Thus the descendant cells of c can be pruned.
- -ves
  - Partitioning and sorting are the major costs in BUC's cube computation. Since recursive partitioning in BUC does not reduce the input size, both partition and aggregation are costly.
  - Moreover, BUC is sensitive to skew in the data: the performance of BUC degrades as skew increases.
  - Unlike MultiWay, the results of a parent cuboid does not help compute that of its children in BUC.
    - For example, the computation of cuboid AB does not help that of ABC. The latter needs to be computed essentially from scratch.

H-Cubing

- H-Cubing uses a hyper-tree structure, called HTree, to facilitate cube computation.
  - Each level in the tree represents a dimension in the base cuboid. A base tuple (cell) of d-dimensions forms one path of length d (i.e., d nodes) in the tree.
- Nodes at the same level of tree that hold the same value are linked together via side-links.
- In addition, a Header Table is associated with each level to record the count of every distinct value in all the dimensions not below the current one, and provides links to the first node of the corresponding values in the H-Tree.
- With this data structure, there are two methods to compute the cube: bottom-up (BOT) vs. top-down (TOP) tree traversal.
- In both methods, the algorithm starts at a particular level of the H-Tree, i.e., a particular dimension, and examines the group-by's that include that level and levels above it in the H-Tree.
- The aggregation is facilitated by the Header Table constructed in the initial pass of the data and Header Tables local to the current group-by.
- During the aggregation, if the count of a particular node is below min sup, it skips itself and jumps to the next node via the side-link.
- The difference between the two traversal methods, H-Cubing-BOT and H-Cubing-TOP, is that the former starts the process at the bottom of the HTree, whereas the latter starts at the top

About H-cube

- +ves
  - One advantage of H-Cubing is that since the internal nodes of the H-Tree structure collapse duplicated data, shared processing and some simultaneous aggregation can be explored.
  - Also, it computes less dimension combinations before proceedings to more dimensions, and thus leads to Apriori pruning for iceberg cube computation.
- -ves
  - However, similar to BUC, it cannot use the intermediate results at computing low dimensions to facilitate the computation of high dimensional cuboids.

Outline

- Introduction
- Overview of cubing algorithms
- **Start-Cubing: An Integration of top-down and bottom-up approach**
- Performance Analysis
- Conclusion

Algorithm	Simultaneous Aggregation	Partition & Prune
MultiWay	Yes	No
BUC	No	Yes
H-Cubing	Weak	Yes
Star-Cubing	Yes	Yes

Table 1: **Summary of Four Algorithms**

Start Cubing

- The Star-Cubing algorithm explores both the topdown and bottom-up models:
- On the global computation order, it uses the top-down model similar to Figure 1.
- However, it has a sub-layer underneath based on the bottom-up model by exploring the notion of shared dimension.
- Main concepts
  - Shared dimensions
  - Star Nodes and Star Trees
  - Cuboid Trees

Shared Dimensions

- Observe Figure 1
  - all the cuboids in the left-most sub-tree of the root include dimensions ABC,
  - all those in the second sub-tree include dimensions AB,
  - and all those in the third include dimension A.
- We call these common dimensions the shared dimensions of those particular sub-trees.
- Based on this concept, Figure 1 is extended to Figure 3, which shows the spanning tree marked with the shared dimensions.
  - For example, ABD/AB means cuboid ABD has shared dimension AB, ACD/A means cuboids ACD has shared dimension A, and so on

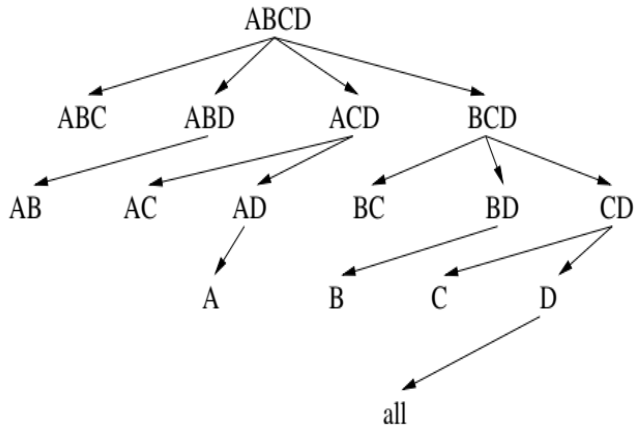


Figure 1: **Top-Down Computation**

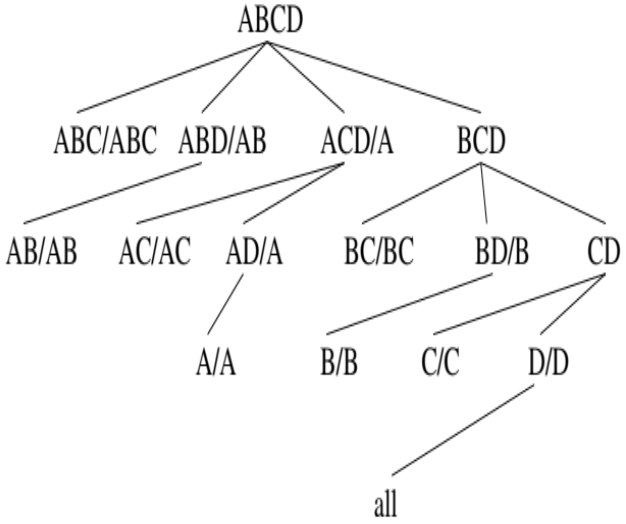


Figure 3: **Star-Cubing: Top-down computation with bottom-up growing shared dimensions**

Apriori Pruning

- Anti-monotonic property of shared dimensions
  - If the measure is *anti-monotonic*, and if the aggregate value on a shared dimension does not satisfy the *iceberg condition*, then all the cells extended from this shared dimension cannot satisfy the condition either
- Intuition: if we can compute the shared dimensions before the actual cuboid, we can use them to do `Apriori` pruning
  - As an example, if the value in the shared dimension A is a1 and it fails to satisfy the iceberg condition, the whole sub-tree rooted at a1CD/a1 (including a1C/a1C, a1D/a1, a1/a1) can be pruned

- ***This is nothing but the Exploitation of BUC***

Cuboid Trees

- We use trees to represent individual cuboids.
- Figure 4 shows a fragment of the cuboid tree of the base cuboid ABCD
- Each node has four flelds:
  - the attribute value, aggregate value, pointer(s) to possible descendant(s), and pointer to possible sibling.
  - Tuples in the cuboid are inserted one by one into the tree. A path from the root to a node represents a tuple. For example, node c2 in the tree has aggregate (count) value of 5, which indicates that there are flve cells of value {a1 b1 c2 *}

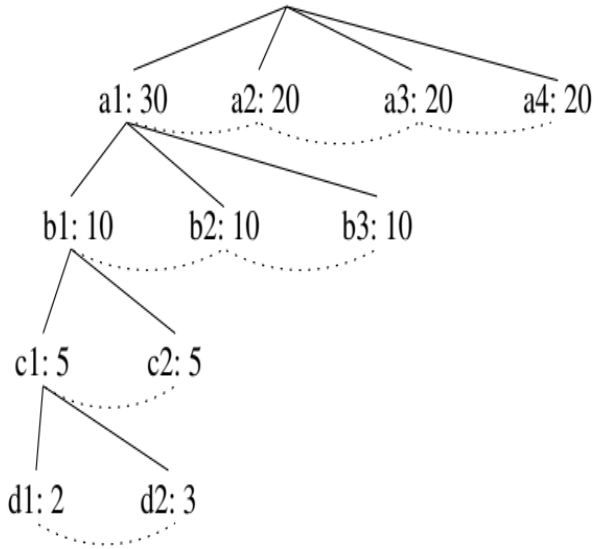


Figure 4: A fragment of the base cuboid tree

- ***This is nothing but the Exploiting of H-cube***

Star Nodes and Star Trees

- If the single dimensional aggregate on an attribute value p does not satisfy the iceberg condition, it is useless to distinguish such nodes in the iceberg cube computation.
- Thus the node p can be replaced by * so that the cuboid tree can be further compressed.
- This motivates us to introduce the concepts of star node and star tree.
- The node p in an attribute A is a star node if the single dimensional aggregate on p does not satisfy the iceberg condition; otherwise, p is a non-star node.
- A cuboid tree that consists of only non-star nodes and star(-replaced) nodes is called a star-tree.

A	B	C	D	Count
$a1$	$b1$	$c1$	$d1$	1
$a1$	$b1$	$c3$	$d3$	1
$a1$	$b2$	$c2$	$d2$	1
$a2$	$b3$	$c3$	$d4$	1
$a2$	$b4$	$c3$	$d4$	1

Table 2: **Base (Cuboid) Table**: Before star reductior

Dimension	Count $= 1$	Count ≥ 2
A	-	$a1(3), a2(2)$
B	$b2, b3, b4$	$b1(2)$
C	$c1, c2$	$c3(3)$
D	$d1, d2, d3$	$d4(2)$

Table 3: **One-Dimensional Aggregates**

A	B	C	D	Count
$a1$	$b1$	$*$	$*$	2
$a1$	$*$	$*$	$*$	1
$a2$	$*$	$c3$	$d4$	2

Table 4: **Compressed Base Table**: After star reduction.

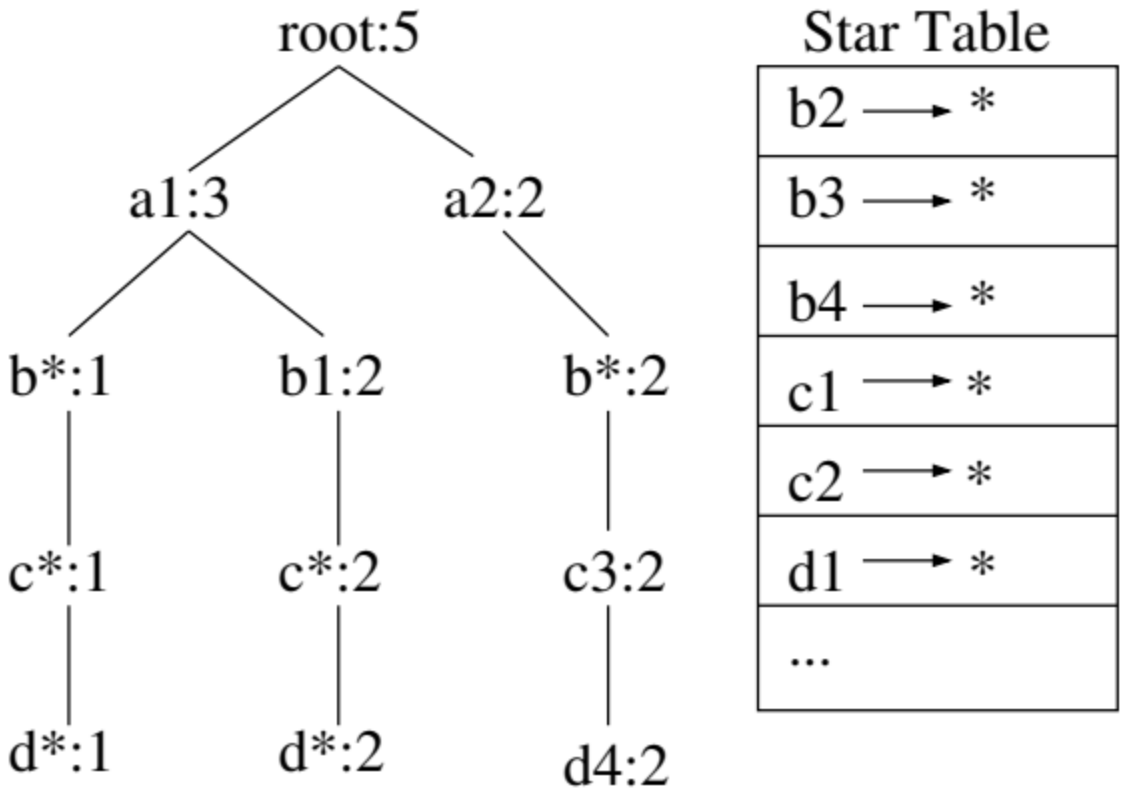


Figure 5: **Star Tree and Star Table**

Performance

- See the paper

Summary of Star CUBE

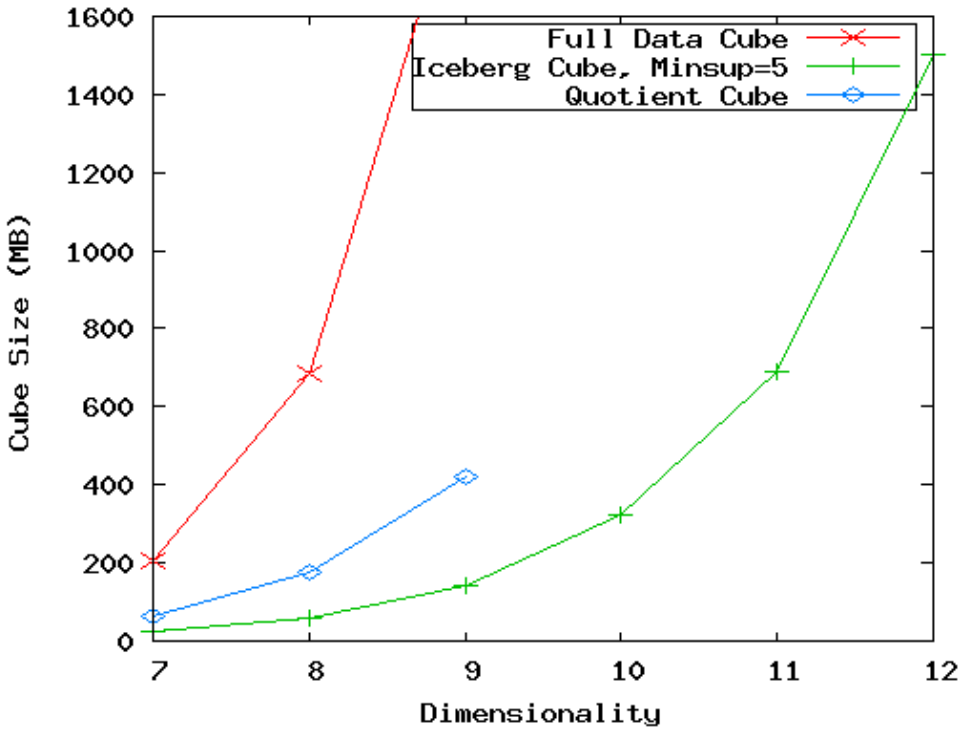
- An interesting cube computation method, Star-Cubing, is proposed
- Integrates the strength of both top-down and bottom-up cube computation
- Exploits
  - shared aggregation by taking advantage of shared dimensions among the current cuboid and its descendant cuboids; and
  - prune as soon as possible the unpromising cells during the cube computation using the anti-monotonic property of the iceberg cube measure.
- Moreover, a new compressed data structure, star-tree is proposed using star nodes. And a few other optimization techniques also contribute to the high performance of the method

Outline

- Data Cube Computation: Preliminary Concepts
- Computing full/iceberg cubes: four methodologies
  - 1.General heuristics: On the computation of multidimensional aggregates (Agarwal et al.'96)
  - 2.Bottom-Up: Multiway Array Aggregation for Full Cube Computation (Zhao et al SIGMOD'97)
  - 3. Top-down: BUC: Computing Iceberg Cubes from the Apex Cuboid Downward (Beyer et al SIGMOD'99)
  - 4. Top-down: H-Cubing: Exploring an H-Tree Structure (Han et al SIGMOD'01)
- Overview of Other methodologies
    - 5.Star-cubing: Computing Iceberg Cubes Using a Dynamic Star-tree Structure (Xin et al VLDB'03)
    - **6.Precomputing Shell Fragments for Fast High-Dimensional OLAP (Li, et al. VLDB'04)**
- Overview of Data Cubes for Advanced Applications
  - 7. Sampling Cubes: OLAP on Sampling Data (Li et al SIGMOD 08)
  - 8. Ranking Cubes: Efficient Computation of Ranking Queries (Xin et al VLDB06)
- Overview of Knowledge Discovery with Data Cubes
  - **9**. Prediction Cubes: Data Mining in Multi-Dimensional Cube Space (Chen et al VLDB'05)
  - 10**.** Complex Aggregation at Multiple Granularity: Multi-feature Cubes (Ross et al EDBT 1998)
  - 11**.** Discovery-Driven Data Cubes (Sarawagi et al EDBT 1998)
- Summary

The Curse of Dimensionality

- None of the previous cubing method can handle high dimensionality!
- A database of 600k tuples. Each dimension has cardinality of 100.



Motivation of High-D OLAP

- X. Li, J. Han, and H. Gonzalez, High-Dimensional OLAP: A Minimal Cubing Approach, VLDB'04
- Challenge to current cubing methods:
  - The “curse of dimensionality” problem
  - Iceberg cube and compressed cubes: only delay the inevitable explosion
  - Full materialization: still significant overhead in accessing results on disk
- High-D OLAP is needed in applications
  - Science and engineering analysis
  - Bio-data analysis: thousands of genes
  - Statistical surveys: hundreds of variables

Fast High-D OLAP with Minimal Cubing

- Observation: OLAP occurs only on a small subset of dimensions at a time
- Semi-Online Computational Model
  1. Partition the set of dimensions into **shell fragments**
  2. Compute data cubes for each shell fragment while retaining **inverted indices** or **value-list indices**
  3. Given the pre-computed **fragment cubes**, dynamically compute cube cells of the high-dimensional data cube *online*

Properties of Proposed Method

- Partitions the data vertically
- Reduces high-dimensional cube into a set of lower dimensional cubes
- Online re-construction of original high-dimensional space
- Lossless reduction
- Offers tradeoffs between the amount of pre-processing and the speed of online computation

Example Computation

- Let the cube aggregation function be `count`

tid	**A**	**B**	**C**	**D**	**E**
1	a1	b1	c1	d1	e1
2	a1	b2	c1	d2	e1
3	a1	b2	c1	d1	e2
4	a2	b1	c1	d1	e2
5	a2	b1	c1	d1	e3

- Divide the 5 dimensions into 2 shell fragments:
  - (`A, B, C`) and (`D, E`)

1-D Inverted Indices

- Build traditional invert index or RID list

Attribute Value	TID List	List Size
a1	1 2 3	3
a2	4 5	2
b1	1 4 5	3
b2	2 3	2
c1	1 2 3 4 5	5
d1	1 3 4 5	4
d2	2	1
e1	1 2	2
e2	3 4	2
e3	5	1

Shell Fragment Cubes: Ideas

- Generalize the 1-D inverted indices to multi-dimensional ones in the data cube sense
- Compute all cuboids for data cubes ABC and DE while retaining the inverted indices
- For example, shell fragment cube ABC contains 7 cuboids:
  - A, B, C
  - AB, AC, BC
  - ABC
- This completes the offline computation stage

Cell	Intersection	TID List	List Size
a1 b1	1 2 3 ∩1 4 5	1	1
a1 b2	1 2 3 ∩2 3	2 3	2
a2 b1	4 5 ∩1 4 5	4 5	2
a2 b2	4 5 ∩2 3	⊗	0

Shell Fragment Cubes: Size and Design

- Given a database of T tuples, D dimensions, and F shell fragment size, the fragment cubes' space requirement is:
  - For F < 5, the growth is sub-linear $O\left(T\left\lceil\frac{D}{F}\right\rceil(2^F-1)\right)$
- Shell fragments do not have to be disjoint
- Fragment groupings can be arbitrary to allow for maximum online performance
  - Known common combinations (e.g.,<city, state>) should be grouped together.
- Shell fragment sizes can be adjusted for optimal balance between offline and online computation

ID_Measure Table

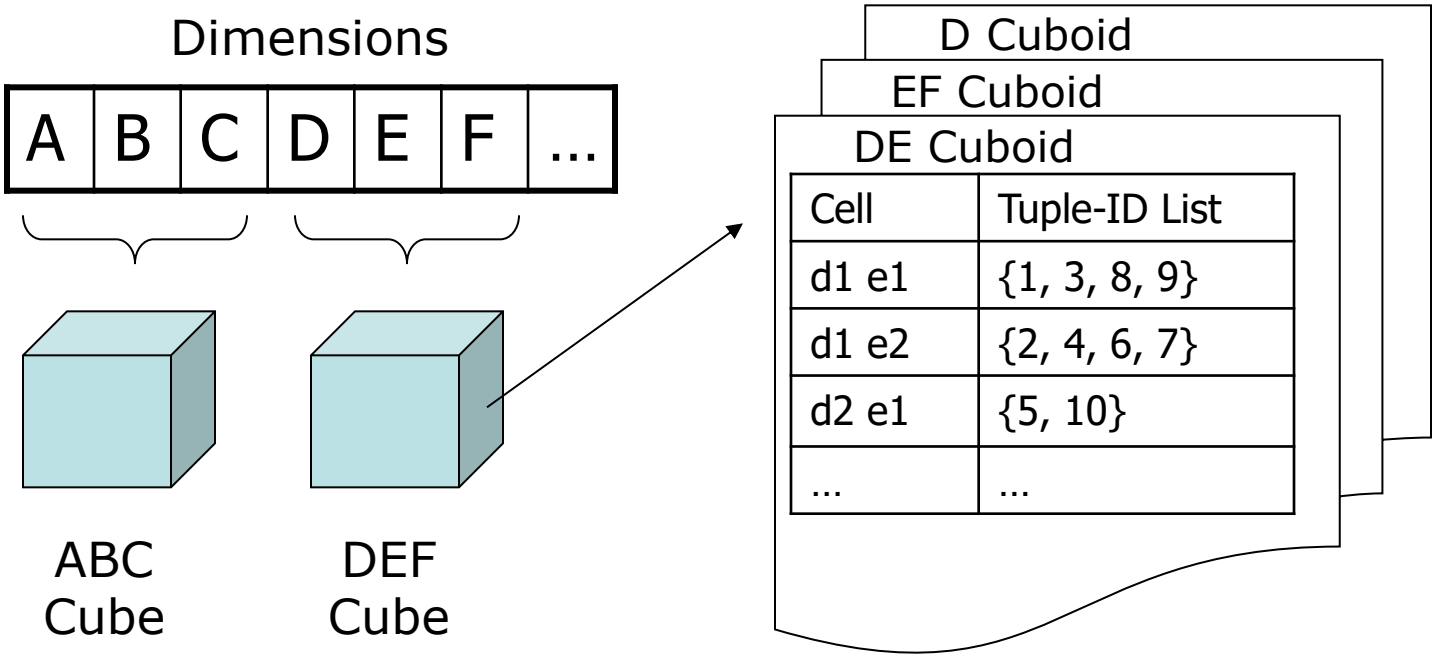
- If measures other than `count` are present, store in *ID_measure* table separate from the shell fragments

tid	count	sum
1	5	70
2	3	10
3	8	20
4	5	40
5	2	30

The Frag-Shells Algorithm

1. Partition set of dimension ($A_1,\ldots,A_n$) into a set of *k* fragments ($P_1,\ldots,P_k$).
2. Scan base table once and do the following
3. insert <tid, measure> into ID_measure table.
4. for each attribute value a_i of each dimension A_i
5. build inverted index entry <a_i, tidlist>
6. For each fragment partition P_i
7. build local fragment cube S_i by intersecting tid-lists in bottom-up fashion.

Frag-Shells (2)



Online Query Computation: Query

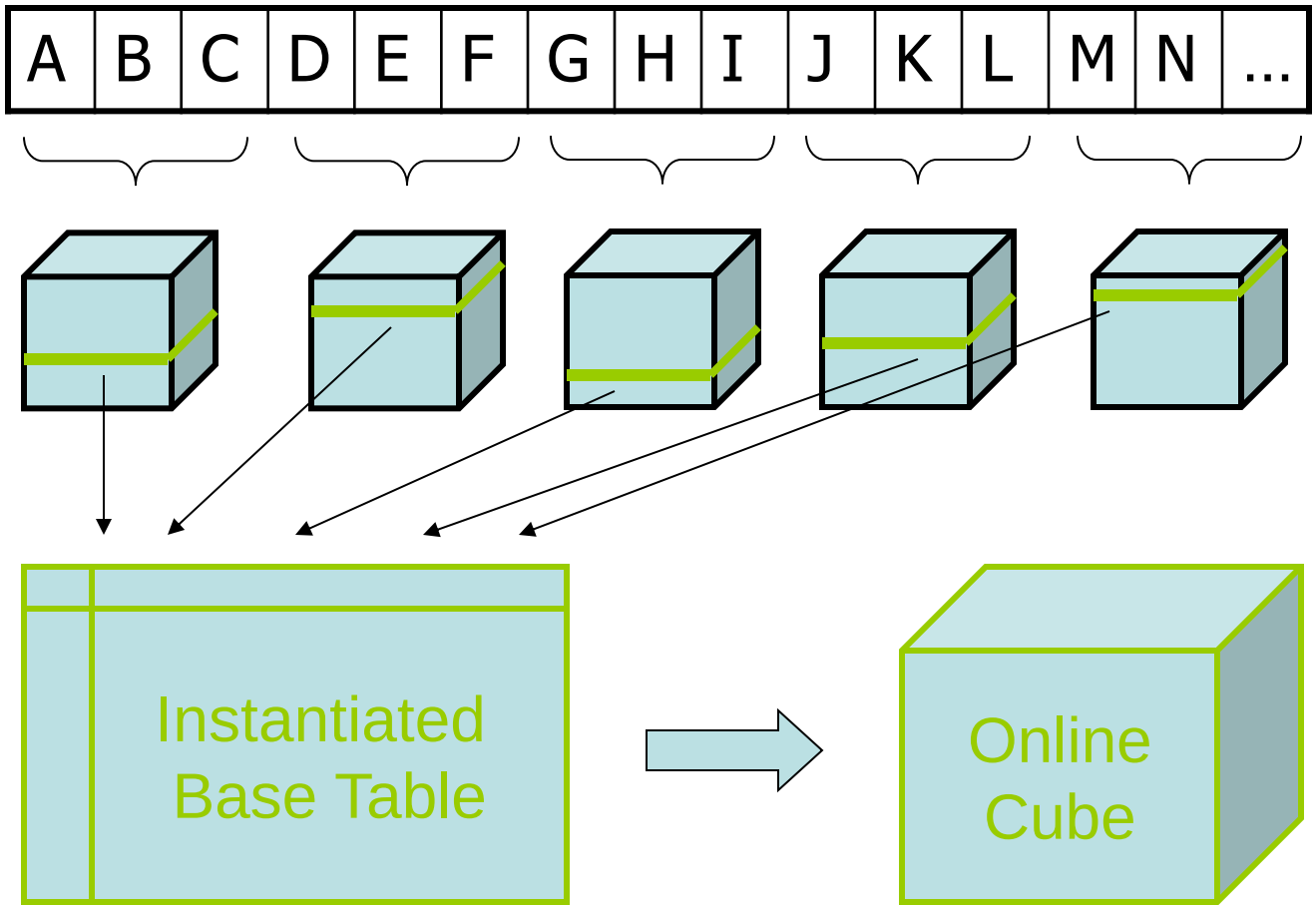
- A query has the general form $\langle a_1, a_2, \ldots, a_n : M \rangle$
- Each a_i has 3 possible values
  1. Instantiated value
  2. Aggregate * function
  3. Inquire ? function

■ For example, $\langle 3 \quad ? \quad ? \quad * \quad 1 : count \rangle$ returns a 2-D data cube.

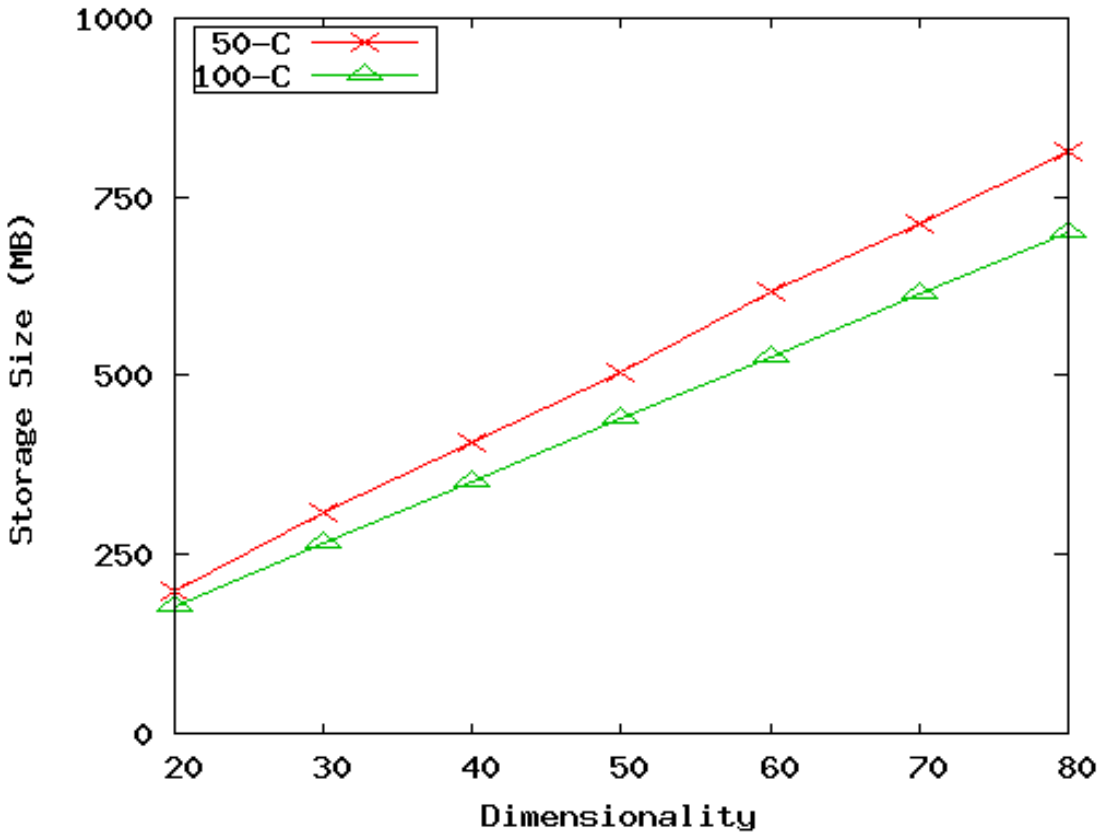
Online Query Computation: Method

- Given the fragment cubes, process a query as follows
  1. Divide the query into fragment, same as the shell
  2. Fetch the corresponding TID list for each fragment from the fragment cube
  3. Intersect the TID lists from each fragment to construct **instantiated base table**
  4. Compute the data cube using the base table with any cubing algorithm

Online Query Computation: Sketch



Experiment: Size vs. Dimensionality (50 and 100 cardinality)



- (50-C): 10^6 tuples, 0 skew, 50 cardinality, fragment size 3.
- (100-C): 10^6 tuples, 2 skew, 100 cardinality, fragment size 2.

Experiments on Real World Data

- UCI Forest CoverType data set
  - 54 dimensions, 581K tuples
  - Shell fragments of size 2 took 33 seconds and 325MB to compute
  - 3-D subquery with 1 instantiate D: 85ms~1.4 sec.
- Longitudinal Study of Vocational Rehab. Data
  - 24 dimensions, 8818 tuples
  - Shell fragments of size 3 took 0.9 seconds and 60MB to compute
  - 5-D query with 0 instantiated D: 227ms~2.6 sec.

Outline

- Data Cube Computation: Preliminary Concepts
- Computing full/iceberg cubes: four methodologies
  - 1.General heuristics: On the computation of multidimensional aggregates (Agarwal et al.'96)
  - 2.Bottom-Up: Multiway Array Aggregation for Full Cube Computation (Zhao et al SIGMOD'97)
  - 3. Top-down: BUC: Computing Iceberg Cubes from the Apex Cuboid Downward (Beyer et al SIGMOD'99)
  - 4. Top-down: H-Cubing: Exploring an H-Tree Structure (Han et al SIGMOD'01)
- Overview of Other methodologies
  - 5.Star-cubing: Computing Iceberg Cubes Using a Dynamic Star-tree Structure (Xin et al VLDB'03)
  - 6.Precomputing Shell Fragments for Fast High-Dimensional OLAP (Li, et al. VLDB'04)
- Overview of Data Cubes for Advanced Applications
  - **7. Sampling Cubes: OLAP on Sampling Data (Li et al SIGMOD 08)**
  - 8. Ranking Cubes: Efficient Computation of Ranking Queries (Xin et al VLDB06)
- Overview of Knowledge Discovery with Data Cubes
  - **9**. Prediction Cubes: Data Mining in Multi-Dimensional Cube Space (Chen et al VLDB'05)
  - 10**.** Complex Aggregation at Multiple Granularity: Multi-feature Cubes (Ross et al EDBT 1998)
  - 11**.** Discovery-Driven Data Cubes (Sarawagi et al EDBT 1998)
- Summary

Processing Advanced Queries by Exploring Data Cube Technology

- Sampling Cube 
  - X. Li, J. Han, Z. Yin, J.-G. Lee, Y. Sun, "Sampling Cube: A Framework for Statistical OLAP over Sampling Data", SIGMOD'08
- Ranking Cube
  - D. Xin, J. Han, H. Cheng, and X. Li. Answering top-k queries with multi-dimensional selections: The ranking cube approach. VLDB'06
- Other advanced cubes for processing data and queries
  - Stream cube, spatial cube, multimedia cube, text cube, RFID cube, etc.

Statistical Surveys and OLAP

- Statistical survey: A popular tool to collect information about a **population** based on a **sample**
  - Ex.: TV ratings, US Census, election polls
- A common tool in politics, health, market research, science, and many more
- An efficient way of collecting information (Data collection is expensive)
- Many **statistical tools** available, to determine validity
  - Confidence intervals
  - Hypothesis tests
- OLAP (multidimensional analysis) on survey data
  - highly desirable but can it be done well?

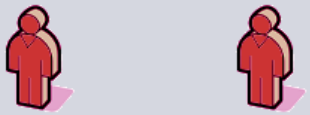
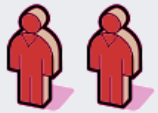
Surveys: Sample vs. Whole Population

Data is only a sample of **population**

Age\Education	High-school	College	Graduate
18			
19			
20			
…			

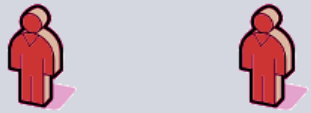
Problems for Drilling in Multidim. Space

Data is only a **sample** of population but samples could be small when drilling to certain multidimensional space

Age\Education	High-school	College	Graduate
18			
19			
20			
...			

OLAP on Survey (i.e., Sampling) Data

- Semantics of query is unchanged
- Input data has changed

Age/Education	High-school	College	Graduate
18			
19			
20			
...			

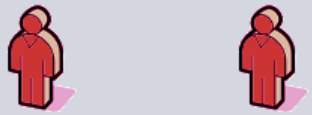
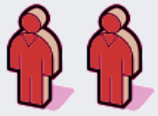
Challenges for OLAP on Sampling Data

- Computing confidence intervals in OLAP context
- No data?
  - Not exactly. No data in subspaces in cube
  - Sparse data
  - Causes include sampling bias and query selection bias
- Curse of dimensionality
  - Survey data can be high dimensional
  - Over 600 dimensions in real world example
  - Impossible to fully materialize

Example 1: Confidence Interval

What is the average income of 19-year-old high-school students?

Return not only query result but also confidence interval

Age/Education	High-school	College	Graduate
18			
19			
20			
...			

Confidence Interval

- *Confidence interval at* $\bar{x}$ $\bar{x} \pm t_c \hat{\sigma}_{\bar{x}}$
  - *x is a sample of data set;* $\bar{x}$ *is the mean of sample*
  - t_c *is the critical t-value, calculated by a look-up*
  - $\hat{\sigma}_{\bar{x}} = \frac{s}{\sqrt{l}}$ the estimated standard error of the mean
- *Example: $50,000 ± $3,000 with 95% confidence*
  - Treat points in cube cell as samples
  - Compute confidence interval as traditional sample set
- Return answer in the form of confidence interval
  - Indicates **quality** of query answer
  - User selects desired confidence interval

Efficient Computing Confidence Interval Measures

- Efficient computation in all cells in data cube
  - Both mean and confidence interval are **algebraic**
  - Why confidence interval measure is algebraic?

$$\bar{x} \pm t_c \hat{\sigma}_{\bar{x}}$$

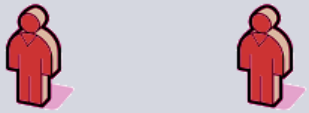
$\bar{x}$ is algebraic

$\hat{\sigma}_{\bar{x}} = \frac{s}{\sqrt{l}}$ where both *s* and *l* *(count)* are algebraic

- Thus one can calculate cells efficiently at more general cuboids without having to start at the base cuboid each time

Example 2: Query Expansion

What is the average income of 19-year-old college students?

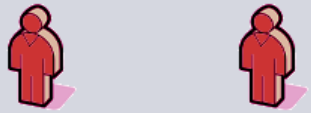
Age/Education	High-school	College	Graduate
18			
19			
20			
...			

Boosting Confidence by Query Expansion

- From the example: The queried cell “19-year-old college students” contains only 2 samples
- Confidence interval is large (i.e., low confidence). why?
  - Small sample size
  - High standard deviation with samples
- Small sample sizes can occur at relatively low dimensional selections
  - Collect more data?— expensive!
  - Use data in other cells? Maybe, but have to be careful

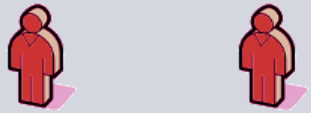
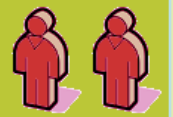
Intra-Cuboid Expansion: Choice 1

Expand query to include **18** and **20** year olds?

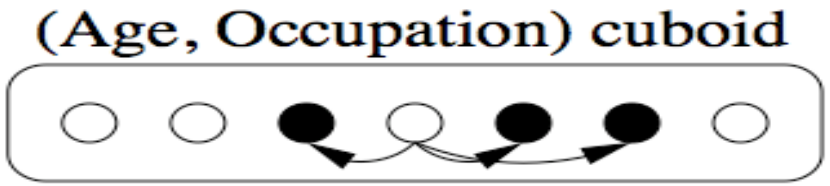
Age/Education	High-school	College	Graduate
18			
19			
20			
...			

Intra-Cuboid Expansion: Choice 2

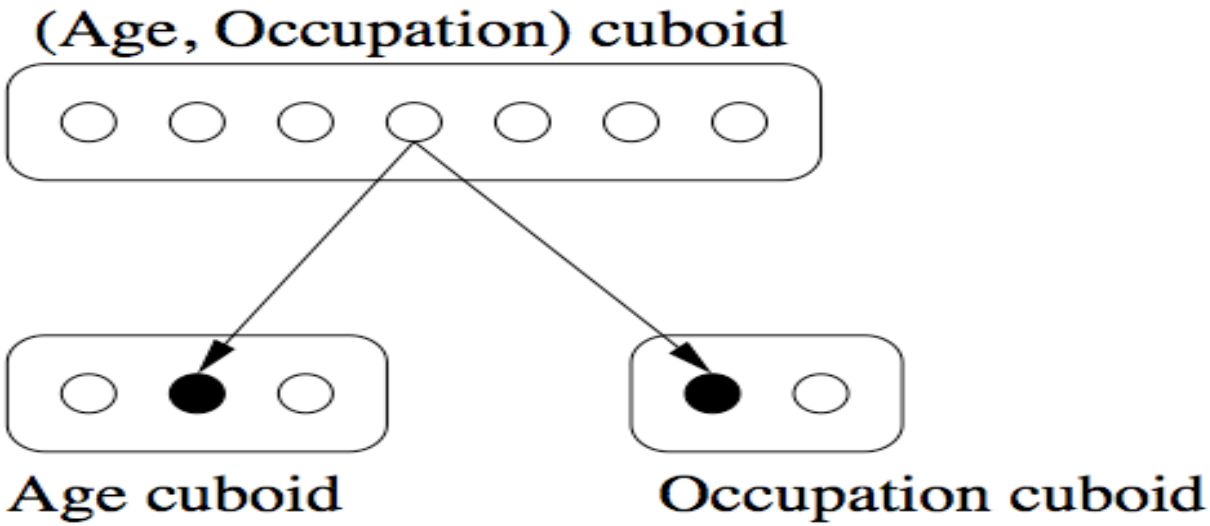
Expand query to include **high-school** and **graduate** students?

Age/Education	High-school	College	Graduate
18			
19			
20			
...			

Query Expansion



(a) Intra-Cuboid Expansion



(b) Inter-Cuboid Expansion

Intra-Cuboid Expansion

- Combine other cells' data into own to "boost" confidence
  - If share semantic and cube similarity
  - Use only if necessary
  - Bigger sample size will decrease confidence interval
- Cell segment similarity
  - Some dimensions are clear: **Age**
  - Some are fuzzy: **Occupation**
  - May need domain knowledge
- Cell value similarity
  - How to determine if two cells' samples come from the same population?
  - Two-sample t-test (confidence-based)

Inter-Cuboid Expansion

- If a query dimension is
  - Not correlated with cube value
  - But is causing small sample size by drilling down too much
- Remove dimension (i.e., generalize to *) and move to a more general cuboid
- Can use two-sample t-test to determine similarity between two cells across cuboids
- Can also use a different method to be shown later

Query Expansion Experiments

- Real world sample data: 600 dimensions and 750,000 tuples
- 0.05% to simulate "sample" (allows error checking)

(a) Intra-Cuboid Expansion with **Age** dimension and **Average Number of Children** cube measure

Query		Average Query Answer Error			Sampling Sizes		
Gender	Marital	No Expand	Expand	% Improve	Population	Sample	Expanded
FEMALE	MARRIED	0.48	0.32	33%	2473.0	2.2	28.3
FEMALE	SINGLE	0.31	0.21	30%	612.6	0.6	6.4
FEMALE	DIVORCED	0.49	0.43	11%	321.1	0.3	3.4
MALE	MARRIED	0.42	0.21	49%	4296.8	4.4	37.6
MALE	SINGLE	0.26	0.21	16%	571.8	0.5	3.6
MALE	DIVORCED	0.33	0.27	19%	224.7	0.2	1.2
	Average	0.38	0.27	26%	1416.7	1.4	13.4

Outline

- Data Cube Computation: Preliminary Concepts
- Computing full/iceberg cubes: four methodologies
  - 1.General heuristics: On the computation of multidimensional aggregates (Agarwal et al.'96)
  - 2.Bottom-Up: Multiway Array Aggregation for Full Cube Computation (Zhao et al SIGMOD'97)
  - 3. Top-down: BUC: Computing Iceberg Cubes from the Apex Cuboid Downward (Beyer et al SIGMOD'99)
  - 4. Top-down: H-Cubing: Exploring an H-Tree Structure (Han et al SIGMOD'01)
- Overview of Other methodologies
  - 5.Star-cubing: Computing Iceberg Cubes Using a Dynamic Star-tree Structure (Xin et al VLDB'03)
  - 6.Precomputing Shell Fragments for Fast High-Dimensional OLAP (Li, et al. VLDB'04)
- Overview of Data Cubes for Advanced Applications
  - **7.** Sampling Cubes: OLAP on Sampling Data (Li et al SIGMOD 08)
  - **8. Ranking Cubes: Efficient Computation of Ranking Queries (Xin et al VLDB06)**
- Overview of Knowledge Discovery with Data Cubes
  - **9**. Prediction Cubes: Data Mining in Multi-Dimensional Cube Space (Chen et al VLDB'05)
  - 10**.** Complex Aggregation at Multiple Granularity: Multi-feature Cubes (Ross et al EDBT 1998)
  - 11**.** Discovery-Driven Data Cubes (Sarawagi et al EDBT 1998)
- Summary

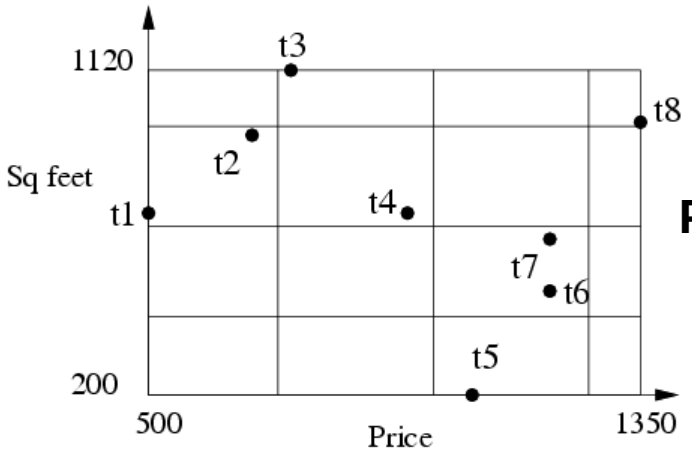
Ranking Cubes – Efficient Computation of Ranking queries

- Data cube helps not only OLAP but also ranked search
- **(top-k) ranking query**: only returns the best k results according to a user-specified preference, consisting of (1) a *selection condition* and (2) a *ranking function*
- Ex.: Search for apartments with expected price 1000 and expected square feet 800
  - Select top 1 from Apartment
  - where City = "LA" and Num_Bedroom = 2
  - order by [price – 1000]^2 + [sq feet - 800]^2 asc
- Efficiency question: Can we only search what we need?
  - Build a ranking cube on both selection dimensions and ranking dimensions

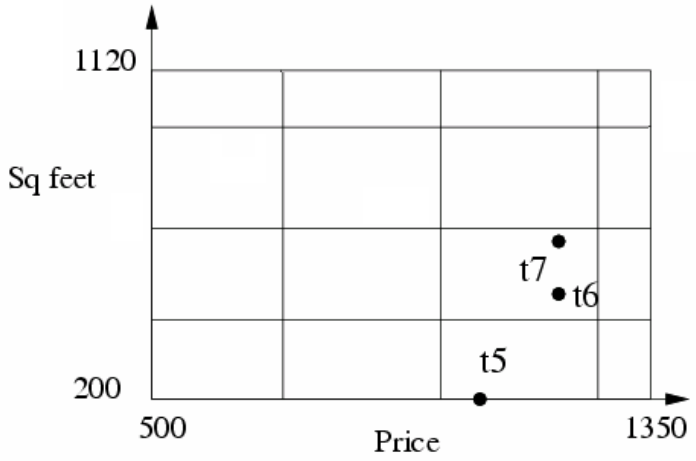
Ranking Cube: Partition Data on Both Selection and Ranking Dimensions

One single data partition as the template

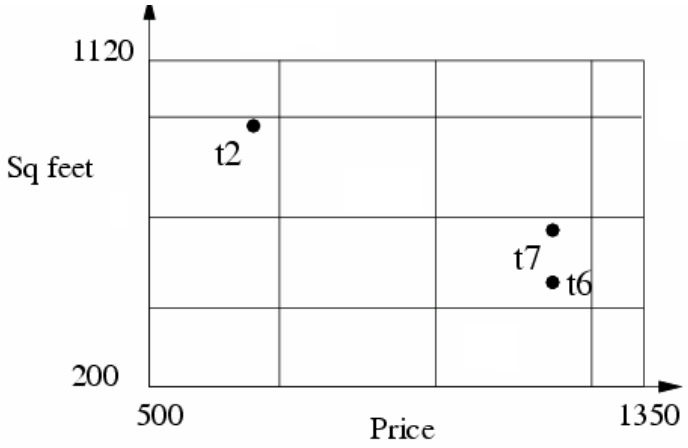
Slice the data partition by selection conditions



Partition for all data



Sliced Partition for city="LA"

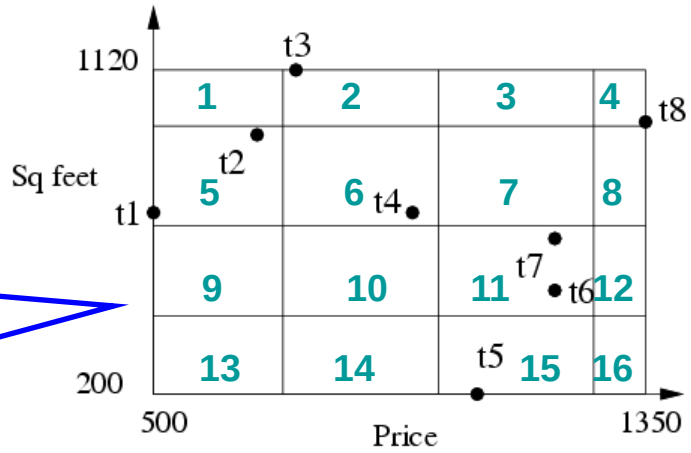


Sliced Partition for BR=2

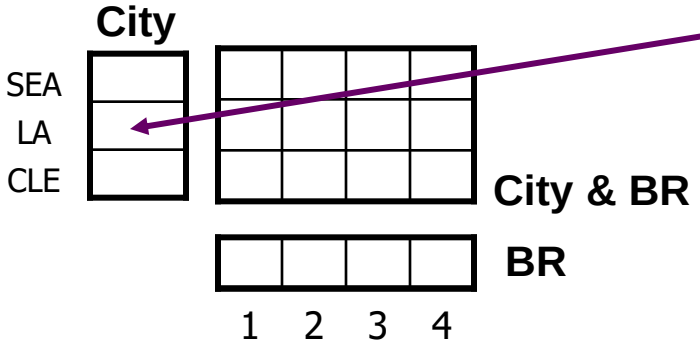
Materialize Ranking-Cube

tid	City	BR	Price	Sq feet
t1	SEA	1	500	600
t2	CLE	2	700	800
t3	SEA	1	800	900
t4	CLE	3	1000	1000
t5	LA	1	1100	200
t6	LA	2	1200	500
t7	LA	2	1200	560
t8	CLE	3	1350	1120

Step 1: Partition Data on Ranking Dimensions



Step 2: Group data by Selection Dimensions

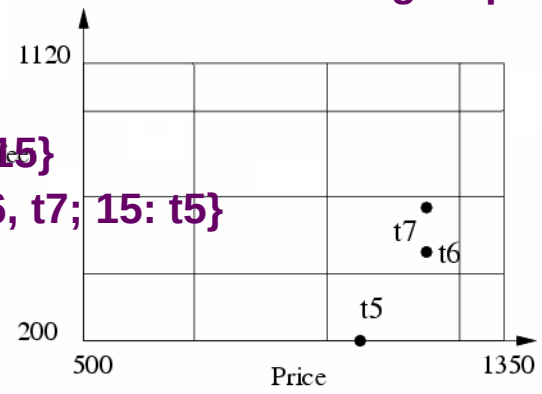


Step 3: Compute Measures for each group

For the cell (LA)

Block-level: {11, 15}

Data-level: {11: t6, t7; 15: t5}

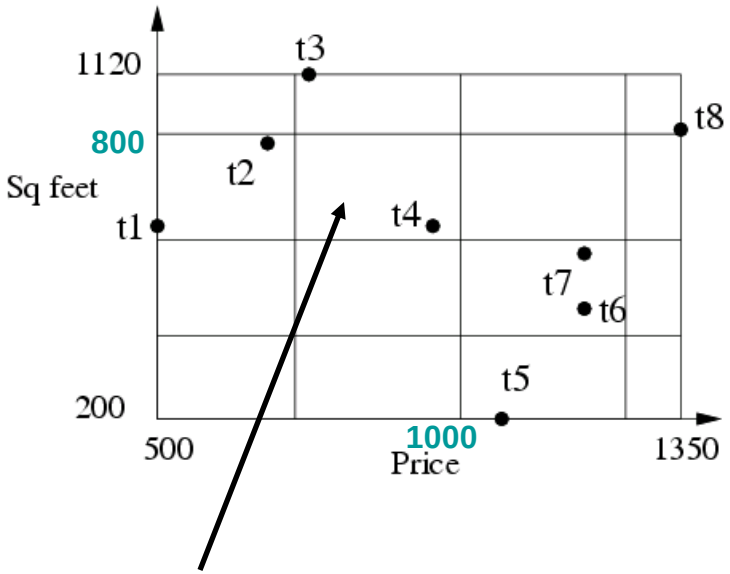


Search with Ranking-Cube: Simultaneously Push Selection and Ranking

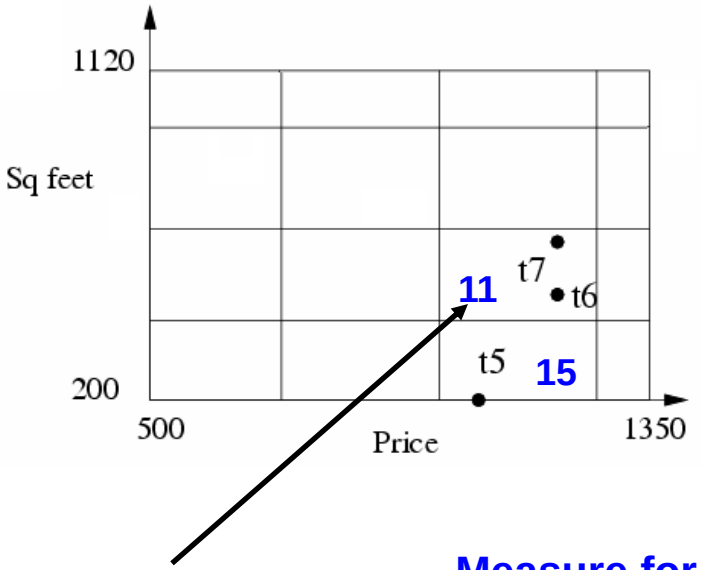
Select top 1 from Apartment
where city = "LA"
order by [price – 1000]^2 + [sq feet - 800]^2 asc

Bin boundary for price	[500, 600, 800, 1100,1350]
Bin boundary for sq feet	[200, 400, 600, 800, 1120]

Given the bin boundaries, locate the block with top score



Without ranking-cube: start search from here



With ranking-cube: start search from here

Measure for LA:
{11, 15}
{11: t6,t7; 15:t5}

Ranking Cube: Methodology and Extension

- Ranking cube methodology
  - Push selection and ranking simultaneously
  - It works for many sophisticated ranking functions
- How to support high-dimensional data?
  - Materialize only those *atomic* cuboids that contain single selection dimensions
    - Uses the idea similar to high-dimensional OLAP
    - Achieves low space overhead and high performance in answering ranking queries with a high number of selection dimensions

Outline

- Data Cube Computation: Preliminary Concepts
- Computing full/iceberg cubes: four methodologies
  - 1.General heuristics: On the computation of multidimensional aggregates (Agarwal et al.'96)
  - 2.Bottom-Up: Multiway Array Aggregation for Full Cube Computation (Zhao et al SIGMOD'97)
  - 3. Top-down: BUC: Computing Iceberg Cubes from the Apex Cuboid Downward (Beyer et al SIGMOD'99)
  - 4. Top-down: H-Cubing: Exploring an H-Tree Structure (Han et al SIGMOD'01)
- Overview of Other methodologies
  - 5.Star-cubing: Computing Iceberg Cubes Using a Dynamic Star-tree Structure (Xin et al VLDB'03)
  - 6.Precomputing Shell Fragments for Fast High-Dimensional OLAP (Li, et al. VLDB'04)
- Overview of Data Cubes for Advanced Applications
  - **7.** Sampling Cubes: OLAP on Sampling Data (Li et al SIGMOD 08)
  - **8**. Ranking Cubes: Efficient Computation of Ranking Queries (Xin et al VLDB06)
- Overview of Knowledge Discovery with Data Cubes
  - **9**. **Prediction Cubes: Data Mining in Multi-Dimensional Cube Space (Chen et al VLDB'05)**
  - 10**.** Complex Aggregation at Multiple Granularity: Multi-feature Cubes (Ross et al EDBT 1998)
  - 11**.** Discovery-Driven Data Cubes (Sarawagi et al EDBT 1998)
- Summary

Data Mining in Cube Space

- Data cube greatly increases the analysis bandwidth
- Four ways to interact OLAP-styled analysis and data mining
  - Using cube space to define data space for mining
  - Using OLAP queries to generate features and targets for mining, e.g., multi-feature cube
  - Using data-mining models as building blocks in a multi-step mining process
  - Using data-cube computation techniques to speed up repeated model construction
    - Cube-space data mining may require building a model for each candidate data space
    - Sharing computation across model-construction for different candidates may lead to efficient mining

Prediction Cubes

- **Prediction cube**: A cube structure that stores prediction models in multidimensional data space and supports prediction in OLAP manner

- Prediction models are used as building blocks to define the interestingness of subsets of data, i.e., to answer which subsets of data indicate better prediction

Reference: B.-C. Chen, L. Chen, Y. Lin, and R. Ramakrishnan. Prediction cubes. VLDB'05

How to Determine the Prediction Power of an Attribute?

- Ex. A customer table **D:**
  - Two dimensions **Z**: *Time* (*Month*, *Year*) and *Location* (*State*, *Country*)
  - Two features **X**: *Gender* and *Salary*
  - One class-label attribute Y: *Valued Customer*
- Q: "Are there times and locations in which the value of a customer depended greatly on the customers gender (i.e., Gender: predictiveness attribute **V**)?"
- Idea:
  - Compute the difference between the model built on that using **X** to predict Y and that built on using **X – V** to predict Y
    - X is a set of attributes and V is an attribute
  - If the difference is large, **V** must play an important role at predicting Y

Efficient Computation of Prediction Cubes

- Naïve method: Fully materialize the prediction cube, i.e., exhaustively build models and evaluate them for each cell and for each granularity

- Better approach: Explore score function decomposition that reduces prediction cube computation to data cube computation

Outline

- Data Cube Computation: Preliminary Concepts
- Computing full/iceberg cubes: four methodologies
  - 1.General heuristics: On the computation of multidimensional aggregates (Agarwal et al.'96)
  - 2.Bottom-Up: Multiway Array Aggregation for Full Cube Computation (Zhao et al SIGMOD'97)
  - 3. Top-down: BUC: Computing Iceberg Cubes from the Apex Cuboid Downward (Beyer et al SIGMOD'99)
  - 4. Top-down: H-Cubing: Exploring an H-Tree Structure (Han et al SIGMOD'01)
- Overview of Other methodologies
  - 5.Star-cubing: Computing Iceberg Cubes Using a Dynamic Star-tree Structure (Xin et al VLDB'03)
  - 6.Precomputing Shell Fragments for Fast High-Dimensional OLAP (Li, et al. VLDB'04)
- Overview of Data Cubes for Advanced Applications
  - **7.** Sampling Cubes: OLAP on Sampling Data (Li et al SIGMOD 08)
  - **8**. Ranking Cubes: Efficient Computation of Ranking Queries (Xin et al VLDB06)
- Overview of Knowledge Discovery with Data Cubes
  - 9. Prediction Cubes: Data Mining in Multi-Dimensional Cube Space (Chen et al VLDB'05)
  - 10**. Complex Aggregation at Multiple Granularity: Multi-feature Cubes (Ross et al EDBT 1998)**
  - 11**.** Discovery-Driven Data Cubes (Sarawagi et al EDBT 1998)
- Summary

Complex Aggregation at Multiple Granularities: Multi-Feature Cubes

- Multi-feature cubes (Ross, et al. 1998): Compute complex queries involving multiple dependent aggregates at multiple granularities
- Ex. Grouping by all subsets of {item, region, month}, find the maximum price in 2010 for each group, and the total sales among all maximum price tuples

```
select item, region, month, max(price),
    sum(R.sales)
from purchases
where year = 2010
cube by item, region, month: R
such that R.price = max(price)
```

- Continuing the last example, among the max price tuples, find the min and max shelf live, and find the fraction of the total sales due to tuple that have min shelf life within the set of all max price tuples

Reference Paper: Ross, K.A., Srivastava, D., Chatziantoniou, D. (1998). Complex aggregation at multiple granularities. EDBT 1998. Lecture Notes in Computer Science, vol 1377. Springer, Berlin, Heidelberg.

Outline

- Data Cube Computation: Preliminary Concepts
- Computing full/iceberg cubes: four methodologies
  - 1.General heuristics: On the computation of multidimensional aggregates (Agarwal et al.'96)
  - 2.Bottom-Up: Multiway Array Aggregation for Full Cube Computation (Zhao et al SIGMOD'97)
  - 3. Top-down: BUC: Computing Iceberg Cubes from the Apex Cuboid Downward (Beyer et al SIGMOD'99)
  - 4. Top-down: H-Cubing: Exploring an H-Tree Structure (Han et al SIGMOD'01)
- Overview of Other methodologies
  - 5.Star-cubing: Computing Iceberg Cubes Using a Dynamic Star-tree Structure (Xin et al VLDB'03)
  - 6.Precomputing Shell Fragments for Fast High-Dimensional OLAP (Li, et al. VLDB'04)
- Overview of Data Cubes for Advanced Applications
  - **7.** Sampling Cubes: OLAP on Sampling Data (Li et al SIGMOD 08)
  - **8**. Ranking Cubes: Efficient Computation of Ranking Queries (Xin et al VLDB06)
- Overview of Knowledge Discovery with Data Cubes
  - 9. Prediction Cubes: Data Mining in Multi-Dimensional Cube Space (Chen et al VLDB'05)
  - 10**.** Complex Aggregation at Multiple Granularity: Multi-feature Cubes (Ross et al EDBT 1998)
  - 11**. Discovery-Driven Data Cubes (Sarawagi et al EDBT 1998)**
- Summary

Discovery-Driven Exploration of Data Cubes

- Hypothesis-driven
  - exploration by user, huge search space
- Discovery-driven
  - Effective navigation of large OLAP data cubes
  - pre-compute measures indicating exceptions, guide user in the data analysis, at all levels of aggregation
  - Exception: significantly different from the value anticipated, based on a statistical model
  - Visual cues such as background color are used to reflect the degree of exception of each cell

Sarawagi, S., Agrawal, R., Megiddo, N. (1998). Discovery-driven exploration of OLAP data cubes. EDBT 1998. Lecture Notes in Computer Science, vol 1377.

Kinds of Exceptions and their Computation

- Parameters
  - SelfExp: surprise of cell relative to other cells at same level of aggregation
  - InExp: surprise beneath the cell
  - PathExp: surprise beneath cell for each drill-down path
- Computation of exception indicator (modeling fitting and computing SelfExp, InExp, and PathExp values) can be overlapped with cube construction
- Exception themselves can be stored, indexed and retrieved like precomputed aggregates

Examples: Discovery-Driven Data Cubes

item	all
region	all

Sum of sales	month											
	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
Total		1%	-1%	0%	1%	3%	-1	-9%	-1%	2%	-4%	3%

Avg sales	month											
item	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
Sony b/w printer		9%	-8%	2%	-5%	14%	-4%	0%	41%	-13%	-15%	-11%
Sony color printer		0%	0%	3%	2%	4%	-10%	-13%	0%	4%	-6%	4%
HP b/w printer		-2%	1%	2%	3%	8%	0%	-12%	-9%	3%	-3%	6%
HP color printer		0%	0%	-2%	1%	0%	-1%	-7%	-2%	1%	-5%	1%
IBM home computer		1%	-2%	-1%	-1%	3%	3%	-10%	4%	1%	-4%	-1%
IBM laptop computer		0%	0%	-1%	3%	4%	2%	-10%	-2%	0%	-9%	3%
Toshiba home computer		-2%	-5%	1%	1%	-1%	1%	5%	-3%	-5%	-1%	-1%
Toshiba laptop computer		1%	0%	3%	0%	-2%	-2%	-5%	3%	2%	-1%	0%
Logitech mouse		3%	-2%	-1%	0%	4%	6%	-11%	2%	1%	-4%	0%
Ergo-way mouse		0%	0%	2%	3%	1%	-2%	-2%	-5%	0%	-5%	8%

item	IBM home computer

Avg sales	month											
region	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
North		-1%	-3%	-1%	0%	3%	4%	-7%	1%	0%	-3%	-3%
South		-1%	1%	-9%	6%	-1%	-39%	9%	-34%	4%	1%	7%
East		-1%	-2%	2%	-3%	1%	18%	-2%	11%	-3%	-2%	-1%
West		4%	0%	-1%	-3%	5%	1%	-18%	8%	5%	-8%	1%

Summary

- Data Cube Computation: Preliminary Concepts
- Computing full/iceberg cubes: four methodologies
  - **1.**General heuristics: On the computation of multidimensional aggregates (Agarwal et al.'96)
  - 2.Bottom-Up: Multiway Array Aggregation for Full Cube Computation (Zhao et al SIGMOD'97)
  - **3**. Top-down: BUC: Computing Iceberg Cubes from the Apex Cuboid Downward (Beyer et al SIGMOD'99)
  - 4**.** Top-down: H-Cubing: Exploring an H-Tree Structure (Han et al SIGMOD'01)
- Overview of Other methodologies
    - 5.Star-cubing: Computing Iceberg Cubes Using a Dynamic Star-tree Structure (Xin et al VLDB'03)
    - 6.Precomputing Shell Fragments for Fast High-Dimensional OLAP (Li, et al. VLDB'04)
- Overview of Data Cubes for Advanced Applications
  - 7. Sampling Cubes: OLAP on Sampling Data (Li et al SIGMOD 08)
  - 8. Ranking Cubes: Efficient Computation of Ranking Queries (Xin et al VLDB06)
- Overview of Knowledge Discovery with Data Cubes
  - **9**. Prediction Cubes: Data Mining in Multi-Dimensional Cube Space (Chen et al VLDB'05)
  - 10**.** Complex Aggregation at Multiple Granularity: Multi-feature Cubes (Ross et al EDBT 1998)
  - 11**.** Discovery-Driven Data Cubes (Sarawagi et al EDBT 1998)
- Summary

Other cubes

- Stream cubes for multi dimensional stream analysis
  - Y. Chen, G. Dong, J. Han, B. W. Wah, and J. Wang, Multi-Dimensional Regression Analysis of Time-Series Data Streams, VLDB'02
- Map cube for visualizing spatial data
  - Map Cube: A Visualization Tool for Spatial Data Warehouses
- Multimedia cube
  - **MultiMediaMiner: A System Prototype for MultiMedia Data Mining**
- Text Cube for analyzing multi dimensional text databases
  - C. X. Lin, B. Ding, J. Han, F. Zhu, and B. Zhao. Text Cube: Computing IR measures for multidimensional text database analysis. ICDM'08
- Topic cube based on topic modelling
  - D. Zhang, C. Zhai, and J. Han. Topic cube: Topic modeling for OLAP on multi-dimensional text databases. SDM'09
- Flow cube for analysing RFID data
  - **FlowCube: Constructing RFID FlowCubes for Multi-Dimensional Analysis of Commodity Flows, VLDB** *06*

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 cube provides opportunity to extract interesting summarization knowledge from multi-dimensional data
- Potential opportunity
  - for research for new technologies

Final Word

Ref.(I) Data Cube Computation Methods

- S. Agarwal, R. Agrawal, P. M. Deshpande, A. Gupta, J. F. Naughton, R. Ramakrishnan, and S. Sarawagi. On the computation of multidimensional aggregates. VLDB'96
- D. Agrawal, A. E. Abbadi, A. Singh, and T. Yurek. Efficient view maintenance in data warehouses. SIGMOD'97
- K. Beyer and R. Ramakrishnan. Bottom-Up Computation of Sparse and Iceberg CUBEs.. SIGMOD'99
- M. Fang, N. Shivakumar, H. Garcia-Molina, R. Motwani, and J. D. Ullman. Computing iceberg queries efficiently. VLDB'98
- J. Gray, S. Chaudhuri, A. Bosworth, A. Layman, D. Reichart, M. Venkatrao, F. Pellow, and H. Pirahesh. Data cube: A relational aggregation operator generalizing group-by, cross-tab and sub-totals. Data Mining and Knowledge Discovery, 1:29–54, 1997.
- J. Han, J. Pei, G. Dong, K. Wang. Efficient Computation of Iceberg Cubes With Complex Measures. SIGMOD'01
- L. V. S. Lakshmanan, J. Pei, and J. Han, Quotient Cube: How to Summarize the Semantics of a Data Cube, VLDB'02
- X. Li, J. Han, and H. Gonzalez, High-Dimensional OLAP: A Minimal Cubing Approach, VLDB'04
- Y. Zhao, P. M. Deshpande, and J. F. Naughton. An array-based algorithm for simultaneous multidimensional aggregates. SIGMOD'97
- K. Ross and D. Srivastava. Fast computation of sparse datacubes. VLDB'97
- D. Xin, J. Han, X. Li, B. W. Wah, Star-Cubing: Computing Iceberg Cubes by Top-Down and Bottom-Up Integration, VLDB'03
- D. Xin, J. Han, Z. Shao, H. Liu, C-Cubing: Efficient Computation of Closed Cubes by Aggregation-Based Checking, ICDE'06

Ref. (II) Advanced Applications with Data Cubes

- D. Burdick, P. Deshpande, T. S. Jayram, R. Ramakrishnan, and S. Vaithyanathan. OLAP over uncertain and imprecise data. VLDB'05
- X. Li, J. Han, Z. Yin, J.-G. Lee, Y. Sun, "Sampling Cube: A Framework for Statistical OLAP over Sampling Data", SIGMOD'08
- C. X. Lin, B. Ding, J. Han, F. Zhu, and B. Zhao. Text Cube: Computing IR measures for multidimensional text database analysis. ICDM'08
- D. Papadias, P. Kalnis, J. Zhang, and Y. Tao. Efficient OLAP operations in spatial data warehouses. SSTD'01
- N. Stefanovic, J. Han, and K. Koperski. Object-based selective materialization for efficient implementation of spatial data cubes. IEEE Trans. Knowledge and Data Engineering, 12:938–958, 2000.
- T. Wu, D. Xin, Q. Mei, and J. Han. Promotion analysis in multidimensional space. VLDB'09
- T. Wu, D. Xin, and J. Han. ARCube: Supporting ranking aggregate queries in partially materialized data cubes. SIGMOD'08
- D. Xin, J. Han, H. Cheng, and X. Li. Answering top-k queries with multi-dimensional selections: The ranking cube approach. VLDB'06
- J. S. Vitter, M. Wang, and B. R. Iyer. Data cube approximation and histograms via wavelets. CIKM'98
- D. Zhang, C. Zhai, and J. Han. Topic cube: Topic modeling for OLAP on multi-dimensional text databases. SDM'09

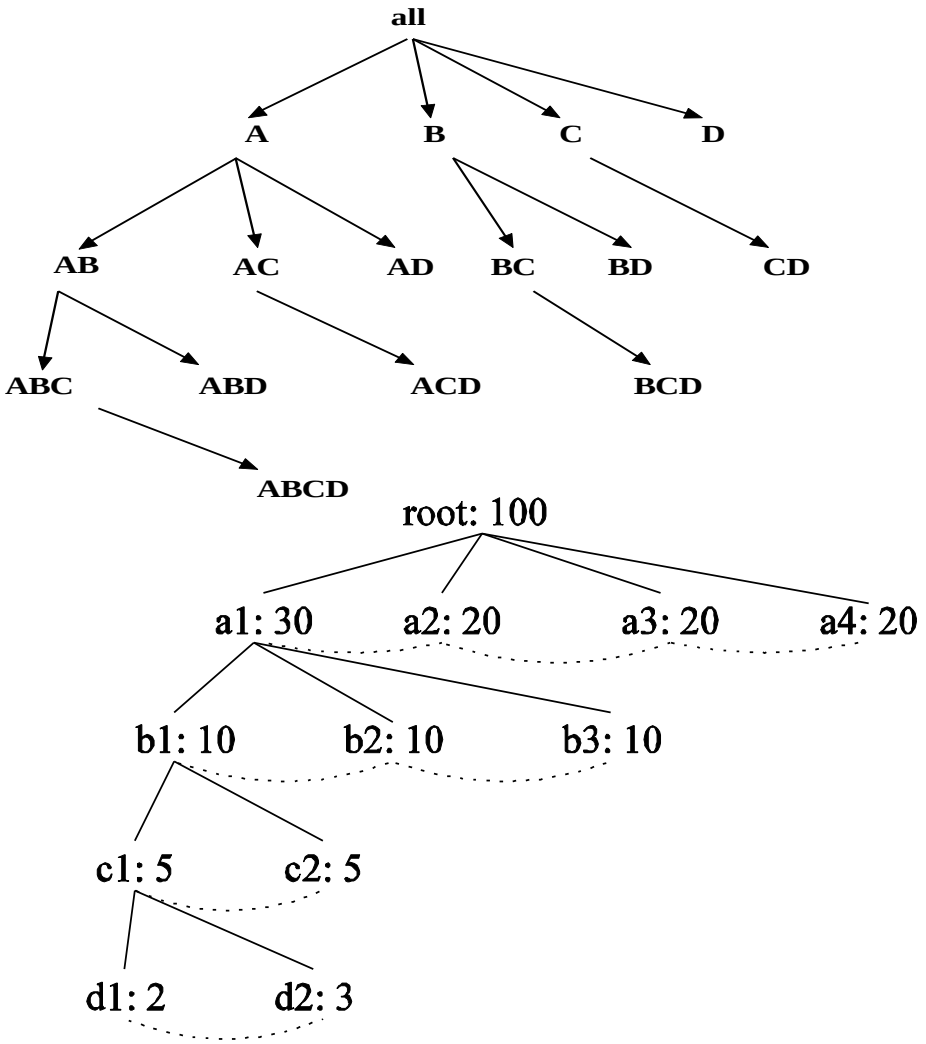
Ref. (III) Knowledge Discovery with Data Cubes

- R. Agrawal, A. Gupta, and S. Sarawagi. Modeling multidimensional databases. ICDE'97
- B.-C. Chen, L. Chen, Y. Lin, and R. Ramakrishnan. Prediction cubes. VLDB'05
- B.-C. Chen, R. Ramakrishnan, J.W. Shavlik, and P. Tamma. Bellwether analysis: Predicting global aggregates from local regions. VLDB'06
- Y. Chen, G. Dong, J. Han, B. W. Wah, and J. Wang, Multi-Dimensional Regression Analysis of Time-Series Data Streams, VLDB'02
- G. Dong, J. Han, J. Lam, J. Pei, K. Wang. Mining Multi-dimensional Constrained Gradients in Data Cubes. VLDB' 01
- R. Fagin, R. V. Guha, R. Kumar, J. Novak, D. Sivakumar, and A. Tomkins. Multi-structural databases. PODS'05
- J. Han. Towards on-line analytical mining in large databases. SIGMOD Record, 27:97–107, 1998
- T. Imielinski, L. Khachiyan, and A. Abdulghani. Cubegrades: Generalizing association rules. Data Mining & Knowledge Discovery, 6:219–258, 2002.
- R. Ramakrishnan and B.-C. Chen. Exploratory mining in cube space. Data Mining and Knowledge Discovery, 15:29–54, 2007.
- K. A. Ross, D. Srivastava, and D. Chatziantoniou. Complex aggregation at multiple granularities. EDBT'98
- S. Sarawagi, R. Agrawal, and N. Megiddo. Discovery-driven exploration of OLAP data cubes. EDBT'98
- G. Sathe and S. Sarawagi. Intelligent Rollups in Multidimensional OLAP Data. *VLDB'01*

Surplus Slides

H-Cubing: Using H-Tree Structure

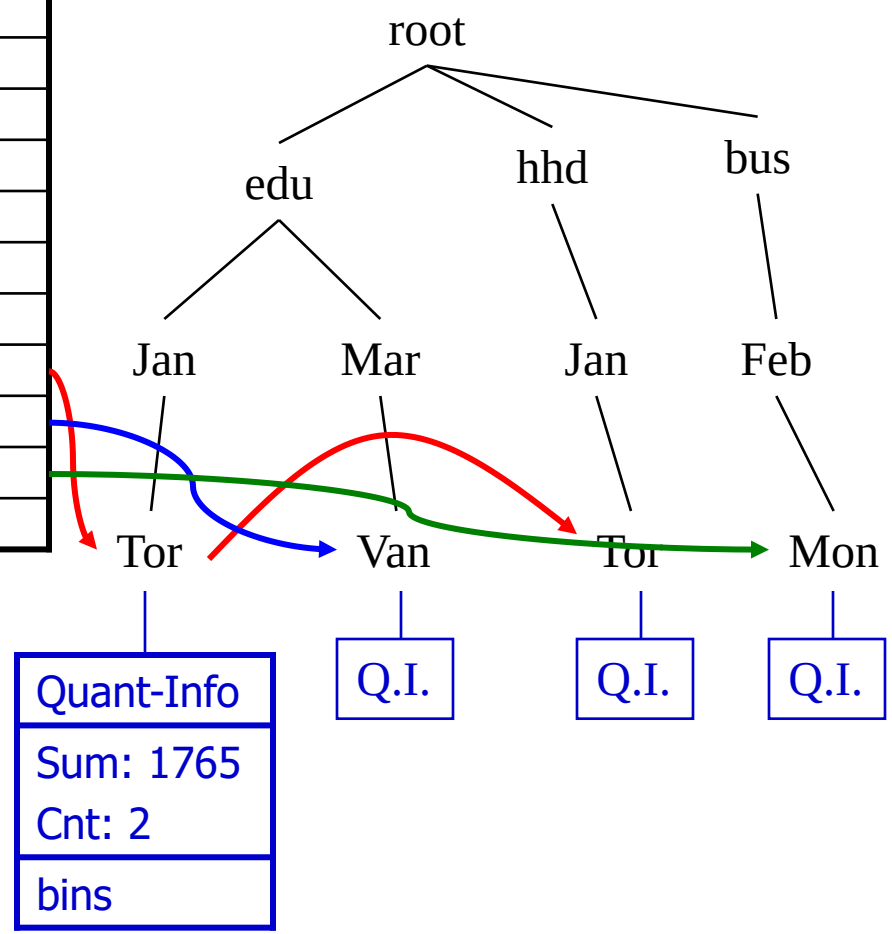
- Bottom-up computation
- Exploring an H-tree structure
- If the current computation of an H-tree cannot pass min_sup, do not proceed further (pruning)
- No simultaneous aggregation



H-tree: A Prefix Hyper-tree

Header table

Attr. Val.	Quant-Info	Side-link
Edu	Sum:2285 …	
Hhd	…	
Bus	…	
…	…	
Jan	…	
Feb	…	
…	…	
Tor	**…**	
Van	**…**	
Mon	**…**	
…	…	



Month	City	Cust_grp	Prod	Cost	Price
Jan	Tor	Edu	Printer	500	485
Jan	Tor	Hhd	TV	800	1200
Jan	Tor	Edu	Camera	1160	1280
Feb	Mon	Bus	Laptop	1500	2500
Mar	Van	Edu	HD	540	520
…	…	…	…	…	…

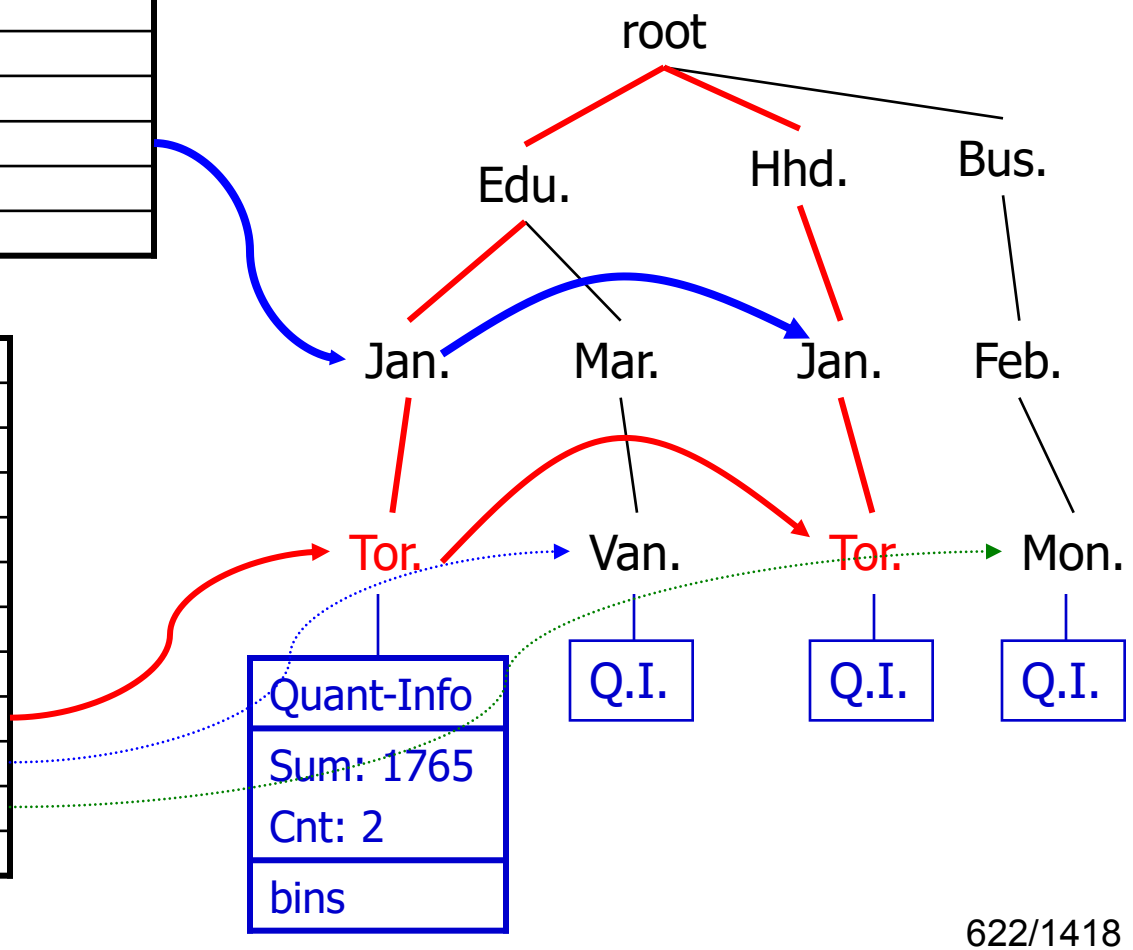
Computing Cells Involving "City"

Header Table H_{Tor}

Attr. Val.	Q.I.	Side-link
Edu	…	
Hhd	…	
Bus	…	
…	…	
Jan	**…**	
Feb	…	
…	…	

From (*, *, Tor) to (*, Jan, Tor)

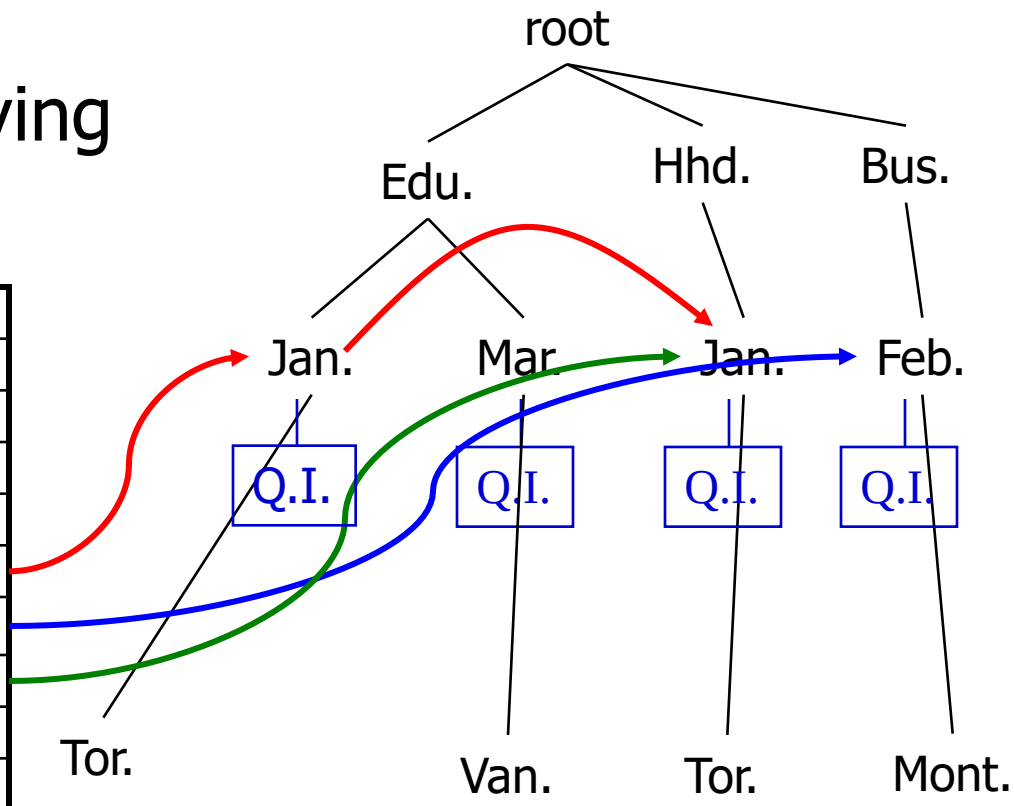
Attr. Val.	Quant-Info	Side-link
Edu	Sum:2285 …	
Hhd	…	
Bus	…	
…	…	
Jan	…	
Feb	…	
…	…	
Tor	**…**	
Van	…	
Mon	…	
…	…	



Computing Cells Involving Month But No City

1. Roll up quant-info
2. Compute cells involving month but no city

Attr. Val.	Quant-Info	Side-link
Edu.	Sum:2285 …	
Hhd.	…	
Bus.	…	
…	…	
Jan.	**…**	
Feb.	**…**	
Mar.	**…**	
…	…	
Tor.	…	
Van.	…	
Mont.	…	
…	…	

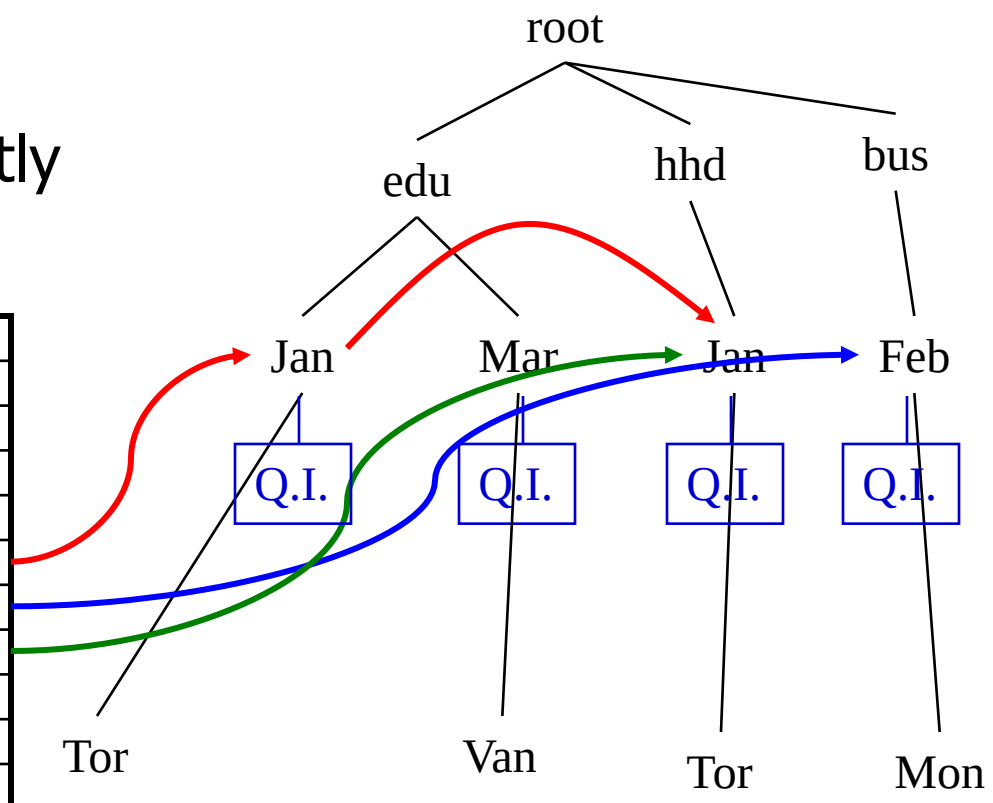


Top-k OK mark: if Q.I. in a child passes top-k avg threshold, so does its parents. No binning is needed!

Computing Cells Involving Only Cust_grp

Check header table directly

Attr. Val.	Quant-Info	Side-link
Edu	**Sum:2285 …**	
Hhd	**…**	
Bus	**…**	
…	…	
Jan	…	
Feb	…	
Mar	…	
…	…	
Tor	…	
Van	…	
Mon	…	
…	…	



Concepts and algorithms for mining patterns and associations (Class #9)

P.Krishna Reddy

IIIT, Hyderabad

Detailed Syllabus

- Introduction (1.5 (3) hour): Definition, KDD framework, Issues in data mining.
- Preprocessing and Data summarization (7.5 (7.5) hrs): Data Types, Preprocessing, Characterization, Discrimination, data warehousing techniques (Data warehousing technology, Data cube computation**)**
- **Concepts and algorithms for mining patterns and associations (9 hours) (Frequent item-set generation, A priori and FP-growth algorithm, Evaluation of Association patterns)**
- Concepts and algorithms related to classification and regression (9hrs) (Overview, Decision tree induction, Over-fitting and under-fitting, Scalable decision tree algorithms, Bayesian Classification, Regression-based Prediction methods (9 hours)
- Concepts and algorithms for clustering the data (9 hours) (Overview, Types of Data, K-means, Aglomerative clustering, Clustering algorithms (DBSCAN, BIRCH, CURE, ROCK, CHAMELEON)).
- Outlier analysis and future trends (graph mining, spatio-temporal mining). (3 hours)

Presentation Outline

- Basic concepts
  - Frequent itemssets, Closed Itemsets, and Association Rules
- Frequent itemset mining methods
  - Apriori Algorithm
    - Generating Association Rules from Frequent Itemsets
    - Improving the Efficiency of Apriori
  - A Pattern-Growth Approach
  - Mining Frequent Itemsets Using the Vertical Data Format
  - Mining Closed and Max Patterns
- Which Patterns are interesting? Pattern Evaluation Methods
- Summary

What Is Frequent Pattern Analysis?

- Use case: You are talking to a customer who recently bought a PC and a digital camera from the store. What should you recommend to her next?
  - Information about which products are frequently purchased by your customers following their purchases of a PC and a digital camera in sequence would be very helpful
  - Frequent patterns and association rules are the knowledge that you want to mine in such a scenario.

- Frequent pattern: a pattern (a set of items, subsequences, substructures, etc.) that occurs frequently in a data set
  - First proposed by Agrawal, Imielinski, and Swami [AIS93] in the context of frequent itemsets and association rule mining

- Motivation: Finding inherent regularities in data
  - What products were often purchased together?— Beer and diapers?!
  - What are the subsequent purchases after buying a PC?
  - What kinds of DNA are sensitive to this new drug?
  - Can we automatically classify web documents?

- Applications
  - Basket data analysis, cross-marketing, catalog design, sale campaign analysis, Web log (click stream) analysis, and DNA sequence analysis.

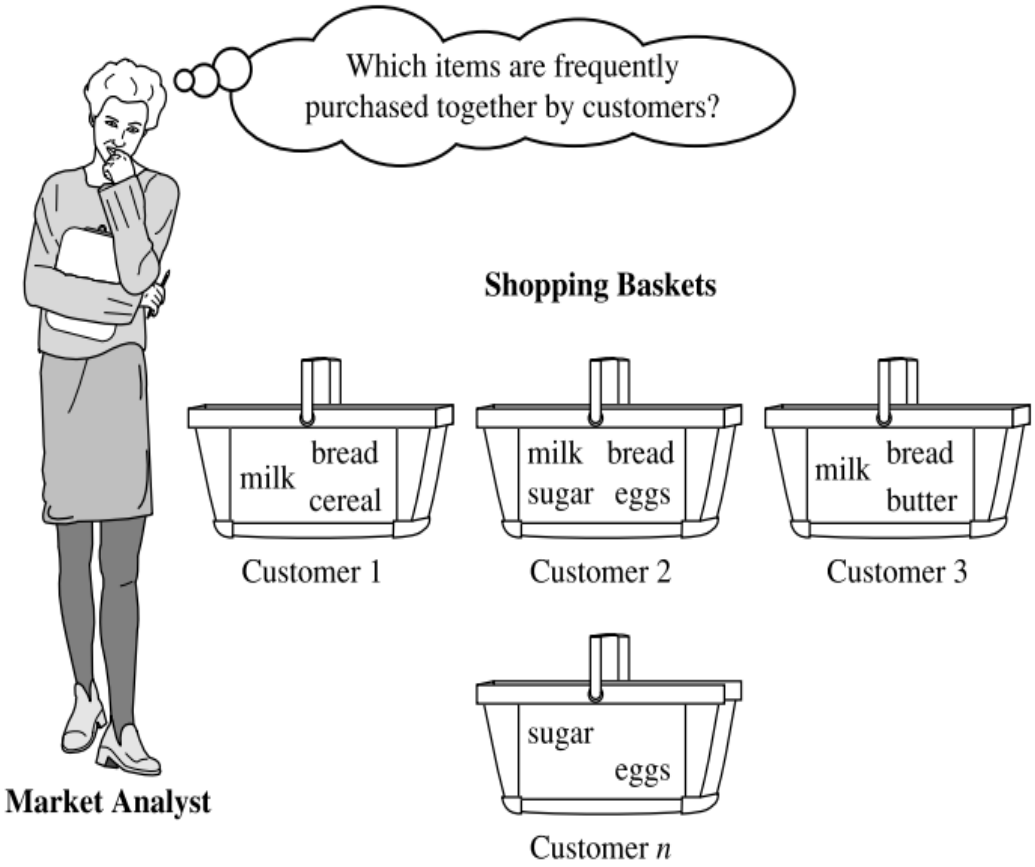


FIGURE 4.1

Market basket analysis.

Why Is Freq. Pattern Mining Important?

- Discloses an intrinsic and important property of data sets
- Forms the foundation for many essential data mining tasks
  - Association, correlation, and causality analysis
  - Sequential, structural (e.g., sub-graph) patterns
  - Pattern analysis in spatiotemporal, multimedia, time-series, and stream data
  - Classification: associative classification
  - Cluster analysis: frequent pattern-based clustering
  - Data warehousing: iceberg cube and cube-gradient
  - Semantic data compression: fascicles
  - Broad applications

What Is Association Mining?

- Association rule mining:
  - Association rule mining searches for interesting relationships among items in a given data set.
  - *Example: 98% of people who purchase tires and auto accessories also get automotive services done*
- Applications:
  - Basket data analysis, sales, catalog design, storage layout, customer segmentation based on buying pattern.

Association Rule: Basic Concepts

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

Formal Definition

- Let $I = \{i_1, i_2, \ldots, i_m\}$ be a set of literals, called items. Let D be a set of transactions, where each transaction T is a set of items (itemsets).
- An association rule is an implication of the form $X \rightarrow Y$, where $X \subset I$, $Y \subset I$, and $X \cap Y = \varnothing$.
- X is called antecedent, Y is called consequence.
- The rule $X \rightarrow Y$ has support *s* if *s*% of transactions in D contains $X \cup Y$.
  - support, *s*, probability that a transaction contains X U Y= % of transactions that contain X U Y

$$\textbf{Support}(X \rightarrow Y) = \frac{\textbf{Number of transactions that contain } X \cup Y}{\textbf{Total number of transactions in D}}$$

- The rule $X \rightarrow Y$ has confidence *c* if *c*% of transactions in D that contains X also contains Y.
  - confidence, *c*, conditional probability that a transaction having X also contains Y

$$\textbf{Confidence}\ (X \rightarrow Y) = \frac{\textbf{Support}\ (X \cup Y)}{\textbf{Support}\ (X)}$$

- Support indicates the frequencies of the occurring patterns while confidence measures the rule's strength.

Association Rule Mining

- Given a set of transactions, find rules that will predict the occurrence of an item based on the occurrences of other items in the transaction

Market-Basket transactions

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

Example of Association Rules

{Diaper} → {Beer},
{Milk, Bread} → {Eggs,Coke},
{Beer, Bread} → {Milk},

Implication means co-occurrence, not causality!

Definition: Frequent Itemset

- **Itemset**
  - A collection of one or more items
    - Example: {Milk, Bread, Diaper}
  - k-itemset
    - An itemset that contains k items
- **Support count (σ)**
  - Frequency of occurrence of an itemset
  - E.g. σ({Milk, Bread,Diaper}) = 2
- **Support**
  - Fraction of transactions that contain an itemset
  - 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 Rule: Example

- **Association Rule**
  - An implication expression of the form $X \rightarrow Y$, where X and Y are itemsets
  - Example:
    {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
  - Confidence (c)
    - Measures how often items in Y appear in transactions that contain X

Example:

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

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

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

Confidence of a rule

- The confidence of a rule provides an accurate prediction on the association of the items in the rule.
- The support of a rule indicates how frequent the rule is in the transactions.
- Rules that have small support are uninteresting, since they do not describe large populations.

Association Rule Mining Task

- Given a set of transactions T, the goal of association rule mining is to find all rules having
  - support $\geq$ *minsup* threshold
  - confidence $\geq$ *minconf* threshold

- Brute-force approach:
  - List all possible association rules
  - Compute the support and confidence for each rule
  - Prune rules that fail the *minsup* and *minconf* thresholds

  $\Rightarrow$ Computationally prohibitive!

Mining Association Rules

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

Example of Rules:

{Milk,Diaper} → {Beer} (s=0.4, c=0.67)
{Milk,Beer} → {Diaper} (s=0.4, c=1.0)
{Diaper,Beer} → {Milk} (s=0.4, c=0.67)
{Beer} → {Milk,Diaper} (s=0.4, c=0.67)
{Diaper} → {Milk,Beer} (s=0.4, c=0.5)
{Milk} → {Diaper,Beer} (s=0.4, c=0.5)

Observations:

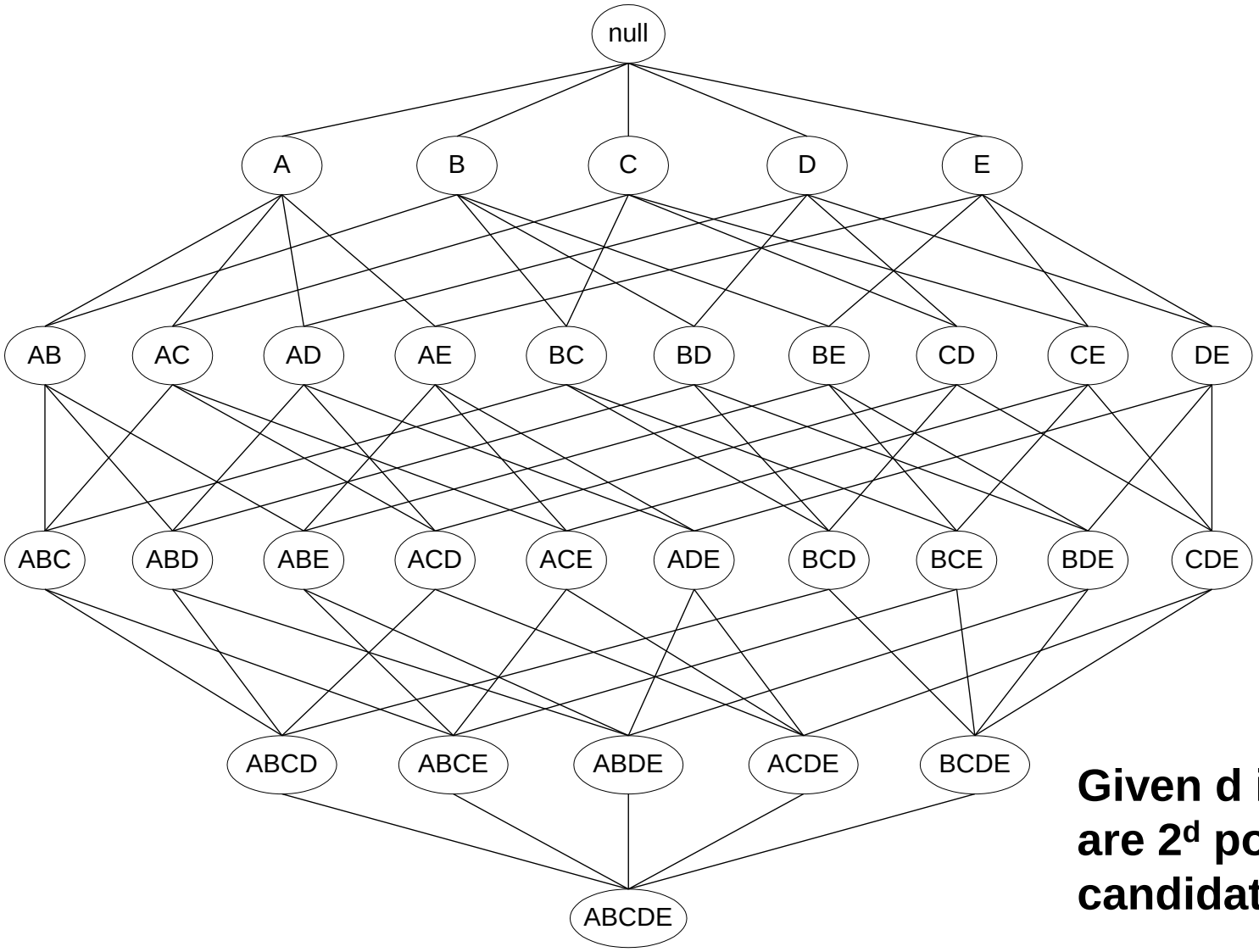
- All the above rules are binary partitions of the same itemset:
  {Milk, Diaper, Beer}
- Rules originating from the same itemset have identical support but can have different confidence
- Thus, we may decouple the support and confidence requirements

Mining Association Rules

- Two-step approach:
  1. Frequent Itemset Generation
     - Generate all itemsets whose support ≥ minsup
  2. Rule Generation
     - Generate high confidence rules from each frequent itemset, where each rule is a binary partitioning of a frequent itemset

- Frequent itemset generation is still computationally expensive

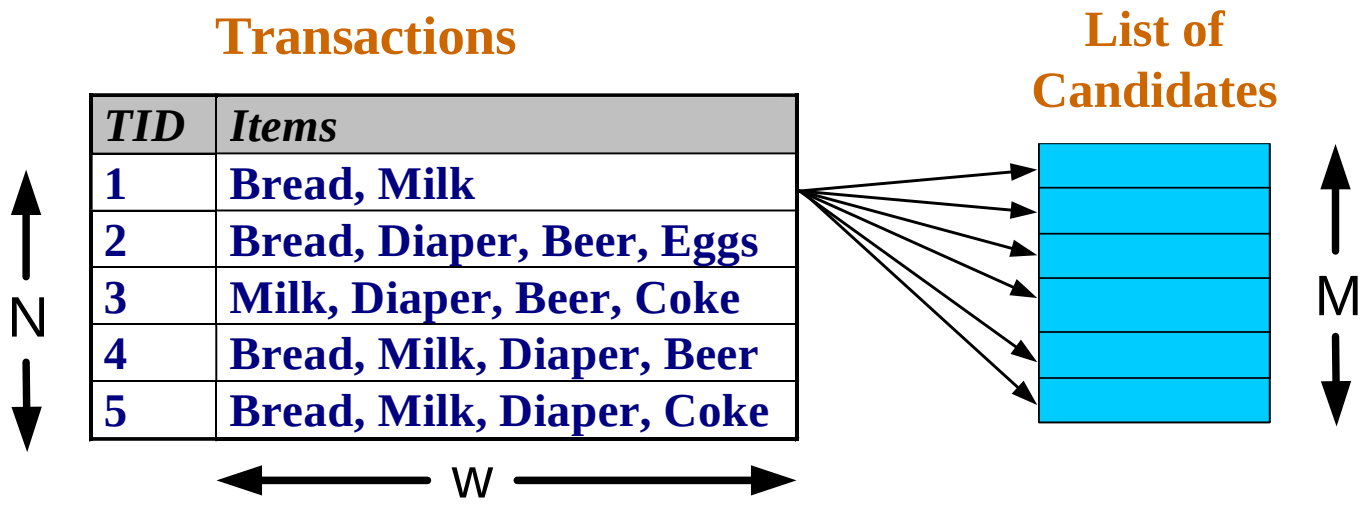
Frequent Itemset Generation



Given d items, there are 2^d possible candidate itemsets

Frequent Itemset Generation

- Brute-force approach:
  - Each itemset in the lattice is a candidate frequent itemset
  - Count the support of each candidate by scanning the database



  - Match each transaction against every candidate
  - Complexity ~ O(NMw) => Expensive since $M = 2^d$!!!

Reducing Number of Candidates

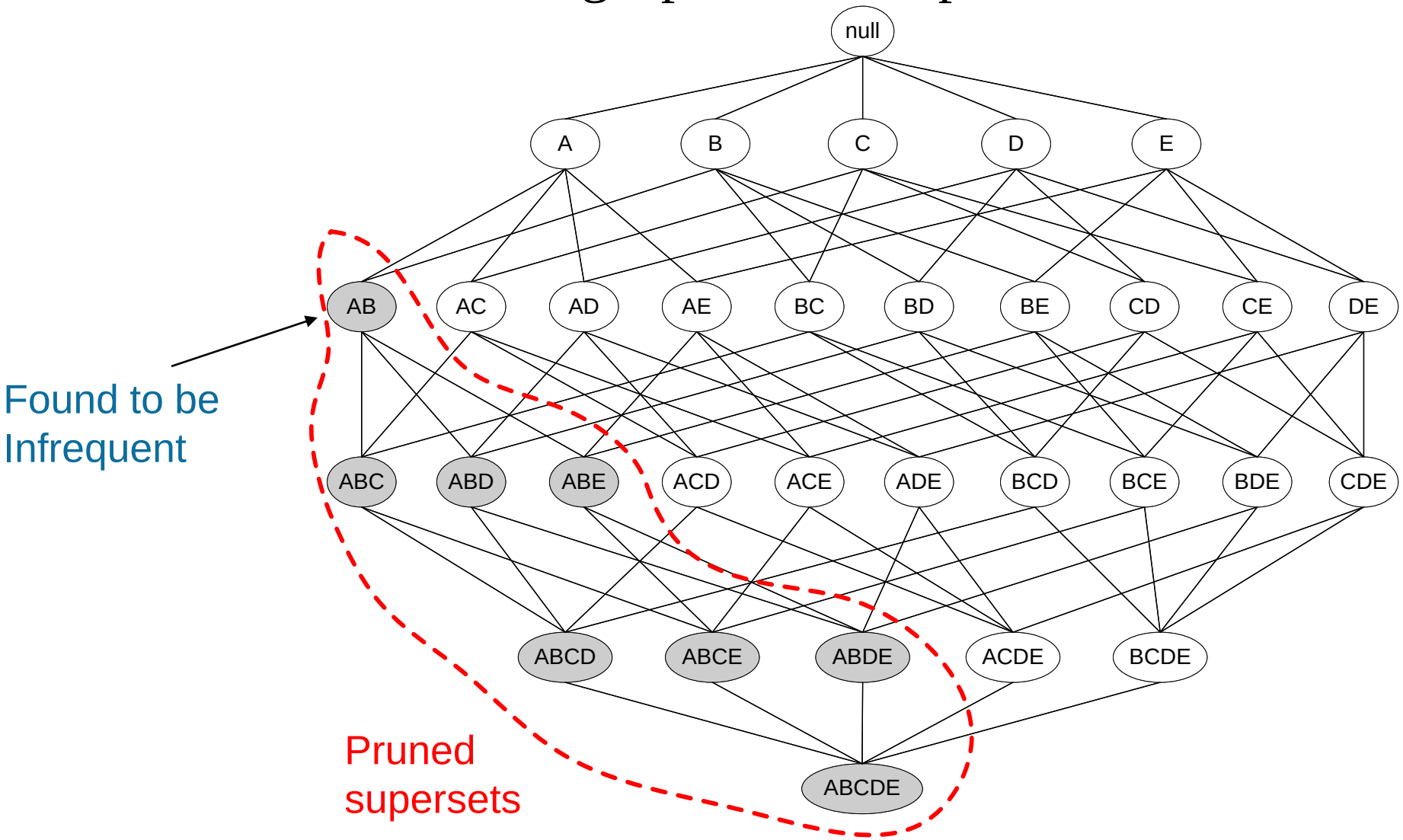
- Apriori principle:
  - If an itemset is frequent, then all of its subsets must also be frequent

- Apriori principle holds due to the following property of the support measure:

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

  - Support of an itemset never exceeds the support of its subsets
  - This is known as the anti-monotone property of support

Illustrating Apriori Principle



Closed Patterns and Max-Patterns

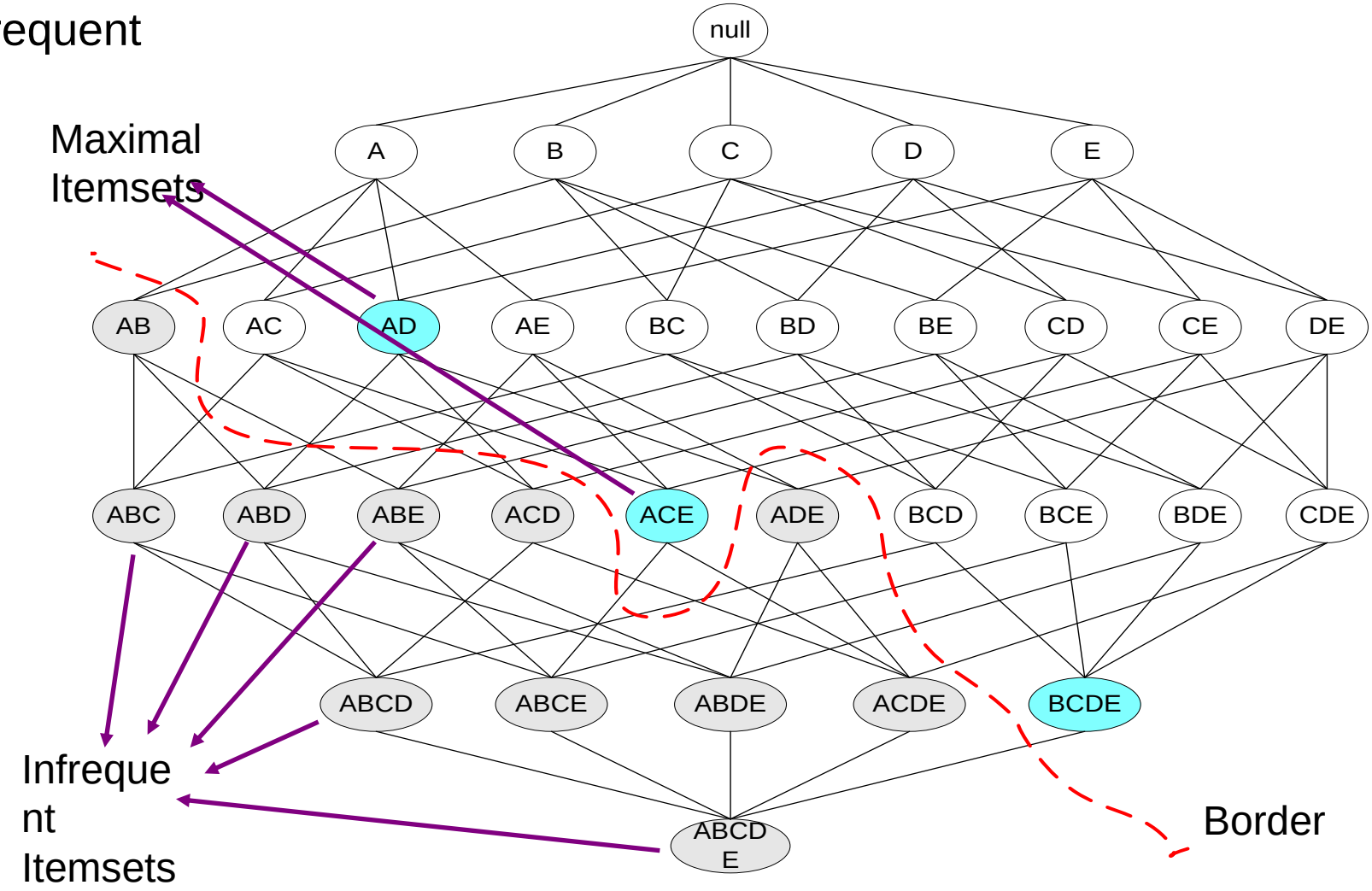
- A long pattern contains a combinatorial number of sub-patterns, e.g., $\{a_1, \ldots, a_{100}\}$ contains $\binom{100}{1} + \binom{100}{2} + \ldots + \binom{1}{1}\binom{0}{0}\binom{0}{0} = 2^{100} - 1 = 1.27*10^{30}$ sub-patterns!
- Solution: *Mine closed patterns and max-patterns instead*
- Closed Pattern:
  - An itemset *X* is **closed** in a data set *D* if there exists no proper super-itemset *Y* such that *Y* has the same support count as *X* in *D*.
    (proposed by Pasquier, et al. @ ICDT'99)
- Closed Frequent Pattern:
  - An itemset *X* is a **closed frequent pattern** in set *D* if *X* is both closed and frequent in *D*.
- Maximal Frequent Pattern:
  - An itemset X is a max-frequent-pattern if X is frequent and there exists no frequent super-pattern Y ⊃ X (proposed by Bayardo @ SIGMOD'98)
- Closed pattern is a lossless compression of freq. patterns
  - Reducing the # of patterns and rules

Closed Patterns and Max-Patterns

- Exercise. DB = {<a_1, …, a_{100}>, < a_1, …, a_{50}>}
  - Min_sup = 1.
- What is the set of closed itemset?
  - <a_1, …, a_{100}>: 1
  - < a_1, …, a_{50}>: 2
- What is the set of max-pattern?
  - <a_1, …, a_{100}>: 1
- What is the set of all patterns?
  - !!

