
2 Data

This chapter discusses several data-related issues that are important for successful data mining:

The Type of Data Data sets differ in a number of ways. For example, the attributes used to describe data objects can be of different types—quantitative or qualitative—and data sets often have special characteristics; e.g., some data sets contain time series or objects with explicit relationships to one another. Not surprisingly, the type of data determines which tools and techniques can be used to analyze the data. Indeed, new research in data mining is often driven by the need to accommodate new application areas and their new types of data.

The Quality of the Data Data is often far from perfect. While most data mining techniques can tolerate some level of imperfection in the data, a focus on understanding and improving data quality typically improves the quality of the resulting analysis. Data quality issues that often need to be addressed include the presence of noise and outliers; missing, inconsistent, or duplicate data; and data that is biased or, in some other way, unrepresentative of the phenomenon or population that the data is supposed to describe.

Preprocessing Steps to Make the Data More Suitable for Data Mining

Often, the raw data must be processed in order to make it suitable for

analysis. While one objective may be to improve data quality, other goals focus on modifying the data so that it better fits a specified data mining technique or tool. For example, a continuous attribute, e.g., length, sometimes needs to be transformed into an attribute with discrete categories, e.g., *short*, *medium*, or *long*, in order to apply a particular technique. As another example, the number of attributes in a data set is often reduced because many techniques are more effective when the data has a relatively small number of attributes.

Analyzing Data in Terms of Its Relationships One approach to data analysis is to find relationships among the data objects and then perform the remaining analysis using these relationships rather than the data objects themselves. For instance, we can compute the similarity or distance between pairs of objects and then perform the analysis—clustering, classification, or anomaly detection—based on these similarities or distances. There are many such similarity or distance measures, and the proper choice depends on the type of data and the particular application.

Example 2.1 (An Illustration of Data-Related Issues).

To further illustrate the importance of these issues, consider the following hypothetical situation. You receive an email from a medical researcher concerning a project that you are eager to work on.

Hi,

I've attached the data file that I mentioned in my previous email. Each line contains the information for a single patient and consists of five fields. We want to predict the last field using the other fields. I don't have time to provide any more information about the data since I'm going out of town for a couple of days, but hopefully that won't slow you down too much. And if you

don't mind, could we meet when I get back to discuss your preliminary results? I might invite a few other members of my team.

Thanks and see you in a couple of days.

Despite some misgivings, you proceed to analyze the data. The first few rows of the file are as follows:

012	232	33.5	0	10.7
020	121	16.9	2	210.1
027	165	24.0	0	427.6
⋮				

A brief look at the data reveals nothing strange. You put your doubts aside and start the analysis. There are only 1000 lines, a smaller data file than you had hoped for, but two days later, you feel that you have made some progress. You arrive for the meeting, and while waiting for others to arrive, you strike up a conversation with a statistician who is working on the project. When she learns that you have also been analyzing the data from the project, she asks if you would mind giving her a brief overview of your results.

Statistician: So, you got the data for all the patients?

Data Miner: Yes. I haven't had much time for analysis, but I do have a few interesting results.

Statistician: Amazing. There were so many data issues with this set of patients that I couldn't do much.

Data Miner: Oh? I didn't hear about any possible problems.

Statistician: Well, first there is field 5, the variable we want to predict.

It's common knowledge among people who analyze this type of data that results are better if you work with the log of the values, but I didn't discover this until later. Was it mentioned to you?

Data Miner: No.

Statistician: But surely you heard about what happened to field 4? It's supposed to be measured on a scale from 1 to 10, with 0 indicating a missing value, but because of a data entry error, all 10's were changed into 0's. Unfortunately, since some of the patients have missing values for this field, it's impossible to say whether a 0 in this field is a real 0 or a 10. Quite a few of the records have that problem.

Data Miner: Interesting. Were there any other problems?

Statistician: Yes, fields 2 and 3 are basically the same, but I assume that you probably noticed that.

Data Miner: Yes, but these fields were only weak predictors of field 5.

Statistician: Anyway, given all those problems, I'm surprised you were able to accomplish anything.

Data Miner: True, but my results are really quite good. Field 1 is a very strong predictor of field 5. I'm surprised that this wasn't noticed before.

Statistician: What? Field 1 is just an identification number.

Data Miner: Nonetheless, my results speak for themselves.

Statistician: Oh, no! I just remembered. We assigned ID numbers after we sorted the records based on field 5. There is a strong connection, but it's meaningless. Sorry.

Although this scenario represents an extreme situation, it emphasizes the importance of "knowing your data." To that end, this chapter will address each

of the four issues mentioned above, outlining some of the basic challenges and standard approaches.

2.1 Types of Data

A **data set** can often be viewed as a collection of **data objects**. Other names for a data object are *record*, *point*, *vector*, *pattern*, *event*, *case*, *sample*, *instance*, *observation*, or *entity*. In turn, data objects are described by a number of **attributes** that capture the characteristics of an object, such as the mass of a physical object or the time at which an event occurred. Other names for an attribute are *variable*, *characteristic*, *field*, *feature*, or *dimension*.

Example 2.2 (Student Information).


Often, a data set is a file, in which the objects are records (or rows) in the file and each field (or column) corresponds to an attribute. For example, **Table 2.1**  shows a data set that consists of student information. Each row corresponds to a student and each column is an attribute that describes some aspect of a student, such as grade point average (GPA) or identification number (ID).

Table 2.1. A sample data set containing student information.

Student ID	Year	Grade Point Average (GPA)	...
	⋮		
1034262	Senior	3.24	...
1052663	Freshman	3.51	...
1082246	Sophomore	3.62	...

Although record-based data sets are common, either in flat files or relational database systems, there are other important types of data sets and systems for storing data. In [Section 2.1.2](#), we will discuss some of the types of data sets that are commonly encountered in data mining. However, we first consider attributes.

2.1.1 Attributes and Measurement

In this section, we consider the types of attributes used to describe data objects. We first define an attribute, then consider what we mean by the type of an attribute, and finally describe the types of attributes that are commonly encountered.

What Is an Attribute?

We start with a more detailed definition of an attribute.

Definition 2.1.

An **attribute** is a property or characteristic of an object that can vary, either from one object to another or from one time to another.

For example, eye color varies from person to person, while the temperature of an object varies over time. Note that eye color is a symbolic attribute with a

small number of possible values *{brown, black, blue, green, hazel, etc.}* , while temperature is a numerical attribute with a potentially unlimited number of values.

At the most basic level, attributes are not about numbers or symbols. However, to discuss and more precisely analyze the characteristics of objects, we assign numbers or symbols to them. To do this in a well-defined way, we need a measurement scale.

Definition 2.2.

A **measurement scale** is a rule (function) that associates a numerical or symbolic value with an attribute of an object.

Formally, the process of **measurement** is the application of a measurement scale to associate a value with a particular attribute of a specific object. While this may seem a bit abstract, we engage in the process of measurement all the time. For instance, we step on a bathroom scale to determine our weight, we classify someone as male or female, or we count the number of chairs in a room to see if there will be enough to seat all the people coming to a meeting. In all these cases, the “physical value” of an attribute of an object is mapped to a numerical or symbolic value.

With this background, we can discuss the type of an attribute, a concept that is important in determining if a particular data analysis technique is consistent with a specific type of attribute.


The Type of an Attribute

It is common to refer to the type of an attribute as the **type of a measurement scale**. It should be apparent from the previous discussion that an attribute can be described using different measurement scales and that the properties of an attribute need not be the same as the properties of the values used to measure it. In other words, the values used to represent an attribute can have properties that are not properties of the attribute itself, and vice versa. This is illustrated with two examples.

Example 2.3 (Employee Age and ID Number).

Two attributes that might be associated with an employee are *ID* and *age* (in years). Both of these attributes can be represented as integers. However, while it is reasonable to talk about the average age of an employee, it makes no sense to talk about the average employee ID. Indeed, the only aspect of employees that we want to capture with the ID attribute is that they are distinct. Consequently, the only valid operation for employee IDs is to test whether they are equal. There is no hint of this limitation, however, when integers are used to represent the employee ID attribute. For the age attribute, the properties of the integers used to represent age are very much the properties of the attribute. Even so, the correspondence is not complete because, for example, ages have a maximum, while integers do not.

Example 2.4 (Length of Line Segments).

Consider [Figure 2.1](#) , which shows some objects—line segments—and how the length attribute of these objects can be mapped to numbers in two different ways. Each successive line segment, going from the top to the bottom, is formed by appending the topmost line segment to itself. Thus,

the second line segment from the top is formed by appending the topmost line segment to itself twice, the third line segment from the top is formed by appending the topmost line segment to itself three times, and so forth. In a very real (physical) sense, all the line segments are multiples of the first. This fact is captured by the measurements on the right side of the figure, but not by those on the left side. More specifically, the measurement scale on the left side captures only the ordering of the length attribute, while the scale on the right side captures both the ordering and additivity properties. Thus, an attribute can be measured in a way that does not capture all the properties of the attribute.

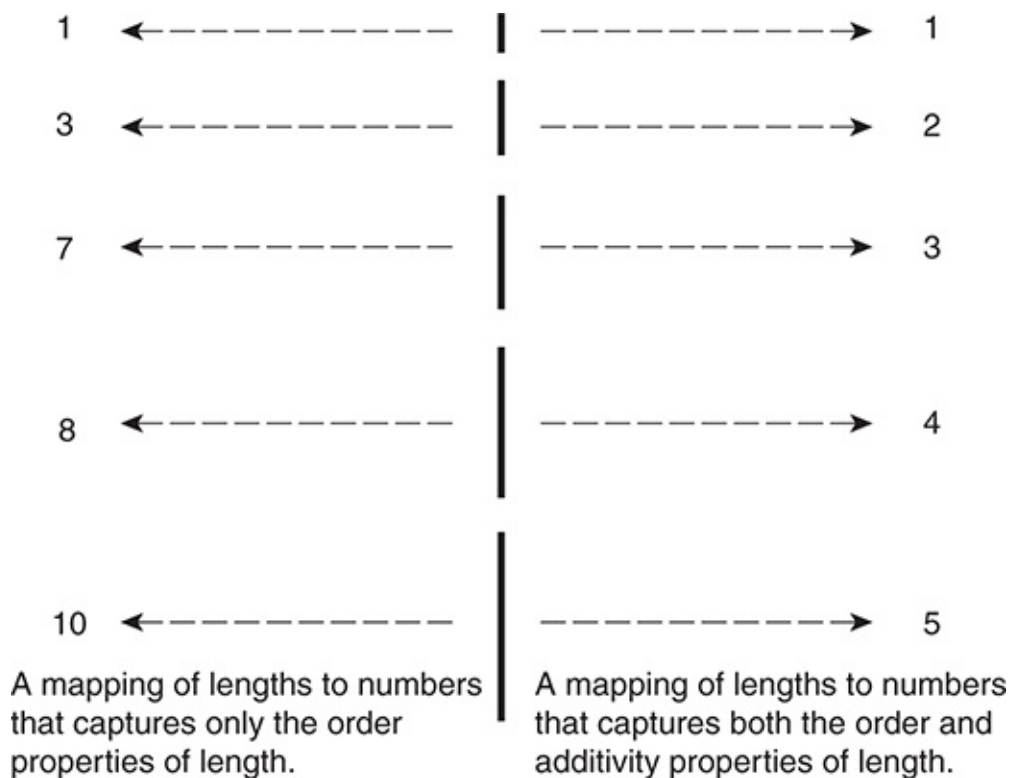


Figure 2.1.

The measurement of the length of line segments on two different scales of measurement.

Knowing the type of an attribute is important because it tells us which properties of the measured values are consistent with the underlying

properties of the attribute, and therefore, it allows us to avoid foolish actions, such as computing the average employee ID.

The Different Types of Attributes

A useful (and simple) way to specify the type of an attribute is to identify the properties of numbers that correspond to underlying properties of the attribute. For example, an attribute such as length has many of the properties of numbers. It makes sense to compare and order objects by length, as well as to talk about the differences and ratios of length. The following properties (operations) of numbers are typically used to describe attributes.

1. **Distinctness** = and \neq
2. **Order** $<$, \leq , $>$, and \geq
3. **Addition** + and -
4. **Multiplication** \times and /

Given these properties, we can define four types of attributes: **nominal** , **ordinal**, **interval** , and **ratio**. [Table 2.2](#) gives the definitions of these types, along with information about the statistical operations that are valid for each type. Each attribute type possesses all of the properties and operations of the attribute types above it. Consequently, any property or operation that is valid for nominal, ordinal, and interval attributes is also valid for ratio attributes. In other words, the definition of the attribute types is cumulative. However, this does not mean that the statistical operations appropriate for one attribute type are appropriate for the attribute types above it.

Table 2.2. Different attribute types.

Attribute Type		Description	Examples	Operations
Categorical	Nominal	The values of a nominal attribute	zip codes,	mode,

(Qualitative)		are just different names; i.e., nominal values provide only enough information to distinguish one object from another. ($=$, \neq)	employee ID numbers, eye color, gender	entropy, contingency correlation, χ^2 test
	Ordinal	The values of an ordinal attribute provide enough information to order objects. ($<$, $>$)	hardness of minerals, <i>{good, better, best}</i> , grades, street numbers	median, percentiles, rank correlation, run tests, sign tests
Numeric (Quantitative)	Interval	For interval attributes, the differences between values are meaningful, i.e., a unit of measurement exists. ($+$, $-$)	calendar dates, temperature in Celsius or Fahrenheit	mean, standard deviation, Pearson's correlation, t and F tests
	Ratio	For ratio variables, both differences and ratios are meaningful. (\times , $/$)	temperature in Kelvin, monetary quantities, counts, age, mass, length, electrical current	geometric mean, harmonic mean, percent variation

Nominal and ordinal attributes are collectively referred to as **categorical** or **qualitative** attributes. As the name suggests, qualitative attributes, such as employee ID, lack most of the properties of numbers. Even if they are represented by numbers, i.e., integers, they should be treated more like symbols. The remaining two types of attributes, interval and ratio, are collectively referred to as **quantitative** or **numeric** attributes. Quantitative attributes are represented by numbers and have most of the properties of

numbers. Note that quantitative attributes can be integer-valued or continuous.

