UNIT-I

Reference: Data Mining – Concepts and Techniques – 3rd Edition, Jiawei Han, Micheline Kamber & Jian Pei-Elsevier

Introduction: Fundamentals of data mining

Why Data Mining?

Moving toward the Information Age

"*We are living in the information age*": Terabytes or petabytes, of data pour into our computer networks, the World Wide Web (WWW), and various data storage devices every day from business, society, science and engineering, medicine, and almost every other aspect of daily life. This explosive growth of available data volume is a result of the computerization of our society and the fast development of powerful data collection and storage tools. Businesses worldwide generate gigantic data sets, including sales transactions, stock trading records, product descriptions, sales promotions, company profiles and performance, and customer feedback. For example, large stores, such as Wal-Mart, handle hundreds of millions of transactions per week at thousands of branches around the world.

Example 1.1 Data mining turns a large collection of data into knowledge. A search engine (e.g., Google) receives hundreds of millions of queries every day. Each query can be viewed as a transaction where the user describes her or his information need. What novel and useful knowledge can a search engine learn from such a huge collection of queries collected from users over time? Interestingly, some patterns found in user search queries can disclose invaluable knowledge that cannot be obtained by reading individual data items alone. For example, Google's *Flu Trends* uses specific search terms as indicators of flu activity. It found a close relationship between the number of people who search for flu-related information and the number of people who actually have flu symptoms. A pattern emerges when all of the search queries related to flu are aggregated. Using aggregated Google search data, *Flu Trends* can estimate flu activity up to two weeks faster than traditional systems can.

Data Mining as the Evolution of Information Technology

Data mining can be viewed as a result of the natural evolution of information technology. The database and data management industry evolved in the development of

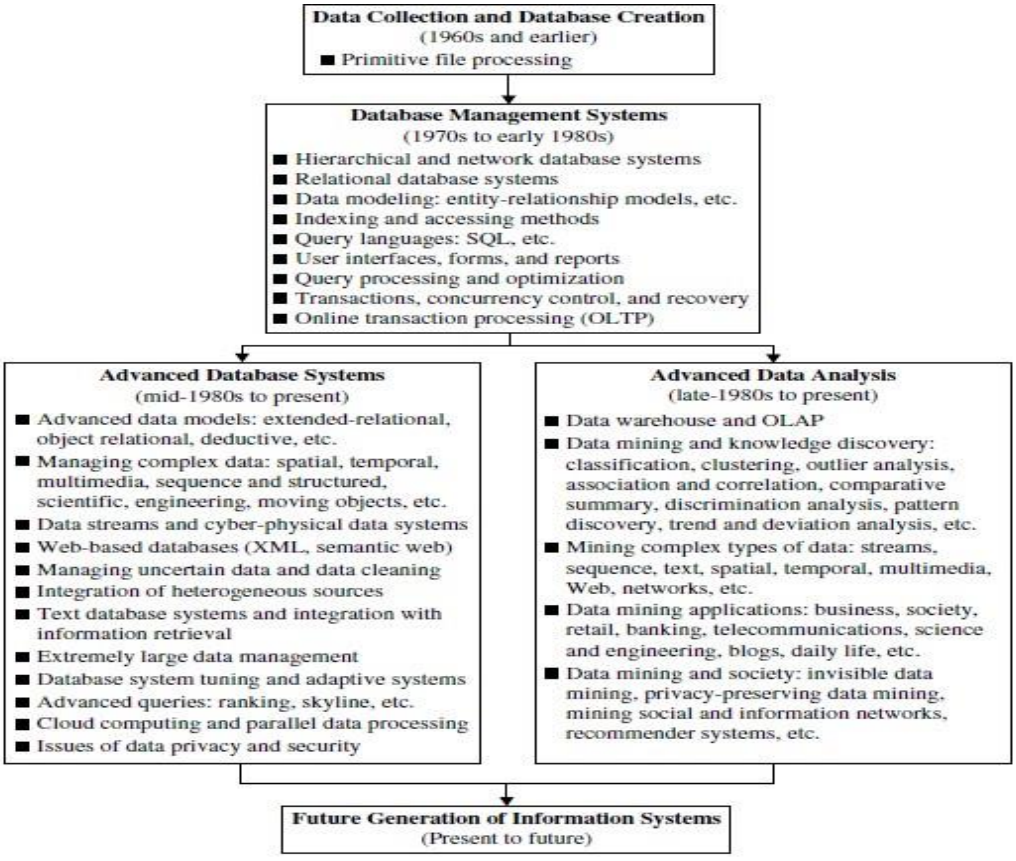


Figure 1.1 The evolution of database system technology.

several critical 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 evolved systematically from primitive file processing systems to sophisticated and powerful database systems. The research and development in database systems since the 1970s progressed from early hierarchical and network database systems to relational database systems (where data are stored in relational table structures; see Section 1.3.1), data modeling tools, and indexing and accessing methods.

After the establishment of database management systems, database technology moved toward the development of *advanced database systems*, *data warehousing*, and *data mining* for advanced data analysis and *web-based databases*. Advanced database systems, for example, resulted from an upsurge of research from the mid-1980s onward. These systems incorporate new and powerful data models such as extended-relational, object-oriented, object-relational, and deductive models. Application-oriented database systems have flourished, including spatial, temporal, multimedia, active, stream and sensor, scientific and engineering databases, knowledge bases, and office information bases. Issues related to the distribution, diversification, and sharing of data have been studied extensively. Data can now be stored in many different kinds of databases and information repositories.

One emerging data repository architecture is the **data warehouse** (Section 1.3.2). This is a repository of multiple heterogeneous data sources organized under a unified schema at a single site to facilitate management decision making. Data warehouse technology includes data cleaning, data integration, and online 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.

Huge volumes of data have been accumulated beyond databases and data warehouses.

During the 1990s, the World Wide Web and web-based databases (e.g., XML databases) began to appear. Internet-based global information bases, such as the WWW and various kinds of interconnected, heterogeneous databases, have emerged and play a vital role in the information industry. The effective and efficient analysis of data from such different forms of data by integration of information retrieval, data mining, and information network analysis technologies is a challenging task.

Figure 1.2 The world is data rich but information poor.

In summary, the abundance of data, coupled with the need for powerful data analysis tools, has been described as a *data rich but information poor* situation (Figure 1.2). 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. 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. Efforts have been made to develop expert system and knowledge-based technologies, which typically rely on users or domain experts to *manually* input knowledge into knowledge bases. Unfortunately, however, the manual knowledge input procedure is prone to biases and errors and is extremely costly and time consuming. The widening gap between data and information calls for the systematic development of *data mining tools* that can turn data tombs into "golden nuggets" of knowledge.

What Is Data Mining?

data mining should have been more appropriately named "knowledge mining from data," which is unfortunately somewhat long. However, the shorter term, *knowledge mining* may not reflect the emphasis on mining from large amounts of data. Many people treat data mining as a synonym for another popularly used term, **knowledge discovery from data**, or **KDD.** The knowledge discovery process is shown in Figure 1.4 as an iterative sequence of the following steps

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

1. Data cleaning (to remove noise and inconsistent data)
2. Data integration (where multiple data sources may be combined)

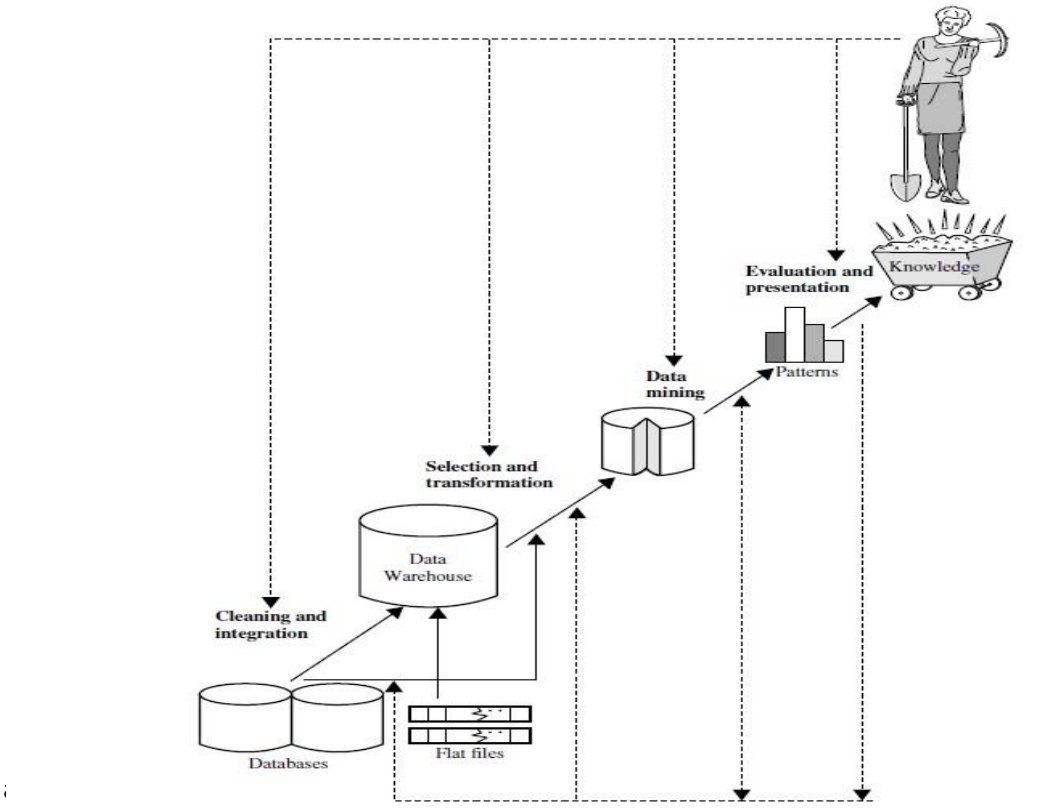


Figure 1.4 Data mining as a step in the process of knowledge discovery.