Maximal Frequent Itemset

An itemset is maximal frequent if none of its immediate supersets is frequent



Closed Itemset

- An itemset is closed if none of its immediate supersets has the same support as the itemset

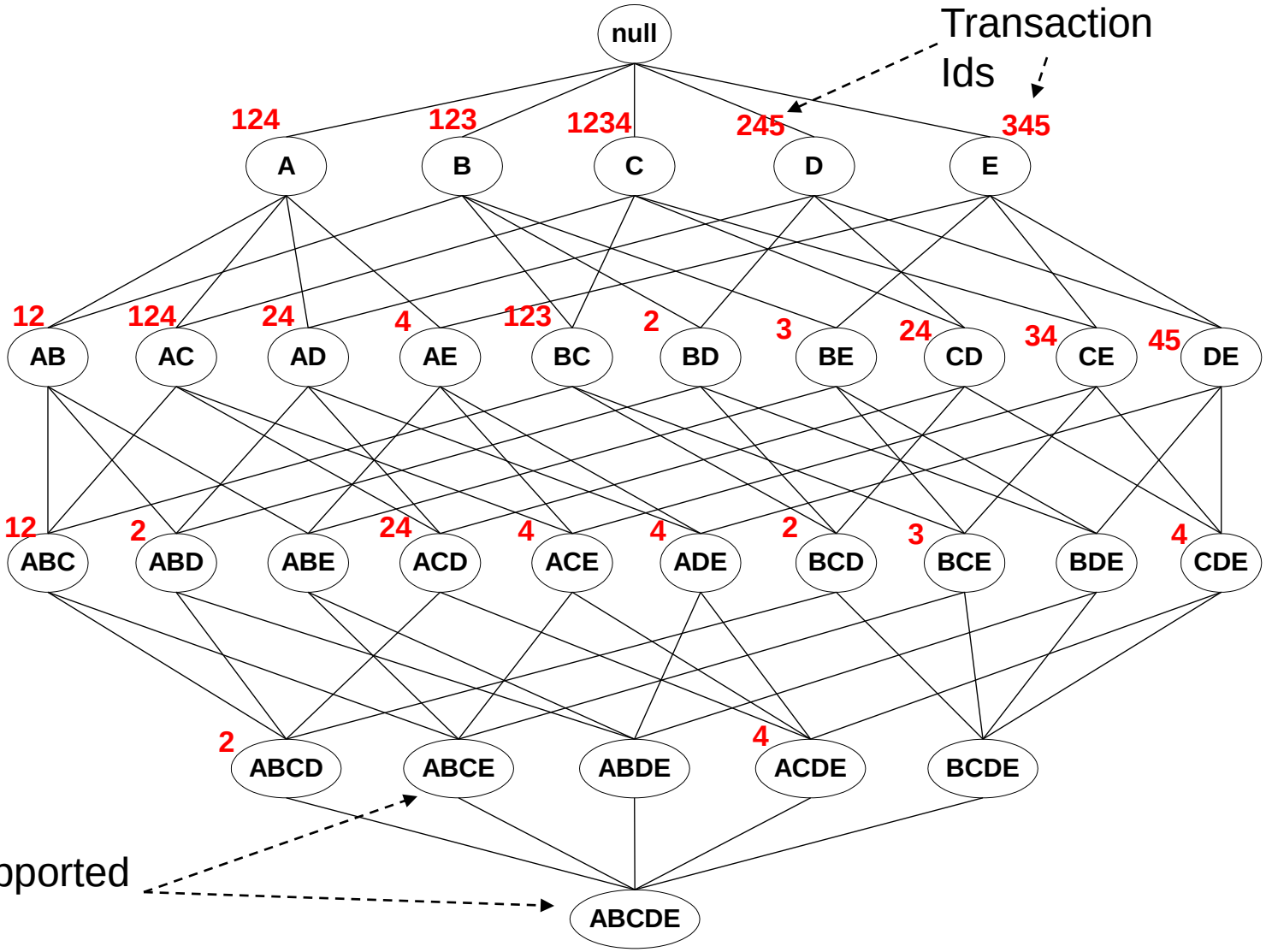
TID	Items
1	{A,B}
2	{B,C,D}
3	{A,B,C,D}
4	{A,B,D}
5	{A,B,C,D}

Itemset	Support
{A}	4
{B}	5
{C}	3
{D}	4
{A,B}	4
{A,C}	2
{A,D}	3
{B,C}	3
{B,D}	4
{C,D}	3

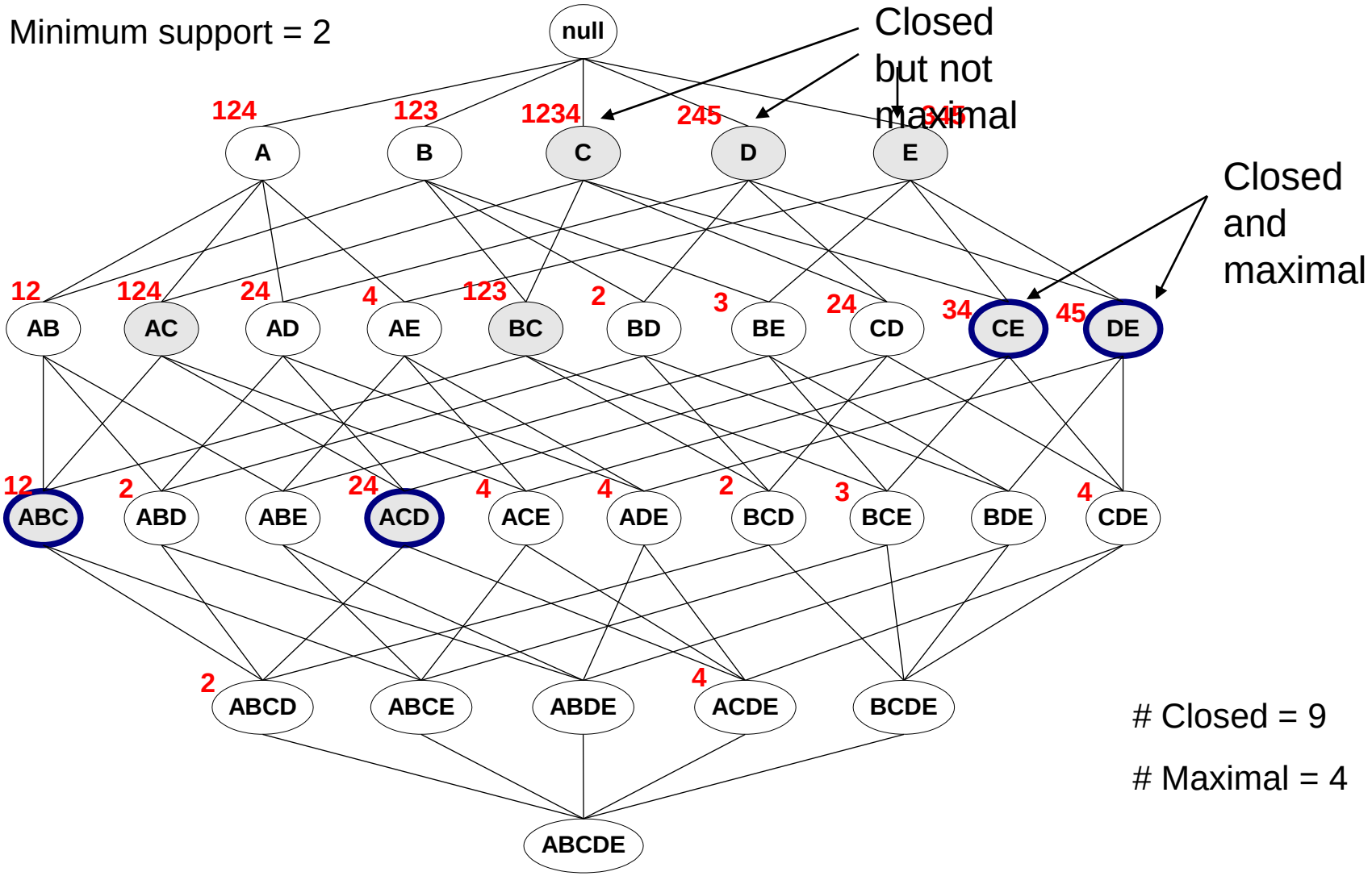
Itemset	Support
{A,B,C}	2
{A,B,D}	3
{A,C,D}	2
{B,C,D}	3
{A,B,C,D}	2

Maximal vs Closed Itemsets

TID	Items
1	ABC
2	ABCD
3	BCE
4	ACDE
5	DE



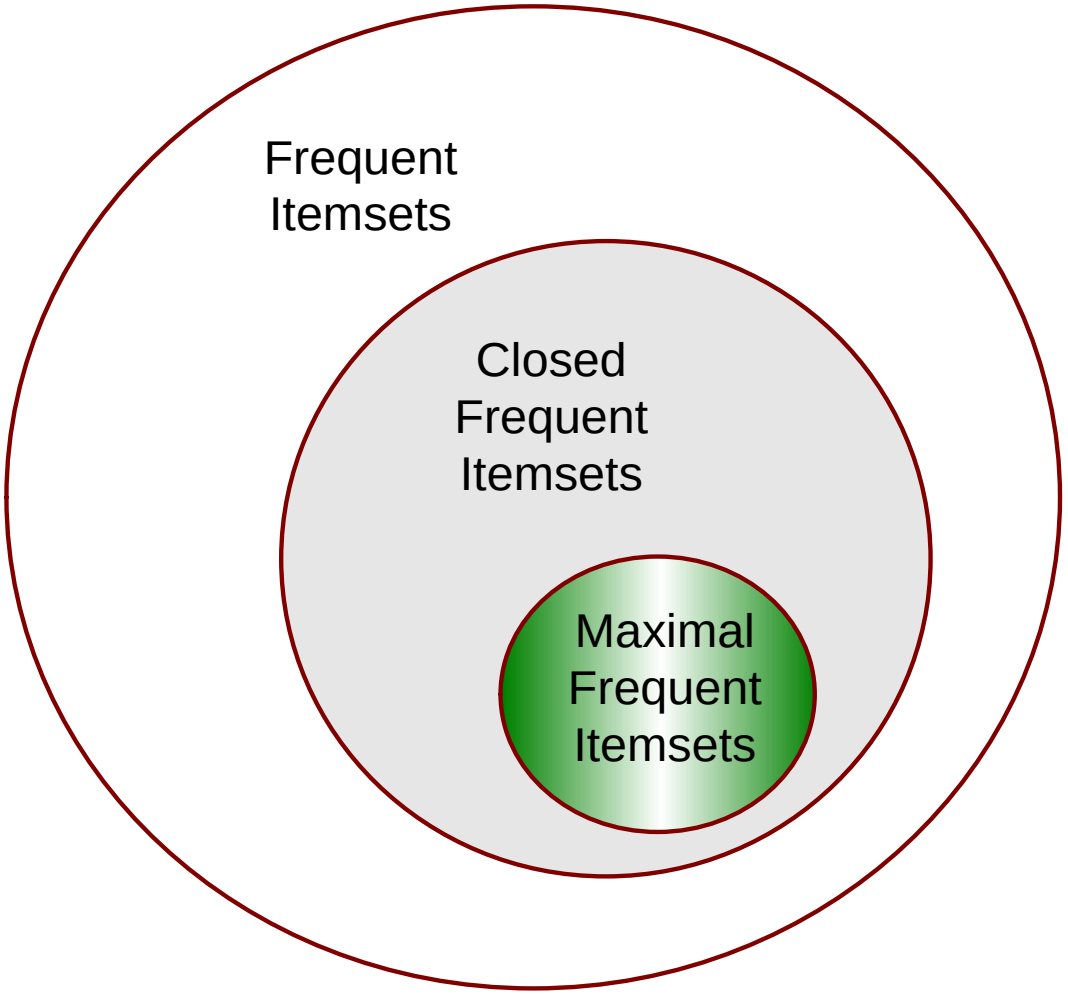
Maximal vs Closed Frequent Itemsets



About Closed frequent itemsets and Maximal frequent itemsets

- Let *C* be the set of closed frequent itemsets for a data set *D* satisfying a minimum support threshold, *min sup*.
- Let *M* be the set of maximal frequent itemsets for *D* satisfying *min sup*.
- Suppose that we have the support count of each itemset in *C* and *M*.
- Notice that C and its count information can be used to derive the whole set of frequent itemsets
  - Thus, we say that C contains complete information regarding its corresponding frequent itemsets.
  - On the other hand, M registers only the support of the maximal itemsets.
    - We can only assert about the support, but exact count is not known

Maximal vs Closed Itemsets



Computational Complexity of Frequent Itemset Mining

- How many itemsets are potentially to be generated in the worst case?
  - The number of frequent itemsets to be generated is senstive to the minsup threshold
  - When minsup is low, there exist potentially an exponential number of frequent itemsets
  - The worst case: M^N where M: # distinct items, and N: max length of transactions

Frequent Itemset Generation Strategies

- Reduce the number of candidates (M)
  - Complete search: $M=2^d$
  - Use pruning techniques to reduce M
- Reduce the number of transactions (N)
  - Reduce size of N as the size of itemset increases
  - Used by DHP and vertical-based mining algorithms
- Reduce the number of comparisons (NM)
  - Use efficient data structures to store the candidates or transactions
  - No need to match every candidate against every transaction

Presentation Outline

- Basic concepts
  - Frequent itemssets, Closed Itemsets, and Association Rules
- Frequent itemset mining methods
  - **Apriori Algorithm**
    - **Generating Association Rules from Frequent Itemsets**
    - Improving the Efficiency of Apriori
  - A Pattern-Growth Approach
  - Mining Frequent Itemsets Using the Vertical Data Format
  - Mining Closed and Max Patterns
- Which Patterns are interesting? Pattern Evaluation Methods
- Summary

Fast Algorithms for Mining Association Rules

Rakesh Agrawal Ramakrishnan Srikant*

IBM Almaden Research Center
650 Harry Road, San Jose, CA 95120

Proceedings of the 20th VLDB Conference
Santiago, Chile, 1994

Apriori: A Candidate Generation & Test Approach

- Apriori pruning principle: If there is any itemset which is infrequent, its superset should not be generated/tested!
- Method:
  - Initially, scan DB once to get frequent 1-itemset
  - Generate length (k+1) candidate itemsets from length k frequent itemsets
  - Test the candidates against DB
  - Terminate when no frequent or candidate set can be generated

The Trick

Any subset of large itemset is large.

Therefore

To find large k-itemset

- Create candidates by combining large k-1 itemsets.
- Delete those that contain any subset that is not large.

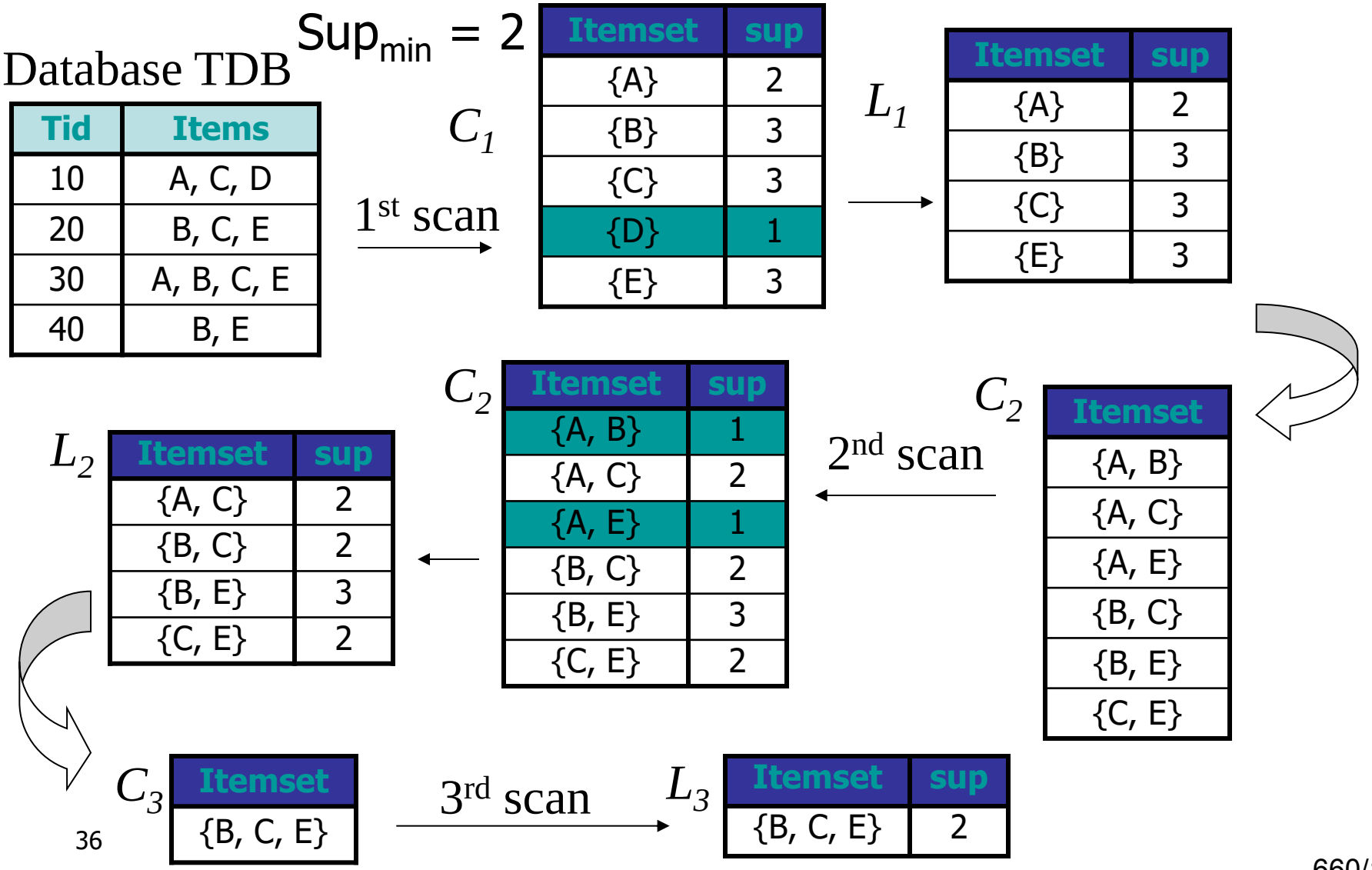
Apriori Algorithm

Notations

- k-itemset: An itemset having k items
- L_k: Set of large k-itemsets (those with minimum support). Each member of this set has two fields: i) itemset and ii) support count
- C_k: Set of candidate k-itemsets (potentially large itemsets). Each member of this set has two fields: i) itemset and ii) support count

- *Candidate* itemsets are generated using only the *large* itemsets of the previous pass without considering the transactions in the database.
- The *large* itemset of the previous pass is joined with itself to generate all itemsets whose size is higher by 1.
- Each generated itemset, that has a subset which is not *large*, is deleted. The remaining itemsets are the *candidate* ones.

The Apriori Algorithm—An Example



The Apriori Algorithm (Pseudo-Code)

1) L_1 = {large 1-itemsets};
2) **for** ($k = 2$; $L_{k-1} \neq \emptyset$; k++) **do begin**
3) C_k = apriori-gen(L_{k-1}); // New candidates
4) **forall transactions** $t \in \mathcal{D}$ **do begin**
5) C_t = subset(C_k, t); // Candidates contained in t
6) **forall candidates** $c \in C_t$ **do**
7) c.count++;
8) **end**
9) $L_k = \{c \in C_k \mid c.\text{count} \geq \text{minsup}\}$
10) **end**
11) Answer = $\bigcup_k L_k$;

Figure 1: Algorithm Apriori

Notations

- k-itemset: An itemset having k items
- L_k: Set of large k-itemsets (those with minimum support). Each member of this set has two fields: i) itemset and ii) support count
- C_k: Set of candidate k-itemsets (potentially large itemsets). Each member of this set has two fields: i) itemset and ii) support count

Apriori-gen(L_{k-1})

The **apriori-gen** function takes as argument L_{k-1}, the set of all large $(k-1)$-itemsets. It returns a superset of the set of all large k-itemsets. The function works as follows. [1] First, in the *join* step, we join L_{k-1} with L_{k-1}:

insert into C_k
select $p.\text{item}_1$, $p.\text{item}_2$, ..., $p.\text{item}_{k-1}$, $q.\text{item}_{k-1}$
from L_{k-1} p, L_{k-1} q
where $p.\text{item}_1 = q.\text{item}_1$, ..., $p.\text{item}_{k-2} = q.\text{item}_{k-2}$,
$p.\text{item}_{k-1} < q.\text{item}_{k-1}$;

Prune Step

- In the Prune step we delete all itemsets c ε C_k Ck such that some (k-1)-subset of c is not in L_{k-1}

forall itemsets $c \in C_k$ **do**
  forall $(k-1)$-subsets s of c **do**
    if $(s \notin L_{k-1})$ **then**
      delete c from C_k;

Self-joining and Pruning: Example

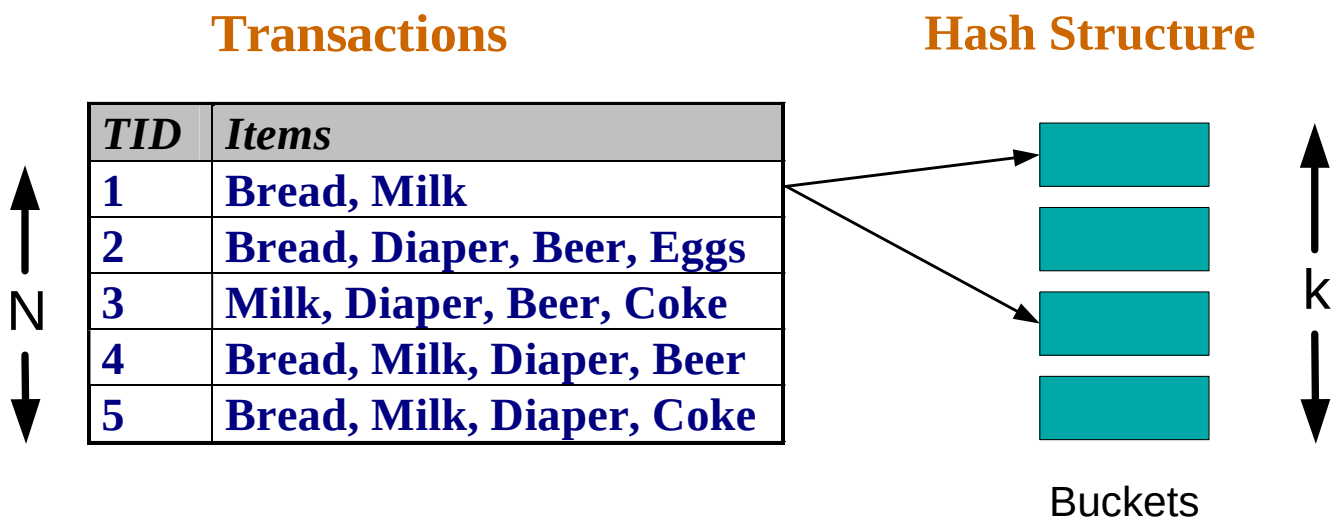
- How to generate candidates?
  - Step 1: self-joining L_k
  - Step 2: pruning
- Example of Candidate-generation
  - L_3={*abc*, *abd*, *acd*, *ace*, *bcd*}
  - Self-joining: L_3*L_3
    - *abcd* from *abc* and *abd*
    - *acde* from *acd* and *ace*
  - Pruning:
    - *acde* is removed because *ade* is not in L_3
  - C_4 = {*abcd*}

How to Count Supports of Candidates?

- Why counting supports of candidates a problem?
  - The total number of candidates can be very huge
  - One transaction may contain many candidates
- Method:
  - Candidate itemsets are stored in a *hash-tree*
  - *Leaf* node of hash-tree contains a list of itemsets and counts
  - *Interior* node contains a hash table
  - *Subset function*: finds all the candidates contained in a transaction

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



How to Count Supports of Candidates?

- Candidate itemsets Ck are stored in a hash-tree.
- A node of the hash-tree either contains a list of itemsets (a leaf node) or a hash table (an interior node).
- In an interior node, each bucket of the hash table points to another node. The root of the hash-tree is defined to be at depth 1.
- An interior node at depth d points to nodes at depth d+1.
- Itemsets are stored in the leaves.
- When we add an itemset c, we start from the root and go down the tree until we reach a leaf. At an interior node at depth d, we decide which branch to follow by applying a hash function to the d'th item of the itemset.
- All nodes are initially created as leaf nodes. When the number of itemsets in a leaf node exceeds a specified threshold, the leaf node is converted to an interior node.

How to Count Supports of Candidates?

- Starting from the root node, the subset function finds all the candidates contained in a transaction t as follows.
  - If we are at a leaf, we finds which of the itemsets in the leaf are contained in t and add references to them to the answer set.
  - If we are at an interior node and we have reached it by hashing the item i, we hash on each item that comes after i in t and recursively apply this procedure to the node in the corresponding bucket. For the root node, we hash on every item in t.

How to Count Supports of Candidates?

- To see why the subset function returns the desired set of references, consider what happens at the root node.
  - For any itemset c contained in transaction t, the first item of c must be in t.
  - At the root, by hashing on every item in t, we ensure that we only ignore itemsets that start with an item not in t. Similar arguments apply at lower depths.
  - The only additional factor is that, since the items in any itemset are ordered, if we reach the current node by hashing the item i, we only need to consider the items in t that occur after i.
  - If k is the size of a candidate itemset in the hash-tree, we can find in O(k) time whether the itemset is contained in a transaction by using a temporary bitmap. Each bit of the bitmap corresponds an item. The bitmap is created once for the data structure, and reinitialized for each transaction. This initialization takes O(size(transaction)) time for each transaction.

Subset Function

- Candidate itemsets - C_k are stored in a hash-tree
- Finds in O(k) time whether a candidate itemset of size k is contained in transaction t.
- Total time O(max(k,size(t))

$L_1 = \{large\ 1\text{-}itemsets\}$
For $(k = 2;\ L_{k-1} \neq \phi\ ;\ k++)$ do begin
    C_k = apriori-gen (L_{k-1});
    forall transactions $t \in D$ do begin
        C_t = subset (C_k, t)
        forall candidates $c \in C_t$ do
            $c.count++$;
        end
    end
    $L_k = \{ c \in C_k | c.count \geq minsup\}$
end
$Answer = \bigcup_k L_k$;

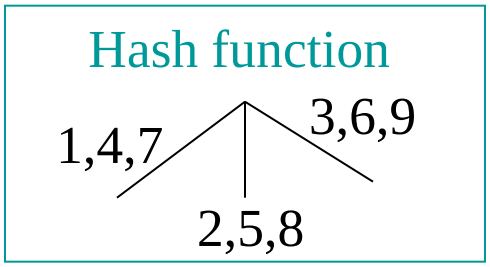
Generate Hash Tree

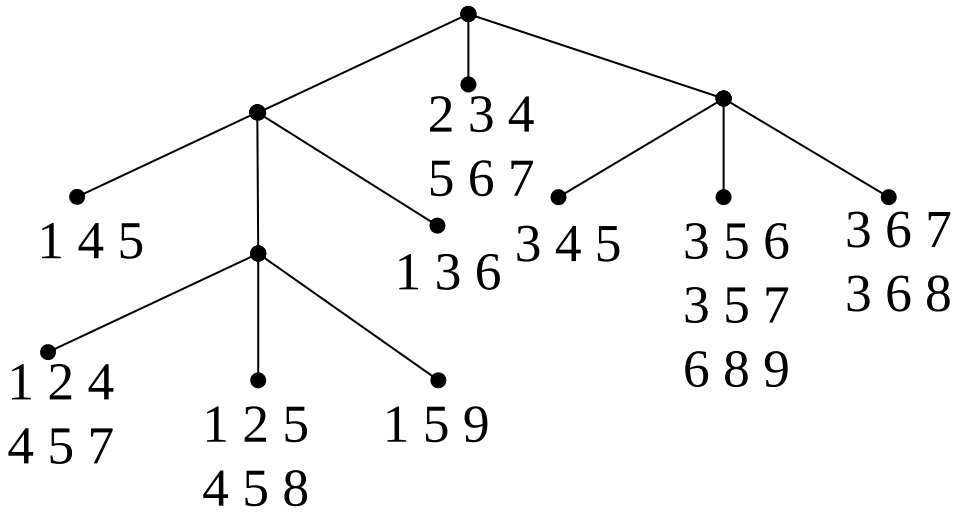
Suppose you have 15 candidate itemsets of length 3:

{1 4 5}, {1 2 4}, {4 5 7}, {1 2 5}, {4 5 8}, {1 5 9}, {1 3 6}, {2 3 4}, {5 6 7}, {3 4 5}, {3 5 6}, {3 5 7}, {6 8 9}, {3 6 7}, {3 6 8}

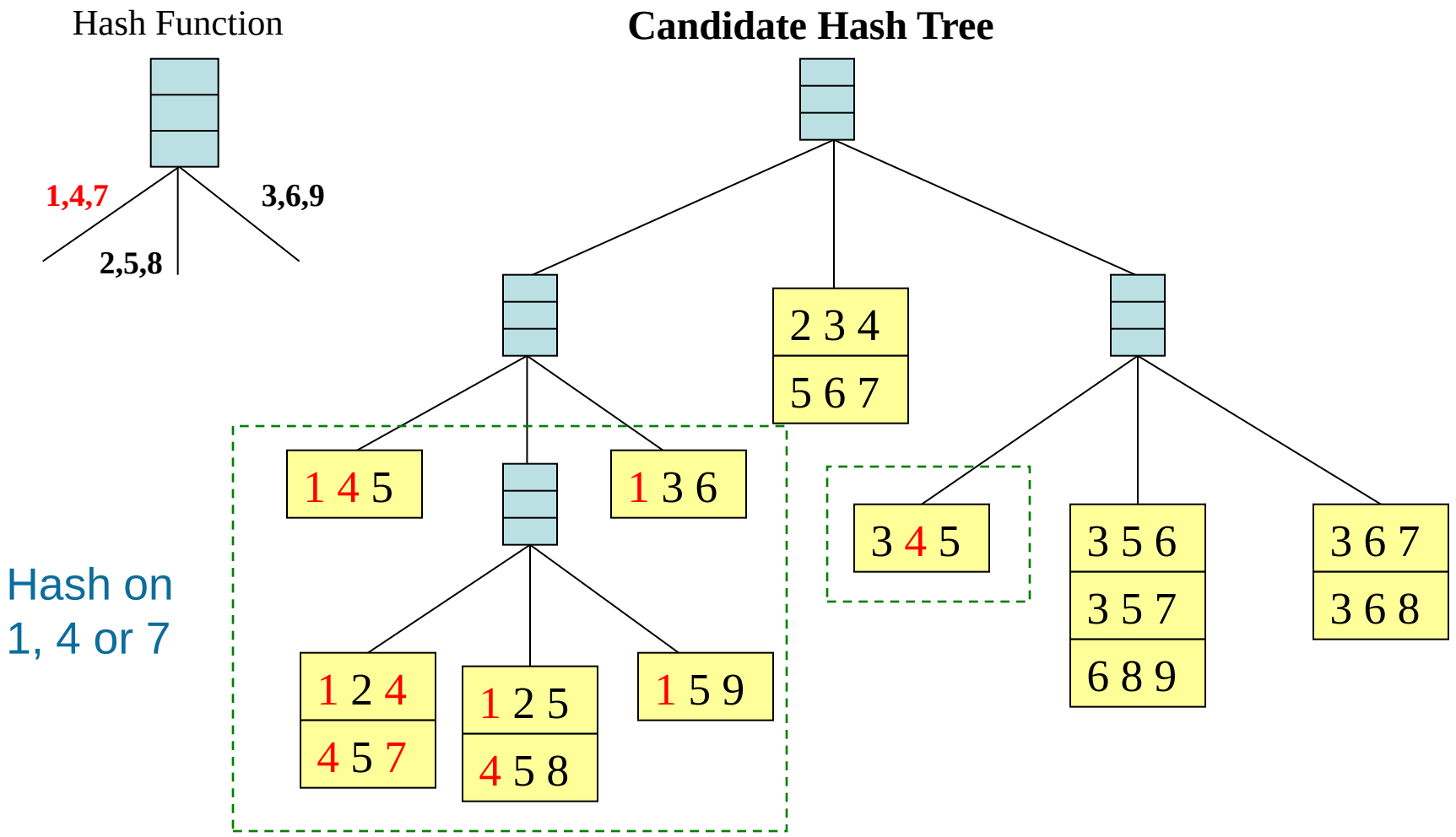
You need:

- **Hash function**
- **Max leaf size: max number of itemsets stored in a leaf node (if number of candidate itemsets exceeds max leaf size, split the node)**

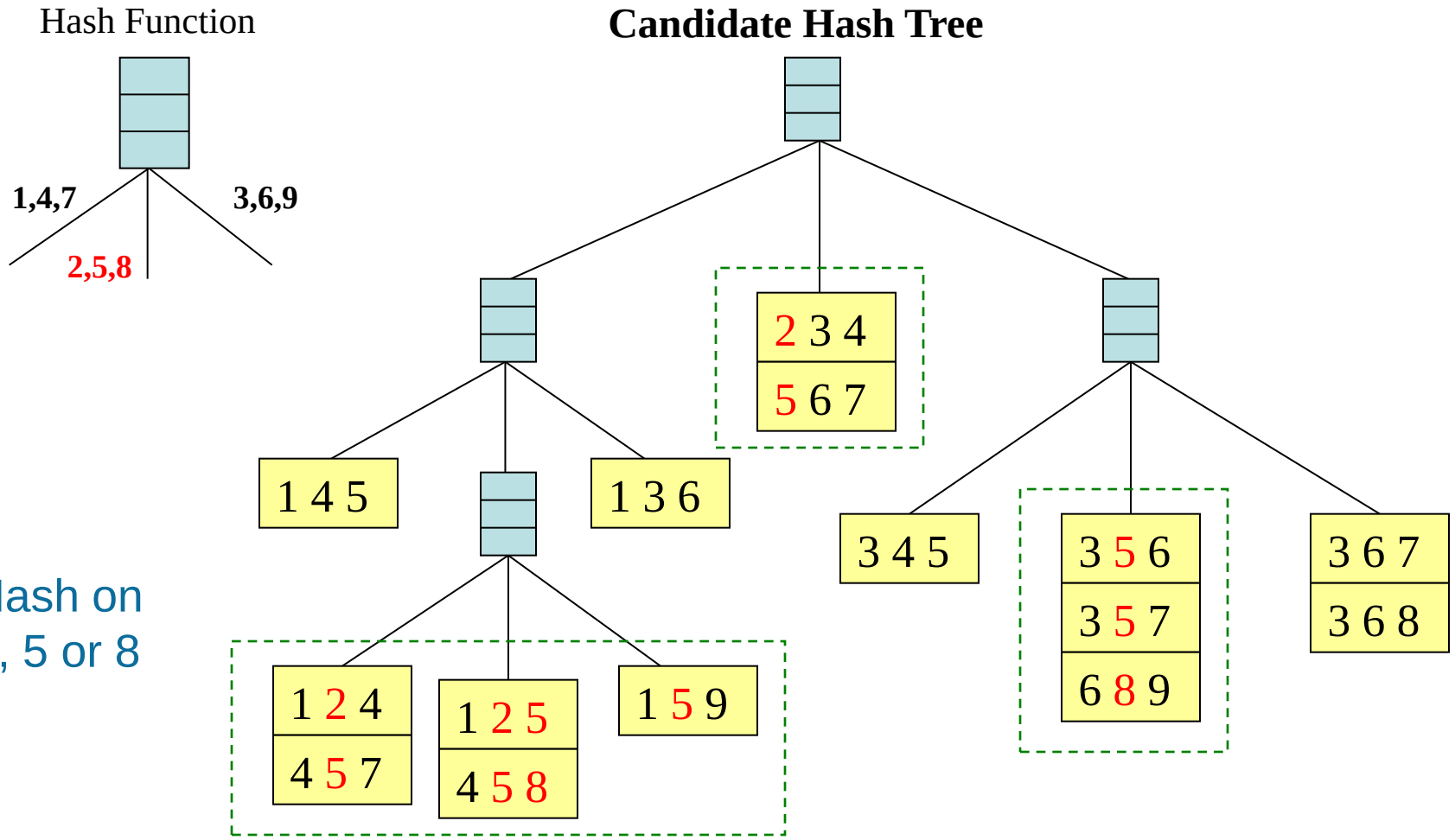




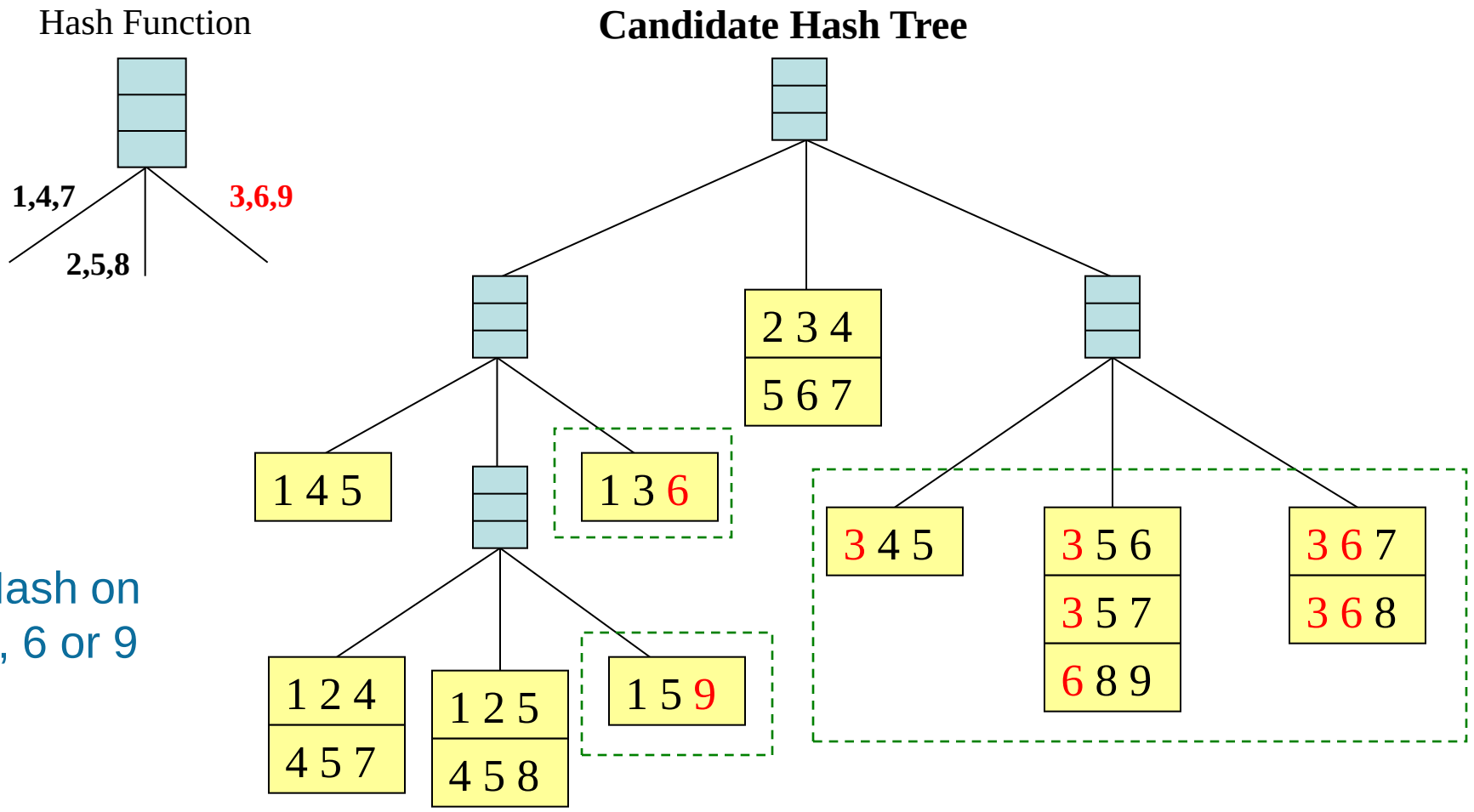
Association Rule Discovery: Hash tree



Association Rule Discovery: Hash tree

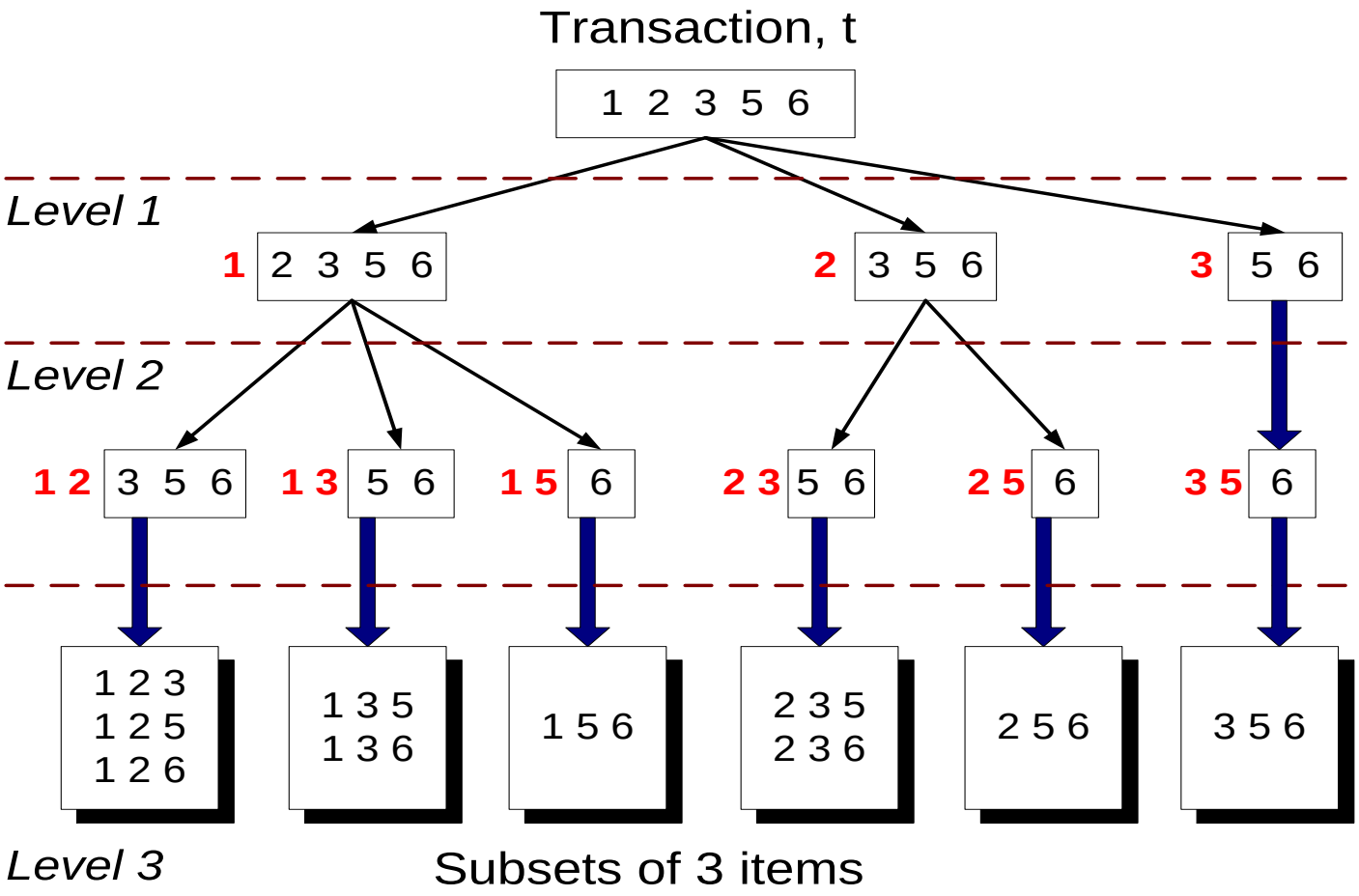


Association Rule Discovery: Hash tree

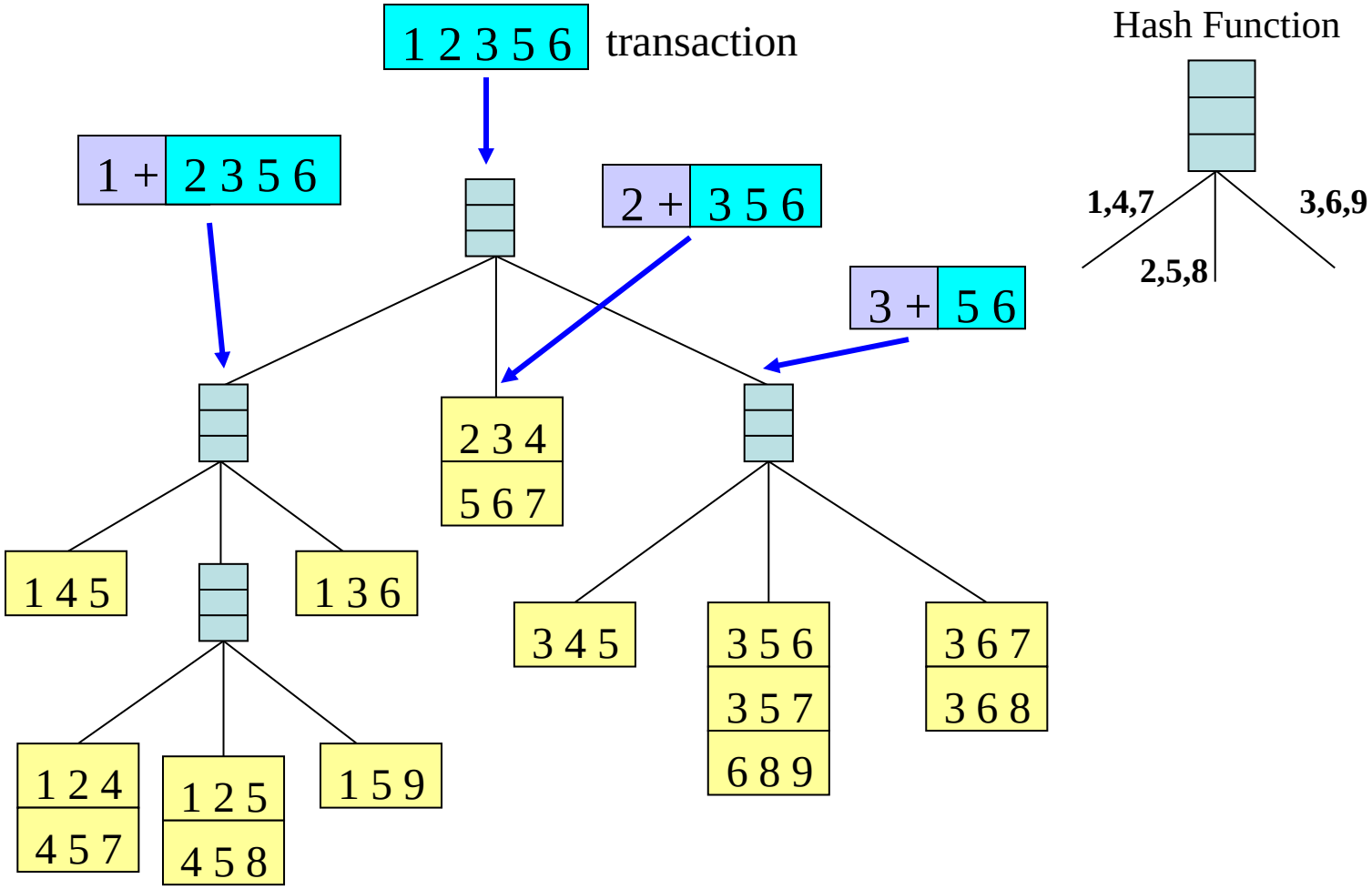


Subset Operation

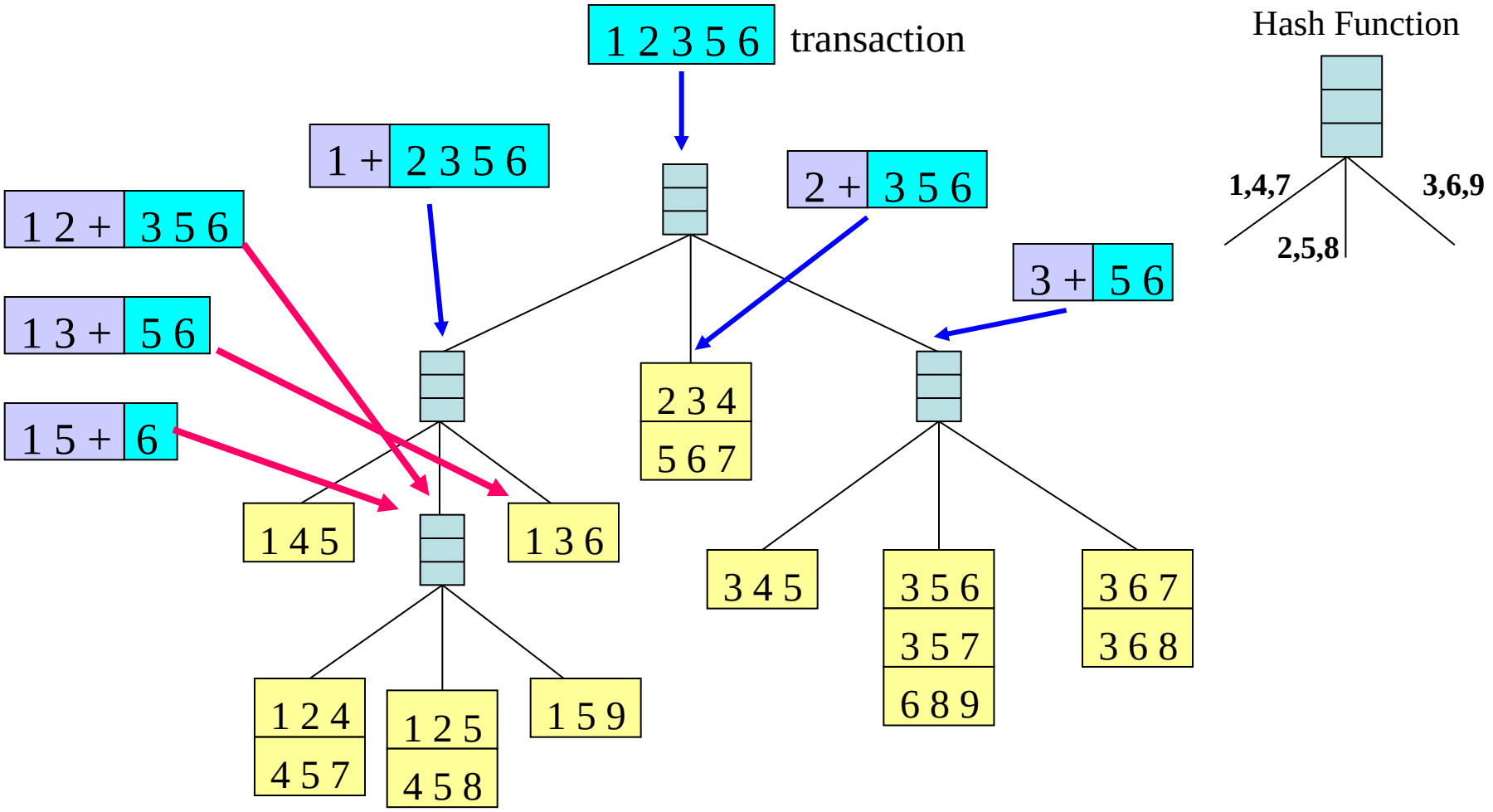
Given a transaction t, what are the possible subsets of size 3?



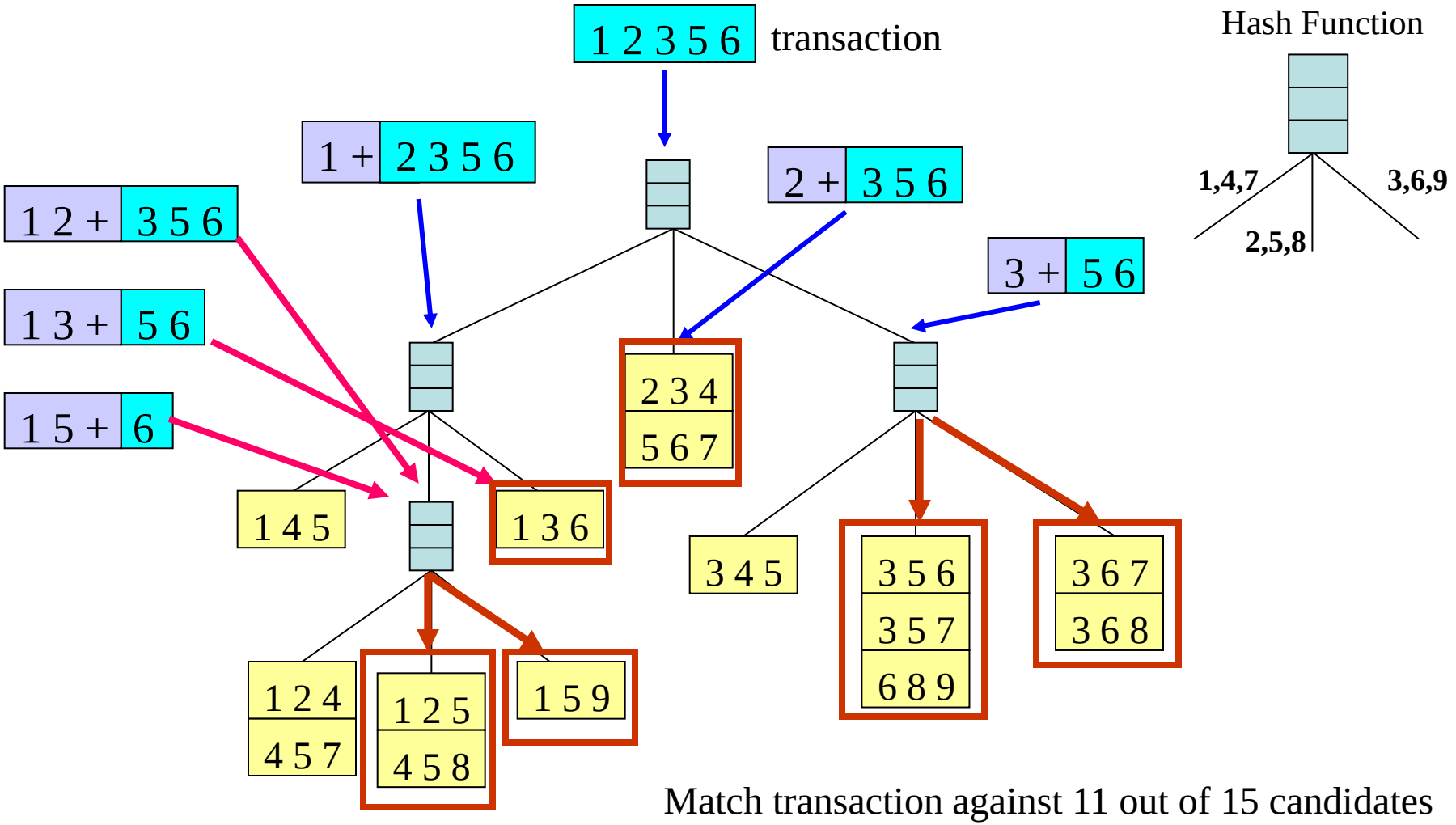
Subset Operation Using Hash Tree



Subset Operation Using Hash Tree

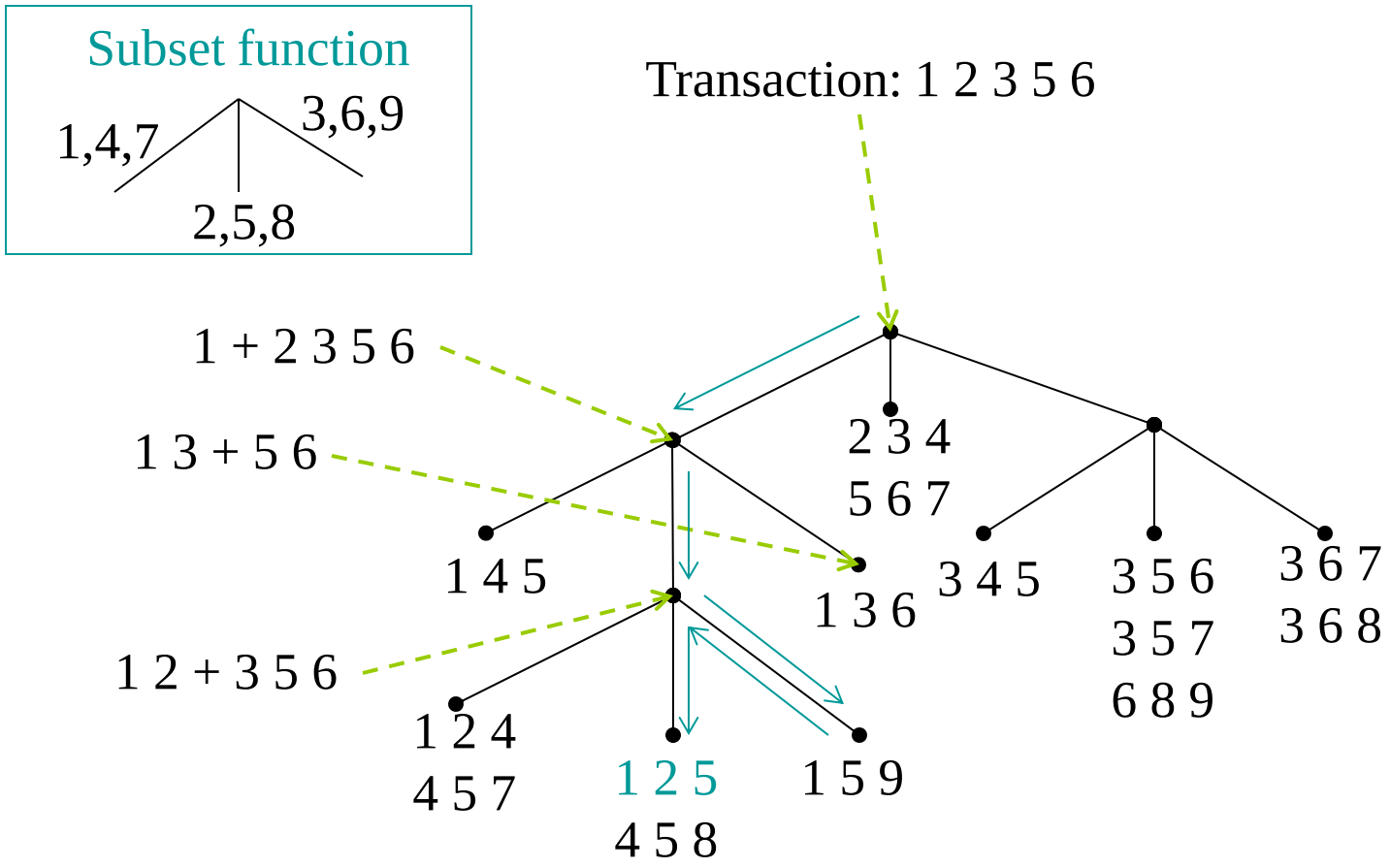


Subset Operation Using Hash Tree



Match transaction against 11 out of 15 candidates

Example: Counting Supports of Candidates



Generating Association Rules

- Once the frequent itemsets are found, it is straightforward to generate strong association rules from them (where strong association rules satisfy both minimum support and minimum confidence).

$$\textbf{Confidence } (X \rightarrow Y) = \frac{\textbf{Support } (X \cup Y)}{\textbf{Support } (X)}$$

- For each frequent itemset *l*, generate all nonempty subsets of *l*.
- For every nonempty subset *s* of *l*, output the rule $l \rightarrow (l-s)$,

  if $\frac{support\text{-}count\ (l)}{support\text{-}count\ (s)} >= min_conf$, where *min conf* is the minimum confidence threshold.
- Because the rules are generated from frequent itemsets, each one automatically satisfies the minimum support.
- Frequent itemsets can be stored ahead of time in hash tables along with their counts so that they can be accessed quickly

Example

- The data contain frequent itemset *X* (I1, I2, I5). What are the association rules that can be generated from *X*?
- Generate the nonempty subsets of *X* are {I1, I2}, {I1, I5}, {I2, I5}, {I1}, {I2}, and {I5}.
- The resulting association rules are
  - {I1, I2}→{I5}= confidence=2/4=50%
  - {I1, I5}→{I2}=2/2=100%
  - {I2, I5}→ {I1}=2/2=100%
  - {I1}→{I2,I5}= 2/6=33%
- ….
- …
- If minimum confidence =70%, second and third are potential rules

Table 6.1 Transactional Data for an *AllElectronics* Branch

TID	*List of item_IDs*
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

Presentation Outline

- Basic concepts
  - Frequent itemssets, Closed Itemsets, and Association Rules
- Frequent itemset mining methods
  - Apriori Algorithm
    - Generating Association Rules from Frequent Itemsets
    - **Improving the Efficiency of Apriori**
  - A Pattern-Growth Approach
  - Mining Frequent Itemsets Using the Vertical Data Format
  - Mining Closed and Max Patterns
- Which Patterns are interesting? Pattern Evaluation Methods
- Summary

Further Improvement of the Apriori Method

- Apriori: Major computational challenges
  - Multiple scans of transaction database
  - Huge number of candidates
  - Tedious workload of support counting for candidates
- Improving Apriori: general ideas
  - Reduce passes of transaction database scans
  - Shrink number of candidates
  - Facilitate support counting of candidates

DHP (Direct Hashing and Pruning): Reduce the Number of Candidates

- Use of hash to reduce the size of candidate *k*-itemsets, *Ck*, for *k* > 1.
- For example, when scanning each transaction in the database to generate the frequent 1-itemsets, *L*1, we can generate all the 2-itemsets for each transaction, hash (i.e., map) them into the different *buckets* of a *hash table* structure, and increase the corresponding bucket counts.
- A 2-itemset with a corresponding bucket count in the hash table that is below the support threshold cannot be frequent (especially when k=2)
- J. Park, M. Chen, and P. Yu. An effective hash-based algorithm for mining association rules. *SIGMOD'95*

Create hash table H_2 using hash function $h(x, y) = ((order\ of\ x) \times 10 + (order\ of\ y))\ mod\ 7$

H_2

bucket address	0	1	2	3	4	5	6
bucket count	2	2	4	2	2	4	4
bucket contents	{I1, I4} {I3, I5}	{I1, I5} {I1, I5}	{I2, I3} {I2, I3} {I2, I3} {I2, I3}	{I2, I4} {I2, I4}	{I2, I5} {I2, I5}	{I1, I2} {I1, I2} {I1, I2} {I1, I2}	{I1, I3} {I1, I3} {I1, I3} {I1, I3}

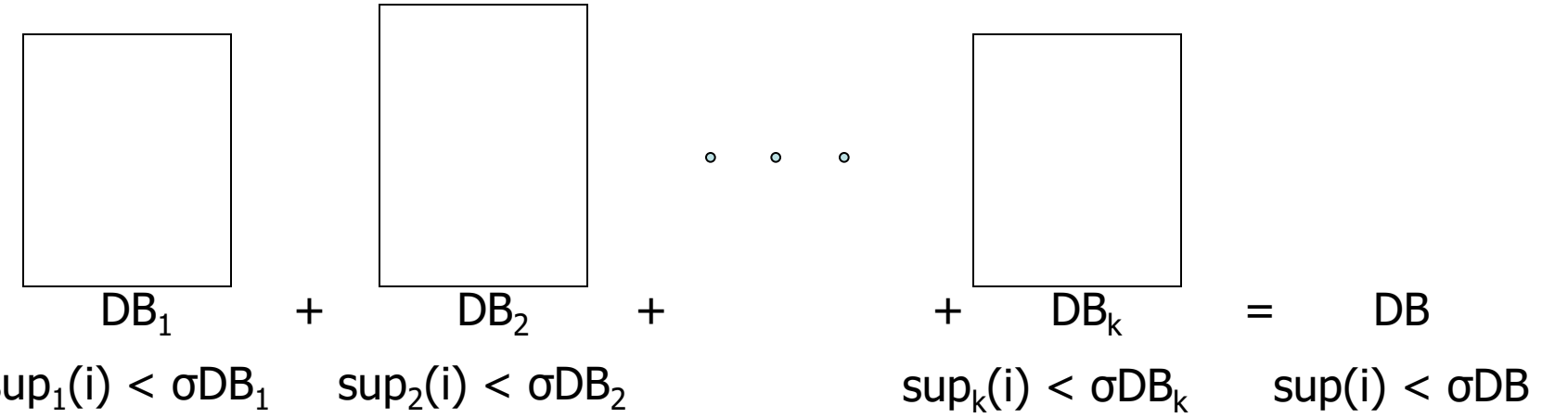
Figure 6.5 Hash table, H_2, for candidate 2-itemsets. This hash table was generated by scanning Table 6.1's transactions while determining L_1. If the minimum support count is, say, 3, then the itemsets in buckets 0, 1, 3, and 4 cannot be frequent and so they should not be included in C_2.

Transaction Reduction

- Reducing the number of transactions scanned in future iterations):
- A transaction that does not contain any frequent *k*-itemsets cannot contain any
  frequent (k+1)-itemsets.
- Therefore, such a transaction can be marked or removed
  from further consideration because subsequent database scans for *j*-itemsets, where

Partition: Scan Database Only Twice

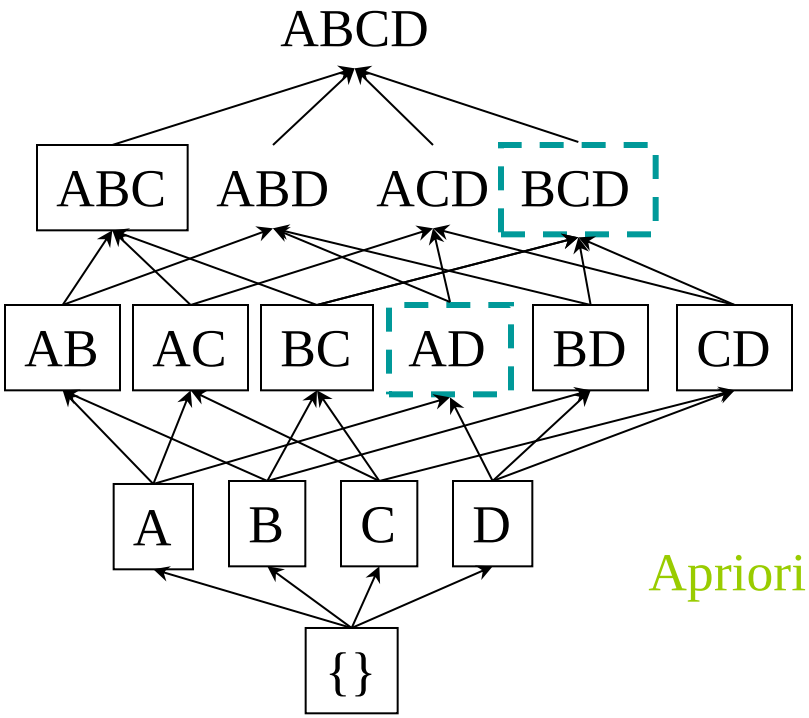
- Any itemset that is potentially frequent in DB must be frequent in at least one of the partitions of DB
  - Scan 1: partition database and find local frequent patterns
  - Scan 2: consolidate global frequent patterns
- A. Savasere, E. Omiecinski and S. Navathe, *VLDB'95*



Sampling for Frequent Patterns

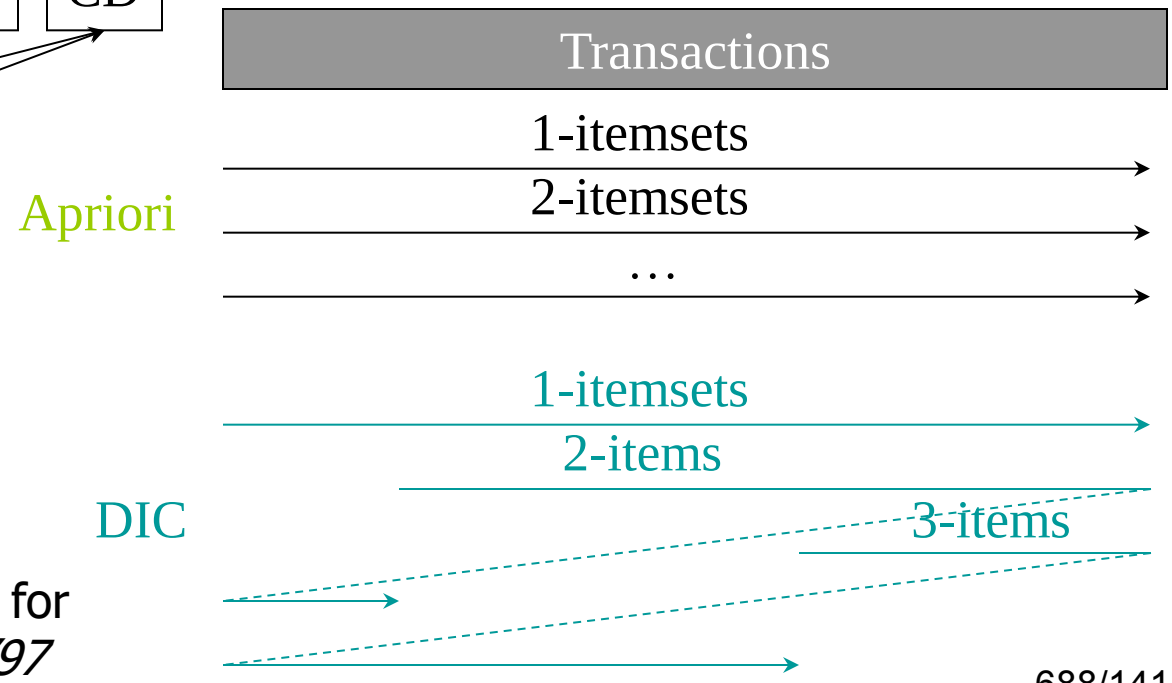
- Select a sample of original database, mine frequent patterns within sample using Apriori
- Scan database once to verify frequent itemsets found in sample, only *borders* of closure of frequent patterns are checked
  - Example: check *abcd* instead of *ab, ac, …, etc.*
- Scan database again to find missed frequent patterns
- H. Toivonen. Sampling large databases for association rules. In *VLDB'96*

Dynamic Itemset Counting (DIC): Reduce Number of Scans



Itemset lattice

- Once both A and D are determined frequent, the counting of AD begins
- Once all length-2 subsets of BCD are determined frequent, the counting of BCD begins



Apriori

DIC

S. Brin R. Motwani, J. Ullman, and S. Tsur. Dynamic itemset counting and implication rules for market basket data. *SIGMOD'97*

Basic Idea of DIC

If we consider Apriori in this metaphor, all itemsets must get on at the start of a pass and get off at the end. The 1-itemsets take the first pass, the 2-itemsets take the second pass, and so on (see Figure 1). In DIC, we have the added flexibility of allowing itemsets to get on at any stop as long as they get off at the same stop the next time the train goes around. Therefore, the itemset has "seen" all the transactions in the file. This means that we can start counting an itemset as soon as we suspect it may be necessary to count it instead of waiting until the end of the previous pass.

For example, if we are mining 40,000 transactions and $M = 10,000$, we will count all the 1-itemsets in the first 40,000 transactions we will read. However, we will begin counting 2-itemsets after the first 10,000 transactions have been read. We will begin counting 3-itemsets after 20,000 transactions. For now, we assume there are no 4-itemsets we need to count. Once we get to the end of the file, we will stop counting the 1-itemsets and go back to the start of the file to count the 2 and 3-itemsets. After the first 10,000 transactions, we will finish counting the 2-itemsets and after 20,000 transactions, we will finish counting the 3-itemsets. In total, we have made 1.5 passes over the data instead of the 3 passes a level-wise algorithm would make.[3]

Presentation Outline

- Basic concepts
  - Frequent itemssets, Closed Itemsets, and Association Rules
- Frequent itemset mining methods
  - Apriori Algorithm
    - Generating Association Rules from Frequent Itemsets
    - Improving the Efficiency of Apriori
  - **A Pattern-Growth Approach**
  - Mining Frequent Itemsets Using the Vertical Data Format
  - Mining Closed and Max Patterns
- Which Patterns are interesting? Pattern Evaluation Methods
- Summary

Mining Frequent Patterns without Candidate Generation

Proc. 2000 ACM-SIGMOD Int. Conf. on Management of Data (SIGMOD'00), Dallas, TX, May 2000.

Jiawei Han, Jian Pei, and Yiwen Yin

(FP Tree method)

Pattern-Growth Approach: Mining 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, 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

Main Concepts of FP-growth

- F-list
- FP tree
- Header table
- Conditional pattern bases
- Conditional FP-trees

Construct FP-tree from a Transaction Database

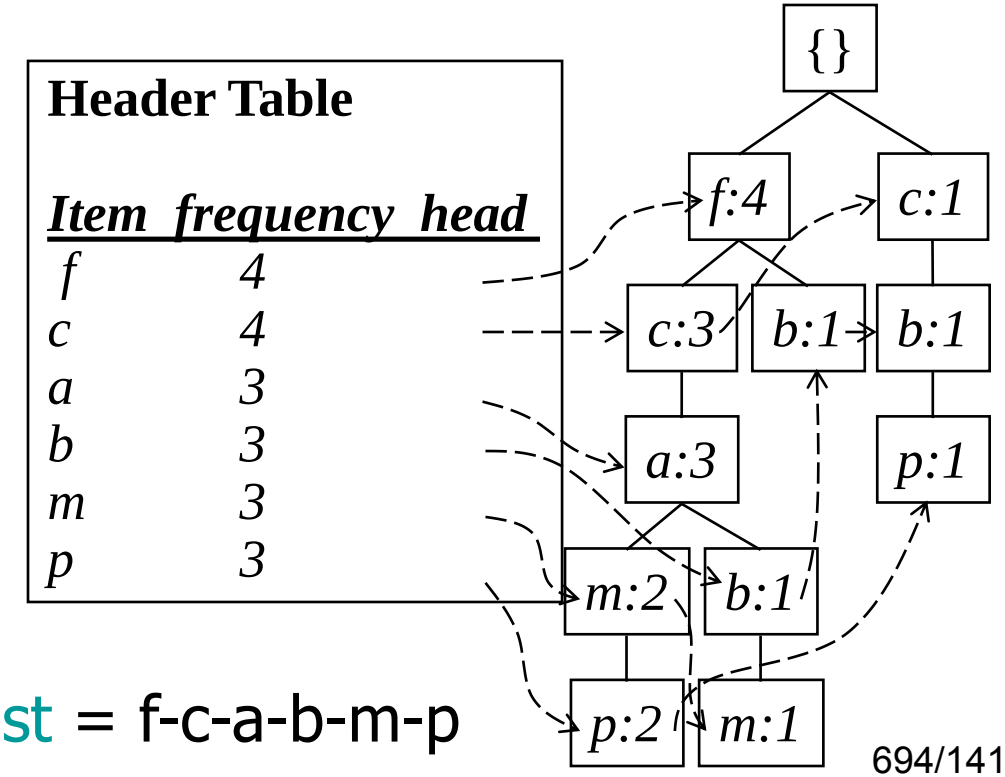
TID	Items bought	(ordered) frequent items
100	{f, a, c, d, g, i, m, p}	{f, c, a, m, p}
200	{a, b, c, f, l, m, o}	{f, c, a, b, m}
300	{b, f, h, j, o, w}	{f, b}
400	{b, c, k, s, p}	{c, b, p}
500	{a, f, c, e, l, p, m, n}	{f, c, a, m, p}

min_support = 3

1. Scan DB once, find frequent 1-itemset (single item pattern)
2. Sort frequent items in frequency descending order, f-list
3. Scan DB again, construct FP-tree

Header Table

Item	frequency	head
f	4	
c	4	
a	3	
b	3	
m	3	
p	3	



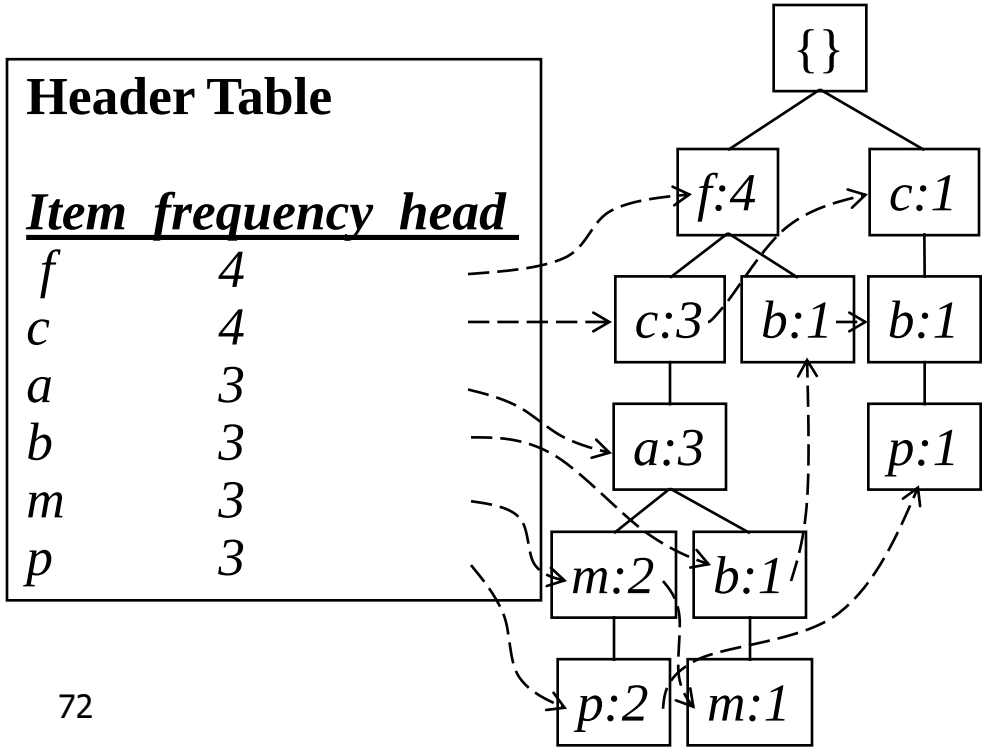
F-list = f-c-a-b-m-p

Partition Patterns and Databases

- Frequent patterns can be partitioned into subsets according to f-list
  - F-list = f-c-a-b-m-p
  - Patterns containing p
  - Patterns having m but no p
  - …
  - Patterns having c but no a nor b, m, p
  - Pattern f
- Completeness and non-redundency

Find Patterns Having P From P-conditional Database

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

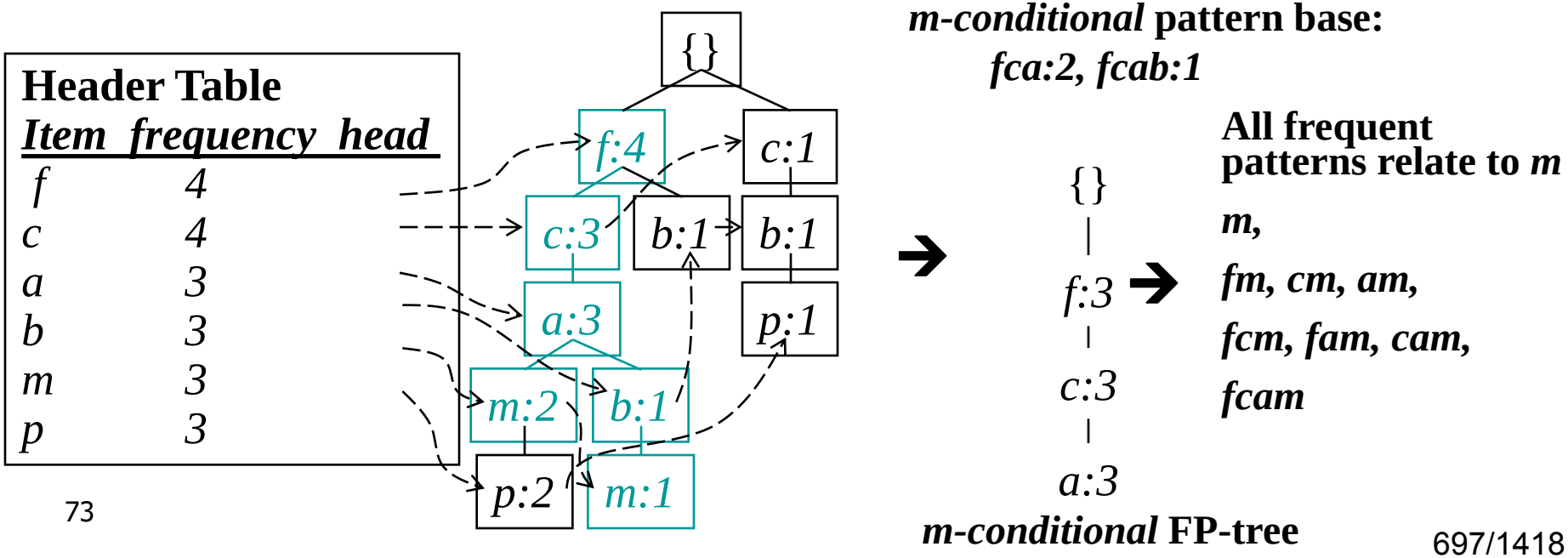


***Conditional* pattern bases**

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

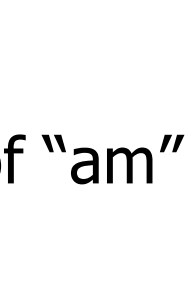
From Conditional Pattern-bases to Conditional FP-trees

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



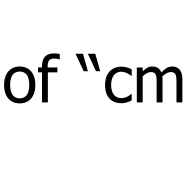
Recursion: Mining Each Conditional FP-tree

Cond. pattern base of "am": (fc:3)

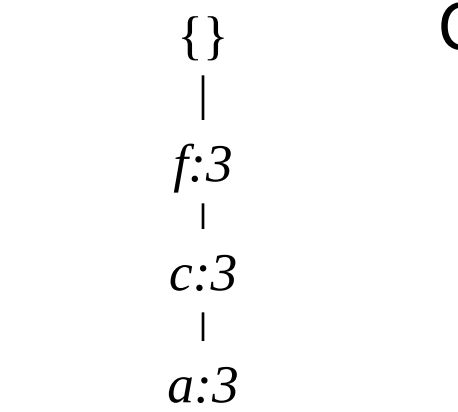


***am-conditional* FP-tree**

Cond. pattern base of "cm": (f:3)



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



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

Cond. pattern base of "cam": (f:3)



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

Presentation Outline

- Basic concepts
  - Frequent itemssets, Closed Itemsets, and Association Rules
- Frequent itemset mining methods
  - Apriori Algorithm
    - Generating Association Rules from Frequent Itemsets
    - Improving the Efficiency of Apriori
  - A Pattern-Growth Approach
  - **Mining Frequent Itemsets Using the Vertical Data Format**
  - Mining Closed and Max Patterns
- Which Patterns are interesting? Pattern Evaluation Methods
- Summary

IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, VOL. 12, NO. 3, MAY/JUNE 2000

Scalable Algorithms for Association Mining

Mohammed J. Zaki, *Member*, *IEEE*

ECLAT: Mining by Exploring Vertical Data Format

(ECLAT: Equivalence CLAss Transformation)

- Transform the horizontally formatted data into the vertical format by scanning the data set once.
- The support count of an itemset is simply the length of the TID set of the itemset.
  - Starting with $k = 1$, the frequent k-itemsets can be used to construct the candidate $(k+1)$-itemsets based on the Apriori property.

Example:

- The minimum support count is 2.

- 10 intersections performed in total, which lead to eight nonempty 2-itemsets.
- Notice that because the itemsets {I1, I4} and {I3, I5} each contain only one transaction, they do not belong to the set of frequent 2-itemsets.
- Based on the Apriori property, a given 3-itemset is a candidate 3-itemset only if every one of its 2-itemset subsets is frequent. The candidate generation process here will generate only two 3-itemsets: {I1, I2, I3} and {I1, I2, I5}.
- By intersecting the TID sets of any two corresponding 2-itemsets of these candidate 3-itemsets, we can extract three itemsets.
- There are only two frequent 3-itemsets: {I1, I2, I3: 2} and {I1, I2, I5: 2}.

Table 6.3 The Vertical Data Format of the Transaction Data Set *D* of Table 6.1

itemset	*TID_set*
I1	{T100, T400, T500, T700, T800, T900}
I2	{T100, T200, T300, T400, T600, T800, T900}
I3	{T300, T500, T600, T700, T800, T900}
I4	{T200, T400}
I5	{T100, T800}

Table 6.4 2-Itemsets in Vertical Data Format

itemset	*TID_set*
{I1, I2}	{T100, T400, T800, T900}
{I1, I3}	{T500, T700, T800, T900}
{I1, I4}	{T400}
{I1, I5}	{T100, T800}
{I2, I3}	{T300, T600, T800, T900}
{I2, I4}	{T200, T400}
{I2, I5}	{T100, T800}
{I3, I5}	{T800}

Table 6.5 3-Itemsets in Vertical Data Format

itemset	*TID_set*
{I1, I2, I3}	{T800, T900}
{I1, I2, I5}	{T100, T800}

About ECLAT

- Advantages
  - Exploits apriori property
  - Scan the database only once
- Negative
  - the TID sets can be quite long, taking substantial memory space as well as computation time for intersecting the long sets.
- Notion of Diffset is proposed
  - Keep track of only Diffset
  - keeps track of only the differences of the TID sets of a *(k*-1*)*-itemset and a corresponding *k*-itemset.
    - For instance, in Example 6.6 we have {I1}={T100, T400, T500, T700, T800, T900} and {I1, I2}= {T100, T400, T800, T900}. The *diffset* between the two is *diffset*({I1, I2}, {I1}})= {T500, T700}.

Presentation Outline

- Basic concepts
  - Frequent itemssets, Closed Itemsets, and Association Rules
- Frequent itemset mining methods
  - Apriori Algorithm
    - Generating Association Rules from Frequent Itemsets
    - Improving the Efficiency of Apriori
  - A Pattern-Growth Approach
  - Mining Frequent Itemsets Using the Vertical Data Format
  - **Mining Closed and Max Patterns**
- Which Patterns are interesting? Pattern Evaluation Methods
- Summary

Basic Idea

- Naïve approach
  - Mine complete set of frequent item sets
  - Filter each frequent itemset which is a subset of another frequent itemset or caries the same support of the existing frequent itemset.
  - High complexity
- Recommended methodology
  - Search for closed itemsets directly during the mining process.
  - Prune the search space as soon as we find the closed itemsets.
- List of papers
  - J. Pei, J. Han & R. Mao. “CLOSET: An Efficient Algorithm for Mining Frequent Closed Itemsets", DMKD’00.
  - R. Bayardo. Efficiently mining long patterns from databases. *SIGMOD’98*

Presentation Outline

- Basic concepts
  - Frequent itemssets, Closed Itemsets, and Association Rules
- Frequent itemset mining methods
  - Apriori Algorithm
    - Generating Association Rules from Frequent Itemsets
    - Improving the Efficiency of Apriori
  - A Pattern-Growth Approach
  - Mining Frequent Itemsets Using the Vertical Data Format
  - Mining Closed and Max Patterns
- **Which Patterns are interesting? Pattern Evaluation Methods**
- Summary

About Interesting patterns

- Association rule mining algorithms employ a support-confidence framework.
- Support-confidence
  - Extracts interesting rules
  - However, the framework also extracts uninteresting rules.
    - Low support and long patterns
- Support-confidence framework can supplemented with additional measures.

Strong Rules always may not be interesting

- A misleading association rule
- 10,000 transactions, 6000 include games, 7500 include videos, 4000 include games and videos
  - Buys(X, computer game) →buys(X, videos)
  - Support=40%, confidence=66%
  - Strong rule
- This rule is uninteresting, because the probability of purchasing videos is 75%
- Normally, computer games and videos are negatively correlated.
  - Purchase of one decreases the chances of purchase of the other.
- So support/confidence framework may be misleading.
- We need measures to measure real strength of the correlation and implication.

Lift: A correlation rule

- A→B (support, confidence, correlation)
- There are several correlation measures. One can choose any rule
- Lift is the simple correlation measure
- Lift(A,B)=P(AUB)/P(A)P(B)
  - < 1 negatively correlated. It means occurrence of one will lead to absence of the other
  - >1 Positively correlated. It means occurrence of one will lead to presence of other
  - =1 A and B are independent. There is no relationship
- Chi-square test is another correlation measure.
  - It tests the hypothesis that both are independent

Example

Table 6.6 2 × 2 Contingency Table Summarizing the Transactions with Respect to Game and Video Purchases

	game	$\overline{game}$	Σ_{row}
video	4000	3500	7500
$\overline{video}$	2000	500	2500
Σ_{col}	6000	4000	10,000

- Lift (game → videos)=0.40/0.60*0.75=0.89
- = <1 negatively correlated.
- Numerator: likelihood of purchasing both
- Denominator: likelihood of independent purchases.

Other interestingness measures

- All confidence
- Max confidence
- Kulczynski
- Cosine
- Each measure is only influenced by the supports of A,B, and AUB, but not by the total number of transactions.
- The value ranges between 0 to 1

Given two itemsets, A and B, the **all_confidence** measure of A and B is defined as

$$all_conf(A, B) = \frac{sup(A \cup B)}{max\{sup(A), sup(B)\}} = min\{P(A|B), P(B|A)\}, \tag{6.9}$$

where $max\{sup(A), sup(B)\}$ is the maximum support of the itemsets A and B. Thus, $all_conf(A, B)$ is also the minimum confidence of the two association rules related to A and B, namely, "$A \Rightarrow B$" and "$B \Rightarrow A$."

Given two itemsets, A and B, the **max_confidence** measure of A and B is defined as

$$max_conf(A, B) = max\{P(A \mid B), P(B \mid A)\}. \tag{6.10}$$

The *max_conf* measure is the maximum confidence of the two association rules, "$A \Rightarrow B$" and "$B \Rightarrow A$."

Given two itemsets, A and B, the **Kulczynski** measure of A and B (abbreviated as **Kulc**) is defined as

$$Kulc(A, B) = \frac{1}{2}(P(A|B) + P(B|A)). \tag{6.11}$$

It was proposed in 1927 by Polish mathematician S. Kulczynski. It can be viewed as an average of two confidence measures. That is, it is the average of two conditional probabilities: the probability of itemset B given itemset A, and the probability of itemset A given itemset B.

Finally, given two itemsets, A and B, the **cosine** measure of A and B is defined as

$$\begin{aligned} cosine(A, B) &= \frac{P(A \cup B)}{\sqrt{P(A) \times P(B)}} = \frac{sup(A \cup B)}{\sqrt{sup(A) \times sup(B)}} \\ &= \sqrt{P(A|B) \times P(B|A)}. \end{aligned} \tag{6.12}$$

Which one is better?

- We introduced six measures? Which one is best in assessing the pattern relationship?

Table 6.8 2 × 2 Contingency Table for Two Items

	milk	$\overline{milk}$	Σ_{row}
coffee	mc	$\overline{m}c$	c
$\overline{coffee}$	$m\overline{c}$	$\overline{mc}$	$\overline{c}$
Σ_{col}	m	$\overline{m}$	Σ

Table 6.9 Comparison of Six Pattern Evaluation Measures Using Contingency Tables for a Variety of Data Sets

Data Set	*mc*	$\overline{m}c$	$m\overline{c}$	$\overline{mc}$	χ^2	*lift*	*all_conf.*	*max_conf.*	*Kulc.*	*cosine*
D_1	10,000	1000	1000	100,000	90557	9.26	0.91	0.91	0.91	0.91
D_2	10,000	1000	1000	100	0	1	0.91	0.91	0.91	0.91
D_3	100	1000	1000	100,000	670	8.44	0.09	0.09	0.09	0.09
D_4	1000	1000	1000	100,000	24740	25.75	0.5	0.5	0.5	0.5
D_5	1000	100	10,000	100,000	8173	9.18	0.09	0.91	0.5	0.29
D_6	1000	10	100,000	100,000	965	1.97	0.01	0.99	0.5	0.10

- M and C
  - are positively correlated in D1 and D2
  - Are negatively associated in D3
  - Neutral in D4
- Lift and chi-square produce different results for D1 and D2 because of $\overline{mc}$
- In real world applications $\overline{mc}$ is huge and unstable.
- So a good measure should not be effected by $\overline{mc}$
- D3: other measures are OK but lift and chisquare contradict
- D4, lift and chisquare show high correlation, others indicate neutral

- Reason for bad performance of lift and chi square
  - Null transactions: A null transaction does not contain the items being examined
  - The measure is strongly influenced by the transactions which do not contain m and c.
- A measure should be independent of null transactions

- Null-invariant measure: A measure is null-invariant if it value is free from the influence of null transactions

Which Null-Invariant Measure Is Better?

- IR (Imbalance Ratio): to assess the imbalance of two itemsets A and B in rule implications

$$IR(A,B) = \frac{|sup(A) - sup(B)|}{sup(A) + sup(B) - sup(A \cup B)}$$

- Suggestion: Kulczynski and Imbalance Ratio (IR) together present a clear picture for all the three datasets D_4 through D_6
  - D_4 is balanced & neutral
  - D_5 is imbalanced & neutral
  - D_6 is very imbalanced & neutral

Data	mc	$\overline{m}c$	$m\overline{c}$	$\overline{mc}$	*all_conf.*	*max_conf.*	*Kulc.*	*cosine*	IR
D_1	10,000	1,000	1,000	100,000	0.91	0.91	0.91	0.91	0.0
D_2	10,000	1,000	1,000	100	0.91	0.91	0.91	0.91	0.0
D_3	100	1,000	1,000	100,000	0.09	0.09	0.09	0.09	0.0
D_4	1,000	1,000	1,000	100,000	0.5	0.5	0.5	0.5	0.0
D_5	1,000	100	10,000	100,000	0.09	0.91	0.5	0.29	0.89
D_6	1,000	10	100,000	100,000	0.01	0.99	0.5	0.10	0.99

Presentation Outline

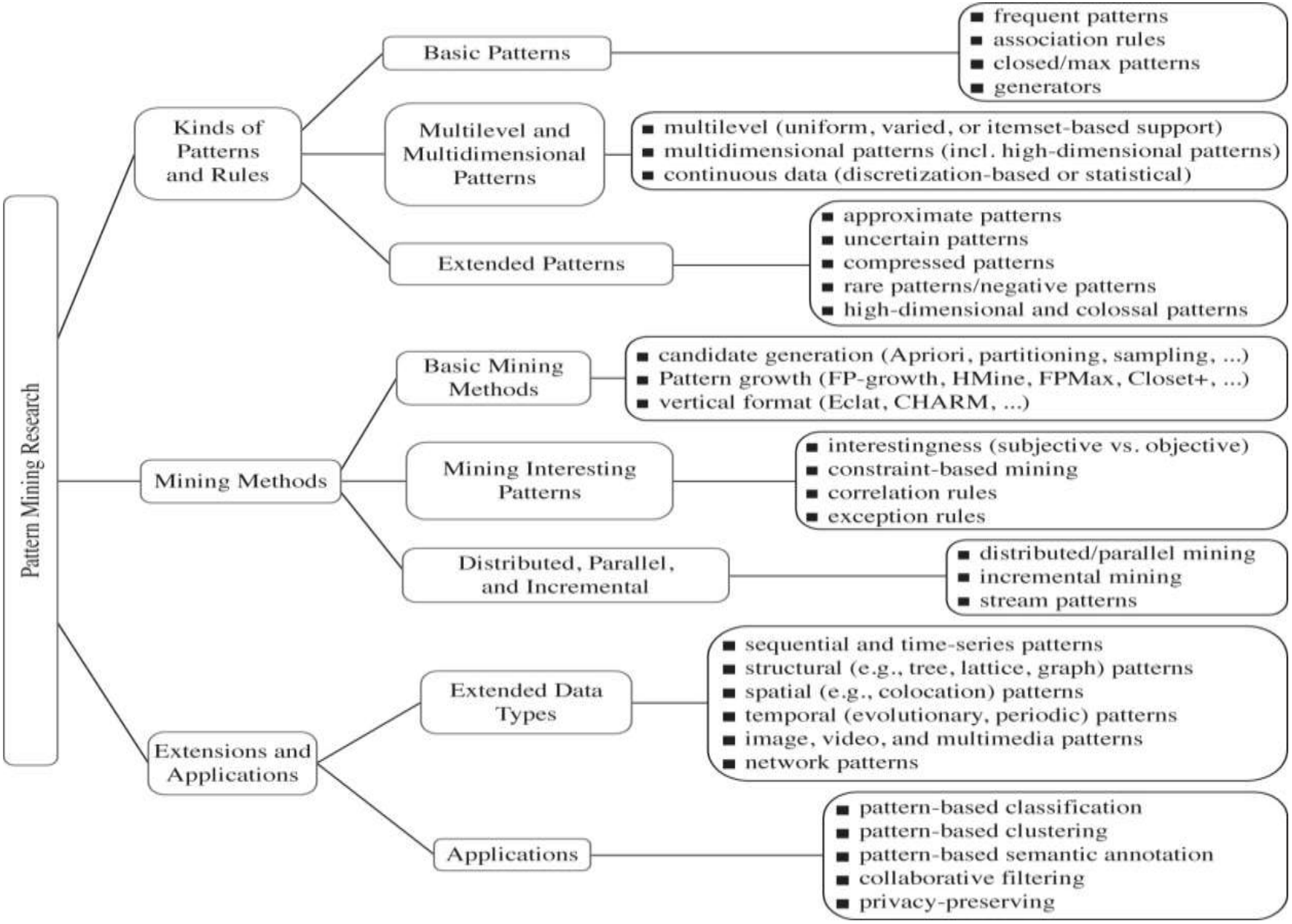
- Basic concepts
  - Frequent itemssets, Closed Itemsets, and Association Rules
- Frequent itemset mining methods
  - Apriori Algorithm
    - Generating Association Rules from Frequent Itemsets
    - **Improving the Efficiency of Apriori**
  - A Pattern-Growth Approach
  - Mining Frequent Itemsets Using the Vertical Data Format
  - Mining Closed and Max Patterns
- Which Patterns are interesting? Pattern Evaluation Methods
- Summary

Summary

- Frequent patterns: association between items
  - Application: DSSs, Market basket analysis
- Exponential problem
- Apriori algorithm
- Frequent pattern growth algorithm
- Eclat: extracting with vertical format
- Null-invariant measures

Advanced Pattern Mining

P. Krishna Reddy, IIIT Hyderabad



Pattern Mining: A Road Map

Outline

- Pattern Mining: A Road Map
- **Coverage Pattern Mining**
- Multilevel Pattern Mining
- Quantitative Pattern Mining
- Multidimensional mining
- Diverse Frequent Pattern Mining
- High Utility Pattern mining
- Mining Rare Patterns and Negative Patterns
- Constraint-Based Frequent Pattern Mining
- Summary

Discovering Coverage Patterns for Banner Advertisement Placement (Class #13)

P. Gowtham Srinivas, P. Krishna Reddy, S. Bhargav,

R. Uday Kiran, and D. Satheesh Kumar

International Institute of Information Technology

Hyderabad (IIIT-H) , India

Presented by P.Krishna Reddy

E-mail: pkreddy@iiit.ac.in

Outline

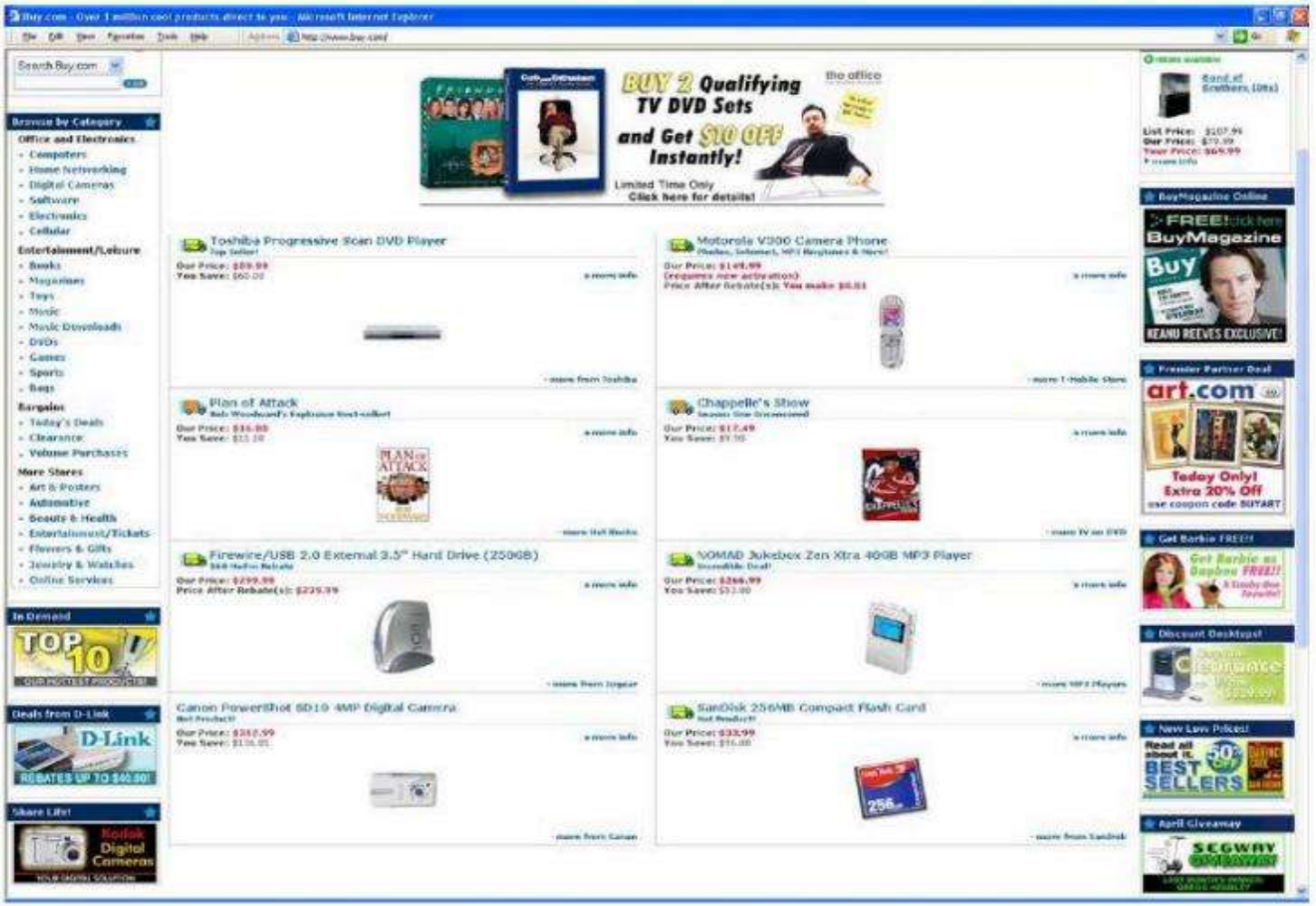
- **Introduction**
- Coverage patterns
- Mining Coverage Patterns
- Experimental Results
- Conclusions and future work

Introduction

- **Online advertising** is a form of promotion that uses the Internet to deliver marketing messages to attract customers.
  - provides sustainability for the e-commerce enterprises
  - a key factor in the growth of internet economy.
- There are three modes of online advertising:
  - Banner Advertising.
    - hyperlinked to the advertisers primary page or one with more information about the specific product or service advertised.
  - Contextual advertising
    - Relevant advertisements are assigned to a web page based on the content of the web page.
  - Sponsored search advertising
    - Advertisements are placed alongside with the search results.

About Banner Advertising

- Banner advertising is one of the dominant modes of online advertising.
- In this paper, we proposed a notion of coverage patterns to help to place the banner advertisements in a better manner.
- Three entities are involved in banner advertising: advertiser, publisher and visitor.
  - An advertiser is interested in endorsing products through banner advertisements.
  - A publisher manages a web site or an advertisement network that sells banner advertisement space.
  - Finally, a visitor visits the web pages of a web site which contains banners.



Banner advertisements on www.buy.com

Problem Description

- Goal of advertiser
  - spreading advertisement to a certain percentage of people visiting a web site.
- Goal of the publisher
  - efficiently sell the advertising space available in the web pages of a web site and meeting the demands of multiple advertisers.
- Opportunity:
  - For a given web site and period, one can analyse the visitors' behaviour by processing the transactions generated based on click stream dataset and identify the sets of web pages that cover a given percentage of visitors' population.
- Research Issue:
  - Investigate the approaches for discovering the sets of web pages which can cover a given percentage of visitors' population by analyzing the click stream transactions

Contributions

- We have proposed methodology to discover coverage patterns from transactional databases.
  - Given a set of data items, a coverage pattern is a set of non-overlapping data items covered by a certain percentage of transactions.
  - An Apriori-like algorithm called CMine is proposed for mining coverage patterns.

Related Work

- Coverage patterns
  - The notion of coverage is being used for solving the set cover problem [2] in set theory and node cover problem [3] in graphs respectively.
  - In [4], the notion of coverage and overlap is used to examine the creation of a tag cloud for exploring and understanding a set of objects.
  - In [5], the notion of coverage and overlap is used to solve the problem of topical query decomposition.
- Online Advertisement
  - Most of the research work on online advertisement has been focused on auction models [7], keyword or phrase identification based on user queries [8], contextual advertising [9] and allocation and scheduling of advertisements [10].
- To our knowledge, not much amount of research work has been carried out on improving the options offered by the publisher to the advertisers

Advantages

- We have proposed a different kind of knowledge patterns.
- The proposed patterns can be employed in improving the performance of several applications such as banner advertisements.
  - help the advertiser by making his advertisement visible to a certain percentage of web site visitors.
  - Ensuring the publisher to meet the demands of multiple advertisers by considering several groups of potential pages.

Outline

- Introduction
- **Coverage patterns**
- Mining Coverage Patterns
- Experimental Results
- Conclusions and future work

Coverage Patterns and Banner Advertisement

- We identify the issue of banner advertisement placement as one of the potential application of coverage patterns.
  - For a given e-commerce web site, transactions generated from click stream dataset can be used to identify the sets of web pages that cover a given percentage of visitors' population.
  - We consider transactions generated from click stream data of a web site. However, the model can be extended to any transactional data set.

Basic Terminology

- Let W = {w1,w2, · · · , wn} be a set of identifiers of web pages and D be a set of transactions, where each transaction T is a set of web pages such that $T \subseteq W$.
- Associated with each transaction is a unique transactional identifier called TID.
- A set of web pages in W i.e., X = {wp, ….. ,wq, wr},

  $1 \leq p \leq q \leq r \leq n$, is called a pattern.
- A pattern containing ‘k’ number of web pages is called a k-pattern. In other words, the length of k-pattern is k.
- The percentage of transactions in D that contain the web page $wi \in W$ is known as the “relative frequency of a web page $wi \in W$” and denoted as RF(wi).
- Let $|T^{wi}|$ indicates the total number of transactions that contain wi. The relative frequency of wi is denoted as RF(wi). That is, $RF(wi) = (|T^{wi}|)/|D|$.

Proposed Model…

TRANSACTIONAL DATABASE

Table 1.

TID	Web pages	TID	Web pages
1	a, b, c	6	b, d
2	a, c, e	7	b, d
3	a, c, e	8	b, e
4	a, c, d	9	b, e
5	b, d, f	10	a, b

Item	Relative Frequency (RF)
a	0.5
b	0.7
c	0.4
d	0.4
e	0.4
f	0.1

- {a, b} is a pattern. Since there are two web pages in this pattern it is a 2-pattern.

Frequent Page

- Note that from the advertisement point of view the pages that are visited by more number of users are interesting. We capture this aspect with the notion of frequent page.

- The frequent web pages are web pages which have relative frequency no less than the user-specified threshold value, called minimum relative frequency(*minRF*) . That is, $RF(w_i) \geq minRF.$

- Example 2: Continuing with the example, the relative frequency of 'a' i.e., $RF(a) = |T^a|/|D| = 5/10 = 0.5$. If the user-specified minRF = 0.5, then 'a' is called a frequent web page because RF(a) ≥ minRF.

Coverage Set

- The coverage set is a set of transactions.
  - Notion: How many users visit at least one web page in the set.
- Coverage set of a pattern X = {w_p, ….. ,w_q, w_r}, $1 \leq p \leq q \leq r \leq n$:
  - The set of distinct TIDs containing at least one web page of X is called the coverage set of pattern X and is denoted as CSet(X). Therefore, $CSet(X) = \{T^{w_p} \cup \cdots \cup T^{w_q} \cup T^{w_r}\}$.

TID	Web pages	TID	Web pages
1	a, b, c	6	b, d
2	a, c, e	7	b, d
3	a, c, e	8	b, e
4	a, c, d	9	b, e
5	b, d, f	10	a, b

Example: The set of tids containing the web page `a' i.e., T^a ={1,2,3,4,10}. Similarly, T^b={1,5,6,7,8,9,10}.

The coverage set of {a,b} i.e., CSet({a,b}) ={1,2,3,4,5,6,7,8,9,10} .

Note: In case of frequent patterns, the support of {a,b} ={1,10}=2

Coverage Support

- A pattern will be interesting if its coverage set contains more than a threshold number of transactions. This aspect is captured through the notion of coverage support.
- ***Coverage-support of a pattern X*** *: The ratio* of size of coverage set of *X* to the transactional database size is called the coverage-support of pattern *X and denoted as CS(X).*
  - *CS(X) = |CSet(X)|/ |D|.*
- For a pattern *X, CS(X)* $\in$ *[0, 1].*
  - If CS(X) = 0, no single web page of X has appeared in the entire transactional database.
  - If CS(X) = 1, every transaction in T contains at least one web page wj $\in$ X.
- Example: Coverage support of {a,b} i.e., CS({a,b}) = |CSet({a,b})|/|D| = 10/10=1.

Overlap ratio

- Adding other web pages to pattern X may not increase the coverage support, significantly.
  - Due to co-occurrence, there is an overlap of coverage set of X and coverage set of new single web page.
  - Such a pattern can be uninteresting to the advertiser because, the advertisement will be displayed to the same user multiple times.
- Example: T^a={1,2,3,4,10}. T^b={1,5,6,7,8,9,10}, and T^c={1,2,3,4}. CS(a)=0.5, CS(b)=0.7, CS(c)=0.4.
  - We can note that CS (a) and CS(a,c)=5/10=0.5. However, this pattern is uninteresting as the pattern `c' has not increased coverage-support of the pattern `a'.
  - We can note that CS(a,b)=1.As compared to {a,c}, the pattern {a,b} is interesting because, the web page `b' has increased the coverage support of a pattern.

Overlap Ratio: first definition

Definition 3 *(Overlap ratio (OR) of a pattern Y.) The OR of a pattern $Y = X \cup \{w_r\}$, where X can be empty or $X = \{w_p, \cdots, w_q\}$, $1 \leq p, q, r \leq n$ is defined as the ratio of the number of transactions common in $CSet(X)$ and $CSet(\{w_r\})$ to $CSet(\{w_r\})$, i.e., $OR(Y) = \frac{|(Cset(X)) \cap (Cset\{w_r\})|}{|Cset\{w_r\}|}$.*

…Overlap Ratio

- For a pattern X, OR(X) ∈ [0, 1]. If OR(X) = 0, there exists no common transactions between $X - \{w_r\}$ and $\{w_r\}$.
- If OR(X) = 1, then w_r has occurred in the transactions where at least one web page $w_j \in (X - \{w_r\})$ has occurred.
- Continuing with the example,
  - The OR({a,b}) = |CSet(b) ∩ CSet(a)|/|CSet(a)| = 2/5 = 0.4.
- Note that a coverage pattern is interesting if it has high coverage support and low overlap ratio.
  - As a result an advertisement is exposed to more number of users by reducing repetitive display of the advertisement. The definition of coverage pattern is as follows

Coverage Pattern

- ***Coverage pattern X*** : A pattern X ={w_p, …. , w_q, w_r}, where 1 ≤ p ≤ q ≤ r ≤ n, is said to be a coverage pattern if CS(x) ≥ minCS and OR(X) ≤ maxOR and RF(w_i) ≥ minRF ∀ w_i ∈ X.

  A coverage pattern *X* having CS(X) = a% and OR(X) = b% is expressed as

  X [CS = a%, OR = b%]

- If minRF=0.4, minCS=0.7 and maxOR=0.5, then the pattern {a,b} is a coverage pattern. It is because RF(a) >= minRF, RF(b) >= minRF, CS({a,b}) >= minCS and OR({a,b}) <= maxOR. This pattern is written as follows:
  - {a,b} [CS=1 (=100%), OR= 0.4 (=40%)]

Problem Statement

- Given a transactional database *D*, set of web pages *W* and user-specified minimum relatively frequency (*minRF)*, minimum coverage support *(minCS)* and maximum overlap ratio (*maxOR)*, discover complete set of coverage patterns such that
  i. If X is a coverage 1-pattern (i.e., k = 1), then $RF(w_i) \geq minRF$, $w_i \in X$.
  ii. Otherwise (i.e., when k > 1), each coverage pattern X must have $CS(X) \geq minCS$, $OR(X) \leq maxOR$ and $RF(w_i) \geq minRF$, where $w_i \in X$.

Outline

- Introduction
- Coverage patterns
- **Mining Coverage Patterns**
- Experimental Results
- Conclusions and future work

Extracting Coverage Patterns

- Naïve approach
  - Extract all possible patterns (leads to combinatorial explosion)
  - Each pattern in CP is added to the coverage pattern set if it satisfies minCS; minRF and maxOR constraints.
- Opportunity
  - The search space can be reduced if the coverage pattern satisfies downward closure property on either coverage support or overlap ratio.

Extracting Coverage Patterns

- Coverage support does not satisfy downward closure property. That is, although a pattern satisfies minCS, it is not necessary that all its non-empty subsets will also satisfy minCS value.
  - Consider the patterns {a}, {e} and {a,e}. The coverage supports of these patterns are 0.5, 0.4 and 0.7, respectively. If the user-specified minCS=0.7, then the pattern {a,e} satisfies minCS value. However, its non-empty subsets do not satisfy minCS value.
- The parameter overlap ratio also does not satisfy downward closure property if a pattern is considered as an unordered set of web pages.
- The Overlap ratio satisfies downward closure property if a pattern is an ordered set, where web pages are sorted in descending order of their frequencies. This property is known as the sorted closure property.

Coverage patterns and sorted closure property Extracting the complete set of CPs for a given $minRF$, $minCS$ and $maxOR$ is a challenging task, because the measure CS does not satisfy the *downward closure property*. Although a pattern satisfies $minCS$, it is not necessary that all its non-empty subsets will also satisfy $minCS$ value. For example, consider the patterns $\{a\}$, $\{e\}$ and $\{a, e\}$ from Table 1. The coverage-supports of these patterns are 0.5, 0.4 and 0.7, respectively. If the user specifies $minCS = 0.7$, the pattern $\{a, e\}$ satisfies $minCS$ value. However, the patterns $\{a\}$ and $\{e\}$ do not satisfy $minCS$ value. It can also be noted that the preceding definition of OR measure also does not satisfy *downward closure property*. From Table 1, $OR(\{a, c, b\}) = \frac{|CSet(\{a,c\}) \cap CSet(\{b\})|}{|CSet(\{b\})|} = \frac{2}{7}$ and $OR(\{a, c\}) = \frac{|CSet(\{a\}) \cap CSet(\{c\}|)}{|CSet(\{c\})|} = \frac{4}{4} = 1$. If the user specifies $maxOR = \frac{2}{7}$, $OR(\{a, c, b\}) \leq maxOR$ value. However, $OR(\{a, c\}) \not\leq maxOR$ value.

It was observed that the OR satisfies *downward closure property*, if the items are ordered in descending order of their frequencies. This property is called *sorted-closure property* [13] which is given in Property 1 and the correctness is shown in Lemma 2.

Property 1 *Sorted Closure Property: Let* $X = \{w_p, \cdots w_q, w_r\}$ *be a pattern such that* $RF(w_p) \geq \cdots RF(w_q) \geq RF(w_r)$ *and* $1 \leq p, q, r \leq n$. *If* $OR(X) \leq maxOR$, *then all its non-empty subsets containing* w_r *and having size* $k \geq 2$ *will also have* overlap ratio *less than or equal to* $maxOR$.

Lemma 1 *If* $X \subset Y$, *then* $CSet(X) \subseteq CSet(Y)$.

Lemma 2 *If* $OR(X) \leq maxOR$, *then* $OR(Y) \leq maxOR$. $\forall\ Y \subset X, w_r \in Y$.

Proof Let w_a, w_b and w_c be the web pages having $RF(w_a) \geq RF(w_b) \geq RF(w_c)$. If $OR(\{w_a\} \cup \{w_c\}) > maxOR$, then $OR((\{w_a\} \cup \{w_b\}) \cup \{w_c\}) > maxOR$ because from Lemma 1, $\frac{|CSet(\{w_a\}) \cap CSet(\{w_c\})|}{|CSet(\{w_c\})|} \leq \frac{|CSet(\{w_a \cup w_b\}) \cap CSet(\{w_c\})|}{|CSet(\{w_c\})|}$.

Overlap ratio: Second Definition

- If $X \subset Y$, then CSet(X) $\subset$= CSet(Y).
- Sorted closure property: Let X = {w_p, ….. ,w_q, w_r}, where $1 \leq p \leq q \leq r \leq n$ be a pattern such that $T^{w_p} >= \cdots >= T^{w_q} >= T^{w_r}$. If OR(X) <= maxOR, all its non-empty subsets containing w_r and having size k>= 2 will also have overlap ratio less than or equal to maxOR.
- Rationale: Let w_a, w_b and w_c be the web pages having $RF(w_a) \geq RF(w_b) \geq RF(w_c)$. If $OR(w_a \cup w_c) > maxOR$, then $OR(\{w_a \cup w_b\} \cup w_c) > maxOR$ because from above property

  $|CSet(w_a) \cap CSet(w_c)| / |CSet(w_c)| <= CSet(\{w_a \cup w_b\}) \cap CSet(w_c)/CSet(w_c)$.
- (Non-overlap pattern X.) A pattern X is said to be *non-overlap* if $OR(X) \leq maxOR$ and $RF(w_i) \geq minRF \ \forall w_i \in X$.

Extracting Coverage Patterns

- Every coverage pattern is a non-overlap pattern
- However, every non-overlap pattern may not be a coverage pattern.
- The sorted closure property of non-overlap patterns is used for minimizing the search space while mining complete set of coverage patterns by designing an algorithm similar to the Apriori algorithm .

Proposed Algorithm

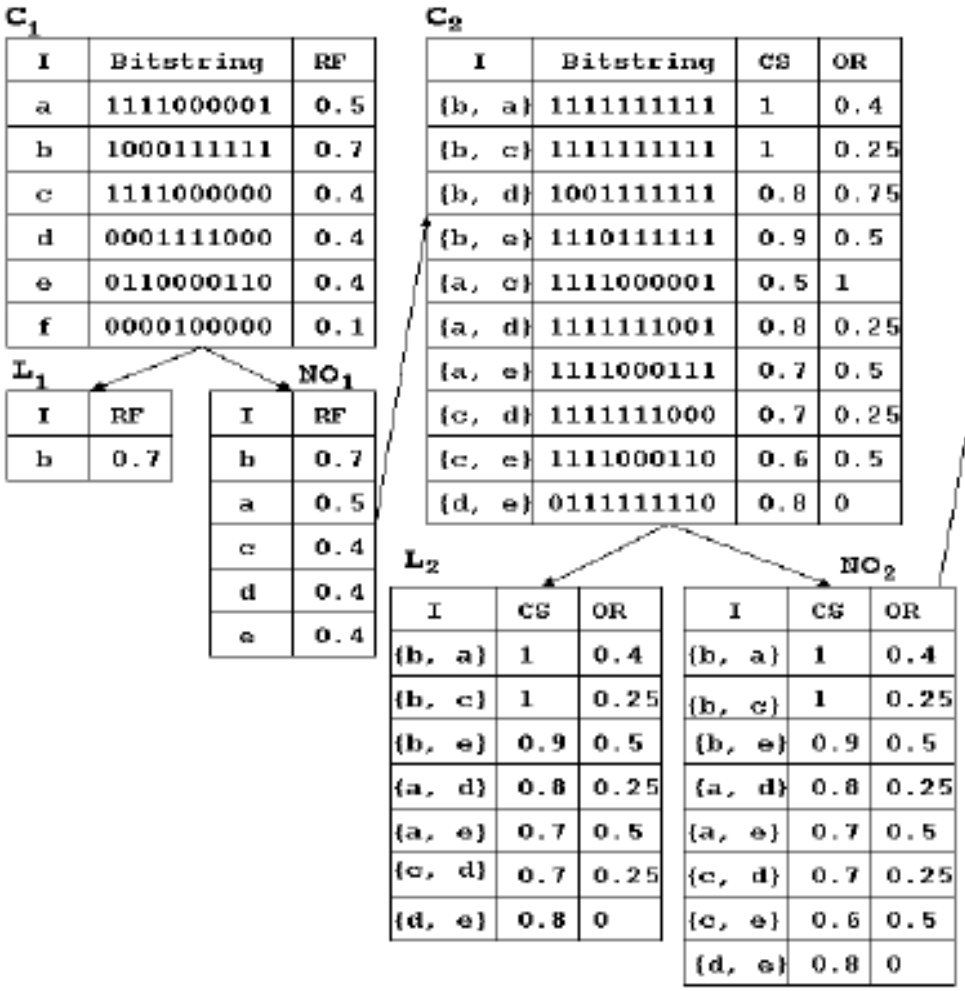
- The proposed algorithm *CMine* employs *level-wise* search to discover the complete set of coverage patterns.
- In *level-wise* search*, k*-patterns are used to explore *(k + 1)*-patterns.
- Notations

F	Set of Frequent items
C_k	Set of Candidate K-patterns
L_k	Set of Coverage K-patterns
NO_k	Set of non-overlap K-patterns

CMine Algorithm

- Step 1: In the first scan of the database, ***CMine*** discovers set of all frequent items (denoted as *F)* and coverage 1-patterns (denoted as L_1). Next, items in NO_1 are sorted in descending order of their frequencies.
- Step 2: Using NO_1 as a seed set, candidate patterns C_2 are generated.
- Step 3: From C_2, the patterns that satisfy *minCS* and *maxOR* are generated as *coverage* 2-patterns, L_2.
- **Simultaneously, all candidate 2-patterns that satisfy *maxOR* are generated as *non-overlap* 2-patterns NO_2.**
- Step 4 : C_3 *is* generated by combining $NO_2 \bowtie NO_2$.
- Step 5 : From C_3, L_3 *and* NO_3 are discovered.
- The above process is repeated until no new coverage pattern is found, or no new candidate pattern can be generated.

