Chapter 1

An Introduction to Data Mining

"*Education is not the piling on of learning, information, data, facts, skills, or abilities – that's training or instruction – but is rather making visible what is hidden as a seed.*"—Thomas More

1.1 Introduction

Data mining is the study of collecting, cleaning, processing, analyzing, and gaining useful insights from data. A wide variation exists in terms of the problem domains, applications, formulations, and data representations that are encountered in real applications. Therefore, "data mining" is a broad umbrella term that is used to describe these different aspects of data processing.

In the modern age, virtually all automated systems generate some form of data either for diagnostic or analysis purposes. This has resulted in a deluge of data, which has been reaching the order of petabytes or exabytes. Some examples of different kinds of data are as follows:

- *World Wide Web:* The number of documents on the indexed Web is now on the order of billions, and the invisible Web is much larger. User accesses to such documents create Web access logs at servers and customer behavior profiles at commercial sites. Furthermore, the linked structure of the Web is referred to as the *Web graph*, which is itself a kind of data. These different types of data are useful in various applications. For example, the Web documents and link structure can be mined to determine associations between different topics on the Web. On the other hand, user access logs can be mined to determine frequent patterns of accesses or unusual patterns of possibly unwarranted behavior.

- *Financial interactions:* Most common transactions of everyday life, such as using an automated teller machine (ATM) card or a credit card, can create data in an automated way. Such transactions can be mined for many useful insights such as fraud or other unusual activity.

C. C. Aggarwal, *Data Mining: The Textbook*, DOI 10.1007/978-3-319-14142-8_1

- *User interactions:* Many forms of user interactions create large volumes of data. For example, the use of a telephone typically creates a record at the telecommunication company with details about the duration and destination of the call. Many phone companies routinely analyze such data to determine relevant patterns of behavior that can be used to make decisions about network capacity, promotions, pricing, or customer targeting.

- *Sensor technologies and the Internet of Things:* A recent trend is the development of low-cost wearable sensors, smartphones, and other smart devices that can communicate with one another. By one estimate, the number of such devices exceeded the number of people on the planet in 2008 [30]. The implications of such massive data collection are significant for mining algorithms.

The deluge of data is a direct result of advances in technology and the computerization of every aspect of modern life. It is, therefore, natural to examine whether one can extract *concise* and possibly *actionable* insights from the available data for application-specific goals. This is where the task of data mining comes in. The raw data may be arbitrary, unstructured, or even in a format that is not immediately suitable for automated processing. For example, manually collected data may be drawn from heterogeneous sources in different formats and yet somehow needs to be processed by an automated computer program to gain insights.

To address this issue, data mining analysts use a pipeline of processing, where the raw data are collected, cleaned, and transformed into a standardized format. The data may be stored in a commercial database system and finally processed for insights with the use of analytical methods. In fact, while data mining often conjures up the notion of analytical algorithms, the reality is that the vast majority of work is related to the data preparation portion of the process. This pipeline of processing is conceptually similar to that of an actual mining process from a mineral ore to the refined end product. The term "mining" derives its roots from this analogy.

From an analytical perspective, data mining is challenging because of the wide disparity in the problems and data types that are encountered. For example, a commercial product recommendation problem is very different from an intrusion-detection application, even at the level of the input data format or the problem definition. Even within related classes of problems, the differences are quite significant. For example, a product recommendation problem in a multidimensional database is very different from a social recommendation problem due to the differences in the underlying data type. Nevertheless, in spite of these differences, data mining applications are often closely connected to one of four "super-problems" in data mining: association pattern mining, clustering, classification, and outlier detection. These problems are so important because they are used as building blocks in a majority of the applications in some indirect form or the other. This is a useful abstraction because it helps us conceptualize and structure the field of data mining more effectively.

The data may have different formats or *types*. The type may be quantitative (e.g., age), categorical (e.g., ethnicity), text, spatial, temporal, or graph-oriented. Although the most common form of data is multidimensional, an increasing proportion belongs to more complex data types. While there is a conceptual portability of algorithms between many data types at a very high level, this is not the case from a practical perspective. The reality is that the precise data type may affect the behavior of a particular algorithm significantly. As a result, one may need to design refined variations of the basic approach for multidimensional data, so that it can be used effectively for a different data type. Therefore, this book will dedicate different chapters to the various data types to provide a better understanding of how the processing methods are affected by the underlying data type.

A major challenge has been created in recent years due to increasing data volumes. The prevalence of continuously collected data has led to an increasing interest in the field of *data streams*. For example, Internet traffic generates large streams that cannot even be stored effectively unless significant resources are spent on storage. This leads to unique challenges from the perspective of processing and analysis. In cases where it is not possible to explicitly store the data, all the processing needs to be performed in real time.

This chapter will provide a broad overview of the different technologies involved in preprocessing and analyzing different types of data. The goal is to study data mining from the perspective of different problem abstractions and data types that are frequently encountered. Many important applications can be converted into these abstractions.

This chapter is organized as follows. Section 1.2 discusses the data mining process with particular attention paid to the data preprocessing phase in this section. Different data types and their formal definition are discussed in Sect. 1.3. The major problems in data mining are discussed in Sect. 1.4 at a very high level. The impact of data type on problem definitions is also addressed in this section. Scalability issues are addressed in Sect. 1.5. In Sect. 1.6, a few examples of applications are provided. Section 1.7 gives a summary.

1.2 The Data Mining Process

As discussed earlier, the data mining process is a pipeline containing many phases such as data cleaning, feature extraction, and algorithmic design. In this section, we will study these different phases. The workflow of a typical data mining application contains the following phases:

1. *Data collection:* Data collection may require the use of specialized hardware such as a sensor network, manual labor such as the collection of user surveys, or software tools such as a Web document crawling engine to collect documents. While this stage is highly application-specific and often outside the realm of the data mining analyst, it is critically important because good choices at this stage may significantly impact the data mining process. After the collection phase, the data are often stored in a database, or, more generally, a *data warehouse* for processing.

2. *Feature extraction and data cleaning:* When the data are collected, they are often not in a form that is suitable for processing. For example, the data may be encoded in complex logs or free-form documents. In many cases, different types of data may be arbitrarily mixed together in a free-form document. To make the data suitable for processing, it is essential to transform them into a format that is friendly to data mining algorithms, such as multidimensional, time series, or semistructured format. The multidimensional format is the most common one, in which different *fields* of the data correspond to the different measured properties that are referred to as *features*, *attributes*, or *dimensions*. It is crucial to extract relevant features for the mining process. The feature extraction phase is often performed in parallel with data cleaning, where missing and erroneous parts of the data are either estimated or corrected. In many cases, the data may be extracted from multiple sources and need to be *integrated* into a unified format for processing. The final result of this procedure is a nicely structured data set, which can be effectively used by a computer program. After the feature extraction phase, the data may again be stored in a database for processing.

3. *Analytical processing and algorithms:* The final part of the mining process is to design effective analytical methods from the processed data. In many cases, it may not be

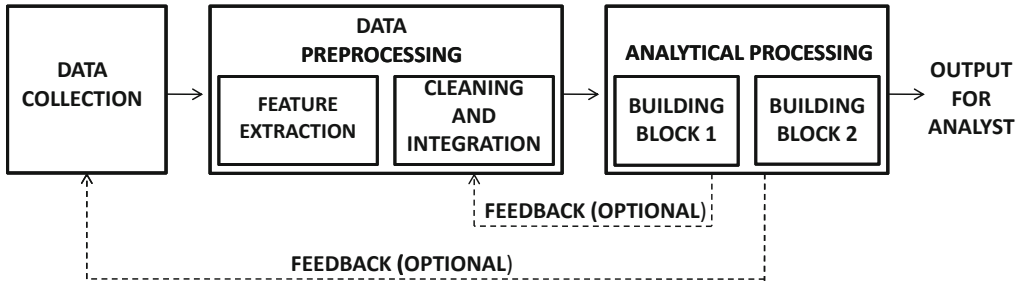


Figure 1.1: The data processing pipeline

possible to directly use a standard data mining problem, such as the four "superproblems" discussed earlier, for the application at hand. However, these four problems have such wide coverage that *many* applications can be broken up into components that use these different building blocks. This book will provide examples of this process.

