# Introduction to Data Mining

## 1.1 What Motivated Data Mining? Why Is It Important?

*Necessity is the mother of invention. —Plato*

- Data mining has attracted a great deal of attention in the information industry and in society as a whole in recent years, due to the wide availability of huge amounts of data and the imminent need for turning such data into useful information and knowledge.
- The information and knowledge gained can be used for applications ranging from market analysis, fraud detection, and customer retention, to production control and science exploration. ***Data mining can be viewed as a result of the natural evolution of information technology.***
- ***The database system industry has witnessed an evolutionary path in the development of the following functionalities (Figure 1.1): data collection and database creation, data management (including data storage and retrieval, and database transaction processing), and advanced data analysis (involving data warehousing and data mining).***
- Since the 1960s, database and information technology has been evolving systematically from primitive file processing systems to sophisticated and powerful database systems.
- The research and development in database systems since the 1970s has progressed from early hierarchical and network database systems to the development of relational database systems (where data are stored in relational table structures; see Section 1.3.1), data modeling tools, and indexing and accessing methods. In addition, users gained convenient and flexible data access through query languages, user inter faces, optimized query processing, and transaction management.

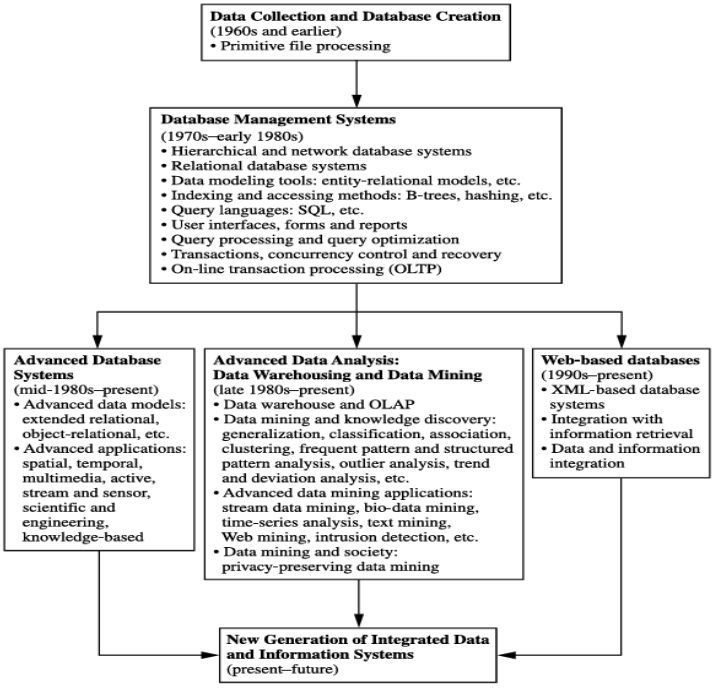



**Figure 1.1** The evolution of database system technology.

**Refer classwork example (Hospital Scenario from Flat files – storage of data to today's Robotic World)**

- Efficient methods for on-line transaction processing (OLTP), where a query is viewed as a read-only transaction, have contributed substantially to the evolution and wide acceptance of relational technology as a major tool for efficient storage, retrieval, and management of large amounts of data.
- ***Database technology since the mid-1980s has been characterized by the popular adoption of relational technology and an upsurge of research and development activities on new and powerful database systems. These promote the development of advanced data models such as extended-relational, object-oriented, object-relational, and deductive models.***

- ***Application-oriented database systems, including spatial, temporal, multimedia, active, stream, and sensor, and scientific and engineering databases, knowledge bases, and office information bases***, have flourished.
- Heterogeneous database systems and Internet-based global information systems such as the World Wide Web (WWW) have also emerged and play a vital role in the information industry.
- Data can now be stored in many different kinds of databases and information repositories.
  - One data repository architecture that has emerged is the data warehouse a repository of multiple heterogeneous data sources organized under a unified schema at a single site in order to facilitate management decision making.
- ***Data warehouse technology includes data cleaning, data integration, and on-line analytical processing (OLAP), that is, analysis techniques with functionalities such as summarization, consolidation, and aggregation as well as the ability to view information from different angles.*** Although OLAP tools support multidimensional analysis and decision making, additional data analysis tools are required for in-depth analysis, such as


**Figure 1.2** We are data rich, but information poor.

data classification, clustering, and the characterization of data changes over time.

- In addition, huge volumes of data can be accumulated beyond databases and data ware houses.
- Typical examples include the World Wide Web and data streams, where data flow in and out like streams, as in applications like video surveillance, telecommunication, and sensor networks. The effective and efficient analysis of data in such different forms becomes a challenging task.
- The abundance of data, coupled with the need for powerful data analysis tools, has been described as a data rich but information poor situation. The fast-growing, tremendous amount of data, collected and stored in large and numerous data repositories, has far exceeded our human ability for comprehension without powerful tools (Figure 1.2).
- As a result, data collected in large data repositories become "data tombs"—data archives that are seldom visited.
- Consequently, important decisions are often made based not on the information-rich data stored in data repositories, but rather on a decision maker's intuition, simply because the decision maker does not have the tools to extract the valuable knowledge embedded in the vast amounts of data.
- In addition, consider expert system technologies, which typically rely on users or domain experts to manually input knowledge into knowledge bases. Unfortunately, this procedure is prone to biases and errors, and is extremely time-consuming and costly.
- Data mining tools perform data analysis and may uncover important data patterns, contributing greatly to business strategies, knowledge bases, and scientific and medical research. The widening gap between data and information calls for a systematic development of data mining tools that will turn data tombs into "golden nuggets" of knowledge.

## 1.2 So, What Is Data Mining?

- Simply stated, data mining refers to extracting or "mining" knowledge from large amounts of data. The term is actually a misnomer. Remember that the

mining of gold from rocks or sand is referred to as gold mining rather than rock or sand mining. Thus, ***data mining should have been more appropriately named "knowledge mining from data," which is, unfortunately, somewhat long. "Knowledge mining," a shorter term, may not reflect the emphasis on mining from large amounts of data.***

- Nevertheless, mining is a vivid term characterizing the process that finds a small set of precious nuggets from a great deal of raw material (Figure 1.3). Thus, such a misnomer that carries both "data" and "mining" became a popular choice. Many other terms carry a similar or slightly different meaning to data mining, such as knowledge mining from data, knowledge extraction, data/pattern analysis, data archaeology, and data dredging.
- ***Many people treat data mining as a synonym for another popularly used term, Knowledge Discovery from Data, or KDD.*** Alternatively, others view data mining as simply an


**Figure 1.3** Data mining—searching for knowledge (interesting patterns) in your data.

essential step in the process of knowledge discovery. Knowledge discovery as a process is depicted in Figure 1.4 and consists of an iterative sequence of the following steps:

***1. Data cleaning (to remove noise and inconsistent data)***

***2. Data integration (where multiple data sources may be combined)***

***3. Data selection (where data relevant to the analysis task are retrieved from the database)***

***4. Data transformation (where data are transformed or consolidated into forms appropriate for mining by performing summary or aggregation operations, for instance)***

***5. Data mining (an essential process where intelligent methods are applied in order to extract data patterns)***

***6. Pattern evaluation (to identify the truly interesting patterns representing knowledge based on some interestingness measures; Section 1.5)***

***7. Knowledge presentation (where visualization and knowledge representation techniques are used to present the mined knowledge to the user)***

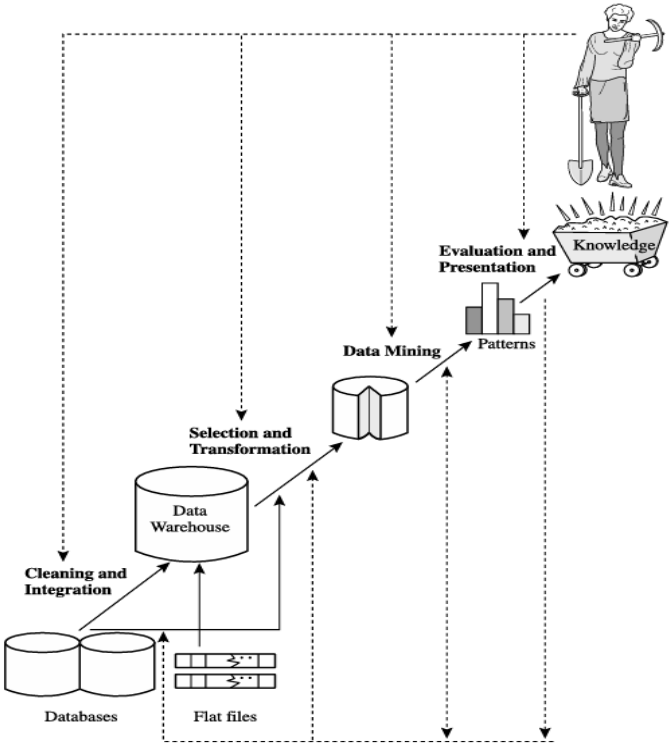



**Figure 1.4** Data mining as a step in the process of knowledge discovery.

Steps 1 to 4 are different forms of data preprocessing, where the data are prepared for mining. The data mining step may interact with the user or a knowledge base.

- The interesting patterns are presented to the user and may be stored as new knowledge in the knowledge base. Note that according to this view, data mining is only one step in the entire process, albeit an essential one because it uncovers hidden patterns for evaluation.
- We agree that data mining is a step in the knowledge discovery process. However, in industry, in media, and in the data base research milieu, the term data mining is becoming more popular than the longer term of knowledge discovery from data. Therefore, in the document/discussion in class, we choose to use the term data mining.

**Data Mining Typical Architecture**

- We adopt a broad view of data mining functionality: data mining is the process of discovering interesting knowledge from large amounts of data stored in databases, data warehouses, or other information repositories.
- Based on this view, the architecture of a typical data mining system may have the following major components (Figure 1.5): Database, data warehouse, World Wide Web, or other information repository: This is one or a set of databases, data warehouses, spreadsheets, or other kinds of information repositories. Data cleaning and data integration techniques may be performed on the data.
- ***Database or data warehouse server:*** The database or data warehouse server is responsible for fetching the relevant data based on the user's data mining request.
- A popular trend in the information industry is to perform data cleaning and data integration as a pre-processing step, where the resulting data are stored in a data warehouse.

- Sometimes data transformation and consolidation are performed before the data selection process, particularly in the case of data warehousing. Data reduction may also be performed to obtain a smaller representation of the original data without sacrificing its integrity.

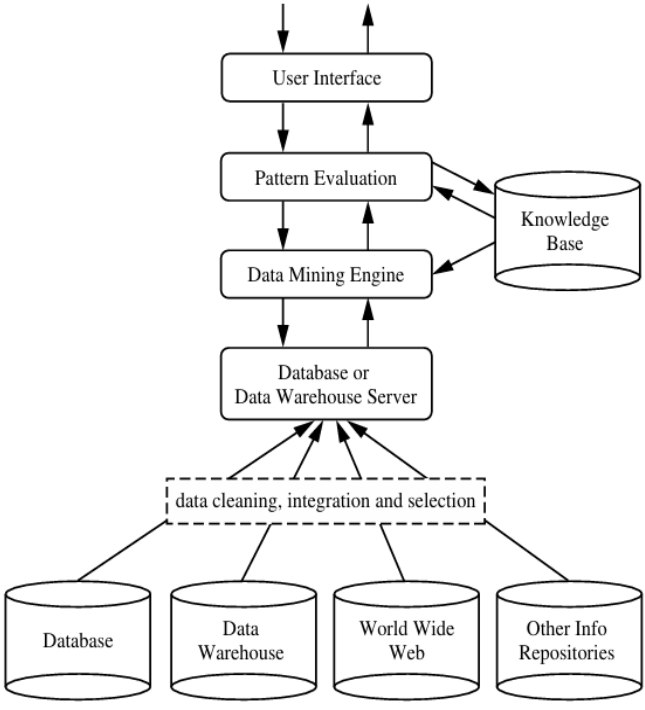



**Figure 1.5** Architecture of a typical data mining system.

**Knowledge base:** ***This is the domain knowledge that is used to guide the search or evaluate the interestingness of resulting patterns. Such knowledge can include concept hierarchies, used to organize attributes or attribute values into different levels of abstraction. Knowledge such as user beliefs, which can be used to assess a pattern's interestingness based on its unexpectedness, may also be included. Other examples of domain***

***knowledge are additional interestingness constraints or thresholds, and metadata (e.g., describing data from multiple heterogeneous sources).***

**Data mining engine**: This is essential to the data mining system and ideally consists of a set of functional modules for tasks such as characterization, association and correlation analysis, classification, prediction, cluster analysis, outlier analysis, and evolution analysis.

**Pattern evaluation module:** This component typically employs interestingness measures (Section 1.5) and interacts with the data mining modules so as to focus the search toward interesting patterns. It may use interestingness thresholds to filter out discovered patterns. Alternatively, the pattern evaluation module may be integrated with the mining module, depending on the implementation of the data mining method used. For efficient data mining, it is highly recommended to push the evaluation of pattern interestingness as deep as possible into the mining process so as to confine the search to only the interesting patterns.

**User interface**: This module communicates between users and the data mining system, allowing the user to interact with the system by specifying a data mining query or task, providing information to help focus the search, and performing exploratory data mining based on the intermediate data mining results. In addition, this component allows the user to browse database and data warehouse schemas or data structures, evaluate mined patterns, and visualize the patterns in different forms.

- From a data warehouse perspective, data mining can be viewed as an advanced stage of on-line analytical processing (OLAP). However, data mining goes far beyond the narrow scope of summarization-style analytical processing of data warehouse systems by incorporating more advanced techniques for data analysis.

- Although there are many "data mining systems" on the market, not all of them can perform true data mining. A data analysis system that does not handle large amounts of data should be more appropriately categorized as a machine learning system, a statistical data analysis tool, or an experimental system prototype.
- ***A system that can only perform data or information retrieval, including finding aggregate values, or that performs deductive query answering in large databases should be more appropriately categorized as a database system, an information retrieval system, or a deductive database system.***
- ***Data mining involves an integration of techniques from multiple disciplines such as database and data warehouse technology, statistics, machine learning, high-performance computing, pattern recognition, neural networks, data visualization, information retrieval, image and signal processing, and spatial or temporal data analysis.***
- For an algorithm to be scalable, its running time should grow approximately linearly in proportion to the size of the data, given the available system resources such as main memory and disk space. By performing data mining, interesting knowledge, regularities, or high-level information can be extracted from databases and viewed or browsed from different angles.
- ***The discovered knowledge can be applied to decision-making, process control, information management, and query processing.*** Therefore, datamining is considered one of the most important frontiers in a database and information systems and one of the most promising interdisciplinary developments in the information technology.

## 1.3 Data Mining—On What Kind of Data?

- ***In principle, data mining should be applicable to any kind of data repository, as well as to transient data, such as data streams. Thus the scope of our examination of data repositories will include relational databases, data warehouses, transactional databases, advanced database systems, flat***

***files, data streams. Advanced database systems include object-relational databases and specific application-oriented databases, such as spatial databases, time-series databases, text databases, and multimedia databases. The challenges and techniques of mining may differ for each of the repository systems.***

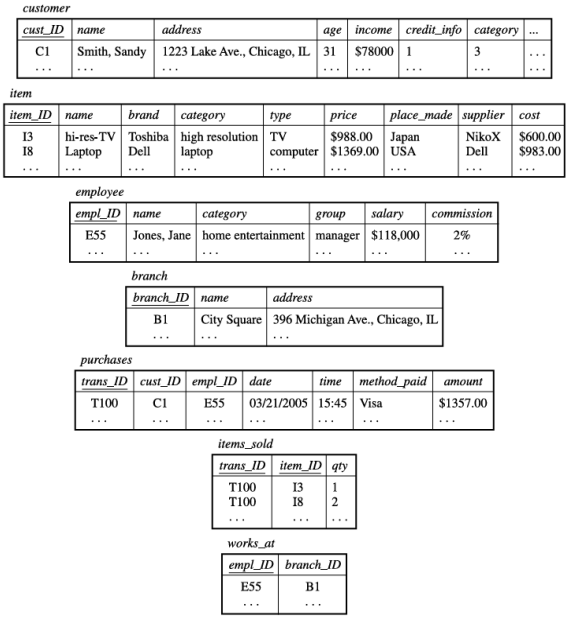
### 1.3.1 Relational Databases

- ***A database system, also called a database management system (DBMS), consists of a collection of interrelated data, known as a database, and a set of software programs to manage and access the data.***
- The software programs involve mechanisms for the definition of database structures; for data storage; for concurrent, shared, or distributed data access; and for ensuring the consistency and security of the information stored, despite system crashes or attempts at unauthorized access.
- A relational data base is a collection of tables, each of which is assigned a unique name. Each table consists of a set of attributes (columns or fields) and usually stores a large set of tuples (records or rows). Each tuple in a relational table represents an object identified by a unique key and described by a set of attribute values.
- ***A semantic data model, such as an entity-relationship (ER) data model, is often constructed for relational databases.*** An ER data model represents the database as a set of entities and their relationships.