3. Data selection (where data relevant to the analysis task are retrieved from the database)
4. Data transformation (where data are transformed and consolidated into forms appropriate for mining by performing summary or aggregation operations)
5. Data mining (an essential process where intelligent methods are applied to extract data patterns)
6. Pattern evaluation (to identify the truly interesting patterns representing knowledge based on *interestingness measures*—see Section 1.4.6)
7. Knowledge presentation (where visualization and knowledge representation techniques are used to present mined knowledge to users).
Steps 1 through 4 are different forms of data preprocessing, where 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.
Data mining is the *process* of discovering interesting patterns and knowledge from *large* amounts of data. The data sources can include databases, data warehouses, theWeb, other information repositories, or data that are streamed into the system dynamically.

What Kinds of Data Can Be Mined?

The most basic forms of data for mining applications are database data , data warehouse data , and transactional data. Data mining can also be applied to other forms of data (e.g., data streams, ordered/sequence data, graph or networked data, spatial data, text data, multimedia data, and the WWW).

Database Data

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 provide mechanisms for defining database structures and data storage; for specifying and managing concurrent, shared, or distributed data access; and for ensuring consistency and security of the information stored despite system crashes or attempts at unauthorized access.
A **relational database** 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.

Example 1.2 A relational database for *AllElectronics*. The fictitious *AllElectronics* store is used to illustrate concepts throughout this book. The company is described by the following relation tables: *customer*, *item*, *employee*, and *branch*.

customer	(*cust_ID, name, address, age, occupation, annual_income, credit_information, category, ...*)
item	(*item_ID, brand, category, type, price, place_made, supplier, cost, ...*)
employee	(*empl_ID, name, category, group, salary, commission, ...*)
branch	(*branch_ID, name, address, ...*)
purchases	(*trans_ID, cust_ID, empl_ID, date, time, method_paid, amount*)
items_sold	(*trans_ID, item_ID, qty*)
works_at	(*empl_ID, branch_ID*)

Figure 1.5 Relational schema for a relational database, *AllElectronics*.

Relational data can be accessed by **database queries** written in a relational query language (e.g., SQL) or with the assistance of graphical user interfaces. A given query is transformed into a set of 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.

When **mining relational databases**, 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.

DataWarehouses

A **data warehouse** is a repository of information collected from multiple sources, stored under a unified schema, and usually residing at a single site. Data warehouses are constructed via a process of data cleaning, data integration, data transformation, data loading, and periodic data refreshing. To facilitate decision making, the data in a data warehouse are organized around *major subjects* (e.g., customer, item, supplier, and activity). The data are stored to provide information froma *historical perspective*, such as in the past 6 to 12 months, and are typically *summarized*.

A data warehouse is usually modeled by a multidimensional data structure, called a **data cube**, in which 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 *sum(sales_amount)*. A data cube provides a multidimensional view of data and allows the precomputation and fast access of summarized data.

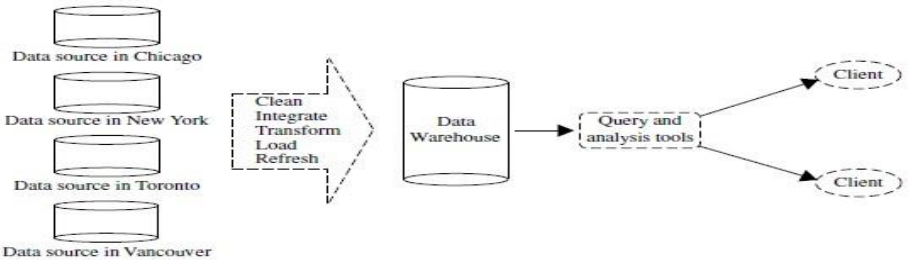


Figure 1.6 Typical framework of a data warehouse for *AllElectronics*.

E is presented in ***F*** *ancouver*), *time* (with quarter values *Q1, Q2, Q3, Q4*), and *item*(with itemtype 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 the items related to security systems in Vancouver is $400, 000, as stored in cell <*Vancouver, Q1, security*>.

Examples of OLAP operations include **drill-down** and **roll-up**, which allow the user to view the data at differing degrees of summarization. For instance, we can drill down on sales data summarized by *quarter* to see data summarized by *month*. Similarly, we can roll up on sales data summarized by *city* to view data summarized by *country*.

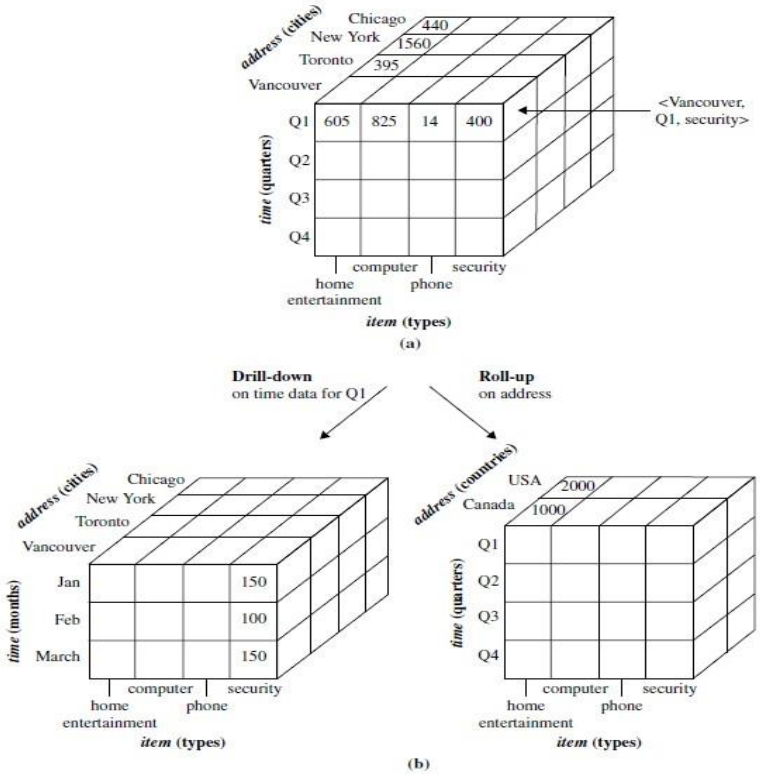


Figure 1.7 A multidimensional data cube, commonly used for data warehousing, (a) showing summarized data for *AllElectronics* and (b) showing summarized data resulting from drill-down and roll-up operations on the cube in (a). For improved readability, only some of the cube cell values are shown.

Prepa
Source

Transactional Data

In general, each record in a **transactional database** captures a transaction, such as a customer's purchase, a flight booking, or a user's clicks on a web page. A transaction typically includes a unique transaction identity number (*trans ID*) and a list of the **items** making up the transaction, such as the items purchased in the transaction.

Example 1.4 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 1.8. From the relational database point of view, the *sales* table in the figure is a nested relation because the attribute *list of item IDs* contains a set of *items*.

As an analyst of *AllElectronics*, you may ask, *"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 boosting sales. For example, given the knowledge that printers are commonly purchased together with computers, you could offer certain printers at a steep discount (or even for free) to customers buying selected computers, in the hopes of selling more computers (which are often more expensive than printers).

trans_ID	*list_of_item_IDs*
T100	I1, I3, I8, I16
T200	I2, I8
...	...

Figure 1.8 Fragment of a transactional database for sales at *AllElectronics*.

Other Kinds of Data

Besides relational database data, data warehouse data, and transaction data, there are many other kinds of data that have versatile forms and structures and rather different semantic meanings. Such kinds of data can be seen in many applications: time-related or sequence data (e.g., historical records, stock exchange data, and time-series and biological sequence data), data streams (e.g., video surveillance and sensor data, which are continuously transmitted), spatial data (e.g., maps), engineering design data (e.g., the design of buildings, system components, or integrated circuits), hypertext and multimedia data (including text, image, video, and audio data), graph and networked data (e.g., social and information networks), and the Web (a huge, widely distributed information repository made available by the Internet). These applications bring about new challenges, like how to handle data carrying special structures (e.g., sequences, trees, graphs, and networks) and specific semantics (such as ordering, image, audio and video contents, and connectivity), and how to mine patterns that carry rich structures and semantics.

What Kinds of Patterns Can Be Mined? Or Datamining Functionalities Or Datamining Task Primitives

Data mining functionalities are used to specify the kinds of patterns to be found in data mining tasks. In general, such tasks can be classified into two categories: **descriptive** and **predictive**. Descriptive mining tasks characterize properties of the data in a target data set. Predictive mining tasks perform induction on the current data in order to make predictions. Data mining functionalities are described below:

1. Class/Concept Description: Characterization and Discrimination
2. Mining Frequent Patterns, Associations, and Correlations
3. Classification and Regression for Predictive Analysis
4. Cluster Analysis
5. Outlier Analysis
6. Are All Patterns Interesting?

1. Class/Concept Description: Characterization and Discrimination:

Data entries 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.*

Data characterization is a summarization of the general characteristics or features of a target class of data.for example, A customer relationship manager at *AllElectronics* may order the following data mining task: *Summarize the characteristics of customers who spend more than $5000 a year at AllElectronics*. The result is a general profile of these customers, such as that they are 40 to 50 years old, employed, and have excellent credit ratings.

Data discrimination is a comparison of the general features of the target class data objects against the general features of objects from one or multiple contrasting classes. The target and contrasting classes can be specified by a user, and the corresponding data objects can be retrieved through database queries. For example, a user may want to compare the general features of software products with sales that increased by 10% last year against those with sales that decreased by at least 30% during the same period. The methods used for data discrimination are similar to those used for data characterization.

