DATA MINING

Concepts, Models, Methods, and Algorithms

MEHMED KANTARDZIC

DATA MINING

DATA MINING

Concepts, Models, Methods, and Algorithms

THIRD EDITION

Mehmed Kantardzic

IEEE PRESS

WILEY

To Belma and Nermin

CONTENTS

PREFACE

Since the second edition of the book published in 2011, there were a lot of advances in the data-mining field. The term Big Data is introduced and widely accepted to describe the amount and the rate at which massive and diverse data has been collected, analyzed, and used. New field of data science is established to describe all multidisciplinary aspects of advanced tools and methodologies, enabling to extract useful and actionable insight from Big Data. The third edition of the book summarizes these new developments in fast-changing data-mining field, as well as presents state-of-the-art data-mining principles required for systematic approach in both an academic environment and advanced applications deployment.

While the core of materials in the third edition of the textbook remains the same, most important changes and additions in this edition highlight the dynamics of the field and include:

- new topics such as Big Data, data science, and deep learning,
- new methodologies including reinforcement learning, cloud computing, and MapReduce framework,
- new highlights given on unbalanced data, fairness of data-mining models, and subjectivity in a clustering validation,
- additional advanced algorithms such as convolutional neural networks (CNN), semisupervised support vector machines (S3VM), Q learning, random forest, and SMOTE algorithm for unbalanced data modeling, and
- additional examples and exercises that have been added to each chapter, as well as bibliography, references for further reading, and appendices that have been updated.

I would like to thank current and former students in our Data Mining Lab at the Computer Engineering and Computer Science Department, University of Louisville, for their contributions in preparation of this third edition. Tegjyot Singh Sethi and Elaheh Arabmakki helped with comments and suggestions based on their TA experiences using the previous editions of the textbook for our data-mining classes. Lingyu

Lyu and Mehmet Akif Gulum helped me with proofreading of the new edition and numerous corrections and updates in appendices of the book. Special thanks to Hanqing Hu who helped me in a preparation of the final version of the text and all additional figures and tables in the third edition. The new edition of the book is a result of previous editions' use as a textbook in active teaching by a large number of my colleagues. They helped me with their experiences and recommendations, and I would like to thank them for their support and encouragements during the preparation of the third edition.

I expect that with this new edition of the book, the reader will increase understanding of modern data-mining technologies and their applications and will identify the recent challenges in the field. The book should serve as the guide in the data-mining field for advanced undergraduate or graduate students, young researchers, and practitioners. While each chapter roughly follows a standard educational template, earlier chapters in the book take more emphasis to introduce fundamental concepts, while later chapters build upon these foundations and gradually introduce the most important techniques and methodologies for data mining. The book provides the fundamental building blocks that will enable the reader to become part of data science community and participate in building killer data-mining applications of tomorrow.

MEHMED KANTARDZIC
Louisville

PREFACE TO THE SECOND EDITION

In the 7 years that have passed since the publication of the first edition of this book, data-mining field has made a good progress both in developing new methodologies and in extending the spectrum of new applications. These changes in data mining motivated me to update my data-mining book with a second edition. Although the core of material in this edition remains the same, the new version of the book attempts to summarize recent developments in our fast-changing field, presenting the state of the art in data mining, both in academic research and in deployment in commercial applications. Most notable amount of changes from the first edition is the addition of:

- new topics such as ensemble learning, graph mining, temporal, spatial, distributed, and privacy preserving data mining,
- new algorithms such as CART, DBSCAN, BIRCH, PageRank, AdaBoost, support vector machines (SVM), Kohonen self-organizing maps (SOM), and latent semantic indexing (LSI),
- more details on practical aspects and business understanding of a data-mining process, discussing important problems of validation, deployment, data understanding, causality, security and privacy, and
- some quantitative measures and methods for comparison of data-mining models such as ROC curve, lift chart, ROI chart, McNemar's test, and K-fold cross-validation paired t-test.

Keeping in mind the educational side of the book, many new exercises have been added. The bibliography and appendices have been updated to include work that has been appeared in the last few years, as well as to reflect the change of emphasis when new topic gained importance.