Consider the following example.

**Example 1.1** A relational database for All Electronics. The All Electronics company is described by the following relation tables: customer, item, employee, and branch. Fragments of the tables described here are shown in Figure 1.6.

- ✓ The relation customer consists of a set of attributes, including a unique customer identity number (cust ID), customer name, address, age, occupation, annual income, credit information, category, and so on.
- ✓ Similarly, each of the relations item, employee, and branch consists of a set of attributes describing their properties.
- ✓ Tables can also be used to represent the relationships between or among multiple relation tables. For our example, these include purchases (customer purchases items, creating a sales transaction that is handled by an employee), items sold (lists the items sold in a given transaction), and works at (employee works at a branch of AllElectronics).
- Relational data can be accessed by database queries written in a relational query language, such as SQL, or with the assistance of graphical user interfaces. In the latter, the user may employ a menu, for example, to specify attributes to be included in the query, and the constraints on these attributes. A given query is transformed into a set of



*customer*

| *cust_ID* | *name* | *address* | *age* | *income* | *credit_info* | *category* | ... |
|---|---|---|---|---|---|---|---|
| C1 | Smith, Sandy | 1223 Lake Ave., Chicago, IL | 31 | $78000 | 1 | 3 | ... |
| ... | ... | ... | ... | ... | ... | ... | ... |

*item*

| *item_ID* | *name* | *brand* | *category* | *type* | *price* | *place_made* | *supplier* | *cost* |
|---|---|---|---|---|---|---|---|---|
| I3 | hi-res-TV | Toshiba | high resolution | TV | $988.00 | Japan | NikoX | $600.00 |
| I8 | Laptop | Dell | laptop | computer | $1369.00 | USA | Dell | $983.00 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... |

*employee*

| *empl_ID* | *name* | *category* | *group* | *salary* | *commission* |
|---|---|---|---|---|---|
| E55 | Jones, Jane | home entertainment | manager | $118,000 | 2% |
| ... | ... | ... | ... | ... | ... |

*branch*

| *branch_ID* | *name* | *address* |
|---|---|---|
| B1 | City Square | 396 Michigan Ave., Chicago, IL |
| ... | ... | ... |

*purchases*

| *trans_ID* | *cust_ID* | *empl_ID* | *date* | *time* | *method_paid* | *amount* |
|---|---|---|---|---|---|---|
| T100 | C1 | E55 | 03/21/2005 | 15:45 | Visa | $1357.00 |
| ... | ... | ... | ... | ... | ... | ... |

*items_sold*

| *trans_ID* | *item_ID* | *qty* |
|---|---|---|
| T100 | I3 | 1 |
| T100 | I8 | 2 |
| ... | ... | ... |

*works_at*

| *empl_ID* | *branch_ID* |
|---|---|
| E55 | B1 |
| ... | ... |

**Figure 1.6** Fragments of relations from a relational database for *AllElectronics*.

relational operations, such as join, selection, and projection, and is then optimized for efficient processing.

- A query allows retrieval of specified subsets of the data. Suppose that your job is to analyze the AllElectronics data. Through the use of relational queries, you can ask things like "Show me a list of all items that were sold in the last quarter."
- ***Relational languages also include aggregate functions such as sum, avg (average), count, max (maximum), and min(minimum).These allow you to ask things like "Show me the total sales of the last month, grouped by branch," or "How many sales transactions occurred in the month of December?" or "Which sales person had the highest amount of sales?"***
- When data mining is applied to relational data bases, we can go further by searching for trends or data patterns. For example, data mining systems can

analyze customer data to predict the credit risk of new customers based on their income, age, and previous credit information.

- Data mining systems may also detect deviations, such as items whose sales are far from those expected in comparison with the previous year. Such deviations can then be further investigated (e.g., has there been a change in packaging of such items, or a significant increase in price?).
- Relational databases are one of the most commonly available and rich information repositories, and thus they are a major data form in our study of data mining.

### 1.3.2 Data Warehouses

- Suppose that AllElectronics is a successful international company, with branches around the world. Each branch has its own set of databases. The president of AllElectronics has asked you to provide an analysis of the company's sales per item type per branch for the third quarter. This is a difficult task, particularly since the relevant data are spread out over several databases, physically located at numerous sites.
- If AllElectronics had a data warehouse, this task would be easy. A data ware house is a repository of information collected from multiple sources, stored under a unified schema, and that usually resides at a single site. Data warehouses are con structed via a process of data cleaning, data integration, data transformation, data loading, and periodic data refreshing.

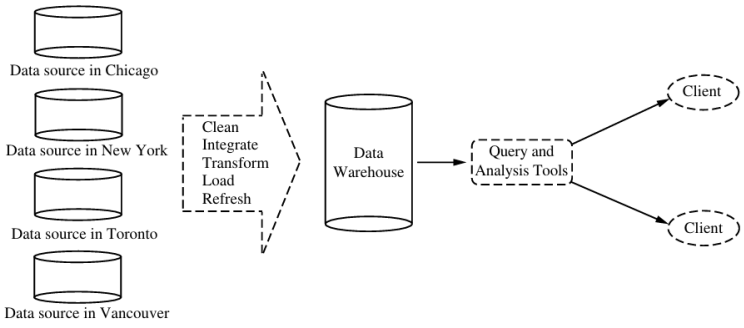



**Figure 1.7** Typical framework of a data warehouse for *AllElectronics*.

- To facilitate decision making, the data in a data warehouse are organized around major subjects, such as customer, item, supplier, and activity. The data are stored to provide information from a historical perspective (such as from the past 5–10 years) and are typically summarized.
- For example, rather than storing the details of each sales transaction, the data warehouse may store a summary of the transactions per item type for each store or, summarized to a higher level, for each sales region.
- A data warehouse is usually modelled by a multidimensional database structure, where each dimension corresponds to an attribute or a set of attributes in the schema, and each cell stores the value of some aggregate measure, such as count or sales amount.
- The actual physical structure of a data warehouse may be a relational data store or a multidimensional data cube. A data cube provides a multidimensional view of data and allows the precomputation and fast accessing of summarized data.

**Example 1.2**

- A data cube for AllElectronics. A data cube for summarized sales data of AllElectronics is presented in Figure 1.8(a). The cube has three dimensions: address (with city values Chicago, New York, Toronto, Vancouver), time (with quarter values Q1, Q2, Q3, Q4), and item (with item type values home

entertainment, computer, phone,security). The aggregate value stored in each cell of the cube is sales amount (in thousands).

- For example, the total sales for the first quarter, Q1, for items relating to security systems in Vancouver is $400,000, as stored in cell Vancouver, Q1, security. Additional cubes may be used to store aggregate sums over each dimension, corresponding to the aggregate values obtained using different SQL group-bys (e.g., the total sales amount per city and quarter, or per city and item, or per quarter and item, or per each individual dimension).

I have also heard about data marts.

***What is the difference between a data warehouse and a data mart?" you may ask***.

- ***A data warehouse collects information about subjects that span an entire organization, and thus its scope is enterprise-wide. A data mart, on the other hand, is a department subset of a data warehouse. It focuses on selected subjects, and thus its scope is department-wide.***
- By providing multidimensional data views and the precomputation of summarized data, data warehouse systems are well suited for on-line analytical processing, or OLAP. OLAP operations use background knowledge regarding the domain of the data being studied in order to allow the presentation of data at different levels of abstraction. Such operations accommodate different user viewpoints.
- Examples of OLAP operations include drill-down and roll-up, which allow the user to view the data at differing degrees of summarization, as illustrated in Figure 1.8(b). For instance, we can drill down on sales data summarized by quarter to see the data summarized by month. Similarly, we can roll up on sales data summarized by city to view the data summarized by country.

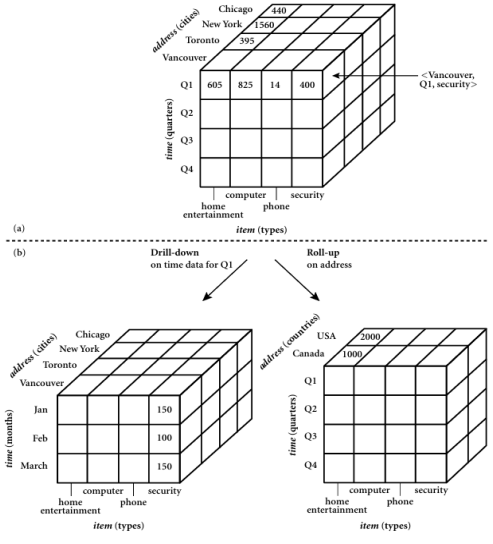



**Figure 1.8** A multidimensional data cube, commonly used for data warehousing, (a) showing summarized data for *AllElectronics* and (b) showing summarized data resulting from drill-down and roll-up operations on the cube in (a). For improved readability, only some of the cube cell values are shown.

### 1.3.3 *Transactional Databases In general, a transactional data base consists of a file where each record represents a transaction. A transaction typically includes a unique transaction identity number (trans ID) and a list of the items making up the transaction (such as items purchased in a store).*

| *trans_ID* | *list of item_IDs* |
|---|---|
| T100 | I1, I3, I8, I16 |
| T200 | I2, I8 |
| ... | ... |

**Figure 1.9** Fragment of a transactional database for sales at *AllElectronics*.

- The transactional database may have additional tables associated with it, which contain other information, such as the date of the transaction, the customer ID number, the ID number of the salesperson and of the branch at which the sale occurred, and so on.

**Example 1.3**

- A transactional database for AllElectronics. Transactions can be stored in a table, with one record per transaction. A fragment of a transactional database for AllElectronics is shown in Figure 1.9.
- From the relational database point of view, the sales table in Figure 1.9 is a nested relation because the attribute list of item IDs contains a set of items. Because most relational data base systems do not support nested relational structures, the transactional database is usually either stored in a flat file in a format similar to that of the table in Figure 1.9 or unfolded into a standard relation in a format similar to that of the items sold table in Figure 1.6.
- As an analyst of the AllElectronics database, you may ask, "Show me all the items purchased by Sandy Smith" or "How many transactions include item number I3?" Answering such queries may require a scan of the entire transactional database.
- Suppose you would like to dig deeper into the data by asking, "Which items sold well together?" This kind of market basket data analysis would enable you to bundle groups of items together as a strategy for maximizing sales.
- For example, given the knowledge that printers are commonly purchased together with computers, you could offer an expensive model of printers at a discount to customers buying selected computers, in the hopes of selling more of the expensive printers. A regular data retrieval system is not able to answer queries like the one above.
- However, data mining systems for transactional data can do so by identifying frequent item sets, that is, sets of items that are frequently sold together.

### 1.3.4 Advanced Data and Information Systems and Advanced Applications

- Relational database systems have been widely used in business applications. With the progress of database technology, various kinds of advanced data and

information systems have emerged and are undergoing development to address the requirements of new applications.

- ***The new database applications include handling spatial data (such as maps), engineering design data (such as the design of buildings, system components, or integrated circuits), hypertext and multimedia data (including text, image, video, and audio data), time-related data (such as historical records or stock exchange data), stream data (such as video surveillance and sensor data, where data flow in and out like streams), and the World Wide Web (a huge, widely distributed information repository made available by the Internet). These applications require efficient data structures and scalable methods for handling complex object structures; variable-length records; semi-structured or unstructured data; text, spatiotemporal, and multimedia data; and database schemas with complex structures and dynamic changes.***
- In response to these needs, advanced data base systems and specific application-oriented database systems have been developed. These include object-relational database systems, temporal and time-series database systems, spatial and spatiotemporal database systems, text and multimedia database systems, heterogeneous and legacy database systems, data stream management systems, and Web-based global information systems.
- While such databases or information repositories require sophisticated facilities to efficiently store, retrieve, and update large amounts of complex data, they also provide fertile grounds and raise many challenging research and implementation issues for data mining.

**Object-Relational Databases**

- ***Object-relational databases are constructed based on an object-relational data model. This model extends the relational model by providing a rich data type for handling com plex objects and object orientation. Because most sophisticated database applications need to handle complex objects***

***and structures, object-relational databases are becoming increasingly popular in industry and applications.***

- Conceptually, the object-relational data model inherits the essential concepts of object-oriented databases, where, in general terms, each entity is considered as an object. Following the AllElectronics example, objects can be individual employees, customers, or items. Data and code relating to an object are encapsulated into a single unit. Each object has associated with it the following:

✓ ***A set of variables that describe the objects. These correspond to attributes in the entity-relationship and relational models.***

✓ ***Aset of messages that the object can use to communicate with other objects, or with the rest of the database system.***

✓ ***A set of methods, where each method holds the code to implement a message. Upon receiving a message, the method returns a value in response.*** For instance, the method for the message get photo(employee) will retrieve and return a photo of the given employee object.

- Objects that share a common set of properties can be grouped into an object class. Each object is an instance of its class. Object classes can be organized into class/subclass hierarchies so that each class represents properties that are common to objects in that class.
- For instance, an employee class can contain variables like name, address, and birth date. Suppose that the class , sales person, is a subclass of the class, employee. A sales person object would inherit all of the variables pertaining to its superclass of employee. In addition, it has all of the variables that pertain specifically to being a salesperson (e.g., commission). Such a class inheritance feature benefits information sharing.
- For data mining in object-relational systems, techniques need to be developed for handling complex object structures, complex data types, class and subclass hierarchies, property inheritance, and methods and procedures.

## Temporal Databases, Sequence Databases, and Time-Series Databases

- ***A temporal database typically stores relational data that include time-related attributes. These attributes may involve several timestamps, each having different semantics. A sequence database stores sequences of ordered events, with or without a concrete notion of time.***
- Examples include customer shopping sequences, Web click streams, and biological sequences. A time-series data base stores sequences of values or events obtained over repeated measurements of time (e.g., hourly, daily, weekly). Examples include data collected from the stock exchange, inventory control, and the observation of natural phenomena (like temperature and wind).
- ***Data mining techniques can be used to find the characteristics of object evolution, or the trend of changes for objects in the database. Such information can be useful in decision making and strategy planning.***
- ***For instance, the mining of banking data may aid in the scheduling of bank tellers according to the volume of customer traffic.*** Stock exchange data can be mined to uncover trends that could help you plan investment strategies (e.g., when is the best time to purchase AllElectronics stock?).
- Such analyses typically require defining multiple granularity of time. For example, time may be decomposed according to fiscal years, academic years, or calendar years. Years may be further decomposed into quarters or months.

## Spatial Databases and Spatiotemporal Databases

- ***Spatial databases contain spatial-related information. Examples include geographic (map) databases, very large-scale integration (VLSI) or computed-aided design databases, and medical and satellite image databases. Spatial data may be represented in raster format, consisting of n-dimensional bit maps or pixel maps.***

- For example, a 2-D satellite image may be represented as raster data, where each pixel registers the rainfall in a given area. Maps can be represented in vector format, where roads, bridges, buildings, and lakes are represented as unions or overlays of basic geometric constructs, such as points, lines, polygons, and the partitions and networks formed by these components.
- Geographic databases have numerous applications, ranging from forestry and ecology planning to providing public service information regarding the location of telephone and electric cables, pipes, and sewage systems.
- In addition, geographic databases are commonly used in vehicle navigation and dispatching systems. An example of such a system for taxis would store a city map with information regarding one-way streets, suggested routes for moving from region A to region B during rush hour, and the location of restaurants and hospitals, as well as the current location of each driver.

"***What kind of data mining can be performed on spatial databases?" you may ask***.

- ***Data mining may uncover patterns describing the characteristics of houses located near a specified kind of location, such as a park, for instance***. Other patterns may describe the climate of mountainous areas located at various altitudes, or describe the change in trend of metropolitan poverty rates based on city distances from major highways.
- ***The relation- ships among a set of spatial objects can be examined in order to discover which subsets of objects are spatially auto-correlated or associated. Clusters and outliers can be identified by spatial cluster analysis.*** Moreover, spatial classification can be performed to construct models for prediction based on the relevant set of features of the spatial objects.
- Further- more, "spatial data cubes" may be constructed to organize data into multidimensional structures and hierarchies, on which OLAP operations (such as drill-down and roll-up) can be performed.