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 frequent itemsets, frequent subsequences (also known as sequential patterns), and frequent substructures. A *frequent itemset* typically refers to a set of items that often appear together in a transactional data set— for example, milk and bread, which are frequently bought together in grocery stores by many customers. A frequently occurring subsequence, such as the pattern that customers, tend to purchase first a laptop, followed by a digital camera, and then a memory card, is a (*frequent*) *sequential pattern.* A substructure can refer to different structural forms (e.g., graphs, trees, or lattices) that may be combined with itemsets or subsequences. If a substructure occurs frequently, it is called a (*frequent*) *structured pattern.*

Association analysis. Suppose that, as a marketing manager at *AllElectronics*, you want to know which items are frequently purchased together (i.e., within the same transaction).
An example of such a rule, mined fromthe *AllElectronics* transactional database, is

$$buys(X, \text{“}computer\text{”}) \Rightarrow buys(X, \text{“}software\text{”}) \; [support = 1\%, confidence = 50\%],$$

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 the transactions under analysis show that computer and software are 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**.
Suppose, instead, that we are given the *AllElectronics* relational database related to purchases. A data mining system may find association rules like

$$age(X, \text{“}20..29\text{”}) \wedge income(X, \text{“}40K..49K\text{”}) \Rightarrow buys(X, \text{“}laptop\text{”})$$
$$[support = 2\%, confidence = 60\%].$$

The rule indicates that of the *AllElectronics* customers under study, 2% are 20 to 29 years old with an income of $40,000 to $49,000 and have purchased a laptop (computer) at *AllElectronics*. There is a 60% probability that a customer in this age and income group will purchase a laptop. Note that this is an association involving 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 analysis can be performed to uncover interesting statistical **correlations** between associated attribute–value pairs.

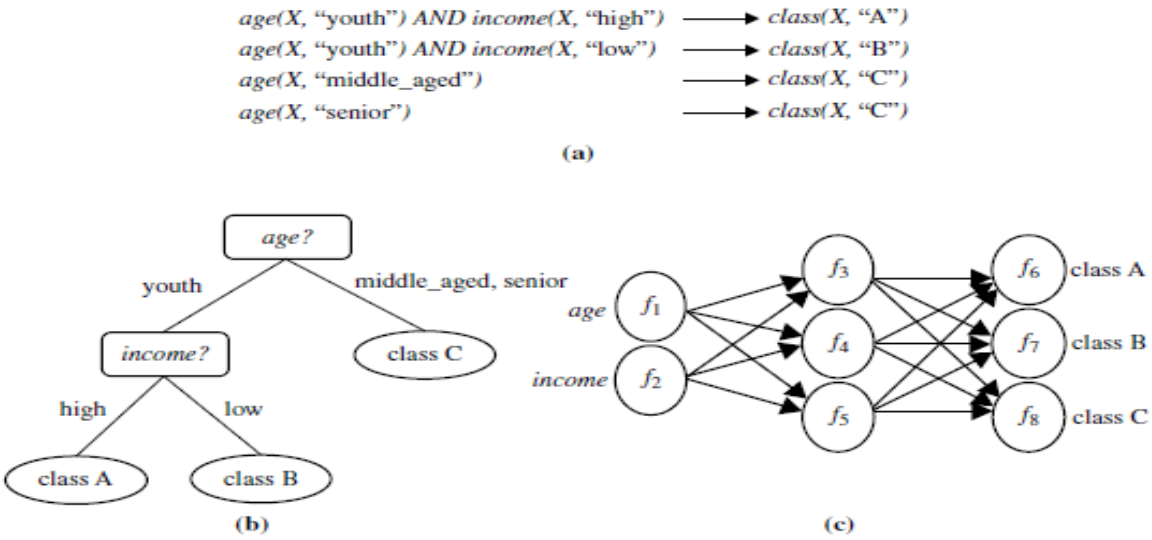
3. Classification and Regression for Predictive Analysis

Classification is the process of finding a **model** (or function) that describes and distinguishes data classes or concepts. The model are derived based on the analysis of a set of **training data** (i.e., data objects for which the class labels are known). The model is used to predict the class label of objects for which the the class label is unknown.

“How is the derived model presented?” The derived model may be represented in various forms, such as

classification rules (i.e., *IF-THEN rules*), *decision trees*, *mathematical formulae*, or *neural networks* (Figure 1.9).A **decisiontree** is a flowchart-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 unitswithweighted connections between the units. There are many other methods for constructing classificationmodels, such as na¨ıve Bayesian classification, support vector machines, and *k*-nearest-neighbor classification.

Whereas classification predicts categorical (discrete, unordered) labels, **regression** models continuous-valued functions. **Regression analysis** is a statistical methodology that is most often used for numeric prediction, although other methods exist as well.

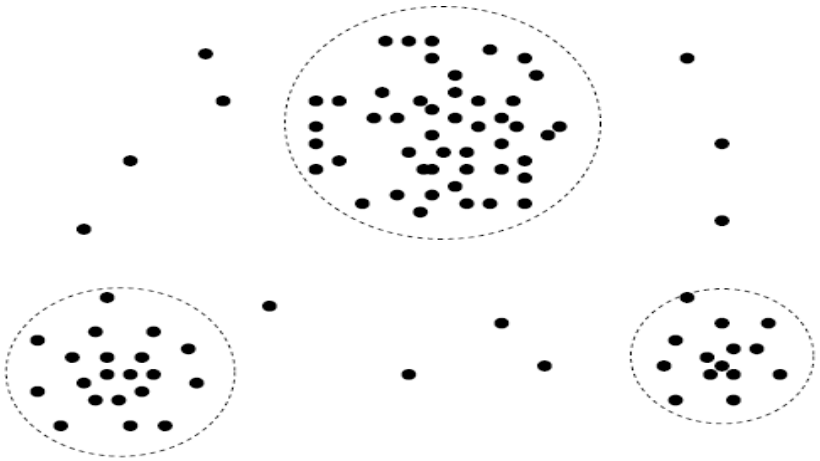


A classification model can be represented in various forms: (a) IF-THEN rules, (b) a decision tree, or (c) a neural network.

For example, **Classification and regression.** Suppose as a sales manager of *AllElectronics* you want to classify a large set of items in the store, based on three kinds of responses to a sales campaign:*good response*, *mild response* and *no response*. You want 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*.

4. Cluster Analysis

Unlike classification and regression, which analyze class-labeled (training) data sets, **clustering** analyzes data objects without consulting class labels. In many cases, classlabeled data may simply not exist at the beginning. Clustering can be used to generate class labels for a group of data. The objects are clustered or grouped based on the principle of *maximizing the intraclass similarity and minimizing the interclass similarity*.



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

5. Outlier Analysis

A data set may contain objects that do not comply with the general behavior or model of the data. These data objects are **outliers**. Many data mining methods discard outliers as noise or exceptions. However, in some applications (e.g., 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 analysis** or **anomaly mining**.

For example, **Outlier analysis.** Outlier analysis may uncover fraudulent usage of credit cards by detecting purchases of unusually 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 locations and types of purchase, or the purchase frequency.

6. Are All Patterns Interesting?

"Are all of the patterns interesting?" Typically, the answer is no—only a small fraction of the patterns potentially generated would actually be of interest to a given user. *"What makes a pattern interesting? Can a data mining system generate all of the interesting patterns? Or, Can the system generate only the interesting ones?"* 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*.

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. 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.

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

Other objective interestingness measures include *accuracy* and *coverage* for classification (IF-THEN) rules. In general terms, accuracy tells us the percentage of data that are correctly classified by a rule. Coverage is similar to support, in that it tells us the percentage of data to which a rule applies.

Although objective measures help identify interesting patterns, they are often insufficient unless combined with subjective measures that reflect a particular user's needs and interests. For example, patterns describing the characteristics of customers who shop frequently at *AllElectronics* should be interesting to the marketing manager, but may be of little interest to other analysts studying the same database for patterns on employee performance. **Subjective interestingness measures** are based on user beliefs in the data. These measures find patterns interesting if the patterns 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**. For example, patterns like "a large earthquake often follows a cluster of small quakes" may be highly actionable if users can act on the information to save lives. Patterns that are **expected** can be interesting if they confirm a hypothesis that the user wishes to validate or they 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 possible patterns.

Finally, the third question—*"Can a data mining system generate only interesting patterns?"*— is an optimization problem in data mining. It is highly desirable for data mining systems to generate only interesting patterns.

Which Technologies Are Used?

As a highly application-driven domain, data mining has incorporated many techniques from other domains such as statistics, machine learning, pattern recognition, database and data warehouse systems, information retrieval, visualization, algorithms, highperformance computing, and many application domains

Statistics

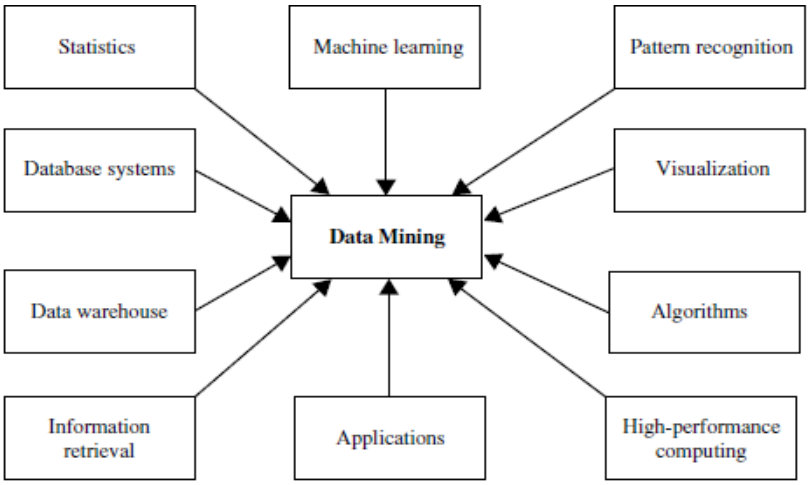
Statistics studies the collection, analysis, interpretation or explanation, and presentation of data. Data mining has an inherent connection with statistics.

A **statistical model** is a set of mathematical functions that describe the behavior of the objects in a target class in terms of random variables and their associated probability distributions. Statistical models are widely used to model data and data classes. For example, in data mining tasks like data characterization and classification, statistical models of target classes can be built. In other words, such statistical models can be the outcome of a data mining task.

