3

Data

In this Chapter, we discuss attribute and data types, data storage, and problems of quantity and quality of data.

1. Introduction

The outcome of data mining and knowledge discovery heavily depends on the quality and quantity of available data. Before we discuss data analysis methods, data organization and related issues need to be addressed first. This Chapter focuses on three issues: **data types, data storage techniques**, and **amount and quality of the data**. These topics form the necessary background for subsequent knowledge discovery process steps such as data preprocessing, data mining, representation of generated knowledge, and assessment of generated models. Upon finishing this Chapter, the reader should be able to understand problems associated with available data.

2. Attributes, Data Sets, and Data Storage

Data can have diverse formats and can be stored using a variety of different storage modes. At the most elementary level, a single unit of information is a **value** of a **feature/attribute**, where each feature can take a number of different values. The objects, described by features, are combined to form **data sets**, which in turn are stored as **flat** (rectangular) **files** and in other formats using **databases** and **data warehouses**. The relationships among the above concepts are depicted in Figure 3.1.

In what follows, we provide a detailed explanation of the terminology and concepts introduced above.

2.1. Values, Features, and Objects

There are two key types of values: **numerical** and **symbolic**. Numerical values are expressed by numbers, for instance, real numbers (–1.09, 123.5), integers (1, 44, 125), prime numbers (1, 3, 5), etc. In contrast, symbolic values usually describe qualitative concepts such as colors (white, red) or sizes (small, medium, big).

Features (also known as attributes) are usually described by a set of corresponding values. For instance, height is usually expressed as a set of real numbers. Features described by both numerical and symbolic values can be either **discrete** (categorical) or **continuous**. Discrete features concern a situation in which the total number of values is relatively small (finite), while with continuous features the total number of values is very large (infinite) and covers a specific interval (range).

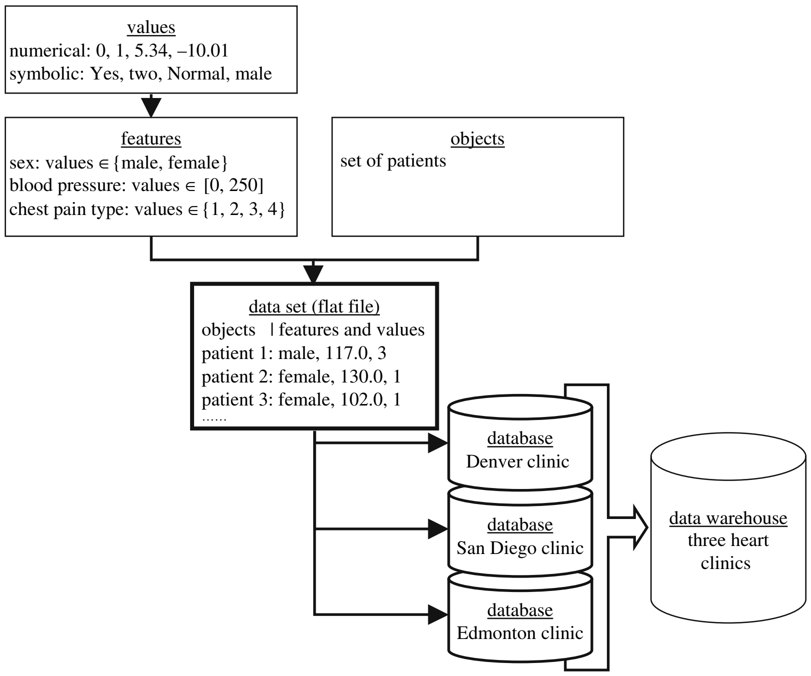
values
numerical: 0, 1, 5.34, –10.01
symbolic: Yes, two, Normal, male
features
sex: values ∈ {male, female}
blood pressure: values ∈ [0, 250]
chest pain type: values ∈ {1, 2, 3, 4}
objects
set of patients
data set (flat file)
objects | features and values
patient 1: male, 117.0, 3
patient 2: female, 130.0, 1
patient 3: female, 102.0, 1
……
database
Denver clinic
database
San Diego clinic
database
Edmonton clinic
data warehouse
three heart
clinics

Figure 3.1. Relationships between values, features, objects, data sets, databases and data warehouses.

A special case of a discrete feature is the **binary** (dichotomous) feature, for which there are only two distinct values. A **nominal** (polytomous) feature implies that there is no natural ordering among its values, while an **ordinal** feature implies that some ordering exists. The values for a given feature can be organized as sets, vectors, or arrays. This categorization of data is important for practical reasons. For instance, some preprocessing and data mining methods are only applicable to data described by discrete features. In those cases a process called **discretization** (see Chapter 8) becomes a necessary preprocessing step to transform continuous features into discrete ones, and this step must be completed before the data mining step is performed.

Objects (also known as records, examples, units, cases, individuals, data points) represent entities described by one or more features. The term *multivariate data* refers to situation in which an object is described by many features, while with *univariate data* a single feature describes an object.

Let us consider an example concerning patients at a heart disease clinic. A patient is an object that can be described by a number of features, such as name, sex, age, diagnostic test results such as blood pressure, cholesterol level, and qualitative evaluations like chest pain and its severity type. An example of a "patient" object is shown in Figure 3.2.

An important issue, from the point of view of the knowledge discovery process, is how different types of features and values are manipulated. In particular, any operation on multiple objects, such as the comparison of feature values or the computation of distance, should be carefully analyzed and designed. For instance, the symbolic value "two" usually cannot be compared with the numerical value 2, unless some conversion is performed. Although computation of the distance between two numerical values is straightforward, performing the same computation between two nominal values (such as "white" and "red", or "chest pain of typ 1" and "chest pain of type 4") requires special attention. In other words, how can we measure distance between colors (this could be done using chromaticity diagrams) or distance between chest pain types (this is much more difficult)? In some cases, it might be impossible to calculate such a distance.

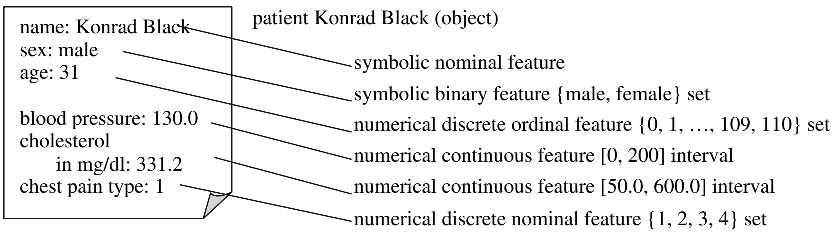
name: Konrad Black
sex: male
age: 31
blood pressure: 130.0
cholesterol in mg/dl: 331.2
chest pain type: 1
patient Konrad Black (object)
symbolic nominal feature
symbolic binary feature {male, female} set
numerical discrete ordinal feature {0, 1, …, 109, 110} set
numerical continuous feature [0, 200] interval
numerical continuous feature [50.0, 600.0] interval
numerical discrete nominal feature {1, 2, 3, 4} set

Figure 3.2. Patient record (object).

An important issue concerning data is the limited comprehension of numbers by humans, who are the ultimate users of the (generated) knowledge. For instance, most people will not comprehend a cholesterol value of 331.2, while they can easily understand the meaning when this numerical value is expressed in terms of aggregated information such as a "high" or "low" level of cholesterol. In other words, information is often "granulated" and represented at a higher level of abstraction (aggregation). In a similar manner, operations or relationships between features can be quantified on an aggregated level. In general, **information granulation** means encapsulation of numeric values into single conceptual entities (see Chapter 5). Examples include encapsulation of elements by sets, or encapsulation of intervals by numbers. Understanding of the concept of encapsulation, also referred to as a *discovery window*, is very important in the framework of knowledge discovery. Continuing with our cholesterol example, we may be satisfied with a single numerical value, say 331.2, which expresses the highest level of granularity (Figure 3.3a). Alternatively, we may want to define this value as belonging to an interval [300, 400], the next, lower, granularity level, which captures meaning of the "high" value of cholesterol (Figure 3.3b). Through the refinement of the discovery window, we can change the "crisp" character of the word "high" by using notion of fuzzy sets (Figure 3.3c) or rough sets (Figure 3.3d). In the case of fuzzy sets, we express the cholesterol value as being high to some degree, or being normal to some other degree. The lowest level of granularity (i.e., the highest generality) occurs when the discovery window covers the entire spectrum of values, which implies that the knowledge discovery process focuses on the entire data set.