- ***A spatial database that stores spatial objects that change with time is called a spatiotemporal database, from which interesting information can be mined.***
- For example, we may be able to group the trends of moving objects and identify some strangely moving vehicles, or distinguish a bioterrorist attack from a normal outbreak of the flu based on the geographic spread of a disease with time.

**Text Databases and Multimedia Databases**

- ***Text databases are databases that contain word descriptions for objects. These word descriptions are usually not simple keywords but rather long sentences or paragraphs, such as product specifications, error or bug reports, warning messages, summary reports, notes, or other documents.***
- Text databases may be highly unstructured (such as some Web pages on the World Wide Web). Some text databases may be somewhat structured, that is, semi-structured (such as e-mail messages and many HTML/XML Web pages), whereas others are relatively well structured (such as library catalogue databases). Text databases with highly regular structures typically can be implemented using relational database systems.

***"What can data mining on text databases uncover?"***

- ***By mining text data, one may uncover general and concise descriptions of the text documents, keyword or content associations, as well as the clustering behavior of text objects.*** To do this, standard data mining methods need to be integrated with information retrieval techniques and the construction or use of hierarchies specifically for text data (such as dictionaries and thesauruses), as well as discipline-oriented term classification systems (such as in biochemistry, medicine, law, or economics).
- Multimedia databases store image, audio, and video data. They are used in applications such as picture content-based retrieval, voice-mail systems, video-on-demand systems, the World Wide Web, and speech-based user interfaces that

recognize spoken commands. Multimedia databases must support large objects, because data objects such as video can require gigabytes of storage.

- Specialized storage and search techniques are also required. Because video and audio data require real-time retrieval at a steady and predetermined rate in order to avoid picture or sound gaps and system buffer overflows, such data are referred to as continuous-media data.
- For multimedia data mining, storage and search techniques need to be integrated with standard data mining methods. Promising approaches include the construction of multimedia data cubes, the extraction of multiple features from multimedia data, and similarity-based pattern matching.

**Heterogeneous Databases and Legacy Databases**

- ***A heterogeneous database consists of a set of interconnected, autonomous component databases. The components communicate in order to exchange information and answer queries.***
- Objects in one component database may differ greatly from objects in other component databases, making it difficult to assimilate their semantics into the overall heterogeneous database.
- Many enterprises acquire legacy databases as a result of the long history of information technology development (including the application of different hardware and operating systems).
- ***A legacy database is a group of heterogeneous databases that com- bines different kinds of data systems, such as relational or object-oriented databases, hierarchical databases, network databases, spreadsheets, multimedia databases, or file systems. The heterogeneous databases in a legacy database may be connected by intra- or inter-computer networks.***
- Information exchange across such databases is difficult because it would require precise transformation rules from one representation to another, considering diverse semantics.

- Consider, for example, the problem in exchanging information regarding student academic performance among different schools. Each school may have its own computer system and use its own curriculum and grading system. One university may adopt a quarter system, offer three courses on database systems, and assign grades from A+ to F, whereas another may adopt a semester system, offer two courses on databases, and assign grades from 1 to 10. It is very difficult to work out precise course-to-grade transformation rules between the two universities, making information exchange difficult.
- Data mining techniques may provide an interesting solution to the information exchange problem by performing statistical data distribution and correlation analysis, and transforming the given data into higher, more generalized, conceptual levels (such as fair, good, or excellent for student grades), from which information exchange can then more easily be performed.

**Data Streams**

- Many applications involve the generation and analysis of a new kind of data, called stream data, where data flow in and out of an observation platform (or window) dynamically. Such data streams have the following unique features: huge or possibly infinite volume, dynamically changing, flowing in and out in a fixed order, allowing only one or a small number of scans, and demanding fast (often real-time) response time.
- ***Typical examples of data streams include various kinds of scientific and engineering data, time-series data, and data produced in other dynamic environments, such as power supply, network traffic, stock exchange, telecommunications, Web click streams, video surveillance, and weather or environment monitoring because data streams are normally not stored in any kind of data repository, effective and efficient management and analysis of stream data poses great challenges to researchers.***

- Currently, many researchers are investigating various issues relating to the development of data stream management systems.
- A typical query model in such a system is the continuous query model, where predefined queries constantly evaluate incoming streams, collect aggregate data, report the current status of data streams, and respond to their changes.
- Mining data streams involves the efficient discovery of general patterns and dynamic changes within stream data. For example, we may like to detect intrusions of a computer network based on the anomaly of message flow, which may be discovered by clustering data streams, dynamic construction of stream models, or comparing the current frequent patterns with that at a certain previous time. Most stream data reside at a rather low level of abstraction, whereas analysts are often more interested in higher and multiple levels of abstraction. Thus, multilevel, multidimensional on-line analysis and mining should be performed on stream data as well.

**The World Wide Web**

- The World Wide Web and its associated distributed information services, such as Yahoo!, Google, America Online, and AltaVista, provide rich, worldwide, on-line information services, where data objects are linked together to facilitate interactive access. Users seeking information of interest traverse from one object via links to another. Such systems provide ample opportunities and challenges for data mining.
- ***For example, understanding user access patterns will not only help improve system design (by providing efficient access between highly correlated objects), but also leads to better marketing decisions (e.g., by placing advertisements in frequently visited documents, or by providing better customer/user classification and behavior analysis). Capturing user access patterns in such distributed information environments is called Web usage mining (or Weblog mining).***

- Although Web pages may appear fancy and informative to human readers, they can be highly unstructured and lack a predefined schema, type, or pattern. Thus it is difficult for computers to understand the semantic meaning of diverse Web pages and structure them in an organized way for systematic information retrieval and data mining. Web services that provide keyword-based searches without understanding the context behind the Web pages can only offer limited help to users.
- For example, a Web search based on a single keyword may return hundreds of Web page pointers containing the keyword, but most of the pointers will be very weakly related to what the user wants to find. Data mining can often provide additional help here than Web search services.
- For example, authoritative Web page analysis based on linkages among Web pages can help rank Web pages based on their importance, influence, and topics. Automated Web page clustering and classification help group and arrange Web pages in a multidimensional manner based on their contents.
- Web community analysis helps identify hidden Web social networks and communities and observe their evolution. Web mining is the development of scalable and effective Web data analysis and mining methods. It may help us learn about the distribution of information on the Web in general, characterize and classify Web pages, and uncover Web dynamics and the association and other relationships among different Web pages, users, communities, and Web-based activities.

## 1.4 Datamining Functionalities—What Kinds of Patterns Can Be Mined?

- Data mining functionalities are used to specify the kind of patterns to be found in data mining tasks. In general, data mining tasks can be classified into two categories: descriptive and predictive.
- ***Descriptive mining tasks characterize the general properties of the data in the database. Predictive mining tasks perform inference on the current data in order to make predictions.***

- ***In some cases, users may have no idea regarding what kinds of patterns in their data may be interesting, and hence may like to search for several different kinds of patterns in parallel. Thus it is important to have a data mining system that can mine multiple kinds of patterns to accommodate different user expectations or applications.***
- ***Furthermore, data mining systems should be able to discover patterns at various granularity (i.e., different levels of abstraction).*** Data mining systems should also allow users to specify hints to guide or focus the search for interesting patterns. Because some patterns may not hold for all of the data in the database, a measure of certainty or "trustworthiness" is usually associated with each discovered pattern. Data mining functionalities, and the kinds of patterns they can discover, are described below.

### 1.4.1 Concept/Class Description: Characterization and Discrimination

- Data can be associated with classes or concepts. For example, in the *AllElectronics* store, classes of items for sale include *computers* and *printers*, and concepts of customers include *bigSpenders* and *budgetSpenders*. It can be useful to describe individual classes and con- cepts in summarized, concise, and yet precise terms. Such descriptions of a class or a concept are called **class/concept descriptions**. These descriptions can be derived via
  (1) *data characterization*, by summarizing the data of the class under study (often called the **target class**) in general terms, or (2) *data discrimination*, by comparison of the target class with one or a set of comparative classes (often called the **contrasting classes**), or (3) both data characterization and discrimination.

- ***Data characterization is a summarization of the general characteristics or features of a target class of data. The data corresponding to the user-specified class are typically collected by a database query.***

- For example, to study the characteristics of software products whose sales increased by 10% in the last year, the data related to such products can be collected by executing an SQL query.There are several methods for effective data summarization and characterization. Simple data summaries based on statistical measures and plots. The data cube–based OLAP roll-up operation (Section 1.3.2) can be used to perform user-controlled data summarization along a specified dimension.
- ***An attribute-oriented induction technique can be used to perform data generalization and characterization without step-by-step user interaction.***
- The output of data characterization can be presented in various forms. Examples include pie charts, bar charts, curves, multidimensional data cubes, and multidimensional tables, including crosstabs. The resulting descriptions can also be presented as generalized relations or in rule form (called characteristic rules).

**Example 1.4 Data characterization**.

- A data mining system should be able to produce a description summarizing the characteristics of customers who spend more than $1,000 a year at AllElectronics. The result could be a general profile of the customers, such as they are 40–50 years old, employed, and have excellent credit ratings.
- The system should allow users to drill down on any dimension, such as on occupation in order to view these customers according to their type of employment.
- ***Data discrimination is a comparison of the general features of target class data objects with the general features of objects from one or a set of contrasting classes. The target and contrasting classes can be specified by the user, and the corresponding data objects retrieved through database queries.***
- For example, the user may like to compare the general features of software products whose sales increased by 10% in the last year with those whose sales

decreased by at least 30% during the same period. The methods used for data discrimination are similar to those used for data characterization.

- "***How are discrimination descriptions output?"*** The forms of output presentation are similar to those for characteristic descriptions, although discrimination descriptions should include comparative measures that help distinguish between the target and contrasting classes. Discrimination descriptions expressed in rule form are referred to as discriminant rules.

***Example 1.5 Data discrimination***.

- A data mining system should be able to compare two groups of AllElectronics customers, such as those who shop for computer products regularly (more than two times a month) versus those who rarely shop for such products (i.e., less than three times a year).
- The resulting description provides a general comparative profile of the customers, such as 80% of the customers who frequently purchase computer products are between 20 and 40 years old and have a university education, whereas 60% of the customers who infrequently buy such products are either seniors or youths, and have no university degree.
- Drilling down on a dimension, such as occupation, or adding new dimensions, such as income level, may help in finding even more discriminative features between the two classes.

### 1.4.2 Mining Frequent Patterns, Associations, and Correlations

***Frequent patterns, as the name suggests, are patterns that occur frequently in data.*** There are many kinds of frequent patterns, including itemsets, subsequences, and substructures.

- ***A frequent itemset typically refers to a set of items that frequently appear together in a transactional data set, such as milk and bread. A frequently occurring subsequence, such as the pattern that customers tend to purchase***

***first a PC, followed by a digital cam- era, and then a memory card, is a (frequent) sequential pattern.***

- A substructure can refer to different structural forms, such as graphs, trees, or lattices, which may be combined with itemsets or subsequences. If a substructure occurs frequently, it is called a (frequent) structured pattern. Mining frequent patterns leads to the discovery of interesting associ- ations and correlations within data.

**Example 1.6** Association analysis. Suppose, as a marketing manager of AllElectronics, you would like to determine which items are frequently purchased together within the same transactions.

- ***An example of such a rule, mined from the AllElectronics transactional database, is***

***buys(X, "computer") ⇒ buys(X, "software") [support = 1%, confidence = 50%]***

***where X is a variable representing a customer.***

- ***A confidence, or certainty, of 50% means that if a customer buys a computer, there is a 50% chance that she will buy software as well. A 1% support means that 1% of all of the transactions under analysis showed that computer and software were purchased together.***
- ***This association rule involves a single attribute or predicate (i.e., buys) that repeats. Association rules that contain a single predicate are referred to as single-dimensional association rules. Dropping the predicate notation, the above rule can be written simply as "computer software [1%, 50%]".***

Suppose, instead, that we are given the AllElectronics relational database relating to purchases. A data mining system may find association rules like

age(X , “20...29”) ∧income(X , “20K...29K”) ⇒ buys(X , “CD player”) [support = 2%, confidence = 60%]

- The rule indicates that of the AllElectronics customers under study, 2% are 20 to 29 years of age with an income of 20,000 to 29,000 and have purchased a CD player at *AllElectronics*. There is a 60% probability that a customer in this age and income group will purchase a CD player. Note that this is an association between more than one attribute, or predicate (i.e., *age, income*, and *buys*).
- Adopting the terminology used in multidimensional databases, where each attribute is referred to as a dimension, the above rule can be referred to as a **multidimensional association rule**.
- Typically, association rules are discarded as uninteresting if they do not satisfy both a minimum support threshold and a minimum confidence threshold. Additional anal- ysis can be performed to uncover interesting statistical correlations between associated attribute-value pairs. Sequential pattern mining and structured pattern mining are considered advanced topics.

### 1.4.3 Classification and Prediction

- Classification is the process of finding a model (or function) that describes and distinguishes data classes or concepts, for the purpose of being able to use the model to predict the class of objects whose class label is unknown. The derived model is based on the analysis of a set of training data (i.e., data objects whose class label is known).

***“How is the derived model presented?”***

- ***The derived model may be represented in various forms, such as classification (IF-THEN) rules, decision trees, mathematical formulae, or neural networks*** (Figure 1.10). A decision tree is a flow-chart-like tree structure, where each node denotes a test on an attribute value, each branch

represents an outcome of the test, and tree leaves represent classes or class distributions.

- ***Decision trees can easily be converted to classification rules. A neural network, when used for classification, is typically a collection of neuron-like processing units with weighted connections between the units. There are many other methods for constructing classification models, such as naïve Bayesian classification, support vector machines, and k-nearest neighbor classification.***
- Whereas classification predicts categorical (discrete, unordered) labels, prediction models continuous-valued functions. That is, it is used to predict missing or unavailable numerical data values rather than class labels.
- Although the term prediction may refer to both numeric prediction and class label prediction. Regression analysis is a statistical methodology that is most often used for numeric prediction, although other methods exist as well. Prediction also encompasses the identification of distribution trends based on the available data.
- Classification and prediction may need to be preceded by relevance analysis, which attempts to identify attributes that do not contribute to the classification or prediction process. These attributes can then be excluded.

**Example 1.7**

- Classification and prediction. Suppose, as sales manager of AllElectronics, you would like to classify a large set of items in the store, based on three kinds of responses to a

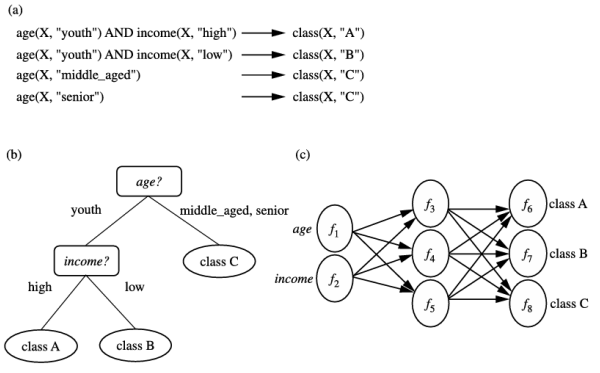



**Figure 1.10** A classification model can be represented in various forms, such as (a) IF-THEN rules, (b) a decision tree, or a (c) neural network.

sales campaign: good response, mild response, and no response.

- You would like to derive a model for each of these three classes based on the descriptive features of the items, such as price, brand, place made, type, and category. The resulting classification should maximally distinguish each class from the others, presenting an organized picture of the data set. Suppose that the resulting classification is expressed in the form of a decision tree.
- The decision tree, for instance, may identify price as being the single factor that best distinguishes the three classes. The tree may reveal that, after price, other features that help further distinguish objects of each class from another include brand and place made. Such a decision tree may help you understand the impact of the given sales campaign and design a more effective campaign for the future.
- Suppose instead, that rather than predicting categorical response labels for each store item, you would like to predict the amount of revenue that each item will generate during an upcoming sale at AllElectronics, based on previous sales data. This is an example of (numeric) prediction because the model constructed will predict a continuous-valued function, or ordered value.

### 1.4.4 Cluster Analysis

***"What is cluster analysis?"***

- Unlike classification and prediction, which analyze class-labeled data objects, **clustering** analyzes data objects without consulting a known class label.