Machine Learning

Machine learning investigates how computers can learn (or improve their performance) based on data. A main research area is for computer programs to *automatically* learn to recognize complex patterns and make intelligent decisions based on data. For example, a typical machine learning problem is to program a computer so that it can automatically recognize handwritten postal codes on mail after learning from a set of examples.

Machine learning is a fast-growing discipline. Here, we illustrate classic problems in machine learning that are highly related to data mining.



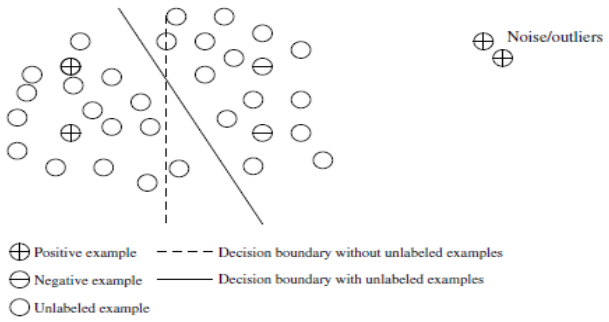
Data mining adopts techniques from many domains.

Supervised learning is basically a synonym for classification. The supervision in the learning comes from the labeled examples in the training data set. For example, in the postal code recognition problem, a set of handwritten postal code images and their corresponding machine-readable translations are used as the training examples, which supervise the learning of the classification model.

Unsupervised learning is essentially a synonym for clustering. The learning process is unsupervised since the input examples are not class labeled. Typically, we may use clustering to discover classes within the data. For example, an unsupervised learning method can take, as input, a set of images of handwritten digits. Suppose that it finds 10 clusters of data. These clusters may correspond to the 10 distinct digits of 0 to 9, respectively. However, since the training data are not labeled, the learned model cannot tell us the semantic meaning of the clusters found.

Semi-supervised learning is a class of machine learning techniques that make use of both labeled and unlabeled examples when learning a model. In one approach, labeled examples are used to learn class models and unlabeled examples are used to refine the boundaries between classes. For a two-class problem, we can think of the set of examples belonging to one class as the *positive examples* and those belonging to the other class as the *negative examples*.

Active learning is a machine learning approach that lets users play an active role in the learning process. An active learning approach can ask a user (e.g., a domain expert) to label an example, which may be from a set of unlabeled examples or synthesized by the learning program. The goal is to optimize the model quality by actively acquiring knowledge from human users, given a constraint on how many examples they can be asked to label.



Semi-supervised learning.

Database Systems and DataWarehouses

Database systems research focuses on the creation, maintenance, and use of databases for organizations and end-users. Particularly, database systems researchers have established highly recognized principles in data models, query languages, query processing and optimization methods, data storage, and indexing and accessing methods. Database systems are often well known for their high scalability in processing very large, relatively structured data sets.
Recent database systems have built systematic data analysis capabilities on database data using data warehousing and data mining facilities. A **data warehouse** integrates data originating from multiple sources and various timeframes. It consolidates data in multidimensional space to form partially materialized data cubes. The data cube model not only facilitates OLAP in multidimensional databases but also promotes *multidimensional data mining.*

Information Retrieval

Information retrieval (**IR**) is the science of searching for documents or information in documents. Documents can be text or multimedia, and may reside on the Web. The differences between traditional information retrieval and database systems are twofold: Information retrieval assumes that (1) the data under search are unstructured; and (2) the queries are formed mainly by keywords, which do not have complex structures.
The typical approaches in information retrieval adopt probabilistic models.(i) **language model (ii) topic model.**
Increasingly large amounts of text and multimedia data have been accumulated and made available online due to the fast growth of the Web and applications such as digital libraries, digital governments, and health care information systems.
Their effective search and analysis have raised many challenging issues in data mining. Therefore, text mining and multimedia data mining, integrated with information retrieval methods, have become increasingly important.

Major Issues in Data Mining *Life is short but art is long. – Hippocrates*

Data mining is a dynamic and fast-expanding field with great strengths. major issues in data mining are partitioned into five groups: *mining methodology, user interaction, efficiency and scalability*, *diversity of data types*, and *data mining and society*.

Mining Methodology

Researchers have been vigorously developing new data mining methodologies. This involves the investigation of new kinds of knowledge, mining in multidimensional space, integrating methods fromother disciplines, and the consideration of semantic ties among data objects. In addition, mining methodologies should consider issues such as data uncertainty, noise, and incompleteness. Some mining methods explore how userspecified measures can be used to assess the interestingness of discovered patterns as well as guide the discovery process.
Various aspects of mining methodology are
Mining various and new kinds of knowledge:
Data mining covers a wide spectrum of data analysis and knowledge discovery tasks, fromdata characterization and discrimination to association and correlation analysis, classification, regression, clustering, outlier analysis, sequence analysis, and trend and evolution analysis. These tasks may use the same database in different ways and require the development of numerous data mining techniques.
Mining knowledge in multidimensional space:
When searching for knowledge in large data sets, we can explore the data in multidimensional space. That is, we can search for interesting patterns among combinations of dimensions (attributes) at varying levels of abstraction. Such mining is known as *(exploratory) multidimensional data mining*.
Data mining—an interdisciplinary effort:
The power of data mining can be substantially enhanced by integrating new methods from multiple disciplines. For example, to mine data with natural language text, it makes sense to fuse data mining methods with methods of information retrieval and natural language processing.
Boosting the power of discovery in a networked environment:
Knowledge derived in one set of objects can be used to boost the discovery of knowledge in a “related” or semantically linked set of objects.
Handling uncertainty, noise, or incompleteness of data:
Data often contain noise, errors, exceptions, or uncertainty, or are incomplete. Errors and noise may confuse the data mining process, leading to the derivation of erroneous patterns.
Pattern evaluation and pattern- or constraint-guided mining:
Not all the patterns generated by data mining processes are interesting. What makes a pattern interesting may vary from ser to user. Therefore, techniques are needed to assess the interestingness of discovered patterns based on subjective measures.

User Interaction

The user plays an important role in the data mining process. Interesting areas of research include *how to interact with a data mining system*, *how to incorporate a user's background knowledge in mining*, and *how to visualize and comprehend data mining results.*

Interactive mining:

The data mining process should be highly *interactive*. Thus, it is important to build flexible user interfaces and an exploratory mining environment, facilitating the user's interaction with the system. A user may like to first sample a set of data, explore general characteristics of the data, and estimate potential mining results.

Incorporation of background knowledge:

Background knowledge, constraints, rules, and other information regarding the domain under study should be incorporated into the knowledge discovery process.

Ad hoc data mining and data mining query languages:

Query languages (e.g., SQL) have played an important role in flexible searching because they allow users to pose ad hoc queries. Similarly, high-level data mining query languages or other high-level flexible user interfaces will give users the freedom to define ad hoc data mining tasks.

Presentation and visualization of data mining results:

How can a data mining system present data mining results, vividly and flexibly, so that the discovered knowledge can be easily understood and directly usable by humans? This is especially crucial if the data mining process is interactive.

Efficiency and Scalablility

Efficiency and scalability are always considered when comparing data mining algorithms. As data amounts continue to multiply, these two factors are especially critical.

Efficiency and scalability of data mining algorithms:

Data mining algorithms must be efficient and scalable in order to effectively extract information from huge amounts of data in many data repositories or in dynamic data streams.

Parallel, distributed, and incremental mining algorithms:

The humongous size of many data sets, the wide distribution of data, and the computational complexity of some data mining methods are factors that motivate the development of **parallel and distributed data-intensive mining algorithms**. Such algorithms first partition the data into "pieces." Each piece is processed, in parallel, by searching for patterns.

Cloud computing and *cluster computing*, which use computers in a distributed and collaborative way to tackle very large-scale computational tasks, are also active research themes in parallel data mining.

Diversity of Database Types

The wide diversity of database types brings about challenges to data mining. These include

Handling complex types of data:

It is unrealistic to expect one data mining system to mine all kinds of data, given the diversity of data types and the different goals of data mining. Domain- or application-dedicated data mining systems are being constructed for indepth mining of specific kinds of data. The construction of effective and efficient data mining tools for diverse applications remains a challenging and active area of research.

Mining dynamic, networked, and global data repositories:

Multiple sources of data are connected by the Internet and various kinds of networks, forming gigantic, distributed, and heterogeneous global information systems and networks. The discovery of knowledge from different sources of structured, semi-structured, or unstructured yet interconnected data with diverse data semantics poses great challenges to data mining.

Data Mining and Society

How does data mining impact society? What steps can data mining take to preserve the privacy of individuals? Do we use data mining in our daily lives without even knowing that we do? These questions raise the following issues:

Social impacts of data mining:

How can we use data mining technology to benefit society? How can we guard against its misuse? The improper disclosure or use of data and the potential violation of individual privacy and data protection rights are areas of concern that need to be addressed.

Privacy-preserving data mining:

Data mining will help scientific discovery, business management, economy recovery, and security protection (e.g., the real-time discovery of intruders and cyberattacks). The philosophy is to observe data sensitivity and preserve people's privacy while performing successful data mining.

Invisible data mining:

We cannot expect everyone in society to learn and master data mining techniques. More and more systems should have data mining functions built within so that people can perform data mining or use data mining results simply by mouse clicking, without any knowledge of data mining algorithms. Intelligent search engines and Internet-based stores perform

such *invisible data mining* by incorporating data mining into their components to improve their functionality and performance. This is done often unbeknownst to the user.

Data Preprocessing

Low-quality data will lead to low-quality mining results. *Data cleaning* can be applied to remove noise and correct inconsistencies in data. *Data integration* merges data from multiple sources into a coherent data store such as a data warehouse. *Data reduction* can reduce data size by, for instance, aggregating, eliminating redundant features, or clustering. *Data transformations* (e.g., normalization) may be applied, where data are scaled to fall within a smaller range like 0.0 to 1.0. This can improve the accuracy and efficiency of mining algorithms involving distancemeasurements.

Data Quality: Why Preprocess the Data?