2.2. Data Sets

Objects described by the same features are grouped to form **data sets**. Many data mining and statistical data analysis tools assume that data sets are organized as **flat files,** in a rectangularly formatted table composed of rows and columns. The rows represent objects and the columns represent features, resulting in a flat file that forms a two-dimensional array. Flat files are used to store data in a simple text file format, and they are often generated from data stored in other, more complex formats, such as spreadsheets or databases.

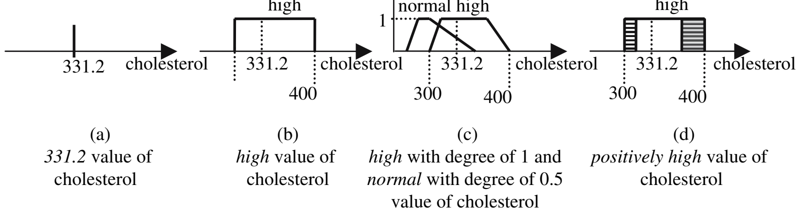
331.2
cholesterol
high
331.2
cholesterol
400
1
normal high
331.2
cholesterol
300
400
high
331.2
cholesterol
300
400
(a)
331.2 value of cholesterol
(b)
high value of cholesterol
(c)
high with degree of 1 and *normal* with degree of 0.5 value of cholesterol
(d)
positively high value of cholesterol

Figure 3.3. Information granularization methods. (a) Numerical; (b) interval based; (c) fuzzy set based; (d) rough set based.

Table 3.1. Flat file data set for heart clinic patients.

Name	Age	Sex	Blood pressure	Blood pressure test date	Cholesterol in mg/dl	Cholesterol test date	Chest pain type	Defect type	Diagnosis
Konrad Black	31	male	130.0	05/05/2005	NULL	NULL	NULL	NULL	NULL
Konrad Black	31	male	130.0	05/05/2005	331.2	05/21/2005	1	normal	absent
Magda Doe	26	female	115.0	01/03/2002	NULL	NULL	4	fixed	present
Magda Doe	26	female	115.0	01/03/2002	407.5	06/22/2005	NULL	NULL	NULL
Anna White	56	female	120.0	12/30/1999	45.0	12/30/1999	2	normal	absent
…	…	…	…	…	…	…	…	…	…

To illustrate, let us design a data set of heart patients, shown in Table 3.1. Each patient (object) is described by the following set of features:

– *name* (symbolic nominal feature)
– *age* (numerical discrete ordinal feature from the {0, 1, …, 109, 110} set)
– *sex* (symbolic binary feature from the {male, female} set)
– *blood pressure* (numerical continuous feature from the [0, 200] interval)
– *blood pressure test date* (date type feature)
– *cholesterol in mg/dl* (numerical continuous feature from the [50.0, 600.0] interval)
– *cholesterol test date* (date type feature)
– *chest pain type* (numerical discrete nominal feature from the {1, 2, 3, 4} set)
– *defect type* (symbolic nominal feature from the {normal, fixed, reversible} set)
– *diagnosis* (symbolic binary feature from the {present, absent} set)

First, let us note that a new feature type, namely, date, has been introduced. Date can be treated as a numerical continuous feature (after some conversion) but usually is stored in a customary format like mm/dd/yyyy, which is between being numerical and symbolic. A number of other special feature types, such as text, image, video, etc., are briefly described in the next section. Second, a new NULL value has been introduced. This value indicates that the corresponding feature is unknown (not measured or missing) for the object. Third, we observe that it is possible that several different objects are related to the same patient. For instance, Konrad Black first came to get a blood pressure test and later came to do the remaining tests and was diagnosed. Finally, for the object associated with Anna White, the cholesterol value is 45.0, which is outside the interval defined/acceptable for this feature, and thus we may suspect that the value may be incorrect, e.g., it may be that 45.0 was entered instead of the correct value of 450.0. These observations regarding different data types and missing and erroneous data lead to certain difficulties associated with the knowledge discovery process, which are described later in the Chapter.

2.3. Data Storage: Databases and Data Warehouses

Although flat files are a popular format in which to store data, data mining tools used in the knowledge discovery process can be applied to a great variety of other data formats, such as those found in databases, data warehouses, advanced database systems, and the World Wide Web (WWW). Advanced database systems include object-oriented and object-relational databases, as well as data-specific databases, such as transactional, spatial, temporal, text, and multimedia databases.

In the knowledge discovery process we often deal with large data sets that require special data storage and management systems. There are four reasons for using a specialized system:

1. *The corresponding data set may not fit into the memory of a computer used for data mining*, and thus a data management system is used to fetch the data.
2. *The data mining methods may need to work on many different subsets of data*, and therefore a data management system is required to efficiently retrieve the required pieces of data.
3. *The data may need to be dynamically added and updated, sometimes by different people in different locations.* A data management system is required to handle updating (and also offers failure and recovery options).
4. *The flat file may include significant portions of redundant information*, which can be avoided if a single data set is stored in multiple tables (a characteristic feature of data storage and management systems).

A poll (September 2005) performed by KDNuggets (http://www.kdnuggets. com) asked which data formats the users had analyzed in the last 12 months. The results showed that flat files, which are often extracted from relational databases, were used 26% of the time, time series (temporal databases) 13%, text databases 11%, transactional databases 10%, and WWW clickstream data, spatial databases, and WWW content data 5% each. The results indicated that although flat files were still the most often used, other formats gained significant interest and therefore are discussed below.

2.3.1. Databases

The above mentioned reasons result in a need to use a specialized system, a **DataBase Management System** (DBMS). It consists of a **database** that stores the data, and a set of programs for management and fast access to the database. The software provides numerous services, such as the ability to define the structure (**schema**) of the database, to store the data, to access the data in concurrent ways (several users may access the data at the same time) and distributed ways (the data are stored in different locations), and to ensure the security and consistency of the stored data (for instance, to protect against unauthorized access or a system crash). The most common database type is a **relational database**, which consists of a set of tables. Each table is rectangular and can be perceived as being analogous to a single flat file. At the same time, the database terminology is slightly different than that used in data mining. The tables consist of **attributes** (also called columns or fields) and **tuples** (also called records and rows). Most importantly, each table is assigned a unique name, and each tuple in a table is assigned a special attribute, called a **key**, that defines its unique identifiers. Relational databases also include the **entity-relational** (ER) data model, which defines a set of entities (tables, tuples, etc.) and their relationships. Next, we define a relational database for a heart clinic, which will extend our flat file–based data storage and allow better understanding of the above concepts. In order to better utilize the capabilities of a relational database, our original flat file is divided into four relational tables. These include the table *patient*, which stores basic information about patients together with their diagnostic information (the key is the *patient ID* attribute), and two tables *blood_pressure_test* and *cholesterol_test* that store the corresponding test values (see Figure 3.4). The fourth table, *performed_tests*, represents the relationship between multiple other tables. In the case of our relational database, the relationship shows which patients had which tests.