The types of attributes can also be described in terms of transformations that do not change the meaning of an attribute. Indeed, S. Smith Stevens, the psychologist who originally defined the types of attributes shown in [Table 2.2](#), defined them in terms of these **permissible transformations**. For example, the meaning of a length attribute is unchanged if it is measured in meters instead of feet.

The statistical operations that make sense for a particular type of attribute are those that will yield the same results when the attribute is transformed by using a transformation that preserves the attribute’s meaning. To illustrate, the average length of a set of objects is different when measured in meters rather than in feet, but both averages represent the same length. [Table 2.3](#) shows the meaning-preserving transformations for the four attribute types of [Table 2.2](#).

Table 2.3. Transformations that define attribute levels.

Attribute Type		Transformation	Comment
Categorical (Qualitative)	Nominal	Any one-to-one mapping, e.g., a permutation of values	If all employee ID numbers are reassigned, it will not make any difference.
	Ordinal	An order-preserving change of values, i.e., $\text{new_value} = f(\text{old_value})$, where f is a monotonic function.	An attribute encompassing the notion of good, better, best can be represented equally well by the values {1, 2, 3} or by {0.5, 1, 10}.
Numeric (Quantitative)	Interval	$\text{new_value} = a \times \text{old_value} + b$, a and b constants.	The Fahrenheit and Celsius temperature scales differ in the

			location of their zero value and the size of a degree (unit).
	Ratio	$\text{new_value} = a \times \text{old_value}$	Length can be measured in meters or feet.

Example 2.5 (Temperature Scales).

Temperature provides a good illustration of some of the concepts that have been described. First, temperature can be either an interval or a ratio attribute, depending on its measurement scale. When measured on the Kelvin scale, a temperature of 2° is, in a physically meaningful way, twice that of a temperature of 1° . This is not true when temperature is measured on either the Celsius or Fahrenheit scales, because, physically, a temperature of 1° Fahrenheit (Celsius) is not much different than a temperature of 2° Fahrenheit (Celsius). The problem is that the zero points of the Fahrenheit and Celsius scales are, in a physical sense, arbitrary, and therefore, the ratio of two Celsius or Fahrenheit temperatures is not physically meaningful.

Describing Attributes by the Number of Values

An independent way of distinguishing between attributes is by the number of values they can take.

Discrete A discrete attribute has a finite or countably infinite set of values. Such attributes can be categorical, such as zip codes or ID numbers, or numeric, such as counts. Discrete attributes are often represented using integer variables. **Binary attributes** are a special case of discrete attributes and assume only two values, e.g., true/false, yes/no, male/female, or 0/1.

Binary attributes are often represented as Boolean variables, or as integer variables that only take the values 0 or 1.

Continuous A continuous attribute is one whose values are real numbers. Examples include attributes such as temperature, height, or weight. Continuous attributes are typically represented as floating-point variables. Practically, real values can be measured and represented only with limited precision.

In theory, any of the measurement scale types—nominal, ordinal, interval, and ratio—could be combined with any of the types based on the number of attribute values—binary, discrete, and continuous. However, some combinations occur only infrequently or do not make much sense. For instance, it is difficult to think of a realistic data set that contains a continuous binary attribute. Typically, nominal and ordinal attributes are binary or discrete, while interval and ratio attributes are continuous. However, **count attributes**, which are discrete, are also ratio attributes.

Asymmetric Attributes

For asymmetric attributes, only presence—a non-zero attribute value—is regarded as important. Consider a data set in which each object is a student and each attribute records whether a student took a particular course at a university. For a specific student, an attribute has a value of 1 if the student took the course associated with that attribute and a value of 0 otherwise. Because students take only a small fraction of all available courses, most of the values in such a data set would be 0. Therefore, it is more meaningful and more efficient to focus on the non-zero values. To illustrate, if students are compared on the basis of the courses they don't take, then most students would seem very similar, at least if the number of courses is large. Binary attributes where only non-zero values are important are called **asymmetric**

binary attributes. This type of attribute is particularly important for association analysis, which is discussed in [Chapter 5](#). It is also possible to have discrete or continuous asymmetric features. For instance, if the number of credits associated with each course is recorded, then the resulting data set will consist of **asymmetric discrete** or **continuous attributes**.

General Comments on Levels of Measurement

As described in the rest of this chapter, there are many diverse types of data. The previous discussion of measurement scales, while useful, is not complete and has some limitations. We provide the following comments and guidance.

- **Distinctness, order, and meaningful intervals and ratios are only four properties of data—many others are possible.** For instance, some data is inherently cyclical, e.g., position on the surface of the Earth or time. As another example, consider set valued attributes, where each attribute value is a set of elements, e.g., the set of movies seen in the last year. Define one set of elements (movies) to be greater (larger) than a second set if the second set is a subset of the first. However, such a relationship defines only a partial order that does not match any of the attribute types just defined.
- **The numbers or symbols used to capture attribute values may not capture all the properties of the attributes or may suggest properties that are not there.** An illustration of this for integers was presented in [Example 2.3](#), i.e., averages of IDs and out of range ages.
- **Data is often transformed for the purpose of analysis—see [Section 2.3.7](#).** This often changes the distribution of the observed variable to a distribution that is easier to analyze, e.g., a Gaussian (normal) distribution. Often, such transformations only preserve the order of the original values, and other properties are lost. Nonetheless, if the desired outcome is a

statistical test of differences or a predictive model, such a transformation is justified.

- **The final evaluation of any data analysis, including operations on attributes, is whether the results make sense from a domain point of view.**

In summary, it can be challenging to determine which operations can be performed on a particular attribute or a collection of attributes without compromising the integrity of the analysis. Fortunately, established practice often serves as a reliable guide. Occasionally, however, standard practices are erroneous or have limitations.

2.1.2 Types of Data Sets

There are many types of data sets, and as the field of data mining develops and matures, a greater variety of data sets become available for analysis. In this section, we describe some of the most common types. For convenience, we have grouped the types of data sets into three groups: record data, graph-based data, and ordered data. These categories do not cover all possibilities and other groupings are certainly possible.

General Characteristics of Data Sets

Before providing details of specific kinds of data sets, we discuss three characteristics that apply to many data sets and have a significant impact on the data mining techniques that are used: dimensionality, distribution, and resolution.

Dimensionality

The dimensionality of a data set is the number of attributes that the objects in the data set possess. Analyzing data with a small number of dimensions tends to be qualitatively different from analyzing moderate or high-dimensional data. Indeed, the difficulties associated with the analysis of high-dimensional data are sometimes referred to as the **curse of dimensionality**. Because of this, an important motivation in preprocessing the data is **dimensionality reduction**. These issues are discussed in more depth later in this chapter and in Appendix B.

Distribution

The distribution of a data set is the frequency of occurrence of various values or sets of values for the attributes comprising data objects. Equivalently, the distribution of a data set can be considered as a description of the concentration of objects in various regions of the data space. Statisticians have enumerated many types of distributions, e.g., Gaussian (normal), and described their properties. (See Appendix C.) Although statistical approaches for describing distributions can yield powerful analysis techniques, many data sets have distributions that are not well captured by standard statistical distributions.

As a result, many data mining algorithms do not assume a particular statistical distribution for the data they analyze. However, some general aspects of distributions often have a strong impact. For example, suppose a categorical attribute is used as a class variable, where one of the categories occurs 95% of the time, while the other categories together occur only 5% of the time. This **skewness** in the distribution can make classification difficult as discussed in Section 4.11. (Skewness has other impacts on data analysis that are not discussed here.)

A special case of skewed data is **sparsity**. For sparse binary, count or continuous data, most attributes of an object have values of 0. In many cases, fewer than 1% of the values are non-zero. In practical terms, sparsity is an advantage because usually only the non-zero values need to be stored and manipulated. This results in significant savings with respect to computation time and storage. Indeed, some data mining algorithms, such as the association rule mining algorithms described in [Chapter 5](#), work well only for sparse data. Finally, note that often the attributes in sparse data sets are asymmetric attributes.

Resolution

It is frequently possible to obtain data at different levels of resolution, and often the properties of the data are different at different resolutions. For instance, the surface of the Earth seems very uneven at a resolution of a few meters, but is relatively smooth at a resolution of tens of kilometers. The patterns in the data also depend on the level of resolution. If the resolution is too fine, a pattern may not be visible or may be buried in noise; if the resolution is too coarse, the pattern can disappear. For example, variations in atmospheric pressure on a scale of hours reflect the movement of storms and other weather systems. On a scale of months, such phenomena are not detectable.

Record Data

Much data mining work assumes that the data set is a collection of records (data objects), each of which consists of a fixed set of data fields (attributes). See [Figure 2.2\(a\)](#). For the most basic form of record data, there is no explicit relationship among records or data fields, and every record (object) has the same set of attributes. Record data is usually stored either in **flat** files or in relational databases. Relational databases are certainly more than a

collection of records, but data mining often does not use any of the additional information available in a relational database. Rather, the database serves as a convenient place to find records. Different types of record data are described below and are illustrated in [Figure 2.2](#).

<i>Tid</i>	<i>Refund</i>	<i>Marital Status</i>	<i>Taxable Income</i>	<i>Defaulted Borrower</i>
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

(a) Record data.

<i>TID</i>	<i>ITEMS</i>
1	Bread, Soda, Milk
2	Beer, Bread
3	Beer, Soda, Diapers, Milk
4	Beer, Bread, Diapers, Milk
5	Soda, Diapers, Milk

(b) Transaction data.

<i>Projection of x Load</i>	<i>Projection of y Load</i>	<i>Distance</i>	<i>Load</i>	<i>Thickness</i>
10.23	5.27	15.22	27	1.2
12.65	6.25	16.22	22	1.1
13.54	7.23	17.34	23	1.2
14.27	8.43	18.45	25	0.9

(c) Data matrix.


	<i>team</i>	<i>coach</i>	<i>play</i>	<i>ball</i>	<i>score</i>	<i>game</i>	<i>win</i>	<i>lost</i>	<i>timeout</i>	<i>season</i>
Document 1	3	0	5	0	2	6	0	2	0	2
Document 2	0	7	0	2	1	0	0	3	0	0
Document 3	0	1	0	0	1	2	2	0	3	0

(d) Document-term matrix.


Figure 2.2.

Different variations of record data.


Transaction or Market Basket Data

Transaction data is a special type of record data, where each record (transaction) involves a set of items. Consider a grocery store. The set of products purchased by a customer during one shopping trip constitutes a transaction, while the individual products that were purchased are the items. This type of data is called **market basket data** because the items in each record are the products in a person's "market basket." Transaction data is a collection of sets of items, but it can be viewed as a set of records whose fields are asymmetric attributes. Most often, the attributes are binary, indicating whether an item was purchased, but more generally, the attributes can be discrete or continuous, such as the number of items purchased or the amount spent on those items. [Figure 2.2\(b\)](#)  shows a sample transaction data set. Each row represents the purchases of a particular customer at a particular time.

The Data Matrix

If all the data objects in a collection of data have the same fixed set of numeric attributes, then the data objects can be thought of as points (vectors) in a multidimensional space, where each dimension represents a distinct attribute describing the object. A set of such data objects can be interpreted as an m by n matrix, where there are m rows, one for each object, and n columns, one for each attribute. (A representation that has data objects as columns and attributes as rows is also fine.) This matrix is called a **data matrix** or a **pattern matrix**. A data matrix is a variation of record data, but because it consists of numeric attributes, standard matrix operation can be applied to transform and manipulate the data. Therefore, the data matrix is the standard data format for most statistical data. [Figure 2.2\(c\)](#)  shows a sample data matrix.


The Sparse Data Matrix

A sparse data matrix is a special case of a data matrix where the attributes are of the same type and are asymmetric; i.e., only non-zero values are important. Transaction data is an example of a sparse data matrix that has only 0–1 entries. Another common example is document data. In particular, if the order of the terms (words) in a document is ignored—the “bag of words” approach—then a document can be represented as a term vector, where each term is a component (attribute) of the vector and the value of each component is the number of times the corresponding term occurs in the document. This representation of a collection of documents is often called a **document-term matrix**. [Figure 2.2\(d\)](#)  shows a sample document-term matrix. The documents are the rows of this matrix, while the terms are the columns. In practice, only the non-zero entries of sparse data matrices are stored.

Graph-Based Data


A graph can sometimes be a convenient and powerful representation for data. We consider two specific cases: (1) the graph captures relationships among data objects and (2) the data objects themselves are represented as graphs.

Data with Relationships among Objects

The relationships among objects frequently convey important information. In such cases, the data is often represented as a graph. In particular, the data objects are mapped to nodes of the graph, while the relationships among objects are captured by the links between objects and link properties, such as direction and weight. Consider web pages on the World Wide Web, which contain both text and links to other pages. In order to process search queries, web search engines collect and process web pages to extract their contents. It is well-known, however, that the links to and from each page provide a great deal of information about the relevance of a web page to a query, and thus, must also be taken into consideration. [Figure 2.3\(a\)](#)  shows a set of linked

web pages. Another important example of such graph data are the social networks, where data objects are people and the relationships among them are their interactions via social media.

Data with Objects That Are Graphs

If objects have structure, that is, the objects contain subobjects that have relationships, then such objects are frequently represented as graphs. For example, the structure of chemical compounds can be represented by a graph, where the nodes are atoms and the links between nodes are chemical bonds. **Figure 2.3(b)**  shows a ball-and-stick diagram of the chemical compound benzene, which contains atoms of carbon (black) and hydrogen (gray). A graph representation makes it possible to determine which substructures occur frequently in a set of compounds and to ascertain whether the presence of any of these substructures is associated with the presence or absence of certain chemical properties, such as melting point or heat of formation. Frequent graph mining, which is a branch of data mining that analyzes such data, is considered in Section 6.5.

