Maharaja Education Trust (R), Mysuru

Maharaja Institute of Technology Mysore

Belawadi, Sriranga Pattana Taluk, Mandya – 571 477

Approved by AICTE, New Delhi,
Affiliated to VTU, Belagavi & Recognized by Government of Karnataka

Lecture Notes on

Data Mining and Data Warehousing (17CS651)

Prepared by

Department of Information Science & Engineering

Vision/ ಆಶಯ

"To be recognized as a premier technical and management institution promoting extensive education fostering research, innovation and entrepreneurial attitude"

□□□□□□□□, □□□□□□□□□ □□□□ □□□□□□□□□□□□□□□□□ □□□□□□□□□□□ □□□□□□□□□□ □□□□□□□□□ □□□□□ □□□□□□ □□□□□□□□ □□□□□□□ □□□□□□□□□□□ □□□□□□□□□□□□□□□□□□□□□.

Mission/ □□□□□□

- To empower students with indispensable knowledge through dedicated teaching and collaborative learning.

  □□□□□□□□ □□□□□□□□ □□□□□□ □□□□ □□□□□□□□□□□□ □□□□□□□□□□□□□□□□ □□□□□□□□□□□□□□□□□□ □□□□□□□□□□□□□ □□□□□□□□□□□□□□□□□□□□□□□□□.

- To advance extensive research in science, engineering and management disciplines.

  □□□□□□□□□□□, □□□□□□□□□□ □□□□□ □□□□□□ □□□□□□□□□ □□□□□□□□□□□□□□ □□□□□□□□□ □□□□□□□□□□□□□□□□ □□□□□□□□□□ □□□□□□□□□□□.

- To facilitate entrepreneurial skills through effective institute - industry collaboration and interaction with alumni.

  □□□□□□ □□□□□□□□□□□□□ □□□□□□, □□□□□□□□ □□□□□□ □□□□□□□□□□□□□□□□□□□□□□ □□□□□□□ □□□□□□□□□□□□ □□□□□□□□□□□□□□□□□□ □□□□□□□□□□□□□ □□□□□□□ □□□□□□□ □□□□□□□□□□□□□.

- To instill the need to uphold ethics in every aspect.

  □□□□□□□□□□□□ □□□□□□ □□□□□□□□□□□□□□ □□□□□□□□□□□□□□□□□□□□□□ □□□□□□□ □□□□□□□ □□□□□□ □□□□□□□□□□□□□.

- To mould holistic individuals capable of contributing to the advancement of the society.

  □□□□□□ □□□□□□□□□□□□□ □□□□□□ □□□□□□□ □□□□□□□□□ □□□□□□□□□□ □□□□□□□□□□□□□□□□□□□□ □□□□□□ □□□□□□□□□□□□□□ □□□□□□□□□□□□□.

Maharaja Institute of Technology Mysore

Department of Information Science & Engineering

Vision / ಆಶಯ

To be recognized as the best centre for technical education and research in the field of information science and engineering.

Mission / □□□□□

- To facilitate adequate transformation in students through a proficient teaching learning process with the guidance of mentors and all-inclusive professional activities.
- To infuse students with professional, ethical and leadership attributes through industry collaboration and alumni affiliation.
- To enhance research and entrepreneurship in associated domains and to facilitate real time problem solving.

.

Maharaja Institute of Technology Mysore

Department of Information Science & Engineering

Program Outcomes

1. **Engineering knowledge**: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis**: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **Design/development of solutions**: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. **Conduct investigations of complex problems**: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. **Modern tool usage**: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
6. **The engineer and society**: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environment and sustainability**: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. **Ethics**: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and team work**: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **Communication**: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. **Project management and finance**: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12. **Life-long learning**: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

Maharaja Institute of Technology Mysore

Department of Information Science & Engineering

Course Overview

Subject: Data Mining and Data Warehousing **Subject Code: 17CS651**

William H. Inmon first coined the concept of Data Warehouse (DW) in 1990. Inmon defined data warehouse as 'a subject-oriented, integrated, time-variant and non-volatile collection of data.' Extremely useful for Data Analysts, this data helps them to take business decisions and other data-related decisions in the organization. It provides the multidimensional view of consolidated data in a warehouse. Additionally, the data warehouse environment supports **ETL (Extraction, Transform and Load)** solutions, data mining capabilities, statistical analysis, reporting and **Online Analytical Processing (OLAP)** Tools, which help in interactive and efficient data analysis in a multifaceted view.

Data mining is the process of turning raw data into useful information. Any numbers, text, facts, web pages or documents that can be processed by a computer are considered data and mining is the process of extracting something useful. Hence, as the name indicates, data mining is the process of extracting useful information from large volumes of data. We will concentrate on discussing a few of the important problems in the topic of data mining. These problems include those of finding associations, clustering and classification. In this section, we will provide a brief introduction to each of these problems and elaborate on them in greater detail in later sections.

Associations: This problem often occurs in the process of finding relationships between different attributes in large customer databases. These attributes may either be 0-1 literals, or they may be quantitative. The idea in the association rule problem is to find the nature of the causalities between the values of the different attributes.

Classification is the process of learning a model that elucidates different predetermined classes of data. It is a two-step process, comprised of a **learning** step and a **classification** step. In learning step, a classification model is constructed and classification step the constructed model is used to **prefigure** the class labels for given data.

Clustering is a technique of organizing a group of data into classes and clusters where the objects reside inside a cluster will have high similarity and the objects of two clusters would be dissimilar to each other. Here the two clusters can be considered as disjoint. The main target of clustering is to divide the whole data into multiple clusters. Unlike classification process, here the class labels of objects are not known before, and clustering pertains to unsupervised learning.

Organizations today are under tremendous pressure to compete in an environment of tight deadlines and reduced profits. Business processes that require data to be extracted and manipulated prior to use will no longer be acceptable. Instead, enterprises need rapid decision support based on the analysis and forecasting

Course Objectives

The objectives of this course is to make students to learn-

1. Define multi-dimensional data models.
2. Explain rules related to association, classification and clustering analysis.
3. Compare and contrast between different classification and clustering algorithms

Course Outcomes

CO's	DESCRIPTION OF THE OUTCOMES
C651.1	Construct various architectures and main components of a data warehouse and address the issues that arise when implementing a data warehouse.
C651.2	Apply data mining techniques for extracting knowledge from a data warehouse.
C651.3	Compare and contrast various classifications and clustering algorithm to solve data mining problem.
C651.4	Analyze the suitability of association rules for a given data pattern.

Dr. Pushpa D	**Prof. Sneha D.P**	**Course Coordinator**

Facilitator	**NBA Coordinator**	**HOD**

Maharaja Institute of Technology Mysore

Department of Information Science & Engineering

Syllabus

Subject: Data Mining and Data Warehousing **Subject Code: 17CS651**

Syllabus	Teaching Hours
Module-1	
Data Warehousing & modelling: Basic Concepts: Data Warehousing: A multitier Architecture, Data warehouse models: Enterprise warehouse, Data mart and virtual warehouse, Extraction, Transformation and loading, Data Cube: A multidimensional data model, Stars, Snowflakes and Fact constellations: Schemas for multidimensional Data models, Dimensions: The role of concept Hierarchies, Measures: Their Categorization and computation, Typical OLAP Operations.	**08 Hours**
Module-2	
Data warehouse implementation& Data mining: Efficient Data Cube computation: An overview, Indexing OLAP Data: Bitmap index and join index, Efficient processing of OLAP Queries, OLAP server Architecture ROLAP versus MOLAP Versus HOLAP. **:** Introduction: What is data mining, Challenges, Data Mining Tasks, Data: Types of Data, Data Quality, Data Pre-processing, Measures of Similarity and Dissimilarity	**08 Hours**
Module-3	
Association Analysis: Association Analysis: Problem Definition, Frequent Item set Generation, Rule generation. Alternative Methods for Generating Frequent Item sets, FP-Growth Algorithm, Evaluation of Association Patterns.	**08 Hours**
Module-4	
Classification: Decision Trees Induction, Method for Comparing Classifiers, Rule Based Classifiers, Nearest Neighbor Classifiers, Bayesian Classifiers.	**08 Hours**
Module-5	
Clustering Analysis: Overview, K-Means, Agglomerative Hierarchical Clustering, DBSCAN, Cluster Evaluation, Density-Based Clustering, Graph-Based Clustering, Scalable Clustering Algorithms	**08 Hours**
List of Text Books	
1. Pang-Ning Tan, Michael Steinbach, Vipin Kumar: Introduction to Data Mining, Pearson, First impression, 2014. 2. Jiawei Han, Micheline Kamber, Jian Pei: Data Mining -Concepts and Techniques, 3rdEdition, Morgan Kaufmann Publisher, 2012.	
List of Reference Books	
1. Sam Anahory, Dennis Murray: Data Warehousing in the Real World, Pearson, Tenth Impression, 2012. 2. Michael.J.Berry, Gordon.S.Linoff: Mastering Data Mining, Wiley Edition, secondedtion, 2012.	

Maharaja Institute of Technology Mysore

Department of Information Science & Engineering

Index

Subject: Data Mining and Data Warehousing **Subject Code:17CS651**

Module-1

Data Warehouse: Basic Concepts

Data warehousing provides architectures and tools for business executives to systematically organize, understand, and use their data to make strategic decisions.

A data warehouse refers to a data repository that is maintained separately from an organization's operational databases.

Data warehouse systems allow for integration of a variety of application systems. They support information processing by providing a solid platform of consolidated historic data for analysis.

William H. Inmon states "A data warehouse is a subject-oriented, integrated, time-variant, and nonvolatile collection of data in support of management's decision making process".

Key Features of Data Warehouse: The key features of Data warehouse are as follows

Subject-oriented: A data warehouse is organized around major subjects such as customer, supplier, product, and sales rather than concentrating on day-to-day operations and Transaction processing of organization.

Data Warehouse focuses on modeling and analysis of data for decision making

Integrated: A data warehouse is usually constructed by integrating multiple heterogeneous sources, such as relational databases, flat files, and online transaction records.

"Data cleaning" and "Data integration" techniques are applied to ensure consistency in naming conventions, encoding structures, attribute measures, and so on.

Time-variant: Data are stored to provide information from an historic perspective (e.g., the past 5–10 years).

Every key structure in the data warehouse contains, either implicitly or explicitly, a time element.

Non-volatile: A data warehouse is always a physically separate store of data transformed from the application data found in operational environment.

Due to this separation, a data warehouse does not require transaction processing, recovery, and concurrency control mechanisms. It usually requires only two operations in data accessing: initial loading of data and access of data.

Differences between Operational Database Systems and Data Warehouses

The major task of online operational database systems is to perform online transaction and query processing. These systems are called **online transaction processing (OLTP)** systems. They cover most of the day-to-day operations of an organization such as purchasing, inventory, manufacturing, banking, payroll, registration, and accounting.

Data warehouse systems, on the other hand, serve users or knowledge workers in the role of data analysis and decision making. Such systems can organize and present data in various formats in order to accommodate the diverse needs of different users. These systems are known as **online analytical processing (OLAP)** systems.

The major distinguishing features of OLTP and OLAP are summarized as follows:

Table 4.1 Comparison of OLTP and OLAP Systems

Feature	*OLTP*	*OLAP*
Characteristic	operational processing	informational processing
Orientation	transaction	analysis
User	clerk, DBA, database professional	knowledge worker (e.g., manager, executive, analyst)
Function	day-to-day operations	long-term informational requirements decision support
DB design	ER-based, application-oriented	star/snowflake, subject-oriented
Data	current, guaranteed up-to-date	historic, accuracy maintained over time
Summarization	primitive, highly detailed	summarized, consolidated
View	detailed, flat relational	summarized, multidimensional
Unit of work	short, simple transaction	complex query
Access	read/write	mostly read
Focus	data in	information out
Operations	index/hash on primary key	lots of scans
Number of records accessed	tens	millions
Number of users	thousands	hundreds
DB size	GB to high-order GB	$\geq$ TB
Priority	high performance, high availability	high flexibility, end-user autonomy
Metric	transaction throughput	query throughput, response time

Data Warehousing: A Multitiered Architecture

Data warehouses often adopt a three-tier architecture such as

- Bottom Tier
- Middle Tier
- Top Tier

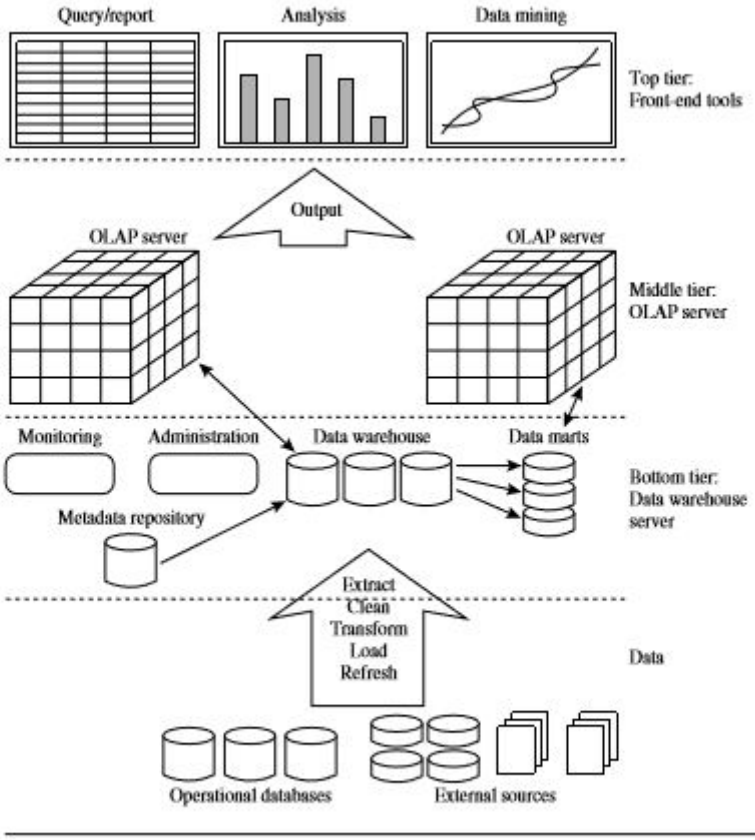


Figure 4.1 A three-tier data warehousing architecture.

1. The **bottom tier** is a warehouse database server that is almost always a relational database system. Back-end tools and utilities are used to feed data into the bottom tier from operational databases or other external sources (e.g., customer profile information provided by external consultants).These tools and utilities perform data extraction, cleaning, and transformation (e.g., to merge similar data from different sources into a unified format), as well as load and refresh functions to update the data warehouse

The data are extracted using application program interfaces known as gateways. A gateway is supported by the underlying DBMS and allows client programs to generate SQL code to be executed at a server. Examples of gateways include ODBC (Open Database Connection) and OLEDB (Object Linking and Embedding Database) by Microsoft and JDBC (Java Database Connection). This tier also contains a metadata repository, which stores information about the data warehouse and its contents

2. The **middle tier** is an OLAP server that is typically implemented using either

- A relational OLAP (ROLAP) model (i.e., an extended relational DBMS that maps operations on multidimensional data to standard relational operations);
- A multidimensional OLAP (MOLAP) model (i.e., a special-purpose server that directly implements multidimensional data and operations).

3. The **top tier** is a front-end client layer, which contains query and reporting tools, analysis tools, and/or data mining tools (e.g., trend analysis, prediction, and so on).

Data Warehouse Models:

Enterprise warehouse:

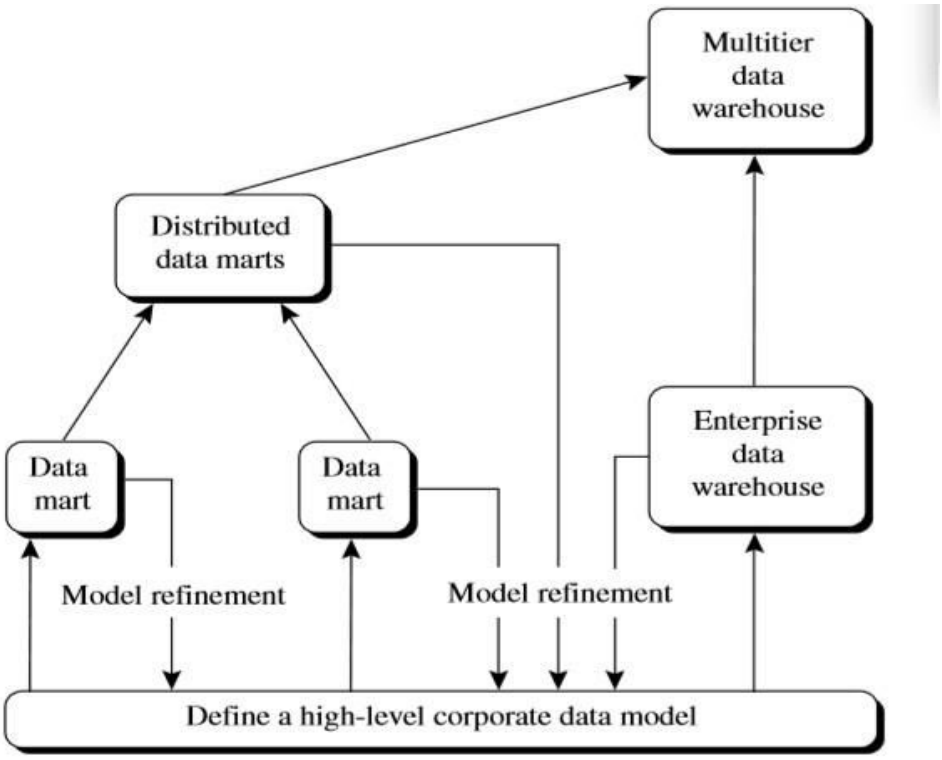
- An enterprise warehouse collects all of the information about subjects spanning the entire organization.
- It provides corporate-wide data integration, usually from one or more operational systems or external information providers, and is cross-functional in scope.
- It typically contains detailed data as well as summarized data, and can range in size from a few gigabytes to hundreds of gigabytes, terabytes, or beyond.
- An enterprise data warehouse may be implemented on traditional mainframes, computer super servers, or parallel architecture platforms. It requires extensive business modeling and may take years to design and build.

Datamart:

- A data mart contains a subset of corporate-wide data that is of value to a specific group of users. The scope is confined to specific selected subjects. For example, a marketing data mart may confine its subjects to customer, item, and sales. The data contained in data marts tend to be summarized.
- Data marts are usually implemented on low-cost departmental servers that are Unix/Linux or Windows based. The implementation cycle of a data mart is more likely to be measured in weeks rather than months or years. However, it may involve complex integration in the long run if its design and planning were not enterprise-wide.
- Depending on the source of data, data marts can be categorized as independent or dependent. Independent data marts are sourced from data captured from one or more operational systems or external information providers, or from data generated locally within a particular department or geographic area. Dependent data marts are sourced directly from enterprise data warehouses.

Virtual warehouse:

- A virtual warehouse is a set of views over operational databases. For efficient query processing, only some of the possible summary views may be materialized.
- A virtual warehouse is easy to build but requires excess capacity on operational database servers.



4.2 A recommended approach for data warehouse development.

A recommended method for the development of data warehouse systems is to implement the warehouse in an incremental and evolutionary manner, as shown in Figure4.2. First, a high-level corporate data model is defined within a reasonably short period (such as one or two months) that provides a corporate-wide, consistent, integrated view of data among different subjects and potential usages. This high-level model, although it will need to be refined in the further development of enterprise data warehouses and departmental data marts, will greatly reduce future integration problems. Second, independent data marts can be implemented in parallel with the enterprise warehouse based on the same corporate data model set noted before. Third, distributed data marts can be constructed to integrate different data marts via hub servers. Finally, a **multitier data warehouse** is constructed where the enterprise warehouse is the sole custodian of all warehouse data, which is then distributed to the various dependent data marts.

Extraction, Transformation, and Loading

Data warehouse systems use back-end tools and utilities to populate and refresh their data. These tools and utilities include the following functions:

- **Data extraction**, which typically gathers data from multiple, heterogeneous, and external sources.
- **Data cleaning ,** which detects errors in the data and rectifies them when possible.
- **Data transformation,** which converts data from legacy or host format to warehouse format.
- **Load,** which sorts, summarizes, consolidates, computes views, checks integrity, and builds indices and partitions.
- **Refresh,** which propagates the updates from the data sources to the warehouse.

Meta Data Repository:

Metadata are data about data. When used in a data warehouse, metadata are the data that define warehouse objects. . Metadata are created for the data names and definitions of the given warehouse. Additional metadata are created and captured for time stamping any extracted data, the source of the extracted data, and missing fields that have been added by data cleaning or integration processes.

A metadata repository should contain the following:

- A description of the structure of the data warehouse, which includes the warehouse schema, view, dimensions, hierarchies, and derived data definitions, as well as data mart locations and contents.
- *Operational metadata*, which include data lineage (history of migrated data and the sequence of transformations applied to it), currency of data (active, archived, or purged), and monitoring information (warehouse usage statistics, error reports, and audit trails).
- The *algorithms* used for summarization, which include measure and dimension definition algorithms, data on granularity, partitions, subject areas, aggregation, summarization, and predefined queries and reports.
- *The mapping from the operational environment to the data warehouse*, which includes source databases and their

contents, gateway descriptions, data partitions, data extraction, cleaning, transformation rules and defaults, data refresh and purging rules, and security (user authorization and access control).

- *Data related to system performance*, which include indices and profiles that improve data access and retrieval performance, in addition to rules for the timing and scheduling of refresh, update, and replication cycles.
- *Business metadata*, which include business terms and definitions, data ownership information, and charging policies.

Data Warehouse Modeling: Data Cube and OLAP

Data warehouses and OLAP tools are based on a **multidimensional data model**. This model views data in the form of a data cube.

Data Cube: A multidimensional Data model

- 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			
	home				home				home				home			
time	ent.	comp.	phone	sec.	ent.	comp.	phone	sec.	ent.	comp.	phone	sec.	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).

Given a set of dimensions, we can generate a cuboid for each of the possible subsets of the given dimensions. The result would form a lattice of cuboids, each showing the data at a different level of summarization, or group-by. The lattice of cuboids is then referred to as a data cube. Figure shows a lattice of cuboids forming a data cube for the dimensions time, item, location, and supplier. The cuboid that holds the lowest level of summarization is called the base cuboid.

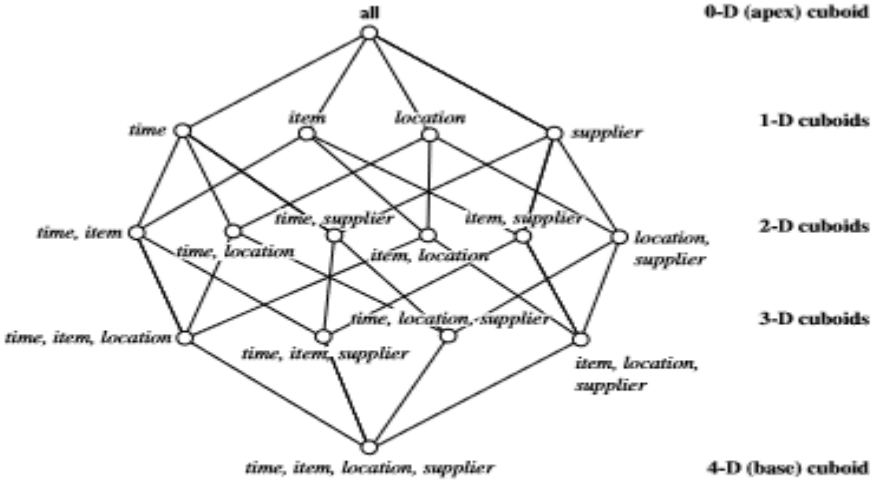


Figure 4.5 Lattice of cuboids, making up a 4-D data cube for *time*, *item*, *location*, and *supplier*. Each cuboid represents a different degree of summarization.

Stars, Snowflakes and Fact Constellations: Schemas for Multidimensional Data Models

The most popular data model for a data warehouse is a multidimensional model, which can exist in the form of a star schema, a snow flake schema, or a fact constellation schema**.**

Schemas for multidimensional data models

- 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

Star schema: The most common modeling paradigm is the star schema, in which the data warehouse contains (1) a large central table (fact table) containing the bulk of the data, with no redundancy, and (2) a set of smaller attendant tables (dimension tables), one for each dimension. The schema graph resembles a starburst, with the dimension tables displayed in a radial pattern around the central fact table.

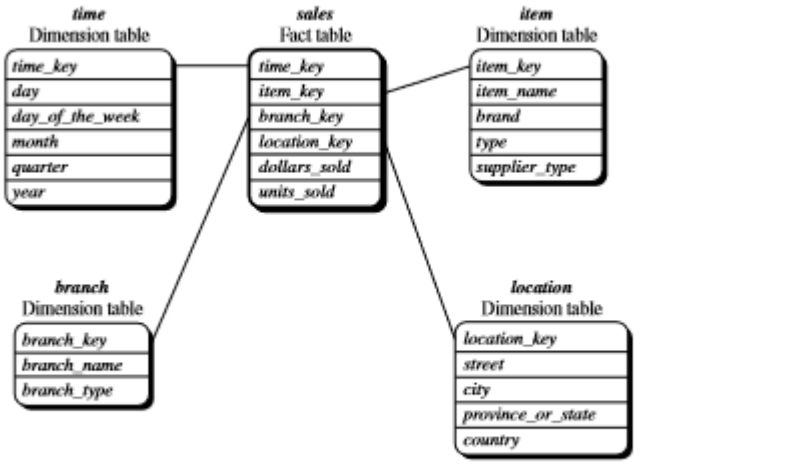


Figure 4.6 Star schema of *sales* data warehouse.

Snowflake schema: The snowflake schema is a variant of the star schema model, where some dimension tables are normalized, thereby further splitting the data into additional tables. The resulting schema graph forms a shape similar to a snowflake.

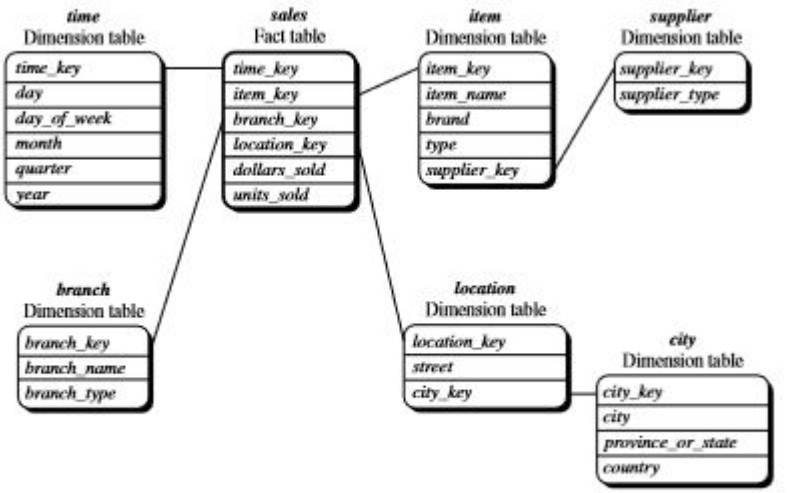


Figure 4.7 Snowflake schema of a *sales* data warehouse.

Fact constellation: Sophisticated applications may require multiple fact tables to share dimension tables. This kind of schema can be viewed as a collection of stars, and hence is called a galaxy schema or a fact constellation.

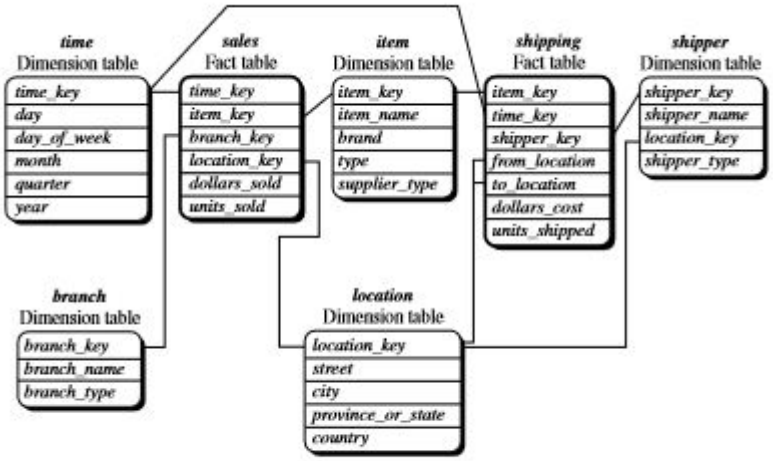


Figure 4.8 Fact constellation schema of a sales and shipping data warehouse.

Dimensions: The Role of Concept Hierarchies

A concept hierarchy defines a sequence of mappings from a set of low-level concepts to higher- level, more general concepts. Consider a concept hierarchy for the dimension location. City values for location include Vancouver, Toronto, New York, and Chicago. Each city, however, can be mapped to the province or state to which it belongs

For example, suppose that the dimension location is described by the attributes number, street, city, province or state, zip code, and country. These attributes are related by a total order, forming a concept hierarchy such as "street < city < province or state < country." This hierarchy is shown in Figure

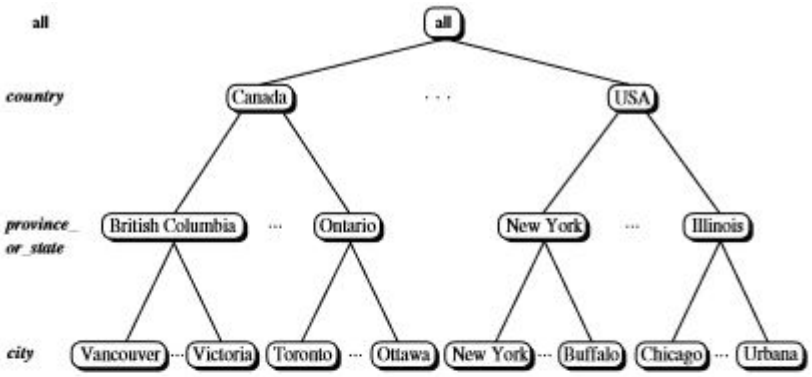


Figure 4.9 A concept hierarchy for *location*. Due to space limitations, not all of the hierarchy nodes are shown, indicated by ellipses between nodes.