Analysis of the above example shows that the use of multiple tables results in the removal of redundant information included in the flat file. At the same time, the data are divided into smaller blocks and thus are easier to manipulate and fit into the available memory. Another important feature of a DBMS is the availability of a specialized language, called **Structured Query Language** (SQL), that aims to provide fast and convenient access to portions of the entire database. For instance, we may want to extract information about the tests performed between specific dates or obtain a list of all patients who have been diagnosed with a certain diagnosis *patient*

Patient ID	Name	Age	Sex	Chest pain type	Defect type	Diagnosis
P1	Konrad Black	31	male	1	normal	absent
P2	Magda Doe	26	female	4	fixed	present
P3	Anna White	56	female	2	normal	absent
…	…	…	…	…	…	…

blood_pressure_test

Blood pressure test ID	Patient ID	Blood pressure	Blood pressure test date
BPT1	P1	130.0	05/05/2005
BPT2	P2	115.0	01/03/2002
BPT3	P3	120.0	12/30/1999
…	…	…	…

cholesterol_test

Cholesterol test ID	Patient ID	Cholesterol in mg/dl	Cholesterol test date
SCT1	P1	331.2	05/21/2005
SCT2	P2	407.5	06/22/2005
SCT3	P3	45.0	12/30/1999
…	…	…	…

performed_tests

Patient ID	Blood pressure test	Cholesterol test
P1	BPT1	SCT1
P2	BPT2	SCT2
P3	BPT3	SCT3
…	…	…

Figure 3.4. Relational database for heart clinic patients

type. This process is relatively simple when a DBMS and an SQL are used. In contrast, with a flat file, the user himself is forced to manipulate the data to select the desired portion – a process that can be tedious and difficult to perform.

SQL allows the user to specify queries that contain a list of relevant attributes and constraints on those attributes. Oftentimes, DBMSs provide a graphical user interface to facilitate query formulation. The user's query is automatically transformed into a set of relational operations, such as join, selection, and projection, optimized for time – and/or resource–efficient processing and executed by the DBMS (see Chapter 6). SQL also provides the ability to aggregate data by computing functions, such as summations, average, count, maximum, and minimum. This ability allows the user to receive answers to more complex and interesting queries. For instance, the user could ask the DBMS to compute how many blood pressure tests were performed in a particular month, or what the average blood pressure is for female patients.

2.3.2. Data Warehouses

Now we consider a more elaborate scenario in which a number of heart clinics in different cities, each having its own database, belong to the same company. Using a database, we are able to analyze (mine) data in individual clinics, but it can be very difficult to perform analysis across all clinics. In this case, a **data warehouse**, which is a repository of data collected in different locations (relational databases) and stored using a unified schema, is used. Data warehouses are usually created by applying a set of processing steps to data coming from multiple databases. The steps usually include data cleaning, data transformation, data integration, data loading, and periodical data update (see Figure 3.5).

While the main purpose of a database is storage of the data, the main purpose of a data warehouse is analysis of the data. Consequently, the data in a data warehouse are organized around a set of subjects of interest to the user. For instance, in case of the heart clinic, these subjects could be patients, clinical test types, and diagnoses. The analysis is performed to provide information

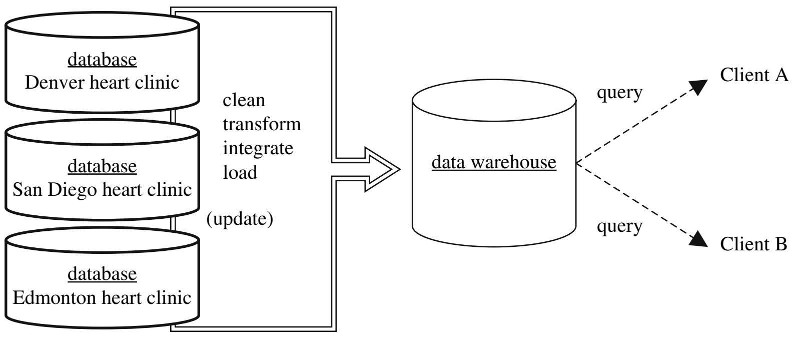

Figure 3.5. Typical architecture of a data warehouse system.

(knowledge) from a historical perspective. For instance, we could ask for a breakdown of the most performed clinical tests in the past five years. Such requests (queries) require the availability of summarized information, and therefore data warehouses do not store the same content as the source databases but rather a summary of these data. For instance, a data warehouse may not store individual blood pressure test values, but rather the number of tests performed by each clinic, the time interval (say a month), and a given set of patient age ranges.

A data warehouse usually uses a multidimensional database structure, where each dimension corresponds to an attribute or a set of attributes selected by the user to be included in the schema. Each cell in the database corresponds to some summarized (aggregated) measure, such as average, count, minimum, etc. The actual implementation of the warehouse can be a relational database or a multidimensional **data cube**. The latter structure provides a three-dimensional view of the data and allows for fast access to the summarized data via precomputation. An example of the data cube for the heart clinic's data warehouse is shown in Figure 3.6. The cube has three dimensions: clinic (Denver, San Diego, Edmonton), time (expressed in months), and age range (0–8, 9–21, 21–45, 45–65, over 65). The values are in thousands and show how many blood pressure tests were performed. Each dimension can be further summarized, e.g., we can collapse months into quarters and can convert age into some numeric intervals (ranges), say, intervals of values within 0–45 and those over 45.

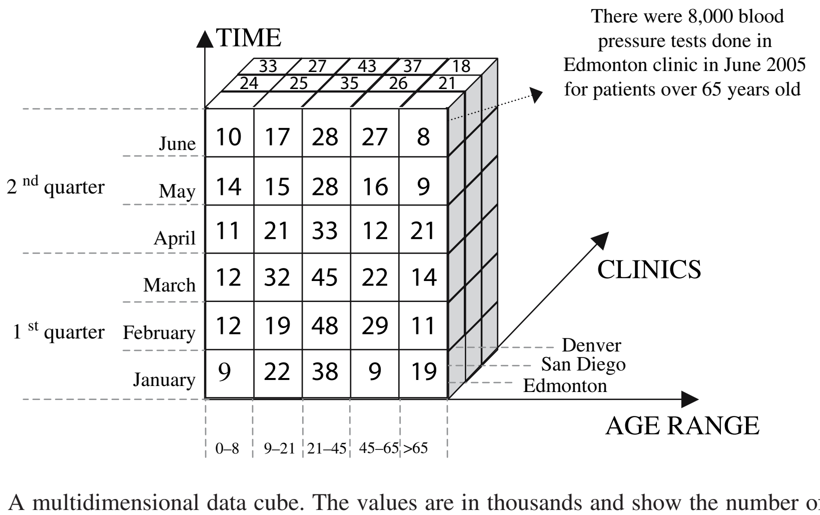

Figure 3.6. A multidimensional data cube. The values are in thousands and show the number of performed blood pressure tests. For readability, only some of the cube cells are shown.

The availability of multidimensional data and precomputation gave rise to **On-Line Analytical Processing** (**OLAP**). OLAP makes use of some background knowledge about the domain (which concerns the data in hand) to present data at different levels of abstraction. Two commonly used OLAP operations are roll-up and drill-down. The first operation merges data at one or more dimensions to provide the user with a higher–level summarization, while the latter breaks down data into subranges to present the user with more detailed information. An example for the heart clinics warehouse is shown in Figure 3.7.

A detailed discussion of data warehouses and OLAP is provided in Chapter 6.

2.4. Advanced Data Storage

While relational databases (warehouses) are usually used by businesses like retail stores and banks, other more specialized and advanced database systems have emerged in recent years. The new breed of databases satisfies the needs of more specialized users who must handle more than just numerical and nominal data. The new databases handle **transactional** data; **spatial** data, such as maps; **hypertext**, such as HTML and XML; **multimedia** data, such as combinations of text, image, video and audio; **temporal** data, such as time–related series concerning stock exchange and historical records; and the **WWW**, which is enormously large and distributed (and accessible via the Internet). The special data types require equally specialized databases that utilize efficient data structures and methods for handling operations on such complex data structures. The challenges for databases are to cope with variable-length objects, structured and semi-structured data, unstructured text in various languages, multimedia data formats, and very large amounts of data.