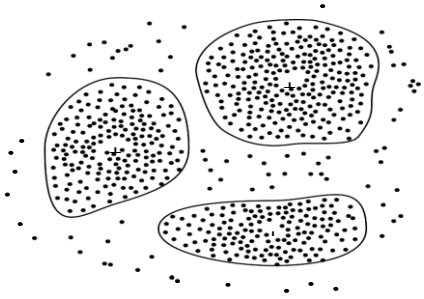


**Figure 1.11** A 2-D plot of customer data with respect to customer locations in a city, showing three data clusters. Each cluster "center" is marked with a "+".

- ***In general, the class labels are not present in the training data simply because they are not known to begin with. Clustering can be used to generate such labels. The objects are clustered or grouped based on the principle of maximizing the intraclass similarity and minimizing the interclass similarity.***
- ***That is, clusters of objects are formed so that objects within a cluster have high similarity in comparison to one another, but are very dissimilar to objects in other clusters.***
- Each cluster that is formed can be viewed as a class of objects, from which rules can be derived. Clustering can also facilitate taxonomy formation, that is, the organization of observations into a hierarchy of classes that group similar events together.

**Example 1.8 Cluster analysis.**

- Cluster analysis can be performed on *AllElectronics* customer data in order to identify homogeneous subpopulations of customers. These clusters may represent individual target groups for marketing. Figure 1.11 shows a 2-D plot of customers with respect to customer locations in a city. Three clusters of data points are evident.

### 1.4.5 Outlier Analysis

- ***A database may contain data objects that do not comply with the general behavior or model of the data. These data objects are outliers. Most data mining methods discard outliers as noise or exceptions. However, in some applications such as fraud detection, the rare events can be more interesting than the more regularly occurring ones. The analysis of outlier data is referred to as outlier mining.***
- Outliers may be detected using statistical tests that assume a distribution or probability model for the data, or using distance measures where objects that are a substantial distance from any other cluster are considered outliers. Rather than using statistical or distance measures, deviation-based methods identify outliers by examining differences in the main characteristics of objects in a group.

**Example 1.9 Outlier analysis**.

- Outlier analysis may uncover fraudulent usage of credit cards by detecting purchases of extremely large amounts for a given account number in comparison to regular charges incurred by the same account. Outlier values may also be detected with respect to the location and type of purchase, or the purchase frequency.

### 1.4.6 Evolution Analysis

- Data evolution analysis describes and models regularities or trends for objects whose behavior changes over time. Although this may include characterization, discrimination, association and correlation analysis, classification, prediction, or

clustering of time- related data, distinct features of such an analysis include time-series data analysis, sequence or periodicity pattern matching, and similarity-based data analysis.

**Example 1.10** Evolution analysis.

- Suppose that you have the major stock market (time-series) data of the last several years available from the New York Stock Exchange and you would like to invest in shares of high-tech industrial companies. A data mining study of stock exchange data may identify stock evolution regularities for overall stocks and for the stocks of particular companies. Such regularities may help predict future trends in stock market prices, contributing to your decision making regarding stock investments.

## 1.5 Are All of the Patterns Interesting?

- A data mining system has the potential to generate thousands or even millions of patterns, or rules.

  *"So,"* you may ask, *"are all of the patterns interesting?"* Typically not—only a small fraction of the patterns potentially generated would actually be of interest to any given user.

  This raises some serious questions for data mining. You may wonder,

  ***"What makes a pattern interesting?***

  ***Can a data mining system generate all of the interesting patterns?***

  ***Can a data mining system generate only interesting patterns?"***

- ***To answer the first question, a pattern is interesting if it is (1) easily understood by humans, (2) valid on new or test data with some degree of certainty, (3) potentially useful,***

and (4) *novel*. A pattern is also interesting if it validates a hypothesis that the user *sought to confirm*. An interesting pattern represents **knowledge**.

Several **objective measures of pattern interestingness** exist. These are based on the structure of discovered patterns and the statistics underlying them. An objective measure for association rules of the form $X \Rightarrow Y$ is rule **support**, representing the percentage of transactions from a transaction database that the given rule satisfies. This is taken to be the probability $P(X \cup Y)$, where $X \cup Y$ indicates that a transaction contains both $X$ and $Y$, that is, the union of itemsets $X$ and $Y$. Another objective measure for association rules is **confidence**, which assesses the degree of certainty of the detected association. This is taken to be the conditional probability $P(Y|X)$, that is, the probability that a transaction containing $X$ also contains $Y$. More formally, support and confidence are defined as

$$support(X \Rightarrow Y) = P(X \cup Y).$$
$$confidence(X \Rightarrow Y) = P(Y|X).$$

- ***In general, each interestingness measure is associated with a threshold, which may be controlled by the user. For example, rules that do not satisfy a confidence threshold of, say, 50% can be considered uninteresting. Rules below the threshold likely reflect noise, exceptions, or minority cases and are probably of less value.***
- ***Although objective measures help identify interesting patterns, they are insufficient unless combined with subjective measures that reflect the needs and interests of a particular user.***
- For example, patterns describing the characteristics of customers who shop frequently at AllElectronics should interest the marketing manager, but may be of little interest to analysts studying the same database for patterns on employee performance.
- Furthermore, many patterns that are interesting by objective standards may represent common knowledge and, therefore, are actually uninteresting. Subjective interesting- ness measures are based on user beliefs in the data. These measures find patterns interesting if they are unexpected (contradicting a user's belief) or offer strategic information on which the user can act. In the latter case, such patterns are referred to as actionable. Patterns that are expected can be interesting if they confirm a hypothesis that the user wished to validate, or resemble a user's hunch.

***The second question—"Can a data mining system generate all of the interesting patterns?"—refers to the completeness of a data mining algorithm***.

- It is often unrealistic and inefficient for data mining systems to generate all of the possible patterns. Instead, user-provided constraints and interestingness measures should be used to focus the search. For some mining tasks, such as association, this is often sufficient to ensure the completeness of the algorithm. Association rule mining is an example where the use of constraints and interestingness measures can ensure the completeness of mining.

***Finally, the third question— "Can a data mining system generate only interesting pat- terns?"—is an optimization problem in data mining.*** It is highly desirable for data mining systems to generate only interesting patterns. This would be much more efficient for users and data mining systems, because neither would have to search through the pat- terns generated in order to identify the truly interesting ones. Progress has been made in this direction; however, such optimization remains a challenging issue in data mining.

## 1.6 Classification of Data Mining Systems

- ***Data mining is an interdisciplinary field, the confluence of a set of disciplines, including database systems, statistics, machine learning, visualization, and information science*** (Figure 1.12).
- Moreover, depending on the data mining approach used, techniques from other disciplines may be applied, such as neural networks, fuzzy and/or rough set theory, knowledge representation, inductive logic programming, or high-performance computing.
- Depending on the kinds of data to be mined or on the given data mining application, the data mining system may also integrate techniques from spatial data analysis, information retrieval, pattern recognition, image analysis, signal processing, computer graphics, Web technology, economics, business, bioinformatics, or psychology, because of the diversity of disciplines

contributing to data mining, data mining research is expected to generate a large variety of data mining systems.

- Therefore, it is necessary to provide a clear classification of data mining systems, which may help potential users distinguish between such systems and identify those that best match their needs. Data mining systems can be categorized according to various criteria, as follows:

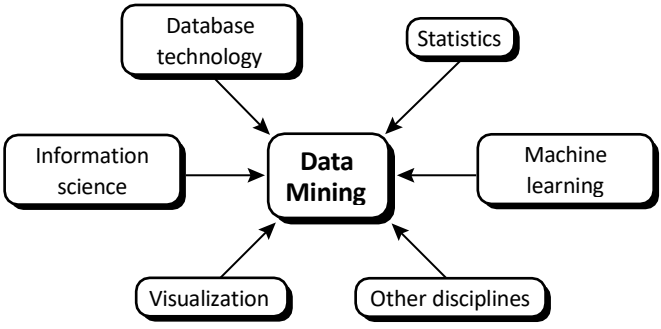



**Figure 1.12** Data mining as a confluence of multiple disciplines.

- Classification according to the kinds of databases mined: A data mining system can be classified according to the kinds of databases mined.
- Database systems can be classified according to different criteria (such as data models, or the types of data or applications involved), each of which may require its own data mining technique. Data mining systems can therefore be classified accordingly.
- For instance, if classifying according to data models, we may have a relational, transactional, object-relational, or data warehouse mining system. If classifying according to the special types of data handled, we may have a spatial, time-series, text, stream data, multimedia data mining system, or a World Wide Web mining system.

***Classification according to the kinds of knowledge mined***:

- ***Data mining systems can be categorized according to the kinds of knowledge they mine, that is, based on data mining functionalities, such as characterization, discrimination, association and correlation analysis, classification, prediction, clustering, outlier analysis, and evolution analysis.*** A comprehensive data mining system usually provides multiple and/or integrated data mining functionalities.
- ***Moreover, data mining systems can be distinguished based on the granularity or levels of abstraction of the knowledge mined, including generalized knowledge (at a high level of abstraction), primitive-level knowledge (at a raw data level), or knowledge at multiple levels (considering several levels of abstraction).***
- An advanced data mining system should facilitate the discovery of knowledge at multiple levels of abstraction. Data mining systems can also be categorized as those that mine data regularities (commonly occurring patterns) versus those that mine data irregularities (such as exceptions, or outliers).
- ***In general, concept description, association and correlation analysis, classification, prediction, and clustering mine data regularities, rejecting outliers as noise. These methods may also help detect outliers.***
- ***Classification according to the kinds of techniques utilized: Data mining systems can be categorized according to the underlying data mining techniques employed. These techniques can be described according to the degree of user interaction involved (e.g., autonomous systems, interactive exploratory systems, query-driven systems) or the methods of data analysis employed (e.g., database-oriented or data warehouse– oriented techniques, machine learning, statistics, visualization, pattern recognition, neural networks, and so on).***

- A sophisticated data mining system will often adopt multiple data mining techniques or work out an effective, integrated technique that combines the merits of a few individual approaches.
- Classification according to the applications adapted: Data mining systems can also be categorized according to the applications they adapt. For example, data mining systems may be tailored specifically for finance, telecommunications, DNA, stock markets, e-mail, and so on. Different applications often require the integration of application-specific methods. Therefore, a generic, all-purpose data mining system may not fit domain-specific mining tasks.

## 1.7 Data-Mining Task Primitives

- Each user will have a data mining task in mind, that is, some form of data analysis that he or she would like to have performed. A data mining task can be specified in the form of a data mining query, which is input to the data mining system.
- A data mining query is defined in terms of data mining task primitives. These primitives allow the user to inter- actively communicate with the data mining system during discovery in order to direct the mining process, or examine the findings from different angles or depths. The data mining primitives specify the following, as illustrated in Figure 1.13.
- ***The set of task-relevant data to be mined: This specifies the portions of the database or the set of data in which the user is interested. This includes the database attributes or data warehouse dimensions of interest (referred to as the relevant attributes or dimensions).***
- ***The kind of knowledge to be mined: This specifies the data mining functions to be per- formed, such as characterization, discrimination, association or correlation analysis, classification, prediction, clustering, outlier analysis, or evolution analysis.***

- The background knowledge to be used in the discovery process: This knowledge about the domain to be mined is useful for guiding the knowledge discovery process and for evaluating the patterns found.
- ***Concept hierarchies are a popular form of back- ground knowledge, which allow data to be mined at multiple levels of abstraction.*** An example of a concept hierarchy for the attribute (or dimension) age is shown in Figure 1.14. User beliefs regarding relationships in the data are another form of background knowledge.
- ***The interestingness measures and thresholds for pattern evaluation: They may be used to guide the mining process or, after discovery, to evaluate the discovered patterns.*** Different kinds of knowledge may have different interestingness measures. For example, interestingness measures for association rules include support and confidence. Rules whose support and confidence values are below user-specified thresholds are considered uninteresting.
- The expected representation for visualizing the discovered patterns: This refers to the form in which discovered patterns are to be displayed, which may include rules, tables, charts, graphs, decision trees, and cubes.
- A data mining query language can be designed to incorporate these primitives, allowing users to flexibly interact with data mining systems. Having a data mining query language provides a foundation on which user-friendly graphical interfaces can be built.

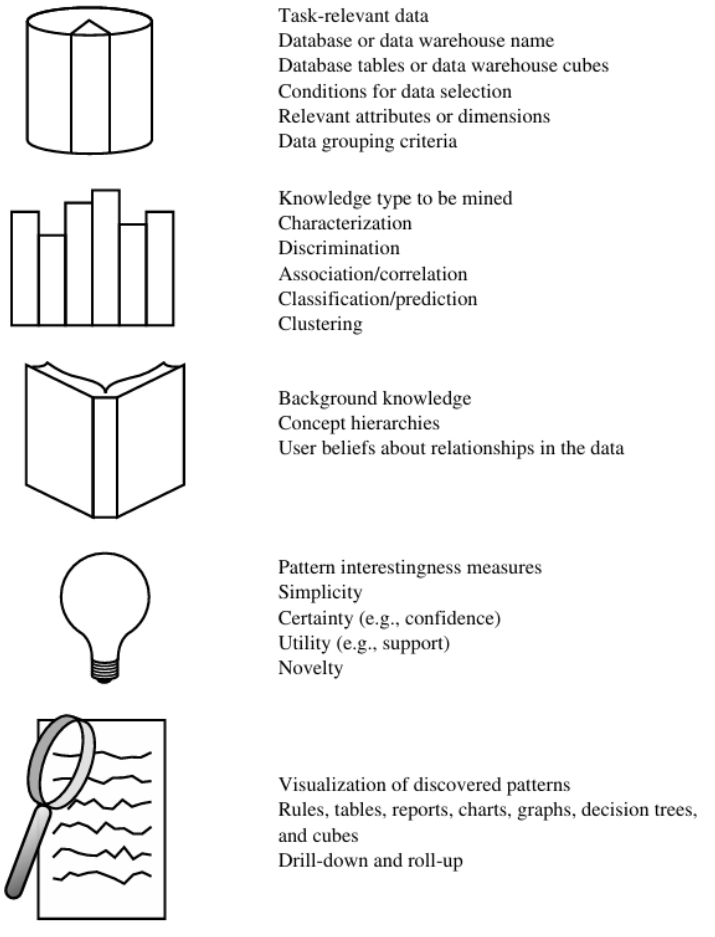



**Figure 1.13** Primitives for specifying a data mining task.

This facilitates a data mining system's communication with other information systems and its integration with the overall information processing environment.

Designing a comprehensive data mining language is challenging because data mining covers a wide spectrum of tasks, from data

characterization to evolution analysis. Each task has different requirements. The design of an effective data mining query language requires a deep understanding of the power, limitation, and underlying mechanisms of the various kinds of data mining tasks.

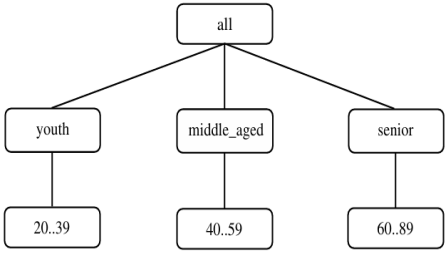



**Figure 1.14** A concept hierarchy for the attribute (or dimension) *age*. The root node represents the most general abstraction level, denoted as **all**.

- There are several proposals on data mining languages and standards. In this book, we use a data mining query language known as DMQL (Data Mining Query Language), which was designed as a teaching tool, based on the above primitives. Examples of its use to specify data mining queries appear throughout this book. The language adopts an SQL-like syntax, so that it can easily be integrated with the relational query language, SQL. Let's look at how it can be used to specify a data mining task.

**Example 1.11 Mining classification rules.**

- Suppose, as a marketing manager of AllElectronics, you would like to classify customers based on their buying patterns. You are especially interested in those customers whose salary is no less than $40,000, and who have bought more than $1,000 worth of items, each of which is priced at no less than $100. In particular, you are interested in the customer's age, income, the types of items purchased, the purchase location, and where the items were made. You would like to view the resulting classification in the form of rules.