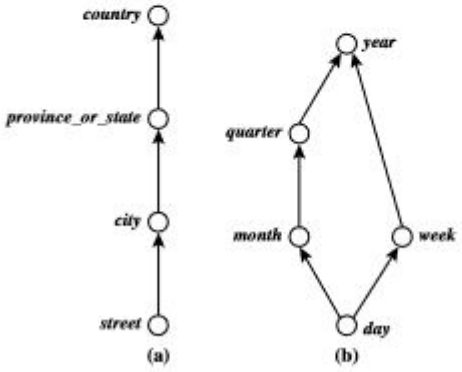


Figure 4.10 Hierarchical and lattice structures of attributes in warehouse dimensions: (a) a hierarchy for *location* and (b) a lattice for *time*.

The attributes of a dimension may be organized in a partial order, forming a lattice. An example of a partial order for the time dimension based on the attributes day, week, month, quarter, and year is "day < {month < quarter; week} < year."1 This lattice structure is shown in Figure 4.10(b). A concept hierarchy that is a total or partial order among attributes in a database schema is called a schema hierarchy

Concept hierarchies may also be defined by discretizing or grouping values for a given dimension or attribute, resulting in a set-grouping hierarchy. A total or partial order can be defined among groups of values. An example of a set-grouping hierarchy is shown in Figure 4.11 for the dimension price, where an interval ($X...$Y] denotes the range from $X (exclusive) to $Y (inclusive).

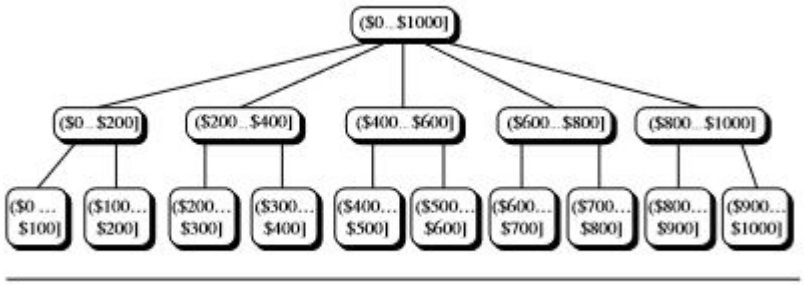


Figure 4.11 A concept hierarchy for *price*.

Measures: Their Categorization and Computation

- **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

  For example, avg() (average) can be computed by sum()/count(), where both sum() and count() are distributive aggregate functions. Similarly, it can be shown that min N () and max N() (which find the N minimum and N maximum values, respectively, in a given set

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

Typical OLAP Operations

- **ROLL-UP**

The roll-up operation (also called the drill-up operation by some vendors) performs aggregation on a data cube, either by climbing up a concept hierarchy for a dimension or by dimension reduction. Figure 4.12 shows the result of a roll-up operation performed on the central cube by climbing up the concept hierarchy for location given in Figure 4.9. This hierarchy was defined as the total order "street < city < province or state < country."

The roll-up operation shown aggregates the data by ascending the location hierarchy from the level of city to the level of country. In other words, rather than grouping the data by city, the resulting cube groups the data by country.

- **DRILL DOWN**

Drill-down is the reverse of roll-up. It navigates from less detailed data to more detailed data. Drill-down can be realized by either stepping down a concept hierarchy for a dimension or introducing additional dimensions.Figure4.12 shows the result of a drill-down operation performed on the central cube by stepping down a concept hierarchy for time defined as "day < month < quarter < year."

Drill-down occurs by descending the time hierarchy from the level of quarter to the more detailed level of month. The resulting data cube details the total sales per month rather than summarizing them by quarter.

Because a drill-down adds more detail to the given data, it can also be performed by adding new dimensions to a cube. For example, a drill-down on the central cube of Figure 4.12 can occur by introducing an additional dimension, such as customer group.

- **SLICE & DICE**

The **slice** operation performs a selection on one dimension of the given cube, resulting in a subcube. Figure 4.12 shows a slice operation where the sales data are selected from the central cube for the dimension time using the criterion time="Q1."

The **dice** operation defines a subcube by performing a selection on two. Figure4.12 shows a dice operation on the central cube based on the following selection criteria that involve three dimensions: (location="Toronto" or "Vancouver") and (time="Q1" or "Q2") and (item="home entertainment" or "computer").

- **PIVOT (OR ROTATE)**

Pivot (also called rotate) is a visualization operation that rotates the data axes in view to provide an alternative data presentation. Figure 4.12 shows a pivot operation where the item and location axes in a 2-D slice are rotated. Other examples include rotating the axes in a 3-D cube, or transforming a 3-D cube into a series of 2-D planes.

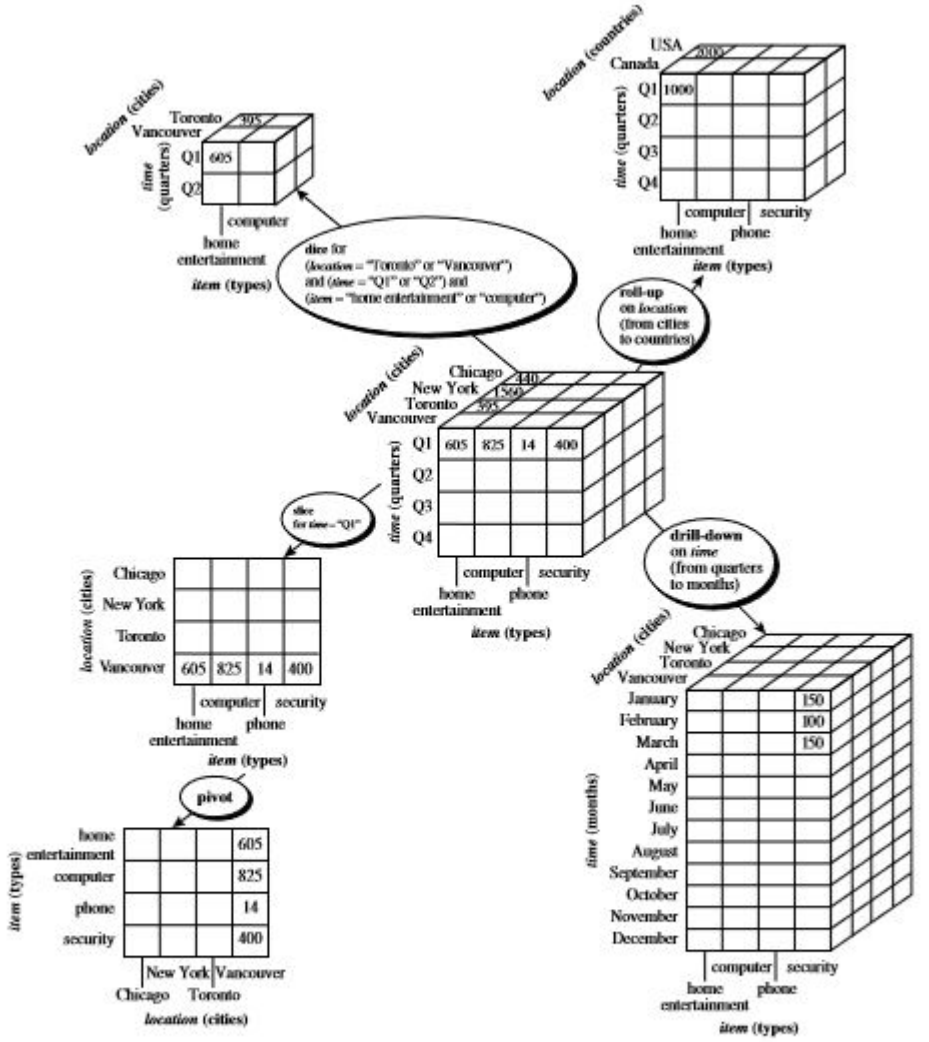


Figure 4.12 Examples of typical OLAP operations on multidimensional data.

Module-2

Why Mine Data?

Commercial Viewpoint:

Lots of data is being collected and warehoused

- Web data, e-commerce
- purchases at department/grocery stores
- Bank/Credit Card transactions

Data mining techniques can be used to support a wide range of business intelligence applications such as customer profiling, targeted marketing, workflow management, store layout, and fraud detection.

It can also help retailers answer important business questions such as

- "Who are the most profitable customers?"
- "What products can be cross-sold or up-sold?"
- "What is the revenue outlook of the company for next year?

Scientific Viewpoint

Data is collected and stored at enormous speeds (GB/hour). E.g.

- remote sensors on a satellite
- telescopes scanning the skies
- scientific simulations

As an important step toward improving our understanding of the Earth's climate system, NASA has deployed a series of Earth orbiting satellites that continuously generate global observations of the Land surface, oceans, and atmosphere.

Techniques developed in data mining can aid Earth scientists in answering questions such as

- "What is the relationship between the frequency and intensity of ecosystem disturbances such as droughts and hurricanes to global warming?"
- "How is land surface precipitation and temperature affected by ocean surface temperature?"
- "How well can we predict the beginning and end of the growing season for a region?"

What Is Data Mining?(Definition)

Data mining is the process of automatically discovering useful information in large data repositories.

- Finding hidden information in a database
  - Data mining techniques are deployed to scour large databases in order to find novel and useful patterns that might otherwise remain unknown. They also provide capabilities to predict
  - outcome of a future observation.

Applications of data mining

- Banking: loan/credit card approval
  - Given a database of 100,000 names, which persons are the least likely to default on their credit cards?
- Customer relationship management:
  - Which of my customers are likely to be the most loyal, and which are most likely to leave for a competitor?
- Targeted marketing:
  - identify likely responders to promotions
- Fraud detection: telecommunications, financial transactions
  - from an online stream of event identify fraudulent events
- Manufacturing and production:
  - automatically adjust knobs when process parameter changes
- Medicine: disease outcome, effectiveness of treatments
  - analyze patient disease history: find relationship between diseases
- Molecular/Pharmaceutical: identify new drugs
- Scientific data analysis:
  - identify new galaxies by searching for sub clusters
- Web site/store design and promotion:
  - find affinity of visitor to pages and modify layout

What is not Data Mining?

- Find all credit applicants with last name of Smith.
- Identify customers who have purchased more than $10,000 in the last month.
- Find all customers who have purchased milk
- Looking up individual records using a database management system Look up

phone number in phone directory.
- finding particular Web pages via a query to an Internet search engine.

What is Data Mining?

- Find all credit applicants who are poor credit risks. (classification)
- Identify customers with similar buying habits. (Clustering)
- Find all items which are frequently purchased with milk. (association rules)
- Discover groups of similar documents on the Web
- Certain names are more popular in certain locations

Data Mining and Knowledge Discovery

Data mining is an integral part of knowledge discovery in databases (KDD), which is the overall process of converting raw data into useful information,

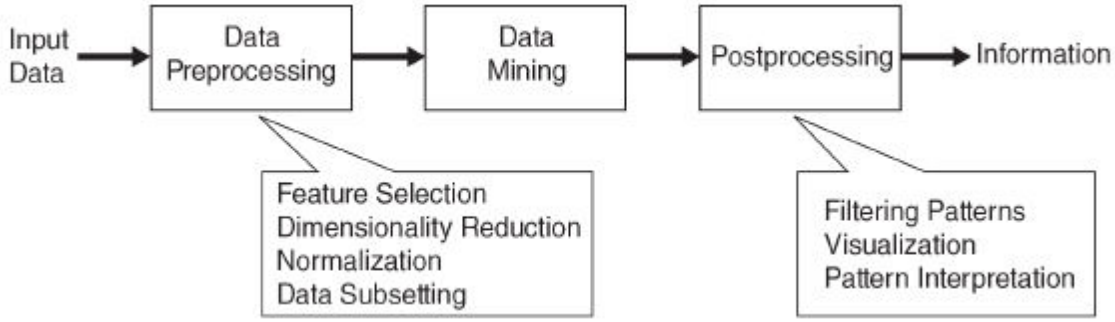


Figure 2.1 KDD

Preprocessing

- The input data can be stored in a variety of formats (flat files, spreadsheets, or relational tables) and may reside in a centralized data repository or be distributed across multiple sites.

- The purpose of preprocessing is to transform the raw input data into an appropriate format for subsequent analysis.
- Data preprocessing include fusing data from multiple sources, cleaning data to remove noise and duplicate observations, and selecting records and features that are relevant to the data mining task at hand.

Post processing:

- Ensures that only valid and useful results are incorporated into the system.
- Which allows analysts to explore the data and the data mining results from a variety of viewpoints.
- Testing methods can also be applied during post processing to eliminate spurious data mining results.

Motivating Challenges

The following are some of the specific challenges that motivated the development of data mining.

Scalability :

Because of advances in data generation and collection, data sets with sizes of gigabytes, terabytes, or even petabytes are becoming common.

If data mining algorithms are to handle these massive data sets, then they must be scalable.

Scalability may also require the implementation of novel data structures to access individual records in an efficient manner.

High Dimensionality:

It is now common to encounter data sets with hundreds or thousands of attributes.

Traditional data analysis techniques that were developed for low-dimensional data often do not work well for such high dimensional data.

Heterogeneous and Complex Data:

Recent years have also seen the emergence of more complex data objects, such as

- Collections of Web pages containing semi-structured text and hyperlinks;
- DNA data with sequential and three-dimensional structure;
- climate data that consists of time series measurements (temperature, pressure, etc.) various locations on the Earth's surface .

Techniques developed for mining such complex objects should take into consideration relationships in the data, such as temporal and spatial autocorrelation, graph connectivity, and parent-child relationships between the elements in semi-structured text.

Data ownership and Distribution:

Sometimes, the data needed for an analysis is not stored in one location or owned by one organization. Instead, the data is geographically distributed among resources belonging to multiple entities. This requires the development of distributed data mining techniques.

Among the key challenges faced by distributed data mining algorithms include

(1) How to reduce the amount of communication needed to perform the distributed computation.

(2)How to effectively consolidate the data mining results obtained from multiple sources.

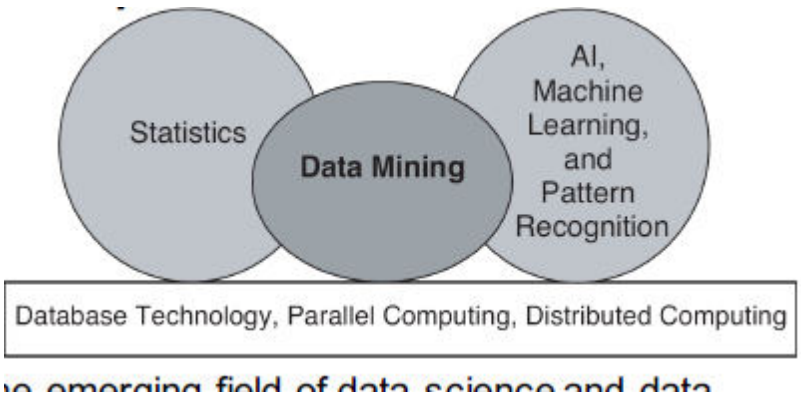
(3) How to address data security issues.

Origins of Data Mining

Draws ideas from machine learning/AI, pattern recognition, statistics, and database systems

Traditional Techniques may be unsuitable due to Enormity of data, High dimensionality of data, Heterogeneous, distributed nature of data.

Figure 2.2 shows the relationship of data mining to other areas.



ve emerging field of data science and data

Data Mining Tasks

Two broad categories

1) Prediction Methods*:* Use some variables to predict unknown or future values of other variables.

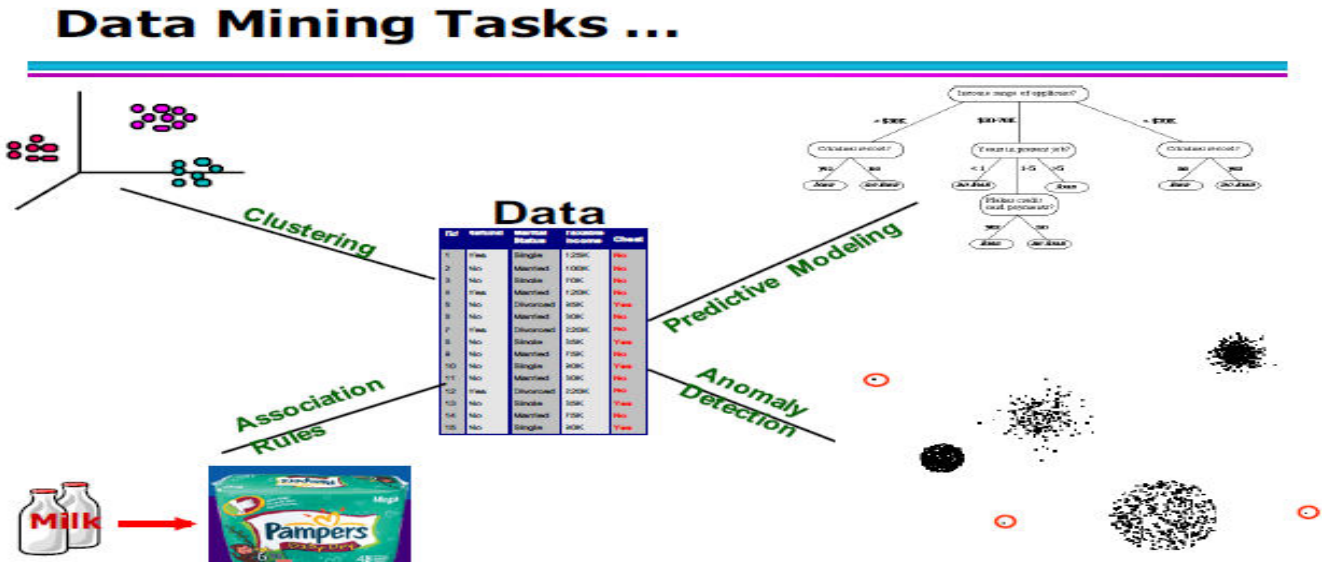
2) Description Methods *:*Find human-interpretable patterns that describe the data

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

Classification: Definition

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

Figure 2.3



Classification: Application 1

Direct Marketing

Goal: Reduce cost of mailing by targeting a set of consumers likely to buy a new cell-phone product. Approach**:**

- ✓ Use the data for a similar product introduced before.

We know which customers decided to buy and which decided otherwise. This **{buy, don't buy}** decision forms the class attribute.

- ✓ Collect various demographic, lifestyle, and company-interaction related information about all such customers, such as type of business, where they stay, how much they earn, etc.
- ✓ Use this information as input attributes to learn a classifier model.

Classification: Application 2

Fraud Detection

Goal: Predict fraudulent cases in credit card transactions.

Approach:

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

Classification: Application 3

Customer Attrition/Churn:

Goal: To predict whether a customer is likely to be lost to a competitor.

Approach:

- ✓ Use detailed record of transactions with each of the past and present customers, to find attributes.
- ✓ How often the customer calls, where he calls, what time-of-the day he calls most, his financial status, marital status, etc.
- ✓ Label the customers as loyal or disloyal.
- ✓ Find a model for loyalty.

Clustering: Definition

Given a set of data points, each having a set of attributes, and a similarity measure among them, find clusters such that

- ✓ Data points in one cluster are more similar to one another.
- ✓ Data points in separate clusters are less similar to one another.
- ✓ Similarity Measures:
  - Euclidean Distance if attributes are continuous.
  - Other Problem-specific Measures.

Clustering: Application 1

Market Segmentation:

Goal: Subdivide a market into distinct subsets of customers where any subset may conceivably be selected as a market target to be reached with a distinct marketing mix.

Approach:

- ✓ Collect different attributes of customers based on their geographical and lifestyle related information.
- ✓ Find clusters of similar customers.
- ✓ Measure the clustering quality by observing buying patterns of customers in same cluster vs. those from different clusters.

Clustering: Application 2

Document Clustering

Goal: To find groups of documents that are similar to each other based on the important terms appearing in them.

Approach:

- ✓ To identify frequently occurring terms in each document. Form a similarity measure based on the frequencies of different terms. Use it to cluster.

Gain: Information Retrieval can utilize the clusters to relate a new document or search term to clustered documents

Association Rule Discovery: Definition

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

other items

Figure 2.4

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

Rules Discovered:

{Milk} --> {Coke}

{Diaper, Milk} --> {Beer}

Association Rule Discovery: Application 1

***Marketing and Sales Promotion*:**

Let the rule discovered be

{Bagels, … } --> {Potato Chips}

(antecedent) (consequent)

- ✓ Potato Chips as consequent => Can be used to determine what should be done to boost its sales.
- ✓ Bagels in the antecedent => Can be used to see which products would be affected if the store discontinues selling bagels.

Bagels in antecedent and Potato chips in consequent

=> Can be used to see what products should be sold with Bagels to promote sale of Potato chips!

Association Rule Discovery: Application 2

Supermarket shelf management

Goal: To identify items that are bought together by sufficiently many customers.

Approach:

- ✓ Process the point-of-sale data collected with barcode scanners to find dependencies among items.
- ✓ A classic rule --
  - If a customer buys diaper and milk, then he is very likely to buy beer.
  - So, don't be surprised if you find six-packs stacked next to diapers!

Regression: Definition

Predict a value of a given continuous valued variable based on the values of other variables,

assuming a linear or nonlinear model of dependency.

Greatly studied in statistics and neural network fields.

Examples:

- ✓ Predicting sales amounts of new product based on advertising expenditure.
- ✓ Predicting wind velocities as a function of temperature, humidity, air pressure, etc.
- ✓ Time series prediction of stock market indices.

Deviation/Anomaly Detection:Definition

Detect significant deviations from normal behavior

Applications: Credit Card Fraud Detection

Network Intrusion Detection

Problems: Discuss whether or not each of the following activities is a data mining task.

(a) Dividing the customers of a company according to their gender.

No. This is a simple database query.

(b) Dividing the customers of a company according to their profitability.

No. This is an accounting calculation, followed by the application of a threshold. However, predicting the profitability of a new customer would be data mining.

(c) Computing the total sales of a company.

No. Again, this is simple accounting.

(d) Sorting a student database based on student identification numbers.

No. Again, this is a simple database query.

(e) Predicting the outcomes of tossing a (fair) pair of dice.

No. Since the die is fair, this is a probability calculation. If the die were not fair, and we needed to estimate the probabilities of each outcome from the data, then this is more like the problems considered by data mining. However, in this specific case, solutions to this problem were developed by mathematicians a long time ago, and thus, we wouldn't consider it to be data mining.

(f) Predicting the future stock price of a company using historical records.

Yes. We would attempt to create a model that can predict the continuous value of the stock price. This is an example of the area of data mining known as predictive modeling. We could use regression for this modeling, although researchers in many fields have developed a wide variety of techniques for predicting time series.

(g) Monitoring the heart rate of a patient for abnormalities.

Yes. We would build a model of the normal behavior of heart rate and raise an alarm when an unusual heart behavior occurred. This would involve the area of data mining known as anomaly detection. This could also be considered as a classification problem if we had examples of both normal and abnormal heart behavior.

(h) Monitoring seismic waves for earthquake activities.

Yes. In this case, we would build a model of different types of seismic wave behavior associated with earthquake activities and raise an alarm when one of these different types of seismic activity was observed. This is an example of the area of data mining known as classification.

(i) Extracting the frequencies of a sound wave.

No. This is signal processing.

What is Data?

- ✓ Collection of data objects and their attributes
- ✓ An attribute is a property or characteristic of an object
  - Examples: eye color of a person, temperature, etc.
  - Attribute is also known as variable, field, characteristic, or feature
- ✓ A collection of attributes describe an object
  - Object is also known as record, point, case, sample, entity, or instance

Attribute Values

- ✓ Attribute values are numbers or symbols assigned to an attribute
- ✓ Distinction between attributes and attribute values
  - Example: height can be measured in feet or meters
- ✓ Different attributes can be mapped to the same set of values
  - Example: Attribute values for ID and age are integers
- ✓ But properties of attribute values can be different
  - ID has no limit but age has a maximum and minimum value

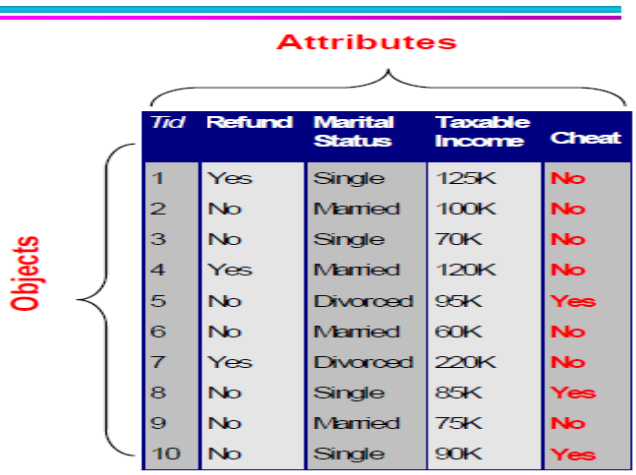
Types of Attributes

There are different types of attributes

- ✓ Nominal

- Examples: ID numbers, eye color, zip codes

Figure 2.5:-Attributes



✓ Ordinal

- Examples: rankings (e.g., taste of potato chips on a scale from 1-10), grades, height in {tall, medium, short}

✓ Interval

- Examples: calendar dates, temperatures in Celsius or Fahrenheit.

✓ Ratio

- Examples: temperature in Kelvin, length, time, counts

Properties of Attribute Values

- The type of an attribute depends on which of the following properties/operations it possesses:
  - Distinctness: = ≠
  - Order: < >
  - Differences are meaningful : + -
  - Ratios are meaningful * /
  - Nominal attribute: distinctness
  - Ordinal attribute: distinctness & order
  - Interval attribute: distinctness, order & meaningful differences

Ratio attribute: all 4 properties/operations

Difference Between Ratio and Interval

- Is it physically meaningful to say that a temperature of 10 ° is twice that of 5° on
  - the Celsius scale?
  - the Fahrenheit scale?

- the Kelvin scale?
- Consider measuring the height above average
- If Bill's height is three inches above average and Bob's height is six inches above average, then would we say that Bob is twice as tall as Bill?
- Is this situation analogous to that of temperature?

	Attribute Type	Description	Examples	Operations
Categorical Qualitative	Nominal	Nominal attribute values only distinguish. (=, ≠)	zip codes, employee ID numbers, eye color, sex: {*male, female*}	mode, entropy, contingency correlation, $\chi 2$ test
	Ordinal	Ordinal attribute values also order objects. (<, >)	hardness of minerals, {*good, better, best*}, grades, street numbers	median, percentiles, rank correlation, run tests, sign tests
Numeric Quantitative	Interval	For interval attributes, differences between values are meaningful. (+, -)	calendar dates, temperature in Celsius or Fahrenheit	mean, standard deviation, Pearson's correlation, *t* and *F* tests
	Ratio	For ratio variables, both differences and ratios are meaningful. (*, /)	temperature in Kelvin, monetary quantities, counts, age, mass, length, current	geometric mean, harmonic mean, percent variation

	Attribute Type	Transformation	Comments
Categorical Qualitative	Nominal	Any permutation of values	If all employee ID numbers were reassigned, would it make any difference?
	Ordinal	An order preserving change of values, i.e., *new_value = f(old_value)* where *f* is a monotonic function	An attribute encompassing the notion of good, better best can be represented equally well by the values {1, 2, 3} or by { 0.5, 1, 10}.
Numeric Quantitative	Interval	*new_value = a * old_value + b* where a and b are constants	Thus, the Fahrenheit and Celsius temperature scales differ in terms of where their zero value is and the size of a unit (degree).
	Ratio	*new_value = a * old_value*	Length can be measured in meters or feet.

Describing Attribute by the number of values

An independent way of distinguishing between attributes is by the number of values they can take

Discrete Attribute

Has only a finite or countable infinite set of values

- ✓ Examples: zip codes, counts, or the set of words in a collection of documents .
- ✓ Often represented as integer variables.
- ✓ Note: binary attributes are a special case of discrete attributes

Continuous Attribute

Has real numbers as attribute values

- ✓ Examples: temperature, height, or weight.
- ✓ Practically, real values can only be measured and represented using a finite number of digits.
- ✓ Continuous attributes are typically represented as floating-point variables.

Types of data sets

Record

- ✓ Collection of records , each of which consists of a fixed set of data fields (attributes).
- ✓ For the most basic form of record data, there is no explicit relationship among records every record (object) has the same set of attributes.
- ✓ Record data is usually stored either in flat files or in relational databases
- ✓ Example:
  - o Data Matrix
  - o Document Data
  - o Transaction Data

Graph-Based Data

- ✓ frequently convey important information. In such cases, the data is often represented as a graph.
- ✓ In particular, the data objects are mapped to nodes of the graph, while the relationships among objects are captured by the links between objects and link properties, such as direction and weight.

(j) Example: World Wide Web, Molecular Structures etc.,
(k)
(l)
(m) Ordered Data
(n) For some types of data, the attributes have relationships that involve order in time or space Different types of ordered data are

(o)
(p) ***Sequential Data:*** Also referred to as temporal data, can be thought of as an extension of record data, where each record has a time associated with it.