Due to these challenges, numerous specialized databases have been developed. These include object-oriented and object-relational databases, spatial databases, temporal databases, text databases, multimedia databases, and the WWW. Each of these databases not only allows the user to store data in the corresponding format but also provides facilities to efficiently store, update,

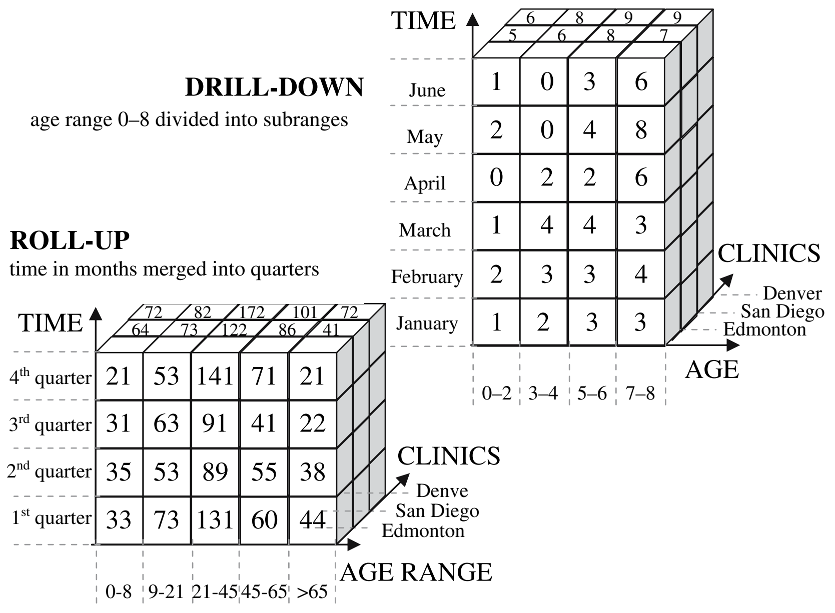

Figure 3.7. Results of the drill-down and roll-up operations for the data cube shown in Figure 3.6. The values are in thousands; for drill-down, we subdivide the age range dimension; for roll-up, we merge the time dimension. For readability, only some of the cube cells are shown.

and retrieve large amount of data. Below we briefly introduce the major types of specialized database systems.

2.4.1. Object-oriented Databases

Object-oriented databases are based on the object-oriented programming paradigm, which treats each stored entity as an object (this object is different than the object or record described earlier in this Chapter). The object encapsulates a set of **variables** (which provide its description), a set of **messages** that the object uses to communicate with other objects, and a set of **methods** that contain code to implement the messages. For instance, in the heart clinic example, the object could be a patient, its variables could be name, address, sex, etc., and methods could implement messages that, for instance, would retrieve particular test values for particular times. The key element of object-oriented databases is the ability to group similar (or identical) objects into classes, which can be organized into hierarchies. For instance, the *patient* class would have the variables name, address, and sex, and its instances would be particular patients. The patient class could have a subclass called *retired patient*, which would inherit all variables of the patient class, but would also have new variables, such as date of home release, that pertain to patients who are not longer under treatment. The inheritance between the classes in a hierarchy allows for better information sharing.

2.4.2. Object-relational Databases

Object-relational databases are based on the object-relational data model. In essence, this is a relational database that has been extended by providing a set of complex data types to handle complex objects. This extension requires availability of a specialized query language to retrieve the complex data from the database. Most common extensions include the ability to handle data types such as trees, graphs, lists, class hierarchies, and inheritance.

2.4.3. Transactional Databases

Transactional databases are stored as flat files and consist of records that represent **transactions**. A transaction includes a unique identifier and a set of **items** that make up the transaction. One example of a transaction is a record of a purchase in a store, which consists of a list of purchased items, the purchase identifier, and some other information about the sale. The additional information is usually stored in a separate file, and may include customer name, cashier name, date, store branch, etc. An example transactional database is shown in Figure 3.8.

The single most important difference between the relational and the transactional databases is that the latter store a set of items (values), rather than a set of values of the related features. A transactional database stores information about the presence/absence of an item, while a relational database stores information about specific feature values that an example (item) possesses. The transactional databases are often used to identify sets of items that frequently coexist in transactions. For instance, given a transactional database for the heart clinic that stores names of tests as items, and where each transaction denotes a single patient's visit to the clinic, we can find out which tests are often performed together. Based on the results, we can modify business practice in a clinic to prescribe these tests together instead of prescribing them separately. Transactional data and corresponding data mining algorithms are described in Chapter 10.

sales

Transation ID	Set of item IDs
TR000001	Item1, Item32, Item52, Item71
TR000002	Item2, Item3, Item4, Item57, Item 92, Item93
TR000003	Item11, Item101
…	…

Figure 3.8. Example transactional database

2.4.4. Spatial Databases

Spatial databases are designed to handle spatially related data. Example data includes geographical maps, satellite images, medical images, and VLSI chip-design data. Spatial data can be represented in two ways: in a **raster format** and in a **vector format**. The raster format concerns using *n*–dimensional pixel maps, while the vector format requires representing all considered objects as simple geometrical objects, such as lines, triangles, polygons, etc., and using vector-based geometry to compute relations between the objects. Spatial databases allow the user to obtain unique information about the stored data. For instance, for the heart clinic, we can ask whether patients living in a specific neighborhood are more susceptible to certain heart conditions, or whether the clinic accepts more patients from the north side than from the south side of town.

2.4.5. Temporal Databases

Temporal databases (also called time-series databases) are used to store time-related data. Similarly, as in the case of object-relational databases, the temporal databases extend the relational databases to handle time-related features. Such attributes may be defined using timestamps of different semantics such as days and months, hours and minutes, days of the week, etc. The database keeps time-related features by storing sequences of their values that change with time. In contrast, a relational database usually stores the most recent value only. The temporal data allow the user to find unique patterns in the data, which are usually related to trends of changes. In the case of the heart clinic, we can ask whether blood pressure has a tendency to increase or decrease with age for particular patients or patient groups–say, those with high cholesterol–or we could compare rates of decrease or increase.

2.4.6. Text Databases

Text databases include features (attributes) that contain word descriptions for objects. These features are not simple nominal but hold long sentences or paragraphs of text. Examples for the heart clinic would be reports from patient interviews, physician's notes and recommendations, descriptions of how a given drug works for a given patient, etc. The text may be either **unstructured**, like sentences in plain written language (English, Polish, Spanish, etc.); **semistructured**, where some words or parts of the sentence are annotated, such as descriptions of how a drug works, which may use special annotations for the drug's name and the dose; and **structured**, where all the words are annotated, like a physician's recommendation that may be a fixed form listing only specific drugs and doses. Other, easy-to-understand examples include text documents as unstructured text, HTML pages and emails as semistructured text, and library data as structured text. Analysis of such databases requires special tools and integration with text data hierarchies, such as dictionaries, thesauruses, and specialized discipline-oriented term-classification systems, such as those in medicine, engineering, economy, etc. The mining of text databases is described in Chapter 14.

2.4.7. Multimedia Databases

Multimedia databases allow storage, retrieval, and manipulation of image, video, and audio data. The main concern regarding such data sources is their very large size, and therefore, specialized storage and search techniques are required. Additionally, both video and audio data are recorded in real time, and thus the database must include mechanisms that assure a steady and predefined rate of acquisition to avoid gaps, system buffer overflows, etc. An example application of a multimedia database for the heart clinic would be to find the relation between a video of heart motion and a recording of the heart beats.

2.4.8. World Wide Web

The World Wide Web (**WWW**) is an enormous distributed repository of data that is linked together via the use of **hyperlinks**. The hyperlinks link together individual data objects of possibly different types, allowing for interactive access to the information. The most specific characteristic of the WWW is that users seek information traversing between objects via links. The WWW also provides specialized query engines such as Google, Yahoo!, AltaVista, and Prodigy.

