2

PREPARING THE DATA

Chapter Objectives

- Analyze basic representations and characteristics of raw and large data sets.
- Apply different normalization techniques on numerical attributes.
- Recognize different techniques for data preparation, including attribute transformation.
- Compare different methods for elimination of missing values.
- Construct a method for uniform representation of time-dependent data.
- Compare different techniques for outlier detection.
- Implement some data preprocessing techniques.

2.1 REPRESENTATION OF RAW DATA

Data samples introduced as rows in Figure 1.4 are basic components in a data-mining process. Every sample is described with several features, and there are different types of values for every feature. We will start with the two most common types: *numeric*

Data Mining: Concepts, Models, Methods, and Algorithms, Second Edition. Mehmed Kantardzic.

and *categorical*. Numeric values include real-value variables or integer variables such as age, speed, or length. A feature with numeric values has two important properties: Its values have an order relation ($2 < 5$ and $5 < 7$) and a distance relation (d [2.3, 4.2] = 1.9).

In contrast, categorical (often called symbolic) variables have neither of these two relations. The two values of a categorical variable can be either equal or not equal: They only support an equality relation (Blue = Blue, or Red ≠ Black). Examples of variables of this type are eye color, sex, or country of citizenship. A categorical variable with two values can be converted, in principle, to a numeric binary variable with two values: 0 or 1. A categorical variable with n values can be converted into n binary numeric variables, namely, one binary variable for each categorical value. These coded categorical variables are known as "dummy variables" in statistics. For example, if the variable eye color has four values (black, blue, green, and brown), they can be coded with four binary digits.

Feature Value	Code
Black	1000
Blue	0100
Green	0010
Brown	0001

Another way of classifying a variable, based on its values, is to look at it as a continuous variable or a discrete variable.

Continuous variables are also known as *quantitative* or *metric variables*. They are measured using either an interval scale or a ratio scale. Both scales allow the underlying variable to be defined or measured theoretically with infinite precision. The difference between these two scales lies in how the 0 point is defined in the scale. The 0 point in the *interval scale* is placed arbitrarily, and thus it does not indicate the complete absence of whatever is being measured. The best example of the interval scale is the temperature scale, where 0 degrees Fahrenheit does not mean a total absence of temperature. Because of the arbitrary placement of the 0 point, the ratio relation does not hold true for variables measured using interval scales. For example, 80 degrees Fahrenheit does not imply twice as much heat as 40 degrees Fahrenheit. In contrast, a *ratio scale* has an absolute 0 point and, consequently, the ratio relation holds true for variables measured using this scale. Quantities such as height, length, and salary use this type of scale. Continuous variables are represented in large data sets with values that are numbers—real or integers.

Discrete variables are also called qualitative variables. Such variables are measured, or their values defined, using one of two kinds of nonmetric scales—*nominal* or *ordinal*. A nominal scale is an orderless scale, which uses different symbols, characters, and numbers to represent the different states (values) of the variable being measured. An example of a nominal variable, a utility, customer-type identifier with possible values is residential, commercial, and industrial. These values can be coded alphabetically as A, B, and C, or numerically as 1, 2, or 3, but they do not have metric

Type	Description	Examples	Operations
Nominal	Uses a label or name to distinguish one object from another.	Zip code, ID, gender	= or not =
Ordinal	Uses values to provide the ordering of objects.	Opinion, grades	< or >
Interval	Uses units of measurement, but the origin is arbitrary.	Celsius or Fahrenheit, calendar dates	+ or –
Ratio	Uses units of measurement, and the origin is not arbitrary.	Temperature in Kelvin, length, counts, age, income	+, –, *, /

Figure 2.1. Variable types with examples.

characteristics as the other numeric data have. Another example of a nominal attribute is the zip code field available in many data sets. In both examples, the numbers used to designate different attribute values have no particular order and no necessary relation to one another.

An *ordinal scale* consists of ordered, discrete gradations, for example, rankings. An ordinal variable is a categorical variable for which an order relation is defined but not a distance relation. Some examples of an ordinal attribute are the rank of a student in a class and the gold, silver, and bronze medal positions in a sports competition. The ordered scale need not be necessarily linear; for example, the difference between the students ranked fourth and fifth need not be identical to the difference between the students ranked 15th and 16th. All that can be established from an ordered scale for ordinal attributes with greater-than, equal-to, or less-than relations. Typically, ordinal variables encode a numeric variable onto a small set of overlapping intervals corresponding to the values of an ordinal variable. These ordinal variables are closely related to the linguistic or fuzzy variables commonly used in spoken English, for example, AGE (with values young, middle aged, and old) and INCOME (with values low-middle class, upper middle class, and rich). More examples are given in Figure 2.1, and the formalization and use of fuzzy values in a data-mining process are given in Chapter 14.

A special class of discrete variables is periodic variables. A *periodic variable* is a feature for which the distance relation exists, but there is no order relation. Examples are days of the week, days of the month, or days of the year. Monday and Tuesday, as the values of a feature, are closer than Monday and Thursday, but Monday can come before or after Friday.

Finally, one additional dimension of classification of data is based on its behavior with respect to time. Some data do not change with time, and we consider them *static data*. On the other hand, there are attribute values that change with time, and this type of data we call *dynamic* or *temporal data*. The majority of data-mining methods are more suitable for static data, and special consideration and some preprocessing are often required to mine dynamic data.

Most data-mining problems arise because there are large amounts of samples with different types of features. Additionally, these samples are very often high dimensional,

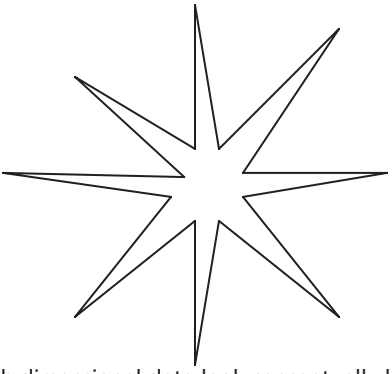

Figure 2.2. High-dimensional data look conceptually like a porcupine.

which means they have extremely large number of measurable features. This additional dimension of large data sets causes the problem known in data-mining terminology as "the curse of dimensionality." The "curse of dimensionality" is produced because of the geometry of high-dimensional spaces, and these kinds of data spaces are typical for data-mining problems. The properties of high-dimensional spaces often appear counterintuitive because our experience with the physical world is in a low-dimensional space, such as a space with two or three dimensions. Conceptually, objects in high-dimensional spaces have a larger surface area for a given volume than objects in low-dimensional spaces. For example, a high-dimensional hypercube, if it could be visualized, would look like a porcupine, as in Figure 2.2. As the dimensionality grows larger, the edges grow longer relative to the size of the central part of the hypercube. Four important properties of high-dimensional data affect the interpretation of input data and data-mining results.