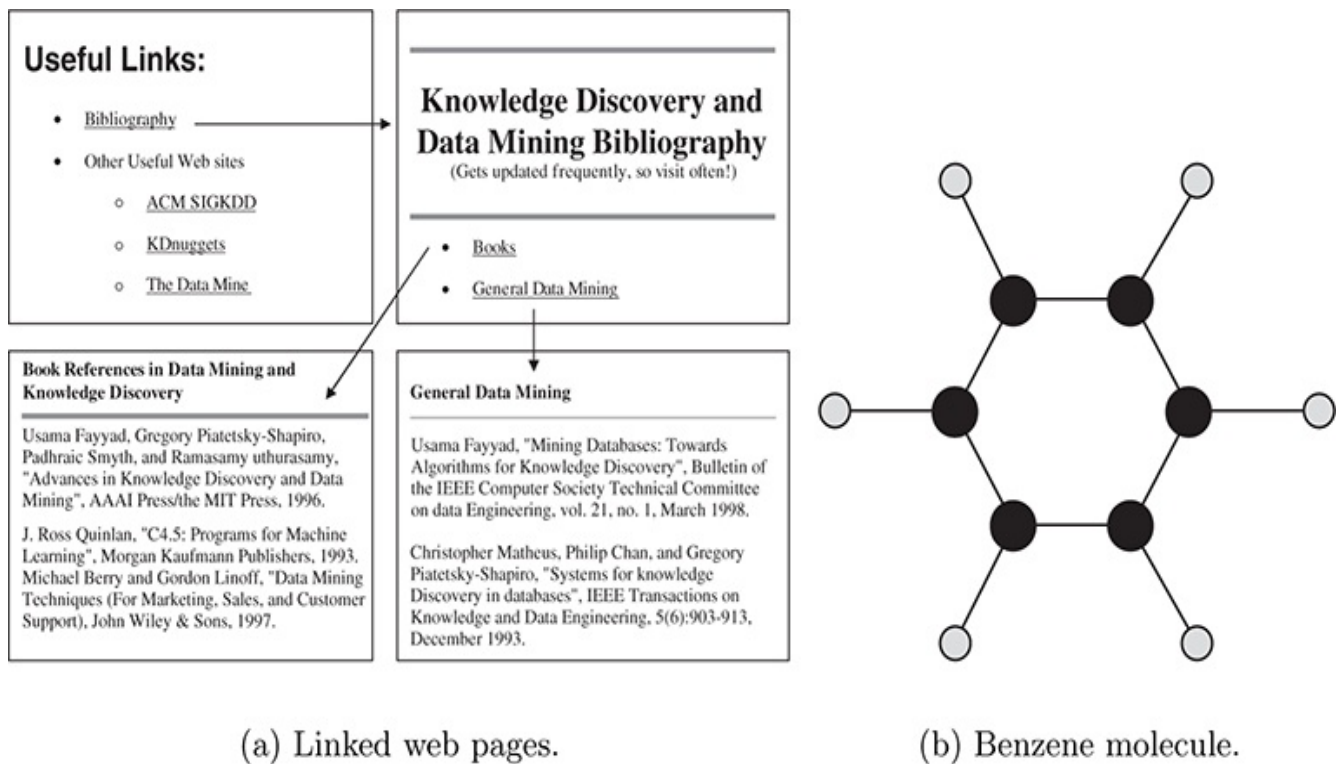


Figure 2.3.

Different variations of graph data.


Ordered Data

For some types of data, the attributes have relationships that involve order in time or space. Different types of ordered data are described next and are shown in [Figure 2.4](#).


Sequential Transaction Data

Sequential transaction data can be thought of as an extension of transaction data, where each transaction has a time associated with it. Consider a retail transaction data set that also stores the time at which the transaction took place. This time information makes it possible to find patterns such as “candy sales peak before Halloween.” A time can also be associated with each attribute. For example, each record could be the purchase history of a

customer, with a listing of items purchased at different times. Using this information, it is possible to find patterns such as “people who buy DVD players tend to buy DVDs in the period immediately following the purchase.”

Figure 2.4(a)  shows an example of sequential transaction data. There are five different times— t_1 , t_2 , t_3 , t_4 , and t_5 ; three different customers—C1, C2, and C3; and five different items—A, B, C, D, and E. In the top table, each row corresponds to the items purchased at a particular time by each customer. For instance, at time t_3 , customer C2 purchased items A and D. In the bottom table, the same information is displayed, but each row corresponds to a particular customer. Each row contains information about each transaction involving the customer, where a transaction is considered to be a set of items and the time at which those items were purchased. For example, customer C3 bought items A and C at time t_2 .

Time Series Data

Time series data is a special type of ordered data where each record is a **time series**, i.e., a series of measurements taken over time. For example, a financial data set might contain objects that are time series of the daily prices of various stocks. As another example, consider **Figure 2.4(c)** , which shows a time series of the average monthly temperature for Minneapolis during the years 1982 to 1994. When working with temporal data, such as time series, it is important to consider **temporal autocorrelation**; i.e., if two measurements are close in time, then the values of those measurements are often very similar.

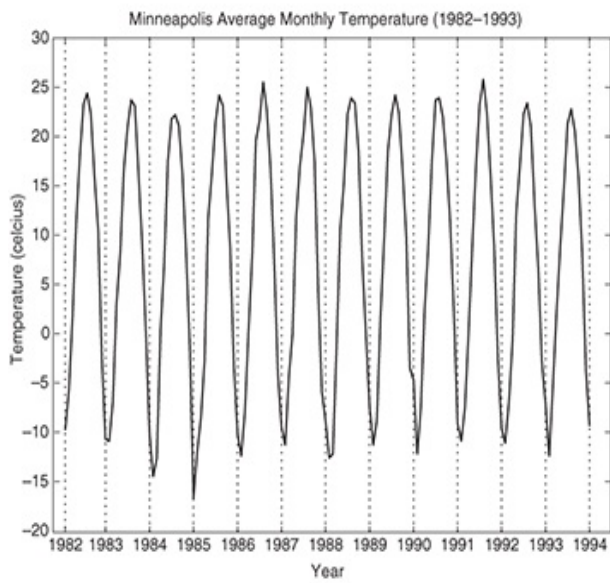
Time	Customer	Items Purchased
t1	C1	A, B
t2	C3	A, C
t2	C1	C, D
t3	C2	A, D
t4	C2	E
t5	C1	A, E

Customer	Time and Items Purchased
C1	(t1: A,B) (t2:C,D) (t5:A,E)
C2	(t3: A, D) (t4: E)
C3	(t2: A, C)

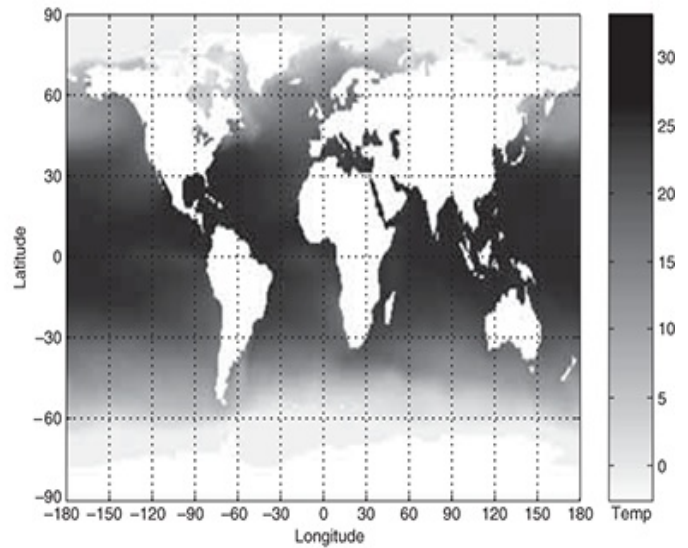
(a) Sequential transaction data.

```
GGTTCCGCCTTCAGCCCCGCGCC
CGCAGGGCCCGCCCCGCGCCGTC
GAGAAGGGCCCGCCTGGCGGGCG
GGGGGAGGCGGGGCGCCCGAGC
CCAACCGAGTCCGACCAGGTGCC
CCCTCTGCTCGGCCTAGACCTGA
GCTCATTAGGCGGCAGCGGACAG
GCCAAGTAGAACACGCGAAGCGC
TGGGCTGCCTGCTGCGACCAGGG
```

(b) Genomic sequence data.



(c) Temperature time series.




(d) Spatial temperature data.

Figure 2.4.

Different variations of ordered data.

Sequence Data

Sequence data consists of a data set that is a sequence of individual entities, such as a sequence of words or letters. It is quite similar to sequential data, except that there are no time stamps; instead, there are positions in an ordered sequence. For example, the genetic information of plants and animals can be represented in the form of sequences of nucleotides that are known as genes. Many of the problems associated with genetic sequence data involve predicting similarities in the structure and function of genes from similarities in nucleotide sequences. **Figure 2.4(b)**  shows a section of the human genetic code expressed using the four nucleotides from which all DNA is constructed: A, T, G, and C.

Spatial and Spatio-Temporal Data

Some objects have spatial attributes, such as positions or areas, in addition to other types of attributes. An example of spatial data is weather data (precipitation, temperature, pressure) that is collected for a variety of geographical locations. Often such measurements are collected over time, and thus, the data consists of time series at various locations. In that case, we refer to the data as spatio-temporal data. Although analysis can be conducted separately for each specific time or location, a more complete analysis of spatio-temporal data requires consideration of both the spatial and temporal aspects of the data.

An important aspect of spatial data is **spatial autocorrelation**; i.e., objects that are physically close tend to be similar in other ways as well. Thus, two points on the Earth that are close to each other usually have similar values for temperature and rainfall. Note that spatial autocorrelation is analogous to temporal autocorrelation.

Important examples of spatial and spatio-temporal data are the science and engineering data sets that are the result of measurements or model output

taken at regularly or irregularly distributed points on a two- or three-dimensional grid or mesh. For instance, Earth science data sets record the temperature or pressure measured at points (grid cells) on latitude–longitude spherical grids of various resolutions, e.g., 1° by 1°. See [Figure 2.4\(d\)](#). As another example, in the simulation of the flow of a gas, the speed and direction of flow at various instants in time can be recorded for each grid point in the simulation. A different type of spatio-temporal data arises from tracking the trajectories of objects, e.g., vehicles, in time and space.

Handling Non-Record Data

Most data mining algorithms are designed for record data or its variations, such as transaction data and data matrices. Record-oriented techniques can be applied to non-record data by extracting features from data objects and using these features to create a record corresponding to each object. Consider the chemical structure data that was described earlier. Given a set of common substructures, each compound can be represented as a record with binary attributes that indicate whether a compound contains a specific substructure. Such a representation is actually a transaction data set, where the transactions are the compounds and the items are the substructures.

In some cases, it is easy to represent the data in a record format, but this type of representation does not capture all the information in the data. Consider spatio-temporal data consisting of a time series from each point on a spatial grid. This data is often stored in a data matrix, where each row represents a location and each column represents a particular point in time. However, such a representation does not explicitly capture the time relationships that are present among attributes and the spatial relationships that exist among objects. This does not mean that such a representation is inappropriate, but rather that these relationships must be taken into consideration during the analysis. For example, it would not be a good idea to use a data mining

technique that ignores the temporal autocorrelation of the attributes or the spatial autocorrelation of the data objects, i.e., the locations on the spatial grid.

2.2 Data Quality

Data mining algorithms are often applied to data that was collected for another purpose, or for future, but unspecified applications. For that reason, data mining cannot usually take advantage of the significant benefits of “addressing quality issues at the source.” In contrast, much of statistics deals with the design of experiments or surveys that achieve a prespecified level of data quality. Because preventing data quality problems is typically not an option, data mining focuses on (1) the detection and correction of data quality problems and (2) the use of algorithms that can tolerate poor data quality. The first step, detection and correction, is often called **data cleaning**.

The following sections discuss specific aspects of data quality. The focus is on measurement and data collection issues, although some application-related issues are also discussed.

2.2.1 Measurement and Data Collection Issues

It is unrealistic to expect that data will be perfect. There may be problems due to human error, limitations of measuring devices, or flaws in the data collection process. Values or even entire data objects can be missing. In other cases, there can be spurious or duplicate objects; i.e., multiple data objects that all correspond to a single “real” object. For example, there might be two different records for a person who has recently lived at two different addresses. Even if

all the data is present and “looks fine,” there may be inconsistencies—a person has a height of 2 meters, but weighs only 2 kilograms.

In the next few sections, we focus on aspects of data quality that are related to data measurement and collection. We begin with a definition of measurement and data collection errors and then consider a variety of problems that involve measurement error: noise, artifacts, bias, precision, and accuracy. We conclude by discussing data quality issues that involve both measurement and data collection problems: outliers, missing and inconsistent values, and duplicate data.

Measurement and Data Collection Errors

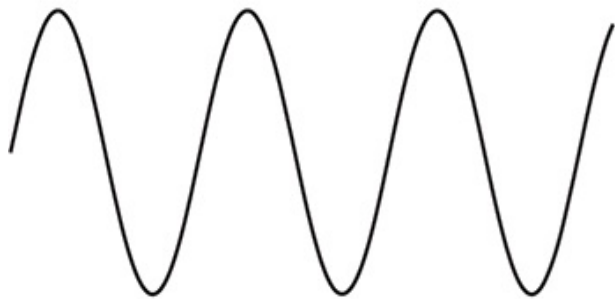
The term **measurement error** refers to any problem resulting from the measurement process. A common problem is that the value recorded differs from the true value to some extent. For continuous attributes, the numerical difference of the measured and true value is called the **error**. The term **data collection error** refers to errors such as omitting data objects or attribute values, or inappropriately including a data object. For example, a study of animals of a certain species might include animals of a related species that are similar in appearance to the species of interest. Both measurement errors and data collection errors can be either systematic or random.