(q) Example:A retail transaction data set that also stores the time at which the transaction took place
(r)
(s)
(t) ***Sequence Data*** : Sequence data consists of a data set that is a sequence of individual entities, such as a sequence of words or letters. It is quite similar to sequential data, except that there are no time stamps; instead, there are positions in an ordered sequence.

(u) Example: the genetic information of plants and animals can be represented in the form of sequences of nucleotides that are known as genes.

(v)

(w) ***Time Series Data*** : Time series data is a special type of sequential data in which each record is a time series, i.e., a series of measurements taken over time.

(x) Example: A financial data set might contain objects that are time series of the daily prices of various stocks.

(y) Example,: consider a time series of the average monthly temperature for a city during the years 1982 to 1994

(z)

(aa) ***Spatial Data*** : Some objects have spatial attributes, such as positions or areas, as well as other types of attributes.

(bb) Example: Weather data (precipitation, temperature, pressure) that is collected for a variety of geographical locations.

General Characteristics of Data Sets

Dimensionality :

- ✓ It is the number of attributes that the objects in the data set possess.
- ✓ Data with a small number of dimensions tends to be qualitatively different than moderate or high-dimensional data.

Sparsity:

- ✓ For some data sets, most attributes of an object have values of 0; in many cases fewer than 1% of the entries are non-zero.
- ✓ In practical terms, sparsity is an advantage because usually only the non-zero values need to be stored and manipulated.
- ✓ This results in significant savings with respect to computation time and storage.

Resolution:

- ✓ It is frequently possible to obtain data at different levels of resolution, and often the properties of the data are different at different resolutions.

- ✓ For instance, the surface of the Earth seems very uneven at a resolution of meters, but is relatively smooth at a resolution of tens of kilometers.
- ✓ The patterns in the data also depend on the level of resolution.
- ✓ If the resolution is too fine, a pattern may not be visible or may be buried in noise; if the resolution is too coarse, the pattern may disappear.

Data Quality:

- ✓ Data mining applications are often applied to data that was collected for another purpose, or for future, but unspecified applications.
- ✓ For that reason ,data mining cannot usually take advantage of the significant benefits of "addressing quality issues at the source."
- ✓ Data mining focuses on (1) the detection and correction of data quality problems (called data cleaning.)
- ✓ (2) the use of algorithms that can tolerate poor data quality.
- ✓ Examples of data quality problems:

(cc) Noise and outliers

(dd) missing values

(ee) duplicate data

Noise: Noise refers to modification of original values

Examples: distortion of a person's voice when talking on a poor phone and "snow" on television screen

Outliers :Outliers are data objects with characteristics that are considerably different than most of the other data objects in the data set

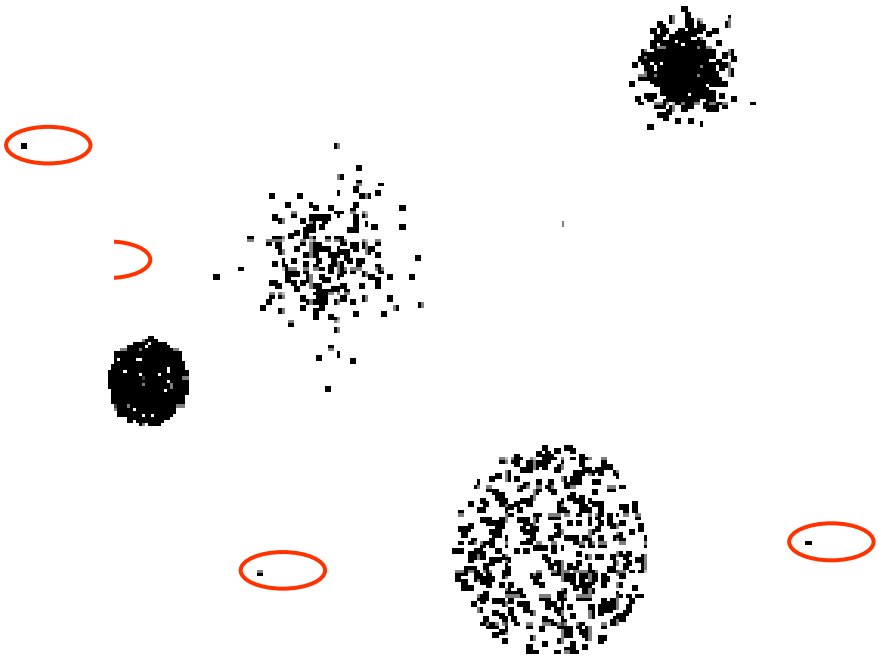


Figure 2.6:-outlier

Missing Values:

- ✓ Reasons for missing values
  - Information is not collected (e.g., people decline to give their age and weight)
  - Attributes may not be applicable to all cases (e.g., annual income is not applicable to children)

- ✓ Handling missing values
  - Eliminate Data Objects
  - Estimate Missing Values
  - Ignore the Missing Value During Analysis
  - Replace with all possible values (weighted by their probabilities)

Duplicate Data:

- ✓ Data set may include data objects that are duplicates, or almost duplicates of one another

- ✓ Major issue when merging data from heterogeneous sources
- ✓ Examples:

  Same person with multiple email addresses
- ✓ Data cleaning

  Process of dealing with duplicate data issues

Precision, Bias, and Accuracy:

(Precision). The closeness of repeated measurements(of the same quantity) to one another

(Bias). A systematic quantity being measured

(Accuracy). The closeness of measurements o the true value of the quantity being measured

Data Preprocessing

- ✓ Preprocessing steps should be applied to make the data more suitable for data mining
- ✓ The most important ideas and approaches are
  - Aggregation
  - Sampling
  - Dimensionality Reduction
  - Feature subset selection
  - Feature creation
  - Discretization and Binarization
  - Attribute Transformation

Aggregation

- ✓ Combining two or more attributes (or objects) into a single attribute (or object)

Purpose:

- o Data reduction
- o Reduce the number of attributes or objects
- o Change of scale
  - Cities aggregated into regions, states, countries, etc
- o More “stable” data
- o Aggregated data tends to have less variability

Sampling

- ✓ Sampling is the main technique employed for data selection.
- ✓ It is often used for both the preliminary investigation of the data and the final data analysis.

- ✓ Statisticians sample because obtaining the entire set of data of interest is too expensive or time consuming.
- ✓ Sampling is used in data mining because processing the entire set of data of interest is too expensive or time consuming.
- ✓ The key principle for effective sampling is the following:
    - using a sample will work almost as well as using the entire data sets, if the sample is representative
    - A sample is representative if it has approximately the same property (of interest) as the original set of data

Types of Sampling

- ✓ Simple Random Sampling

  There is an equal probability of selecting any particular item
- ✓ Sampling without replacement

  As each item is selected, it is removed from the population
- ✓ Sampling with replacement

  Objects are not removed from the population as they are selected for the sample.

  In sampling with replacement, the same object can be picked up more than once
- ✓ Stratified sampling

  Split the data into several partitions; then draw random samples from each partition

Dimensionality Reduction:

Purpose:

- o Avoid curse of dimensionality
- o Reduce amount of time and memory required by data mining algorithms
- o Allow data to be more easily visualized
- o May help to eliminate irrelevant features or reduce noise

- ✓ **The Curse of Dimensionality**: the curse of dimensionality refers to the phenomenon that many types of data analysis become significantly harder as the dimensionality of the data increases. Specifically, as dimensionality increases, the data becomes increasingly sparse in the space that it occupies.

Feature Subset Selection:

- ✓ Another way to reduce dimensionality of data

- ✓ Redundant features
  - duplicate much or all of the information contained in one or more other attributes
  - Example: purchase price of a product and the amount of sales tax paid

- ✓ Irrelevant features
  - contain no information that is useful for the data mining task at hand
  - Example: students' ID is often irrelevant to the task of predicting students' GPA

- ✓ Techniques of feature subset selection:

Brute-force approach:

Try all possible feature subsets as input to data mining algorithm

Embedded approaches:

Feature selection occurs naturally as part of the data mining algorithm

Filter approaches:

Features are selected before data mining algorithm is run

Wrapper approaches:

Use the data mining algorithm as a black box to find best subset of attributes

Feature Creation

- ✓ Create new attributes that can capture the important information in a data set much more efficiently than the original attributes

- ✓ Three general methodologies:
    - Feature Extraction
    - domain-specific
    - Mapping Data to New Space
    - Feature Construction
    - combining features

Discretization and Binarization:

- ✓ Some data mining algorithms require that the data be in the form of categorical attributes. Algorithms that find association patterns require that the data be in the form of binary attributes.
- ✓ Thus, it is often necessary to transform a continuous attribute into a categorical attribute (discretization).
- ✓ Both continuous and discrete attributes may need to be transformed into one or more binary attributes (binarization).

Attribute Transformation

- ✓ A function that maps the entire set of values of a given attribute to a new set of replacement values such that each old value can be identified with one of the new values

Similarity and Dissimilarity: Definition

- ✓ **Similarity** between two objects is a numerical measure of how alike two data objects are.
- ✓ Similarities are higher for pairs of objects that are more alike.
- ✓ Similarities are usually non-negative and are often between 0 (no similarity) and 1 (complete similarity).

- ✓ **Dissimilarity** between two objects is a Numerical measure of how different are two data objects.
- ✓ Dissimilarities are lower for more similar pairs of objects.

- ✓ Minimum dissimilarity is often 0, Upper limit varies

Similarity/Dissimilarity for Simple Attributes

The following table shows the similarity and dissimilarity between two objects, *x* and *y*, with respect to a single, simple attribute.

Attribute Type	Dissimilarity	Similarity
Nominal	$d = \begin{cases} 0 & \text{if } x = y \\ 1 & \text{if } x \neq y \end{cases}$	$s = \begin{cases} 1 & \text{if } x = y \\ 0 & \text{if } x \neq y \end{cases}$
Ordinal	$d = \lvert x - y \rvert / (n-1)$ (values mapped to integers 0 to $n-1$, where n is the number of values)	$s = 1 - d$
Interval or Ratio	$d = \lvert x - y \rvert$	$s = -d,\ s = \frac{1}{1+d},\ s = e^{-d},$ $s = 1 - \frac{d - min\_d}{max\_d - min\_d}$

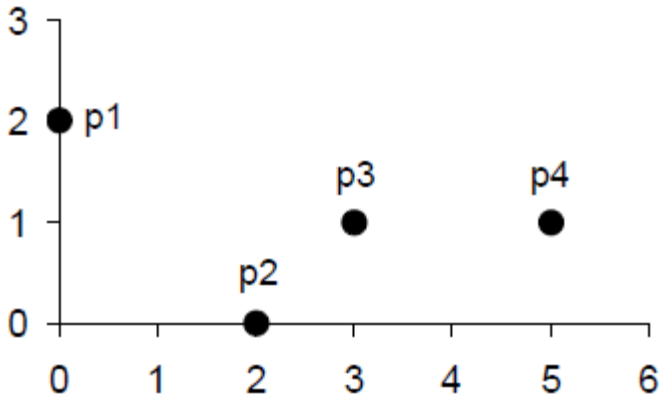
- Euclidean Distance

where *n* is the number of dimensions (attributes) and x_k and y_k are, respectively, the k^{th} attributes (components) or data objects **x** and **y**.

$$d(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{k=1}^{n} (x_k - y_k)^2},$$

Where n is the number of dimensions (attributes) and xk and yk are, respectively, the kth attributes (components) or data objects x and y.

Example:



point	x	y
p1	0	2
p2	2	0
p3	3	1
p4	5	1

	p1	p2	p3	p4
p1	0	2.828	3.162	5.099
p2	2.828	0	1.414	3.162
p3	3.162	1.414	0	2
p4	5.099	3.162	2	0

Distance Matrix

Where r is a parameter. Where n is the number of dimensions (attributes) and xk and yk are, respectively, the kth attributes (components) or data objects x and y.

$$d(\mathbf{x}, \mathbf{y}) = \left(\sum_{k=1}^{n} |x_k - y_k|^r \right)^{1/r},$$

The following are the three most common examples of Minkowski distances.

1) r = 1. City block (Manhattan, taxicab, L1 norm) distance.

   A common example of this is the Hamming distance, which is just the number of bits that are different between two binary vectors

2) r = 2. Euclidean distance
3) $r \to \infty$. "supremum" (Lmax norm, L∞ norm) distance.

This is the maximum difference between any component of the vectors

Minkowski Distance: Example

point	x	y
p1	0	2
p2	2	0
p3	3	1
p4	5	1

L1	p1	p2	p3	p4
p1	0	4	4	6
p2	4	0	2	4
p3	4	2	0	2
p4	6	4	2	0

L2	p1	p2	p3	p4
p1	0	2.828	3.162	5.099
p2	2.828	0	1.414	3.162
p3	3.162	1.414	0	2
p4	5.099	3.162	2	0

L_∞	p1	p2	p3	p4
p1	0	2	3	5
p2	2	0	1	3
p3	3	1	0	2
p4	5	3	2	0

Common Properties of a Distance

- ✓ Distances, such as the Euclidean distance, have some well-known properties.
- ✓ If d(x, y) is the distance between two points, x and y, then the following properties hold.

1. Positivity

(a) d(x, y) >= 0 for all x and y,

(b) d(x, y) = 0 only if x = y

2. Symmetry

d(x,y)= d(y,x) for all x and y.

3. Triangle Inequality

d(x,z) <= d(x, y) + d(y, z) for all points x, y, and z.

Measures that satisfy all three properties are known as metrics.

Common Properties of a Similarity

If s(x, y) is the similarity between points x and y, then the typical properties of similarities are the following:

1. s(x,y) = 1 only if x= y. (0 <= s <= 1)
2. s(x,y)= s(y, x) for all x and y. (Symmetry)

Similarity Measures for Binary Data

Let x and y be two objects that consist of n binary attributes. The comparison of two such objects, i.e., two binary vectors, Leads to the following four quantities (frequencies:)

f_{00} = the number of attributes where **x** is 0 and **y** is 0
f_{01} = the number of attributes where **x** is 0 and **y** is 1
f_{10} = the number of attributes where **x** is 1 and **y** is 0
f_{11} = the number of attributes where **x** is 1 and **y** is 1

Simple Matching Coefficient (SMC):

SMC = number of matches / number of attributes
= $(f_{11} + f_{00}) / (f_{01} + f_{10} + f_{11} + f_{00})$

J = number of 11 matches / number of non-zero attributes
= $(f_{11}) / (f_{01} + f_{10} + f_{11})$

x = 1 0 0 0 0 0 0 0 0 0
y = 0 0 0 0 0 0 1 0 0 1

$f_{01} = 2$ (the number of attributes where **x** was 0 and **y** was 1)
$f_{10} = 1$ (the number of attributes where **x** was 1 and **y** was 0)
$f_{00} = 7$ (the number of attributes where **x** was 0 and **y** was 0)
$f_{11} = 0$ (the number of attributes where **x** was 1 and **y** was 1)

SMC $= (f_{11} + f_{00}) / (f_{01} + f_{10} + f_{11} + f_{00})$
$= (0+7) / (2+1+0+7) = 0.7$

$J = (f_{11}) / (f_{01} + f_{10} + f_{11}) = 0 / (2 + 1 + 0) = 0$

Cosine Similarity

- If $\mathbf{d}_1$ and $\mathbf{d}_2$ are two document vectors, then

$$\cos(\mathbf{d}_1, \mathbf{d}_2) = \langle \mathbf{d}_1, \mathbf{d}_2 \rangle / \|\mathbf{d}_1\| \|\mathbf{d}_2\|,$$

where $\langle \mathbf{d}_1, \mathbf{d}_2 \rangle$ indicates inner product or vector dot product of vectors, $\mathbf{d}_1$ and $\mathbf{d}_2$, and $\| \mathbf{d} \|$ is the length of vector $\mathbf{d}$.

- Example:

$\mathbf{d}_1 = 3\ 2\ 0\ 5\ 0\ 0\ 0\ 2\ 0\ 0$

$\mathbf{d}_2 = 1\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 2$

$\langle \mathbf{d}_1, \mathbf{d2} \rangle = 3*1 + 2*0 + 0*0 + 5*0 + 0*0 + 0*0 + 0*0 + 2*1 + 0*0 + 0*2 = 5$

$\| \mathbf{d}_1 \| = (3*3+2*2+0*0+5*5+0*0+0*0+0*0+2*2+0*0+0*0)^{0.5} = (42)^{0.5} = 6.481$

$\| \mathbf{d}_2 \| = (1*1+0*0+0*0+0*0+0*0+0*0+0*0+1*1+0*0+2*2)^{0.5} = (6)^{0.5} = 2.449$

$\cos(\mathbf{d}_1, \mathbf{d}_2) = 0.3150$

x and y are two document vectors, then

$$\cos(\mathbf{x}, \mathbf{y}) = \frac{\mathbf{x} \cdot \mathbf{y}}{\|\mathbf{x}\| \|\mathbf{y}\|}, \qquad (2.7)$$

where $\cdot$ indicates the vector dot product, $\mathbf{x} \cdot \mathbf{y} = \sum_{k=1}^{n} x_k y_k$, and $\|\mathbf{x}\|$ is the length of vector $\mathbf{x}$, $\|\mathbf{x}\| = \sqrt{\sum_{k=1}^{n} x_k^2} = \sqrt{\mathbf{x} \cdot \mathbf{x}}$.

$$\text{corr}(\mathbf{x}, \mathbf{y}) = \frac{\text{covariance}(\mathbf{x}, \mathbf{y})}{\text{standard\_deviation}(\mathbf{x}) * \text{standard\_deviation}(\mathbf{y})} = \frac{s_{xy}}{s_x \; s_y}, \quad (2.11$$

where we are using the following standard statistical notation and definitions

$$\text{covariance}(\mathbf{x}, \mathbf{y}) = s_{xy} = \frac{1}{n-1} \sum_{k=1}^{n} (x_k - \overline{x})(y_k - \overline{y}) \quad (2.12$$

$$\text{standard\_deviation}(\mathbf{x}) = s_x = \sqrt{\frac{1}{n-1} \sum_{k=1}^{n} (x_k - \overline{x})^2}$$

$$\text{standard\_deviation}(\mathbf{y}) = s_y = \sqrt{\frac{1}{n-1} \sum_{k=1}^{n} (y_k - \overline{y})^2}$$

$$\overline{x} = \frac{1}{n} \sum_{k=1}^{n} x_k \text{ is the mean of } \mathbf{x}$$

$$\overline{y} = \frac{1}{n} \sum_{k=1}^{n} y_k \text{ is the mean of } \mathbf{y}$$

Question Bank

MODULE 1

1What is data warehouse? Discuss key features
2. Differentiate between Operational Database Systems and Data Warehouses.
3. Differentiate between OLAP and OLTP
4. Why multidimensional views of data and data-cubes are used? With a neat diagram, explain data-cube implementations.
5. Describe the Multitiered Architecture of data warehousing.
6. Explain the data warehouse models
7. What is ETL? Explain the steps in ETL
8. Write a note on Metadata Repository
9. Explain in detail data warehouse design and implementation. Explain star schema and snow

flake schema in detail?
10. Explain the schemas for multidimensional data models
11. Discuss the role of concept hierarchies
12. Explain OLAP operations with an example.
13. Describe the steps in data warehouse design process.
14. Explain Bitmap indexing and join indexing
15. Describe the difference between ROLAP & MOLAP

MODULE 2

1.What is data mining ? what are the applications of data mining.

2.Explain Knowledge data discovery KDD with a neat diagram. .

3.Discuss the challenges that motivate the development of Data Mining. .

4.Explain the origin of data mining .

5.What is data mining? Explain various data mining task with examples .

6.What are data and data attributes ? Explain the types and properties of attributes. .

7.Differentiate between discrete and continuous attributes.

8.Distinguish between categorical and numerical attributes. .

9.Explain the types of data sets.

10.List and explain general characteristics of data sets.

11. What is data quality? What are the dimension that asses the data quality.

12.Describe any five data preprocessing approaches. .

13. What is sampling? Explain simple random sampling v/s stratified sampling v/s progressive sampling.

14.Describe the various approaches for feature selection. .

15. What is curse of dimensionality? Explain .

16. What is similarity and dissimilarity? Explain similarity and dissimilarity measures between

simple attributes based on different types of attributes. .

17. Discuss the measures of proximity between objects that involve multiple attribute.

18. Explain the cosine similarity for calculating the similarity of two documents with an example.
.

19. Consider the following vectors. Find a) Simple Matching Co-efficient b) Jaccard Co-efficient c) Hamming Distance .

i)X: 0101010001 Y: 0100011000

ii)X: 1000000000 Y: 0000001001

20. Distinguish between:

a) Noise and Outlier

b) Jaccard Co-efficient and SMC

c)Discretization and Binarization

MODULE-3

Data Mining Association Analysis: Basic Concepts and Algorithms

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

1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
	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}

Definition: Item Set and support count

Itemset and Support Count Let I ={i1,i2,….id} be the set of all itemsin a market basket data and
T : {t1,t2,..-,tN} be the set of all transactions. Each transaction ti contains a subset of items
chosen from I

In association analysis, a collection of zero or more items is termed an itemset. If an itemsetcontains k- items, it is called a k-itemset

Example: {Milk, Bread, Diaper}

Association Rule An association rule is an implication expression of the form $X \longrightarrow Y$, where X and Y are disjoint itemsets, i.e., $X \cap Y = \emptyset$. The strength of an association rule can be measured in terms of its **support** and **confidence**. Support determines how often a rule is applicable to a given data set, while confidence determines how frequently items in Y appear in transactions that contain X. The formal definitions of these metrics are

$$\text{Support, } s(X \longrightarrow Y) = \frac{\sigma(X \cup Y)}{N}; \quad (6.1)$$

$$\text{Confidence, } c(X \longrightarrow Y) = \frac{\sigma(X \cup Y)}{\sigma(X)}. \quad (6.2)$$

Rule Evaluation Metrics:

Support count (·)

- Frequency of occurrence of an itemset
-

E.g. · ({Milk, Bread,Diaper}) = 2 Support(s)

- Fraction of transactions that contain an itemset
- E.g.

s({Milk,Bread,Dipe r}) = 2/5

Frequent Itemset

- An itemset whose support is greater than or equal to a minsup threshold

Example:

{ Milk ,Diaper } · Beer

$$s \cdot \frac{(\text{Milk, Diaper, Beer})}{|T|} \quad \frac{2}{5} = 0.4$$

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

Association Rule Mining Task

Given a set of transactions T, the goal of association rule mining is to find all rules having

- support ≥ minsupthreshold
- confidence ≥

minconfthresholdBrute-force approach:

- List all possible association rules
- Compute the support and confidence for each rule
- Prune rules that fail the minsup and minconf thresholds · Computationally prohibitive!

More specifically, the total number of possible rules extracted from a data set that contains d items is

$$R = 3^d - 2^{d+1} + 1.$$

Even for the small data set with 6 items, this approach requires us to compute the support and confidence for 36 - 27 * 1 = 602 rules.

More than 80% of the rules are discarded after applying minsup : 20Vo andminconf : 5070, thus making most of the computations become wasted.

To avoid performing needless computations, it would be useful to prune the rules early without having to compute their support and confidence values

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

If the item set is infrequent, then all six candidate rules can be pruned immediately without our having to compute their confidence values.

Therefore, a common strategy adopted by many association rule mining algorithms is to decompose the problem into two major subtasks:

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.

2. Consider the data set shown in Table 6.1.

(a) Compute the support for itemsets $\{e\}$, $\{b, d\}$, and $\{b, d, e\}$ by treating each transaction ID as a market basket.

Answer:

Table 6.1. Example of market basket transactions.

Customer ID	Transaction ID	Items Bought
1	0001	$\{a, d, e\}$
1	0024	$\{a, b, c, e\}$
2	0012	$\{a, b, d, e\}$
2	0031	$\{a, c, d, e\}$
3	0015	$\{b, c, e\}$
3	0022	$\{b, d, e\}$
4	0029	$\{c, d\}$
4	0040	$\{a, b, c\}$
5	0033	$\{a, d, e\}$
5	0038	$\{a, b, e\}$

$$s(\{e\}) = \frac{8}{10} = 0.8$$
$$s(\{b, d\}) = \frac{2}{10} = 0.2$$
$$s(\{b, d, e\}) = \frac{2}{10} = 0.2$$

(6.1)

(b) Use the results in part (a) to compute the confidence for the association rules $\{b,d\} \longrightarrow \{e\}$ and $\{e\} \longrightarrow \{b,d\}$. Is confidence a symmetric measure?

Answer:

$$c(bd \longrightarrow e) = \frac{0.2}{0.2} = 100\%$$
$$c(e \longrightarrow bd) = \frac{0.2}{0.8} = 25\%$$

No, confidence is not a symmetric measure.

(c) Repeat part (a) by treating each customer ID as a market basket. Each item should be treated as a binary variable (1 if an item appears in at least one transaction bought by the customer, and 0 otherwise.)

Answer:

$$s(\{e\}) = \frac{4}{5} = 0.8$$
$$s(\{b,d\}) = \frac{5}{5} = 1$$
$$s(\{b,d,e\}) = \frac{4}{5} = 0.8$$

(d) Use the results in part (c) to compute the confidence for the association rules $\{b,d\} \longrightarrow \{e\}$ and $\{e\} \longrightarrow \{b,d\}$.

Answer:

$$c(bd \longrightarrow e) = \frac{0.8}{1} = 80\%$$
$$c(e \longrightarrow bd) = \frac{0.8}{0.8} = 100\%$$

Table 6.2. Market basket transactions.

Transaction ID	Items Bought
1	{Milk, Beer, Diapers}
2	{Bread, Butter, Milk}
3	{Milk, Diapers, Cookies}
4	{Bread, Butter, Cookies}
5	{Beer, Cookies, Diapers}
6	{Milk, Diapers, Bread, Butter}
7	{Bread, Butter, Diapers}
8	{Beer, Diapers}
9	{Milk, Diapers, Bread, Butter}
10	{Beer, Cookies}

6. Consider the market basket transactions shown in Table 6.2.

(a) What is the maximum number of association rules that can be extracted from this data (including rules that have zero support)?
Answer: There are six items in the data set. Therefore the total number of rules is 602.

(b) What is the maximum size of frequent itemsets that can be extracted (assuming $minsup > 0$)?
Answer: Because the longest transaction contains 4 items, the maximum size of frequent itemset is 4.

(c) Write an expression for the maximum number of size-3 itemsets that can be derived from this data set.
Answer: $\binom{6}{3} = 20$.

(d) Find an itemset (of size 2 or larger) that has the largest support.
Answer: {Bread, Butter}.

(e) Find a pair of items, a and b, such that the rules $\{a\} \longrightarrow \{b\}$ and $\{b\} \longrightarrow \{a\}$ have the same confidence.
Answer: (Beer, Cookies) or (Bread, Butter).

Frequent Itemset Generation:

A lattice structure can be used to enumerate the list of all possible item sets. Figure 6.1 shows an itemset lattice for 1: {a,b,c.,d,e}.In general, a data setthat contains k items can potentially generate up to 2^k- 1 frequent itemsets, excluding the null set. Because k can be very large in many practical applications, the search space of itemsets that need to be explored is exponentially Large.

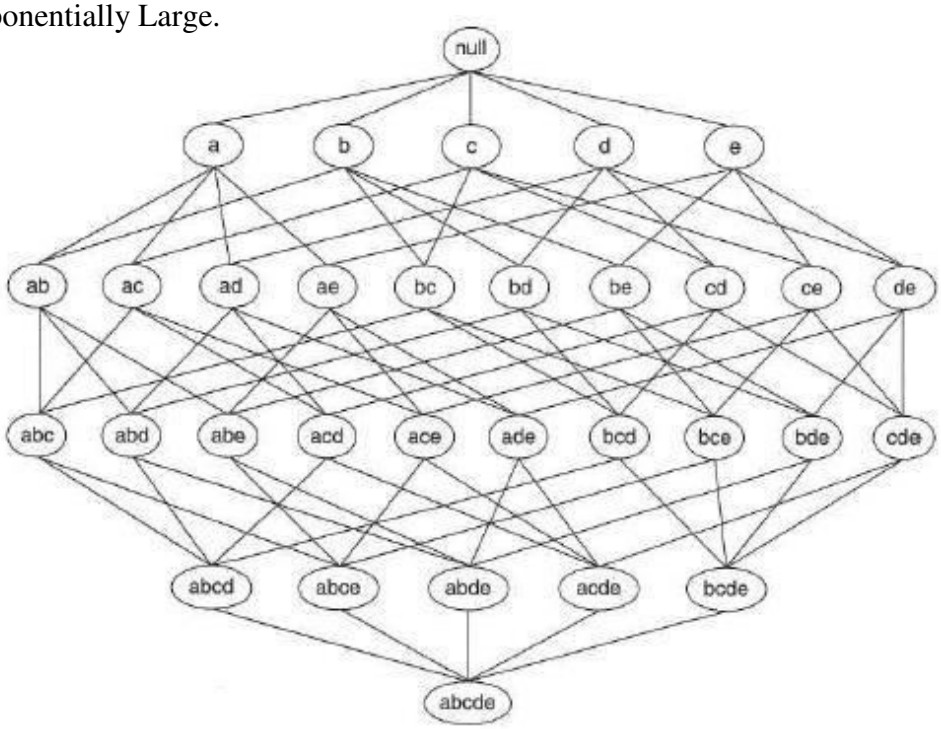
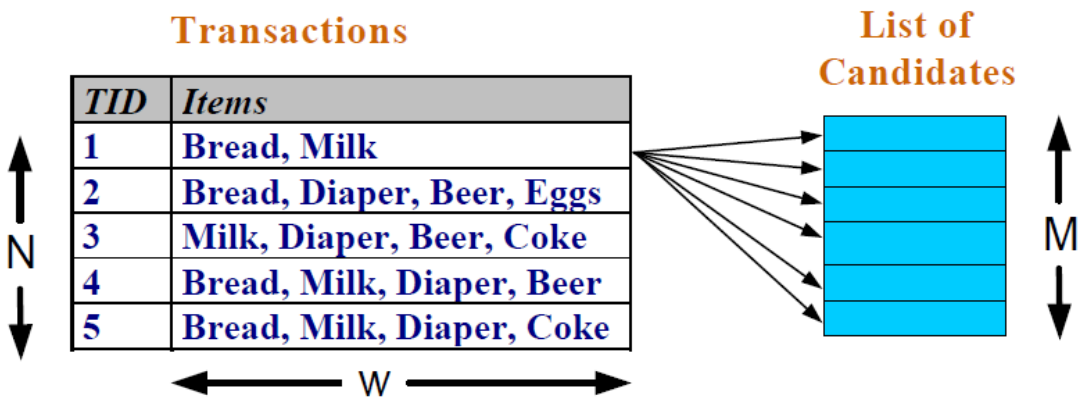


Figure 6.1. An itemset lattice.

