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

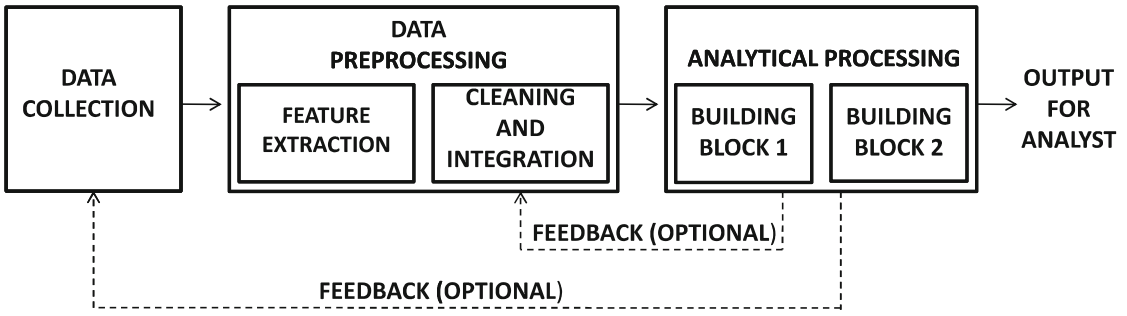

Figure 1.1: The data processing pipeline

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.*