We will only consider general types of errors. Within particular domains, certain types of data errors are commonplace, and well-developed techniques often exist for detecting and/or correcting these errors. For example, keyboard errors are common when data is entered manually, and as a result, many data entry programs have techniques for detecting and, with human intervention, correcting such errors.

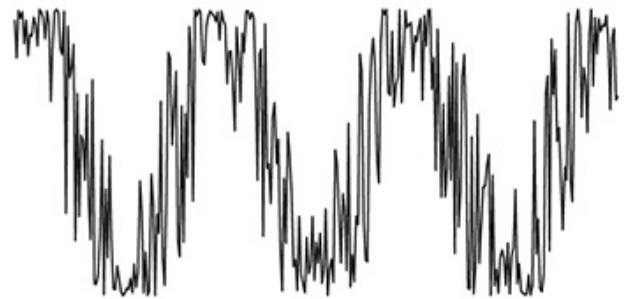
Noise and Artifacts

Noise is the random component of a measurement error. It typically involves the distortion of a value or the addition of spurious objects. **Figure 2.5** shows a time series before and after it has been disrupted by random noise. If a bit more noise were added to the time series, its shape would be lost.

Figure 2.6 shows a set of data points before and after some noise points (indicated by '+'s) have been added. Notice that some of the noise points are intermixed with the non-noise points.



(a) Time series.



(b) Time series with noise.

Figure 2.5.

Noise in a time series context.

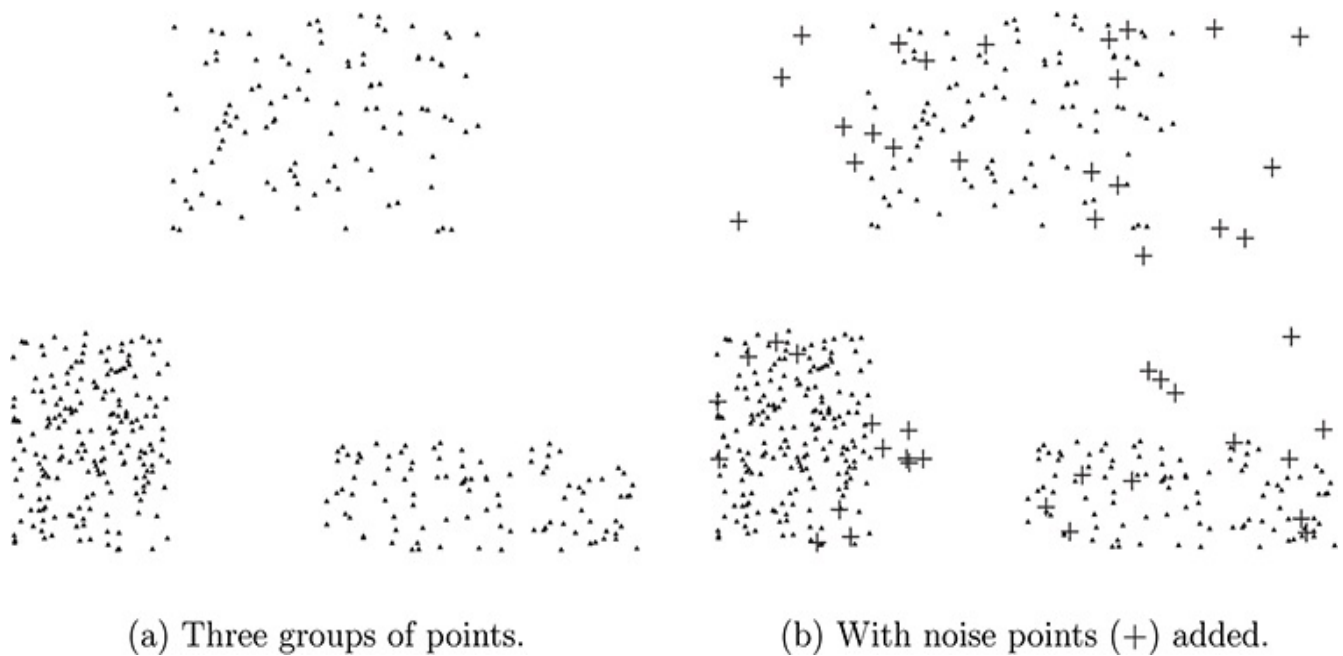


Figure 2.6.

Noise in a spatial context.

The term noise is often used in connection with data that has a spatial or temporal component. In such cases, techniques from signal or image processing can frequently be used to reduce noise and thus, help to discover patterns (signals) that might be “lost in the noise.” Nonetheless, the elimination of noise is frequently difficult, and much work in data mining focuses on devising **robust algorithms** that produce acceptable results even when noise is present.

Data errors can be the result of a more deterministic phenomenon, such as a streak in the same place on a set of photographs. Such deterministic distortions of the data are often referred to as **artifacts**.

Precision, Bias, and Accuracy

In statistics and experimental science, the quality of the measurement process and the resulting data are measured by precision and bias. We provide the

standard definitions, followed by a brief discussion. For the following definitions, we assume that we make repeated measurements of the same underlying quantity.

Definition 2.3 (Precision).

The closeness of repeated measurements (of the same quantity) to one another.

Definition 2.4 (Bias).

A systematic variation of measurements from the quantity being measured.

Precision is often measured by the standard deviation of a set of values, while bias is measured by taking the difference between the mean of the set of values and the known value of the quantity being measured. Bias can be determined only for objects whose measured quantity is known by means external to the current situation. Suppose that we have a standard laboratory weight with a mass of 1g and want to assess the precision and bias of our new laboratory scale. We weigh the mass five times, and obtain the following five values: { 1.015, 0.990, 1.013, 1.001, 0.986}. The mean of these values is

1.001, and hence, the bias is 0.001. The precision, as measured by the standard deviation, is 0.013.

It is common to use the more general term, **accuracy**, to refer to the degree of measurement error in data.

Definition 2.5 (Accuracy)

The closeness of measurements to the true value of the quantity being measured.


Accuracy depends on precision and bias, but there is no specific formula for accuracy in terms of these two quantities.

One important aspect of accuracy is the use of **significant digits**. The goal is to use only as many digits to represent the result of a measurement or calculation as are justified by the precision of the data. For example, if the length of an object is measured with a meter stick whose smallest markings are millimeters, then we should record the length of data only to the nearest millimeter. The precision of such a measurement would be $\pm 0.5\text{mm}$. We do not review the details of working with significant digits because most readers will have encountered them in previous courses and they are covered in considerable depth in science, engineering, and statistics textbooks.

Issues such as significant digits, precision, bias, and accuracy are sometimes overlooked, but they are important for data mining as well as statistics and science. Many times, data sets do not come with information about the

precision of the data, and furthermore, the programs used for analysis return results without any such information. Nonetheless, without some understanding of the accuracy of the data and the results, an analyst runs the risk of committing serious data analysis blunders.

Outliers

Outliers are either (1) data objects that, in some sense, have characteristics that are different from most of the other data objects in the data set, or (2) values of an attribute that are unusual with respect to the typical values for that attribute. Alternatively, they can be referred to as **anomalous** objects or values. There is considerable leeway in the definition of an outlier, and many different definitions have been proposed by the statistics and data mining communities. Furthermore, it is important to distinguish between the notions of noise and outliers. Unlike noise, outliers can be legitimate data objects or values that we are interested in detecting. For instance, in fraud and network intrusion detection, the goal is to find unusual objects or events from among a large number of normal ones. [Chapter 9](#)  discusses anomaly detection in more detail.

Missing Values

It is not unusual for an object to be missing one or more attribute values. In some cases, the information was not collected; e.g., some people decline to give their age or weight. In other cases, some attributes are not applicable to all objects; e.g., often, forms have conditional parts that are filled out only when a person answers a previous question in a certain way, but for simplicity, all fields are stored. Regardless, missing values should be taken into account during the data analysis.

There are several strategies (and variations on these strategies) for dealing with missing data, each of which is appropriate in certain circumstances. These strategies are listed next, along with an indication of their advantages and disadvantages.

Eliminate Data Objects or Attributes

A simple and effective strategy is to eliminate objects with missing values. However, even a partially specified data object contains some information, and if many objects have missing values, then a reliable analysis can be difficult or impossible. Nonetheless, if a data set has only a few objects that have missing values, then it may be expedient to omit them. A related strategy is to eliminate attributes that have missing values. This should be done with caution, however, because the eliminated attributes may be the ones that are critical to the analysis.

Estimate Missing Values

Sometimes missing data can be reliably estimated. For example, consider a time series that changes in a reasonably smooth fashion, but has a few, widely scattered missing values. In such cases, the missing values can be estimated (interpolated) by using the remaining values. As another example, consider a data set that has many similar data points. In this situation, the attribute values of the points closest to the point with the missing value are often used to estimate the missing value. If the attribute is continuous, then the average attribute value of the nearest neighbors is used; if the attribute is categorical, then the most commonly occurring attribute value can be taken. For a concrete illustration, consider precipitation measurements that are recorded by ground stations. For areas not containing a ground station, the precipitation can be estimated using values observed at nearby ground stations.

Ignore the Missing Value during Analysis

Many data mining approaches can be modified to ignore missing values. For example, suppose that objects are being clustered and the similarity between pairs of data objects needs to be calculated. If one or both objects of a pair have missing values for some attributes, then the similarity can be calculated by using only the attributes that do not have missing values. It is true that the similarity will only be approximate, but unless the total number of attributes is small or the number of missing values is high, this degree of inaccuracy may not matter much. Likewise, many classification schemes can be modified to work with missing values.

Inconsistent Values


Data can contain inconsistent values. Consider an address field, where both a zip code and city are listed, but the specified zip code area is not contained in that city. It is possible that the individual entering this information transposed two digits, or perhaps a digit was misread when the information was scanned from a handwritten form. Regardless of the cause of the inconsistent values, it is important to detect and, if possible, correct such problems.

Some types of inconsistencies are easy to detect. For instance, a person's height should not be negative. In other cases, it can be necessary to consult an external source of information. For example, when an insurance company processes claims for reimbursement, it checks the names and addresses on the reimbursement forms against a database of its customers.

Once an inconsistency has been detected, it is sometimes possible to correct the data. A product code may have "check" digits, or it may be possible to double-check a product code against a list of known product codes, and then

correct the code if it is incorrect, but close to a known code. The correction of an inconsistency requires additional or redundant information.

Example 2.6 (Inconsistent Sea Surface Temperature).

This example illustrates an inconsistency in actual time series data that measures the sea surface temperature (SST) at various points on the ocean. SST data was originally collected using ocean-based measurements from ships or buoys, but more recently, satellites have been used to gather the data. To create a long-term data set, both sources of data must be used. However, because the data comes from different sources, the two parts of the data are subtly different. This discrepancy is visually displayed in [Figure 2.7](#) , which shows the correlation of SST values between pairs of years. If a pair of years has a positive correlation, then the location corresponding to the pair of years is colored white; otherwise it is colored black. (Seasonal variations were removed from the data since, otherwise, all the years would be highly correlated.) There is a distinct change in behavior where the data has been put together in 1983. Years within each of the two groups, 1958–1982 and 1983–1999, tend to have a positive correlation with one another, but a negative correlation with years in the other group. This does not mean that this data should not be used, only that the analyst should consider the potential impact of such discrepancies on the data mining analysis.

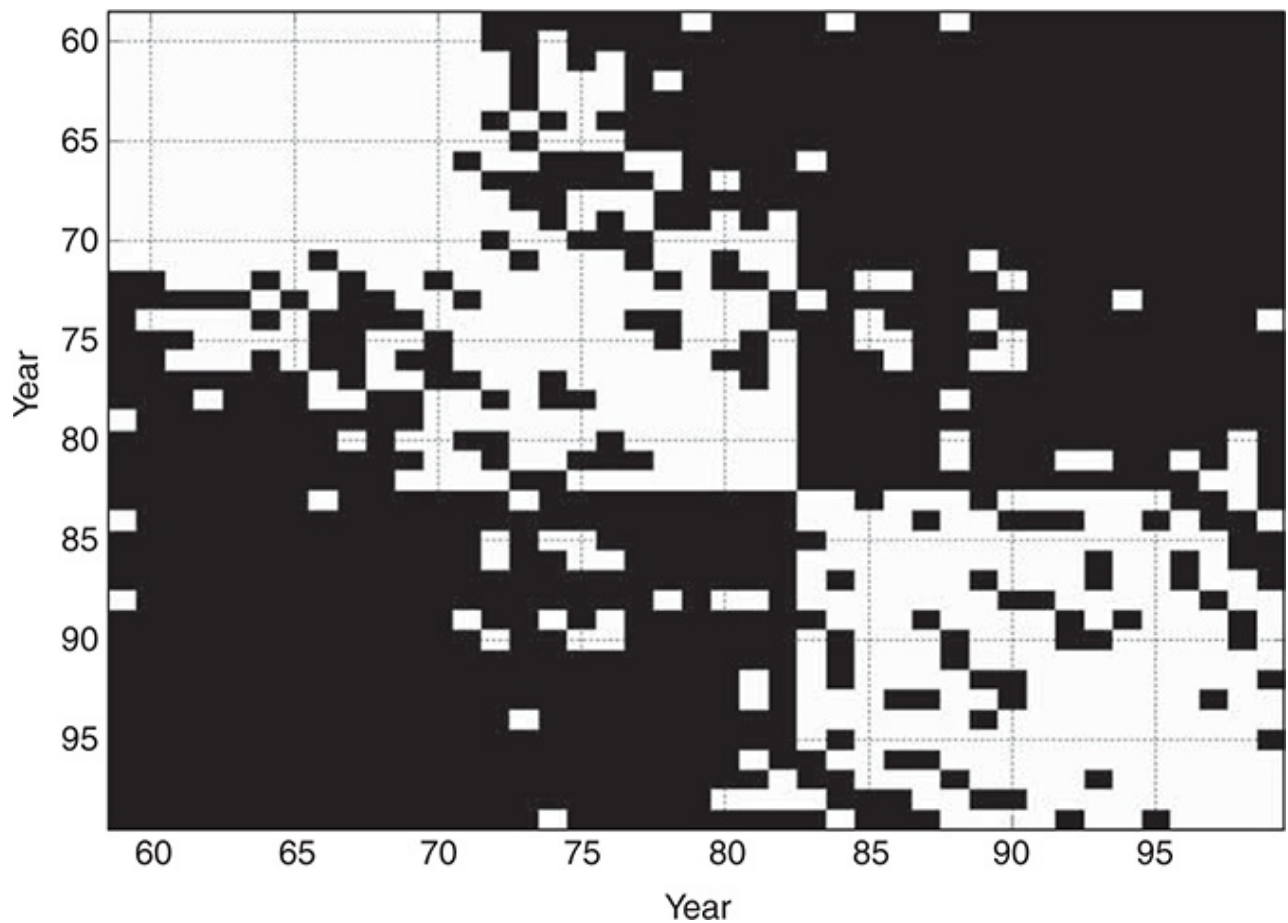


Figure 2.7.