This data mining query is expressed in DMQL3 as follows, where each line of the query has been enumerated to aid in our discussion.

```
(1) use database AllElectronics_db
(2) use hierarchy location_hierarchy for T.branch, age_hierarchy for C.age
(3) mine classification as promising_customers
(4) in relevance to C.age, C.income, I.type, I.place_made, T.branch
(5) from customer C, item I, transaction T
(6) where I.item_ID = T.item_ID and C.cust_ID = T.cust_ID
        and C.income ≥ 40,000 and I.price ≥ 100
(7) group by T.cust_ID

(8) having sum(I.price) ≥ 1,000
(9) display as rules
```

- The data mining query is parsed to form an SQL query that retrieves the set of task-relevant data specified by lines 1 and 4 to 8. That is, line 1 specifies the *All- Electronics* database, line 4 lists the relevant attributes (i.e., on which mining is to be performed) from the relations specified in line 5 for the conditions given in lines 6 and 7. Line 2 specifies that the concept hierarchies *location hierarchy* and *age hierarchy* be used as background knowledge to generalize branch locations and customer age values, respectively. Line 3 specifies that the kind of knowledge to be mined for this task is classification.
- Note that we want to generate a classification model for "promising customers" versus "non promising customers." In classification, often, an attribute may be specified as the **class label attribute**, whose values *explicitly* represent the classes. However, in this example, the two classes are *implicit*.
- That is, the specified data are retrieved and considered examples of "promising customers," whereas the remaining customers in the customer table are considered as "non-promising customers." Classification is performed based on this training set. Line 9 specifies that the mining results are to be displayed as a set of rules.
- There is no standard data mining query language as of yet; however, researchers and industry have been making good progress in this direction. Microsoft's OLE DB for Data Mining includes DMX, an XML-styled data mining language. Other standardization efforts include PMML (Programming

data Model Markup Language) and CRISP-DM (CRoss-Industry Standard Process for Data Mining).

- **Integration of a Data Mining System with a Database or Data Warehouse System :** A good system architecture will facilitate the data mining system to make best use of the software environment, accomplish data mining tasks in an efficient and timely manner, interoperate and exchange information with other information systems, be adaptable to users' diverse requirements, and evolve with time.

- ***A critical question in the design of a data mining (DM) system is how to integrate or couple the DM system with a database (DB) system and/or a data warehouse (DW) system. If a DM system works as a stand-alone system or is embedded in an application program, there are no DB or DW systems with which it has to communicate.***

- This simple scheme is called *no coupling*, where the main focus of the DM design rests on developing effective and efficient algorithms for mining the available data sets. However, when a DM system works in an environment that requires it to communicate with other information system components, such as DB and DW systems, possible integration schemes include no coupling, loose coupling, semitight coupling, and tight coupling. We examine each of these schemes, as follows:

**No coupling**:

- ***No coupling means that a DM system will not utilize any function of a DB or DW system. It may fetch data from a particular source (such as a file system), process data using some data mining algorithms, and then store the mining results in another file.***

Such a system, though simple, suffers from several drawbacks. First, a DB system provides a great deal of flexibility and efficiency at storing, organizing, accessing, and processing data. Without using a DB/DW system, a DM system

may spend a substantial amount of time finding, collecting, cleaning, and transforming data.

- In DB and/or DW systems, data tend to be well organized, indexed, cleaned, integrated, or consolidated, so that finding the task-relevant, high-quality data becomes an easy task. Second, there are many tested, scalable algorithms and data structures implemented in DB and DW systems. It is feasible to realize efficient, scalable implementations using such systems.
- Moreover, most data have been or will be stored in DB/DW systems. Without any coupling of such systems, a DM system will need to use other tools to extract data, making it difficult to integrate such a system into an information processing environment. Thus, no coupling represents a poor design.

**Loose coupling:**

- ***Loose coupling means that a DM system will use some facilities of a DB or DW system, fetching data from a data repository managed by these systems, performing data mining, and then storing the mining results either in a file or in a designated place in a database or data warehouse.***
- Loose coupling is better than no coupling because it can fetch any portion of data stored in databases or data warehouses by using query processing, indexing, and other system facilities. It incurs some advantages of the flexibility, efficiency, and other features provided by such systems. However, many loosely coupled mining systems are main memory-based. Because mining does not explore data structures and query optimization methods provided by DB or DW systems, it is difficult for loose coupling to achieve high scalability and good performance with large data sets.

**Semi-tight coupling**:

- ***Semi-tight coupling means that besides linking a DM system to a DB/DW system, efficient implementations of a few essential data mining primitives (identified by the analysis of frequently encountered data mining functions) can be provided in the DB/DW system. These primitives can include sorting, indexing, aggregation, histogram analysis, multiway join, and***

***precomputation of some essential statistical measures, such as sum, count, max, min, standard deviation, and so on.***

- Moreover, some frequently used intermediate mining results can be precomputed and stored in the DB/DW system. Because these intermediate mining results are either precomputed or can be computed efficiently, this design will enhance the performance of a DM system.

**Tight coupling:**

- Tight coupling means that a DM system is smoothly integrated into the DB/DW system. The data mining subsystem is treated as one functional component of an information system. Data mining queries and functions are optimized based on mining query analysis, data structures, indexing schemes, and query processing methods of a DB or DW system. With further technology advances, DM, DB, and DW systems will evolve and integrate together as one information system with multiple functionalities. This will provide a uniform information processing environment.

  This approach is highly desirable because it facilitates efficient implementations of data mining functions, high system performance, and an integrated information processing environment.

## 1.7 Major Issues in Data Mining

- Mining methodology and user interaction issues: These reflect the kinds of knowledge mined, the ability to mine knowledge at multiple granularities, the use of domain knowledge, ad hoc mining, and knowledge visualization.
- ***Mining different kinds of knowledge in databases:*** Because different users can be interested in different kinds of knowledge, data mining should cover a wide spectrum of data analysis and knowledge discovery tasks, including data characterization, discrimination, association and correlation analysis, classification, prediction, clustering, outlier analysis, and evolution analysis (which includes trend and similarity analysis). These tasks may use the same database in different ways and require the development of numerous data

mining techniques.

- ***Interactive mining of knowledge at multiple levels of abstraction:*** Because it is difficult to know exactly what can be discovered within a database, the data mining process should be interactive. For databases containing a huge amount of data, appropriate sampling techniques can first be applied to facilitate inter- active data exploration. Interactive mining allows users to focus the search for patterns, providing and refining data mining requests based on returned results. Specifically, knowledge should be mined by drilling down, rolling up, and pivoting through the data space and knowledge space interactively, similar to what OLAP can do on data cubes. In this way, the user can interact with the data mining system to view data and discovered patterns at multiple granularities and from different angles.
- ***Incorporation of background knowledge:*** Background knowledge, or information regarding the domain under study, may be used to guide the discovery process and allow discovered patterns to be expressed in concise terms and at different levels of abstraction. ***Domain knowledge related to databases, such as integrity constraints and deduction rules, can help focus and speed up a data mining process, or judge the interestingness of discovered patterns.***
- ***Data mining query languages and ad hoc data mining: Relational query languages (such as SQL) allow users to pose ad hoc queries for data retrieval. In a similar vein, high-level data mining query languages need to be developed to allow users to describe ad hoc data mining tasks by facilitating the specification of the relevant sets of data for analysis, the domain knowledge, the kinds of knowledge to be mined, and the conditions and constraints to be enforced on the discovered patterns.*** Such a language should be integrated with a database or data warehouse query language and optimized for efficient and flexible data mining.
- ***Presentation and visualization of data mining results: Discovered knowledge should be expressed in high-level languages, visual***

***representations, or other expressive forms so that the knowledge can be easily understood and directly usable by humans. This is especially crucial if the data mining system is to be interactive. This requires the system to adopt expressive knowledge representation techniques, such as trees, tables, rules, graphs, charts, crosstabs, matrices, or curves.***

- ***Handling noisy or incomplete data:*** The data stored in a database may reflect noise, exceptional cases, or incomplete data objects. When mining data regularities, these objects may confuse the process, causing the knowledge model constructed to overfit the data. As a result, the accuracy of the discovered patterns can be poor. Data cleaning methods and data analysis methods that can handle noise are required, as well as outlier mining methods for the discovery and analysis of exceptional cases.
- ***Pattern evaluation—the interestingness problem:*** A data mining system can uncover thousands of patterns. Many of the patterns discovered may be uninteresting to the given user, either because they represent common knowledge or lack novelty. Several challenges remain regarding the development of techniques to assess the interestingness of discovered patterns, particularly with regard to subjective measures that estimate the value of patterns with respect to a given user class, based on user beliefs or expectations. The use of interestingness measures or user-specified constraints to guide the discovery process and reduce the search space is another active area of research.

**Performance Issues:** These include efficiency, scalability, and parallelization of data mining algorithms.

- ***Efficiency and scalability of data mining algorithms:*** To effectively extract information from a huge amount of data in databases, data mining algorithms must be efficient and scalable. In other words, the running time of a data mining algorithm must be predictable and acceptable in large databases. From a database perspective on knowledge discovery, efficiency and scalability are key issues in the implementation of data mining systems. Many of the issues

discussed above under *mining methodology and user interaction* must also consider efficiency and scalability.

- ***Parallel, distributed, and incremental mining algorithms:*** The huge size of many databases, the wide distribution of data, and the computational complexity of some data mining methods are factors motivating the development of **parallel and distributed data mining algorithms**. Such algorithms divide the data into partitions, which are processed in parallel. The results from the partitions are then merged. Moreover, the high cost of some data mining processes promotes the need for **incremental** data mining algorithms that incorporate database updates with- out having to mine the entire data again "from scratch." Such algorithms perform knowledge modification incrementally to amend and strengthen what was previously discovered.

**Issues relating to the diversity of database types:**

- ***Handling of relational and complex types of data:*** Because relational databases and data warehouses are widely used, the development of efficient and effective data mining systems for such data is important. However, other databases may contain complex data objects, hypertext and multimedia data, spatial data, temporal data, or transaction data. It is unrealistic to expect one system to mine all kinds of data, given the diversity of data types and different goals of data mining. Specific data mining systems should be constructed for mining specific kinds of data. Therefore, one may expect to have different data mining systems for different kinds of data.

- ***Mining information from heterogeneous databases and global information systems:*** Local- and wide-area computer networks (such as the Internet) connect many sources of data, forming huge, distributed, and heterogeneous databases. The discovery of knowledge from different sources of structured, semi-structured, or unstructured data with diverse data semantics poses great challenges to data mining. Data mining may help disclose high-level data

regularities in multiple heterogeneous databases that are unlikely to be discovered by simple query systems and may improve information exchange and interoperability in heterogeneous databases. Web mining, which uncovers interesting knowledge about Web contents, Web structures, Web usage, and Web dynamics, becomes a very challenging and fast-evolving field in data mining.

- Today's real-world databases are highly susceptible to noisy, missing, and inconsistent data due to their typically huge size (often several gigabytes or more) and their likely origin from multiple, heterogenous sources. Low-quality data will lead to low-quality mining results.
- "How can the data be preprocessed in order to help improve the quality of the data and, consequently, of the mining results? How can the data be preprocessed so as to improve the efficiency and ease of the mining process?"
- There are a number of data preprocessing techniques. Data cleaning can be applied to remove noise and correct inconsistencies in the data. Data integration merges data from multiple sources into a coherent data store, such as a data warehouse.
- Data transformations, such as normalization, may be applied. For example, normalization may improve the accuracy and efficiency of mining algorithms involving distance measurements.
- Data reduction can reduce the data size by aggregating, eliminating redundant features, or clustering, for instance. These techniques are not mutually exclusive; they may work together. For example, data cleaning can involve transformations to correct wrong data, such as by transforming all entries for a date field to a common format.
- Data processing techniques, when applied before mining, can substantially improve the overall quality of the patterns mined and/or the time required for the actual mining.
- ***Descriptive data summarization, which serves as a foundation for data preprocessing. Descriptive data summarization helps us study the general***

***characteristics of the data and identify the presence of noise or outliers, which is useful for successful data cleaning and data integration. The methods for data preprocessing are organized into the following categories: data cleaning , data integration and transformation , and data reduction .***

- ***Concept hierarchies can be used in an alternative form of data reduction where we replace low-level data (such as raw values for age) with higher-level concepts (such as youth, middle-aged, or senior).***

**Why Preprocess the Data?**

- Imagine that you are a manager at AllElectronics and have been charged with analyzing the company's data with respect to the sales at your branch. You immediately set out to perform this task.
- You carefully inspect the company's database and data warehouse, identifying and selecting the attributes or dimensions to be included in your analysis, such as item, price, and units sold. Alas! You notice that several of the attributes for various tuples have no recorded value. For your analysis, you would like to include information as to whether each item purchased was advertised as on sale, yet you discover that this information has not been recorded.
- Furthermore***, users of your database system have reported errors, unusual values, and inconsistencies in the data recorded for some transactions. In other words, the data you wish to analyze by data mining techniques are incomplete (lacking attribute values or certain attributes of interest, or containing only aggregate data), noisy (containing errors, or outlier values that deviate from the expected), and inconsistent (e.g., containing discrepancies in the department codes used to categorize items). Welcome to the real world! Incomplete, noisy, and inconsistent data are commonplace properties of large real- world databases and data warehouses. Incomplete data can occur for a number of rea- sons.***
- Attributes of interest may not always be available, such as customer

information for sales transaction data. Other data may not be included simply because it was not considered important at the time of entry. Relevant data may not be recorded due to a misunderstanding, or because of equipment malfunctions.

- Data that were inconsistent with other recorded data may have been deleted. Furthermore, the recording of the his- tory or modifications to the data may have been overlooked. Missing data, particularly for tuples with missing values for some attributes, may need to be inferred.
- There are many possible reasons for noisy data (having incorrect attribute values). The data collection instruments used may be faulty. There may have been human or computer errors occurring at data entry.
- Errors in data transmission can also occur. There may be technology limitations, such as limited buffer size for coordinating synchronized data transfer and consumption. Incorrect data may also result from inconsistencies in naming conventions or data codes used, or inconsistent formats for input fields, such as date. Duplicate tuples also require data cleaning.
- ***Data cleaning routines work to "clean" the data by filling in missing values, smoothing noisy data, identifying or removing outliers, and resolving inconsistencies. If users believe the data are dirty, they are unlikely to trust the results of any data mining that has been applied to it.***
- Furthermore, dirty data can cause confusion for the mining procedure, resulting in unreliable output. Although most mining routines have some procedures for dealing with incomplete or noisy data, they are not always robust.
- Instead, they may concentrate on avoiding overfitting the data to the function being modeled. Therefore, a useful preprocessing step is to run your data through some data cleaning routines.
- Getting back to your task at AllElectronics, suppose that you would like to include data from multiple sources in your analysis. This would involve

integrating multiple databases, data cubes, or files, that is, data integration. Yet some attributes representing a given concept may have different names in different databases, causing inconsistencies and redundancies.

- For example, the attribute for customer identification may be referred to as customer id in one data store and cust id in another. Naming inconsistencies may also occur for attribute values. For example, the same first name could be registered as "Bill" in one database, but "William" in another, and "B." in the third. Furthermore, you suspect that some attributes may be inferred from others (e.g., annual revenue).
- Having a large amount of redundant data may slow down or confuse the knowledge discovery process. Clearly, in addition to data cleaning, steps must be taken to help avoid redundancies during data integration. Typically, data cleaning and data integration are performed as a preprocessing step when preparing the data for a data warehouse. Additional data cleaning can be performed to detect and remove redundancies that may have resulted from data integration.
- Getting back to your data, you have decided, say, that you would like to use a distance- based mining algorithm for your analysis, such as neural networks, nearest-neighbor classifiers, or clustering.1 Such methods provide better results if the data to be analyzed have been normalized, that is, scaled to a specific range such as [0.0, 1.0].
- Your customer data, for example, contain the attributes age and annual salary. The annual salary attribute usually takes much larger values than age. Therefore, if the attributes are left unnormalized, the distance measurements taken on annual salary will generally out- weigh distance measurements taken on age. Furthermore, it would be useful for your analysis to obtain aggregate information as to the sales per customer region—something that is not part of any precomputed data cube in your data warehouse.
- You soon realize that data transformation operations, such as normalization and aggregation, are additional data preprocessing procedures that would

contribute toward the success of the mining process.