1. The size of a data set yielding the same density of data points in an n-dimensional space increases exponentially with dimensions. For example, if a one-dimensional (1-D) sample containing n data points has a satisfactory level of density, then to achieve the same density of points in k dimensions, we need n^k data points. If integers 1 to 100 are values of 1-D samples, where the domain of the dimension is [0, 100], then to obtain the same density of samples in a 5-D space we will need $100^5 = 10^{10}$ different samples. This is true even for the largest real-world data sets; because of their large dimensionality, the density of samples is still relatively low and, very often, unsatisfactory for data-mining purposes.
2. A larger radius is needed to enclose a fraction of the data points in a high-dimensional space. For a given fraction of samples, it is possible to determine the edge length *e* of the hypercube using the formula

$$e(p) = p^{1/d}$$

where p is the prespecified fraction of samples, and d is the number of dimensions. For example, if one wishes to enclose 10% of the samples (p = 0.1), the

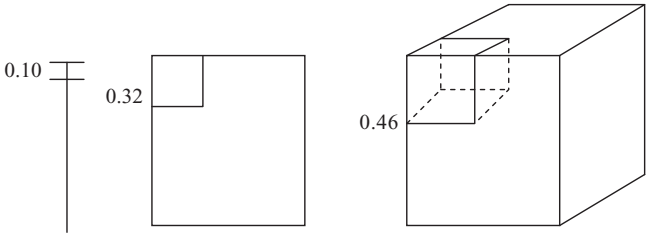

Figure 2.3. Regions enclose 10% of the samples for one-, two-, and three-dimensional spaces.

corresponding edge for a 2-D space will be $e_2(0.1) = 0.32$, for a 3-D space $e_3(0.1) = 0.46$, and for a 10-D space $e_{10}(0.1) = 0.80$. Graphical interpretation of these edges is given in Figure 2.3.

This shows that a very large neighborhood is required to capture even a small portion of the data in a high-dimensional space.

3. Almost every point is closer to an edge than to another sample point in a high-dimensional space. For a sample of size n, the expected distance D between data points in a d-dimensional space is

$$D(d, n) = \tfrac{1}{2}(1/n)^{1/d}$$

For example, for a 2-D space with 10,000 points the expected distance is D(2,10,000) = 0.005 and for a 10-D space with the same number of sample points D(10,10,000) = 0.4. Keep in mind that the maximum distance from any point to the edge occurs at the center of the distribution, and it is 0.5 for normalized values of all dimensions.

4. Almost every point is an outlier. As the dimension of the input space increases, the distance between the prediction point and the center of the classified points increases. For example, when d = 10, the expected value of the prediction point is 3.1 standard deviations away from the center of the data belonging to one class. When d = 20, the distance is 4.4 standard deviations. From this standpoint, the prediction of every new point looks like an outlier of the initially classified data. This is illustrated conceptually in Figure 2.2, where predicted points are mostly in the edges of the porcupine, far from the central part.

These rules of the "curse of dimensionality" most often have serious consequences when dealing with a finite number of samples in a high-dimensional space. From properties (1) and (2) we see the difficulty of making local estimates for high-dimensional samples; we need more and more samples to establish the required data density for performing planned mining activities. Properties (3) and (4) indicate the difficulty of predicting a response at a given point, since any new point will on average be closer to an edge than to the training examples in the central part.

One interesting experiment, performed recently by a group of students, shows the importance of understanding curse-of-dimensionality concepts for data-mining tasks.

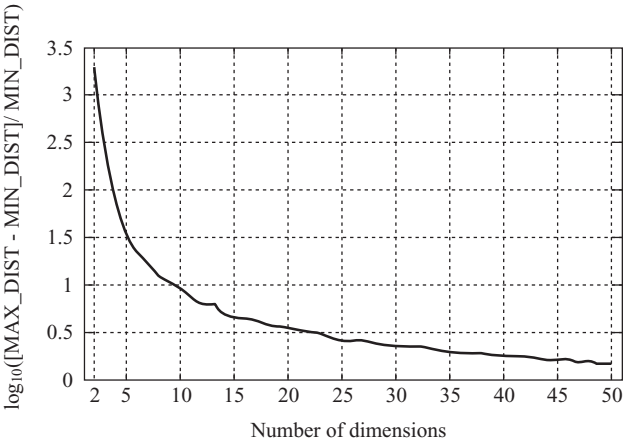

Figure 2.4. With large number of dimensions, the concept of a distance changes the meaning.

They generated randomly 500 points for different n-dimensional spaces. The number of dimensions was between 2 and 50. Then, they measured in each space all distances between any pair of points and calculated the parameter P:

$$P_n = \log_{10}([\text{MAX\_DIST}_n - \text{MIN\_DIST}_n]/\text{MIN\_DIST}_n)$$

where n is the number of dimensions, and MAX-DIST and MIN-DIST are maximum and minimum distances in the given space, respectively. The results are presented in Figure 2.4. What is interesting from the graph is that as the number of dimensions increases, the parameter P_n approaches the value of 0. That means maximum and minimum distances are becoming very close in these spaces; in other words, there are no differences in distances between any two points in these large-dimensional spaces. It is an experimental confirmation that traditional definitions of density and distance between points, which are critical for many data-mining tasks, change their meaning. When dimensionality of a data set increases, data become increasingly sparse, with mostly outliers in the space that they occupy. Therefore, we have to revisit and reevaluate traditional concepts from statistics: distance, similarity, data distribution, mean, standard deviation, and so on.

2.2 CHARACTERISTICS OF RAW DATA

All raw data sets initially prepared for data mining are often large; many are related to human beings and have the potential for being messy. A priori, one should expect to find missing values, distortions, misrecording, inadequate sampling, and so on in these initial data sets. Raw data that do not appear to show any of these problems should immediately arouse suspicion. The only real reason for the high quality of data could be that the presented data have been cleaned up and preprocessed before the analyst sees them, as in data of a correctly designed and prepared data warehouse.

Let us see what the sources and implications of messy data are. First, data may be *missing* for a huge variety of reasons. Sometimes there are mistakes in measurements or recordings, but in many cases, the value is unavailable. To cope with this in a data-mining process, one must be able to model with the data that are presented, even with their values missing. We will see later that some data-mining techniques are more or less sensitive to missing values. If the method is robust enough, then the missing values are not a problem. Otherwise, it is necessary to solve the problem of missing values before the application of a selected data-mining technique. The second cause of messy data is *misrecording of data*, and that is typical in large volumes of data. We have to have mechanisms to discover some of these "unusual" values, and in some cases, even to work with them to eliminate their influence on the final results. Further, data may *not* be *from the population* they are supposed to be from. Outliers are typical examples here, and they require careful analysis before the analyst can decide whether they should be dropped from the data-mining process as anomalous or included as unusual examples from the population under study.

It is very important to examine the data thoroughly before undertaking any further steps in formal analysis. Traditionally, data-mining analysts had to familiarize themselves with their data before beginning to model them or use them with some data-mining algorithms. However, with the large size of modern data sets, this is less feasible or even entirely impossible in many cases. Here we must rely on computer programs to check the data for us.

Distorted data, incorrect choice of steps in methodology, misapplication of data-mining tools, too idealized a model, a model that goes beyond the various sources of uncertainty and ambiguity in the data—all these represent possibilities for taking the wrong direction in a data-mining process. Therefore, data mining is not just a matter of simply applying a directory of tools to a given problem, but rather a process of critical assessments, exploration, testing, and evaluation. The data should be well-defined, consistent, and nonvolatile in nature. The quantity of data should be large enough to support data analysis, querying, reporting, and comparisons of historical data over a long period of time.

Many experts in data mining will agree that one of the most critical steps in a data-mining process is the preparation and transformation of the initial data set. This task often receives little attention in the research literature, mostly because it is considered too application-specific. But, in most data-mining applications, some parts of a data-preparation process or, sometimes, even the entire process can be described independently of an application and a data-mining method. For some companies with extremely large and often distributed data sets, most of the data-preparation tasks can be performed during the design of the data warehouse, but many specialized transformations may be initialized only when a data-mining analysis is requested.