Correlation of SST data between pairs of years. White areas indicate positive correlation. Black areas indicate negative correlation.

Duplicate Data

A data set can include data objects that are duplicates, or almost duplicates, of one another. Many people receive duplicate mailings because they appear in a database multiple times under slightly different names. To detect and eliminate such duplicates, two main issues must be addressed. First, if there are two objects that actually represent a single object, then one or more values of corresponding attributes are usually different, and these inconsistent values must be resolved. Second, care needs to be taken to avoid accidentally combining data objects that are similar, but not duplicates, such

as two distinct people with identical names. The term **deduplication** is often used to refer to the process of dealing with these issues.

In some cases, two or more objects are identical with respect to the attributes measured by the database, but they still represent different objects. Here, the duplicates are legitimate, but can still cause problems for some algorithms if the possibility of identical objects is not specifically accounted for in their design. An example of this is given in [Exercise 13](#) on page [108](#).

2.2.2 Issues Related to Applications

Data quality issues can also be considered from an application viewpoint as expressed by the statement “data is of high quality if it is suitable for its intended use.” This approach to data quality has proven quite useful, particularly in business and industry. A similar viewpoint is also present in statistics and the experimental sciences, with their emphasis on the careful design of experiments to collect the data relevant to a specific hypothesis. As with quality issues at the measurement and data collection level, many issues are specific to particular applications and fields. Again, we consider only a few of the general issues.

Timeliness

Some data starts to age as soon as it has been collected. In particular, if the data provides a snapshot of some ongoing phenomenon or process, such as the purchasing behavior of customers or web browsing patterns, then this snapshot represents reality for only a limited time. If the data is out of date, then so are the models and patterns that are based on it.

Relevance

The available data must contain the information necessary for the application. Consider the task of building a model that predicts the accident rate for drivers. If information about the age and gender of the driver is omitted, then it is likely that the model will have limited accuracy unless this information is indirectly available through other attributes.

Making sure that the objects in a data set are relevant is also challenging. A common problem is **sampling bias**, which occurs when a sample does not contain different types of objects in proportion to their actual occurrence in the population. For example, survey data describes only those who respond to the survey. (Other aspects of sampling are discussed further in [Section 2.3.2](#).) Because the results of a data analysis can reflect only the data that is present, sampling bias will typically lead to erroneous results when applied to the broader population.

Knowledge about the Data

Ideally, data sets are accompanied by documentation that describes different aspects of the data; the quality of this documentation can either aid or hinder the subsequent analysis. For example, if the documentation identifies several attributes as being strongly related, these attributes are likely to provide highly redundant information, and we usually decide to keep just one. (Consider sales tax and purchase price.) If the documentation is poor, however, and fails to tell us, for example, that the missing values for a particular field are indicated with a -9999, then our analysis of the data may be faulty. Other important characteristics are the precision of the data, the type of features (nominal, ordinal, interval, ratio), the scale of measurement (e.g., meters or feet for length), and the origin of the data.

2.3 Data Preprocessing

In this section, we consider which preprocessing steps should be applied to make the data more suitable for data mining. Data preprocessing is a broad area and consists of a number of different strategies and techniques that are interrelated in complex ways. We will present some of the most important ideas and approaches, and try to point out the interrelationships among them. Specifically, we will discuss the following topics:

- Aggregation
- Sampling
- Dimensionality reduction
- Feature subset selection
- Feature creation
- Discretization and binarization
- Variable transformation

Roughly speaking, these topics fall into two categories: selecting data objects and attributes for the analysis or for creating/changing the attributes. In both cases, the goal is to improve the data mining analysis with respect to time, cost, and quality. Details are provided in the following sections.

A quick note about terminology: In the following, we sometimes use synonyms for attribute, such as feature or variable, in order to follow common usage.

2.3.1 Aggregation

Sometimes “less is more,” and this is the case with **aggregation** , the combining of two or more objects into a single object. Consider a data set consisting of transactions (data objects) recording the daily sales of products in various store locations (Minneapolis, Chicago, Paris, ...) for different days over the course of a year. See [Table 2.4](#) . One way to aggregate transactions for this data set is to replace all the transactions of a single store with a single storewide transaction. This reduces the hundreds or thousands of transactions that occur daily at a specific store to a single daily transaction, and the number of data objects per day is reduced to the number of stores.

Table 2.4. Data set containing information about customer purchases.

Transaction ID	Item	Store Location	Date	Price	...
⋮	⋮	⋮	⋮	⋮	
101123	Watch	Chicago	09/06/04	\$25.99	...
101123	Battery	Chicago	09/06/04	\$5.99	...
101124	Shoes	Minneapolis	09/06/04	\$75.00	...



An obvious issue is how an aggregate transaction is created; i.e., how the values of each attribute are combined across all the records corresponding to a particular location to create the aggregate transaction that represents the sales of a single store or date. Quantitative attributes, such as *price*, are typically aggregated by taking a sum or an average. A qualitative attribute, such as *item*, can either be omitted or summarized in terms of a higher level category, e.g., televisions versus electronics.

The data in [Table 2.4](#) can also be viewed as a multidimensional array, where each attribute is a dimension. From this viewpoint, aggregation is the process of eliminating attributes, such as the type of item, or reducing the

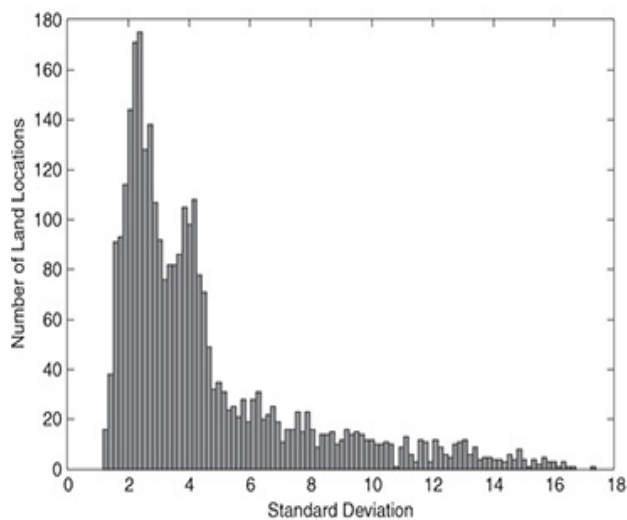
number of values for a particular attribute; e.g., reducing the possible values for *date* from 365 days to 12 months. This type of aggregation is commonly used in Online Analytical Processing (OLAP). References to OLAP are given in the bibliographic Notes.

There are several motivations for aggregation. First, the smaller data sets resulting from data reduction require less memory and processing time, and hence, aggregation often enables the use of more expensive data mining algorithms. Second, aggregation can act as a change of scope or scale by providing a high-level view of the data instead of a low-level view. In the previous example, aggregating over store locations and months gives us a monthly, per store view of the data instead of a daily, per item view. Finally, the behavior of groups of objects or attributes is often more stable than that of individual objects or attributes. This statement reflects the statistical fact that aggregate quantities, such as averages or totals, have less variability than the individual values being aggregated. For totals, the actual amount of variation is larger than that of individual objects (on average), but the percentage of the variation is smaller, while for means, the actual amount of variation is less than that of individual objects (on average). A disadvantage of aggregation is the potential loss of interesting details. In the store example, aggregating over months loses information about which day of the week has the highest sales.

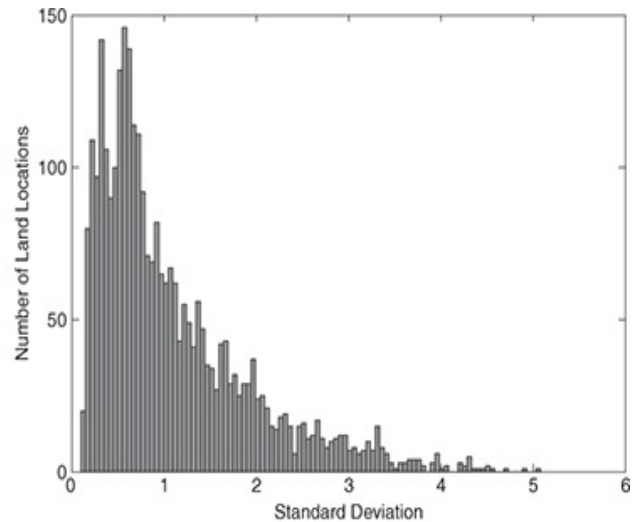
Example 2.7 (Australian Precipitation).

This example is based on precipitation in Australia from the period 1982–1993. **Figure 2.8(a)**  shows a histogram for the standard deviation of average monthly precipitation for 3,030 0.5° by 0.5° grid cells in Australia, while **Figure 2.8(b)**  shows a histogram for the standard deviation of the average yearly precipitation for the same locations. The average yearly precipitation has less variability than the average monthly precipitation. All

precipitation measurements (and their standard deviations) are in centimeters.



(a) Histogram of standard deviation of average monthly precipitation



(b) Histogram of standard deviation of average yearly precipitation

Figure 2.8.

Histograms of standard deviation for monthly and yearly precipitation in Australia for the period 1982–1993.

2.3.2 Sampling

Sampling is a commonly used approach for selecting a subset of the data objects to be analyzed. In statistics, it has long been used for both the preliminary investigation of the data and the final data analysis. Sampling can also be very useful in data mining. However, the motivations for sampling in statistics and data mining are often different. Statisticians use sampling because obtaining the entire set of data of interest is too expensive or time consuming, while data miners usually sample because it is too computationally expensive in terms of the memory or time required to process

all the data. In some cases, using a sampling algorithm can reduce the data size to the point where a better, but more computationally expensive algorithm can be used.




The key principle for effective sampling is the following: Using a sample will work almost as well as using the entire data set if the sample is representative. In turn, **a sample is representative** if it has approximately the same property (of interest) as the original set of data. If the mean (average) of the data objects is the property of interest, then a sample is representative if it has a mean that is close to that of the original data. Because sampling is a statistical process, the representativeness of any particular sample will vary, and the best that we can do is choose a sampling scheme that guarantees a high probability of getting a representative sample. As discussed next, this involves choosing the appropriate sample size and sampling technique.

Sampling Approaches

There are many sampling techniques, but only a few of the most basic ones and their variations will be covered here. The simplest type of sampling is **simple random sampling**. For this type of sampling, there is an equal probability of selecting any particular object. There are two variations on random sampling (and other sampling techniques as well): (1) **sampling without replacement** —as each object is selected, it is removed from the set of all objects that together constitute the **population** , and (2) **sampling with replacement** —objects are not removed from the population as they are selected for the sample. In sampling with replacement, the same object can be picked more than once. The samples produced by the two methods are not much different when samples are relatively small compared to the data set size, but sampling with replacement is simpler to analyze because the probability of selecting any object remains constant during the sampling process.

When the population consists of different types of objects, with widely different numbers of objects, simple random sampling can fail to adequately represent those types of objects that are less frequent. This can cause problems when the analysis requires proper representation of all object types. For example, when building classification models for rare classes, it is critical that the rare classes be adequately represented in the sample. Hence, a sampling scheme that can accommodate differing frequencies for the object types of interest is needed. **Stratified sampling**, which starts with prespecified groups of objects, is such an approach. In the simplest version, equal numbers of objects are drawn from each group even though the groups are of different sizes. In another variation, the number of objects drawn from each group is proportional to the size of that group.

Example 2.8 (Sampling and Loss of Information).

Once a sampling technique has been selected, it is still necessary to choose the sample size. Larger sample sizes increase the probability that a sample will be representative, but they also eliminate much of the advantage of sampling. Conversely, with smaller sample sizes, patterns can be missed or erroneous patterns can be detected. **Figure 2.9(a)**  shows a data set that contains 8000 two-dimensional points, while **Figures 2.9(b)**  and **2.9(c)**  show samples from this data set of size 2000 and 500, respectively. Although most of the structure of this data set is present in the sample of 2000 points, much of the structure is missing in the sample of 500 points.