Data have quality if they satisfy the requirements of the intended use. There are many factors comprising **data quality**, including *accuracy*, *completeness*, *consistency*, *timeliness*, *believability*, and *interpretability*.
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); *inaccurate* or *noisy* (containing errors, or values that deviate from the expected); and *inconsistent* (e.g., containing discrepancies in the department codes used to categorize items) – Real World data.
three of the elements defining data quality: **accuracy**, **completeness**, and **consistency**.
Many possible reasons for inaccurate data (i.e., having incorrect attribute values). The data collection instruments used may be faulty. There may have been human or computer errors occurring at data entry. Users may purposely submit incorrect data values for mandatory fields when they do not wish to submit personal information (e.g., by choosing the default value "January 1" displayed for birthday). This is known as *disguised missing data*.
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, or inconsistent formats for input fields (e.g., *date*). Duplicate tuples also require data cleaning.
Incomplete data can occur for a number of reasons. Attributes of interest may not always be available, such as customer information for sales transaction data. Other data may not be included simply because they were not considered important at the time of entry. Relevant data may not be recorded due to a misunderstanding or because of equipment malfunctions.
Timeliness also affects data quality. Suppose that you are overseeing the distribution of monthly sales bonuses to the top sales representatives at *AllElectronics*. Several sales representatives, however, fail to submit their sales records on time at the end of the month.
Two other factors affecting data quality are believability and interpretability. **Believability** reflects how much the data are trusted by users, while **interpretability** reflects how easy the data are understood.

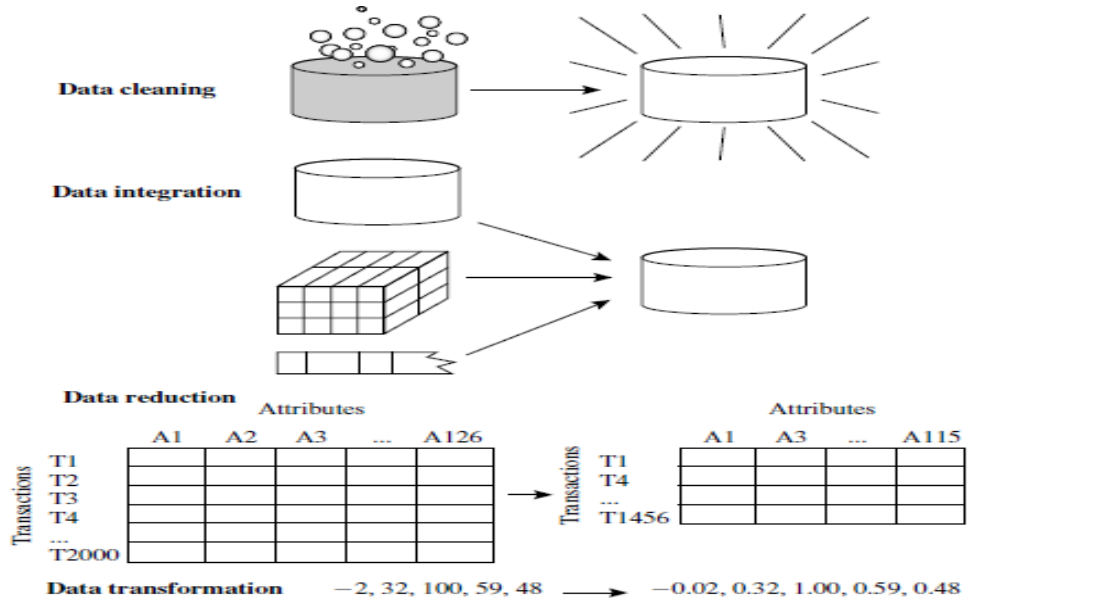
Major Tasks in Data Preprocessing

They are four major tasks in preprocessing. They are

1. Data cleaning
2. data integration
3. Data reduction
4. data transformation

Data cleaning routines work to "clean" the data by filling in missing values, smoothing noisy data, identifying or removing outliers, and resolving inconsistencies.
suppose that you would like to include data from multiple sources in your analysis. This would involve integrating multiple databases, data cubes, or files (i.e., **data integration**). Typically, data cleaning and data integration are performed as a preprocessing step when preparing data for a data warehouse. Additional
data cleaning can be performed to detect and remove redundancies that may have resulted from data integration.
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. Data reduction strategies include *dimensionality reduction* and *numerosity reduction.*
In **dimensionality reduction**, data encoding schemes are applied so as to obtain a reduced or "compressed" representation of the original data. Examples include data compression techniques (e.g., *wavelet transforms* and *principal components analysis*), *attribute subset selection* (e.g., removing irrelevant attributes), and *attribute construction* (e.g., where a small set of more useful attributes is derived from the original set).
In **numerosity reduction**, the data are replaced by alternative, smaller representations using parametric models (e.g., *regression* or *log-linear models*) or nonparametric models (e.g., *histograms*, *clusters*, *sampling*, or *data aggregation*).

Normalization, data discretization, and concept hierarchy generation are forms of **data transformation**.
Normalization -scaled to a smaller range such as [0.0, 1.0].
Discretization and *concept hierarchy generation* can also be useful, where raw data values for attributes are replaced by ranges or higher conceptual levels. For example, raw values for *age* may be replaced by higher-level concepts, such as *youth*, *adult*, or *senior*.



Forms of data preprocessing.

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 identifying outliers, and correct inconsistencies in the data.

Missing Values

How can you go about filling in the missing values for this attribute? Methods are

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

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."*

4. Use a measure of central tendency for the attribute (e.g., the mean or median) to fill in the missing value: measures of central tendency are, the "middle" value of a data distribution. For normal (symmetric) data distributions, the mean can be used, while skewed data distribution should employ the median.

5. Use the attribute mean or median for all samples belonging to the same class as the given tuple: For example, if classifying customers according to *credit risk*, we may replace the missing value with the mean *income* value for customers in the same credit risk category as that of the given tuple. If the data distribution for a given class is skewed, the median value is a better choice.

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 through 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.

Noisy Data

"What is noise?" **Noise** is a random error or variance in a measured variable.

Binning: Binning methods smooth a sorted data value by consulting its "neighborhood," 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. 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.
In **smoothing by bin boundaries**, the minimum 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.

Regression: Data smoothing can also be done by regression, a technique that conforms data values to a function. *Linear regression* involves finding the "best" line to fit two attributes (or variables) so that one attribute can be used to predict the other. *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.

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

Binning methods for data smoothing.

Outlier analysis: Outliers may be detected by clustering, for example, where similar values are organized into groups, or "clusters." Intuitively, values that fall outside of the set of clusters may be considered outliers.

Data Cleaning as a Process

The two-step process of discrepancy detection and data transformation (to correct discrepancies) iterates. This process, however, is error-prone and time consuming.
The first step in data cleaning as a process is *discrepancy detection*. Discrepancies can be caused by several factors, including poorly designed data entry forms that have many optional fields, human error in data entry, deliberate errors (e.g., respondents not wanting to divulge information about themselves), and data decay (e.g., outdated addresses).

The data should also be examined regarding unique rules, consecutive rules, and null rules. A **unique rule** says that each value of the given attribute must be different from all other values for that attribute. A **consecutive rule** says that there can be no missing values between the lowest and highest values for the attribute, and that all values must also be unique (e.g., as in check numbers). A **null rule** specifies the use of blanks, question marks, special characters, or other strings that may indicate the null condition (e.g., where a value for a given attribute is not available), and how such values should be handled.

There are a number of different commercial tools that can aid in the discrepancy detection step. **Data scrubbing tools** use simple domain knowledge (e.g., knowledge of postal addresses and spell-checking) to detect errors and make corrections in the data. These tools rely on parsing and fuzzy matching techniques when cleaning data from multiple sources. **Data auditing tools** find discrepancies by analyzing the data to discover rules and relationships, and detecting data that violate such conditions.

Commercial tools can assist in the data transformation step. **Data migration tools** allow simple transformations to be specified such as to replace the string *"gender"* by *"sex."* **ETL (extraction/transformation/loading) tools** allow users to specify transforms through a graphical user interface (GUI). These tools typically support only a restricted set of transforms so that, often, we may also choose to write custom scripts for this step of the data cleaning process.

The two-step process of discrepancy detection and data transformation (to correct discrepancies) iterates. This process, however, is error-prone and time consuming. Some transformations may introduce more discrepancies. Some

nested discrepancies may only be detected after others have been fixed. For example, a typo such as "20010" in a year field may only surface once all date values have been converted to a uniform format.

Data Integration

the merging of data from multiple data stores. Careful integration can help reduce and avoid redundancies and inconsistencies in the resulting data set. This can help improve the accuracy and speed of the subsequent data mining process.

Entity Identification Problem

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. Such metadata can be used to help avoid errors in schema integration.

When matching attributes from one database to another during integration, special attention must be paid to the *structure* of the data. This is to ensure that any attribute functional dependencies and referential constraints in the source system match those in the target system. For example, in one system, a *discount* may be applied to the order, whereas in another system it is applied to each individual line item within the order. If this is not caught before integration, items in the target system may be improperly discounted.

Redundancy and Correlation Analysis

Redundancy is another important issue in data integration. 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 nominal data, we use the χ^2 (*chi-square*) test. For numeric attributes, we can use the *correlation coefficient* and *covariance*, both of which access how one attribute's values vary from those of another.

χ^2 Correlation Test for Nominal Data

For nominal data, a correlation relationship between two attributes, *A* and *B*, can be discovered by a χ^2 (**chi-square**) test. Suppose *A* has *c* distinct values, namely $a_1, a_2, ..., a_c$. *B* has *r* distinct values, namely $b_1, b_2, ..., b_r$. The data tuples described by *A* and *B* can be shown as a **contingency table**, with the *c* values of *A* making up the columns and the *r* values of *B* making up the rows. Let (A_i, B_j) denote the joint event that attribute *A* takes on value a_i and attribute *B* takes on value b_j, that is, where $(A = a_i, B = b_j)$.