Raw data are not always (in our opinion very seldom) the best data set for data mining. Many transformations may be needed to produce features more useful for selected data-mining methods such as prediction or classification. Counting in different ways, using different sampling sizes, taking important ratios, varying data-window sizes for time-dependent data, and including changes in moving averages (MA) may all contribute to better data-mining results. Do not expect that the machine will find the

best set of transformations without human assistance, and do not expect that transformations used in one data-mining application are the best for another.

The preparation of data is sometimes dismissed as a minor topic in the data-mining literature and used just formally as a phase in a data-mining process. In the real world of data-mining applications, the situation is reversed. More effort is expended preparing data than applying data-mining methods. There are two central tasks for the preparation of data:

1. organizing data into a standard form that is ready for processing by data-mining and other computer-based tools (a standard form is a relational table), and
2. preparing data sets that lead to the best data-mining performances.

2.3 TRANSFORMATION OF RAW DATA

We will review a few general types of transformations of data that are not problem-dependent and that may improve data-mining results. Selection of techniques and use in particular applications depend on types of data, amounts of data, and general characteristics of the data-mining task.

2.3.1 Normalizations

Some data-mining methods, typically those that are based on distance computation between points in an n-dimensional space, may need normalized data for best results. The measured values can be scaled to a specific range, for example, [−1, 1], or [0, 1]. If the values are not normalized, the distance measures will overweight those features that have, on average, larger values. There are many ways of normalizing data. The following are three simple and effective normalization techniques.

Decimal Scaling. Decimal scaling moves the decimal point but still preserves most of the original digit value. The typical scale maintains the values in a range of −1 to 1. The following equation describes decimal scaling, where v(i) is the value of the feature v for case i and v′(i) is a scaled value

$$v'(i) = v(i)/10^k$$

for the smallest k such that *max* (|v′(i)|) < 1.

First, the maximum |v′(i)| is found in the data set, and then the decimal point is moved until the new, scaled, maximum absolute value is less than 1. The divisor is then applied to all other v(i). For example, if the largest value in the set is 455, and the smallest value is −834, then the maximum absolute value of the feature becomes .834, and the divisor for all v(i) is 1000 (k = 3).

Min–Max Normalization. Suppose that the data for a feature v are in a range between 150 and 250. Then, the previous method of normalization will give all

normalized data between .15 and .25, but it will accumulate the values on a small subinterval of the entire range. To obtain better distribution of values on a whole normalized interval, for example, [0,1], we can use the min–max formula

$$v'(i) = (v(i) - min[v(i)]) / (max[v(i)] - min[v(i)])$$

where the minimum and the maximum values for the feature v are computed on a set automatically, or they are estimated by an expert in a given domain. Similar transformation may be used for the normalized interval [−1, 1]. The automatic computation of min and max values requires one additional search through the entire data set, but computationally, the procedure is very simple. On the other hand, expert estimations of min and max values may cause unintentional accumulation of normalized values.

Standard Deviation Normalization. Normalization by standard deviation often works well with distance measures but transforms the data into a form unrecognizable from the original data. For a feature v, the mean value *mean*(v) and the standard deviation *sd*(v) are computed for the entire data set. Then, for a case i, the feature value is transformed using the equation

$$v(i) = (v[i] - mean[v]) / sd(v)$$

For example, if the initial set of values of the attribute is v = {1, 2, 3}, then *mean*(v) = 2, *sd*(v) = 1, and the new set of normalized values is v* = {−1, 0, 1}.

Why not treat normalization as an implicit part of a data-mining method? The simple answer is that normalizations are useful for several diverse methods of data mining. Also very important is that the normalization is not a one-time or a one-phase event. If a method requires normalized data, available data should be initially transformed and prepared for the selected data-mining technique, but an identical normalization must be applied in all other phases of data-mining and with all new and future data. Therefore, the normalization parameters must be saved along with a solution.

2.3.2 Data Smoothing

A numeric feature, y, may range over many distinct values, sometimes as many as the number of training cases. For many data-mining techniques, minor differences among these values are not significant and may degrade the performance of the method and the final results. They may be considered as random variations of the same underlying value. Hence, it can be advantageous sometimes to smooth the values of the variable.

Many simple smoothers can be specified that average similar measured values. For example, if the values are real numbers with several decimal places, rounding the values to the given precision could be a simple smoothing algorithm for a large number of samples, where each sample has its own real value. If the set of values for the given feature F is {0.93, 1.01, 1.001, 3.02, 2.99, 5.03, 5.01, 4.98}, then it is obvious that smoothed values will be $F_{smoothed}$ = {1.0, 1.0, 1.0, 3.0, 3.0, 5.0, 5.0, 5.0}. This simple

transformation is performed without losing any quality in a data set, and, at the same time, it reduces the number of different real values for the feature to only three.

Some of these smoothing algorithms are more complex, and they are explained in Section 3.2. Reducing the number of distinct values for a feature means reducing the dimensionality of the data space at the same time. Reduced values are particularly useful for logic-based methods of data mining, as will be explained in Chapter 6. Smoothers in this case can be used to discretize continuous features into a set of features with binary true–false values.

2.3.3 Differences and Ratios

Even small changes to features can produce significant improvement in data-mining performances. The effects of relatively minor transformations of input or output features are particularly important in the specification of the data-mining goals. Two types of simple transformations, *differences* and *ratios,* could make improvements in goal specification, especially if they are applied to the output features.

These transformations sometimes produce better results than the simple, initial goal of predicting a number. In one application, for example, the objective is to move the controls for a manufacturing process to an optimal setting. But instead of optimizing the absolute magnitude specification for the output $s(t + 1)$, it is more effective to set the goal of a relative move from current value to a final optimal $s(t + 1) - s(t)$. The range of values for the relative moves is generally much smaller than the range of values for the absolute control setting. Therefore, for many data-mining methods, a smaller number of alternatives will improve the efficiency of the algorithm and will very often give better results.

Ratios are the second simple transformation of a target or output features. Using $s(t + 1)/s(t)$ as the output of a data-mining process instead of absolute value $s(t + 1)$ means that the level of increase or decrease in the values of a feature may also improve the performances of the entire mining process.

Differences and ratio transformations are not only useful for output features but also for inputs. They can be used as changes in time for one feature or as a composition of different input features. For example, in many medical data sets, there are two features of a patient (height and weight) that are taken as input parameters for different diagnostic analyses. Many applications show that better diagnostic results are obtained when an initial transformation is performed using a new feature called the body mass index (BMI), which is the weighted ratio between weight and height. This composite feature is better than the initial parameters to describe some of the characteristics of the patient, such as whether or not the patient is overweight.

Logical transformations can also be used to compose new features. For example, sometimes it is useful to generate a new feature that will determine the logical value of the relation $A > B$ between existing features A and B. But there are no universally best data-transformation methods. The lesson to be learned is that a major role remains for human insight while defining the problem. Attention should be paid to composing features, because relatively simple transformations can sometimes be far more effective for the final performance than switching to some other techniques of data mining.

2.4 MISSING DATA

For many real-world applications of data mining, even when there are huge amounts of data, the subset of cases with complete data may be relatively small. Available samples and also future cases may have values missing. Some of the data-mining methods accept missing values and satisfactorily process data to reach a final conclusion. Other methods require that all values be available. An obvious question is whether these missing values can be filled in during data preparation, prior to the application of the data-mining methods. The simplest solution for this problem is the reduction of the data set and the elimination of all samples with missing values. That is possible when large data sets are available, and missing values occur only in a small percentage of samples. If we do not drop the samples with missing values, then we have to find values for them. What are the practical solutions?