I would like to give thanks to all my colleagues all over the world who used the first edition of the book for their classes and sent me support, encouragement, and suggestions to put together this revised version. My sincere thanks to all my colleagues and students in the Data Mining Lab and Computer Science Department for their reviews of this edition and numerous helpful suggestions. Special thanks to the graduate students Brent Wenerstrom, Chamila Walgampaya, and Wael Emara for patience

in proofreading through this new edition and useful discussions about the content of new chapters, numerous corrections, and additions. Dr. Joung Woo Ryu helped me enormously in a preparation of the final version of the text and all additional figures and tables, and I would like to express my deepest gratitude.

I believe this book can serve as a valuable guide to the field for undergraduate, graduate students, researchers, and practitioners. I hope that the wide range of topics covered will allow readers to appreciate the extent of the impact of data mining on modern business, science, even the entire society.

MEHMED KANTARDZIC
Louisville
July 2010

PREFACE TO THE FIRST EDITION

The modern technologies of computers, networks, and sensors have made data collection and organization an almost effortless task. However, the captured data needs to be converted into information and knowledge from recorded data to become useful. Traditionally, the task of extracting useful information from recorded data has been performed by analysts; however, the increasing volume of data in modern businesses and sciences calls for computer-based methods for this task. As data sets have grown in size and complexity, so there had been an inevitable shift away from direct hands-on data analysis toward indirect, automatic data analysis in which the analyst works via more complex and sophisticated tools. The entire process of applying computer-based methodology, including new techniques for knowledge discovery from data, is often called data mining.

The importance of data mining arises from the fact that the modern world is a data-driven world. We are surrounded by data, numerical and otherwise, which must be analyzed and processed to convert it into information that informs, instructs, answers, or otherwise aids understanding and decision-making. In the age of Internet, intranets, data warehouses, and data marts, the fundamental paradigms of classical data analysis are ripe for changes. Very large collections of data—millions or even hundreds of millions of individual records—are now being stored into centralized data warehouses, allowing analysts to make use of powerful data-mining methods to examine data more comprehensively. The quantity of such data is huge and growing, the number of sources is effectively unlimited, and the range of areas covered is vast: industrial, commercial, financial, and scientific activities are all generating such data.

The new discipline of data mining has developed especially to extract valuable information from such huge data sets. In recent years there has been an explosive growth of methods for discovering new knowledge from raw data. This is not surprising given the proliferation of low-cost computers (for implementing such methods in software), low-cost sensors, communications, and database technology (for collecting and storing data) and highly computer-literate application experts who can pose "interesting" and "useful" application problems.

Data-mining technology is currently a hot favorite in the hands of decision-makers as it can provide valuable hidden business and scientific "intelligence" from large amount of historical data. It should be remembered, however, that fundamentally

data mining is not a new technology. The concept of extracting information and knowledge discovery from recorded data is a well-established concept in scientific and medical studies. What is new is the convergence of several disciplines and corresponding technologies that have created a unique opportunity for data mining in scientific and corporate world.

The origin of this book was a wish to have a single introductory source in which we could direct students, rather than having to direct them to multiple sources. However, it soon becomes apparent that wide interest existed and the potential readers other than our students would appreciate a compilation of some of the most important methods, tools, and algorithms in data mining. Such readers include people from a wide variety of backgrounds and positions, who find themselves confronted by the need to make sense of large amount of raw data. This book can be used by a wide range of readers, from students wishing to learn about basic processes and techniques in data mining to analysts and programmers who will be engaged directly in interdisciplinary teams for selected data-mining applications. This book reviews state-of-the-art techniques for analyzing enormous quantities of raw data in a high-dimensional data spaces to extract new information useful in decision-making process. Most of the definitions, classifications, and explanations of the techniques, covered in this book, are not new, and they are already presented in references at the end of the book. One of the author's main goals was to concentrate on systematic and balanced approach to all phases of a data-mining process and present them with enough illustrative examples. We expect that carefully prepared examples should give the reader additional arguments and guidelines in a selection and structuring of techniques and tools for its own data-mining applications. Better understanding of implementational details for most of the introduced techniques challenges the reader to build its own tools or to improve applied methods and techniques.