CMine Algorithm Example



C_1

I	Bitstring	RF
a	1111000001	0.5
b	1000111111	0.7
c	1111000000	0.4
d	0001111000	0.4
e	0110000110	0.4
f	0000100000	0.1

L_1

I	RF
b	0.7

NO_1

I	RF
b	0.7
a	0.5
c	0.4
d	0.4
e	0.4

C_2

I	Bitstring	CS	OR
{b, a}	1111111111	1	0.4
{b, c}	1111111111	1	0.25
{b, d}	1001111111	0.8	0.75
{b, e}	1110111111	0.9	0.5
{a, c}	1111000001	0.5	1
{a, d}	1111111001	0.8	0.25
{a, e}	1111000111	0.7	0.5
{c, d}	1111111000	0.7	0.25
{c, e}	1111000110	0.6	0.5
{d, e}	0111111110	0.8	0

L_2

I	CS	OR
{b, a}	1	0.4
{b, c}	1	0.25
{b, e}	0.9	0.5
{a, d}	0.8	0.25
{a, e}	0.7	0.5
{c, d}	0.7	0.25
{d, e}	0.8	0

NO_2

I	CS	OR
{b, a}	1	0.4
{b, c}	1	0.25
{b, e}	0.9	0.5
{a, d}	0.8	0.25
{a, e}	0.7	0.5
{c, d}	0.7	0.25
{c, e}	0.6	0.5
{d, e}	0.8	0

C_3

I	Bitstring	CS	OR
{b, a, c}	1111111111	1	1
{b, a, e}	1111111111	1	1
{b, c, e}	1111111111	1	1
{a, d, e}	1111111111	1	0.5
{c, d, e}	1111111110	0.9	0.5

L_3/NO_3

I	CS	OR
{a, d, e}	1	0.5
{c, d, e}	0.9	0.5

Note:

(i){b,a} will be joined with {b,c} and forms a pattern {b,a,c} as RF(a) > RF(c).

(ii) {b,c} can not be joined with {b,a} to form a pattern {b,c,a} as RF(c) < RF(a).

minRF = 0.4, minCS = 0.7, maxOR = 0.5 for database given in Table 1.

Outline

- Introduction
- Coverage patterns
- Mining Coverage Patterns
- **Experimental Results**
- Conclusions and future work

Experimental Results

- For experimental purposes we have chosen two kinds of datasets: (i)real world dataset and (ii)synthetic-dataset.
- The four real world datasets considered for the experiments are described below.
  - Kosarak dataset is a sparse dataset with 9,90,002 number of transactions containing 41,270 distinct items.
  - MSNBC dataset contains data from Internet Information Server (IIS) logs for msnbc.com and news-related portions of msn.com for the entire day of September, 28, 1999 . Requests are at the level of page category. The number of categories are 17 and the number of transactions are 989,818
  - Mushroom dataset is a dense dataset containing 8,124 transactions and 119 distinct items.
  - BMS-POS dataset contains click stream data of a dotcom company . The dataset contains 515,597 number of transactions and 1656 distinct items.
- The synthetic dataset T40I10D100K dataset which is generated by the dataset generator [9]. The dataset contains 100000 transactions and 941 distinct items.
- The CMine algorithm was written in Java and run with Windows XP on a 2.66 GHz machine with 2GB memory.

Usefulness of coverage patterns

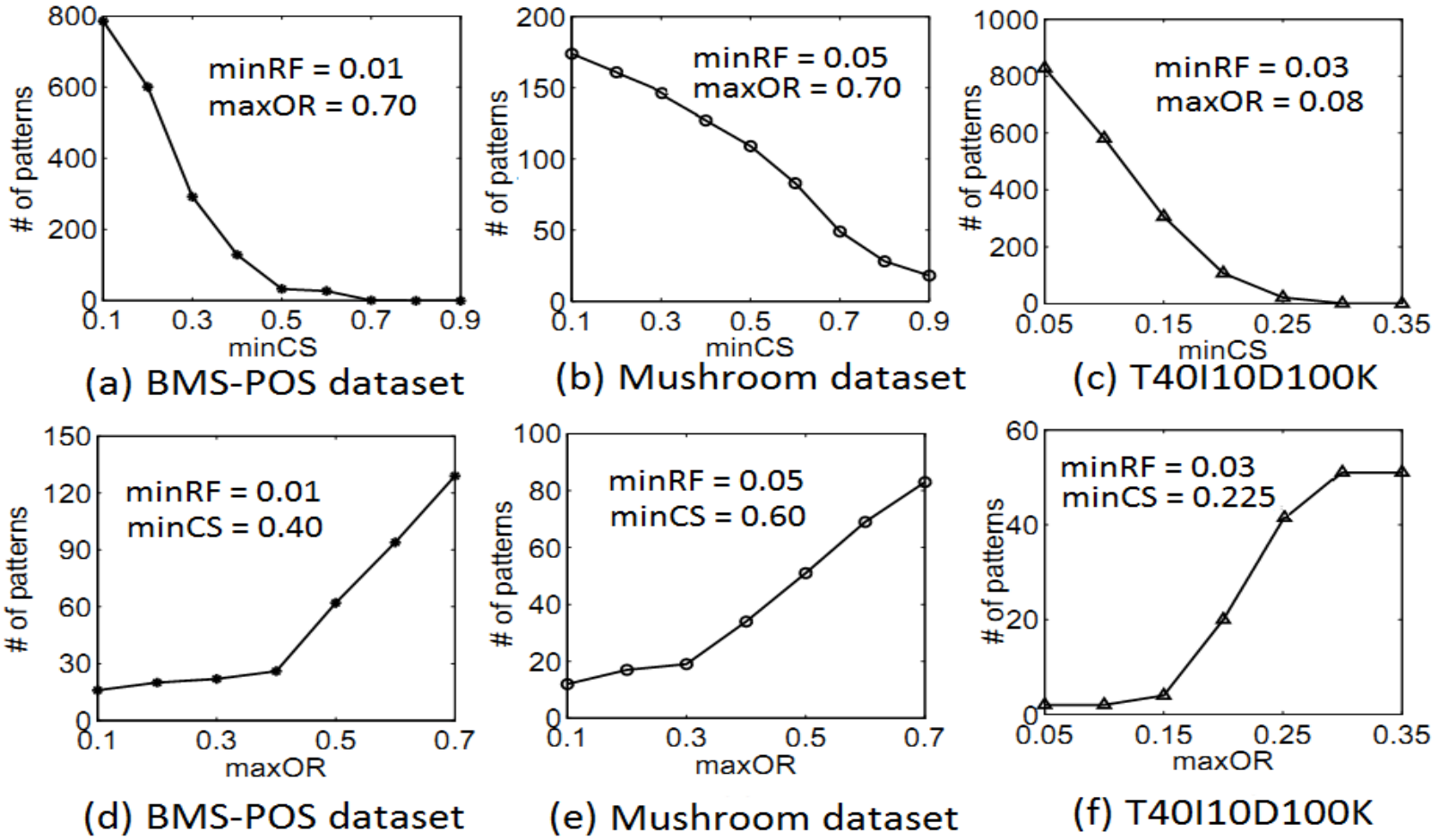
- The names of web page categories involved in MSNBC are
  - "frontpage", "news", "tech", "local", "opinion", "on-air", "misc", "weather", "health", "living", "business", "sports", "summary", "bbs" (bulletin board service), "travel", "msn-news", and "msn-sports".

S.No	Coverage Pattern	CS
1	{local, misc, front page}	0.42
2	{news, health, front page}	0.43
3	{tech, opinion, front page}	0.41
4	{on-air, news, misc}	0.40
5	{tech, weather, on-air}	0.41
6	{sports, misc, opinion}	0.43

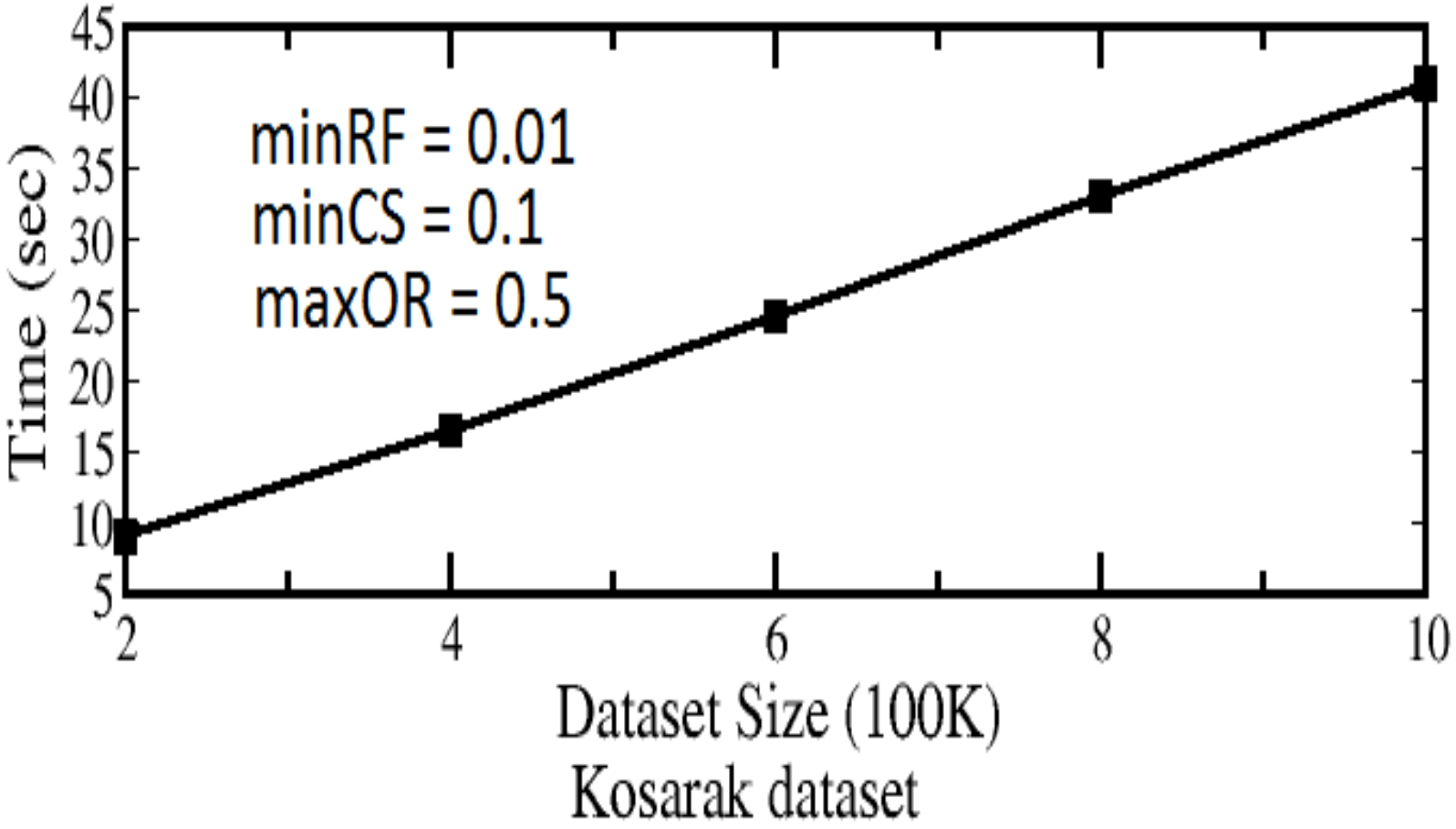
- Advantage:

Proposed approach provides flexibility to the publisher to meet the demands of multiple advertisers by considering different sets of web pages.

Coverage pattern generation



Scalability of CMine Algorithm



Conclusions and Future Work

- Proposed a framework of data mining pattern called “coverage pattern”
- Proposed a methodology to extract the same from transactional databases.
- Through experimental results, we have shown that the proposed model and methodology can effectively discover coverage patterns.
- The coverage patterns help the publisher to meet the demands of multiple advertisers.

Conclusions and Future Work

- We are going to investigate how both frequent and coverage pattern knowledge can be used for efficient banner advertisement placement.
- How the content of the web page and search query can be exploited to explore content specific coverage patterns.
- The notion of coverage patterns can be extended to other domains like bio-informatics for extracting potential knowledge patterns.

Outline

- Pattern Mining: A Road Map
- Coverage Pattern Mining
- **Multilevel Pattern Mining**
- Quantitative Pattern Mining
- Mining Multidimensional Associations
- Diverse Frequent Pattern Mining
- High Utility Pattern mining
- Mining Rare Patterns and Negative Patterns
- Constraint-Based Frequent Pattern Mining
- Summary

About Multilevel Pattern Mining

- Applications require association rules at many levels of abstraction
  - “80 percent of customers that purchase milk may also purchase Bread”
  - “75 percent of people buy wheat bread if they buy “milk(2%)”
- The association relationship in the latter statement is expressed at a lower level of abstraction but carries more specific and concrete information than that in the former.
- Therefore, a data mining system should provide efficient methods for mining multiple-level association rules.

Why Multiple-Level Association Rules?

TID	items
T1	{m1, b2}
T2	{m2, b1}
T3	{b2}

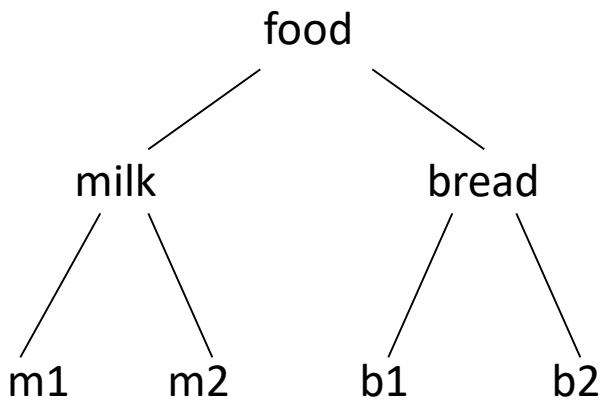
Frequent itemset: {b2}

Association rules: none

Is this database useless?

Why Multiple-Level Association Rules?

What if we have this abstraction tree?



TID	items
T1	{milk, bread}
T2	{milk, bread}
T3	{bread}

minisup = 50% miniconf = 50%

Frequent itemset: {milk, bread}

A.A rules: milk <=> bread

Why Multiple-Level Association Rules?

- Sometimes, at primitive data level, data does not show any significant pattern. But there are useful information hiding behind.

- The goal of Multiple-Level Association Analysis is to find the hidden information in or between levels of abstraction

Input

- Input to Multi-Level Association Rule Mining
  - 1) data at multiple levels of abstraction, and
  - 2) efficient methods for multiple-level rule mining.
- The first requirement can be satisfied by providing concept taxonomies from the primitive level concepts to higher levels.
- In many applications, the taxonomy information is either
  - stored implicitly in the database
  - provided by experts or users,
  - or computed by applying some cluster analysis methods
- We assume that concept taxonomies exist.

Requirements in Multiple-Level Association Analysis

Two general requirements to do multiple-level association rule mining:

1) Provide data at multiple levels of abstraction. (a common practice now)

2) Find efficient methods for multiple-level rule mining.

Observation

- One choice
  - Direct application of the existing single-level association rule mining methods to multiple-level association mining.
  - For example, one may apply the Apriori algorithm to examine data items at multiple levels of abstraction under the same minimum support and minimum confidence thresholds.
- Apriori with single minimum support and minimum confidence may lead to some undesirable results.

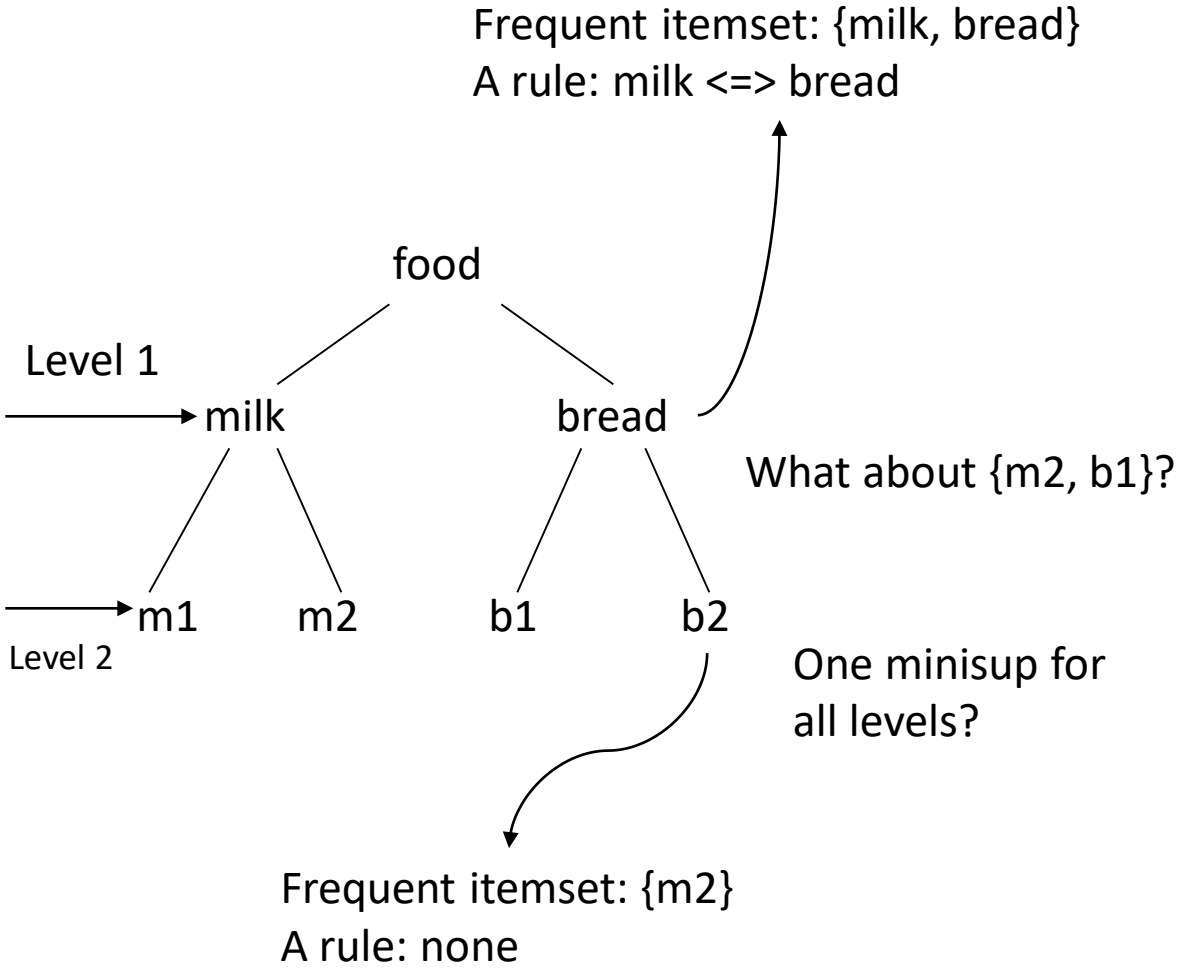
Observation..

- Undesirable results with apriori
  - First, large support is more likely to exist at high levels of abstraction.
    - If one wants to find strong associations at relatively low levels of abstraction, the minimum support threshold must be reduced substantially, which may lead to the generation of many uninteresting associations at high or intermediate levels.
  - Second, since it is unlikely to find many strong association rules at a primitive concept level, mining strong associations should be performed at a rather high concept level. But, rules at high concept levels,
    - may often lead to the rules corresponding to prior knowledge and expectations, such as “milk->bread”, (which could be common sense),
    - or lead to some uninteresting attribute combinations if the minimum support is allowed to be rather small, such as **“toy -> milk**”,

Algorithm : observation

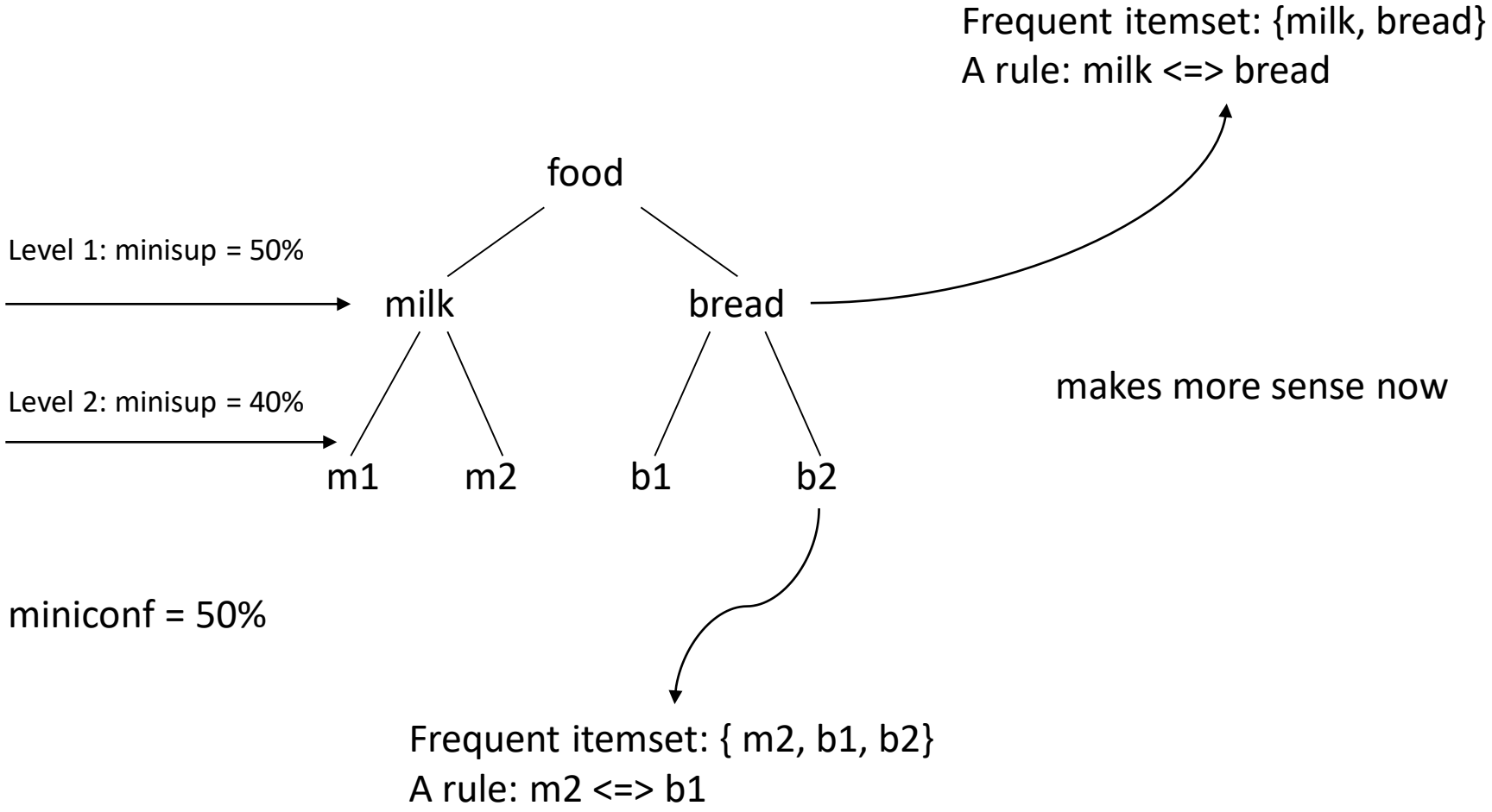
TID	items
T1	{milk, bread}
T2	{milk, bread}
T3	{bread}
T4	{milk, bread}
T5	{milk}

TID	items
T1	{m1, b2}
T2	{m2, b1}
T3	{b2}
T4	{m2, b1}
T5	{m2}



minisup = 50% miniconf = 50%

Algorithm : observation



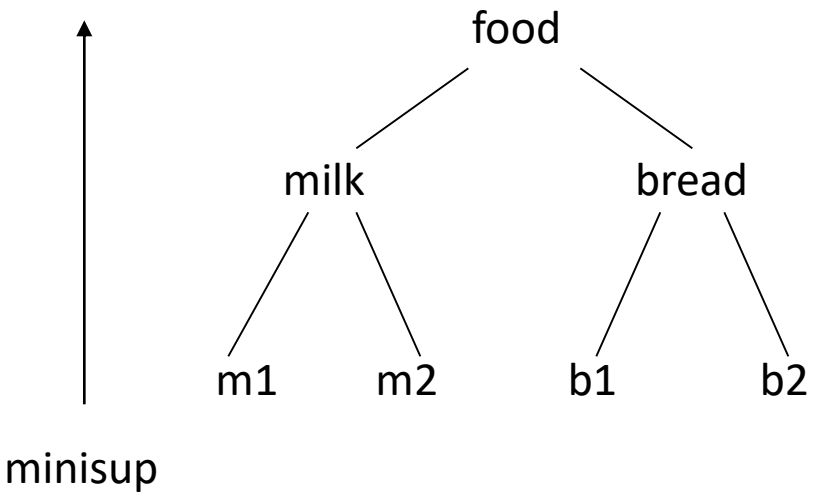
Algorithm: observation

Drawbacks to use only one minisup:

- If the minisup is too high, we are losing information from lower levels

- If the minisup is too low, we are gaining too many rules from higher levels, many of them are useless

Approach: ascending minisup on each level



Research Paper: Mining Multiple-level Association Rules in Large Databases

IEEE Transactions on Knowledge and Data Engineering, 1999

JIAWEI HAN and YONGJIAN FU

Multi-level Association Rules

We assume that the database contains: 1) an item data set which contains the description of each item in $\mathcal{I}$ in the form of $\langle A_i, description_i \rangle$, where $A_i \in \mathcal{I}$, and 2) a transaction data set, $\mathcal{T}$, which consists of a set of transactions $\langle T_i, \{A_p, \ldots, A_q\} \rangle$, where T_i is a transaction identifier and $A_i \in \mathcal{I}$ (for $i = p, \ldots, q$).

Definition 2.2. *A pattern A is **frequent** in set S at level l if the support of A is no less than its corresponding minimum support threshold σ'_l. A rule "$A \Rightarrow B/S$" is **strong** if, for a set S, each ancestor (i.e., the corresponding high-level item) of every item in A and B, if any, is frequent at its corresponding level, "$A \wedge B/S$" is frequent (at the current level), and the confidence of "$A \Rightarrow B/S$" is no less than minimum confidence threshold at the current level.*

Multi-level Association Rules..

- The definition 2.2
  - patterns to be examined at lower levels to be only those with large supports at their corresponding high levels (and thus avoids the generation of many meaningless combinations formed by the descendants of the infrequent patterns).
- For example, in a sales transaction data set,
  - if milk is a frequent pattern, its lower level patterns, such as 2 percent milk, will be examined;
  - whereas if fish is an infrequent pattern, its descendants, such as salmon, will not be examined further.

About the Method

- Used hierarchy information encoded transaction table
- Collect task relevant data and mine
- Encoding can be done during the collection

Encoding Method

- Purpose: To find multiple-level frequent item sets for mining strong association rules in a transaction database
- Input
  - T[1]: a hierarchy-information encoded transaction table of form <TID,Item-set>
  - minisup threshold for each level L in the form: (minsup[L])
- Output: Multiple-level frequent item sets
- Method: A top-down, progressively deepening process which collects frequent item sets at different concept levels as follows.

Algorithm: An Example

An entry of **sales_transaction** Table

Transaction_id	Bar_code_set
351428	{17325,92108,55349,88157,…}

A **sales_item** Description Relation

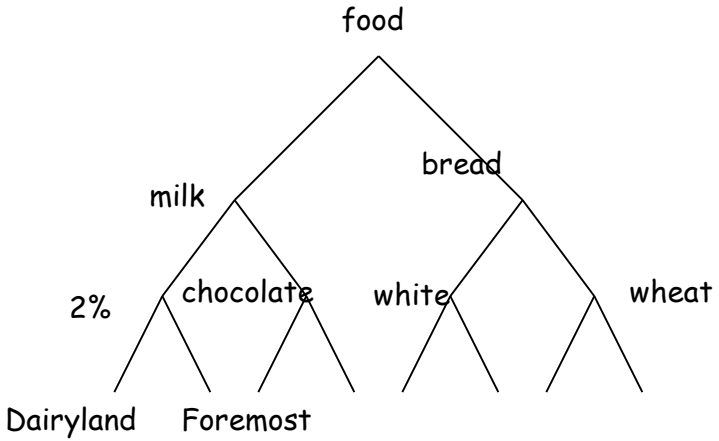
Bar_code	category	Brand	Content	Size	Storage_pd	price
17325	Milk	Foremost	2%	1ga.	14(days)	$3.89

Example 2.1. Let the query be to find multiple-level strong associations in the database in Table 1 for the purchase patterns related to category, content, and brand of the foods which can only be stored for less than three weeks.

Algorithm: An Example

Encode the database with layer information

GID	bar_code	category	content	brand
112	17325	Milk	2%	Foremost



First 1: implies milk

Second 1: implies 2% content

2: implies Foremost brand

Algorithm: An Example

Encoded Transaction Table:T[1]

TID	Items
T1	{111,121,211,221}
T2	{111,211,222,323}
T3	{112,122,221,411}
T4	{111,121}
T5	{111,122,211,221,413}
T6	{211,323,524}
T7	{323,411,524,713}

Algorithm: An Example

The frequent 1-itemset on level 1

L[1,1]

Level-1 minsup = 4

T[2]

Itemset	Support
{1**}	5
{2**}	5

only keep items
in L[1,1] from T[1]

TID	Items
T1	{111,121,211,221}
T2	{111,211,222}
T3	{112,122,221}
T4	{111,121}
T5	{111,122,211,221}
T6	{211}

L[1,2]

Itemset	Support
{1**,2**}	4

Use Apriori on each level

Algorithm: An Example

Level-2 minsup = 3

L[2,1]

Itemset	Support
{11*}	5
{12*}	4
{21*}	4
{22*}	4

L[2,2]

Itemset	Support
{11*,12*}	4
{11*,21*}	3
{11*,22*}	4
{12*,22*}	3
{21*,22*}	3

L[2,3]

Itemset	Support
{11*,12*,22*}	3
{11*,21*,22*}	3

Frequent Item Sets at Level 3

Level-3 minsup = 3

L[3,1]

Itemset	Support
{111}	4
{211}	4
{221}	3

L[3,2]

Itemset	Support
{111,211}	3

Only generate T[1] & T[2], all frequent itemsets after level 2 is generated from T[2]

E.g.

Level-1:

80% of customers that purchase milk also purchase bread.

milk ➔ bread with Confidence= 80%

Level-2:

75% of people who buy 2% milk also buy wheat bread.

2% milk ➔ wheat bread with Confidence= 75%

Multi-level Association: Flexible Support and Redundancy filtering

- Flexible min-support thresholds: Some items are more valuable but less frequent
  - Use non-uniform, group-based min-support
  - E.g., {diamond, watch, camera}: 0.05%; {bread, milk}: 5%; …
- Redundancy Filtering: Some rules may be redundant due to "ancestor" relationships between items
  - milk $\Rightarrow$ wheat bread [support = 8%, confidence = 70%]
  - 2% milk $\Rightarrow$ wheat bread [support = 2%, confidence = 72%]

  The first rule is an ancestor of the second rule
- A rule is *redundant* if its support is close to the "expected" value, based on the rule's ancestor

MIDSEM
(UP TO The Preceding Slide)

Outline

- Pattern Mining: A Road Map
- Coverage Pattern Mining
- Multilevel Pattern Mining
- **Quantitative Pattern Mining**
- Diverse Frequent Pattern Mining
- High Utility Pattern mining
- Mining Rare Patterns and Negative Patterns
- Constraint-Based Frequent Pattern Mining
- Summary

- Class #14: no class (mid preparation)
- Class #15

Boolean Association Rules

Trans-ID	Items
1	A C D
2	B C E
3	A B C E
4	B E
5	A B C E

- The table has an attribute correspond to each item. A record correspond to each transaction.
- Value of attribute is “1” if it exists on record.
- All the attributes are Boolean.

Boolean Association Rules

TID	A	B	C	D	E
1	1	0	1	1	0
2	0	1	1	0	1
3	1	1	1	0	1
4	0	1	0	0	1
5	1	1	1	0	1

- Attribute has a value of “1” if the transaction contains the corresponding item; “0” otherwise.

787/1418

Large Relational Tables in real-life.

- Relational tables in most business and scientific domains have richer attribute types.
  - Quantitative attributes (e.g. age, income)
  - Categorical attributes (e.g. zip code, marriage status, make of car)
- Can not apply existing methods to extract Boolean association rules to mine association rules over quantitative and categorical attributes.
  - This paper present techniques for discovering such rules.

Quantitative Association Rules: Example

RecordID	Age	Married	NumCars
100	23	No	1
200	25	Yes	1
300	29	No	0
400	34	Yes	2
500	38	Yes	2

Sample Rules	Support	Confidence
<age:30..39> and <married: yes> ==> <numCars:2>	40%	100%
<NumCars: 0..1> ==> <Married: No>	40%	66.70%

Mapping to Boolean Association Rules Problem

- Using <attribute: value> as new attribute, which has only Boolean values
- Categorical: The value of Boolean field corresponding to <attribute1, value1> would be “1”, if atribute1 had value 1 in the original record, and “0” otherwise.
- If the domain of values is large, partition the values into intervals and map each pair to Boolean attribute.

- Use any exiting algorithm a priori or FP Tree.

RecordID	Age: 20..29	Age: 30..39	Married: Yes	Married: No	NumCars: 0	NumCars: 1
100	1	0	0	1	0	1
200	1	0	1	0	0	1
300	1	0	0	1	1	0
400	0	1	1	0	0	0
500	0	1	1	0	0	0

First Problem with Direct Mapping

- **MinSup**: If the number of intervals for a quantitative attribute is large, the support for any single interval can be low. Some rules involving this attribute may not be found because they lack minimum support.

Second Problem with Direct Mapping

- **MinConf.** There is some information lost whenever we partition values into intervals. This information loss increases as the interval sizes become larger.

Example "*MinConf*" problem.

Rec-ID	Age: 20..29	Age: 30..39	Married: Yes	Married: No	NumCars: 0	NumCars: 1	NumCars: 2
100	1	0	0	1	0	1	0
200	1	0	1	0	0	1	0
300	1	0	0	1	1	0	0
400	0	1	1	0	0	0	1
500	0	1	1	0	0	0	1

- The rule

  < NumCars: 0> => <Married: No> has 100% confidence.

Example "*MinConf*" problem. (con't)

Rec-ID	Age: 20..29	Age: 30..39	Married: Yes	Married: No	NumCars: 0..1	NumCars: 2..3
100	1	0	0	1	1	1
200	1	0	1	0	1	1
300	1	0	0	1	1	0
400	0	1	1	0	0	0

- Closest rule is

  <NumCars: 0..1> => <Married: No>

  has only 66.66% confidence.

“catch-22” Situation.

- Mapping problems create “catch-22” situation.
  - If the intervals are too large, some rules may not have minimum confidence.
  - If the intervals are too small, some rules may not have minimum support.

Dynamic discretization based on data distribution (quantitative rules, e.g., Agrawal & Srikant@SIGMOD96)

- Minsupport problem: Adjacent intervals can be combined till minsup is satisfies
- Minconfidence problem: Exploit generalization/specialization, Expected support/expected confidence

Mining Quantitative Associations

Techniques can be categorized by how numerical attributes, such as age or salary are treated

1. Static discretization based on predefined concept hierarchies (data cube methods)
2. Dynamic discretization based on data distribution (quantitative rules, e.g., Agrawal & Srikant@SIGMOD96)
3. Clustering: Distance-based association (e.g., Yang & Miller@SIGMOD97)
   - One dimensional clustering then association
4. Deviation: (such as Aumann and Lindell@KDD99)

   Sex = female => Wage: mean=$7/hr (overall mean = $9)

Mining Multi-Dimensional Association

- Single-dimensional rules:

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

- Multi-dimensional rules: $\geq$ 2 dimensions or predicates
  - Inter-dimension assoc. rules (*no repeated predicates*)

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

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

- Categorical Attributes: finite number of possible values, no ordering among values—data cube approach
- Quantitative Attributes: Numeric, implicit ordering among values—discretization, clustering, and gradient approaches

Outline

- Pattern Mining: A Road Map
- Coverage Pattern Mining
- Multilevel Pattern Mining
- Quantitative Pattern Mining
- **Diverse Frequent Pattern Mining**
- High Utility Pattern mining
- Mining Rare Patterns and Negative Patterns
- Constraint-Based Frequent Pattern Mining
- Summary

Model of Diverse Frequent Patterns

International Journal of Data Science and Analytics
https://doi.org/10.1007/s41060-019-00203-2

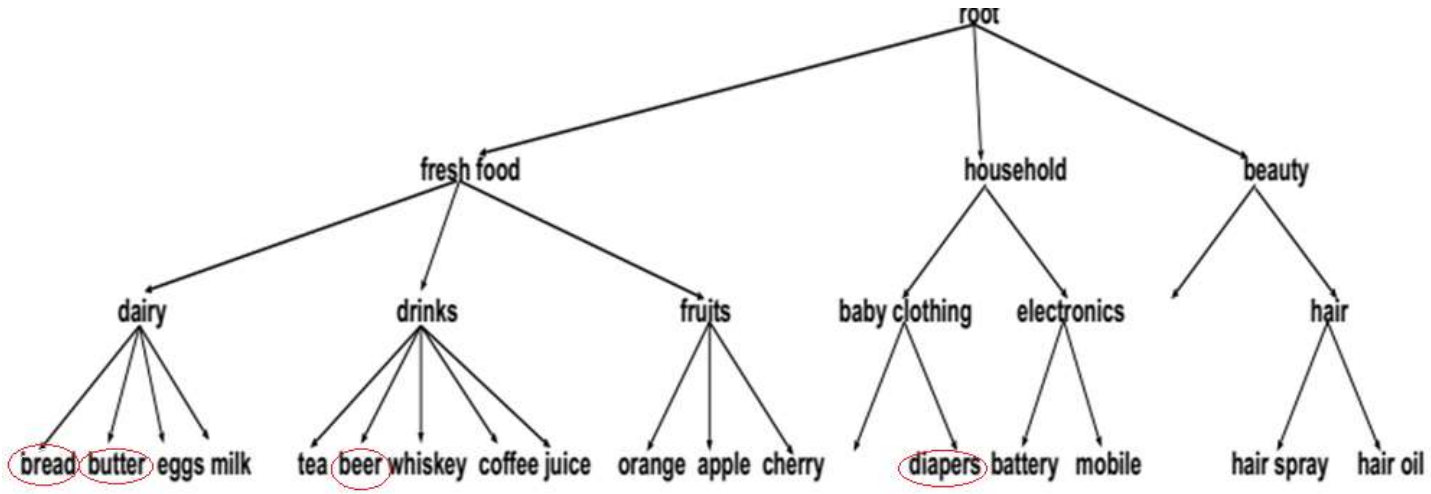
REGULAR PAPER

A model of concept hierarchy-based diverse patterns with applications to recommender system

M. Kumara Swamy[1] · P. Krishna Reddy[1]

Notion of Diversity

- Consider patterns, {***bread***, ***butter***} and {***beer***, ***diaper***} with the same *support*
- Normally {***beer***, ***diaper***} is more interesting than {***bread***, ***butter***}
- Reason
  - The items ***bread*** and ***butter*** are belong to same category ***Dairy*** (***Food Product***)
  - The items ***beer*** and ***diaper*** belong to different categories ***drinks*** and ***baby clothing***
- We say that {***beer***, ***diaper***} is more ***diverse*** than {***bread***, ***butter***}



Notion of Diversity …

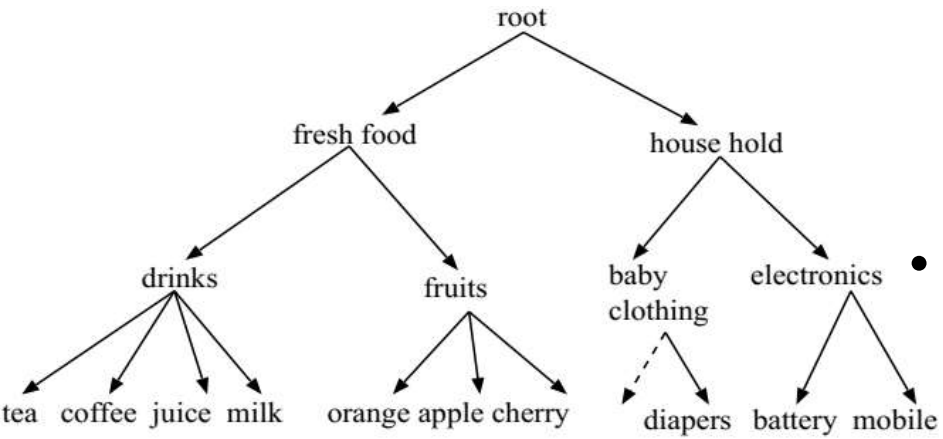
- For certain types of applications, it may be useful to distinguish between the pattern with items belonging to different categories and the pattern with items belonging to few categories.
- The existing pattern extraction approaches (frequent, sequential, periodic, etc.) fail to make such distinction.
- It is possible to rank the patterns by analysing the extent to which the items in the patterns belong to different categories.

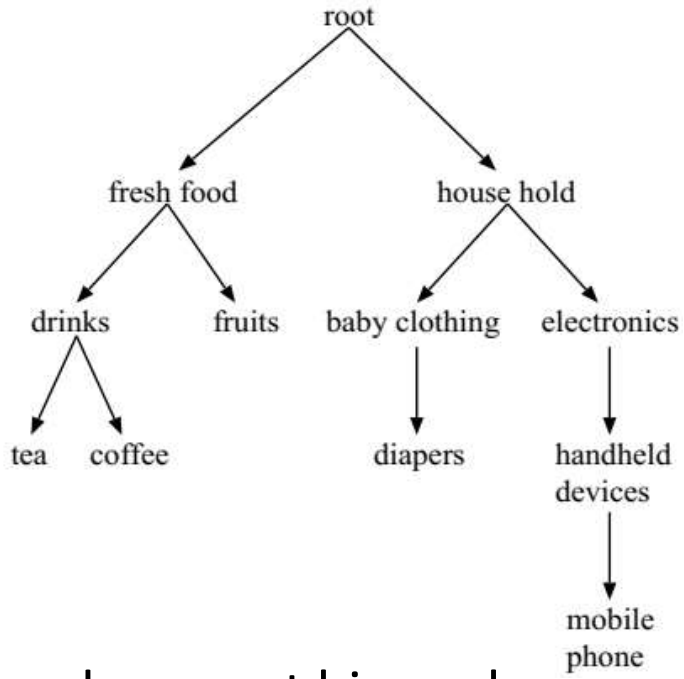
About Computation of Diversity

- A pattern is an itemset (set of items)
- The diversity value of a pattern is called diverse rank (*drank*)
- To compute the *drank* of a pattern, the following issues are to be resolved
  - To determine the category of items, category level relationship among the items is needed
    - We employ *concept hierarchy* to find the relationship of the items at higher level categories
  - Given a concept hierarchy, a framework is required to compute the *drank* value of the pattern
  - Given a set of items, concept hierarchy, and transactional database, an approach has to be developed to extract *diverse patterns* and *diverse frequent patterns*

About Balanced and Unbalanced Concept Hierarchy

- Balanced concept hierarchy
  - The depth of all leaf-level items are equal





- Unbalanced concept hierarchy
  - The depth of all leaf-level items are not equal

Model of Diverse Pattern

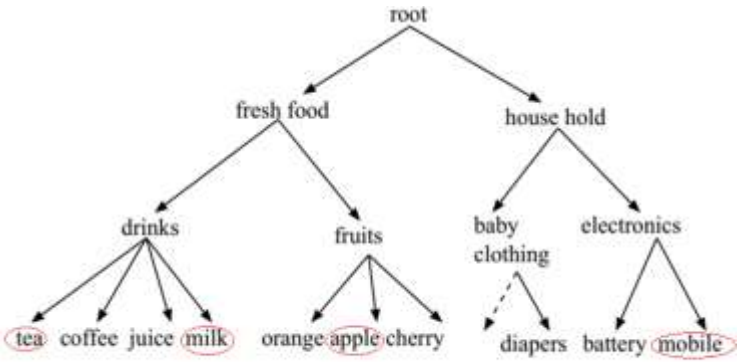
- The diversity of a pattern measures the extent of the items belong to multiple higher level categories.
  - The items of a pattern are mapped to the same/few categories in a concept hierarchy, the pattern has low diversity
  - Relatively, the items of a pattern are mapped to multiple categories, the pattern has more diversity

- Merging behavior: The speed of mapping of items from low level items to high level categories

- The question: how to measure the merging behavior?
  - The pattern merges into one/few higher level categories quickly, and ultimately all paths join at the *root*, the pattern has low diversity value
  - The pattern merges directly at *root* slowly by crossing several intermediary categories, it has relatively high diversity values

Model of Diverse of Pattern …

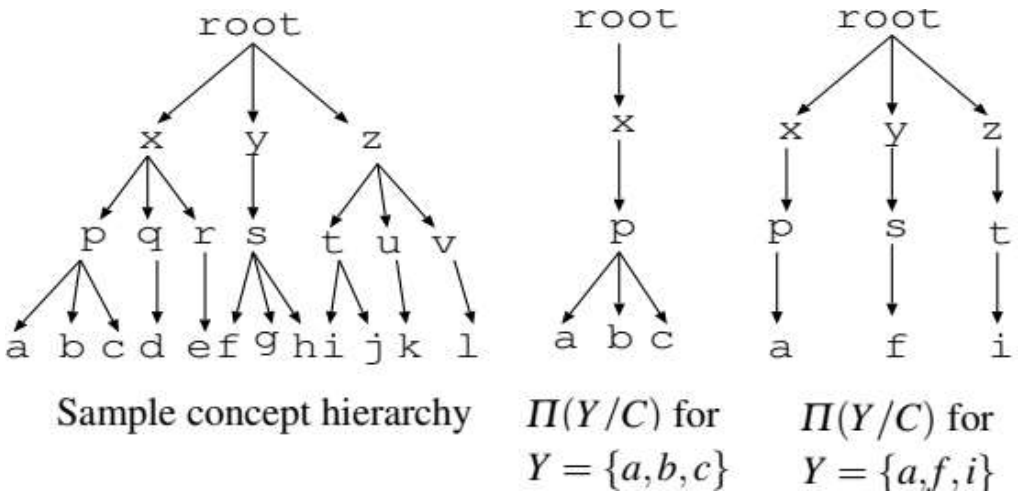
- Example
  - Consider the patterns {*tea*, *milk*}, {*milk*, *apple*} and {*milk*, *mobile*}
  - The pattern {*tea*, *milk*} quickly merges to parent, the diversity of the pattern is low.
  - The pattern {*milk*, *apple*} merges slowly as compared to the pattern {*tea*, *milk*}, the diversity of the pattern is medium.
  - Similarly, the pattern {*milk*, *mobile*} merges slowly at *root* crossing several internal nodes, the diversity of the pattern is high



A balanced concept hierarchy

Computing the Diversity of a Pattern

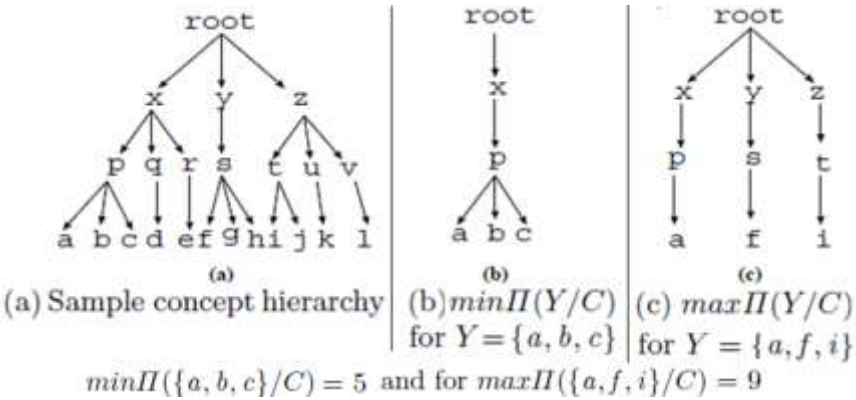
- Terminology
  - Balanced patterns (BP): Let $I = \{i_1, i_2 \cdots , i_n\}$ be a set of items, D be a transactional database on I, and C be a concept hierarchy of I with height h. A pattern $Y = \{i_1, i_2 \cdots , i_m\} \subseteq I$ with m items is called balanced pattern, if the height of all the items in Y is equal to h.
  - Projection of Pattern (Y) on concept hierarchy (C): It is denoted by $\pi(Y/C)$. It is a sub-tree which contains the portion of C concerning to the pattern Y. All the nodes and edges exist in the paths of the items of Y to the *root.*



Sample concept hierarchy $\Pi(Y/C)$ for $Y = \{a,b,c\}$ $\Pi(Y/C)$ for $Y = \{a,f,i\}$

Framework to Compute *drank* using Balanced Concept Hierarchy

- Given the pattern *Y* of certain size and concept hierarchy *C*, two extreme projections are possible
- Minimal projection
  - All the items of *Y* merge at immediate higher level
  - The number of edges $|min\ \pi(Y/C)|$ is equal to $(|Y| + h - 1)$
- Maximal projection
  - All the leaf level nodes merges at *root.*
  - The number of edges $|max\ \pi(Y/C)|$ is equal to $(|Y|\ \text{x}\ h)$
- Observation: Given a pattern Y of certain size
  - Its *minimal projection* contains minimum number of edges and *maximal project* contains the maximum number of edges



(a) Sample concept hierarchy (b) $minΠ(Y/C)$ for $Y = \{a, b, c\}$ (c) $maxΠ(Y/C)$ for $Y = \{a, f, i\}$

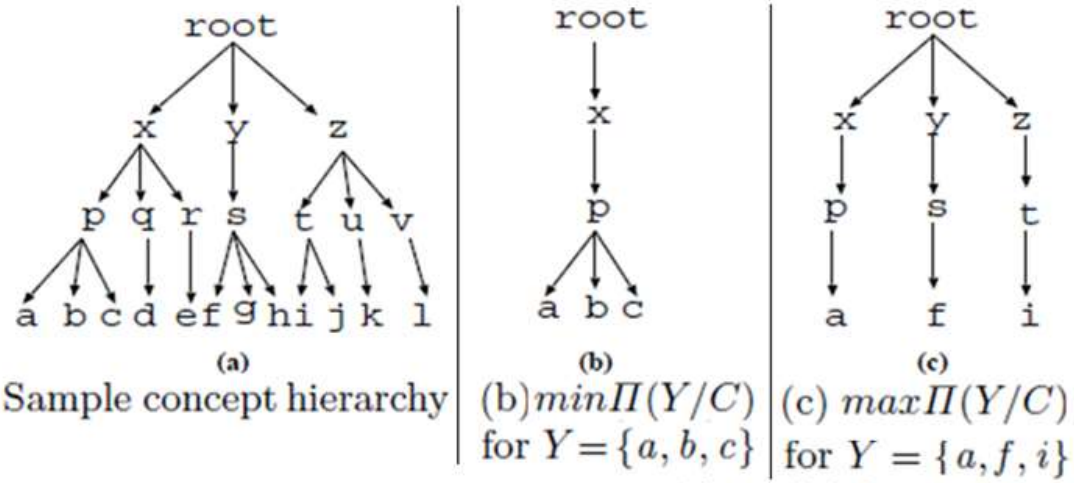
$minΠ(\{a, b, c\}/C) = 5$ and for $maxΠ(\{a, f, i\}/C) = 9$

Framework to Compute *drank* using Balanced Concept Hierarchy …

- Given a pattern Y, the *drank*(*Y*) is the ratio of number of edges in its projection ($\pi(Y/C)$) and the maximal projection ($max\ \pi(Y/C)$) which is equal to $\frac{|\Pi(Y/C)|}{|max\Pi(Y/C)|}$
- The minimum value of this ratio is equal to $\frac{|min\Pi(Y/C)|}{|max\Pi(Y/C)|}$ which is equal to 0.
- The maximum value of this ratio is equal to $\frac{|max\Pi(Y/C)|}{|max\Pi(Y/C)|}$ which is equal to 1.
- On applying the min-max normalization, we get following formula. $drank(Y) = \frac{(|\Pi(Y/C)|) - (|Y| + h - 1)}{(h-1)(|Y|-1)}$

$$drank(Y) = \left(\frac{\left(\frac{|\Pi(Y/C)|}{|Y|\times h}\right) - \left(\frac{|Y|+h-1}{|Y|\times h}\right)}{1 - \left(\frac{|Y|+h-1}{|Y|\times h}\right)} \right)(1-0) + 0$$

Example



(a) Sample concept hierarchy | (b) $minΠ(Y/C)$ for $Y=\{a,b,c\}$ | (c) $maxΠ(Y/C)$ for $Y=\{a,f,i\}$

$minΠ(\{a,b,c\}/C)=5$ and for $maxΠ(\{a,f,i\}/C)=9$

$$drank(Y)=\frac{(|Π(Y/C)|)-(|Y|+h-1)}{(h-1)(|Y|-1)}$$

$drank(\{a,b,c\})=\frac{5-5}{2\times 2}=0.0$

$drank(\{a,b,e\})=0.25$

$drank(\{a,b,f\})=0.5$

$drank(\{a,e,f\})=0.75$

$drank(\{a,f,i\})=1.0$

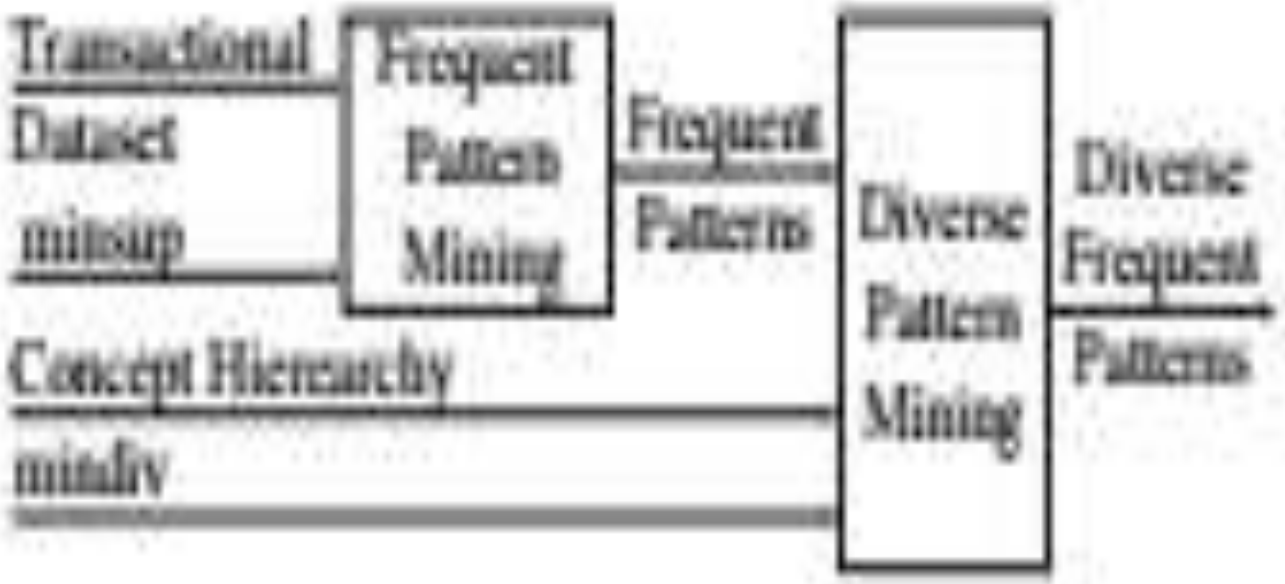


Fig. 3 Extraction of diverse frequent patterns

Experiments

- Dataset from MovieLens project (http://www.grouplens.org)
  : contains 100,000 ratings of 943 users on 1682 movies

Table 1 Top-10 patterns of size 3 sorted w.r.t. *drank*

Top 10 diverse patterns	*drank*	*Support* (%)
{G.I. Jane, The Game, Contact}	1	5.5
{G.I. Jane, Titanic, Contact}	1	5.3
{Amistad, L.A. Confidential, Titanic}	1	5.3
{One Fine Day, Return of the Jedi, Star Wars}	1	5.1
{One Fine Day, Star Wars, Return of the Jedi}	1	5.1
{The First Wives Club, Return of the Jedi, Star Wars}	1	5.1
{Volcano, Star Wars, Return of the Jedi}	1	5.0
{Welcome to the Dollhouse, Trainspotting, Star Wars}	1	4.8
{Austin Powers: International Man of Mystery, Fargo, Star Wars}	1	4.6
{Austin Powers: Int'l. Man of Mystery, Return of the Jedi, Star Wars}	1	4.3

Table 2 Top 10 patterns of size 3 sorted w.r.t. support value

Top 10 frequent patterns	*Support* (%)	*drank*
{Raiders of the Lost Ark, Return of the Jedi, Star Wars}	23.2	0.33
{The Empire Strikes Back, Raiders of the Lost Ark, Star Wars}	20.6	0.33
{The Rock, Return of the Jedi, Star Wars}	17.7	0.33
{Independence Day, Return of the Jedi, Star Wars}	18.2	0.33
{Aliens, Star Wars, Raiders of the Lost Ark}	16.1	0.75
{Braveheart, Indiana Jones and the Last Crusade, Return of the Jedi}	14.9	0.33
{Aliens, Return of the Jedi, Star Wars}	14.8	0.33
{The Terminator, Pulp Fiction, Raiders of the Lost Ark}	14.8	0.33
{The Princess Bride, The Empire Strikes Back, Raiders of the Lost Ark}	14.5	0.33
{Independence Day, Raiders of the Lost Ark, Return of the Jedi}	14.4	0.33

Outline

- Pattern Mining: A Road Map
- Coverage Pattern Mining
- Multilevel Pattern Mining
- Quantitative Pattern Mining
- Diverse Frequent Pattern Mining
- **High Utility Pattern mining**
- Mining Rare Patterns and Negative Patterns
- Constraint-Based Frequent Pattern Mining
- Summary

HIGH UTILITY MINING

A Two-Phase Algorithm for Fast Discovery of High Utility Itemsets

Ying Liu, Wei-keng Liao, and Alok Choudhary

PAKDD 2005

Limitations of Frequent Itemset Mining

- Purchase quantities are not taken into account.
  - Thus, an item may only appear once or zero time in a transaction. Thus, if a customer has bought five breads, ten breads or twenty breads, it is viewed as the same.
- All items are viewed as having the same importance, utility of weight.
  - For example, if a customer buys a very expensive bottle of wine or just a piece of bread, it is viewed as being equally important.
- Frequent pattern mining may find many frequent patterns that are not interesting.
  - For example, one may find that {bread, milk} is a frequent pattern. However, from a business perspective, this pattern may be uninteresting because it does not generate much profit.
  - Moreover, frequent pattern mining algorithms may miss the rare patterns that generate a high profit such as perhaps {caviar, wine}

High-utility itemset mining

- **High-utility itemset mining addresses the limitations** .
- A transaction database contains transactions where purchase quantities are taken into account as well as the unit profit of each item. For example, consider the following transaction database.
- Local transaction utility of an item
- External utility of an item
- Utility of the item in a transaction= Local utility * external utility

Table 1. A transaction database

(a) Transaction table. Each row is a transaction. The columns represent the number of items in a particular transaction. TID is the transaction identification number

TID \ ITEM	A	B	C	D	E
T_1	0	0	18	0	1
T_2	0	6	0	1	1
T_3	2	0	1	0	1
T_4	1	0	0	1	1
T_5	0	0	4	0	2
T_6	1	1	0	0	0
T_7	0	10	0	1	1
T_8	3	0	25	3	1
T_9	1	1	0	0	0
T_{10}	0	6	2	0	2

(b) The utility table. The right column displays the profit of each item per unit in dollars

ITEM	PROFIT ($) (per unit)
A	3
B	10
C	1
D	6
E	5

(c) Transaction utility (TU) of the transaction database

TID	TU	TID	TU
T_1	23	T_6	13
T_2	71	T_7	111
T_3	12	T_8	57
T_4	14	T_9	13
T_5	14	T_{10}	72

Utility Mining Problem

- Utility mining is to find all the itemsets whose utility values are beyond a user-specified threshold. An itemset X is a high utility itemset if $u(X) \geq \varepsilon$, where $X \subseteq I$ and ε is the minimum utility threshold, otherwise, it is a low utility itemset.
- Non-existence of "downward closure property" (anti-monotone property) in the *utility mining* model

- $I = \{i_1, i_2, \ldots, i_m\}$ is a set of items.
- $D = \{T_1, T_2, \ldots, T_n\}$ be a transaction database where each transaction $T_i \in D$ is a subset of I.
- $o(i_p, T_q)$, *local transaction utility value*, represents the quantity of item i_p in transaction T_q. For example, $o(\text{A}, T_8) = 3$, in Table 1(a).
- $s(i_p)$, *external utility,* is the value associated with item i_p in the Utility Table. This value reflects the importance of an item, which is independent of transactions. For example, in Table 1(b), the external utility of item A, $s(\text{A})$, is 3.
- $u(i_p, T_q)$, *utility*, the quantitative measure of utility for item i_p in transaction T_q, is defined as $o(ip,Tq) \times s(ip)$. For example, $u(A, T_8) = 3 \times 3 = 9$, in Table 1
- $u(X, T_q)$, *utility of an itemset X in transaction* T_q, is defined as $\sum_{ip \in X} u(ip,Tq)$, where

  $X = \{i_1, i_2, \ldots, i_k\}$ is a k-itemset, $X \subseteq T_q$ and $1 \leq k \leq m$.
- $u(X)$, *utility of an itemset X*, is defined as $\sum_{T_q \in D \wedge X \subseteq T_q} u(X,T_q)$. (3.1)

Two-Phase algorithm

- Transaction utility= sum of all utilities of items
- Transaction weighted utility of itemset X, twu(X)= Summation of utilities all transactions Ti such that X is belongs to Ti.
- High Transaction-weighted Utilization Itemset: Transaction weighted Utility of the itemset should be greater than the user given threshold.
- Transaction-weighted Downward Closure Property: If k-itemset is high transaction-weighted utilization itemset, k-1 itemset will be high transaction-weighted utilization itemset.
- Theorem: Let HTWU be the collection of all high transaction-weighted utilization itemsets in a transaction database D, and HU be the collection of high utility itemsets in D. Then HU is the subset of HTWU.

Theorem 1. (Transaction-weighted Downward Closure Property) Let I^k be a k-itemset and I^{k-1} be a $(k-1)$-itemset such that $I^{k-1} \subset I^k$. If I^k is a high transaction-weighted utilization itemset, I^{k-1} is a high transaction-weighted utilization itemset.

Proof: Let T_{I^k} be the collection of the transactions containing I^k and $T_{I^{k-1}}$ be the collection containing I^{k-1}. Since $I^{k-1} \subset I^k$, $T_{I^{k-1}}$ is a superset of T_{I^k}. According to Definition 2, $twu(I^{k-1}) = \sum_{I^{k-1} \subseteq T_q \in D} tu(T_q) \geq \sum_{I^k \subseteq T_p \in D} tu(T_p) = twu(I^k) \geq \varepsilon'$ ❑

The *Transaction-weighted Downward Closure Property* indicates that any superset of a low transaction-weighted utilization itemset is low in transaction-weighted utilization. That is, only the combinations of high transaction-weighted utilization $(k-1)$-itemsets could be added into the candidate set C_k at each level.

Theorem 2. Let *HTWU* be the collection of all high transaction-weighted utilization itemsets in a transaction database *D*, and *HU* be the collection of high utility itemsets in *D*. If $\varepsilon' = \varepsilon$, then $HU \subseteq HTWU$.

Proof: $\forall X \in HU$, if X is a high utility itemset, then

$$\varepsilon' = \varepsilon \leq u(X) = \sum_{X \subseteq T_q} u(X, T_q) = \sum_{X \subseteq T_q} \sum_{i_p \in X} u(i_p, T_q) \leq \sum_{X \subseteq T_q} \sum_{i_p \in T_q} u(i_p, T_q) = \sum_{X \subseteq T_q} tu(T_q) = twu(X)$$

Thus, X is a high transaction-weighted utilization itemset and $X \in HTWU$. ❑

According to Theorem 2, we can utilize the *Transaction-weighted Downward Closure Property* in our *transaction-weighted utilization mining* in Phase I by assuming $\varepsilon' = \varepsilon$ and prune those overestimated itemsets in Phase II.

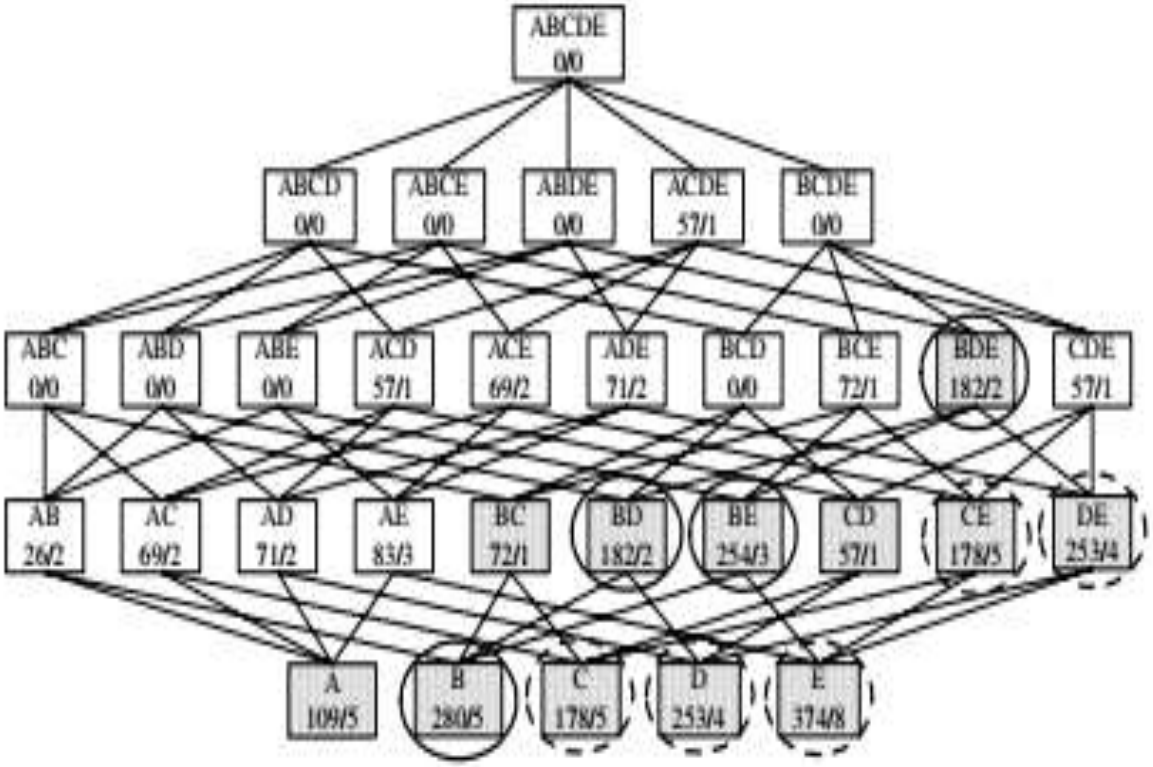


Fig. 1. Itemsets lattice related to the example in Table 1. $\varepsilon' = 120$. Itemsets in circles (solid and dashed) are the high transaction-weighted utilization itemsets in *transaction-weighted utilization mining model*. Gray-shaded boxes denote the search space. Itemsets in solid circles are high utility itemsets. Numbers in each box are transaction-weighted utilization / number of occurrence

Outline

- Pattern Mining: A Road Map
- Coverage Pattern Mining
- Multilevel Pattern Mining
- Quantitative Pattern Mining
- Diverse Frequent Pattern Mining
- High Utility Pattern mining
- **Mining Rare Patterns and Negative Patterns**
- Constraint-Based Frequent Pattern Mining
- Summary

Negative and Rare Patterns

- Rare patterns: Very low support but interesting
  - E.g., buying Rolex watches
  - Mining: Setting individual-based or special group-based support threshold for valuable items
- Negative patterns
  - Since it is unlikely that one buys Ford Expedition (an SUV car) and Toyota Prius (a hybrid car) together, Ford Expedition and Toyota Prius are likely negatively correlated patterns
- Negatively correlated patterns that are infrequent tend to be more interesting than those that are frequent

Defining Negative Correlated Patterns (I)

- Definition 1 (support-based)
  - If itemsets X and Y are both frequent but rarely occur together, i.e.,

    $sup(X \cup Y) < sup(X) * sup(Y)$
  - Then X and Y are negatively correlated
- Problem: A store sold two needle 100 packages A and B, only one transaction containing both A and B.
  - When there are in total 200 transactions, we have

    $s(A \cup B) = 0.005$, $s(A) * s(B) = 0.25$, $s(A \cup B) < s(A) * s(B)$
  - When there are 10^5 transactions, we have

    $s(A \cup B) = 1/10^5$, $s(A) * s(B) = 1/10^3 * 1/10^3$, $s(A \cup B) > s(A) * s(B)$
  - Where is the problem? —Null transactions, i.e., the support-based definition is not null-invariant!

Defining Negative Correlated Patterns (II)

- Definition 2 (negative itemset-based)
  - X is a *negative itemset* if (1) X = $\bar{A}$ U B, where B is a set of positive items, and $\bar{A}$ is a set of negative items, $|\bar{A}| \geq 1$, and (2) $s(X) \geq \mu$
  - Itemsets X is negatively correlated, if

$$s(X) < \prod_{i=1}^{k} s(x_i), where\ x_i \in X,\ and\ s(x_i)\ is\ the\ support\ of\ x_i$$

- This definition suffers a similar null-invariant problem
- Definition 3 (Kulzynski measure-based) If itemsets X and Y are frequent, but $(P(X|Y) + P(Y|X))/2 < \epsilon$, where ϵ is a negative pattern threshold, then X and Y are negatively correlated.
- Ex. For the same needle package problem, when no matter there are 200 or 10^5 transactions, if $\epsilon = 0.01$, we have

$$(P(A|B) + P(B|A))/2 = (0.01 + 0.01)/2 < \epsilon$$

Mining Compressed Patterns

Pat-ID	Item-Sets	Support
P1	{38,16,18,12}	205227
P2	{38,16,18,12,17}	205211
P3	{39,38,16,18,12,17}	101758
P4	{39,16,18,12,17}	161563

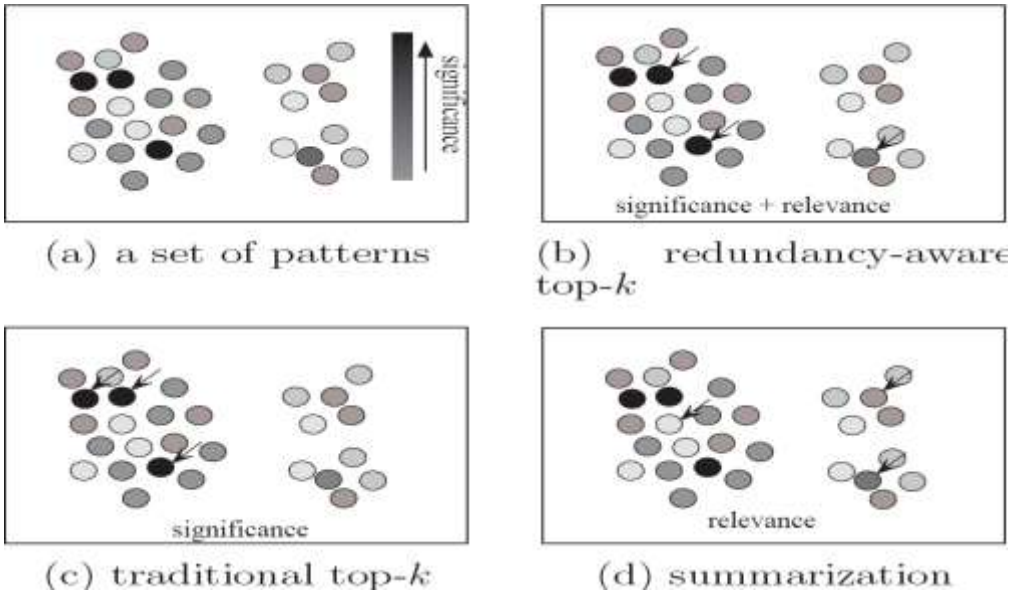
- Closed patterns
  - P1, P2, P3, P4, P5
  - Emphasizes too much on support
- Max-patterns
  - P3: information loss
- Desired output (a good balance):
  - P2, P3, P4

- Why mining compressed patterns? Too many scattered patterns but not so meaningful
- Pa $Dist(P_1, P_2) = 1 - \frac{|T(P_1) \cap T(P_2)|}{|T(P_1) \cup T(P_2)|}$

- δ-clustering: For each pattern P, find all patterns which can be expressed by P and whose distance to P is within δ (δ-cover)
- All patterns in the cluster can be represented by P

Redundancy-Aware Top-k Patterns

- Desired patterns: high significance & low redundancy



(a) a set of patterns (b) redundancy-aware top-k

(c) traditional top-k (d) summarization

- Method: Use MMS (Maximal Marginal Significance) for measuring the combined significance of a pattern set
- Xin et al., Extracting Redundancy-Aware Top-K Patterns, KDD'06

Outline

- Pattern Mining: A Road Map
- Coverage Pattern Mining
- Multilevel Pattern Mining
- Quantitative Pattern Mining
- Diverse Frequent Pattern Mining
- High Utility Pattern mining
- Mining Rare Patterns and Negative Patterns
- **Constraint-Based Frequent Pattern Mining**
- Summary

Constraint-based (Query-Directed) Mining

- Finding all the patterns in a database autonomously? — unrealistic!
  - The patterns could be too many but not focused!
- Data mining should be an interactive process
  - User directs what to be mined using a data mining query language (or a graphical user interface)
- Constraint-based mining
  - User flexibility: provides constraints on what to be mined
  - Optimization: explores such constraints for efficient mining — constraint-based mining: constraint-pushing, similar to push selection first in DB query processing
  - Note: still find all the answers satisfying constraints, not finding some answers in “heuristic search”

Constraints in Data Mining

- Knowledge type constraint:
  - classification, association, etc.
- Data constraint — using SQL-like queries
  - find product pairs sold together in stores in Chicago this year
- Dimension/level constraint
  - in relevance to region, price, brand, customer category
- Rule (or pattern) constraint
  - small sales (price < $10) triggers big sales (sum > $200)
- Interestingness constraint
  - strong rules: min_support ≥ 3%, min_confidence ≥ 60%

Meta-Rule Guided Mining

- Meta-rule can be in the rule form with partially instantiated predicates and constants

  $P_1(X, Y) \wedge P_2(X, W) \Rightarrow buys(X, \text{“iPad”})$

- The resulting rule derived can be

  $age(X, \text{“15-25”}) \wedge profession(X, \text{“student”}) \Rightarrow buys(X, \text{“iPad”})$

- In general, it can be in the form of

  $P_1 \wedge P_2 \wedge \ldots \wedge P_l \Rightarrow Q_1 \wedge Q_2 \wedge \ldots \wedge Q_r$

- Method to find meta-rules
  - Find frequent (l+r) predicates (based on min-support threshold)
  - Push constants deeply when possible into the mining process (see the remaining discussions on constraint-push techniques)
  - Use confidence, correlation, and other measures when possible

Constraint-Based Frequent Pattern Mining

- Pattern space pruning constraints
  - Anti-monotonic: If constraint c is violated, its further mining can be terminated
  - Monotonic: If c is satisfied, no need to check c again
  - Succinct: c must be satisfied, so one can start with the data sets satisfying c
  - Convertible: c is not monotonic nor anti-monotonic, but it can be converted into it if items in the transaction can be properly ordered
- Data space pruning constraint
  - Data succinct: Data space can be pruned at the initial pattern mining process
  - Data anti-monotonic: If a transaction t does not satisfy c, t can be pruned from its further mining

What Constraints Are Convertible?

Constraint	Convertible anti-monotone	Convertible monotone	Strongly convertible
avg(S) ≤ , ≥ v	Yes	Yes	Yes
median(S) ≤ , ≥ v	Yes	Yes	Yes
sum(S) ≤ v (items could be of any value, v ≥ 0)	Yes	No	No
sum(S) ≤ v (items could be of any value, v ≤ 0)	No	Yes	No
sum(S) ≥ v (items could be of any value, v ≥ 0)	No	Yes	No
sum(S) ≥ v (items could be of any value, v ≤ 0)	Yes	No	No
……			

Constraint-Based Mining — A General Picture

Constraint	Anti-monotone	Monotone	Succinct
$v \in S$	no	yes	yes
$S \supseteq V$	no	yes	yes
$S \subseteq V$	yes	no	yes
$\min(S) \leq v$	no	yes	yes
$\min(S) \geq v$	yes	no	yes
$\max(S) \leq v$	yes	no	yes
$\max(S) \geq v$	no	yes	yes
$\text{count}(S) \leq v$	yes	no	weakly
$\text{count}(S) \geq v$	no	yes	weakly
$\text{sum}(S) \leq v$ ($a \in S, a \geq 0$)	yes	no	no
$\text{sum}(S) \geq v$ ($a \in S, a \geq 0$)	no	yes	no
$\text{range}(S) \leq v$	yes	no	no
$\text{range}(S) \geq v$	no	yes	no
$\text{avg}(S)\ \theta\ v, \theta \in \{=, \leq, \geq\}$	convertible	convertible	no
$\text{support}(S) \geq \xi$	yes	no	no
$\text{support}(S) \leq \xi$	no	yes	no

Summary

- Pattern Mining: A Road Map
- Coverage Pattern Mining
- Multilevel Pattern Mining
- Quantitative Pattern Mining
- Diverse Frequent Pattern Mining
- High Utility Pattern mining
- Mining Rare Patterns and Negative Patterns
- Constraint-Based Frequent Pattern Mining
- Summary

Summary: Pattern Mining: Advanced Methods (I)

- Mining Diverse Patterns
  - Mining Multiple-Level Associations
  - Mining Multi-Dimensional Associations
  - Mining Quantitative Associations
  - Mining Negative Correlations
  - Mining Compressed and Redundancy-Aware Patterns

- Constraint-Based Frequent Pattern Mining
  - Why Constraint-Based Mining?
  - Constrained Mining with Pattern Anti-Monotonicity
  - Constrained Mining with Pattern Monotonicity
  - Constrained Mining with Data Anti-Monotonicity
  - Constrained Mining with Succinct Constraints
  - Constrained Mining with Convertible Constraints
  - Handling Multiple Constraints
  - Constraint-Based Sequential-Pattern Mining

Summary: Pattern Mining: Advanced Methods (II)

- Sequential Pattern Mining
  - Sequential Pattern and Sequential Pattern Mining
  - GSP: Apriori-Based Sequential Pattern Mining
  - SPADE: Sequential Pattern Mining in Vertical Data Format
  - PrefixSpan: Sequential Pattern Mining by Pattern-Growth
  - CloSpan: Mining Closed Sequential Patterns
  - Constraint-Based Sequential Pattern Mining

- Graph Pattern Mining
  - Graph Pattern and Graph Pattern Mining
  - Apriori-Based Graph Pattern Mining Methods
  - gSpan: A Pattern-Growth-Based Method
- CloseGraph: Mining Closed Graph Patterns
  - Graph Pattern Applications
- Application I: Mining Software Copy-and-Paste Bugs
- Application II: Phrase Mining

References: Mining Diverse Patterns

- R. Srikant and R. Agrawal, “Mining generalized association rules”, VLDB'95
- Y. Aumann and Y. Lindell, “A Statistical Theory for Quantitative Association Rules”, KDD'99
- K. Wang, Y. He, J. Han, “Pushing Support Constraints Into Association Rules Mining”, IEEE Trans. Knowledge and Data Eng. 15(3): 642-658, 2003
- D. Xin, J. Han, X. Yan and H. Cheng, "On Compressing Frequent Patterns", Knowledge and Data Engineering, 60(1): 5-29, 2007
- D. Xin, H. Cheng, X. Yan, and J. Han, "Extracting Redundancy-Aware Top-K Patterns", KDD'06
- J. Han, H. Cheng, D. Xin, and X. Yan, "Frequent Pattern Mining: Current Status and Future Directions", Data Mining and Knowledge Discovery, 15(1): 55-86, 2007
- F. Zhu, X. Yan, J. Han, P. S. Yu, and H. Cheng, “Mining Colossal Frequent Patterns by Core Pattern Fusion”, ICDE'07

References: Constraint-Based Frequent Pattern Mining

- R. Srikant, Q. Vu, and R. Agrawal, “Mining association rules with item constraints”, KDD'97
- R. Ng, L.V.S. Lakshmanan, J. Han & A. Pang, “Exploratory mining and pruning optimizations of constrained association rules”, SIGMOD’98
- G. Grahne, L. Lakshmanan, and X. Wang, “Efficient mining of constrained correlated sets”, ICDE'00
- J. Pei, J. Han, and L. V. S. Lakshmanan, “Mining Frequent Itemsets with Convertible Constraints”, ICDE'01
- J. Pei, J. Han, and W. Wang, “Mining Sequential Patterns with Constraints in Large Databases”, CIKM'02
- F. Bonchi, F. Giannotti, A. Mazzanti, and D. Pedreschi, “ExAnte: Anticipated Data Reduction in Constrained Pattern Mining”, PKDD'03
- F. Zhu, X. Yan, J. Han, and P. S. Yu, “gPrune: A Constraint Pushing Framework for Graph Pattern Mining”, PAKDD'07

References: Sequential Pattern Mining

- R. Srikant and R. Agrawal, “Mining sequential patterns: Generalizations and performance improvements”, EDBT’96
- M. Zaki, “SPADE: An Efficient Algorithm for Mining Frequent Sequences”, Machine Learning, 2001
- J. Pei, J. Han, B. Mortazavi-Asl, J. Wang, H. Pinto, Q. Chen, U. Dayal, and M.-C. Hsu, "Mining Sequential Patterns by Pattern-Growth: The PrefixSpan Approach", IEEE TKDE, 16(10), 2004
- X. Yan, J. Han, and R. Afshar, “CloSpan: Mining Closed Sequential Patterns in Large Datasets”, SDM'03
- J. Pei, J. Han, and W. Wang, "Constraint-based sequential pattern mining: the pattern-growth methods", J. Int. Inf. Sys., 28(2), 2007
- M. N. Garofalakis, R. Rastogi, K. Shim: Mining Sequential Patterns with Regular Expression Constraints. IEEE Trans. Knowl. Data Eng. 14(3), 2002
- H. Mannila, H. Toivonen, and A. I. Verkamo, “Discovery of frequent episodes in event sequences”, Data Mining and Knowledge Discovery, 1997

References: Graph Pattern Mining

- C. Borgelt and M. R. Berthold, Mining molecular fragments: Finding relevant substructures of molecules, ICDM'02
- J. Huan, W. Wang, and J. Prins. Efficient mining of frequent subgraph in the presence of isomorphism, ICDM'03
- A. Inokuchi, T. Washio, and H. Motoda. An apriori-based algorithm for mining frequent substructures from graph data, PKDD'00
- M. Kuramochi and G. Karypis. Frequent subgraph discovery, ICDM'01
- S. Nijssen and J. Kok. A Quickstart in Frequent Structure Mining can Make a Difference. KDD'04
- N. Vanetik, E. Gudes, and S. E. Shimony. Computing frequent graph patterns from semistructured data, ICDM'02
- X. Yan and J. Han, gSpan: Graph-Based Substructure Pattern Mining, ICDM'02
- X. Yan and J. Han, CloseGraph: Mining Closed Frequent Graph Patterns, KDD'03
- X. Yan, P. S. Yu, J. Han, Graph Indexing: A Frequent Structure-based Approach, SIGMOD'04
- X. Yan, P. S. Yu, and J. Han, Substructure Similarity Search in Graph Databases, SIGMOD'05

References: Phrase Mining

- S. Bergsma, E. Pitler, D. Lin, Creating Robust Supervised Classifiers via Web-scale N-gram Data, ACL'2010
- D. M. Blei and J. D. Lafferty. Visualizing Topics with Multi-word Expressions. arXiv:0907.1013, 2009
- D.M. Blei, A. Y. Ng, M. I. Jordan, J. D. Lafferty, Latent Dirichlet Allocation. JMLR 2003
- M. Danilevsky, C. Wang, N. Desai, X. Ren, J. Guo, J. Han. Automatic Construction and Ranking of Topical Keyphrases on Collections of Short Documents. SDM'14
- A. El-Kishky, Y. Song, C. Wang, C. R. Voss, and J. Han. Scalable Topical Phrase Mining from Text Corpora. VLDB'15
- R. V. Lindsey, W. P. Headden, III, M. J. Stipicevic. A Phrase-Discovering Topic Model Using Hierarchical Pitman-Yor Processes. EMNLP-CoNLL'12.
- J. Liu, J. Shang, C. Wang, X. Ren, J. Han, Mining Quality Phrases from Massive Text Corpora. SIGMOD'15
- A. Parameswaran, H. Garcia-Molina, and A. Rajaraman. Towards the Web of Concepts: Extracting Concepts from Large Datasets. VLDB'10
- X. Wang, A. McCallum, X. Wei. Topical N-grams: Phrase and Topic Discovery, With and Application to Information Retrieval. ICDM'07
- J. Shang, J. Liu, M. Jiang, X. Ren, C. R Voss, J. Han, "Automated Phrase Mining from Massive Text Corpora", IEEE Transactions on Knowledge and Data Engineering, 30(10):1825-1837 (2018)

Classification: Basic Concepts
Class #16

P. Krishna Reddy, IIIT Hyderabad

Topics

- Introduction (1.5 hour): Definition, KDD framework, Issues in data mining.
- Data summarization (7.5 hrs): Data Types, Preprocessing, Characterization, Discrimination, data warehousing techniques (Multidimensional data model, Data warehousing architecture, Data cube computation and OLAP technology)
- Concepts and algorithms for mining patterns and associations (9 hours) (Frequent item-set generation, A priori and FP-growth algorithm, Evaluation of Association patterns) and preprocessing
- **Concepts and algorithms related to classification and regression (9hrs) (Overview, Decision tree induction, Over-fitting and under-fitting, Scalable decision tree algorithms, Bayesian Classification, Regression-based Prediction methods (9 hours)**
- Concepts and algorithms for clustering the data (9 hours) (Overview, Types of Data, K-means, Aglomerative clustering, Clustering algorithms (DBSCAN, BIRCH, CURE, ROCK, CHAMELEON)).
- Outlier analysis and future trends (graph mining, spatio-temporal mining). (3 hours)

Presentation Outline

- **Background**
- Basic concepts
- Decision Tree Induction
- Bayes Classification Methods
- Lazy Learners (k-nearest neighbours)
- Linear Classifiers
- Model Evaluation and Selection
- Techniques to Improve Accuracy
- Summary

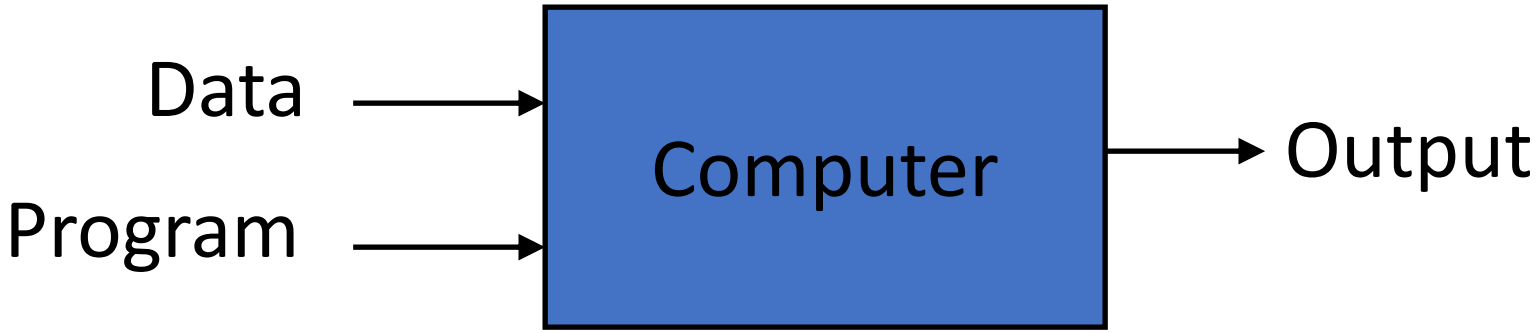
A simple question

- 1, 3, 5, 7, 9, …. What is the next number?
  - Ans: 11. ; Odd numbers, Or 2n+1
- 1, 3, 9, 19, 33, ... What is the next number?
  - Ans: 51; $2n^2+1$
- How do we solve such problems?
  - Find a pattern from the examples.
    - (function f(n) = 2n+1. Or model the data)
  - Use it to predict the next number (or solve the problem)
- How do we design a computational procedure?

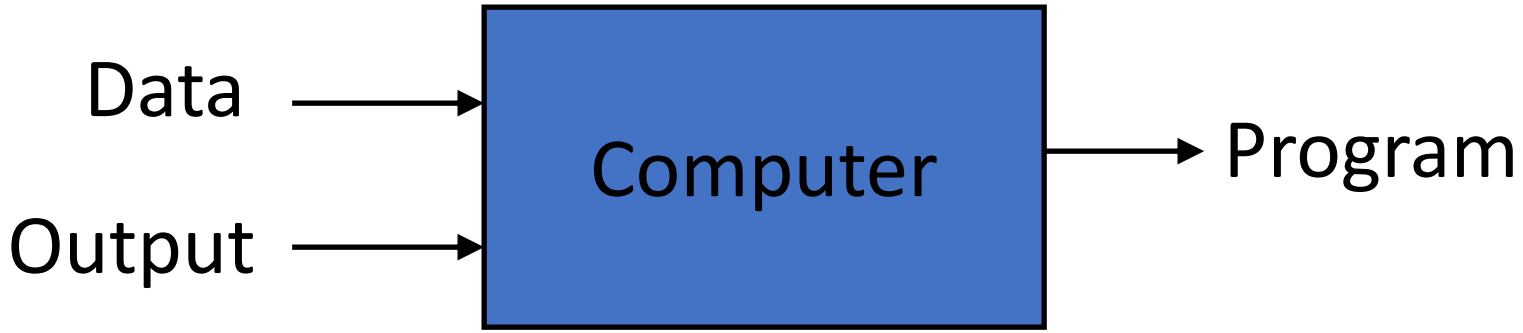
A simple question (cont.)

- We know: 1, 3, 9, 19, 33, ... What is the next number?
  - Ans: 51; $2n^2+1$
- 0.99, 3.02, 9.00, 18.98, 33.01, ... What next?
- Consider a series of 2D points
  - (1,3), (2,6), (3,9), (4,12), ...
  - What is the next point?
- When does the problem become difficult?
  - When numbers are “uncertain”. Noise in measurements
  - When numbers are not just “simple numbers”?

Traditional Programming



Machine Learning



The machine learning framework

- Apply a prediction function to a feature representation of the “sample” to get the desired output:

f() = “apple”

f() = “tomato”

f() = “cow”

Slide credit: L. Lazebnik

The machine learning/Classification framework

$$y = f(\mathbf{x})$$



- **Training:** given a *training set* of labeled examples $\{(\mathbf{x}_1,y_1), \ldots, (\mathbf{x}_N,y_N)\}$, estimate the prediction function f by minimizing the prediction error.

- **Testing:** apply f to a never before seen *test example* $\mathbf{x}$ and output the predicted value $y = f(\mathbf{x})$

Slide credit: L. Lazebnik

Steps

Training

Training Data



Training Labels

Features

Training

Learned model

Testing



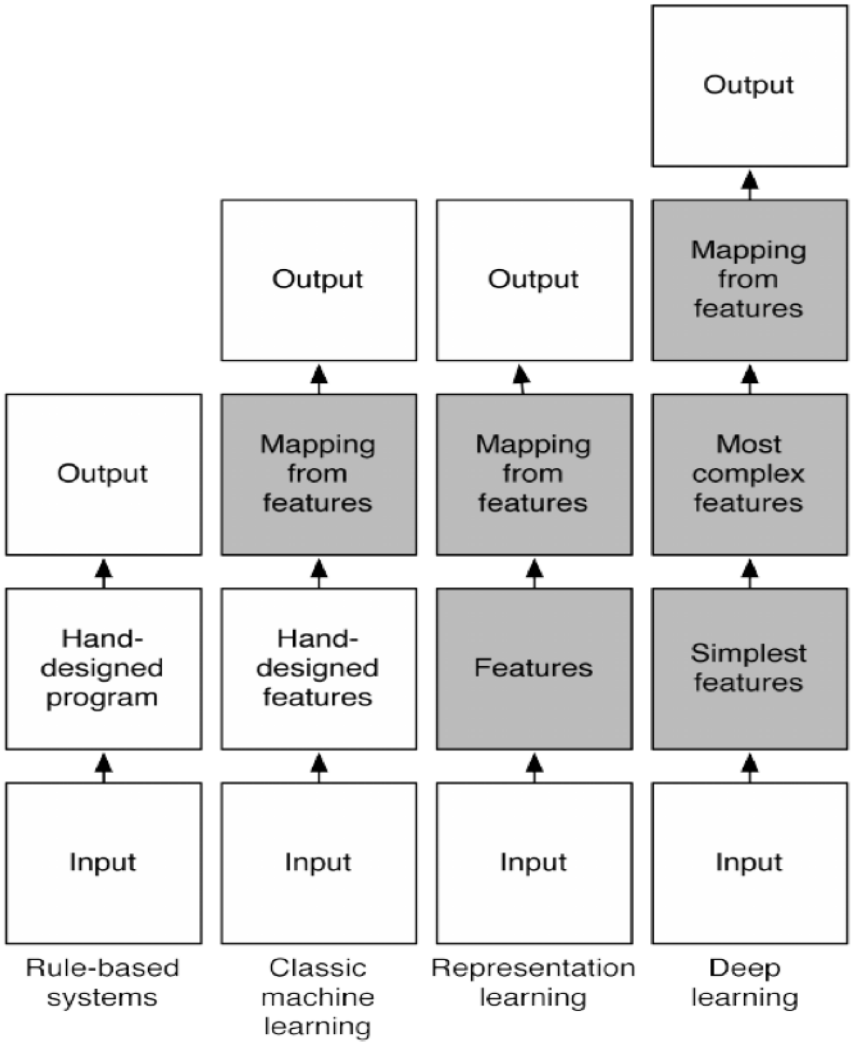
Test sample

Features

Learned model

Prediction

What is deep learning?



Y. Bengio et al, ``Deep Learning", MIT Press, 2015

Hand-crafted features, Representative Learning and Deep Learning

- "**Hand Crafted" features** refer to properties derived using various algorithms using the information present in the data itself.
- **Representation learning**, also known as feature learning, is a process that allows a machine to identify the most useful features or representations from raw data automatically. This process is crucial in machine learning because it can significantly improve the performance of learning algorithms.
- **Deep learning** is a method in artificial intelligence (AI) that teaches computers to process data in a way that is inspired by the human brain. Deep learning models can recognize complex patterns in pictures, text, sounds, and other data to produce accurate insights and predictions.

Presentation Outline

- Background
- **Basic concepts**
- Decision Tree Induction
- Bayes Classification Methods
- Lazy Learners (k-nearest neighbours)
- Linear Classifiers
- Model Evaluation and Selection
- Techniques to Improve Accuracy
- Summary

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.

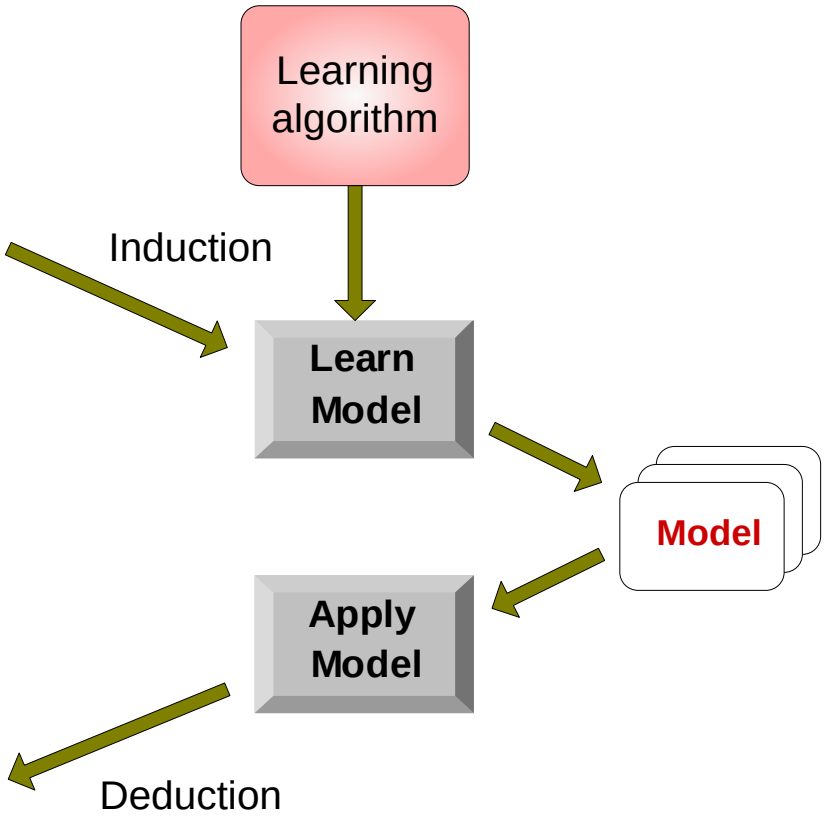
Illustrating Classification Task

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

Training Set

Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

Test Set



Supervised vs. Unsupervised Learning

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

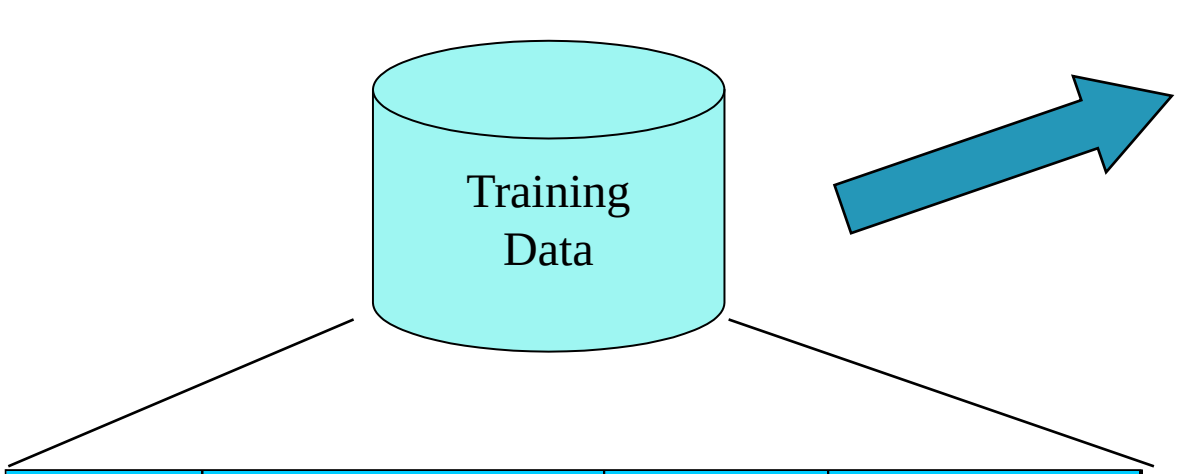
Classification vs. Prediction

- 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
- Prediction
  - 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

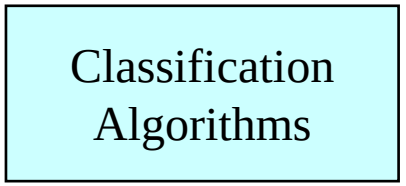
Classification—A Two-Step Process

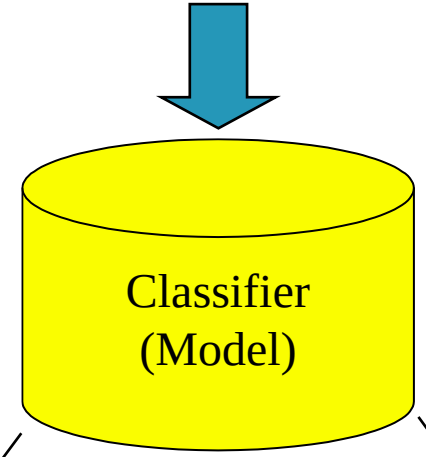
- Model construction: describing a set of predetermined classes
  - Each tuple/sample is assumed to belong to a predefined class, as determined by the class label attribute
  - The set of tuples used for model construction is training set
  - The model is represented as classification rules, decision trees, or mathematical formulae
- Model usage: for classifying future or unknown objects
  - Estimate accuracy of the model
    - The known label of test sample is compared with the classified result from the model
    - Accuracy rate is the percentage of test set samples that are correctly classified by the model
    - Test set is independent of training set, otherwise over-fitting will occur
  - If the accuracy is acceptable, use the model to classify data tuples whose class labels are not known

Process (1): Model Construction



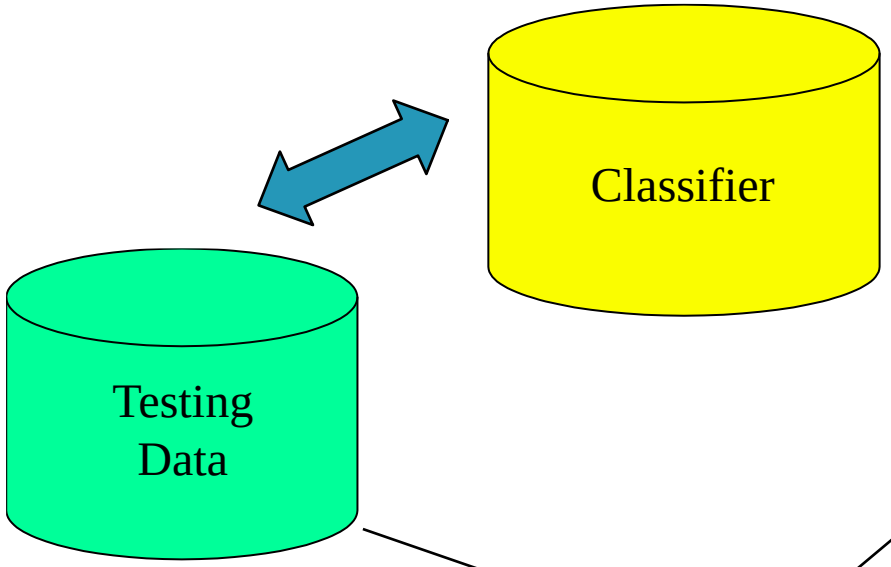
NAME	RANK	YEARS	TENURED
Mike	Assistant Prof	3	no
Mary	Assistant Prof	7	yes
Bill	Professor	2	yes
Jim	Associate Prof	7	yes
Dave	Assistant Prof	6	no
Anne	Associate Prof	3	no





IF rank = ‘professor’
OR years > 6
THEN tenured = ‘yes’

Process (2): Using the Model in Prediction



(Jeff, Professor, 4)

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

Tenured?

Yes

Classification Issues: Data Preparation

- Data cleaning
  - Preprocess data in order to reduce noise and handle missing values
- Relevance analysis (feature selection)
  - Remove the irrelevant or redundant attributes
- Data transformation
  - Generalize and/or normalize data

Issues: Evaluating Classification Methods

- Estimate accuracy of the model
  - The known label of test sample is compared with the classified result from the model
  - Accuracy rate is the percentage of test set samples that are correctly classified by the model
  - Test set is independent of training set (otherwise overfitting)
  - If the accuracy is acceptable, use the model to classify new data
- Note: If *the test set* is used to select models, it is called validation (test) set

Presentation Outline

- Background
- Basic concepts
- **Decision Tree Induction**
- Bayes Classification Methods
- Rule-Based Classification
- Model Evaluation and Selection
- Techniques to Improve Accuracy
- Summary

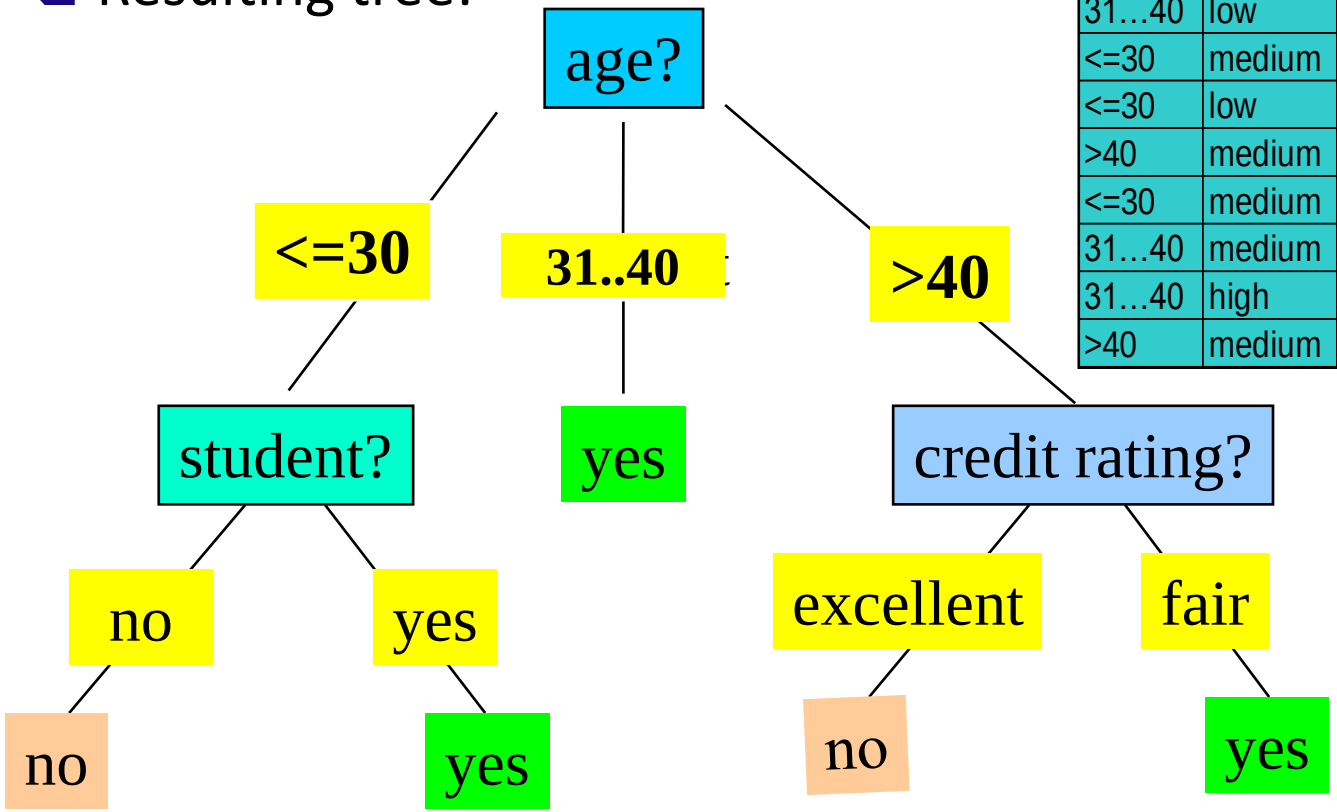
Decision tree induction (Class #17)

- **Decision tree induction** is the learning of decision trees from class-labeled training tuples.
- A **decision tree** is a flowchart-like tree structure
  - Each **internal node** (nonleaf node) denotes a test on an attribute,
  - Each **branch** represents an outcome of the test
  - Each **leaf node** (or terminal node) holds a class label.
  - **Root node** is the topmost node.

Decision Tree Induction: An Example

- Training data set: Buys_computer
- The data set follows an example of Quinlan's ID3
- Resulting tree:

age	income	student	credit_rating	buys_computer
<=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



Use of decision tree

- For a tuple X, the attribute values are tested against the decision tree.
- A path is traced from the root to leaf.
- Reason for popularity
  - Easy to construct, simple and fast
    - No domain knowledge is required
  - Easy to understand; interpretable
  - They can handle multidimensional data
  - Good accuracy
  - Several applications

History

- Introduced by J.Ross Quinlan
  - Developed ID3 algorithm (Iterative Dichotomiser)
- Quinlan also presented C4.5, which became a benchmark
- A CART algorithm is published by four researchers independently
  - IBM Intelligent Miner
- Three algorithms: ID3, C4.5, and CART adopt a greedy approach (non-backtracking)
  - Top-down, divide and conquer

Algorithm for Decision Tree Induction

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

- Conditions for stopping partitioning
  - All samples for a given node belong to the same class
  - There are no remaining attributes for further partitioning – majority voting is employed for classifying the leaf
  - There are no samples left

- Inputs: D, attribute list, and attribute selection method
- D is the complete set of training tuples and associated class lablels
- Attribute list: list of attributes describing the tuples

- Attribute selection method specifies the method to select the attributes.

Splitting possibilities

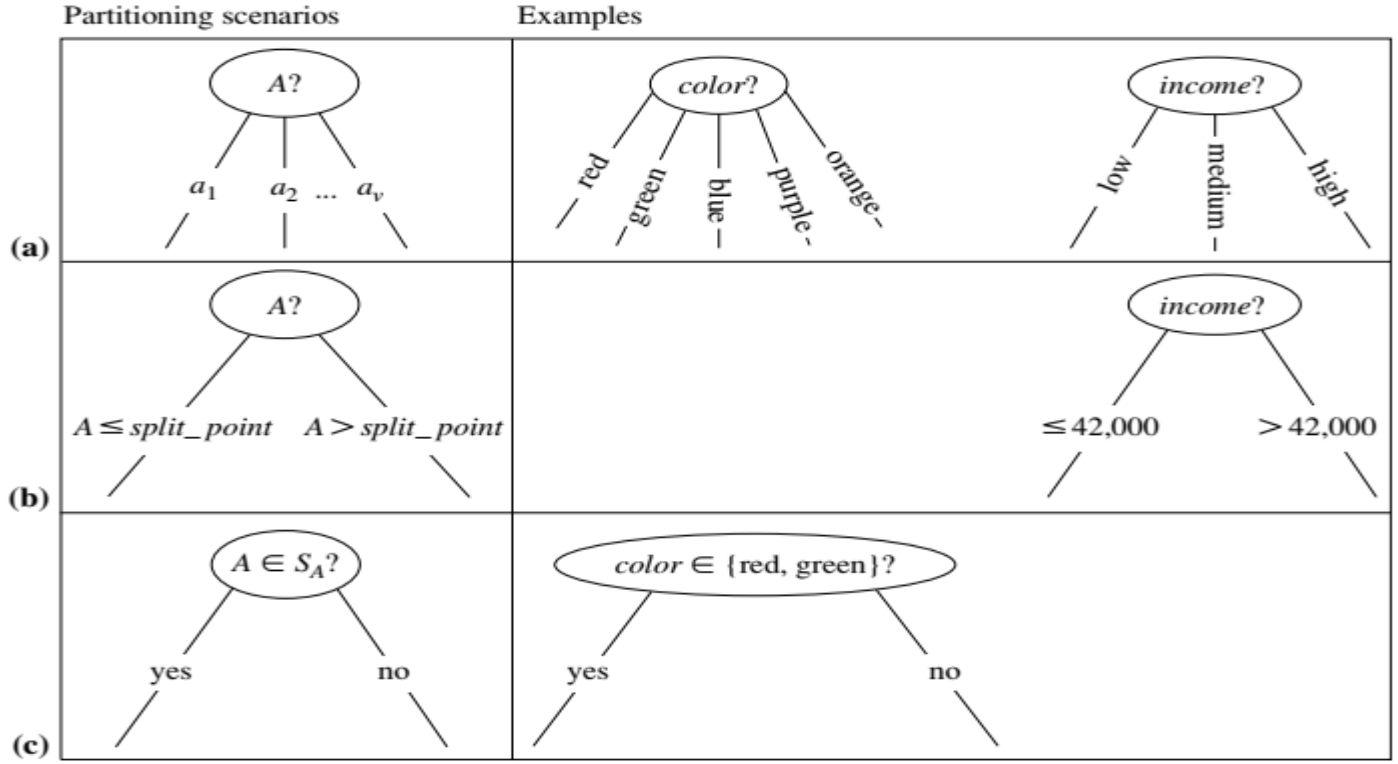
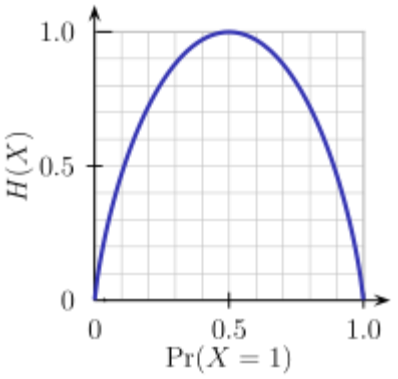


Figure 8.4 This figure shows three possibilities for partitioning tuples based on the splitting criterion, each with examples. Let A be the splitting attribute. (a) If A is discrete-valued, then one branch is grown for each known value of A. (b) If A is continuous-valued, then two branches are grown, corresponding to $A \leq split_point$ and $A > split_point$. (c) If A is discrete-valued and a binary tree must be produced, then the test is of the form $A \in S_A$, where S_A is the splitting subset for A.

Brief Review of Entropy

- Entropy (Information Theory)
  - A measure of uncertainty associated with a random variable
  - Calculation: For a discrete random variable *Y* taking *m* distinct values $\{y_1, \ldots, y_m\}$,
    - $H(Y) = -\sum_{i=1}^{m} p_i \log(p_i)$, where $p_i = P(Y = y_i)$
  - Interpretation:
    - Higher entropy => higher uncertainty
    - Lower entropy => lower uncertainty
- Conditional Entropy
  - $H(Y|X) = \sum_x p(x) H(Y|X = x)$



m = 2

About Shannon's entropy

- Consider transmission of a1, a2, a3, a4
  - Option 1: a1= 00, a2=01, a3=10, a4= 11
  - Option 2: a1=0, a2=10, a3=110, a4=111

(i) Option 1: If p(a1)=p(a2)=p(a3)=p(a4)=1/4

Expected code length= 2 bits.

Option 2: Expected code length=2.25 bits

(ii) If p(a1)= 1/2; p(a2)=1/4; p(a3)=1/8; p(a4)=1/8

Option 1: 2 bits (if we do not consider probabilities)

Option 2= 1.75 bits =(-1/2*log(1/2) -1/4*log(1/4)-1/8*log(1/8)-1/8*log(1/8)=1/2+2/4+ 3/8+3/8=1.75

(Equal to Shannon's entropy)

The idea is that *frequent letters should be coded with smaller lengths*.

The **Shannon entropy** *of* A *is*

$$H(A) = -\sum_{i=1}^{n} p_i \log_2 p_i.$$

pi= probability of event X

log(pi) is the amount of information of event x

Information content of event x with probability p=log(1/p(x))=-log(p(x))

Attribute Selection Measure: Information Gain (ID3/C4.5)

- Select the attribute with the highest information gain
- Let p_i be the probability that an arbitrary tuple in D belongs to class C_i, estimated by $|C_{i,D}|/|D|$
- Expected information (entropy) needed to classify a tuple in D:

$$Info(D) = -\sum_{i=1}^{m} p_i \log_2(p_i)$$

- Information needed (after using A to split D into v partitions) to classify D:

$$Info_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times Info(D_j)$$

- Information gained by branching on attribute A

$$Gain(A) = Info(D) - Info_A(D)$$

Attribute Selection: Information Gain

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

$$Info(D) = I(9,5) = -\frac{9}{14}\log_2(\frac{9}{14}) - \frac{5}{14}\log_2(\frac{5}{14}) = 0.940$$

age	p_i	n_i	$I(p_i, n_i)$
<=30	2	3	0.971
31…40	4	0	0
>40	3	2	0.971

$$Info_{age}(D) = \frac{5}{14}I(2,3) + \frac{4}{14}I(4,0) + \frac{5}{14}I(3,2) = 0.694$$

$\frac{5}{14}I(2,3)$ means "age <=30" has 5 out of 14 samples, with 2 yes'es and 3 no's. Hence

age	income	student	credit_rating	buys_computer
<=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

$$Gain(age) = Info(D) - Info_{age}(D) = 0.246$$

Similarly,

$$Gain(income) = 0.029$$
$$Gain(student) = 0.151$$
$$Gain(credit_rating) = 0.048$$

Computing Information-Gain for Continuous-Valued Attributes

- Let attribute A be a continuous-valued attribute
- Must determine the *best split point* for A
  - Sort the value A in increasing order
  - Typically, the midpoint between each pair of adjacent values is considered as a possible *split point*
    - $(a_i+a_{i+1})/2$ is the midpoint between the values of a_i and a_{i+1}
  - The point with the *minimum expected information requirement* for A is selected as the split-point for A
- Split:
  - D1 is the set of tuples in D satisfying A ≤ split-point, and D2 is the set of tuples in D satisfying A > split-point

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.