Brute-force approach:

- Each itemset in the lattice is a candidate frequent itemset
- Count the support of each candidate by scanning the database

Such an approach can be very expensive because it requires O(N Mw) comparisons, where N is the number of transactions, M =2^k - 1 is the number of candidate itemsets, and w is the maximum transaction width.



There are several ways to reduce the computational complexity of frequent itemset generation.

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 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

Apriori principle : Reducing Number of Candidates

Apriory principle: If an itemset is frequent, then all of its subsets must also be frequent

To illustrate the idea behind the Apriory principle, consider the itemsetIattice shown in Figure

6.3. Suppose {c, d, e} is a frequent itemset. Clearly,any transaction that contains {c,d,e} must also contain its subsets, {c,d},{c,e}, {d,e}, {c}, {d}, and

{e}. As a result, if {c,d,e} is frequent, then all subsets of {c, d,e} (i.e., the shaded itemsets in this figure) must also be frequent.

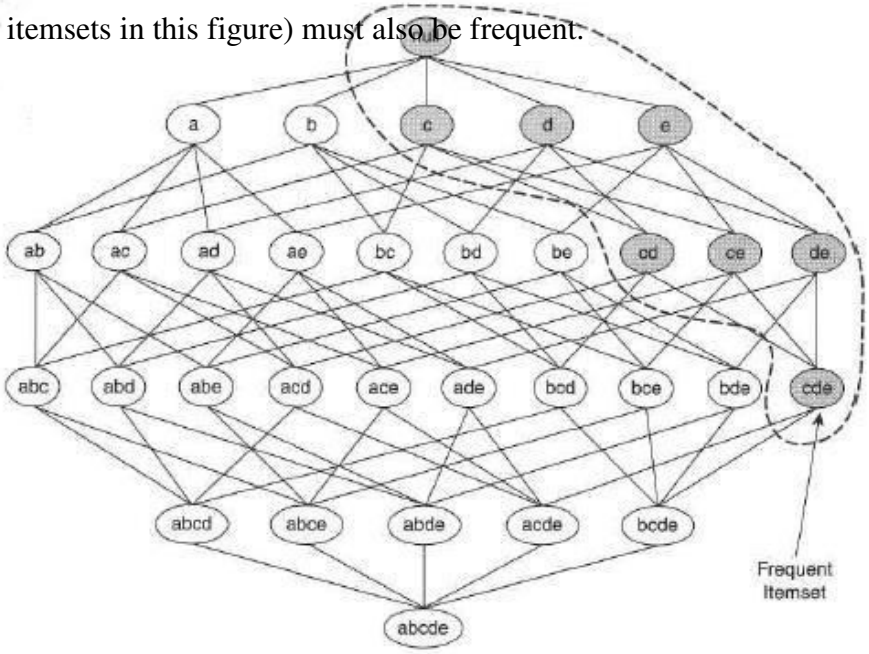


Figure 6.3. An illustration of the *Apriori* principle. If $\{c, d, e\}$ is frequent, then all subsets of this itemset are frequent.

Conversely, if an itemset such as {a, b} is infrequent, then all of its supersets must be infrequent too. As illustrated in Figure 6.4, the entire subgraph containing the supersets of {a, b} can be pruned immediately once {a, b} is found to be infrequent. This strategy of trimming the exponential search space based on the support measure is known as support-based pruning.

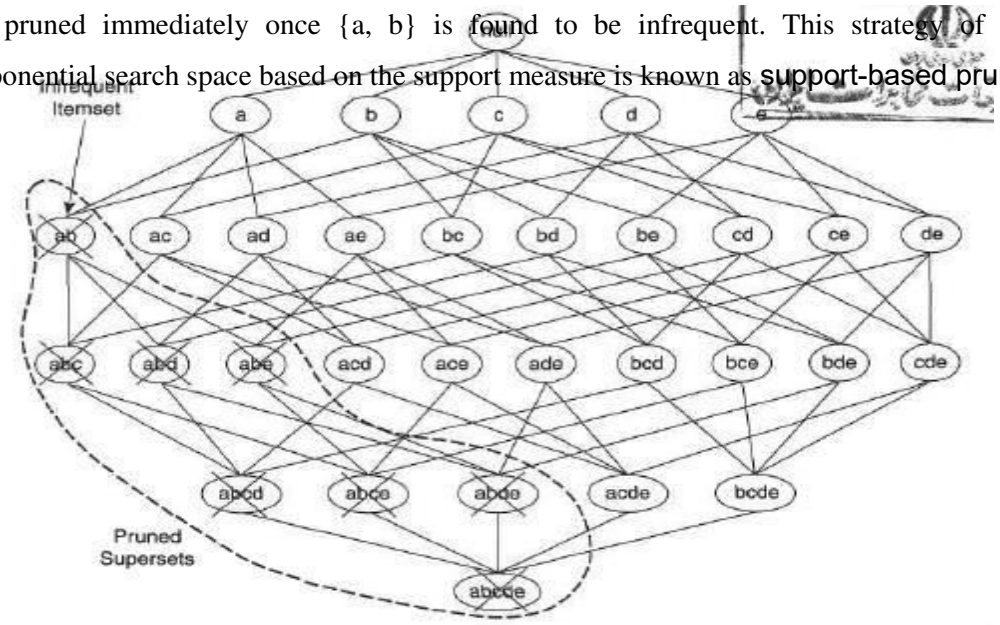


Figure 6.4. An illustration of support-based pruning. If $\{a, b\}$ is infrequent, then all supersets of $\{a, b\}$ are infrequent.

Frequent Itemset Generation in the Apriori Algorithm: Illustration with example.

Figure 6.5 provides a high-level illustration of the frequent item set generation part of the Apriori algorithm for the transactions shown inTable 6.1. We assume that the support threshold is 60 To, which is equivalent to a minimum support count equal to 3.

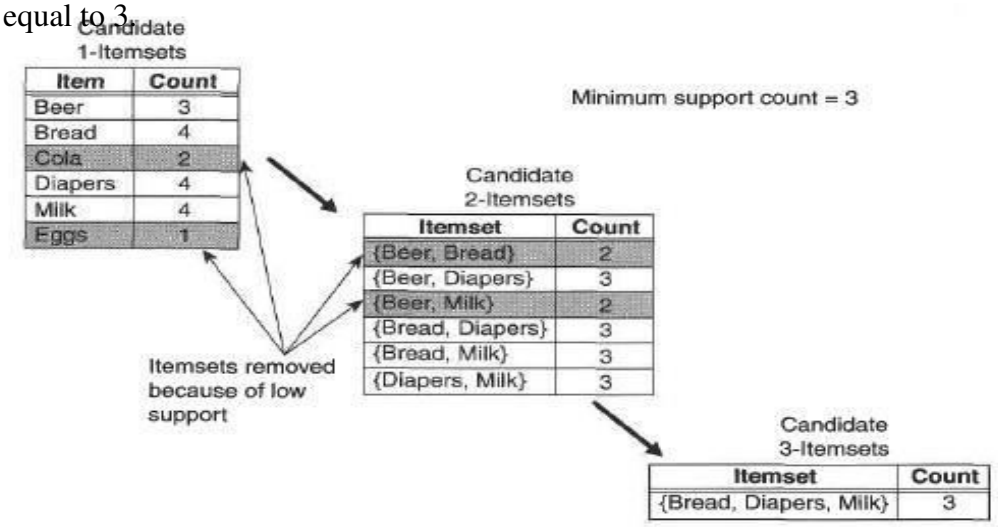
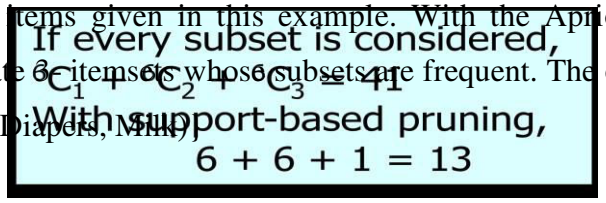


Figure 6.5. *Illustration of frequent itemset generation using the Apriori algorithm.*

Initially, every item is considered as a candidate l-itemset. Aftercounting their supports, the candidate itemsets {Co1a} and {Eggs} are discarded because they appear in fewer than three transactions.

In the next iteration, candidate 2-itemsets are generated using only the frequent 1-itemsets because the Apriory principle ensures that all supersets of the infrequent 1-itemsets must be infrequent. Because there are only four frequent 1-itemsets, the number of candidate 2-itemsets generated by the algorithm is 6. Two of these six candidates, {Beer, Bread} and {Beer, Milk}, are subsequently found to be infrequent after computing their support values. The remaining four candidates are frequent, and thus will be used to generate candidate 3-itemsets.

Without support-based. pruning, there are 20 candidate3-itemsets that can be formed using the six items given in this example. With the Apriory principle, we only need to keep candidate 3-itemsets whose subsets are frequent. The only candidate that has this property is {Bread,Diapers, Milk}



.

The effectiveness of the Apriory pruning strategy can be shown by counting the number of candidate itemsets generated.

A brute-force strategy of enumerating all itemsets(up to size3) as candidates w ill produce 41 candidates.

With the Apriory principle, this number decreases t o candidates, which epresents a 68% reduction in the number of candidate itemsets even in this simple example.

Apriori Algorithm:

Input: set of items I, set of transactions T, number of transactions N, minimum support minsup. Output: frequent k-itemsetsFk, k=1…

Method:

K=1

✓ Compute support for each 1-itemset (item) by scanning the transactions ✓ F1 = items that have support above minsup

✓ Repeat until no new frequent itemsets are identified

1. Ck+1 = candidate k+1 -itemsets generated from length k frequent itemsetsFk
2. Compute the support of each candidate in Ck+1 by scanning the transactions T
3. Fk+1 = Candidates in Ck+1 that have support above minsup.

8. The *Apriori* algorithm uses a generate-and-count strategy for deriving frequent itemsets. Candidate itemsets of size $k+1$ are created by joining a pair of frequent itemsets of size k (this is known as the candidate generation step). A candidate is discarded if any one of its subsets is found to be infrequent during the candidate pruning step. Suppose the *Apriori* algorithm is applied to the data set shown in Table 6.3 with $minsup = 30\%$, i.e., any itemset occurring in less than 3 transactions is considered to be infrequent.

Table 6.3. Example of market basket transactions.

Transaction ID	Items Bought
1	$\{a, b, d, e\}$
2	$\{b, c, d\}$
3	$\{a, b, d, e\}$
4	$\{a, c, d, e\}$
5	$\{b, c, d, e\}$
6	$\{b, d, e\}$
7	$\{c, d\}$
8	$\{a, b, c\}$
9	$\{a, d, e\}$
10	$\{b, d\}$

(a) Draw an itemset lattice representing the data set given in Table 6.3. Label each node in the lattice with the following letter(s):

- **N**: If the itemset is not considered to be a candidate itemset by the *Apriori* algorithm. There are two reasons for an itemset not to be considered as a candidate itemset: (1) it is not generated at all during the candidate generation step, or (2) it is generated during the candidate generation step but is subsequently removed during the candidate pruning step because one of its subsets is found to be infrequent.
- **F**: If the candidate itemset is found to be frequent by the *Apriori* algorithm.
- **I**: If the candidate itemset is found to be infrequent after support counting.

Answer:

The lattice structure is shown below.

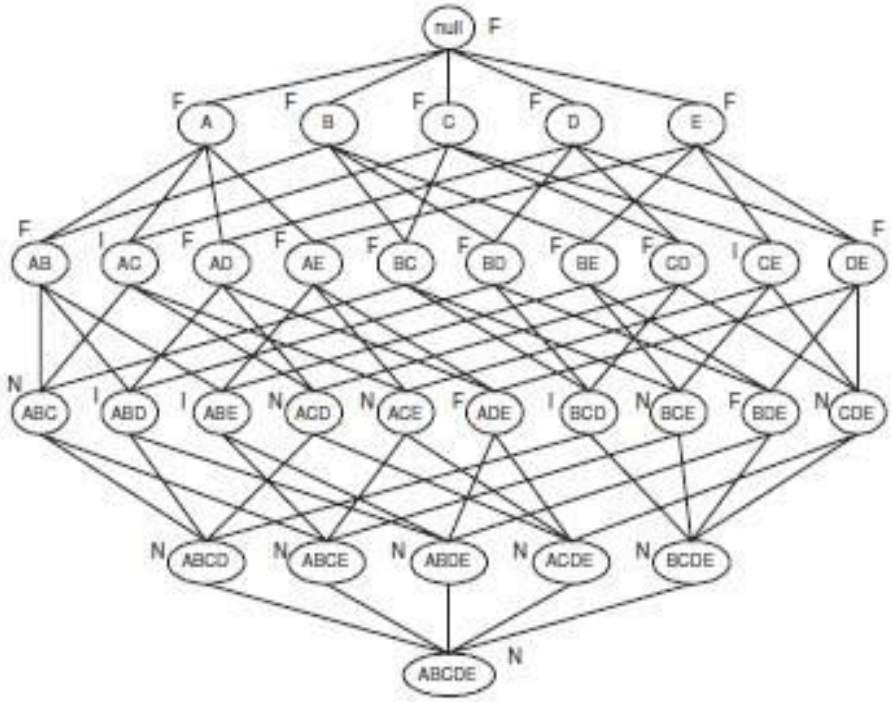


Figure 6.1. Solution.

(b) What is the percentage of frequent itemsets (with respect to all itemsets in the lattice)?

Answer:

Percentage of frequent itemsets = 16/32 = 50.0% (including the null set).

(c) What is the pruning ratio of the *Apriori* algorithm on this data set? (Pruning ratio is defined as the percentage of itemsets not considered to be a candidate because (1) they are not generated during candidate generation or (2) they are pruned during the candidate pruning step.)

Answer:

Pruning ratio is the ratio of N to the total number of itemsets. Since the count of $N = 11$, therefore pruning ratio is $11/32 = 34.4\%$.

(d) What is the false alarm rate (i.e, percentage of candidate itemsets that are found to be infrequent after performing support counting)?

Answer:

False alarm rate is the ratio of I to the total number of itemsets. Since the count of $I = 5$, therefore the false alarm rate is $5/32 = 15.6\%$.

Candidate Generation and Pruning

Candidate Generation: This operation generates new candidate kitemsets based on the frequent (k - l)-itemsets found in the previous iteration.

Candidate Pruning: This operation eliminates some of the candidate k-itemsets using the support-based pruning strategy.

The folIowing is a list of requirements for an effective candidate generation procedure:

➢ It should avoid generating too many unnecessary candidates.
➢ It must ensure that the candidate set is complete, i.e., no frequent itemsetsare
left out by the candidate generation procedure.
➢ It should not generate the same candidate itemset more than once.
e.g. {a,b,c,d} can be generated by merging {a,b,c} with {d}
or {b,d} with {a,c}, {a,b} with {c,d}

Several candidate generation strategies are discussed below. Brute-Force Method: The brute-force method considers every k-itemset asa potential candidate and then applies the candidate pruning step to removeany unnecessary candidates (see Figure).

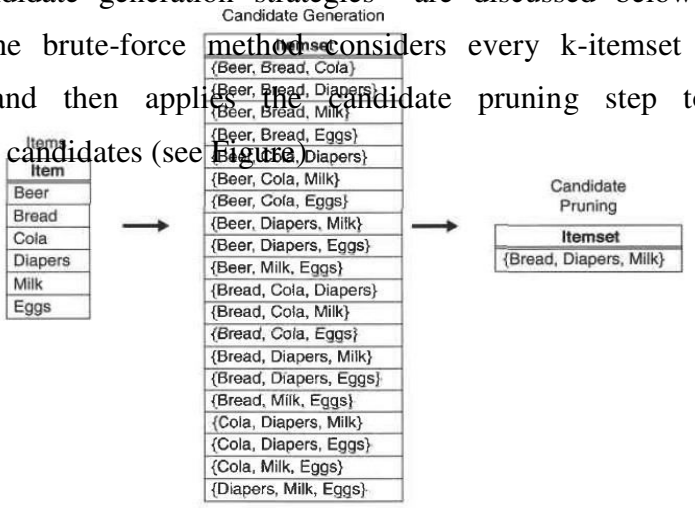


Figure 6.6. A brute-force method for generating candidate 3-itemsets.

Fk-1 x F1 Method:

Combine frequent k-1 -itemsets with frequent 1- itemsets

Figure 6.7 illustrates how a frequent 2-itemset such as {Beer, Diapers} can be augmented with a frequent item such as Bread to produce a candidate 3-itemset {Beer, Diapers, Bread}.

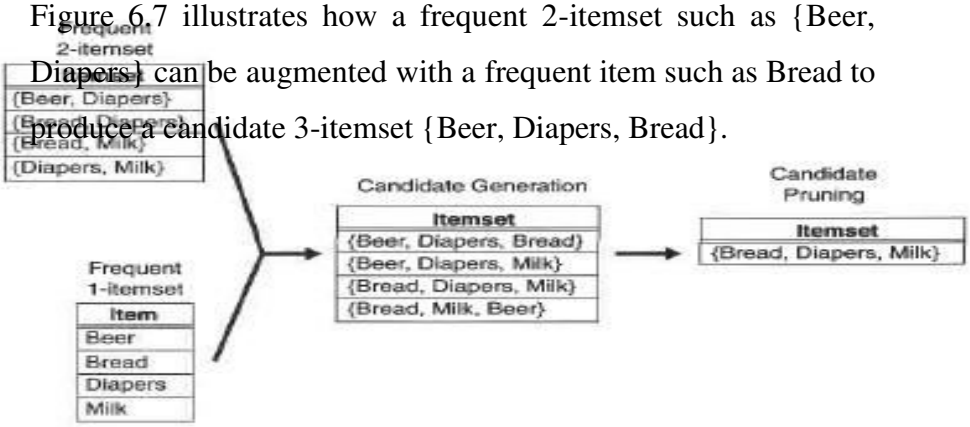


Figure 6.7. Generating and pruning candidate k-itemsets by merging a frequent $(k-1)$-itemset with a frequent item. Note that some of the candidates are unnecessary because their subsets are infrequent.

Satisfaction of our requirements

1)while many k-itemsets are left ungenerated, can still generate unnecessary candidates

e.g. merging {Beer, Diapers} with {Milk} is unnecessary, since {Beer, Milk} is infrequent.

2) Method is complete: each frequent itemset consists of a frequent k-1 -itemset and a frequent 1-itemset.

3) Can generate the same set twice

e.g. {Bread, Diapers, Milk} can be generated by merging {Bread,Diapers} with {Milk} or {Bread,Milk} with {Diapers} or {Diapers, Milk} with {Bread}

This can be circumvented by keeping all frequent itemsets in their lexicographical order (\

- e.g. {Bread,Diapers} can be merged with {Milk} as 'Milk' comes after 'Bread' and 'Diapers' in lexicographical order

- {Diapers, Milk} is not merged with {Bread}, {Bread, Milk} is not merged with {Diapers} as that would violate the lexicographical ordering

Fk-1 x Fk-1 Method:

➢ Combine a frequent k-1 -itemset with another frequent k-1 -itemset➢ Items are stored in lexicographical order in the itemset

➢ When considering for merging, only pairs that share first k-2 items are considered

- e.g. {Bread, Diapers} is merged with {Bread,Milk}
- if the pairs share fewer than k-2 items, the resulting itemset would be larger than k, so we do not need to generate it yet

➢ The resulting k-itemset has k subsets of size k-1, which will be checked against support threshold

- the merging ensures that at least two of the subsets are frequent
- An additional check is made that the remaining k-2 subsets are frequent as well

In Figure 6.8, the frequent itemsets {Bread, Diapers} and {Bread, Milk} are merged to form a candidate 3-itemset {Bread, Diapers, Milk}.

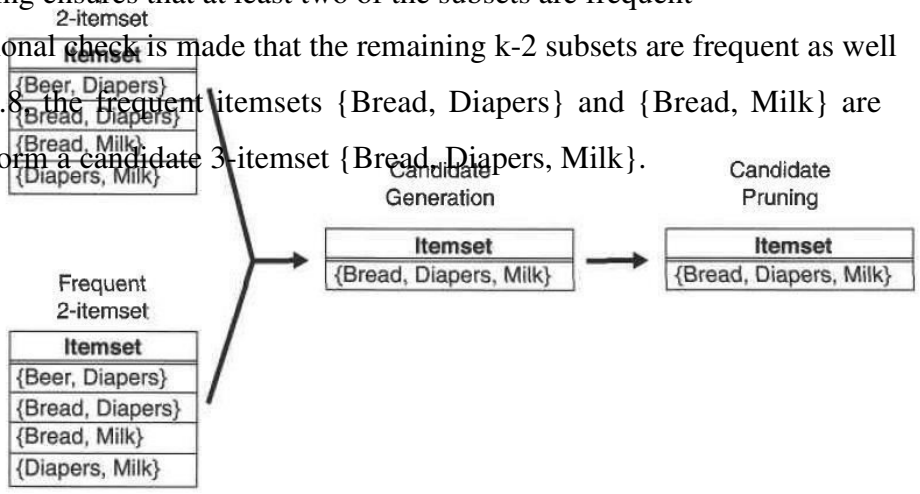


Figure 6.8. Generating and pruning candidate k-itemsets by merging pairs of frequent $(k-1)$-itemsets.

Satisfaction of our requirements

1) Avoids the generation of many unnecessary candidates that are generated by the Fk-1 x F1 method

e.g. will not generate {Beer, Diapers, Milk} as {Beer,Milk} is infrequent

2. Method is complete: every frequent k-itemset can be formed of two frequent k-1 –itemsets differing in their last item.

3. Each candidate itemset is generated only once.

(a) List all candidate 4-itemsets obtained by a candidate generation procedure using the $F_{k-1} \times F_1$ merging strategy.

Answer:

$\{1,2,3,4\},\{1,2,3,5\},\{1,2,3,6\}.$
$\{1,2,4,5\},\{1,2,4,6\},\{1,2,5,6\}.$

$\{1,3,4,5\},\{1,3,4,6\},\{2,3,4,5\}.$
$\{2,3,4,6\},\{2,3,5,6\}.$

(b) List all candidate 4-itemsets obtained by the candidate generation procedure in *Apriori*.

Answer:

$\{1,2,3,4\}, \{1,2,3,5\}, \{1,2,4,5\}, \{2,3,4,5\}, \{2,3,4,6\}.$

(c) List all candidate 4-itemsets that survive the candidate pruning step of the *Apriori* algorithm.

Answer:

$\{1,2,3,4\}$

Support counting using hash tree:

Given the candidate itemsetsCk and the set of transactions T, we need to compute the support counts σ(X) for each itemset X in Ck.

Brute-force algorithm would compare each transaction against each itemset.

➢ large amount of comparisons.

An alternative approach

➢ Divide the candidate itemsets Ck into buckets by using a hash function for each transaction t.

➢ Hash the itemsets contained in t into buckets using the same hash function. ➢ Compare the corresponding buckets of candidates and the transaction. ➢ Increment the support counts of each matching candidate itemset.

➢ A hash tree is used to implement the hash function.

An alternative approach is to enumerate the itemsets contained in each transaction and use them to update the support counts oftheir respective candidate itemsets. To illustrate, consider a transaction t that contains five items, {1,2,3,5,6}.

Figure 6.9 shows a systematic way for enumerating the 3-itemsets contained in t. Assuming that each itemset keeps its items in increasing lexicographic order, an itemset can be enumerated by specifying the smallest item first, followed by the larger items. For instance, given t : {1,2,3,5,6}, all the 3- itemsets contained in f must begin with item 1, 2, or 3.

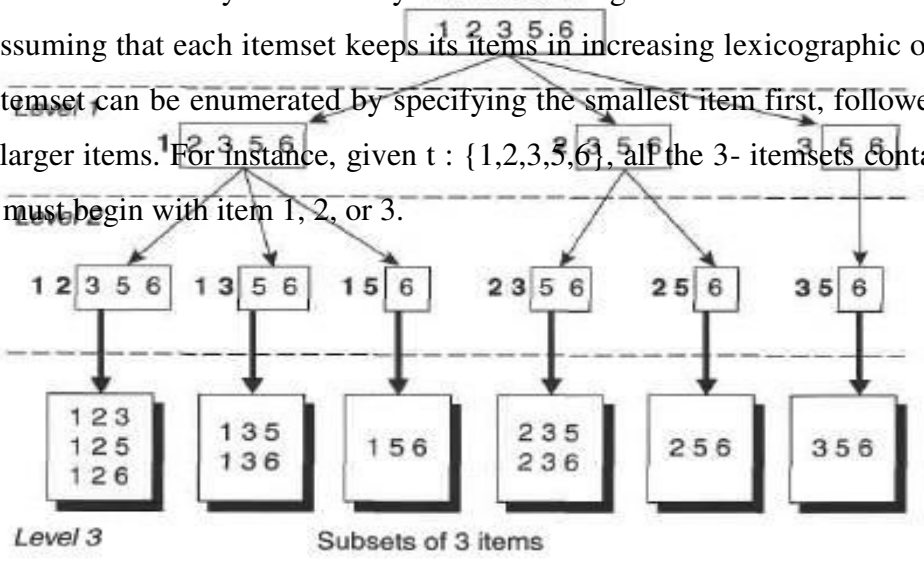


Figure 6.9. Enumerating subsets of three items from a transaction t.

Figure 6.11 shows an example of a hash tree structure.

Each internal node of the tree uses the following hash function, h(p) : p mod 3, to determine which branch of the current node should be followed next.

For example, items 1, 4, and 7 are hashed to the same branch (i.e., the leftmost branch) because they have the same remainder after dividing the number by 3.

All candidate itemsets are stored at the leaf nodes of the hash tree. The hash tree shown in Figure 6.11 contains 15 candidate 3-itemsets, distributed across 9 leaf nodes.

Consider a transaction, t, : {1,2,3,5,6}. To update the support counts of the candidate itemsets, the hash tree must be traversed in such a way that all the leaf nodes containing candidate 3-itemsets belonging to t must be visited at least once.

At the root node of the hash tree, the items 1, 2, and 3 of the transaction are hashed separately. Item 1 is hashed to the left child of the root node, item 2 is hashed to the middle child, and item 3 is hashed to the right child.

At the next level of the tree, the transaction is hashed on the second item listed in the Level 2 structures shown in Figure 6.9.

For example, after hashing on item 1 at the root node, items 2, 3, and 5 of the transaction are hashed. Items 2 and 5 are hashed to the middle child, while item 3 is hashed to the right child, as shown in Figure 6.12. This process continues until the leaf nodes of the hash tree are reached.

The candidate item sets stored at the visited leaf nodes are compared against the transaction. If a candidate is a subset of the transaction, its support count is incremented.

In this example, 5 out of the 9 leaf nodes are visited and 9 out of the 15 item sets are compared against the transaction.

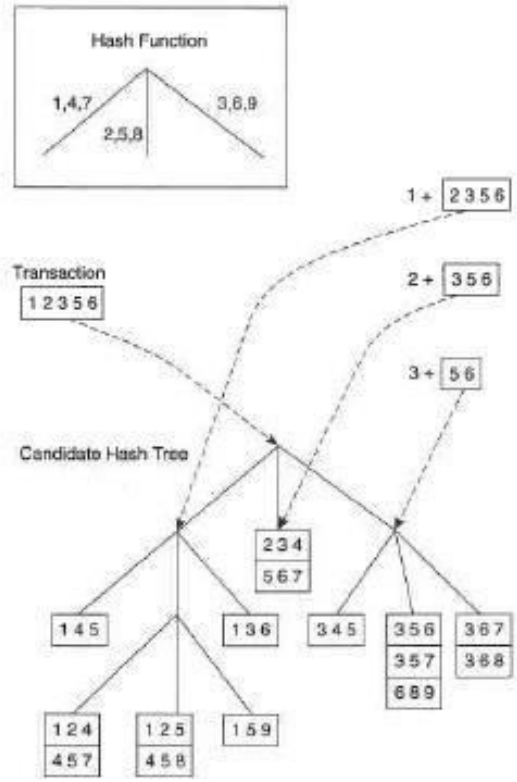


Figure 6.11. Hashing a transaction at the root node of a hash tree.

9. The *Apriori* algorithm uses a hash tree data structure to efficiently count the support of candidate itemsets. Consider the hash tree for candidate 3-itemsets shown in Figure 6.2.

   (a) Given a transaction that contains items $\{1,3,4,5,8\}$, which of the hash tree leaf nodes will be visited when finding the candidates of the transaction?