First, a data miner, together with the domain expert, can manually examine samples that have no values and enter a reasonable, probable, or expected value, based on a domain experience. The method is straightforward for small numbers of missing values and relatively small data sets. But, if there is no obvious or plausible value for each case, the miner is introducing noise into the data set by manually generating a value.

The second approach gives an even simpler solution for elimination of missing values. It is based on a formal, often automatic replacement of missing values with some constants, such as:

1. replace all missing values with a single global constant (a selection of a global constant is highly application dependent);
2. replace a missing value with its feature mean; and
3. replace a missing value with its feature mean for the given class (this approach is possible only for classification problems where samples are classified in advance).

These simple solutions are tempting. Their main flaw is that the substituted value is not the correct value. By replacing the missing value with a constant or changing the values for a few different features, the data are biased. The replaced value (values) will homogenize the cases with missing values into a uniform subset directed toward the class with the most missing values (an artificial class). If missing values are replaced with a single global constant for all features, an unknown value may be implicitly made into a positive factor that is not objectively justified.

One possible interpretation of missing values is that they are "don't care" values. In other words, we suppose that these values do not have any influence on the final data-mining results. In that case, a sample with the missing value may be extended to the set of artificial samples, where, for each new sample, the missing value is replaced with one of the possible feature values of a given domain. Although this interpretation may look more natural, the problem with this approach is the combinatorial explosion of artificial samples. For example, if one 3-D sample X is given as X = {1, ?, 3}, where the second feature's value is missing, the process will generate five artificial samples for the feature domain [0, 1, 2, 3, 4]

$$X_1 = \{1, 0, 3\}, X_2 = \{1, 1, 3\}, X_3 = \{1, 2, 3\}, X_4 = \{1, 3, 3\}, \text{and } X_5 = \{1, 4, 3\}$$

Finally, the data miner can generate a predictive model to predict each of the missing values. For example, if three features A, B, and C are given for each sample, then based on samples that have all three values as a training set, the data miner can generate a model of correlation between features. Different techniques, such as regression, Bayesian formalism, clustering, or decision-tree induction, may be used depending on data types (all these techniques are explained later in Chapters 5, 6, and 7). Once you have a trained model, you can present a new sample that has a value missing and generate a "predictive" value. For example, if values for features A and B are given, the model generates the value for the feature C. If a missing value is highly correlated with the other known features, this process will generate the best value for that feature. Of course, if you can always predict a missing value with certainty, this means that the feature is redundant in the data set and not necessary in further data-mining analyses. In real-world applications, you should expect an imperfect correlation between the feature with the missing value and other features. Therefore, all automatic methods fill in values that may not be correct. Such automatic methods are among the most popular in the data-mining community. In comparison to the other methods, they use the most information from the present data to predict missing values.

In general, it is speculative and often misleading to replace missing values using a simple, artificial schema of data preparation. It is best to generate multiple solutions of data mining with and without features that have missing values and then analyze and interpret them.

2.5 TIME-DEPENDENT DATA

Practical data-mining applications will range from those having strong time-dependent relationships to those with loose or no time relationships. Real-world problems with time dependencies require special preparation and transformation of data, which are, in many cases, critical for successful data mining. We will start with the simplest case—a single feature measured over time. This feature has a series of values over fixed time units. For example, a temperature reading could be measured every hour, or the sales of a product could be recorded every day. This is the classical univariate time-series problem, where it is expected that the value of the variable X at a given time can be related to previous values. Because the time series is measured at fixed units of time, the series of values can be expressed as

$$X = \{t(1), t(2), t(3), \ldots, t(n)\}$$

where t(n) is the most recent value.

For many time-series problems, the goal is to forecast t(n + 1) from previous values of the feature, where these values are directly related to the predicted value. One of the most important steps in the preprocessing of raw, time-dependent data is the specification of a window or a time lag. This is the number of previous values that influence

the prediction. Every window represents one sample of data for further analysis. For example, if the time series consists of the 11 measurements

$$X = \{t(0), t(1), t(2), t(3), t(4), t(5), t(6), t(7), t(8), t(9), t(10)\}$$

and if the window for analysis of the time-series is five, then it is possible to reorganize the input data into a tabular form with six samples, which is more convenient (standardized) for the application of data-mining techniques. Transformed data are given in Table 2.1.

The best time lag must be determined by the usual evaluation techniques for a varying complexity measure using independent test data. Instead of preparing the data once and turning them over to the data-mining programs for prediction, additional iterations of data preparation have to be performed. Although the typical goal is to predict the next value in time, in some applications, the goal can be modified to predict values in the future, several time units in advance. More formally, given the time-dependent values t(n − i), . . . , t(n), it is necessary to predict the value t(n + j). In the previous example, taking j = 3, the new samples are given in Table 2.2.

In general, the further in the future, the more difficult and less reliable is the forecast. The goal for a time series can easily be changed from predicting the next value in the time series to classification into one of predefined categories. From a data-

TABLE 2.1. Transformation of Time Series to Standard Tabular Form (Window = 5)

Sample	*WINDOW*					Next Value
	M1	M2	M3	M4	M5	
1	t(0)	t(1)	t(2)	t(3)	t(4)	t(5)
2	t(1)	t(2)	t(3)	t(4)	t(5)	t(6)
3	t(2)	t(3)	t(4)	t(5)	t(6)	t(7)
4	t(3)	t(4)	t(5)	t(6)	t(7)	t(8)
5	t(4)	t(5)	t(6)	t(7)	t(8)	t(9)
6	t(5)	t(6)	t(7)	t(8)	t(9)	t(10)

TABLE 2.2. Time-Series Samples in Standard Tabular Form (Window = 5) with Postponed Predictions (j = 3)

Sample	*WINDOW*					Netxt Value
	M1	M2	M3	M4	M5	
1	t(0)	t(1)	t(2)	t(3)	t(4)	t(7)
2	t(1)	t(2)	t(3)	t(4)	t(5)	t(8)
3	t(2)	t(3)	t(4)	t(5)	t(6)	t(9)
4	t(3)	t(4)	t(5)	t(6)	t(7)	t(10)

preparation perspective, there are no significant changes. For example, instead of predicted output value t(i + 1), the new classified output will be binary: *T* for t(i + 1) ≥ threshold value and *F* for t(i + 1) < threshold value.

The time units can be relatively small, enlarging the number of artificial features in a tabular representation of time series for the same time period. The resulting problem of high dimensionality is the price paid for precision in the standard representation of the time-series data.

In practice, many older values of a feature may be historical relics that are no longer relevant and should not be used for analysis. Therefore, for many business and social applications, new trends can make old data less reliable and less useful. This leads to a greater emphasis on recent data, possibly discarding the oldest portions of the time series. Now we are talking not only of a fixed window for the presentation of a time series but also on a fixed size for the data set. Only the n most recent cases are used for analysis, and, even then, they may not be given equal weight. These decisions must be given careful attention and are somewhat dependent on knowledge of the application and past experience. For example, using 20-year-old data about cancer patients will not give the correct picture about the chances of survival today.

Besides standard tabular representation of time series, sometimes it is necessary to additionally preprocess raw data and summarize their characteristics before application of data-mining techniques. Many times it is better to predict the difference t(n + 1) − t(n) instead of the absolute value t(n + 1) as the output. Also, using a ratio, t(n + 1)/t(n), which indicates the percentage of changes, can sometimes give better prediction results. These transformations of the predicted values of the output are particularly useful for logic-based data-mining methods such as decision trees or rules. When differences or ratios are used to specify the goal, features measuring differences or ratios for input features may also be advantageous.