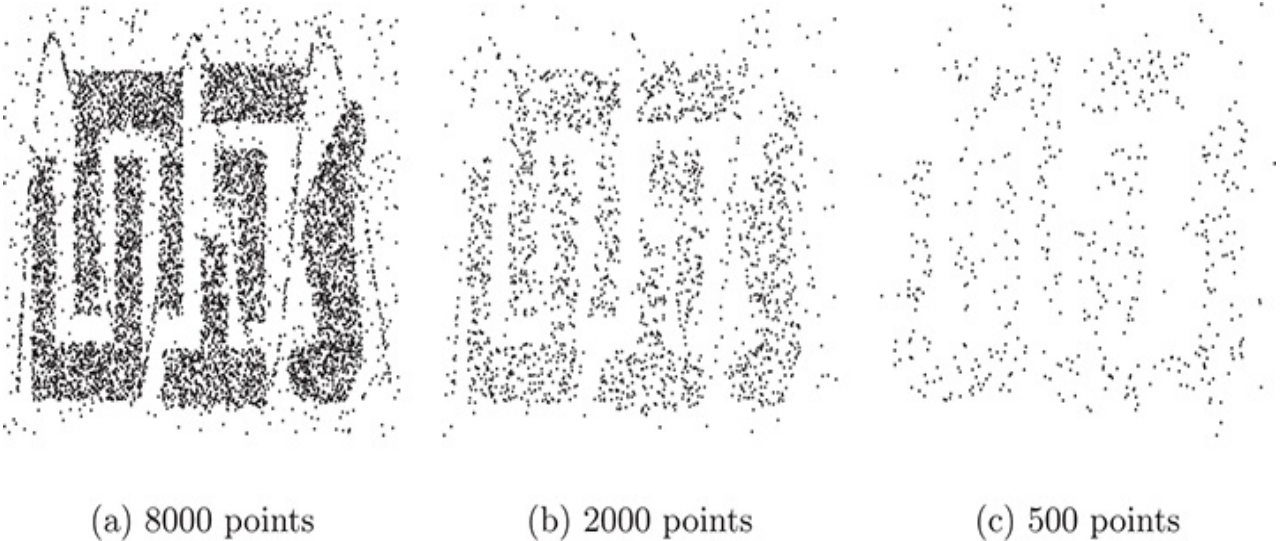



Figure 2.9.

Example of the loss of structure with sampling.

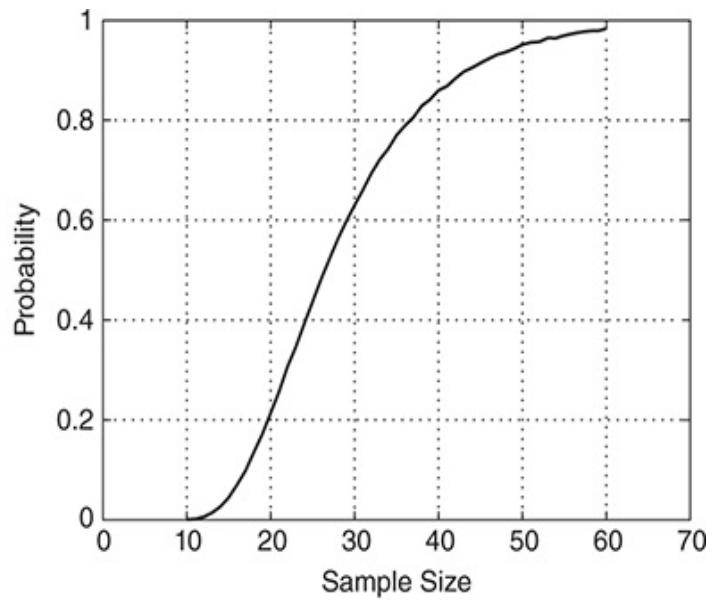
Example 2.9 (Determining the Proper Sample Size).

To illustrate that determining the proper sample size requires a methodical approach, consider the following task.

Given a set of data consisting of a small number of almost equal-sized groups, find at least one representative point for each of the groups. Assume that the objects in each group are highly similar to each other, but not very similar to objects in different groups. [Figure 2.10\(a\)](#)  shows an idealized set of clusters (groups) from which these points might be drawn.





(a) Ten groups of points.



(b) Probability a sample contains points from each of 10 groups.

Figure 2.10.

Finding representative points from 10 groups.

This problem can be efficiently solved using sampling. One approach is to take a small sample of data points, compute the pairwise similarities between points, and then form groups of points that are highly similar. The desired set of representative points is then obtained by taking one point from each of these groups. To follow this approach, however, we need to determine a sample size that would guarantee, with a high probability, the desired outcome; that is, that at least one point will be obtained from each cluster. **Figure 2.10(b)**  shows the probability of getting one object from each of the 10 groups as the sample size runs from 10 to 60. Interestingly, with a sample size of 20, there is little chance (20%) of getting a sample that includes all 10 clusters. Even with a sample size of 30, there is still a moderate chance (almost 40%) of getting a sample that doesn't contain objects from all 10 clusters. This issue is further explored in the context of clustering by **Exercise 4**  on page **603**.

Progressive Sampling


The proper sample size can be difficult to determine, so **adaptive** or **progressive sampling** schemes are sometimes used. These approaches start with a small sample, and then increase the sample size until a sample of sufficient size has been obtained. While this technique eliminates the need to determine the correct sample size initially, it requires that there be a way to evaluate the sample to judge if it is large enough.

Suppose, for instance, that progressive sampling is used to learn a predictive model. Although the accuracy of predictive models increases as the sample size increases, at some point the increase in accuracy levels off. We want to stop increasing the sample size at this leveling-off point. By keeping track of the change in accuracy of the model as we take progressively larger samples, and by taking other samples close to the size of the current one, we can get an estimate of how close we are to this leveling-off point, and thus, stop sampling.

2.3.3 Dimensionality Reduction

Data sets can have a large number of features. Consider a set of documents, where each document is represented by a vector whose components are the frequencies with which each word occurs in the document. In such cases, there are typically thousands or tens of thousands of attributes (components), one for each word in the vocabulary. As another example, consider a set of time series consisting of the daily closing price of various stocks over a period of 30 years. In this case, the attributes, which are the prices on specific days, again number in the thousands.

There are a variety of benefits to dimensionality reduction. A key benefit is that many data mining algorithms work better if the dimensionality—the number of attributes in the data—is lower. This is partly because dimensionality reduction can eliminate irrelevant features and reduce noise and partly because of the curse of dimensionality, which is explained below. Another benefit is that a reduction of dimensionality can lead to a more understandable model because the model usually involves fewer attributes. Also, dimensionality reduction may allow the data to be more easily visualized. Even if dimensionality reduction doesn't reduce the data to two or three dimensions, data is often visualized by looking at pairs or triplets of attributes, and the number of such combinations is greatly reduced. Finally, the amount of time and memory required by the data mining algorithm is reduced with a reduction in dimensionality.

The term dimensionality reduction is often reserved for those techniques that reduce the dimensionality of a data set by creating new attributes that are a combination of the old attributes. The reduction of dimensionality by selecting attributes that are a subset of the old is known as feature subset selection or feature selection. It will be discussed in [Section 2.3.4](#) .

In the remainder of this section, we briefly introduce two important topics: the curse of dimensionality and dimensionality reduction techniques based on linear algebra approaches such as principal components analysis (PCA). More details on dimensionality reduction can be found in Appendix B.

The Curse of Dimensionality

The curse of dimensionality refers to the phenomenon that many types of data analysis become significantly harder as the dimensionality of the data increases. Specifically, as dimensionality increases, the data becomes increasingly sparse in the space that it occupies. Thus, the data objects we

observe are quite possibly not a representative sample of all possible objects. For classification, this can mean that there are not enough data objects to allow the creation of a model that reliably assigns a class to all possible objects. For clustering, the differences in density and in the distances between points, which are critical for clustering, become less meaningful. (This is discussed further in Sections 8.1.2, 8.4.6, and 8.4.8.) As a result, many clustering and classification algorithms (and other data analysis algorithms) have trouble with high-dimensional data leading to reduced classification accuracy and poor quality clusters.

Linear Algebra Techniques for Dimensionality Reduction

Some of the most common approaches for dimensionality reduction, particularly for continuous data, use techniques from linear algebra to project the data from a high-dimensional space into a lower-dimensional space.

Principal Components Analysis (PCA) is a linear algebra technique for continuous attributes that finds new attributes (principal components) that (1) are linear combinations of the original attributes, (2) are **orthogonal** (perpendicular) to each other, and (3) capture the maximum amount of variation in the data. For example, the first two principal components capture as much of the variation in the data as is possible with two orthogonal attributes that are linear combinations of the original attributes. **Singular Value Decomposition (SVD)** is a linear algebra technique that is related to PCA and is also commonly used for dimensionality reduction. For additional details, see Appendices A and B.

2.3.4 Feature Subset Selection

Another way to reduce the dimensionality is to use only a subset of the features. While it might seem that such an approach would lose information, this is not the case if redundant and irrelevant features are present.

Redundant features duplicate much or all of the information contained in one or more other attributes. For example, the purchase price of a product and the amount of sales tax paid contain much of the same information. **Irrelevant features** contain almost no useful information for the data mining task at hand. For instance, students' ID numbers are irrelevant to the task of predicting students' grade point averages. Redundant and irrelevant features can reduce classification accuracy and the quality of the clusters that are found.

While some irrelevant and redundant attributes can be eliminated immediately by using common sense or domain knowledge, selecting the best subset of features frequently requires a systematic approach. The ideal approach to feature selection is to try all possible subsets of features as input to the data mining algorithm of interest, and then take the subset that produces the best results. This method has the advantage of reflecting the objective and bias of the data mining algorithm that will eventually be used. Unfortunately, since the number of subsets involving n attributes is 2^n , such an approach is impractical in most situations and alternative strategies are needed. There are three standard approaches to feature selection: embedded, filter, and wrapper.

Embedded approaches

Feature selection occurs naturally as part of the data mining algorithm. Specifically, during the operation of the data mining algorithm, the algorithm itself decides which attributes to use and which to ignore. Algorithms for building decision tree classifiers, which are discussed in [Chapter 3](#), often operate in this manner.

Filter approaches


Features are selected before the data mining algorithm is run, using some approach that is independent of the data mining task. For example, we might select sets of attributes whose pairwise correlation is as low as possible so that the attributes are non-redundant.

Wrapper approaches

These methods use the target data mining algorithm as a black box to find the best subset of attributes, in a way similar to that of the ideal algorithm described above, but typically without enumerating all possible subsets.

Because the embedded approaches are algorithm-specific, only the filter and wrapper approaches will be discussed further here.

An Architecture for Feature Subset Selection

It is possible to encompass both the filter and wrapper approaches within a common architecture. The feature selection process is viewed as consisting of four parts: a measure for evaluating a subset, a search strategy that controls the generation of a new subset of features, a stopping criterion, and a validation procedure. Filter methods and wrapper methods differ only in the way in which they evaluate a subset of features. For a wrapper method, subset evaluation uses the target data mining algorithm, while for a filter approach, the evaluation technique is distinct from the target data mining algorithm. The following discussion provides some details of this approach, which is summarized in [Figure 2.11](#) .

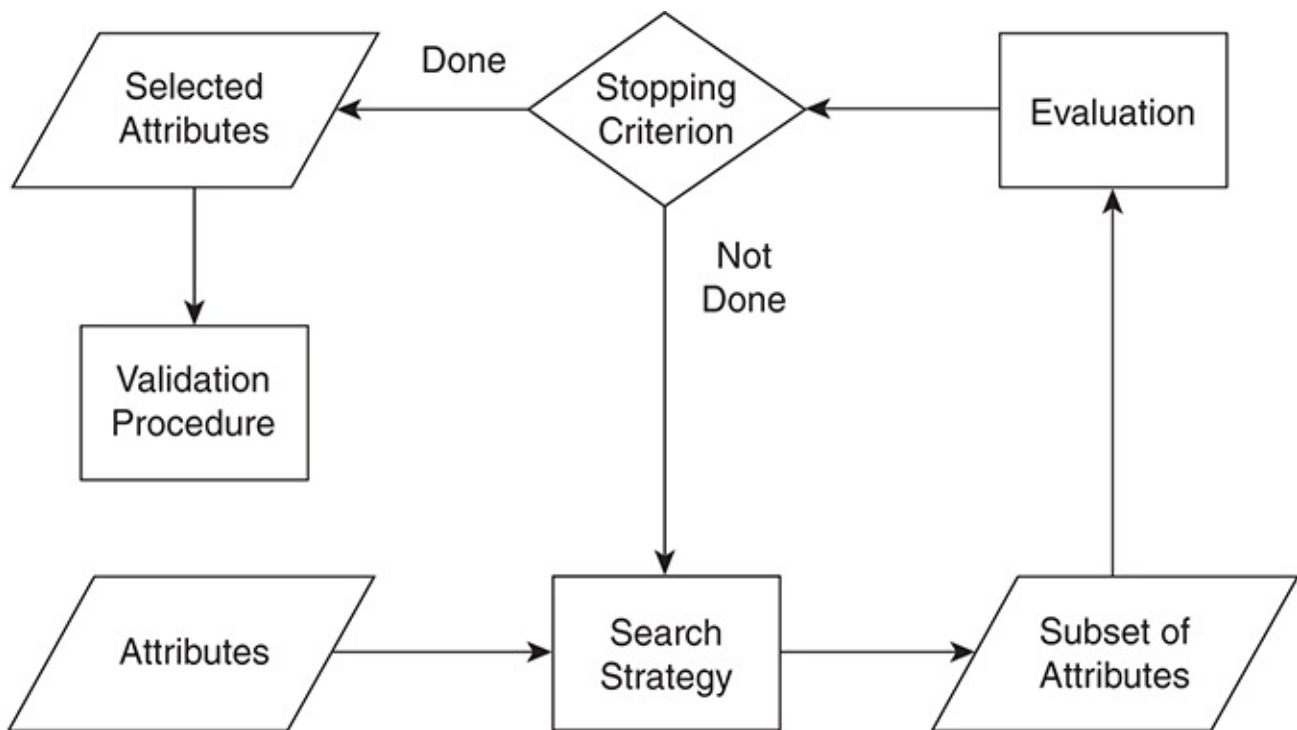


Figure 2.11.