   Answer:

   The leaf nodes visited are L1, L3, L5, L9, and L11.

   (b) Use the visited leaf nodes in part (b) to determine the candidate itemsets that are contained in the transaction $\{1,3,4,5,8\}$.

   Answer:

   The candidates contained in the transaction are $\{1,4,5\}$, $\{1,5,8\}$, and $\{4,5,8\}$.

Rule Generation

Given a frequent itemset L, find all non-empty subsets f · L such that f · L - f satisfies the minimum confidence requirement

- If {A,B,C,D} is a frequent itemset, candidate rules:

ABC · D, ABD · C, ACD · B, BCD · A,

A · BCD, B · ACD, C · ABD, D · ABC

AB · CD, AC · BD, AD · BC, BC · AD,

BD · AC, CD · AB,

If |L| = k, then there are 2^k - 2 candidate association rules (ignoring L · · and · · L)

How to efficiently generate rules from frequent itemsets?

- In general, confidence does not have an anti-monotone property

  c(ABC · D) can be larger or smaller than c(AB · D)

- But confidence of rules generated from the same itemset has an anti-monotone property

  e.g., L = {A,B,C,D}:

  c(ABC · D) · c(AB · CD) · c(A · BCD)

· Confidence is anti-monotone w.r.t. number of items on the RHS of the rule

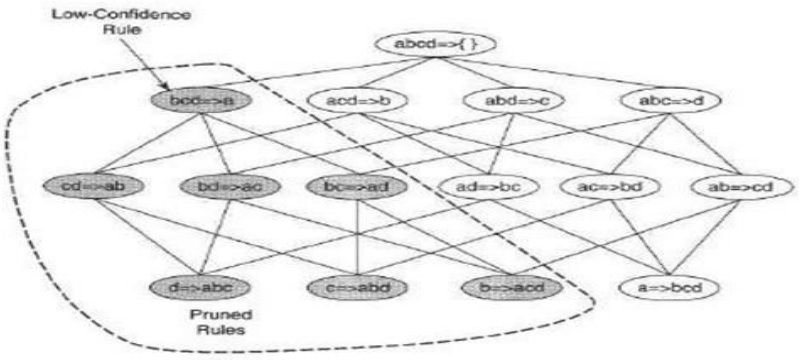


Figure 6.15. Pruning of association rules using the confidence measure.

Candidate rule is generated by merging two rules that share the same prefix in the rule consequent

join(CD=>AB,BD=>AC) would produce the candidate rule D => ABC Prune rule D=>ABC if its subset AD=>BC does not have high confidence

Alternative Methods for Generating Frequent Itemsets

Traversal of Itemset Lattice::A search for frequent itemsets can be conceptually viewed as a traversal on the itemset lattice.

The search strategy employed by an algorithm dictates how the lattice structure is traversed during the frequent itemset generation process. Some search strategies are better than others, depending on the configuration of frequent itemsets in the lattice.

Equivalence classes :Equivalence Classes can also be defined according to the prefix or suffix labels of an itemset.

In this case, two itemsets belong to the same equivalence class if they share a common prefix or suffix of length k. In the prefix-based approach, the algorithm can search for frequent itemsetsstarting with the prefix a before looking for those starting with prefixes b, c and so on

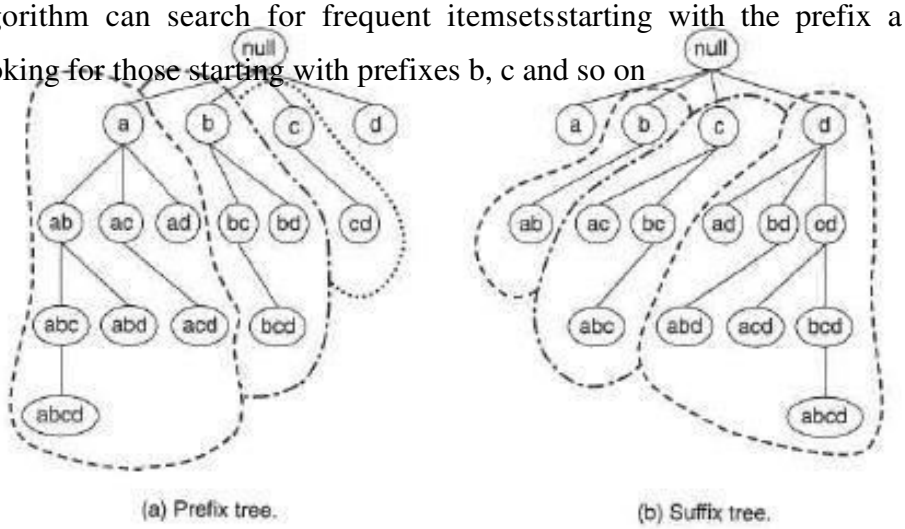


Figure 6.20. Equivalence classes based on the prefix and suffix labels of itemsets.

Breadth-First versus Depth-First: The Apriori, algorithm traverses the lattice in a breadth-first manner) as shown in Figure 6.2L(a). It first discovers all the frequent 1-itemsets, followed by the frequent 2-itemsets, and so on, until no new frequent itemsets are generated.

The algorithm can start from, say, node a, in Figure 6.22, and count its support to determine whether it is frequent. If so, the algorithm progressively expands the next level of nodes, i.e., ab, abc, and so on, until an infrequent node is reached, say, abcd. It then backtracks to another branch, say, abce, and continues the search from there.

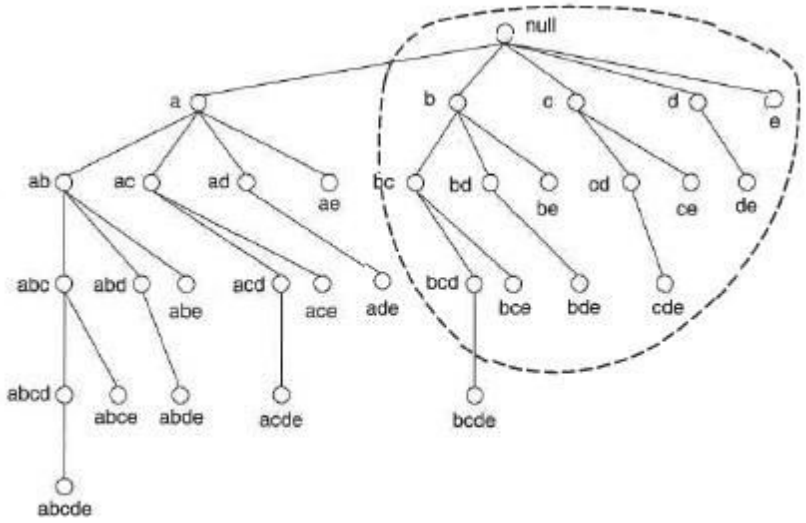


Figure 6.22. Generating candidate itemsets using the depth-first approach.

FP-Growth Algorithm

➢Apriori: uses a generate-and-test approach - generates candidate itemsets and tests if they are frequent

- Generation of candidate itemsets is expensive(in both space and time)
- Support counting is expensive
- Subset checking (computationally expensive) Multiple Database scans

➢ FP-Growth: allows frequent itemset discovery without candidate itemset generation. Two step approach:

- Step 1: Build a compact data structure called the FP-tree
- Built using 2 passes over the data-set.

– Step 2: Extracts frequent itemsets directly from the FP-tree

Step 1: FP-Tree Construction

➢ FP-Tree is constructed using 2 passes over the data-set: Pass 1: Scan data and find support for each item.

- Discard infrequent items.
- Sort frequent items in decreasing order based on their support.

Use this order when building the FP-Tree, so common prefixes can be shared.

Pass 2:

Nodes correspond to items and have a counter

1. FP-Growth reads 1 transaction at a time and maps it to a path

2. Fixed order is used, so paths can overlap when transactions share items (when they have the same prfix).

- In this case, counters are incremented

3. Pointers are maintained between nodes containing the same item, creating singly linked lists (dotted lines)

- The more paths that overlap, the higher the compression. FP-tree may fit in memory.

4. Frequent itemsets extracted from the FP-Tree.

Figure 6.24 shows a data set that contains ten transactions and five items.

Initially, the FP-tree contains only the root node represented by the null symbol. The FP-tree is subsequently extended in the following way:

1. The data set is scanned once to determine the support count of each item. Infrequent items are discarded, while the frequent items are sorted in decreasing support counts. For the data set shown in Figure 6.24, a is the most frequent item, followed by b, c, d, and e.

2.The algorithm makes a second pass over the data to construct the FP tree.

After reading the first transaction, {a,b), the nodes labeled as a and b are created. A path is then formed from null ,a, b to encode the transaction. Every node along the path has a frequency count of 1.

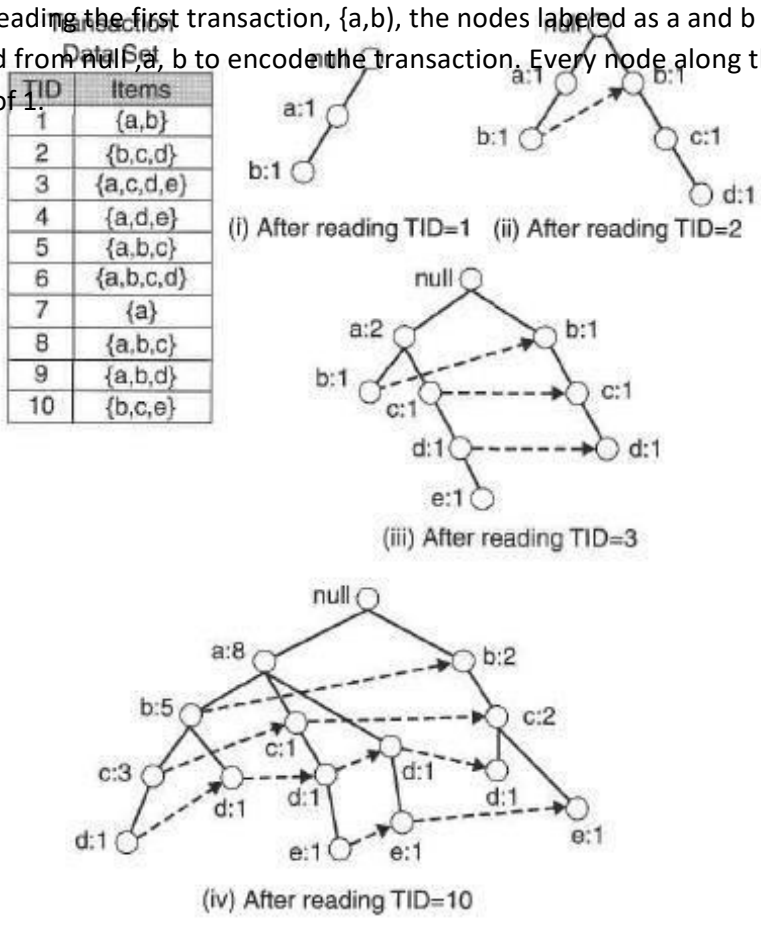


Figure 6.24. Construction of an FP-tree.

3. After reading the second transaction, {b,c,d}, a new set of nodes is created for items b, c, and

d. A path is then formed to represent the transaction by connecting the nodes null ,b,c, d. Every node along this path also has a frequency count equal to one. Although the first two transactions have an item in common, which is b, their paths are disjoint because the transactions do not share a common prefix.

The third transaction, {a,c,d,e}, shares a common prefix item (which is a) with the first transaction. As a result, the path for the third transaction null , a,c,d, e, overlaps with the path for the first transaction, nul,a ,b. Because of their overlapping path, the frequency count for node a is incremented to two, while the frequency counts for the newly created nodes, c, d, and e) are equal to one.

This process continues until every transaction has been mapped onto one of the paths given in the FP-tree. The resulting FP-tree after reading all the transactions is shown at the bottom of Figure 6.25.

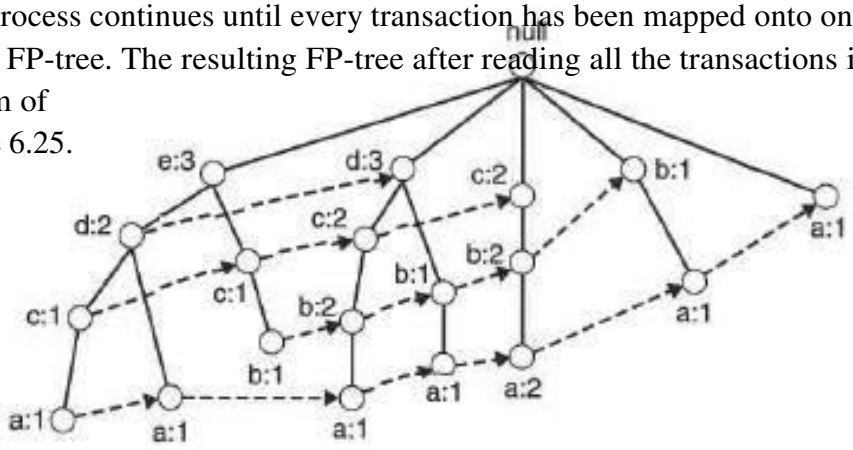


Figure 6.25. An FP-tree representation for the data set shown in Figure 6.24 with a different item ordering scheme.

Step 2: Frequent Itemset Generation

FP-growth is an algorithm that generates frequent itemsets from an FP-tree by exploring the tree in a bottom-up fashion.

Given the example tree shown in Figure 6.24, the algorithm looks for frequent itemsets ending in e first, followed by d, c, b, and finally, a. This bottom-up strategy for finding frequent itemsets ending with a particular item is equivalent to the suffix-based approach.

Since every transaction is mapped onto a path in the FP-tree, we can derive the frequent itemsets ending with a particular item, say e, by examining only the paths containing node e. These paths can be accessed rapidly using the pointers associated with node e. The extracted paths are shown in Figure 6.26(a)

After finding the frequent itemsets ending in e, the algorithm proceeds to look for frequent

itemsets ending in d by processing the paths associated with node d. The corresponding paths are shown in Figure 6.26(b). This process continues until all the paths associated with nodes c, b, and finally a are processed.

The paths for these items are shown in Figures 6.26(c), (d), and (e), while their corresponding frequent itemsets are summarized in Table 6.6

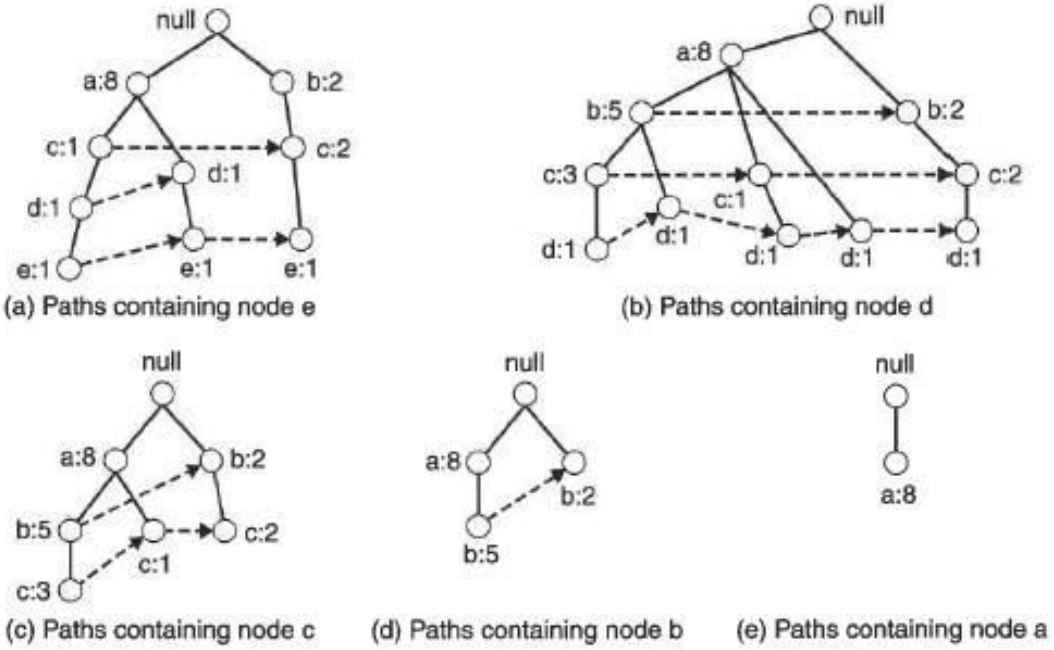


Figure 6.26. Decomposing the frequent itemset generation problem into multiple subproblems, where each subproblem involves finding frequent itemsets ending in e, d, c, b, and a.

Table 6.6. The list of frequent itemsets ordered by their corresponding suffixes.

Suffix	Frequent Itemsets
e	{e}, {d,e}, {a,d,e}, {c,e},{a,e}
d	{d}, {c,d}, {b,c,d}, {a,c,d}, {b,d}, {a,b,d}, {a,d}
c	{c}, {b,c}, {a,b,c}, {a,c}
b	{b}, {a,b}
a	{a}

Evaluation of Association Pattern :

Association rule algorithms tend to produce too many rules

- many of them are uninteresting or redundant

- Redundant if {A,B,C} · {D} and {A,B} · {D}

have same support & confidence

Interestingness measures can be used to prune/rank the derived patterns

Objective measures of interestingness

Given a rule X · Y, information needed to compute rule interestingness can be obtained from a contingency table $I(A,B) = \frac{s(A,B)}{s(A) \times s(B)} = \frac{Nf_{11}}{f_{1+}f_{+1}}$.

Contingency table for X · Y

Interest Factor

$$I(A,B) \begin{cases} = 1, & \text{if } A \text{ and } B \text{ are independent;} \\ > 1, & \text{if } A \text{ and } B \text{ are positively correlated;} \\ < 1, & \text{if } A \text{ and } B \text{ are negatively correlated.} \end{cases}$$

Example:

	Coffee	Coffee	
Tea	15	5	20
Tea	75	5	80
	90	10	100

For the tea-coffee example shown in Table 6.8, $I = \frac{0.15}{0.2 \times 0.8} = 0.9375$, thus suggesting a slight negative correlation between tea drinkers and coffee drinkers.

IS Measure *IS* is an alternative measure that has been proposed for handling asymmetric binary variables. The measure is defined as follows:

$$IS(A,B) = \sqrt{I(A,B) \times s(A,B)} = \frac{s(A,B)}{\sqrt{s(A)s(B)}}. \quad (6.9)$$

Correlation Analysis

For binary variables, correlation can be measured using the d-coefficient. which is defined as

$$\phi = \frac{f_{11}f_{00} - f_{01}f_{10}}{\sqrt{f_{1+}f_{+1}f_{0+}f_{+0}}}$$

The value of correlation ranges from -1 (perfect negative correlation) to +1 (perfect positive correlation). If the variables are statistically independent, then it is 0.

12. The original association rule mining formulation uses the support and confidence measures to prune uninteresting rules.

(a) Draw a contingency table for each of the following rules using the transactions shown in Table 6.4.

Rules: $\{b\} \longrightarrow \{d\}$, $\{e\} \longrightarrow \{c\}$, $\{c\} \longrightarrow \{a\}$.

Answer:

	c	$\overline{c}$
b	3	4
$\overline{b}$	2	1

	d	$\overline{d}$
a	4	1
$\overline{a}$	5	0

	d	$\overline{d}$
b	6	1
$\overline{b}$	3	0

	c	$\overline{c}$
e	2	4
$\overline{e}$	3	1

	a	$\overline{a}$
c	2	3
$\overline{c}$	3	2

(b) Use the contingency tables in part (a) to compute and rank the rules in decreasing order according to the following measures.

17. Suppose we have market basket data consisting of 100 transactions and 20 items. If the support for item a is 25%, the support for item b is 90% and the support for itemset $\{a, b\}$ is 20%. Let the support and confidence thresholds be 10% and 60%, respectively.

(a) Compute the confidence of the association rule $\{a\} \rightarrow \{b\}$. Is the rule interesting according to the confidence measure?

Answer:

Confidence is $0.2/0.25 = 80\%$. The rule is interesting because it exceeds the confidence threshold.

(b) Compute the interest measure for the association pattern $\{a, b\}$. Describe the nature of the relationship between item a and item b in terms of the interest measure.

Answer:

The interest measure is $0.2/(0.25 \times 0.9) = 0.889$. The items are negatively correlated according to interest measure.

$b \longrightarrow d$	0.6	1
$e \longrightarrow e$	0.2	4
$c \longrightarrow a$	0.2	4

ii. Confidence.

Answer:

Rules	Confidence	Rank
$b \longrightarrow c$	3/7	3
$a \longrightarrow d$	4/5	2
$b \longrightarrow d$	6/7	1
$e \longrightarrow e$	2/6	5
$c \longrightarrow a$	2/5	4

iii. $\text{Interest}(X \longrightarrow Y) = \frac{P(X,Y)}{P(X)}P(Y)$.

Answer:

Rules	Interest	Rank
$b \longrightarrow c$	0.771	3
$a \longrightarrow d$	0.72	2
$b \longrightarrow d$	0.771	1
$e \longrightarrow e$	0.96	5
$c \longrightarrow a$	0.2	4

iv. $\text{IS}(X \longrightarrow Y) = \frac{P(X,Y)}{\sqrt{P(X)P(Y)}}$.

Answer:

Rules	IS	Rank
$b \longrightarrow c$	0.507	3
$a \longrightarrow d$	0.596	2
$b \longrightarrow d$	0.756	1
$e \longrightarrow e$	0.365	5
$c \longrightarrow a$	0.4	4

Question Bank:

1. What is association analysis? Define support and confidence with an example.
2. Develop the appriori algorithm for frequent itemset generation, with an example.
3. Explain the various measure of evaluating association patterns.
4. Explain in detail frequent itemset generation and rule generation with reference to appriori along with an example.
5. Define following: a) Support b) Confidence.
6. Explain FP growth algorithm for discovering frequent item sets. What are its limitation.
7. Consider following transaction data set

TID	ITEM

1	{a, b}
2	{b, c, d}
3	{a, c, d, e}
4	{a, d, e}
5	{a, b, c}
6	{a, b, c, d}
7	{a}
8	{a, b, c}

9 {a, b, d}

10 {b, c, e}

Construct the FP tress by showing the tress separately after reading each transaction.

8. Illustrate the limitations of support confidence framework for evaluation of an association rule

9. Define cross support pattern. Suppose the support for milk is 70%, support for sugar is 10% and support for bread is 0.04%. givenh_c= 0.01. is the frequent item set {milk, sugar, bread} the cross-support pattern?

10.Which are the factors affecting the computational complexity of appriori algorithm? Explain them.

11.Define a frequent pattern tree. Discuss the method of computing a FP-Tree, with an algorithm.

12.Give an example to show that items in a strong association rule may actually be negatively corelated.

13.A database has five transactions. Let min-sup = 60% and min-conf = 80%

TID	ITEM
T1	{M, O, N, K, E, Y}
T2	{D, O, N, K, E, Y}
T3	{M, A, K, E}
T4	{M, U, C, K, Y}
T5	{C, O, O, K, I, E}

Find all frequent item sets using appriori and FP growth respectively,

14.Explain various alternative methods for generating frequent item sets.

15.A database has four transactions. Let min-sup = 40% and min-conf = 60%

TID	DATE	ITEM
T1	01/01/10	{K, A, D, B}
T2	01/01/10	{D, A, C, E, B}
T3	01/15/10	{C, A, B, E}
T4	01/22/10	{B, A, D}

Find all frequent item sets using appriori and FP growth algorithms. Compare the efficiency of two measuring process.

16.Explain various Candidate Generation and Pruning techniques.

17.Explain the various properties of objective measures.

18.Comprehend the Simpson's Paradox.

19.Illustrate the nature of Simpson's paradox for the following two-way contingency table

Buy HDTV	Buy Exercise machine		
	yes	no	
yes	99	81	180
No	54	66	120
	153	147	300

20.What is appriori algorithm? Give an example. A database has six transactions of purchase

of books from a book shop as given below

TID	ITEM
T1	{ANN, CC, JC, CG}
T2	{CC, D, CG}
T3	{ANN, D, CC, TC}
T4	{ANN, CC, D, CG}
T5	{ANN, CC, D, TC, CG}

T6	{C, D, TC}

Let X= {CC, TC} and Y= {ANN, TC, CC} find confidence and support of the association rule X· Y and inverse rule Y· X

21.Consider the following transaction data set:

TID	ITEM
T100	I_1, I_2, I_5
T200	I_2, I_4
T300	I_2, I_3
T400	I_1, I_2, I_4
T500	I_1, I_3
T600	I_2, I_3
T700	I_1, I_3
T800	I_1, I_2, I_3, I_5
T900	I_1, I_2, I_3

Construct FP Tree.Generate List of frequent item set ordered by their corresponding suffixes.

22.Consider following set of frequent 3 item sets

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

Assume that there are only 5 items in data set.

a) List all candidate 4 item sets obtained by a candidate generation procedure

using F_{k-1} X F_1 merging strategy

b) List all candidate 4 item sets obtained by the candidate generation procedure

in appriori,

23.Apply appriori algorithm for

TID	ITEM
101	Milk, Bread, Eggs
102	Milk, Juice
103	Juice, Butter
104	Milk, Bread, Eggs
105	Coffee, Eggs
106	Coffee
107	Coffee, Juice
108	Milk, Bread, Cookies, Eggs
109	Cookies, Butter
110	Milk, Bread

Item set = {Milk, Bread, Eggs, Cookies, Coffee, Butter, Juice}, use 0.2 for min-sup

MODULE-4

Classification: Basic Concepts, Decision Trees, and Model Evaluation.

Classification: Definition

- ✓ Classification, which is the task of assigning objects to one of several predefined categories.
- ✓ 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. The input data for a classification task is a collection of records. Each record,

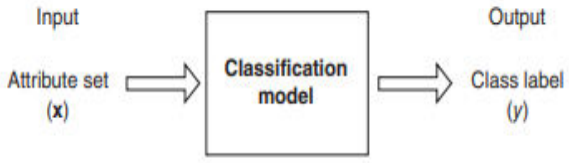


Figure 3.2. A schematic illustration of a classification task.

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.

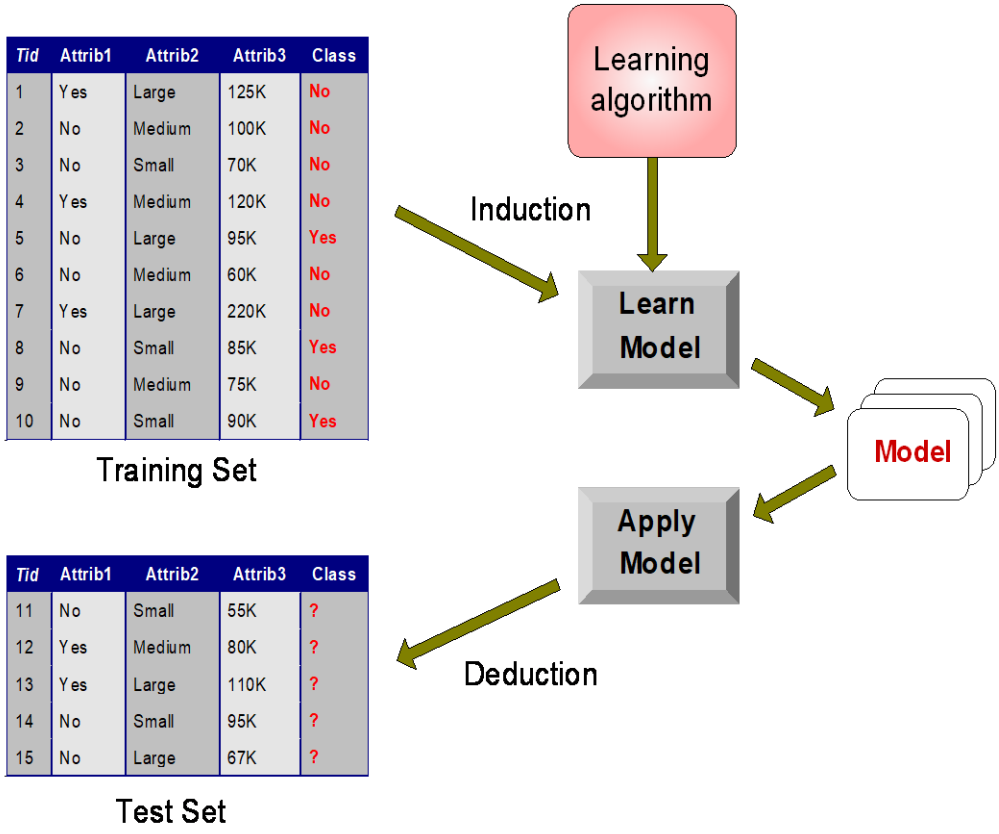
Applications:

Examples

- ✓ Detecting spam email messages based upon the message header and content.
- ✓ Categorizing cells as malignant or benign based upon the results of MRI scans.
- ✓ Classifying galaxies based upon their shapes.
- ✓ Categorizing news stories as finance, weather, entertainment, sports, etc
- ✓ Classifying credit card transactions as legitimate or fraudulent.

General Approach to Solving a Classification

- ✓ A classification technique (or classifier) is a systematic approach to building
- ✓ Classification models from an input data set.
- ✓ Each technique employs a learning algorithm to identify a model that best fits the relationship between the attribute set and class label of the input data.
- ✓ The model generated by a learning algorithm should both fit the input data well and correctly predict the class labels of records it has never seen before.
- ✓ Therefore, a key objective of the learning algorithm is to build models with good generalization capability; i.e., models that accurately predict the class labels of previously unknown records



Evaluation of the performance of a classification model is based on the counts of test records correctly and incorrectly predicted by the model. These counts are tabulated in a table known as a confusion matrix.