Time-dependent cases are specified in terms of a goal and a time lag or a window of size m. One way of summarizing features in the data set is to average them, producing MA. A single average summarizes the most recent m feature values for each case, and for each increment in time, its value is

$$\mathrm{MA}(i, m) = 1/m \sum_{j=i-m+1}^{i} t(j)$$

Knowledge of the application can aid in specifying reasonable sizes for m. Error estimation should validate these choices. MA weight all time points equally in the average. Typical examples are MA in the stock market, such as 200 days MA for the DOW or NASDAQ. The objective is to smooth neighboring time points by an MA to reduce the random variation and noise components

$$\mathrm{MA}(i, m) = t(i) = \mathrm{mean}(i) + \mathrm{error}$$

Another type of average is an *exponential moving average* (EMA) that gives more weight to the most recent time periods. It is described recursively as

$$\mathrm{EMA}(i,\ m) = p^{*}\, t(i) + (1-p) * \mathrm{EMA}\ (i-1,\ m-1)$$
$$\mathrm{EMA}\ (i,1) = t(i)$$

where p is a value between 0 and 1. For example, if p = 0.5, the most recent value t(i) is equally weighted with the computation for all previous values in the window, where the computation begins with averaging the first two values in the series. The computation starts with the following two equations:

$$\mathrm{EMA}\ (i,2) = 0.5\, t(i) + 0.5\, t(i-1)$$
$$\mathrm{EMA}\ (i,3) = 0.5\, t(i) + 0.5\,[0.5\, t(i-1) + 0.5\, t(i-2)]$$

As usual, application knowledge or empirical validation determines the value of p. The exponential MA has performed very well for many business-related applications, usually producing results superior to the MA.

An MA summarizes the recent past, but spotting a change in the trend of the data may additionally improve forecasting performances. Characteristics of a trend can be measured by composing features that compare recent measurements with those of the more distant past. Three simple comparative features are

1. t(i) – MA(i, m), the difference between the current value and an MA,
2. MA(i, m) – MA(i – k, m), the difference between two MAs, usually of the same window size, and
3. t(i)/MA(i, m), the ratio between the current value and an MA, which may be preferable for some applications.

In general, the main components of the summarizing features for a time series are

1. current values,
2. smoothed values using MA, and
3. derived trends, differences, and ratios.

The immediate extension of a univariate time series is to a multivariate one. Instead of having a single measured value at time i, t(i), multiple measurements t[a(i), b(j)] are taken at the same time. There are no extra steps in data preparation for the multivariate time series. Each series can be transformed into features, and the values of the features at each distinct time A(i) are merged into a sample i. The resultant transformations yield a standard tabular form of data such as the table given in Figure 2.5.

While some data-mining problems are characterized by a single time series, hybrid applications are more frequent in real-world problems, having both time series and features that are not dependent on time. In these cases, standard procedures for time-dependent transformation and summarization of attributes are performed. High dimensions of data generated during these transformations can be reduced through the next phase of a data-mining process: data reduction.

Time	a	b
1	5	117
2	8	113
3	4	116
4	9	118
5	10	119
6	12	120

(a)

Sample	a(n−2)	a(n−1)	a(n)	b(n−2)	b(n−1)	b(n)
1	5	8	4	117	113	116
2	8	4	9	113	116	118
3	4	9	8	116	118	119
4	9	10	12	118	119	120

(b)

Figure 2.5. Tabulation of time-dependent features. (a) Initial time-dependent data; (b) samples prepared for data mining with time window = 3.

Some data sets do not include a time component explicitly, but the entire analysis is performed in the time domain (typically based on several dates that are attributes of described entities). One very important class of data belonging to this type is *survival data*. Survival data are data concerning how long it takes for a particular event to happen. In many medical applications, the event is the death of a patient, and therefore we analyze the patient's survival time. In industrial applications, the event is often the failure of a component in a machine. Thus, the output in these sorts of problems is the survival time. The inputs are the patient's records in medical applications and characteristics of the machine component in industrial applications. There are two main characteristics of survival data that make them different from the data in other data-mining problems. The first characteristic is called *censoring*. In many studies, the event has not happened by the end of the study period. So, for some patients in a medical trial, we may know that the patient was still alive after 5 years, but do not know when the patient died. This sort of observation would be called a censored observation. If the output is censored, we do not know the value of the output, but we do have some information about it. The second characteristic of survival data is that the input values are *time-dependent*. Since collecting data entails waiting until the event happens, it is possible for the inputs to change its value during the waiting period. If a patient stops smoking or starts with a new drug during the study, it is important to know what data to include into the study and how to represent these changes in time. Data-mining analysis for these types of problems concentrates on the survivor function or the hazard function. The survivor function is the probability of the survival time being greater than the time t. The hazard function indicates how likely a failure (of the industrial component) is at time t, given that a failure has not occurred before time t.

2.6 OUTLIER ANALYSIS

Very often in large data sets, there exist samples that do not comply with the general behavior of the data model. Such samples, which are significantly different or inconsistent with the remaining set of data, are called outliers. Outliers can be caused by measurement error, or they may be the result of inherent data variability. If, for example, the display of a person's age in the database is −1, the value is obviously not correct, and the error could have been caused by a default setting of the field "unrecorded age"

in the computer program. On the other hand, if in the database the number of children for one person is 25, this datum is unusual and has to be checked. The value could be a typographical error, or it could be correct and represent real variability for the given attribute.

Many data-mining algorithms try to minimize the influence of outliers on the final model, or to eliminate them in the preprocessing phases. Outlier detection methodologies have been used to detect and, where appropriate, remove anomalous observations from data. Outliers arise due to mechanical faults, changes in system behavior, fraudulent behavior, human error, instrument error, or simply through natural deviations in populations. Their detection can identify system faults and fraud before they escalate with potentially catastrophic consequences. The literature describes the field with various names, including outlier detection, novelty detection, anomaly detection, noise detection, deviation detection, or exception mining. Efficient detection of such outliers reduces the risk of making poor decisions based on erroneous data, and aids in identifying, preventing, and repairing the effects of malicious or faulty behavior. Additionally, many data-mining techniques may not work well in the presence of outliers. Outliers may introduce skewed distributions or complexity into models of the data, which may make it difficult, if not impossible, to fit an accurate model to the data in a computationally feasible manner.

The data-mining analyst has to be very careful in the automatic elimination of outliers because if the data are correct, that could result in the loss of important hidden information. Some data-mining applications are focused on outlier detection, and it is the essential result of a data analysis. The process consists of two main steps: (1) Build a profile of the "normal" behavior, and (2) use the "normal" profile to detect outliers. The profile can be patterns or summary statistics for the overall population. The assumption is that there are considerably more "normal" observations than "abnormal"—outliers/anomalies in the data. For example, when detecting fraudulent credit card transactions at a bank, the outliers are typical examples that may indicate fraudulent activity, and the entire data-mining process is concentrated on their detection. But, in many of the other data-mining applications, especially if they are supported with large data sets, outliers are not very useful, and they are more the result of errors in data collection than a characteristic of a data set.

Outlier detection and potential removal from a data set can be described as a process of the selection of k out of n samples that are considerably dissimilar, exceptional, or inconsistent with respect to the remaining data ($k << n$). The problem of defining outliers is nontrivial, especially in multidimensional samples. The main types of outlier detection schemes are

- graphical or visualization techniques,
- statistical-based techniques,
- distance-based techniques, and
- model-based techniques.