Teaching in data mining has to have emphasis on the concepts, and properties of the applied methods, rather than on the mechanical details of how to apply different data-mining tools. Despite all of their attractive bells and whistles, the computer-based tools alone will never provide the entire solution. There will always be the need for the practitioner to make important decisions regarding how the whole process will be designed and how and what tools will be employed. Obtaining a deeper understanding of the methods and models, how they behave and why they behave the way they do, is a prerequisite for efficient and successful application of data-mining technology. The premise of this book is that there are just a handful of important principles and issues in the field of data mining. Any researcher or practitioner in this field needs to be aware of these issues in order to successfully apply a particular methodology, understand a method's limitations, or develop new techniques. This book is an attempt to present and discuss such issues and principles and then describe representative and popular methods originating from statistics, machine learning, computer graphics, databases, information retrieval, neural networks, fuzzy logic, and evolutionary computation.

In this book, we describe how best to prepare environments for performing data mining and discuss approaches that have proven to be critical in revealing important

patterns, trends, and models in large data sets. It is our expectation that once a reader has completed this text, he/she will be able to initiate and perform basic activities in all phases of a data-mining process successfully and effectively. Although it is easy to focus on the technologies, as you read through the book, have in mind that technology alone does not provide the entire solution. One of our goals in writing this book was to minimize the hype associated with data mining. Rather than making false promises that overstep the bounds of what can reasonably be expected from data mining, we have tried to make a more objective approach. We describe with enough information the processes and algorithms that are necessary to produce reliable and useful results in data-mining applications. We do not advocate the use of any particular product or technique over another; the designer of data-mining process has to have enough background for selection of appropriate methodologies and software tools.

MEHMED KANTARDZIC
Louisville
August 2002

1

DATA-MINING CONCEPTS

Chapter Objectives

- Understand the need for analyses of large, complex, information-rich data sets.
- Identify the goals and primary tasks of the data-mining process.
- Describe the roots of data-mining technology.
- Recognize the iterative character of a data-mining process and specify its basic steps.
- Explain the influence of data quality on a data-mining process.
- Establish the relation between data warehousing and data mining.
- Discuss concepts of big data and data science.

1.1   INTRODUCTION

Modern science and engineering are based on using *first-principle models* to describe physical, biological, and social systems. Such an approach starts with a basic scientific model, such as Newton's laws of motion or Maxwell's equations in electromagnetism, and then builds upon them various applications in mechanical engineering or electrical engineering. In this approach, experimental data are used to verify the underlying first-principle models and to estimate some of the parameters that are difficult or sometimes impossible to measure directly. However, in many domains the underlying first principles are unknown, or the systems under study are too complex to be mathematically formalized. With the growing use of computers, there is a great amount of data being generated by such systems. In the absence of first-principle models, such readily available data can be used to derive models by estimating useful relationships between a system's variables (i.e., unknown input–output dependencies). *Thus there is currently a paradigm shift from classical modeling and analyses based on first principles to developing models and the corresponding analyses directly from data.*

We have grown accustomed gradually to the fact that there are tremendous volumes of data filling our computers, networks, and lives. Government agencies, scientific institutions, and businesses have all dedicated enormous resources to collecting and storing data. In reality, only a small amount of these data will ever be used because, in many cases, the volumes are simply too large to manage or the data structures themselves are too complicated to be analyzed effectively. How could this happen? The primary reason is that the original effort to create a data set is often focused on issues such as storage efficiency; it does not include a plan for how the data will eventually be used and analyzed.

The need to understand large, complex, information-rich data sets is common to virtually all fields of business, science, and engineering. In the business world, corporate and customer data are becoming recognized as a strategic asset. The ability to extract useful knowledge hidden in these data and to act on that knowledge is becoming increasingly important in today's competitive world. The entire process of applying a computer-based methodology, including new techniques, for discovering knowledge from data is called data mining.

Data mining is an iterative process within which progress is defined by discovery, through either automatic or manual methods. Data mining is most useful in an exploratory analysis scenario in which there are no predetermined notions about what will constitute an "interesting" outcome. Data mining is the search for new, valuable, and nontrivial information in large volumes of data. It is a cooperative effort of humans and computers. Best results are achieved by balancing the knowledge of human experts in describing problems and goals with the search capabilities of computers.