Table 3.4. Confusion matrix for a binary classification problem.

		Predicted Class	
		Class = 1	Class = 0
Actual Class	Class = 1	f_{11}	f_{10}
	Class = 0	f_{01}	f_{00}

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}}. \tag{3.1}$$

For binary classification problems, the accuracy of a model is given by

$$\text{Accuracy} = \frac{f_{11} + f_{00}}{f_{11} + f_{10} + f_{01} + f_{00}}. \tag{3.2}$$

Error rate is another related metric, which is defined as follows for binary classification problems:

$$\text{Error rate} = \frac{\text{Number of wrong predictions}}{\text{Total number of predictions}} = \frac{f_{10} + f_{01}}{f_{11} + f_{10} + f_{01} + f_{00}}. \tag{3.3}$$

Most classification algorithms seek models that attain the highest accuracy, or equivalently, the lowest error rate when applied to the test set.

Classification Techniques:

- Decision Tree based Methods
- Rule-based Methods
- Memory based reasoning
- Neural Networks
- Naïve Bayes and Bayesian Belief Networks
- Support Vector Machines

Decision Tree

The tree has three types of nodes:

A ***root node,*** that has no incoming edges and zero or more outgoing edges.

Internal nodes, each of which has exactly one incoming edge and two or more outgoing edges.

Leaf or terminal nodes, each of which has exactly one incoming edge and no outgoing edges.

In a decision tree, each leaf node is assigned a class label. The non terminal nodes, which include the root and other internal nodes, contain attribute test conditions to separate records that have different characteristics

In principle, there are exponentially many decision trees that can be constructed from a given set of attributes.

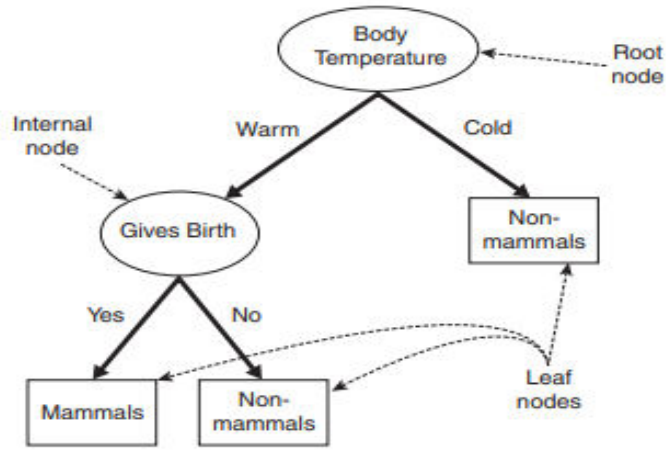


Figure 3.4. A decision tree for the mammal classification problem.

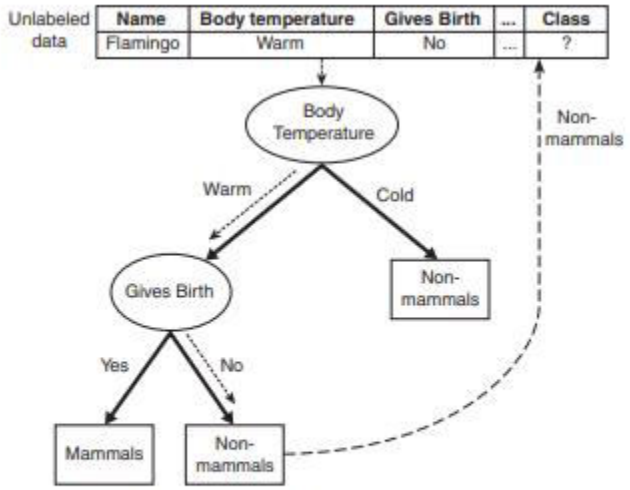


Figure 3.5. Classifying an unlabeled vertebrate. The dashed lines represent the outcomes of applying various attribute test conditions on the unlabeled vertebrate. The vertebrate is eventually assigned to the `Non-mammals` class.

Hunt's Algorithm

In Hunt's algorithm, a decision tree is grown in a recursive fashion by partitioning the training records into successively purer subsets. Let Di be the set of training records that are associated with node t and y= {“y1,y2,y3….yc"} be the class labels. The following is a recursive definition of Hunt's algorithm.

Step 1: If all the records in Dt belong to the same class yt, then t is a leaf node labeled as yt.

Step 2: If Di contains records that belong to more than one class, an attribute test condition is selected to partition the records into smaller subsets. A child node is created for each outcome of the test condition and the records in Dt are distributed to the children based on the outcomes. The algorithm is then recursively applied to each ***child node.***

	binary	categorical	continuous	class
Tid	Home Owner	Marital Status	Annual Income	Defaulted Borrower
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

Figure 4.6. Training set for predicting borrowers who will default on loan payments.

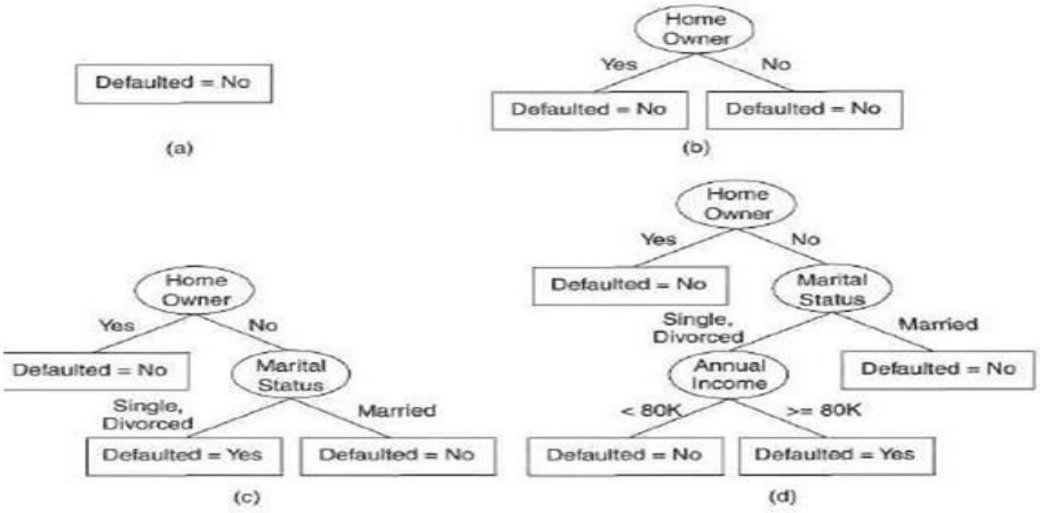


Figure 4.7. Hunt's algorithm for inducing decision trees.

To illustrate how the algorithm works, consider the problem of predicting whether a loan applicant will repay her loan obligations or become delinquent, subsequently defaulting on her loan.

The initial tree for the classification problem contains a single node with class label Defaulted = No (see Figure 4.7a), which means that most of the borrowers successfully repaid their loans. The tree, however, needs to be refined since the root node contains records from both classes.

The records are subsequently divided into smaller subsets based on the outcomes of the Home Owner test condition as shown in Figure 4.7(b). The justification for choosing this attribute test condition will be discussed later. For now, we will assume that this is the best criterion for splitting the data at this point.

Hunt's algorithm is then applied recursively to each child of the root node. From the training set given in Figure 4.6, notice that all borrowers who are home owners successfully repaid their loans. The left child of the root is therefore a leaf node labeled Defaulted = No (see Figure 4.7(b)).

For the right child, we need to continue applying the recursive step of Hunt's algorithm until all the records belong to the same class. The trees resulting from each recursive step are shown in Figures 4.7(c) and (d).

Design Issues of Decision Tree Induction:

A learning algorithm for inducing decision trees must address the following two issues.

1) Should the training records be split?

Each recursive step of the tree-growing process must select an attribute test condition to divide the records into smaller subsets. To implement this step, the algorithm must provide a method for specifying the test condition for different attribute types as well as an objective measure for evaluating the goodness of each test condition.

2) How should the splitting procedure stop?

A stopping condition is needed to terminate the tree-growing process. A possible strategy is to continue expanding a node until either all the records belong to the same class or all the records have identical attribute values. Although both conditions are sufficient to stop any decision tree induction algorithm, other criteria can be imposed to allow the tree-growing procedure to terminate earlier.

Methods for Expressing Attribute Test Conditions:

Decision tree induction algorithms must provide a method for expressing an attribute test condition and its corresponding outcomes for different attribute types.

Binary Attributes: The test condition for a binary attribute generates two potential outcomes, as shown in Figure 4.8.

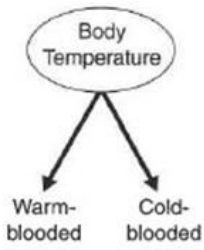


Figure 4.8. Test condition for binary attributes.

Nominal Attributes: Since a nominal attribute can have many values; its test condition can be expressed in two ways, as shown in Figure 4.9. For a multiway split (Figure 4.9(a)), the number of outcomes depends on the number of distinct values for the corresponding attribute. For example, if an attribute such as marital status has three distinct values-single, married, or divorced-its test condition will produce a three-way split.

Figure 4.9(b) illustrates three different ways of grouping the attribute values for marital status into two subsets.

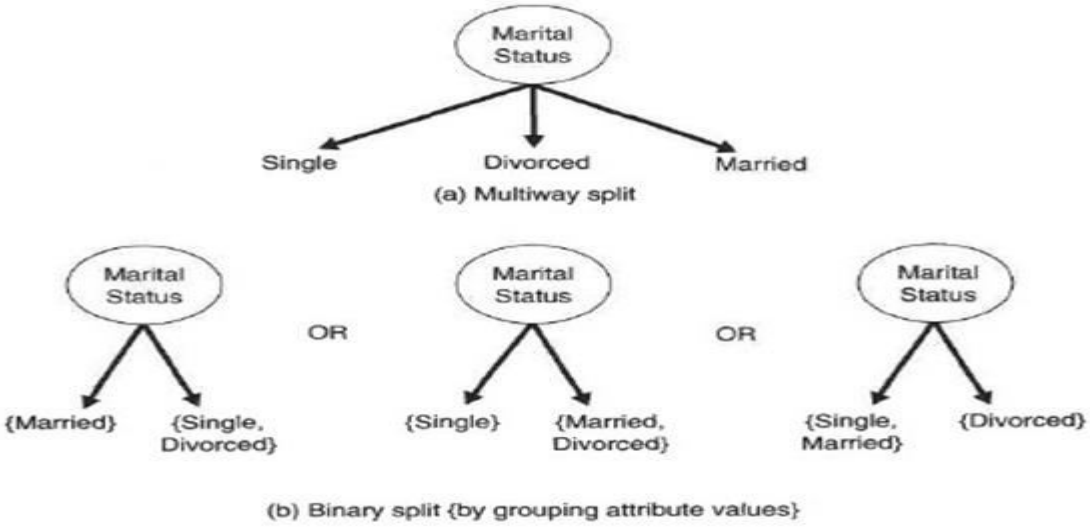


Figure 4.9. Test conditions for nominal attributes.

Ordinal Attributes: Ordinal attributes can also produce binary or multiway splits. Ordinal attribute values can be grouped as long as the grouping does not violate the order property of the attribute values. Figure 4.10 illustrates various ways of splitting training records based on the Shirt Size attribute.

The groupings shown in Figures 4.10(a) and (b) preserve the order among the attribute values, whereas the grouping shown in Figure a.10(c) violates this property because it combines the attribute values Small and Large into the same partition while Medium and Extra Large are combined into another partition.

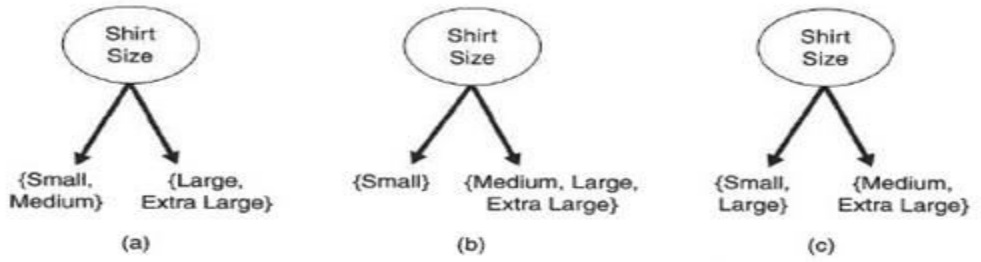


Figure 4.10. Different ways of grouping ordinal attribute values.

Continuous Attributes: For continuous attributes, the test condition can be expressed as a comparison test (A < V) or (A>=V ,) with binary outcomes, or a range query with outcomes of the form Vi<=A<Vi+1, for i=1,2…k. The difference between these approaches is shown in Figure 4.11.

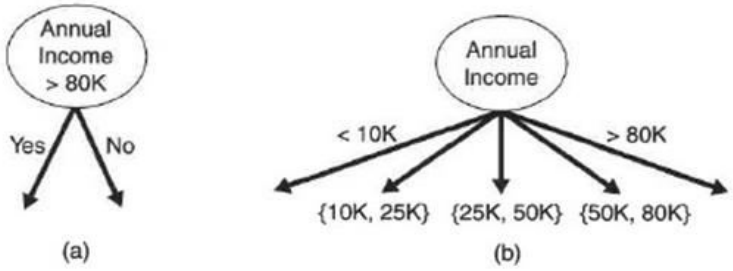


Figure 4.11. Test condition for continuous attributes.

How to determine the Best Split:

Greedy approach:

- Nodes with homogeneous class distribution are preferred Need a measure of node impurity:

Non-homogeneous, High degree of impurity

Homogeneous, Low degree of impurity

Measures of Node Impurity:

- Gini Index
- Entropy
- Misclassification error

$$\text{Entropy}(t) = -\sum_{i=0}^{c-1} p(i|t)\log_2 p(i|t),$$

$$\text{Gini}(t) = 1 - \sum_{i=0}^{c-1}[p(i|t)]^2,$$

$$\text{Classification error}(t) = 1 - \max_i[p(i|t)],$$

where c is the number of classes and $0 \log_2 0 = 0$ in entropy calculations.

Where **p(i|t)** denote the fraction of records belonging to class **i** at a given node **t** and where c is the number of classes.

The measures developed for selecting the best split are often based on the degree of impurity of the child nodes. The smaller the degree of impurity, the more skewed the class distribution.

Node N_1	Count
Class=0	0
Class=1	6

Gini $= 1 - (0/6)^2 - (6/6)^2 = 0$
Entropy $= -(0/6)\log_2(0/6) - (6/6)\log_2(6/6) = 0$
Error $= 1 - \max[0/6, 6/6] = 0$

Node N_2	Count
Class=0	1
Class=1	5

Gini $= 1 - (1/6)^2 - (5/6)^2 = 0.278$
Entropy $= -(1/6)\log_2(1/6) - (5/6)\log_2(5/6) = 0.650$
Error $= 1 - \max[1/6, 5/6] = 0.167$

Node N_3	Count
Class=0	3
Class=1	3

Gini $= 1 - (3/6)^2 - (3/6)^2 = 0.5$
Entropy $= -(3/6)\log_2(3/6) - (3/6)\log_2(3/6) = 1$
Error $= 1 - \max[3/6, 3/6] = 0.5$

Node N1 has the lowest impurity value, followed by N2 and N3.

To determine how well a test condition performs, we need to compare the degree of impurity of the parent node (before splitting) with the degree of impurity of the child nodes (after splitting). The larger their difference, the better the test condition. The gain is a criterion that can be used to determine the goodness of a split.

$$\Delta = I(\text{parent}) - \sum_{j=1}^{k} \frac{N(v_j)}{N} I(v_j), \quad (4.6)$$

where $I(\cdot)$ is the impurity measure of a given node, N is the total number of records at the parent node, k is the number of attribute values, and $N(v_j)$ is the number of records associated with the child node, v_j. Decision tree

Characteristics of Decision Tree Based Classification:

Advantages:

- ✓ Decision tree induction is a nonparametric approach for building classification models. In other words, it does not require any prior assumptions regarding the type of probability distributions satisfied by the class and other attributes.
- ✓ Finding an optimal decision tree is an NP-complete problem
- ✓ Techniques developed for constructing decision trees are computationally inexpensive, making it possible to quickly construct models even when the training set size is very large. Once a decision tree has been built, classifying a test record is extremely fast, with a worst-case complexity of O(W), where ,W is the maximum depth of the tree.
- ✓ Decision trees, especially smaller-sized trees, are relatively easy to interpret.
- ✓ Decision tree algorithms are quite robust to the presence of noise.
- ✓ The presence of redundant attributes does not adversely affect the accuracy of decision trees.

Disadvantages:

- ✓ Since most decision tree algorithms employ a top-down, recursive partitioning approach, the number of records becomes smaller as we traverse down the tree. At the leaf nodes, the number of records may be too small to make a statistically significant decision about the class representation of the nodes.

- ✓ A subtree can be replicated multiple times in a decision tree, as illustrated in Figure 4.19. This makes the decision tree more complex than necessary and perhaps more difficult to interpret. Such a situation can arise from decision tree implementations that rely on a single attribute test condition at each internal node.

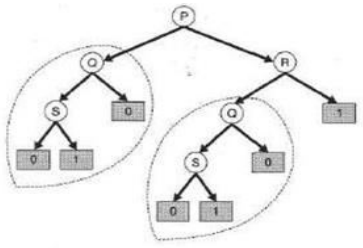


Figure 4.19. Tree replication problem. The same subtree can appear at different branches.

Model Over fitting:

The errors committed by a classification model are generally divided into two types: training errors and generalization errors.

Training error, is the number of misclassification errors committed on training records,

Generalization error is the expected error of the model on test records.

A good model must have low training error as well as low generalization error.

Underfitting: The training and test error rates of the model are large when the size of the tree is very small. This situation is known as model underfitting. Underfitting occurs because the model has yet to learn the true structure ofthe data. As a result, it performs poorly on both the training and the test sets.

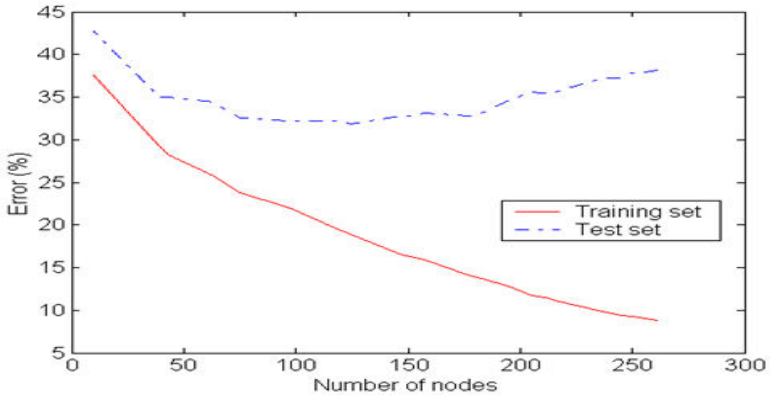
Overfitting.: As the number of nodes in the decision tree increases, the tree will have fewer training and test error . However, once the tree becomes too large, its test error rate begins to increase even though its training error rate continues to decrease. This phenomenon is known as model over fitting.

Reasons for over fitting:

- Presence of Noise

- Lack of Representative Samples

Figure shows the training and test error rates of the decision tree.



Estimating Generalization Errors:

Generalization errors: error on testing (Σ e'(t))

Methods for estimating generalization errors:

1) Optimistic approach: e'(t) = e(t)

2) Pessimistic approach:

- ✓ For each leaf node: e'(t) = (e(t)+0.5)
- ✓ Total errors: e'(T) = e(T) + N × 0.5 (N: number of leaf nodes)
  - Ex: For a tree with 30 leaf nodes and 10 errors on training (out of 1000 instances): Training error = 10/1000 = 1%
  - Generalization error = (10 + 30×0.5)/1000 = 2.5%

- Reduced error pruning (REP):
  - ✓ Uses validation data set to estimate generalization error

How to address over fitting:

1. **Pre-Pruning (Early Stopping Rule)**
   - Stop the algorithm before it becomes a fully-grown tree
   - Typical stopping conditions for a node:
     - Stop if all instances belong to the same class
     - Stop if all the attribute values are the same
   - More restrictive conditions:
     - Stop if number of instances is less than some user-specified threshold
     - Stop if class distribution of instances are independent of the available features (e.g., using χ^2 test)
     - Stop if expanding the current node does not improve impurity measures (e.g., Gini or information gain).
2. **Post-pruning**
   - Grow decision tree to its entirety
   - Trim the nodes of the decision tree in a bottom-up fashion
   - If generalization error improves after trimming, replace sub-tree by a leaf node.
   - Class label of leaf node is determined from majority class of instances in the sub-tree

Rule-Based Classifier

A rule-based classifier is a technique for classifying records using a collection of "if . . .then. . ." rules.

The rules for the model are represented in a disjunctive normal form, where R is known as the rule set and r;'s are the classification rules or disjuncts

Each classification rule can be expressed in the following way:

$$r_i : \quad (Condition_i) \longrightarrow y_i.$$

The left-hand side of the rule is called the rule antecedent or precondition.

The right-hand side of the rule is called the rule consequent, which contains the predicted class yi

Rule-based Classifier (Example)

Name	Blood Type	Give Birth	Can Fly	Live in Water	Class
human	warm	yes	no	no	mammals
python	cold	no	no	no	reptiles
salmon	cold	no	no	yes	fishes
whale	warm	yes	no	yes	mammals
frog	cold	no	no	sometimes	amphibians
komodo	cold	no	no	no	reptiles
bat	warm	yes	yes	no	mammals
pigeon	warm	no	yes	no	birds
cat	warm	yes	no	no	mammals
leopard shark	cold	yes	no	yes	fishes
turtle	cold	no	no	sometimes	reptiles
penguin	warm	no	no	sometimes	birds
porcupine	warm	yes	no	no	mammals
eel	cold	no	no	yes	fishes
salamander	cold	no	no	sometimes	amphibians
gila monster	cold	no	no	no	reptiles
platypus	warm	no	no	no	mammals
owl	warm	no	yes	no	birds
dolphin	warm	yes	no	yes	mammals
eagle	warm	no	yes	no	birds

R1: (Give Birth = no) $\wedge$ (Can Fly = yes) $\longrightarrow$ Birds
R2: (Give Birth = no) $\wedge$ (Live in Water = yes) $\longrightarrow$ Fishes
R3: (Give Birth = yes) $\wedge$ (Blood Type = warm) $\longrightarrow$ Mammals
R4: (Give Birth = no) $\wedge$ (Can Fly = no) $\longrightarrow$ Reptiles
R5: (Live in Water = sometimes) $\longrightarrow$ Amphibians

Name	Blood Type	Give Birth	Can Fly	Live in Water	Class
hawk	warm	no	yes	no	?
grizzly bear	warm	yes	no	no	?

The rule R1 covers a hawk => Bird

The rule R3 covers the grizzly bear => Mammal

Name	Blood Type	Give Birth	Can Fly	Live in Water	Class
lemur	warm	yes	no	no	?
turtle	cold	no	no	sometimes	?
dogfish shark	cold	yes	no	yes	?

A lemur triggers rule R3, so it is classified as a mammal

A turtle triggers both R4 and R5

A dogfish shark triggers none of the rules

Rule Coverage and Accuracy

Coverage of a rule:

Fraction of records that satisfy the antecedent of a rule

Accuracy of a rule:

Fraction of records that satisfy both the antecedent and consequent of a rule

$$\text{Coverage}(r) = \frac{|A|}{|D|}$$

$$\text{Accuracy}(r) = \frac{|A \cap y|}{|A|}, \qquad (5.3)$$

where $|A|$ is the number of records that satisfy the rule antecedent, $|A \cap y|$ is the number of records that satisfy both the antecedent and consequent, and $|D|$ is the total number of records.

ID	Home Owner	Marital Status	Annual Income	Defaulted Borrower
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

Mutually Exclusive Rules The rules in a rule set R are mutually exclusive if no two rules in .R are triggered by the same record. This property ensures that every record is covered by at most one rule in R.

Exhaustive Rules A rule set -R has exhaustive coverage if there is a rule for each combination of attribute values. This property ensures that every record is covered by at least one rule in –R

Ordered Rules In this approach, the rules in a rule set are ordered in decreasing order of their priority, which can be defined in many ways (e.g., based on accuracy, coverage, total description length, or the order in which the rules are generated). An ordered rule set is also known as a decision list. When a test record is presented, it is classified by the highest-ranked rule that covers the record. This avoids the problem of having conflicting classes predicted by multiple classification rules

Rule-Ordering Schemes

Rule-based ordering

Individual rules are ranked based on their quality

- This approach orders the individual rules by some rule quality measure.
- This ordering scheme ensures that every test record is classified by the "best" rule covering it.

Class-based ordering

Rules that belong to the same class appear together

In this approach, rules that belong to the same class appear together in the rule set R. The rules are then collectively sorted on the basis of their class information.

Rule-based Ordering

(Refund=Yes) ==> No

(Refund=No, Marital Status={Single,Divorced}, Taxable Income<80K) ==> No

(Refund=No, Marital Status={Single,Divorced},

Class-based Ordering

(Refund=Yes) ==> No

(Refund=No, Marital Status={Single,Divorced}, Taxable Income<80K) ==> No

(Refund=No, Marital Status={Married}) ==> No

Characteristics of Rule-Based Classifiers:

A rule-based classifier has the following characteristics:

- The expressiveness of a rule set is almost equivalent to that of a decision tree because a decision tree can be represented by a set of mutually exclusive and exhaustive rules. Both rule-based and decision tree classifiers create rectilinear partitions of the attribute space and assign a class to each partition. Nevertheless, if the rule-based classifier allows multiple rules to be triggered for a given record, then a more complex decision boundary can be constructed.

- Rule-based classifiers are generally used to produce descriptive models that are easier to interpret, but gives comparable performance to the decision tree classifier.

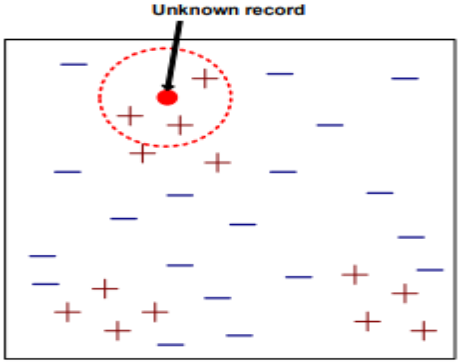
Nearest-Neighbor Classifiers

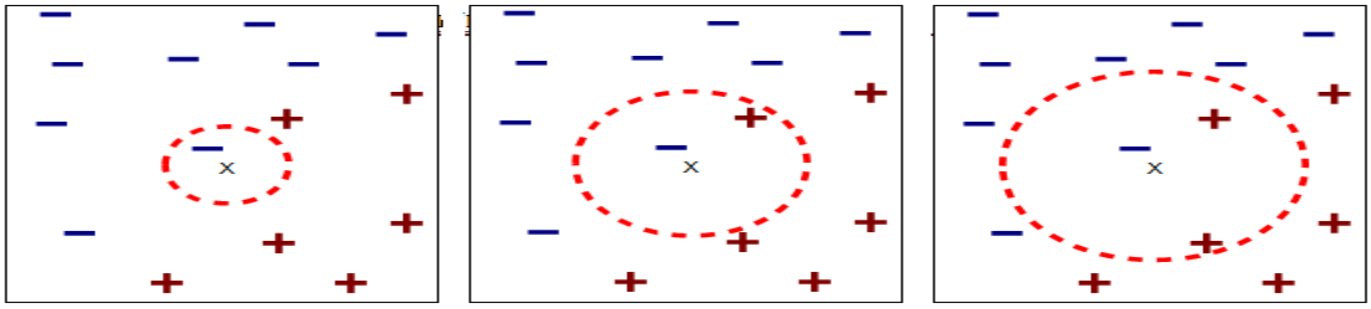
Requires three things

- 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)





(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 Compute distance between two points:

- Euclidean distance

$$d(x,y) = \sqrt{\sum_i (x_i - y_i)^2}$$

Determine the class from nearest neighbor list

- take the majority vote of class labels among the k-nearest neighbors

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

Characteristics of Nearest-Neighbor Classifiers:

- Nearest-neighbor classification is part of a more general technique known as instance-based learning, which uses specific training instances to make predictions without having to maintain an abstraction (or model) derived from data. Instance-based learning algorithms require a proximity measure to determine the similarity or distance between instances and a classification function that returns the predicted class of a test instance based on its proximity to other instances.
- Lazy learners such as nearest-neighbor classifiers do not require model building. However, classifying a test example can be quite expensive because we need to compute the proximity values individually between the test and training examples.
- Nearest-neighbor classifiers can produce arbitrarily shaped decision boundaries. Such boundaries provide a more flexible model representation compared to decision tree and rule-based classifiers that are often constrained to rectilinear decision boundaries.
- Nearest-neighbor classifiers can produce wrong predictions unless the appropriate proximity measure and data preprocessing steps are taken.

Bayes' Theorem:

Bayes' theorem is a way to figure out conditional probability. Conditional probability is the probability of an event happening, given that it has some relationship to one or more other events.

$$P(C \mid A) = \frac{P(A \mid C)P(C)}{P(A)}$$

Bayes' Theorem Problems Example #1

In a particular pain clinic, 10% of patients are prescribed narcotic pain killers. Overall, five percent of the clinic's patients are addicted to narcotics (including pain killers and illegal substances). Out of all the people prescribed pain pills, 8% are addicts. *If a patient is an addict, what is the probability that they will be prescribed pain pills?*

Step 1: **Figure out what your event "A" is from the question.** That information is in the italicized part of this particular question. The event that happens first (A) is being prescribed pain pills. That's given as 10%.