Examples of visualization methods include Boxplot (1-D), Scatter plot (2-D), and Spin plot (3-D), and they will be explained in the following chapters. Data visualization

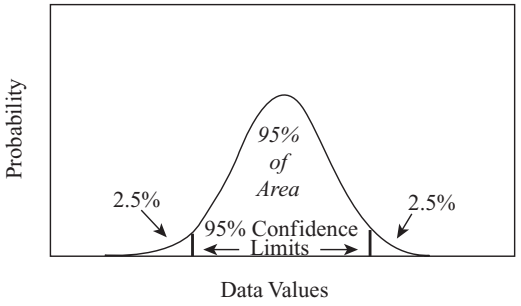

Figure 2.6. Outliers for univariate data based on mean value and standard deviation.

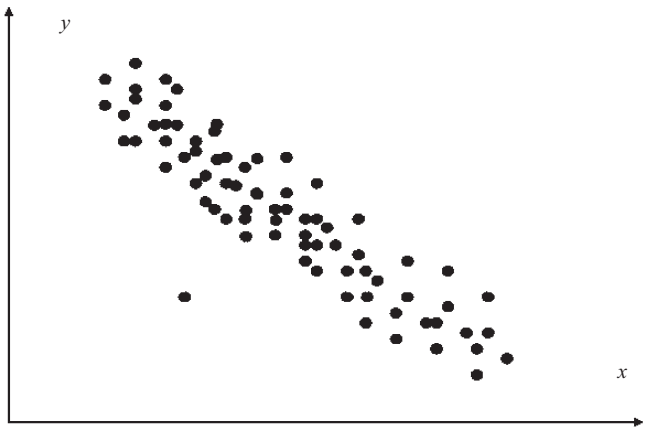

Figure 2.7. Two-dimensional data set with one outlying sample.

methods that are useful in outlier detection for one to three dimensions are weaker in multidimensional data because of a lack of adequate visualization methodologies for n-dimensional spaces. An illustrative example of a visualization of 2-D samples and visual detection of outliers is given in Figures 2.6 and 2.7. The main limitations of the approach are its time-consuming process and the subjective nature of outlier detection.

Statistically based outlier detection methods can be divided between *univariate methods*, proposed in earlier works in this field, and *multivariate methods*, which usually form most of the current body of research. Statistical methods either assume a known underlying distribution of the observations or, at least, they are based on statistical estimates of unknown distribution parameters. These methods flag as outliers those observations that deviate from the model assumptions. The approach is often unsuitable for high-dimensional data sets and for arbitrary data sets without prior knowledge of the underlying data distribution.

Most of the earliest univariate methods for outlier detection rely on the assumption of an underlying known distribution of the data, which is assumed to be identically and independently distributed. Moreover, many discordance tests for detecting univariate

outliers further assume that the distribution parameters and the type of expected outliers are also known. Although traditionally the normal distribution has been used as the target distribution, this definition can be easily extended to any unimodal symmetric distribution with positive density function. Traditionally, the sample mean and the sample variance give good estimation for data location and data shape if it is not contaminated by outliers. When the database is contaminated, those parameters may deviate and significantly affect the outlier-detection performance. Needless to say, in real-world data-mining applications, these assumptions are often violated.

The simplest approach to outlier detection for 1-D samples is based on traditional unimodal statistics. Assuming that the distribution of values is given, it is necessary to find basic statistical parameters such as mean value and variance. Based on these values and the expected (or predicted) number of outliers, it is possible to establish the threshold value as a function of variance. All samples out of the threshold value are candidates for outliers as presented in Figure 2.6. The main problem with this simple methodology is an a priori assumption about data distribution. In most real-world examples, the data distribution may not be known.

For example, if the given data set represents the feature age with 20 different values:

$$\text{Age} = \{3, 56, 23, 39, 156, 52, 41, 22, 9, 28, 139, 31, 55, 20, -67, 37, 11, 55, 45, 37\}$$

then, the corresponding statistical parameters are

$$\text{Mean} = 39.9$$

$$\text{Standard deviation} = 45.65$$

If we select the threshold value for normal distribution of data as

$$\text{Threshold} = \text{Mean} \pm 2 \times \text{Standard deviation}$$

then, all data that are out of range [−54.1, 131.2] will be potential outliers. Additional knowledge of the characteristics of the feature (age is always greater then 0) may further reduce the range to [0, 131.2]. In our example there are three values that are outliers based on the given criteria: 156, 139, and −67. With a high probability we can conclude that all three of them are typo errors (data entered with additional digits or an additional "–" sign).

An additional single-dimensional method is Grubbs' method (Extreme Studentized Deviate), which calculates a Z value as the difference between the mean value for the attribute and the analyzed value divided by the standard deviation for the attribute. The Z value is compared with a 1% or 5% significance level showing an outlier if the Z parameter is above the threshold value.

In many cases multivariable observations cannot be detected as outliers when each variable is considered independently. Outlier detection is possible only when multivariate analysis is performed, and the interactions among different variables are compared

within the class of data. An illustrative example is given in Figure 2.7 where separate analysis of each dimension will not give any outlier, but analysis of 2-D samples (x,y) gives one outlier detectable even through visual inspection.

Statistical methods for multivariate outlier detection often indicate those samples that are located relatively far from the center of the data distribution. Several distance measures can be implemented for such a task. The Mahalanobis distance measure includes the inter-attribute dependencies so the system can compare attribute combinations. It is a well-known approach that depends on estimated parameters of the multivariate distribution. Given n observations x_i from a p-dimensional data set (often $n >> p$), denote the sample mean vector by $\overline{x}_n$ and the sample covariance matrix by V_n, where

$$V_n = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \overline{x}_n)(x_i - \overline{x}_n)^T$$

The *Mahalanobis* distance for each multivariate data point i ($i = 1, \ldots, n$) is denoted by M_i and given by

$$M_i = \left(\sum_{i=1}^{n}(x_i - \overline{x}_n)^T V_n^{-1}(x_i - \overline{x}_n)\right)^{1/2}$$

Accordingly, those n-dimensional samples with a large *Mahalanobis* distance are indicated as outliers. Many statistical methods require data-specific parameters representing a priori data knowledge. Such information is often not available or is expensive to compute. Also, most real-world data sets simply do not follow one specific distribution model.

Distance-based techniques are simple to implement and make no prior assumptions about the data distribution model. However, they suffer exponential computational growth as they are founded on the calculation of the distances between all samples. The computational complexity is dependent on both the dimensionality of the data set m and the number of samples n and usually is expressed as $O(n^2m)$. Hence, it is not an adequate approach to use with very large data sets. Moreover, this definition can lead to problems when the data set has both dense and sparse regions. For example, as the dimensionality increases, the data points are spread through a larger volume and become less dense. This makes the convex hull harder to discern and is known as the "curse of dimensionality."

Distance-based outlier detection method, presented in this section, eliminates some of the limitations imposed by the statistical approach. The most important difference is that this method is applicable to multidimensional samples while most of statistical descriptors analyze only a single dimension, or several dimensions, but separately. The basic computational complexity of this method is the evaluation of distance measures between all samples in an n-dimensional data set. Then, a sample s_i in a data set S is an outlier if at least a fraction p of the samples in S lies at a distance greater than d. In other words, distance-based outliers are those samples that do not have enough

neighbors, where neighbors are defined through the multidimensional distance between samples. Obviously, the criterion for outlier detection is based on two parameters, p and d, which may be given in advance using knowledge about the data, or which may be changed during the iterations (trial-and-error approach) to select the most representative outliers.