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

where o_{ij} is the *observed frequency* (i.e., actual count) of the joint event (A_i, B_j) and e_{ij} is the *expected frequency* of (A_i, B_j), which can be computed as

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

where *n* is the number of data tuples, *count* $(A = a_i)$ is the number of tuples having value a_i for *A*, and *count* $(B = b_j)$ is the number of tuples having value b_j for *B*.

The χ^2 statistic tests the hypothesis that *A* and *B* are *independent*, that is, there is no correlation between them. The test is based on a significance level, with (r-1) * (c-1) degrees of freedom. If the hypothesis can be rejected, then we say that *A* and *B* are statistically correlated.

Example 3.1 Correlation analysis of nominal attributes using χ^2. Suppose that a group of 1500 people was surveyed. The gender of each person was noted. Each person was polled as to whether his or her preferred type of reading material

was fiction or nonfiction. Thus, we have two attributes, *gender* and *preferred_reading*. The observed frequency (or count) of each possible joint event is summarized in the contingency table shown below, where the numbers in parentheses are the expected frequencies. The expected frequencies are calculated based on the data distribution for both attributes using Eq. e_{ij}.
we can verify the expected frequencies for each cell. For example, the expected frequency for the cell (*male*, *fiction*) is

$$e_{11} = \frac{count(male) \times count(fiction)}{n} = \frac{300 \times 450}{1500} = 90,$$

and so on. Notice that in any row, the sum of the expected frequencies must equal the total observed frequency for that row, and the sum of the expected frequencies in any column must also equal the total observed frequency for that column.

Table 3.1 Example 2.1's 2 × 2 Contingency Table Data

	male	*female*	*Total*
fiction	250 (90)	200 (360)	450
non_fiction	50 (210)	1000 (840)	1050
Total	300	1200	1500

Note: Are *gender* and *preferred_reading* correlated?

χ^2 computation, we get

$$\chi^2 = \frac{(250-90)^2}{90} + \frac{(50-210)^2}{210} + \frac{(200-360)^2}{360} + \frac{(1000-840)^2}{840}$$
$$= 284.44 + 121.90 + 71.11 + 30.48 = 507.93.$$

For this 2*2 table, the degrees of freedom are (2-1)(2-1)=1. For 1 degree of freedom, the χ^2 value needed to reject the hypothesis at the 0.001 significance level is 10.828. Since our computed value is above this, we can reject the hypothesis that *gender* and *preferred reading* are independent and conclude that the two attributes are (strongly) correlated for the given group of people.

Correlation Coefficient for Numeric Data

For numeric 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

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

where *n* is the number of tuples, a_i and b_i are the respective values of *A* and *B* in tuple *i*, and *B* are the respective mean values of *A* and *B*, σ_A and σ_B are the respective standard deviations of *A* and *B*. $-1 \le r_{A,B} \le +1$.

Note that 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.

Covariance of Numeric Data

In probability theory and statistics, correlation and covariance are two similar measures for assessing howmuch two attributes change together. Consider two numeric attributes *A* and *B*, and a set of *n* observations $\{(a_1,b_1),\ldots,(a_n,b_n)\}$ The mean values of *A* and *B*, respectively, are also known as the **expected values** on *A* and *B*, that is,

$$E(A) = \bar{A} = \frac{\sum_{i=1}^{n} a_i}{n}$$

And

$$E(B) = \bar{B} = \frac{\sum_{i=1}^{n} b_i}{n}.$$

The **covariance** between *A* and *B* is defined as

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

If we compare $r_{A,B}$ (correlation coefficient) covariance, we see that

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

where σ_A and σ_B are the standard deviations of *A* and *B*, respectively. It can also be shown that

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

Table 3.2 Stock Prices for *AllElectronics* and *HighTech*

Time point	*AllElectronics*	*HighTech*
t1	6	20
t2	5	10
t3	4	14
t4	3	5
t5	2	5

mply independence.

Example 3.2 Covariance analysis of numeric attributes. Consider Table 3.2, which presents a simplified example of stock prices observed at five time points for *AllElectronics* and *HighTech*, a high-tech company. If the stocks are affected by the same industry trends, will their prices rise or fall together?

$$E(AllElectronics) = \frac{6+5+4+3+2}{5} = \frac{20}{5} = \$4$$

and

$$E(HighTech) = \frac{20+10+14+5+5}{5} = \frac{54}{5} = \$10.80.$$

Thus, using Eq. (3.4), we compute

$$Cov(AllElectroncis, HighTech) = \frac{6\times 20 + 5\times 10 + 4\times 14 + 3\times 5 + 2\times 5}{5} - 4\times 10.80$$
$$= 50.2 - 43.2 = 7.$$

Therefore, given the positive covariance we can say that stock prices for both companies rise together. ■

Variance is a special case of covariance, where the two attributes are identical (i.e., the covariance of an attribute with itself). Variance was discussed in Chapter 2.

Tuple Duplication

In addition to detecting redundancies between attributes, duplication should also be detected at the tuple level (e.g., where there are two or more identical tuples for a given unique data entry case). The use of denormalized tables (often done to improve performance by avoiding joins) is another source of data redundancy.

Data Value Conflict Detection and Resolution

Data integration also involves the *detection and resolution of data value conflicts*. For example, for the same real-world entity, attribute values from different sources may differ. This may be due to differences in representation, scaling, or encoding. For instance, a *weight* attribute may be stored in metric units in one system and British imperial units in another. For a hotel chain, the *price* of rooms in different cities may involve not only different currencies but also different services (e.g., free breakfast) and taxes. When exchanging information between schools, for example, each school may have its own curriculum and grading scheme. One university may adopt a quarter system, offer three courses on database systems, and assign grades from AC to F, whereas another may adopt a semester system, offer two courses on databases, and assign grades from 1 to 10.

Data Reduction

Data reduction techniques can be applied to obtain a reduced representation of the data set that ismuch 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.

Overview of Data Reduction Strategies

Data reduction strategies include *dimensionality reduction*, *numerosity reduction*, and *data compression*.

Dimensionality reduction is the process of reducing the number of randomvariables or attributes under consideration. Dimensionality reduction methods include *wavelet transforms* and *principal components analysis* which transform or project the original data onto a smaller space. *Attribute subset selection* is a method of dimensionality reduction in which irrelevant, weakly relevant, or redundant attributes or dimensions are detected and removed.

Numerosity reduction techniques replace the original data volume by alternative, smaller forms of data representation. These techniques may be parametric or nonparametric. For *parametric methods*, a model is used to estimate the data, so that typically only the data parameters need to be stored, instead of the actual data. (Outliers may also be stored.) Regression and log-linear models are examples. *Nonparametric methods* for storing reduced representations of the data include *histograms, clustering, sampling,* and *data cube aggregation.*

In **data compression**, transformations are applied so as to obtain a reduced or "compressed" representation of the original data. If the original data can be *reconstructed* from the compressed data without any information loss, the data reduction is called **lossless**. If, instead, we can reconstruct only an approximation of the original data, then the data reduction is called **lossy**.

Wavelet Transforms

The **discrete wavelet transform (DWT)** is a linear signal processing technique that, when applied to a data vector $\boldsymbol{X}$, transforms it to a numerically different vector, $\boldsymbol{X}'$, of **wavelet coefficients**. The two vectors are of the same length. When applying this technique to data reduction, we consider each tuple as an n-dimensional data vector, that is, $X = (x_1,x_2,x_3...x_n)$, depicting n measurements made on the tuple from n database attributes.

"How can this technique be useful for data reduction if the wavelet transformed data are of the same length as the original data?" The usefulness lies in the fact that the wavelet transformed data can be truncated. A compressed approximation of the data can be retained by storing only a small fraction of the strongest of the wavelet coefficients. For example, all wavelet coefficients larger than some user-specified threshold can be retained. All other coefficients are set to 0. The resulting data representation is therefore very sparse, so that operations that can take advantage of data sparsity are computationally very fast if performed in wavelet space.

There is only one DFT, yet there are several families of DWTs. Figure 3.4 shows some wavelet families. Popular wavelet transforms include the Haar-2, Daubechies-4, and Daubechies-6. The general procedure for applying a discrete wavelet transform uses a hierarchical *pyramid algorithm* that halves the data at each iteration, resulting in fast computational speed. The method is as follows:

1. The length, L, of the input data vector must be an integer power of 2. This condition can be met by padding the data vector with zeros as necessary ($L >= n$).

2. Each transform involves applying two functions. The first applies some data smoothing, such as a sum or weighted average. The second performs a weighted difference, which acts to bring out the detailed features of the data.

3. The two functions are applied to pairs of data points in $\boldsymbol{X}$, that is, to all pairs of measurements (x_{2i}, x_{2i+1}). This results in two data sets of length $L/2$. In general, these represent a smoothed or low-frequency version of the input data and the highfrequency content of it, respectively.

4. The two functions are recursively applied to the data sets obtained in the previous loop, until the resulting data sets obtained are of length 2.

5. Selected values from the data sets obtained in the previous iterations are designated the wavelet coefficients of the transformed data.

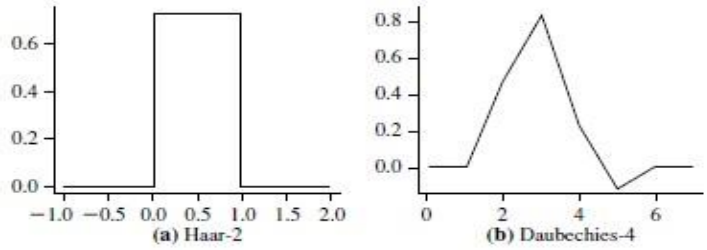


Figure 3.4 Examples of wavelet families. The number next to a wavelet name is the number of *vanishing moments* of the wavelet. This is a set of mathematical relationships that the coefficients must satisfy and is related to the number of coefficients.

Equivalently, a matrix multiplication can be applied to the input data in order to obtain the wavelet coefficients, where the matrix used depends on the given DWT. The matrix must be **orthonormal**, meaning that the columns are unit vectors and are mutually orthogonal, so that the matrix inverse is just its transpose.