Step 2: **Figure out what your event "B" is from the question.** That information is also in the italicized part of this particular question. Event B is being an addict. That's given as 5%.

Step 3: **Figure out what the probability of event B (Step 2) given event A (Step 1)**. In other words, find what (B|A) is. We want to know "Given that people are prescribed pain pills, what's the probability they are an addict?" That is given in the question as 8%, or .8.

Step 4: **Insert your answers from Steps 1, 2 and 3 into the formula and solve.**

P(A|B) = P(B|A) * P(A) / P(B) = (0.08 * 0.1)/0.05 = 0.16

The probability of an addict being prescribed pain pills is 0.16 (16%).

Bayes' Theorem Problems Example #2

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 stiff neck, what's the probability he/she has meningitis?

$$P(M \mid S) = \frac{P(S \mid M)P(M)}{P(S)} = \frac{0.5 \times 1/50000}{1/20} = 0.0002$$

Using the Bayes Theorem for Classification:

Let X denotes the attribute set and Y denote the class variable. If the class variable has a non-deterministic relationship with the attributes, then we can treat X and Y as random variables and capture their relationship probabilistically using P(Y/X). This conditional probability is also known as the posterior probability for Y, as opposed to its prior probability, P(Y).

During the training phase, we need to learn the posterior probabilities P(Y/X) for every combination of X and Y based on information gathered from the training data.

By knowing these probabilities, a test record X' can be classified by finding the class Y' that maximizes the posterior probability, P(Y'/X').

To illustrate this approach, consider the task of predicting whether a loan borrower will default on their payments.

Figure 5.9 shows a training set with the following attributes: House Owner, Marital Status, and Annual Income. Loan borrowers who defaulted on their payments are classified as Yes, while those who repaid their loans are classified as No

	binary	categorical	continuous	class
Tid	Home Owner	Marital Status	Annual Income	Defaulted Borrower
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

Figure 5.9. Training set for predicting the loan default problem.

Suppose we are given a test record with the following attribute set:

X :(Home Owner : No, Marital Status : Married, Annual Income : $120K).

To classify the record, we need to compute the posterior probabilities P(Yes/X) and P(No/X) based on information available in the training data.

If **P(Yes/X) > P(No/X),** then the record is classified as Yes; otherwise, it is classified as No

5.3.3 Naïve Bayes Classifier

A naïve Bayes classifier estimates the class-conditional probability by assuming that the attributes are conditionally independent, given the class label y. The conditional independence assumption can be formally stated as follows:

$$P(\mathbf{X}|Y = y) = \prod_{i=1}^{d} P(X_i|Y = y), \tag{5.12}$$

where each attribute set $\mathbf{X} = \{X_1, X_2, \ldots, X_d\}$ consists of d attributes.

How a Naïve Bayes Classifier Works

With the conditional independence assumption, instead of computing the class-conditional probability for every combination of $\mathbf{X}$, we only have to estimate the conditional probability of each X_i, given Y. The latter approach is more practical because it does not require a very large training set to obtain a good estimate of the probability.

To classify a test record, the naïve Bayes classifier computes the posterior probability for each class Y:

$$P(Y|\mathbf{X}) = \frac{P(Y)\prod_{i=1}^{d} P(X_i|Y)}{P(\mathbf{X})}. \tag{5.15}$$

Estimating Conditional Probabilities for Categorical Attributes

For a categorical attribute X_i, the conditional probability $P(X_i = x_i|Y = y)$ is estimated according to the fraction of training instances in class y that take on a particular attribute value x_i. For example, in the training set given in Figure 5.9, three out of the seven people who repaid their loans also own a home. As a result, the conditional probability for $P(\texttt{Home Owner=Yes}|\texttt{No})$ is equal to 3/7. Similarly, the conditional probability for defaulted borrowers who are single is given by $P(\texttt{Marital Status} = \texttt{Single}|\texttt{Yes}) = 2/3$.

Estimating Conditional Probabilities for Continuous Attributes:

There are two ways to estimate the class-conditional probabilities for continuous Attributes in naive Bayes classifiers:

1. We can discretize each continuous attribute and then replace the continuous attribute value with its corresponding discrete interval. This approach transforms the continuous attributes into ordinal attributes. The conditional probability $P(X_i|Y = y)$ is estimated by computing the fraction of training records belonging to class y that falls within the corresponding interval for X_i. The estimation error depends on the dis-

2. We can assume a certain form of probability distribution for the continuous variable and estimate the parameters of the distribution using the training data. A Gaussian distribution is usually chosen to represent the class-conditional probability for continuous attributes. The distribution is characterized by two parameters, its mean, μ, and variance, σ^2. For each class y_j, the class-conditional probability for attribute X_i is

$$P(X_i = x_i|Y = y_j) = \frac{1}{\sqrt{2\pi}\sigma_{ij}} \exp^{-\frac{(x_i-\mu_{ij})^2}{2\sigma_{ij}^2}}. \qquad (5.16)$$

The parameter μ_{ij} can be estimated based on the sample mean of X_i ($\bar{x}$) for all training records that belong to the class y_j. Similarly, σ_{ij}^2 can be estimated from the sample variance (s^2) of such training records. For

Given a test record with taxable income equal to \$120K, we can compute its class-conditional probability as follows:

$$P(\texttt{Income=120}|\texttt{No}) = \frac{1}{\sqrt{2\pi}(54.54)} \exp^{-\frac{(120-110)^2}{2\times 2975}} = 0.0072.$$

M-estimate of Conditional Probability:

$$P(x_i|y_j) = \frac{n_c + mp}{n + m},$$

where n is the total number of instances from class Yj, nc is the number of training examples from class Yi that take on the value Xi, m is a parameter known as the equivalent sample size, and p is a user-specified parameter.

Characteristics of Naive Bayes Classifiers:

Naive Bayes classifiers generally have the following characteristics:

- They are robust to isolated noise points because such points are averaged out when estimating conditional probabilities from data.
- Naive Bayes classifiers can also handle missing values by ignoring the example during model building and classification.
- They are robust to irrelevant attributes.
- Correlated attributes can degrade the performance of naive Bayes classifiers because the conditional independence assumption no longer holds for such attributes.

Bayesian Belief Networks

Bayesian networks represent an advanced form of general Bayesian probability

A Bayesian network is a graphical model that encodes probabilistic relationships among variables of interest

A Bayesian belief network (BBN), or simply, Bayesian network, provides a graphical representation of the probabilistic relationships among a set of random variables. There are two key elements of a Bayesian network:

1. A directed acyclic graph (dag) encoding the dependence relationships among a set of variables.

2. A probability table associating each node to its immediate parent nodes

Consider three random variables,A, B, and C, in which A and B are independent variables and each has a direct influence on a third variable, C.

The relationships among the variables can be summarized into the directed acyclic graph shown in Figure 5.12(a).

Each node in the graph represents a variable, and each arc asserts the dependence relationship between the pair of variables. If there is a directed arc from X to Y, then X is the parent of Y and Y is the child of X. F\rrthermore, if there is a directed path in the network from X to Z, then X is an ancestor of Z, whlle Z is a descendant of X.

For example, in the diagram shown in Figure 5.12(b), A is a descendant of D and D is an ancestor of B. Both B and D arc also non-descendants of A.

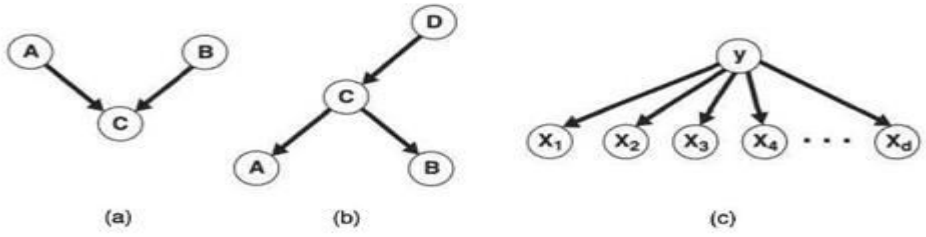


Figure 5.12. Representing probabilistic relationships using directed acyclic graphs.

In the diagram shown in Figure 5.12(b), A is conditionally independent of both B and D given C because the nodes for B and D are non-descendants of node A.

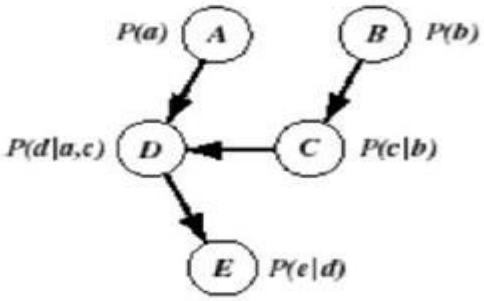
The conditional independence assumption made by a naive Bayes classifier can also be represented using a Bayesian network, as shown in Figure 5.12(c), where gr is the target class and {Xt,Xz,...,Xa} is the attribute set.

Besides the conditional independence conditions imposed by the network topology, each node is also associated with a probability table.

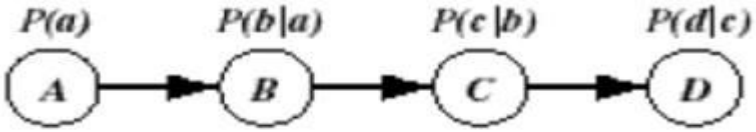
1. If a node X does not have any parents, then the table contains only the prior probability P(X).

2. If a node X has only one parent, Y, then the table contains the conditional probability P(XIY).

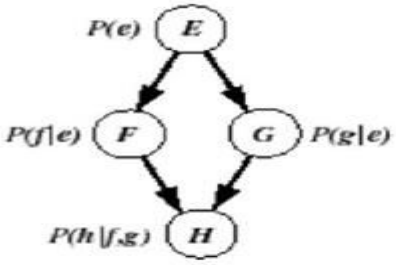
3. If a node X has multiple parents, {Y1,,Y2, . . . ,Yn}, then the table contains the conditional probability P(XlY1,Y2,. . ., Yn.).



$$P(a, b, c, d, e) = P(a)P(b)P(c|b)P(d|a,c)P(e|d)$$



$$P(a, b, c, d) = P(a)P(b|a)P(c|b)P(d|c)$$



$$P(e, f, g, h) = P(e)P(f|e)P(g|e)P(h|f, g)$$

Characteristics of BBN

Following are some of the general characteristics of the BBN method:

- ✓ BBN provides an approach for capturing the prior knowledge of a particular domain using a graphical model. The network can also be used to encode causal dependencies among variables.

- ✓ Constructing the network can be time consuming and requires a large amount of effort. However, once the structure of the network has been determined, adding a new variable is quite straightforward.

- ✓ Bayesian networks are well suited to dealing with incomplete data. Instances with missing attributes can be handled by summing or integrating the probabilities over all possible values of the attribute.

- ✓ Because the data is combined probabilistically with prior knowledge, the method is quite robust to model over fitting.

Cluster Analysis - MODULE 5

Cluster analysis divides data into groups (clusters) that are meaningful, useful, or both. If meaningful groups are the goal, then the clusters should capture the natural structure of the data.

Clustering for Understanding Classes, or conceptually meaningful groups of objects that share common characteristics, play an important role in how people analyze and describe the world. Indeed, human beings are skilled at dividing objects into groups (clustering) and assigning particular objects to these groups (classification). For example, even relatively young children can quickly label the objects in a photograph. In the context of understanding data, clusters are potential classes and cluster analysis is the study of tech- niques for automatically finding classes.

The following are some examples:

Biology. Biologists have spent many years creating a taxonomy (hi- erarchical classification) of all living things: kingdom, phylum, class, order, family, genus, and species. biologists have applied clustering to analyze the large amounts of genetic information that are now available. For example, clustering has been used to find groups of genes that have similar functions.

Information Retrieval. The World Wide Web consists of billions of web pages, and the results of a query to a search engine can return thousands of pages. Clustering can be used to group these search results into a small number of clusters, each of which captures a particular aspect of the query.

- **Climate.** Understanding the Earth's climate requires finding patterns in the atmosphere and ocean. To that end, cluster analysis has been applied to find patterns in atmospheric pressure and ocean temperature that have a significant impact on climate.
- **Psychology and Medicine.** An illness or condition frequently has a number of variations, and cluster analysis can be used to identify these different subcategories. For example, clustering has been used to identify different types of depression. Cluster analysis can also be used to detect patterns in the spatial or temporal distribution of a disease.
- **Business.** Businesses collect large amounts of information about current and potential customers. Clustering can be used to segment customers into a small number of groups for additional analysis and marketing activities.

Overview

Before discussing specific clustering techniques, we provide some necessary background.

Then we explore two important topics: (1) different ways to group a set of objects into a set of clusters, and (2) types of clusters.

What Is Cluster Analysis?

Cluster analysis groups data objects based on information found only in the data that describes the objects and their relationships. The goal is that the objects within a group be similar (or related) to one another and different from (or unrelated to) the objects in other groups. The greater the similarity (or homogeneity) within a group and the greater the difference between groups, the better or more distinct the clustering.

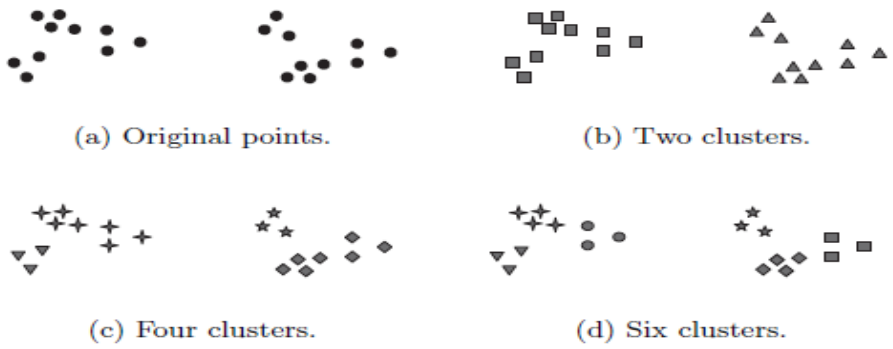


Figure 7.1. Three different ways of clustering the same set of points.

Figure 7.1, which shows 20 points and three different ways of dividing them into clusters. The shapes of the markers indicate cluster membership. Figures 7.1(b) and 7.1(d) divide the data into two and six parts, respectively. However, the apparent division of each of the two larger clusters into three sub clusters may simply be an artifact of the human visual system. Also, it may not be unreasonable to say that the points form four clusters, as shown in Figure 7.1(c). This figure illustrates that the definition of a cluster is imprecise and that the best definition depends on the nature of data and the desired results. Cluster analysis is related to other techniques that are used to divide data objects into groups. For instance, clustering can be regarded as a form of classification in that it creates a labeling of objects with class (cluster) labels

Different Types of Clustering's

An entire collection of clusters is commonly referred to as a **clustering**, and in this section, we distinguish various types of clusterings:

Hierarchical versus Partitional The most commonly discussed distinc- tion among different types of clusterings is whether the set of clusters is nested or unnested, or in more traditional terminology, hierarchical or partitional. A **partitional clustering** is simply a division of the set of data objects into non-overlapping subsets (clusters) such that each data object is in exactly one subset. Taken individually, each collection of clusters in Figures 7.1 (b–d) is a partitional clustering.

If we permit clusters to have subclusters, then we obtain a **hierarchical clustering**, which is a set of nested clusters that are organized as a tree. Each node (cluster) in the tree (except for the leaf nodes) is the union of its children (subclusters), and the root of the tree is the cluster containing all the objects. Often, but not always, the leaves of the tree are singleton clusters of individual data objects. If we allow clusters to be nested, then one interpretation of Figure 7.1(a) is that it has two subclusters (Figure 7.1(b)), each of which, in turn, has three subclusters (Figure 7.1(d)). The clusters shown in Figures 7.1 (a–d), when taken in that order, also form a hierarchical (nested) clustering with, respectively, 1, 2, 4, and 6 clusters on each level. Finally, note that a hierarchical clustering can be viewed as a sequence of partitional clusterings and a partitional clustering can be obtained by taking any member of that sequence; i.e., by cutting the hierarchical tree at a particular level.

Exclusive versus Overlapping versus Fuzzy The clusterings shown in Figure 7.1 are all **exclusive**, as they assign each object to a single cluster. There are many situations in which a point could reasonably be placed in more than one cluster, and these situations are better addressed by non-exclusive clustering. In the most general sense, an **overlapping** or **non-exclusive clustering** is used to reflect the fact that an object can *simultaneously* belong to more than one group (class). For instance, a person at a university can be both an enrolled student and an employee of the university. A non-exclusive clustering is also often used when, for example, an object is "between" two or more clusters and could reasonably be assigned to any of these clusters. Imagine a point halfway between two of the clusters of Figure 7.1. Rather than make a somewhat arbitrary assignment of the object to a single cluster, it is placed in all of the "equally good" clusters.

In a **fuzzy clustering** (Section 8.2.1), every object belongs to every cluster with a membership weight that is between 0 (absolutely doesn't belong) and 1 (absolutely belongs). In other words, clusters are treated as fuzzy sets. (Mathematically, a fuzzy set is one in which an object belongs to every set with a weight that is between 0 and 1. In fuzzy clustering, we often impose the additional constraint that the sum of the weights for each object must equal 1.) Similarly, probabilistic clustering techniques (Section 8.2.2) compute the probability with which each point belongs to each cluster, and these probabilities must also sum to 1. Because the membership weights or probabilities for any object sum to 1, a fuzzy or probabilistic clustering does

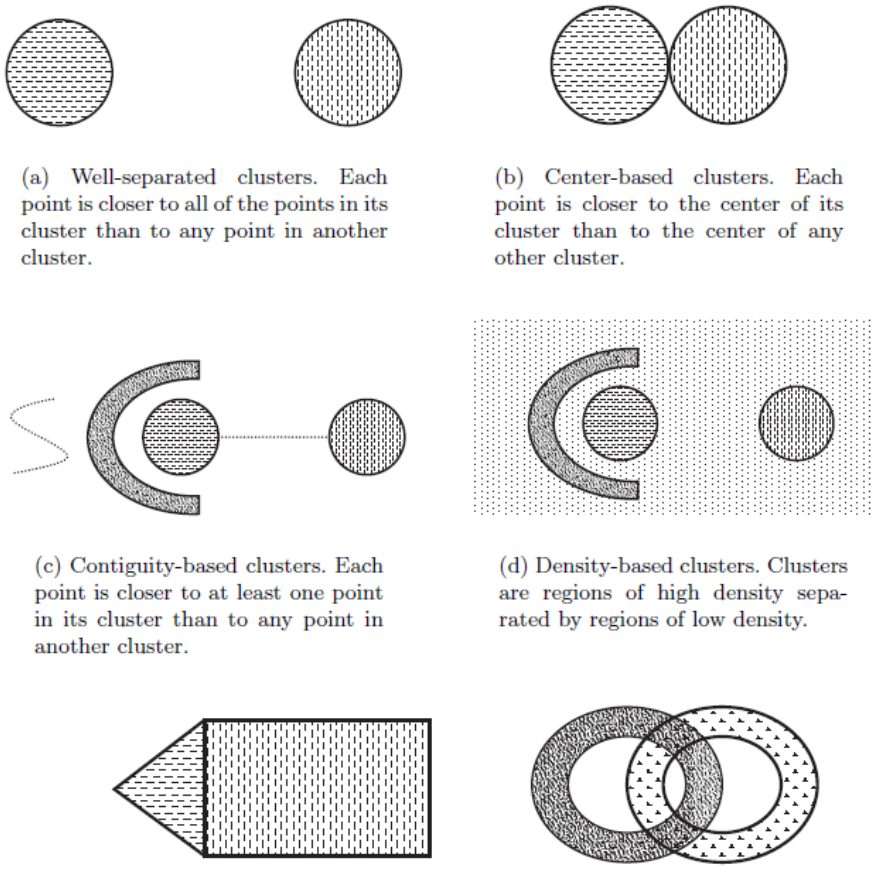
not address true multiclass situations, such as the case of a student employee, where an object belongs to multiple classes. Instead, these approaches are most appropriate for avoiding the arbitrariness of assigning an object to only one cluster when it is close to several. In practice, a fuzzy or probabilistic clustering is often converted to an exclusive clustering by assigning each object to the cluster in which its membership weight or probability is highest.

Complete versus Partial A **complete clustering** assigns every object to a cluster, whereas a **partial clustering** does not. The motivation for a partial clustering is that some objects in a data set may not belong to well- defined groups. Many times objects in the data set represent noise, outliers, or "uninteresting background." For example, some newspaper stories share a common theme, such as global warming, while other stories are more generic or one-of-a-kind. Thus, to find the important topics in last month's stories, we often want to search only for clusters of documents that are tightly related by a common theme. In other cases, a complete clustering of the objects is desired. For example, an application that uses clustering to organize documents for browsing needs to guarantee that all documents can be browsed.

7.1.3 Different Types of Clusters

Clustering aims to find useful groups of objects (clusters), where usefulness is defined by the goals of the data analysis. Not surprisingly, several different notions of a cluster prove useful in practice. In order to visually illustrate the differences among these types of clusters, we use two-dimensional points, as shown in Figure 7.2, as our data objects..

Well-Separated A cluster is a set of objects in which each object is closer (or more similar) to every other object in the cluster than to any object not in the cluster. Sometimes a threshold is used to specify that all the objects in a cluster must be sufficiently close (or similar) to one another. This idealistic definition of a cluster is satisfied only when the data contains natural clusters that are quite far from each other. Figure 7.2(a) gives an example of well- separated clusters that consists of two groups of points in a two-dimensional space. The distance between any two points in different groups is larger than the distance between any two points within a group. Well-separated clusters do not need to be globular, but can have any shape.



(a) Well-separated clusters. Each point is closer to all of the points in its cluster than to any point in another cluster.

(b) Center-based clusters. Each point is closer to the center of its cluster than to the center of any other cluster.

(c) Contiguity-based clusters. Each point is closer to at least one point in its cluster than to any point in another cluster.

(d) Density-based clusters. Clusters are regions of high density separated by regions of low density.

(e) Conceptual clusters. Points in a cluster share some general property that derives from the entire set of points. (Points in the intersection of the circles belong to both.)

Figure 7.2. Different types of clusters as illustrated by sets of two-dimensional points.

Prototype-Based A cluster is a set of objects in which each object is closer (more similar) to the prototype that defines the cluster than to the prototype of any other cluster. For data with continuous attributes, the prototype of a cluster is often a centroid, i.e., the average (mean) of all the points in the clus- ter. When a centroid is not meaningful, such as when the data has categorical attributes, the prototype is often a medoid, i.e., the most representative point of a cluster. For many types of data, the prototype can be regarded as the most central point, and in such instances, we commonly refer to prototype- based clusters as **center-based clusters**. Not surprisingly, such clusters tend to be globular. Figure 7.2(b) shows an example of center-based clusters.

Graph-Based If the data is represented as a graph, where the nodes are objects and the links represent connections among objects (see Section 2.1.2), then a cluster can be defined as a **connected component**; i.e., a group of objects that are connected to one another, but that have no connection to objects outside the

group. An important example of graph-based clusters is a **contiguity-based cluster**, where two objects are connected only if they are within a specified distance of each other. This implies that each object in a contiguity-based cluster is closer to some other object in the cluster than to

any point in a different cluster. Figure 7.2(c) shows an example of such clusters for two-dimensional points. This definition of a cluster is useful when clusters are irregular or intertwined. However, this approach can have trouble when noise is present since, as illustrated by the two spherical clusters of Figure 7.2(c), a small bridge of points can merge two distinct clusters.

Other types of graph-based clusters are also possible. One such approach (Section 7.3.2) defines a cluster as a **clique**; i.e., a set of nodes in a graph that are completely connected to each other. Specifically, if we add connections between objects in the order of their distance from one another, a cluster is formed when a set of objects forms a clique. Like prototype-based clusters, such clusters tend to be globular.

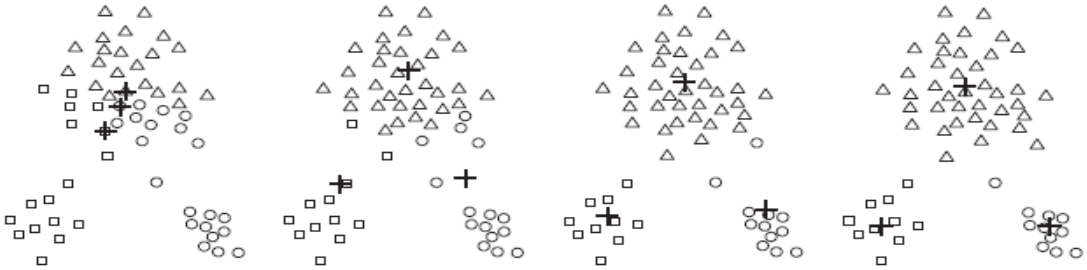
Density-Based A cluster is a dense region of objects that is surrounded by a region of low density. Figure 7.2(d) shows some density-based clusters for data created by adding noise to the data of Figure 7.2(c). The two circular clusters are not merged, as in Figure 7.2(c), because the bridge between them fades into the noise. Likewise, the curve that is present in Figure 7.2(c) also fades into the noise and does not form a cluster in Figure 7.2(d). A density- based definition of a cluster is often employed when the clusters are irregular or intertwined, and when noise and outliers are present. By contrast, a contiguity- based definition of a cluster would not work well for the data of Figure 7.2(d) because the noise would tend to form bridges between clusters.

Shared-Property (Conceptual Clusters) More generally, we can define a cluster as a set of objects that share some property. This definition encompasses all the previous definitions of a cluster; e.g., objects in a center-based cluster share the property that they are all closest to the same centroid or medoid. However, the shared-property approach also includes new types of clusters. Consider the clusters shown in Figure 7.2(e). A triangular area (cluster) is adjacent to a rectangular one, and there are two intertwined circles (clusters).

K-means

Prototype-based clustering techniques create a one-level partitioning of the data objects. There are a number of such techniques, but two of the most prominent are K-means and K-medoid. K-means defines a prototype in terms of a centroid, which is usually the mean of a group of points, and is typically applied to objects in a continuous n-dimensional space

The Basic K-means Algorithm



(a) Iteration 1. (b) Iteration 2. (c) Iteration 3. (d) Iteration 4.

Figure 7.3. Using the K-means algorithm to find three clusters in sample data.

. K-means is formally described by Algorithm 7.1. The operation of K-means is illustrated in Figure 7.3, which shows how, starting from three centroids, the final clusters are found in four assignment-update steps. In these and other figures displaying K-means clustering, each subfigure shows (1) the centroids at the start of the iteration and (2) the assignment of the points to those centroids. The centroids are indicated by the "+" symbol; all points belonging to the same cluster have the same marker shape.

Algorithm 7.1 Basic K-means algorithm.

1: Select K points as initial centroids.
2: **repeat**
3: Form K clusters by assigning each point to its closest centroid.
4: Recompute the centroid of each cluster.
5: **until** Centroids do not change.

In the first step, shown in Figure 7.3(a), points are assigned to the initial centroids, which are all in the larger group of points. For this example, we use the mean as the centroid. After points are assigned to a centroid, the centroid is then updated. Again, the figure for each step shows the centroid at the beginning of the step and the assignment of points to those centroids. In the second step, points are assigned to the updated centroids, and the centroids are updated again. In steps 2, 3, and 4, which are shown in Figures 7.3 (b), (c), and (d), respectively, two of the centroids move to the two small groups of points at the bottom of the figures. When the K-means algorithm terminates in Figure 7.3(d), because no more changes occur, the centroids have identified the natural groupings of points.

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.

$$\text{SSE} = \sum_{i=1}^{K} \sum_{\mathbf{x} \in C_i} dist(\mathbf{c}_i, \mathbf{x})^2$$

x is a data point in cluster Ci and mi is the representative point for cluster Ci can show that mi 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.

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

If clusters are the same size, n, then

$$P = \frac{\text{number of ways to select one centroid from each cluster}}{\text{number of ways to select } K \text{ centroids}} = \frac{K! n^K}{(Kn)^K} = \frac{K!}{K^K}$$

K-means: Additional Issues

Handling Empty Clusters

Basic K-means algorithm can yield empty clusters

Several strategies to address this..

- Choose the point that contributes most to SSE
- Choose a point from the cluster with the highest SSE
- If there are several empty clusters, the above can be repeated several times

Updating Centers Incrementally

In the basic K-means algorithm, centroids are updated after all points are assigned to a centroid

An alternative is to update the centroids after each assignment (incremental approach)

- Each assignment updates zero or two centroids
- More expensive
- Introduces an order dependency
- Never get an empty cluster
- Can use "weights" to change the impact

Pre-processing and Post-processing

Pre-processing

- Normalize the data
- Eliminate outliers

Post-processing

- Eliminate small clusters that may represent outliers
- Split 'loose' clusters, i.e., clusters with relatively high SSE
- Merge clusters that are 'close' and that have relatively low SSE
- Can use these steps during the clustering process

Strengths and Weaknesses of K-means (Limitations)

K-means is simple and can be used for a wide variety of data types.

- It is also quite efficient, even though multiple runs are often performed.
- K-means is not suitable for all types of data.
- K-means has problems when clusters are of differing
  - Sizes
  - Densities
  - Non-globular shapes
- K-means has problems when the data contains outliers.

BisectingK-means

The bisecting K-means algorithm is a straightforward extension of the basic K-means algorithm that is based on a simple idea: to obtain K clusters, split the set of all points into two clusters, select one of these clusters to split, and so on, until K clusters have been produced.