To illustrate the approach we can analyze a set of 2-D samples S, where the requirements for outliers are the values of thresholds: $p \geq 4$ and $d > 3$.

$$S = \{s_1, s_2, s_3, s_4, s_5, s_6, s_7\} = \{(2,4), (3,2), (1,1), (4,3), (1,6), (5,3), (4,2)\}$$

The table of Euclidian distances, $d = [(x1 - x2)^2 + [y1 - y2]^2]^{1/2}$, for the set S is given in Table 2.3 and, based on this table, we can calculate a value for the parameter p with the given threshold distance (d = 3) for each sample. The results are represented in Table 2.4.

Using the results of the applied procedure and given threshold values, it is possible to select samples s_3 and s_5 as outliers (because their values for p is above the threshold value: p = 4). The same results could be obtained by visual inspection of a data set, represented in Figure 2.8. Of course, the given data set is very small and a 2-D graphical representation is possible and useful. For n-dimensional, real- world data analyses the visualization process is much more difficult, and analytical approaches in outlier detection are often more practical and reliable.

TABLE 2.3. Table of Distances for Data Set S

	s_1	s_2	s_3	s_4	s_5	s_6	s_7
s_1		2.236	3.162	2.236	2.236	3.162	2.828
s_2			2.236	1.414	4.472	2.236	1.000
s_3				3.605	5.000	4.472	3.162
s_4					4.242	1.000	1.000
s_5						5.000	5.000
s_6							1.414

TABLE 2.4. The Number of Points p with the Distance Greater Than d for Each Given Point in S

Sample	p
s_1	2
s_2	1
s_3	5
s_4	2
s_5	5
s_6	3
s_7	2

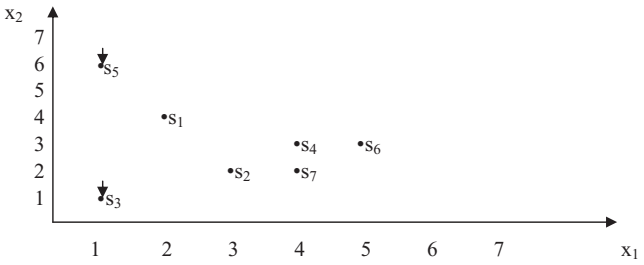

Figure 2.8. Visualization of two-dimensional data set for outlier detection.

There is a possibility for reducing complexity of the algorithm by partitioning the data into n-dimensonal cells. If any cell and its directly adjacent neighbors contain more than k points, then the points in the cell are deemed to lie in a dense area of the distribution so the points contained are unlikely to be outliers. If the number of points is less than k, then all points in the cell are potential outliers. Hence, only a small number of cells need to be processed and only a relatively small number of distances need to be calculated for outlier detection.

Model-based techniques are the third class of outlier-detection methods. These techniques simulate the way in which humans can distinguish unusual samples from a set of other similar samples. These methods define the basic characteristics of the sample set, and all samples that deviate from these characteristics are outliers. The sequential-exception technique is one possible approach that is based on a dissimilarity function. For a given set of n samples, a possible dissimilarity function is the total variance of the sample set. Now, the task is to define the smallest subset of samples whose removal results in the greatest reduction of the dissimilarity function for the residual set. The general task of finding outliers using this method can be very complex (combinational explosion of different selections of the set of potential outliers—the so called exception set), and it can be theoretically defined as an NP-hard problem (i.e., intractable). If we settle for a less-than-optimal answer, the algorithm's complexity can be reduced to the linear level, using a sequential approach. Using the greedy method, the algorithm reduces the size sequentially, sample by sample (or subset by subset), by selecting at each step the one that causes the greatest decrease in the total variance.

Many data-mining algorithms are robust and as such tolerant to outliers but were specifically optimized for clustering or classification in large data sets. It includes clustering algorithms such as Balanced and Iterative Reducing and Clustering Using Hierarchies (BIRCH) and Density-Based Spatial Clustering of Applications with Noise (DBSCAN), k nearest neighbor (kNN) classification algorithms, and different neural networks. These methodologies are explained with more details later in the book, but the reader has to be aware about applicability of these techniques as powerful tools for outliers' detection. For example, in the data set represented in Figure 2.9, clustering-based methods consider a cluster of small sizes, including the size of one sample, as clustered outliers. Note that since their main objective is clustering, these methods are

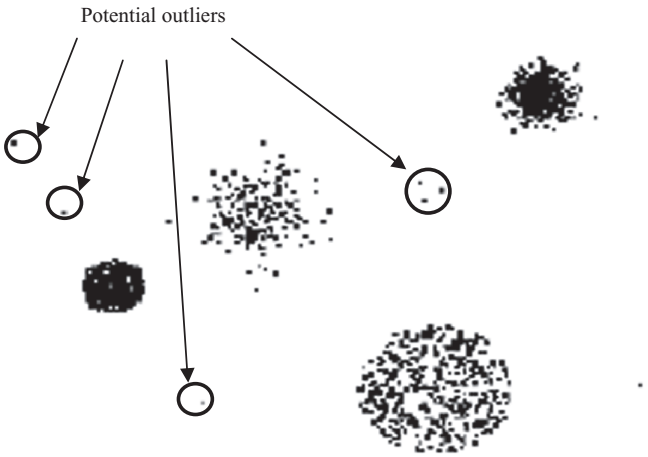

Figure 2.9. Determining outliers through clustering.

not always optimized for outlier detection. In most cases, the outlier detection criteria are implicit and cannot easily be inferred from the clustering procedures.

Most of outlier detection techniques have only focused on continuous real-valued data attributes, and there has been little focus on categorical data. Most approaches require cardinal or at the least ordinal data to allow vector distances to be calculated, and have no mechanism for processing categorical data with no implicit ordering.

2.7 REVIEW QUESTIONS AND PROBLEMS

1. Generate the tree structure of data types explained in Section 2.1.
2. If one attribute in the data set is student grade with values A, B, C, D, and F, what type are these attribute values? Give a recommendation for preprocessing of the given attribute.
3. Explain why "the curse of dimensionality" principles are especially important in understanding large data sets.
4. Every attribute in a 6-D sample is described with one out of three numeric values {0, 0.5, 1}. If there exist samples for all possible combinations of attribute values, what will be the number of samples in a data set and what will be the expected distance between points in a 6-D space?
5. Derive the formula for min–max normalization of data on [−1, 1] interval.
6. Given 1-D data set X = {−5.0, 23.0, 17.6, 7.23, 1.11}, normalize the data set using
   (a) decimal scaling on interval [−1, 1],
   (b) min–max normalization on interval [0, 1],

(c) min–max normalization on interval [−1, 1], and
(d) standard deviation normalization.

Compare the results of previous normalizations and discuss the advantages and disadvantages of the different techniques.

7. Perform data smoothing using a simple rounding technique for a data set

$$Y = \{1.17, 2.59, 3.38, 4.23, 2.67, 1.73, 2.53, 3.28, 3.44\}$$

and present the new data set when the rounding is performed to the precision of
(a) 0.1 and
(b) 1.

8. Given a set of 4-D samples with missing values,

$$X1 = \{0, 1, 1, 2\}$$
$$X2 = \{2, 1, -, 1\}$$
$$X3 = \{1, -, -, 0\}$$
$$X4 = \{-, 2, 1, -\}$$

if the domains for all attributes are [0, 1, 2], what will be the number of "artificial" samples if missing values are interpreted as "don't care values" and they are replaced with all possible values for a given domain?