Principal Components Analysis

Suppose that the data to be reduced consist of tuples or data vectors described by *n* attributes or dimensions. **Principal components analysis** (**PCA**; also called the Karhunen-Loeve, or K-L, method) searches for *k n*-dimensional orthogonal vectors that can best be used to represent the data, where *k* <= *n*. The original data are thus projected onto a much smaller space, resulting in dimensionality reduction. Unlike attribute subset selection, which reduces the attribute set size by retaining a subset of the initial set of attributes, PCA "combines" the essence of attributes by creating an alternative, smaller set of variables. The initial data can then be projected onto this smaller set. PCA often reveals relationships that were not previously suspected and thereby allows interpretations that would not ordinarily result.

The basic procedure is as follows:

1. The input data are normalized, so that each attribute falls within the same range. This step helps ensure that attributes with large domains will not dominate attributes with smaller domains.

2. PCA computes *k* orthonormal vectors that provide a basis for the normalized input data. These are unit vectors that each point in a direction perpendicular to the others. These vectors are referred to as the *principal components*. The input data are a linear combination of the principal components.

3. The principal components are sorted in order of decreasing "significance" or strength. The principal components essentially serve as a new set of axes for the data, providing important information about variance. That is, the sorted axes are such that the first axis shows the most variance among the data, the second axis shows the next highest variance, and so on. For example, Figure 3.5 shows the first two principal components, $\boldsymbol{Y}_1$ and $\boldsymbol{Y}_2$, for the given set of data originally mapped to the axes X_1 and X_2. This information helps identify groups or patterns within the data.

4. Because the components are sorted in decreasing order of "significance," the data size can be reduced by eliminating the weaker components, that is, those with low variance. Using the strongest principal components, it should be possible to reconstruct a good approximation of the original data.

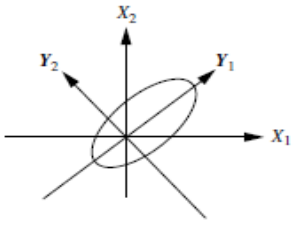


Figure 3.5 Principal components analysis. $\boldsymbol{Y}_1$ and $\boldsymbol{Y}_2$ are the first two principal components for the given data.

d data. Multidimensional rincipal components may be used as inputs to multiple regression and cluster analysis.

Attribute Subset Selection

Attribute subset selection[4] 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.

"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 prohibitively expensive, especially as n and the number of data classes increase.

The "best" (and "worst") attributes are typically determined using tests of statistical significance, which assume that the attributes are independent of one another. Many other attribute evaluation measures can be used such as the *information gain* measure used in building decision trees for classification

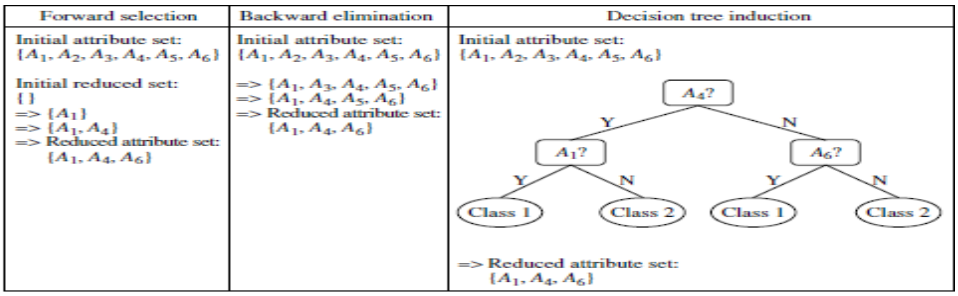


Figure 3.6 Greedy (heuristic) methods for attribute subset selection.

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 remaining attributes.

4. Decision tree induction: Decision tree algorithms (e.g., ID3, C4.5, and CART) were originally intended for classification. Decision tree induction constructs a flowchartlike 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.

Regression and Log-Linear Models: Parametric Data Reduction

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

$$y = wx + b,$$

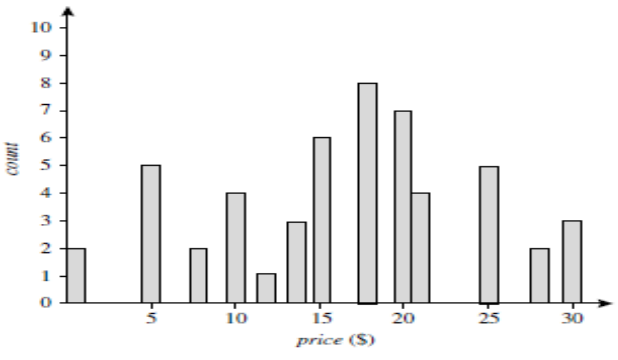
where the variance of y is assumed to be constant. In the context of data mining, x and y are numeric database attributes. The coefficients, w and b (called *regression coefficients*), specify the slope of the line and the y-intercept, respectively. These coefficients can be solved for by the *method 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 distributions. 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.

Histograms

A **histogram** for an attribute, A, partitions the data distribution of A into disjoint subsets, referred to as *buckets* or *bins*. 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.

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



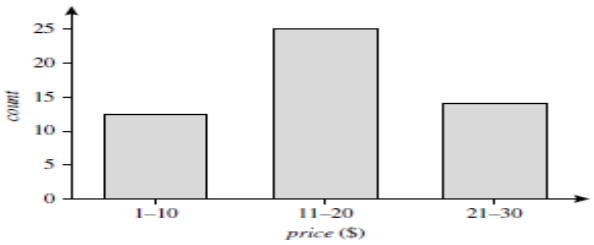


Figure 3.8 An equal-width histogram for *price*, where values are aggregated so that each bucket has a uniform width of $10.

"How are the buckets determined and the attribute values partitioned?" There are several partitioning rules, including the following:

Equal-width: In an equal-width histogram, the width of each bucket range is uniform (e.g., the width of $10 for the buckets in Figure 3.8).

Equal-frequency (or equal-depth): In an equal-frequency histogram, the buckets are created so that, roughly, the frequency of each bucket is constant

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.

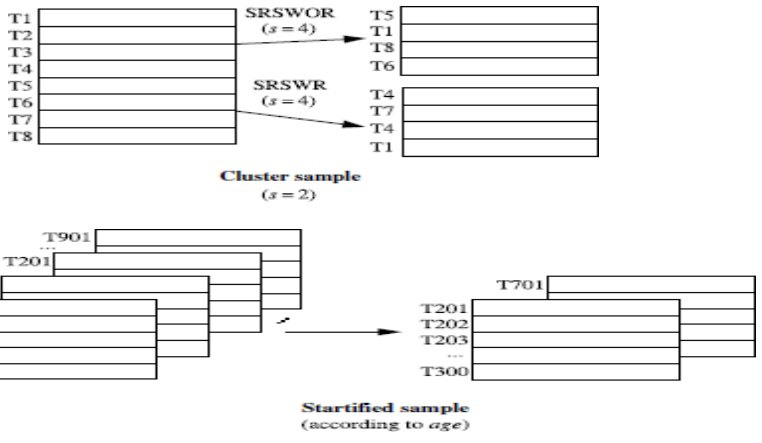
Sampling

Sampling can be used as a data reduction technique because it allows a large data set to be represented by a much smaller random data sample (or subset). Suppose that a large data set, *D*, contains *N* tuples. Let's look at the most common ways that we could sample*D* for data reduction,

Simple random sample without replacement (SRSWOR) of size *s*: This is created by drawing *s* of the *N* tuples from *D* ($s < N$), where the probability of drawing any tuple in *D* is $1/N$, that is, all tuples are equally likely to be sampled.

Simple random sample with replacement (SRSWR) of size *s*: This is similar to SRSWOR, except that each time a tuple is drawn from *D*, it is recorded and then *replaced*. That is, after a tuple is drawn, it is placed back in *D* so that it may be drawn again.

Cluster sample: If the tuples in *D* are grouped into *M* mutually disjoint "clusters," then an SRS of *s* clusters can be obtained, where $s < M$. For example, tuples in a database are usually retrieved a page at a time, so that each page can be considered a cluster. A reduced data representation can be obtained by applying, say, SRSWOR to the pages, resulting in a cluster sample of the tuples.



T38	youth
T256	youth
T307	youth
T391	youth
T96	middle_aged
T117	middle_aged
T138	middle_aged
T263	middle_aged
T290	middle_aged
T308	middle_aged
T326	middle_aged
T387	middle_aged
T69	senior
T284	senior

T38	youth
T391	youth
T117	middle_aged
T138	middle_aged
T290	middle_aged
T326	middle_aged
T69	senior

Figure 3.9 Sampling can be used for data reduction.

Stratified sample: If *D* is divided intomutually disjoint parts called *strata*, a stratified sample of *D* is generated by obtaining an SRS at each stratum. This helps ensure a representative sample, especially when the data are skewed. For example, a stratified sample may be obtained fromcustomer data, where a stratum is created for each customer age group.

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 2008 to 2010. 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.

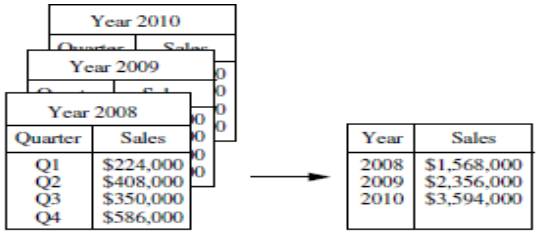


Figure 3.10 Sales data for a given branch of *AllElectronics* for the years 2008 through 2010. On the *left*, the sales are shown per quarter. On the *right*, the data are aggregated to provide the annual sales.

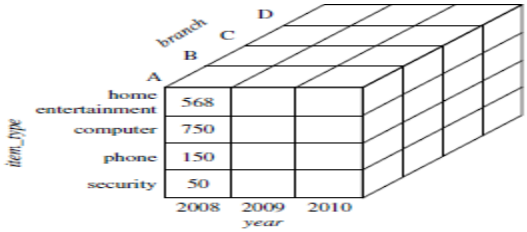


Figure 3.11 A data cube for sales at *AllElectronics*.