Finally, a comment on **heterogeneous databases**. These consist of a set of interconnected databases that communicate between themselves in order to exchange data and provide answers to user queries. The individual databases are connected via intra- and intercomputer networks, and usually they are of different types. The biggest challenge for a heterogeneous database is that objects in the component databases usually differ substantially, resulting in difficulties in developing common semantics to facilitate communication between them. The handling of such databases is beyond the scope of this book.

2.5. Data Storage and Data Mining

We address here several issues concerning data mining from the perspective of the data storage system used. From the point of view of the data warehouse and database users, data mining is often seen as an execution of a set of OLAP commands, but this perception is incorrect. Similarly, the ability to perform data or information retrieval (selection of data from a large repository, such as the WWW) or to find aggregate values should not be confused with data mining but instead should be categorized as using a database or an information retrieval system, respectively. Although the capabilities of the described storage systems may seem advanced and may appear to provide everything a user might need when analyzing data, the field of data mining goes much further. Data mining provides more complex techniques for understanding data and generating new knowledge than the simple summarization-like processing offered by OLAP. It allows for automated discovery of patterns and trends in the data. For instance, based on the information available in the heart clinic database, data mining may allow a user to learn which patients are susceptible to a particular heart condition, to discover that increased blood pressure over a period of time may lead to certain heart defects, or to identify deviations such as patients with unusually high or low blood pressure associated with their given heart condition.

3. Issues Concerning the Amount and Quality of Data

Several fundamental issues related to the data have a significant impact on the quality of the outcome of a knowledge discovery process. These include the following:

- the huge volume of data, and the related problem of scalability of data mining methods
- the dynamic nature of the data, which are constantly being updated/changed
- problems related to data quality, such as imprecision, incompleteness, noise, missing values, and redundancy

The above issues must be addressed before we perform any further work on the data; otherwise, we may encounter serious problems later in the knowledge discovery process. For instance, only some of the data mining methods can be used on data that contain missing values or noise. In this case, we need to make sure that a suitable method is available to analyze the data. When developing a real-time system, we need either to select an appropriate data mining system that can analyze the data in the specified amount of time or to reduce the processing time, for instance, by reducing the size of the data.

In the last decade, a significant amount of work has been devoted to data security and privacy issues. Data miners must be aware of the existing standards for protection against unauthorized access to data and must know how to prepare the data to guarantee confidentiality of the information present therein. The best example is medical data, for which the Health Insurance Portability and Accountability (HIPAA) dictates, in a very stringent way, how to design databases so as to store data securely and to preserve patient confidentiality. These issues are described in Chapter 16.

3.1. High Dimensionality

Although many data mining systems are available, only some can be considered as "truly" mining the data. The distinguishing factor is their ability to handle massive amount of data, which requires the use of **scalable** algorithms and methods. The scalability issue is not related to efficient storage and retrieval of the data (these are handled by using DBMSs) but instead to the algorithm design. Systems that are not capable of handling large quantities of data are known by the terms *machine learning* or *statistical data analysis systems*. For each data mining method under consideration, one needs to evaluate the sensitivity to the size of the data, which translates into the amount of time required for processing such data. Application domains such as the analysis of data from large retail store chains (e.g., Wal-Mart) deal with hundreds of millions of objects, while in the field of bioinformatics, data sets are described by several thousand features (e.g., microarray data analysis). Published results of a survey on the largest and most heavily used commercial databases show that the average size of Unix databases experienced a 6-fold, and of Windows databases a 14-fold increase, between 2001 and 2003. Large commercial databases now average 10 billion objects. Although only a selected portion of these objects is used for data mining purposes, the ability to cope with large quantities of data remains one of the most important data mining issues. Three dimensions are usually considered:

– The number of objects, which may range from a few hundred to a few billion
– The number of features, which may range from a few to several thousand
– The number of values a feature assumes, which may range from two to a few million

The ability of a particular data mining algorithm to cope with highly dimensional inputs is usually described by its asymptotic complexity, i.e., an estimate of an "order of magnitude" of the total number of operations, which translates into the specific duration of run time. The asymptotic complexity describes the growth rate of the algorithm's run time as the size of each dimension increases. The most commonly performed asymptotic complexity analysis describes scalability with respect to the number of objects.

The following example illustrates the importance of the asymptotic complexity of a data mining algorithm. Assume the user wants to generate knowledge in terms of rules from the data using either a decision tree or a rule algorithm (for details, see Chapter 12). Assume we want to use the following algorithms:

– *See5*, which has log-linear complexity, i.e., $O(n^* \log(n))$, n is the number of objects
– *DataSqueezer*, which has linear complexity, i.e., $O(n)$
– *CLIP4*, which has quadratic complexity, i.e., $O(n^2)$
– *C4.5 rules*, which has cubic complexity, i.e., $O(n^3)$

To make the example more compelling, assume that the constant for the linear and log-linear algorithms is 100 times larger than for the quadratic and cubic algorithms–a situation that strongly favors the latter two algorithms. Table 3.2 shows relative difference in the running time in seconds required to compute the rules for the increasing number of objects for the four algorithms.

Table 3.2. Comparison of running time with respect to the "number of objects" for four rule-learning algorithms.

Number of objects in thousands	Running time [sec]			
	DataSqueezer $y = 100^{*}a^{*}x$	**See5** $y = 100^{*}a^{*}x^{*}\log(x)$	**CLIP4** $y = a^{*}x^{2}$	**C4.5 rules** $y = a^{*}x^{3}$
100	1000.0	2000.0	1000.0	100000.0
200	2000.0	4602.1	4000.0	800000.0
300	3000.0	7431.4	9000.0	2700000.0
400	4000.0	10408.2	16000.0	6400000.0
500	5000.0	13494.9	25000.0	12500000.0
600	6000.0	16668.9	36000.0	21600000.0
700	7000.0	19915.7	49000.0	34300000.0
800	8000.0	23224.7	64000.0	51200000.0
900	9000.0	26588.2	81000.0	72900000.0
1000	10000.0	30000.0	100000.0	100000000.0
1100	11000.0	33455.3	121000.0	133100000.0
…	…	…	…	…

Thus, for 100,000 objects, the fastest linear algorithm computes the results in 1,000 seconds, while the slowest cubic algorithm needs 100,000 seconds. When the number of objects increases tenfold, the time needed to compute the results increases to 10,000 seconds for the linear algorithm and to 100,000,000 seconds for the cubic algorithm. This example shows that for the linear algorithm, doubling the number of the objects results in doubling the run time, while for the cubic algorithm the run time increases eightfold. Note that the rate at which the run time increases as the number of the objects increases (or any other dimension) is more important than the absolute value of the run time. The functional relationship between the number of objects and the run time is shown in Figure 3.9.

A number of techniques are available to improve the scalability of data mining algorithms. These can be divided into two groups:

– Techniques that **speed up the algorithm**. This outcome can be achieved through use of heuristics, optimization, and parallelization. Heuristics simplify the processing of the data

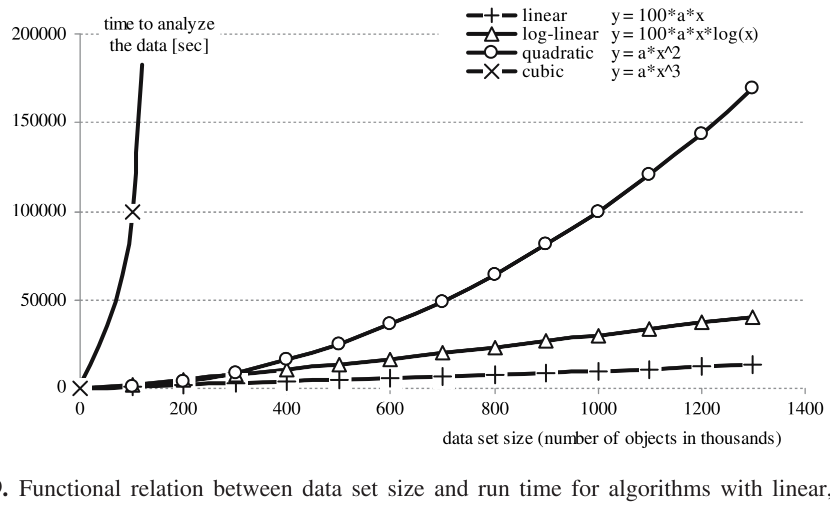

Figure 3.9. Functional relation between data set size and run time for algorithms with linear, log-linear, quadratic, and cubic asymptotic complexity.

performed by the algorithm, e.g., only rules of a certain maximal length are considered, and longer rules are never generated. Optimization of the algorithm is often achieved by using efficient data structures, such as bit vectors, hash tables, and binary search trees, to store and manipulate the data. Finally, parallelization aims to distribute the processing of the data by the algorithm into several processors that work in parallel and thus can speed up the computations.