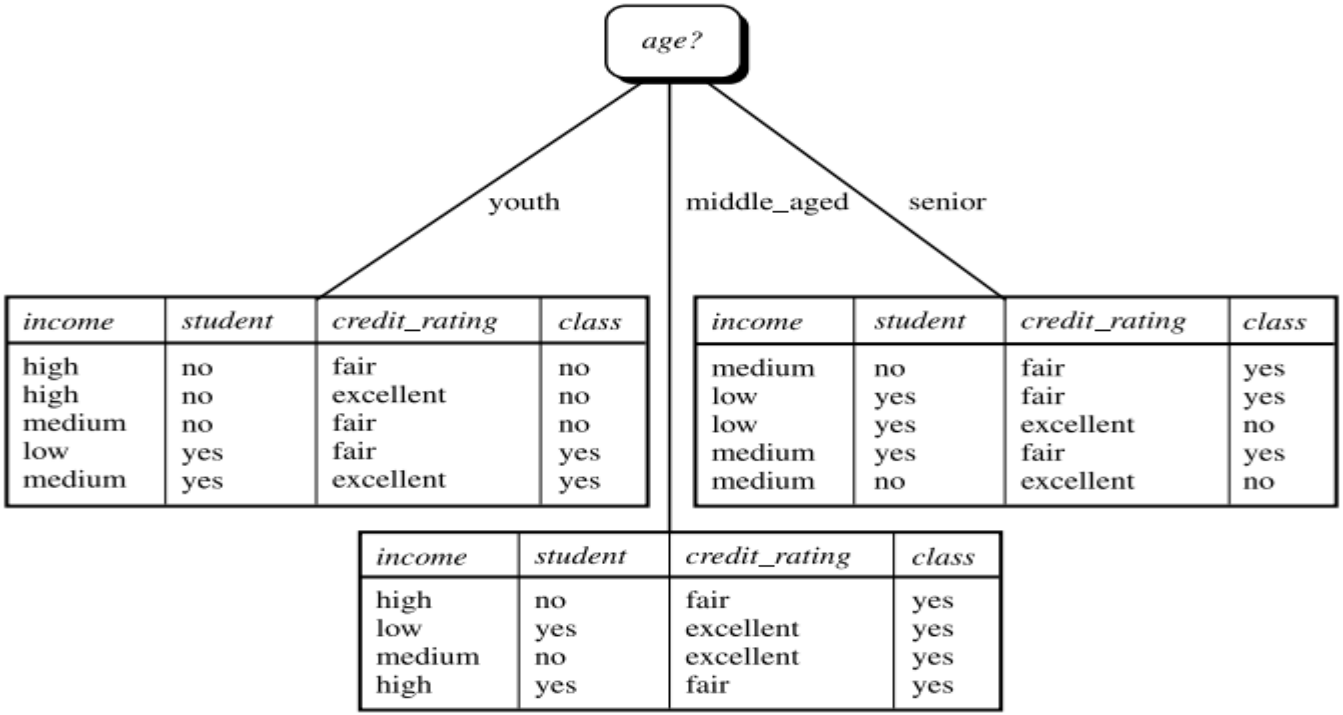
$$SplitInfo_{income}(D) = -\frac{4}{14} \times \log_2\left(\frac{4}{14}\right) - \frac{6}{14} \times \log_2\left(\frac{6}{14}\right) - \frac{4}{14} \times \log_2\left(\frac{4}{14}\right) = 1.557$$

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

87/1418

Table 8.1 Class-Labeled Training Tuples from the *AllElectronics* Customer Database

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



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

- Reduction in Impurity:

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

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

- 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

$$gini_{income \in \{low,medium\}}(D) = \left(\frac{10}{14}\right) Gini(D_1) + \left(\frac{4}{14}\right) Gini(D_2)$$

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

$$= 0.443$$

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

$Gini_{\{low,high\}}$ 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

Other Attribute Selection Measures

- CHAID: a popular decision tree algorithm, measure based on χ^2 test for independence
- G-statistic: has a close approximation to χ^2 distribution
- MDL (Minimal Description Length) principle (i.e., the simplest solution is preferred):
  - The best tree as the one that requires the fewest # of bits to both (1) encode the tree, and (2) encode the exceptions to the tree
- Multivariate splits (partition based on multiple variable combinations)
  - CART: finds multivariate splits based on a linear comb. of attrs.
- Which attribute selection measure is the best?
  - Most give good results, none is significantly superior than others

Enhancements to Basic Decision Tree Induction

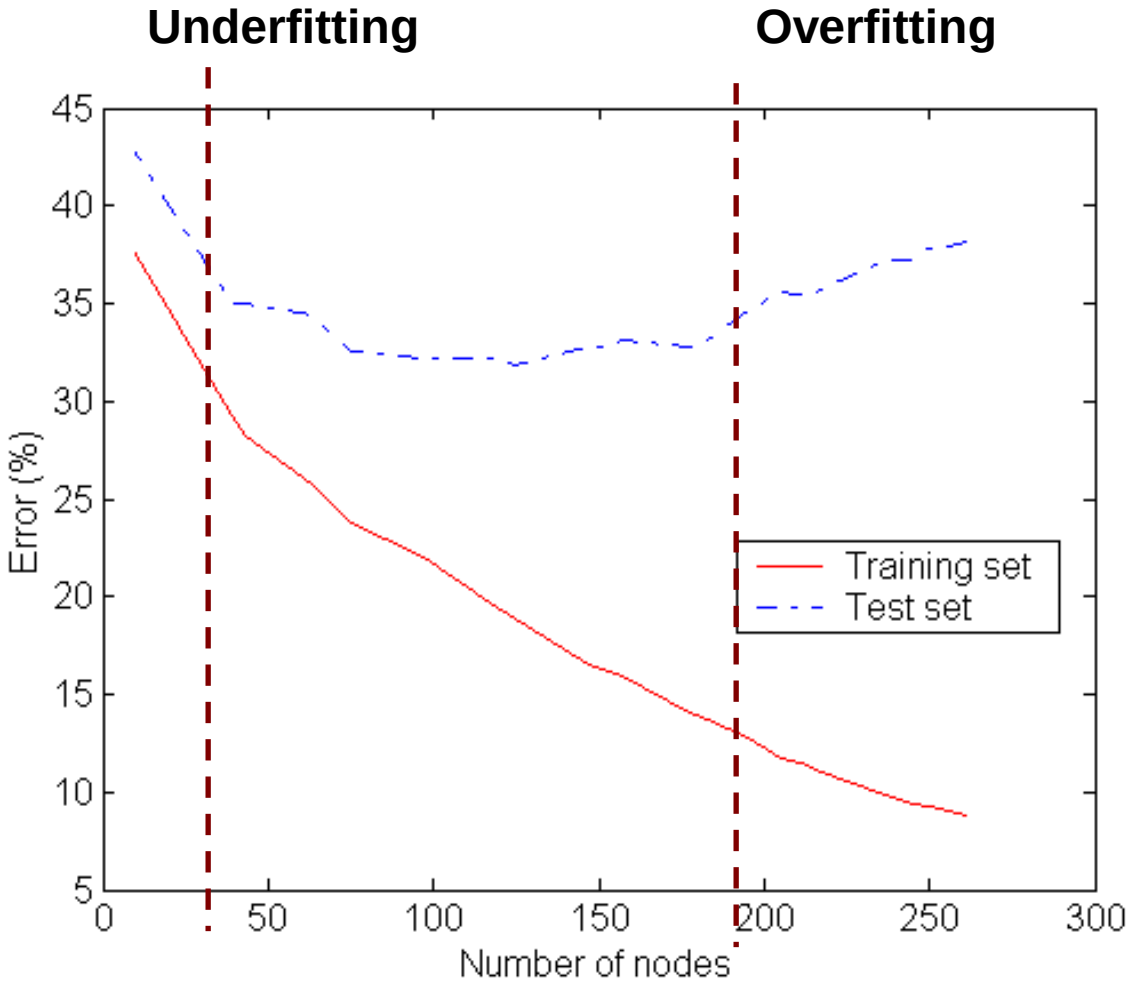
- Allow for **continuous-valued attributes**
  - Dynamically define new discrete-valued attributes that partition the continuous attribute value into a discrete set of intervals
- Handle **missing attribute values**
  - Assign the most common value of the attribute
  - Assign probability to each of the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

Underfitting and Overfitting

- Underfitting and overfitting are two problems that will adversely affect the accuracy and usefulness of the model.

- Classification requires three data sets
  - Training data set
    - This is the data that are analyzed to first produce a model.
  - Test data set
    - Once a model is developed it is then tested on the evaluation data set. The second data set is an evaluation data set important for refining the model if necessary, and to get a sense of how well the model will perform on subsequent data that it will be applied on.
  - Data on which classification is applied

Underfitting and Overfitting



Underfitting: when model is too simple, both training and test errors are large

Overfitting: once the tree becomes too large, its test error rates begins to increase even though its training error rate continues to decrease.

Underfitting

- **Underfitting** is a situation where training and test error rates of the model are large when the size of the tree is very small.
- Underfitting occurs because the model has yet to learn the true structure of the data
- It performs poorly on both training and test sets
- As number of nodes in the decision tree increases the tree will have a fewer training and test errors.
- However, once the tree becomes too large, its test error rates begins to increase even though its training error rate continues to decrease. It is called **overfitting.**

Underfitting

- Underfitting refers to a model that is too general and fails to find interesting patterns in the data.
  - This can result from not including important variables as inputs during the model building process.
  - In terms of a loan application scenario, the analyst or model-builder may include annual salary as an important factor, but may exclude information about the applicant's job, which may turn out to be important.
  - For example, jobs that are seasonal (swimming pool maintenance, landscaping, etc.) may affect the person's ability to submit monthly payments during the times when work is slower -- information that is not reflected by their annual salary.
  - This might suggest that as much information as possible should be included as inputs during the model building process to avoid underfitting.

More on Underfitting

- Training error can be reduced by increasing the model complexity.
  - Leaf-nodes can be expanded until it perfectly fits the training data.
  - As a result the test error may be large because the tree may fit some of the noise points in the data.
  - Such nodes degrade the performance as they do not generalize well to the test examples.

Overfitting

- Overfitting: once the tree becomes too large, its test error rates begins to increase even though its training error rate continues to decrease.
- Including as much information as possible to develop a model can lead to the problem of overfitting.
- Overfitting refers to models that are too specific, or too sensitive to the particulars of the data (the training set) used to build the model.
- This can be due to having too many variables used as inputs and/or a non-representative training set.
  - In the loan application scenario, if in the training set, many people that defaulted on their loans happened to be named "Smith" (a popular name), then the model (e.g. a decision tree) may decide that if the applicant's last name is "Smith", then deny the loan.
  - In refining the model, perhaps last name should not serve as an input. More importantly, the characteristics of the data used to build the model have to be representative of the data at large -- it's unlikely that people named Smith have a disproportionately high rate of defaulting on loans.

Overfitting

- Overfitting can be corrected using the evaluation data set. If accurate performance on the training set is due to particular characteristics in that data (e.g. person's last name), then performance will be poor on the evaluation set as long as the evaluation set does not share these idiosyncrasies.
- Refining the model (e.g. by pruning the decision tree) involves setting performance to be generally equivalent on both the training and evaluation data sets.
- This will generally give the analyst the idea of how well the model may perform on "real" data.

Overfitting Examples

- Overfitting can still occur despite the use of a training and evaluation data set.

- This can be the result of poorly creating the training and evaluation sets -- so that neither is **representative of subsequent data that the model will be applied to.**

- For example, of predicting stock market performance.
- One year's worth of data were partitioned into equal sized training and evaluation sets. A quantitative model was developed based on the training set -- and its predictive power on this dataset was impressive. It was equally impressive on the evaluation dataset, apparently requiring no refinement. But when applied to the next year's data it performed miserably despite the fact that there were not significant events related to the stock market. The reason for this is had to do with how the training and evaluation data sets were created. The training and evaluation sets were based on the daily closing values of alternate days. Day 1's close was assigned to the training set, day 2 to the evaluation set, day 3 to the training set, day 4 to the evaluation set, and so forth. As a result the overfitting that occurred when the model picked up factors tied to the temporal fluctuations in the stock market's closing values were carried over into the evaluation set as well.

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
  - Poor accuracy for unseen samples
- Two approaches to avoid overfitting
  - Prepruning: *Halt tree construction early*-do not split a node if this would result in the goodness measure falling below a threshold
    - Difficult to choose an appropriate threshold
  - Postpruning: *Remove branches* from a "fully grown" tree—get a sequence of progressively pruned trees
    - Use a set of data different from the training data to decide which is the "best pruned tree"

Classification in Large Databases

- Classification—a classical problem extensively studied by statisticians and machine learning researchers
- Scalability: Classifying data sets with millions of examples and hundreds of attributes with reasonable speed
- Why is decision tree induction popular?
  - relatively faster learning speed (than other classification methods)
  - convertible to simple and easy to understand classification rules
  - can use SQL queries for accessing databases
  - comparable classification accuracy with other methods
- RainForest (VLDB'98 — Gehrke, Ramakrishnan & Ganti)
  - Builds an AVC-list (attribute, value, class label)

Scalability Framework for RainForest

- Separates the scalability aspects from the criteria that determine the quality of the tree
- Builds an AVC-list**: AVC (Attribute, Value, Class_label)**
- **AVC-set** (of an attribute *X*)
  - Projection of training dataset onto the attribute *X* and class label where counts of individual class label are aggregated
- **AVC-group** (of a node *n*)
  - Set of AVC-sets of all predictor attributes at the node *n*

Rainforest: Training Set and Its AVC Sets

Training Examples

age	income	student	credit_rating	com
<=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

AVC-set on *Age*

Age	Buy_Computer	
	yes	no
<=30	2	3
31..40	4	0
>40	3	2

AVC-set on *income*

income	Buy_Computer	
	yes	no
high	2	2
medium	4	2
low	3	1

AVC-set on *Student*

student	Buy_Computer	
	yes	no
yes	6	1
no	3	4

AVC-set on *credit_rating*

Credit rating	Buy_Computer	
	yes	no
fair	6	2
excellent	3	3

BOAT (Bootstrapped Optimistic Algorithm for Tree Construction)

- Use a statistical technique called *bootstrapping* to create several smaller samples (subsets), each fits in memory
- Each subset is used to create a tree, resulting in several trees
- These trees are examined and used to construct a new tree *T'*
  - It turns out that *T'* is very close to the tree that would be generated using the whole data set together
- Adv: requires only two scans of DB, an incremental alg.

Presentation Outline (Class #18)

- Background
- Basic concepts
- Decision Tree Induction
- **Bayes Classification Methods**
- Rule-Based Classification
- Model Evaluation and Selection
- Techniques to Improve Accuracy
- Summary

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

Derivation of Bays' theorem

- Definition: The conditional probability that A is true, given that B, with the notion P(A/B) (read "probability of A given B). Here, P(A$\wedge$B) is probability of the joint intersection of events *A* and *B*
  - P(A/B)= P(A $\wedge$ B)/P(B)
  - Or P(A $\wedge$ B)= P(A/B) * P(B)
- Bays theorem
  - P(A $\wedge$ B)= P(B $\wedge$ A)
  - P(B$\wedge$ A)=P(B/A) * P(A)
- So, we have P(A/B)=P(B/A)*P(A) / P(B)

- Let H be some hypothesis, such as the data sample X belongs to a specified class C. For this we want to determine P(H/X), the probability that the hypothesis H holds given the observed sample X.
  - P(H/X)=P(X/H)*P(H)/P(X)

Examples of conditional probability

- Example 1: Suppose you are drawing three marbles—red, blue, and green—from a bag. Each marble has an equal chance of being drawn. What is the conditional probability of drawing the red marble after already drawing the blue one?
  - First, the probability of drawing a blue marble is about 33% because it is one possible outcome out of three. Assuming this first event occurs, there will be two marbles remaining, with each having a 50% chance of being drawn. So the chance of drawing a blue marble after already drawing a red marble would be about 16.5% (33% x 50%).
- Example 2: Consider that a fair die has been rolled and you are asked to give the probability that it was a five. There are six equally likely outcomes, so your answer is 1/6.
  - But imagine if before you answer, you get extra information that the number rolled was odd. Since there are only three odd numbers that are possible, one of which is five, you would certainly revise your estimate for the likelihood that a five was rolled from 1/6 to 1/3.
- Example 3: Suppose a student is applying for admission to a university and hopes to receive an academic scholarship. The school to which they are applying accepts 100 of every 1,000 applicants (10%) and awards academic scholarships to 10 of every 500 students who are accepted (2%). Of the scholarship recipients, 50% of them also receive university stipends for books, meals, and housing.
  - For the students, the chance of them being accepted and then receiving a scholarship is .2% (.1 x .02). The chance of them being accepted, receiving the scholarship, then also receiving a stipend for books, etc. is .1% (.1 x .02 x .5).

Bayesian Classification: Simple Overview

- "The essence of the Bayesian approach is to provide a mathematical rule explaining how you should change your existing beliefs in the light of new evidence.
- In other words, it allows scientists to combine new data with their existing knowledge or expertise.

- The canonical example is to imagine that a precocious newborn observes his first sunset, and wonders whether the sun will rise again or not. He assigns equal prior probabilities to both possible outcomes, and represents this by placing one white and one black marble into a bag. The following day, when the sun rises, the child places another white marble in the bag. The probability that a marble plucked randomly from the bag will be white (ie, the child's degree of belief in future sunrises) has thus gone from a half to two-thirds. After sunrise the next day, the child adds another white marble, and the probability (and thus the degree of belief) goes from two-thirds to three-quarters. And so on. Gradually, the initial belief that the sun is just as likely as not to rise each morning is modified to become a near-certainty that the sun will always rise."

Bayesian Classification: Simple overview

- Suppose your data consist of fruits, described by their color and shape. Bayesian classifiers operate by saying "If you see a fruit that is red and round, which type of fruit is it most likely to be, based on the observed data sample? In future, classify red and round fruit as that type of fruit.“
- A difficulty arises when you have more than a few variables and classes -- you would require an enormous number of observations (records) to estimate these probabilities.
- Naive Baye’s classification gets around this problem by not requiring that you have lots of observations for each possible combination of the variables. Rather, the variables are assumed to be independent of one another and, therefore the probability that a fruit that is red, round, firm, 3" in diameter, etc. will be an apple can be calculated from the independent probabilities that a fruit is red, that it is round, that it is firm, that is 3" in diameter, etc.
- In other words, Naive Bayes classifiers assume that the effect of an variable value on a given class is independent of the values of other variable. This assumption is called class conditional independence. It is made to simplify the computation and in this sense considered to be naïve.

Bayesian Classification: Simple overview…

- This assumption is a fairly strong assumption and is often not applicable. However, bias in estimating probabilities often may not make a difference in practice -- it is the order of the probabilities, not their exact values, that determine the classifications.
- Studies comparing classification algorithms have found the Naïve Bayesian classifier to be comparable in performance with classification trees and with neural network classifiers. They have also exhibited high accuracy and speed when applied to large databases.

Bays Theorem

Let X be the data record (case) whose class label is unknown. Let H be some hypothesis, such as "data record X belongs to a specified class C.“

For classification, we want to determine P (H|X) – the probability that the hypothesis H holds, given the observed data record X.

P (H|X) is the posterior probability of H conditioned on X. For example, the probability that a fruit is an apple, given the condition that it is red and round.

In contrast, P(H) is the prior probability, or a priori probability, of H. In this example P(H) is the probability that any given data record is an apple, regardless of how the data record looks.

The posterior probability, P (H|X), is based on more information (such as background knowledge) than the prior probability, P(H), which is independent of X.

Bayesian Classification: Simple introduction…

Similarly, P (X|H) is posterior probability of X conditioned on H. That is, it is the probability that X is red and round given that we know that it is true that X is an apple.

P(X) is the prior probability of X, i.e., it is the probability that a data record from our set of fruits is red and round.

Bayes theorem is useful in that it provides a way of calculating the posterior probability, P(H|X), from P(H), P(X), and P(X|H). Bayes theorem is

$$P(H|X) = P(X|H)\, P(H) / P(X)$$

Bayes Classifier

- A probabilistic framework for solving classification problems
- Conditional Probability:

$$P(C \mid A) = \frac{P(A,C)}{P(A)}$$

$$P(A \mid C) = \frac{P(A,C)}{P(C)}$$

- Bayes theorem:

$$P(C \mid A) = \frac{P(A \mid C)P(C)}{P(A)}$$

Example of Bayes Theorem

- Given:
  - A doctor knows that meningitis causes stiff neck 50% of the time
  - Prior probability of any patient having meningitis is 1/50,000
  - Prior probability of any patient having stiff neck is 1/20

- If a patient has a stiff neck, what's the probability he/she has meningitis?

$$P(M \mid S) = \frac{P(S \mid M)P(M)}{P(S)} = \frac{0.5 \times 1/50000}{1/20} = 0.0002$$

Bayesian Classifiers

- Consider each attribute and class label as random variables

- Given a record with attributes (A_1, A_2,…,A_n)
  - Goal is to predict class C
  - Specifically, we want to find the value of C that maximizes P(C| A_1, A_2,…,A_n)

- Can we estimate P(C| A_1, A_2,…,A_n) directly from data?

Bayesian Classifiers

- Approach:
  - compute the posterior probability $P(C \mid A_1, A_2, \ldots, A_n)$ for all values of C using the Bayes theorem

$$P(C \mid A_1 A_2 \ldots A_n) = \frac{P(A_1 A_2 \ldots A_n \mid C) P(C)}{P(A_1 A_2 \ldots A_n)}$$

  - Choose value of C that maximizes
    $P(C \mid A_1, A_2, \ldots, A_n)$

  - Equivalent to choosing value of C that maximizes
    $P(A_1, A_2, \ldots, A_n \mid C)\, P(C)$

- How to estimate $P(A_1, A_2, \ldots, A_n \mid C)$?

Naïve Bayes Classifier

- Assume independence among attributes A_i when class is given:
  - $P(A_1, A_2, \ldots, A_n \mid C) = P(A_1 \mid C_j)\, P(A_2 \mid C_j) \ldots P(A_n \mid C_j)$
  - Can estimate $P(A_i \mid C_j)$ for all A_i and C_j.
  - New point is classified to C_j if $P(C_j) \prod P(A_i \mid C_j)$ is maximal.

How to Estimate Probabilities from Data?

Tid	Refund	Marital Status	Taxable Income	Evade
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

- Class: $P(C) = N_c/N$
  - e.g., P(No) = 7/10,
    P(Yes) = 3/10

- For discrete attributes:

$$P(A_i \mid C_k) = |A_{ik}| / N_c$$

  - where $|A_{ik}|$ is number of instances having attribute A_i and belongs to class C_k
  - Examples:

    P(Status=Married|No) = 4/7
    P(Refund=Yes|Yes)=0

How to Estimate Probabilities from Data?

- For continuous attributes:
  - Discretize the range into bins
    - one ordinal attribute per bin
    - violates independence assumption
  - Two-way split: $(A < v)$ or $(A > v)$
    - choose only one of the two splits as new attribute
  - Probability density estimation:
    - Assume attribute follows a normal distribution
    - Use data to estimate parameters of distribution (e.g., mean and standard deviation)
    - Once probability distribution is known, can use it to estimate the conditional probability $P(A_i|c)$

How to Estimate Probabilities from Data?

Tid	Refund	Marital Status	Taxable Income	Evade
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

- Normal distribution:

$$P(A_i \mid c_j) = \frac{1}{\sqrt{2\pi\sigma_{ij}^2}} e^{-\frac{(A_i-\mu_{ij})^2}{2\sigma_{ij}^2}}$$

  - One for each (A_i, c_i) pair

- For (Income, Class=No):
  - If Class=No
    - sample mean = 110
    - sample variance = 2975

$$P(Income = 120 \mid No) = \frac{1}{\sqrt{2\pi}(54.54)} e^{-\frac{(120-110)^2}{2(2975)}} = 0.0072$$

Example of Naïve Bayes Classifier

Given a Test Record:

$$X = (\text{Refund} = \text{No}, \text{Married}, \text{Income} = 120\text{K})$$

naive Bayes Classifier:

P(Refund=Yes|No) = 3/7
P(Refund=No|No) = 4/7
P(Refund=Yes|Yes) = 0
P(Refund=No|Yes) = 1
P(Marital Status=Single|No) = 2/7
P(Marital Status=Divorced|No)=1/7
P(Marital Status=Married|No) = 4/7
P(Marital Status=Single|Yes) = 2/7
P(Marital Status=Divorced|Yes)=1/7
P(Marital Status=Married|Yes) = 0

For taxable income:
If class=No: sample mean=110
sample variance=2975
If class=Yes: sample mean=90
sample variance=25

- P(X|Class=No) = P(Refund=No|Class=No)
  × P(Married| Class=No)
  × P(Income=120K| Class=No)
  = 4/7 × 4/7 × 0.0072 = 0.0024

- P(X|Class=Yes) = P(Refund=No| Class=Yes)
  × P(Married| Class=Yes)
  × P(Income=120K| Class=Yes)
  = 1 × 0 × 1.2 × 10^{-9} = 0

Since P(X|No)P(No) > P(X|Yes)P(Yes)

Therefore P(No|X) > P(Yes|X)

=> Class = No

Naïve Bayes Classifier

- If one of the conditional probability is zero, then the entire expression becomes zero
- Probability estimation:

$$\text{Original} : P(A_i \mid C) = \frac{N_{ic}}{N_c}$$

$$\text{Laplace} : P(A_i \mid C) = \frac{N_{ic} + 1}{N_c + c}$$

$$\text{m-estimate} : P(A_i \mid C) = \frac{N_{ic} + mp}{N_c + m}$$

c: number of classes

p: prior probability

m: parameter

Naïve Bayes Classifier: Example 1

Name	Give Birth	Can Fly	Live in Water	Have Legs	Class
human	yes	no	no	yes	mammals
python	no	no	no	no	non-mammals
salmon	no	no	yes	no	non-mammals
whale	yes	no	yes	no	mammals
frog	no	no	sometimes	yes	non-mammals
komodo	no	no	no	yes	non-mammals
bat	yes	yes	no	yes	mammals
pigeon	no	yes	no	yes	non-mammals
cat	yes	no	no	yes	mammals
leopard shark	yes	no	yes	no	non-mammals
turtle	no	no	sometimes	yes	non-mammals
penguin	no	no	sometimes	yes	non-mammals
porcupine	yes	no	no	yes	mammals
eel	no	no	yes	no	non-mammals
salamander	no	no	sometimes	yes	non-mammals
gila monster	no	no	no	yes	non-mammals
platypus	no	no	no	yes	mammals
owl	no	yes	no	yes	non-mammals
dolphin	yes	no	yes	no	mammals
eagle	no	yes	no	yes	non-mammals

A: attributes

M: mammals

N: non-mammals

$$P(A \mid M) = \frac{6}{7} \times \frac{6}{7} \times \frac{2}{7} \times \frac{2}{7} = 0.06$$

$$P(A \mid N) = \frac{1}{13} \times \frac{10}{13} \times \frac{3}{13} \times \frac{4}{13} = 0.0042$$

$$P(A \mid M)P(M) = 0.06 \times \frac{7}{20} = 0.021$$

$$P(A \mid N)P(N) = 0.004 \times \frac{13}{20} = 0.0027$$

Give Birth	Can Fly	Live in Water	Have Legs	Class
yes	no	yes	no	?

P(A|M)P(M) > P(A|N)P(N)

=> Mammals

Naïve Bayes Classifier: Example 2: estimating $P(x_i|C)$

Outlook	Temperature	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

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

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

Example 2….

- Classify an unseen sample X = <rain, hot, high, false>

- P(X|p)·P(p) =
  P(rain|p)·P(hot|p)·P(high|p)·P(false|p)·P(p) =
  3/9·2/9·3/9·6/9·9/14 = 0.010582
- P(X|n)·P(n) =
  P(rain|n)·P(hot|n)·P(high|n)·P(false|n)·P(n) =
  2/5·2/5·4/5·2/5·5/14 = 0.018286

- Sample X is classified in class n (don't play)

Naïve Bayes (Summary)

- Robust to isolated noise points
- Handle missing values by ignoring the instance during probability estimate calculations
- Robust to irrelevant attributes
- Independence assumption may not hold for some attributes
  - … makes computation possible
  - … yields optimal classifiers when satisfied
  - … but is seldom satisfied in practice, as attributes (variables) are often correlated.
  - Attempts to overcome this limitation:
    - Bayesian networks, that combine Bayesian reasoning with causal relationships between attributes
    - Decision trees, that reason on one attribute at the time, considering most important attributes first

Presentation Outline

- Background
- Basic concepts
- Decision Tree Induction
- Bayes Classification Methods
- **Lazy Learners (k-nearest neighbours)**
- Linear Classifiers
- Model Evaluation and Selection
- Techniques to Improve Accuracy
- Summary

Lazy vs. Eager Learning

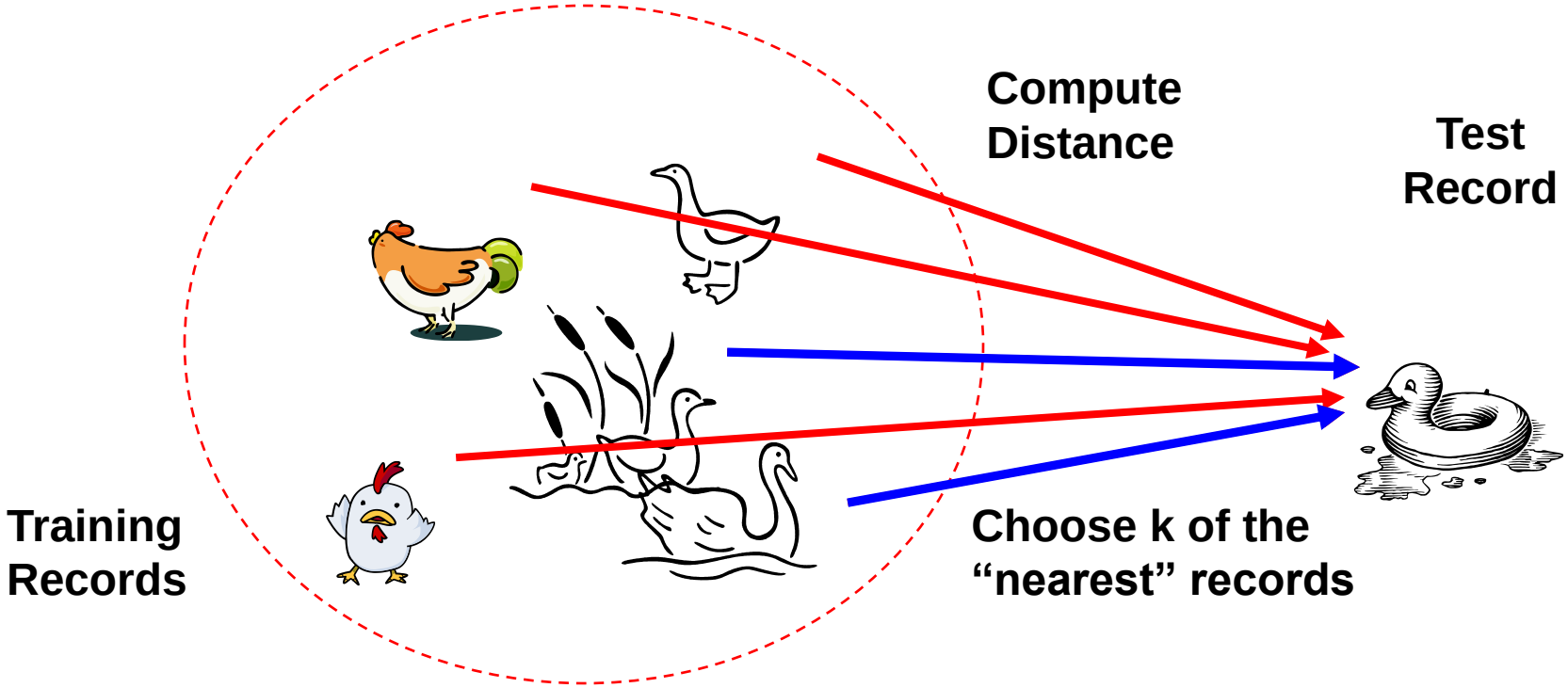
- Lazy vs. eager learning
  - **Lazy learning** (e.g., instance-based learning): Simply stores training data (or only minor processing) and waits until it is given a test tuple
  - **Eager learning** (the above discussed methods): Given a set of training tuples, constructs a classification model before receiving new (e.g., test) data to classify
- Lazy: less time in training but more time in predicting
- Accuracy
  - Lazy method effectively uses a richer hypothesis space since it uses many local linear functions to form an implicit global approximation to the target function
  - Eager: must commit to a single hypothesis that covers the entire instance space

Lazy Learner: Instance-Based Methods

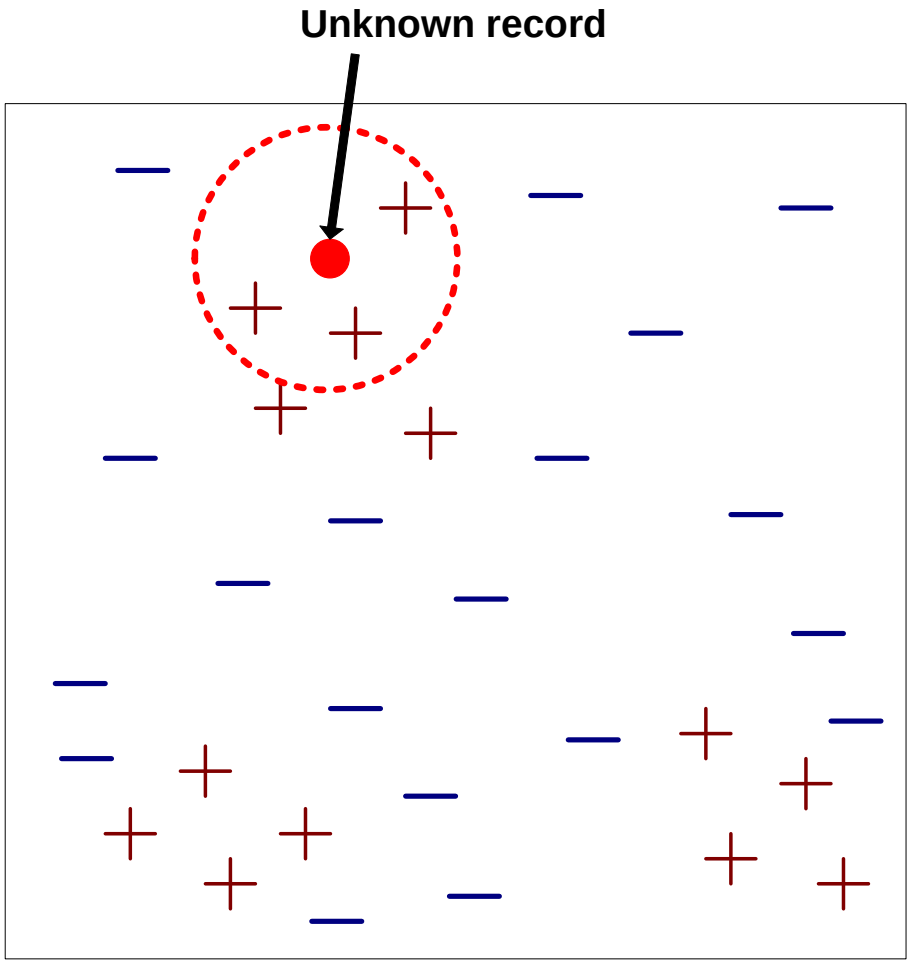
- Instance-based learning:
  - Store training examples and delay the processing ("lazy evaluation") until a new instance must be classified
- Typical approaches
  - *k*-nearest neighbor approach
    - Instances represented as points in a Euclidean space.
  - Case-based reasoning
    - Uses symbolic representations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

k-Nearest Neighbor Classifiers

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

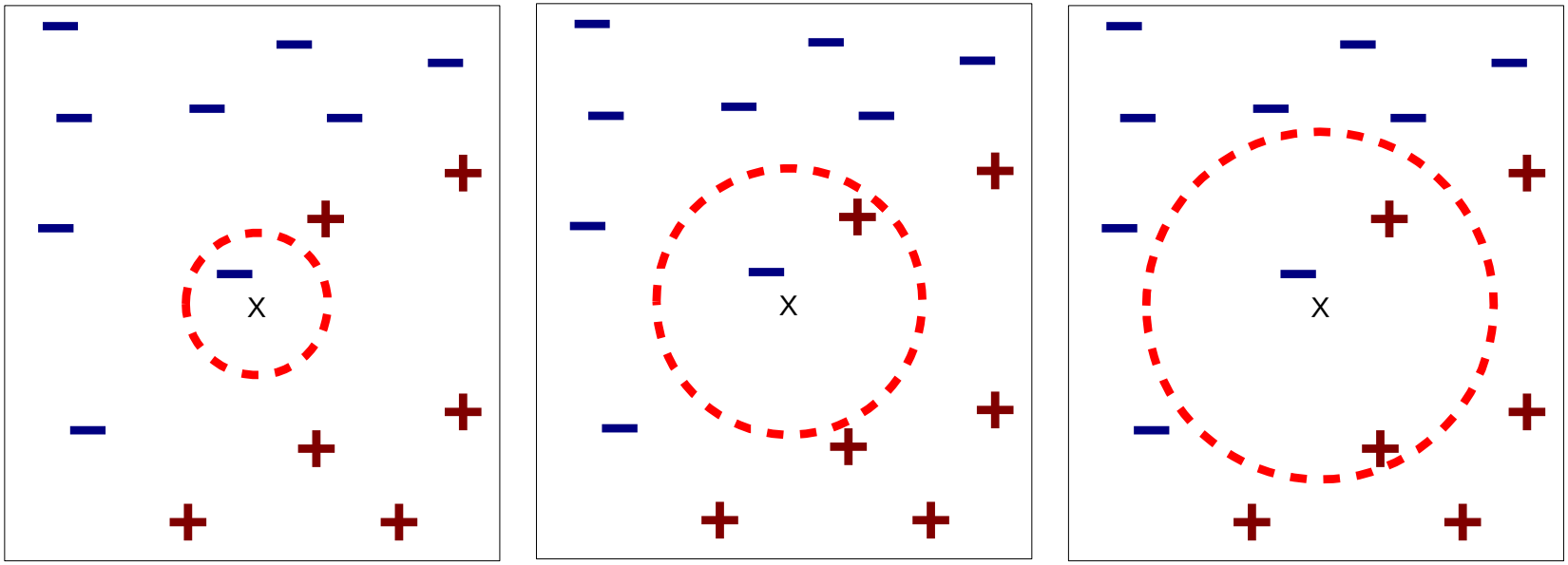


k-Nearest-Neighbor Classifiers



- Requires three things
  - The set of stored records
  - Distance Metric to compute distance between records
  - The value of *k*, the number of nearest neighbors to retrieve
- To classify an unknown record:
  - Compute distance to other training records
  - Identify *k* nearest neighbors
  - Use class labels of nearest neighbors to determine the class label of unknown record (e.g., by taking majority vote)

Definition of Nearest Neighbor



(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

Nearest Neighbor Classification

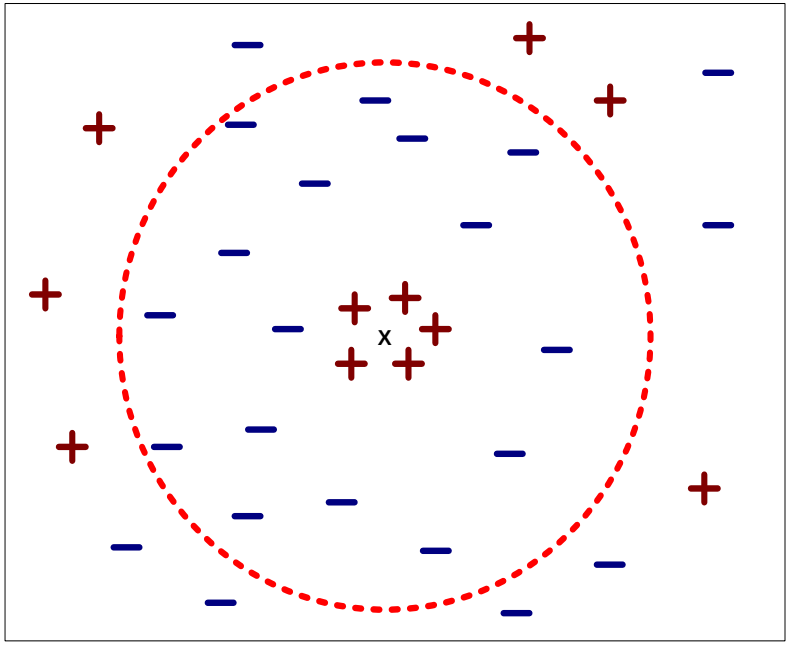
- Compute distance between two points:
  - Euclidean distance

$$d(p,q) = \sqrt{\sum_i (p_i - q_i)^2}$$

- Determine the class from nearest neighbor list
  - take the majority vote of class labels among the k-nearest neighbors
  - Weigh the vote according to distance
    - weight factor, $w = 1/d^2$

Nearest Neighbor Classification…

- Choosing the value of k:
  - If k is too small, sensitive to noise points
  - If k is too large, neighborhood may include points from other classes



Nearest Neighbor Classification…

- Scaling issues
  - Attributes may have to be scaled to prevent distance measures from being dominated by one of the attributes
  - Example:
    - height of a person may vary from 1.5m to 1.8m
    - weight of a person may vary from 90lb to 300lb
    - income of a person may vary from $10K to $1M

Nearest Neighbor Classification…

- Problem with Euclidean measure:
  - High dimensional data
    - curse of dimensionality
  - Can produce counter-intuitive results

1 1 1 1 1 1 1 1 1 1 1 0		1 0 0 0 0 0 0 0 0 0 0 0	
0 1 1 1 1 1 1 1 1 1 1 1	vs	0 0 0 0 0 0 0 0 0 0 0 1	
d = 1.4142		**d = 1.4142**	

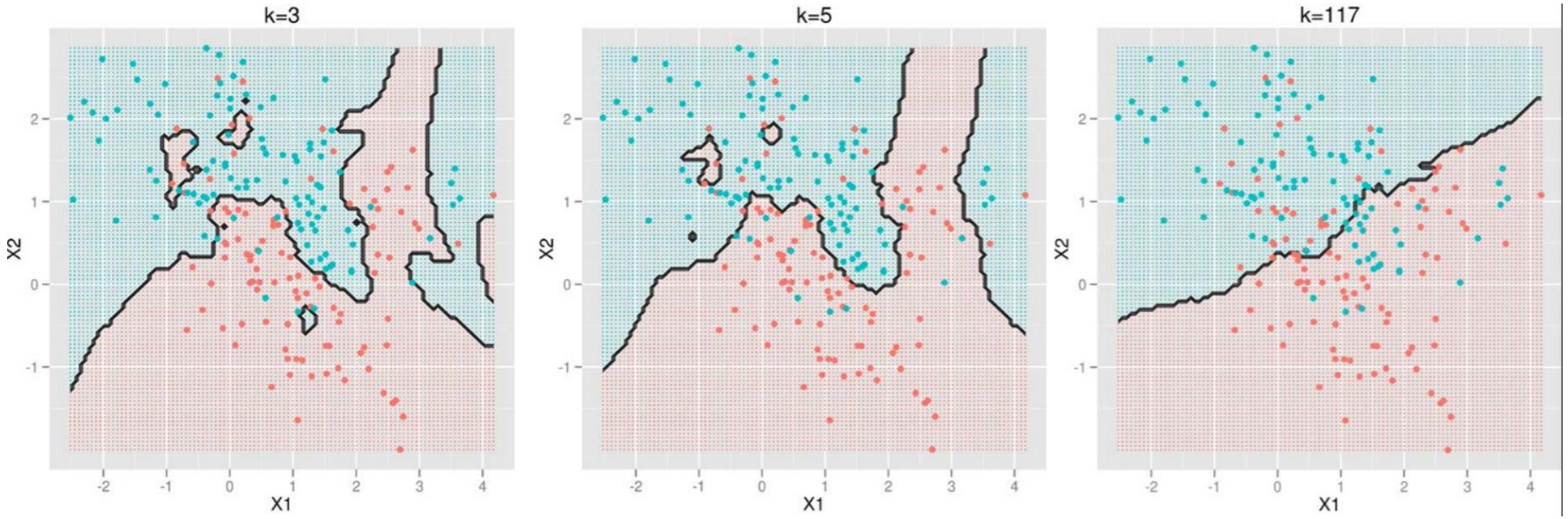
- Solution: Normalize the vectors to unit length

Nearest neighbor Classification…

- k-NN classifiers are lazy learners
  - It does not build models explicitly
  - Unlike eager learners such as decision tree induction and rule-based systems
  - Classifying unknown records are relatively expensive

Selection of k for kNN

- The number of neighbors k
  - Small k: overfitting (high var., low bias)
  - Big k: bringing too many irrelevant points (high bias, low var.)



http://scott.fortmann-roe.com/docs/BiasVariance.html

Case-Based Reasoning (CBR)

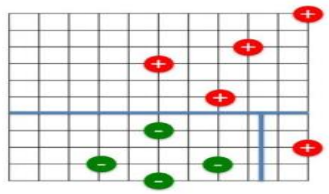
- **CBR**: Uses a database of problem solutions to solve new problems
- Store symbolic description (tuples or cases)—not points in a Euclidean space
- Applications: Customer-service (product-related diagnosis), legal ruling
- Methodology
  - Instances represented by rich symbolic descriptions (e.g., function graphs)
  - Search for similar cases, multiple retrieved cases may be combined
  - Tight coupling between case retrieval, knowledge-based reasoning, and problem solving
- Challenges
  - Find a good similarity metric
  - Indexing based on syntactic similarity measure, and when failure, backtracking, and adapting to additional cases

Presentation Outline (Class #19)

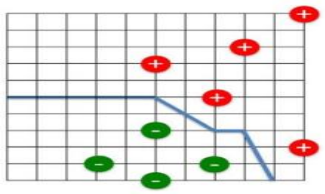
- Background
- Basic concepts
- Decision Tree Induction
- Bayes Classification Methods
- Lazy Learners (k-nearest neighbours)
- **Linear Classifiers**
- Model Evaluation and Selection
- Techniques to Improve Accuracy
- Summary

About Linear classifiers

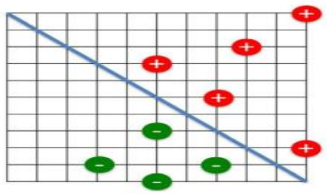
- Classifies generate complex decision boundaries.
- Decision tree might generate hyperrectangular shape boundary
- 1-nearest neighbour classifier may generate hyper polygonal boundary
- Linear classifier may general linear boundary
  - Better generalization performance, better interpretability
- We discuss three classifiers: Linear regression, perceptron, logistic regression (widely used)



(a) Decision Tree Classifier



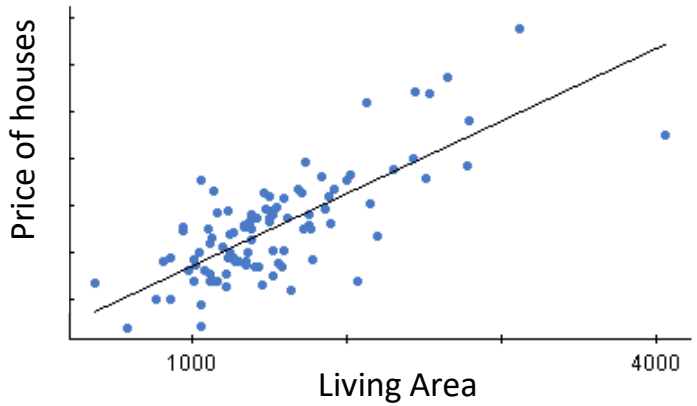
(b) 1—Nearest-Neighbor Classifier

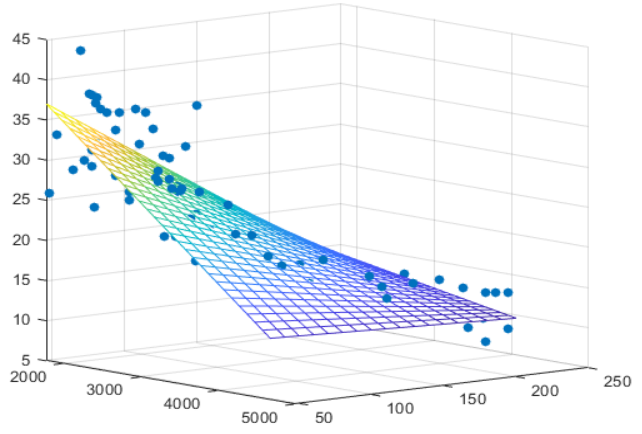


(c) Linear Classifier

Linear Regression Problem: Example

- Mapping from independent attributes to **continuous value**: x => y
- {living area} => Price of the house
- {college; major; GPA} => Future Income





Linear Regression Problem: Model

- Linear regression
  - Data: n independent objects
    - Observed Value: $y_i, i = 1,2,3, \cdots, n$
    - p-dimensional attributes: $x_i = \left(x_{i1}, x_{i2}, \cdots, x_{ip}\right)^T$, $i = 1,2,3 \cdots, n$
  - Model:
    - Weight vector: $w = \left(w_1, w_2, \cdots, w_p\right)$
    - $y_i = w^T x_i + b$
  - The weight vector w and bias b are the model parameter learnt by data

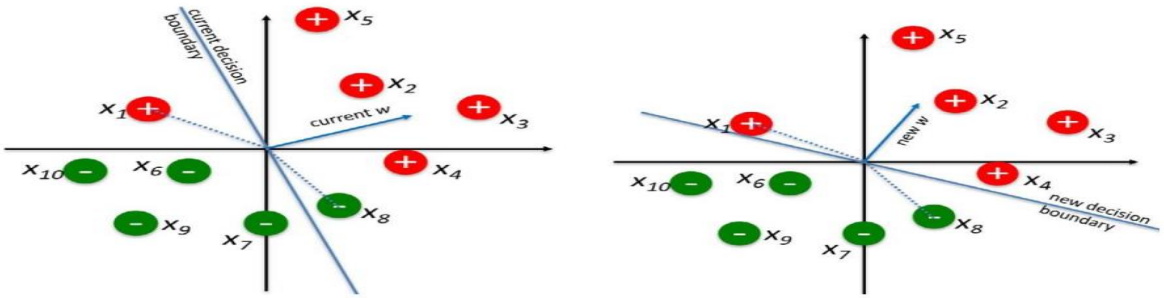
Linear Regression Model: Solution

- Least Square Method
  - Cost / Loss Function: $\mathrm{L}(w,b) = \Sigma_{i=1}^{n}(y_i - wx_i - b)^2$
  - Optimization Goal: argmin $\mathrm{L}(w,b) = \Sigma_{i=1}^{n}(y_i - wx_i - b)^2$
  - Closed-form solution:
    - $w = \frac{\Sigma_{i=1}^{n} x_i(y_i - \bar{y})}{\Sigma_{i=1}^{n} x_i^2 - n\left(\Sigma_{i=1}^{n} x_i\right)^2}$ $\quad b = \frac{1}{n}\Sigma_{i=1}^{n}(y_i - wx_i)$

- For multiple attributes, multiple linear regression methods are applied.

Perceptron

- Consider a binary classification task
- The output value *yi* for a given tuple is a binary variable: *y*i = +1 indicates the *i*th tuple is a positive tuple (e.g., *buy computer*) and *yi* = 0 indicates the *i*th tuple is a negative one (e.g., *not buy computer*).
- Output of regression function is the sign of regression function
- Sign=+1 if yi is the predicted class label, sign=0, otherwise
- If we know the weight vector W, we can predict the class label.
- W is interactively learned from the training set.
- If the training tuples are linearly separable, the perceptron algorithm is guaranteed to find a weight vector (i.e., a hyperplane decision boundary)



(a) current weight vector w (b) updated weight vector w

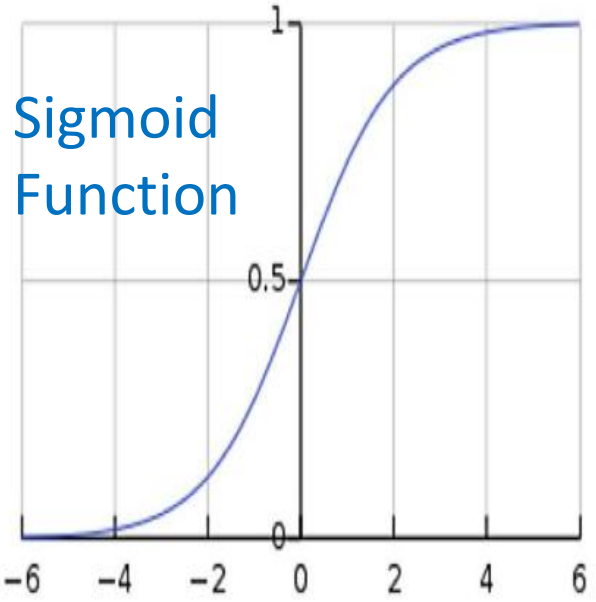
Logistic Regression

- Perceptron predicts the binary class label of a given tuple. However, can we also tell how confident such a prediction is?
- Logistic regression estimates the probability of an event occurring, such as voted or didn't vote, based on a given dataset of independent variables.
- To convert the output to 0 to 1, a sigmoid function is introduced.
  - Sigmoid function (differentiable function) :
  - $\sigma(z) = \frac{1}{1+e^{-z}} = \frac{e^z}{e^z+1}$
  - Projects $(-\infty, +\infty)$ to $[0, 1]$
  - Not only LR uses this function,

    but also neural network, deep learning

- To determine the optimal W vector, the maximum likelihood estimation method is employed.
- It aims to solve the following optimization problem: "choosing the the best weight vector *w* that maximizes the likelihood of the training set.

$$S(x) = \frac{1}{1+e^{-x}}$$



Presentation Outline

- Background
- Basic concepts
- Decision Tree Induction
- Bayes Classification Methods
- Lazy Learners
  - K-nearest neighbors
  - Case based reasoning
- Linear Classifiers
  - Linear regression
  - Perceptron: turning linear regression to classification
  - Logistic regression
- **Model Evaluation and Selection**
- Techniques to Improve Accuracy
- Summary

Model Evaluation and Selection

- Evaluation metrics: How can we measure accuracy? Other metrics to consider?
- Use **validation test set** of class-labeled tuples instead of training set when assessing accuracy
- Methods for estimating a classifier's accuracy:
  - Holdout method, random subsampling
  - Cross-validation
  - Bootstrap
- Comparing classifiers:
  - Confidence intervals
  - Cost-benefit analysis and ROC Curves

Confusion Matrix

- **True positives** (TP): These refer to the positive tuples that were correctly labeled by the classifier.
- **True negatives** (TN): These are the negative tuples that were correctly labeled by the classifier.
- **False positives** (FP): These are the negative tuples that were incorrectly labeled as positive (e.g., tuples of class buys_computer = no for which the classifier predicted buys_computer = yes).
- **False negatives (FN):** These are the positive tuples that were mislabeled as negative (e.g., tuples of class buys_computer = yes for which the classifier predicted buys_computer = no).

Classifier Evaluation Metrics: Confusion Matrix

Confusion Matrix:

Actual class\Predicted class	C_1	¬ C_1
C_1	**True Positives (TP)**	**False Negatives (FN)**
¬ 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)/P+N

- **Error rate:** *1 – accuracy*, or

  Error rate = (FP + FN)/P+N

- **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**

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?

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

- Perfect score is 1.0
- ***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

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

More about Precision and Recall

- Precession (Exactness): Suppose you have done duty for 10 hours, e.g., digging a well. How many hours out of total duty have you spent doing useful work? Ideally, you are supposed to spend 10 hours doing useful work. Total duty time (TP+FP) = time spent on useful work (TP)+ time spent on other work/mistakes (FP). Precision=TP/TP+FP
- Recall (completeness): Suppose the objective is to dig the well for 10 meters depth. You have spent some total time. Recall is about how much of the well is being dug. Total time = time spent digging work (TP) plus other work/mistakes (FN). Recall= TP/TP+FN.
- For a classifier, it is possible to have 100% precision and low recall. For example, you have spent 10 hours (the entire time) digging well. But, only a small portion of the well is being dug (recall is low).
- For a classifier, it is possible to have 100% recall and low precision. For example, you have dug the 10 meters well as required but have spent several days making mistakes (several wrong results).
- The objective is to get good precision and recall, which is a good F1 metric.

Classifier Evaluation Metrics: Example

Actual Class\Predicted class	cancer = yes	cancer = no	Total
cancer = yes	**90**	**210**	300
cancer = no	**140**	**9560**	9700
Total	230	9770	10000

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

- Calculate the measure just introduced
  - Sensitivity = TP/P = 90/300 = 30%
  - Specificity = TN/N = 9560/9700 = 98.56%
  - Accuracy = (TP + TN)/All = (90+9560)/10000 = 96.50%
  - Error rate = (FP + FN)/All = (140 + 210)/10000 = 3.50%
  - Precision = TP/(TP + FP) = 90/(90 + 140) = 90/230 = 39.13%
  - Recall = TP/ (TP + FN) = 90/(90 + 210) = 90/300 = 30.00%
  - F1 = 2 P × R /(P + R) = 2 × 39.13% × 30.00%/(39.13% + 30%) = 33.96%

Evaluating Classifier Accuracy: Holdout & Cross-Validation Methods

- **Holdout method**
  - Given data is randomly partitioned into two independent sets
    - Training set (e.g., 2/3) for model construction
    - Test set (e.g., 1/3) for accuracy estimation
  - Random sampling: a variation of holdout
    - Repeat holdout k times, accuracy = avg. of the accuracies obtained
- **Cross-validation** (*k*-fold, where k = 10 is most popular)
  - Randomly partition the data into *k mutually exclusive* subsets, each approximately equal size
  - At *i*-th iteration, use D_i as test set and others as training set
  - Leave-one-out: *k* folds where *k* = # of tuples, for small sized data
  - ***Stratified cross-validation*: folds are stratified so that class dist. in each fold is approx. the same as that in the initial data (popular method)**

Evaluating Classifier Accuracy: Bootstrap

- **Bootstrap**
  - Works well with small data sets
  - Samples the given training tuples uniformly *with replacement*
    - i.e., each time a tuple is selected, it is equally likely to be selected again and re-added to the training set
- Several bootstrap methods, and a common one is **.632 boostrap**
  - A data set with *d* tuples is sampled *d* times, with replacement, resulting in a training set of *d* samples. The data tuples that did not make it into the training set end up forming the test set. About 63.2% of the original data end up in the bootstrap, and the remaining 36.8% form the test set (since $(1 - 1/d)^d \approx e^{-1} = 0.368$)
  - Repeat the sampling procedure *k* times, overall accuracy of the model:

$$Acc(M) = \frac{1}{k}\sum_{i=1}^{k}(0.632 \times Acc(M_i)_{test_set} + 0.368 \times Acc(M_i)_{train_set})$$

Estimating Confidence Intervals: Classifier Models M_1 vs. M_2

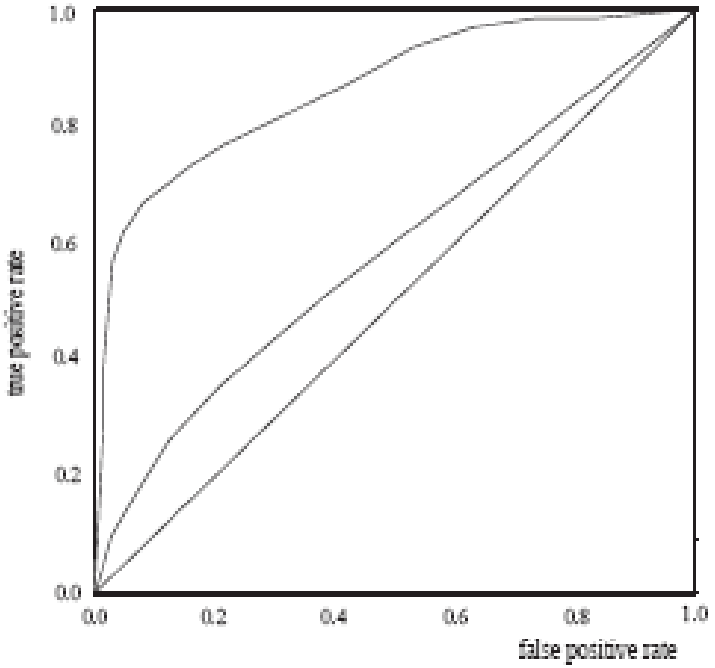
- Suppose we have 2 classifiers, M_1 and M_2, which one is better?
- Use 10-fold cross-validation to obtain $\overline{err}(M_1)$ and $\overline{err}(M_2)$
- These mean error rates are just *estimates* of error on the true population of *future* data cases
- What if the difference between the error rates is just attributed to *chance*?
  - Use a **test of statistical significance**
  - Obtain **confidence limits** for our error estimates

Estimating Confidence Intervals: Null Hypothesis

- Perform 10-fold cross-validation
- Assume samples follow a **t distribution** with *k–1* **degrees of freedom**
- Use **t-test** (or **Student's t-test**)
- **Null Hypothesis**: M_1 & M_2 are the same
- If we can **reject** null hypothesis, then
  - we conclude that the difference between M_1 & M_2 is **statistically significant**
  - Chose model with lower error rate

Model Selection: ROC Curves

- **ROC** (Receiver Operating Characteristics) curves: for visual comparison of classification models
- Shows the trade-off between the true positive rate (TPR) and the false positive rate (FPR)
- *TPR* is the proportion of positive (or "yes") tuples that **are** correctly labeled by the model;
- FPR is the proportion of negative (or "no") tuples that are mislabeled as positive. Recall that T P, FP, P, and N are the number of true positive, false positive, positive, and negative tuples, respectively.
- 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 TPR
- Horizontal axis rep. the FPR
- The plot also shows a diagonal line
- A model with perfect accuracy will have an area of 1.0

Tuple #	Class	Prob.	*TP*	*FP*	*TN*	*FN*	*TPR*	*FPR*
1	*P*	0.90	1	0	5	4	0.2	0
2	*P*	0.80	2	0	5	3	0.4	0
3	*N*	0.70	2	1	4	3	0.4	0.2
4	*P*	0.60	3	1	4	2	0.6	0.2
5	*P*	0.55	4	1	4	1	0.8	0.2
6	*N*	0.54	4	2	3	1	0.8	0.4
7	*N*	0.53	4	3	2	1	0.8	0.6
8	*N*	0.51	4	4	1	1	0.8	0.8
9	*P*	0.50	5	4	1	0	1.0	0.8
10	*N*	0.40	5	5	0	0	1.0	1.0

FIGURE 6.20

Tuples sorted by decreasing score, where the score is the value returned by a probabilistic classifier.

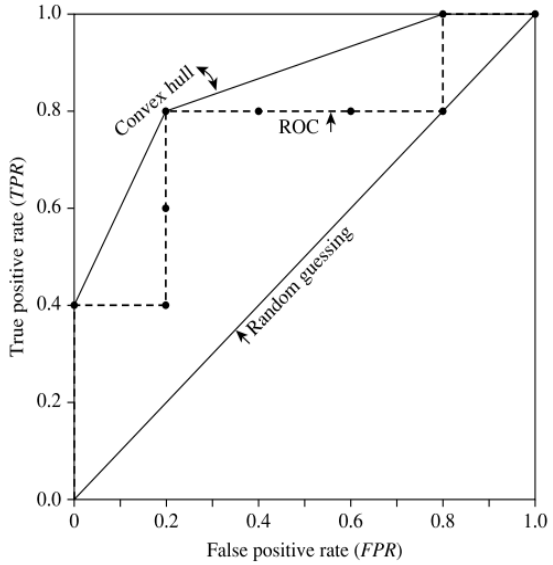


FIGURE 6.21

ROC curve for the data in Figure 6.20.

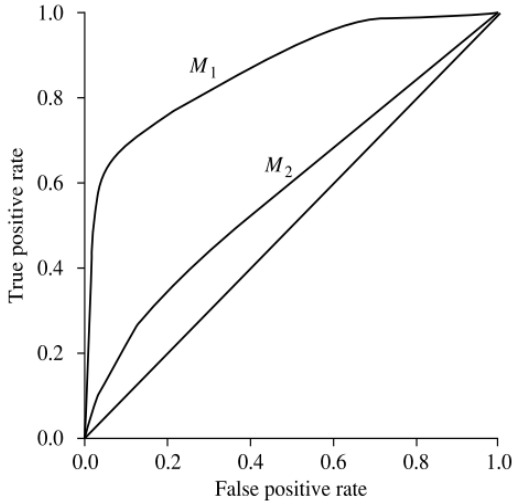


FIGURE 6.22

ROC curves of two classification models, M_1 and M_2. The diagonal shows where, for every true positive, we are equally likely to encounter a false positive. The closer a ROC curve is to the diagonal line, the less accurate the model is. Thus $M1$ is more accurate here.

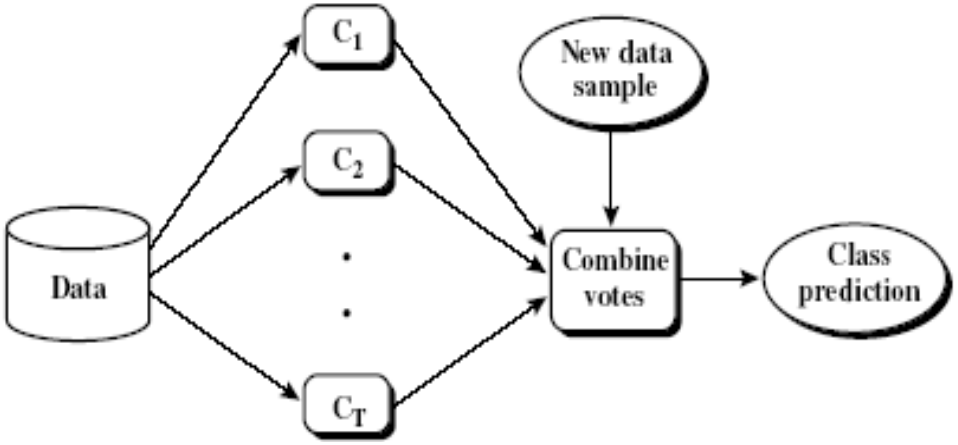
Issues Affecting Model Selection

- **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

Presentation Outline

- Background
- Basic concepts
- Decision Tree Induction
- Bayes Classification Methods
- Rule-Based Classification
- Model Evaluation and Selection
- **Techniques to Improve Accuracy**
- Summary

Ensemble Methods: Increasing the Accuracy



- Ensemble methods
  - Use a combination of models to increase accuracy
  - Combine a series of k learned models, M_1, M_2, …, M_k, with the aim of creating an improved model M*

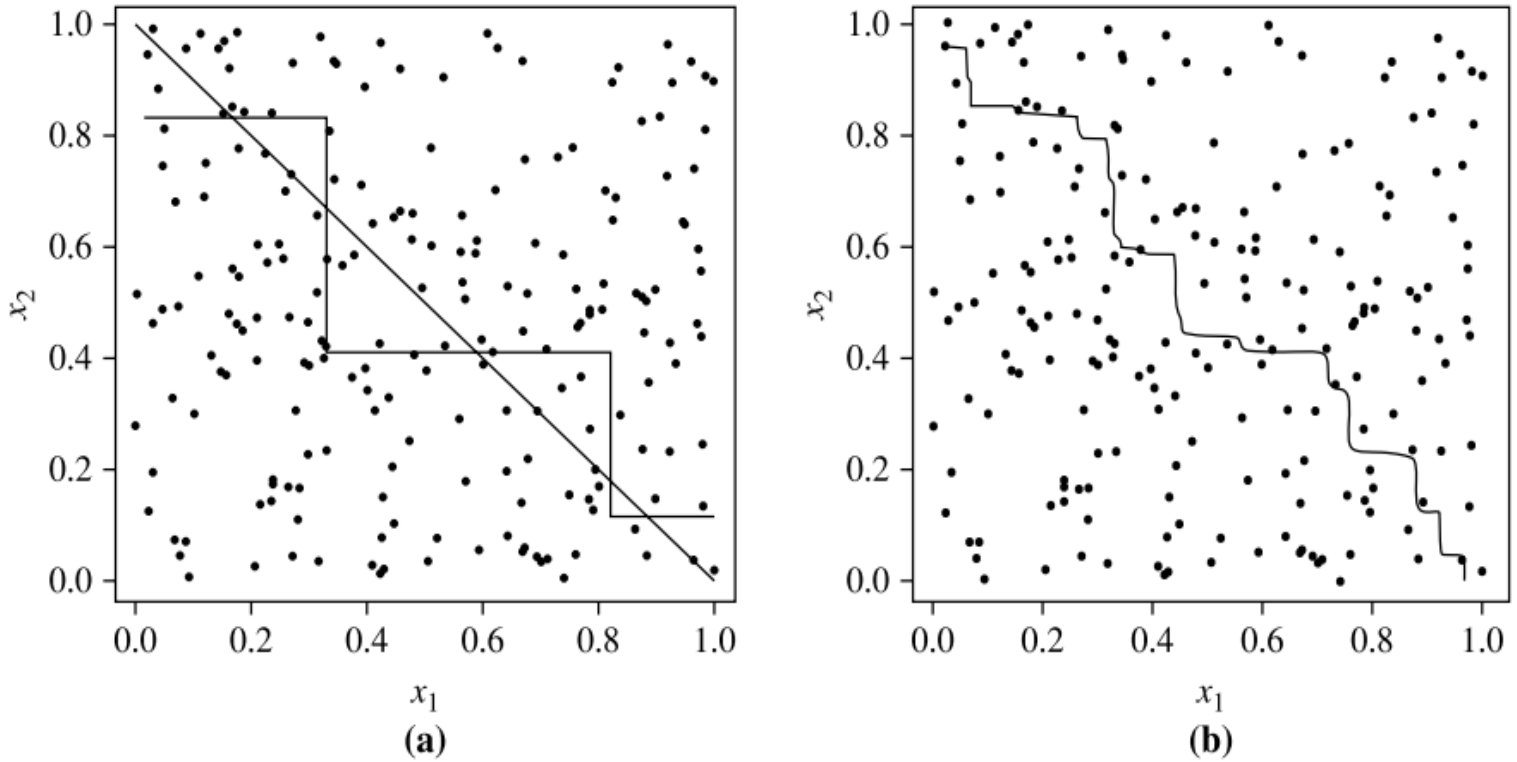
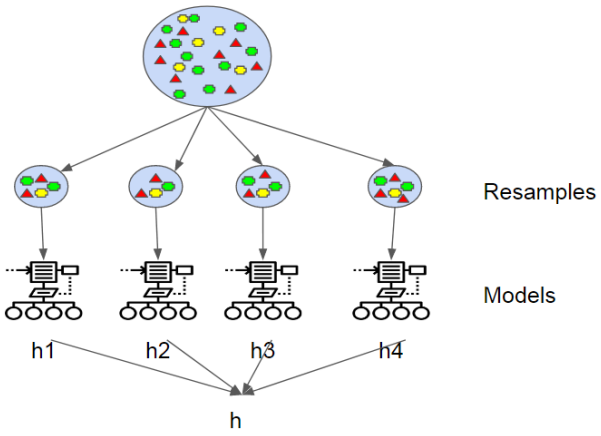


FIGURE 6.24

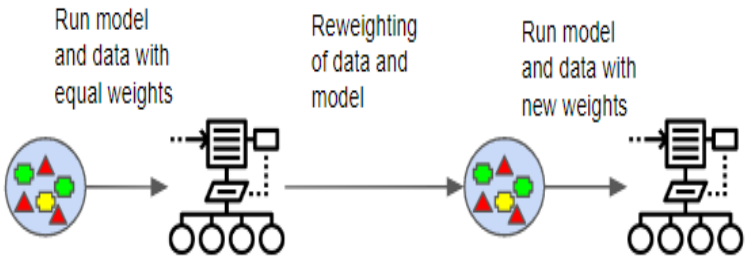
Decision boundary by (a) a single decision tree and (b) an ensemble of decision trees for a linearly separable problem (i.e., where the actual decision boundary is a straight line). The decision tree struggles with approximating a linear boundary. The decision boundary of the ensemble is closer to the true boundary. *Source:* From Seni and Elder [SE10]. © 2010 Morgan & Claypool Publishers; used with permission.

Ensemble Methods: Increasing the Accuracy

- Popular ensemble methods
  - Bagging: Trains each model using a subset of the training set, and models learned in parallel
  - Boosting: Trains each new model instance to emphasize the training instances that previous models mis-classified, and models learned in order



Bagging



Boosting

Bagging: Bootstrap Aggregation

- Analogy: Diagnosis based on multiple doctors' majority vote
- Training
  - Given a set D of *d* tuples, at each iteration *i*, a training set D_i of *d* tuples is sampled with replacement from D (i.e., bootstrap)
  - A classifier model M_i is learned for each training set D_i
- Classification: classify an unknown sample **X**
  - Each classifier M_i returns its class prediction
  - The bagged classifier M* counts the votes and assigns the class with the most votes to **X**
- Prediction: can be applied to the prediction of continuous values by taking the average value of each prediction for a given test tuple
- Accuracy
  - Often significantly better than a single classifier derived from D
  - For noise data: not considerably worse, more robust
  - Proved improved accuracy in prediction

Algorithm: Bagging. The bagging algorithm—create an ensemble of classification models for a learning scheme where each model gives an equally weighted prediction.

Input:

- D, a set of d training tuples;
- k, the number of models in the ensemble;
- a classification learning scheme (e.g., decision tree algorithm, naïve Bayesian, etc.).

Output: The ensemble—a composite model, M^*.

Method:

(1) **for** $i = 1$ to k **do** // create k models:
(2) create bootstrap sample, D_i, by sampling D with replacement;
(3) use D_i and the learning scheme to derive a model, M_i.
(4) **endfor**

To use the ensemble to classify a tuple, X:

let each of the k models classify X and return the majority vote;

FIGURE 6.25

Bagging.

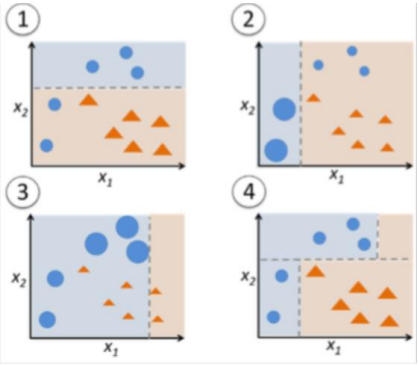
Boosting

- Analogy: Consult several doctors, based on a combination of weighted diagnoses—weight assigned based on the previous diagnosis accuracy
- How boosting works?
  - **Weights** are assigned to each training tuple
  - A series of k classifiers is iteratively learned
  - After a classifier M_i is learned, the weights are updated to allow the subsequent classifier, M_{i+1}, to **pay more attention to the training tuples that were misclassified** by M_i
  - The final **M* combines the votes** of each individual classifier, where the weight of each classifier's vote is a function of its accuracy
- Boosting algorithm can be extended for numeric prediction
- Comparing with bagging: Boosting tends to have greater accuracy, but it also risks overfitting the model to misclassified data

Adaboost (Freund and Schapire, 1997)

1. Assign initial weights to each training tuple

3. Update weights based on current model



$\{w_n^{(1)}\}$ $\{w_n^{(2)}\}$ ... $\{w_n^{(k)}\}$

M_1 M_2 ... M_k

2. Train base classifier on weighted dataset

Two 'weighting' strategy:

1. Assign weights to each training example
2. Sample dataset based on weight distribution

$$M^*(x) = \text{sign}\left(\sum_{i=1}^{k} \alpha_i M_i(x)\right)$$

4. After base classifiers are trained, they are combined to give the final classifier

Adaboost (Freund and Schapire, 1997)
Adaptive boosting

- Given a set of *d* class-labeled tuples, ($\mathbf{X_1}$, y_1), …, ($\mathbf{X_d}$, y_d)
- Initially, all the weights of tuples are set the same (1/d)
- Generate k classifiers in k rounds. At round i,
  - Tuples from D are sampled (with replacement) to form a training set D_i of the same size
  - Each tuple's chance of being selected is based on its weight
  - A classification model M_i is derived from D_i
  - Its error rate is calculated using D_i as a test set
  - If a tuple is misclassified, its weight is increased, o.w. it is decreased
- Error rate: err($\mathbf{X_j}$) is the misclassification error of tuple $\mathbf{X_j}$. Classifier M_i error rate is the sum of the weights of the misclassified tuples:

$$error(M_i) = \sum_j^d w_j \times err(\mathbf{X_j})$$

- The weight of classifier M_i's vote is $\log \frac{1-error(M_i)}{error(M_i)}$

Algorithm: AdaBoost. A boosting algorithm—create an ensemble of classifiers. Each one gives a weighted vote.

Input:

- D, a set of d class-labeled training tuples;
- k, the number of rounds (one classifier is generated per round);
- a classification learning scheme.

Output: A composite model.

Method:

(1) initialize the weight of each tuple in D to $1/d$;
(2) **for** $i = 1$ to k **do** // for each round:
(3) sample D with replacement according to the tuple weights to obtain D_i;
(4) use training set D_i to derive a model, M_i;
(5) compute $error(M_i)$, the error rate of M_i (Eq. (6.34))
(6) **if** $error(M_i) > 0.5$ **then**
(7) abort the loop;
(8) **endif**
(9) **for** each tuple in D that was correctly classified **do**
(10) multiply the weight of the tuple by $error(M_i)/(1 - error(M_i))$; // update weights
(11) normalize the weight of each tuple.
(12) **endfor**

To use the ensemble to classify tuple, X:

(1) initialize weight of each class to 0;
(2) **for** $i = 1$ to k **do** // for each classifier:
(3) $w_i = log \frac{1-error(M_i)}{error(M_i)}$; // weight of the classifier's vote
(4) $c = M_i(X)$; // get class prediction for X from M_i
(5) add w_i to the weight for class c
(6) **endfor**
(7) return the class with the largest weight.

FIGURE 6.26

AdaBoost, a boosting algorithm.

Gradient Boosting and XGboost

- **Gradient boosting** is another powerful boosting technique, which can be used for classification, regression, and ranking.
  - If we use a tree (e.g., decision tree for classification, regression tree for regression) as the base model (i.e., the weak learner), it is called **gradient tree boosting**, or **gradient boosted tree**
- A highly scalable end-to-end gradient tree boosting system is called **XGBoost**, which is capable to handle a billion-scale training set.
- XGBoost has made a number of innovations for training gradient tree boosting, including a new tree construction algorithm designed for sparse data, feature subsampling (as opposed to training tuple subsampling in stochastic gradient boosting)
- A highly efficient cacheaware block structure. XGBoost has been successfully used by data scientists in many data mining challenges, often leading to top competitive results.

Random Forest (Breiman 2001)

- Random Forest:
  - Each classifier in the ensemble is a *decision tree* classifier and is generated using a random selection of attributes at each node to determine the split
  - During classification, each tree votes and the most popular class is returned
- Two Methods to construct Random Forest:
  - Forest-RI (*random input selection*): Randomly select, at each node, F attributes as candidates for the split at the node. The CART methodology is used to grow the trees to maximum size
  - Forest-RC (*random linear combinations*): Creates new attributes (or features) that are a linear combination of the existing attributes (reduces the correlation between individual classifiers)
- Comparable in accuracy to Adaboost, but more robust to errors and outliers
- Insensitive to the number of attributes selected for consideration at each split, and faster than bagging or boosting

Ensemble Methods Recap

- Random forest and XGBoost are the most commonly used algorithms for tabular data
- Pros
  - Good performance for tabular data, requires no data scaling
  - Can scale to large datasets
  - Can handle missing data to some extent
- Cons
  - Can overfit to training data if not tuned properly
  - Lack of interpretability (compared to decision trees)

Classification of Class-Imbalanced Data Sets

- Class-imbalance problem: Rare positive example but numerous negative ones, e.g., medical diagnosis, fraud, oil-spill, fault, etc.
- Traditional methods assume a balanced distribution of classes and equal error costs: not suitable for class-imbalanced data
- Typical methods for imbalance data in 2-class classification:
  - **Oversampling**: re-sampling of data from positive class so that the training set contains equal number of positive and negative samples.
  - **Under-sampling**: randomly eliminate tuples from negative class so that the training set contains equal number of positive and negative samples.
  - **Threshold-moving**: moves the decision threshold, t, so that the rare class tuples are easier to classify, and hence, less chance of costly false negative errors
  - Ensemble techniques: Ensemble multiple classifiers introduced above
- Still difficult for class imbalance problem on multiclass tasks

Presentation Outline

- Background
- Basic concepts
- Decision Tree Induction
- Bayes Classification Methods
- Rule-Based Classification
- Model Evaluation and Selection
- Techniques to Improve Accuracy
- **Summary**

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. Bagging and boosting can be used to increase overall accuracy by learning and combining a series of individual models.

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

Classification: Overview of Advanced Methods (Class #20)

P. Krishna Reddy, IIIT Hyderabad

Topics

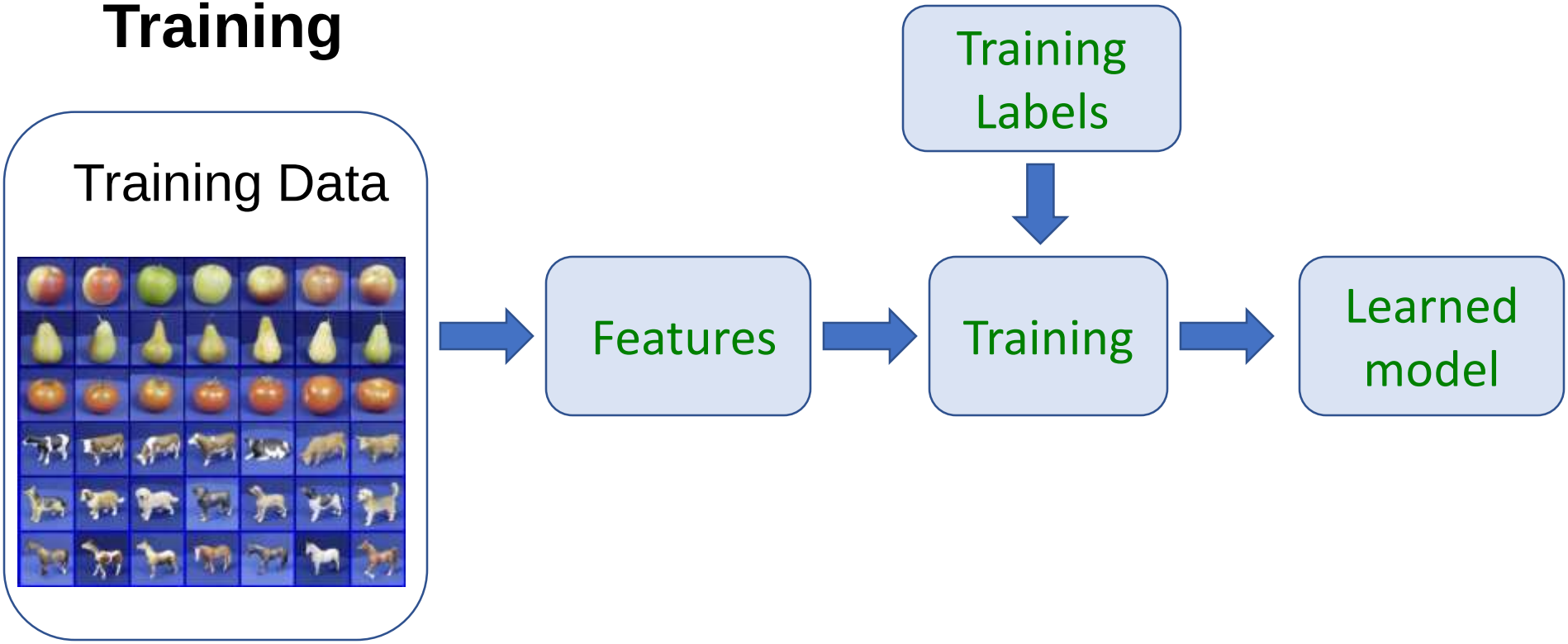
- Introduction (1.5 hour): Definition, KDD framework, Issues in data mining.
- Data summarization (7.5 hrs): Data Types, Preprocessing, Characterization, Discrimination, data warehousing techniques (Multidimensional data model, Data warehousing architecture, Data cube computation methods and OLAP technology)
- Concepts and algorithms for mining patterns and associations (9 hours) (Frequent item-set generation, A priori and FP-growth algorithm, Evaluation of Association patterns) and preprocessing, Overview of advanced mehods
- **Concepts and algorithms related to classification and regression (9hrs) (Overview, Decision tree induction, Over-fitting and under-fitting, Scalable decision tree algorithms, Bayesian Classification, Regression-based Prediction methods, Overview of Advanced Methods (9 hours)**
- Concepts and algorithms for clustering the data (9 hours) (Overview, Types of Data, K-means, Aglomerative clustering, Clustering algorithms (DBSCAN, BIRCH, CURE, ROCK, CHAMELEON)).
- Outlier analysis and future trends (graph mining, spatio-temporal mining). (3 hours)

Outline

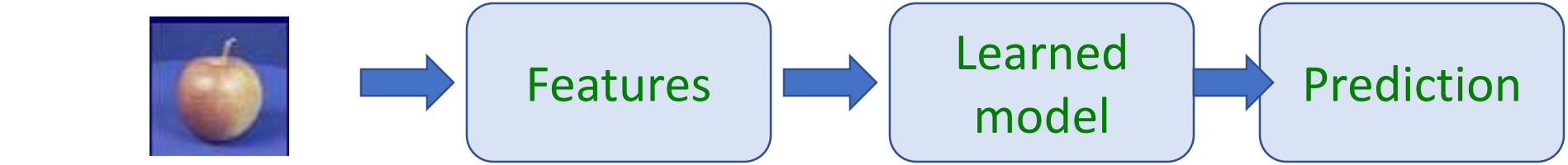
- **Feature Selection and Engineering**
- Bayesian Belief Networks
- Support Vector Machines
- Rule based and pattern based classification
- Classification by weak supervision
- Classification with rich data type
- Other Related Methods
- Summary

Steps

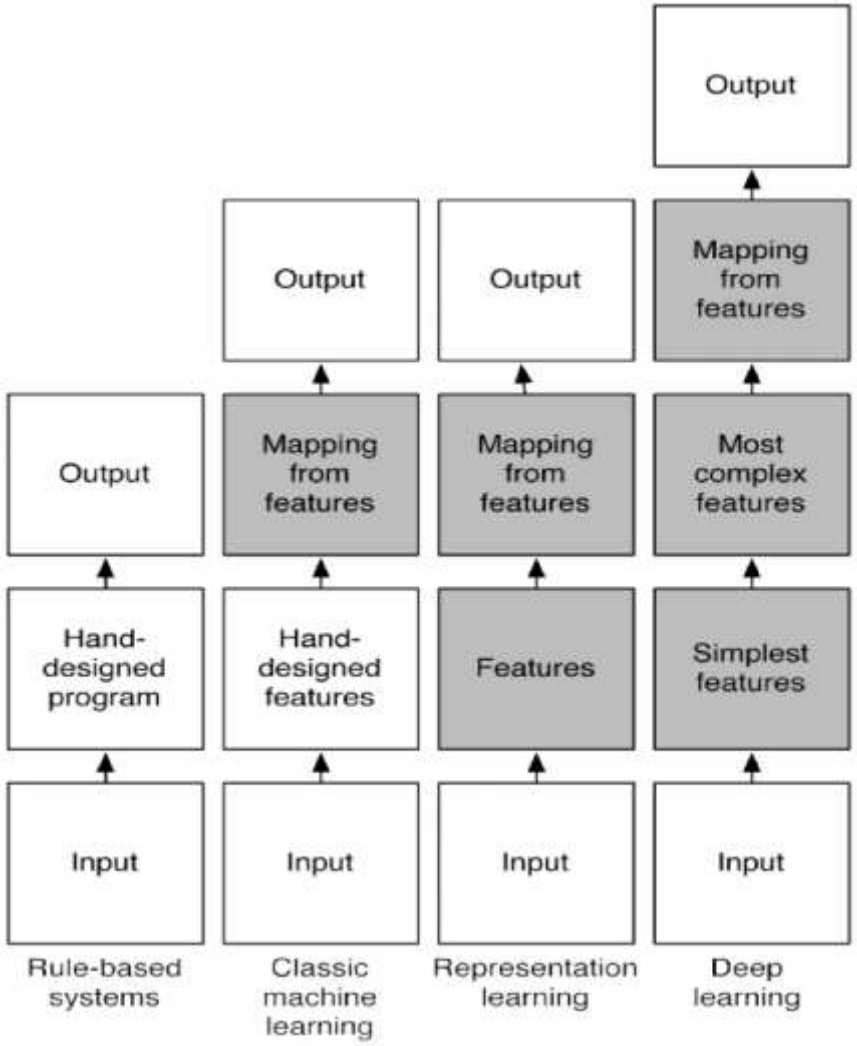
Training



Testing



What is deep learning?



Y. Bengio et al, ``Deep Learning", MIT Press, 2015

Tabular Data

age	income	student	credit_rating	_comp
<=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

Issues

- Assumption: attributes are known
  - Realty: Attributes are unknown
- Assumption: Training set is given
  - Realty: Training set is not given

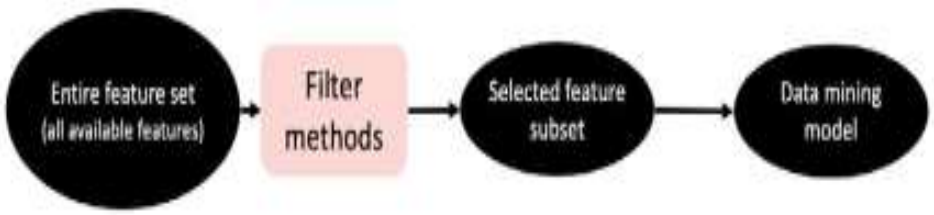
About feature selection and Engineering

- So far, we assumed
  - A training set of with n tuples for p attributes (features) is available.
- Question, where do these features have come?
- **Feature selection:**
  - Large number of features are collected (thousands or more)
  - Question: how to select a subset of features from initial set of p features to improve generalization (classification) performance?
  - Example:
    - Student ID is irrelevant to predict whether a student will drop the course.
    - Monthly income and yearly income of redundant features to predict "Buy Computer"
  - Irrelevant attributes make classifier sensitive to noise.

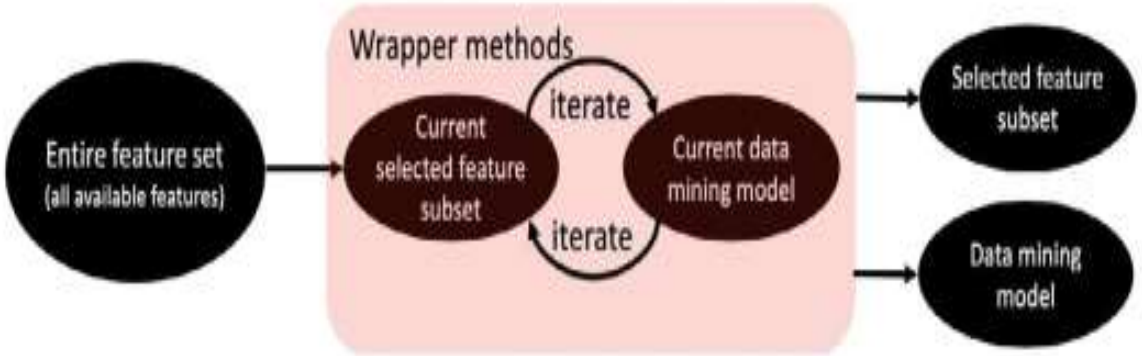
- **Feature Engineering:** How can I construct *p* features so that all of them are critical for the classification task I have?
- Given set of p features, can I transform them to p' attributes so that these transformed features will be more effective for classification?
- Example: Health surveillance data which includes number of daily positive cases, number of daily tests, number of daily hospitalization.
  - It has turned out that weekly positive rate is powerful indicator for disease outbreak.
  - It can be derived from the daily data

Feature selection methods

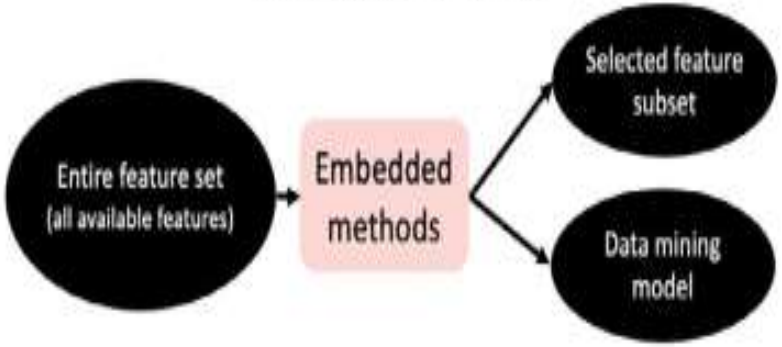
- Filter methods
  - Independent of classification model
- Wrapper methods
  - Combines feature selection and classifier model
- Embedded methods
  - Embeds the feature selection model during the classification step



(a) Filter methods



(b) Wrapper methods



(c) Embedded methods

Filter Method

- Selects features based on certain goodness measure of the input features
- Have a goodness score for each feature and select features with highest goodness score.
- Method 1: A feature is good if it highly correlated with the class label we want to predict.
  - Higher chi-square value indicates stronger corelation.
  - Select k features with highest chi-square values.
  - For continuous attribute
    - Option 1: discretize and compute chi-square
    - Option 2: Compute Fisher score (computes correlation between continuous value and a categorical attribute
- Method 2: Information theoretic measures
  - Information gain
- Limitations
  - Independent of classification model
  - Does not consider the interaction between the attributes.

Wrapper method

- It is an iterative process
  - In each iteration
    - Build a classifier based on currently selected feature set and build a classifier
    - Update the feature set (add, remove, swap)
- Straightforward method
  - Try all subsets of p features = 2^p-1 → exponential!
- Stepwise forward selection method
  - Start with empty set.
  - Iteratively, include additional feature if it improves accuracy.
- Stepwise backward selection
  - Start with initial P features
  - Iteratively eliminate a feature, if it improves accuracy
- We can combine both
- Other techniques
  - Genetic algorithms
  - Simulated annealing: probabilistic algorithm designed for nonconvex optimization problem

Embedded methods

- Combines advantages of filter methods and wrapper methods
- Embedded method performs feature selection and classification model construction simultaneously.
- Example: decision tree
  - After termination, all features at non-leaf nodes are selected features.
- Sparse Learning
  - We build data mining models by minimizing some objective function.
  - Example:
    - Least square linear regression: we want to find a weight vector w by minimizing the sum of the squared difference between the predicted output and the actual output.
    - Logistic regression: find the optimal vector w by minimizing the negative log likelihood.
- By making the corresponding weight of the variable zero, we can remove the feature and calculate the loss function.

Outline

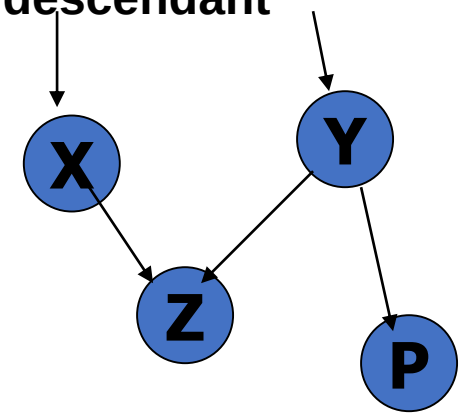
- Feature Selection and Engineering
- **Bayesian Belief Networks**
- Support Vector Machines
- Rule based and pattern based classification
- Classification by weak supervision
- Classification with rich data type
- Other Related Methods
- Summary

Bayesian belief networks

- Assumption of Bayesian classifier is class conditional independence
  - Given the class label of a tuple, the values of the attributes are assumed to be conditionally independent of one another. This simplifies computation.
- When the assumption holds true, then the naïve Bayesian classifier is the most accurate in comparison with all other classifiers.
- In practice, however, dependencies can exist between variables.
- **Bayesian belief networks** specify joint conditional probability distributions.
  - allow class conditional independencies to be defined between subsets of variables.
  - Provide a graphical model of causal relationships, on which learning can be performed.
- Trained Bayesian belief networks can be used for classification.
- Other names are **belief networks**, **Bayesian networks**, and **probabilistic networks**.
- We will refer to them as belief networks

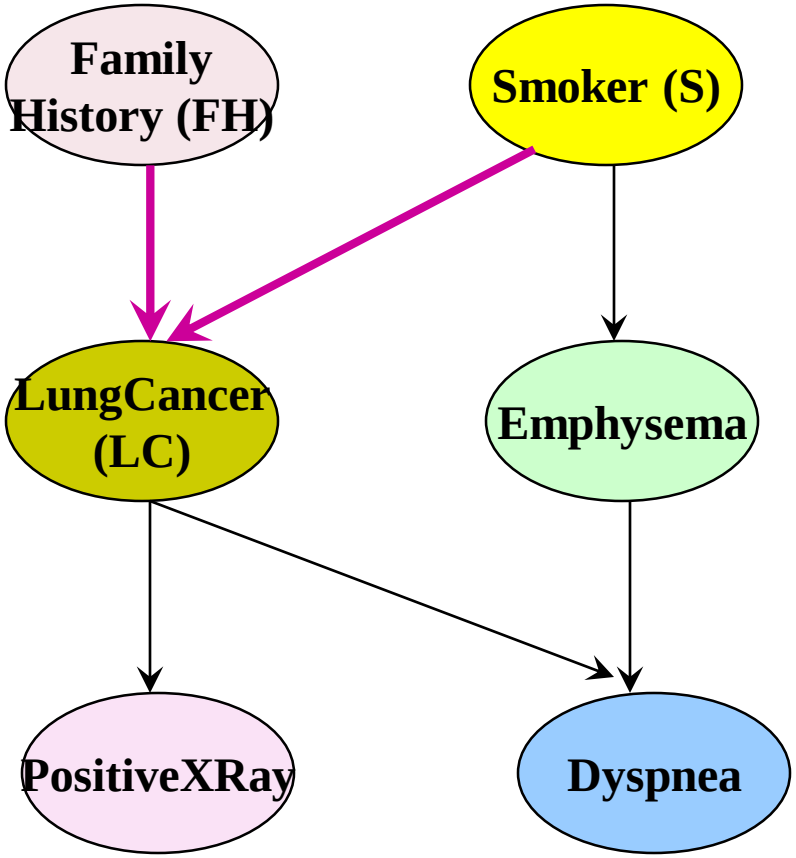
Bayesian Belief Networks

- A belief network is defined by two components—
  - a directed acyclic graph and a set of conditional probability tables.
- Each node in the directed acyclic graph represents a random variable.
  - The variables may be discrete- or continuous-valued.
  - They may correspond to actual attributes given in the data or to “hidden variables” believed to form a relationship
  - About hidden variables
    - In the case of medical data, a hidden variable may indicate a syndrome, representing a number of symptoms that, together, characterize a specific disease.
- Each arc represents a probabilistic dependence. If an arc is drawn from a node *Y* to a node *Z*, then *Y* is a **parent** or **immediate predecessor** of *Z*, and *Z* is a **descendant**



- ❑ Nodes: random variables
- ❑ Links: dependency
- ❑ X and Y are the parents of Z, and Y is the parent of P
- ❑ No dependency between Z and P
- ❑ Has no loops/cycles

Bayesian Belief Network: An Example



Bayesian Belief Network

CPT: **Conditional Probability Table** for variable LungCancer:

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

shows the conditional probability for each possible combination of its parents

Derivation of the probability of a particular combination of values of **X**, from CPT:

$$P(x_1, \ldots, x_n) = \prod_{i=1}^{n} P(x_i \mid Parents(Y_i))$$

About Example

- The arcs allow the representation of casual knowledge
  - For example, having lung cancer is influenced by a person's family history of lung cancer, as well as whether or not the person is a smoker.
  - Note that the variable PositiveXRay is independent of whether the patient has a family history of lung cancer or is a smoker, given that we know the patient has lung cancer.
- In other words, once we know the outcome of the variable LungCancer, then the variables FamilyHistory and Smoker do not provide any additional information regarding PositiveXRay.
- The arcs also show that the variable LungCancer is conditionally independent of Emphysema, given its parents, FamilyHistory and Smoker

About Conditional Probability Table (CPT)

- The CPT for a variable *Y* specifies the conditional distribution *P(Y|Parents(Y))*, where *Parents(Y)* are the parents of *Y*.
- in CPT, the conditional probability for each known value of *LungCancer* is given for each possible combination of the values of its parents.
- Example

P(LungCancer =yes| FamilyHistory =yes, Smoker =yes)= 0.8
P(LungCancer =no| FamilyHistory =no, Smoker =no)= 0.9.

Let $\boldsymbol{X} = (x_1, \ldots, x_n)$ be a data tuple described by the variables or attributes $Y_1, \ldots, Y_n$, respectively. Recall that each variable is conditionally independent of its nondescendants in the network graph, given its parents. This allows the network to provide a complete representation of the existing joint probability distribution with the following equation:

$$P(x_1, \ldots, x_n) = \prod_{i=1}^{n} P(x_i | Parents(Y_i)), \qquad (9.1)$$

where $P(x_1, \ldots, x_n)$ is the probability of a particular combination of values of $\boldsymbol{X}$, and the values for $P(x_i|Parents(Y_i))$ correspond to the entries in the CPT for Y_i.

Applications

- Genetic linkage analysis (e.g., the mapping of genes onto a chromosomes.
- Computer vision (e.g., image restoration and stereo vision)
- document and text analysis
- decision support systems
- sensitivity analysis.

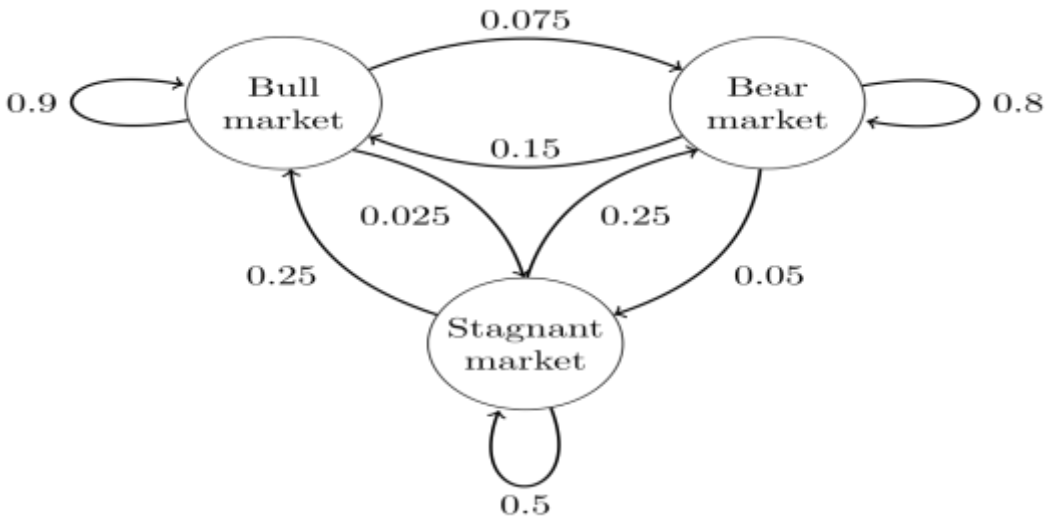
Training Bayesian Networks: Several Scenarios

- **Scenario 1: Given both the network structure and all variables observable: *compute only the CPT entries***
  - **Similar to naïve Bayesian classification**
- Scenario 2: Network structure known, some variables hidden: *gradient descent* (greedy hill-climbing) method, i.e., search for a solution along the steepest descent of a criterion function
  - Weights are initialized to random probability values
  - At each iteration, it moves towards what appears to be the best solution at the moment, w.o. backtracking
  - Weights are updated at each iteration & converge to local optimum
- Scenario 3: Network structure unknown, all variables observable: search through the model space to *reconstruct network topology*
- Scenario 4: Unknown structure, all hidden variables: No good algorithms known for this purpose
- D. Heckerman. A Tutorial on Learning with Bayesian Networks. In *Learning in Graphical Models*, M. Jordan, ed.. MIT Press, 1999.

Markov Chains

- Undirected graph
- A Markov chain is **a stochastic model that uses mathematics to predict the probability of a sequence of events occurring based on the most recent event**.
- Memory less
- Example 1: A common example of a Markov chain in action is the way Google predicts the next word in your sentence based on your previous entry within Gmail

- Example 2:The states represent whether a hypothetical stock market is exhibiting a bull market, bear market, or stagnant market trend during a given week. According to the figure, a bull week is followed by another bull week 90% of the time, a bear week 7.5% of the time, and a stagnant week the other 2.5% of the time.



Outline

- Feature Selection and Engineering
- Bayesian Belief Networks
- **Support Vector Machines**
- Rule based and pattern based classification
- Classification by weak supervision
- Classification with rich data type
- Other Related Methods
- Summary

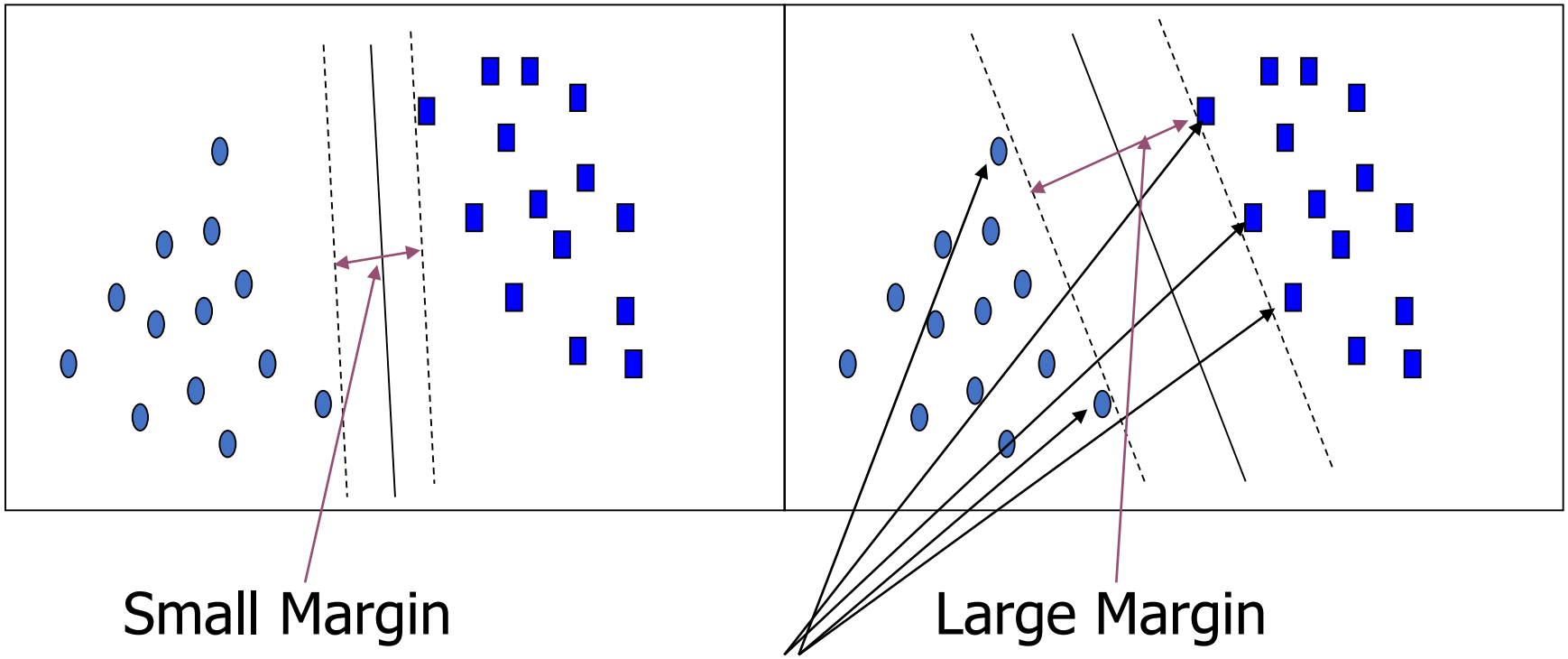
SVM—Support Vector Machines

- A relatively new classification method for both linear and nonlinear data
- It uses a nonlinear mapping to transform the original training data into a higher dimension
- With the new dimension, it searches for the linear optimal separating **hyperplane** (i.e., "decision boundary")
- With an appropriate nonlinear mapping to a sufficiently high dimension, data from two classes can always be separated by a hyperplane
- SVM finds this hyperplane using **support vectors** ("essential" training tuples) and **margins** (defined by the support vectors)

SVM—History and Applications

- Vapnik and colleagues (1992)—groundwork from Vapnik & Chervonenkis' statistical learning theory in 1960s
- Features: training can be slow but accuracy is high owing to their ability to model complex nonlinear decision boundaries (margin maximization)
- Used for: classification and numeric prediction
- Applications:
  - handwritten digit recognition, object recognition, speaker identification, benchmarking time-series prediction tests

SVM—General Philosophy



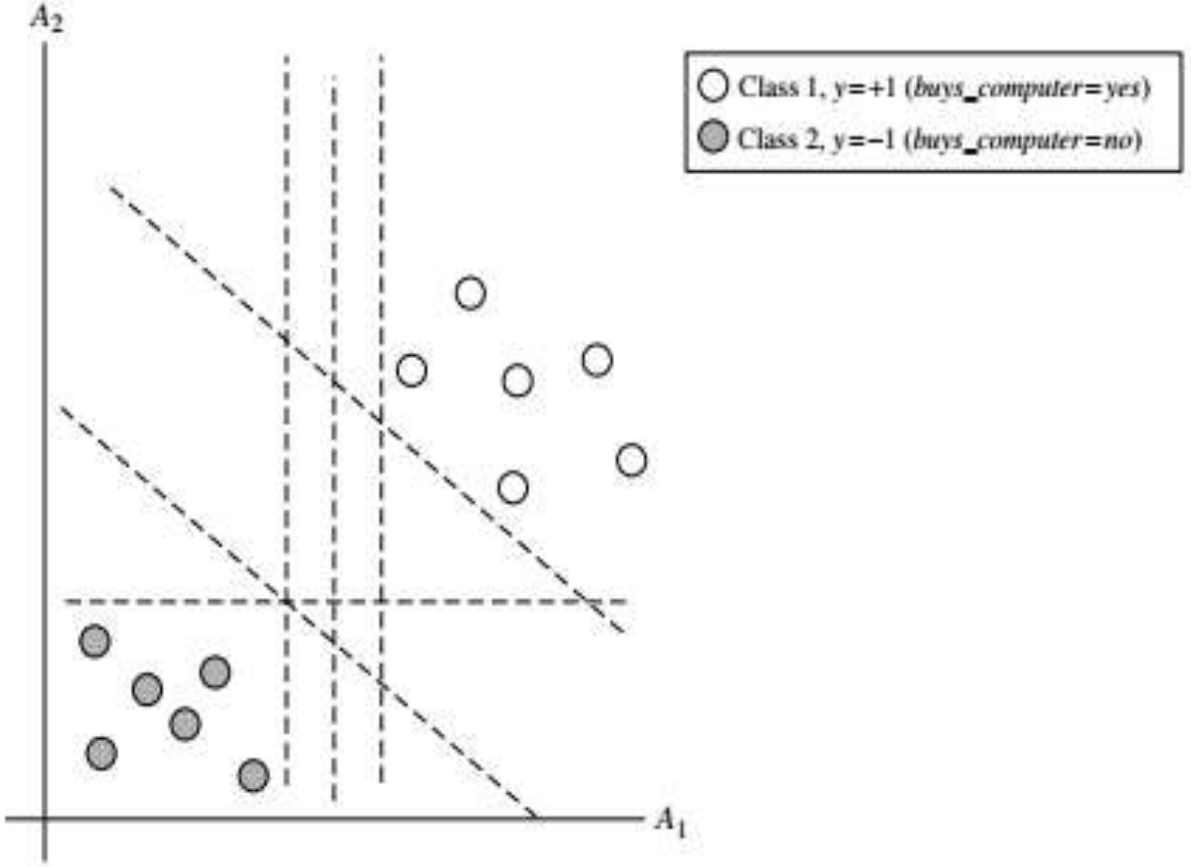


Figure 9.7 The 2-D training data are linearly separable. There are an infinite number of possible separating hyperplanes or "decision boundaries," some of which are shown here as dashed lines. Which one is best?

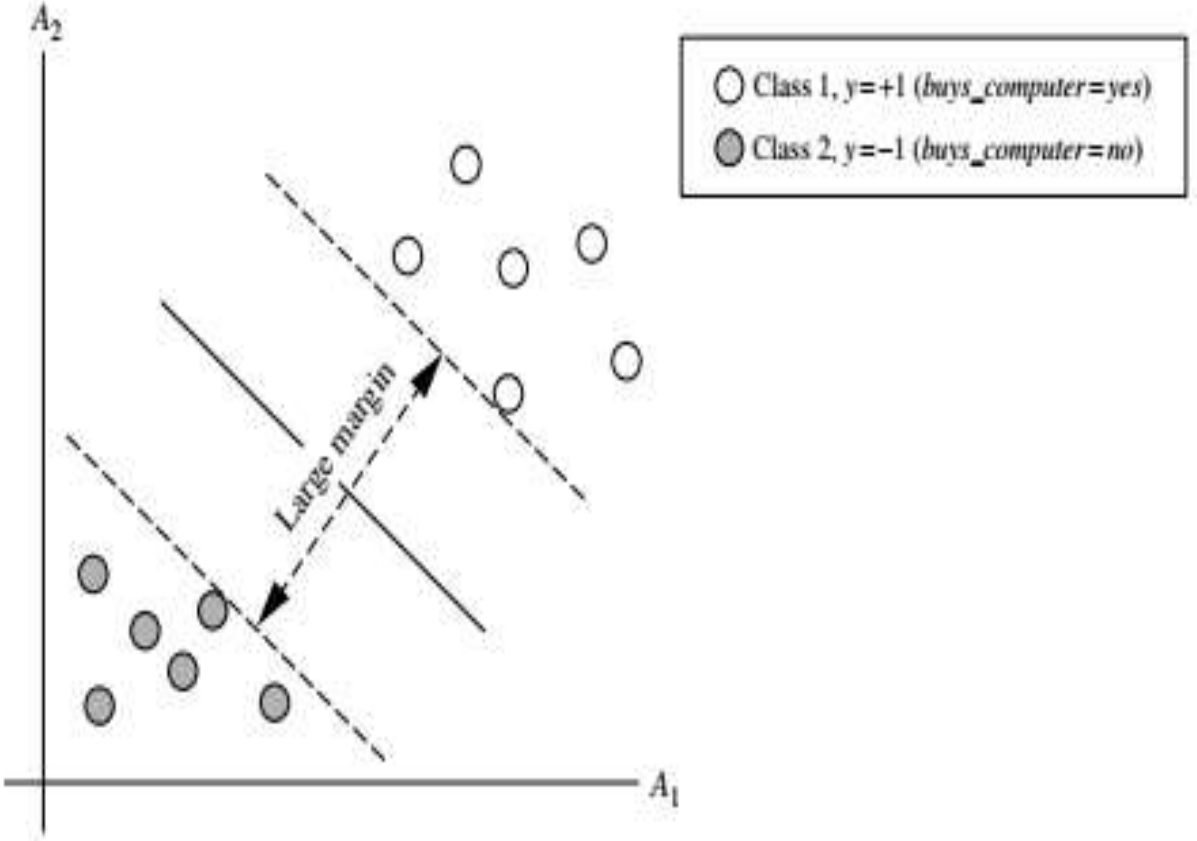
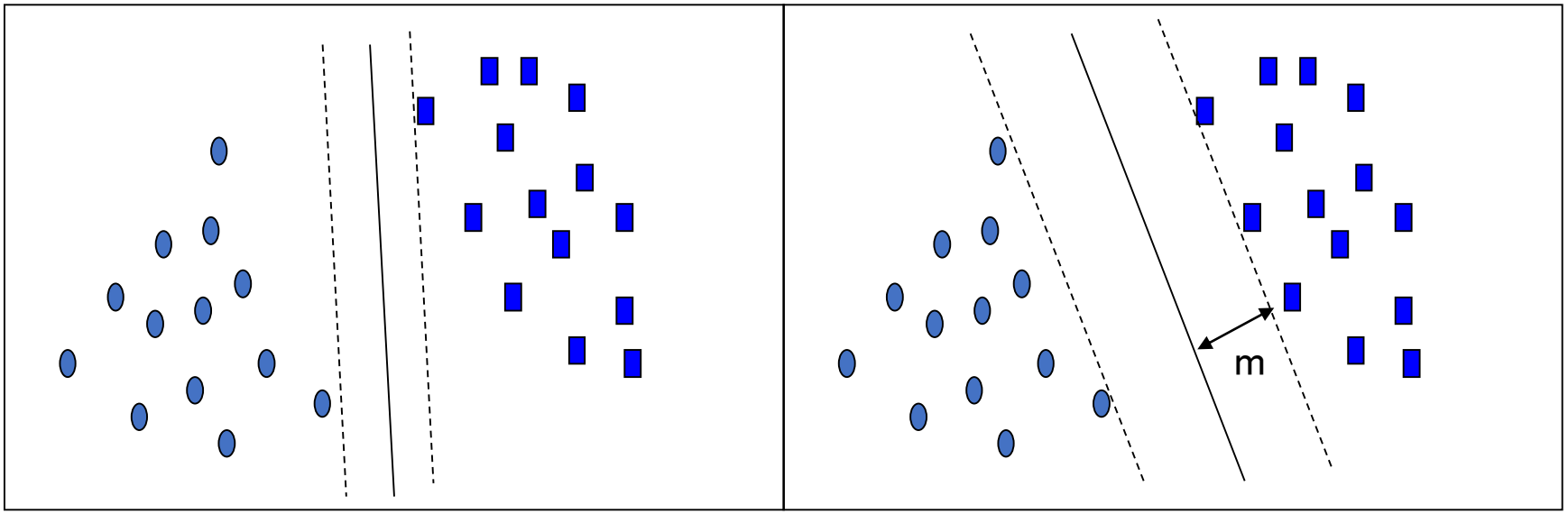


Figure 9.9 Support vectors. The SVM finds the maximum separating hyperplane, that is, the one with maximum distance between the nearest training tuples. The support vectors are shown with a thicker border.