Data Transformation and Data Discretization

In this preprocessing step, the data are transformed or consolidated so that the resulting mining process may be more efficient, and the patterns found may be easier to understand. Data discretization, a form of data transformation.

Data Transformation Strategies Overview

In *data transformation*, the data are transformed or consolidated into forms appropriate for mining. Strategies for data transformation include the following:

1. Smoothing, which works to remove noise fromthe data. Techniques include binning, regression, and clustering.

2. Attribute construction (or *feature construction*), where new attributes are constructed and added from the given set of attributes to help the mining process.

3. 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 data analysis at multiple abstraction levels.

4. Normalization, where the attribute data are scaled so as to fall within a smaller range, such as -1.0 to 1.0, or 0.0 to 1.0.
5. Discretization, where the raw values of a numeric attribute (e.g., *age*) are replaced by interval labels (e.g., 0–10, 11–20, etc.) or conceptual labels (e.g., *youth*, *adult*, *senior*). The labels, in turn, can be recursively organized into higher-level concepts, resulting in a *concept hierarchy* for the numeric attribute. Figure 3.12 shows a concept hierarchy for the attribute *price*. More than one concept hierarchy can be defined for the same
attribute to accommodate the needs of various users.
6. Concept hierarchy generation for nominal data, where attributes such as *street* can be generalized to higher-level concepts, like *city* or *country*. Many hierarchies for nominal attributes are implicit within the database schema and can be automatically defined at the schema definition level.

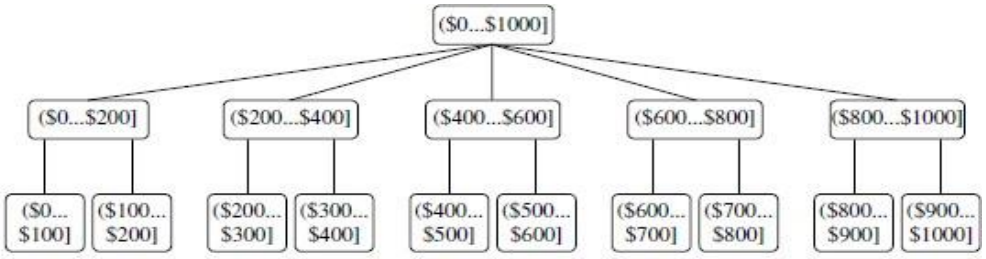


Figure 3.12 A concept hierarchy for the attribute *price*, where an interval ($X ... $Y] denotes the range from $X (exclusive) to $Y (inclusive).

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*.
Data discretization and concept hierarchy generation are also forms of data reduction. The raw data are replaced by a smaller number of interval or concept labels. This simplifies the original data and makes the mining more efficient.

Data Transformation by Normalization

The measurement unit used can affect the data analysis. For example, changing measurement units from meters to inches for *height*, or fromkilograms to pounds for *weight*, may lead to very different results. To help avoid dependence on the choice of measurement units, the data should be *normalized* or *standardized*. This involves transforming the data to fall within a smaller or common range such as [-1, 1] or [0.0, 1.0].
Normalizing the data attempts to give all attributes an equal weight. 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.
There are many methods for data normalization. We study *min-max normalization*, *z-score normalization*, and *normalization by decimal scaling.* For our discussion, let *A* be a numeric attribute with *n* observed values, $v_1, v_2, ..., v_n$.
Min-max normalization performs a linear transformation on the original data. Suppose that min_A and max_A are the minimum and maximum values of an attribute, *A*. Min-max normalization maps a value, v_i , of *A* to v_i' in the range $[new\_min_A, new\_max_A]$ by computing

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

Min-max normalization preserves the relationships among the original data values.

Example 3.4 Min-max normalization. Suppose that the minimum and maximum values for the attribute *income* are $12,000 and $98,000, respectively. We would like to map *income* to the range [0.0, 1.0]. By min-max normalization, a value of $73,600 for *income* is transformed to $\frac{73{,}600 - 12{,}000}{98{,}000 - 12{,}000}(1.0 - 0) + 0 = 0.716.$

In ***z*-score normalization** (or *zero-mean normalization*), the values for an attribute, *A*, are normalized based on the mean (i.e., average) and standard deviation of *A*. A value, v_i , of *A* is normalized to v_i' by computing

$$v_i' = \frac{v_i - \bar{A}}{\sigma_A},$$

where and σ_A are the mean and standard deviation, respectively, of attribute A.

Example 3.5 z-score normalization. Suppose that the mean and standard deviation of the values for the attribute *income* are $54,000 and $16,000, respectively. With z-score normalization, a value of $73,600 for *income* is transformed to $\frac{73,600 - 54,000}{16,000} = 1.225.$

A variation of this z-score normalization replaces the standard deviation of above Eq. by the *mean absolute deviation* of A. The *mean absolute deviation* of A, denoted s_A, is

$$s_A = \frac{1}{n}(|v_1 - \bar{A}| + |v_2 - \bar{A}| + \cdots + |v_n - \bar{A}|).$$

Thus, z-score normalization using the mean absolute deviation is

$$v_i' = \frac{v_i - \bar{A}}{s_A}.$$

The mean absolute deviation, s_A, is more robust to outliers than the standard deviation, σ_A.

Normalization by decimal scaling normalizes by moving the decimal point of values of attribute A. The number of decimal points moved depends on the maximum absolute value of A. A value, v_i, of A is normalized to v_i' by computing

$$v_i' = \frac{v_i}{10^j},$$

where j is the smallest integer such that $max(|v_i'|) < 1.$

Example 3.6 Decimal scaling. Suppose that the recorded values of A range from -986 to 917. The maximum absolute value of A is 986. To normalize by decimal scaling, we therefore divide each value by 1000 (i.e., j = 3) so that -986 normalizes to -0.986 and 917 normalizes to 0.917.

Discretization by Binning

Binning is a top-down splitting technique based on a specified number of bins. These methods are also used as discretization methods for data 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 to generate concept hierarchies. Binning does not use class information and is therefore an unsupervised discretization technique.

Discretization by Histogram Analysis

Like binning, histogram analysis is an unsupervised discretization technique because it does not use class information. A histogram partitions the values of an attribute, A, into disjoint ranges called *buckets* or *bins*.

Various partitioning rules can be used to define histograms . In an *equal-width* histogram, for example, the values are partitioned into equal-size partitions or ranges (e.g., earlier in Figure 3.8 for *price*, where each bucket has a width of $10). With an *equal-frequency* histogram, the values are partitioned so that, ideally, each partition contains the same number of data tuples.

Discretization by Cluster, Decision Tree, and Correlation Analyses

Cluster analysis is a popular data discretization method. A clustering algorithm can be applied to discretize a numeric attribute, A, by partitioning the values of A into clusters or groups.

Clustering can be used to generate a concept hierarchy for A by following either a top-down splitting strategy or a bottom-up merging strategy, where each cluster forms a node of the concept hierarchy.

Techniques to generate decision trees for classification can be applied to discretization. Such techniques employ a top-down splitting approach. Unlike the other methods mentioned so far, decision tree approaches to discretization are

supervised, that is, they make use of class label information. For example, we may have a data set of patient symptoms (the attributes) where each patient has an associated *diagnosis* class label. *Entropy* is the most commonly used measure for this purpose. To discretize a numeric attribute, *A*, the method selects the value of *A* that has the minimum entropy as a split-point, and recursively partitions the resulting intervals to arrive at a hierarchical discretization. Such discretization forms a concept hierarchy for *A*. Because decision tree–based discretization uses class information, it is more likely that the interval boundaries (split-points) are defined to occur in places that may help improve classification accuracy.

Measures of correlation can be used for discretization. *ChiMerge* is a χ^2-based discretization method. The discretization methods that we have studied up to this point have all employed a top-down, splitting strategy. This contrasts with ChiMerge, which employs a bottom-up approach by finding the best neighboring intervals and then merging them to form larger intervals, recursively. As with decision tree analysis, ChiMerge is supervised in that it uses class information.

Concept Hierarchy Generation for Nominal Data

Nominal attributes have a finite (but possibly large) number of distinct values, with no ordering among the values. Examples include *geographic_location*, *job_category*, and *item_type*.

1. **Specification of a partial ordering of attributes explicitly at the schema level by users or experts:** Concept hierarchies for nominal 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. 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*.
2. **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.

3. 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 systemcan 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 nominal attributes be found?" Consider the 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*).

Example 3.7 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*—fromthe *AllElectronics* database, but does not specify the hierarchical ordering among the attributes.

4. Specification of only a partial set of attributes: Sometimes a user can be careless 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.

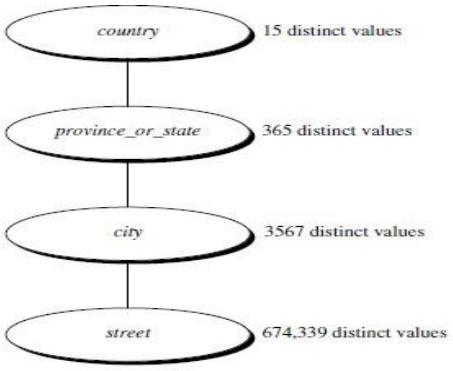


Figure 3.13 Automatic generation of a schema concept hierarchy based on the number of distinct attribute values.

Example 3.8 Concept hierarchy generation using prespecified semantic connections. Suppose that a data mining expert (serving as an administrator) has pinned together the five attributes *number*, *street*, *city*, *province or state*, and*country*, because they are closely linked semantically regarding the notion of *location*. If a user were to specify only the attribute *city* for a hierarchy defining *location*, the systemcan automatically drag in all five semantically related attributes to form a hierarchy. The user may choose to drop any of these attributes (e.g., *number* and *street*) from the hierarchy, keeping *city* as the lowest conceptual level.