– Techniques that **partition the data set**. This outcome can be achieved by reducing the dimensionality of the input data set through reduction of the number of objects, features, number of values per feature, and sequential or parallel processing of data divided into subsets. With dimensionality reduction, the data are sampled and only a representative subset of objects and/or features is used (see Chapter 7). This approach is especially viable when dealing with large data sets, in which the objects and/or attributes may be redundant, very similar, or irrelevant, and thus a subset of the objects and/or features may provide representative information about the entire data set. Alternatively, for some algorithms, it is beneficial or necessary to reduce the number of values an attribute assumes by using discretization algorithms (see Chapter 8). Finally, the division of data into subsets and the processing of these subsets sequentially or in parallel is beneficial when the complexity of the data mining algorithms is worse than linear. As an example, let us consider a cubic complexity algorithm from Table 3.2. When used to analyze 1,000,000 objects, the algorithm takes 100,000,000 seconds to compute the result. However, if the data set is partitioned into 10 subsets of 100,000 objects each and the subsets are processed sequentially (processing of each subset takes "only" 100,000 seconds), then all the data can be analyzed in $10^{*}100,000 = 1,000,000$ seconds – a whopping 99,000,000 seconds savings. Note that the division of the data into subsets should be performed only when the results generated for each of the subsets can be combined into a result that spans the entire data set.

3.2. Dynamic Data

Data sets are often dynamic in nature, i.e., new objects and/or features may be added and some objects and/or features may be removed or replaced by new ones. In this case, data mining algorithms should also evolve with time, which means that the knowledge derived so far should be also incrementally updated. The difference between **incremental** and **nonincremental** DM algorithms is shown in Figure 3.10.

The main challenge in incremental data mining methods is merging the newly generated knowledge from new data with the existing, previous knowledge. The merger may be as simple as adding the new knowledge to the existing knowledge, but most often it requires modifying the existing knowledge to preserve consistency.

3.3. Imprecise, Incomplete, and Redundant Data

Real data often include **imprecise data** objects. For instance, we may not know the exact value of a given medical test, but instead we know whether the value is high, average, or low. In such cases, fuzzy and/or rough sets can be used to process such imprecise information (see Chapter **??**).

The term **incomplete data** refers to the situation in which the available data do not contain enough information to discover new (desired) knowledge. Incompleteness may be a result of an insufficient feature description of the objects, insufficient number of objects, or missing values for a given feature. For instance, when analyzing heart patient data, if one wanted to distinguish between sick and healthy patients but only demographic information was available, the task could be impossible to complete. In dealing with incomplete data, we first need to identify the problem and then take measures to remove it. To detect incompleteness, the user must analyze the existing data set and determine whether the existing features and objects give a sufficiently rich representation of the problem at hand. A common sign of incompleteness is when generated

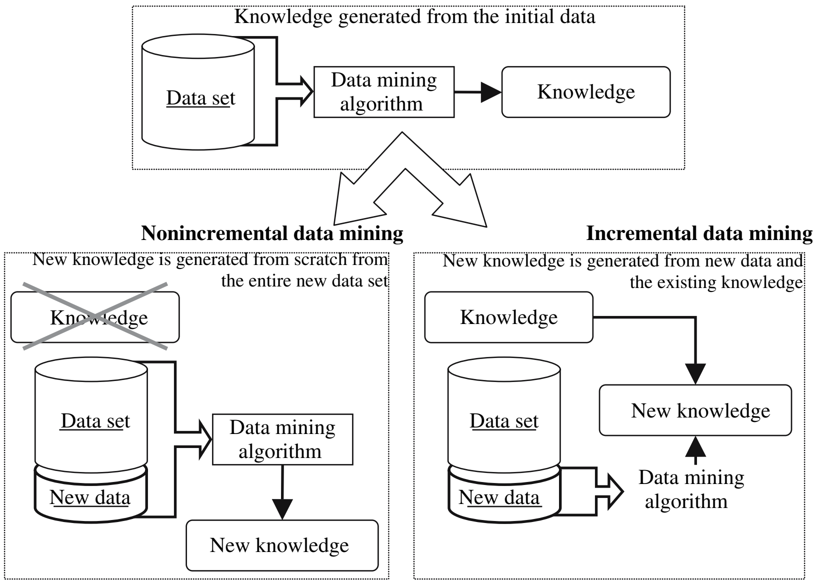
Knowledge generated from the initial data
Data set
Data mining algorithm
Knowledge
Nonincremental data mining
New knowledge is generated from scratch from the entire new data set
Knowledge
Data set
New data
Data mining algorithm
New knowledge
Incremental data mining
New knowledge is generated from new data and the existing knowledge
Knowledge
Data set
New data
New knowledge
Data mining algorithm

Figure 3.10. Incremental vs. nonincremental data mining.

(new) knowledge is of low quality and cannot be improved by using other data mining methods. The most obvious way to improve the incomplete data is to gather and record additional data, such as new features and/or new objects. A discussion of missing values can be found in the next section.

The term **redundant data** refers to a problem when there are two or more identical objects, or two or more features that are strongly correlated. Redundant data (objects, one of the correlated features) are thus removed to speed up the processing time. In some cases, however, redundant data may reveal useful information, i.e., frequency of identical objects may provide useful information. A special case of redundant data is **irrelevant data**, where some objects and/or features are insignificant with respect to data analysis. For instance, we can expect that a patient's name is irrelevant with respect to his/her heart condition and thus can be removed. Redundant data can be identified by feature selection and extraction algorithms (see Chapter 7).

3.4. Missing Values

Many data sets are plagued by the problem of **missing values**. These can result from incomplete manual data entry, incorrect measurements, equipment errors, etc. Such values are usually denoted by "NULL", "*" and "?" values. In some domains (as in medicine), it is common to encounter data with a large percentage of missing values, even over 50% of all values. The methods for dealing with these missing values can be divided into two categories:

– **Removal of missing data**. In this case the objects and/or features with missing values are simply discarded. This approach is effective only when the removed features are not crucial to the analysis, since the removal would than result in decreasing the information content of the data. It is practical only when the data contain small amounts of missing values and when analysis of the remaining complete data will not be biased by the removal. For example, in distinguishing between sick and healthy heart patients, the removal of the blood pressure feature will result in biasing the discovered knowledge towards other features, although obviously the blood pressure feature is important.

– **Imputation (filling in) of missing data**. Imputation is performed using a number of different algorithms, which can be subdivided into single and multiple imputation methods. In single-imputation methods a missing value is imputed by a single value, while with multiple-imputation methods, several likelihood- ordered choices for imputing the missing value are computed and one "best" value is selected.