SVM—When Data Is Linearly Separable

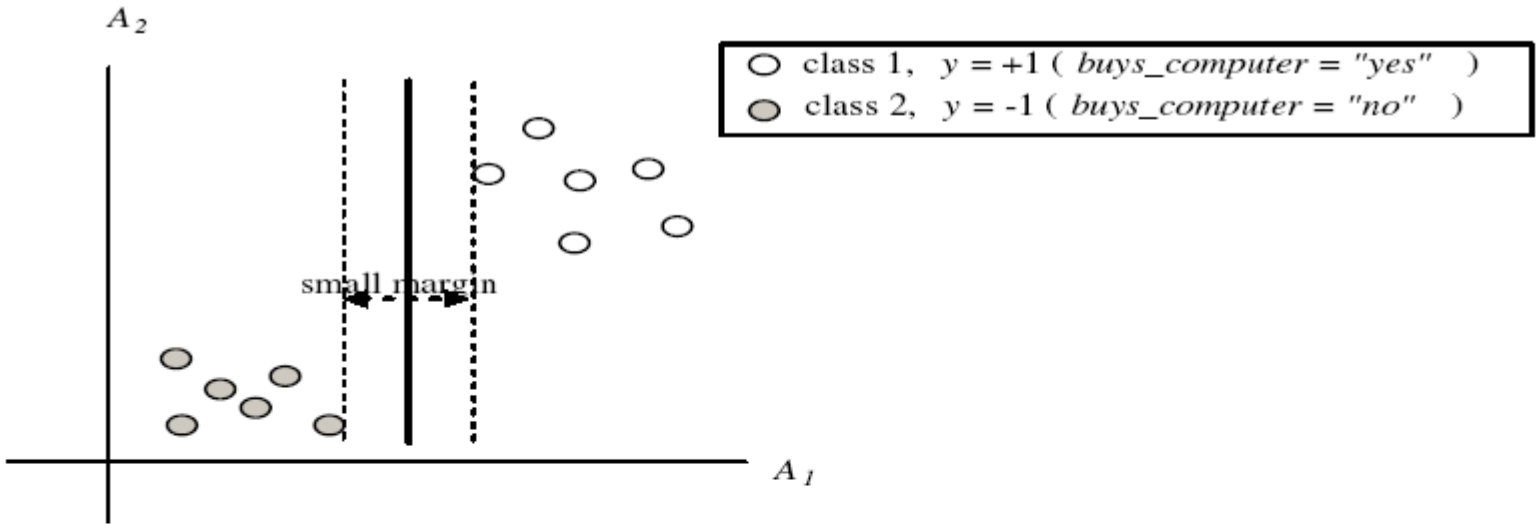


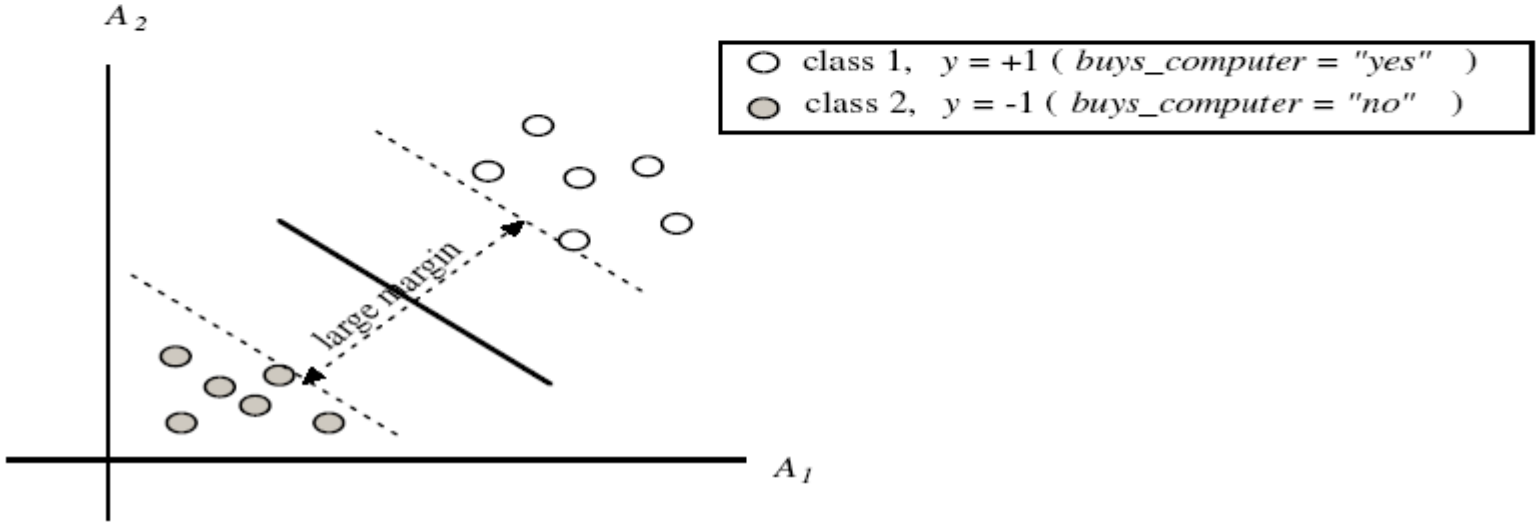
Let data D be $(\mathbf{X}_1, y_1), \ldots, (\mathbf{X}_{|D|}, y_{|D|})$, where $\mathbf{X}_i$ is the set of training tuples associated with the class labels y_i

There are infinite lines (hyperplanes) separating the two classes but we want to find the best one (the one that minimizes classification error on unseen data)

SVM searches for the hyperplane with the largest margin, i.e., **maximum marginal hyperplane** (MMH)

SVM—Margins and Support Vectors





SVM—Linearly Separable

- A separating hyperplane can be written as

  $\mathbf{W} \bullet \mathbf{X} + b = 0$

  where $\mathbf{W}=\{w_1, w_2, \ldots, w_n\}$ is a weight vector and b a scalar (bias)
- For 2-D it can be written as

  $w_0 + w_1 x_1 + w_2 x_2 = 0$
- The hyperplane defining the sides of the margin:

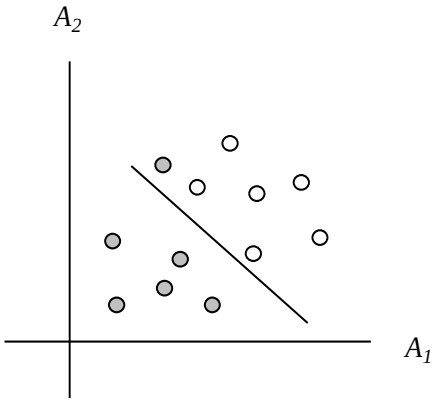
  $H_1: w_0 + w_1 x_1 + w_2 x_2 \geq 1$ for $y_i = +1$, and

  $H_2: w_0 + w_1 x_1 + w_2 x_2 \leq -1$ for $y_i = -1$
- Any training tuples that fall on hyperplanes H_1 or H_2 (i.e., the sides defining the margin) are **support vectors**
- This becomes a **constrained (convex) quadratic optimization** problem: Quadratic objective function and linear constraints → *Quadratic Programming (QP)* → Lagrangian multipliers method

Why Is SVM Effective on High Dimensional Data?

- The **complexity** of trained classifier is characterized by the # of support vectors rather than the dimensionality of the data
- The **support vectors** are the essential or critical training examples —they lie closest to the decision boundary, maximum marginal hyperplane (MMH)
- If all other training examples are removed and the training is repeated, the same separating hyperplane would be found
- The number of support vectors found can be used to compute an (upper) bound on the expected error rate of the SVM classifier, which is independent of the data dimensionality
- Thus, an SVM with a small number of support vectors can have good generalization, even when the dimensionality of the data is high

SVM—Linearly Inseparable



- Transform the original input data into a higher dimensional space

Example 6.8 Nonlinear transformation of original input data into a higher dimensional space. Consider the following example. A 3D input vector $\mathbf{X} = (x_1, x_2, x_3)$ is mapped into a 6D space Z using the mappings $\phi_1(X) = x_1, \phi_2(X) = x_2, \phi_3(X) = x_3, \phi_4(X) = (x_1)^2, \phi_5(X) = x_1x_2$, and $\phi_6(X) = x_1x_3$. A decision hyperplane in the new space is $d(\mathbf{Z}) = \mathbf{WZ} + b$, where $\mathbf{W}$ and $\mathbf{Z}$ are vectors. This is linear. We solve for $\mathbf{W}$ and b and then substitute back so that we see that the linear decision hyperplane in the new ($\mathbf{Z}$) space corresponds to a nonlinear second order polynomial in the original 3-D input space,

$$\begin{aligned} d(Z) &= w_1x_1 + w_2x_2 + w_3x_3 + w_4(x_1)^2 + w_5x_1x_2 + w_6x_1x_3 + b \\ &= w_1z_1 + w_2z_2 + w_3z_3 + w_4z_4 + w_5z_5 + w_6z_6 + b \end{aligned}$$

- Search for a linear separating hyperplane in the new space
- Computation is costly.

Using SVM for classification

"*Once I've got a trained support vector machine, how do I use it to classify test (i.e., new) tuples?*" Based on the Lagrangian formulation mentioned before, the MMH (maximum marginal hyperplane) can be rewritten as the decision bou

$$d(X^T) = \sum_{i=1}^{l} y_i \alpha_i X_i X^T + b_0, \tag{9.19}$$

where y_i is the class label of support vector X_i; X^T is a test tuple; α_i and b_0 are numeric parameters that were determined automatically by the optimization or SVM algorithm noted before; and l is the number of support vectors.

Given a test tuple, X^T, we plug it into Eq. (9.19), and then check to see the sign of the result. This tells us on which side of the hyperplane the test tuple falls. If the sign is positive, then X^T falls on or above the MMH, and so the SVM predicts that X^T belongs to class +1 (representing *buys_computer* = *yes*, in our case). If the sign is negative, then X^T falls on or below the MMH and the class prediction is −1 (representing *buys_computer* = *no*).

SVM vs. Neural Network

- **SVM**
  - Deterministic algorithm
  - Nice generalization properties
  - Hard to learn – learned in batch mode using quadratic programming techniques
  - Using kernels can learn very complex functions

- **Neural Network**
  - Nondeterministic algorithm
  - Generalizes well but doesn't have strong mathematical foundation
  - Can easily be learned in incremental fashion
  - To learn complex functions—use multilayer perceptron (nontrivial)

SVM Related Links

- SVM Website: http://www.kernel-machines.org/
- Representative implementations
  - **LIBSVM**: an efficient implementation of SVM, multi-class classifications, nu-SVM, one-class SVM, including also various interfaces with java, python, etc.
  - **SVM-light**: simpler but performance is not better than LIBSVM, support only binary classification and only in C
  - **SVM-torch**: another recent implementation also written in C

Outline

- Feature Selection and Engineering
- Bayesian Belief Networks
- Support Vector Machines
- **Rule based and pattern based classification**
- Classification by weak supervision
- Classification with rich data type
- Other Related Methods
- Summary

Rule-based classification

- A classifier is a set of rules
- A rule based classifier uses a set of IF THEN rules for classification.
- An IF-THEN rule is an expression of the form

  IF condition THEN conclusion

- Example

R1: IF age=youth AND student=yes THEN buys_computer= yes

- The IF part of the rule is antecedent or precondition
- The THEN part of the rule is the rule consequent.

Using IF-THEN Rules for Classification

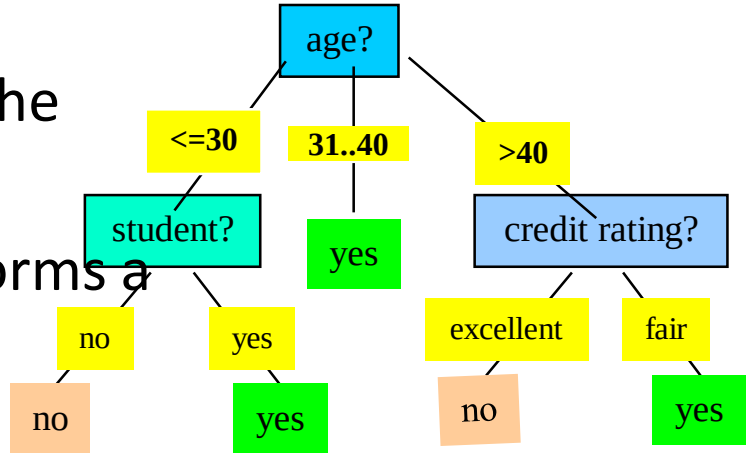
- Assessment of a rule: *coverage* and *accuracy*
  - n_{covers} = # of tuples covered by R
  - $n_{correct}$ = # of tuples correctly classified by R

  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 the highest priority to the 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

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

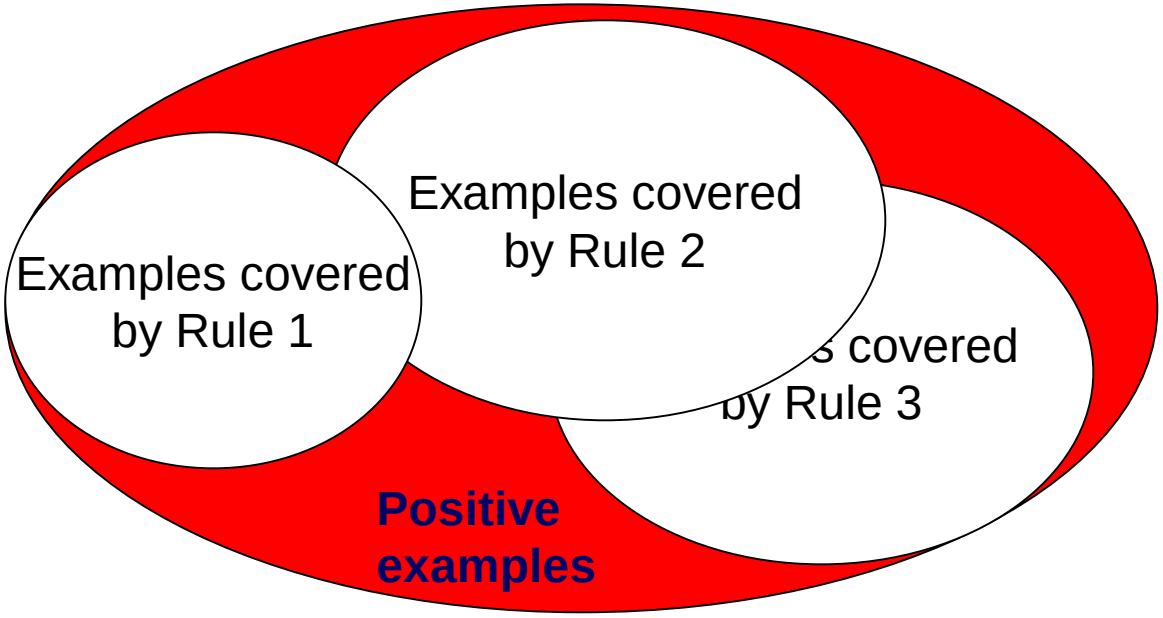
IF *age* = young AND *student* = *no* THEN *buys_computer* = *no*
IF *age* = young AND *student* = *yes* THEN *buys_computer* = *yes*
IF *age* = mid-age THEN *buys_computer* = *yes*
IF *age* = old AND *credit_rating* = *excellent* THEN *buys_computer* = *no*
IF *age* = old AND *credit_rating* = *fair* THEN *buys_computer* = *yes*

Rule Induction: Sequential Covering Method

- Sequential covering algorithm: Extracts rules directly from training data
- Typical sequential covering algorithms: FOIL, AQ, CN2, RIPPER
- Rules are learned *sequentially*, each for a given class C_i will cover many tuples of C_i but none (or few) of the tuples of other classes
- Steps:
  - Rules are learned one at a time
  - Each time a rule is learned, the tuples covered by the rules are removed
  - Repeat the process on the remaining tuples until *termination condition*, e.g., when no more training examples or when the quality of a rule returned is below a user-specified threshold
- Comp. w. decision-tree induction: learning a set of rules *simultaneously*

Sequential Covering Algorithm

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



Rule Generation

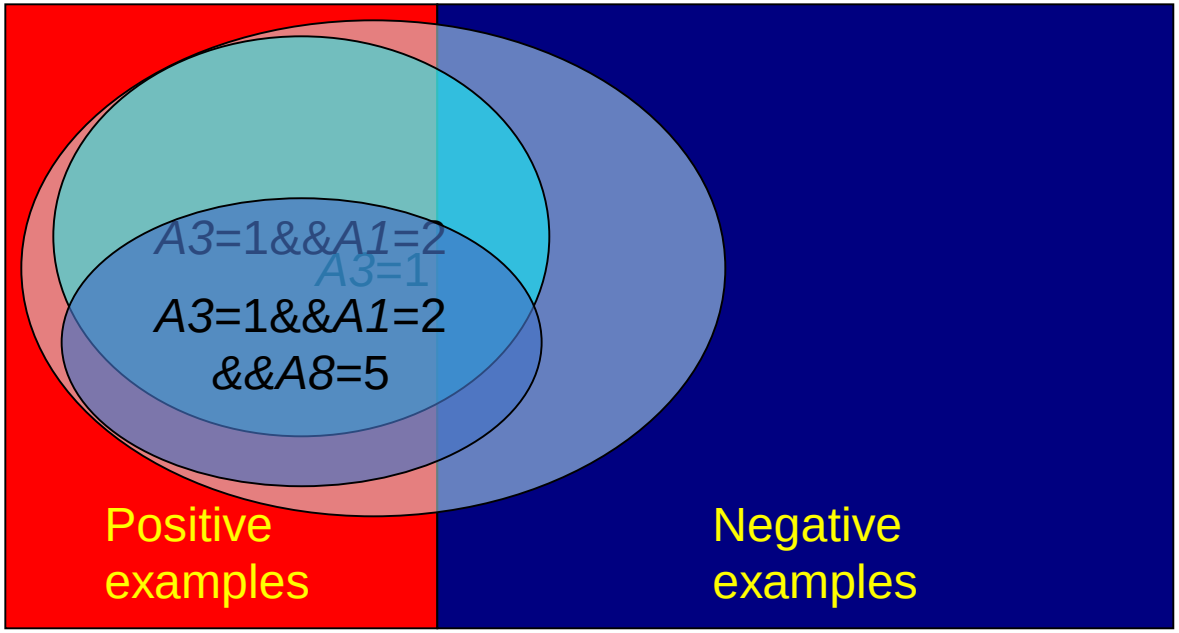
- To generate a rule

  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
- FOIL- First Order Inductive Learner
- Rule-Quality measures: consider both coverage and accuracy
  - Let **pos** and **neg** be the number of positive and negative tuples learned by R and **pos'** and **neg'** be the number of positive and negative tuples learned by R'
  - 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. Given R

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

Pos/neg are # of positive/negative tuples covered by R.
If *FOIL_Prune* is higher for the pruned version of R, prune R

Associative Classification

- Associative classification: Major steps
  - Mine data to find strong associations between frequent patterns (conjunctions of attribute-value pairs) and class labels
  - Association rules are generated in the form of

    $P_1 \wedge p_2 \ldots \wedge p_l \rightarrow$ "$A_{class} = C$" (conf, sup)
  - Organize the rules to form a rule-based classifier
- Why effective?
  - It explores highly confident associations among multiple attributes and may overcome some constraints introduced by decision-tree induction, which considers only one attribute at a time
  - Associative classification has been found to be often more accurate than some traditional classification methods, such as C4.5

Typical Associative Classification Methods

- **CBA** (Classification Based on Associations: Liu, Hsu & Ma, KDD'98)
  - Mine possible association rules in the form of
    - Cond-set (a set of attribute-value pairs) → class label
  - Build classifier: Organize rules according to decreasing precedence based on confidence and then support
- **CMAR** (Classification based on Multiple Association Rules: Li, Han, Pei, ICDM'01)
  - Classification: Statistical analysis on multiple rules
- **CPAR** (Classification based on Predictive Association Rules: Yin & Han, SDM'03)
  - Generation of predictive rules (FOIL-like analysis) but allow covered rules to retain with reduced weight
  - Prediction using best k rules
  - High efficiency, accuracy similar to CMAR

Quiz 2 (Class #21): Till the preceding slide

Discriminative Frequent Pattern Classification

- This topic is not included.

Outline

- Feature Selection and Engineering
- Bayesian Belief Networks
- Support Vector Machines
- Rule based and pattern based classification
- **Classification by weak supervision**
- Classification with rich data type
- Other Related Methods
- Summary

Weak Supervision (Class #22)

- SVM, Logistic regression, k-NN
  - Strong supervision: require large number of high quality training tuples annotated by domain experts.
- In real work, training set is not available.
  - Document classification, speech recognition, computer vision
  - One minute of speech requires 10 minutes to label
  - For annotating sounds, it takes 400 times more
- Classification approaches are required for limited number of training samples or no training samples.

Semi-Supervised Classification

- Semi-supervised: Uses labeled and unlabeled data to build a classifier
- Self-training:
  - Build a classifier using the labeled data
  - Use it to label the unlabeled data, and those with the most confident label prediction are added to the set of labeled data
  - Repeat the above process
  - Adv: easy to understand; disadv: may reinforce errors
- Co-training: Use two or more classifiers to teach each other
  - Each learner uses a mutually independent set of features of each tuple to train a good classifier, say f1 and f2
  - Then f1 and f2 are used to predict the class label for unlabeled data X
  - **Teach each other: The tuple having the most confident prediction from f_1 is added to the set of labeled data for f_2, & vice versa**
  - Example: Web page classification
    - Text features (f1) and image-based features (f2)

Self-training

1. Select a learning method such as Bayesian classification. Build the classifier using the labeled data, X_l.
2. Use the classifier to label the unlabeled data, X_u.
3. Select the tuple $x \in X_u$ having the highest confidence (most confident prediction). Add it and its predicted label to X_l.
4. Repeat (i.e., retrain the classifier using the augmented set of labeled data).

Cotraining

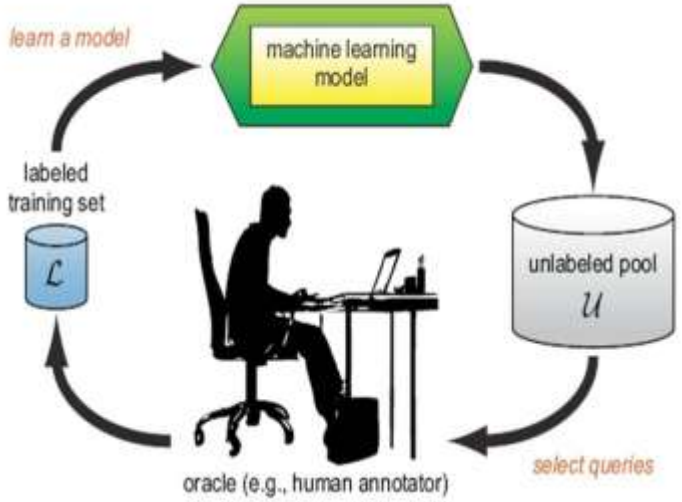
1. Define two separate nonoverlapping feature sets for the labeled data, X_l.
2. Train two classifiers, f_1 and f_2, on the labeled data, where f_1 is trained using one of the feature sets and f_2 is trained using the other.
3. Classify X_u with f_1 and f_2 separately.
4. Add the most confident $(x, f_1(x))$ to the set of labeled data used by f_2, where $x \in X_u$. Similarly, add the most confident $(x, f_2(x))$ to the set of labeled data used by f_1.
5. Repeat.

FIGURE 7.16

Self-training and cotraining methods of semisupervised classification.

Active Learning

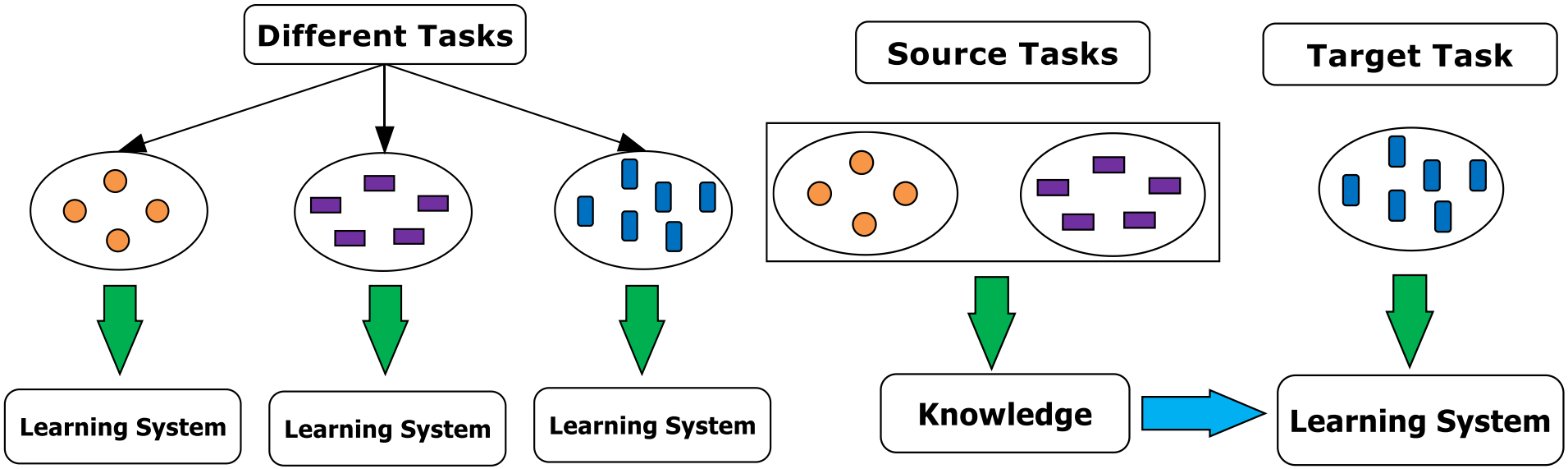
- Class labels are expensive to obtain
- Active learner: query human (oracle) for labels
- Pool-based approach: Uses a pool of unlabeled data
  - L: a small subset of D is labeled, U: a pool of unlabeled data in D
  - Use a query function to carefully select one or more tuples from U and request labels from an oracle (a human annotator)
  - The newly labeled samples are added to L, and learn a model
  - Goal: Achieve high accuracy using as few labeled data as possible



- Evaluated using *learning curves*: Accuracy as a function of the number of instances queried (# of tuples to be queried should be small)
- Research issue: How to choose the data tuples to be queried?
  - Uncertainty sampling: choose the least certain ones
  - Query by committee:
    - constructs multiple (say five) classification models
    - Selects the unlabeled tuple that constructed classification models have the most disagreement in terms of its predicted class labels
  - Reduce expected entropy over U: Find the greatest reduction in the total number of incorrect predictions

Transfer Learning: Conceptual Framework

- Transfer learning: Extract knowledge from one or more source tasks and apply the knowledge to a target task
  - Classification of camera reviews to classification of TV reviews
- Traditional learning: Build a new classifier for each new task
- Transfer learning: Build new classifier by applying existing knowledge learned from source tasks



Transfer Learning: Methods and Applications

- Applications: Especially useful when data is outdated or distribution changes, e.g., Web document classification, e-mail spam filtering
- *Instance-based transfer learning*: Reweight some of the data from source tasks and use it to learn the target task
- TrAdaBoost (Transfer AdaBoost)
  - Assume source and target data each described by the same set of attributes (features) & class labels, but rather diff. distributions
  - Require only labeling a small amount of target data
  - Use source data in training: When a source tuple is misclassified, reduce the weight of such tupels so that they will have less effect on the subsequent classifier
- Research issues
  - Negative transfer: When it performs worse than no transfer at all
  - Heterogeneous transfer learning: Transfer knowledge from different feature space or multiple source domains
  - Large-scale transfer learning

Distant Supervision

- Automatic generation of large number of labels.
- Exploits external knowledge base to automatically generate labels.
- Examples:
  - Classification of tweets
    - :) sign means negative :(sign means positive
    - If a tweet contains a URL, one can consider URL category as a label
  - Classification of tweets into different categories (e.g., news, health, science, games, etc.)
    - We could explore the Open Directory Project (ODP, http://odp.org), which maintains a directory for web links by volunteers.
    - If a tweet contains a url (e.g., http://nytimes.com), we can automatically find its ODP category (e.g., news), which is treated as the label of the corresponding tweet.
    - A large labeled training set can be generated automatically
- Issues: it is noisy

Zero-shot learning

- Example:
  - Suppose that we have a collection of animal images, each of which has a unique label, including “owl,” “dog,” or “fish.”
  - Using this training data set, we can build a classifier, say SVMs or logistic regression classifier.
  - We can use the trained classifier to predict its class label, that is, which one of the three possible animals (owl, dog, or fish).
- *But, what if the test image is actually about a cat?*
- In other words, the class label of the test data *never* appears in the training data.
- *zero-shot learning*
  - Zero-shot learning (ZSL) is a machine learning technique that allows models to recognize and categorize new concepts without having seen examples of those concepts before.
- Solution:
  - we might have some high-level description about the novel classes.
    - For example, for “cat,” we can learn from the Wikipedia that a cat has retractable claws and super night vision.
  - Zero-shot learning tries to leverage such external knowledge or side-information to build a classifier that can recognize such novel class labels

Zero-shot learning: example

- n training images each of which is represented by a d-D feature vector and a 3-D label vector.
  - The label vector indicates which of the three known classes the training image belongs to. For example, for an image about a "dog," its label vector is [1,0,0].
- In addition, we have the external knowledge about the class label, where each class label (animal) can be described by four semantic attributes. including whether the animal "has four legs," "has wings," "has retractable claws," and "has super night vision."
  - For example, since a dog has four legs, but no wings or retractable claws or super night vision, the class label "dog" can be described by a 4-D semantic attribute vector [1,0,0,0].
- Likewise, the class label "cat" can be described by a 4-D semantic attribute vector [1,0,1,1], meaning that a cat has four legs, retractable claws and super night vision but no wings.
- Such external knowledge is available for both known class labels (e.g., "owl," "dog," and "fish") and novel class labels (e.g., "cat" and "rooster")
- We train a Semantic Attribute Classifier F, which predicts a 4-D semantic attribute vector for an input image represented by a d-dimensional feature vector.
- We can employ a two-level neural network
- Applications
  - Neural activity recognition

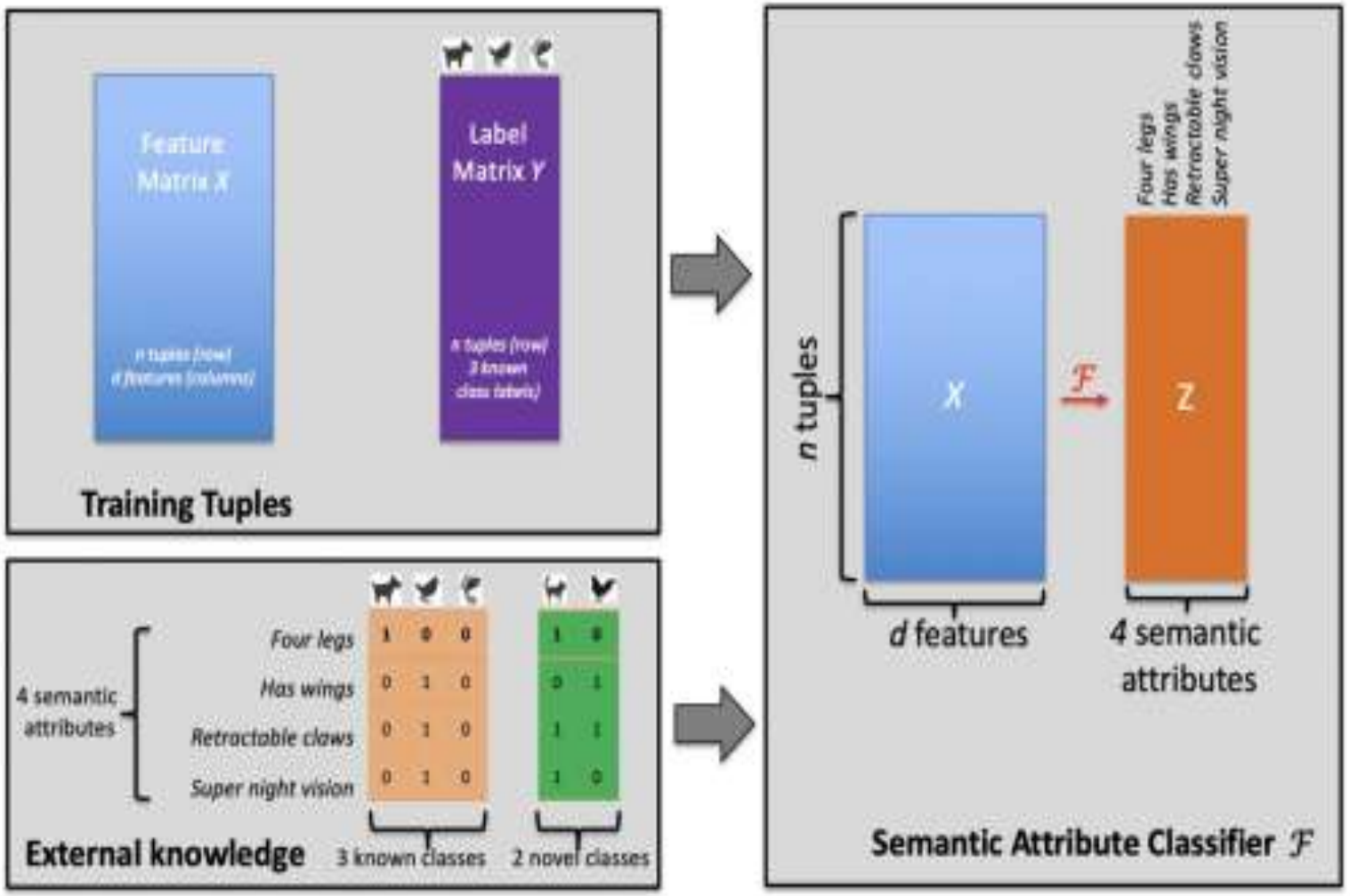


FIGURE 7.19

Top left: input n training tuples in d-dimensional feature space, each of which is labeled by one of the three known classes (i.e., "dog," "owl," and "fish"). Bottom left: external knowledge where each known and novel class is described by four semantic attributes. Right: the trained semantic attribute classifier $\mathcal{F}$.

Outline

- Feature Selection and Engineering
- Bayesian Belief Networks
- Support Vector Machines
- Rule based and pattern based classification
- Classification by weak supervision
- **Classification with rich data type**
- Other Related Methods
- Summary

Classification with rich data type

- Assumption so far
  - Each training tuple is represented by a feature vector and a class label. We will predict the class label
  - Each training tuple can be considered as a data point in the feature space.
- Other types of data
  - Stream data, sequence data, graph data, grid (image) data, spatial-temporal data (video)
  - Deep learning techniques are powerful tools to classify rich data types.

Stream data classification

- Transactions arrive sequentially
  - Example bank transactions
  - A sequence of transactions may be fraud.
- Classifier has to classify the transactions and update the labels confirmed by bank expert.
- Challenges
  - Speed of data arrival
  - Impossible to store all data tuples/transactions
  - One-pass constraint
- Ensemble is effective method
- Applications
  - Marketing, network monitoring and sensor networks

Ensemble method

- Partition the data into equal sized chunks (i=0,1,…,k) where i=0 is the current i=k is the oldest chunk
- Each chunk i has a training set and labels for I
- We train each classifier (example: Naïve Bayes) which gives posterior probability of a tuple belongs to a particular class.
- The output is the weighted sum of all classifiers.
- When a new chunk comes, we train a new classifier.
- Use newly arrived chunk as a test set and evaluate the performance of k+1 classifiers.
  - We select k classifiers with lowest classification error.
  - Discard the classifier with highest error rate.

Sequence Classification

- A sequence is an ordered list of values.
  - Example: sentence, DNA segment (value is one of the amino acid: A,C,G,T.
- Goal of sequence mining
  - Predict a label of a sequence
    - Sentiment of a sentence, gene coding are/noncoding area, high value/ordinary customer
- Approach:
  - Convert the input feature into a vector of features and fed into a conventional classifier.
- K-nearest neighbour classifier with commonly used distance measures gives competitive performance
- Selection of features is important
  - Recurrent neural networks is powerful technique
- Number of features may be huge
  - Feature engineering plays a major role.

Sequence Classification

	Unigrams				Bigrams															
Index	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Candidate features	A	C	G	T	AC	AG	AT	CG	CT	CA	GA	GC	GT	TA	TC	TG	AA	CC	GG	TT
Feature vector (binary)	1	1	1	1	1	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0
Feature vector (frequency)	1	5	1	1	1	0	0	1	0	0	0	0	1	0	0	0	0	4	0	0

FIGURE 7.21

An example of using *n*-gram for sequence data feature engineering. Given a DNA segment "ACCCCCGT," we want to convert it to a feature vector, using *n*-grams. There are 20 candidate features in total, including 4 unigrams and 16 bigrams. For the binary feature vector (the third row), an element indicates whether the corresponding feature appears (1) or not (0) in the input sequence. For the weighted feature vector (the fourth row), an element indicates how many times (the frequency) that the corresponding feature appears in the input sequence.

Graph data classification

- Graph is a ubiquitous data type
- Graph classification: predict the label of nodes (node-level classification) or entire graph (graph level classification).
- Node-level classification
  - Given a web graph (with pages and hyperlinks), predict a category of a page
- Graph-level classification
  - Given a collection f molecular graphs, predict whether a given molecule is toxic.
- Two approaches: feature engineering or proximity measures
- Feature Engineering
  - Extract a set of features for each node or a graph and feed into conventional classifier.
  - Nodel–level: number of neighbouring nodes linked to the given node, node importance measure, node clustering coefficient
  - Graph-level: size of the graph (nodes, edges, weights), diameter of the graph, tital number of triangles in the graph
    - Recent technique: Graph neural networks

Outline

- Feature Selection and Engineering
- Bayesian Belief Networks
- Support Vector Machines
- Rule based and pattern based classification
- Classification by weak supervision
- Classification with rich data type
- **Other Related Methods**
- Summary

Other methods

- Multiclass classification
  - How to build a classifier for multiple classes?
- Distance metric learning
  - How to learn the distance automatically?
- How to render interpretability?
- Genetic algorithm
- Reinforcement learning
  - Learning receives evaluation feedback

Multiclass Classification

- Classification involving more than two classes (i.e., > 2 Classes)
- Method 1. **One-vs.-all** (OVA): Learn a classifier one at a time
  - Given m classes, train m classifiers: one for each class
  - Classifier j: treat tuples in class j as *positive* & all others as *negative*
  - To classify a tuple **X**, the set of classifiers vote as an ensemble
- Method 2. **All-vs.-all** (AVA): Learn a classifier for each pair of classes
  - Given m classes, construct m(m-1)/2 binary classifiers
  - A classifier is trained using tuples of the two classes
  - To classify a tuple **X**, each classifier votes. X is assigned to the class with maximal vote
- Comparison
  - All-vs.-all tends to be superior to one-vs.-all
  - Problem: Binary classifier is sensitive to errors, and errors affect vote count

Error-Correcting Codes for Multiclass Classification

- Originally designed to correct errors during data transmission for communication tasks by exploring data redundancy
- Example
  - A 7-bit codeword associated with classes 1-4

Class	Error-Corr. Codeword						
C_1	1	1	1	1	1	1	1
C_2	0	0	0	0	1	1	1
C_3	0	0	1	1	0	0	1
C_4	0	1	0	1	0	1	0

  - Given a unknown tuple **X**, the 7-trained classifiers output: 0001010
  - Hamming distance: # of different bits between two codewords
  - $H(\mathbf{X}, C_1) = 5$, by checking # of bits between [1111111] & [0001010]
  - $H(\mathbf{X}, C_2) = 3$, $H(\mathbf{X}, C_3) = 3$, $H(\mathbf{X}, C_4) = 1$, thus C_4 as the label for **X**
- Error-correcting codes can correct up to (h-1)/h 1-bit error, where h is the minimum Hamming distance between any two codewords
- If we use 1-bit per class, it is equiv. to one-vs.-all approach, the code are insufficient to self-correct
- When selecting error-correcting codes, there should be good row-wise and col.-wise separation between the codewords

Interpretability

- Interpretability: The models' ability to explain the classification results or process in a user-understandable way.
- Decision trees are interpretable; Linear classifiers are interpretable due to the weight of the attribute.
- Approach: LIME: Local Interpretable Model-agnostic Explanation

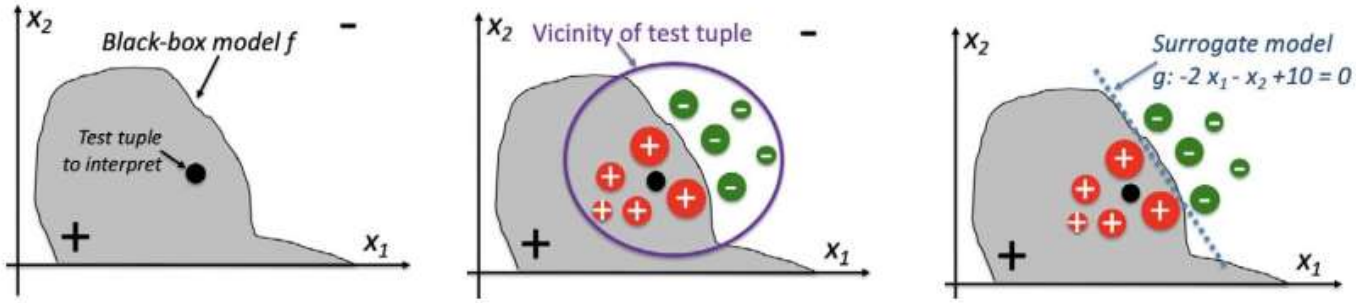


FIGURE 7.28

An example of LIME: Local Interpretable Model-agnostic Explanation.

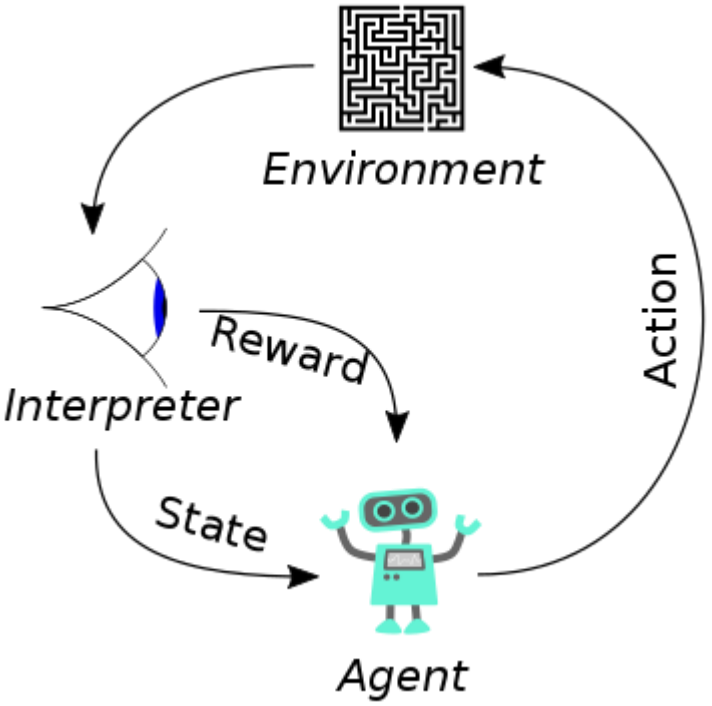
- Beyond classification, interpretability also plays an important role in other data mining tasks, including clustering, outlier detection, ranking, and recommendation. In addition to making the data mining models transparent so as to gain the users' trust of the model, interpretability is also intimately related to other important aspects of data mining, such as fairness, robustness, causality, privacy.

Genetic Algorithms (GA)

- Genetic Algorithm: based on an analogy to biological evolution
- An initial **population** is created consisting of randomly generated rules
  - Each rule is represented by a string of bits
  - E.g., if A_1 and $\neg A_2$ then C_2 can be encoded as 100
  - If an attribute has k > 2 values, k bits can be used
- Based on the notion of survival of the **fittest**, a new population is formed to consist of the fittest rules and their offspring
- The ***fitness of a rule*** is represented by its classification accuracy on a set of training examples
- Offspring are generated by ***crossover*** and ***mutation***
- The process continues until a population P evolves *when each rule in P satisfies a prespecified threshold*
- Slow but easily parallelizable

Reinforcement learning

- In reinforcement learning, the learning agent tries to figure out what to do (e.g., choose the best product to advertise in order to maximize the overall increased revenue) by interacting with the environment (e.g., trying a few different advertisements, observing their rewards, and adjusting the advertisement for next day accordingly).
- In classification, classifier receives the *instructive feedback* (i.e., the true labels for the training tuples) in order to construct the best classifier.
- In reinforcement learning, the learning agent receives *evaluative feedback*

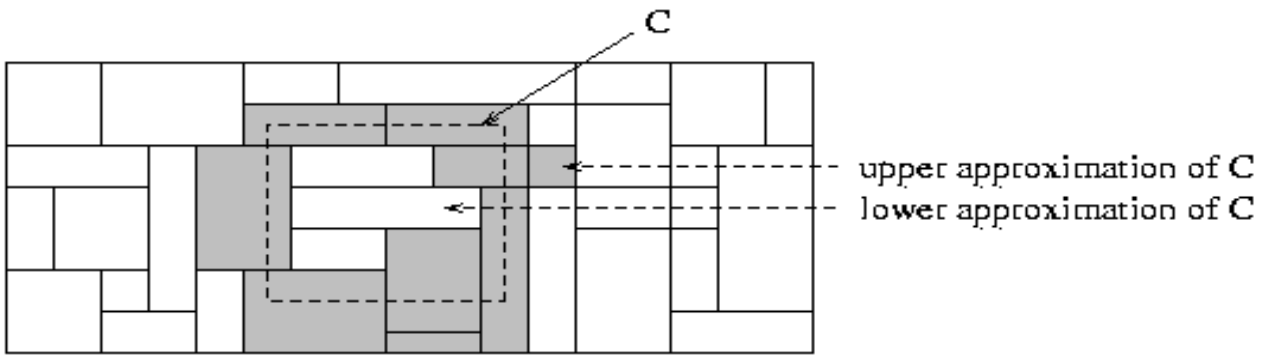


Outline

- Feature Selection and Engineering
- Bayesian Belief Networks
- Support Vector Machines
- Rule based and pattern based classification
- Classification by weak supervision
- Classification with rich data type
- Other Related Methods
- Summary

Rough Set Approach

- Rough sets are used to **approximately or "roughly" define equivalent classes**
- A rough set for a given class C is approximated by two sets: a lower approximation (certain to be in C) and an upper approximation (cannot be described as not belonging to C)
- Finding the minimal subsets (**reducts**) of attributes for feature reduction is NP-hard but a **discernibility matrix** (which stores the differences between attribute values for each pair of data tuples) is used to reduce the computation intensity



Fuzzy sets

- The word "fuzzy" means "vaguness (ambiguity)".
- Fuzziness occurs when the boundary of a piece of information is not clear-cut.
- Fuzzy sets - 1965 Lotfi Zadeh as an extension of classical notation set.
- Classical set theory allows the membership of the elements in the set in **binary terms**.
- Fuzzy set theory permits membership function valued in the interval [0,1].

Fuzzy sets

Example:

Words like young, tall, good or high are fuzzy.

- There is no single quantitative value which defines the term young.
- For some people, age 25 is young, and for others, age 35 is young.
  The concept young has no clean boundary.
- Age 35 has some possibility of being young and usually
- depends on the context in which it is being considered.

Fuzzy set theory is an extension of classical set theory where elements have degree of membership.

Introduction

- In real world, there exist much fuzzy knowledge (i.e. vague, uncertain inexact etc).
- Human **thinking** and **reasoning** (analysis, logic, interpretation) frequently involved **fuzzy** information.
- Human can give satisfactory answers, which are probably true.
- Our systems are unable to answer many question because the systems are designed based upon classical set theory (Unreliable and incomplete).

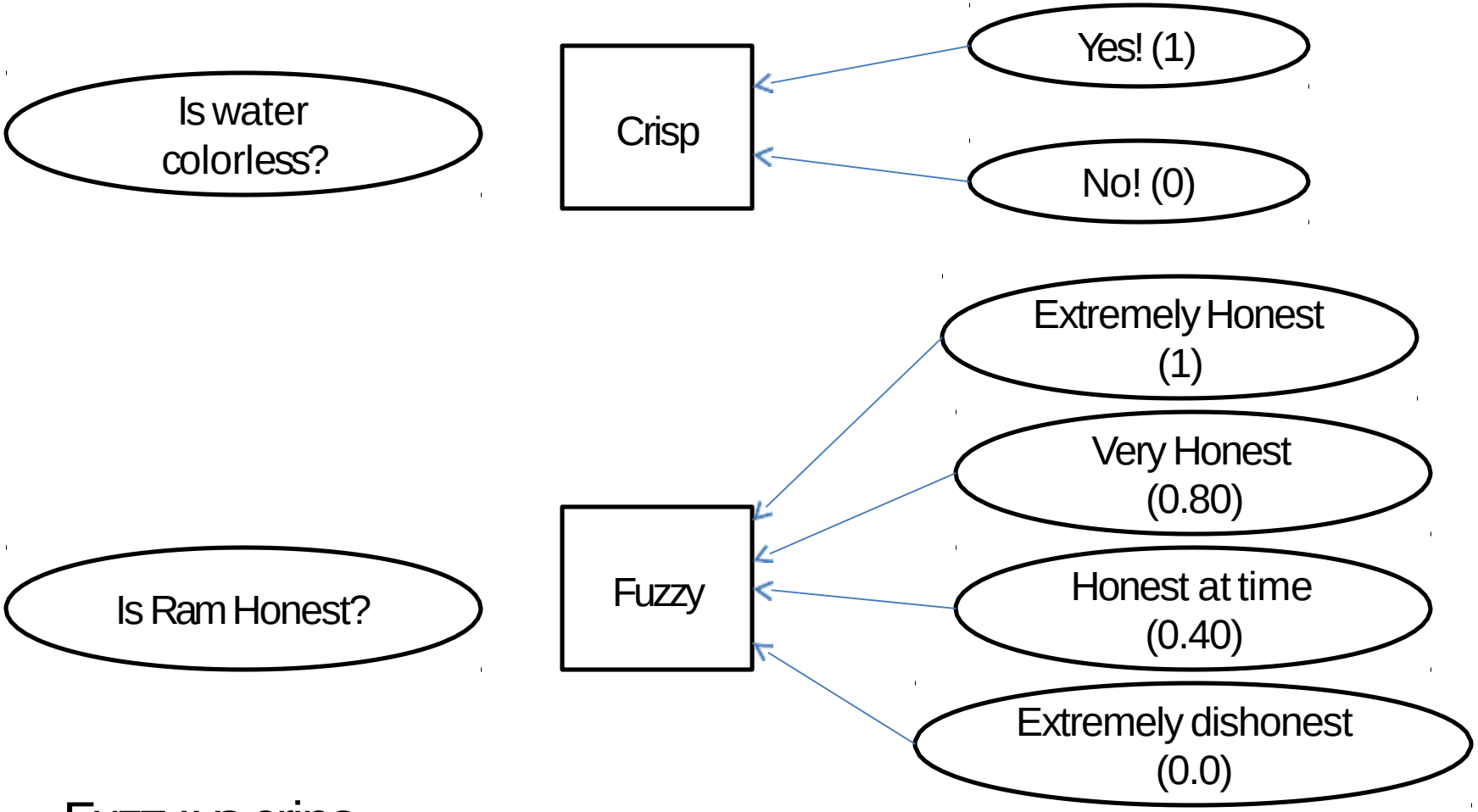
- We want, our system should be able to cope with unreliable and incomplete information.
- Fuzzy system have been provide solution.

Introduction

Classical set theory	Fuzzy set theory
• Classes of objects with sharp boundaries.	• Classes of objects with un- sharp boundaries.
• A classical set is defined by crisp(exact) boundaries, i.e., there is no uncertainty about the location of the set boundaries.	• A fuzzy set is defined by its ambiguous boundaries, i.e., there exists uncertainty about the location of the set boundaries.
• Widely used in digital system design	• Used in fuzzy controllers.

Introduction (Continue)

Example



Fuzzy vs crips

Classical set theory

- A Set is any well defined collection of objects.
- An object in a set is called an element or member of that set.
  - Sets are defined by a simple statement,
  - Describing whether a particular element having a certain
  - propertybelongs to that particular set.

$$A = \{a_1, a_2, a_3, \ldots\ldots, a_n\}$$

- If the elements a_i $(i = 1,2,3,\ldots,n)$ of a set A are subset of universal set X, then set A can be represented for all elements $x \epsilon X$ by its characteristics function

$$\mu_A(x) = 1 \text{ if } x \epsilon X \text{ otherwise } 0$$

Operations on classical set theory

Union: the union of two sets A and B is given as

$$A \cup B = \{ x \mid x \in A \text{ or } x \in B \}$$

Intersection: the intersection of two sets A and B is given as

$$A \cap B = \{ x \mid x \in A \text{ and } x \in B \}$$

Complement: It is denoted by $\tilde{A}$ and is defined as

$$\tilde{A} = \{ x \mid x \text{ does not belongs } A \text{ and } x \in X \}$$

Fuzzy Sets

- Fuzzy sets theory is an extension of classical set theory.
- Elements have varying degree of membership. A logic based on two truth values,
- ***True*** and ***False*** is sometimes insufficient when describing human reasoning.
- Fuzzy Logic uses the whole interval between 0 (false) and 1 (true) to describe human reasoning.
- A Fuzzy Set is any set that allows its members to have different degree of membership, called **membership function**, having interval [0,1].

Fuzzy Sets

- **Fuzzy Logic** is derived from fuzzy set theory
- Many degree of membership (between 0 to 1) are allowed.
- Thus a membership function $\mu_A(x)$ is associated with a fuzzy sets Ã such that the function maps every element of universe of discourse X to the interval [0,1].
- The mapping is written as: $\mu_{\tilde{A}}(x)$: X □ [0,1].

- Fuzzy Logic is capable of handing inherently imprecise (vague or inexact or rough or inaccurate) concepts

Fuzzy Sets

- **Fuzzy set** is defined as follows:

- If X is an universe of discourse and x is a particular element of X, then a fuzzy set A defined on X and can be written as a collection of ordered pairs

$$A = \{(x, \mu_{\tilde{A}}(x)), x \epsilon X\}$$

Fuzzy Sets (Continue)

Example

- Let $X = \{g_1, g_2, g_3, g_4, g_5\}$ be the reference set of students.
- Let $\tilde{A}$ be the fuzzy set of "smart" students, where "smart" is fuzzy term.

$$\tilde{A} = \{(g_1,0.4)(g_2,0.5)(g_3,1)(g_4,0.9)(g_5,0.8)\}$$

Here $\tilde{A}$ indicates that the smartness of g_1 is 0.4 and soon

Fuzzy Sets (Continue)

Membership Function

- The membership function fully defines the fuzzyset
- A membership function provides a measure of *the degree of similarity* of an element to a fuzzyset

Membership functions can

- either be chosen by the user arbitrarily, based on the user's experience (MF chosen by two users could be different depending upon their experiences, perspectives, etc.)
- Or be designed using machine learning methods (e.g., artificial neural networks, genetic algorithms, etc.)

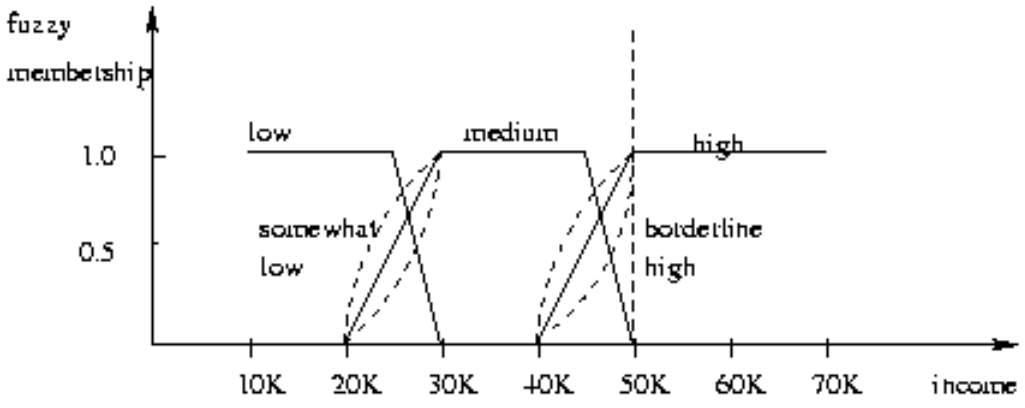
Fuzzy Sets (Continue)

There are different shapes of membership functions;

- **Triangular,**
- **Trapezoidal,**
- **Gaussian, etc**

Fuzzy Set Classification

- Fuzzy logic uses truth values between 0.0 and 1.0 to represent the degree of membership (such as in a *fuzzy membership graph*)
- Attribute values are converted to fuzzy values. Ex.:
  - Income, *x*, is assigned a fuzzy membership value to each of the discrete categories {low, medium, high}, e.g. $49K belongs to "medium income" with fuzzy value 0.15 but belongs to "high income" with fuzzy value 0.96
  - Fuzzy membership values do not have to sum to 1.
- Each applicable rule contributes a vote for membership in the categories
- Typically, the truth values for each predicted category are summed, and these sums are combined



Clustering Analysis: basic concepts and methods

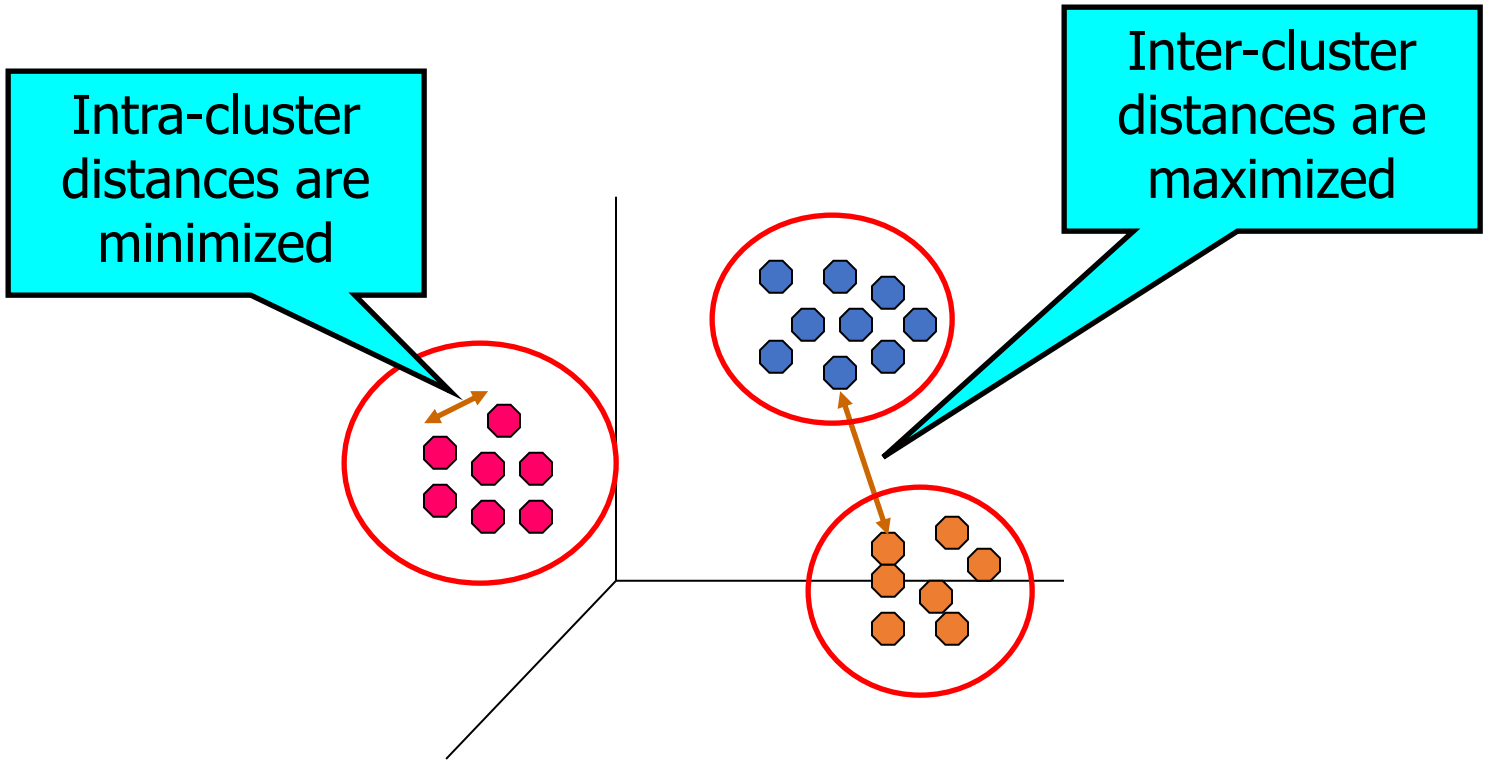
P. Krishna Reddy, IIIT Hyderabad

Outline (Class #23)

- **Cluster Analysis: Basic Concepts**
- Partitioning Methods
- Hierarchical Methods
- Density-Based Methods
- Grid-Based Methods
- Evaluation of Clustering
- Summary

What is Cluster Analysis?

- Finding groups of objects such that the objects in a group will be similar (or related) to one another and different from (or unrelated to) the objects in other groups



About clustering

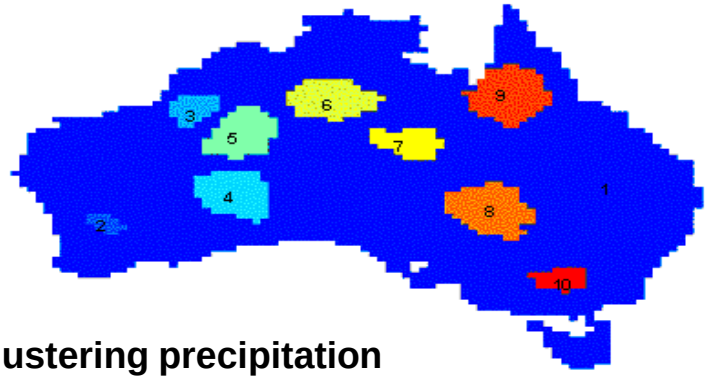
- Human being may have one or few perspectives on the data
- But, different clustering methods with different similarity methods may lead to the discovery of previously unknown groups in the data
  - Interesting knowledge

Applications of Cluster Analysis

- **Understanding**
  - Group related documents for browsing, group genes and proteins that have similar functionality, or group stocks with similar price fluctuations

- **Summarization**
  - Reduce the size of large data sets

	Discovered Clusters	*Industry Group*
1	Applied-Matl-DOWN,Bay-Network-Down,3-COM-DOWN, Cabletron-Sys-DOWN,CISCO-DOWN,HP-DOWN, DSC-Comm-DOWN,INTEL-DOWN,LSI-Logic-DOWN, Micron-Tech-DOWN,Texas-Inst-Down,Tellabs-Inc-Down, Natl-Semiconduct-DOWN,Oracl-DOWN,SGI-DOWN, Sun-DOWN	Technology1-DOWN
2	Apple-Comp-DOWN,Autodesk-DOWN,DEC-DOWN, ADV-Micro-Device-DOWN,Andrew-Corp-DOWN, Computer-Assoc-DOWN,Circuit-City-DOWN, Compaq-DOWN, EMC-Corp-DOWN, Gen-Inst-DOWN, Motorola-DOWN,Microsoft-DOWN,Scientific-Atl-DOWN	Technology2-DOWN
3	Fannie-Mae-DOWN,Fed-Home-Loan-DOWN, MBNA-Corp-DOWN,Morgan-Stanley-DOWN	Financial-DOWN
4	Baker-Hughes-UP,Dresser-Inds-UP,Halliburton-HLD-UP, Louisiana-Land-UP,Phillips-Petro-UP,Unocal-UP, Schlumberger-UP	Oil-UP



Clustering precipitation in Australia

Examples of Clustering Applications

- **Biology:** Biologists have spent many years creating a taxonomy of all living things.
  - Recently, clustering has been used find groups of genes that have similar functions.
- **Information Retrieval**: Clustering is used to group thousands of search results extracted for a given query from billions of web pages.
  - Example: Web pages returned for query “Movie” can be grouped into categories such as reviews, trailers, starts and theaters. Each category can be broken into sub-categories.
- **Climate**: finding patterns in atmosphere and ocean
  - Cluster analysis has been applied to find patterns in the atmospheric pressure of polar regions and areas of ocean that have significant impact on land climate.
- **Psychology and Medicine:** Clustering analysis can be used to understand different subcategories of illness.
  - Example: cluster analysis can be used to identify different types of depression.
  - Cluster analysis can be used to detect patterns in the spatial and temporal distribution of a disease.

Broad Applications of Cluster Analysis

- Data summarization, compression, and reduction
  - Examples: Image processing or vector quantization
- Collaborative filtering, recommendation systems, or customer segmentation
  - Finding like-minded users or similar products
- Dynamic trend detection
  - Clustering stream data and detecting trends and patterns
- Multimedia data analysis, biological data analysis, and social network analysis
  - Examples: Clustering video/audio clips or gene/protein sequences
- A key intermediate step for other data mining tasks
  - Generating a compact summary of data for classification, pattern discovery, and hypothesis generation and testing
  - Outlier detection: Outliers are those “far away” from any cluster

Clustering for Utility

- Cluster analysis provides an abstraction from individual data objects into the clusters.
- Some clustering algorithms characterize each cluster in terms of cluster prototype.
  - A cluster prototype is a data object that is representative of the other objects in the cluster.

Clustering for Utility

- The cluster prototype can be used as the basis for a number of data analysis or data processing techniques.
  - Summarization:
    - Regression and PCA have a time and space complexity of $O(m^2)$, where m is a number of objects. Not practical for large data sets.
    - It can be applied to reduced data set with cluster prototypes.
  - Compression
    - Clustering prototypes can be used in data compression.
    - A table is created that consists of the prototypes for each cluster. Each object is represented by the index of the prototype associated with the cluster. It is called vector quantization and applied to image, sound, video clippings where
      - Many data objects are highly similar
      - Loss of information is acceptable
      - Substantial reduction in data size is needed
  - Efficiently finding nearest neighbors
    - Finding nearest neighbors require computing pair-wise distance between all pairs.
    - We can use prototypes to reduce the number of computations.

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 by the method
  - its implementation, and
  - Its ability to discover some or all of the hidden patterns

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

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

Data Matrix and Dissimilarity Matrix

- Data matrix
  - n data points with p dimensions
  - Two modes

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

- Dissimilarity matrix
  - n data points, but registers only the distance
  - A triangular matrix
  - Single mode

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

Proximity Measure for Nominal Attributes

- Can take 2 or more states
  - Example: Map colour (red, yellow, blue, green (generalization of a binary attribute)
- Let i and j are objects with p variables
- Method 1: Simple matching
  - *m*: # of matches, *p*: total # of variables
  - The dissimilarity

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

- Assign more weight to the attributes having large number of states.
- Method 2: Use a large number of binary attributes
  - creating a new binary attribute for each of the *M* nominal states

Proximity Measure for Binary Attributes

- A contingency table for binary data
- q: # of attributes equal to 1 in i and j.
- Distance measure for symmetric binary variables:
- Distance measure for asymmetric binary variables:
- Jaccard coefficient (*similarity* measure for *asymmetric* binary variables):

Object *i* \ Object *j*	1	0	sum
1	q	r	$q+r$
0	s	t	$s+t$
sum	$q+s$	$r+t$	p

$$d(i,j) = \frac{r+s}{q+r+s+t}$$

$$d(i,j) = \frac{r+s}{q+r+s}$$

$$sim_{Jaccard}(i,j) = \frac{q}{q+r+s}$$

- Note: Jaccard coefficient is the same as "coherence":

$$coherence(i,j) = \frac{sup(i,j)}{sup(i)+sup(j)-sup(i,j)} = \frac{q}{(q+r)+(q+s)-q}$$

Dissimilarity between Binary Variables

- Example

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

- Gender is a symmetric attribute
- The remaining attributes are asymmetric binary
- Let the values Y and P be 1, and the value N 0

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

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

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

Standardizing Numeric Data

- Z-score: $$z = \frac{x - \mu}{\sigma}$$
  - X: raw score to be standardized, μ: mean of the population, σ: standard deviation
  - the distance between the raw score and the population mean in units of the standard deviation
  - negative when the raw score is below the mean, "+" when above
- An alternative way: Calculate the mean absolute deviation

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

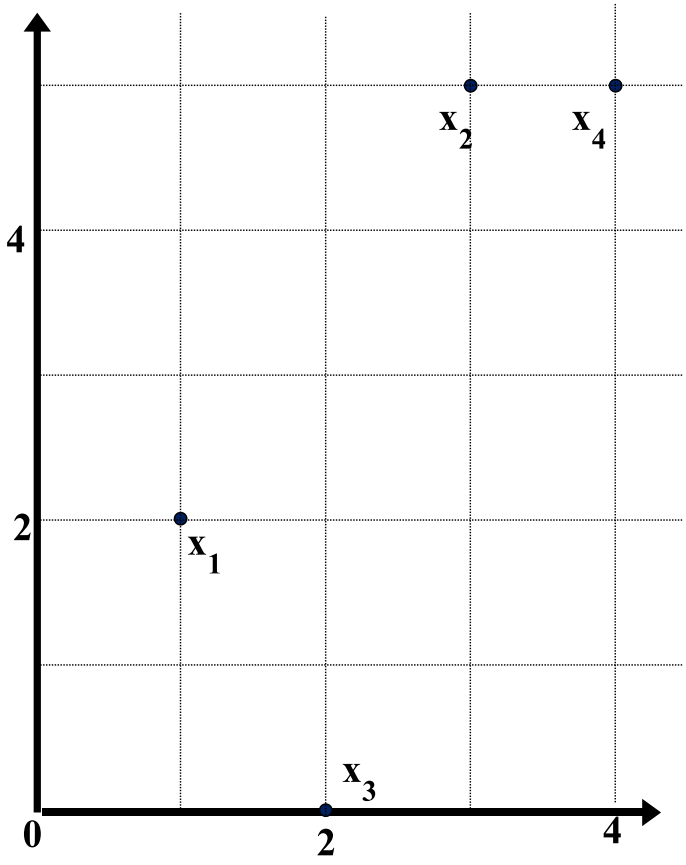
where $$m_f = \frac{1}{n}(x_{1f} + x_{2f} + \ldots + x_{nf})$$

  - standardized measure (*z-score*):

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

- Using mean absolute deviation is more robust than using standard deviation

Example:
Data Matrix and Dissimilarity Matrix



Data Matrix

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

Dissimilarity Matrix

(with Euclidean Distance)

	x1	*x2*	*x3*	*x4*
x1	0			
x2	3.61	0		
x3	5.1	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)

- Properties
  - d(i, j) > 0 if i ≠ j, and d(i, i) = 0 (Positive definiteness)
  - d(i, j) = d(j, i) (Symmetry)
  - d(i, j) ≤ d(i, k) + d(k, j) (Triangle Inequality)
- A distance that satisfies these properties is a metric

Special Cases of Minkowski Distance

- *h* = 1: Manhattan (city block, L_1 norm) distance
  - E.g., the Hamming distance: the number of bits that are different between two binary vectors

$$d(i,j)=|x_{i_1}-x_{j_1}|+|x_{i_2}-x_{j_2}|+...+|x_{i_p}-x_{j_p}|$$

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

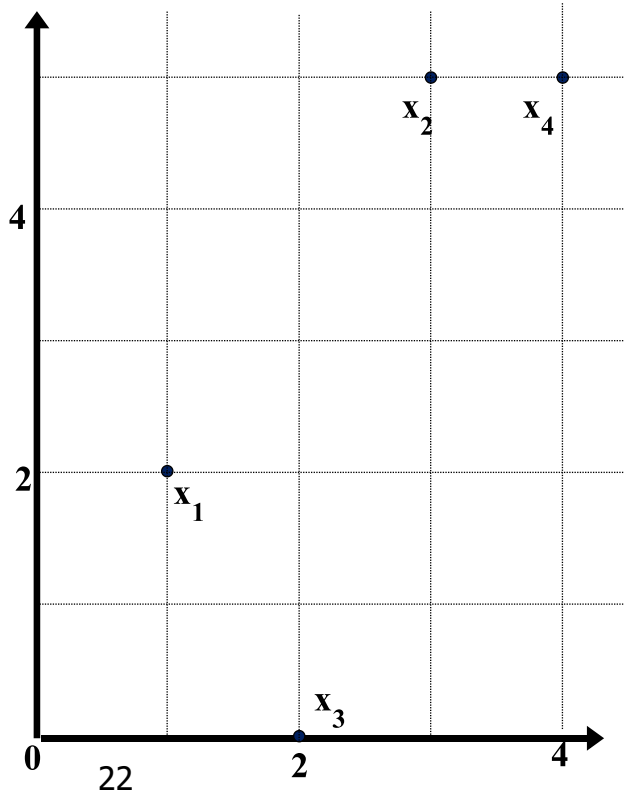
$$d(i,j)=\sqrt{(|x_{i_1}-x_{j_1}|^2+|x_{i_2}-x_{j_2}|^2+...+|x_{i_p}-x_{j_p}|^2)}$$

- $h \to \infty$. "supremum" (L_{max} norm, L_∞ norm) distance.
  - This is the maximum difference between any component (attribute) of the vectors

$$d(i,j)=\lim_{h\to\infty}\left(\sum_{f=1}^{p}|x_{if}-x_{jf}|^h\right)^{\frac{1}{h}}=\max_{f}^{p}|x_{if}-x_{jf}|$$

Example: Minkowski Distance

point	attribute 1	attribute 2
x1	1	2
x2	3	5
x3	2	0
x4	4	5



Dissimilarity Matrices

Manhattan (L_1)

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

Euclidean (L_2)

L2	x1	x2	x3	x4
x1	0			
x2	3.61	0		
x3	2.24	5.1	0	
x4	4.24	1	5.39	0

Supremum

L_∞	x1	x2	x3	x4
x1	0			
x2	3	0		
x3	2	5	0	
x4	3	1	5	0

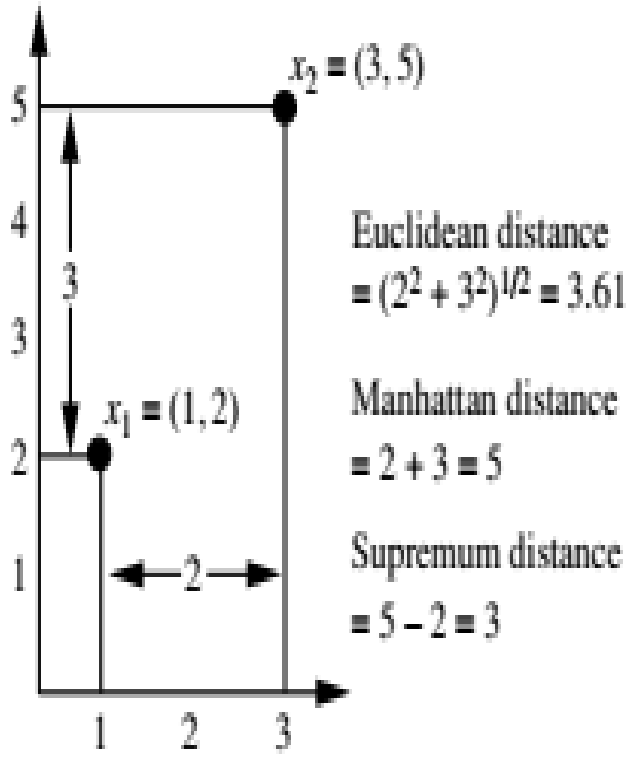


Figure 2.23 Euclidean, Manhattan, and supremum distances between two objects.

Ordinal Variables

- An ordinal variable can be discrete or continuous
- Order is important, e.g., rank
- Can be treated like interval-scaled
  - replace x_{if} by their rank $r_{if} \in \{1, \ldots, M_f\}$
  - map the range of each variable onto [0, 1] by replacing *i*-th object in the *f*-th variable by

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

  - compute the dissimilarity using methods for interval-scaled variables

Attributes of Mixed Type

- A database may contain all attribute types
  - Nominal, symmetric binary, asymmetric binary, numeric, ordinal
- One may use a weighted formula to combine their effects

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

  - f is binary or nominal:

    $d_{ij}^{(f)} = 0$ if $x_{if} = x_{jf}$, or $d_{ij}^{(f)} = 1$ otherwise
  - f is numeric: use the normalized distance
  - f is ordinal
    - Compute ranks r_{if} and
    - Treat z_{if} as interval-scaled

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

Cosine Similarity

- A **document** can be represented by thousands of attributes, each recording the *frequency* of a particular word (such as keywords) or phrase in the document.

Document	*team*	*coach*	*hockey*	*baseball*	*soccer*	*penalty*	*score*	*win*	*loss*	*season*
Document1	5	0	3	0	2	0	0	2	0	0
Document2	3	0	2	0	1	1	0	1	0	1
Document3	0	7	0	2	1	0	0	3	0	0
Document4	0	1	0	0	1	2	2	0	3	0

- Other vector objects: gene features in micro-arrays, …
- Applications: information retrieval, biologic taxonomy, gene feature mapping, ...
- Cosine measure: If d_1 and d_2 are two vectors (e.g., term-frequency vectors), then

$$\cos(d_1, d_2) = (d_1 \bullet d_2) / ||d_1|| \, ||d_2|| ,$$

where $\bullet$ indicates vector dot product, $||d||$: the length of vector d

Example: Cosine Similarity

- $\cos(d_1, d_2) = (d_1 \bullet d_2) / ||d_1|| \, ||d_2||$,
  where $\bullet$ indicates vector dot product, $||d|$: the length of vector d

- Ex: Find the **similarity** between documents 1 and 2.

d_1 = (5, 0, 3, 0, 2, 0, 0, 2, 0, 0)
d_2 = (3, 0, 2, 0, 1, 1, 0, 1, 0, 1)

$d_1 \bullet d_2$ = 5*3+0*0+3*2+0*0+2*1+0*1+0*1+2*1+0*0+0*1 = 25
$||d_1|| = (5*5+0*0+3*3+0*0+2*2+0*0+0*0+2*2+0*0+0*0)^{0.5} = (42)^{0.5} = 6.481$
$||d_2|| = (3*3+0*0+2*2+0*0+1*1+1*1+0*0+1*1+0*0+1*1)^{0.5} = (17)^{0.5} = 4.12$
$\cos(d_1, d_2) = 0.94$

Requirements of Clustering in Data Mining

- Scalability
  - Works well on small data sets containing fewer than 200 objects.
  - Highly scalable clustering algorithms are needed (for millions of data objects).
- Ability to deal with different types of attributes
  - Many algorithms are designed to cluster interval-based (numerical) data.
  - Applications may require clustering other types of data, such as binary, categorical (nominal) and ordinal data, and mixtures of these data types.
- Discovery of clusters with arbitrary shape
  - Many clustering algorithm mine clusters based on Euclidean or Manhattan distance measures.
  - Find spherical clusters with small size and density.
  - Development of algorithms to develop arbitrary shapes are needed.

Requirements of Clustering in Data Mining

- Minimal requirements for domain knowledge to determine input parameters.
  - # of clusters
- Able to deal with noise and outliers
  - Some clustering algorithms are sensitive to outliers, missing, unknown, or erroneous data.
- Incremental clustering and Insensitive to order of input records
- High dimensionality
  - Many clustering algorithms are good at handling low-dimensional data.
- Constraint-based clustering
  - Clustering customers based on city rivers and highway networks.
- Interpretability and usability
  - The results should be interpretable, comprehensive and usable.

Comparison of clustering methods

- 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)

Overview of basic Clustering Methods

- 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
- Hierarchical approach:
  - Create a hierarchical decomposition of the set of data (or objects) using some criterion
  - Typical methods: Diana, Agnes, BIRCH, CAMELEON
- Density-based approach:
  - Based on connectivity and density functions
  - Typical methods: DBSACN, OPTICS, DenClue
- Grid-based approach:
  - based on a multiple-level granularity structure
  - Typical methods: STING, WaveCluster, CLIQUE

Outline

- Cluster Analysis: Basic Concepts
- **Partitioning Methods**
- Hierarchical Methods
- Density-Based Methods
- Grid-Based Methods
- Evaluation of Clustering
- Summary

Algorithms

- K-means: A centroid based technique
- K-mediods: A representative Object-based technique
- CLARA: Clustering LARge Applications
- CLARANS: Clustering Large Applications based upon RANdomized search

Partitioning Algorithms: Basic Concept

- Partitioning method: Discovering the groupings in the data by optimizing a specific objective function and iteratively improving the quality of partitions
- K-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)

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

- Problem: Given *k*, find a partition of *k clusters* that optimizes the chosen partitioning criterion
  - Global optimal: exhaustively enumerate all partitions
    - NP-hard
- Alternative
  - 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

K-means Clustering

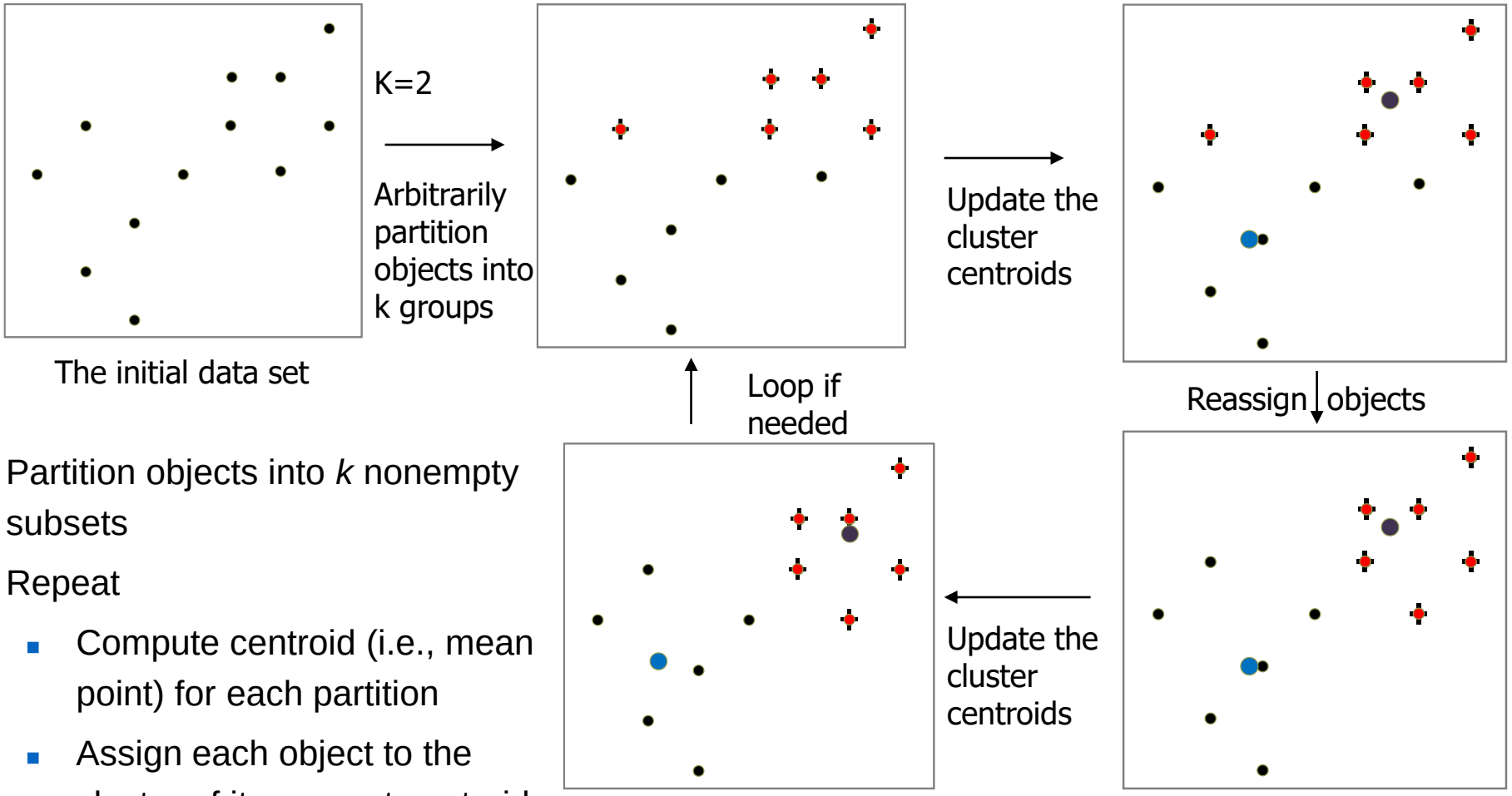
- Partitional clustering approach
- Each cluster is associated with a centroid (center point)
- Each point is assigned to the cluster with the closest centroid
- Number of clusters, K, must be specified
- The basic algorithm is very simple

1: Select K points as the initial centroids.
2: **repeat**
3: Form K clusters by assigning all points to the closest centroid.
4: Recompute the centroid of each cluster.
5: **until** The centroids don't change

K-means Clustering – Details

- Initial centroids are often chosen randomly.
  - Clusters produced vary from one run to another.
- The centroid is (typically) the mean of the points in the cluster.
- ‘Closeness’ is measured by Euclidean distance, Manhattan distance, cosine similarity, correlation, etc.
- K-means will converge for common similarity measures mentioned above.
- Most of the convergence happens in the first few iterations.
  - Often the stopping condition is changed to ‘Until relatively few points change clusters’
- Complexity is O(n * K * I * d)
  - n = number of points, K = number of clusters,
    I = number of iterations, d = number of attributes
- Space complexity O((n+K)d)

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

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*

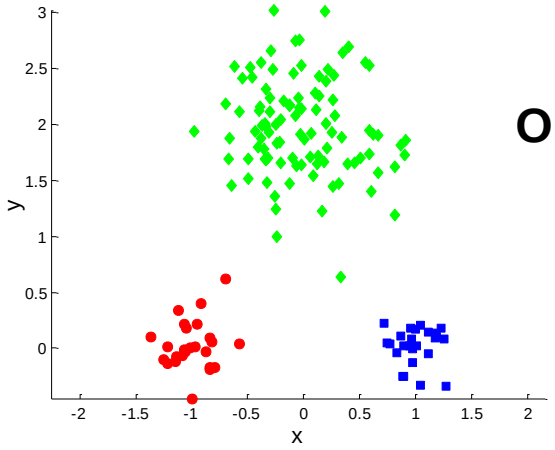
Evaluating K-means Clusters

- Most common measure is Sum of Squared Error (SSE)
  - For each point, the error is the distance to the nearest cluster
  - To get SSE, we square these errors and sum them.

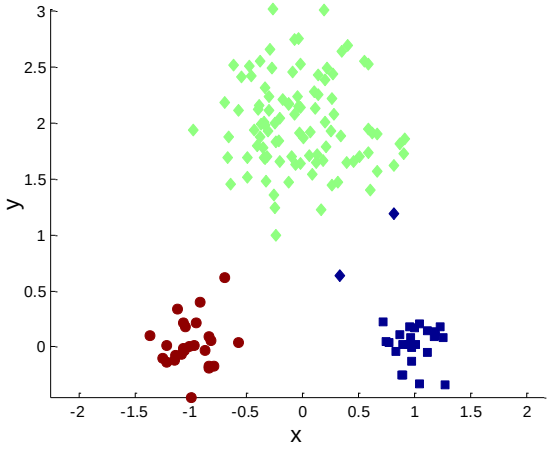
$$SSE = \sum_{i=1}^{K} \sum_{x \in C_i} dist^2(m_i, x)$$

  - x is a data point in cluster C_i and m_i is the representative point for cluster C_i
    - can show that m_i corresponds to the center (mean) of the cluster
  - Given two clusters, we can choose the one with the smallest error
  - One easy way to reduce SSE is to increase K, the number of clusters
    - A good clustering with smaller K can have a lower SSE than a poor clustering with higher K

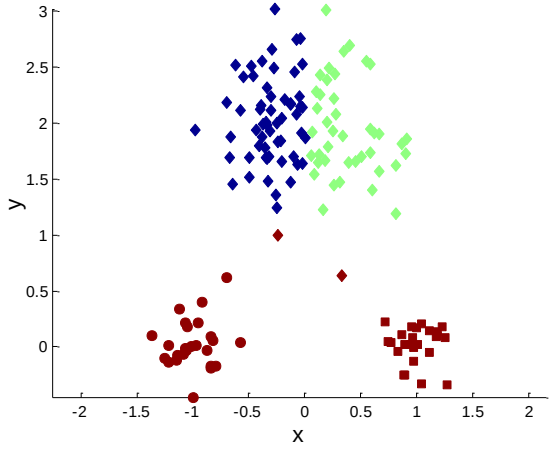
Two different K-means Clusterings



Original Points

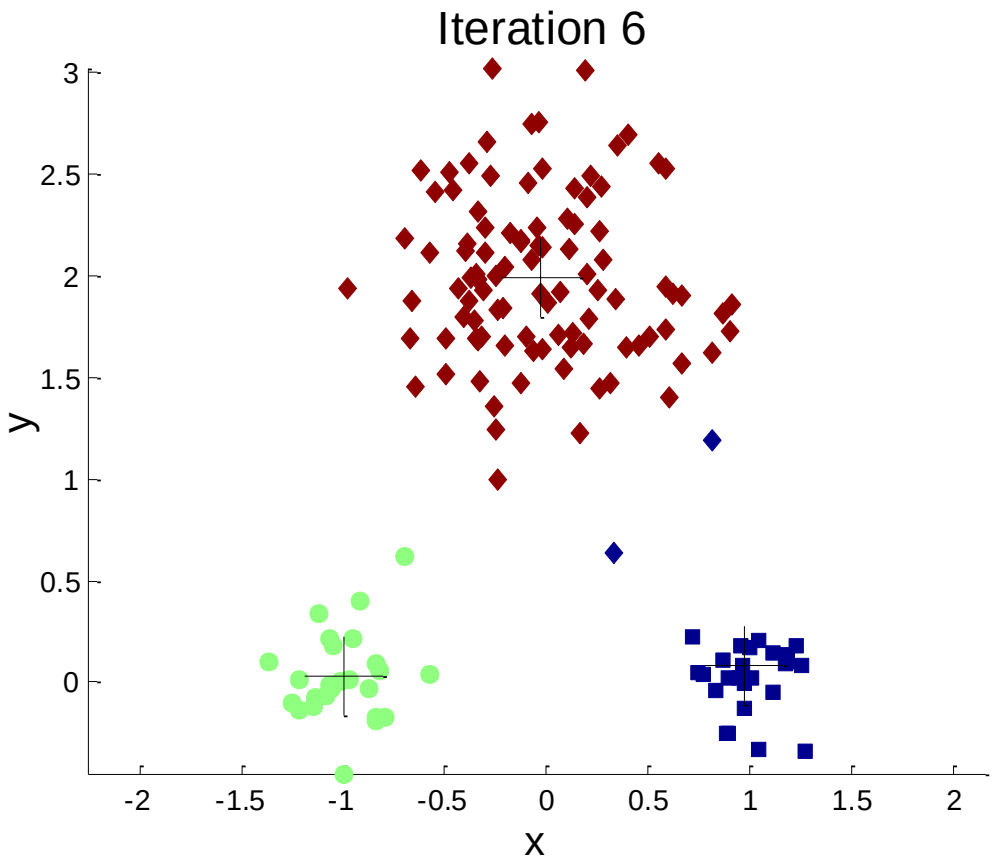


Optimal Clustering

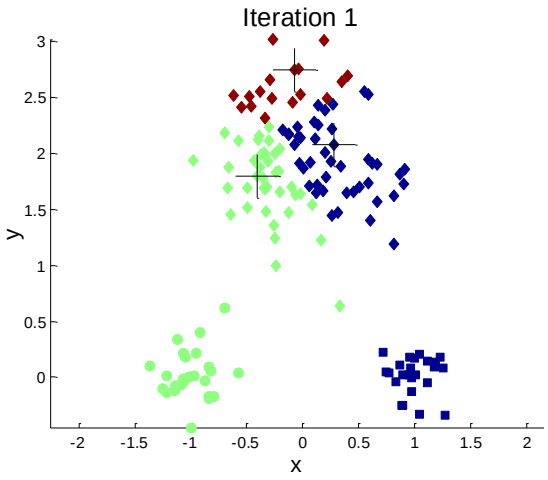


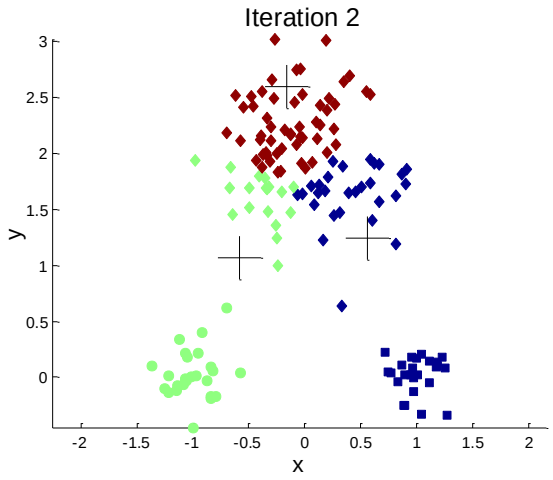
Sub-optimal Clustering

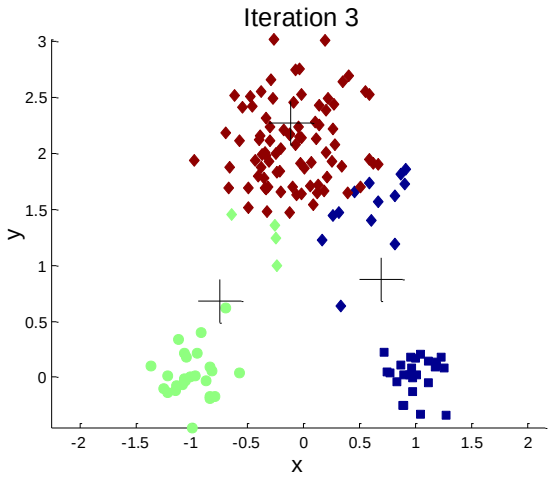
Importance of Choosing Initial Centroids

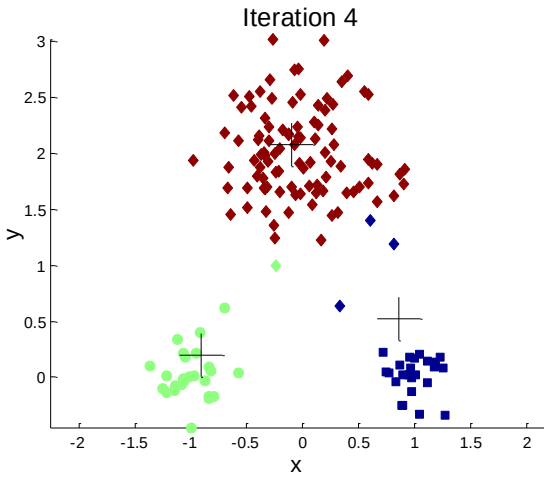


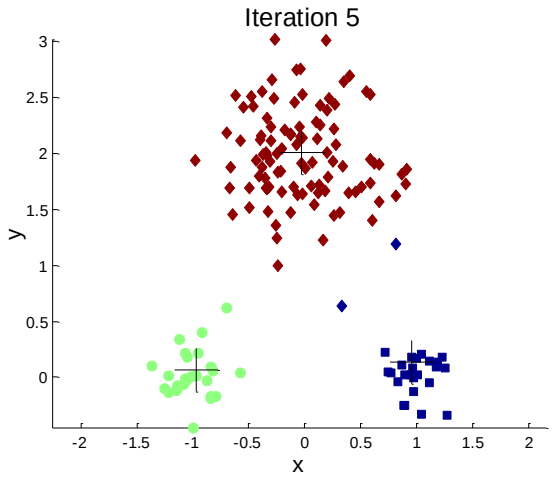
Importance of Choosing Initial Centroids

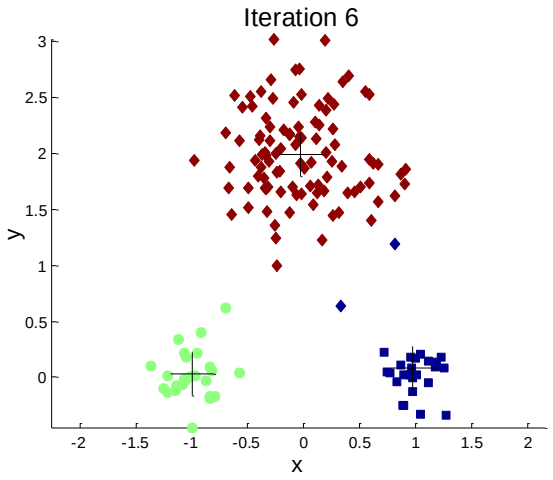












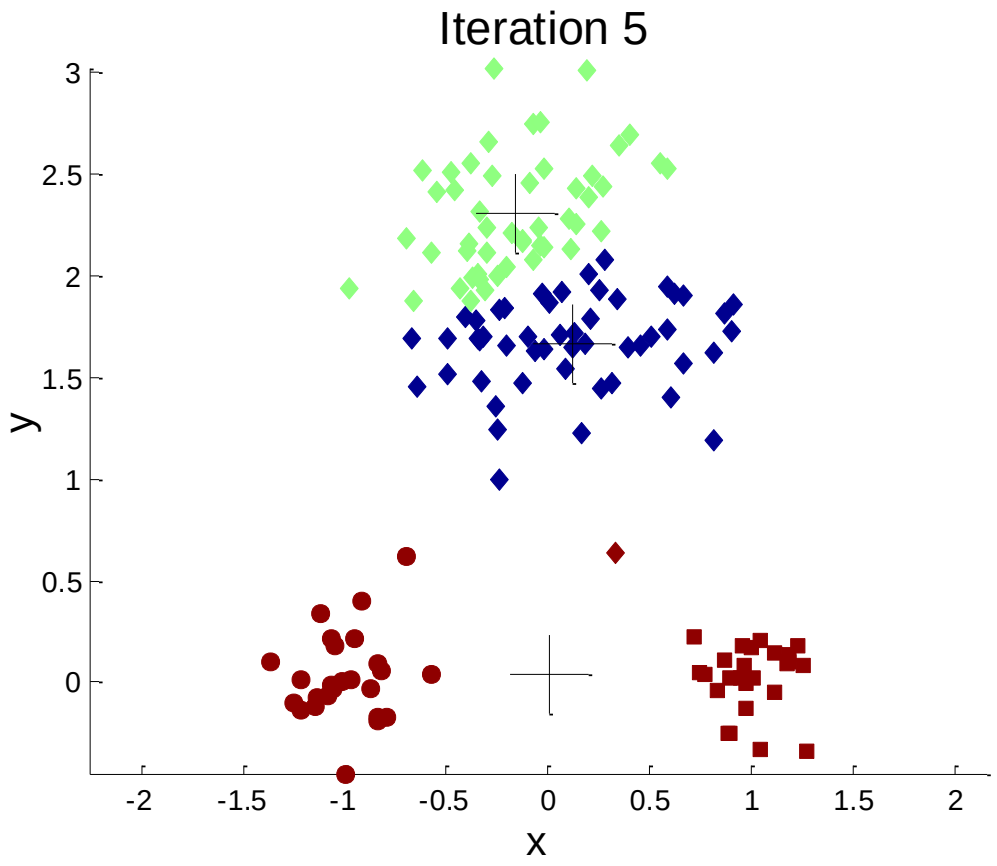
Evaluating K-means Clusters

- Most common measure is Sum of Squared Error (SSE)
  - For each point, the error is the distance to the nearest cluster
  - To get SSE, we square these errors and sum them.

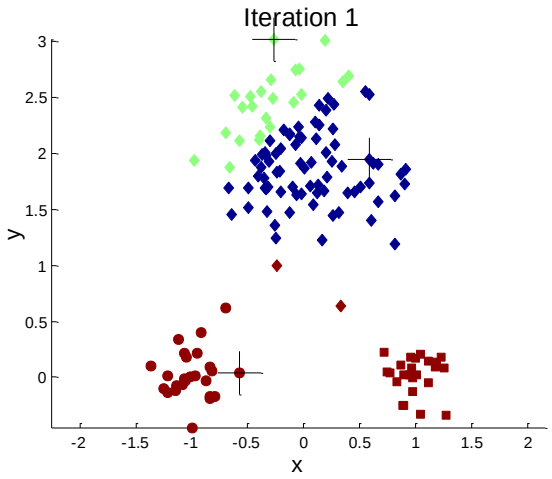
$$SSE = \sum_{i=1}^{K} \sum_{x \in C_i} dist^2(m_i, x)$$

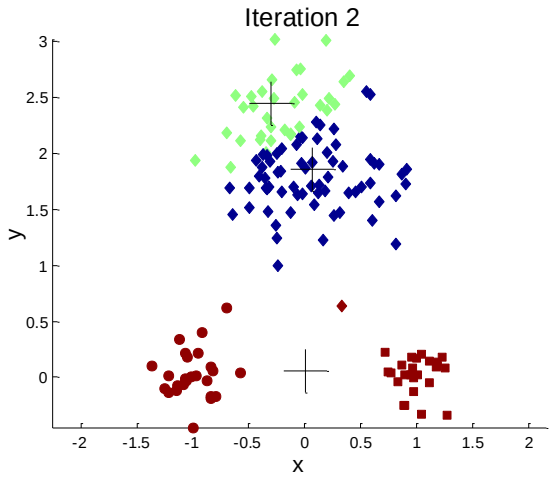
  - *x* is a data point in cluster C_i and m_i is the representative point for cluster C_i
    - can show that m_i corresponds to the center (mean) of the cluster
  - Given two clusters, we can choose the one with the smallest error
  - One easy way to reduce SSE is to increase K, the number of clusters
    - A good clustering with smaller K can have a lower SSE than a poor clustering with higher K

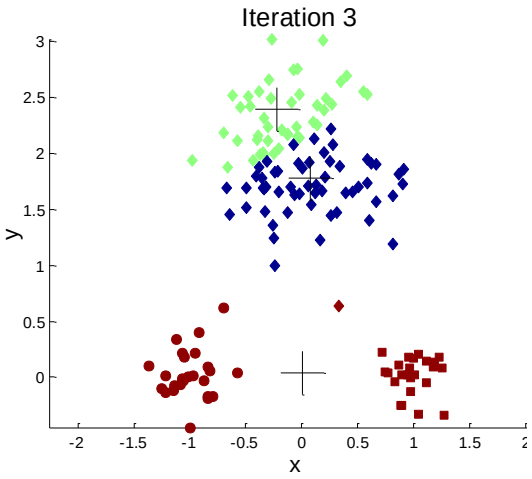
Importance of Choosing Initial Centroids …

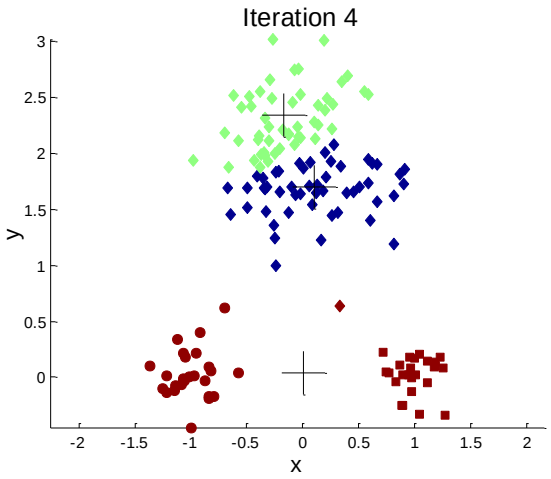


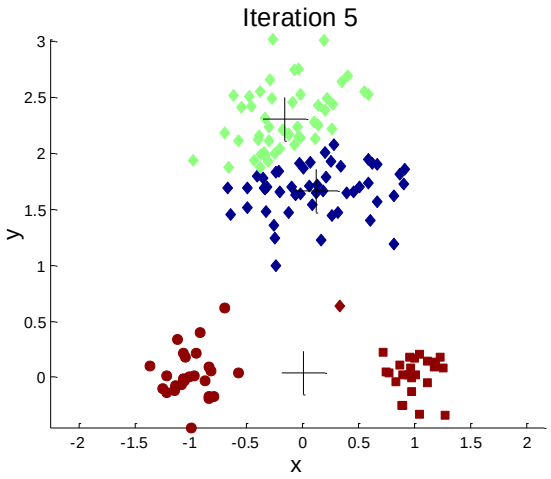
Importance of Choosing Initial Centroids …











Problems with Selecting Initial Points

- If there are K ‘real’ clusters then the chance of selecting one centroid from each cluster is small.
  - Chance is relatively small when K is large
  - Sometimes the initial centroids will readjust themselves in ‘right’ way, and sometimes they don’t

Bisecting K-means

- Bisecting K-means algorithm
  - Variant of K-means that can produce a partitional or a hierarchical clustering
  - Less susceptible to initialization problems

1: Initialize the list of clusters to contain the cluster containing all points.
2: **repeat**
3: Select a cluster from the list of clusters
4: **for** $i = 1$ to *number_of_iterations* **do**
5: Bisect the selected cluster using basic K-means
6: **end for**
7: Add the two clusters from the bisection with the lowest SSE to the list of clusters.
8: **until** Until the list of clusters contains K clusters

Strengths of K-means

- Simple
- Can be used for a wide variety of data types
- Efficient even though multiple runs are performed.
- Some variants including bisecting K-means are even more efficient and are less susceptible to initialization problems.

Limitations of K-means

- K should be known
- K-means has problems when clusters are of differing
  - Sizes
  - Densities
  - Non-globular shapes
- K-means has problems when the data contains outliers.

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

K-modes: Handling categorical data

- Replacing means of clusters with modes
- Using new dissimilarity measures to deal with categorical objects
- The number of mismatches of the corresponding attribute categories of the two objects
- Using a frequency-based method to update modes of clusters
- A mixture of categorical and numerical data: *k-prototype* method

Issue with k-means

- The k-means algorithm is sensitive to outliers!
  - Since an object with an extremely large value may substantially distort the distribution of the data
- Example
  - Clustering of 1,2,3,8,9,10 and 25
  - Visual method: (1,2,3) and (8,9,10)
  - However, k means with k=3
    - First iteration: (1,2,3) and (8,9,10,25)
    - Second iteration (1,2,3,8) and (9,10,25)
    - Here, 25 is outlier

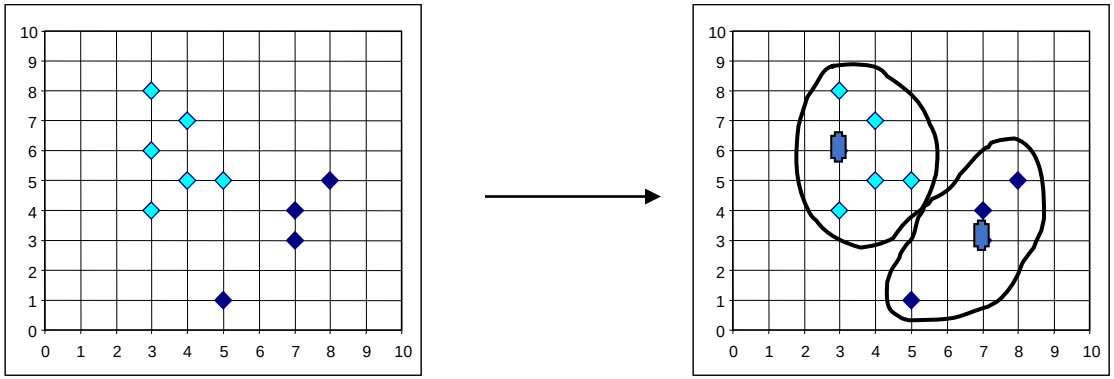
K-medoids

- Instead of taking the mean value of the objects in a cluster as a reference point, we can pick actual objects to represent the clusters.
- Objective is to minimize the sum of dissimilarities between each object p and its corresponding representative object.
- The absolute error condition is

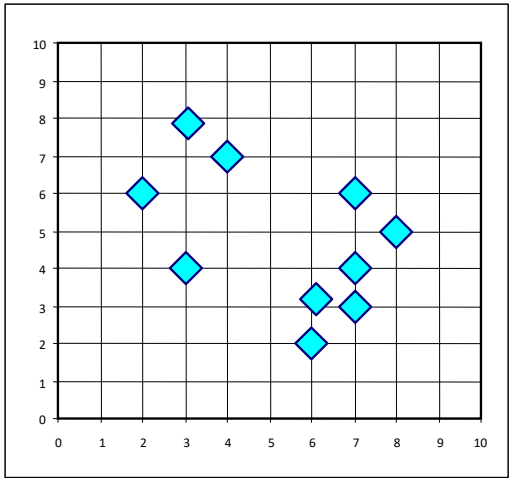
$$E = \sum_{i=1}^{k} \sum_{p \in C_i} dist(p, o_i),$$

K-medoids

- 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

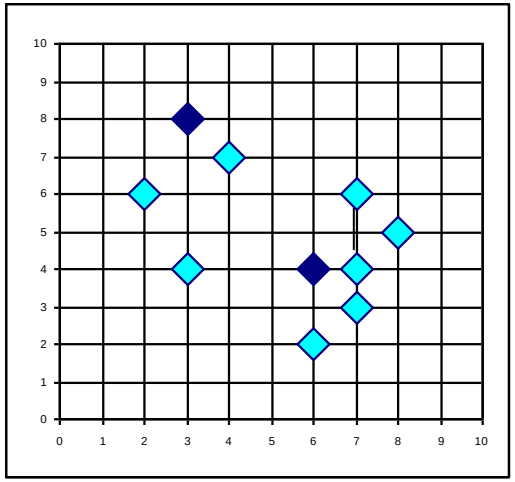


PAM: A Typical K-Medoids Algorithm



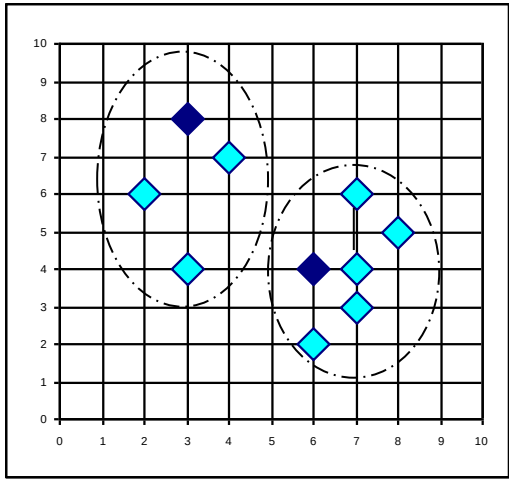
K=2

Arbitrary choose k objects as initial medoids

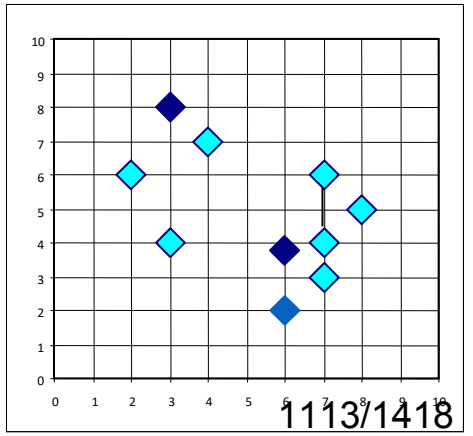


Assign each remaining object to nearest medoids

Total Cost = 20

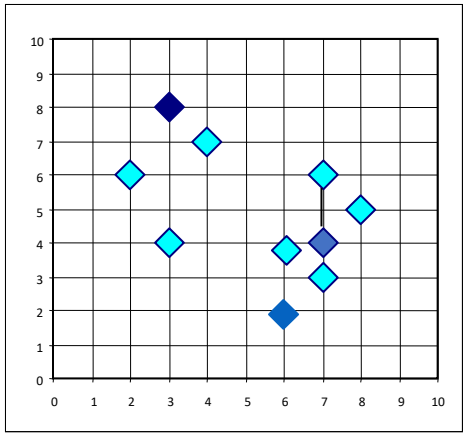


Randomly select a nonmedoid object, O_{ramdom}



Compute total cost of swapping

Total Cost = 26



Swapping O and O_{ramdom}

If quality is improved.

Do loop

Until no change

The K-Medoid Clustering Method

- *Basic idea:* Find *representative* objects (medoids) in clusters
- *PAM* (Partitioning Around Medoids, Kaufmann & Rousseeuw 1987) built in Splus
  - 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

Algorithm: *k*-medoids. PAM, a *k*-medoids algorithm for partitioning based on medoid or central objects.

Input:

- *k*: the number of clusters,
- *D*: a data set containing *n* objects.

Output: A set of *k* clusters.

Method:

(1) arbitrarily choose *k* objects in *D* as the initial representative objects or seeds;
(2) **repeat**
(3) assign each remaining object to the cluster with the nearest representative object;
(4) randomly select a nonrepresentative object, o_{random};
(5) compute the total cost, *S*, of swapping representative object, o_j, with o_{random};
(6) **if** $S < 0$ **then** swap o_j with o_{random} to form the new set of *k* representative objects;
(7) **until** no change;

Figure 10.5 PAM, a *k*-medoids partitioning algorithm.

What Is the Problem with PAM?

- PAM is more robust than k-means in the presence of noise and outliers because a medoid is less influenced by outliers or other extreme values than a mean
- PAM works efficiently for small data sets but does not **scale well** for large data sets.
  - Complexity: $O(k(n-k)^2)$ for each iteration

    where n is # of data,k is # of clusters
- Efficiency improvement on PAM
  - *CLARA* (Kaufmann & Rousseeuw, 1990): PAM on samples
  - *CLARANS* (Ng & Han, 1994): Randomized re-sampling

CLARA (Clustering Large Applications) (1990)

- PAM has a scaling problems
- *CLARA* (Kaufmann and Rousseeuw in 1990)
  - Built in statistical analysis packages, such as Splus
- It draws *multiple samples* of the data set, applies *PAM* on each sample, and gives the best clustering as the output
- Complexity of CLARA for each iteration $O(ks^2+k(n-k))$
  - "s" is a size of sample, k is number of clusters, and n is number of objects.
- Strength: deals with larger data sets than *PAM*
- Weakness:
  - Efficiency depends on the sample size
  - A good clustering based on samples will not necessarily represent a good clustering of the whole data set if the sample is biased

CLARANS (Clustering Large Applications based on RANdomized Search)

- CLARA confines the candidate medoids to only a random sample of the data set.

Algorithm

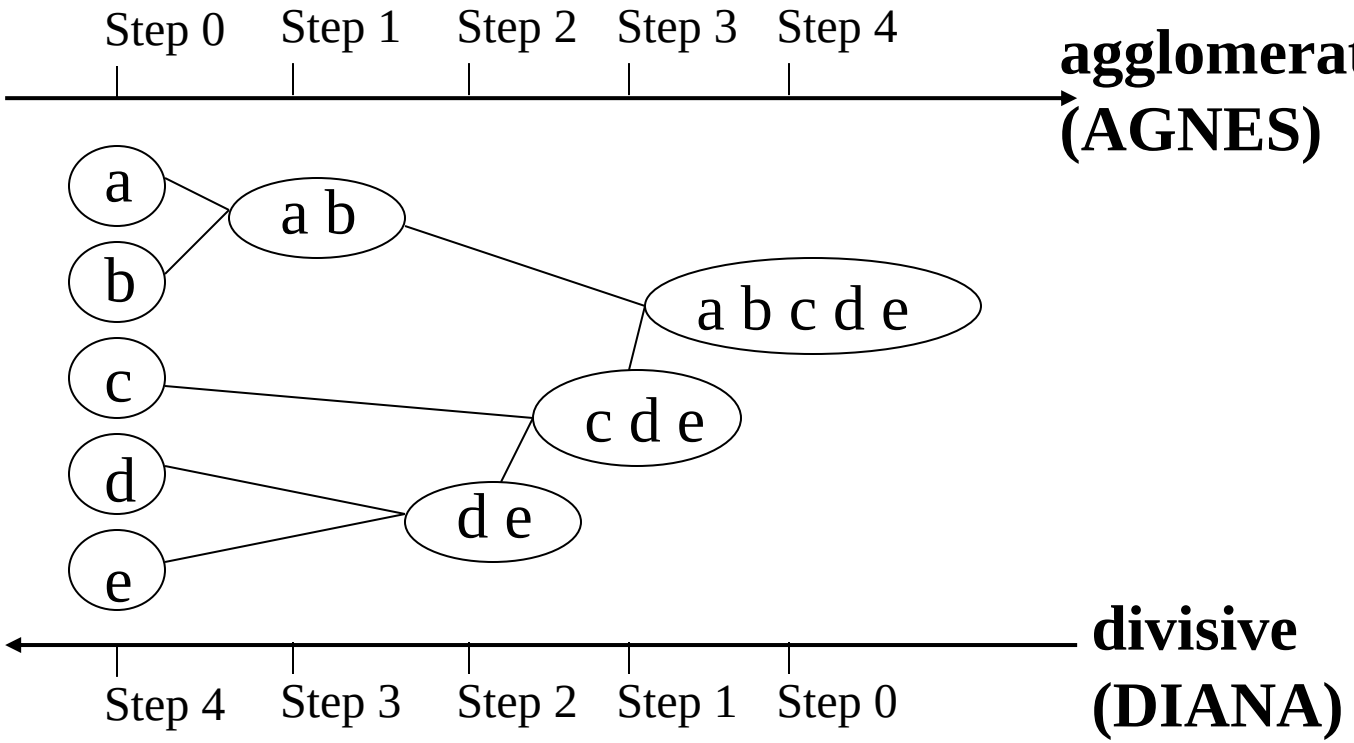
- It randomly selects *k* objects in the data set as the current medoids.
- It then randomly selects a current medoid ***x*** and an object ***y*** that is not one of the current medoids.
- Can replacing ***x*** by ***y*** improve the absolute-error criterion? If yes, the replacement is made.
- CLARANS conducts such a randomized search *l* times.
- The set of the current medoids after the *l* steps is considered a local optimum.
- CLARANS repeats this randomized process *m* times and returns the best local optimal as the final result.