Flowchart of a feature subset selection process.

Conceptually, feature subset selection is a search over all possible subsets of features. Many different types of search strategies can be used, but the search strategy should be computationally inexpensive and should find optimal or near optimal sets of features. It is usually not possible to satisfy both requirements, and thus, trade-offs are necessary.

An integral part of the search is an evaluation step to judge how the current subset of features compares to others that have been considered. This requires an evaluation measure that attempts to determine the goodness of a subset of attributes with respect to a particular data mining task, such as classification or clustering. For the filter approach, such measures attempt to predict how well the actual data mining algorithm will perform on a given set of attributes. For the wrapper approach, where evaluation consists of actually running the target data mining algorithm, the subset evaluation function is simply the criterion normally used to measure the result of the data mining.

Because the number of subsets can be enormous and it is impractical to examine them all, some sort of stopping criterion is necessary. This strategy is usually based on one or more conditions involving the following: the number of iterations, whether the value of the subset evaluation measure is optimal or exceeds a certain threshold, whether a subset of a certain size has been obtained, and whether any improvement can be achieved by the options available to the search strategy.

Finally, once a subset of features has been selected, the results of the target data mining algorithm on the selected subset should be validated. A straightforward validation approach is to run the algorithm with the full set of features and compare the full results to results obtained using the subset of features. Hopefully, the subset of features will produce results that are better than or almost as good as those produced when using all features. Another validation approach is to use a number of different feature selection algorithms to obtain subsets of features and then compare the results of running the data mining algorithm on each subset.

Feature Weighting

Feature weighting is an alternative to keeping or eliminating features. More important features are assigned a higher weight, while less important features are given a lower weight. These weights are sometimes assigned based on domain knowledge about the relative importance of features. Alternatively, they can sometimes be determined automatically. For example, some classification schemes, such as support vector machines ([Chapter 4](#)), produce classification models in which each feature is given a weight. Features with larger weights play a more important role in the model. The normalization of objects that takes place when computing the cosine similarity ([Section 2.4.5](#)) can also be regarded as a type of feature weighting.


2.3.5 Feature Creation

It is frequently possible to create, from the original attributes, a new set of attributes that captures the important information in a data set much more effectively. Furthermore, the number of new attributes can be smaller than the number of original attributes, allowing us to reap all the previously described benefits of dimensionality reduction. Two related methodologies for creating new attributes are described next: feature extraction and mapping the data to a new space.

Feature Extraction

The creation of a new set of features from the original raw data is known as **feature extraction**. Consider a set of photographs, where each photograph is to be classified according to whether it contains a human face. The raw data is a set of pixels, and as such, is not suitable for many types of classification algorithms. However, if the data is processed to provide higher-level features, such as the presence or absence of certain types of edges and areas that are highly correlated with the presence of human faces, then a much broader set of classification techniques can be applied to this problem.


Unfortunately, in the sense in which it is most commonly used, feature extraction is highly domain-specific. For a particular field, such as image processing, various features and the techniques to extract them have been developed over a period of time, and often these techniques have limited applicability to other fields. Consequently, whenever data mining is applied to a relatively new area, a key task is the development of new features and feature extraction methods.

Although feature extraction is often complicated, [Example 2.10](#)  illustrates that it can be relatively straightforward.

Example 2.10 (Density).

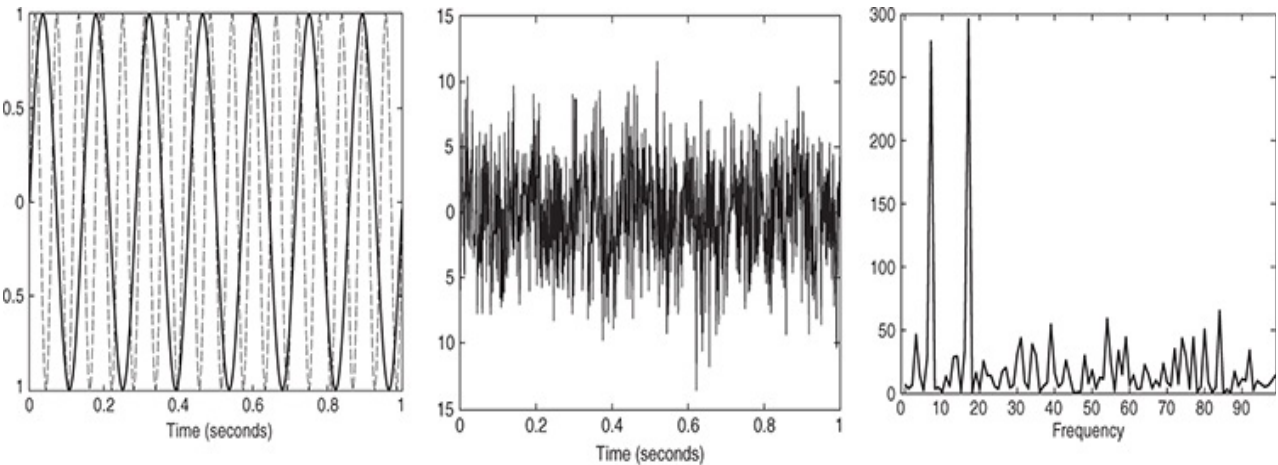
Consider a data set consisting of information about historical artifacts, which, along with other information, contains the volume and mass of each artifact. For simplicity, assume that these artifacts are made of a small number of materials (wood, clay, bronze, gold) and that we want to classify the artifacts with respect to the material of which they are made. In this case, a density feature constructed from the mass and volume features, i.e., $density = mass/volume$, would most directly yield an accurate classification. Although there have been some attempts to automatically perform such simple feature extraction by exploring basic mathematical combinations of existing attributes, the most common approach is to construct features using domain expertise.

Mapping the Data to a New Space

A totally different view of the data can reveal important and interesting features. Consider, for example, time series data, which often contains periodic patterns. If there is only a single periodic pattern and not much noise, then the pattern is easily detected. If, on the other hand, there are a number of periodic patterns and a significant amount of noise, then these patterns are hard to detect. Such patterns can, nonetheless, often be detected by applying a **Fourier transform** to the time series in order to change to a representation in which frequency information is explicit. In [Example 2.11](#) , it will not be necessary to know the details of the Fourier transform. It is enough to know that, for each time series, the Fourier transform produces a new data object whose attributes are related to frequencies.

Example 2.11 (Fourier Analysis).

The time series presented in [Figure 2.12\(b\)](#) is the sum of three other time series, two of which are shown in [Figure 2.12\(a\)](#) and have frequencies of 7 and 17 cycles per second, respectively. The third time series is random noise. [Figure 2.12\(c\)](#) shows the power spectrum that can be computed after applying a Fourier transform to the original time series. (Informally, the power spectrum is proportional to the square of each frequency attribute.) In spite of the noise, there are two peaks that correspond to the periods of the two original, non-noisy time series. Again, the main point is that better features can reveal important aspects of the data.



(a) Two time series.

(b) Noisy time series.

(c) Power spectrum.

Figure 2.12.

Application of the Fourier transform to identify the underlying frequencies in time series data.

Many other sorts of transformations are also possible. Besides the Fourier transform, the **wavelet transform** has also proven very useful for time series and other types of data.

2.3.6 Discretization and Binarization

Some data mining algorithms, especially certain classification algorithms, require that the data be in the form of categorical attributes. Algorithms that find association patterns require that the data be in the form of binary attributes. Thus, it is often necessary to transform a continuous attribute into a categorical attribute (**discretization**), and both continuous and discrete attributes may need to be transformed into one or more binary attributes (**binarization**). Additionally, if a categorical attribute has a large number of values (categories), or some values occur infrequently, then it can be beneficial for certain data mining tasks to reduce the number of categories by combining some of the values.

As with feature selection, the best discretization or binarization approach is the one that “produces the best result for the data mining algorithm that will be used to analyze the data.” It is typically not practical to apply such a criterion directly. Consequently, discretization or binarization is performed in a way that satisfies a criterion that is thought to have a relationship to good performance for the data mining task being considered. In general, the best discretization depends on the algorithm being used, as well as the other attributes being considered. Typically, however, the discretization of each attribute is considered in isolation.

Binarization

A simple technique to binarize a categorical attribute is the following: If there are m categorical values, then uniquely assign each original value to an integer in the interval $[0, m-1]$. If the attribute is ordinal, then order must be maintained by the assignment. (Note that even if the attribute is originally represented using integers, this process is necessary if the integers are not in

the interval $[0, m-1]$.) Next, convert each of these m integers to a binary number. Since $n = \lceil \log_2(m) \rceil$ binary digits are required to represent these integers, represent these binary numbers using n binary attributes. To illustrate, a categorical variable with 5 values $\{awful, poor, OK, good, great\}$ would require three binary variables $x_1, x_2,$ and x_3 . The conversion is shown in [Table 2.5](#).

Table 2.5. Conversion of a categorical attribute to three binary attributes.

Categorical Value	Integer Value	x_1	x_2	x_3
<i>awful</i>	0	0	0	0
<i>poor</i>	1	0	0	1
<i>OK</i>	2	0	1	0
<i>good</i>	3	0	1	1
<i>great</i>	4	1	0	0

Such a transformation can cause complications, such as creating unintended relationships among the transformed attributes. For example, in [Table 2.5](#), attributes x_2 and x_3 are correlated because information about the *good* value is encoded using both attributes. Furthermore, association analysis requires asymmetric binary attributes, where only the presence of the attribute (value = 1) is important. For association problems, it is therefore necessary to introduce one asymmetric binary attribute for each categorical value, as shown in [Table 2.6](#). If the number of resulting attributes is too large, then the techniques described in the following sections can be used to reduce the number of categorical values before binarization.

Table 2.6. Conversion of a categorical attribute to five asymmetric binary

attributes.

Categorical Value	Integer Value	x1	x2	x3	x4	x5
<i>awful</i>	0	1	0	0	0	0
<i>poor</i>	1	0	1	0	0	0
<i>OK</i>	2	0	0	1	0	0
<i>good</i>	3	0	0	0	1	0
<i>great</i>	4	0	0	0	0	1


Likewise, for association problems, it can be necessary to replace a single binary attribute with two asymmetric binary attributes. Consider a binary attribute that records a person's gender, male or female. For traditional association rule algorithms, this information needs to be transformed into two asymmetric binary attributes, one that is a 1 only when the person is male and one that is a 1 only when the person is female. (For asymmetric binary attributes, the information representation is somewhat inefficient in that two bits of storage are required to represent each bit of information.)

Discretization of Continuous Attributes


Discretization is typically applied to attributes that are used in classification or association analysis. Transformation of a continuous attribute to a categorical attribute involves two subtasks: deciding how many categories, n , to have and determining how to map the values of the continuous attribute to these categories. In the first step, after the values of the continuous attribute are sorted, they are then divided into n intervals by specifying $n-1$ **split points**. In the second, rather trivial step, all the values in one interval are mapped to the same categorical value. Therefore, the problem of discretization is one of

deciding how many split points to choose and where to place them. The result can be represented either as a set of intervals $\{(x_0, x_1], (x_1, x_2], \dots, (x_{n-1}, x_n)\}$, where x_0 and x_n can be $+\infty$ or $-\infty$, respectively, or equivalently, as a series of inequalities $x_0 < x \leq x_1, \dots, x_{n-1} < x < x_n$.

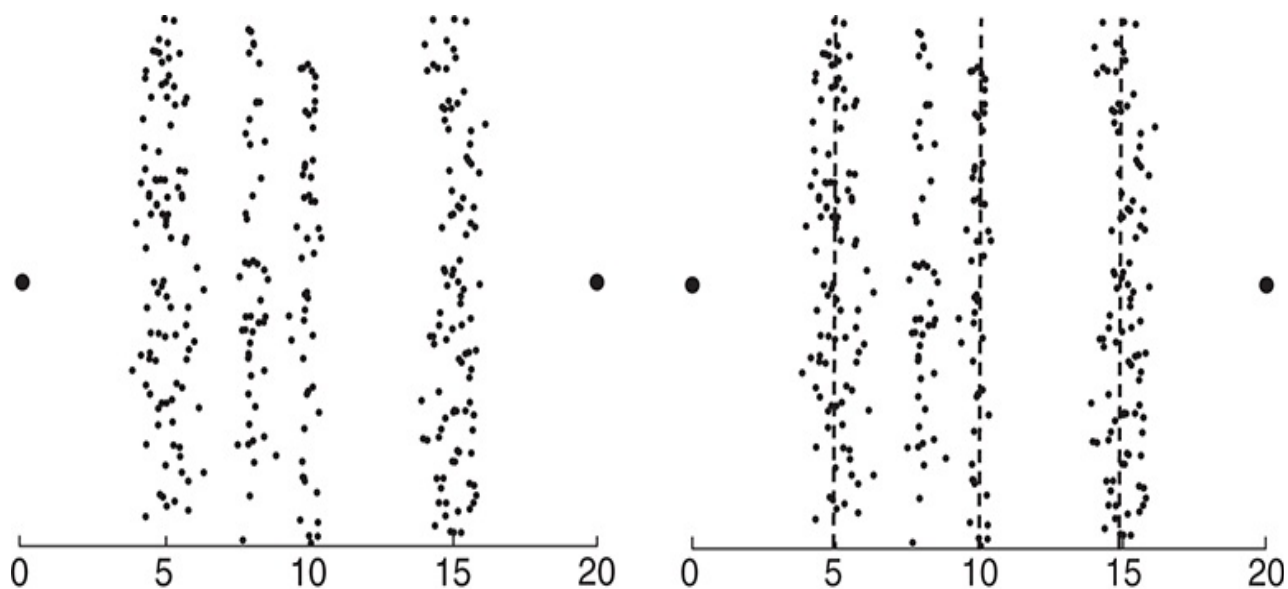
Unsupervised Discretization

A basic distinction between discretization methods for classification is whether class information is used (supervised) or not (unsupervised). If class information is not used, then relatively simple approaches are common. For instance, the **equal width** approach divides the range of the attribute into a user-specified number of intervals each having the same width. Such an approach can be badly affected by outliers, and for that reason, an **equal frequency (equal depth)** approach, which tries to put the same number of objects into each interval, is often preferred. As another example of unsupervised discretization, a clustering method, such as K-means (see [Chapter 7](#) ) , can also be used. Finally, visually inspecting the data can sometimes be an effective approach.