The overall data mining process is illustrated in Fig. 1.1. Note that the analytical block in Fig. 1.1 shows multiple building blocks representing the design of the solution to a particular application. This part of the algorithmic design is dependent on the skill of the analyst and often uses one or more of the four major problems as a building block. This is, of course, not always the case, but it is frequent enough to merit special treatment of these four problems within this book. To explain the data mining process, we will use an example from a recommendation scenario.

Example 1.2.1 *Consider a scenario in which a retailer has Web logs corresponding to customer accesses to Web pages at his or her site. Each of these Web pages corresponds to a product, and therefore a customer access to a page may often be indicative of interest in that particular product. The retailer also stores demographic profiles for the different customers. The retailer wants to make targeted product recommendations to customers using the customer demographics and buying behavior.*

Sample Solution Pipeline In this case, the first step for the analyst is to collect the relevant data from two different sources. The first source is the set of Web logs at the site. The second is the demographic information within the retailer database that were collected during Web registration of the customer. Unfortunately, these data sets are in a very different format and cannot easily be used together for processing. For example, consider a sample log entry of the following form:

```
98.206.207.157 - - [31/Jul/2013:18:09:38 -0700] "GET /productA.htm
HTTP/1.1" 200 328177 "-" "Mozilla/5.0 (Mac OS X) AppleWebKit/536.26
(KHTML, like Gecko) Version/6.0 Mobile/10B329 Safari/8536.25"
"retailer.net"
```

The log may contain hundreds of thousands of such entries. Here, a customer at IP address 98.206.207.157 has accessed productA.htm. The customer from the IP address can be identified using the previous login information, by using cookies, or by the IP address itself, but this may be a noisy process and may not always yield accurate results. The analyst would need to design algorithms for deciding how to filter the different log entries and use only those which provide accurate results as a part of the *cleaning and extraction* process. Furthermore, the raw log contains a lot of additional information that is not necessarily

of any use to the retailer. In the *feature extraction* process, the retailer decides to create one record for each customer, with a specific choice of features extracted from the Web page accesses. For each record, an attribute corresponds to the number of accesses to each product description. Therefore, the raw logs need to be processed, and the accesses need to be aggregated during this *feature extraction* phase. Attributes are added to these records for the retailer's database containing demographic information in a *data integration phase*. Missing entries from the demographic records need to be estimated for further *data cleaning*. This results in a single data set containing attributes for the customer demographics and customer accesses.

At this point, the analyst has to decide how to use this cleaned data set for making recommendations. He or she decides to determine similar groups of customers, and make recommendations on the basis of the buying behavior of these similar groups. In particular, the *building block* of clustering is used to determine similar groups. For a given customer, the most frequent items accessed by the customers in that group are recommended. This provides an example of the entire data mining pipeline. As you will learn in Chap. 18, there are many elegant ways of performing the recommendations, some of which are more effective than the others depending on the specific definition of the problem. Therefore, the entire data mining process is an art form, which is based on the skill of the analyst, and cannot be fully captured by a single technique or building block. In practice, this skill can be learned only by working with a diversity of applications over different scenarios and data types.

1.2.1 The Data Preprocessing Phase

The data preprocessing phase is perhaps the most crucial one in the data mining process. Yet, it is rarely explored to the extent that it deserves because most of the focus is on the analytical aspects of data mining. This phase begins after the collection of the data, and it consists of the following steps:

1. *Feature extraction:* An analyst may be confronted with vast volumes of raw documents, system logs, or commercial transactions with little guidance on how these raw data should be transformed into meaningful database features for processing. This phase is highly dependent on the analyst to be able to abstract out the features that are most relevant to a particular application. For example, in a credit-card fraud detection application, the amount of a charge, the repeat frequency, and the location are often good indicators of fraud. However, many other features may be poorer indicators of fraud. Therefore, extracting the right features is often a skill that requires an understanding of the specific application domain at hand.

2. *Data cleaning:* The extracted data may have erroneous or missing entries. Therefore, some records may need to be dropped, or missing entries may need to be estimated. Inconsistencies may need to be removed.

3. *Feature selection and transformation:* When the data are very high dimensional, many data mining algorithms do not work effectively. Furthermore, many of the high-dimensional features are noisy and may add errors to the data mining process. Therefore, a variety of methods are used to either remove irrelevant features or transform the current set of features to a new data space that is more amenable for analysis. Another related aspect is data *transformation*, where a data set with a particular set of attributes may be transformed into a data set with another set of attributes of the same or a different type. For example, an attribute, such as age, may be partitioned into ranges to create discrete values for analytical convenience.

The data cleaning process requires statistical methods that are commonly used for missing data estimation. In addition, erroneous data entries are often removed to ensure more accurate mining results. The topics of data cleaning is addressed in Chap. 2 on data preprocessing.

Feature selection and transformation should not be considered a part of data preprocessing because the feature selection phase is often highly dependent on the specific analytical problem being solved. In some cases, the feature selection process can even be tightly integrated with the specific algorithm or methodology being used, in the form of a *wrapper model* or *embedded model*. Nevertheless, the feature selection phase is usually performed before applying the specific algorithm at hand.

1.2.2 The Analytical Phase

The vast majority of this book will be devoted to the analytical phase of the mining process. A major challenge is that each data mining application is unique, and it is, therefore, difficult to create general and reusable techniques across different applications. Nevertheless, many data mining formulations are repeatedly used in the context of different applications. These correspond to the major "superproblems" or building blocks of the data mining process. It is dependent on the skill and experience of the analyst to determine how these different formulations may be used in the context of a particular data mining application. Although this book can provide a good overview of the fundamental data mining models, the ability to apply them to real-world applications can only be learned with practical experience.

1.3 The Basic Data Types

One of the interesting aspects of the data mining process is the wide variety of data types that are available for analysis. There are two broad types of data, of varying complexity, for the data mining process:

1. *Nondependency-oriented data:* This typically refers to simple data types such as multidimensional data or text data. These data types are the simplest and most commonly encountered. In these cases, the data records do not have any specified dependencies between either the data items or the attributes. An example is a set of demographic records about individuals containing their age, gender, and ZIP code.

2. *Dependency-oriented data:* In these cases, implicit or explicit relationships may exist between data items. For example, a social network data set contains a set of *vertices* (data items) that are connected together by a set of *edges* (relationships). On the other hand, time series contains implicit dependencies. For example, two successive values collected from a sensor are likely to be related to one another. Therefore, the time attribute implicitly specifies a dependency between successive readings.

In general, dependency-oriented data are more challenging because of the complexities created by preexisting relationships between data items. Such dependencies between data items need to be incorporated directly into the analytical process to obtain contextually meaningful results.