In practice, the two primary goals of data mining tend to be *prediction* and *description*. *Prediction* involves using some variables or fields in the data set to predict unknown or future values of other variables of interest. *Description*, on the other

hand, focuses on finding patterns describing the data that can be interpreted by humans. Therefore, it is possible to put data-mining activities into one of two categories:

1. Predictive data mining, which *produces the model* of the system described by the given data set, or
2. Descriptive data mining, which *produces new, nontrivial information* based on the available data set.

On the predictive end of the spectrum, the goal of data mining is to produce a model, expressed as an executable code, which can be used to perform classification, prediction, estimation, or other similar tasks. On the other, descriptive end of the spectrum, the goal is to gain an understanding of the analyzed system by uncovering patterns and relationships in large data sets. The relative importance of prediction and description for particular data-mining applications can vary considerably. The goals of prediction and description are achieved by using data-mining techniques, explained later in this book, for the following *primary data-mining tasks:*

1. *Classification*—Discovery of a predictive learning function that classifies a data item into one of several predefined classes.
2. *Regression*—Discovery of a predictive learning function, which maps a data item to a real-value prediction variable.
3. *Clustering*—A common descriptive task in which one seeks to identify a finite set of categories or clusters to describe the data.
4. *Summarization*—An additional descriptive task that involves methods for finding a compact description for a set (or subset) of data.
5. *Dependency modeling*—Finding a local model that describes significant dependencies between variables or between the values of a feature in a data set or in a part of a data set.
6. *Change and deviation detection*—Discovering the most significant changes in the data set.

The more formal approach, with graphical interpretation of data-mining tasks for complex and large data sets and illustrative examples, is given in Chapter 4. Current introductory classifications and definitions are given here only to give the reader a feeling of the wide spectrum of problems and tasks that may be solved using data-mining technology.

The success of a data-mining engagement depends largely on the amount of energy, knowledge, and creativity that the designer puts into it. In essence, data mining is like solving a puzzle. The individual pieces of the puzzle are not complex structures in and of themselves. Taken as a collective whole, however, they can constitute very elaborate systems. As you try to unravel these systems, you will probably get frustrated, start forcing parts together, and generally become annoyed at the entire process; but once you know how to work with the pieces, you realize that it was

not really that hard in the first place. The same analogy can be applied to data mining. In the beginning, the designers of the data-mining process probably do not know much about the data sources; if they did, they would most likely not be interested in performing data mining. Individually, the data seem simple, complete, and explainable. But collectively, they take on a whole new appearance that is intimidating and difficult to comprehend, like the puzzle. Therefore, being an analyst and designer in a data-mining process requires, besides thorough professional knowledge, creative thinking and a willingness to see problems in a different light.

Data mining is one of the fastest growing fields in the computer industry. Once a small interest area within computer science and statistics, it has quickly expanded into a field of its own. One of the greatest strengths of data mining is reflected in its wide range of methodologies and techniques that can be applied to a host of problem sets. Since data mining is a natural activity to be performed on large data sets, one of the largest target markets is the entire data-warehousing, data-mart, and decision-support community, encompassing professionals from such industries as retail, manufacturing, telecommunications, healthcare, insurance, and transportation. In the business community, data mining can be used to discover new purchasing trends, plan investment strategies, and detect unauthorized expenditures in the accounting system. It can improve marketing campaigns, and the outcomes can be used to provide customers with more focused support and attention. Data-mining techniques can be applied to problems of business process reengineering, in which the goal is to understand interactions and relationships among business practices and organizations.

Many law enforcement and special investigative units, whose mission is to identify fraudulent activities and discover crime trends, have also used data mining successfully. For example, these methodologies can aid analysts in the identification of critical behavior patterns, the communication interactions of narcotics organizations, the monetary transactions of money laundering and insider trading operations, the movements of serial killers, and the targeting of smugglers at border crossings. Data-mining techniques have also been employed by people in the intelligence community who maintain many large data sources as a part of the activities relating to matters of national security. Appendix B of the book gives a brief overview of typical commercial applications of data-mining technology today. Despite a considerable level of over-hype and strategic misuse, data mining has not only persevered but also matured and adapted for practical use in the business world.