Cluster Analysis: Basic Concepts and Methods

- Cluster Analysis: Basic Concepts
- Partitioning Methods
- **Hierarchical Methods**
- Density-Based Methods
- Grid-Based Methods
- Evaluation of Clustering
- Summary

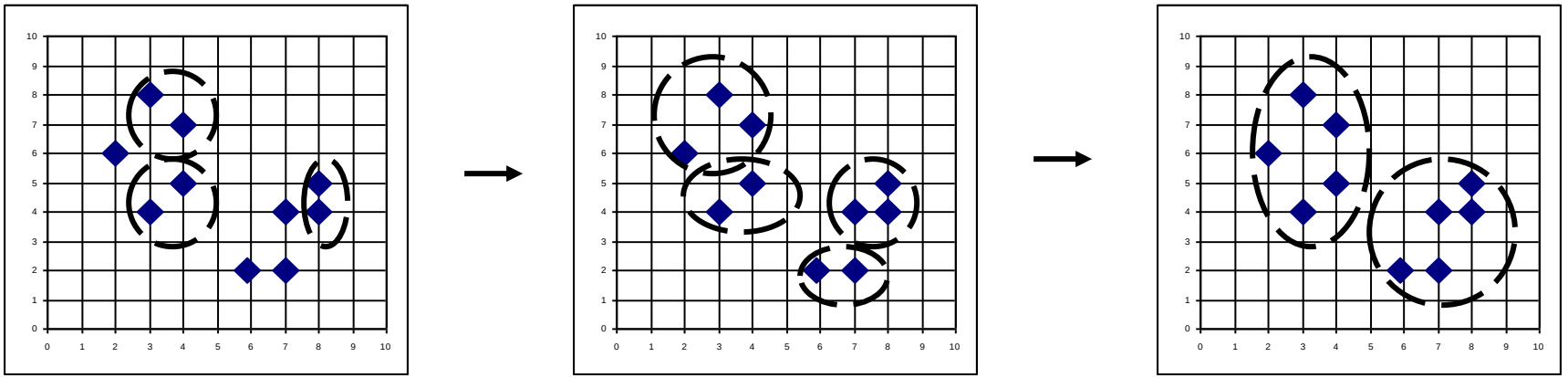
Hierarchical Clustering

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



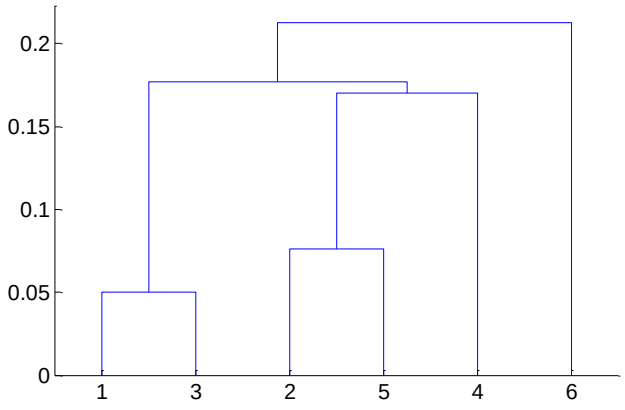
AGNES (Agglomerative Nesting)

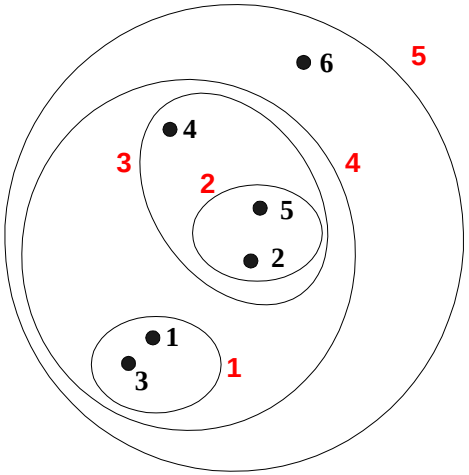
- Introduced in Kaufmann and Rousseeuw (1990)
- Implemented in statistical packages, e.g., Splus
- Use the **single-link** method and the dissimilarity matrix
- Merge nodes that have the least dissimilarity
- Go on in a non-descending fashion
- Eventually all nodes belong to the same cluster



Dendrogram

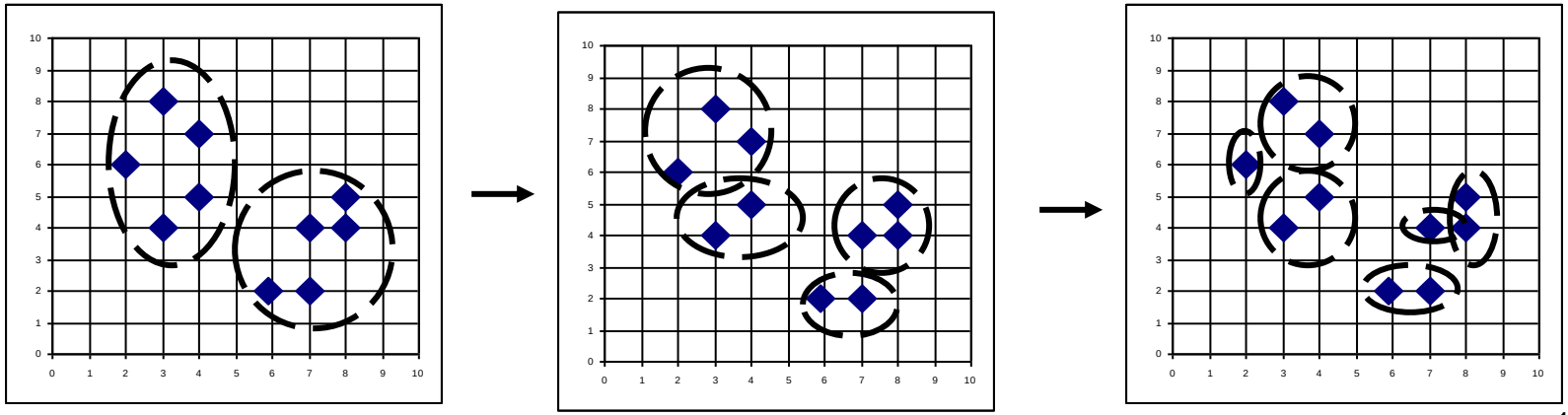
- Produces a set of nested clusters organized as a hierarchical tree
- Can be visualized as a dendrogram
  - A tree like diagram that records the sequences of merges or splits
- A clustering of the data objects is obtained by cutting the dendrogram at the desired level, then each connected component forms a cluster





DIANA (Divisive Analysis)

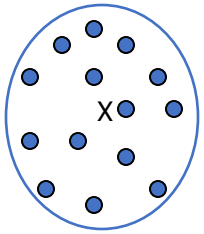
- Introduced in Kaufmann and Rousseeuw (1990)
- Implemented in statistical analysis packages, e.g., Splus
- Inverse order of AGNES
  - Start with one, all-inclusive cluster
  - At each step, split a cluster until each cluster contains a point (or there are k clusters)

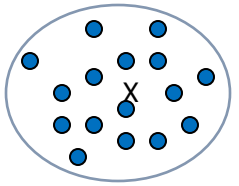


About divisive methods

- Many possibilities and complex
- Employs heuristics
- Not widely used

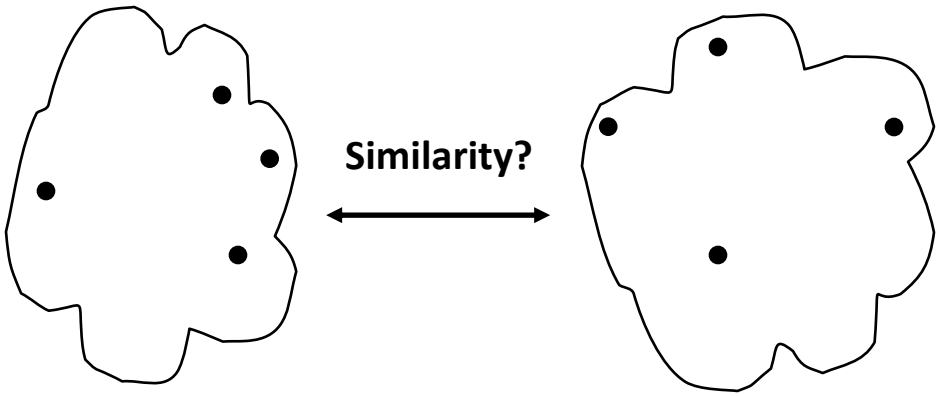
Distance between Clusters





- 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

How to Define Inter-Cluster Similarity

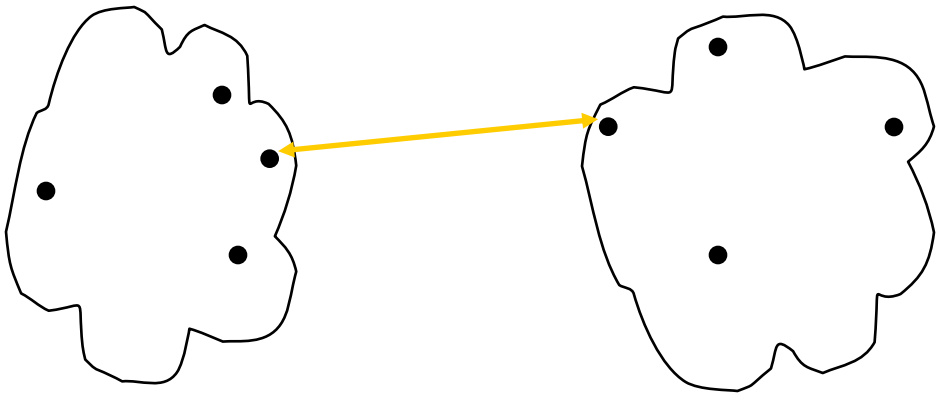


	p1	p2	p3	p4	p5	. . .
p1						
p2						
p3						
p4						
p5						
.						

Proximity Matrix

- MIN
- MAX
- Group Average
- Distance Between Centroids
- Other methods driven by an objective function
  - Ward's Method uses squared error

How to Define Inter-Cluster Similarity

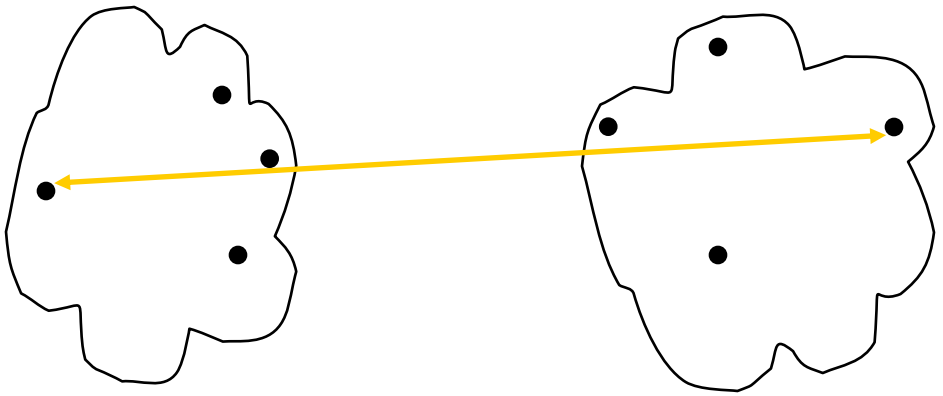


	p1	p2	p3	p4	p5	. . .
p1						
p2						
p3						
p4						
p5						
.						
.						
.						

Proximity Matrix

- MIN
- MAX
- Group Average
- Distance Between Centroids
- Other methods driven by an objective function
  - Ward's Method uses squared error

How to Define Inter-Cluster Similarity



	p1	p2	p3	p4	p5	. . .
p1						
p2						
p3						
p4						
p5						
.						

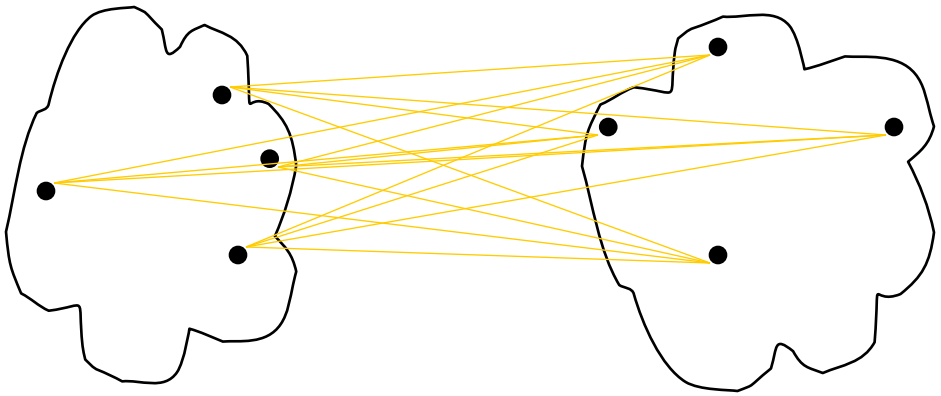
.

.

Proximity Matrix

- MIN
- MAX
- Group Average
- Distance Between Centroids
- Other methods driven by an objective function
  - Ward's Method uses squared error

How to Define Inter-Cluster Similarity



	p1	p2	p3	p4	p5	. . .
p1						
p2						
p3						
p4						
p5						
.						

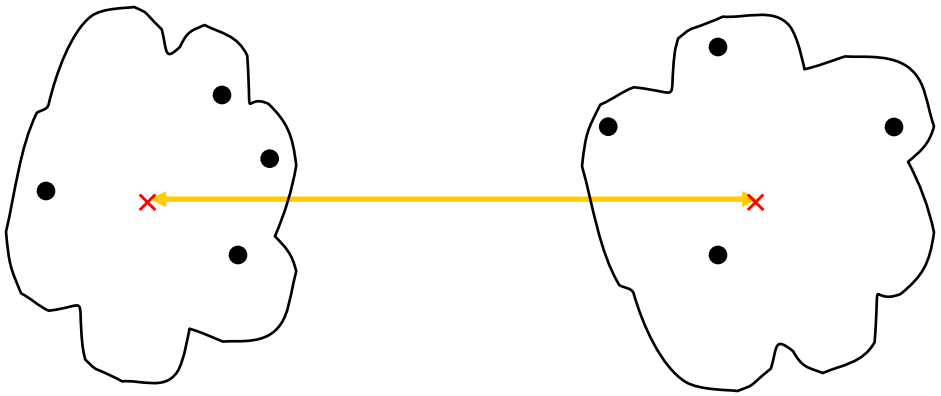
.

.

Proximity Matrix

- MIN
- MAX
- Group Average
- Distance Between Centroids
- Other methods driven by an objective function
  - squared error

How to Define Inter-Cluster Similarity



- MIN
- MAX
- Group Average
- Distance Between Centroids

	p1	p2	p3	p4	p5	. . .
p1						
p2						
p3						
p4						
p5						
.						

.

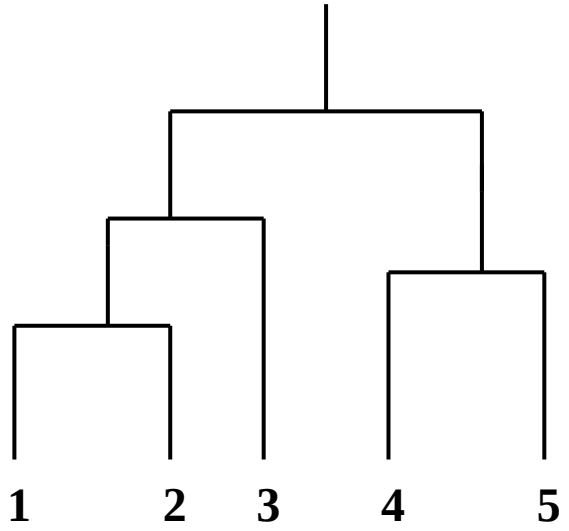
.

Proximity Matrix

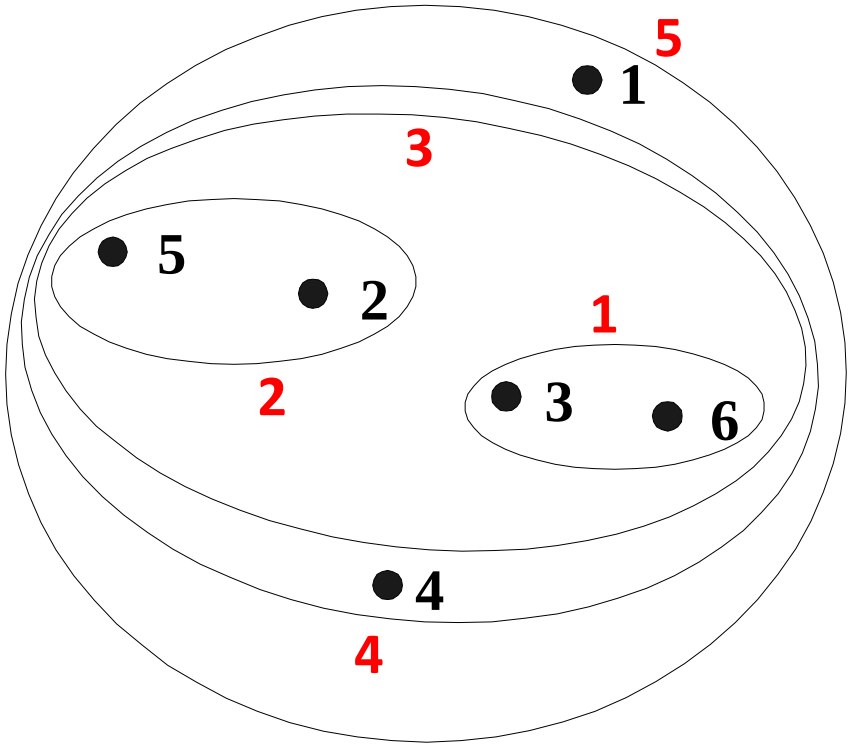
Cluster Similarity: MIN or Single Link

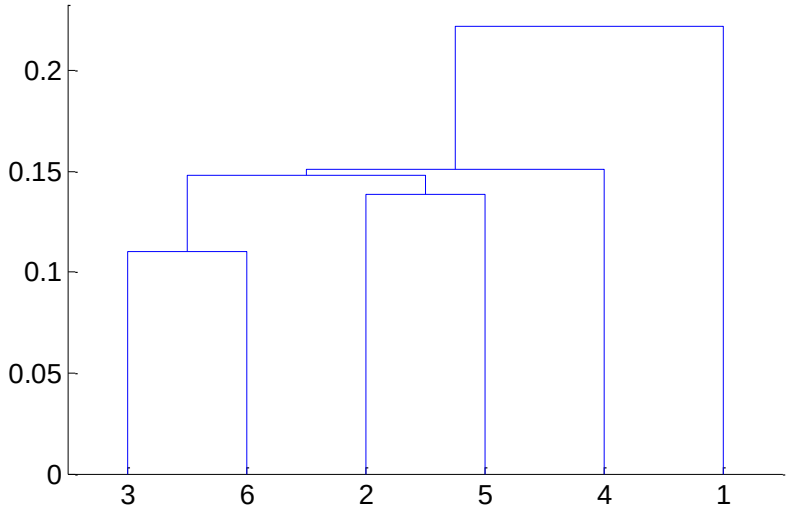
- Similarity of two clusters is based on the two most similar (closest) points in the different clusters
  - Determined by one pair of points, i.e., by one link in the proximity graph.

	I1	I2	I3	I4	I5
I1	1.00	0.90	0.10	0.65	0.20
I2	0.90	1.00	0.70	0.60	0.50
I3	0.10	0.70	1.00	0.40	0.30
I4	0.65	0.60	0.40	1.00	0.80
I5	0.20	0.50	0.30	0.80	1.00



Hierarchical Clustering: MIN

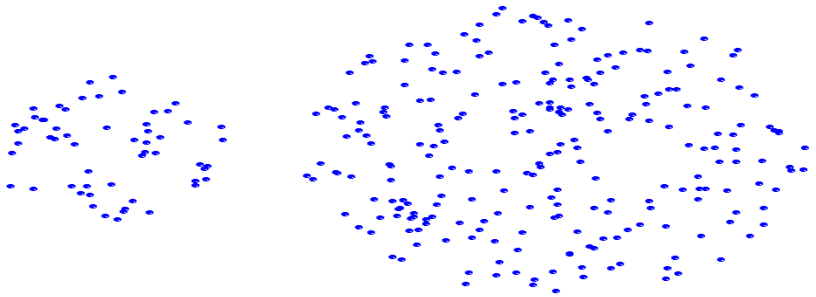




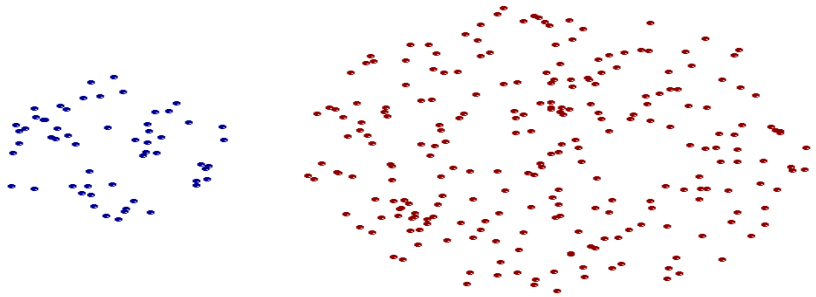
Nested Clusters

Dendrogram

Strength of MIN



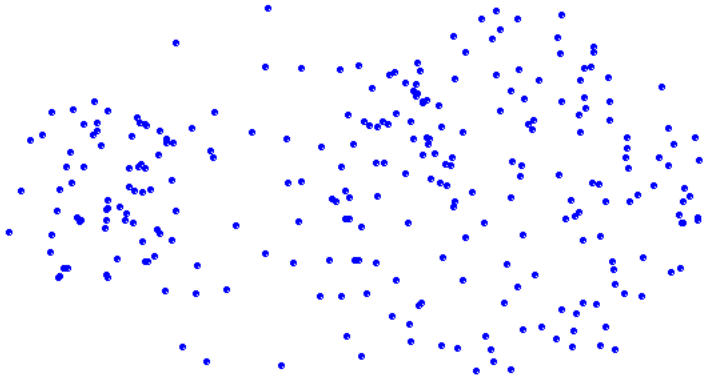
Original Points



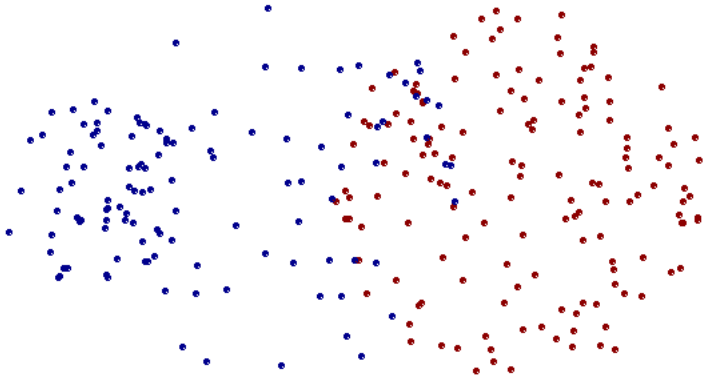
Two Clusters

- **Can handle non-elliptical shapes**

Limitations of MIN



Original Points



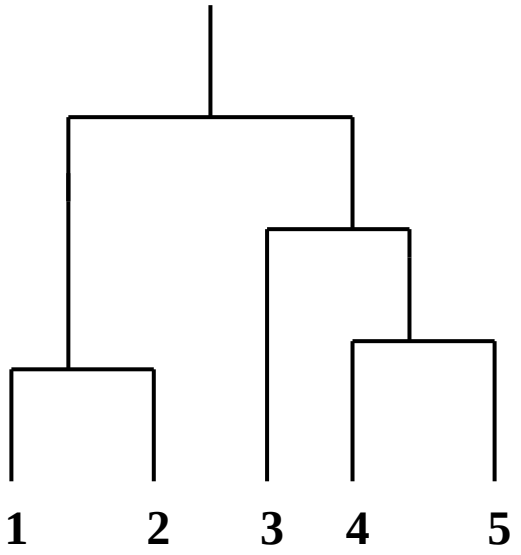
Two Clusters

- **Sensitive to noise and outliers**

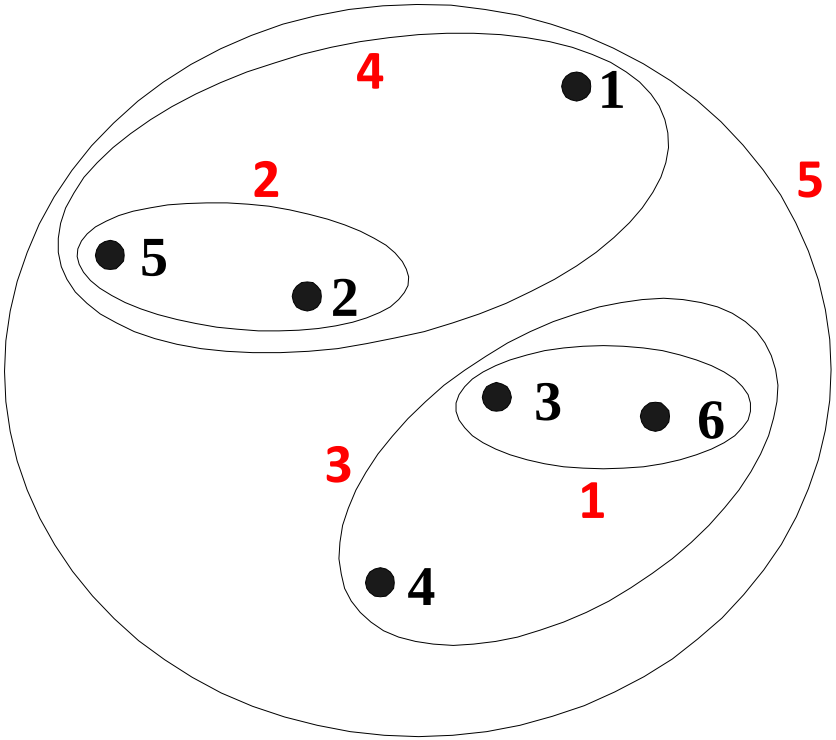
Cluster Similarity: MAX or Complete Linkage

- Similarity of two clusters is based on the two least similar (most distant) points in the different clusters
  - Determined by all pairs of points in the two clusters

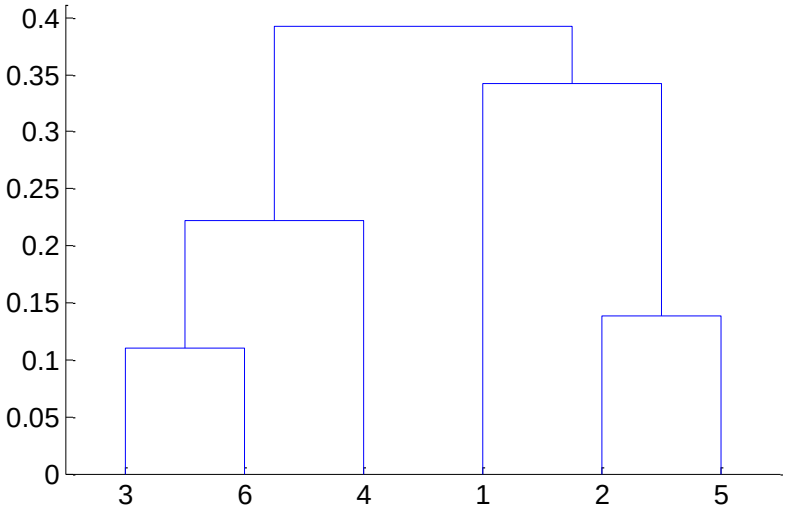
	I1	I2	I3	I4	I5
I1	1.00	0.90	0.10	0.65	0.20
I2	0.90	1.00	0.70	0.60	0.50
I3	0.10	0.70	1.00	0.40	0.30
I4	0.65	0.60	0.40	1.00	0.80
I5	0.20	0.50	0.30	0.80	1.00



Hierarchical Clustering: MAX

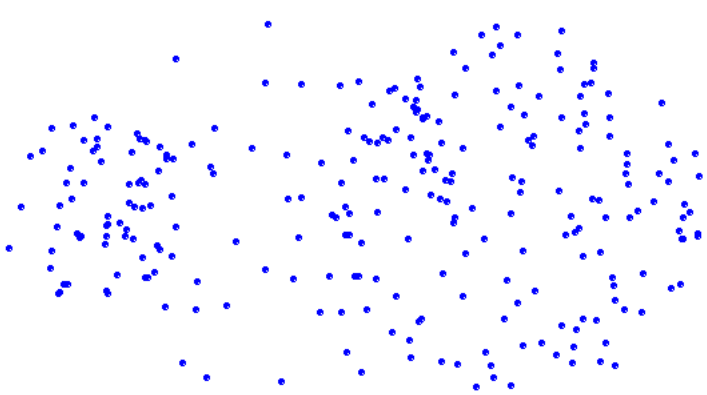


Nested Clusters

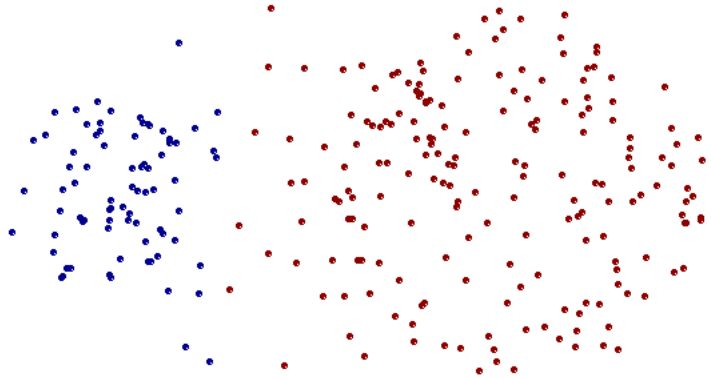


Dendrogram

Strength of MAX



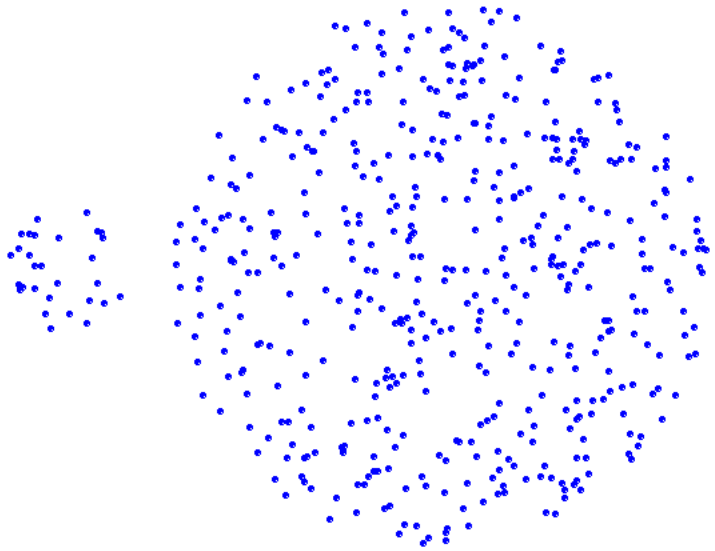
Original Points



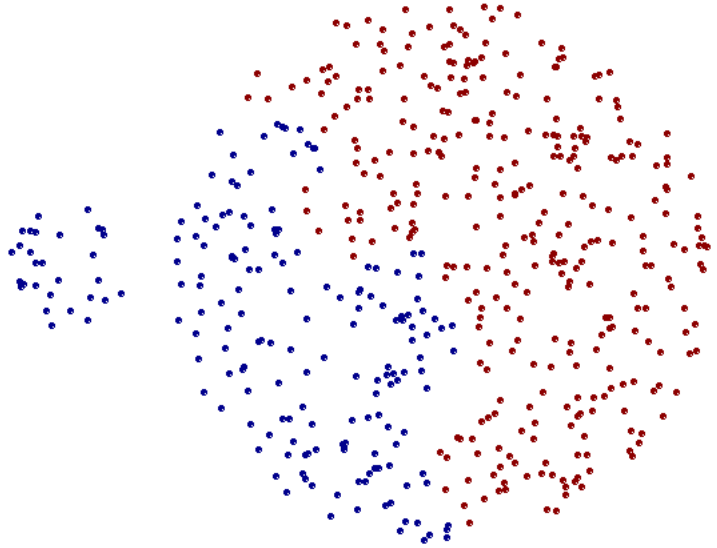
Two Clusters

- **Less susceptible to noise and outliers**

Limitations of MAX



Original Points

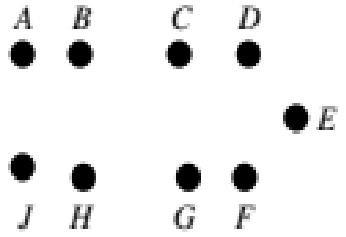


Two Clusters

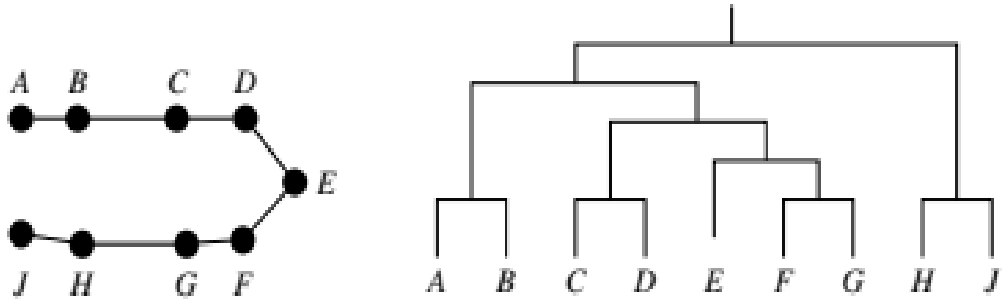
- **Tends to break large clusters**
- **Biased towards globular clusters**

Single linkage and complete-linkage algorithm

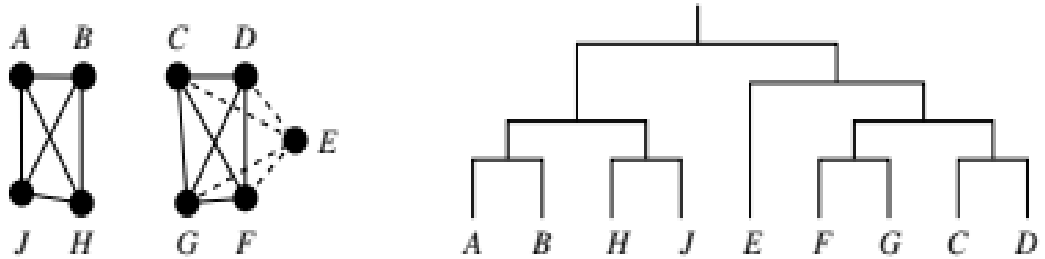
- Nearest-neighbour clustering algorithm.
  - When a clustering algorithm uses minimum distance to measure the distance between the clusters, it is called nearest-neighbour clustering algorithm.
- Single linkage algorithm
  - If the clustering process is terminated, if the minimum distance between the nearest clusters exceeds a user-defined threshold, it is called a single-linkage algorithm
- Agglomerative clustering algorithm that uses minimum distance measure is also called a minimum spanning tree algorithm
- Farthest neighbour clustering algorithm
  - Clustering algorithm uses maximum distance measure to measure the distance between the clusters
- Complete linkage algorithm
  - If the clustering process is terminated, if the maximum distance between the nearest clusters exceeds a user-defined threshold, it is called a complete-linkage algorithm
- Minimum distance and maximum distance measure are two extremes
  - Very sensitive to outliers and noisy data
- Mean and average distance is a compromise between the two and overcomes the outlier and noisy data problem
- Mean distance is simple to compute
- Average distance can handle numeric and categorical data



(a) Data set



(b) Clustering using single linkage



(c) Clustering using complete linkage

Figure 10.8 Hierarchical clustering using single and complete linkages.

Centroid, Radius and Diameter of a Cluster (for numerical data sets)

- Centroid: the "middle" of a cluster

$$C_m = \frac{\sum_{i=1}^{N} (t_{ip})}{N}$$

- Radius: square root of average distance from any point of the cluster to its centroid

$$R_m = \sqrt{\frac{\sum_{i=1}^{N} (t_{ip} - c_m)^2}{N}}$$

- Diameter: square root of average mean squared distance between all pairs of points in the cluster

$$D_m = \sqrt{\frac{\sum_{i=1}^{N} \sum_{i=1}^{N} (t_{ip} - t_{iq})^2}{N(N-1)}}$$

Strengths of Hierarchical Clustering

- Do not have to assume any particular number of clusters
  - Any desired number of clusters can be obtained by ‘cutting’ the dendrogram at the proper level

- They may correspond to meaningful taxonomies
  - Example in biological sciences (e.g., animal kingdom, phylogeny reconstruction, …)

About Agglomerative Clustering Algorithms

- Use a similarity or distance matrix
  - Merge or split one cluster at a time
- Key operation is the computation of the proximity of two clusters
  - Different approaches to defining the distance between clusters distinguish the different algorithms

Hierarchical Clustering: Problems and Limitations

- Once a decision is made to combine two clusters, it cannot be undone
- No objective function is directly minimized
- Different schemes have problems with one or more of the following:
  - Sensitivity to noise and outliers
  - Difficulty handling different sized clusters and convex shapes
  - Breaking large clusters

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

BIRCH (Balanced Iterative Reducing and Clustering Using Hierarchies)

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

Clustering Feature Vector in BIRCH

Clustering Feature (CF): ***CF = (N, LS, SS)***

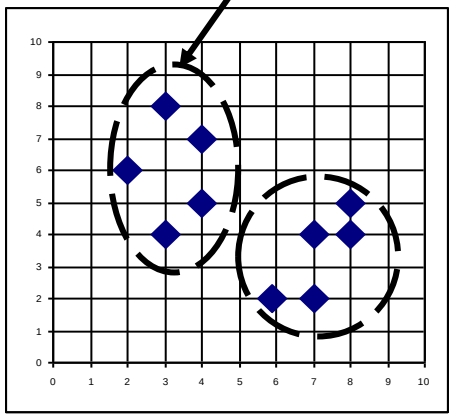
N: **Number of data points**

LS: linear sum of N points: $\sum_{i=1}^{N} X_i$

SS: square sum of N points

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

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

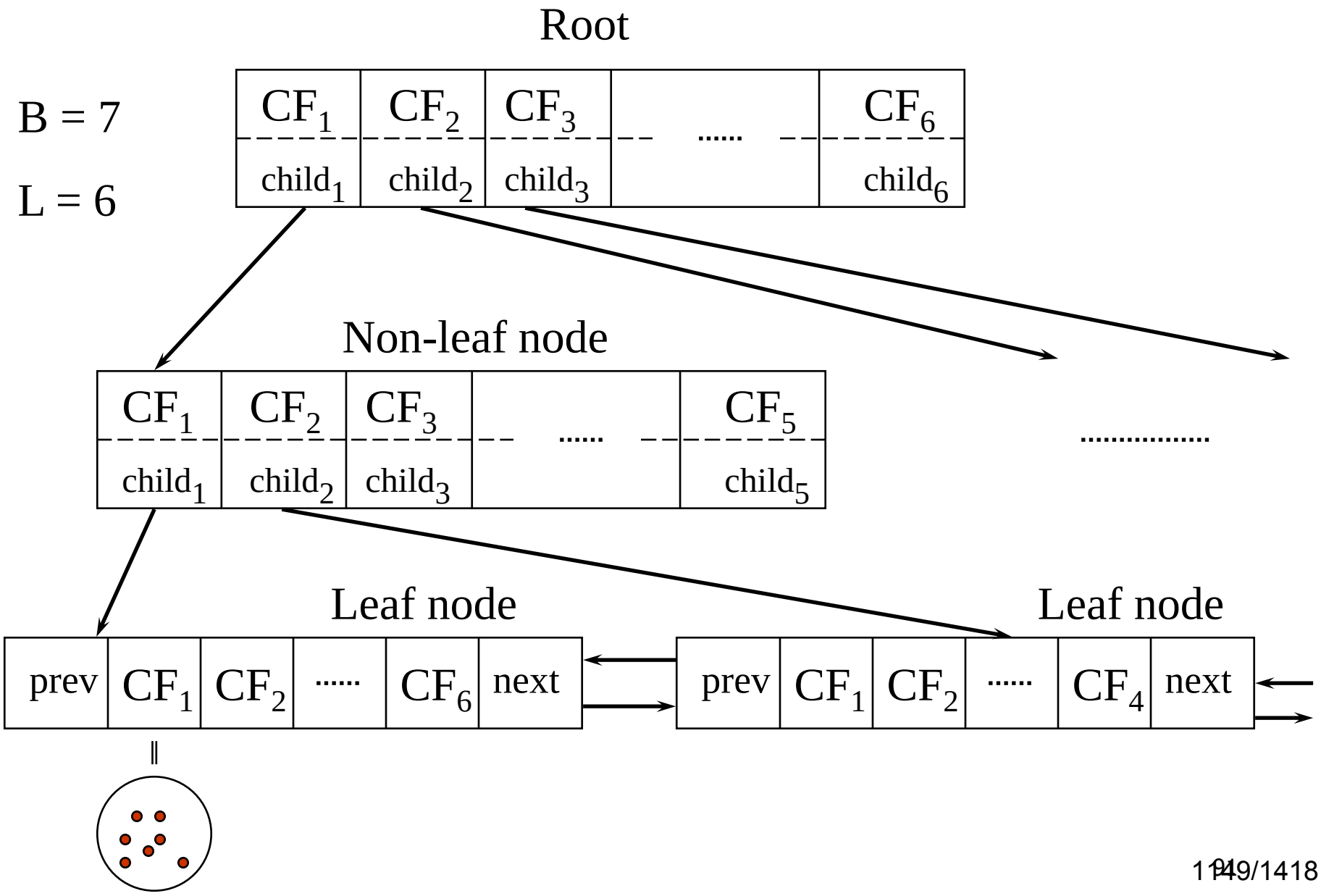


(3,4)
(2,6)
(4,5)
(4,7)
(3,8)

CF-Tree in BIRCH

- Clustering feature:
  - Summary of the statistics for a given subcluster: the 0-th, 1st, and 2nd moments of the subcluster from the statistical point of view
  - Registers crucial measurements for computing cluster and utilizes storage efficiently
- A CF tree is a height-balanced tree that stores the clustering features for a hierarchical clustering
  - A nonleaf node in a tree has descendants or "children"
  - The nonleaf nodes store sums of the CFs of their children
- A CF tree has two parameters
  - Branching factor: max # of children
  - Threshold: max diameter of sub-clusters stored at the leaf nodes

The CF Tree Structure



The Birch Algorithm

- Cluster Diameter

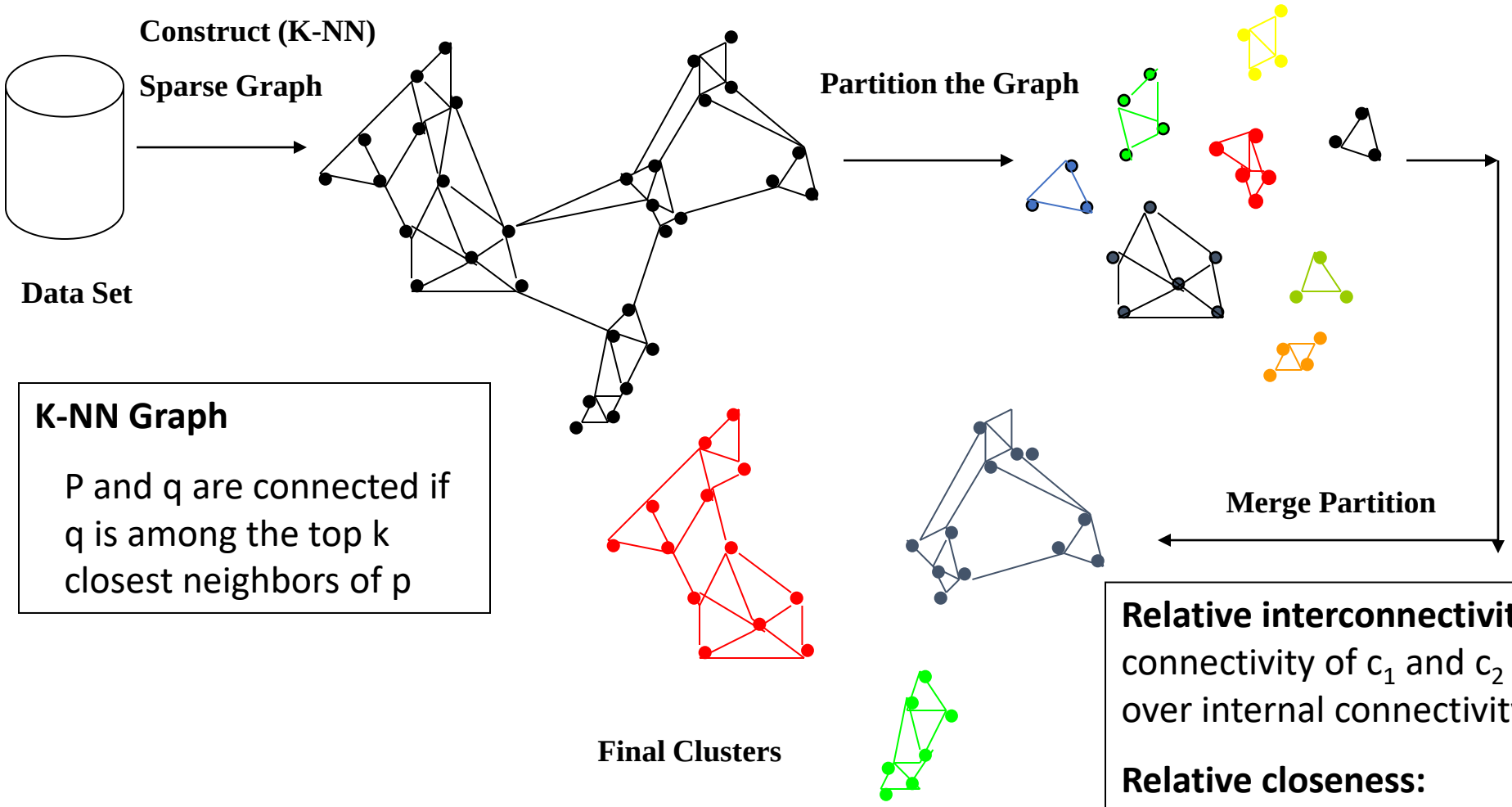
$$\sqrt{\frac{1}{n(n-1)}\sum(x_i - x_j)^2}$$

- For each point in the input
  - Find closest leaf entry
  - Add point to leaf entry and update CF
  - If entry diameter > max_diameter, then split leaf, and possibly parents
- Algorithm is O(n)
- Concerns
  - Sensitive to insertion order of data points
  - Since we fix the size of leaf nodes, so clusters may not be so natural
  - Clusters tend to be spherical given the radius and diameter measures

CHAMELEON: Hierarchical Clustering Using Dynamic Modeling (1999)

- CHAMELEON: G. Karypis, E. H. Han, and V. Kumar, 1999
- Measures the similarity based on a dynamic model
  - Two clusters are merged only if the *interconnectivity* and *closeness (proximity)* between two clusters are high *relative to* the internal interconnectivity of the clusters and closeness of items within the clusters
- Graph-based, and a two-phase algorithm
  1. Use a graph-partitioning algorithm: cluster objects into a large number of relatively small sub-clusters
  2. Use an agglomerative hierarchical clustering algorithm: find the genuine clusters by repeatedly combining these sub-clusters

Overall Framework of CHAMELEON



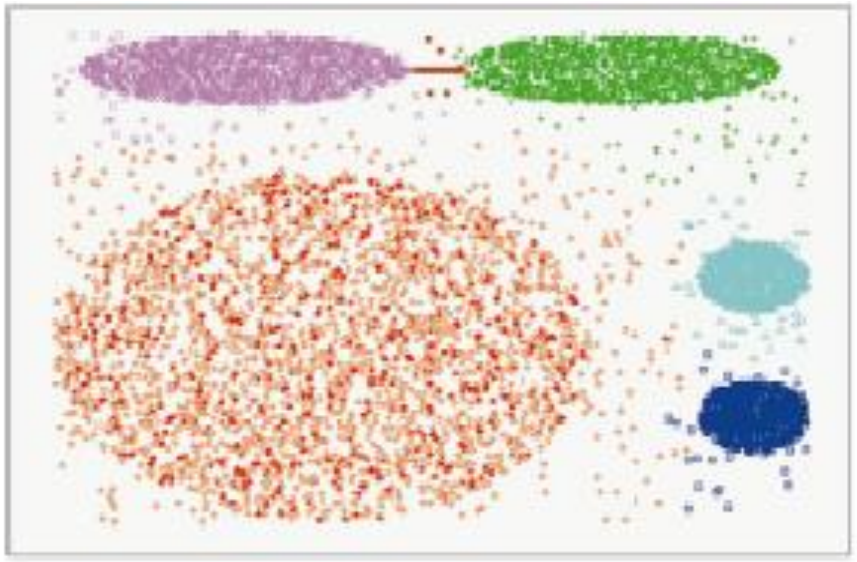
K-NN Graph

P and q are connected if q is among the top k closest neighbors of p

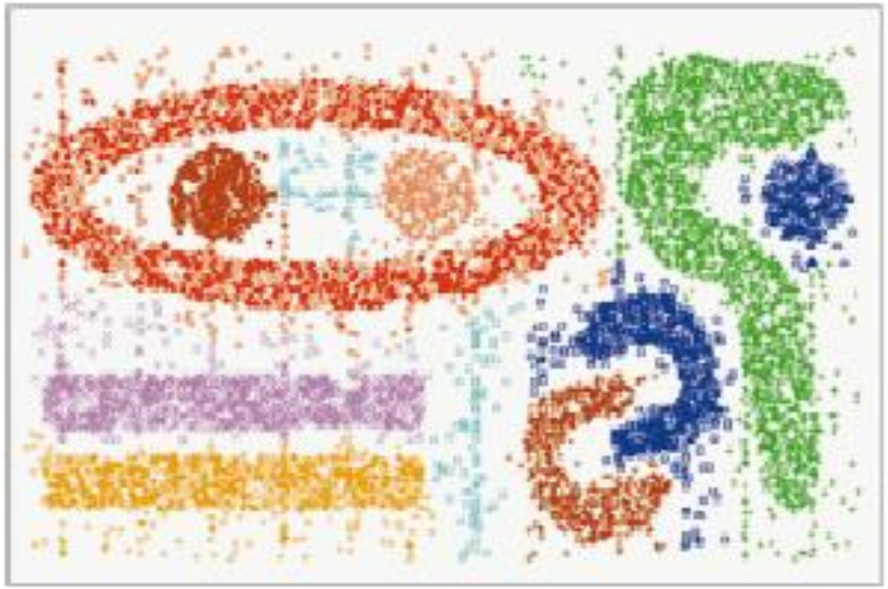
Relative interconnectivity: connectivity of c_1 and c_2 over internal connectivity

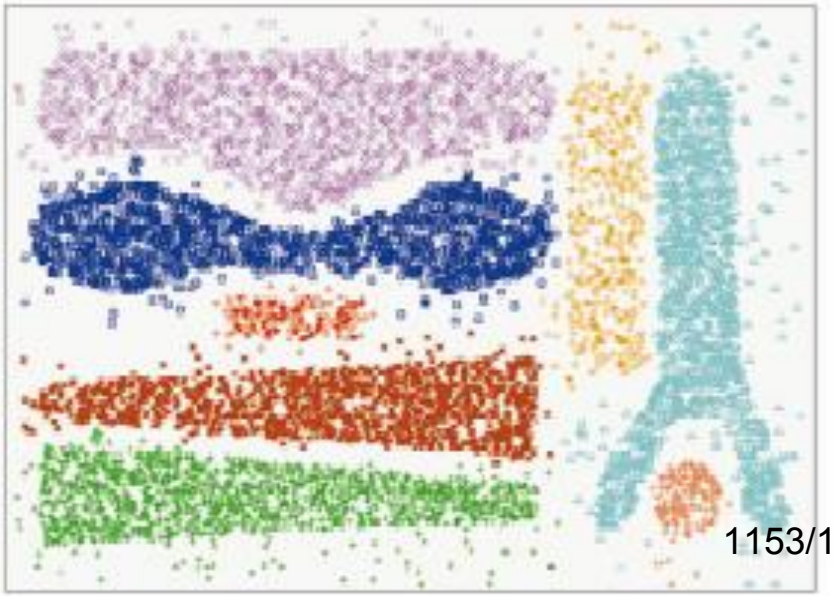
Relative closeness: closeness of c_1 and c_2 over internal closeness

CHAMELEON (Clustering Complex Objects)









Probabilistic Hierarchical Clustering

- Algorithmic hierarchical clustering
  - Nontrivial to choose a good distance measure
  - Hard to handle missing attribute values
  - Optimization goal not clear: heuristic, local search
- Probabilistic hierarchical clustering
  - Use probabilistic models to measure distances between clusters
  - Generative model: Regard the set of data objects to be clustered as a sample of the underlying data generation mechanism to be analyzed
  - Easy to understand, same efficiency as algorithmic agglomerative clustering method, can handle partially observed data
- In practice, assume the generative models adopt common distributions functions, e.g., Gaussian distribution or Bernoulli distribution, governed by parameters

Generative Model

- Given a set of 1-D points $X = \{x_1, \ldots, x_n\}$ for clustering analysis & assuming they are generated by a Gaussian distribution:

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

- The probability that a point $x_i \in X$ is generated by the model

$$P(x_i|\mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x_i-\mu)^2}{2\sigma^2}}$$

- The likelihood that X is generated by the model:

$$L(\mathcal{N}(\mu, \sigma^2) : X) = P(X|\mu, \sigma^2) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x_i-\mu)^2}{2\sigma^2}}$$

- The task of learning the generative model: find the parameters μ and σ^2 such that

the maximum likelihood

$$\mathcal{N}(\mu_0, \sigma_0^2) = \arg\max\{L(\mathcal{N}(\mu, \sigma^2) : X)\}$$

A Probabilistic Hierarchical Clustering Algorithm

- For a set of objects partitioned into *m* clusters $C_1, \ldots, C_m$, the quality can be measured by,

$$Q(\{C_1, \ldots, C_m\}) = \prod_{i=1}^{m} P(C_i)$$

  where *P*() is the maximum likelihood

- Distance between clusters C_1 and C_2:

$$dist(C_i, C_j) = -\log \frac{P(C_1 \cup C_2)}{P(C_1)P(C_2)}$$

- Algorithm: Progressively merge points and clusters

  Input: $D = \{o_1, \ldots, o_n\}$: a data set containing n objects

  Output: A hierarchy of clusters

  Method

    Create a cluster for each object $C_i = \{o_i\}$, $1 \le i \le n$;

    For i = 1 to n {

      Find pair of clusters C_i and C_j such that

        $C_i, C_j = \text{argmax}_{i \neq j} \{\log (P(C_i \cup C_j)/(P(C_i)P(C_j))\}$;

      If $\log (P(C_i \cup C_j)/(P(C_i)P(C_j)) > 0$ then merge C_i and C_j }

Chapter 10. Cluster Analysis: Basic Concepts and Methods

- Cluster Analysis: Basic Concepts
- Partitioning Methods
- Hierarchical Methods
- Density-Based Methods
- Grid-Based Methods
- Evaluation of Clustering
- Summary

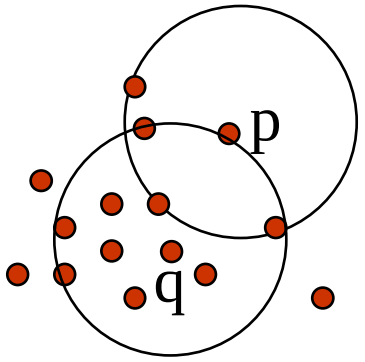
Density-Based Clustering Methods

- 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)

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:

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

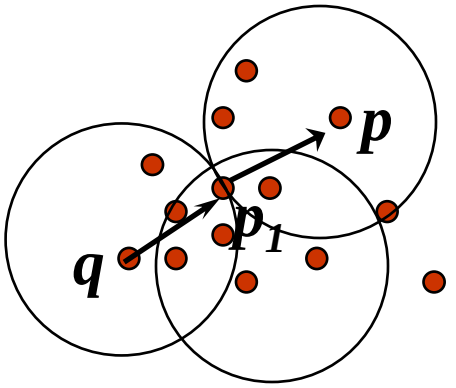


MinPts = 5

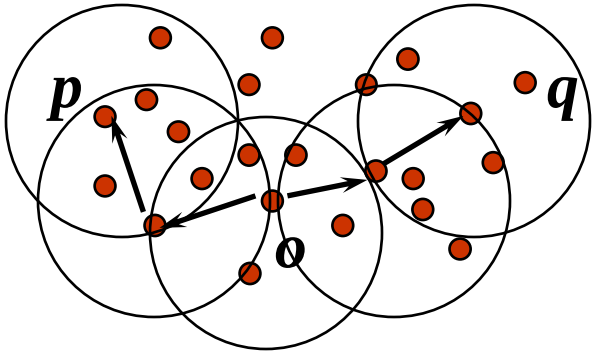
Eps = 1 cm

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 $p_1, \ldots, p_n, p_1 = q, p_n = p$ such that p_{i+1} is directly density-reachable from p_i

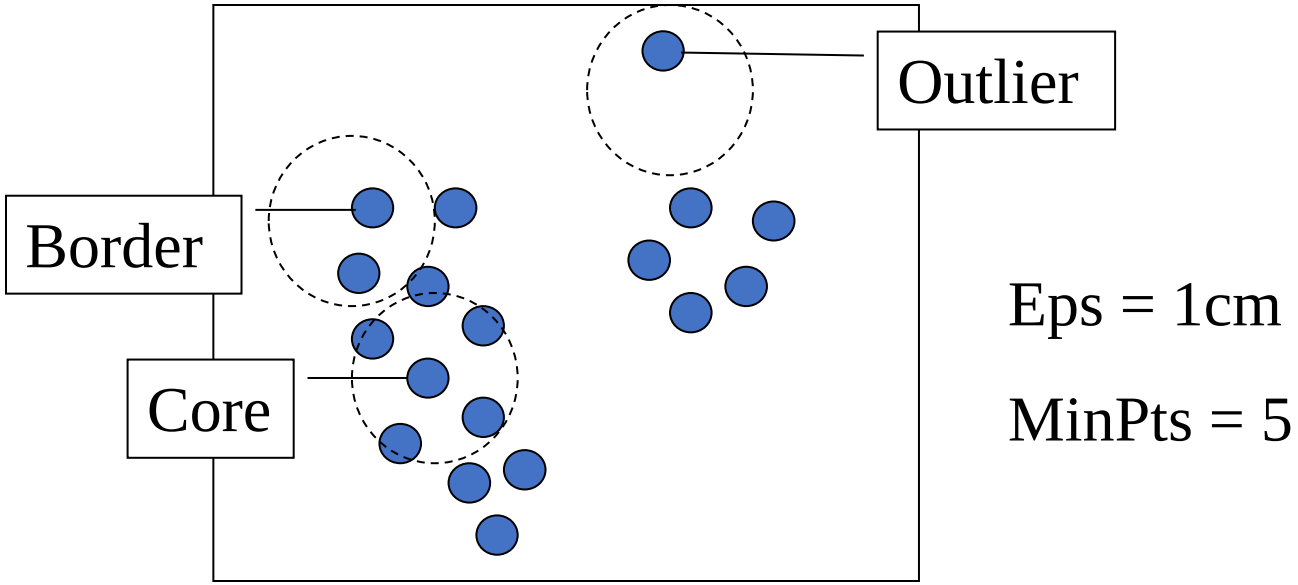


- Density-connected
  - A point *p* is density-connected to a point *q* w.r.t. *Eps*, *MinPts* if there is a point *o* such that both, *p* and *q* are density-reachable from *o* w.r.t. *Eps* and *MinPts*



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

DBSCAN: Sensitive to Parameters

Figure 8. DBScan results for DS1 with MinPts at 4 and Eps at (a) 0.5 and (b) 0.4.

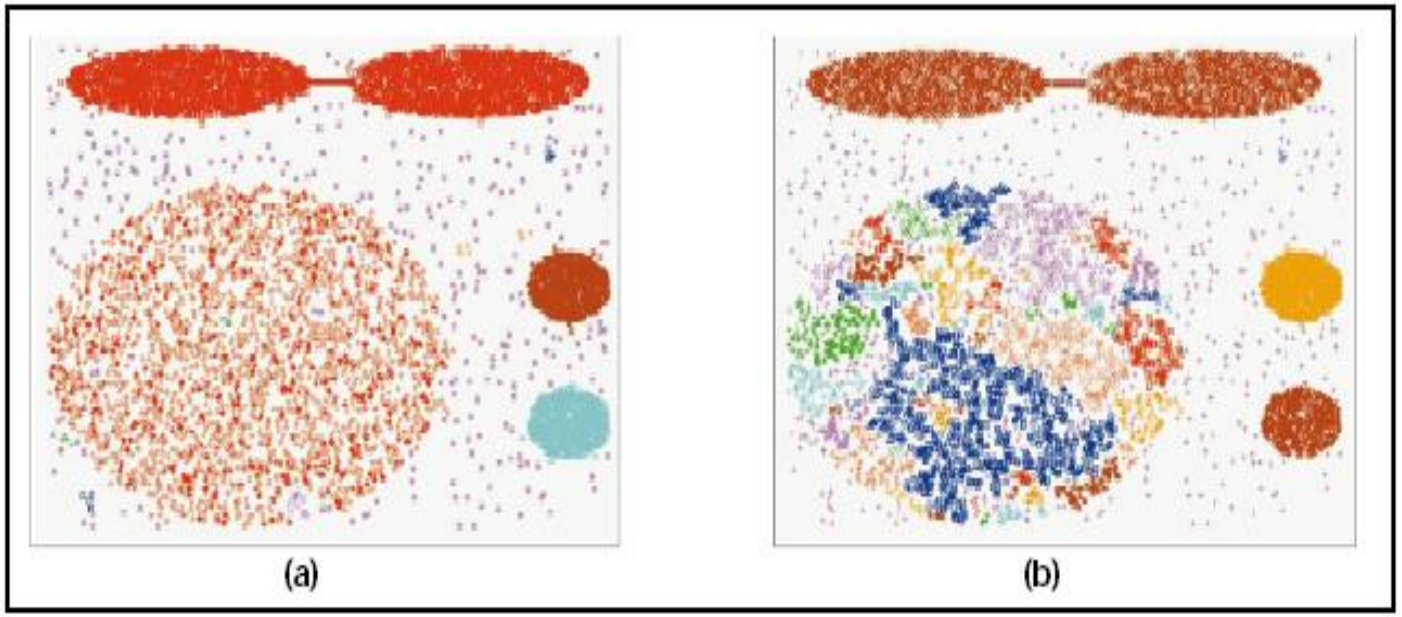
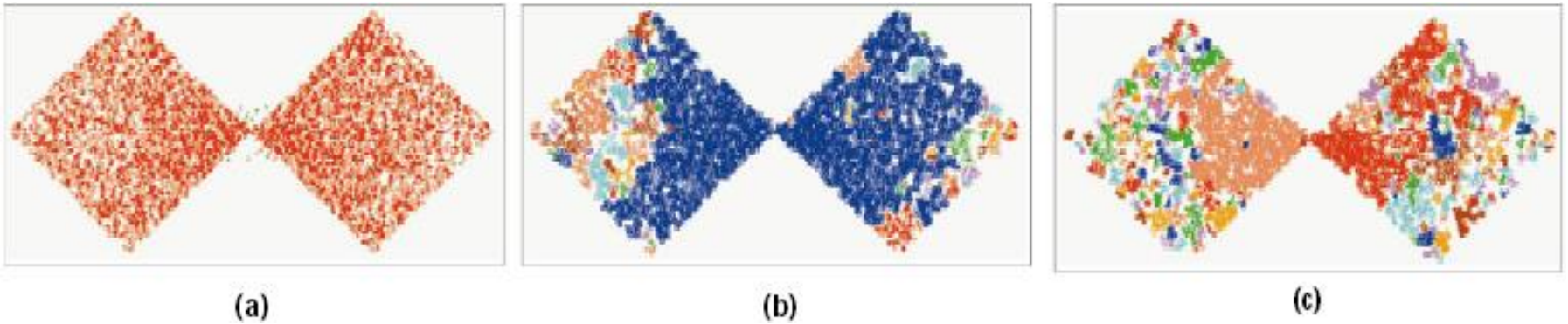


Figure 9. DBScan results for DS2 with MinPts at 4 and Eps at (a) 5.0, (b) 3.5, and (c) 3.0.



OPTICS: A Cluster-Ordering Method (1999)

- OPTICS: Ordering Points To Identify the Clustering Structure
  - Ankerst, Breunig, Kriegel, and Sander (SIGMOD'99)
  - Produces a special order of the database wrt its density-based clustering structure
  - This cluster-ordering contains info equiv to the density-based clusterings corresponding to a broad range of parameter settings
  - Good for both automatic and interactive cluster analysis, including finding intrinsic clustering structure
  - Can be represented graphically or using visualization techniques

3.2.1 Density-Based Cluster-Ordering

To introduce the notion of a density-based cluster-ordering, we first make the following observation: for a constant *MinPts*-value, density-based clusters with respect to a higher density (i.e. a lower value for ε) are completely contained in density-connected sets with respect to a lower density (i.e. a higher value for ε). This fact is illustrated in figure 3, where C_1 and C_2 are density-based clusters with respect to $\varepsilon_2 < \varepsilon_1$ and C is a density-based cluster with respect to ε_1 completely containing the sets C_1 and C_2.

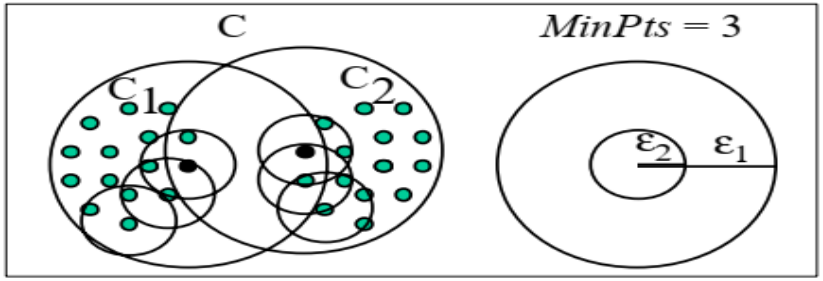
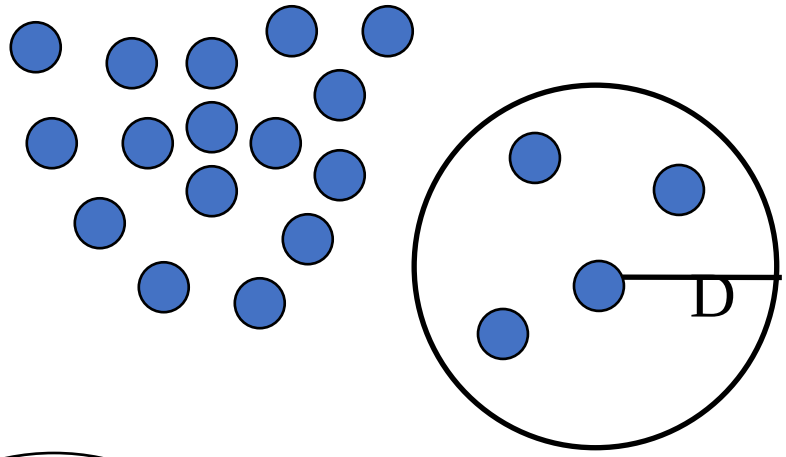


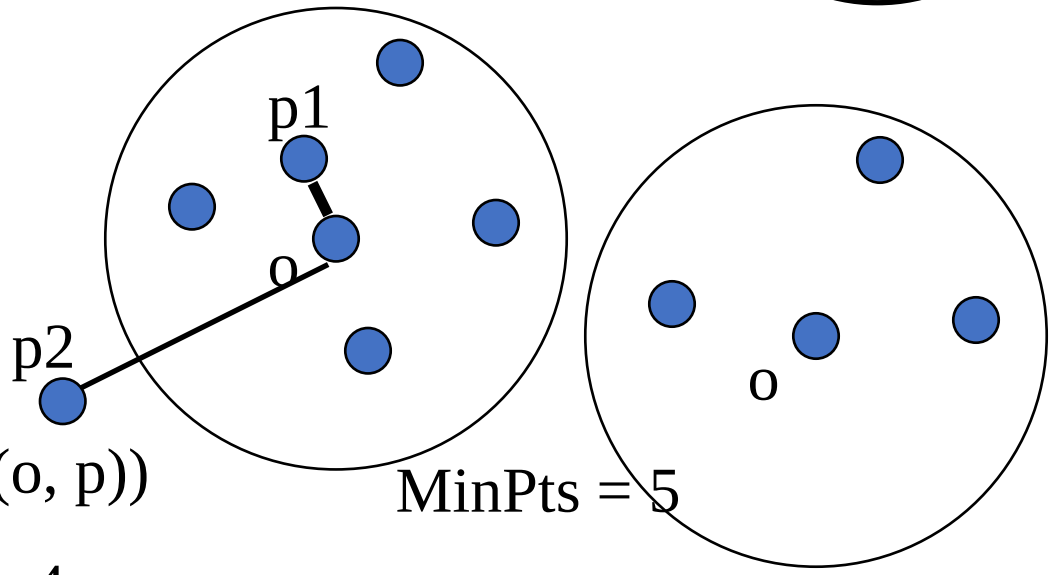
Figure 3. Illustration of "nested" density-based clusters

Consequently, we could extend the DBSCAN algorithm such that several distance parameters are processed at the same time, i.e. the density-based clusters with respect to different densities are constructed simultaneously. To produce a consistent result, however, we would have to obey a specific *order* in which objects are processed when expanding a cluster. We always have to select an object which is density-reachable with respect to the lowest ε value to guarantee that clusters with respect to higher density (i.e. smaller ε values) are finished first.

OPTICS: Some Extension from DBSCAN

- Index-based:
  - k = number of dimensions
  - N = 20
  - p = 75%
  - M = N(1-p) = 5
  - Complexity: O(*NlogN*)
- Core Distance:
  - min eps s.t. point is core
- Reachability Distance



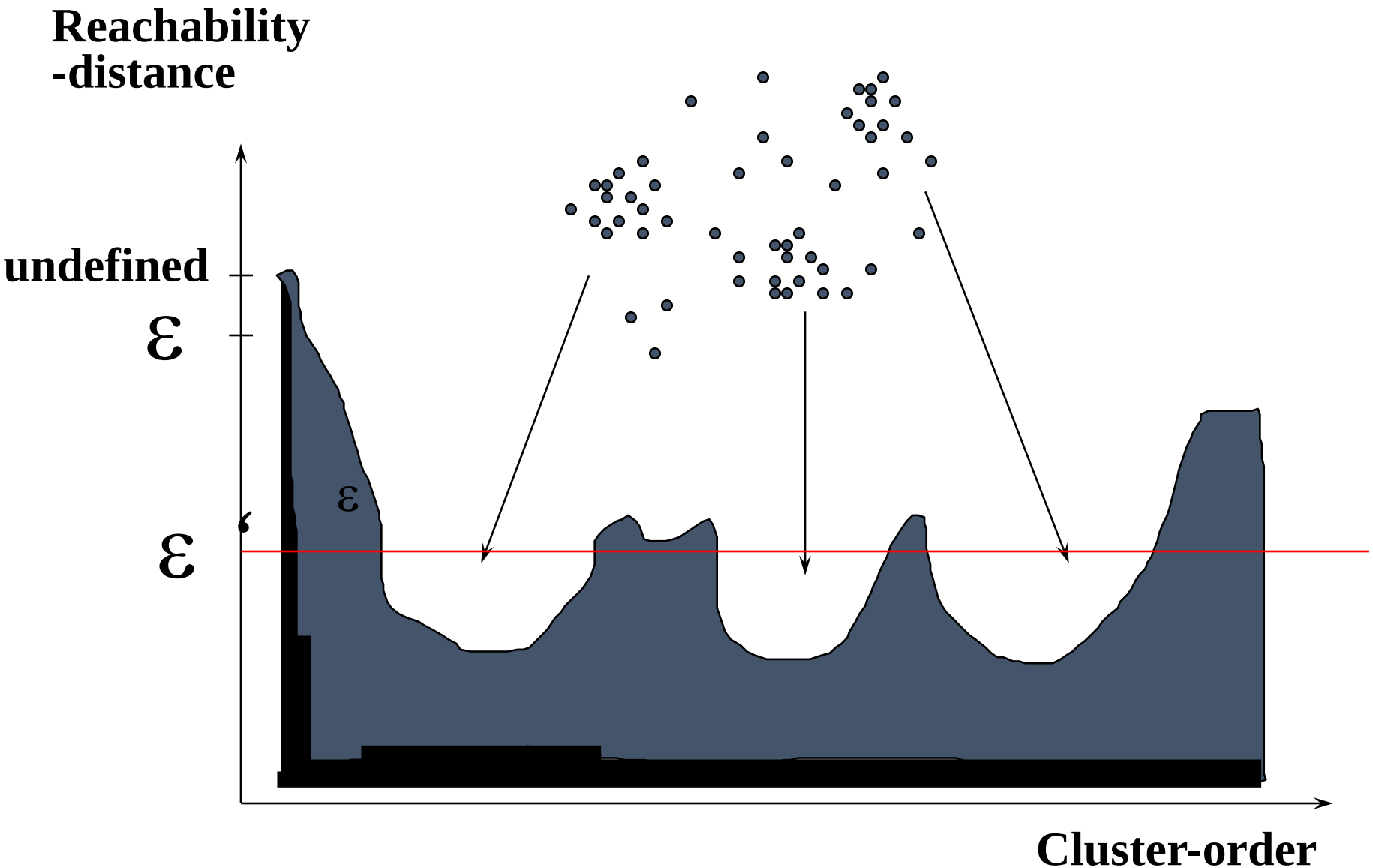


Max (core-distance (o), d (o, p))

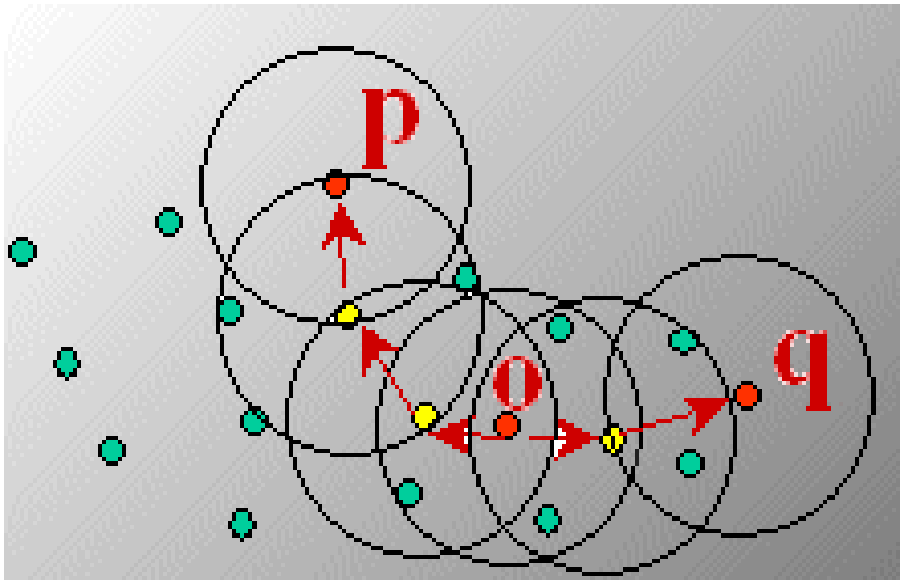
r(p1, o) = 2.8cm. r(p2,o) = 4cm

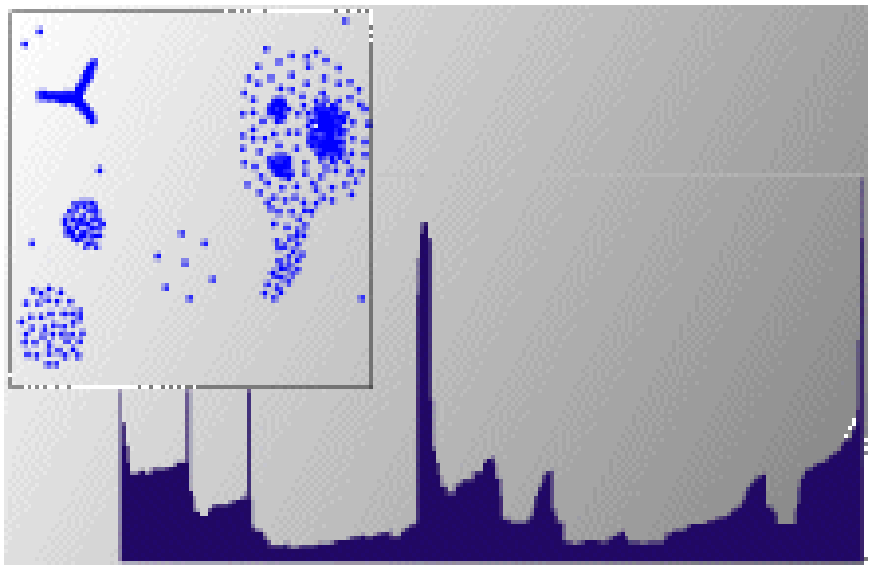
MinPts = 5

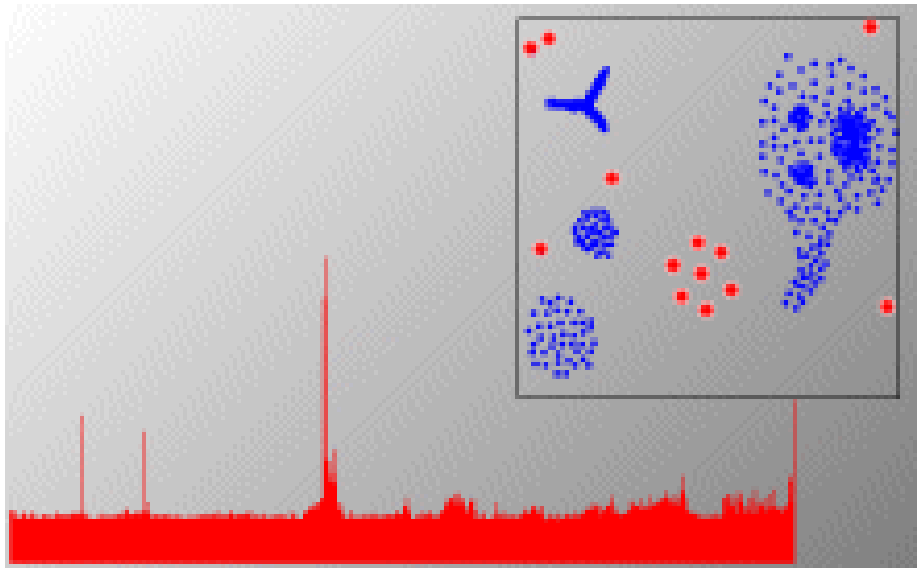
ε = 3 cm

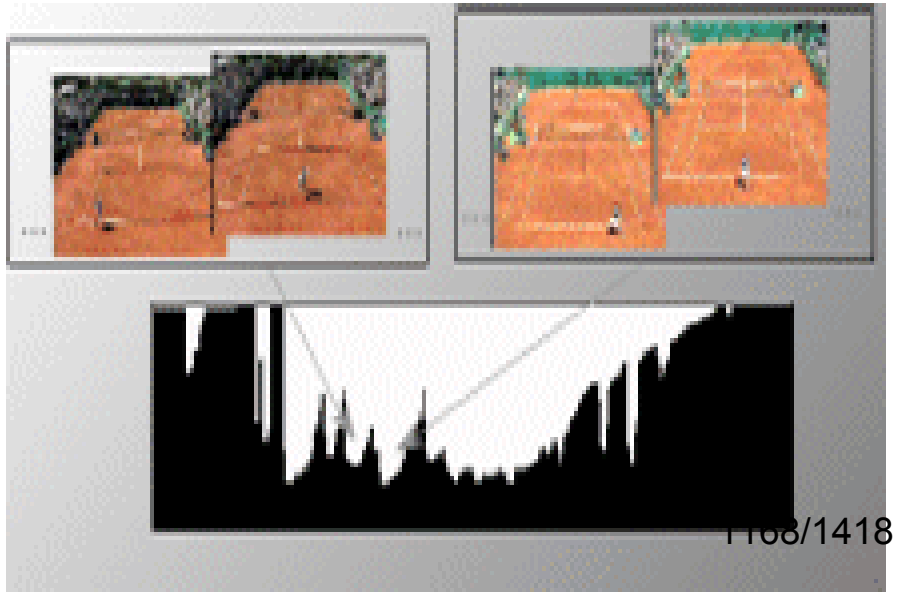

Reachability
-distance
undefined
ε
ε
ε
Cluster-order
of the objects

Density-Based Clustering: OPTICS & Its Applications









DENCLUE: Using Statistical Density Functions

- DENsity-based CLUstEring by Hinneburg & Keim (KDD'98)
- Using statistical density functions:

$$f_{Gaussian}(x,y) = e^{-\frac{d(x,y)^2}{2\sigma^2}}$$

influence of y on x

$$f^{D}_{Gaussian}(x) = \sum_{i=1}^{N} e^{-\frac{d(x,x_i)^2}{2\sigma^2}}$$

total influence on x

$$\nabla f^{D}_{Gaussian}(x,x_i) = \sum_{i=1}^{N} (x_i - x) \cdot e^{-\frac{d(x,x_i)^2}{2\sigma^2}}$$

gradient of x in the direction of x_i

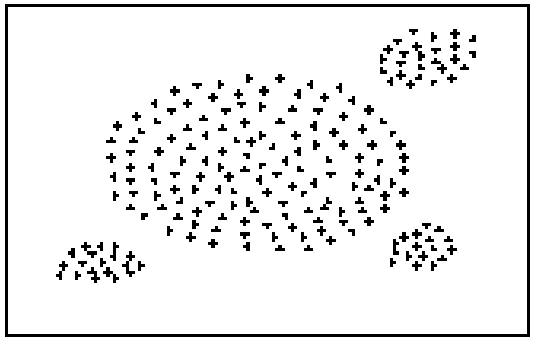
- Major features
  - Solid mathematical foundation
  - Good for data sets with large amounts of noise
  - Allows a compact mathematical description of arbitrarily shaped clusters in high-dimensional data sets
  - Significant faster than existing algorithm (e.g., DBSCAN)
  - But needs a large number of parameters

Denclue: Technical Essence

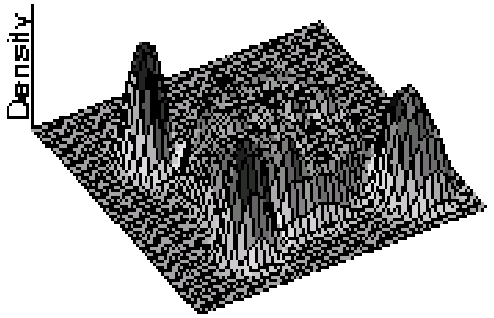
- Uses grid cells but only keeps information about grid cells that do actually contain data points and manages these cells in a tree-based access structure
- **Influence function: describes the impact of a data point within its neighborhood**
- Overall density of the data space can be calculated as the sum of the influence function of all data points
- Clusters can be determined mathematically by **identifying density attractors**
- Density attractors are local maximal of the overall density function
- Center defined clusters: assign to each density attractor the points density attracted to it
- Arbitrary shaped cluster: **merge density attractors that are connected through paths of high density (> threshold)**

1-17/0/1418

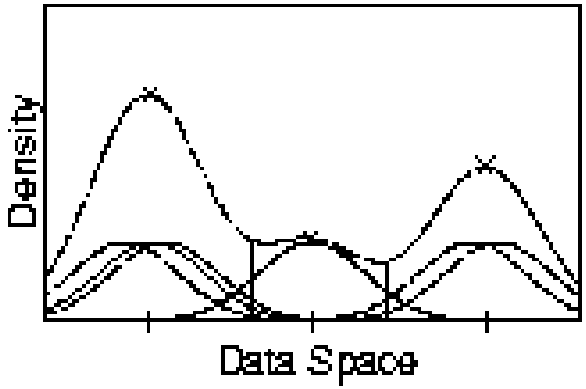
Density Attractor



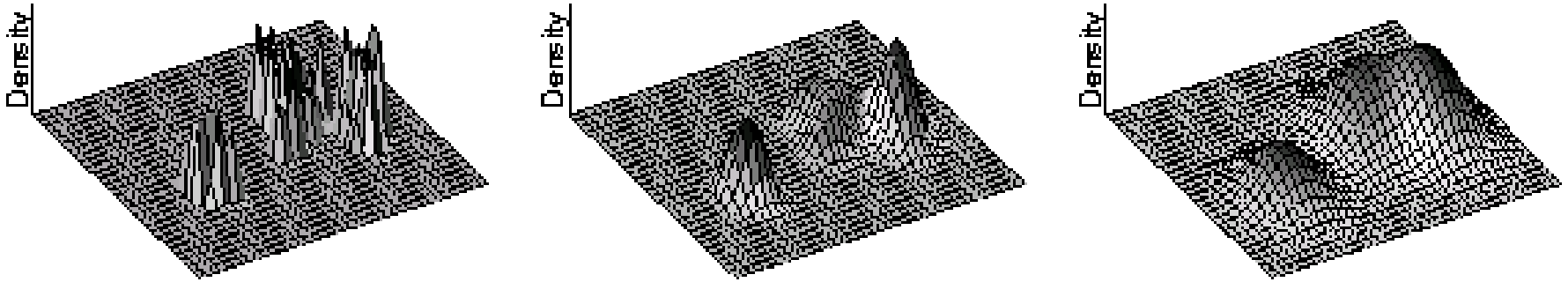
(a) Data Set



(c) Gaussian

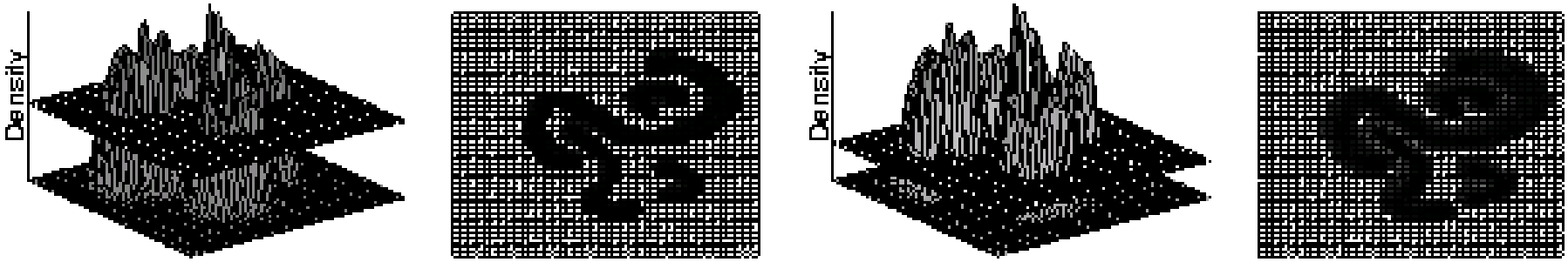


Center-Defined and Arbitrary



(a) $\sigma = 0.2$ (b) $\sigma = 0.6$ (d) $\sigma = 1.5$

Figure 3: Example of Center-Defined Clusters for different σ



(a) $\xi = 2$ (b) $\xi = 2$ (c) $\xi = 1$ (d) $\xi = 1$

Figure 4: Example of Arbitray-Shape Clusters for different ξ

Chapter 10. Cluster Analysis: Basic Concepts and Methods

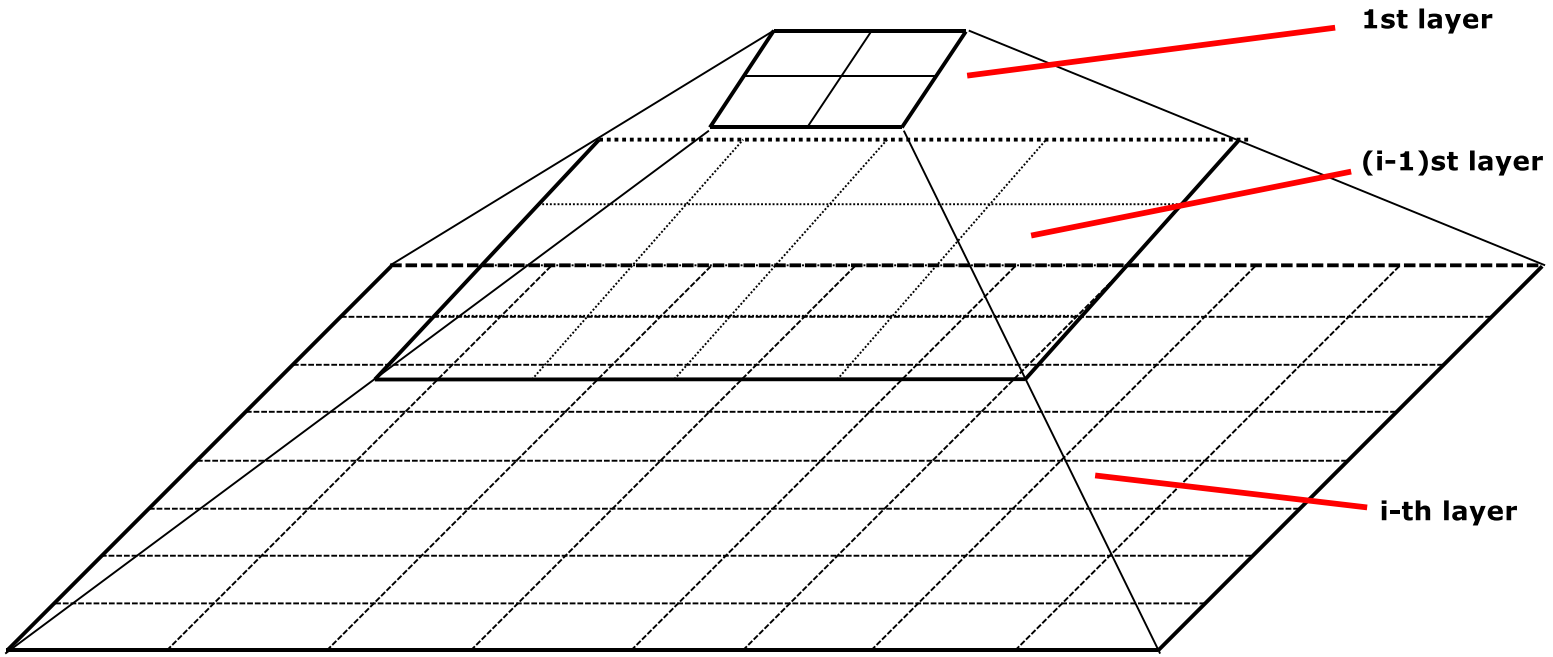
- Cluster Analysis: Basic Concepts
- Partitioning Methods
- Hierarchical Methods
- Density-Based Methods
- **Grid-Based Methods**
- Evaluation of Clustering
- Summary

Grid-Based Clustering Method

- Using multi-resolution grid data structure
- Several interesting methods
  - STING (a STatistical INformation Grid approach) by Wang, Yang and Muntz (1997)
  - CLIQUE: Agrawal, et al. (SIGMOD'98)
    - Both grid-based and subspace clustering

STING: A Statistical Information Grid Approach

- Wang, Yang and Muntz (VLDB'97)
- The spatial area is divided into rectangular cells
- There are several levels of cells corresponding to different levels of resolution



The STING Clustering Method

- Each cell at a high level is partitioned into a number of smaller cells in the next lower level
- Statistical info of each cell is calculated and stored beforehand and is used to answer queries
- Parameters of higher level cells can be easily calculated from parameters of lower level cell
  - *count*, *mean*, *s*, *min*, *max*
  - type of distribution—*normal*, *uniform*, etc.
- Use a top-down approach to answer spatial data queries
- Start from a pre-selected layer—typically with a small number of cells
- For each cell in the current level compute the confidence interval

STING Algorithm and Its Analysis

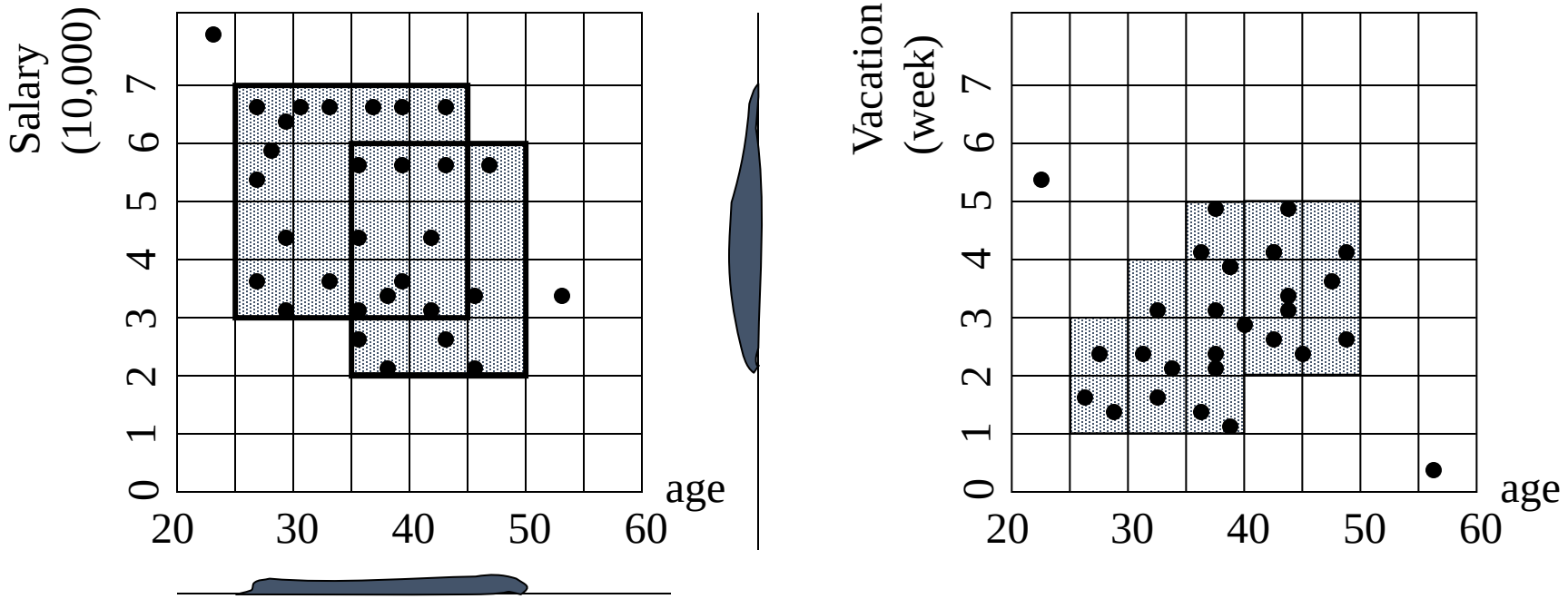
- Remove the irrelevant cells from further consideration
- When finish examining the current layer, proceed to the next lower level
- Repeat this process until the bottom layer is reached
- Advantages:
  - Query-independent, easy to parallelize, incremental update
  - *O(K),* where *K* is the number of grid cells at the lowest level
- Disadvantages:
  - All the cluster boundaries are either horizontal or vertical, and no diagonal boundary is detected

CLIQUE (Clustering In QUEst)

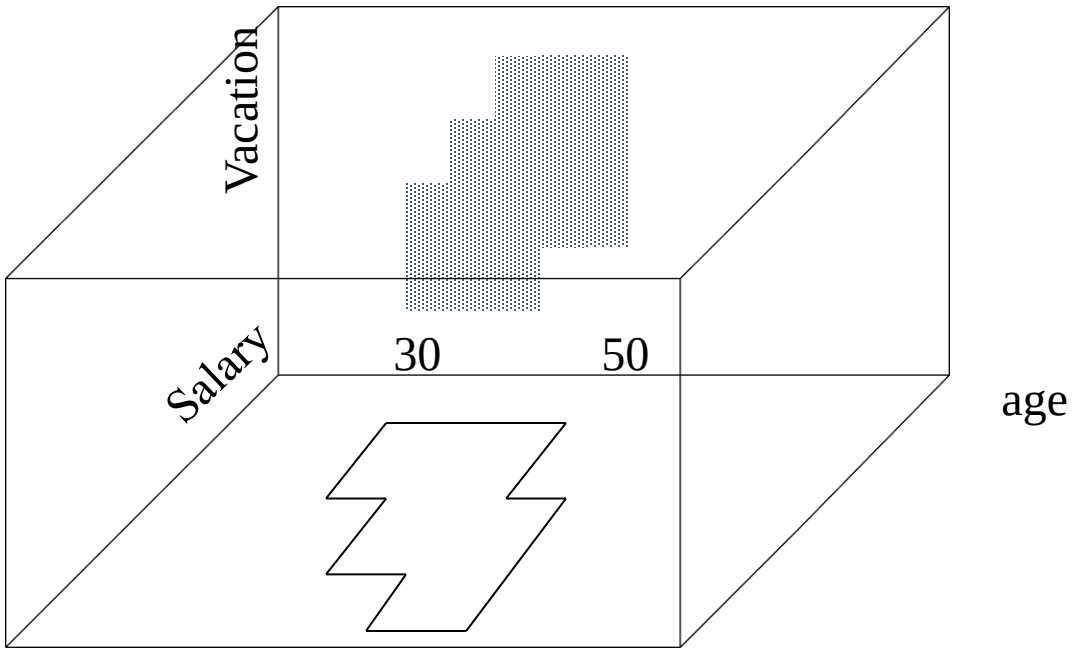
- Agrawal, Gehrke, Gunopulos, Raghavan (SIGMOD'98)
- Automatically identifying subspaces of a high dimensional data space that allow better clustering than original space
- CLIQUE can be considered as both density-based and grid-based
  - It partitions each dimension into the same number of equal length interval
  - It partitions an m-dimensional data space into non-overlapping rectangular units
  - A unit is dense if the fraction of total data points contained in the unit exceeds the input model parameter
  - A cluster is a maximal set of connected dense units within a subspace

CLIQUE: The Major Steps

- Partition the data space and find the number of points that lie inside each cell of the partition.
- Identify the subspaces that contain clusters using the Apriori principle
- Identify clusters
  - Determine dense units in all subspaces of interests
  - Determine connected dense units in all subspaces of interests.
- Generate minimal description for the clusters
  - Determine maximal regions that cover a cluster of connected dense units for each cluster
  - Determination of minimal cover for each cluster


Salary
(10,000)
0 1 2 3 4 5 6 7
20 30 40 50 60
age
Vacation
(week)
0 1 2 3 4 5 6 7
20 30 40 50 60
age

$\tau = 3$


Vacation
Salary
30
50
age

Strength and Weakness of *CLIQUE*

- Strength
  - *automatically* finds subspaces of the highest dimensionality such that high density clusters exist in those subspaces
  - *insensitive* to the order of records in input and does not presume some canonical data distribution
  - scales *linearly* with the size of input and has good scalability as the number of dimensions in the data increases
- Weakness
  - The accuracy of the clustering result may be degraded at the expense of simplicity of the method

Chapter 10. Cluster Analysis: Basic Concepts and Methods

- Cluster Analysis: Basic Concepts
- Partitioning Methods
- Hierarchical Methods
- Density-Based Methods
- Grid-Based Methods
- **Evaluation of Clustering**
- Summary

Assessing Clustering Tendency

- Assess if non-random structure exists in the data by measuring the probability that the data is generated by a uniform data distribution
- Test spatial randomness by statistic test: Hopkins Static
  - Given a dataset D regarded as a sample of a random variable o, determine how far away o is from being uniformly distributed in the data space
  - Sample n points, $p_1, \ldots, p_n$, uniformly from D. For each p_i, find its nearest neighbor in D: $x_i = min\{dist(p_i, v)\}$ where v in D
  - Generate a simulated dataset $q_1, \ldots, q_n$, uniformly from D. For each q_i, find its nearest neighbor in D – $\{q_i\}$: $y_i = min\{dist(q_i, v)\}$ where v in D and $v \neq q_i$
  - Calculate the Hopkins Statistic: $$H = \frac{\sum_{i=1}^{n} y_i}{\sum_{i=1}^{n} x_i + \sum_{i=1}^{n} y_i}$$
  - If D is uniformly distributed, $\sum x_i$ and $\sum y_i$ will be close to each other and H is close to 0.5. If D is highly skewed, H is close to zero.

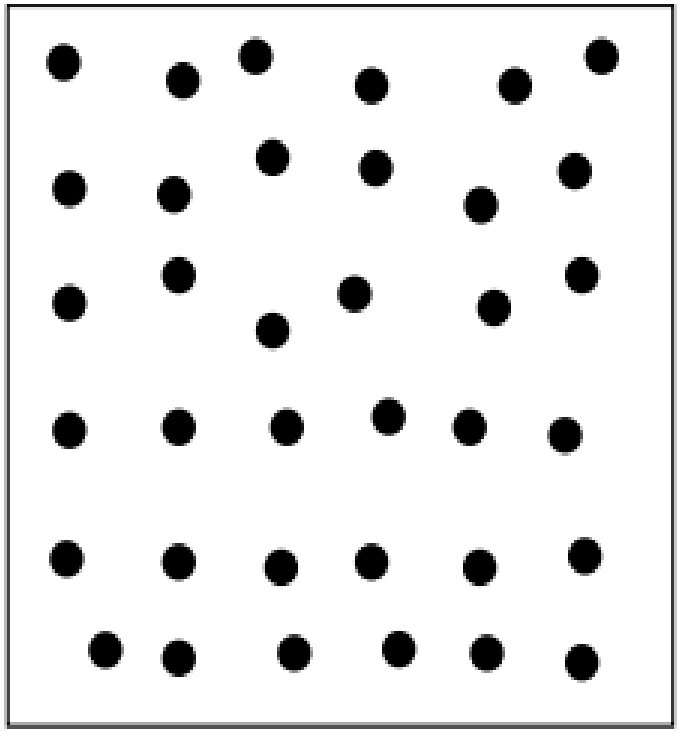
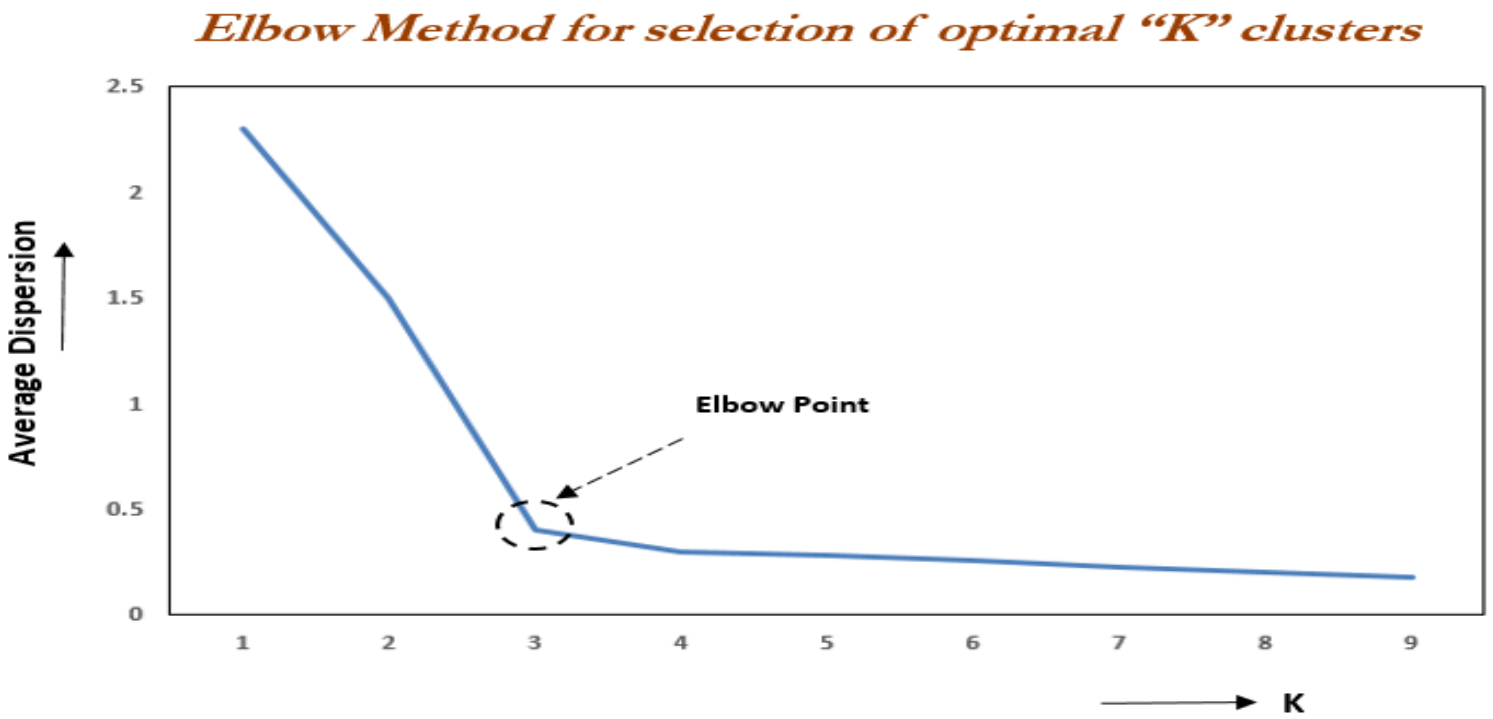


Figure 10.21 A data set that is uniformly distributed in the data space.

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



Determine the Number 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

Extrinsic Methods: BCubed Precession and Recall Metrics

- BCubed evaluates precesion and recall of every object based on ground truth.
- Precession of the object: How many other objects in the same cluster belong to the same category as the object.
- Recall of the object: how many objects of the same category are assigned to the same cluster.

Formally, let $D = \{o_1, \ldots, o_n\}$ be a set of objects, and C be a clustering on D. Let $L(o_i)$ $(1 \leq i \leq n)$ be the category of o_i given by ground truth, and $C(o_i)$ be the *cluster_ID* of o_i in C. Then, for two objects, o_i and o_j, $(1 \leq i, j, \leq n, i \neq j)$, the *correctness* of the relation between o_i and o_j in clustering C is given by

$$\text{Correctness}(o_i, o_j) = \begin{cases} 1 & \text{if } L(o_i) = L(o_j) \Leftrightarrow C(o_i) = C(o_j) \\ 0 & \text{otherwise.} \end{cases} \tag{10.28}$$

BCubed precision is defined as

$$\text{Precision BCubed} = \frac{\sum_{i=1}^{n} \frac{\sum_{o_j : i \neq j, C(o_i) = C(o_j)} \text{Correctness}(o_i, o_j)}{\|\{o_j | i \neq j, C(o_i) = C(o_j)\}\|}}{n}. \tag{10.29}$$

BCubed recall is defined as

$$\text{Recall BCubed} = \frac{\sum_{i=1}^{n} \frac{\sum_{o_j : i \neq j, L(o_i) = L(o_j)} \text{Correctness}(o_i, o_j)}{\|\{o_j | i \neq j, L(o_i) = L(o_j)\}\|}}{n}. \tag{10.30}$$

Intrinsic Methods

- When the ground truth of a data set is not available, we have to use an intrinsic method to assess the clustering quality.
- In general, intrinsic methods evaluate a clustering by examining how well the clusters are separated and how compact the clusters are.

The **silhouette coefficient** is such a measure. For a data set, D, of n objects, suppose D is partitioned into k clusters, $C_1, \ldots, C_k$. For each object $o \in D$, we calculate $a(o)$ as the average distance between o and all other objects in the cluster to which o belongs. Similarly, $b(o)$ is the minimum average distance from o to all clusters to which o does not belong. Formally, suppose $o \in C_i$ $(1 \leq i \leq k)$; then

$$a(o) = \frac{\sum_{o' \in C_i, o \neq o'} dist(o, o')}{|C_i| - 1} \tag{10.31}$$

and

$$b(o) = \min_{C_j : 1 \leq j \leq k, j \neq i} \left\{ \frac{\sum_{o' \in C_j} dist(o, o')}{|C_j|} \right\}. \tag{10.32}$$

The **silhouette coefficient** of o is then defined as

$$s(o) = \frac{b(o) - a(o)}{\max\{a(o), b(o)\}}. \tag{10.33}$$

High S(o) value is preferable

Summary

- Cluster Analysis: Basic Concepts
- Partitioning Methods
- Hierarchical Methods
- Density-Based Methods
- Grid-Based Methods
- Evaluation of Clustering
- Summary

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

BIRCH:
An Efficient Data Clustering Method for Very Large Databases

Tian Zhang, Raghu Ramakrishnan,
Miron Livny

ACM SIGMOD 1996

Problem Introduction

- Data clustering
  - *How do we divide* n *data points into* k *groups?*
    - *How do we minimize the difference within the groups?*
    - *How do we maximize the difference between different groups?*
  - *How do we avoid trying all possible solutions?*

➢Very large data sets

➢Limited computational resources

Outline

- Problem introduction
- Previous work
- Introduction to *BIRCH*
- The algorithm
- Experimental results
- Conclusions & practical use

Previous Work

- Unsupervised conceptual learning
- Two classes of clustering algorithms:
  - *Probability-Based*
    - *COBWEB and CLASSIT*
  - *Distance-Based*
    - *KMEANS, KMEDOIDS and CLARANS*

Previous work: COBWEB

- Probabilistic approach to make decisions
  - *Probabilistic measure: Category Utility*
  - *Clusters are represented with probabilistic description*
- Incrementally generates a hierarchy
  - *Cluster data points one at a time*
  - *Maximizes Category Utility at each decision point*

Previous work: COBWEB

- Computing category utility is very expensive
  - *Attributes are assumed to be statistically independent*
- Only works with discrete values
  - *CLASSIT is similar, but is adapted to only handle continuous data*
- Every instance translates into a terminal node in the hierarchy
  - *Infeasible for large data sets*
  - *Large hierarchies tend to over fit data*

Previous work: KMEANS

- Distance based approach
  - *There must be a distance measurement between any two instances (data points)*
- Iteratively groups instances towards the nearest centroid to minimize distances
- Converges on a local minimum
  - *Sensitive to instance order*
- May have exponential run time (worst case)

Previous work: KMEANS

- Some assumptions:
  - All instances must be initially available
  - Instances must be stored in memory
  - Frequent scan (non-incremental)
- Global methods at the granularity of data points
  - All instances are considered individually
  - ➢ Not all data are equally important for clustering
  - ➢ Close or dense ones could be considered collectively!

Introduction to BIRCH

- *BIRCH*: Balanced Iterative Reducing and Clustering using Hierarchies
- Only works with "metric" attributes
  - *Must have Euclidean coordinates*
- Designed for very large data sets
  - *Time and memory constraints are explicit*
  - *Treats dense regions of data points as sub-clusters*
    - Not all data points are important for clustering
  - *Only one scan of data is necessary*

Introduction to BIRCH

- Incremental, distance-based approach
  - Decisions are made without scanning all data points, or all currently existing clusters
  - Does not need the whole data set in advance
  - Unique approach: Distance-based algorithms generally need all the data points to work
- Make best use of available memory while minimizing I/O costs
- Does not assume that the probability distributions on attributes is independent

Background

Given a cluster of instances $\{\vec{X}_i\}$*, we define:*

Centroid: $\vec{X}0 = \frac{\sum_{i=1}^{N} \vec{X}_i}{N}$

Radius: *average distance from member points to centroid*

$$R = \left(\frac{\sum_{i=1}^{N} (\vec{X}_i - \vec{X}0)^2}{N}\right)^{\frac{1}{2}}$$

Diameter: *average pair-wise distance within a cluster*

$$D = \left(\frac{\sum_{i=1}^{N} \sum_{j=1}^{N} (\vec{X}_i - \vec{X}_j)^2}{N(N-1)}\right)^{\frac{1}{2}}$$

Background

*We define the **centroid Euclidean distance** and **centroid Manhattan distance** between any two clusters as:*

$$D0 = ((\vec{X}0_1 - \vec{X}0_2)^2)^{\frac{1}{2}}$$

$$D1 = |\vec{X}0_1 - \vec{X}0_2| = \sum_{i=1}^{d} |\vec{X}0_1^{(i)} - \vec{X}0_1^{(i)}|$$

Background

*We define the **average inter-cluster**, the **average intra-cluster**, and the **variance increase** distances as:*

$$D2 = \left(\frac{\sum_{i=1}^{N_1}\sum_{j=N_1+1}^{N_1+N_2}(\vec{X}_i - \vec{X}_j)^2}{N_1 N_2}\right)^{\frac{1}{2}}$$

$$D3 = \left(\frac{\sum_{i=1}^{N_1+N_2}\sum_{j=1}^{N_1+N_2}(\vec{X}_i - \vec{X}_j)^2}{(N_1+N_2)(N_1+N_2-1)}\right)^{\frac{1}{2}}$$

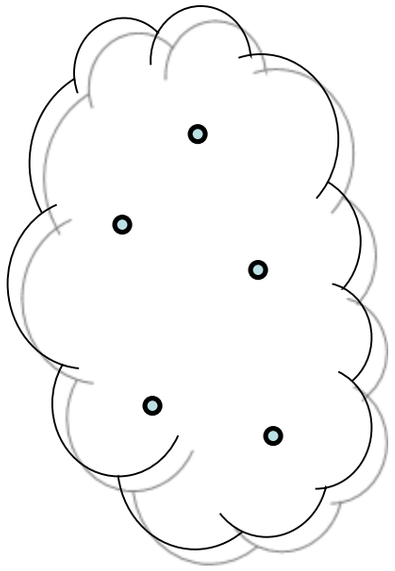
$$D4 = \left(\sum_{k=1}^{N_1+N_2}\left(\vec{X}_k - \frac{\sum_{l=1}^{N_1+N_2}\vec{X}_l}{N_1+N_2}\right)^2 - \sum_{i=1}^{N_1}\left(\vec{X}_i - \frac{\sum_{l=1}^{N_1}\vec{X}_l}{N_1}\right)^2 - \sum_{j=N_1+1}^{N_1+N_2}\left(\vec{X}_j - \frac{\sum_{l=N_1+1}^{N_1+N_2}\vec{X}_l}{N_2}\right)^2\right)^{\frac{1}{2}}$$

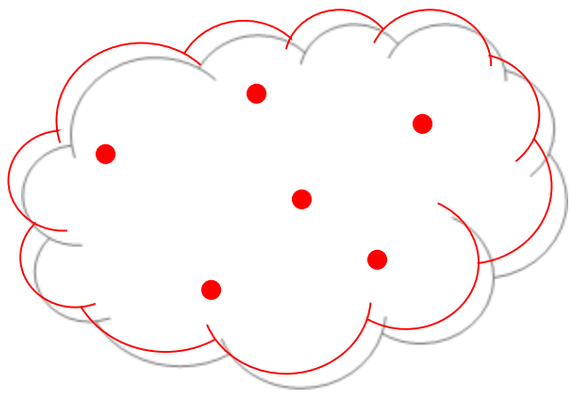
Background

- *Cluster {*X_i*}:*
- i = *1, 2,* ..., N_1

Cluster {X_j*}:*

j = N_1+*1*, N_1+*2*, ..., N_1+N_2

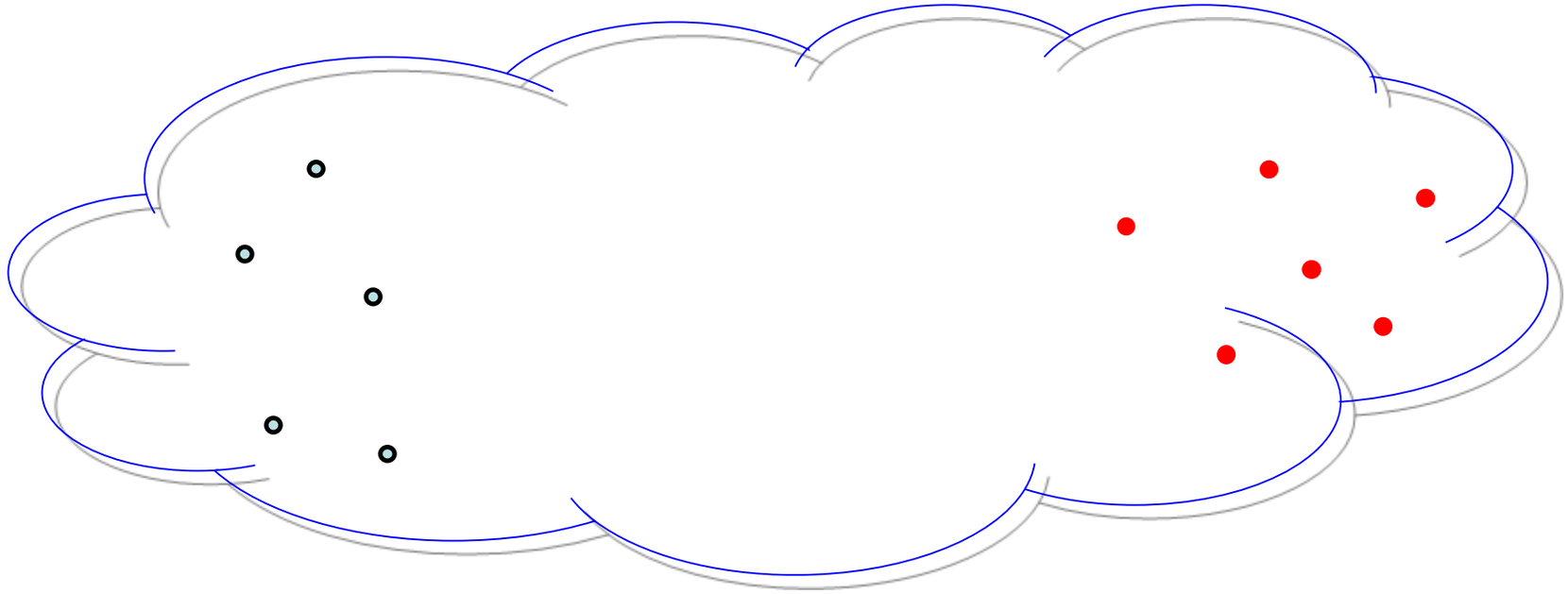




Background

Cluster $X_l = \{X_i\} + \{X_j\}$*:*

$l = 1, 2, \ldots, N_1, N_1+1, N_1+2, \ldots, N_1+N_2$



Background

- *Optional Data Preprocessing (Normalization)*
  - Can not affect relative placement
    - *If point A is left of B, then after pre-processing A must still be to the left of B*
  - Avoids bias caused by dimensions with a large spread

➢Large spread may naturally describe data!