Table 1.1: An example of a multidimensional data set

Name	Age	Gender	Race	ZIP code
John S.	45	M	African American	05139
Manyona L.	31	F	Native American	10598
Sayani A.	11	F	East Indian	10547
Jack M.	56	M	Caucasian	10562
Wei L.	63	M	Asian	90210

1.3.1 Nondependency-Oriented Data

This is the simplest form of data and typically refers to *multidimensional data*. This data typically contains a set of *records*. A record is also referred to as a *data point*, *instance*, *example*, *transaction*, *entity*, *tuple*, *object*, or *feature-vector*, depending on the application at hand. Each record contains a set of *fields*, which are also referred to as *attributes*, *dimensions*, and *features*. These terms will be used interchangeably throughout this book. These fields describe the different properties of that record. Relational database systems were traditionally designed to handle this kind of data, even in their earliest forms. For example, consider the demographic data set illustrated in Table 1.1. Here, the demographic properties of an individual, such as age, gender, and ZIP code, are illustrated. A multidimensional data set is defined as follows:

Definition 1.3.1 (Multidimensional Data) *A multidimensional data set $\mathcal{D}$ is a set of n records, $\overline{X_1} \ldots \overline{X_n}$, such that each record $\overline{X_i}$ contains a set of d features denoted by $(x_i^1 \ldots x_i^d)$.*

Throughout the early chapters of this book, we will work with multidimensional data because it is the simplest form of data and establishes the broader principles on which the more complex data types can be processed. More complex data types will be addressed in later chapters of the book, and the impact of the dependencies on the mining process will be explicitly discussed.

1.3.1.1 Quantitative Multidimensional Data

The attributes in Table 1.1 are of two different types. The age field has values that are numerical in the sense that they have a natural ordering. Such attributes are referred to as *continuous*, *numeric*, or *quantitative*. Data in which all fields are quantitative is also referred to as *quantitative data* or *numeric data*. Thus, when each value of x_i^j in Definition 1.3.1 is quantitative, the corresponding data set is referred to as quantitative multidimensional data. In the data mining literature, this particular subtype of data is considered the most common, and many algorithms discussed in this book work with this subtype of data. This subtype is particularly convenient for analytical processing because it is much easier to work with quantitative data from a statistical perspective. For example, the mean of a set of quantitative records can be expressed as a simple average of these values, whereas such computations become more complex in other data types. Where possible and effective, many data mining algorithms therefore try to convert different kinds of data to quantitative values before processing. This is also the reason that many algorithms discussed in this (or virtually any other) data mining textbook assume a quantitative multidimensional representation. Nevertheless, in real applications, the data are likely to be more complex and may contain a mixture of different data types.

1.3.1.2 Categorical and Mixed Attribute Data

Many data sets in real applications may contain categorical attributes that take on *discrete unordered* values. For example, in Table 1.1, the attributes such as gender, race, and ZIP code, have discrete values without a natural ordering among them. If each value of x_i^j in Definition 1.3.1 is categorical, then such data are referred to as *unordered discrete-valued* or *categorical*. In the case of *mixed attribute* data, there is a combination of categorical and numeric attributes. The full data in Table 1.1 are considered mixed-attribute data because they contain both numeric and categorical attributes.

The attribute corresponding to gender is special because it is categorical, but with only two possible values. In such cases, it is possible to impose an artificial ordering between these values and use algorithms designed for numeric data for this type. This is referred to as *binary* data, and it can be considered a special case of either numeric or categorical data. Chap. 2 will explain how binary data form the "bridge" to transform numeric or categorical attributes into a common format that is suitable for processing in many scenarios.

1.3.1.3 Binary and Set Data

Binary data can be considered a special case of either multidimensional categorical data or multidimensional quantitative data. It is a special case of multidimensional categorical data, in which each categorical attribute may take on one of at most two discrete values. It is also a special case of multidimensional quantitative data because an ordering exists between the two values. Furthermore, binary data is also a representation of setwise data, in which each attribute is treated as a set element indicator. A value of 1 indicates that the element should be included in the set. Such data is common in market basket applications. This topic will be studied in detail in Chaps. 4 and 5.

1.3.1.4 Text Data

Text data can be viewed either as a string, or as multidimensional data, depending on how they are represented. In its raw form, a text document corresponds to a *string*. This is a dependency-oriented data type, which will be described later in this chapter. Each string is a sequence of characters (or words) corresponding to the document. However, text documents are rarely represented as strings. This is because it is difficult to directly use the ordering between words in an efficient way for large-scale applications, and the additional advantages of leveraging the ordering are often limited in the text domain.

In practice, a *vector-space representation* is used, where the frequencies of the words in the document are used for analysis. Words are also sometimes referred to as *terms*. Thus, the precise ordering of the words is lost in this representation. These frequencies are typically normalized with statistics such as the length of the document, or the frequencies of the individual words in the collection. These issues will be discussed in detail in Chap. 13 on text data. The corresponding $n \times d$ data matrix for a text collection with n documents and d terms is referred to as a *document-term matrix*.

When represented in vector-space form, text data can be considered multidimensional quantitative data, where the attributes correspond to the words, and the values correspond to the frequencies of these attributes. However, this kind of quantitative data is special because most attributes take on zero values, and only a few attributes have nonzero values. This is because a single document may contain only a relatively small number of words out of a dictionary of size 10^5. This phenomenon is referred to as *data sparsity*, and it significantly impacts the data mining process. The direct use of a quantitative data mining

algorithm is often unlikely to work with sparse data without appropriate modifications. The sparsity also affects how the data are represented. For example, while it is possible to use the representation suggested in Definition 1.3.1, this is not a practical approach. Most values of x_i^j in Definition 1.3.1 are 0 for the case of text data. Therefore, it is inefficient to explicitly maintain a d-dimensional representation in which most values are 0. A bag-of-words representation is used containing only the words in the document. In addition, the frequencies of these words are explicitly maintained. This approach is typically more efficient. Because of data sparsity issues, text data are often processed with specialized methods. Therefore, text mining is often studied as a separate subtopic within data mining. Text mining methods are discussed in Chap. 13.

1.3.2 Dependency-Oriented Data

Most of the aforementioned discussion in this chapter is about the multidimensional scenario, where it is assumed that the data records can be treated independently of one another. In practice, the different data values may be (implicitly) related to each other temporally, spatially, or through explicit network relationship links between the data items. The knowledge about *preexisting* dependencies greatly changes the data mining process because data mining is all about finding relationships between data items. The presence of preexisting dependencies therefore changes the *expected* relationships in the data, and what may be considered *interesting* from the perspective of these expected relationships. Several types of dependencies may exist that may be either *implicit* or *explicit*:

1. *Implicit dependencies:* In this case, the dependencies between data items are not explicitly specified but are known to "typically" exist in that domain. For example, consecutive temperature values collected by a sensor are likely to be extremely similar to one another. Therefore, if the temperature value recorded by a sensor at a particular time is significantly different from that recorded at the next time instant then this is extremely unusual and may be interesting for the data mining process. This is different from multidimensional data sets where each data record is treated as an independent entity.

2. *Explicit dependencies:* This typically refers to graph or network data in which edges are used to specify explicit relationships. Graphs are a very powerful abstraction that are often used as an intermediate representation to solve data mining problems in the context of other data types.

In this section, the different dependency-oriented data types will be discussed in detail.

1.3.2.1 Time-Series Data

Time-series data contain values that are typically generated by continuous measurement over time. For example, an environmental sensor will measure the temperature continuously, whereas an electrocardiogram (ECG) will measure the parameters of a subject's heart rhythm. Such data typically have *implicit* dependencies built into the values received over time. For example, the adjacent values recorded by a temperature sensor will usually vary smoothly over time, and this factor needs to be explicitly used in the data mining process.

The nature of the temporal dependency may vary significantly with the application. For example, some forms of sensor readings may show periodic patterns of the measured

attribute over time. An important aspect of time-series mining is the extraction of such dependencies in the data. To formalize the issue of dependencies caused by temporal correlation, the attributes are classified into two types:

1. *Contextual attributes:* These are the attributes that define the *context* on the basis of which the implicit dependencies occur in the data. For example, in the case of sensor data, the time stamp at which the reading is measured may be considered the contextual attribute. Sometimes, the time stamp is not explicitly used, but a position index is used. While the time-series data type contains only one contextual attribute, other data types may have more than one contextual attribute. A specific example is *spatial data*, which will be discussed later in this chapter.