Two missing data imputation methods, the mean and the hot deck, are described next. In **mean imputation**, the mean of the values of a feature that contains missing data is used to fill in the missing values. For a symbolic (categorical) feature, a mode, which is the most frequent value, is used instead of the mean. The algorithm imputes missing values for each attribute separately. Mean imputation can be conditional or unconditional. The conditional mean method imputes a mean value that depends on the values of the complete features for the incomplete object. In **hot deck imputation**, for each object that contains missing values, the most similar object is found, and the missing values are imputed from that object. If the most similar record also contains missing values for the same features as in the original object, then the similar record is discarded and another closest object is found. The procedure is repeated until all the missing values are successfully imputed or the entire data set is searched. When no similar object with the required values is found, the closest object with the minimum number of missing values is chosen to impute the missing values. The similarity between objects is usually measured using **distance**. One of the commonly used measures for both numerical and categorical values assumes a distance of 0 between two corresponding features if both have the same value; otherwise, the distance is 1. A distance of 1 is also assumed for a feature for which at least one object has a missing value. The data set for heart clinic patients given in Table 3.1 can be used to illustrate the working of the missing data methods (see Figure 3.11). An example conditional-mean imputation rule could state that the mean value for the *chest pain type* feature is imputed only if the *diagnosis* feature has the value *absent*; otherwise, a 2*mean value is imputed. Another popular conditional-mean imputation rule constrains the computation of the mean value to a certain subset of the objects: for instance, the imputed mean values of features for a given object might be computed by using only objects with the same value of the *diagnosis* feature, i.e., *absent* or *present*.

Several other more complex missing-data imputation methods can also be used. Some compute the most probable value to replace the missing value based on information from the complete portion of the data. An example is a regression imputation, which is performed by regression of the missing values using complete values for a given object, e.g., missing value for the *cholesterol* feature of object 3 would be computed by using a linear function of values for the remaining numerical features, such as *age*, *blood pressure*, and *chest pain type*. Several regression models can be used, including linear, logistic, polytomous, etc. Usually, the logistic regression model is applied for binary features, linear regression for numerical continuous features, and polytomous regression for discrete features (see Chapter 11 for information about regression methods).

3.5. Noise

Noise in the data is defined as a value that is a random error or variance in a measured feature. Depending on the amount in the data, noise it can be a substantial problem that can jeopardize the knowledge discovery process. The influence of noise on the data can be prevented by imposing constraints on features in order to detect anomalies when the data are entered. For instance, DBMS usually provides means to define customized constrains for individual attributes. When noise is already present, it can be removed by using one of the following methods: manual inspection with the use of predefined constraints on feature values, binning, and clustering.

Original data set with missing values

Object ID	Name	Age	Sex	Blood pressure	Cholesterol in mg/dl	Chest pain type	Defect type	Diagnosis
1	Konrad Black	31	male	130.0	NULL	NULL	NULL	NULL
2	Konrad Black	31	male	130.0	331.2	1	normal	absent
3	Magda Doe	26	female	115.0	NULL	4	fixed	present
4	Magda Doe	26	female	115.0	407.5	NULL	NULL	NULL
5	Anna White	56	female	120.0	45.0	2	normal	absent

Data set with removed objects with missing values

Object ID	Name	Age	Sex	Blood pressure	Cholesterol in mg/dl	Chest pain type	Defect type	Diagnosis
2	Konrad Black	31	male	130.0	331.2	1	normal	absent
5	Anna White	56	female	120.0	45.0	2	normal	absent

Data set with removed features with missing values

Object ID	Name	Age	Sex	Blood pressure
1	Konrad Black	31	male	130.0
2	Konrad Black	31	male	130.0
3	Magda Doe	26	female	115.0
4	Magda Doe	26	female	115.0
5	Anna White	56	female	120.0

Data set with missing values imputed using the mean imputation method

Object ID	Name	Age	Sex	Blood pressure	Cholesterol in mg/dl	Chest pain type	Defect type	Diagnosis
1	Konrad Black	31	male	130.0	261.2	2	normal	absent
2	Konrad Black	31	male	130.0	331.2	1	normal	absent
3	Magda Doe	26	female	115.0	261.2	4	fixed	present
4	Magda Doe	26	female	115.0	407.5	2	normal	absent
5	Anna White	56	female	120.0	45.0	2	normal	absent

The imputed values for the “cholesterol” feature equal (331.2+407.5+45)/3.

The imputed values for the “chest pain type” equal (1+4+2)/3 rounded to the nearest integer.

The imputed values for the “defect type” equal the most frequent value “normal”.

Data set with missing values imputed using the hot deck imputation

Object ID	Name	Age	Sex	Blood pressure	Cholesterol in mg/dl	Chest pain type	Defect type	Diagnosis
1	Konrad Black	31	male	130.0	331.2	1	normal	absent
2	Konrad Black	31	male	130.0	331.2	1	normal	absent
3	Magda Doe	26	female	115.0	407.5	4	fixed	present
4	Magda Doe	26	female	115.0	407.5	4	fixed	present
5	Anna White	56	female	120.0	45.0	2	normal	absent

The distance between objects 1 and 2 (object ID is not considered as a feature)
equals 0+0+0+0+1+1+1+1=4.

The distance between objects 1 and 3 equals 1+1+1+1+1+1+1+1=8.

The object most similar to object 3 that has a complete value for “cholesterol” is object 4 (distance 4).

Figure 3.11. Results of using missing-data handling methods for a data set describing heart clinic patients.

In **manual inspection**, the user checks feature values against predefined constraints and manually removes (or changes into missing values) all values that do not satisfy these constraints. For example, for object 5 in Figure 3.11, the *cholesterol* value is 45.0, which is outside the predefined acceptable interval for this feature, namely, within [50.0, 600.0].

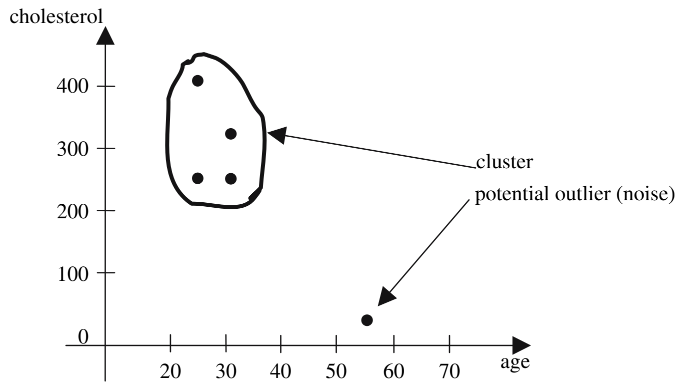

Figure 3.12. Noise detection with use of clustering.

Binning first orders the values of the noisy feature and then replaces the values with a mean or median value for the predefined bins. As an example, let us consider the *cholesterol* feature, with its values 45.0, 261.2, 331.2, and 407.5. If the bin size equals two, two bins are created: bin_1 with 45.0 and 261.2, and bin_2 with 331.2 and 407.5. For bin_1 the mean value is 153.1, and for bin_2 it is 369.4. Therefore the values 45.0 and 261.2 would be replaced with 153.1 and the values 331.2 and 407.5 with 369.4. Note that the two new values are within the acceptable interval.

Clustering finds groups of similar objects and simply removes (or changes into missing values) all values that fall outside the clusters. For instance, clustering that uses the *age* and *cholesterol* features from Figure 3.11 (in the data set with missing values imputed using the mean imputation) is shown in Figure 3.12. See Chapter 9 for more information on clustering.

4. Summary and Bibliographical Notes

In this Chapter, we defined concepts related to **data** and **data sets**. We also covered various data storage techniques and issues related to quality and quantity of data that are used for data mining purposes. The most important topics discussed in this Chapter are the following:

- different data types, including **numerical** and **symbolic** values, which can be subdivided into **categorical** (discrete), **binary** (dichotomous), **nominal** (polytomous), **ordinal** and **continuous**
- data are organized into rectangularly formatted tables called **data sets**, where rows represent **objects** and columns represent **features/attributes**, which describe the objects
- data sets are stored as **flat** (rectangular) **files** and in other formats using **databases** and **data warehouses**
- specialized data types, including **transactional** data, **spatial** data, **hypertext**, **multimedia** data, **temporal** data, and the **WWW**
- important issues related to the data used in data mining including
  - **large quantities** of data and the related problem of **scalability** of data mining methods
  - **dynamic data**, which require the use of **incremental** data mining methods
  - data quality problems, which include **imprecision**, **incompleteness**, **redundancy** **missing values**, and **noise**

A tutorial-style article that introduces data types, their organization into data sets, and their relation to DM can be found in [3]. Another good source for fundamental concepts concerning data types, data sets and data quality and quantity issues is provided in [4], in particular Chapters 4, 6, and 14.