Clustering Feature

*A **Clustering Feature** (CF) summarizes a sub-cluster of data points:*

Given a cluster $\{\vec{X}_1, \vec{X}_2, \ldots, \vec{X}_N\}$

$$\mathbf{CF} = (N, \vec{LS}, SS)$$

N is the number of data points

$$\vec{LS} = \sum_{i=1}^{N} \vec{X}_i$$

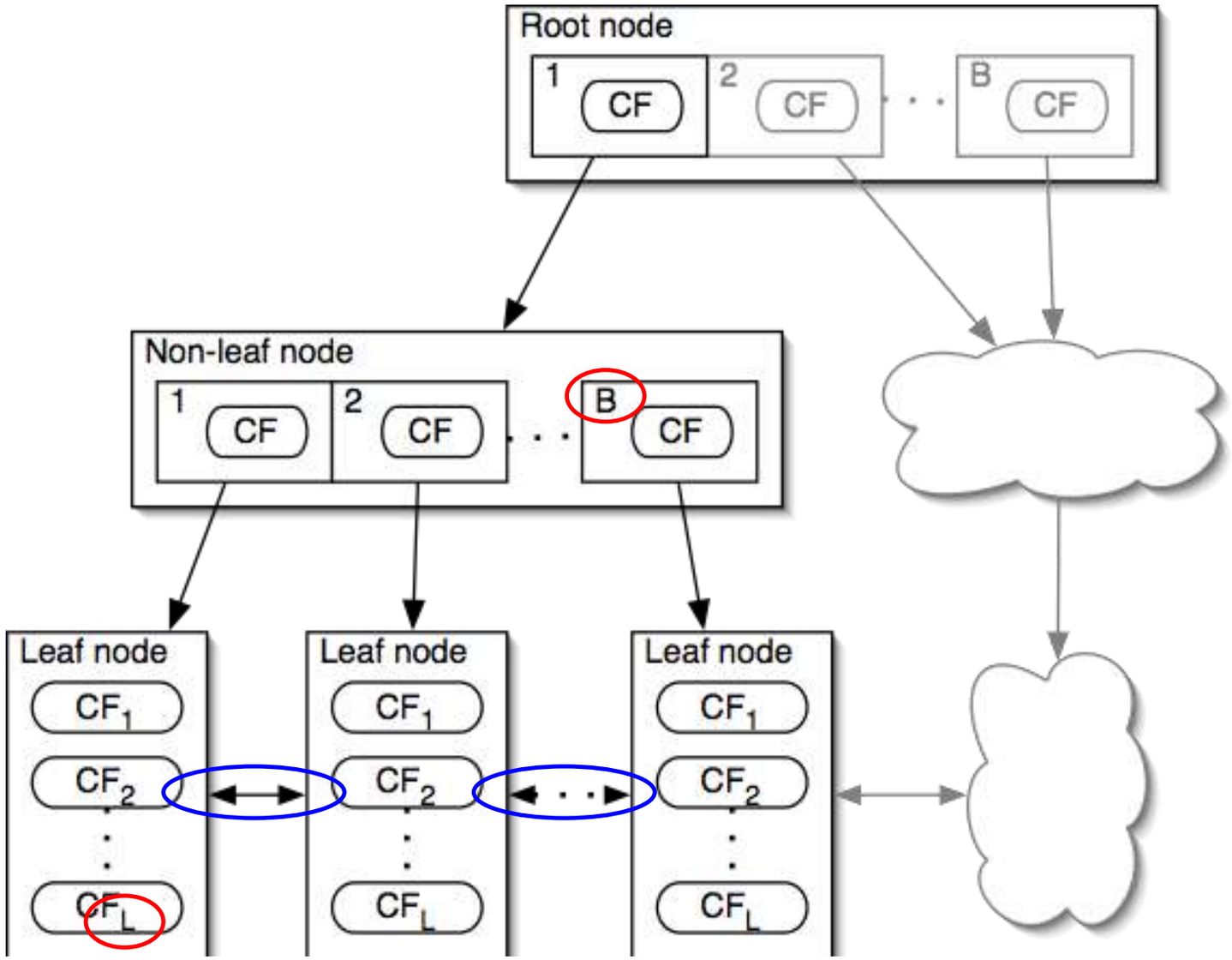
$$SS = \sum_{i=1}^{N} \vec{X}_i^{\,2}$$

$$\mathbf{CF_1 + CF_2} = (N_1 + N_2, \vec{LS}_1 + \vec{LS}_2, SS_1 + SS_2)$$

Properties of Clustering Feature

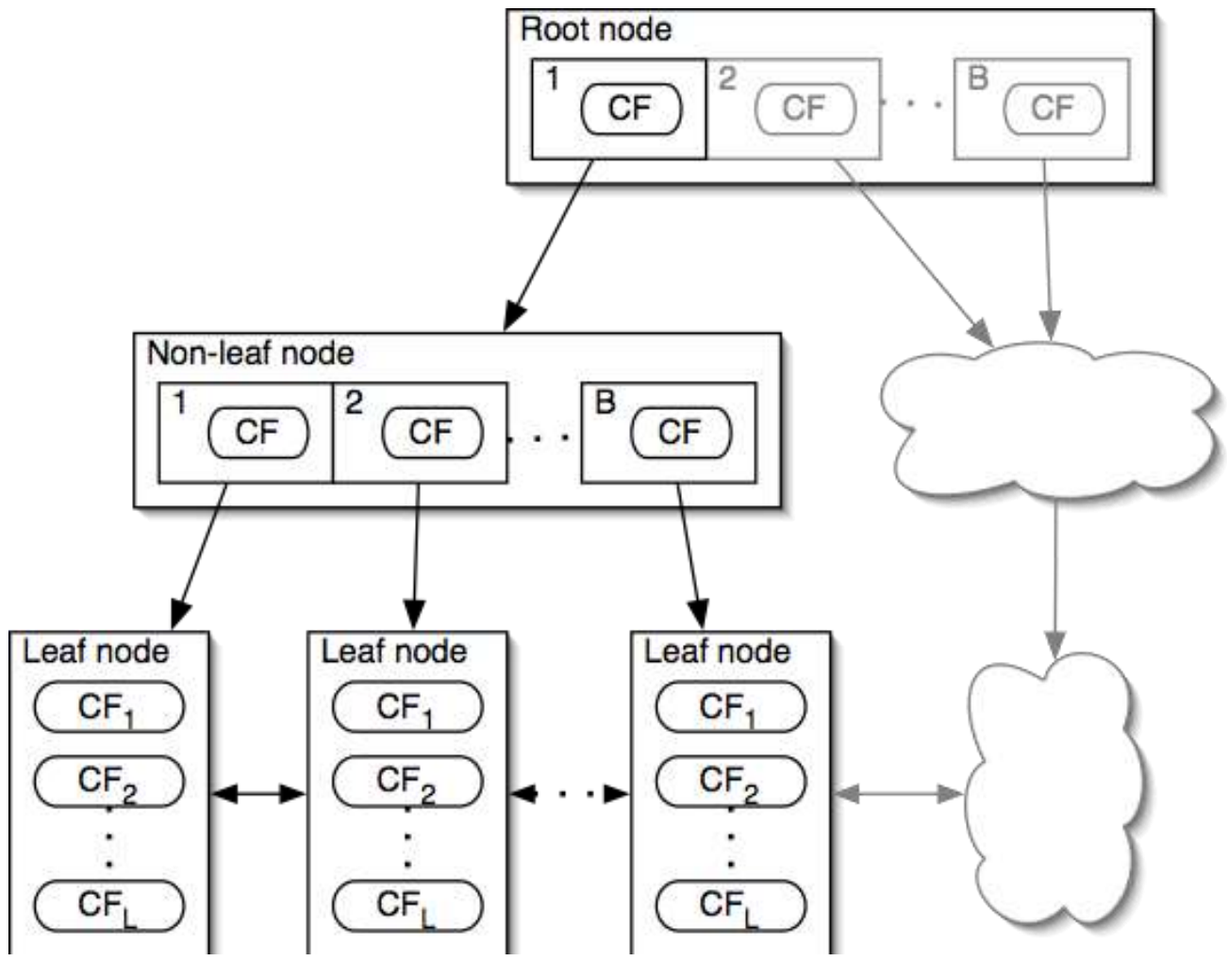
- CF entry is more compact
  - *Stores significantly less then all of the data points in the sub-cluster*
- A CF entry has sufficient information to calculate D*0*-D*4*
- Additivity theorem allows us to merge sub-clusters incrementally & consistently

CF-Tree


Root node
1
CF
2
CF
B
CF
Non-leaf node
1
CF
2
CF
B
CF
Leaf node
CF1
CF2
CFL
Leaf node
CF1
CF2
CFL
Leaf node
CF1
CF2
CFL

Properties of CF-Tree

- Each non-leaf node has at most B entries
- Each leaf node has at most L CF entries which each satisfy threshold T
- Node size is determined by dimensionality of data space and input parameter P (page size)



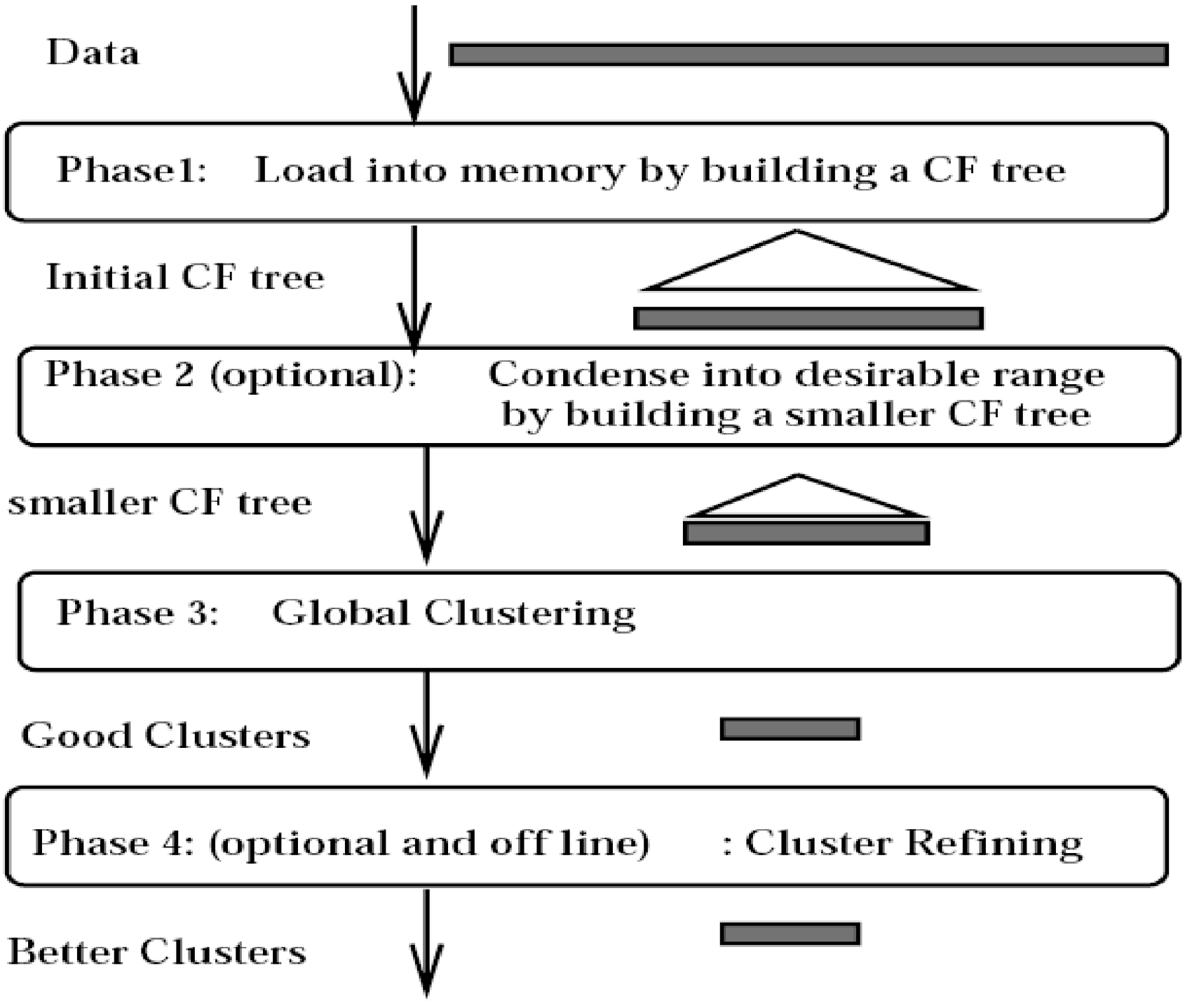
CF-Tree Insertion

- Recurse down from root, find the appropriate leaf
  - *Follow the* "closest"-CF *path, w.r.t.* D*0 / ... /* D*4*
- Modify the leaf
  - *If the* closest-CF *leaf cannot absorb, make a new CF entry. If there is no room for new leaf, split the parent node*
- Traverse back & up
  - *Updating CFs on the path or splitting nodes*

CF-Tree Rebuilding

- If we run out of space, increase threshold T
  - *By increasing the threshold, CFs absorb more data*
  - Goal: Keep CF tree in main memory
- Rebuilding "pushes" CFs over
  - *The larger* T *allows different CFs to group together*
- Reducibility theorem
  - *Increasing* T *will result in a CF-tree as small or smaller then the original*
  - *Rebuilding needs at most* “h” *extra pages of memory*

BIRCH Overview



The Algorithm: BIRCH

- Phase 1: Load data into memory
  - *Build an initial in-memory CF-tree with the data (one scan)*
  - *Subsequent phases become* fast, accurate, less order sensitive
- Phase 2: Condense data
  - *Rebuild the CF-tree with a larger* T
  - *Condensing is optional*

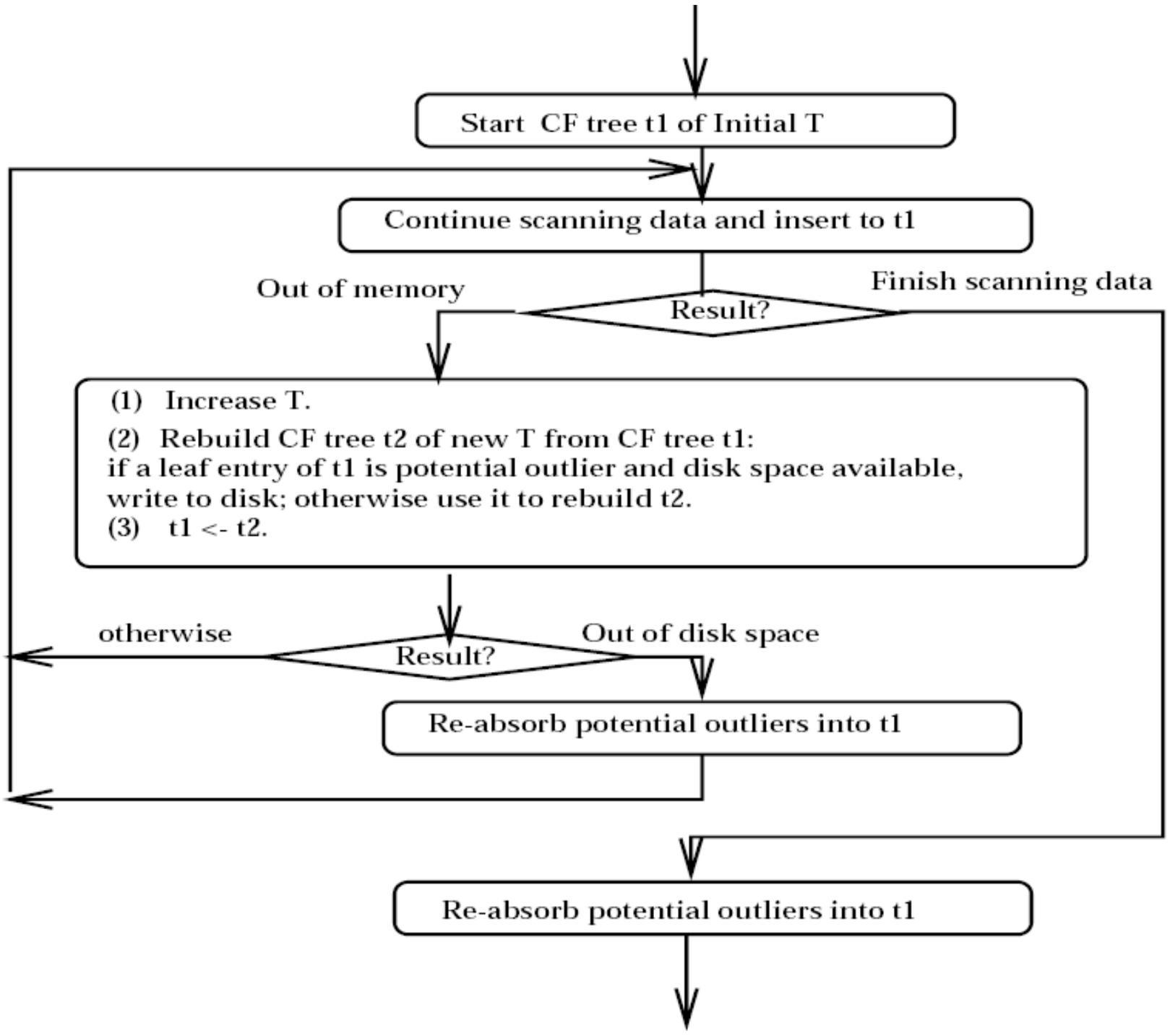
The Algorithm: BIRCH

- Phase 3: Global clustering
  - *Use existing clustering algorithm on CF entries*
  - *Helps fix problem where natural clusters span nodes*
- Phase 4: Cluster refining
  - *Do additional passes over the dataset & reassign data points to the closest centroid from phase 3*
  - *Refining is optional*

The Algorithm: BIRCH

- Why have optional phases?
  - *Phase 2 allows us to resize the data set so Phase 3 runs on an optimally sized data set*
  - *Phase 4 fixes a problem with CF-trees where some data points may be assigned to different leaf entries*
    - ➢ *Phase 4 will always converge to a minimum*
    - ➢ *Phase 4 allows us to discard outliers*

The Algorithm: BIRCH



The Algorithm: BIRCH

- Rebuild CF-tree with smallest T
  - *Start with* T *= 0 and try rebuilding the tree*

- Get rid of outliers
  - *Write outliers to special place outside of the tree*

- Delayed split
  - *Treat data points that force a split like outliers*

Experimental Results

- Input parameters:
  - *Memory (*M*): 5% of data set*
  - *Disk space (*R*): 20% of* M
  - *Distance equation:* D2
  - *Quality equation: weighted average diameter (*D*)*
  - *Initial threshold (*T*): 0.0*
  - *Page size (*P*): 1024 bytes*

Experimental Results

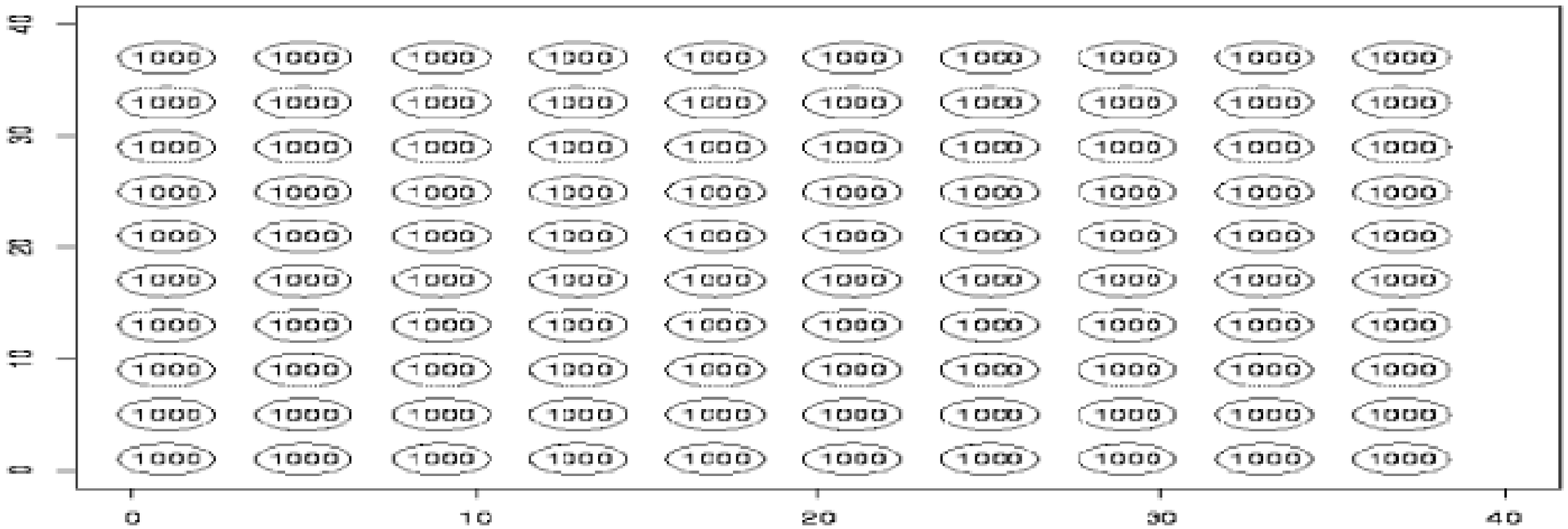
- The Phase 3 algorithm
  - *An agglomerative Hierarchical Clustering (HC) algorithm*
- One refinement pass
  - *Outlier discarding is off*
- Delay-split is on
  - *This is what we use disk space* R *for*

Experimental Results

- Create 3 synthetic data sets for testing
  - *Also create an ordered copy for testing input order*
- KMEANS and CLARANS require entire data set to be in memory
  - *Initial scan is from disk, subsequent scans are in memory*

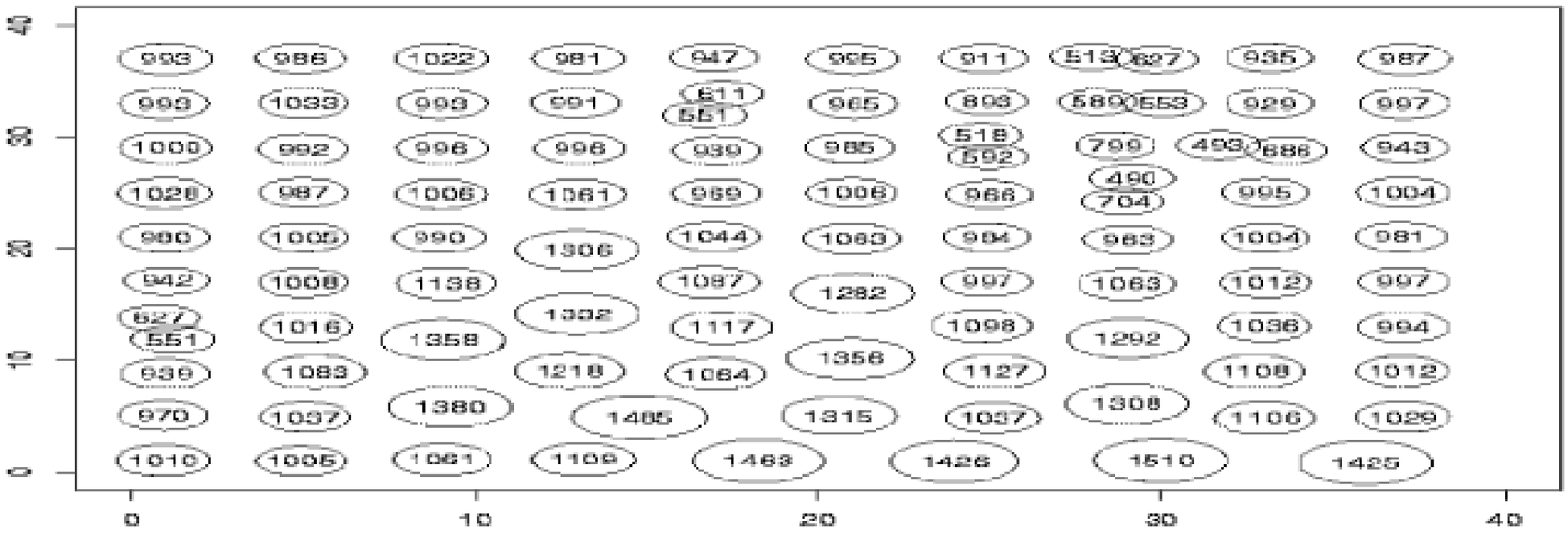
Experimental Results

Intended clustering



Experimental Results

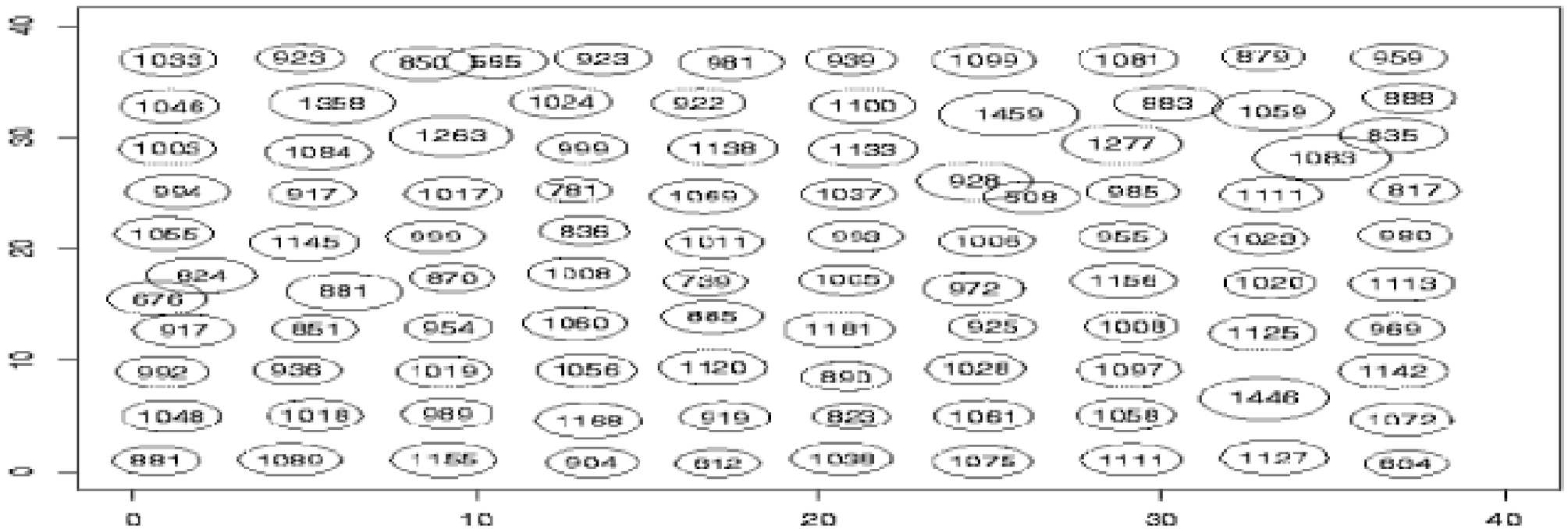
KMEANS clustering



DS	*Time*	*D*	*# Scan*	*DS*	*Time*	*D*	*# Scan*
1	**43.9**	**2.09**	**289**	1o	33.8	1.97	197
2	13.2	4.43	51	2o	12.7	4.20	29
3	32.9	3.66	187	3o	36.0	4.35	241

Experimental Results

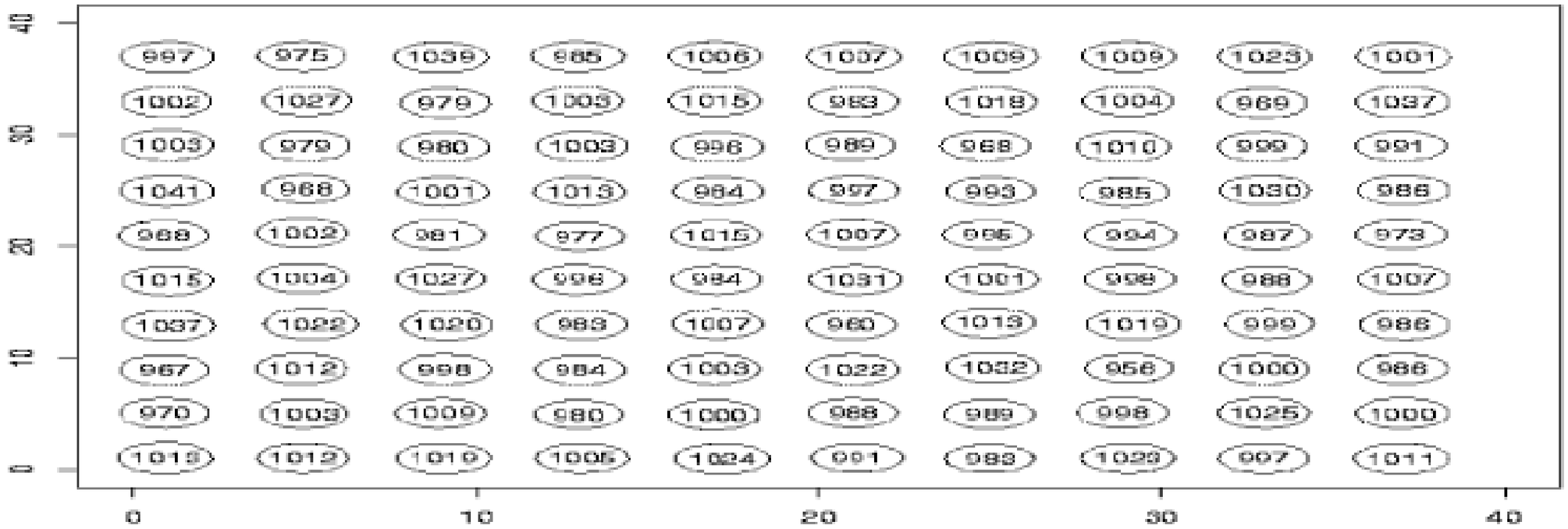
CLARANS clustering



DS	*Time*	*D*	*# Scan*	*DS*	*Time*	*D*	*# Scan*
1	**932**	**2.10**	**3307**	1o	794	2.11	2854
2	758	2.63	2661	2o	816	2.31	2933
3	835	3.39	2959	3o	924	3.28	3369

Experimental Results

BIRCH clustering



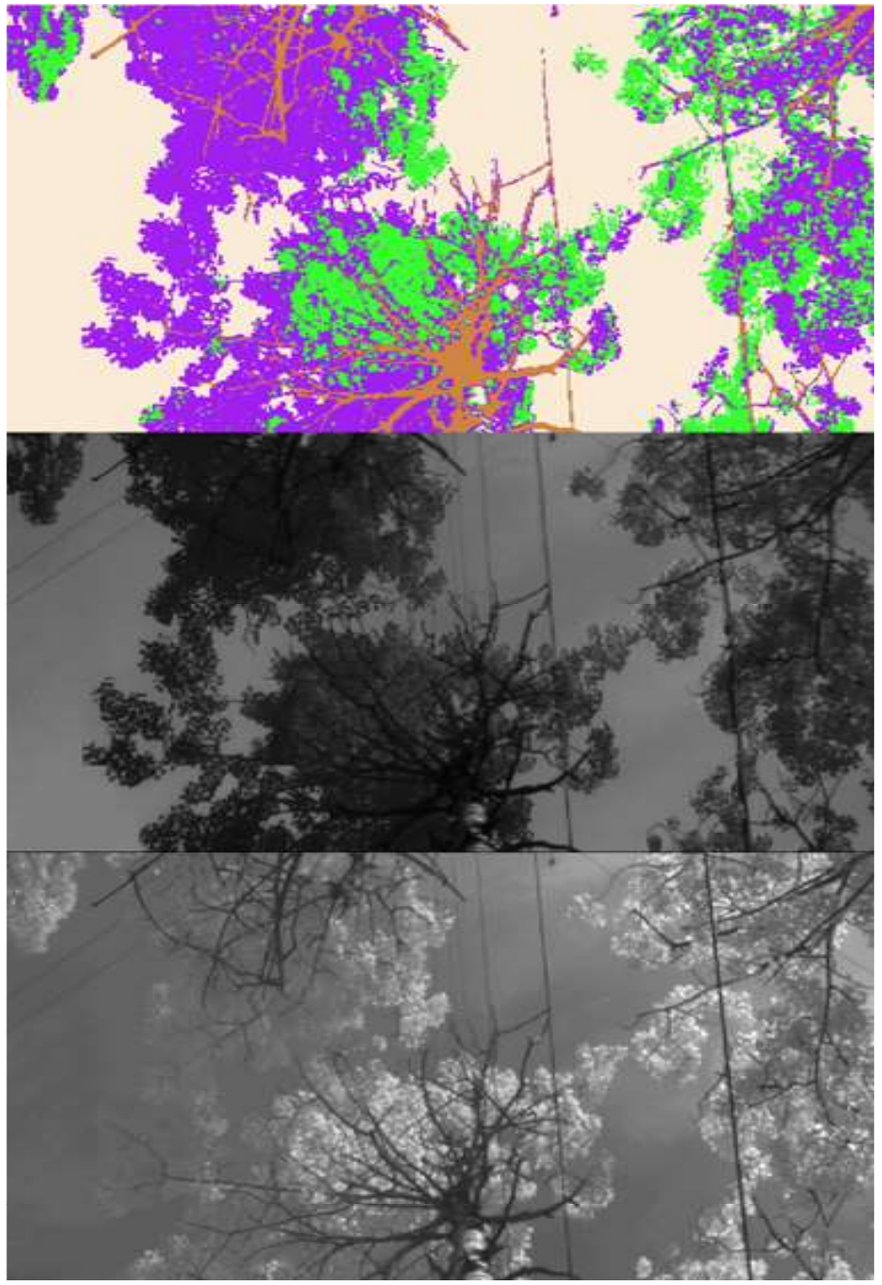
DS	*Time*	*D*	*# Scan*	*DS*	*Time*	*D*	*# Scan*
1	**11.5**	**1.87**	**2**	1o	13.6	1.87	2
2	10.7	1.99	2	2o	12.1	1.99	2
3	11.4	3.95	2	3o	12.2	3.99	2

Experimental Results

- Page size
  - *When using Phase 4,* P *can vary from 256 to 4096 without much effect on the final results*
- Memory vs. Time
  - *Results generated with low memory can be compensated for by multiple iterations of Phase 4*
- Scalability

Conclusions & Practical Use

- Pixel classification in images
- From top to bottom:
  - *BIRCH classification*
  - *Visible wavelength band*
  - *Near-infrared band*



Conclusions & Practical Use

- Image compression using vector quantization
  - *Generate codebook for frequently occurring patterns*
  - *BIRCH performs faster then CLARANS or LBG, while getting better compression and nearly as good quality*

Conclusions & Practical Use

- BIRCH works with very large data sets
- Explicitly bounded by computational resources
  - *Runs with specified amount of memory (*P*)*
- Superior to CLARANS and KMEANS
  - *Quality, speed, stability and scalability*

CHAMELEON: A Hierarchical Clustering Algorithm Using Dynamic Modeling

Paper written by
George Karypis, Eui-Hong (Sam) Han, Vipin Kumar,
and
appeared in IEEE Computer, 1999

Presenter: P.Krishna Reddy

Clustering

- Clustering in data mining is a discovery process that groups a set of data such that the intra-cluster similarity is maximized and the inter-cluster similarity is minimized. Existing clustering algorithms, such as *K*-means, PAM,
- CLARANS, DBSCAN, CURE, and ROCK are designed to find clusters that fit some static models.
- Can breakdown
  - if the choice of parameters in the static model is incorrect with respect to the data set being clustered
  - if the model is not adequate to capture the characteristics of clusters.
  - when the data consists of clusters that are of diverse shapes, densities, and sizes.

Existing algorithms

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

BIRCH (1996)

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

Clustering Feature Vector

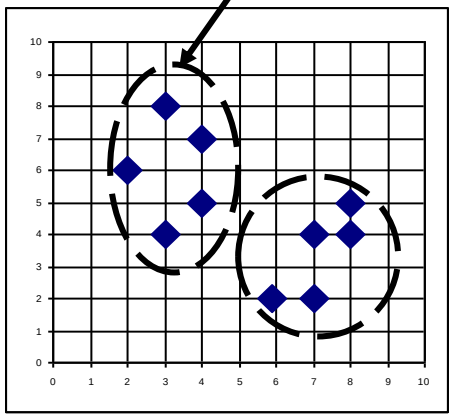
Clustering Feature: $CF = (N, \overrightarrow{LS}, SS)$

N: **Number of data points**

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

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

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

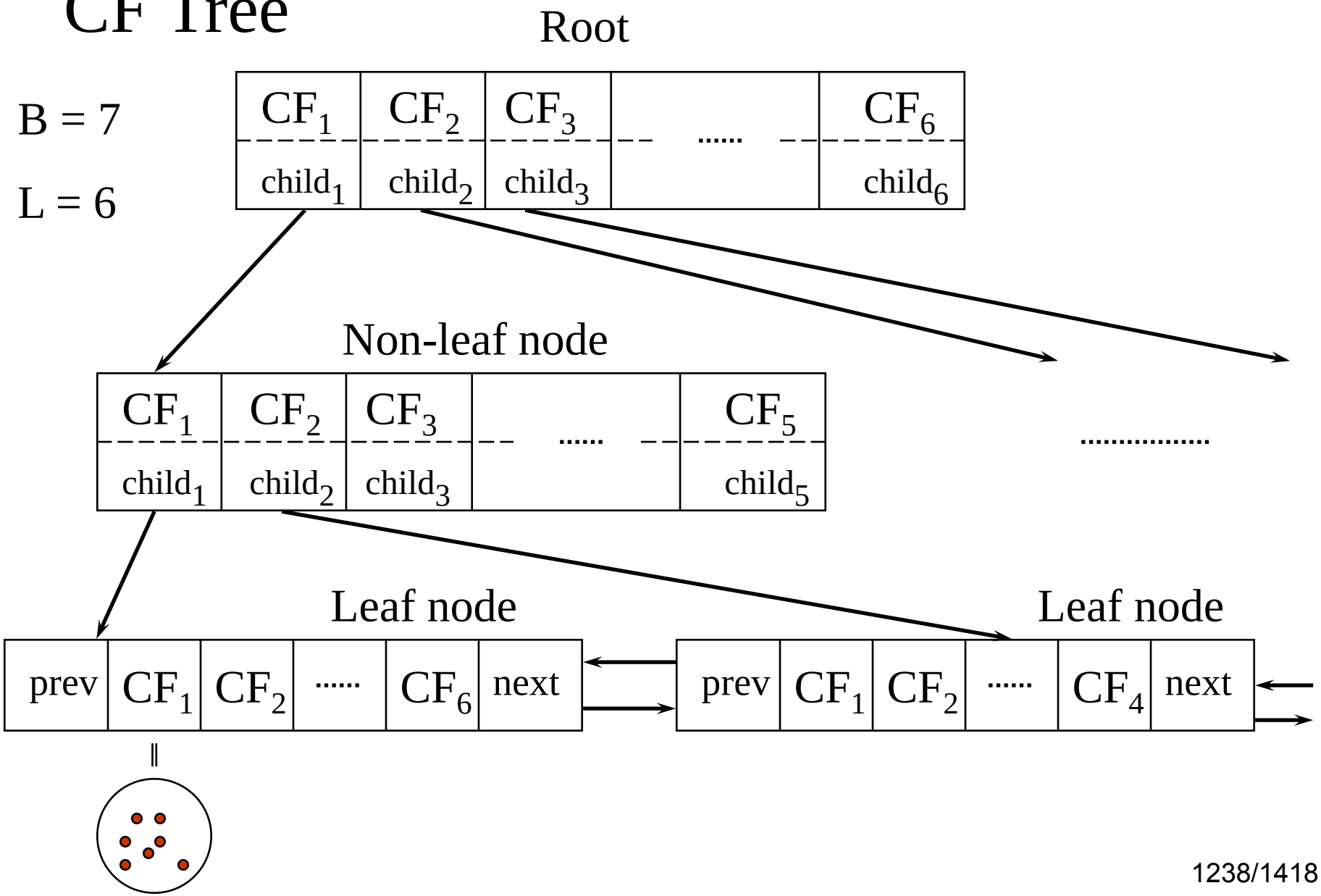


(3,4)
(2,6)
(4,5)
(4,7)
(3,8)

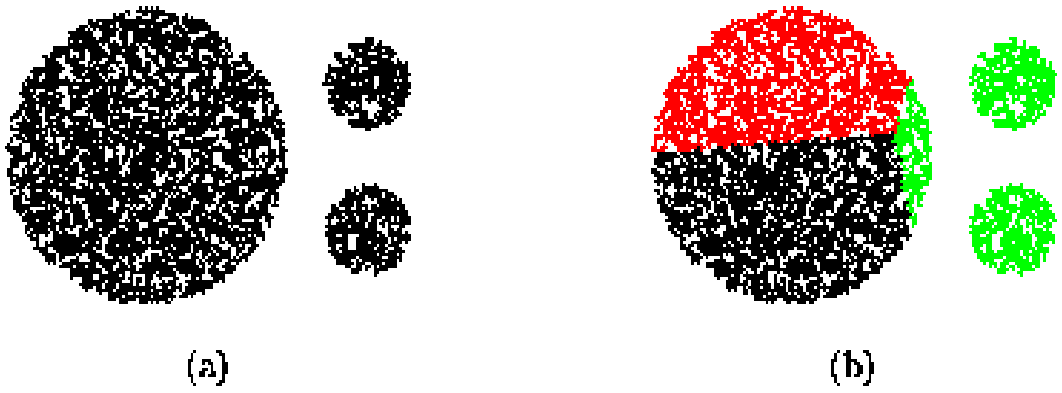
CF Tree

B = 7

L = 6


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

CURE (Clustering Using REpresentatives)



- CURE: proposed by Guha, Rastogi & Shim, 1998
  - Stops the creation of a cluster hierarchy if a level consists of *k* clusters
  - Uses multiple representative points to evaluate the distance between clusters, adjusts well to arbitrary shaped clusters and avoids single-link effect

Drawbacks of Distance-Based Method

- Drawbacks of square-error based clustering method
  - Consider only one point as representative of a cluster
  - Good only for convex shaped, similar size and density, and if *k* can be reasonably estimated
- In CURE
  - the distance between two sub-clusters is the smallest pair-wise distance between their representative points.
  - Hierarchical clustering works by merging the closest pair of sub-clusters.
  - Once the clusters are merged, a new set of representative are computed for the merged cluster.
- Complexity is O(n)

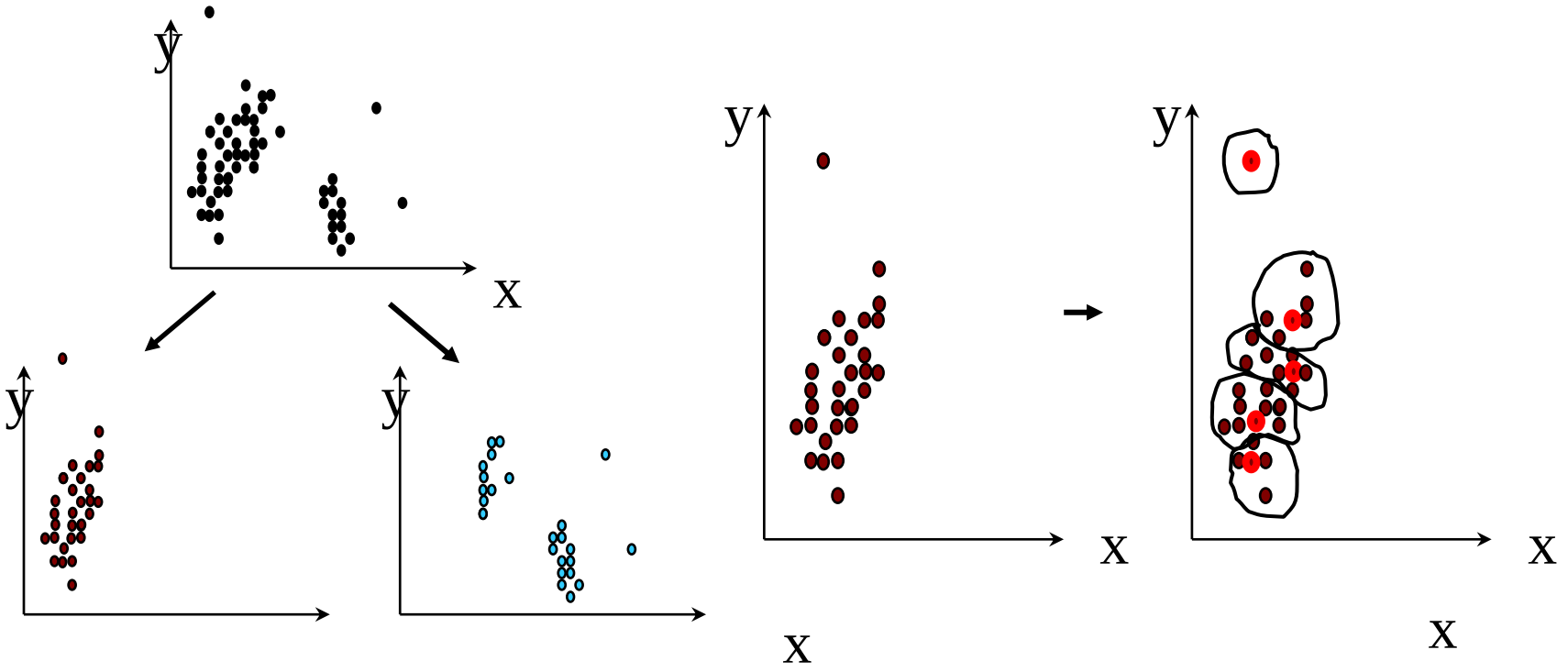
Cure: The Algorithm

- Draw random sample *s*.
- Partition sample to *p* partitions with size *s/p*
- Partially cluster partitions into *s/pq* clusters
- Eliminate outliers
  - By random sampling
  - If a cluster grows too slow, eliminate it.
- Cluster partial clusters.
- Label data in disk

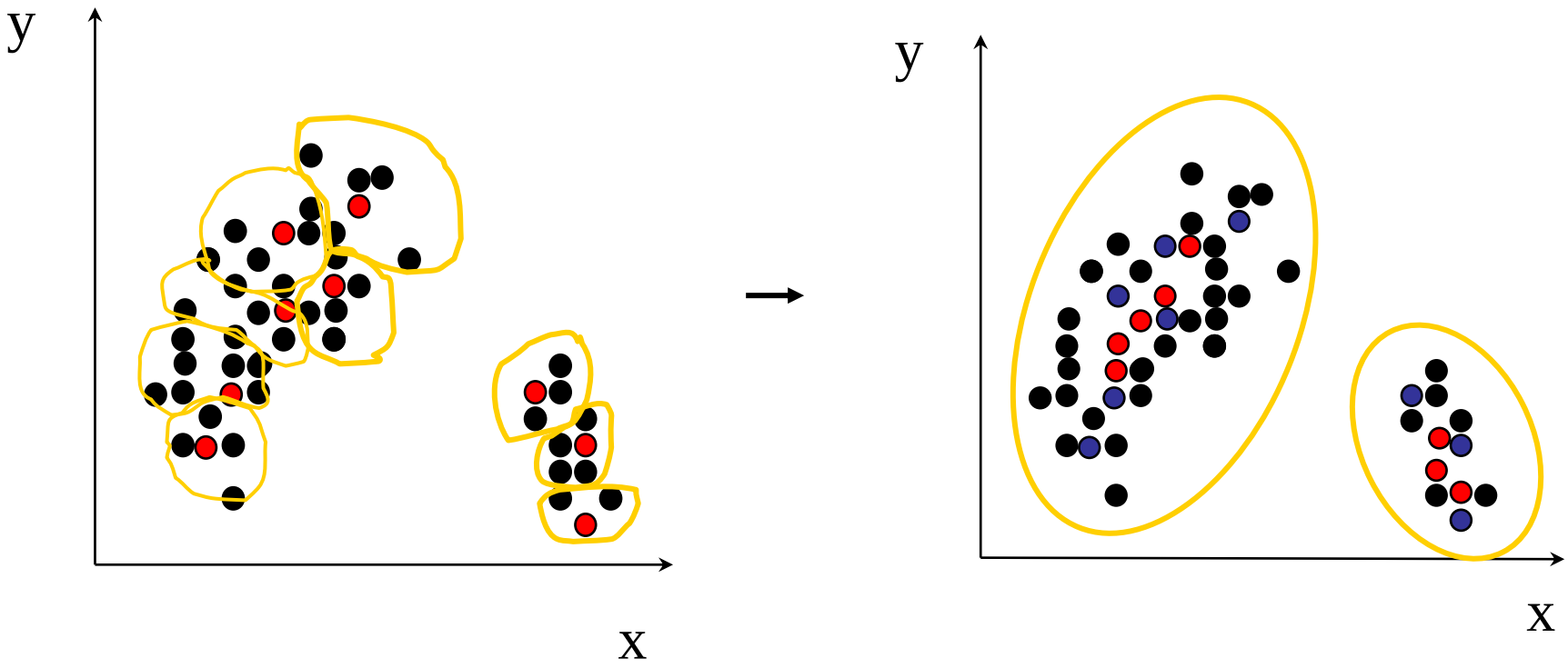
Data Partitioning and Clustering

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

- s/pq = 5



Cure: Shrinking Representative Points



- Shrink the multiple representative points towards the gravity center by a fraction of α.
- Multiple representatives capture the shape of the cluster

Clustering Categorical Data: ROCK

- ROCK: Robust Clustering using links,
  by S. Guha, R. Rastogi, K. Shim (ICDE'99).
  - Use links to measure similarity/proximity
  - Not distance based
  - Computational complexity: $O(n^2 + nm_m m_a + n^2 \log n)$
- Basic ideas:
  - Similarity function and neighbors:

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

$$Sim(T_1, T_2) = \frac{|T_1 \cap T_2|}{|T_1 \cup T_2|}$$

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

Rock: Algorithm

- Neighbors: An objects neighbors are those objects that are considerably similar to it.
- Links: The number of common neighbors for the two points.

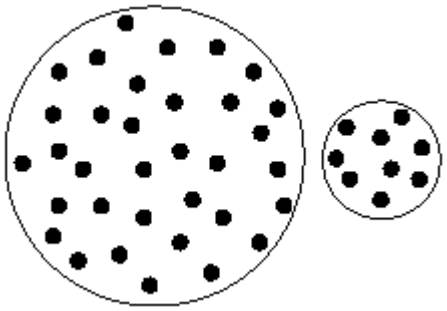
{1,2,3}, {1,2,4}, {1,2,5}, {1,3,4}, {1,3,5}
{1,4,5}, {2,3,4}, {2,3,5}, {2,4,5}, {3,4,5}

- Algorithm $\{1,2,3\} \xleftrightarrow{3} \{1,2,4\}$
  - Draw random sample
  - Cluster with links
  - Label data in disk

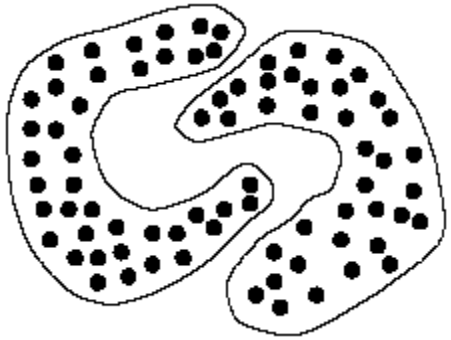
Drawbacks of Partitioning Techniques.

- Centroid-based approaches, as typified by K means or Medoid-based methods, as typified by PAM or CLARENS
- Major drawback
  - they fail for data in which points in a given cluster are closer to the center of another cluster than to the center of their own cluster.

- for example,
  - if there is a large variation in cluster sizes.
  - or when cluster shapes are convex.



a) Clusters of widely differente sizes

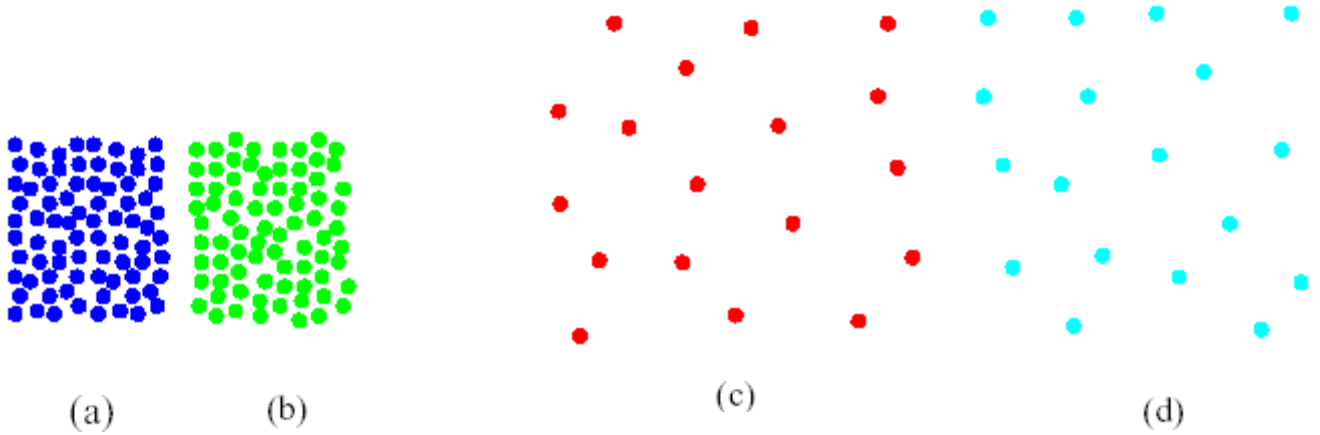


b) Clusters with convex shapes

Drawbacks of Hierarchical Techniques

- Agglomerative hierarchical algorithms
- CURE : a constant number of representative points are chosen to represent a cluster.
  - work implicitly with the *n***n* similarity matrix
  - merging decisions are based upon static modeling of the clusters to be merged.
  - fail to take into account special characteristics of individual clusters
  - make incorrect merging decisions when the underlying data does not follow the assumed model and when noise is present.
  - In schemes based upon group averaging (eg. ROCK) key limitation is assume a static, user supplied inter-connectivity model, which is inflexible.

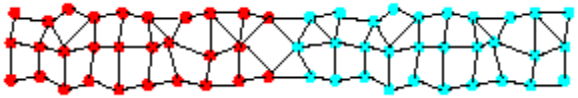
- model under- or over-estimates the inter-connectivity of the data when different clusters exhibit different inter-connectivity characteristics.



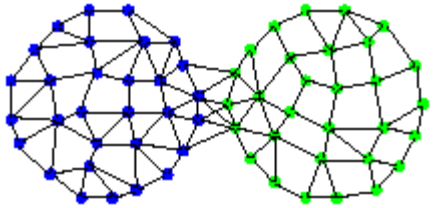
- CURE merges clusters (a) and (b) over merging clusters (c) and (d). But clusters (c) and (d) are better candidates for merging.

- the minimum distances between the boundary points of (c) and (d) are of the same order as the average of the minimum distances of any points within these clusters to other points.

- Ex. 2 (For ROCK):-



(a) (b)



(c) (d)

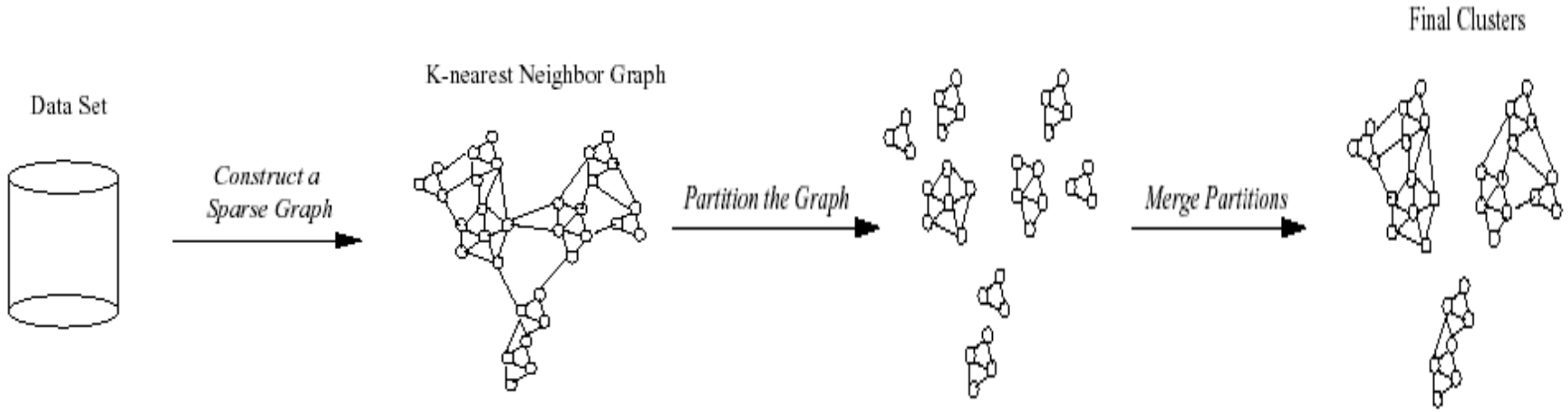
ROCK selection mechanism (irrespective of the assumed model of connectivity) and the group averaging method will select pair {(c),(d)} for merging,whereas the pair{(a),(b)} is a better choice.

CHAMELEON : An Intro

- hierarchical clustering algorithm
- measures the similarity of two clusters based on a dynamic model.
- operates on a sparse graph in which nodes represent data items, and weighted edges represent similarities among the data items.
- uses a two phase algorithm.

- During the first phase, CHAMELEON uses a ***graph partitioning algorithm*** to cluster the data items into a large number of relatively small sub-clusters.
- During the second phase, it uses an ***agglomerative hierarchical clustering algorithm*** to find the genuine clusters by repeatedly combining together these sub-clusters.
- The purpose of the first phase is to cluster the data items into a ***large number of sub-clusters*** that contain a sufficient number of items to allow dynamic modeling.
- The purpose of the second phase, is to ***discover the genuine clusters*** in the dataset by using the ***dynamic modeling framework*** to merge together these sub-clusters in a hierarchical fashion.

- The key feature of CHAMELEON's agglomerative hierarchical clustering algorithm is that it determines the pair of most similar sub-clusters by taking into account both the ***inter-connectivity*** as well as the ***closeness*** of the clusters.
- uses a novel approach to model the degree of inter-connectivity and closeness between each pair of clusters
- does not depend on a static user supplied model.



Overall Framework CHAMELEON

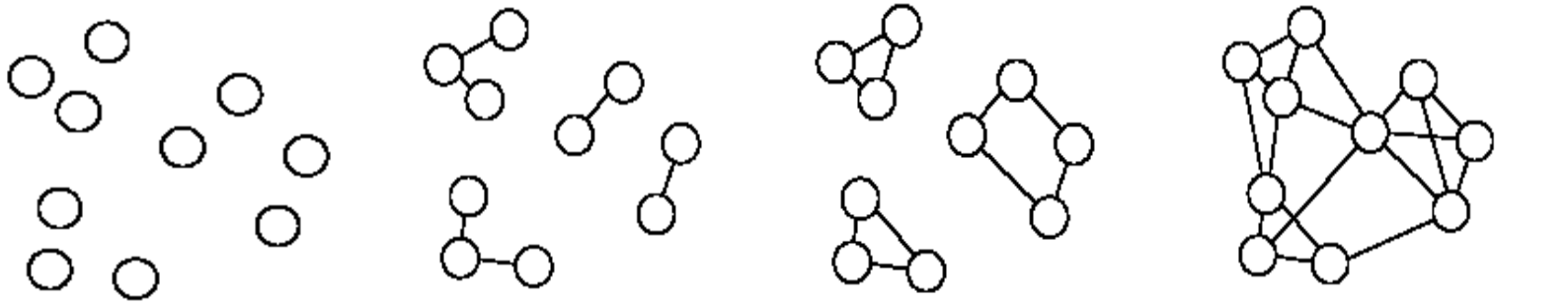
- dynamically model the similarity between the clusters by computing their *relative inter-connectivity* and *relative closeness*.
- graph partitioning is used to obtain the initial fine-grain clustering solution.

Modeling Data

- CHAMELEON's sparse graph representation of the data items is based on the commonly used *k*-nearest neighbor graph approach.
- Each vertex of the *k*-nearest neighbor graph represents a data item, and there exists an edge between two vertices, if data items corresponding to either of the nodes is among the *k*-most similar data points of the data point corresponding to the other node.

Modeling the Data

- K-nearest graphs from an original data in 2D



(a) Original Data in 2D (b) 1-nearest neighbor graph (c) 2-nearest neighbor graph (d) 3-nearest neighbor graph

Advantages of k-nearest neighbour graph

- data points that are far apart are completely disconnected in the *Gk*
- *Gk* captures the concept of neighborhood dynamically. In a dense region, the neighborhood is defined narrowly and in a sparse region, the neighborhood is defined more widely.
- *Gk* captures more natural neighborhood.

- the density of the region is recorded as the weights of the edges. As the consequence, a min-cut bisection of the graph represents the interface layer of sparse region of the graph.
- Finally, *Gk* provides a computational advantage over a full graph in many algorithms operating on graphs, including graph partitioning and partitioning refinement algorithms.

Modeling Cluster Similarity

- **Relative Inter-Connectivity**: defined as the *absolute inter-connectivity* between *Ci* and *Cj* normalized with respect to the internal inter-connectivity of the two clusters *Ci* and *Cj* .
- absolute inter-connectivity between a pair of clusters *Ci* and *Cj* is defined to be as the *sum of the weight of the edges* that connect vertices in *Ci* to vertices in *Cj*.(or Edge Cut)

- Thus the relative inter-connectivity between a pair of clusters *Ci* and *Cj* is given by

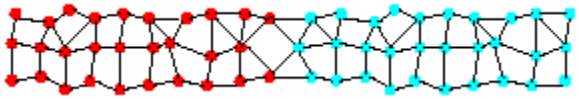
$$RI(C_i, C_j) = \frac{|EC_{\{C_i,C_j\}}|}{\frac{|EC_{C_i}|+|EC_{C_j}|}{2}},$$

which *normalizes the absolute inter-connectivity* with the average internal inter-connectivity of the two clusters.

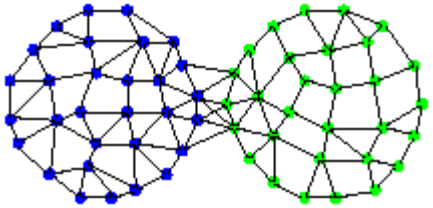
-EC{ci,cj} is the sum of the weight of the edges in edge-cut between two clusters.

-EC{ci} internal inter-connectivity of a cluster Ci. Captured by size of the mincut bisector (i.e., the weighted sum of edges that partition the graph into two roughly equal parts.)

- Thus in Example 2 CHAMELEON correctly merges (a) and (b) over (c) and (d) since relative inter-connectivity between clusters (a) and (b) is higher than the relative inter-connectivity between clusters (c) and (d).



(a) (b)



(c) (d)

- **Relative Closeness**
  - defined as the absolute closeness between *Ci* and *Cj* normalized with respect to the internal closeness of the two clusters *Ci* and *Cj* .
  - CHAMELEON measures the closeness of two clusters by computing the average similarity between the points in *Ci* that are connected to points in *Cj* .
  - their average strength provides a very good measure of the affinity between the data items
  - internal closeness of each cluster *Ci* can also be measured as look at all the edges connecting vertices in *Ci* (*i.e.*, edges that are internal to the cluster), and compute the internal closeness of a cluster as the average weight of these edges.

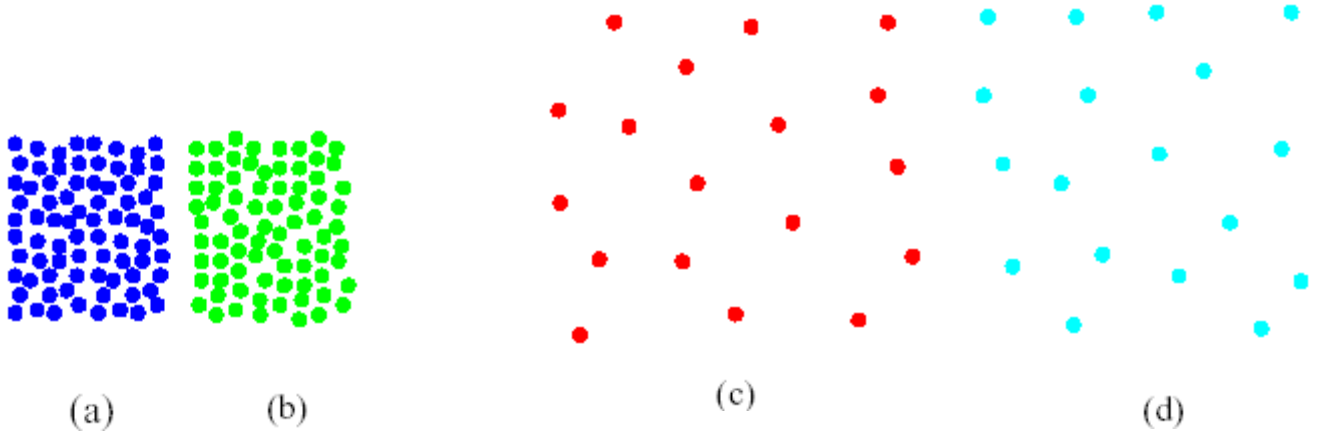
- Hence in CHAMELEON, the relative closeness between a pair of clusters *Ci* and *Cj* is computed as,

$$RC(C_i, C_j) = \frac{\overline{S}_{EC_{\{C_i,C_j\}}}}{\frac{|C_i|}{|C_i|+|C_j|}\overline{S}_{EC_{C_i}} + \frac{|C_j|}{|C_i|+|C_j|}\overline{S}_{EC_{C_j}}},$$

where $\overline{S}_{EC_{Ci}}$ and $\overline{S}_{EC_{Cj}}$ are the average weights of the edges that belong in the min-cut bisector of clusters *Ci* and *Cj* ,respectively.

$\overline{S}_{EC_{\{Ci,Cj\}}}$ is the average weight of the edges that connect vertices in *Ci* to vertices in *Cj* .

- Thus in Example 1 CHAMELEON will merge the clusters (c) and (d) over the clusters (a) and (b).
- Because, the relative closeness of clusters (c) and (d) is higher than the relative closeness between clusters (a) and (b), even though the later pair of clusters have a higher absolute closeness.



The Algorithm

1. CHAMELEON finds the initial sub-clusters using a graph partitioning algorithm to partition the *k*-nearest neighbor graph of the data set into a large number of partitions such that the *edgecut* (*i.e.*, the sum of the weight of the edges that straddle partitions) is minimized.

2) utilizes multilevel graph partitioning algorithms to find the initial sub-clusters
   - initially starts with all the points belonging to the same cluster.
   - then repeatedly selects the largest sub-cluster among the current set of sub-clusters.
   - This process terminates when the larger sub-cluster contains fewer than a specified number of vertices, that is referred to as MINSIZE.

3) CHAMELEON then switches to an agglomerative hierarchical clustering that combines together these small sub-clusters.

- First scheme merges only those pairs of clusters whose relative inter-connectivity and relative closeness are both above some user specified threshold *TRI* and *TRC*, respectively.

- In this approach, CHAMELEON visits each cluster *Ci* ,and checks to see if any one of its adjacent clusters *Cj* satisfy the following two conditions:

$$RI(C_i, C_j) \geq T_{RI} \text{ and } RC(C_i, C_j) \geq T_{RC}.$$

•If more than one of the adjacent clusters satisfy the above conditions, then CHAMELEON selects to merge *Ci* with the cluster that it is most connected to.

•*i.e.*, it selects the cluster *C j* such that the absolute inter-connectivity between these two clusters is the highest.

- Second scheme implemented in CHAMELEON uses a function to combine the relative inter-connectivity and relative closeness, and then selects to merge the pair of clusters that maximizes this function.
  - select the pair of clusters *Ci* and *Cj* to merge that maximize

$$RI(C_i, C_j) * RC(C_i, C_j)^{\alpha},$$

Performance Analysis

- The amount of time required to compute
  - K-nearest neighbor graph
  - Two-phase clustering

Performance Analysis(cont'd)

- The amount of time required to compute
  - K-nearest neighbor graph
    - Low-dimensional data sets = O(n log n)
    - High-dimensional data sets = $O(n^2)$

Performance Analysis(cont'd)

- The amount of time required to compute
  - Two-phase clustering
    - Computing internal inter-connectivity and closeness for each cluster: O(nm)
    - Selecting the most similar pair of cluster: O(n log n + m^2 log m)
  - Total time = O(nm + n log n + m^2 log m)

Remarks

- Does not consider the issue of scaling to large data sets that cannot fit in the main memory
- CHAMELEON can discover natural clusters of different shapes and sizes
- It is possible to use other algorithms instead of k-nearest neighbor graph
- Different domains may require different models for capturing closeness and inter-connectivity

Conclusions

- CHAMELEON discovers natural clusters of different shapes and sizes.

- Experimental results show that Chameleon can discover natural clusters that many existing algorithms fail to find.
  - CURE
  - ROCK
  - DBSCAN

Concepts of Deep learning

P. Krishna Reddy, IIIT Hyderabad

Outline

- What is deep learning?
- Backpropagation algorithm
- Key challenges for training deep learning models
- Overview of deep learning architecture

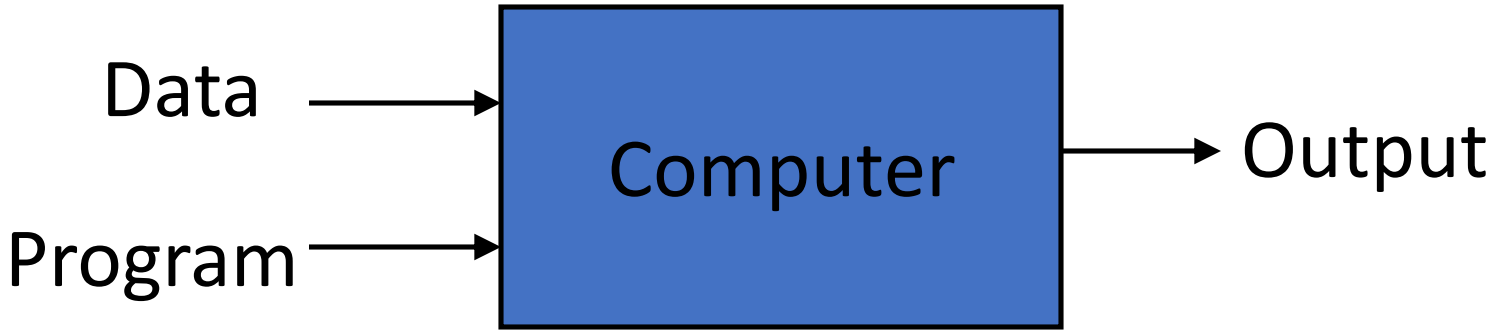
A simple question

- 1, 3, 5, 7, 9, …. What is the next number?
  - Ans: 11. ; Odd numbers, Or 2n+1
- 1, 3, 9, 19, 33, ... What is the next number?
  - Ans: 51; $2n^2+1$
- How do we solve such problems?
  - Find a pattern from the examples.
    - (function f(n) = 2n+1. Or model the data)
  - Use it to predict the next number (or solve the problem)
- How do we design a computational procedure?

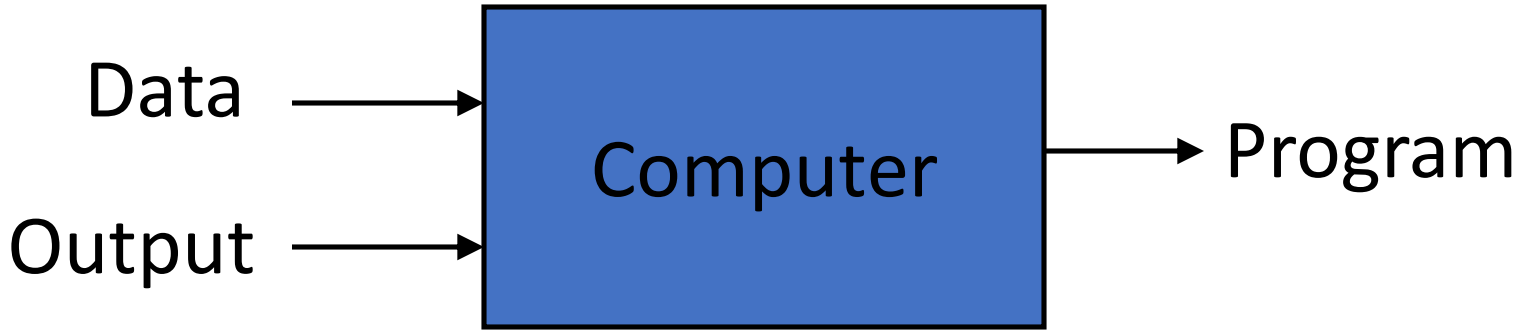
A simple question (cont.)

- We know: 1, 3, 9, 19, 33, ... What is the next number?
  - Ans: 51; $2n^2+1$
- 0.99, 3.02, 9.00, 18.98, 33.01, ... What next?
- Consider a series of 2D points
  - (1,3), (2,6), (3,9), (4,12), ...
  - What is the next point?
- When does the problem become difficult?
  - When numbers are “uncertain”. Noise in measurements
  - When numbers are not just “simple numbers”?

Traditional Programming



Machine Learning



The machine learning framework

- Apply a prediction function to a feature representation of the "sample" to get the desired output:

f() = "apple"

f() = "tomato"

f() = "cow"

Slide credit: L. Lazebnik

The machine learning/Classification framework

$$y = f(\mathbf{x})$$



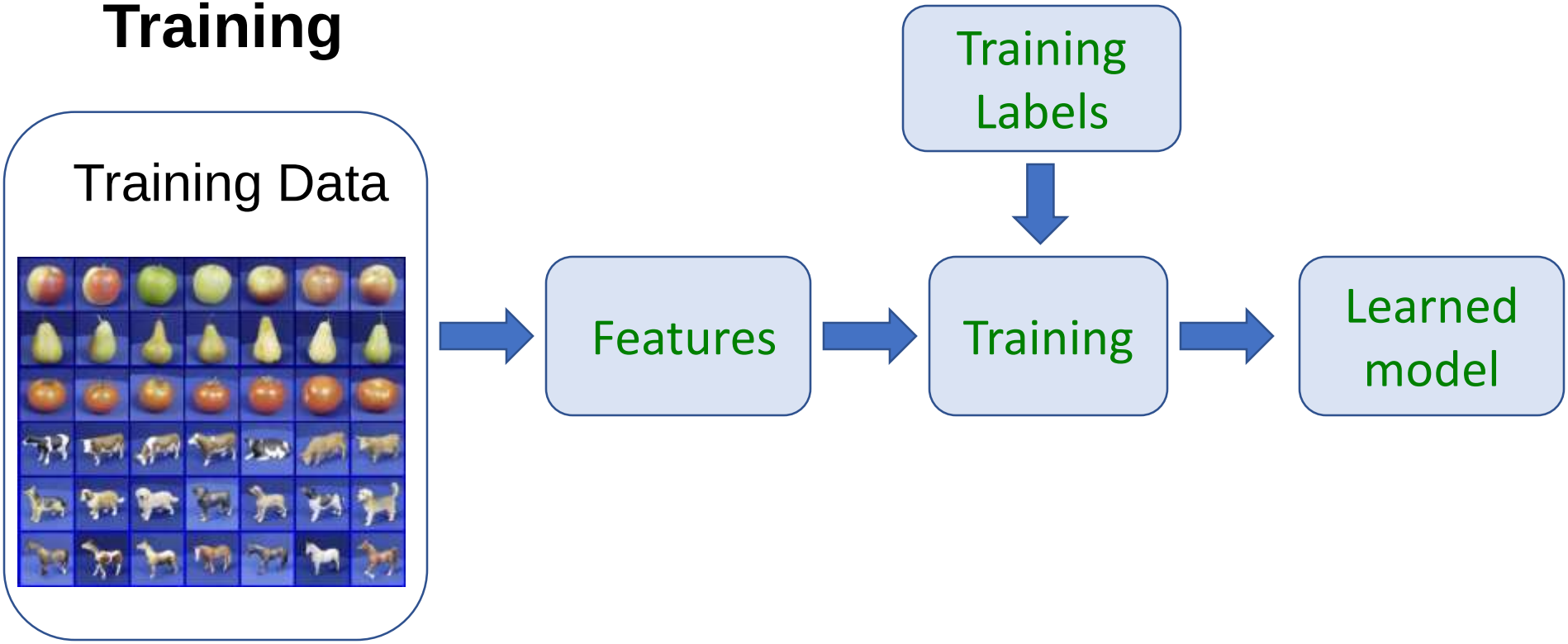
- **Training:** given a *training set* of labeled examples $\{(\mathbf{x}_1,y_1), \ldots, (\mathbf{x}_N,y_N)\}$, estimate the prediction function f by minimizing the prediction error.

- **Testing:** apply f to a never before seen *test example* $\mathbf{x}$ and output the predicted value $y = f(\mathbf{x})$

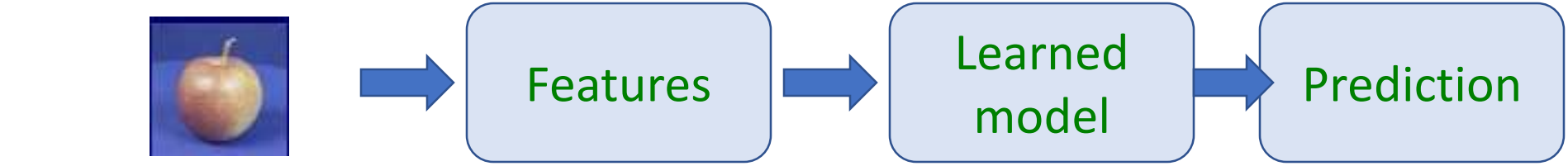
Slide credit: L. Lazebnik

Steps

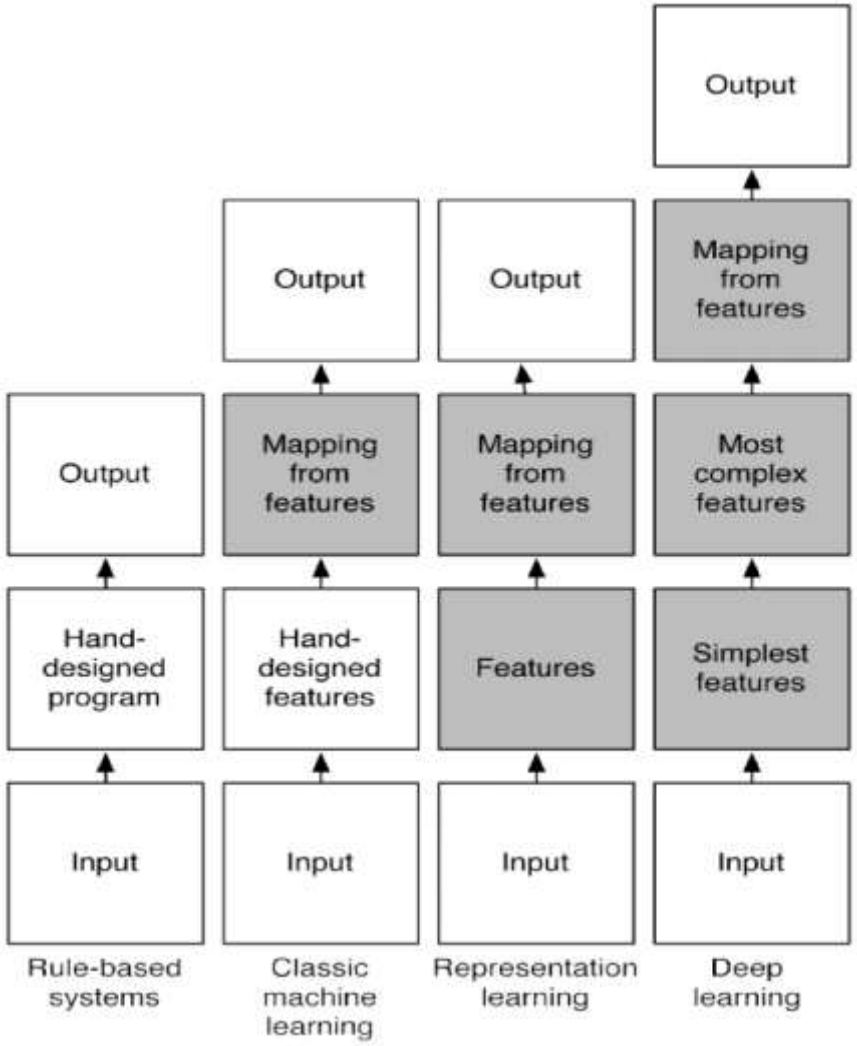
Training



Testing



What is deep learning?



Neural Networks

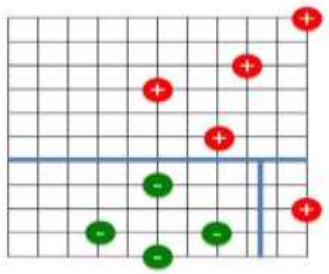
Y. Bengio et al, ``Deep Learning", MIT Press, 2015

What is deep learning?

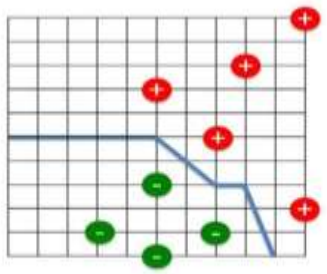
- Deep learning is based on artificial neural networks
- A neural network: A set of connected input/output units where each connection has a **weight** associated with it
- During the learning phase, the **network learns by adjusting the weights** so as to be able to predict the correct class label of the input tuples.
- What is a unit?
  - It is like a perceptron or logistic regression classifier

About Linear classifiers

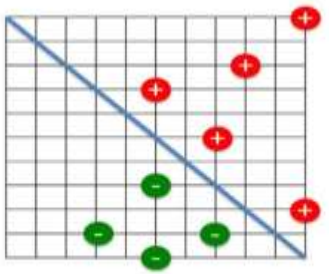
- Classifies generate complex decision boundaries.
- Decision tree might generate hyperrectangular shape boundary
- 1-nearest neighbour classifier may generate hyper polygonal boundary
- Linear classifier may general linear boundary
  - Better generalization performance, better interpretability
- We discuss three classifiers: Linear regression, perceptron, logistic regression (widely used)



(a) Decision Tree Classifier



(b) 1−Nearest-Neighbor Classifier



(c) Linear Classifier

Linear Regression

- Linear regression is a statistical technique that predicts a continuous value based on one or more independent attributes.
  - For example, we might want to predict the housing price based on the living
  - Since linear regression aims to predict a *continuous* value, it cannot be directly applied to the classification task, where the output is a categorical variable
- Suppose we have *n* tuples, each of which is represented by *p* attributes

$$x_i = (x_{i,1}, \ldots, x_{i,p})^T$$

- A continuous output value *yi* (*i* = 1,...,*n)*. In linear regression, we want to learn a linear function that maps the *p* input attributes *xi* to the output variable *yi*, that is,

$$\hat{y}_i = w^T x_i + b = \sum_{j=1}^{p} w_j x_{i,j} + b,$$

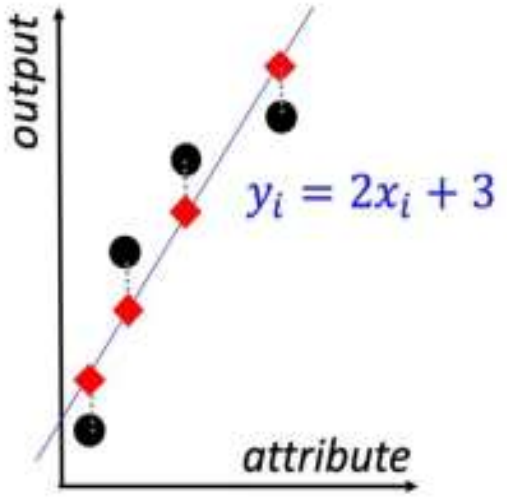
where *y^i* is the predicted output value for the *i*th tuple, *w* = *(w*1*, ...,wp)T* is a *p*-dimensional weight vector and *b* is the bias scalar.

- Algorithm: Least square regression, which aims to minimize the loss function. Therefore the best weight vector *w* and the bias scalar *b* are the ones that minimize the loss function *L(w, b)*

$$L(w,b) = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 = \sum_{i=1}^{n} (y_i - (w^T x_i + b))^2$$

Index (i)	1	2	3	4
Attribute (x_i)	1	3	5	7
Output (y_i)	4	10	14	16

(a) Training tuples



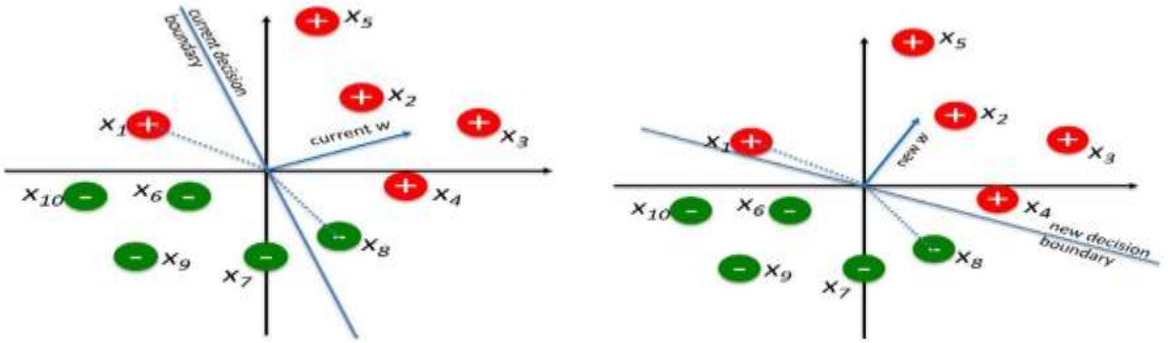
(b) Least square regression

FIGURE 6.11

An example of least square regression. (a) Four training tuples. (b) Scatter-plot of the training tuples (black dots) and least square regression model (the blue line). Red diamonds are the predicted output $\hat{y}_i$ $(i = 1, 2, 3, 4)$ and dashed lines indicate the prediction errors ($|y_i - \hat{y}_i|$) of the corresponding training tuples.

Perceptron

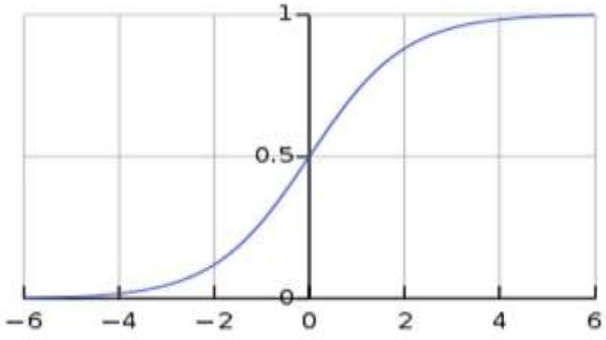
- Consider a binary classification task
- The output value *yi* for a given tuple is a binary variable: *yi* = +1 indicates the *i*th tuple is a positive tuple (e.g., *buy computer*) and *yi* = 0 indicates the *i*th tuple is a negative one (e.g., *not buy computer*).
- Output of regression function is the sign of regression function
- Sign=+1 if yi is the predicted class label, sign=0, otherwise
- If we know the weight vector W, we can predict the class label.
- W is interactively learned from the training set.
- If the training tuples are linearly separable, the perceptron algorithm is guaranteed to find a weight vector (i.e., a hyperplane decision boundary)



(a) current weight vector w (b) updated weight vector w

Logistic Regression

- Perceptron predicts the binary class label of a given tuple. However, can we also tell how confident such a prediction is?
- Logistic regression estimates the probability of an event occurring, such as voted or didn't vote, based on a given dataset of independent variables.
- To convert the output from, 0 to 1 sigmoid function is introduced.
  - Output of linear regression model is mapped to a number between 0 to 1
- To determine optimal W vector, **maximum likelihood estimation** method is employed.
- It aims to solve the following optimization problem: choosing the the best weight vector *w* that maximizes the likelihood of the training set.



Unit of Neural Network

- Both perceptron and logistic regression can be viewed **as a unit** in a neural network
- A unit is a mathematical function
  - Takes linear weighted sum of input and
  - Passes the sum through an activation function.
  - An activation function is a non-linear function
    - Sign function in perceptron or sigmoid function in logistic regression
    - Several activation functions exist.

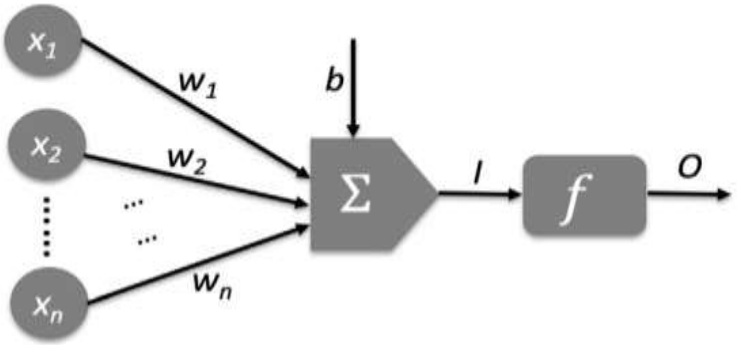


FIGURE 10.2

An illustrative example of unit. Given the inputs $X_1, \ldots, X_n$, a unit takes a linear weighted sum and then passes the sum through an activation function $f(\cdot)$. $I = \sum_{i=1}^{n} X_i w_i + b$ is the net input of the activation function and $O = f(I)$ is the output of the unit. w_i $(i = 1, \ldots, n)$ are weights and b is the bias scalar.

About Neural Network

- A neural network is a collection of interconnected units
- Type of network depends on the type of interconnection
- Popular one
  - Multi-Layer Feed-Forward Neural Network
- Consist of
  - Input layer
  - One or more hidden layers
  - An output layer
- Each layer is made-up of units

Multi-Layer Feed-Forward Neural Network

- Each layer is made up of units. The inputs to the network correspond to the attributes measured for each training tuple.
- The inputs are fed simultaneously into the units making up the **input layer**.
- These inputs pass through the input layer and are then weighted and fed simultaneously to a second layer of "neuronlike" units, known as a **hidden layer**.
- The outputs of the hidden layer units can be input to another hidden layer, and so on.
- The number of hidden layers could be arbitrary (1 hidden layer in this example), outputs of the last hidden layer are input to units making up the **output layer**, which emits the network's predictions for given tuples
- Feed forward network
  - None of the output connects to the input

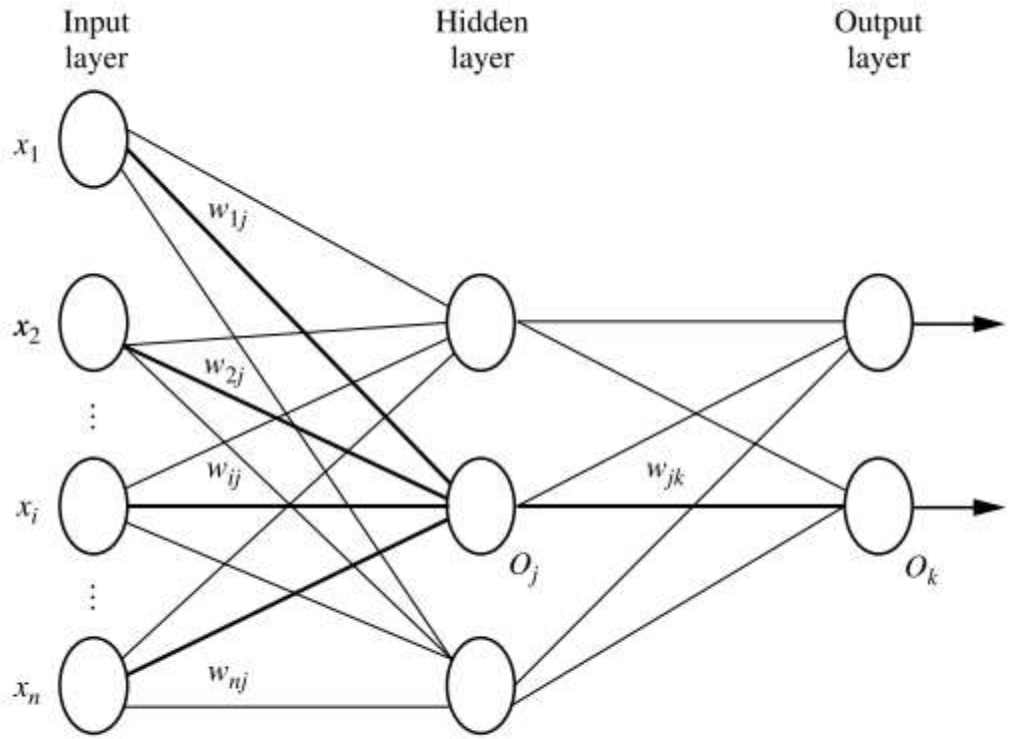
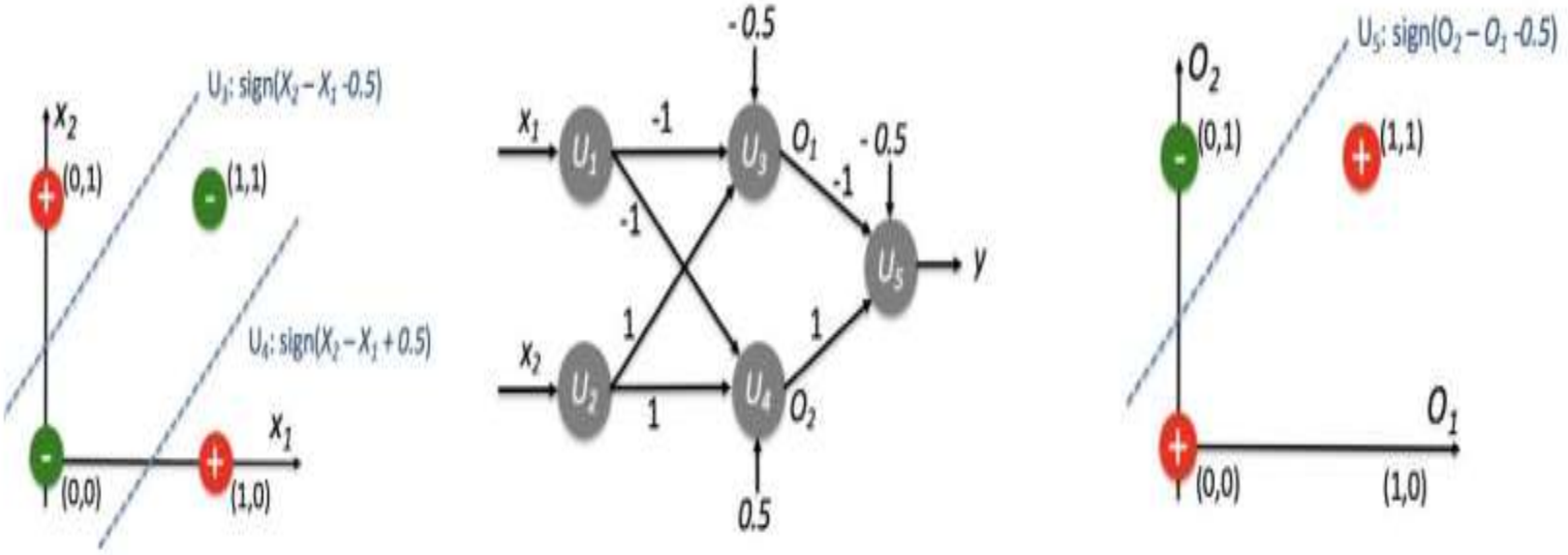


FIGURE 10.1

Multilayer feed-forward neural network.

- **It applies a nonlinear activation function to the weighted sum of the inputs.**
- If we know the weights connecting to each layer and the activation function of each unit, we can feed any data tuple into the input layer of the neural network and calculate its output from the output layer, which can be used for classification, clustering, and so on.
- Multilayer feed-forward neural networks are able to model the class prediction as a nonlinear combination of the inputs.
- **From a statistical point of view, they perform nonlinear regression.**
- **Multilayer feed-forward networks with sufficient depth and width, given enough hidden units and enough training samples, can closely approximate any function.**
- In a multilayer feed-forward network, the output of a hidden unit is used as the input to units in the next layer.
- This gives neural network an incredible ability to learn more complicated, often semantically more meaningful features from the simpler ones.
- From left to right, the outputs of units at each layer of a multilayer feed-forward neural networks form a hierarchy of learned features at increasingly more complex levels

- Compared logistic regression and Support Vector Machines (SVMs), the following are advantages
  - being able to learn *any* nonlinear function that maps the input data tuple to the output (e.g., the class label) and
  - the automatic feature learning
- Consider XOR problem
- In Fig. 10.3(a), there are four training tuples, each of which is represented by two attributes $\boldsymbol{X}1$ and $\boldsymbol{X}2$. Two positive training tuples are at $(0,1)$ and $(1,0)$, respectively, and two negative training tuples are at $(0,0)$ and $(1,1)$, respectively.
- This training set is linearly inseparable, meaning that there is no linear classifier (e.g., perceptron) that is able to separate the two positive tuples from the two negative ones.
- However, we can use two perceptrons (the two dashed lines in Fig. 10.3(a)) to separate the positive tuples from the negative tuples.
- The first perceptron is in the form of $O1 = \text{sign}(\boldsymbol{X}2 - \boldsymbol{X}1 - 0.5)$. The second perceptron is in the form of $O2 = \text{sign}(\boldsymbol{X}2 - \boldsymbol{X}1 + 0.5)$.
- By combining these two perceptrons (each of which is a linear classifier) together, we have a nonlinear classifier that perfectly separates the two positive tuples from the two negative tuples; that is, if $\boldsymbol{X}2 - \boldsymbol{X}1 - 0.5 \geq 0$ or $\boldsymbol{X}2 - \boldsymbol{X}1 + 0.5 < 0$, predict it as a positive tuple; otherwise, predict it as a negative tuple.



(a) Input 4 tuples (b) MLP (c) Outputs of two hidden units

FIGURE 10.3

Solving XOR problem with a two-layer feed-forward neural network with *sign* as the activation function, which is also called multilayer perception (MLP). Note that in (c), both negative tuples share the same representation (O_1, O_2), both being at (0, 1).

- In this example, the third perceptron at unit $U5$ takes $O1$ and $O2$ (i.e., the outputs of unit $U3$ and $U4$), as its input features.
- In other words, both $O1$ and $O2$ can be viewed as the *learned* features by the two hidden units.
- $O1$ represents whether or not the input tuple is above ($O1 = 1$) or below ($O1 = 0$) of the decision boundary of the first perceptron (the upper dashed line in Fig. 10.3(a)).
- Likewise, $O2$ represents whether or not the input tuple is above ($O2 = 1$) or below ($O2 = 0$) of the decision boundary of the second perceptron (the lower dashed line in Fig. 10.3(a)).
- Compared with the original input attributes ($\boldsymbol{X}1$ and $\boldsymbol{X}2$), these two learned features are semantically more meaningful. As we can see from Fig. 10.3(c), the tuples with these two newly learned features are now linearly separable.
- In contrast, in the original feature space (Fig. 10.3(a)), these training tuples cannot be separated from each other by any linear classifier.

Classification by Backpropagation

- Backpropagation: A **neural network** learning algorithm
- Started by psychologists and neurobiologists to develop and test computational analogues of neurons
- A neural network: A set of connected input/output units where 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 the input tuples
- Also referred to as **connectionist learning** due to the connections between units

About Deep Neural Networks

- The ability of neural networks to automatically learn features naturally motivates to use neural networks with many hidden layers, namely, **deep neural networks**.
- Deep neural networks have an incredible capability to learn and represent features at different abstraction levels (e.g., one level at each hidden layer), where the more complicated features are learned based on the simpler ones.
- Example 1: In computer vision, we can build a type of deep neural network called convolutional neural networks to recognize different objects (e.g., car, person, animal) from the input image.
  - The input layer of the neural network contains the raw pixels (e.g., one pixel for each input unit), which represent the feature at the lowest semantic level.
  - The output (i.e., the learned feature from the raw pixels at the input layer) of the first hidden layer might correspond to edges of the input image.
  - The output (i.e., the learned feature from the edges of the first hidden layer) of the second hidden layer might correspond to contours or corners of the input image.
  - The output (i.e., the learned feature from the edges and contours of the second hidden layer) of the third hidden layer might correspond to parts of the objects (e.g., nose, car wheel) of the input image.
  - Finally, the output layer can easily learn a classifier to recognize different objects based on the object parts at the output layer

Example 2: Text Mining

- For text mining, we can build a type of deep neural network called recurrent neural networks to classify text documents into different categories.
  - The first layer might take the raw characters as input and output the tokens (e.g., words, punctuation);
  - the second layer might take the tokens as input and output phrases;
  - and the final output layer might take the phrases as input and output category that the input document belongs to.
- It is very difficult to directly train a classifier (e.g., logistic regression for object recognition or document categorization) based on the raw features of the input data (e.g., the raw pixels of images, the characters of documents).
- However, a deep neural network decomposes **this task into a sequence of interdependent subtasks** (one at each hidden or output layer), each of which learns a semantically more meaningful feature from the feature at the previous layer.

Algorithm framework for deep learning

- Backpropagation algorithm

How A Multi-Layer Neural Network Works

- The **inputs** to the network correspond to the attributes measured for each training tuple
- Inputs are fed simultaneously into the units making up the **input layer**
- They are then weighted and fed simultaneously to a **hidden layer**
- The number of hidden layers is arbitrary, although usually only one
- The weighted outputs of the last hidden layer are input to units making up the **output layer**, which emits the network's prediction
- The network is **feed-forward**: None of the weights cycles back to an input unit or to an output unit of a previous layer
- From a statistical point of view, networks perform **nonlinear regression**: Given enough hidden units and enough training samples, they can closely approximate any function

Defining a Network Topology

- Decide the **network topology:** Specify # of units in the *input layer*, # of *hidden layers* (if > 1), # of units in *each hidden layer*, and # of units in the *output layer*
- Normalize the input values for each attribute measured in the training tuples to [0.0—1.0]
- One **input** unit per domain value
- **Output**, if for classification and more than two classes, one output unit per class is used
- Once a network has been trained and its accuracy is **unacceptable**, repeat the training process with a *different network topology* or a *different set of initial weights*

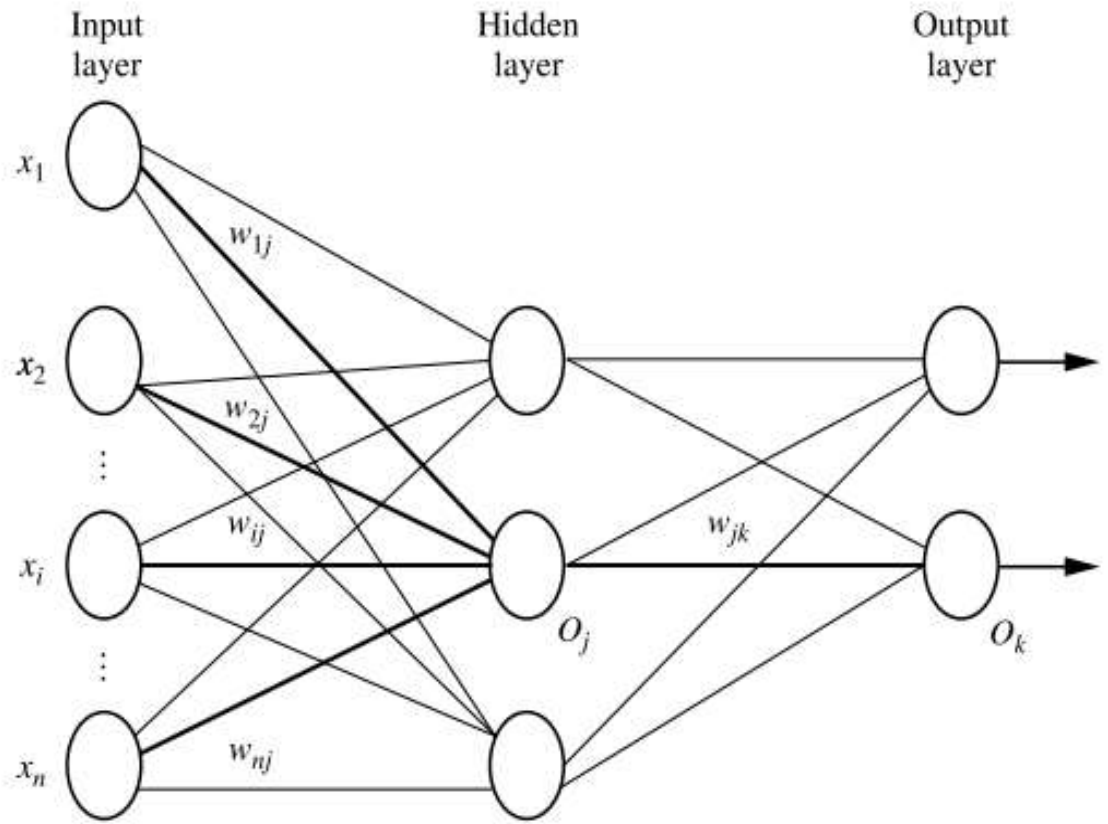


FIGURE 10.1

Multilayer feed-forward neural network.

Backpropagation

- Iteratively process a set of training tuples & compare the network's prediction with the actual known target value
- For each training tuple, the weights are modified to **minimize the mean squared error** between the network's prediction and the actual target value
- Modifications are made in the "**backwards**" direction: from the output layer, through each hidden layer down to the first hidden layer, hence "**backpropagation**"
- Steps
  - Initialize weights to small random numbers, associated with biases
  - For each training tuple X
    - Propagate the inputs forward (by applying activation function)
    - Backpropagate the error (by updating weights and biases)
    - Terminating condition (when error is very small, etc.)

Initialize the weights

- The network weights are initialized to small random numbers
- Each unit has a bias associated with it. The biases are similarly initialized to small numbers.

Propagate the inputs forward

- For each tuple
- The inputs pass through the input units, unchanged. That is, for an input unit, *j*, its output, *Oj*, is equal to its input value, *Ij*.
- Next, the net input to a unit in the hidden or output layers is computed as a linear weighted sum of its inputs.
- Each such unit has a number of inputs to it that are, in fact, the outputs of the units connected to it in the previous layer. Each connection has a weight.
- To compute the net input to the unit, each input connected to the unit is multiplied by its corresponding weight, and this is summed. Given a unit, *j* in a hidden or output layer, the net input, *Ij*, to unit *j* is

$$I_j = \sum_i w_{ij} O_i + b_j, \qquad (10.1)$$

wij is the weight of the connection from i to j.

Oi is the output of the unit i from the previous layer. bj is the bias

Activation function

- Each unit in the hidden and output layer takes its net input and then applies an **activation function** to it.

$$O_j = \frac{1}{1 + e^{-I_j}}.$$

- It maps large input value onto the range of 0 to 1.
  - This nonlinear function.

- We compute the output values, *Oj*, for each hidden layer, up to and including the output layer, which gives the network's prediction.

- In practice, it is a good idea to cache (i.e., save) the intermediate output values at each unit as they are required again later when backpropagating the error. This trick can substantially reduce the amount of computation required.

Backpropagation error

- The error is propagated back by updating the weights and biases to reflect the error of the network prediction. For a unit *j* in the output layer
, the error δ_j is computed by

$$\delta_j = O_j(1 - O_j)(O_j - T_j), \qquad (10.3)$$

- Tj is the known target value of the given tuple.
- To compute the error of hidden layer unit j, the weighted sum of the errors of the units connected to unit j in the next layer are considered. The error of a hidden layer unit j is

$$\delta_j = O_j(1 - O_j)\left(\sum_l \delta_l w_{jl}\right), \qquad (10.4)$$

Updating weights

The weights and biases are updated to reflect the propagated errors. Weights are updated by the following equations, where Δw_{ij} is the change in weight w_{ij}.

$$\Delta w_{ij} = \eta \delta_j O_i. \quad (10.5)$$

$$w_{ij} = w_{ij} - \Delta w_{ij}. \quad (10.6)$$

- Here, n is the learning rate. It is a constant typically having a value between 0.0 and 1.0.
- If the learning rate is low, learning will occur at slow pace.
- If the learning rate is high, inadequate solutions may occur.
- Rule of thumb= 1/t, where t is the number iterations through the training set.

Terminating condition: Training stops when

- All Δw_{ij} in the previous epoch are so small as to be below some specified threshold, or
- The percentage of tuples misclassified in the previous epoch is below some threshold, or
- A prespecified number of epochs has expired.

Algorithm: Backpropagation. Neural network learning for classification or numeric prediction, using the backpropagation algorithm.

Input:

- D, a data set consisting of the training tuples and their associated target values;
- η, the learning rate;
- *network*, a multilayer feed-forward network.

Output: A trained neural network (i.e., the weights w_{ij} and bias b_j for each hidden or output unit).

Method:

(1) Initialize all weights and biases in *network*;
(2) **while** terminating condition is not satisfied {
(3) **for** each training tuple $\boldsymbol{X}$ with target output T in D {
(4) // Propagate the inputs forward:
(5) **for** each input layer unit j {
(6) $O_j = I_j$; // output of an input unit is its actual input value
(7) **for** each hidden or output layer unit j {
(8) $I_j = \sum_i w_{ij} O_i + b_j$; // compute the net input of unit j with respect to the units of the previous layer, i
(9) $O_j = \frac{1}{1+e^{-I_j}}$; } // compute the output of each unit j (sigmoid activation)
(10) // Backpropagate the errors:
(11) **for** each unit j in the output layer
(12) $\delta_j = O_j(1 - O_j)(T_j - O_j)$; // compute the error (mean-squared loss)
(13) **for** each unit j in the hidden layers, from the last to the first hidden layer
(14) $\delta_j = O_j(1 - O_j)(\sum_l \delta_l w_{jl})$; // compute the error with respect to the next higher layer, l
(15) **for** each weight w_{ij} in *network* {
(16) $\Delta w_{ij} = \eta \delta_j O_i$; // weight increment
(17) $w_{ij} = w_{ij} - \Delta w_{ij}$; } // weight update
(18) **for** each bias b_j in *network* {
(19) $\Delta b_j = \eta \delta_j$; // bias increment
(20) $b_j = b_j - \Delta b_j$; } // bias update
(21) } }

FIGURE 10.4

Backpropagation algorithm.

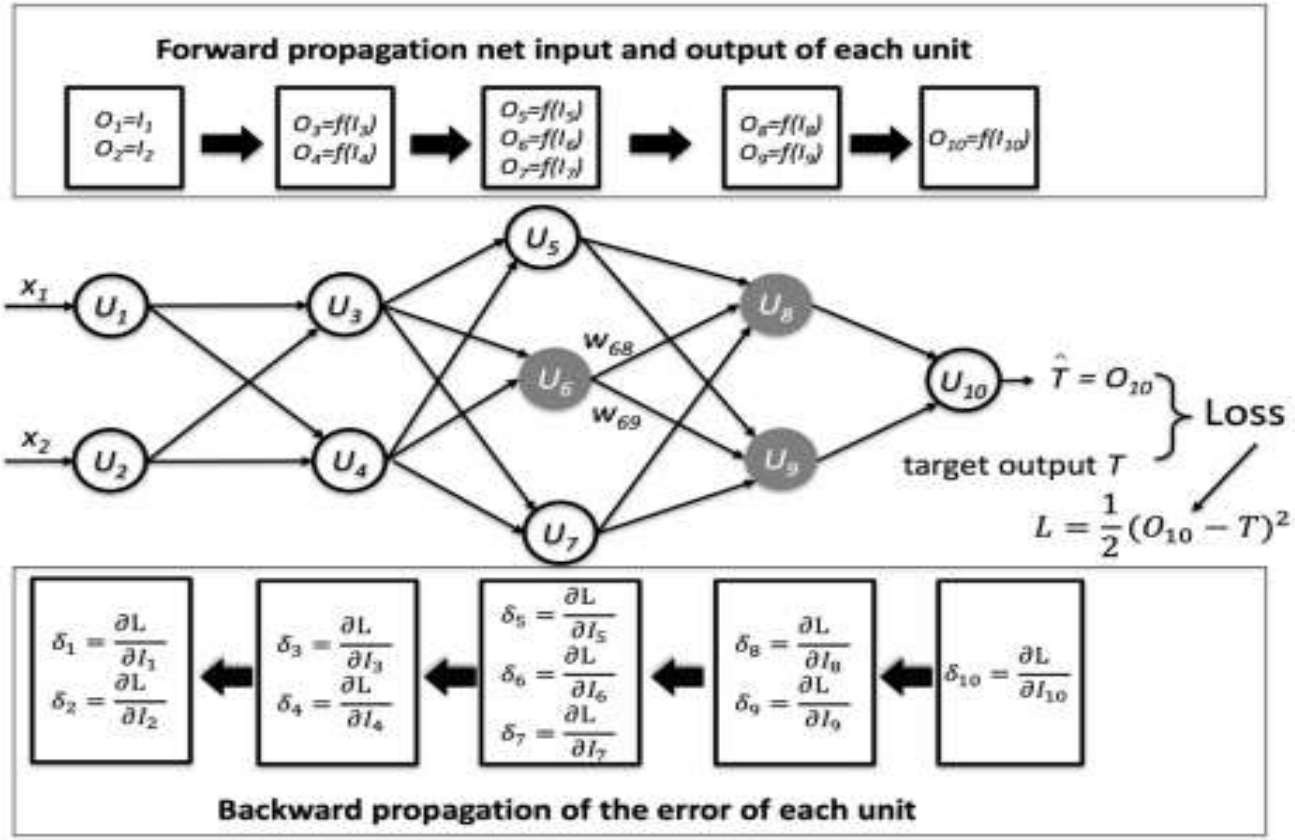


FIGURE 10.5

An illustration of the backpropagation algorithm. Given a training tuple, the algorithm first forward propagates the net input and output of each unit (the upper part of the figure). Then, it backward propagates the error of each unit (the bottom part of the figure). The output O_{10} of unit 10 gives the predicted target value $\hat{T} = O_{10}$, which is a function of the weights and bias scalars. The goal of backpropagation is to adjust the weights and bias scalars so that the predicted output matches the actual target output T as well as possible. That is, to minimize the loss $L = \frac{1}{2}(\hat{T} - T)^2$. The error of a given unit can be recursively computed based on the errors of the units it connects to in the next higher layer. For example, the error of unit 6 can be computed based on the errors of unit 8 and unit 9 (shaded in the figure). That is, $\delta_6 = O_6(1 - O_6)(w_{68}\delta_8 + w_{69}\delta_9)$. For clarity, the bias scalar or the weights between other units are not shown in the figure. Note that we actually do not need to calculate $\delta_1 = \delta_2 = 0$, since the input units (U_1 and U_2) just pass through the input attributes: $O_1 = I_1 = X_1$ and $O_2 = I_2 = X_2$.

Example

Training tuple <x1=1, x2=0,x3=1> with a class label of 1.
Learning rate (n)=0.9

Table 10.1 Initial inputs, weights, and bias values.

x_1	x_2	x_3	w_{14}	w_{15}	w_{24}	w_{25}	w_{34}	w_{35}	w_{46}	w_{56}	b_4	b_5	b_6
1	0	1	0.2	−0.3	0.4	0.1	−0.5	0.2	−0.3	−0.2	−0.4	0.2	0.1

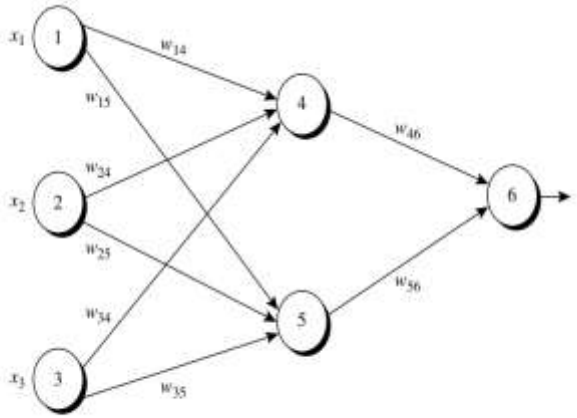


Figure 9.5 Example of a multilayer feed-forward neural network.

Table 10.2 Net input and output calculations.

Unit, j	Net Input, I_j	Output, O_j
4	$0.2+0-0.5-0.4=-0.7$	$1/(1+e^{0.7})=0.332$
5	$-0.3+0+0.2+0.2=0.1$	$1/(1+e^{-0.1})=0.525$
6	$(-0.3)(0.332)-(0.2)(0.525)+0.1=-0.105$	$1/(1+e^{0.105})=0.474$

Table 10.3 Calculation of the error at each node.

Unit, j	Error, δ_j
6	$(0.474)(1-0.474)(1-0.474)=0.1311$
5	$(0.525)(1-0.525)(0.1311)(-0.2)=-0.0065$
4	$(0.332)(1-0.332)(0.1311)(-0.3)=-0.0087$

Table 10.4 Calculations for weight and bias updating.