2. *Behavioral attributes:* These represent the values that are measured in a particular context. In the sensor example, the temperature is the behavioral attribute value. It is possible to have more than one behavioral attribute. For example, if multiple sensors record readings at synchronized time stamps, then it results in a multidimensional time-series data set.

The contextual attributes typically have a strong impact on the dependencies between the behavioral attribute values in the data. Formally, time-series data are defined as follows:

Definition 1.3.2 (Multivariate Time-Series Data) *A time series of length n and dimensionality d contains d numeric features at each of n time stamps $t_1 \ldots t_n$. Each time-stamp contains a component for each of the d series. Therefore, the set of values received at time stamp t_i is $\overline{Y_i} = (y_i^1 \ldots y_i^d)$. The value of the jth series at time stamp t_i is y_i^j.*

For example, consider the case where two sensors at a particular location monitor the temperature and pressure every second for a minute. This corresponds to a multidimensional series with $d = 2$ and $n = 60$. In some cases, the time stamps $t_1 \ldots t_n$ may be replaced by index values from 1 through n, especially when the time-stamp values are equally spaced apart.

Time-series data are relatively common in many sensor applications, forecasting, and financial market analysis. Methods for analyzing time series are discussed in Chap. 14.

1.3.2.2 Discrete Sequences and Strings

Discrete sequences can be considered the categorical analog of time-series data. As in the case of time-series data, the contextual attribute is a time stamp or a position index in the ordering. The behavioral attribute is a categorical value. Therefore, discrete sequence data are defined in a similar way to time-series data.

Definition 1.3.3 (Multivariate Discrete Sequence Data) *A discrete sequence of length n and dimensionality d contains d discrete feature values at each of n different time stamps $t_1 \ldots t_n$. Each of the n components $\overline{Y_i}$ contains d discrete behavioral attributes $(y_i^1 \ldots y_i^d)$, collected at the ith time-stamp.*

For example, consider a sequence of Web accesses, in which the Web page address and the originating IP address of the request are collected for 100 different accesses. This represents a discrete sequence of length $n = 100$ and dimensionality $d = 2$. A particularly common case in sequence data is the *univariate* scenario, in which the value of d is 1. Such sequence data are also referred to as *strings*.

It should be noted that the aforementioned definition is almost identical to the time-series case, with the main difference being that discrete sequences contain categorical attributes. In theory, it is possible to have series that are mixed between categorical and numerical data. Another important variation is the case where a sequence does not contain categorical attributes, but a *set* of any number of unordered categorical values. For example, supermarket transactions may contain a sequence of sets of items. Each set may contain any number of items. Such setwise sequences are not really multivariate sequences, but are univariate sequences, in which each element of the sequence is a *set* as opposed to a unit element. Thus, discrete sequences can be defined in a wider variety of ways, as compared to time-series data because of the ability to define sets on discrete elements.

In some cases, the contextual attribute may not refer to time explicitly, but it might be a position based on physical placement. This is the case for biological sequence data. In such cases, the time stamp may be replaced by an index representing the position of the value in the string, counting the leftmost position as 1. Some examples of common scenarios in which sequence data may arise are as follows:

- *Event logs:* A wide variety of computer systems, Web servers, and Web applications create event logs on the basis of user activity. An example of an event log is a sequence of user actions at a financial Web site:

  ```
  Login Password Login Password Login Password ....
  ```

  This particular sequence may represent a scenario where a user is attempting to break into a password-protected system, and it may be interesting from the perspective of anomaly detection.

- *Biological data:* In this case, the sequences may correspond to strings of nucleotides or amino acids. The ordering of such units provides information about the characteristics of protein function. Therefore, the data mining process can be used to determine interesting patterns that are reflective of different biological properties.

Discrete sequences are often more challenging for mining algorithms because they do not have the smooth value continuity of time-series data. Methods for sequence mining are discussed in Chap. 15.

1.3.2.3 Spatial Data

In spatial data, many nonspatial attributes (e.g., temperature, pressure, image pixel color intensity) are measured at spatial locations. For example, sea-surface temperatures are often collected by meteorologists to forecast the occurrence of hurricanes. In such cases, the spatial coordinates correspond to contextual attributes, whereas attributes such as the temperature correspond to the behavioral attributes. Typically, there are two spatial attributes. As in the case of time-series data, it is also possible to have multiple behavioral attributes. For example, in the sea-surface temperature application, one might also measure other behavioral attributes such as the pressure.

Definition 1.3.4 (Spatial Data) *A d-dimensional spatial data record contains d behavioral attributes and one or more contextual attributes containing the spatial location. Therefore, a d-dimensional spatial data set is a set of d dimensional records $\overline{X_1} \ldots \overline{X_n}$, together with a set of n locations $L_1 \ldots L_n$, such that the record $\overline{X_i}$ is associated with the location L_i.*

The aforementioned definition provides broad flexibility in terms of how record $\overline{X_i}$ and location L_i may be defined. For example, the behavioral attributes in record $\overline{X_i}$ may be numeric or categorical, or a mixture of the two. In the meteorological application, $\overline{X_i}$ may contain the temperature and pressure attributes at location L_i. Furthermore, L_i may be specified in terms of precise spatial coordinates, such as latitude and longitude, or in terms of a logical location, such as the city or state.

Spatial data mining is closely related to time-series data mining, in that the behavioral attributes in most commonly studied spatial applications are continuous, although some applications may use categorical attributes as well. Therefore, value continuity is observed across contiguous spatial locations, just as value continuity is observed across contiguous time stamps in time-series data.

Spatiotemporal Data

A particular form of spatial data is spatiotemporal data, which contains both spatial and temporal attributes. The precise nature of the data also depends on which of the attributes are contextual and which are behavioral. Two kinds of spatiotemporal data are most common:

1. *Both spatial and temporal attributes are contextual:* This kind of data can be viewed as a direct generalization of both spatial data and temporal data. This kind of data is particularly useful when the spatial and temporal dynamics of particular behavioral attributes are measured simultaneously. For example, consider the case where the variations in the sea-surface temperature need to be measured over time. In such cases, the temperature is the behavioral attribute, whereas the spatial and temporal attributes are contextual.