9. A 24-h, time-dependent data set X is collected as a training data set to predict values 3 h in advance. If the data set X is

$$X = \{7, 8, 9, 10, 9, 8, 7, 9, 11, 13, 15, 17, 16, 15, 14, 13, 12, 11, 10, 9, 7, 5, 3, 1),$$

(a) What will be a standard tabular representation of data set X if
  (i) the window width is 6, and a prediction variable is based on the difference between the current value and the value after 3 h. What is the number of samples?
  (ii) the window 4width is 12, and the prediction variable is based on ratio. What is the number of samples?
(b) Plot discrete X values together with computed 6- and 12-h MA.
(c) Plot time-dependent variable X and its 4-h EMA.

10. The number of children for different patients in a database is given with a vector

$$C = \{3, 1, 0, 2, 7, 3, 6, 4, -2, 0, 0, 10, 15, 6\}.$$

Find the outliers in set C using standard statistical parameters mean and variance.

If the threshold value is changed from ±3 standard deviations to ±2 standard deviations, what additional outliers are found?

11. For a given data set X of 3-D samples,

$$X = [\{1, 2, 0\}, \{3, 1, 4\}, \{2, 1, 5\}, \{0, 1, 6\}, \{2, 4, 3\},$$
$$\{4, 4, 2\}, \{5, 2, 1\}, \{7, 7, 7\}, \{0, 0, 0\}, \{3, 3, 3\}].$$

(a) find the outliers using the distance-based technique if
   (i) the threshold distance is 4, and threshold fraction p for non-neighbor samples is 3, and
   (ii) the threshold distance is 6, and threshold fraction p for non-neighbor samples is 2.

(b) Describe the procedure and interpret the results of outlier detection based on mean values and variances for each dimension separately.

12. Discuss the applications in which you would prefer to use EMA instead of MA.

13. If your data set contains missing values, discuss the basic analyses and corresponding decisions you will take in the preprocessing phase of the data-mining process.

14. Develop a software tool for the detection of outliers if the data for preprocessing are given in the form of a flat file with n-dimensional samples.

15. The set of seven 2-D samples is given in the following table. Check if we have outliers in the data set. Explain and discuss your answer.

Sample #	X	Y
1	1	3
2	7	1
3	2	4
4	6	3
5	4	2
6	2	2
7	7	2

16. Given the data set of 10 3-D samples: {(1,2,0), (3,1,4), (2,1,5), (0,1,6), (2,4,3), (4,4,2), (5,2,1), (7,7,7), (0,0,0), (3,3,3)}, is the sample S4 = (0,1,6) outlier if the threshold values for the distance d = 6, and for the number of samples in the neighborhood p > 2? (Note: Use distance-based outlier-detection technique.)

17. What is the difference between *nominal* and *ordinal* data? Give examples.

18. Using the method of *distance-based outliers detection* find the outliers in the set

$$X = \{(0, 0), (1, 1), (3, 2), (6, 3), (5, 4), (2, 4)\}$$

if the criterion is that at least the fraction $p \geq 3$ of the samples in X lies at a distance d greater than 4.

19. What will be normalized values (using *min-max normalization* of data for the range [−1, 1]) for the data set X?

$$X = \{-5, 11, 26, 57, 61, 75\}$$

20. Every attribute in 6-D samples is described with one out of three numerical values: {0, 0.5, 1}. If there exist samples for all possible combinations of attribute values
(a) What will be the number of samples in a data set, and
(b) What will be the expected distance between points in a 6-D space?

21. Classify the following attributes as *binary, discrete, or continuous*. Also, classify them as *qualitative* (*nominal or ordinal*) or *quantitative* (*interval or ratio*). Some cases may have more than one interpretation, so briefly indicate your reasoning (e.g., age in years; answer: discrete, quantitative, ratio).
(a) Time in terms of AM or PM.
(b) Brightness as measured by a light meter.
(c) Brightness as measured by people's judgment.
(d) Angles as measured in degrees between 0 and 360.
(e) Bronze, Silver, and Gold medals at the Olympics.
(f) Height above sea level.
(g) Number of patients in a hospital.
(h) ISBN numbers for books.
(i) Ability to pass light in terms of the following values: opaque, translucent, transparent.
(j) Military rank.
(k) Distance from the center of campus.
(l) Density of a substance in grams per cubic centimeter.
(m) Coats check number when you attend the event.

2.8 REFERENCES FOR FURTHER STUDY

Bischoff, J., T. Alexander, *Data Warehouse: Practical Advice from the Experts*, Prentice Hall, Upper Saddle River, NJ, 1997.

The objective of a data warehouse is to provide any data, anywhere, anytime in a timely manner at a reasonable cost. Different techniques used to preprocess the data in warehouses reduce the effort in initial phases of data mining.

Cios, K.J., W. Pedrycz, R. W. Swiniarski, L. A. Kurgan, *Data Mining: A Knowledge Discovery Approach*, Springer, New York, 2007.

This comprehensive textbook on data mining details the unique steps of the knowledge discovery process that prescribe the sequence in which data-mining projects should be performed. *Data Mining* offers an authoritative treatment of all development phases from problem and data understanding through data preprocessing to deployment of the results. This knowledge-discovery approach is what distinguishes this book from other texts in the area. It concentrates on data preparation, clustering and association-rule learning (required for processing unsupervised data), decision trees, rule induction algorithms, neural networks, and many other data-mining methods, focusing predominantly on those which have proven successful in data-mining projects.

Hand, D., H. Mannila, P. Smith, *Principles of Data Mining*, MIT Press, Cambridge, MA, 2001.

The book consists of three sections. The first, foundations, provides a tutorial overview of the principles underlying data-mining algorithms and their applications. The second section,

data-mining algorithms, shows how algorithms are constructed to solve specific problems in a principled manner. The third section shows how all of the preceding analyses fit together when applied to real-world data-mining problems.

Hodge, J. V., J. Austin, A Survey of Outlier Detection Methodologies, *Artificial Intelligence Review*, Vol. 22, No. 2, October 2004, pp. 85–127.

The original outlier-detection methods were arbitrary but now, principled and systematic techniques are used, drawn from the full gamut of Computer Science and Statistics. In this paper, a survey of contemporary techniques for outlier detection is introduced. The authors identify respective motivations and distinguish advantages and disadvantages of these techniques in a comparative review.

Ben-Gal, I., Outlier Detection, in *Data Mining and Knowledge Discovery Handbook: A Complete Guide for Practitioners and Researchers*, O. Maimon and L. Rockach eds., Kluwer Academic Publishers, Boston, 2005.

The author presents several methods for outlier detection, while distinguishing between univariate versus multivariate techniques and parametric versus nonparametric procedures. In the presence of outliers, special attention should be taken to assure the robustness of the used estimators. Outlier detection for data mining is often based on distance measures, clustering, and spatial methods.

Kennedy, R. L. et al., *Solving Data Mining Problems through Pattern Recognition*, Prentice Hall, Upper Saddle River, NJ, 1998.

The book takes a practical approach to overall data-mining project development. The rigorous, multistep methodology includes defining the data set; collecting, preparing, and preprocessing data; choosing the appropriate technique and tuning the parameters; and training, testing, and troubleshooting.

Weiss, S. M., N. Indurkhya, *Predictive Data Mining: A Practical Guide*, Morgan Kaufman Publishers, San Francisco, CA, 1998.

This book focuses on the data-preprocessing phase in successful data-mining applications. Preparation and organization of data and development of an overall strategy for data mining are not only time-consuming processes, but also fundamental requirements in real-world data mining. The simple presentation of topics with a large number of examples is an additional strength of the book.