Weight or Bias	New Value
w_{46}	$-0.3-(0.9)(0.1311)(0.332)=-0.339$
w_{56}	$-0.2-(0.9)(0.1311)(0.525)=-0.262$
w_{14}	$0.2-(0.9)(-0.0087)(1)=0.208$
w_{15}	$-0.3-(0.9)(-0.0065)(1)=-0.294$
w_{24}	$0.4-(0.9)(-0.0087)(0)=0.4$
w_{25}	$0.1-(0.9)(-0.0065)(0)=0.1$
w_{34}	$-0.5-(0.9)(-0.0087)(1)=-0.492$
w_{35}	$0.2-(0.9)(-0.0065)(1)=0.206$
b_6	$0.1-(0.9)(0.1311)=-0.180$
b_5	$0.2-(0.9)(-0.0065)=0.206$
b_4	$-0.4-(0.9)(-0.0087)=-0.392$

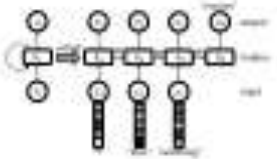
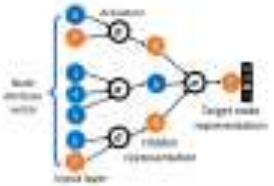
Key Challenges for Training Deep Learning Models

- Challenge 1
  - We do not know weights and biases
  - Backpropagation learns by minimizing training error
  - The overall training error is the average loss among all training tuples.
  - How to minimize the error with minimum number of epochs?
  - It is a nonconvex problem.
- Challenge 2: Minimizing generalization error
  - Overfitting problem
  - Training error is minimum and testing error is large

Overview of deep learning architectures

- Feed-forward is for multidimensional data
- For grid data, convolutional neural network is used
- For sequence data, recurrent neural network is used
- Graph neural networks are used for relational data (graph)

Table 10.5 An overview of typical deep learning architectures.

Data Type	Multidimensional	Grid	Sequence	Graph
	$x = (4.5, 500, 3, 5)$		x = "I love watching movies."	
DL Architecture	Feed-forward Network	CNN	RNN	GNN
				

Neural Network as a Classifier

- Weakness
  - Long training time
  - Require a number of parameters typically best determined empirically, e.g., the network topology or “structure.”
  - Poor interpretability: Difficult to interpret the symbolic meaning behind the learned weights and of “hidden units” in the network
- Strength
  - High tolerance to noisy data
  - Ability to classify untrained patterns
  - Well-suited for continuous-valued inputs *and outputs*
  - Successful on an array of real-world data, e.g., hand-written letters
  - Algorithms are inherently parallel
  - Techniques have recently been developed for the extraction of rules from trained neural networks

Advanced Cluster Analysis: Overview

P. Krishna Reddy, IIIT Hyderabad

Outline

- Fuzzy sets
- Clustering High-Dimensional Data
- Clustering Graphs and Network Data
- Clustering with Constraints
- Summary

Fuzzy sets

- The word "fuzzy" means "vaguness (ambiguity)".
- Fuzziness occurs when the boundary of a piece of information is not clear-cut.
- Fuzzy sets - 1965 Lotfi Zadeh as an extension of classical notation set.
- Classical set theory allows the membership of the elements in the set in **binary terms**.
- Fuzzy set theory permits membership function valued in the interval [0,1].

Fuzzy sets

Example:

Words like young, tall, good or high are fuzzy.

- There is no single quantitative value which defines the term young.
- For some people, age 25 is young, and for others, age 35 is young.
  The concept young has no clean boundary.
- Age 35 has some possibility of being young and usually
- depends on the context in which it is being considered.

Fuzzy set theory is an extension of classical set theory where elements have degree of membership.

Introduction

- In real world, there exist much fuzzy knowledge (i.e. vague, uncertain inexact etc).
- Human **thinking** and **reasoning** (analysis, logic, interpretation) frequently involved **fuzzy** information.
- Human can give satisfactory answers, which are probably true.
- Our systems are unable to answer many question because the systems are designed based upon classical set theory (Unreliable and incomplete).

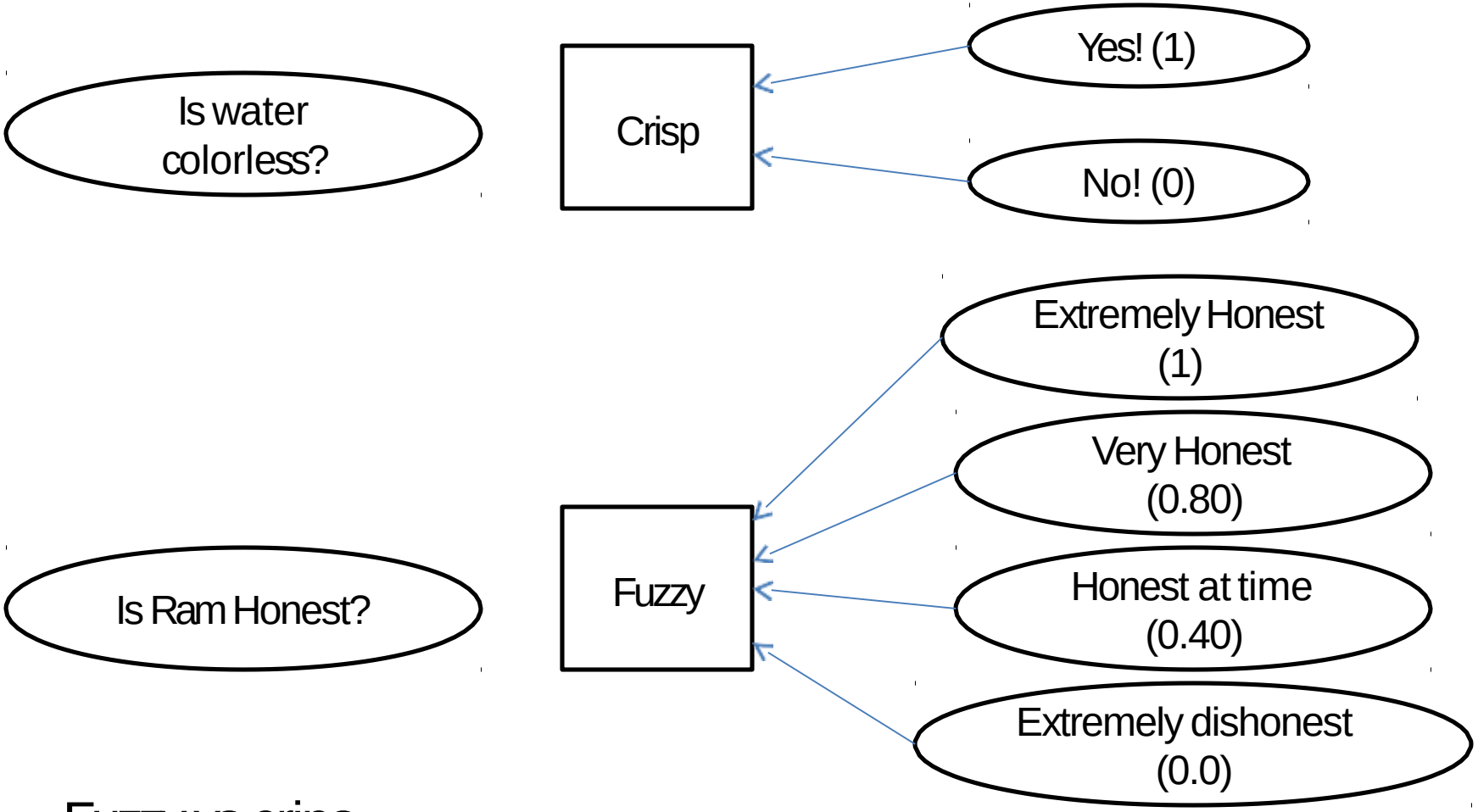
- We want, our system should be able to cope with unreliable and incomplete information.
- Fuzzy system have been provide solution.

Introduction

Classical set theory	Fuzzy set theory
• Classes of objects with sharp boundaries.	• Classes of objects with un- sharp boundaries.
• A classical set is defined by crisp(exact) boundaries, i.e., there is no uncertainty about the location of the set boundaries.	• A fuzzy set is defined by its ambiguous boundaries, i.e., there exists uncertainty about the location of the set boundaries.
• Widely used in digital system design	• Used in fuzzy controllers.

Introduction (Continue)

Example



Fuzzy vs crips

- Fuzzy sets can be used for cluster analysis
- An element may belong to two clusters

Clustering High-Dimensional Data

- Clustering high-dimensional data (How high is high-D in clustering?)
  - A data object may have more than 10 attributes
    - Many applications: text documents, DNA micro-array data
  - Major challenges:
    - Many irrelevant dimensions may mask clusters
    - Distance measure becomes meaningless—due to equi-distance
    - Clusters may exist only in some subspaces

Traditional Distance Measures May Not Be Effective on High-D Data

- Traditional distance measure could be dominated by noises in many dimensions
- Ex. Which pairs of customers are more similar?

Customer	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9	P_{10}
Ada	1	0	0	0	0	0	0	0	0	0
Bob	0	0	0	0	0	0	0	0	0	1
Cathy	1	0	0	0	1	0	0	0	0	1

- By Euclidean distance, we get.

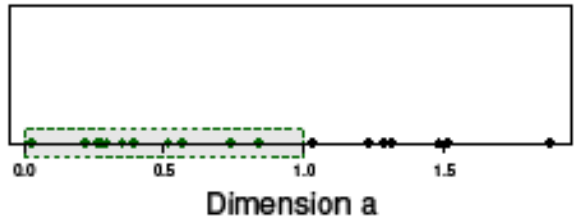
$$dist(\text{Ada}, \text{Bob}) = dist(\text{Bob}, \text{Cathy}) = dist(\text{Ada}, \text{Cathy}) = \sqrt{2}$$

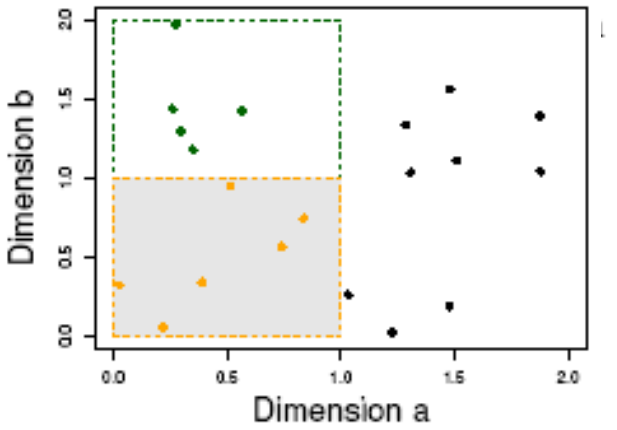
  - despite Ada and Cathy look more similar
- Challenge
  - Finding clusters, but, for each cluster, a set of attributes that manifest the cluster.
  - A cluster on high-dimensional data often is defined using a small set of attributes instead of the full data space.
  - **Clustering high-dimensional data should return groups of objects as clusters (as conventional cluster analysis does), *in addition to*, for each cluster, the set of attributes that characterize the cluster.**
  - **Challenge**
    - **Exponential number of possible subspaces**

The Curse of Dimensionality

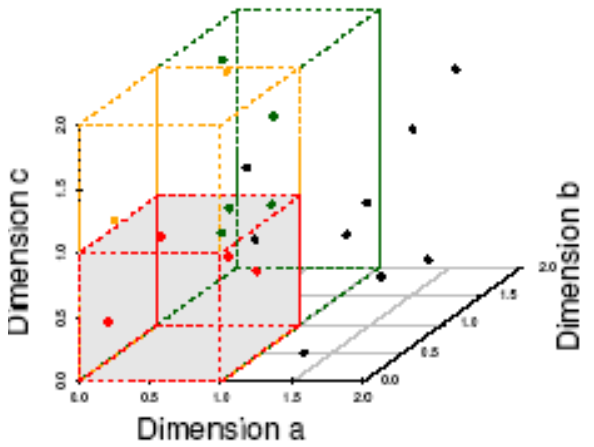
(graphs adapted from Parsons et al. KDD Explorations 2004)

- Data in only one dimension is relatively packed
- Adding a dimension "stretch" the points across that dimension, making them further apart
- Adding more dimensions will make the points further apart—high dimensional data is extremely sparse
- Distance measure becomes meaningless—due to equi-distance

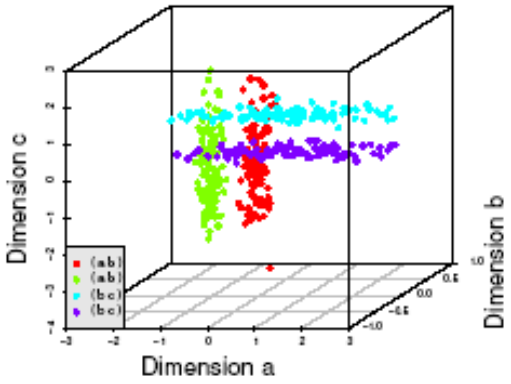




(b) 6 Objects in One Unit Bin



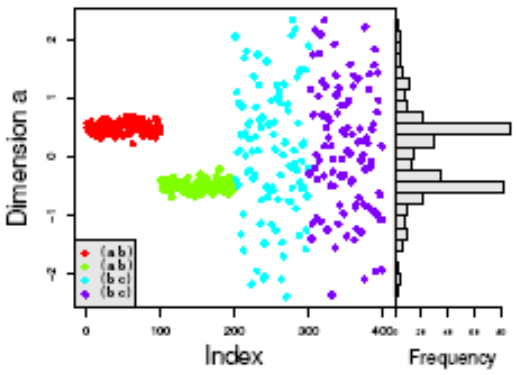
(c) 4 Objects in One Unit Bin



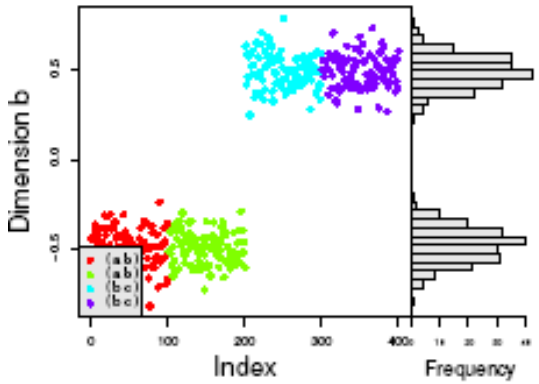
Why Subspace Clustering?

(adapted from Parsons et al. SIGKDD Explorations 2004)

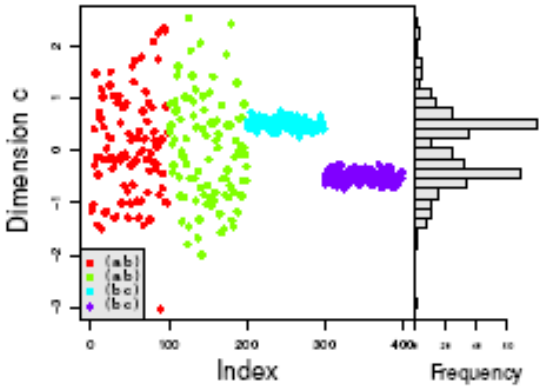
- Clusters may exist only in some subspaces
- Subspace-clustering: find clusters in all the subspaces



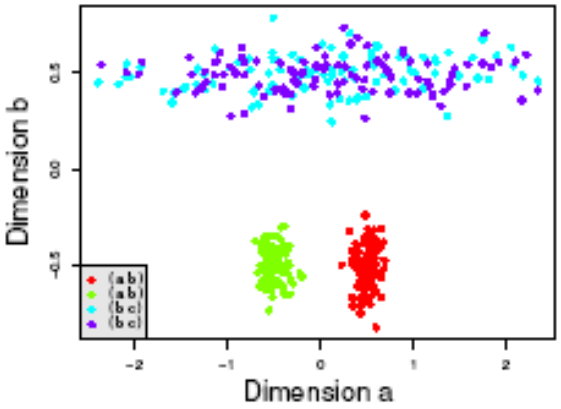
(a) Dimension a



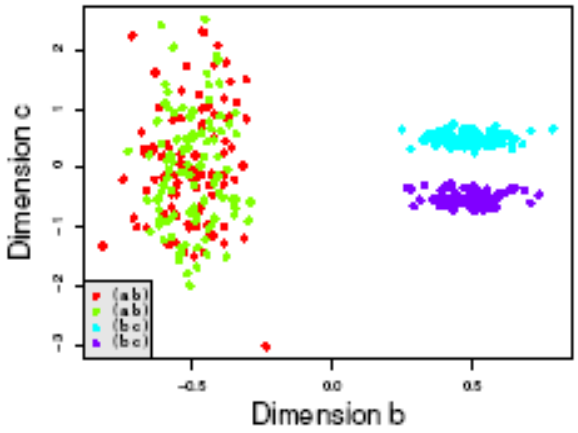
(b) Dimension b



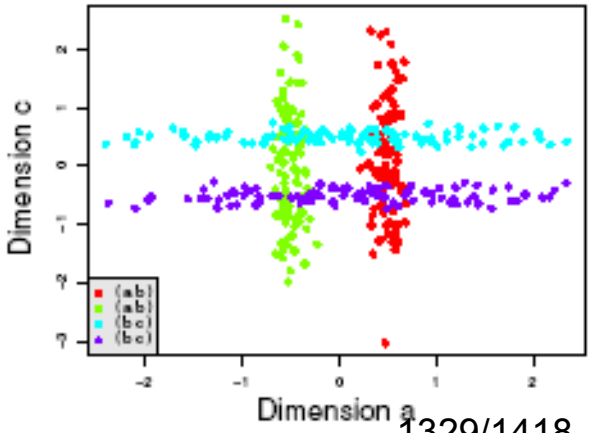
(c) Dimension c



(a) Dims a & b



(b) Dims b & c



(c) Dims a & c

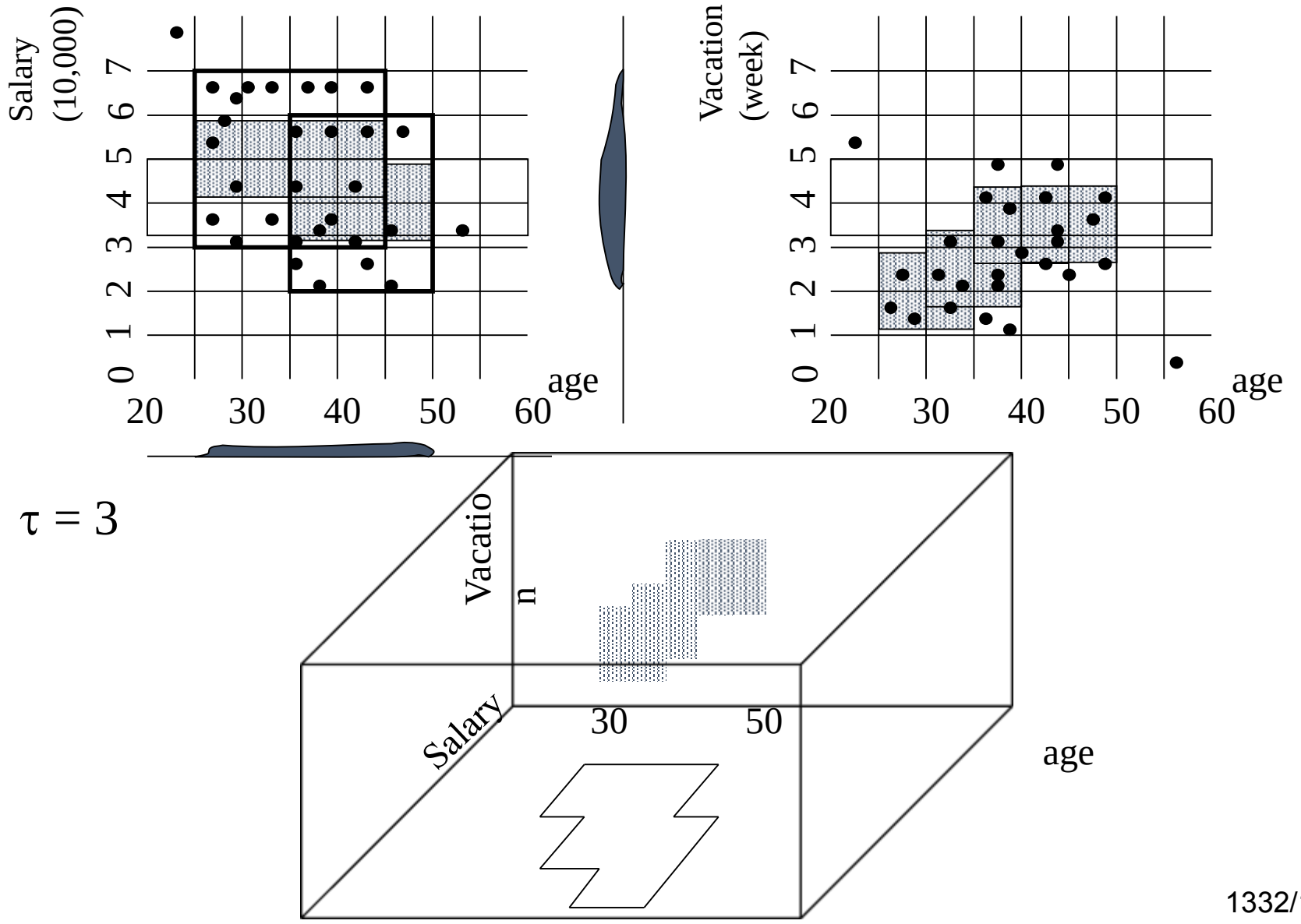
Subspace Clustering Methods

- Subspace search methods: Search various subspaces to find clusters
  - Bottom-up approaches
  - Top-down approaches
- Correlation-based clustering methods
  - E.g., PCA based approaches
- Bi-clustering methods
  - Optimization-based methods
  - Enumeration methods

Subspace Clustering Method (I): Subspace Search Methods

- Search various subspaces to find clusters
- *Bottom-up approaches*
  - Start from low-D subspaces and search higher-D subspaces only when there may be clusters in such subspaces
  - Various pruning techniques to reduce the number of higher-D subspaces to be searched
  - Ex. CLIQUE (Agrawal et al. 1998)
- *Top-down approaches*
  - Start from full space and search smaller subspaces recursively
  - Effective only if the *locality assumption* holds: restricts that the subspace of a cluster can be determined by the local neighborhood
  - Ex. PROCLUS (Aggarwal et al. 1999): a *k*-medoid-like method

CLIQUE: SubSpace Clustering with Aprori Pruning

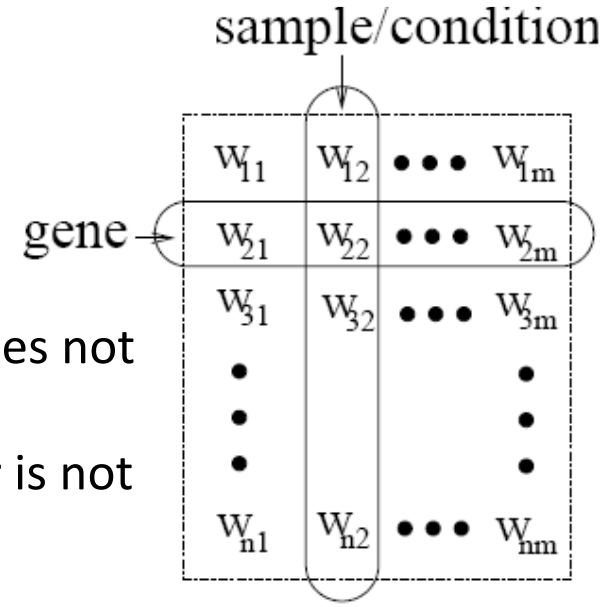


Subspace Clustering Method (II): Correlation-Based Methods

- Subspace search method: similarity based on distance or density
- Correlation-based method: based on advanced correlation models
- Ex. PCA-based approach:
  - Apply PCA (for Principal Component Analysis) to derive a set of new, uncorrelated dimensions,
  - then mine clusters in the new space or its subspaces
- Other space transformations:
  - Hough transform
  - Fractal dimensions

Subspace Clustering Method (III): Bi-Clustering Methods

- Bi-clustering: Cluster both objects and attributes simultaneously (treat objs and attrs in symmetric way)
- Four requirements:
  - Only a small set of objects participate in a cluster
  - A cluster only involves a small number of attributes
  - An object may participate in multiple clusters, or does not participate in any cluster at all
  - An attribute may be involved in multiple clusters, or is not involved in any cluster at all



- Ex 1. *Gene expression* or *microarray data:* a gene sample/condition matrix.
  - Each element in the matrix, a real number, records the expression level of a gene under a specific condition
- Ex. 2. Clustering customers and products
  - Another bi-clustering problem

products

customers

$$\begin{matrix} w_{11} & w_{12} & \cdots & w_{1m} \\ w_{21} & w_{22} & \cdots & w_{2m} \\ \cdots & \cdots & \cdots & \cdots \\ w_{n1} & w_{n2} & \cdots & w_{nm} \end{matrix}$$

Types of Bi-clusters

- Let A = {a_1, ..., a_n} be a set of genes, B = {b_1, …, b_n} a set of conditions
- A bi-cluster: A submatrix where genes and conditions follow some consistent patterns
- 4 types of bi-clusters (ideal cases)
  - Bi-clusters with constant values:
    - for any *i* in *I* and *j in J*, $e_{ij} = c$
  - Bi-clusters with constant values on rows:
    - $e_{ij} = c + \alpha_i$
    - Also, it can be constant values on columns
  - Bi-clusters with *coherent values* (aka. *pattern-based clusters*)
    - $e_{ij} = c + \alpha_i + \beta_j$
  - Bi-clusters with *coherent* evolutions on rows
    - e_{ij} $(e_{i1j1} - e_{i1j2})(e_{i2j1} - e_{i2j2}) \geq 0$
    - i.e., only interested in the up- or down- regulated cha conditions without constraining on the exact values

10	10	10	10	10
20	20	20	20	20
50	50	50	50	50
0	0	0	0	0

10	50	30	70	20
20	60	40	80	30
50	90	70	110	60
0	40	20	60	10

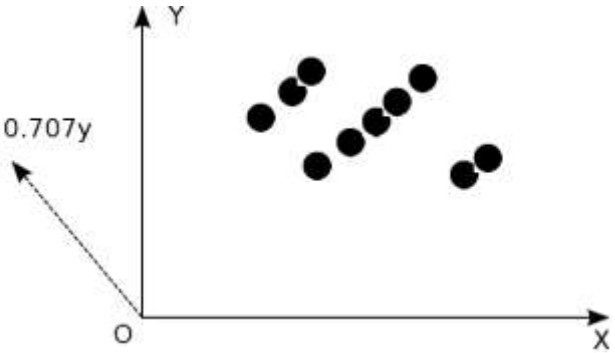
10	50	30	70	20
20	100	50	1000	30
50	100	90	120	80
0	80	20	100	10

Bi-Clustering Methods

- Real-world data is noisy: Try to find approximate bi-clusters
- Methods: Optimization-based methods vs. enumeration methods
- Optimization-based methods
  - Try to find a submatrix at a time that achieves the best significance as a bi-cluster
  - Due to the cost in computation, greedy search is employed to find local optimal bi-clusters
  - Ex. δ-Cluster Algorithm (Cheng and Church, ISMB'2000)
- Enumeration methods
  - Use a tolerance threshold to specify the degree of noise allowed in the bi-clusters to be mined
  - Then try to enumerate all submatrices as bi-clusters that satisfy the requirements
  - Ex. δ-pCluster Algorithm (H. Wang et al.' SIGMOD'2002, MaPle: Pei et al., ICDM'2003)

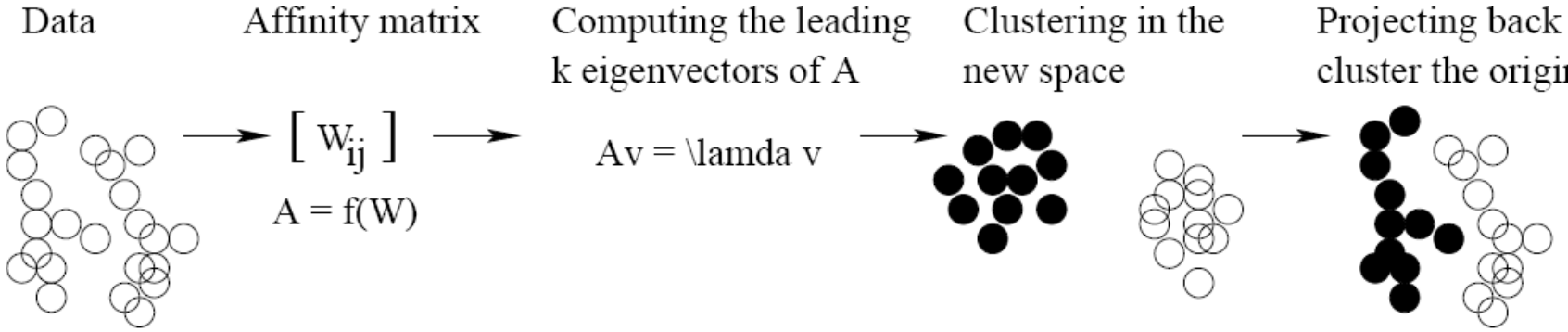
Dimensionality-Reduction Methods

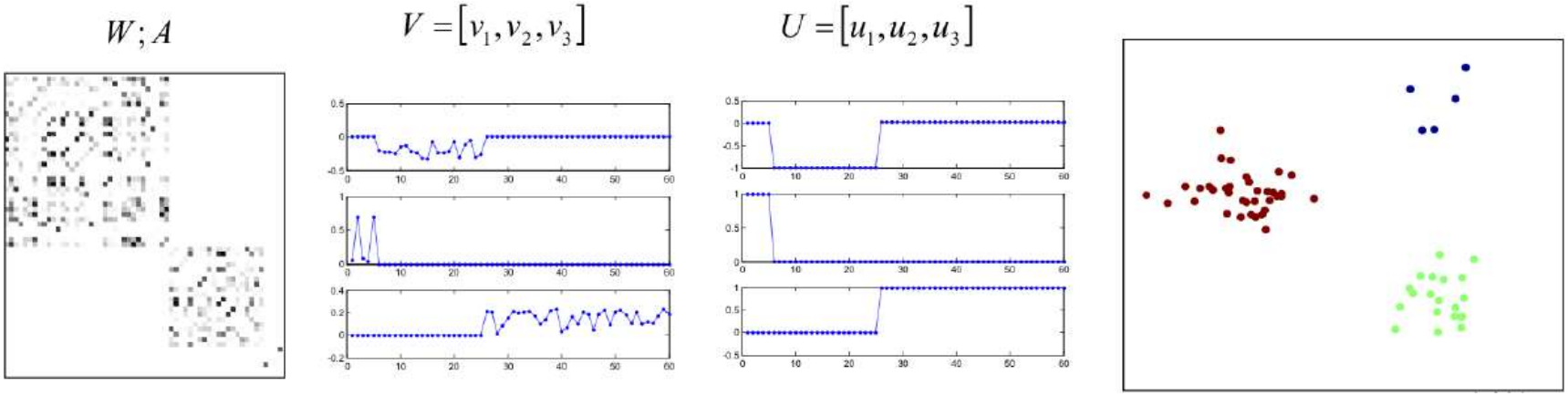
- Dimensionality reduction: In some situations, it is more effective to construct a new space instead of using some subspaces of the original data



- Ex. To cluster the points in the right figure, any subspace of the original one, X and Y, cannot help, since all the three clusters will be projected into the overlapping areas in X and Y axes.
  - Construct a new dimension as the dashed one, the three clusters become apparent when the points projected into the new dimension
- Dimensionality reduction methods
  - Feature selection and extraction: But may not focus on clustering structure finding
  - Spectral clustering: Combining feature extraction and clustering (i.e., use the *spectrum* of the similarity matrix of the data to perform dimensionality reduction for clustering in fewer dimensions)
    - Normalized Cuts (Shi and Malik, CVPR'97 or PAMI'2000)
    - The Ng-Jordan-Weiss algorithm (NIPS'01)

Spectral Clustering: Illustration and Comments





- Spectral clustering: Effective in tasks like image processing
- Scalability challenge: Computing eigenvectors on a large matrix is costly
- Can be combined with other clustering methods, such as bi-clustering

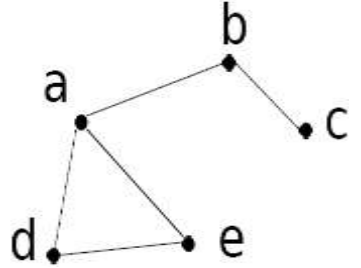
Outline

- Clustering High-Dimensional Data
- Clustering Graphs and Network Data
- Clustering with Constraints
- Summary

Clustering Graphs and Network Data

- Applications
  - Bi-partite graphs, e.g., customers and products, authors and conferences
  - Web search engines, e.g., click through graphs and Web graphs
  - Social networks, friendship/coauthor graphs
- Similarity measures
  - Geodesic distances
  - Distance based on random walk (SimRank)
- Graph clustering methods
  - Minimum cuts: FastModularity (Clauset, Newman & Moore, 2004)
  - Density-based clustering: SCAN (Xu et al., KDD'2007)

Similarity Measure (I): Geodesic Distance



- Geodesic distance (A, B): length (i.e., # of edges) of the shortest path between A and B (if not connected, defined as infinite)
- **Eccentricity** of v, eccen(v): The largest geodesic distance between v and any other vertex u ∈ V − {v}.
  - E.g., eccen(a) = eccen(b) = 2; eccen(c) = eccen(d) = eccen(e) = 3
- **Radius** of graph G: The minimum eccentricity of all vertices, i.e., the distance between the "most central point" and the "farthest border"
  - r = min $_{v \in V}$ eccen(v)
  - E.g., radius (G) = 2
- **Diameter** of graph G: The maximum eccentricity of all vertices, i.e., the largest distance between any pair of vertices in G
  - d = max $_{v \in V}$ eccen(v)
  - E.g., diameter (G) = 3
- A **peripheral vertex** is a vertex that achieves the diameter.
  - E.g., Vertices c, d, and e are peripheral vertices

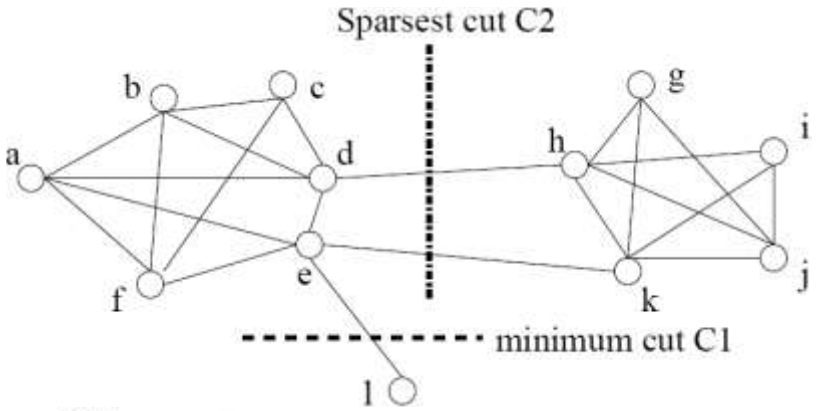
SimRank: Similarity Based on Random Walk and Structural Context

- SimRank: structural-context similarity, i.e., based on the similarity of its neighbors
- In a directed graph G = (V,E),
  - *individual in-neighborhood* of v: I(v) = {u | (u, v) ∈ E}
  - *individual out-neighborhood* of v: O(v) = {w | (v, w) ∈ E}
- Similarity in SimRank: $$s(u,v) = \frac{C}{|I(u)||I(v)|} \sum_{x \in I(u)} \sum_{y \in I(v)} s(x,y)$$
  - Initialization: $$s_0(u,v) = \begin{cases} 0 & \text{if } u \neq v \\ 1 & \text{if } u = v \end{cases}$$
  - Then we can compute s_{i+1} from s_i based on the definition
- Similarity based on random walk: in a strongly connected component
  - Expected distance: $$d(u,v) = \sum_{t:u \rightsquigarrow v} P[t] l(t)$$
  - Expected meeting distance: $$m(u,v) = \sum_{t:(u,v) \rightsquigarrow (x,x)} P[t] l(t)$$
  - Expected meeting probability: $$p(u,v) = \sum_{t:(u,v) \rightsquigarrow (x,x)} P[t] C^{l(t)}$$

$$P[t] = \begin{cases} \prod_{i=1}^{k-1} \frac{1}{|O(w_i)|} & \text{if } l(t) > 0 \\ 0 & \text{if } l(t) = 0. \end{cases}$$

P[t] is the probability of the tour

Graph Clustering: Sparsest Cut



- G = (V,E). The *cut set* of a cut is the set of edges {(u, v) ∈ E | u ∈ S, v ∈ T } and S and T are in two partitions
- *Size* of the cut: # of edges in the cut set
- Min-cut (e.g., C_1) is not a good partition
- A better measure: **Sparsity**:

$$\Phi = \frac{\text{the size of the cut}}{\min\{|S|, |T|\}}$$

- A cut is ***sparsest*** if its sparsity is not greater than that of any other cut
- Ex. Cut C2 = ({a, b, c, d, e, f, l}, {g, h, i, j, k}) is the sparsest cut
- For k clusters, the **modularity** of a clustering assesses the quality of the clustering:

$$Q = \sum_{i=1}^{k} \left(\frac{l_i}{|E|} - \left(\frac{d_i}{2|E|}\right)^2\right)$$

l_i: # edges between vertices in the i-th cluster
d_i: the sum of the degrees of the vertices in the i-th cluster

- The *modularity* of a clustering of a graph is the difference between the fraction of all edges that fall into individual clusters and the fraction that would do so if the graph vertices were randomly connected
- The optimal clustering of graphs maximizes the modularity

Graph Clustering: Challenges of Finding Good Cuts

- High computational cost
  - Many graph cut problems are computationally expensive
  - The sparsest cut problem is NP-hard
  - Need to tradeoff between efficiency/scalability and quality
- Sophisticated graphs
  - May involve weights and/or cycles.
- High dimensionality
  - A graph can have many vertices. In a similarity matrix, a vertex is represented as a vector (a row in the matrix) whose dimensionality is the number of vertices in the graph
- Sparsity
  - A large graph is often sparse, meaning each vertex on average connects to only a small number of other vertices
  - A similarity matrix from a large sparse graph can also be sparse

Two Approaches for Graph Clustering

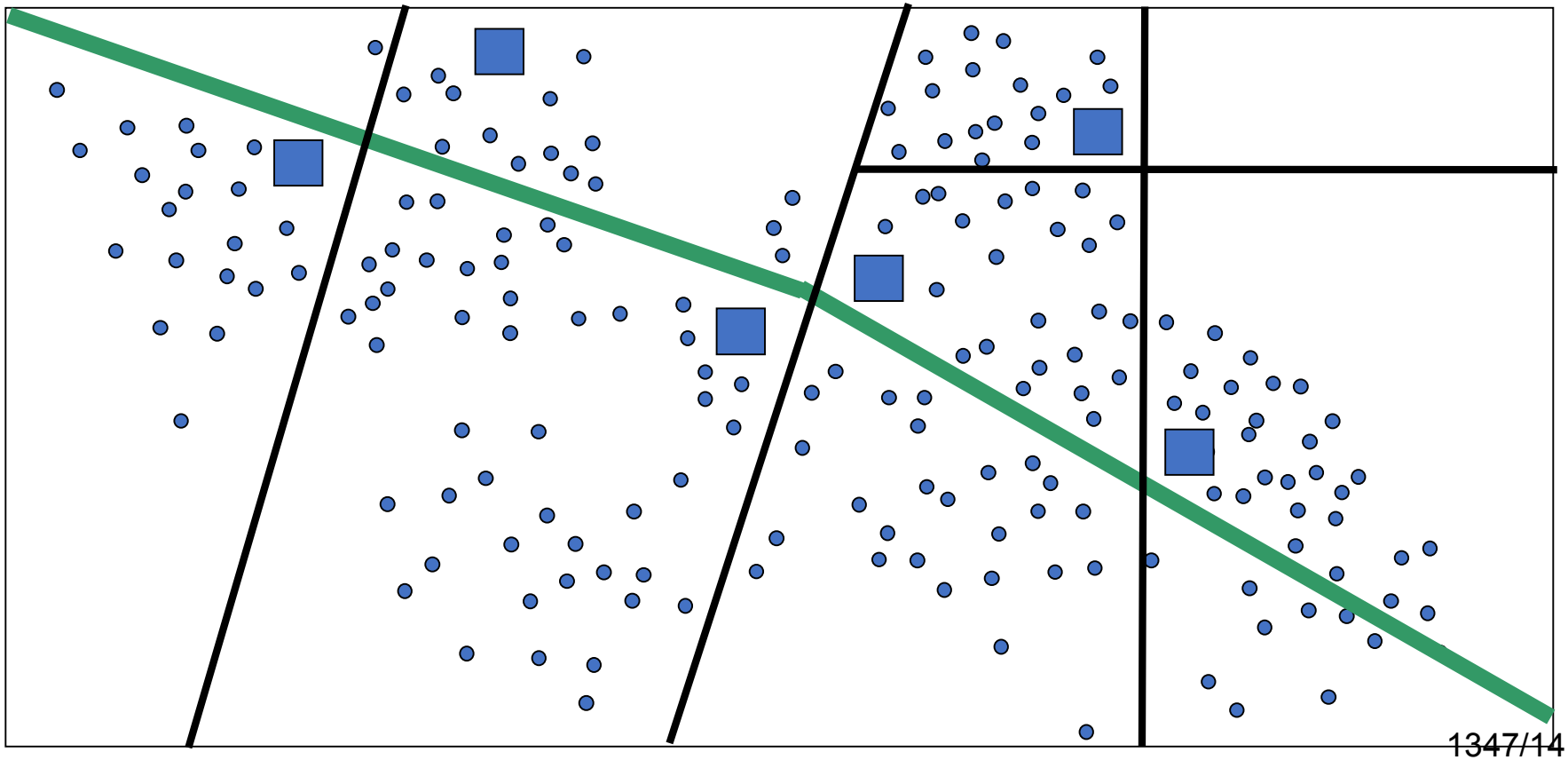
- Two approaches for clustering graph data
  - Use *generic clustering methods* for high-dimensional data
  - *Designed specifically for clustering graphs*
- Using clustering methods for high-dimensional data
  - Extract a similarity matrix from a graph using a similarity measure
  - A generic clustering method can then be applied on the similarity matrix to discover clusters
  - Ex. Spectral clustering: approximate optimal graph cut solutions
- Methods specific to graphs
  - Search the graph to find well-connected components as clusters
  - Ex. SCAN (Structural Clustering Algorithm for Networks)
    - X. Xu, N. Yuruk, Z. Feng, and T. A. J. Schweiger, “SCAN: A Structural Clustering Algorithm for Networks”, KDD'07

Outline

- Clustering High-Dimensional Data
- Clustering Graphs and Network Data
- **Clustering with Constraints**
- Summary

Why Constraint-Based Cluster Analysis?

- Need user feedback: Users know their applications the best
- Less parameters but more user-desired constraints, e.g., an ATM allocation problem: obstacle & desired clusters



Categorization of Constraints

- Constraints on instances: specifies how a pair or a set of instances should be grouped in the cluster analysis
  - Must-link vs. cannot link constraints
    - must-link(x, y): x and y should be grouped into one cluster
  - Constraints can be defined using variables, e.g.,
    - cannot-link(x, y) if dist(x, y) > d
- Constraints on clusters: specifies a requirement on the clusters
  - E.g., specify the min # of objects in a cluster, the max diameter of a cluster, the shape of a cluster (e.g., a convex), # of clusters (e.g., k)
- Constraints on similarity measurements: specifies a requirement that the similarity calculation must respect
  - E.g., driving on roads, obstacles (e.g., rivers, lakes)
- Issues: Hard vs. soft constraints; conflicting or redundant constraints

Constraint-Based Clustering Methods (I): Handling Hard Constraints

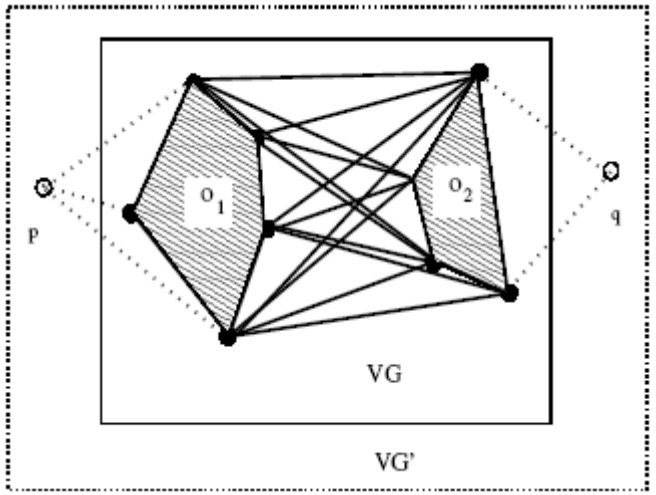
- Handling hard constraints: Strictly respect the constraints in cluster assignments
- Example: The COP-k-means algorithm
  - Generate super-instances for must-link constraints
    - Compute the transitive closure of the must-link constraints
    - To represent such a subset, replace all those objects in the subset by the mean.
    - The super-instance also carries a weight, which is the number of objects it represents
  - Conduct modified k-means clustering to respect cannot-link constraints
    - Modify the center-assignment process in k-means to a *nearest feasible center assignment*
    - An object is assigned to the nearest center so that the assignment respects all cannot-link constraints

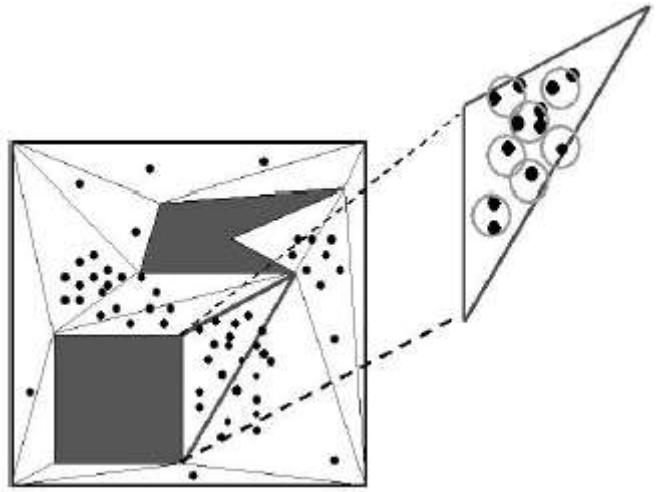
Constraint-Based Clustering Methods (II): Handling Soft Constraints

- Treated as an optimization problem: When a clustering violates a soft constraint, a penalty is imposed on the clustering
- Overall objective: Optimizing the clustering quality, and minimizing the constraint violation penalty
- Ex. CVQE (Constrained Vector Quantization Error) algorithm: Conduct k-means clustering while enforcing constraint violation penalties
- Objective function: Sum of distance used in k-means, adjusted by the constraint violation penalties
  - Penalty of a *must-link* violation
    - If objects x and y must-be-linked but they are assigned to two different centers, c_1 and c_2, dist(c_1, c_2) is added to the objective function as the penalty
  - Penalty of a cannot-link violation
    - If objects x and y cannot-be-linked but they are assigned to a common center c, dist(c, c'), between c and c' is added to the objective function as the penalty, where c' is the closest cluster to c that can accommodate x or y

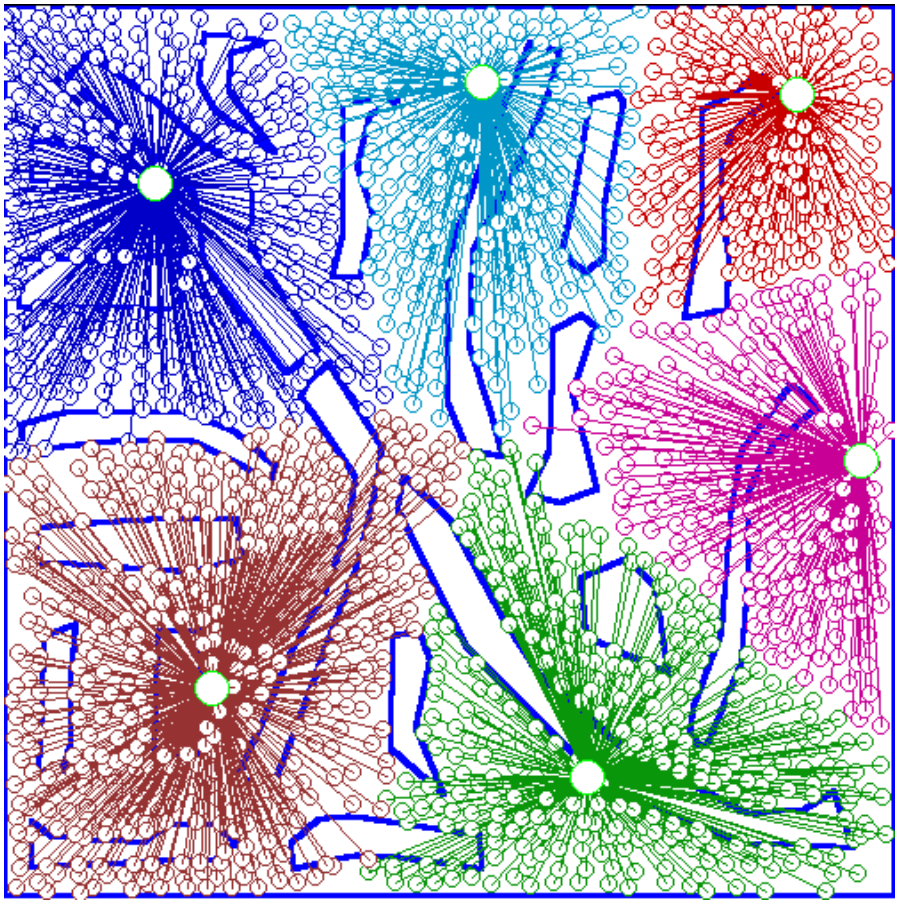
Speeding Up Constrained Clustering

- It is costly to compute some constrained clustering
- Ex. Clustering with obstacle objects: Tung, Hou, and Han. Spatial clustering in the presence of obstacles, ICDE'01
- K-medoids is more preferable since k-means may locate the ATM center in the middle of a lake
- Visibility graph and shortest path
- Triangulation and micro-clustering
- Two kinds of join indices (shortest-paths) worth pre-computation
  - VV index: indices for any pair of obstacle vertices
  - MV index: indices for any pair of micro-cluster and obstacle indices

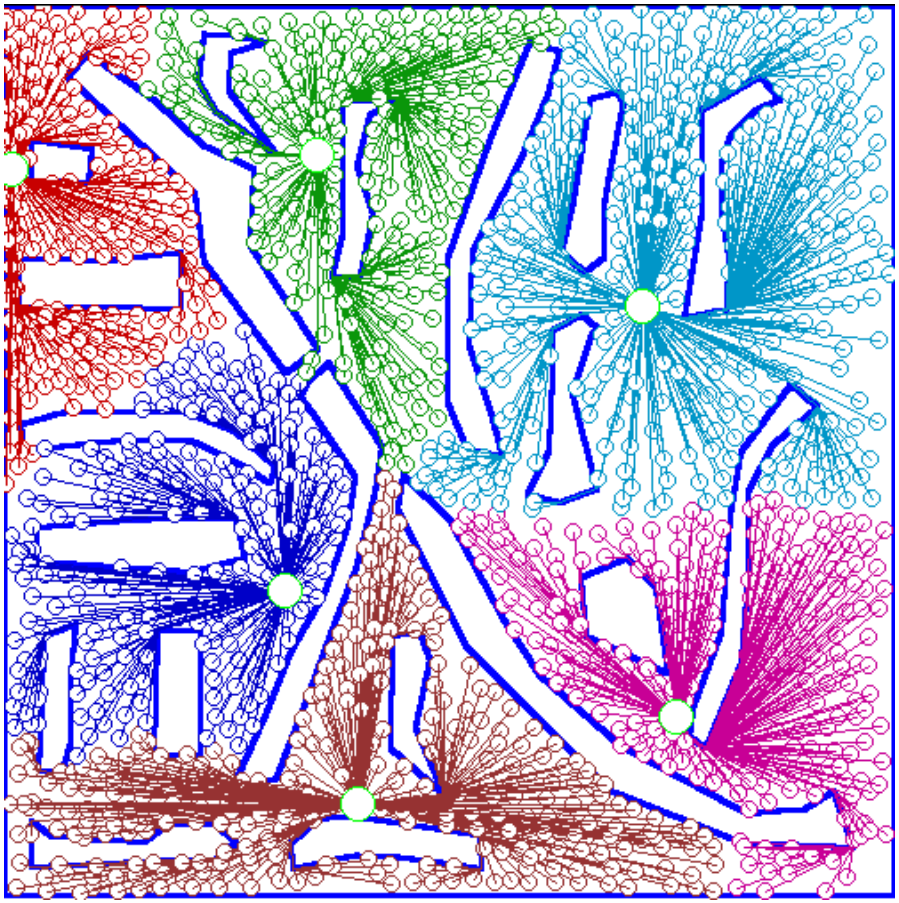




An Example: Clustering With Obstacle Objects



Not Taking obstacles into account



Taking obstacles into account

Summary

- Clustering High-Dimensional Data
  - Subspace clustering: bi-clustering methods
  - Dimensionality reduction: Spectral clustering
- Clustering Graphs and Network Data
  - Graph clustering: min-cut vs. sparsest cut
  - High-dimensional clustering methods
  - Graph-specific clustering methods, e.g., SCAN
- Clustering with Constraints
  - Constraints on instance objects, e.g., Must link vs. Cannot Link
  - Constraint-based clustering algorithms

Outlier Analysis: Overview

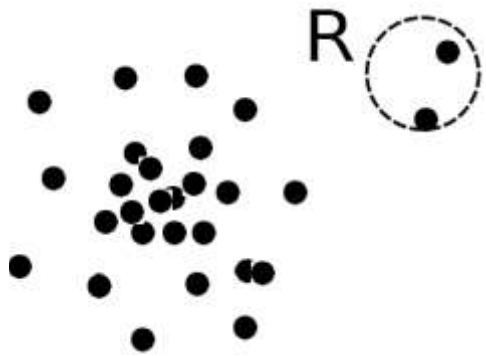
P. Krishna Reddy, IIIT Hyderabad

Outline

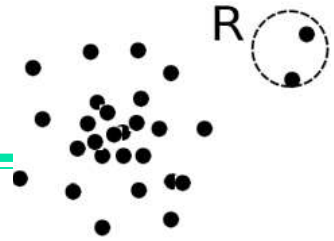
- Outlier and Outlier Analysis
- Overview of Outlier Detection Methods
- Summary

What Are Outliers?

- **Outlier**: A data object that **deviates significantly** from the normal objects as if it were **generated by a different mechanism**
  - Ex.: Unusual credit card purchase, sports: Michael Jordon, Wayne Gretzky, ...
- Outliers are different from the noise data
  - Noise is random error or variance in a measured variable
  - Noise should be removed before outlier detection
- Outliers are interesting: It violates the mechanism that generates the normal data
- Outlier detection vs. *novelty detection*: early stage, outlier; but later merged into the model
- Applications:
  - Credit card fraud detection
  - Telecom fraud detection
  - Customer segmentation
  - Medical analysis



Types of Outliers (I)

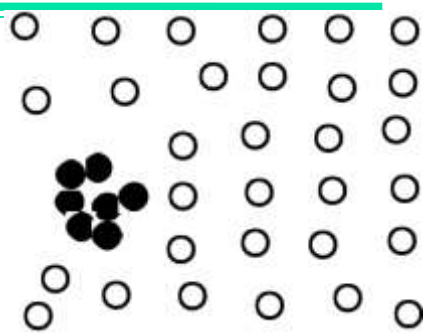


Global Outlier

- Three kinds: *global, contextual* and *collective* outliers
- **Global outlier** (or point anomaly)
  - Object is O_g if it significantly deviates from the rest of the data set
  - Ex. Intrusion detection in computer networks
  - Issue: Find an appropriate measurement of deviation
- **Contextual outlier** (or *conditional outlier*)
  - Object is O_c if it deviates significantly based on a selected context
    - Ex. 80^o F in Urbana: outlier? (depending on summer or winter?)
  - Attributes of data objects should be divided into two groups
    - Contextual attributes: defines the context, e.g., time & location
    - Behavioral attributes: characteristics of the object, used in outlier evaluation, e.g., temperature
  - Can be viewed as a generalization of *local outliers*—whose density significantly deviates from its local area
  - Issue: How to define or formulate meaningful context?

Types of Outliers (II)

- **Collective Outliers**
  - A subset of data objects *collectively* deviate significantly from the whole data set, even if the individual data objects may not be outliers
  - Applications: E.g., *intrusion detection*:
    - When a number of computers keep sending denial-of-service packages to each other



Collective Outlier

  - Detection of collective outliers
    - Consider not only behavior of individual objects, but also that of groups of objects
    - Need to have the background knowledge on the relationship among data objects, such as a distance or similarity measure on objects.
- A data set may have multiple types of outlier
- One object may belong to more than one type of outlier

Challenges of Outlier Detection

- Modeling normal objects and outliers properly
  - Hard to enumerate all possible normal behaviors in an application
  - The border between normal and outlier objects is often a gray area
- Application-specific outlier detection
  - Choice of distance measure among objects and the model of relationship among objects are often application-dependent
  - E.g., clinic data: a small deviation could be an outlier; while in marketing analysis, larger fluctuations
- Handling noise in outlier detection
  - Noise may distort the normal objects and blur the distinction between normal objects and outliers. It may help hide outliers and reduce the effectiveness of outlier detection
- Understandability
  - Understand why these are outliers: Justification of the detection
  - Specify the degree of an outlier: the unlikelihood of the object being generated by a normal mechanism

Outline

- Outlier and Outlier Analysis
- **Overview of Outlier Detection Methods**
- Summary

Outlier Detection I: Supervised Methods

- Two ways to categorize outlier detection methods:
  - Based on whether user-*labeled* examples of outliers can be obtained:
    - Supervised, semi-supervised vs. unsupervised methods
  - Based on *assumptions about normal data and outliers*:
    - Statistical, proximity-based, and clustering-based methods
- **Outlier Detection I: Supervised Methods**
  - Modeling outlier detection as a classification problem
    - Samples examined by domain experts used for training & testing
  - Methods for Learning a classifier for outlier detection effectively:
    - Model normal objects & report those not matching the model as outliers, or
    - Model outliers and treat those not matching the model as normal
  - Challenges
    - Imbalanced classes, i.e., outliers are rare: Boost the outlier class and make up some artificial outliers
    - Catch as many outliers as possible, i.e., recall is more important than accuracy (i.e., not mislabeling normal objects as outliers)

Outlier Detection II: Unsupervised Methods

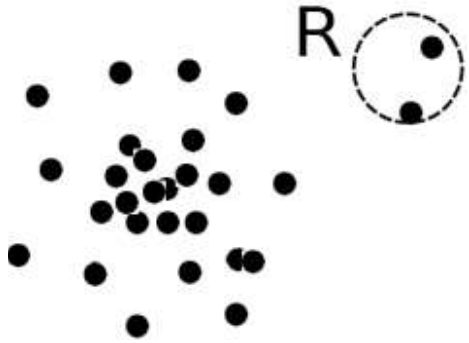
- Assume the normal objects are somewhat ``clustered'' into multiple groups, each having some distinct features
- An outlier is expected to be far away from any groups of normal objects
- Weakness: Cannot detect collective outlier effectively
  - Normal objects may not share any strong patterns, but the collective outliers may share high similarity in a small area
- Ex. In some intrusion or virus detection, normal activities are diverse
  - Unsupervised methods may have a high false positive rate but still miss many real outliers.
  - Supervised methods can be more effective, e.g., identify attacking some key resources
- Many clustering methods can be adapted for unsupervised methods
  - Find clusters, then outliers: not belonging to any cluster
  - Problem 1: Hard to distinguish noise from outliers
  - Problem 2: Costly since first clustering: but far less outliers than normal objects
    - Newer methods: tackle outliers directly

Outlier Detection III: Semi-Supervised Methods

- Situation: In many applications, the number of labeled data is often small: Labels could be on outliers only, normal objects only, or both
- Semi-supervised outlier detection: Regarded as applications of semi-supervised learning
- If some labeled normal objects are available
  - Use the labeled examples and the proximate unlabeled objects to train a model for normal objects
  - Those not fitting the model of normal objects are detected as outliers
- If only some labeled outliers are available, a small number of labeled outliers many not cover the possible outliers well
  - To improve the quality of outlier detection, one can get help from models for normal objects learned from unsupervised methods

Outlier Detection (1): Statistical Methods

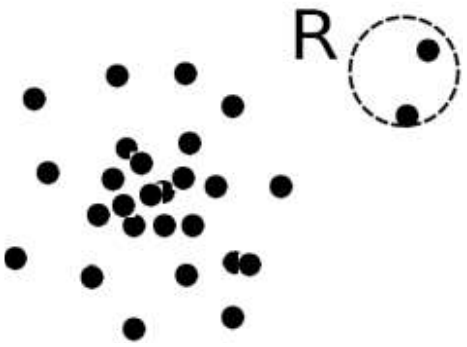
- Statistical methods (also known as model-based methods) assume that the normal data follow some statistical model (a stochastic model)
  - The data not following the model are outliers.
- Example (right figure): First use Gaussian distribution to model the normal data
  - For each object y in region R, estimate $g_D(y)$, the probability of y fits the Gaussian distribution
  - If $g_D(y)$ is very low, y is unlikely generated by the Gaussian model, thus an outlier



- Effectiveness of statistical methods: highly depends on whether the assumption of statistical model holds in the real data
- There are rich alternatives to use various statistical models
  - E.g., parametric vs. non-parametric

Outlier Detection (2): Proximity-Based Methods

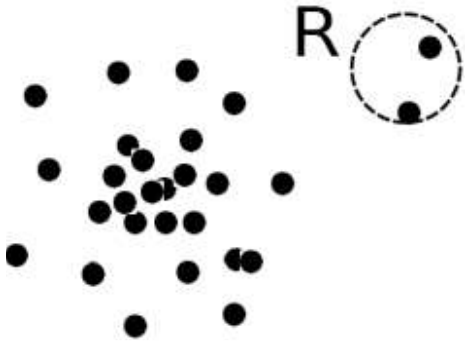
- An object is an outlier if the nearest neighbors of the object are far away, i.e., the **proximity** of the object is **significantly deviates** from the proximity of most of the other objects in the same data set
- Example (right figure): Model the proximity of an object using its 3 nearest neighbors
  - Objects in region R are substantially different from other objects in the data set.
  - Thus the objects in R are outliers



- The effectiveness of proximity-based methods highly relies on the proximity measure.
- In some applications, proximity or distance measures cannot be obtained easily.
- Often have a difficulty in finding a group of outliers which stay close to each other
- Two major types of proximity-based outlier detection
  - Distance-based vs. density-based

Outlier Detection (3): Clustering-Based Methods

- Normal data belong to large and dense clusters, whereas outliers belong to small or sparse clusters, or do not belong to any clusters
- Example (right figure): two clusters
  - All points not in R form a large cluster
  - The two points in R form a tiny cluster, thus are outliers



- Since there are many clustering methods, there are many clustering-based outlier detection methods as well
- Clustering is expensive: straightforward adaption of a clustering method for outlier detection can be costly and does not scale up well for large data sets

Summary

- Types of outliers
  - global, contextual & collective outliers
- Outlier detection
  - supervised, semi-supervised, or unsupervised
- Statistical (or model-based) approaches
- Proximity-base approaches
- Clustering-base approaches

Trends of Data Mining: An Overview



P. Krishna Reddy, IIIT Hyderabad

Outline

- Mining Complex Types of Data
- Other Methodologies of Data Mining
- Data Mining Applications
- Data Mining and Society
- Data Mining Trends
- Summary

Mining Complex Types of Data

- Mining Sequence Data
  - Mining Time Series
  - Mining Symbolic Sequences
  - Mining Biological Sequences
- Mining Graphs and Networks
- Mining Other Kinds of Data

Mining Sequence Data

- Similarity Search in Time Series Data
  - Subsequence match, dimensionality reduction, query-based similarity search, motif-based similarity search
- Regression and Trend Analysis in Time-Series Data
  - long term + cyclic + seasonal variation + random movements
- Sequential Pattern Mining in Symbolic Sequences
  - GSP, PrefixSpan, constraint-based sequential pattern mining
- Sequence Classification
  - Feature-based vs. sequence-distance-based vs. model-based
- Alignment of Biological Sequences
  - Pair-wise vs. multi-sequence alignment, substitution matirces, BLAST
- Hidden Markov Model for Biological Sequence Analysis
  - Markov chain vs. hidden Markov models, forward vs. Viterbi vs. Baum-Welch algorithms

Mining Graphs and Networks

- Graph Pattern Mining
  - Frequent subgraph patterns, closed graph patterns, gSpan vs. CloseGraph
- Statistical Modeling of Networks
  - Small world phenomenon, power law (log-tail) distribution, densification
- Clustering and Classification of Graphs and Homogeneous Networks
  - Clustering: Fast Modularity vs. SCAN
  - Classification: model vs. pattern-based mining
- Clustering, Ranking and Classification of Heterogeneous Networks
  - RankClus, RankClass, and meta path-based, user-guided methodology
- Role Discovery and Link Prediction in Information Networks
  - PathPredict
- Similarity Search and OLAP in Information Networks: PathSim, GraphCube
- Evolution of Social and Information Networks: EvoNetClus

Mining Other Kinds of Data

- Mining Spatial Data
  - Spatial frequent/co-located patterns, spatial clustering and classification
- Mining Spatiotemporal and Moving Object Data
  - Spatiotemporal data mining, trajectory mining, periodica, swarm, …
- Mining Cyber-Physical System Data
  - Applications: healthcare, air-traffic control, flood simulation
- Mining Multimedia Data
  - Social media data, geo-tagged spatial clustering, periodicity analysis, …
- Mining Text Data
  - Topic modeling, i-topic model, integration with geo- and networked data
- Mining Web Data
  - Web content, web structure, and web usage mining
- Mining Data Streams
  - Dynamics, one-pass, patterns, clustering, classification, outlier detection

Outline

- Mining Complex Types of Data
- **Other Methodologies of Data Mining**
- Data Mining Applications
- Data Mining and Society
- Data Mining Trends
- Summary

Other Methodologies of Data Mining

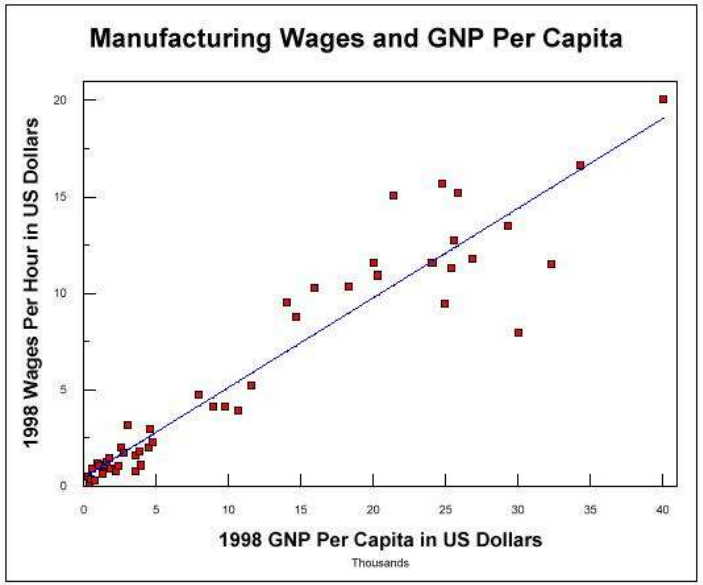
- Statistical Data Mining 
- Views on Data Mining Foundations
- Visual and Audio Data Mining

Major Statistical Data Mining Methods

- Regression
- Generalized Linear Model
- Analysis of Variance
- Mixed-Effect Models
- Factor Analysis
- Discriminant Analysis
- Survival Analysis

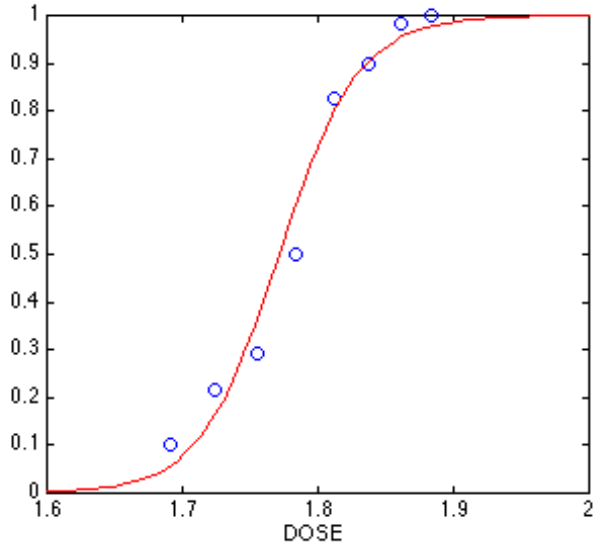
Statistical Data Mining (1)

- There are many well-established statistical techniques for data analysis, particularly for numeric data
  - applied extensively to data from scientific experiments and data from economics and the social sciences
- **Regression**
  - predict the value of a response (dependent) variable from one or more predictor (independent) variables where the variables are numeric
  - forms of regression: linear, multiple, weighted, polynomial, nonparametric, and robust



Scientific and Statistical Data Mining (2)

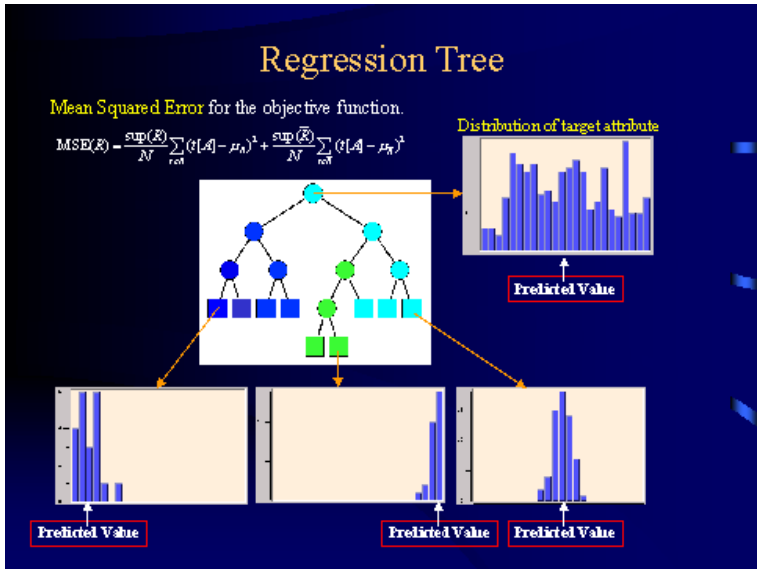
- **Generalized linear models**
  - allow a categorical response variable (or some transformation of it) to be related to a set of predictor variables
  - similar to the modeling of a numeric response variable using linear regressior
  - include logistic regression and Poisson regression

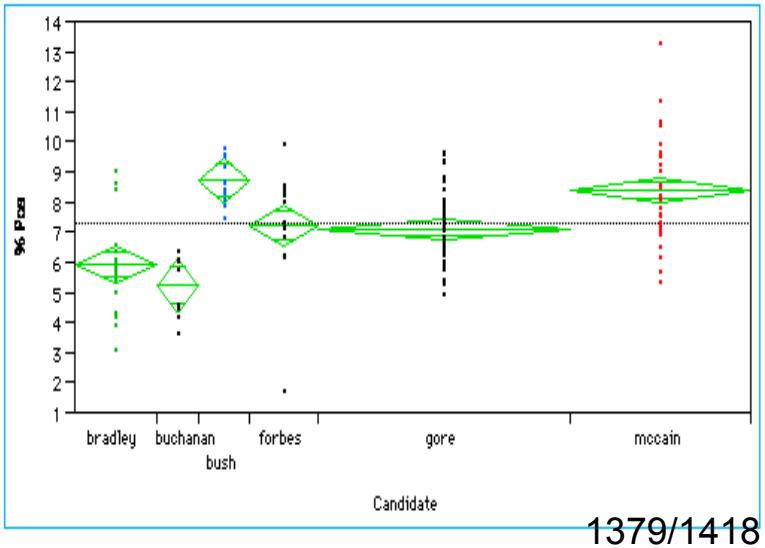


- **Mixed-effect models**
  - For analyzing grouped data, i.e. data that can be classified according to one or more grouping variables
  - Typically describe relationships between a response variable and some covariates in data grouped according to one or more factors

Scientific and Statistical Data Mining (3)

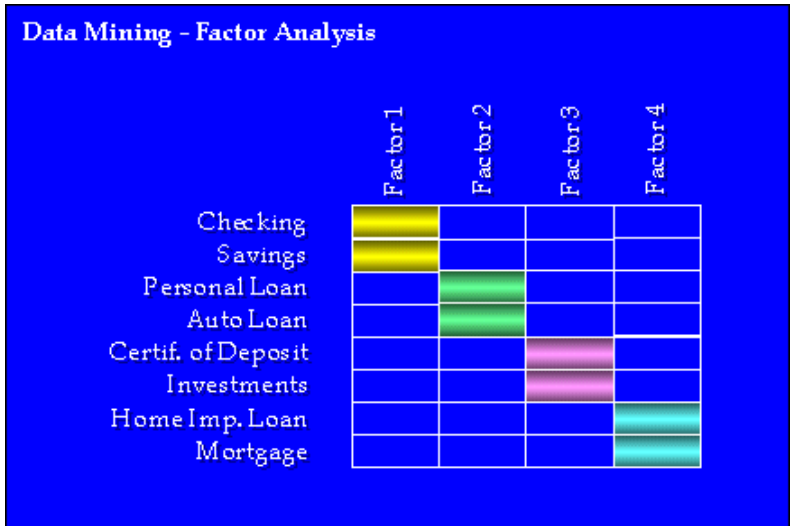
- **Regression trees**
  - Binary trees used for classification and prediction
  - Similar to decision trees:Tests are performed at the internal nodes
  - In a regression tree the mean of the objective attribute is computed and used as the predicted value
- **Analysis of variance**
  - Analyze experimental data for two or more populations described by a numeric response variable and one or more categorical variables (factors)

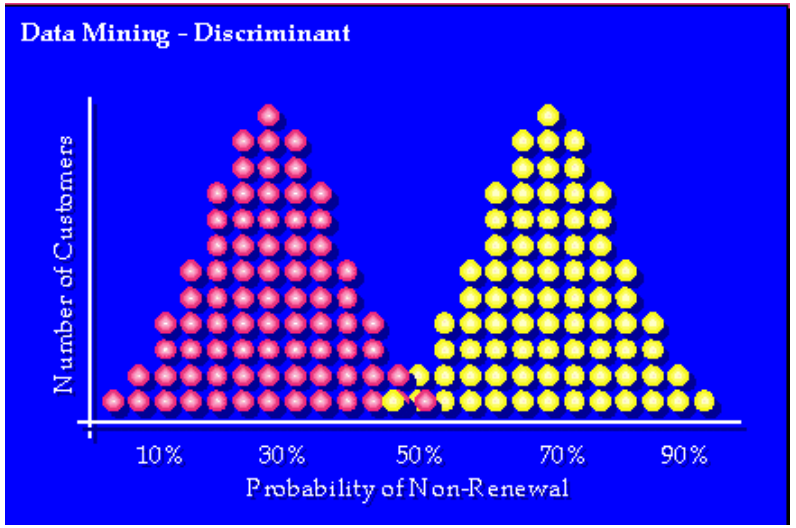




Statistical Data Mining (4)

- **Factor analysis**
  - determine which variables are combined to generate a given factor
  - e.g., for many psychiatric data, one can indirectly measure other quantities (such as test scores) that reflect the factor of interest
- **Discriminant analysis**
  - predict a categorical response variable, commonly used in social science
  - Attempts to determine several discriminant functions (linear combinations of the independent variables) that discriminate among the groups defined by the response variable

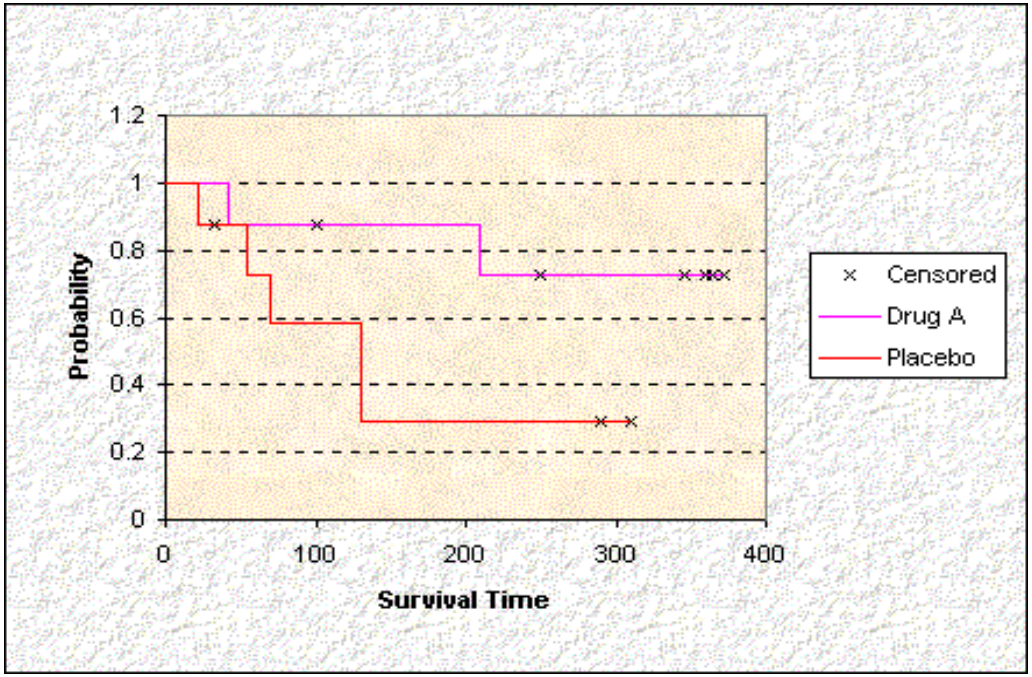




www.spss.com/datamine/factor.htm

Statistical Data Mining (5)

- **Time series**: many methods such as autoregression, ARIMA (Autoregressive integrated moving-average modeling), long memory time-series modeling
- **Quality control:** displays group summary charts
- **Survival analysis**
  - Predicts the probability that a patient undergoing a medical treatment would survive at least to time *t* (life span prediction)



Other Methodologies of Data Mining

- Statistical Data Mining
- Views on Data Mining Foundations
- Visual and Audio Data Mining

Views on Data Mining Foundations (I)

- Data reduction
  - Basis of data mining: Reduce data representation
  - Trades accuracy for speed in response
- Data compression
  - Basis of data mining: Compress the given data by encoding in terms of bits, association rules, decision trees, clusters, etc.
- Probability and statistical theory
  - Basis of data mining: Discover joint probability distributions of random variables

Views on Data Mining Foundations (II)

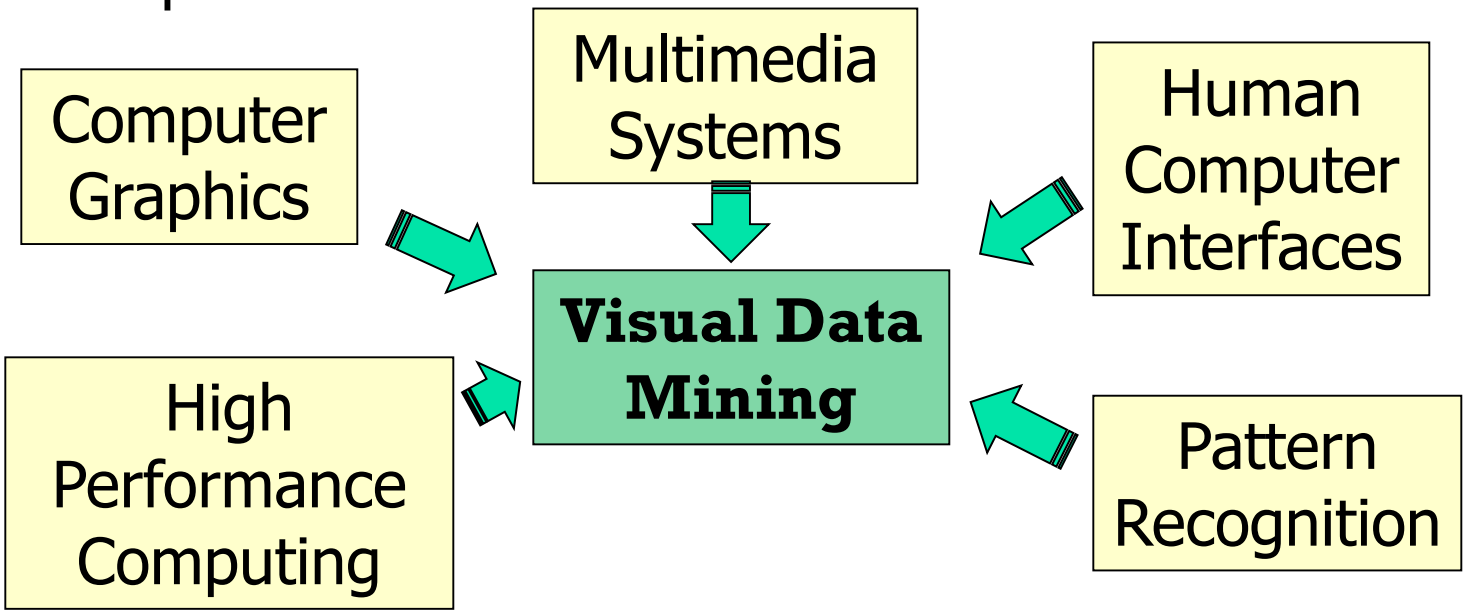
- Microeconomic view
  - A view of utility: Finding patterns that are interesting only to the extent in that they can be used in the decision-making process of some enterprise
- Pattern Discovery and Inductive databases
  - Basis of data mining: Discover patterns occurring in the database, such as associations, classification models, sequential patterns, etc.
  - Data mining is the problem of performing inductive logic on databases
  - The task is to query the data and the theory (i.e., patterns) of the database
  - Popular among many researchers in database systems

Other Methodologies of Data Mining

- Statistical Data Mining
- Views on Data Mining Foundations
- Visual and Audio Data Mining

Visual Data Mining

- Visualization: Use of computer graphics to create visual images which aid in the understanding of complex, often massive representations of data
- Visual Data Mining: discovering implicit but useful knowledge from large data sets using visualization techniques



Visualization

- Purpose of Visualization
  - Gain insight into an information space by mapping data onto graphical primitives
  - Provide qualitative overview of large data sets
  - Search for patterns, trends, structure, irregularities, relationships among data.
  - Help find interesting regions and suitable parameters for further quantitative analysis.
  - Provide a visual proof of computer representations derived

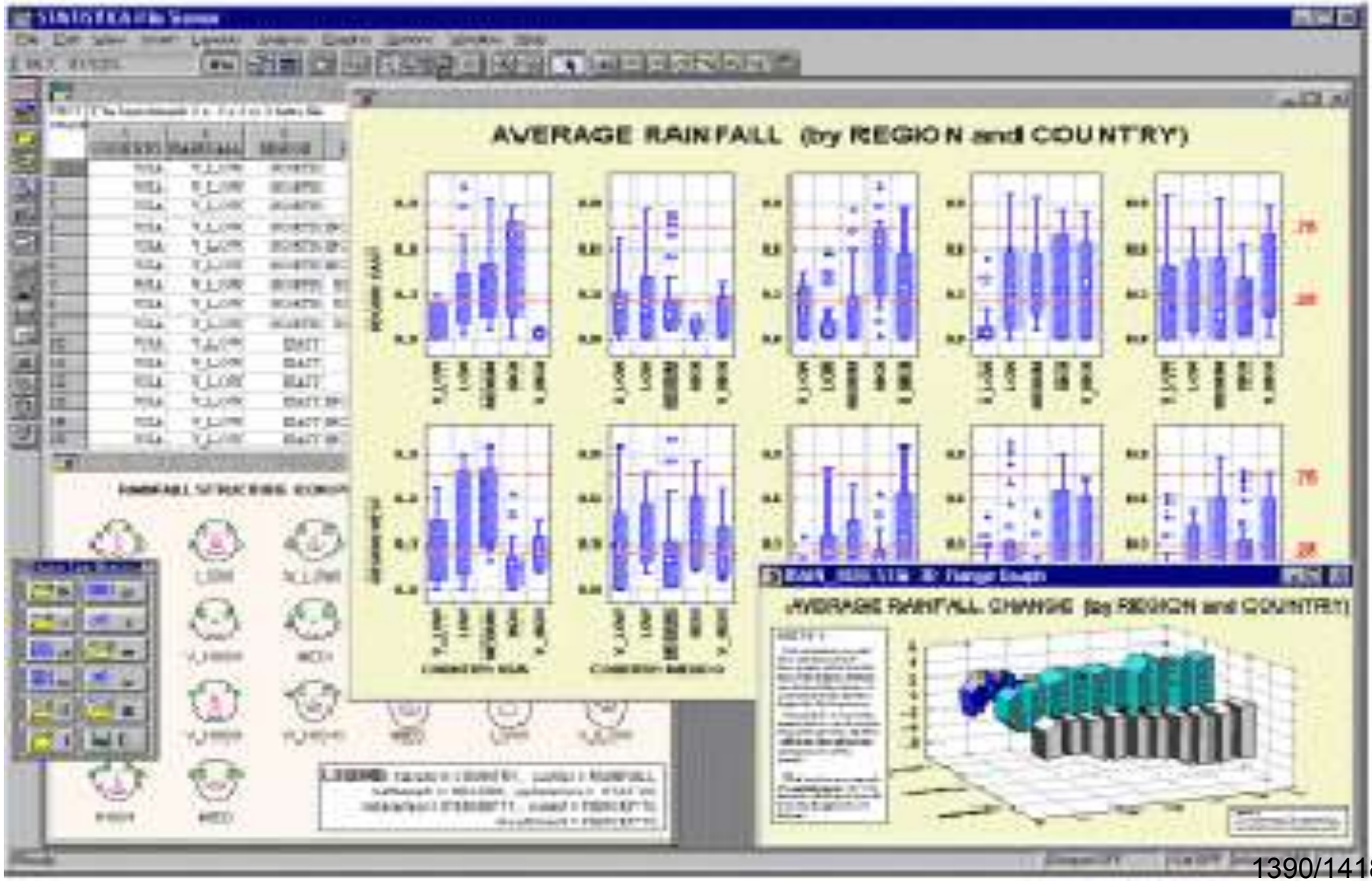
Visual Data Mining & Data Visualization

- Integration of visualization and data mining
  - data visualization
  - data mining result visualization
  - data mining process visualization
  - interactive visual data mining
- Data visualization
  - Data in a database or data warehouse can be viewed
    - at different levels of abstraction
    - as different combinations of attributes or dimensions
  - Data can be presented in various visual forms

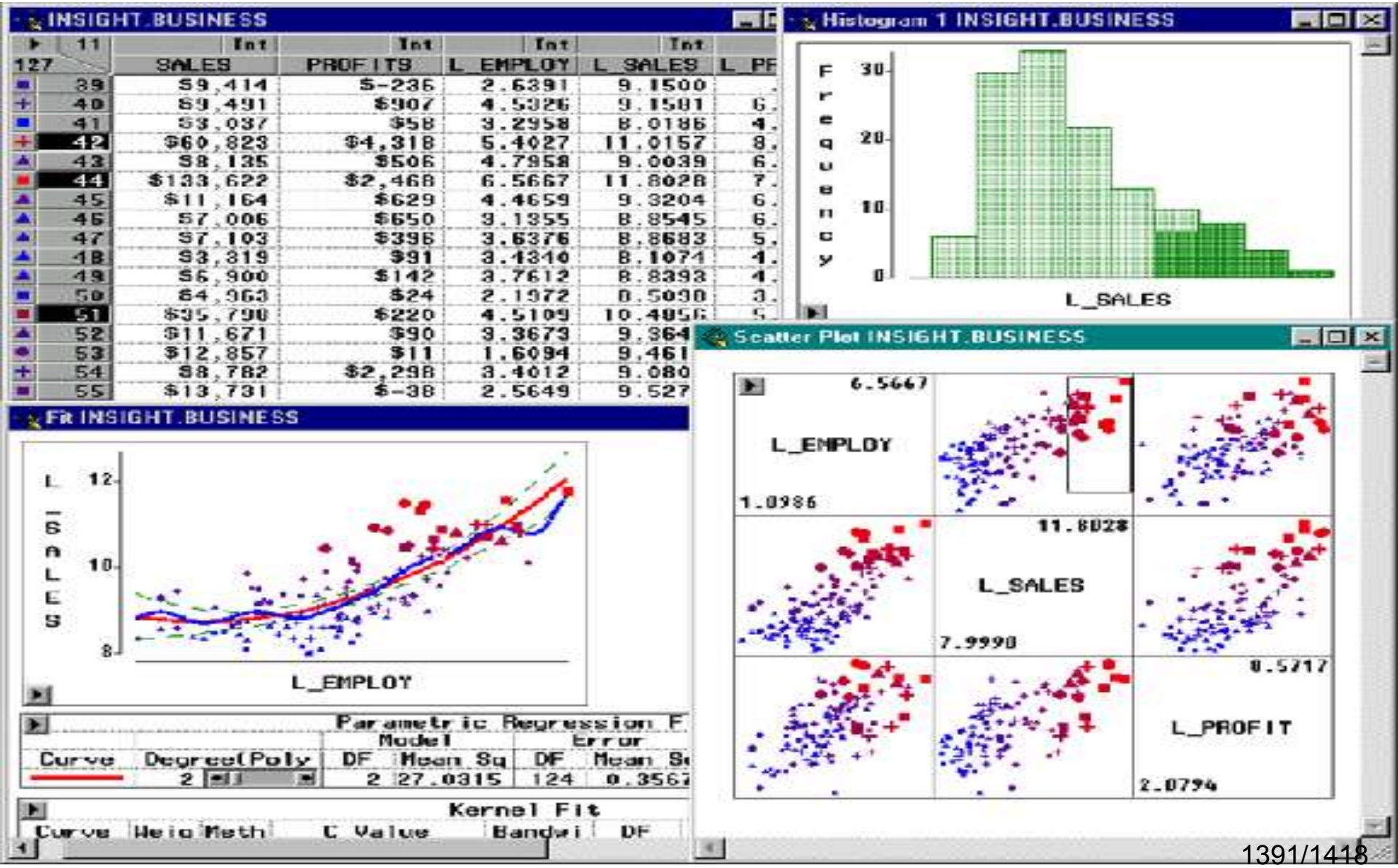
Data Mining Result Visualization

- Presentation of the results or knowledge obtained from data mining in visual forms
- Examples
  - Scatter plots and boxplots (obtained from descriptive data mining)
  - Decision trees
  - Association rules
  - Clusters
  - Outliers
  - Generalized rules

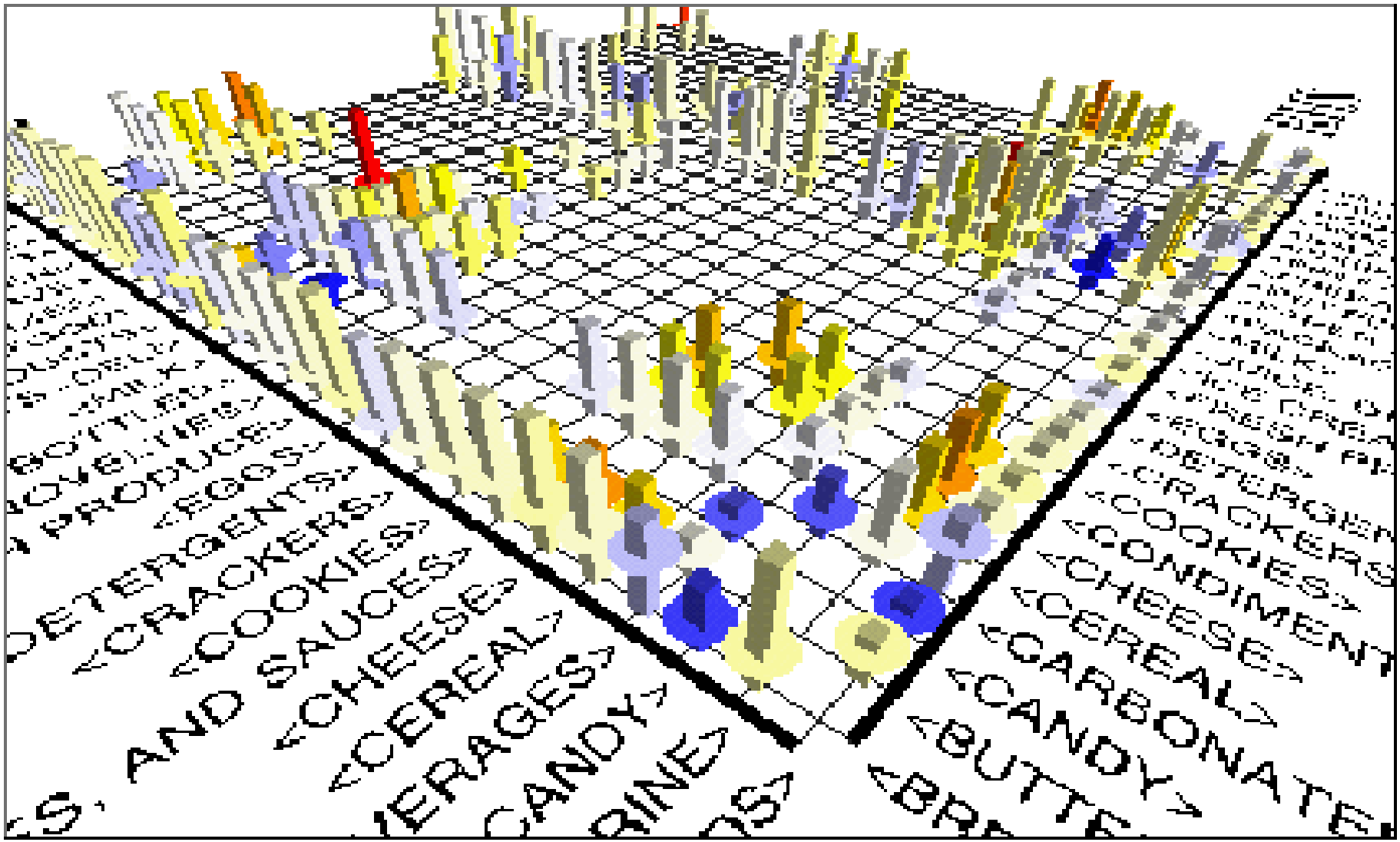
Boxplots from Statsoft: Multiple Variable Combinations



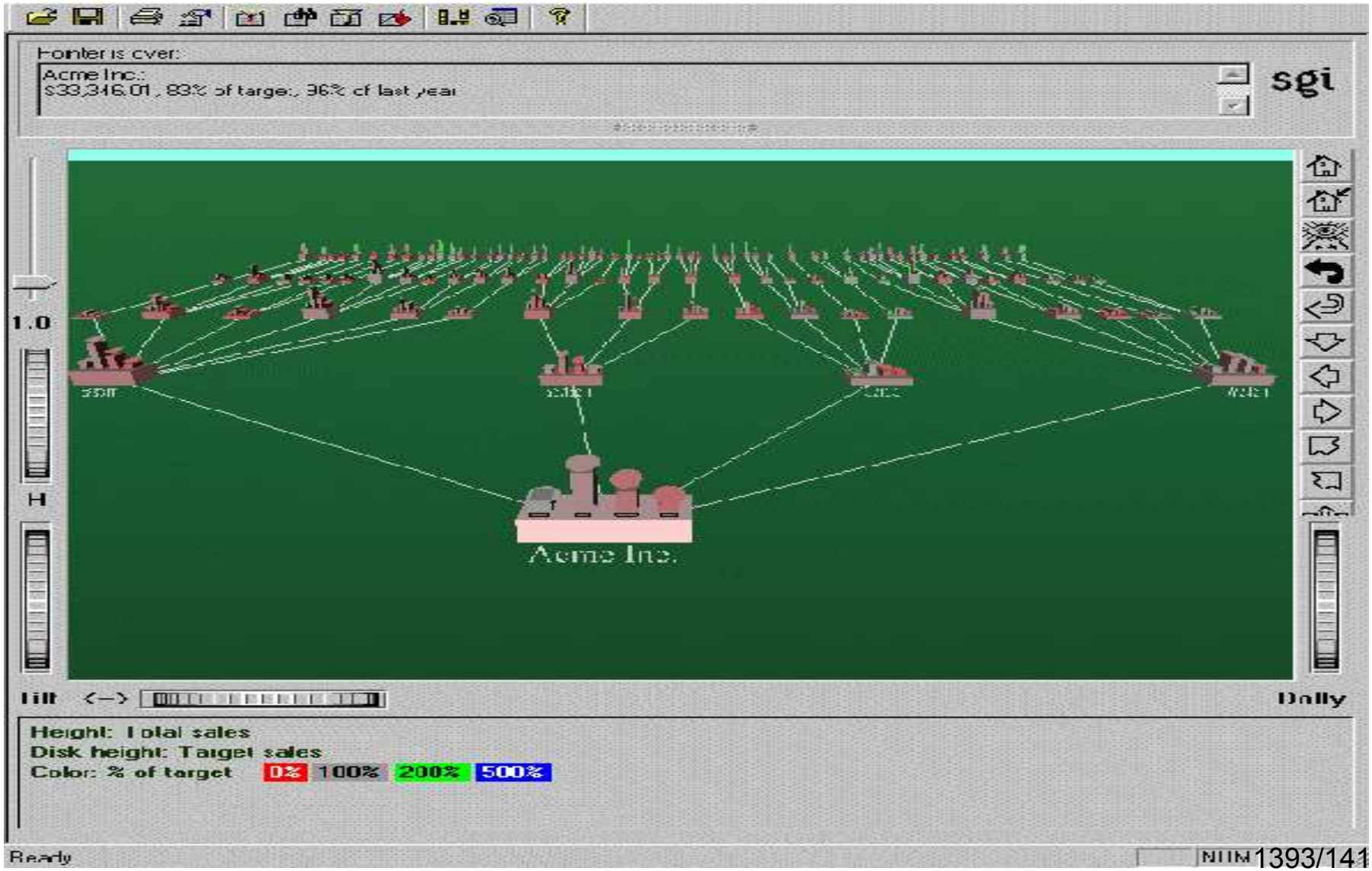
Visualization of Data Mining Results in SAS Enterprise Miner: Scatter Plots



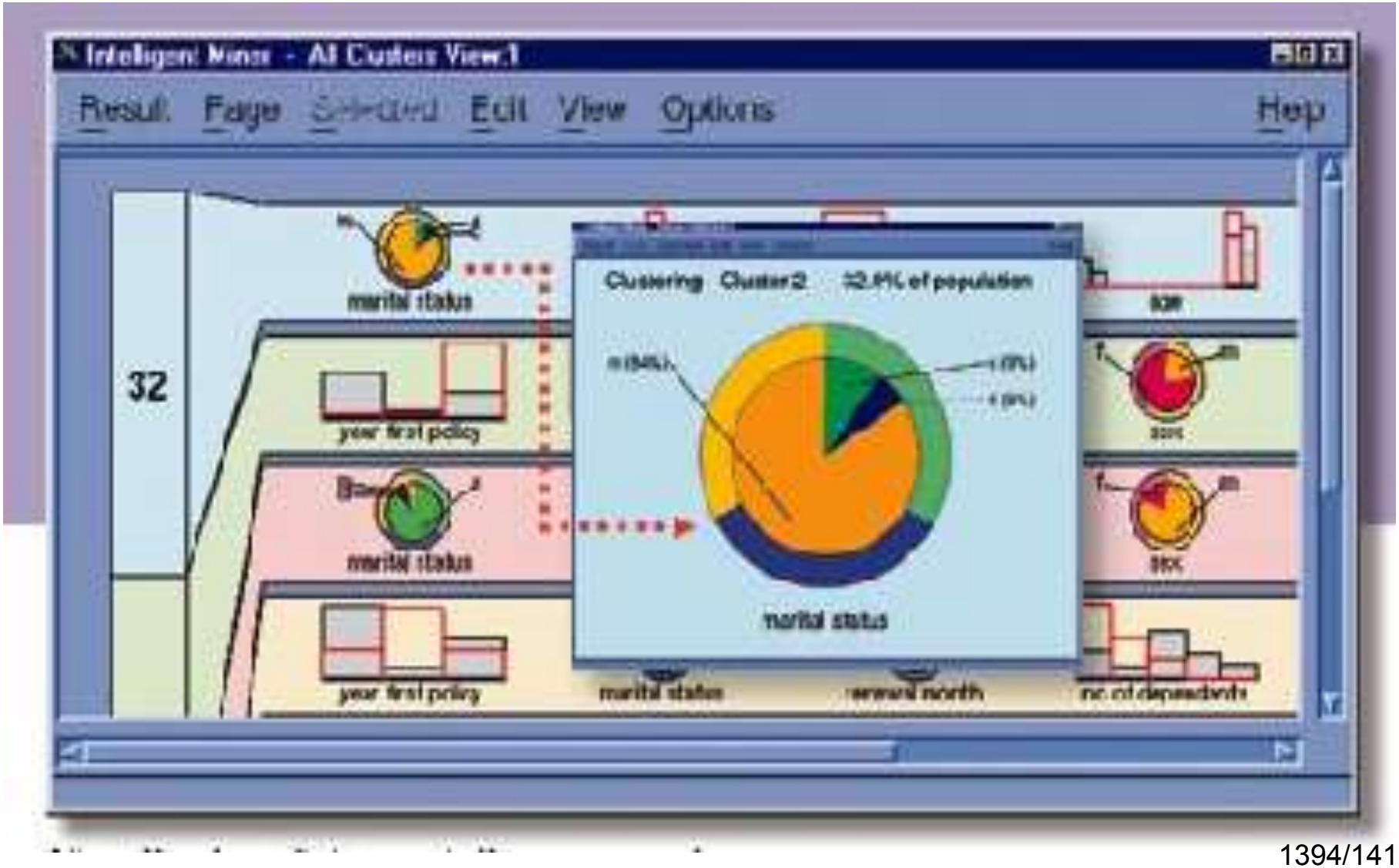
Visualization of Association Rules in SGI/MineSet 3.0



Visualization of a Decision Tree in SGI/MineSet 3.0



Visualization of Cluster Grouping in IBM Intelligent Miner



Data Mining Process Visualization

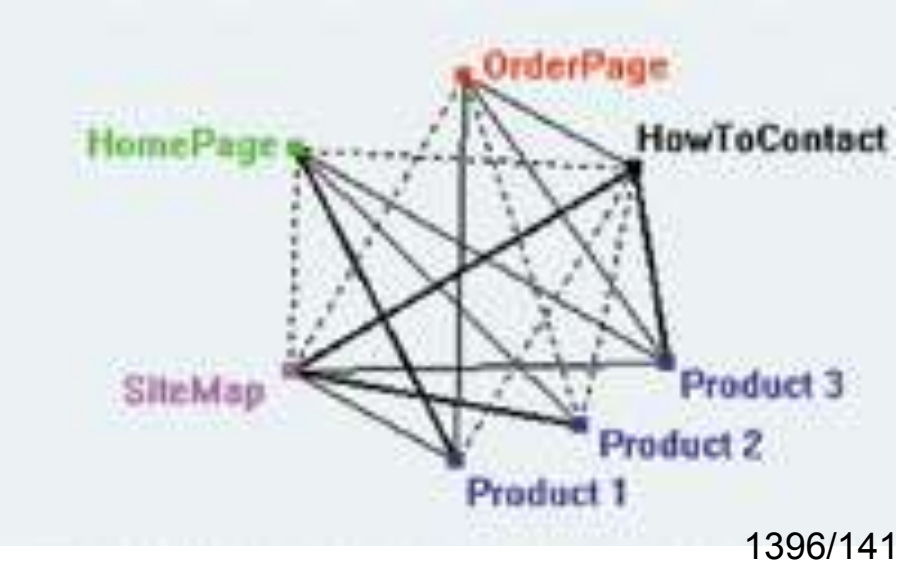
- Presentation of the various processes of data mining in visual forms so that users can see
  - Data extraction process
  - Where the data is extracted
  - How the data is cleaned, integrated, preprocessed, and mined
  - Method selected for data mining
  - Where the results are stored
  - How they may be viewed

Visualization of Data Mining Processes by Clementine



See your solution discovery process clearly

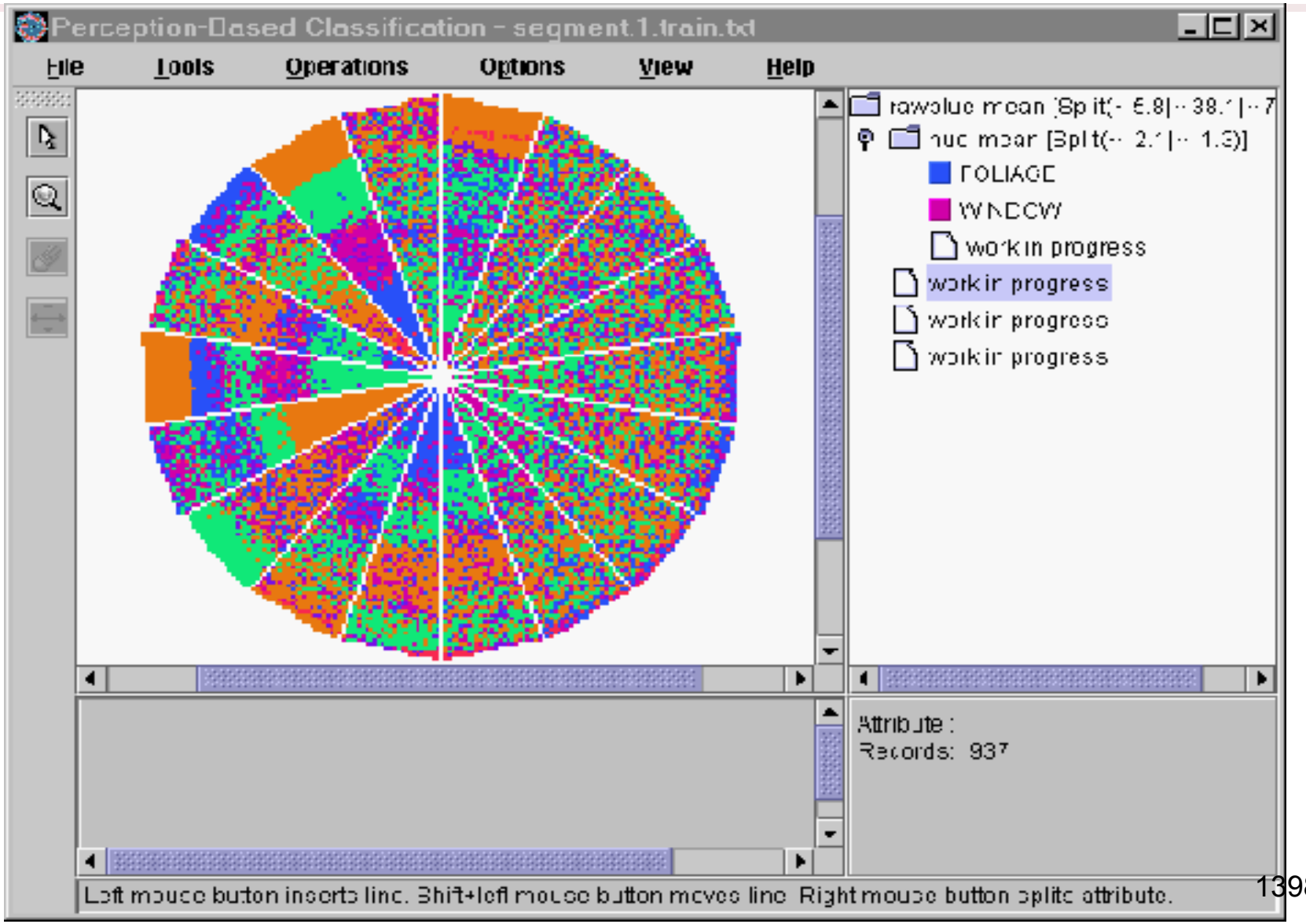
Understand variations with visualized data



Interactive Visual Data Mining

- Using visualization tools in the data mining process to help users make smart data mining decisions
- Example
  - Display the data distribution in a set of attributes using colored sectors or columns (depending on whether the whole space is represented by either a circle or a set of columns)
  - Use the display to which sector should first be selected for classification and where a good split point for this sector may be

Interactive Visual Mining by Perception-Based Classification (PBC)



Audio Data Mining

- Uses audio signals to indicate the patterns of data or the features of data mining results
- An interesting alternative to visual mining
- An inverse task of mining audio (such as music) databases which is to find patterns from audio data
- Visual data mining may disclose interesting patterns using graphical displays, but requires users to concentrate on watching patterns
- Instead, transform patterns into sound and music and listen to pitches, rhythms, tune, and melody in order to identify anything interesting or unusual

Outline

- Mining Complex Types of Data
- Other Methodologies of Data Mining
- Data Mining Applications
- Data Mining and Society
- Data Mining Trends
- Summary

Data Mining Applications

- Data mining: A young discipline with broad and diverse applications
  - There still exists a nontrivial gap between generic data mining methods and effective and scalable data mining tools for domain-specific applications
- Some application domains (briefly discussed here)
  - Data Mining for Financial data analysis
  - Data Mining for Retail and Telecommunication Industries
  - Data Mining in Science and Engineering
  - Data Mining for Intrusion Detection and Prevention
  - Data Mining and Recommender Systems

Data Mining for Financial Data Analysis (I)

- Financial data collected in banks and financial institutions are often relatively complete, reliable, and of high quality
- Design and construction of data warehouses for multidimensional data analysis and data mining
  - View the debt and revenue changes by month, by region, by sector, and by other factors
  - Access statistical information such as max, min, total, average, trend, etc.
- Loan payment prediction/consumer credit policy analysis
  - feature selection and attribute relevance ranking
  - Loan payment performance
  - Consumer credit rating

Data Mining for Financial Data Analysis (II)

- Classification and clustering of customers for targeted marketing
  - multidimensional segmentation by nearest-neighbor, classification, decision trees, etc. to identify customer groups or associate a new customer to an appropriate customer group
- Detection of money laundering and other financial crimes
  - integration of from multiple DBs (e.g., bank transactions, federal/state crime history DBs)
  - Tools: data visualization, linkage analysis, classification, clustering tools, outlier analysis, and sequential pattern analysis tools (find unusual access sequences)

Data Mining for Retail & Telcomm. Industries (I)

- Retail industry: huge amounts of data on sales, customer shopping history, e-commerce, etc.
- Applications of retail data mining
  - Identify customer buying behaviors
  - Discover customer shopping patterns and trends
  - Improve the quality of customer service
  - Achieve better customer retention and satisfaction
  - Enhance goods consumption ratios
  - Design more effective goods transportation and distribution policies
- Telcomm. and many other industries: Share many similar goals and expectations of retail data mining

Data Mining Practice for Retail Industry

- Design and construction of data warehouses
- Multidimensional analysis of sales, customers, products, time, and region
- Analysis of the effectiveness of sales campaigns
- Customer retention: Analysis of customer loyalty
  - Use customer loyalty card information to register sequences of purchases of particular customers
  - Use sequential pattern mining to investigate changes in customer consumption or loyalty
  - Suggest adjustments on the pricing and variety of goods
- Product recommendation and cross-reference of items
- Fraudulent analysis and the identification of usual patterns
- Use of visualization tools in data analysis

Data Mining in Science and Engineering

- Data warehouses and data preprocessing
  - Resolving inconsistencies or incompatible data collected in diverse environments and different periods (e.g. eco-system studies)
- Mining complex data types
  - Spatiotemporal, biological, diverse semantics and relationships
- Graph-based and network-based mining
  - Links, relationships, data flow, etc.
- Visualization tools and domain-specific knowledge
- Other issues
  - Data mining in social sciences and social studies: text and social media
  - Data mining in computer science: monitoring systems, software bugs, network intrusion

Data Mining for Intrusion Detection and Prevention

- Majority of intrusion detection and prevention systems use
  - Signature-based detection: use signatures, attack patterns that are preconfigured and predetermined by domain experts
  - Anomaly-based detection: build profiles (models of normal behavior) and detect those that are substantially deviate from the profiles
- What data mining can help
  - New data mining algorithms for intrusion detection
  - Association, correlation, and discriminative pattern analysis help select and build discriminative classifiers
  - Analysis of stream data: outlier detection, clustering, model shifting
  - Distributed data mining
  - Visualization and querying tools

Data Mining and Recommender Systems

- Recommender systems: Personalization, making product recommendations that are likely to be of interest to a user
- Approaches: Content-based, collaborative, or their hybrid
  - Content-based: Recommends items that are similar to items the user preferred or queried in the past
  - Collaborative filtering: Consider a user's social environment, opinions of other customers who have similar tastes or preferences
- Data mining and recommender systems
  - Users C × items S: extract from known to unknown ratings to predict user-item combinations
  - Memory-based method often uses k-nearest neighbor approach
  - Model-based method uses a collection of ratings to learn a model (e.g., probabilistic models, clustering, Bayesian networks, etc.)
  - Hybrid approaches integrate both to improve performance (e.g., using ensemble)

Chapter 13: Data Mining Trends and Research Frontiers

- Mining Complex Types of Data
- Other Methodologies of Data Mining
- Data Mining Applications
- Data Mining and Society
- Data Mining Trends
- Summary

Ubiquitous and Invisible Data Mining

- Ubiquitous Data Mining
  - Data mining is used everywhere, e.g., online shopping
  - Ex. Customer relationship management (CRM)
- Invisible Data Mining
  - Invisible: Data mining functions are built in daily life operations
  - Ex. Google search: Users may be unaware that they are examining results returned by data
  - Invisible data mining is highly desirable
  - Invisible mining needs to consider efficiency and scalability, user interaction, incorporation of background knowledge and visualization techniques, finding interesting patterns, real-time, …
  - Further work: Integration of data mining into existing business and scientific technologies to provide domain-specific data mining tools

Privacy, Security and Social Impacts of Data Mining

- Many data mining applications do not touch personal data
  - E.g., meteorology, astronomy, geography, geology, biology, and other scientific and engineering data
- Many DM studies are on developing scalable algorithms to find general or statistically significant patterns, not touching individuals
- The real privacy concern: unconstrained access of individual records, especially privacy-sensitive information
- Method 1: Removing sensitive IDs associated with the data
- Method 2: Data security-enhancing methods
  - Multi-level security model: permit to access to only authorized level
  - Encryption: e.g., *blind signatures*, *biometric encryption*, and *anonymous databases* (personal information is encrypted and stored at different locations)
- Method 3: Privacy-preserving data mining methods

Privacy-Preserving Data Mining

- Privacy-preserving (privacy-enhanced or privacy-sensitive) mining:
  - Obtaining valid mining results without disclosing the underlying sensitive data values
  - Often needs trade-off between information loss and privacy
- Privacy-preserving data mining methods:
  - Randomization (e.g., perturbation): Add noise to the data in order to mask some attribute values of records
  - K-anonymity and l-diversity: Alter individual records so that they cannot be uniquely identified
    - k-anonymity: Any given record maps onto at least k other records
    - l-diversity: enforcing intra-group diversity of sensitive values
  - Distributed privacy preservation: Data partitioned and distributed either horizontally, vertically, or a combination of both
  - Downgrading the effectiveness of data mining: The output of data mining may violate privacy
    - Modify data or mining results, e.g., hiding some association rules or slightly distorting some classification models

Outline

- Mining Complex Types of Data
- Other Methodologies of Data Mining
- Data Mining Applications
- Data Mining and Society
- **Data Mining Trends**
- Summary

Trends of Data Mining

- Application exploration: Dealing with application-specific problems
- Scalable and interactive data mining methods
- Integration of data mining with Web search engines, database systems, data warehouse systems and cloud computing systems
- Mining social and information networks
- Mining spatiotemporal, moving objects and cyber-physical systems
- Mining multimedia, text and web data
- Mining biological and biomedical data
- Data mining with software engineering and system engineering
- Visual and audio data mining
- Distributed data mining and real-time data stream mining
- Privacy protection and information security in data mining

Summary

- We present a high-level overview of mining complex data types
- Statistical data mining methods, such as regression, generalized linear models, analysis of variance, etc., are popularly adopted
- Researchers also try to build theoretical foundations for data mining
- Visual/audio data mining has been popular and effective
- Application-based mining integrates domain-specific knowledge with data analysis techniques and provide mission-specific solutions
- Ubiquitous data mining and invisible data mining are penetrating our data lives
- Privacy and data security are importance issues in data mining, and privacy-preserving data mining has been developed recently
- Our discussion on trends in data mining shows that data mining is a promising, young field, with great, strategic importance

Course Outline
(CS4.405: Data Analytics-I)

Prerequisite Course / Knowledge:

(i) Data and Applications, *or equivalent courses that cover Data modelling, normalization, SQL*

(ii) First courses on programming, data-structures and algorithms

(iii) Basics of Python language, to be able to use relevant libraries and toolkits for data analytics

Objective:

- In a computerized and networked society, vast amount of data is being collected every day in multiple domains.
- We are drowning in data, but starving for knowledge or actionable insights.
- Data mining or data analytics constitute a collection of concepts and algorithms, which are being developed to answer "how" questions by extracting interesting and useful knowledge of from large data.
- Data analytics based platforms are being operated in multiple domains to extract valuable and actionable insights from the data to improve the business performance.
- **The objective** of the course is to learn the important concepts and algorithms related to data analytics and data mining functionalities such as summarization, pattern mining, classification, clustering.

Detailed Syllabus

- Unit1:Introduction, data summarization through characterization, discrimination and data cube techniques (9 hours)
- Unit 2: Concepts and algorithms for mining patterns and associations (10 hours)
- Unit 3: Concepts and algorithms related to classification and regression (10 hours)
- Unit 4: Concepts and algorithms for clustering the data (10 hours)
- Unit 5: Outlier analysis and future trends. (3 hours)

Course Outcomes (COs)

- After completing the course successfully, the students are able to
  - CO-1. describe the concepts of data summarization, data warehousing, pattern mining, classification and clustering approaches
  - CO-2. perform the task of data summarization, pattern mining, classification and clustering based on the requirement.
  - CO-3. prescribe a single or a combination of data summarization, pattern mining, classification and clustering approaches for the problem scenario of a business/ organization.