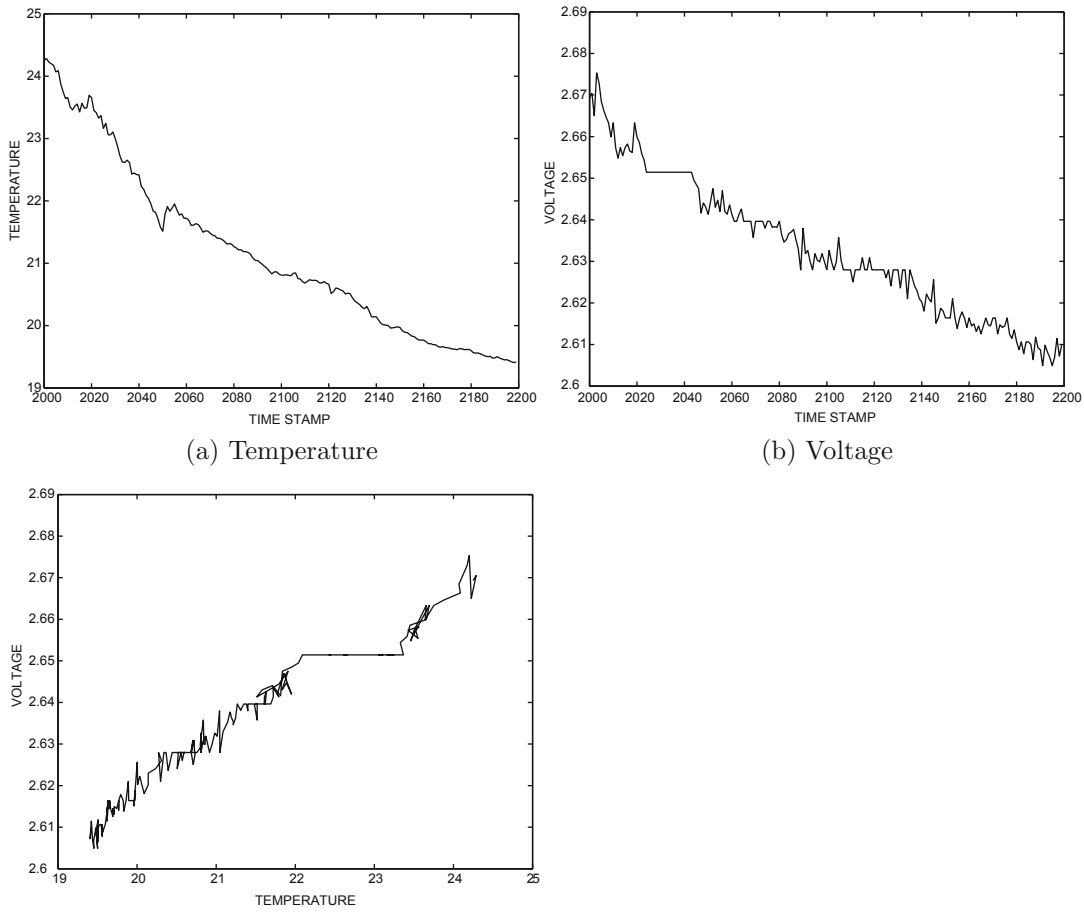
2. *The temporal attribute is contextual, whereas the spatial attributes are behavioral:* Strictly speaking, this kind of data can also be considered time-series data. However, the spatial nature of the behavioral attributes also provides better interpretability and more focused analysis in many scenarios. The most common form of this data arises in the context of *trajectory analysis*.

It should be pointed out that any 2- or 3-dimensional time-series data can be mapped onto trajectories. This is a useful transformation because it implies that trajectory mining algorithms can also be used for 2- or 3-dimensional time-series data. For example, the *Intel Research Berkeley data set* [556] contains readings from a variety of sensors. An example of a pair of readings from a temperature and voltage sensor are illustrated in Figs. 1.2a and b, respectively. The corresponding temperature–voltage trajectory is illustrated in Fig. 1.2c. Methods for spatial and spatiotemporal data mining are discussed in Chap. 16.

1.3.2.4 Network and Graph Data

In network and graph data, the data values may correspond to nodes in the network, whereas the relationships among the data values may correspond to the edges in the network. In some cases, attributes may be associated with nodes in the network. Although it is also possible to associate attributes with edges in the network, it is much less common to do so.

Definition 1.3.5 (Network Data) *A network $G = (N, A)$ contains a set of nodes N and a set of edges A, where the edges in A represent the relationships between the nodes. In*



(a) Temperature

(b) Voltage

(c) Temperature-voltage trajectory

Figure 1.2: Mapping of multivariate time series to trajectory data

some cases, an attribute set $\overline{X_i}$ *may be associated with node* i*, or an attribute set* $\overline{Y_{ij}}$ *may be associated with edge* (i, j)*.*

The edge (i, j) may be directed or undirected, depending on the application at hand. For example, the Web graph may contain directed edges corresponding to directions of hyperlinks between pages, whereas friendships in the Facebook social network are undirected.

A second class of graph mining problems is that of a database containing many small graphs such as chemical compounds. The challenges in these two classes of problems are very different. Some examples of data that are represented as graphs are as follows:

- *Web graph:* The nodes correspond to the Web pages, and the edges correspond to hyperlinks. The nodes have text attributes corresponding to the content in the page.
- *Social networks:* In this case, the nodes correspond to social network actors, whereas the edges correspond to friendship links. The nodes may have attributes corresponding to social page content. In some specialized forms of social networks, such as email or

chat-messenger networks, the edges may have content associated with them. This content corresponds to the communication between the different nodes.

- *Chemical compound databases:* In this case, the nodes correspond to the elements and the edges correspond to the chemical bonds between the elements. The structures in these chemical compounds are very useful for identifying important reactive and pharmacological properties of these compounds.

Network data are a very general representation and can be used for solving many similarity-based applications on other data types. For example, multidimensional data may be converted to network data by creating a node for each record in the database, and representing similarities between nodes by edges. Such a representation is used quite often for many similarity-based data mining applications, such as clustering. It is possible to use community detection algorithms to determine clusters in the network data and then map them back to multidimensional data. Some spectral clustering methods, discussed in Chap. 19, are based on this principle. This generality of network data comes at a price. The development of mining algorithms for network data is generally more difficult. Methods for mining network data are discussed in Chaps. 17, 18, and 19.

1.4 The Major Building Blocks: A Bird's Eye View

As discussed in the introduction Sect. 1.1, four problems in data mining are considered fundamental to the mining process. These problems correspond to clustering, classification, association pattern mining, and outlier detection, and they are encountered repeatedly in the context of many data mining applications. What makes these problems so special? Why are they encountered repeatedly? To answer these questions, one must understand the nature of the typical relationships that data scientists often try to extract from the data.

Consider a multidimensional database $\mathcal{D}$ with n records, and d attributes. Such a database $\mathcal{D}$ may be represented as an $n \times d$ matrix D, in which each row corresponds to one record and each column corresponds to a dimension. We generally refer to this matrix as the *data matrix*. This book will use the notation of a data matrix D, and a database $\mathcal{D}$ interchangeably. Broadly speaking, data mining is all about finding summary relationships between the entries in the data matrix that are either unusually frequent or unusually infrequent. Relationships between data items are one of two kinds:

- *Relationships between columns:* In this case, the frequent or infrequent relationships between the values in a particular row are determined. This maps into either the positive or negative association pattern mining problem, though the former is more commonly studied. In some cases, one particular column of the matrix is considered more important than other columns because it represents a target attribute of the data mining analyst. In such cases, one tries to determine how the relationships in the other columns relate to this special column. Such relationships can be used to predict the value of this special column, when the value of that special column is unknown. This problem is referred to as *data classification*. A mining process is referred to as *supervised* when it is based on treating a particular attribute as special and predicting it.

- *Relationships between rows:* In these cases, the goal is to determine subsets of rows, in which the values in the corresponding columns are related. In cases where these subsets are similar, the corresponding problem is referred to as *clustering*. On the other hand,

when the entries in a row are very different from the corresponding entries in other rows, then the corresponding row becomes interesting as an unusual data point, or as an *anomaly*. This problem is referred to as *outlier analysis*. Interestingly, the clustering problem is closely related to that of classification, in that the latter can be considered a supervised version of the former. The discrete values of a special column in the data correspond to the group identifiers of different *desired* or *supervised* groups of application-specific similar records in the data. For example, when the special column corresponds to whether or not a customer is interested in a particular product, this represents the two groups in the data that one is interested in *learning*, with the use of *supervision*. The term "supervision" refers to the fact that the special column is used to direct the data mining process in an application-specific way, just as a teacher may supervise his or her student toward a specific goal.

Thus, these four problems are important because they seem to cover an exhaustive range of scenarios representing different kinds of positive, negative, supervised, or unsupervised relationships between the entries of the data matrix. These problems are also related to one another in a variety of ways. For example, association patterns may be considered indirect representations of (overlapping) clusters, where each pattern corresponds to a cluster of data points of which it is a subset.