There are three main **repositories** of flat file data sets from different domains, donated by data mining practitioners:

– http://www.ics.uci.edu/~mlearn/ (**Machine Learning repository**)
– http://kdd.ics.uci.edu/ (**Knowledge Discovery in Databases archive**)
– http://lib.stat.cmu.edu/ (**StatLib repository**)

These sites provide free access to numerous data sets and are used mainly for benchmarking and comparative purposes. Data sets are posted by the original authors along with results concerning the data. One prominent example of data mining in specialized data is a prototype multimedia data mining system called MultiMediaMiner [16].

The issues related to the quantity and quality of data used in data mining have been addressed in numerous publications. A survey of the largest commercial databases was published in [15]. The availability of huge data sets that can be extracted from these databases calls for the development of **scalable algorithms** for data mining. A good overview of scalable algorithms is given in [2]. Data-quality research can be divided into two streams: data quality assessment and methods for improving data quality. A method that combines subjective and objective **assessments of data quality** and proposes practical data quality matrices was introduced in [7]. A framework for the identification and analysis of data quality issues, covering management responsibilities, operation and assurance costs, research and development, production, distribution, personnel management, and legal function, is described in [14]. The second approach addresses **data cleaning** methods. A methodology for data cleaning for relational data is reported in [5]. An important branch of data cleaning is related to the handling of missing values. Traditional imputation methods for missing values are based on statistical analysis and include simple algorithms such as mean and hot deck imputation, as well as more complex methods including regression and the Expectation-Maximization (EM) algorithm. The **multiple imputations** method was first described by Rubin [8], while subsequent methods [6, 10] used Bayesian algorithms that perform multiple imputations using posterior predictive distribution of the missing data based on the complete data. Another method imputes each incomplete attribute by cubic spline regression [1]. A detailed description of multiple imputation algorithms can be found in [9, 12, 17], and a primer can be found in [13].

There is significant industrial and academic interest in data cleaning research and development. Several groups exist that aim to develop data cleaning solutions, including the following:

– The *Data Cleaning* group at Microsoft:
  http://research.microsoft.com/dmx/DataCleaning
– The *Data Quality* group at AT&T:
  http://www.dataquality-research.com
– The *Total Data Quality Management* group at MIT:
  http://web.mit.edu/tdqm/www/ index.shtml

Finally, data security and privacy issues with a focus on HIPAA rules in the United States, Canada and Europe are discussed in [11].

References

1. Alzola, C., and Harrell, F. 1999. An introduction of S-Plus and the Hmisc and design libraries (download from http://www.med.virginia.edu/medicine/clinical/hes)
2. Ganti, V., Gehrke, J., and Ramakrishnan, R. Aug. 1999. Mining very large databases. *IEEE Computer*, 32(8):38–45
3. Holsheimer, M., and Siebes, A. 1994. *Data Mining: The Search for Knowledge in Databases*. Report CS-R9406, ISSN 0169–118X, CWI: Dutch National Research Center, Amersterdam, Netherlands

4. Klosgen, W., and Zytkow, J. (Eds.). 2002. *Handbook of Data Mining and Knowledge Discovery*, Oxford University Press New York, USA
5. Lee, M-L., Ling, T.W., Lu, H., and Ko, Y.T. 1999. Cleansing data for mining and warehousing, *Proceedings of the International Conference on Database and Expert Systems Applications* (DEXA), Florence, Italy, *Lecture Notes in Computer Science*, 1677:751–760
6. Li, K-H. 1988. Imputation using markov chains. *Journal of Statistical Computation and Simulation*, 30:57–79
7. Pipino, L., Lee, Y., and Wang, R. 2002. Data quality assessment. *Communications of the ACM*, 45(4):211–218
8. Rubin, D.B. 1977. Formalizing subjective notions about the effect of nonrespondents in sample surveys. *Journal of American Statistical Association*, 72:538–543
9. Rubin, D.B. 1987. *Multiple Imputations for Nonresponse in Surveys*, John Wiley and Sons: New York
10. Rubin, D.B., and Schafer, J.L. 1990. Efficiently creating multiple imputation for incomplete multivariate normal data. *Proceedings of the Statistical Computing Section*, Alexandria: ASA, 83–8
11. Saul, J.M. 2000. Legal policy and security issues in the handling of medical data. In Cios, K.J. (Ed.), *Medical Data Mining and Knowledge Discovery*, Springer Verlag, 21–40
12. Shafer, J.L. 1997. *Analysis of Incomplete Multivariate Data*, Chapman and Hall Heidelberg, Germany
13. Shafer, J.L. 1999. Multiple imputations: a primer. *Statistical Methods in Medical Research*, 8:3–15
14. Wang, R., Storey, V., and Firth C. 1995. A framework for analysis of data quality research. *IEEE Transactions on Knowledge and Data Engineering*, 7(4):623–640
15. Winter, R., and Auerbach, K. May 2004. Contents under pressure. *Intelligent Enterprise*, available online at http://www.intelligententerprise.com/showArticle.jhtml?articleID=18902161
16. Zaïane, O., Han, J., Li, Z.N., and Hou, J. 1998. Mining multimedia data, *Proceeding of the CASCON'98: Meeting of Minds*, 83–96, Toronto, Canada
17. Zhang, P. 2003. Multiple imputation: theory and method (with discussion). *International Statistical Review*, 71(3):581–592

5. Exercises

1. Discuss feature types and organization of data in the *Adult*, *Mushrooms*, and *SPECT* data sets from the ML repository (http://www.ics.uci.edu/~mlearn/). Using this repository, find at least three data sets that consist of only symbolic features, and another three that consists only of numerical features. Finally, find the data set that includes the largest number of features.
2. Convert the data set from Table 3.1 into the transactional format (an example transactional database is shown in Figure 3.8). Briefly describe how such a conversion could be performed. Remember to use the appropriate vocabulary.
3. Considering the data given in Table 3.2, calculate the best partitioning of a data set containing 1,000,000 objects with respect to minimizing the total run time when applying the See5 algorithm to these subsets sequentially. The data set can be divided into two or more subsets, where each subset size is limited to those listed in Table 3.2, i.e. 100,000, 200,000, …, 900,000, that is, the data set can be partitioned into ten subsets of 100,000, one of 800,000 and one of 200,000, or one of 800,000 and two 100,000. You must use the estimated run time values from Table 3.2 and add an overhead of 1,000 seconds for the processing of each subset. For example, division of the data set into ten 100,000 subsets gives a total run time of $10{*}2,000 + 10{*}1,000 = 30,000$ seconds compared with $30,000 + 1,000 = 31,000$ seconds if the entire data set is used. Compute how many seconds can be saved by such sequential processing.
4. Write a program that performs imputation of missing values using both the mean and hot deck methods for the *Water Treatment Plant* data set from the ML repository (http://www.ics.uci.edu/~mlearn/). After performing the imputation, discuss which imputation method, in your opinion, gave better results based on a comparison between the two resulting complete data sets.

5. Investigate other methods (except manual, binning, and clustering) for dealing with noise in the data sets. Find and briefly summarize two alternative methods. Write a one-page report that includes links and references to the source of the information that you have used.
6. Briefly define and describe the data *inconsistency* problem. This data quality issue was not described in this Chapter but may be important with respect to some data mining projects. You report should be similar to the description of the data redundancy problem from this Chapter and should include an example that is based on the heart clinic data set.