Example 2.12 (Discretization Techniques).

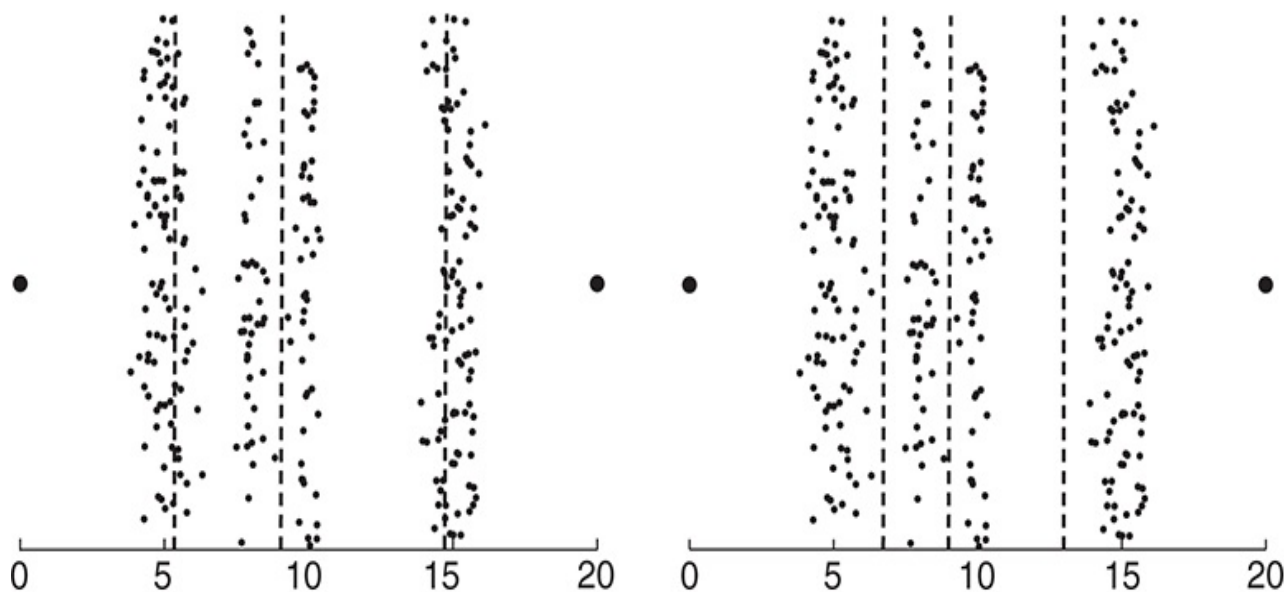
This example demonstrates how these approaches work on an actual data set. [Figure 2.13\(a\)](#)  shows data points belonging to four different groups, along with two outliers—the large dots on either end. The techniques of the previous paragraph were applied to discretize the x values of these data points into four categorical values. (Points in the data set have a random y component to make it easy to see how many points are in each group.) Visually inspecting the data works quite well, but is not automatic, and thus, we focus on the other three approaches. The split points produced by the techniques equal width, equal frequency, and K-means are shown in

Figures 2.13(b) □, 2.13(c) □, and 2.13(d) □, respectively. The split points are represented as dashed lines.



(a) Original data.

(b) Equal width discretization.



(c) Equal frequency discretization.

(d) K-means discretization.

Figure 2.13.

Different discretization techniques.

In this particular example, if we measure the performance of a discretization technique by the extent to which different objects that clump together have the same categorical value, then K-means performs best, followed by equal frequency, and finally, equal width. More generally, the best discretization will depend on the application and often involves domain-specific discretization. For example, the discretization of people into low income, middle income, and high income is based on economic factors.

Supervised Discretization

If classification is our application and class labels are known for some data objects, then discretization approaches that use class labels often produce better classification. This should not be surprising, since an interval constructed with no knowledge of class labels often contains a mixture of class labels. A conceptually simple approach is to place the splits in a way that maximizes the purity of the intervals, i.e., the extent to which an interval contains a single class label. In practice, however, such an approach requires potentially arbitrary decisions about the purity of an interval and the minimum size of an interval.

To overcome such concerns, some statistically based approaches start with each attribute value in a separate interval and create larger intervals by merging adjacent intervals that are similar according to a statistical test. An alternative to this bottom-up approach is a top-down approach that starts by bisecting the initial values so that the resulting two intervals give minimum entropy. This technique only needs to consider each value as a possible split point, because it is assumed that intervals contain ordered sets of values. The splitting process is then repeated with another interval, typically choosing the interval with the worst (highest) entropy, until a user-specified number of intervals is reached, or a stopping criterion is satisfied.

Entropy-based approaches are one of the most promising approaches to discretization, whether bottom-up or top-down. First, it is necessary to define **entropy**. Let k be the number of different class labels, m_i be the number of values in the i^{th} interval of a partition, and m_{ij} be the number of values of class j in interval i . Then the entropy e_i of the i^{th} interval is given by the equation




$$e_i = -\sum_{j=1}^k p_{ij} \log_2 p_{ij},$$

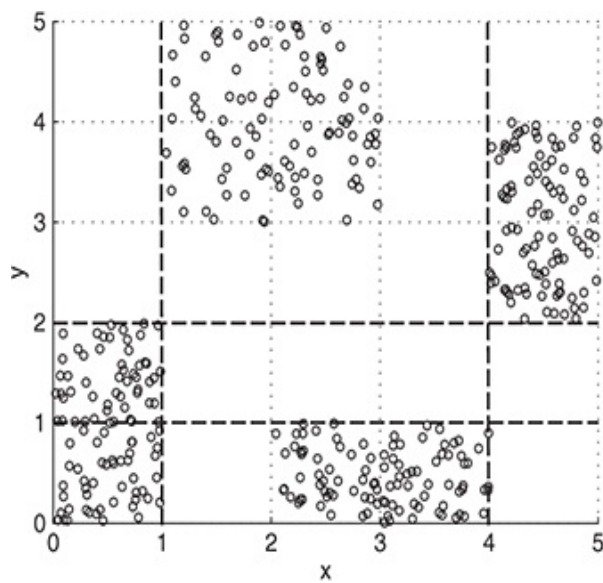
where $p_{ij} = m_{ij}/m_i$ is the probability (fraction of values) of class j in the i^{th} interval. The total entropy, e , of the partition is the weighted average of the individual interval entropies, i.e.,

$$e = \sum_{i=1}^n w_i e_i,$$

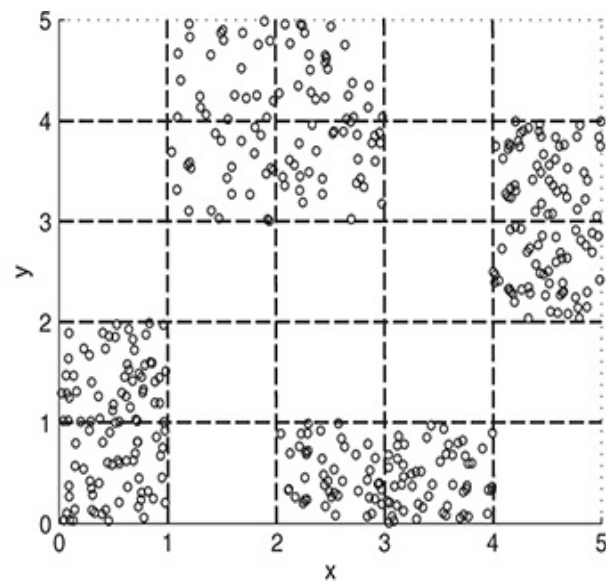
where m is the number of values, $w_i = m_i/m$ is the fraction of values in the i^{th} interval, and n is the number of intervals. Intuitively, the entropy of an interval is a measure of the purity of an interval. If an interval contains only values of one class (is perfectly pure), then the entropy is 0 and it contributes nothing to the overall entropy. If the classes of values in an interval occur equally often (the interval is as impure as possible), then the entropy is a maximum.

Example 2.13 (Discretization of Two Attributes).

The top-down method based on entropy was used to independently discretize both the x and y attributes of the two-dimensional data shown in **Figure 2.14** . In the first discretization, shown in **Figure 2.14(a)** , the x and y attributes were both split into three intervals. (The dashed lines indicate the split points.) In the second discretization, shown in **Figure 2.14(b)** , the x and y attributes were both split into five intervals.



(a) Three intervals



(b) Five intervals

Figure 2.14.

Discretizing x and y attributes for four groups (classes) of points.

This simple example illustrates two aspects of discretization. First, in two dimensions, the classes of points are well separated, but in one dimension, this is not so. In general, discretizing each attribute separately often guarantees suboptimal results. Second, five intervals work better than three, but six intervals do not improve the discretization much, at least in terms of entropy. (Entropy values and results for six intervals are not shown.) Consequently, it is desirable to have a stopping criterion that automatically finds the right number of partitions.

Categorical Attributes with Too Many Values

Categorical attributes can sometimes have too many values. If the categorical attribute is an ordinal attribute, then techniques similar to those for continuous attributes can be used to reduce the number of categories. If the categorical attribute is nominal, however, then other approaches are needed. Consider a

university that has a large number of departments. Consequently, a *department name* attribute might have dozens of different values. In this situation, we could use our knowledge of the relationships among different departments to combine departments into larger groups, such as *engineering*, *social sciences*, or *biological sciences*. If domain knowledge does not serve as a useful guide or such an approach results in poor classification performance, then it is necessary to use a more empirical approach, such as grouping values together only if such a grouping results in improved classification accuracy or achieves some other data mining objective.

2.3.7 Variable Transformation

A **variable transformation** refers to a transformation that is applied to all the values of a variable. (We use the term variable instead of attribute to adhere to common usage, although we will also refer to attribute transformation on occasion.) In other words, for each object, the transformation is applied to the value of the variable for that object. For example, if only the magnitude of a variable is important, then the values of the variable can be transformed by taking the absolute value. In the following section, we discuss two important types of variable transformations: simple functional transformations and normalization.

Simple Functions

For this type of variable transformation, a simple mathematical function is applied to each value individually. If x is a variable, then examples of such transformations include x^k , $\log x$, e^x , x , $1/x$, $\sin x$, or $|x|$. In statistics, variable transformations, especially \sqrt{x} , \log , and $1/x$, are often used to transform data that does not have a Gaussian (normal) distribution into data that does. While

this can be important, other reasons often take precedence in data mining. Suppose the variable of interest is the number of data bytes in a session, and the number of bytes ranges from 1 to 1 billion. This is a huge range, and it can be advantageous to compress it by using a \log_{10} transformation. In this case, sessions that transferred 108 and 109 bytes would be more similar to each other than sessions that transferred 10 and 1000 bytes ($9-8=1$ versus $3-1=3$). For some applications, such as network intrusion detection, this may be what is desired, since the first two sessions most likely represent transfers of large files, while the latter two sessions could be two quite distinct types of sessions.

Variable transformations should be applied with caution because they change the nature of the data. While this is what is desired, there can be problems if the nature of the transformation is not fully appreciated. For instance, the transformation $1/x$ reduces the magnitude of values that are 1 or larger, but increases the magnitude of values between 0 and 1. To illustrate, the values $\{1, 2, 3\}$ go to $\{1, 1/2, 1/3\}$, but the values $\{1, 1/2, 1/3\}$ go to $\{1, 2, 3\}$. Thus, for all sets of values, the transformation $1/x$ reverses the order. To help clarify the effect of a transformation, it is important to ask questions such as the following: What is the desired property of the transformed attribute? Does the order need to be maintained? Does the transformation apply to all values, especially negative values and 0? What is the effect of the transformation on the values between 0 and 1? [Exercise 17](#) on [page 109](#) explores other aspects of variable transformation.

Normalization or Standardization

The goal of standardization or normalization is to make an entire set of values have a particular property. A traditional example is that of “standardizing a variable” in statistics. If \bar{x} is the mean (average) of the attribute values and s_x is their standard deviation, then the transformation $x'=(x-\bar{x})/s_x$ creates a new

variable that has a mean of 0 and a standard deviation of 1. If different variables are to be used together, e.g., for clustering, then such a transformation is often necessary to avoid having a variable with large values dominate the results of the analysis. To illustrate, consider comparing people based on two variables: age and income. For any two people, the difference in income will likely be much higher in absolute terms (hundreds or thousands of dollars) than the difference in age (less than 150). If the differences in the range of values of age and income are not taken into account, then the comparison between people will be dominated by differences in income. In particular, if the similarity or dissimilarity of two people is calculated using the similarity or dissimilarity measures defined later in this chapter, then in many cases, such as that of Euclidean distance, the income values will dominate the calculation.

The mean and standard deviation are strongly affected by outliers, so the above transformation is often modified. First, the mean is replaced by the **median**, i.e., the middle value. Second, the standard deviation is replaced by the **absolute standard deviation**. Specifically, if x is a variable, then the absolute standard deviation of x is given by $\sigma_A = \frac{1}{m} \sum_{i=1}^m |x_i - \mu|$, where x_i is the i th value of the variable, m is the number of objects, and μ is either the mean or median. Other approaches for computing estimates of the location (center) and spread of a set of values in the presence of outliers are described in statistics books. These more robust measures can also be used to define a standardization transformation.