It should be pointed out that the aforementioned discussion assumes the (most commonly encountered) multidimensional data type, although these problems continue to retain their relative importance for more complex data types. However, the more complex data types have a wider variety of problem formulations associated with them because of their greater complexity. This issue will be discussed in detail later in this section.

It has consistently been observed that many application scenarios determine such relationships between rows and columns of the data matrix as an intermediate step. This is the reason that a good understanding of these building-block problems is so important for the data mining process. Therefore, the first part of this book will focus on these problems in detail before generalizing to complex scenarios.

1.4.1 Association Pattern Mining

In its most primitive form, the association pattern mining problem is defined in the context of *sparse binary databases*, where the data matrix contains only 0/1 entries, and most entries take on the value of 0. Most customer transaction databases are of this type. For example, if each column in the data matrix corresponds to an item, and a customer transaction represents a row, the (i, j)th entry is 1, if customer transaction i contains item j as one of the items that was bought. A particularly commonly studied version of this problem is the frequent pattern mining problem or, more generally, the association pattern mining problem. In terms of the binary data matrix, the frequent pattern mining problem may be formally defined as follows:

Definition 1.4.1 (Frequent Pattern Mining) *Given a binary $n \times d$ data matrix D, determine all subsets of columns such that all the values in these columns take on the value of 1 for at least a fraction s of the rows in the matrix. The relative frequency of a pattern is referred to as its support. The fraction s is referred to as the minimum support.*

Patterns that satisfy the minimum support requirement are often referred to as *frequent patterns*, or *frequent itemsets*. Frequent patterns represent an important class of association patterns. Many other definitions of relevant association patterns are possible that do not use

absolute frequencies but use other statistical quantifications such as the χ^2 measure. These measures often lead to generation of more *interesting* rules from a statistical perspective. Nevertheless, this particular definition of association pattern mining has become the most popular one in the literature because of the ease in developing algorithms for it. This book therefore refers to this problem as *association pattern mining* as opposed to *frequent pattern mining*.

For example, if the columns of the data matrix D corresponding to *Bread*, *Butter*, and *Milk* take on the value of 1 together frequently in a customer transaction database, then it implies that these items are often bought together. This is very useful information for the merchant from the perspective of physical placement of the items in the store, or from the perspective of product promotions. Association pattern mining is not restricted to the case of binary data and can be easily generalized to quantitative and numeric attributes by using appropriate data transformations, which will be discussed in Chap. 4.

Association pattern mining was originally proposed in the context of *association rule mining*, where an additional step was included based on a measure known as the *confidence* of the rule. For example, consider two sets of items A and B. The confidence of the rule $A \Rightarrow B$ is defined as the fraction of transactions containing A, which also contain B. In other words, the confidence is obtained by dividing the support of the pattern $A \cup B$ with the support of pattern A. A combination of support and confidence is used to define association rules.

Definition 1.4.2 (Association Rules) *Let A and B be two sets of items. The rule $A \Rightarrow B$ is said to be valid at support level s and confidence level c, if the following two conditions are satisfied:*

1. *The support of the item set A is at least s.*
2. *The confidence of $A \Rightarrow B$ is at least c.*

By incorporating supervision in association rule mining algorithms, it is possible to provide solutions for the classification problem. Many variations of association pattern mining are also related to clustering and outlier analysis. This is a natural consequence of the fact that horizontal and vertical analysis of the data matrix are often related to one another. In fact, many variations of the association pattern mining problem are used as a subroutine to solve the clustering, outlier analysis, and classification problems. These issues will be discussed in Chaps. 4 and 5.

1.4.2 Data Clustering

A rather broad and informal definition of the clustering problem is as follows:

Definition 1.4.3 (Data Clustering) *Given a data matrix D (database $\mathcal{D}$), partition its rows (records) into sets $\mathcal{C}_1 \ldots \mathcal{C}_k$, such that the rows (records) in each cluster are "similar" to one another.*

We have intentionally provided an informal definition here because clustering allows a wide variety of definitions of similarity, some of which are not cleanly defined in closed form by a similarity function. A clustering problem can often be defined as an optimization problem, in which the variables of the optimization problem represent cluster memberships of data points, and the objective function maximizes a concrete mathematical quantification of intragroup similarity in terms of these variables.

An important part of the clustering process is the design of an appropriate similarity function for the computation process. Clearly, the computation of similarity depends heavily on the underlying data type. The issue of similarity computation will be discussed in detail in Chap. 3. Some examples of relevant applications are as follows:

- *Customer segmentation:* In many applications, it is desirable to determine customers that are similar to one another in the context of a variety of product promotion tasks. The segmentation phase plays an important role in this process.
- *Data summarization:* Because clusters can be considered similar groups of records, these similar groups can be used to create a summary of the data.
- *Application to other data mining problems:* Because clustering is considered an unsupervised version of classification, it is often used as a building block to solve the latter. Furthermore, this problem is also used in the context of the outlier analysis problem, as discussed below.

The data clustering problem is discussed in detail in Chaps. 6 and 7.

1.4.3 Outlier Detection

An outlier is a data point that is significantly different from the remaining data. Hawkins formally defined [259] the concept of an outlier as follows:
"*An outlier is an observation that deviates so much from the other observations as to arouse suspicions that it was generated by a different mechanism.*"

Outliers are also referred to as *abnormalities*, *discordants*, *deviants*, or *anomalies* in the data mining and statistics literature. In most applications, the data are created by one or more generating processes that can either reflect activity in the system or observations collected about entities. When the generating process behaves in an unusual way, it results in the creation of outliers. Therefore, an outlier often contains useful information about abnormal characteristics of the systems and entities that impact the data-generation process. The recognition of such unusual characteristics provides useful application-specific insights. The outlier detection problem is informally defined in terms of the data matrix as follows:

Definition 1.4.4 (Outlier Detection) *Given a data matrix D, determine the rows of the data matrix that are very different from the remaining rows in the matrix.*

The outlier detection problem is related to the clustering problem by complementarity. This is because outliers correspond to dissimilar data points from the main groups in the data. On the other hand, the main groups in the data are clusters. In fact, a simple methodology to determine outliers uses clustering as an intermediate step. Some examples of relevant applications are as follows:

- *Intrusion-detection systems:* In many networked computer systems, different kinds of data are collected about the operating system calls, network traffic, or other activity in the system. These data may show unusual behavior because of malicious activity. The detection of such activity is referred to as intrusion detection.
- *Credit card fraud:* Unauthorized use of credit cards may show different patterns, such as a buying spree from geographically obscure locations. Such patterns may show up as outliers in credit card transaction data.

- *Interesting sensor events:* Sensors are often used to track various environmental and location parameters in many real applications. The sudden changes in the underlying patterns may represent events of interest. Event detection is one of the primary motivating applications in the field of sensor networks.

- *Medical diagnosis:* In many medical applications, the data are collected from a variety of devices such as magnetic resonance imaging (MRI), positron emission tomography (PET) scans, or electrocardiogram (ECG) time series. Unusual patterns in such data typically reflect disease conditions.

- *Law enforcement:* Outlier detection finds numerous applications in law enforcement, especially in cases where unusual patterns can only be discovered over time through multiple actions of an entity. The identification of fraud in financial transactions, trading activity, or insurance claims typically requires the determination of unusual patterns in the data generated by the actions of the criminal entity.

- *Earth science:* A significant amount of spatiotemporal data about weather patterns, climate changes, or land-cover patterns is collected through a variety of mechanisms such as satellites or remote sensing. Anomalies in such data provide significant insights about hidden human or environmental trends that may have caused such anomalies.

The outlier detection problem is studied in detail in Chaps. 8 and 9.