- "Hmmm," you wonder, as you consider your data even further. "The data set I have selected for analysis is HUGE, which is sure to slow down the mining process. Is there any way I can reduce the size of my data set, without jeopardizing the data mining results?" Data reduction obtains a reduced representation of the data set that is much smaller in volume, yet produces the same (or almost the same) analytical results. There are a number of strategies for data reduction.
- These include data aggregation (e.g., building a data cube), attribute subset selection (e.g., removing irrelevant attributes through correlation analysis), dimensionality reduction (e.g., using encoding schemes such as minimum length encoding or wavelets), and numerosity reduction (e.g., "replacing" the data by alternative, smaller representations such as clusters or parametric models).
- Data can also be "reduced" by generalization with the use of concept hierarchies, where low-level concepts, such as city for customer location, are replaced with higher-level concepts, such as region or province or state.
- A concept hierarchy organizes the concepts into varying levels of abstraction. Data discretization is

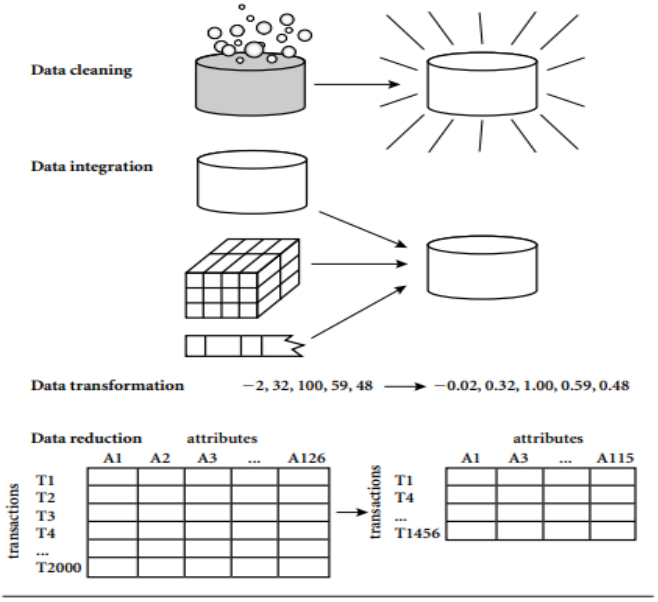



**Figure 2.1** Forms of data preprocessing.

form of data reduction that is very useful for the automatic generation of concept hierarchies from numerical data.

- Figure 2.1 summarizes the data preprocessing steps described here. Note that the above categorization is not mutually exclusive. For example, the removal of redundant data may be seen as a form of data cleaning, as well as data reduction.
- In summary, real-world data tend to be dirty, incomplete, and inconsistent. Data preprocessing techniques can improve the quality of the data, thereby helping to improve the accuracy and efficiency of the subsequent mining process. Data preprocessing is an important step in the knowledge discovery process, because quality decisions must be based on quality data. Detecting data anomalies, rectifying them early, and reducing the data to be analyzed can lead to huge payoffs for decision making.

**Descriptive Data Summarization**

- For data preprocessing to be successful, it is essential to have an overall picture of your data. ***Descriptive data summarization techniques can be used to identify the typical properties of your data and highlight which data values should be treated as noise or outliers.***
- For many data preprocessing tasks, users would like to learn about data characteristics regarding both central tendency and dispersion of the data. Measures of central tendency include mean, median, mode, and midrange, while measures of data dispersion include quartiles, interquartile range (IQR), and variance. These descriptive statistics are of great help in understanding the distribution of the data. Such measures have been studied extensively in the statistical literature.
- ***From the data mining point of view, we need to examine how they can be computed efficiently in large databases.*** In particular, it is necessary to introduce the notions of distributive measure, algebraic measure, and holistic measure. Knowing what kind of measure, we are dealing with can help us choose an efficient implementation for it.
- **Today's real-world databases** are highly susceptible to noisy, missing, and inconsistent data due to their typically huge size (often several gigabytes or more) and their likely origin from multiple, heterogenous sources. Low-quality data will lead to low-quality mining results.
- *"**How can the data be preprocessed in order to help improve the quality of the data and, consequently, of the mining results? How can the data be preprocessed so as to improve the efficiency and ease of the mining process?"***
- There are a number of data preprocessing techniques. *Data cleaning* can be applied to remove noise and correct inconsistencies in the data. *Data integration* merges data from multiple sources into a coherent data store, such as a data warehouse.
- *Data transformations*, such as normalization, may be applied. For example,

normalization may improve the accuracy and efficiency of mining algorithms involving distance measurements.

- **Data reduction can reduce the data size by aggregating, eliminating redundant features, or clustering, for instance.** These techniques are not mutually exclusive; they may work together. For example, data cleaning can involve transformations to correct wrong data, such as by transforming all entries for a *date* field to a common format.
- Data processing techniques, when applied before mining, can substantially improve the overall quality of the patterns mined and/or the time required for the actual mining.
- *Descriptive data summarization*, which serves as a foundation for data preprocessing. Descriptive data summarization helps us study the general characteristics of the data and identify the presence of noise or outliers, which is useful for successful data cleaning and data integration.
- The methods for data preprocessing are organized into the following categories: *data cleaning* , *data integration and transformation* , and *data reduction* .
- ***Concept hierarchies can be used in an alternative form of data reduction where we replace low-level data (such as raw values for age) with higher-level concepts (such as youth, middle-aged, or senior).***

### 2.1.1 Measuring the Central Tendency

In this section, we look at various ways to measure the central tendency of data. The

most common and most effective numerical measure of the "center" of a set of data is the *(arithmetic) mean*. Let $x_1, x_2, \ldots, x_N$ be a set of $N$ values or observations, such as for some attribute, like *salary*. The **mean** of this set of values is

$$\bar{x} = \frac{\sum_{i=1}^{N} x_i}{N} = \frac{x_1 + x_2 + \cdots + x_N}{N}.$$

- This corresponds to the built-in aggregate function, *average* (avg() in SQL), provided in relational database systems.
- A **distributive measure** is a measure (i.e., function) that can be computed for a given data set by partitioning the data into smaller subsets, computing the measure for each subset, and then merging the results in order to arrive at the measure's value for the original (entire) data set.
- Both sum() and count() are distributive measures because they can be computed in this manner. Other examples include max() and min(). An **algebraic measure** is a measure that can be computed by applying an algebraic function to one or more distributive measures.

$$\bar{x} = \frac{\sum_{i=1}^{N} w_i x_i}{\sum_{i=1}^{N} w_i} = \frac{w_1 x_1 + w_2 x_2 + \cdots + w_N x_N}{w_1 + w_2 + \cdots + w_N}. \qquad (2.2)$$

Hence, *average* (or mean()) is an algebraic measure because it can be computed by sum()/count(). When computing data cubes[2], sum() and count() are typically saved in precomputation. Thus, the derivation of *average* for data cubes is straightforward.

Sometimes, each value $x_i$ in a set may be associated with a weight $w_i$, for $i = 1, \ldots, N$.

The weights reflect the significance, importance, or occurrence frequency attached to their respective values.

- This is called the **weighted arithmetic mean** or the **weighted average**. Note that the weighted average is another example of an algebraic measure.
- Although the mean is the single most useful quantity for describing a data set, it is not always the best way of measuring the center of the data. A major

problem with the *mean* is its sensitivity to extreme (e.g., outlier) values. Even a small number of extreme values can corrupt the mean.

- For example, the mean salary at a company may be substantially pushed up by that of a few highly paid managers. Similarly, the average score of a class in an exam could be pulled down quite a bit by a few very low scores.
- To offset the effect caused by a small number of extreme values, we can instead use the **trimmed mean**, which is the mean obtained after chopping off values at the high and low extremes. For example, we can sort the values observed for *salary* and remove the top and bottom 2% before computing the mean. We should avoid trimming too large a portion (such as 20%) at both ends as this can result in the loss of valuable information.
- ***For skewed (asymmetric) data, a better measure of the center of data is the median.*** Suppose that a given data set of $N$ distinct values is sorted in numerical order. If $N$ is odd, then the **median** is the *middle value* of the ordered set; otherwise (i.e., if $N$ is even), the median is the average of the middle two values.
- A **holistic measure** is a measure that must be computed on the entire data set as a whole. It cannot be computed by partitioning the given data into subsets and merging the values obtained for the measure in each subset. The median is an example of a holistic measure. Holistic measures are much more expensive to compute than distributive measures such as those listed above.
- We can, however, easily *approximate* the median value of a dataset. Assume that data are grouped in intervals according to their $x_i$ data values and that the frequency (i.e., number of data values) of each interval is known. For example, people may be grouped according to their annual salary in intervals such as 10–20K, 20–30K, and so on. Let the interval that contains the median frequency be the *median interval*. We can approximate the median of the entire data set (e.g., the median salary) by

  interpolation using the formula:

$$median = L_1 + \left( \frac{N/2 - (\sum freq)_l}{freq_{median}} \right) width, \qquad (2.3)$$

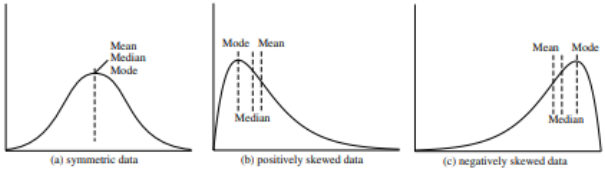



**Figure 2.2** Mean, median, and mode of symmetric versus positively and negatively skewed data.

where $L_1$ is the lower boundary of the median interval, $N$ is the number of values in the entire data set, ($\sum freq)_l$ is the sum of the frequencies of all of the intervals that are lower than the median interval, $freq_{median}$ is the frequency of the median interval, and *width* is the width of the median interval.

Another measure of central tendency is the *mode*. The **mode** for a set of data is the value that occurs most frequently in the set. It is possible for the greatest frequency to correspond to several different values, which results in more than one mode. Data sets with one, two, or three modes are respectively called **unimodal**, **bimodal**, and **trimodal**. In general, a data set with two or more modes is **multimodal**. At the other extreme, if each data value occurs only once, then there is no mode.

For unimodal frequency curves that are moderately skewed (asymmetrical), we have the following empirical relation:

$$\text{mean} - \text{mode} = 3 \times (\text{mean} - \text{median}). \qquad (2.4)$$

This implies that the mode for unimodal frequency curves that are moderately skewed can easily be computed if the mean and median values are known.

In a unimodal frequency curve with perfect symmetric data distribution, the mean, median, and mode are all at the same center value, as shown in Figure 2.2(a). However, data in most real applications are not symmetric. They may instead be either positively skewed, where the mode occurs at a

value that is smaller than the median (Figure 2.2(b)), or negatively skewed, where the mode occurs at a value greater than the median (Figure 2.2(c)).

The **midrange** can also be used to assess the central tendency of a data set. It is the average of the largest and smallest values in the set. This algebraic measure is easy to compute using the SQL aggregate functions, max() and min().

**Data Cleaning**

Real-world data tend to be incomplete, noisy, and inconsistent. *Data cleaning* (or *data cleansing*) routines attempt to fill in missing values, smooth out noise while identify- ing outliers, and correct inconsistencies in the data. In this section, you will study basic methods for data cleaning.

### 2.1.2 Missing Values

Imagine that you need to analyze *AllElectronics* sales and customer data. You note that many tuples have no recorded value for several attributes, such as customer *income*. How can you go about filling in the missing values for this attribute? Let's look at the following methods:

1. **Ignore the tuple**: This is usually done when the class label is missing (assuming the mining task involves classification). This method is not very effective, unless the tuple contains several attributes with missing values. It is especially poor when the percent- age of missing values per attribute varies considerably.
2. **Fill in the missing value manually**: In general, this approach is time-consuming and may not be feasible given a large data set with many missing values.
3. **Use a global constant to fill in the missing value**: Replace all missing attribute values by the same constant, such as a label like *"Unknown"* or □. If missing values are replaced by, say, *"Unknown,"* then the mining program may

mistakenly think that they form an interesting concept, since they all have a value in common—that of *"Unknown."* Hence, although this method is simple, it is not foolproof.

4. **Use the attribute mean to fill in the missing value**: For example, suppose that the average income of *AllElectronics* customers is $56,000. Use this value to replace the missing value for *income*.
5. **Use the attribute mean for all samples belonging to the same class as the given tuple**: For example, if classifying customers according to *credit risk*, replace the missing value with the average *income* value for customers in the same credit risk category as that of the given tuple.
6. **Use the most probable value to fill in the missing value**: This may be determined with regression, inference-based tools using a Bayesian formalism, or decision tree induction. For example, using the other customer attributes in your data set, you may construct a decision tree to predict the missing values for *income*.

Methods 3 to 6 bias the data. The filled-in value may not be correct. Method 6, however, is a popular strategy. In comparison to the other methods, it uses the most information from the present data to predict missing values. By considering the values of the other attributes in its estimation of the missing value for *income*, there is a greater chance that the relationships between *income* and the other attributes are preserved.

- It is important to note that, in some cases, a missing value may not imply an error in the data! For example, when applying for a credit card, candidates may be asked to supply their driver's license number. Candidates who do not have a driver's license may naturally leave this field blank. Forms should allow respondents to specify values such as "not applicable".
- Software routines may also be used to uncover other null values, such as "don't know", "?", or "none". Ideally, each attribute should have one or more rules regarding the *null* condition. The rules may specify whether or not nulls

are allowed, and/or how such values should be handled or transformed. Fields may also be inten- tionally left blank if they are to be provided in a later step of the business process. Hence, although we can try our best to clean the data after it is seized, good design of databases and of data entry procedures should help minimize the number of missing values or errors in the first place.

### 2.1.3 Noisy Data

***"What is noise?" Noise is a random error or variance in a measured variable. Given a numerical attribute such as, say, price, how can we "smooth" out the data to remove the noise?*** **Let's look at the following data smoothing techniques:**

**Sorted data for *price* (in dollars)**: 4, 8, 15, 21, 21, 24, 25, 28, 34

**Partition into (equal-frequency) bins**: Bin 1: 4, 8, 15

Bin 2: 21, 21, 24
Bin 3: 25, 28, 34

Smoothing by bin means:

Bin 1: 9, 9, 9
Bin 2: 22, 22, 22
Bin 3: 29, 29, 29

Smoothing by bin boundaries:

Bin 1: 4, 4, 15
Bin 2: 21, 21, 24
Bin 3: 25, 25, 34

**Figure 2.11** Binning methods for data smoothing.

1. **Binning:** Binning methods smooth a sorted data value by consulting its "neighbor- hood," that is, the values around it. The sorted values are distributed into a number of "buckets," or *bins*. Because binning methods consult the neighborhood of values, they perform *local* smoothing. Figure 2.11 illustrates some binning techniques. In this example, the data for *price* are first sorted and then partitioned into *equal-frequency* bins of size 3 (i.e., each bin contains three values). In **smoothing by bin means**, each value in a bin is replaced by the mean value of the bin. For example, the mean of the values 4, 8, and 15 in Bin 1 is 9. Therefore, each original value in this bin is replaced by the value 9. Similarly, **smoothing by bin medians** can be employed, in which each bin value is replaced by the bin median. In **smoothing by bin boundaries**, the mini- mum and maximum values in a given bin are identified as the *bin boundaries*. Each bin value is then replaced by the closest boundary value. In general, the larger the width, the greater the effect of the smoothing. Alternatively, bins may be *equal-width*, where the interval range of values in each bin is constant. Binning is also used as a discretization technique and is further discussed in Section 2.6.

2. **Regression:** ***Data can be smoothed by fitting the data to a function, such as with regression. Linear regression involves finding the "best" line to fit two attributes (or variables), so that one attribute can be used to predict the other.***

3. ***Multiple linear regression is an extension of linear regression, where more than two attributes are involved and the data are fit to a multidimensional surface.***

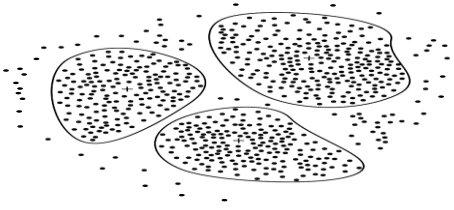


**Figure 2.12** A 2-D plot of customer data with respect to customer locations in a city, showing three data clusters. Each cluster centroid is marked with a "+", representing the average point in space for that cluster. Outliers may be detected as values that fall outside of the sets of clusters.

4. **Clustering:** ***Outliers may be detected by clustering, where similar values are organized into groups, or "clusters."*** Intuitively, values that fall outside of the set of clusters may be considered outliers (Figure 2.12). Chapter 7 is dedicated to the topic of clustering and outlier analysis. Many methods for data smoothing are also methods for data reduction involving discretization. For example, the binning techniques described above reduce the number of distinct values per attribute. This acts as a form of data reduction for logic-based data mining methods, such as decision tree induction, which repeatedly make value comparisons on sorted data.
5. ***Concept hierarchies are a form of data discretization that can also be used for data smoothing. A concept hierarchy for price, for example, may map real price values into inexpensive, moderately priced, and expensive, thereby reducing the number of data values to be handled by the mining process.***

**Data Integration and Transformation**

Data mining often requires data integration—the merging of data from multiple data stores. The data may also need to be transformed into forms appropriate for mining. This section describes both data integration and data transformation.