Algorithm 7.3 Bisecting K-means algorithm.

```
1: Initialize the list of clusters to contain the cluster consisting of all points.
2: repeat
3:    Remove a cluster from the list of clusters.
4:    {Perform several "trial" bisections of the chosen cluster.}
5:    for i = 1 to number of trials do
6:       Bisect the selected cluster using basic K-means.
7:    end for
8:    Select the two clusters from the bisection with the lowest total SSE.
9:    Add these two clusters to the list of clusters.
10: until The list of clusters contains K clusters.
```

There are a number of different ways to choose which cluster to split. We can choose the largest cluster at each step, choose the one with the largest SSE, or use a criterion based on both size

and SSE. Different choices result in different clusters

Exercise 1. ***K-means clustering***

Use the k-means algorithm and Euclidean distance to cluster the following 8 examples into 3 clusters:
A1=(2,10), A2=(2,5), A3=(8,4), A4=(5,8), A5=(7,5), A6=(6,4), A7=(1,2), A8=(4,9).
The distance matrix based on the Euclidean distance is given below:

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

Suppose that the initial seeds (centers of each cluster) are A1, A4 and A7. Run the k-means algorithm for 1 epoch only. At the end of this epoch show:
a) The new clusters (i.e. the examples belonging to each cluster)
b) The centers of the new clusters
c) Draw a 10 by 10 space with all the 8 points and show the clusters after the first epoch and the new centroids.

.

Solution:

a)

d(a,b) denotes the Eucledian distance between a and b. It is obtained directly from the distance matrix or calculated as follows: $d(a,b)=sqrt((x_b-x_a)^2+(y_b-y_a)^2))$

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

epoch1 – start:

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

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

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

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

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

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

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

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

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

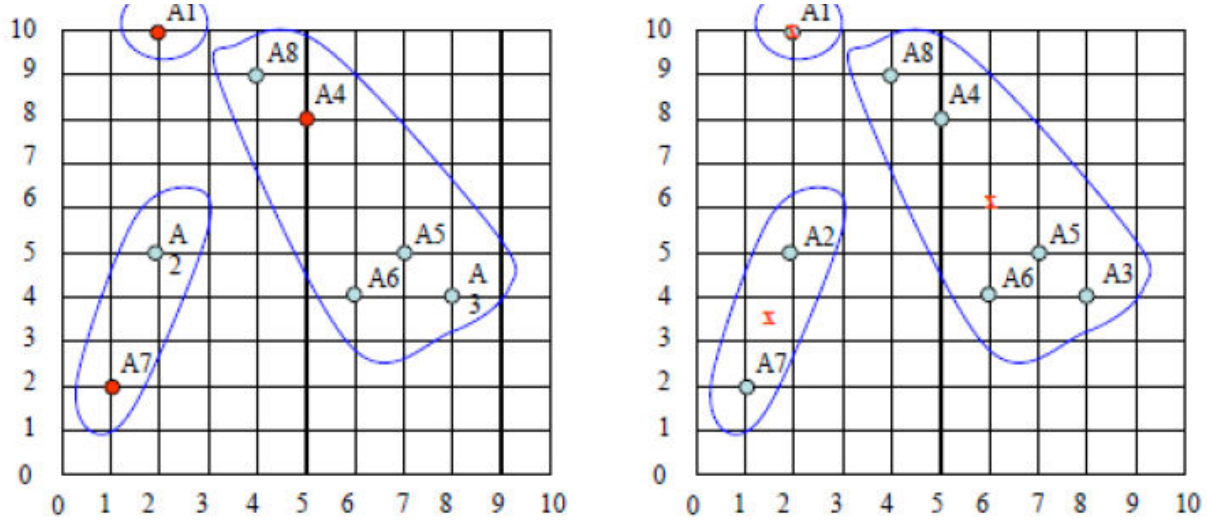
A8:
d(A8, seed1)= $\sqrt{5}$
d(A8, seed2)= $\sqrt{2}$ ← smaller
d(A8, seed3)= $\sqrt{58}$
→ A8 ∈ cluster2

end of epoch1

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

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

c)



Agglomerative Hierarchical Clustering

Hierarchical clustering techniques are a second important category of cluster- ing methods. As with K-means, these approaches are relatively old compared to many clustering algorithms, but they still enjoy widespread use. There are two basic approaches for generating a hierarchical clustering:

Agglomerative: Start with the points as individual clusters and, at each step, merge the closest pair of clusters. This requires defining a notion of cluster proximity.

Do not have to assume any particular number of clusters

– Any desired number of clusters can be obtained by 'cutting' the dendogram at the proper level

Basic algorithm is

Algorithm 8.3 Basic agglomerative hierarchical clustering algorithm.

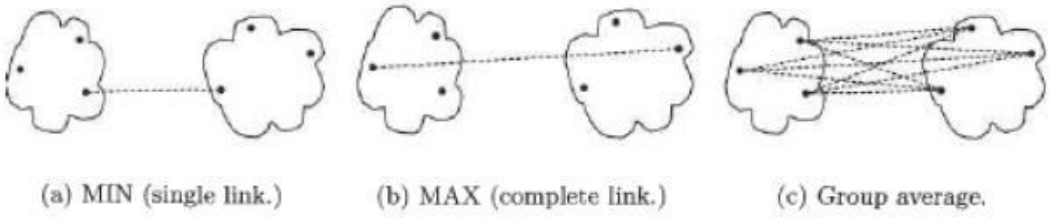
1: Compute the proximity matrix, if necessary.
2: **repeat**
3: Merge the closest two clusters.
4: Update the proximity matrix to reflect the proximity between the new cluster and the original clusters.
5: **until** Only one cluster remains.

How to Define Inter-Cluster Similarity (Proximity of two clusters):

MIN

MAX

Group Average



(a) MIN (single link.) (b) MAX (complete link.) (c) Group average.

Figure 8.14. Graph-based definitions of cluster proximity

1) Single Link or MIN

For the single link or MIN version of hierarchical clustering, the proximity of two clusters is defined as the minimum of the distance (maximum of the similarity) between any two points in the two different clusters.

we shall use sample data that consists of 6 two-dimensional points, which are shown in Figure 8.15. The r and g coordinates of the points and the Euclidean distances between them are shown in Tables 8.3 and 8.4. respectively.

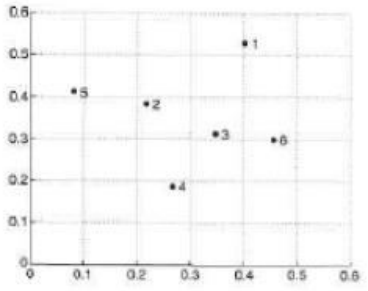


Figure 8.15. Set of 6 two-dimensional points.

Point	x Coordinate	y Coordinate
p1	0.40	0.53
p2	0.22	0.38
p3	0.35	0.32
p4	0.26	0.19
p5	0.08	0.41
p6	0.45	0.30

Table 8.3. xy coordinates of 6 points.

	p1	p2	p3	p4	p5	p6
p1	0.00	0.24	0.22	0.37	0.34	0.23
p2	0.24	0.00	0.15	0.20	0.14	0.25
p3	0.22	0.15	0.00	0.15	0.28	0.11
p4	0.37	0.20	0.15	0.00	0.29	0.22
p5	0.34	0.14	0.28	0.29	0.00	0.39
p6	0.23	0.25	0.11	0.22	0.39	0.00

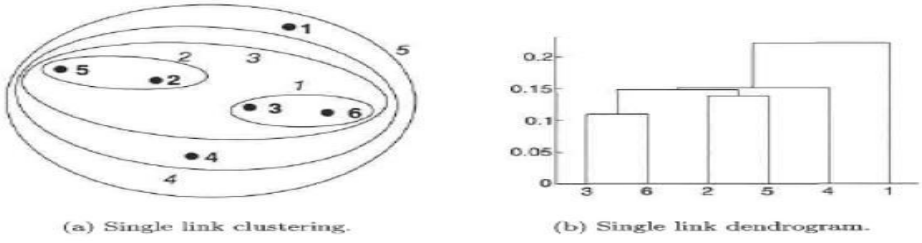
Table 8.4. Euclidean distance matrix for 6 points.

Figure 8.16 shows the result of applying the single link technique to our example data set of six points. Figure 8.16(a) shows the nested clusters as a sequence of nested ellipses, where the numbers associated with the ellipses indicate the order of the clustering. Figure S.16(b) shows the same information, but as a dendrogram.

The height at which two clusters are merged in the dendrogram reflects the distance of the two clusters.

For instance, from Table 8.4, we see that the distance between points 3 and 6 is 0.11, and that is the height at which they are joined into one cluster in the dendrogram. As another example, the distance between clusters {3,6} and {2,5} is given by

$$\begin{aligned} dist(\{3,6\},\{2,5\}) &= \min(dist(3,2), dist(6,2), dist(3,5), dist(6,5)) \\ &= \min(0.15, 0.25, 0.28, 0.39) \\ &= 0.15. \end{aligned}$$



(a) Single link clustering. (b) Single link dendrogram.

Figure 8.16. Single link clustering of the six points shown in Figure 8.15.

2) Complete Link or MAX or CLIQUE

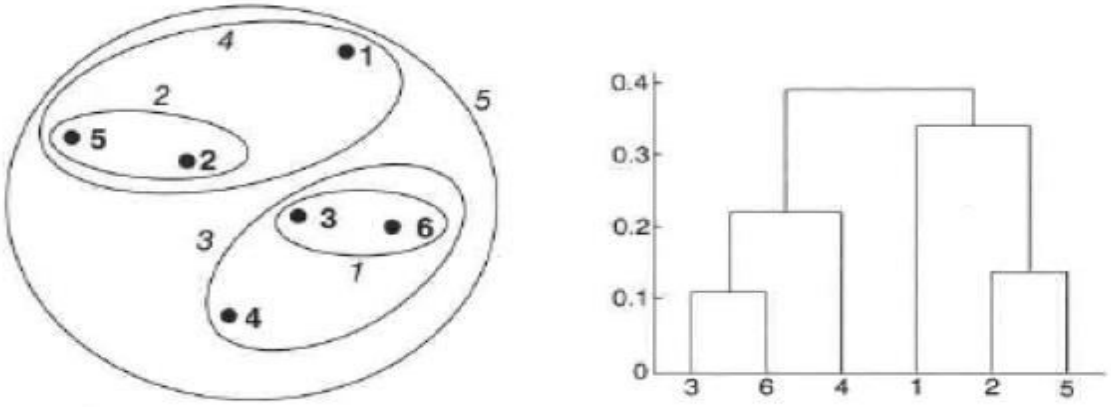
For the complete link or MAX version of hierarchical clustering, the proximity of two clusters is defined as the maximum of the distance (minimum of the similarity) between

any two points in the two different clusters. Using graph terminology, if you start with all points as singleton clusters and add links between points one at a time, shortest links first, then a group of points is not a cluster until all the points in it are completely linked, i.e., form a clique.

Example 8.5 (Complete Link). Figure 8.17 shows the results of applying MAX to the sample data set of six points. As with single link, points 3 and 6 are merged first. However, {3,6} is merged with {4}, instead of {2,5} or {1}.

are merged first. However, $\{3,6\}$ is merged with $\{4\}$, instead of $\{2,5\}$ or $\{1\}$ because

$$\begin{aligned} dist(\{3,6\},\{4\}) &= \max(dist(3,4), dist(6,4)) \\ &= \max(0.15, 0.22) \\ &= 0.22. \\ dist(\{3,6\},\{2,5\}) &= \max(dist(3,2), dist(6,2), dist(3,5), dist(6,5)) \\ &= \max(0.15, 0.25, 0.28, 0.39) \\ &= 0.39. \\ dist(\{3,6\},\{1\}) &= \max(dist(3,1), dist(6,1)) \\ &= \max(0.22, 0.23) \\ &= 0.23. \end{aligned}$$



(a) Complete link clustering. (b) Complete link dendrogram.

Figure 8.17. Complete link clustering of the six points shown in Figure 8.15.

3) Group Average

For the group average version of hierarchical clustering, the proximity of two clusters is defined as the average pairwise proximity among all pairs of points in the different clusters.

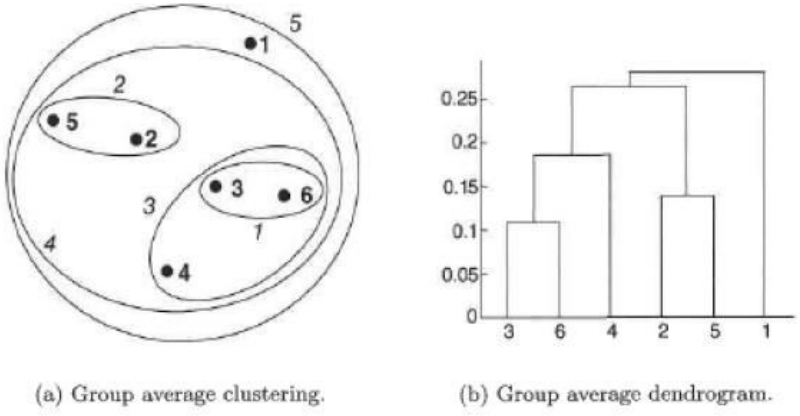
This is an intermediate approach between the single and complete link approaches.

Thus, for group average, the cluster proximity proximity(Ci,Cj) of clusters C,i and Cj', which are of size mi and mj, respectively, is expressed by the following equation

$$proximity(C_i, C_j) = \frac{\sum_{\substack{\mathbf{x} \in C_i \\ \mathbf{y} \in C_j}} proximity(\mathbf{x}, \mathbf{y})}{m_i * m_j}.$$

Figure 8.18 shows the results of applying the group average approach to the sample data set of six points. To illustrate how group average works, we calculate the distance between some clusters.

$$\begin{aligned} dist(\{3,6,4\},\{1\}) &= (0.22 + 0.37 + 0.23)/(3 * 1) \\ &= 0.28 \\ dist(\{2,5\},\{1\}) &= (0.2357 + 0.3421)/(2 * 1) \\ &= 0.2889 \\ dist(\{3,6,4\},\{2,5\}) &= (0.15 + 0.28 + 0.25 + 0.39 + 0.20 + 0.29)/(6 * 2) \\ &= 0.26 \end{aligned}$$



(a) Group average clustering. (b) Group average dendrogram.

Figure 8.18. Group average clustering of the six points shown in Figure 8.15.

Key Issues in Agglomerative Hierarchical Clustering (Strengths and Weaknesses)

- 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

The DBSCAN Algorithm

DBSCAN is a density-based algorithm.

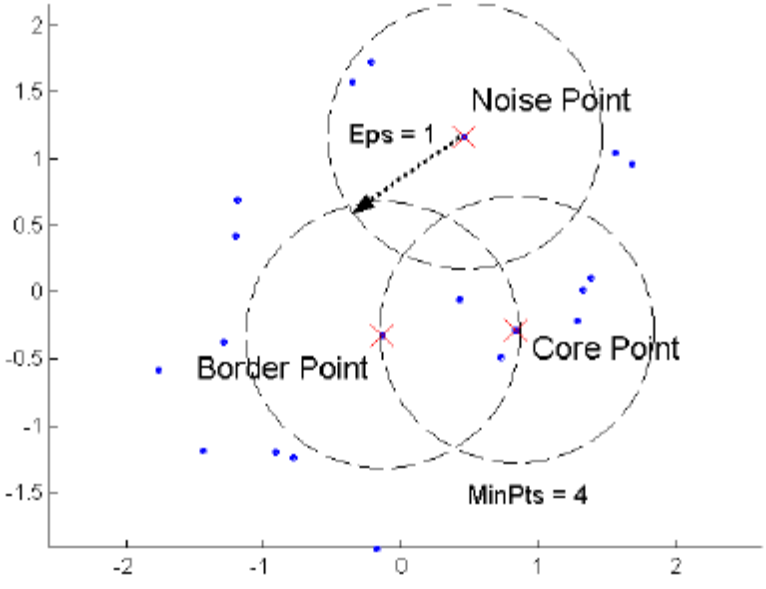
- Density = number of points within a specified radius (Eps)

point is a *core point* if it has more than a specified number of points (MinPts) within Eps

- These are points that are at the interior of a cluster
- A *border point* has fewer than MinPts within Eps, but is in the neighborhood of a core point

- A *noise point* is any point that is not a core point or a border point.

- Eliminate noise points
- Perform clustering on the remaining points



```
current_cluster_label ← 1
for all core points do
  if the core point has no cluster label then
    current_cluster_label ← current_cluster_label + 1
    Label the current core point with cluster label current_cluster_label
  end if
  for all points in the Eps-neighborhood, except i^th the point itself do
    if the point does not have a cluster label then
      Label the point with cluster label current_cluster_label
    end if
  end for
end for
```

Strengths and weaknesses of DBSCAN

- It is relatively Resistant to Noise.
- It can Can handle clusters of different shapes and sizes
- Does NOT Work Well when the clusters having Varying densities
- Does NOT Work Well With High-dimensional data.

Density-Based Clustering

- Grid-Based Clustering
- Subspace Clustering
- CLIQUE
- DENCLUE: A Kernel-Based Scheme for Density-Based
- Clustering

Grid-Based Clustering

The idea is to split the possible values of each attribute into a number of contiguous intervals, creating a set of grid cells.

Objects can be assigned to grid cells in one pass through the data, and information about each cell, such as the number of points in the cell, can also be gathered at the same time.

Defining Grid Cells: This is a key step in the process, but also the least well defined, as there are many ways to split the possible values of each attribute into a number of contiguous intervals.

For continuous attributes, one common approach is to split the values into equal width intervals. If this approach is applied to each attribute, then the resulting grid cells all have the same olume, and the density of a cell is conveniently defined as the number of points in the cell.

The Density of Grid Cells: A natural way to define the density of a grid cell (or a more generally shaped region) is as the number of points divided by the volume of the region. In other words, density is the number of points per amount of space, regardless of the dimensionality of that space

Example: Figure 9.10 shows two sets of two dimensional points divided into 49 cells using a 7-

by-7 grid. The first set contains 200 points generated from a uniform distribution over a circle centered at (2, 3) of radius 2, while the second set has 100 points generated from a uniform distribution over a circle centered at (6, 3) of radius 1. The counts for the grid cells are shown in Table 9.2.

Algorithm 9.4 Basic grid-based clustering algorithm.

1: Define a set of grid cells.
2: Assign objects to the appropriate cells and compute the density of each cell.
3: Eliminate cells having a density below a specified threshold, τ.
4: Form clusters from contiguous (adjacent) groups of dense cells.

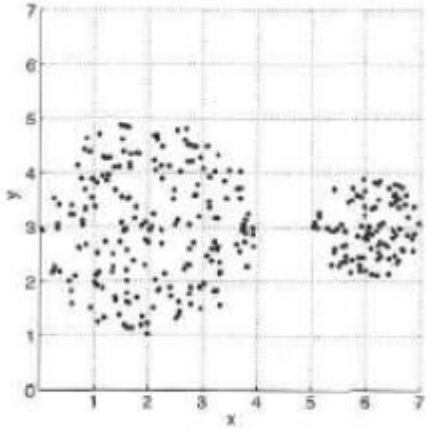


Figure 9.10. Grid-based density.

0	0	0	0	0	0	0
0	0	0	0	0	0	0
4	17	18	6	0	0	0
14	14	13	13	0	18	27
11	18	10	21	0	24	31
3	20	14	4	0	0	0
0	0	0	0	0	0	0

Table 9.2. Point counts for grid cells.

CLIQUE

CLIQUE (Clustering In QUEst) is a grid-based clustering algorithm that methodically finds subspace clusters. It is impractical to check each subspace for clusters since the number of such subspaces is exponential in the number of dimensions. Instead, CLIQUE relies on the following property;

Monotonicity property of density-based clusters If a set of points forms a density-

based cluster in k dimensions (attributes), then the same set of points is also part of a density-based cluster in all possible subsets of those dimensions.

Algorithm 9.5 CLIQUE.

1: Find all the dense areas in the one-dimensional spaces corresponding to each attribute. This is the set of dense one-dimensional cells.
2: $k \leftarrow 2$
3: **repeat**
4: Generate all candidate dense k-dimensional cells from dense $(k-1)$-dimensional cells.
5: Eliminate cells that have fewer than ξ points.
6: $k \leftarrow k+1$
7: **until** There are no candidate dense k-dimensional cells.
8: Find clusters by taking the union of all adjacent, high-density cells.
9: Summarize each cluster using a small set of inequalities that describe the attribute ranges of the cells in the cluster.

Graph-Based Clustering

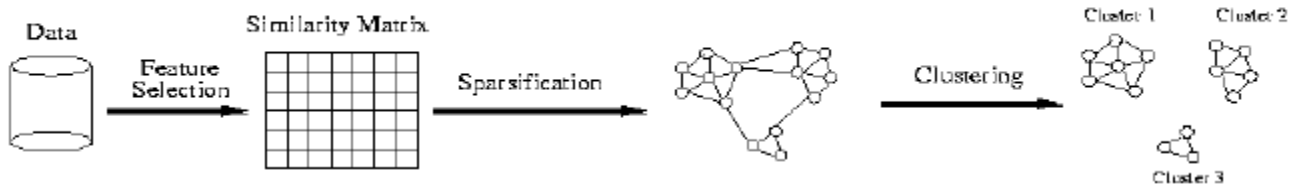
Graph-Based clustering uses the proximity graph

- Start with the proximity matrix
- Consider each point as a node in a graph
- Each edge between two nodes has a weight which is the proximity between the two points
- Initially the proximity graph is fully connected
- MIN (single-link) and MAX (complete-link) can be viewed as starting with this graph
- Sparsification can eliminate more than 99% of the entries in a proximity matrix
- The amount of time required to cluster the data is drastically reduced
- The size of the problems that can be handled is increased.
- Sparsification techniques keep the connections to the most similar (nearest) neighbors of a point while breaking the connections to less similar points.
- The nearest neighbors of a point tend to belong to the same class as the point itself.

- This reduces the impact of noise and outliers and sharpens the distinction between clusters.

Sparsification facilitates the use of graph partitioning algorithms (or algorithms based on graph partitioning algorithms.

– Chameleon and Hypergraph-based Clustering



Limitations of Current Merging Schemes

Existing merging schemes in hierarchical clustering algorithms are static in nature

– MIN or CURE:

merge two clusters based on their *closeness* (or minimum distance)

– GROUP-AVERAGE:

merge two clusters based on their average

connectivity Chameleon: Clustering Using Dynamic Modeling

Adapt to the characteristics of the data set to find the natural clusters Use a dynamic model to measure the similarity between clusters

- Main property is the relative closeness and relative inter-connectivity of the cluster
- Two clusters are combined if the resulting cluster shares certain *properties* with the constituent clusters
- The merging scheme preserves *self-similarity*

Steps

Preprocessing Step:

Represent the Data by a Graph

- Given a set of points, construct the k-nearest-neighbor (k-NN) graph to capture the relationship between a point and its k nearest neighbors

- Concept of neighborhood is captured dynamically (even if region is sparse)

Phase 1: Use a multilevel graph partitioning algorithm on the graph to find a large number of clusters of well-connected vertices

- Each cluster should contain mostly points from one "true" cluster, i.e., is a sub-cluster of a "real" cluster.

Two clusters are combined if the *resulting cluster shares certain properties with the constituent clusters*

- Two key properties used to model cluster similarity:

Relative Interconnectivity: Absolute interconnectivity of two clusters normalized by the internal connectivity of the clusters

Relative Closeness: Absolute closeness of two clusters normalized by the internal closeness of the clusters

SNN Clustering Algorithm:

1)Compute the similarity matrix

This corresponds to a similarity graph with data points for nodes and edges whose weights are the similarities between data points

2)Sparsify the similarity matrix by keeping only the *k* most similar neighbors

This corresponds to only keeping the *k* strongest links of the similarity graph

3)Construct the shared nearest neighbor graph from the sparsified similarity matrix.

At this point, we could apply a similarity threshold and find the connected components to obtain the clusters (Jarvis-Patrick algorithm)

4)Find the SNN density of each Point.

Using a user specified parameters, *Eps*, find the number points that have an SNN imilarity of *Eps* or greater to each point. This is the SNN density of the point.

5)Find the core points

Using a user specified parameter, *MinPts*, find the core points, i.e., all points that have an SNN density greater than *MinPts.*

6)Form clusters from the core points.

If two core points are within a radius, *Eps*, of each other they are place in the same cluster

7)Discard all noise points.

All non-core points that are not within a radius of *Eps* of a core point are discarded .

8)Assign all non-noise, non-core points to clusters.

This can be done by assigning such points to the nearest core point

Algorithm 9.10 Computing shared nearest neighbor similarity

1: Find the k-nearest neighbors of all points.
2: **if** two points, $\mathbf{x}$ and $\mathbf{y}$ are *not* among the k-nearest neighbors of each other **then**
3: $similarity(\mathbf{x}, \mathbf{y}) \leftarrow 0$
4: **else**
5: $similarity(\mathbf{x}, \mathbf{y}) \leftarrow$ number of shared neighbors
6: **end if**

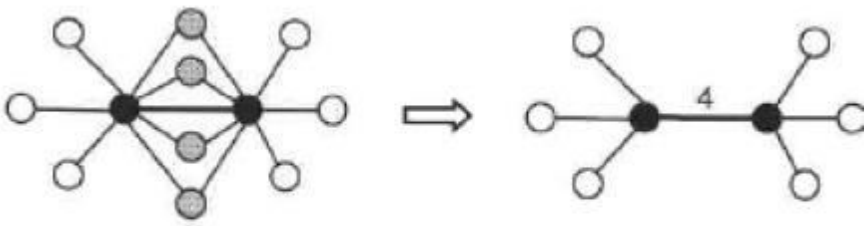


Figure 9.23. Computation of SNN similarity between two points.

Limitations of SNN Clustering

Complexity of SNN Clustering is high

- O(n * time to find numbers of neighbor within *Eps*)

– In worst case, this is $O(n^2)$

Scalable Clustering Algorithms

BIRCH

BIRCH (Balanced Iterative Reducing and Clustering using Hierarchies) is a highly efficient clustering technique for data in Euclidean vector spaces,

i'e., data for which averages make sense. BIRCH can efficiently cluster such data with one pass and can improve that clustering with additional passes. BIRCH can also deal effectively with outliers.

Algorithm 9.13 BIRCH.

1: **Load the data into memory by creating a CF tree that summarizes the data.**
2: **Build a smaller CF tree if it is necessary for phase 3.** T is increased, and then the leaf node entries (clusters) are reinserted. Since T has increased, some clusters will be merged.
3: **Perform global clustering.** Different forms of global clustering (clustering that uses the pairwise distances between all the clusters) can be used. However, an agglomerative, hierarchical technique was selected. Because the clustering features store summary information that is important to certain kinds of clustering, the global clustering algorithm can be applied as if it were being applied to all the points in a cluster represented by the CF.
4: **Redistribute the data points using the centroids of clusters discovered in step 3, and thus, discover a new set of clusters.** This overcomes certain problems that can occur in the first phase of BIRCH. Because of page size constraints and the T parameter, points that should be in one cluster are sometimes split, and points that should be in different clusters are sometimes combined. Also, if the data set contains duplicate points, these points can sometimes be clustered differently, depending on the order in which they are encountered. By repeating this phase multiple times, the process converges to a locally optimum solution.

CURE

CURE (Clustering Using REpresentatives) is a clustering algorithm that uses a variety of different techniques to create an approach that can handle large data sets, outliers, and clusters with non-

spherical shapes and non-uniform sizes. CURE represents a cluster by using multiple representative points from the cluster. These points will, in theory, capture the geometry and shape of the cluster. The first representative point is chosen to be the point farthest from

the center of the cluster, while the remaining points are chosen so that they are farthest from all the previously chosen points. In this way, the representative points are naturally relatively well distributed.

Algorithm 9.14 CURE.

1: **Draw a random sample from the data set.** The CURE paper is notable for explicitly deriving a formula for what the size of this sample should be in order to guarantee, with high probability, that all clusters are represented by a minimum number of points.
2: **Partition the sample into p equal-sized partitions.**
3: **Cluster the points in each partition into $\frac{m}{pq}$ clusters using CURE's hierarchical clustering algorithm to obtain a total of $\frac{m}{q}$ clusters.** Note that some outlier elimination occurs during this process.
4: **Use CURE's hierarchical clustering algorithm to cluster the $\frac{m}{q}$ clusters found in the previous step until only K clusters remain.**
5: **Eliminate outliers.** This is the second phase of outlier elimination.
6: **Assign all remaining data points to the nearest cluster to obtain a complete clustering.**

Question Bank: Clustering Analysis

1. Explain desired features of cluster analysis.
2. Explain how distance between a pair of points can be computed.
3. Write a short note on density-based methods.
4. Write and explain basic K-Means algorithm.
5. Explain DBSCAN clustering algorithm.
6. What are the limitations of K Means algorithm.
7. Explain cluster analysis methods briefly.
8. Explain agglomerative hierarchical clustering.

9. Explain bisecting K Means algorithm.
10. Distinguish between various types of clustering.
11. What are unsupervised, supervised and relative evaluation measures that are applied to judge various aspects of cluster validity.
12. Explain different types of defining proximity between clusters.
13. Differentiate between exclusive and overlapping clustering.
14. What are the various issues considered for cluster validation? Explain different evaluation measures used for cluster validity.
15. Explain unsupervised cluster evaluation using cohesion and separation.

16. Explain unsupervised cluster evaluation using proximity matrix.
17. List and explain classification-oriented measures of cluster validity.
18. Explain similarity – oriented measures of cluster validity.
19. Explain grid-based clustering algorithm.
20. Explain subspace clustering.
21. Write and explain CLIQUE algorithm.
22. Write and explain DENCLUE algorithm.
23. Explain different graph-based clustering.