1.4.4 Data Classification

Many data mining problems are directed toward a specialized goal that is sometimes represented by the value of a particular feature in the data. This particular feature is referred to as the *class label*. Therefore, such problems are *supervised*, wherein the relationships of the remaining features in the data with respect to this special feature are *learned*. The data used to learn these relationships is referred to as the *training data*. The *learned model* may then be used to determine the estimated class labels for records, where the label is missing.

For example, in a target marketing application, each record may be tagged by a particular *label* that represents the interest (or lack of it) of the customer toward a particular product. The labels associated with customers may have been derived from the previous buying behavior of the customer. In addition, a set of features corresponding the customer demographics may also be available. The goal is to predict whether or not a customer, whose buying behavior is unknown, will be interested in a particular product by relating the demographic features to the class label. Therefore, a *training model* is constructed, which is then used to predict class labels. The classification problem is informally defined as follows:

Definition 1.4.5 (Data Classification) *Given an $n \times d$ training data matrix D (database $\mathcal{D}$), and a class label value in $\{1 \ldots k\}$ associated with each of the n rows in D (records in $\mathcal{D}$), create a training model $\mathcal{M}$, which can be used to predict the class label of a d-dimensional record $\overline{Y} \notin \mathcal{D}$.*

The record whose class label is unknown is referred to as the *test record*. It is interesting to examine the relationship between the clustering and the classification problems. In the case of the clustering problem, the data are partitioned into k groups on the basis of similarity. In the case of the classification problem, a (test) record is also categorized into one of k groups, except that this is achieved by learning a model from a training database $\mathcal{D}$, rather than on the basis of similarity. In other words, the supervision from the training data redefines the

notion of a group of "similar" records. Therefore, from a learning perspective, clustering is often referred to as *unsupervised learning* (because of the lack of a special training database to "teach" the model about the notion of an appropriate grouping), whereas the classification problem is referred to as *supervised learning*.

The classification problem is related to association pattern mining, in the sense that the latter problem is often used to solve the former. This is because if the entire training database (including the class label) is treated as an $n \times (d+1)$ matrix, then frequent patterns containing the class label in this matrix provide useful hints about the correlations of other features to the class label. In fact, many forms of classifiers, known as *rule-based classifiers*, are based on this broader principle.

The classification problem can be mapped to a specific version of the outlier detection problem, by incorporating supervision in the latter. While the outlier detection problem is assumed to be unsupervised by default, many variations of the problem are either partially or fully supervised. In supervised outlier detection, some examples of outliers are available. Thus, such data records are tagged to belong to a *rare class*, whereas the remaining data records belong to the *normal class*. Thus, the supervised outlier detection problem maps to a binary classification problem, with the caveat that the class labels are highly *imbalanced*.

The incorporation of supervision makes the classification problem unique in terms of its *direct* application specificity due to its use of application-specific class labels. Compared to the other major data mining problems, the classification problem is relatively self-contained. For example, the clustering and frequent pattern mining problem are more often used as intermediate steps in larger application frameworks. Even the outlier analysis problem is sometimes used in an exploratory way. On the other hand, the classification problem is often used directly as a stand-alone tool in many applications. Some examples of applications where the classification problem is used are as follows:

- *Target marketing:* Features about customers are related to their buying behavior with the use of a training model.
- *Intrusion detection:* The sequences of customer activity in a computer system may be used to predict the possibility of intrusions.
- *Supervised anomaly detection:* The rare class may be differentiated from the normal class when previous examples of outliers are available.

The data classification problem is discussed in detail in Chaps. 10 and 11.

1.4.5 Impact of Complex Data Types on Problem Definitions

The specific data type has a profound impact on the kinds of problems that may be defined. In particular, in dependency-oriented data types, the dependencies often play a critical role in the problem definition, the solution, or both. This is because the contextual attributes and dependencies are often fundamental to how the data may be evaluated. Furthermore, because complex data types are much richer, they allow the formulation of novel problem definitions that may not even exist in the context of multidimensional data. A tabular summary of the different variations of data mining problems for dependency-oriented data types is provided in Table 1.2. In the following, a brief review will be provided as to how the different problem definitions are affected by data type.

Table 1.2: Some examples of variation in problem definition with data type

Problem	Time series	Spatial	Sequence	Networks
Patterns	Motif-mining Periodic pattern	Colocation patterns	Sequential patterns Periodic Sequence	Structural patterns
	Trajectory patterns			
Clustering	Shape clusters	Spatial clusters	Sequence clusters	Community detection
	Trajectory clusters			
Outliers	Position outlier Shape outlier	Position outlier Shape outlier	Position outlier Combination outlier	Node outlier Linkage outlier Community outliers
	Trajectory outliers			
Classification	Position classification Shape classification	Position classification Shape classification	Position classification Sequence classification	Collective classification Graph classification
	Trajectory classification			

1.4.5.1 Pattern Mining with Complex Data Types

The association pattern mining problem generally determines the patterns from the underlying data in the form of sets; however, this is not the case when dependencies are present in the data. This is because the dependencies and relationships often impose ordering among data items, and the direct use of frequent pattern mining methods fails to recognize the relationships among the different data values. For example, when a larger number of time series are made available, they can be used to determine different kinds of *temporally* frequent patterns, in which a temporal ordering is imposed on the items in the pattern. Furthermore, because of the presence of the additional contextual attribute representing time, temporal patterns may be defined in a much richer way than a set-based pattern as in association pattern mining. The patterns may be temporally contiguous, as in *time-series motifs*, or they may be periodic, as in *periodic patterns*. Some of these methods for temporal pattern mining will be discussed in Chap. 14. A similar analogy exists for the case of discrete sequence mining, except that the individual pattern constituents are categorical, as opposed to continuous. It is also possible to define 2-dimensional motifs for the spatial scenario, and such a formulation is useful for image processing. Finally, structural patterns are commonly defined in networks that correspond to frequent subgraphs in the data. Thus, the dependencies between the nodes are included within the definition of the patterns.

1.4.5.2 Clustering with Complex Data Types

The techniques used for clustering are also affected significantly by the underlying data type. Most importantly, the similarity function is significantly affected by the data type. For example, in the case of time series, sequential, or graph data, the similarity between a pair of time series cannot be easily defined by using straightforward metrics such as the Euclidean metric. Rather, it is necessary to use other kinds of metrics, such as the edit distance or structural similarity. In the context of spatial data, trajectory clustering is particularly useful in finding the relevant patterns for mobile data, or for multivariate

time series. For network data, the clustering problem discovers densely connected groups of nodes, and is also referred to as *community detection*.

1.4.5.3 Outlier Detection with Complex Data Types

Dependencies can be used to define expected values of data items. Deviations from these expected values are outliers. For example, a sudden jump in the value of a time series will result in a position outlier at the specific spot at which the jump occurs. The idea in these methods is to use *prediction-based techniques* to forecast the value at that position. Significant deviation from the prediction is reported as a *position outlier*. Such outliers can be defined in the context of time-series, spatial, and sequential data, where significant deviations from the corresponding neighborhoods can be detected using autoregressive, Markovian, or other models. In the context of graph data, outliers may correspond to unusual properties of nodes, edges, or entire subgraphs. Thus, the complex data types show significant richness in terms of how outliers may be defined.

1.4.5.4 Classification with Complex Data Types

The classification problem also shows a significant amount of variation in the different complex data types. For example, class labels can be attached to specific positions in a series, or they can be attached to the entire series. When the class labels are attached to a specific position in the series, this can be used to perform supervised event detection, where the first occurrence of an event-specific label (e.g., the breakdown of a machine as suggested by the underlying temperature and pressure sensor) of a particular series represents the occurrence of the event. For the case of network data, the labels may be attached to individual nodes in a very large network, or to entire graphs in a collection of multiple graphs. The former case corresponds to the classification of nodes in a social network, and is also referred to as *collective classification*. The latter case corresponds to the chemical compound classification problem, in which labels are attached to compounds on the basis of their chemical properties.