### 2.1.4 Data Integration

- It is likely that your data analysis task will involve *data integration*, which combines data from multiple sources into a coherent data store, as in data warehousing. These sources may include multiple

databases, data cubes, or flat files.

- There are a number of issues to consider during data integration. *Schema integration* and *object matching* can be tricky. How can equivalent real-world entities from multiple data sources be matched up? This is referred to as the **entity identification problem**.
- For example, how can the data analyst or the computer be sure that *customer id* in one database and *cust number* in another refer to the same attribute? Examples of metadata for each attribute include the name, meaning, data type, and range of values permitted for the attribute, and null rules for handling blank, zero, or null values (Section 2.3). Such metadata can be used to help avoid errors in schema integration.
- The metadata may also be used to help transform the data (e.g., where data codes for *pay type* in one database may be *"H"* and *"S"*, and *1* and *2* in another). Hence, this step also relates to data cleaning, as described earlier.

***Redundancy is another important issue. An attribute (such as annual revenue, for instance) may be redundant if it can be "derived" from another attribute or set of attributes. Inconsistencies in attribute or dimension naming can also cause redundancies in the resulting data set.***

***Some redundancies can be detected by correlation analysis. Given two attributes, such analysis can measure how strongly one attribute implies the other, based on the available data. For numerical attributes, we can evaluate the correlation between two attributes, A and B, by computing the correlation coefficient (also known as Pearson's product moment coefficient, named after its inventer, Karl Pearson). This is***

$$r_{A,B} = \frac{\sum_{i=1}^{N}(a_i - \bar{A})(b_i - \bar{B})}{N\sigma_A\sigma_B} = \frac{\sum_{i=1}^{N}(a_i b_i) - N\bar{A}\bar{B}}{N\sigma_A\sigma_B}, \qquad (2.8)$$

where $N$ is the number of tuples, $a_i$ and $b_i$ are the respective values of $A$ and $B$ in tuple $i$, $\bar{A}$ and $\bar{B}$ are the respective mean values of $A$ and $B$, $\sigma_A$ and $\sigma_B$ are the respective standard deviations of $A$ and $B$ (as defined in Section 2.2.2), and $\Sigma(a_i b_i)$ is the sum of the $AB$ cross-product (that is, for each tuple, the value for $A$ is multiplied by the value for $B$ in that tuple). Note that $-1 \leq r_{A,B} \leq +1$. If $r_{A,B}$ is greater than 0, then $A$ and $B$ are positively correlated, meaning that the values of $A$ increase as the values of $B$ increase. The higher the value, the stronger the correlation (i.e., the more each attribute implies the other). Hence, a higher value may indicate that $A$ (or $B$) may be removed as a redundancy. If the resulting value is equal to 0, then $A$ and $B$ are independent and there is no correlation between them. If the resulting value is less than 0, then $A$ and $B$ are negatively correlated, where the values of one attribute increase as the values of the other attribute decrease. This means that each attribute discourages the other. Scatter plots can also be used to view correlations between attributes (Section 2.2.3).

### 2.1.5 Data Transformation

In *data transformation*, the data are transformed or consolidated into forms appropriate for mining. Data transformation can involve the following:

**Smoothing**, which works to remove noise from the data. Such techniques include binning, regression, and clustering.

**Aggregation**, where summary or aggregation operations are applied to the data. For example, the daily sales data may be aggregated so as to compute monthly and annual total amounts. This step is typically used in constructing a data cube for analysis of the data at multiple granularities.

**Generalization** of the data, where low-level or "primitive" (raw) data are replaced by higher-level concepts through the use of concept hierarchies. For example, categorical

attributes, like *street*, can be generalized to higher-level concepts, like *city* or *country*. Similarly, values for numerical attributes, like *age*, may be mapped to higher-level concepts, like *youth*, *middle-aged*, and *senior*.

- **Normalization**, where the attribute data are scaled so as to fall within a small specified range, such as —1.0 to 1.0, or 0.0 to 1.0.
- **Attribute construction** (or *feature construction*), where new attributes are constructed and added from the given set of attributes to help the mining process.

- An attribute is normalized by scaling its values so that they fall within a small specified range, such as 0.0 to 1.0. Normalization is particularly useful for classification algorithms involving neural networks, or distance measurements such as nearest-neighbor classification and clustering.
- If using the neural network backpropagation algorithm for classification mining , normalizing the input values for each attribute measured in the training tuples will help speed up the learning phase.
- For distance-based methods, normalization helps prevent attributes with initially large ranges (e.g., *income*) from out- weighing attributes with initially smaller ranges (e.g., binary attributes). There are many methods for data normalization. We study three: *min-max normalization, z-score normalization*, and *normalization by decimal scaling.*
- ***In attribute construction,[5] new attributes are constructed from the given attributes and added in order to help improve the accuracy and understanding of structure in high-dimensional data.***
- For example, we may wish to add the attribute *area* based on the attributes *height* and *width*. By combining attributes, attribute construction can discover missing information about the relationships between data attributes that can be useful for knowledge discovery.

**Data Reduction**

Imagine that you have selected data from the *AllElectronics* data warehouse for analysis. The data set will likely be huge! Complex data analysis and mining on huge amounts of data can take a long time, making such analysis

impractical or infeasible. [5]In the machine learning literature, attribute construction is known as *feature construction*.

**Data reduction** techniques can be applied to obtain a reduced representation of the data set that is much smaller in volume, yet closely maintains the integrity of the original data. That is, mining on the reduced data set should be more efficient yet produce the same (or almost the same) analytical results.

Strategies for data reduction include the following:

1. **Data cube aggregation**, where aggregation operations are applied to the data in the construction of a data cube.
2. **Attribute subset selection**, where irrelevant, weakly relevant, or redundant attributes or dimensions may be detected and removed.
3. **Dimensionality reduction**, where encoding mechanisms are used to reduce the data set size.
4. **Numerosity reduction**, where the data are replaced or estimated by alternative, smaller data representations such as parametric models (which need store only the model parameters instead of the actual data) or nonparametric methods such as clustering, sampling, and the use of histograms.
5. **Discretization and concept hierarchy generation**, where raw data values for attributes are replaced by ranges or higher conceptual levels. Data discretization is a form of numerosity reduction that is very useful for the automatic generation of concept hier- archies. Discretization and concept hierarchy generation are powerful tools for data mining, in that they allow the mining of data at multiple levels of abstraction. We therefore defer the discussion of discretization and concept hierarchy generation to Section 2.6, which is devoted entirely to this topic.

Strategies 1 to 4 above are discussed in the remainder of this section. The

computational time spent on data reduction should not outweigh or "erase" the time saved by mining on a reduced data set size.

### 2.1.6 Data Cube Aggregation

- Imagine that you have collected the data for your analysis. These data consist of the *AllElectronics* sales per quarter, for the years 2002 to 2004. You are, however, interested in the annual sales (total per year), rather than the total per quarter. Thus the data can be *aggregated* so that the resulting data summarize the total sales per year instead of per quarter. This aggregation is illustrated in Figure 2.13. The resulting data set is smaller in volume, without loss of information necessary for the analysis task.
- Data cubes store multidimensional aggregated information. For example, Figure 2.14 shows a data cube for multidimensional analysis of sales data with respect to annual sales per item type for each *AllElectronics* branch. Each cell holds an aggregate data value, corresponding to the data point in multidimensional space.
- Concept hierarchies may exist for each attribute, allowing the analysis of data at multiple levels of abstraction. For example, a hierarchy for *branch* could allow branches to be grouped into regions, based on their address. Data cubes provide fast access to precomputed, summarized data, thereby benefiting on-line analytical processing as well as data mining.

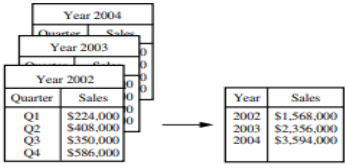



**Figure 2.13** Sales data for a given branch of *AllElectronics* for the years 2002 to 2004. On the left, the sales are shown per quarter. On the right, the data are aggregated to provide the annual sales.

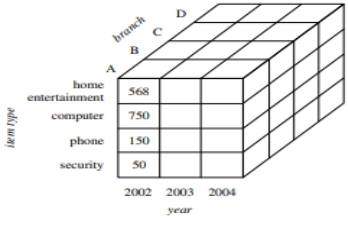



**Figure 2.14** A data cube for sales at *AllElectronics*.

**Figure 2.14** A data cube for sales at *AllElectronics*

- The cube created at the lowest level of abstraction is referred to as the *base cuboid.* The base cuboid should correspond to an individual entity of interest, such as *sales* or *customer*. In other words, the lowest level should be usable, or useful for the analysis.
- A cube at the highest level of abstraction is the *apex cuboid.* For the sales data of Figure 2.14, the apex cuboid would give one total—the total *sales* for all three years, for all item types, and for all branches. Data cubes created for varying levels of abstraction are often referred to as *cuboids*, so that a data cube may instead refer to a *lattice of cuboids*. Each higher level of abstraction further reduces the resulting data size. When replying to data mining requests, the *smallest* available cuboid

relevant to the given task should be used.

### 2.1.7 Attribute Subset Selection

- Data sets for analysis may contain hundreds of attributes, many of which may be irrelevant to the mining task or redundant. For example, if the task is to classify customers as to whether or not they are likely to purchase a popular new CD at *AllElectronics* when notified of a sale, attributes such as the customer's telephone num- ber are likely to be irrelevant, unlike attributes such as *age* or *music taste*.

- Although it may be possible for a domain expert to pick out some of the useful attributes, this can be a difficult and time-consuming task, especially when the behavior of the data is not well known (hence, a reason behind its analysis!). Leaving out relevant attributes or keeping irrelevant attributes may be detrimental, causing confusion for the mining algorithm employed. This can result in discovered patterns of poor quality. In addition, the added volume of irrelevant or redundant attributes can slow down the mining process.

***Attribute subset selection[6] reduces the data set size by removing irrelevant or redundant attributes (or dimensions). The goal of attribute subset selection is to find a minimum set of attributes such that the resulting probability distribution of the data classes is as close as possible to the original distribution obtained using all attributes.***

- Mining on a reduced set of attributes has an additional benefit. It reduces the number of attributes appearing in the discovered patterns, helping to make the patterns easier to understand.

*"**How can we find a 'good' subset of the original attributes?" For n attributes, there are $2^n$ possible subsets. An exhaustive search for the optimal subset of attributes can be pro- hibitively expensive, especially as***

***n and the number of data classes increase. Therefore, heuristic methods that explore a reduced search space are commonly used for attribute subset selection. These methods are typically greedy in that, while searching through attribute space, they always make what looks to be the best choice at the time. Their strategy is to make a locally optimal choice in the hope that this will lead to a globally optimal solution. Such greedy methods are effective in practice and may come close to estimating an optimal solution.***

***The "best" (and "worst") attributes are typically determined using tests of statistical significance, which assume that the attributes are independent of one another. Many***

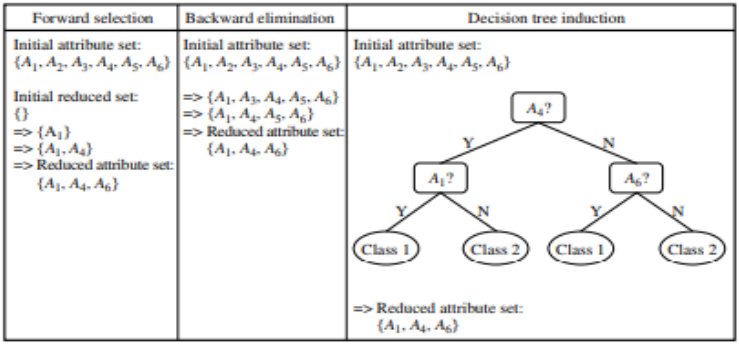



**Figure 2.15** Greedy (heuristic) methods for attribute subset selection.

other attribute evaluation measures can be used, such as the *information gain* measure used in building decision trees for classification.[7]

Basic heuristic methods of attribute subset selection include the following techniques, some of which are illustrated in Figure 2.15.

1. **Stepwise forward selection**: The procedure starts with an empty set of

attributes as the reduced set. The best of the original attributes is determined and added to the reduced set. At each subsequent iteration or step, the best of the remaining original attributes is added to the set.

2. **Stepwise backward elimination**: The procedure starts with the full set of attributes. At each step, it removes the worst attribute remaining in the set.
3. **Combination of forward selection and backward elimination**: The stepwise forward selection and backward elimination methods can be combined so that, at each step, the procedure selects the best attribute and removes the worst from among the remain- ing attributes.
4. **Decision tree induction**: Decision tree algorithms, such as ID3, C4.5, and CART, were originally intended for classification. Decision tree induction constructs a flowchart- like structure where each internal (nonleaf) node denotes a test on an attribute, each branch corresponds to an outcome of the test, and each external (leaf) node denotes a class prediction. At each node, the algorithm chooses the "best" attribute to partition the data into individual classes.

   When decision tree induction is used for attribute subset selection, a tree is constructed from the given data. All attributes that do not appear in the tree are assumed to be irrelevant. The set of attributes appearing in the tree form the reduced subset of attributes.

The stopping criteria for the methods may vary. The procedure may employ a thresh- old on the measure used to determine when to stop the attribute selection process.

**Regression and Log-Linear Models**

Regression and log-linear models can be used to approximate the given data. In (simple) **linear regression**, the data are modeled to fit a straight line.

For example, a random vari- able, *y* (called a *response variable*), can be modeled as a linear function of another random variable, *x* (called a *predictor variable*), with the equation

$$y = wx+b, \quad (2.14)$$

where the variance of *y* is assumed to be constant. In the context of data mining, *x* and *y* are numerical database attributes. The coefficients, *w* and *b* (called *regression coefficients*), specify the slope of the line and the *Y* - intercept, respectively. These coefficients can be solved for by the *method of least squares*, which minimizes the error between the actual line separating the data and the estimate of the line. **Multiple linear regression** is an extension of (simple) linear regression, which allows a response variable, *y*, to be modeled as a linear function of two or more predictor variables.

***Log-linear models approximate discrete multidimensional probability distribu- tions. Given a set of tuples in n dimensions (e.g., described by n attributes), we can consider each tuple as a point in an n-dimensional space. Log-linear models can be used to estimate the probability of each point in a multidimensional space for a set of discretized attributes, based on a smaller subset of dimensional combinations. This allows a higher-dimensional data space to be constructed from lower- dimensional spaces. Log-linear models are therefore also useful for dimensionality reduction (since the lower-dimensional points together typically occupy less space than the original data points) and data smoothing (since aggregate estimates in the lower-dimensional space are less subject to sampling variations than the estimates in the higher-dimensional space).***

Regression and log-linear models can both be used on sparse data, although their application may be limited. While both methods can handle

skewed data, regression does exceptionally well. Regression can be computationally intensive when applied to high- dimensional data, whereas log-linear models show good scalability for up to 10 or so mdimensions. Regression and log-linear models are further discussed in Section 6.11.

**Histograms**

Histograms use binning to approximate data distributions and are a popular form of data reduction. A **histogram** for an attribute, *A*, partitions the data distribution of *A* into disjoint subsets, or *buckets*. If each bucket represents only a single attribute-value/frequency pair, the buckets are called *singleton buckets*. Often, buckets instead represent continuous ranges for the given attribute.

**Clustering**

Clustering techniques consider data tuples as objects. They partition the objects into groups or *clusters*, so that objects within a cluster are "similar" to one another and "dissimilar" to objects in other clusters. Similarity is commonly defined in terms of how "close" the objects are in space, based on a distance function. The "quality" of a cluster may be represented by its *diameter*, the maximum distance between any two objects in the cluster. *Centroid distance* is an alternative measure of cluster quality and is defined as the average distance of each cluster object from the cluster centroid (denoting the "aver- age object," or average point in space for the cluster). Data Discretization and Concept Hierarchy Generation

**Data discretization techniques** can be used to reduce the number of values for a given continuous attribute by dividing the range of the attribute into intervals. Interval labels can then be used to replace actual data

values. Replacing numerous values of a continuous attribute by a small number of interval labels thereby reduces and simplifies the original data. This leads to a concise, easy-to-use, knowledge-level representation of **Mining results.**

***Discretization techniques can be categorized based on how the discretization is performed, such as whether it uses class information or which direction it proceeds (i.e., top-down vs. bottom-up). If the discretization process uses class information, then we say it is supervised discretization. Otherwise, it is unsupervised. If the process starts by first finding one or a few points (called split points or cut points) to split the entire attribute range, and then repeats this recursively on the resulting intervals, it is called top-down discretization or splitting. This contrasts with bottom-up discretization or merging, which starts by considering all of the continuous values as potential split-points, removes some by merging neighborhood values to form intervals, and then recursively applies this process to the resulting intervals. Discretization can be performed recursively on an attribute to provide a hierarchical or multiresolution partitioning of the attribute values, known as a concept hierarchy. Concept hierarchies are useful for mining at multiple levels of abstraction.***

***A concept hierarchy for a given numerical attribute defines a discretization of the attribute. Concept hierarchies can be used to reduce the data by collecting and replac- ing low-level concepts (such as numerical values for the attribute age) with higher-level concepts (such as youth, middle-aged, or senior). Although detail is lost by such data generalization, the generalized data may be more meaningful and easier to interpret. This contributes to a consistent representation of data mining results among multiple mining tasks, which is a common requirement.***

In addition, mining on a reduced data set requires fewer input/output operations and is more efficient than mining on a larger, ungeneralized data set. Because of these benefits, discretization techniques and concept hierarchies are typically applied before data mining as a preprocessing step, rather than during mining.

An example of a concept hierarchy for the attribute *price* is given in Figure 2.22. More than one concept hierarchy can be defined for the same attribute in order to accommodate the needs of various users.

Manual definition of concept hierarchies can be a tedious and time-consuming task for a user or a domain expert. Fortunately, several discretization methods can be used to automatically generate or dynamically refine concept hierarchies for numerical attributes. Furthermore, many hierarchies for categorical attributes are

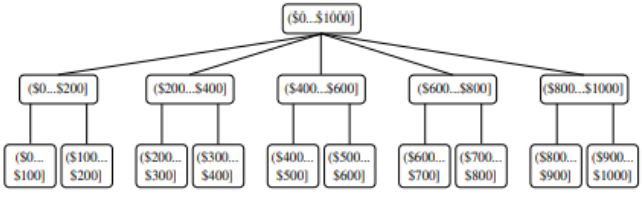



**Figure 2.22** A concept hierarchy for the attribute *price*, where an interval ($X ... $Y] denotes the range from $X (exclusive) to $Y (inclusive).

**Figure 2.22** A concept hierarchy for the attribute *price*, where an interval ($X … $Y ] denotes the range from $X (exclusive) to $Y (inclusive).

implicit within the database schema and can be automatically defined at the schema definition level.

Let's look at the generation of concept hierarchies for numerical and categorical data.

### 2.1.8 Discretization and Concept Hierarchy Generation for Numerical Data

It is difficult and laborious to specify concept hierarchies for numerical attributes because of the wide diversity of possible data ranges and the frequent updates of data values. Such manual specification can also be quite arbitrary.

Concept hierarchies for numerical attributes can be constructed automatically based on data discretization. We examine the following methods: *binning*, *histogram analysis*, *entropy-based discretization*, □2-*merging*, *cluster analysis*, and *discretization by intuitive partitioning*. In general, each method assumes that the values to be discretized are sorted in ascending order.

#### Binning

Binning is a top-down splitting technique based on a specified number of bins. Section 2.3.2 discussed binning methods for data smoothing. These methods are also used as discretization methods for numerosity reduction and concept hierarchy generation. For example, attribute values can be discretized by applying equal-width or equal-frequency binning, and then replacing each bin value by the bin mean or median, as in *smoothing by bin means* or *smoothing by bin medians*, respectively. These techniques can be applied recursively to the resulting partitions in order to gener- ate concept hierarchies. Binning does not use class information and is therefore an unsupervised discretization technique. It is sensitive to the user-specified number of bins, as well as the presence of outliers.

**Concept Hierarchy Generation for Categorical Data**

Categorical data are discrete data. Categorical attributes have a finite (but possibly large) number of distinct values, with no ordering among the values. Examples include *geo- graphic location*, *job category*, and *item type*. There are several methods for the generation of concept hierarchies for categorical data.

**Specification of a partial ordering of attributes explicitly at the schema level by users or experts:** Concept hierarchies for categorical attributes or dimensions typically involve a group of attributes. A user or expert can easily define a concept hierarchy by specifying a partial or total ordering of the attributes at the schema level. For example, a relational database or a dimension *location* of a data warehouse may contain the following group of attributes: *street, city, province or state*, and *country*. A hierarchy can be defined by specifying the total ordering among these attributes at the schema level, such as *street* < *city* < *province or state* < *country*.

**Specification of a portion of a hierarchy by explicit data grouping:** This is essentially the manual definition of a portion of a concept hierarchy. In a large database, it

is unrealistic to define an entire concept hierarchy by explicit value enumeration. On the contrary, we can easily specify explicit groupings for a small portion of intermediate-level data. For example, after specifying that *province* and *country* form a hierarchy at the schema level, a user could define some intermediate levels manually, such as "{*Alberta, Saskatchewan, Manitoba*} ⊂ *prairies Canada*" and "{*British Columbia, prairies Canada*} ⊂ *Western Canada*".

**Specification of a *set of attributes*, but not of their partial ordering:** A user may specify a set of attributes forming a concept hierarchy, but omit to explicitly state their partial ordering. The system can then try to automatically generate the attribute ordering so as to construct a meaningful concept hierarchy. *"Without knowledge of data semantics, how can a hierarchical ordering for an arbitrary set of categorical attributes be found?*

Consider the following observation that since higher-level concepts generally cover several subordinate lower-level concepts, an attribute defining a high concept level (e.g., *country*) will usually contain a smaller number of distinct values than an attribute defining a lower concept level (e.g., *street*).

Based on this observation, a concept hierarchy can be automatically generated based on the number of distinct values per attribute in the given attribute set. The attribute with the most distinct values is placed at the lowest level of the hierarchy. The lower the number of distinct values an attribute has, the higher it is in the generated concept hierarchy. This heuristic rule works well in many cases. Some local-level swapping or adjustments may be applied by users or experts, when necessary, after examination of the generated hierarchy.

Let's examine an example of this method.

**Concept hierarchy generation based on the number of distinct values per attribute.** Suppose a user selects a set of location-oriented attributes, *street, country, province or state,* and *city,* from the *AllElectronics* database, but does not specify the hierarchical ordering among the attributes.

A concept hierarchy for *location* can be generated automatically, as illustrated in Figure 2.24. First, sort the attributes in ascending order based

on the number of distinct values in each attribute. This results in the following (where the number of distinct values per attribute is shown in parentheses): *country* (15), *province or state* (365), *city* (3567), and *street* (674,339). Second, generate the hierarchy from the top down according to the sorted order, with the first attribute at the top level and the last attribute at the bottom level.

Finally, the user can examine the generated hierarchy, and when necessary, modify it to reflect desired semantic relationships among the attributes. In this example, it is obvious that there is no need to modify the generated hierarchy.

Note that this heuristic rule is not foolproof. For example, a time dimension in a database may contain 20 distinct years, 12 distinct months, and 7 distinct days of the
week. However, this does not suggest that the time hierarchy should be "*year* < *month*
< *days of the week*", with *days of the week* at the top of the hierarchy.

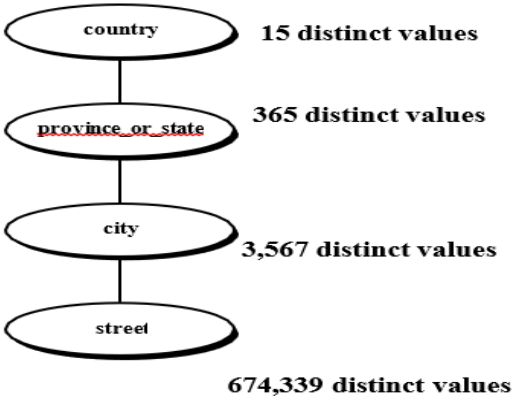


**Figure 2.24** Automatic generation of a schema concept hierarchy based on the number of distinct attribute values.

**Specification of only a partial set of attributes:**

- Sometimes a user can be sloppy when defining a hierarchy, or have only a vague idea about what should be included in a hierarchy. Consequently, the user may have included only a small subset of the relevant attributes in the hierarchy specification.
- For example, instead of including all of the hierarchically relevant attributes for *location*, the user may have specified only *street* and city. To handle such partially specified hierarchies, it is important to embed data semantics in the database schema so that attributes with tight semantic connections can be pinned together. In this way, the specification of one attribute may trigger a whole group of semantically tightly linked attributes to be "dragged in" to form a complete hierarchy. Users, however, should have the option to override this feature, as necessary.

# Data mining knowledge representation

## 1 What Defines a Data Mining Task?

- Task relevant data: where and how to retrieve the data to be used for mining
- Background knowledge: Concept hierarchies
- Interestingness measures: informal and formal selection techniques to be applied to the output knowledge
- Representing input data and output knowledge: the structures used to represent the input of the output of the data mining techniques
- Visualization techniques: needed to best view and document the results of the whole process

## 2 Task relevant data

- Database or data warehouse name: where to find the data
- Database tables or data warehouse cubes
- Condition for data selection, relevant attributes or dimensions and data grouping criteria: all this is used in the SQL query to retrieve the data

## 3 Background knowledge: Concept hierarchies

The concept hierarchies are induced by a *partial order*[1] over the values of a given attribute. Depending on the type of the ordering relation we distinguish several types of concept hierarchies.

### 3.1 Schema hierarchy

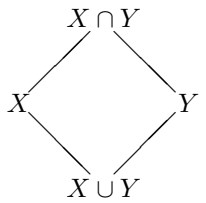
- Relating concept generality. The ordering reflects the generality of the attribute values, e.g. $street < city < state < country$.

### 3.2 Set-grouping hierarchy

- The ordering relation is the *subset* relation ($\subseteq$). Applies to set values.
- Example:

  $\{13, ..., 39\} = young$; $\{13, ..., 19\} = teenage$;
  $\{13, ..., 19\} \subseteq \{13, ..., 39\} \Rightarrow teenage < young$.

- Theory:
  - *power set*: the set of all subsets of a set, $X$.
  - *lattice* $(2^X, \subseteq)$, $sup(X, Y) = X \cap Y$, $inf(X, Y) = X \cup Y$.




  - top element $\top = \{\}$ (empty set), bottom element $\bot = X$.

[1] Consider a set $A$ and an ordering relation $R$. $R$ is a *full order* if for any $x, y \in A$, $xRy$ exists. $R$ is a *partial order* if for any $x \in A$, there exists $y \in A$, such that either $xRy$ or $yRx$ exists.

### 3.3 Operation-derived hierarchy

Produced by applying an operation (encoding, decoding, information extraction). For example:

markovz@cs.ccsu.edu

instantiates the hierarcy $user\_name < department < university < usa\_univeristy$.

### 3.4 Rule-based hierarchy

Using rules to define the partial order, for example:

if *antecedent* then *consequent*

defines the order $antecedent < consequent$.

## 4 Interestingness measures

Criteria to evaluate *hypotheses* (knowledge extracted from data when applying data mining techniques). This issue will be discussed in more detail in Lecture notes - Chapter 9: "Evaluating what's been learned".

### 4.1 Bayesian evaluation

- $E$ - data
- $H = \{H_1, H_2, ..., H_n\}$ - hypotheses
- $H_{best} = argmax_i\ P(H_i|E)$
- Bayes theorem:

$$P(H_i|E) = \frac{P(H_i)P(E|H_i)}{\sum_{i=1}^{n} P(H_i)P(E|H_i)}$$

### 4.2 Simplicity

**Occam's Razor**

Consider for example, association rule length, decision tree size, number and length of classification rules. The intuition suggests that the best hypotesis is the simplest (shortest) one. This is the so called *Occam's Razor Principle* also expressed as a mathematical theorem (Occam's Razor Theorem). Here is an example of applying this principle to grammars:

- Data:
  $E = \{0, 000, 00000, 0000000, 000000000\}$
- Hypotheses:
  $G_1 : S \rightarrow 0|000|00000|0000000|000000000$
  $G_2 : S \rightarrow 00S|0$
- Best hypothesis: $G_2$ (fewer and simpler rules)

However, as simplicity is a subjective measure we need formal criteria to define it.

**Formal criteria for simplicity**

- Bayesian approach: need of large volume of experimental results (statistics) to define prior probabilities.
- Algorithmic (Kolmogorov) complexity of an object (bit string): the length of the shortest program of Universal Turing Machine, that generates the string. Problems: computational complexity.
- Information-based approches: Minimum Description Length Principle (MDL). Most often used in practice.

### 4.3 Minimum Description Length Principle (MDL)

- Bayes Theorem:

$$P(H_i|E) = \frac{P(H_i)P(E|H_i)}{\sum_{i=1}^{n} P(H_i)P(E|H_i)}$$

- Take a $-\log$ of both sides of Bayes ($C$ is a constant):

$$-\log_2 P(H_i|E) = -\log_2 P(H_i) - \log_2 P(E|H_i) + C$$

- $I(A)$ – information in message $A$, $L(A)$ – min length of $A$ in bits: $\log_2 P(A) = I(A) = L(A)$
- Then: $L(H_i|E) = L(H_i) + L(E|H_i) + C$
- MDL: The hypothesis must reduce the information needed to encode the data, i.e.

$$L(E) > L(H_i) + L(E|H_i)$$

- The best hypothesis must maximize *information compression*:

$$H_{best} = argmax_i \ (L(E) - L(H_i) - L(E|H_i))$$

### 4.4 Certainty

- Confidence of association "if $A$ then $B$":

$$P(B|A) = \frac{\#\ of\ tuples\ containing\ both\ A\ and\ B}{\#\ of\ tupples\ containing\ A}$$

- Classification accuracy: Use a training set to generate the hypothesis, then test it on a separate test set.

$$Accuracy = \frac{\#\ of\ correct\ classifications}{\#\ of\ tuples\ in\ the\ test\ set}$$

- Utility (support) of association "if A then B":

$$P(A, B) = \frac{\#\ of\ tupples\ containing\ both\ A\ and\ B}{total\ \#\ of\ tupples}$$

# 5 Representing input data and output knowledge

## 5.1 Concepts (classes, categories, hypotheses): things to be mined/learned

- *Classification* mining/learning: predicting a discrete class, a kind of supervised learning, success is measured on new data for which class labels are known (test data).
- *Association* mining/learning: detecting associations between attributes, can be used to predict any attribute value and more than one attribute values, hence more rules can be generated, therefore we need constraints (minimum support and minimum confidence).
- *Clustering*: grouping similar instances into clusters, a kind of unsupervised learning, success is measured subjectively or by objective functions.
- *Numeric prediction*: predicting a numeric quantity, a kind of supervised learning, success is measured on test data.
- *Concept description*: output of the learning scheme

### 5.2 Instances (examples, tuples, transactions)

- Things to be classified, associated, or clustered.
- Individual, independent examples of the concept to be learned (target concept).
- Described by predetermined set of attributes.
- Input to the learning scheme: set of instances (dataset), represented as a single relation (table).
- Independence assumption: no relationships between attributes.
- Positive and negative examples for a concept, Closed World Assumption (CWA): $\{negative\} = \{all\} \backslash \{positive\}$.
- Relational (First Order Logic) descriptions:
  - Using variables (more compact representation). For example: $< a, b, b >$, $< a, c, c >$, $< b, a, a >$ can be represented as one relational tuple $< X, Y, Y >$.
  - Multiple relation concepts (FOIL, Inductive Logic Programming, see Lecture Notes - Chapter 11). Example:

    $grandfather(X, Z) \leftarrow father(X, Y) \wedge (father(Y, Z) \vee mother(Y, Z))$

### 5.3 Attributes (features)

- Predefined set of features to describe an instance.
- Nominal (categorical, enumerated, discrete) attributes:
  - Values are distinct symbols.
  - No relation among nominal values.

  - Only equality test can be performed.
  - Special case: boolean attributes, transforming nominal to boolean.
- Structured:
  - Partial order among nominal values
  - Example: concept hierarchy
- Numeric:
  - Continuous: full order (e.g. integer or real numbers).
  - Interval: partial order.

### 5.4 Output knowledge representation

- Association rules
- Decision trees
- Classification rules
- Rules with relations
- Prediction schemes:
  - Nearest neighbor
  - Bayesian classification
  - Neural networks
  - Regression
- Clusters:
  - Type of grouping: partitions/hierarchical
  - Grouping or describing: agglomerative/conceptual
  - Type of descriptions: statistical/structural

## 6 Visualization techniques: Why visualize data?

- Identifying problems:
  - Histograms for nominal attributes: is the distribution consistent with background knowledge?
  - Graphs for numeric values: detecting outliers.
- Visualization show dependencies
- Consulting domain experts
- If data are too much, take a sample