1.5 Scalability Issues and the Streaming Scenario

Scalability is an important concern in many data mining applications due to the increasing sizes of the data in modern-day applications. Broadly speaking, there are two important scenarios for scalability:

1. The data are stored on one or more machines, but it is too large to process efficiently. For example, it is easy to design efficient algorithms in cases where the entire data can be maintained in main memory. When the data are stored on disk, it is important to be design the algorithms in such a way that random access to the disk is minimized. For very large data sets, big data frameworks, such as MapReduce, may need to be used. This book will touch upon this kind of scalability at the level of disk-resident processing, where needed.

2. The data are generated continuously over time in high volume, and it is not practical to store it entirely. This scenario is that of *data streams*, in which the data need to be processed with the use of an online approach.

The latter scenario requires some further exposition. The streaming scenario has become increasingly popular because of advances in data collection technology that can collect large amounts of data over time. For example, simple transactions of everyday life such as using a credit card or the phone may lead to automated data collection. In such cases, the volume of the data is so large that it may be impractical to store directly. Rather, all algorithms must be executed in a single pass over the data. The major challenges that arise in the context of data stream processing are as follows:

1. *One-pass constraint:* The algorithm needs to process the entire data set in one pass. In other words, after a data item has been processed and the relevant summary insights have been gleaned, the raw item is discarded and is no longer available for processing. The amount of data that may be processed at a given time depends on the storage available for retaining segments of the data.
2. *Concept drift:* In most applications, the data distribution changes over time. For example, the pattern of sales in a given hour of a day may not be similar to that at another hour of the day. This leads to changes in the output of the mining algorithms as well.

It is often challenging to design algorithms for such scenarios because of the varying rates at which the patterns in the data may change over time and the continuously evolving patterns in the underlying data. Methods for stream mining are addressed in Chap. 12.

1.6 A Stroll Through Some Application Scenarios

In this section, some common application scenarios will be discussed. The goal is to illustrate the wide diversity of problems and applications, and how they might map onto some of the building blocks discussed in this chapter.

1.6.1 Store Product Placement

The application scenario may be stated as follows:

Application 1.6.1 (Store Product Placement) *A merchant has a set of d products together with previous transactions from the customers containing baskets of items bought together. The merchant would like to know how to place the product on the shelves to increase the likelihood that items that are frequently bought together are placed on adjacent shelves.*

This problem is closely related to frequent pattern mining because the analyst can use the frequent pattern mining problem to determine groups of items that are frequently bought together at a particular support level. An important point to note here is that the determination of the frequent patterns, while providing useful insights, does not provide the merchant with precise guidance in terms of how the products may be placed on the different shelves. This situation is quite common in data mining. The building block problems often do not directly solve the problem at hand. In this particular case, the merchant may choose from a variety of heuristic ideas in terms of how the products may be stocked on the different shelves. For example, the merchant may already have an existing placement, and may use the frequent patterns to create a numerical score for the quality of the placement. This placement can be successively optimized by making incremental changes to the current placement. With an appropriate initialization methodology, the frequent pattern mining approach can be leveraged as a very useful subroutine for the problem. These parts of data mining are often application-specific and show such wide variations across different domains that they can only be learned through practical experience.

1.6.2 Customer Recommendations

This is a very commonly encountered problem in the data mining literature. Many variations of this problem exist, depending on the kind of input data available to that application. In the following, we will examine a particular instantiation of the recommendation problem and a straw-man solution.

Application 1.6.2 (Product Recommendations) *A merchant has an $n \times d$ binary matrix D representing the buying behavior of n customers across d items. It is assumed that the matrix is sparse, and therefore each customer may have bought only a few items. It is desirable to use the product associations to make recommendations to customers.*

This problem is a simple version of the collaborative filtering problem that is widely studied in the data mining and recommendation literature. There are literally hundreds of solutions to the vanilla version of this problem, and we provide three sample examples of varying complexity below:

1. A simple solution is to use association rule mining at particular levels of support and confidence. For a particular customer, the relevant rules are those in which all items in the left-hand side were previously bought by this customer. Items that appear frequently on the right-hand side of the relevant rules are reported.

2. The previous solution does not use the similarity across different customers to make recommendations. A second solution is to determine the most similar rows to a target customer, and then recommend the most common item occurring in these similar rows.

3. A final solution is to use clustering to create segments of similar customers. Within each similar segment, association pattern mining may be used to make recommendations.

Thus, there can be multiple ways of solving a particular problem corresponding to different analytical paths. These different paths may use different kinds of building blocks, which are all useful in different parts of the data mining process.

1.6.3 Medical Diagnosis

Medical diagnosis has become a common application in the context of data mining. The data types in medical diagnosis tend to be complex, and may correspond to image, time-series, or discrete sequence data. Thus, dependency-oriented data types tend to be rather common in medical diagnosis applications. A particular case is that of ECG readings from heart patients.

Application 1.6.3 (Medical ECG Diagnosis) *Consider a set of ECG time series that are collected from different patients. It is desirable to determine the anomalous series from this set.*

This application can be mapped to different problems, depending upon the nature of the input data available. For example, consider the case where no previous examples of anomalous ECG series are available. In such cases, the problem can be mapped to the outlier detection problem. A time series that differs significantly from the remaining series in the data may be considered an outlier. However, the solution methodology changes significantly

if previous examples of normal and anomalous series are available. In such cases, the problem maps to a classification problem on time-series data. Furthermore, the class labels are likely to be imbalanced because the number of abnormal series are usually far fewer than the number of normal series.

1.6.4 Web Log Anomalies

Web logs are commonly collected at the hosts of different Web sites. Such logs can be used to detect unusual, suspicious, or malicious activity at the site. Financial institutions regularly analyze the logs at their site to detect intrusion attempts.

Application 1.6.4 (Web Log Anomalies) *A set of Web logs is available. It is desired to determine the anomalous sequences from the Web logs.*

Because the data are typically available in the form of raw logs, a significant amount of data cleaning is required. First, the raw logs need to be transformed into sequences of symbols. These sequences may then need to be decomposed into smaller windows to analyze the sequences at a particular level of granularity. Anomalous sequences may be determined by using a sequence clustering algorithm, and then determining the sequences that do not lie in these clusters [5]. If it is desired to find specific positions that correspond to anomalies, then more sophisticated methods such as Markovian models may be used to determine the anomalies [5].

As in the previous case, the analytical phase of this problem can be modeled differently, depending on whether or not examples of Web log anomalies are available. If no previous examples of Web log anomalies are available, then this problem maps to the unsupervised temporal outlier detection problem. Numerous methods for solving the unsupervised case for the temporal outlier detection problem are introduced in [5]. The topic is also briefly discussed in Chaps. 14 and 15 of this book. On the other hand, when examples of previous anomalies are available, then the problem maps to the rare class-detection problem. This problem is discussed in [5] as well, and in Chap. 11 of this book.

1.7 Summary

Data mining is a complex and multistage process. These different stages are data collection, preprocessing, and analysis. The data preprocessing phase is highly application-specific because the different formats of the data require different algorithms to be applied to them. The processing phase may include data integration, cleaning, and feature extraction. In some cases, feature selection may also be used to sharpen the data representation. After the data have been converted to a convenient format, a variety of analytical algorithms can be used.

A number of data mining building blocks are often used repeatedly in a wide variety of application scenarios. These correspond to the frequent pattern mining, clustering, outlier analysis, and classification problems, respectively. The final design of a solution for a particular data mining problem is dependent on the skill of the analyst in mapping the application to the different building blocks, or in using novel algorithms for a specific application. This book will introduce the fundamentals required for gaining such analytical skills.