1.2   DATA-MINING ROOTS

Looking at how different authors describe data mining, it is clear that we are far from a universal agreement on the definition of data mining or even what constitutes data mining. Is data mining a form of statistics enriched with learning theory, or is it a revolutionary new concept? In our view, most data-mining problems and corresponding solutions have roots in classical data analysis. Data mining has its origins in various disciplines, of which the two most important are *statistics* and *machine learning*.

Statistics has its roots in mathematics, and therefore, there has been an emphasis on mathematical rigor, a desire to establish that something is sensible on theoretical grounds before testing it in practice. In contrast, the machine-learning community has its origins very much in computer practice. This has led to a practical orientation, a willingness to test something out to see how well it performs, without waiting for a formal proof of effectiveness.

If the place given to mathematics and formalizations is one of the major differences between statistical and machine-learning approaches to data mining, another is in the relative emphasis they give to models and algorithms. Modern statistics is almost entirely driven by the notion of a model. This is a postulated structure, or an approximation to a structure, which could have led to the data. In place of the statistical emphasis on models, machine learning tends to emphasize algorithms. This is hardly surprising; the very word "learning" contains the notion of a process, an implicit algorithm.

Basic modeling principles in data mining also have roots in *control theory*, which is primarily applied to engineering systems and industrial processes. The problem of determining a mathematical model for an unknown system (also referred to as the target system) by observing its input–output data pairs is generally referred to as system identification. The purposes of system identification are multiple, and, from a standpoint of data mining, the most important are to predict a system's behavior and to explain the interaction and relationships between the variables of a system.

System identification generally involves two top-down steps:

1. *Structure identification*—In this step, we need to apply a priori knowledge about the target system to determine a class of models within which the search for the most suitable model is to be conducted. Usually this class of models is denoted by a parameterized function $y = f(u,t)$, where y is the model's output, u is an input vector, and t is a parameter vector. The determination of the function f is problem dependent, and the function is based on the designer's experience, intuition, and the laws of nature governing the target system.
2. *Parameter identification*—In the second step, when the structure of the model is known, all we need to do is apply optimization techniques to determine parameter vector t such that the resulting model $y* = f(u,t*)$ can describe the system appropriately.

In general, system identification is not a one-pass process: both structure and parameter identification need to be done repeatedly until a satisfactory model is found. This iterative process is represented graphically in Figure 1.1. Typical steps in every iteration are as follows:

1. Specify and parameterize a class of formalized (mathematical) models, $y* = f(u,t)$, representing the system to be identified.
2. Perform parameter identification to choose the parameters that best fit the available data set (the difference $y - y*$ is minimal).
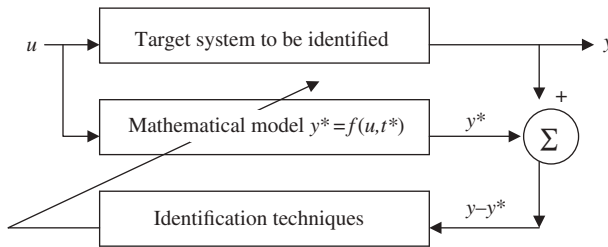
Figure 1.1. Block diagram for parameter identification.

3. Conduct validation tests to see if the model identified responds correctly to an unseen data set (often referred as test, validating, or checking data set).
4. Terminate the process once the results of the validation test are satisfactory.

If we do not have any a priori knowledge about the target system, then structure identification becomes difficult, and we have to select the structure by trial and error. While we know a great deal about the structures of most engineering systems and industrial processes, in a vast majority of target systems where we apply data-mining techniques, these structures are totally unknown, or they are so complex that it is impossible to obtain an adequate mathematical model. Therefore, new techniques were developed for parameter identification, and they are today a part of the spectra of data-mining techniques.

Finally, we can distinguish between how the terms "model" and "pattern" are interpreted in data mining. A model is a "large-scale" structure, perhaps summarizing relationships over many (sometimes all) cases, whereas a pattern is a local structure, satisfied by few cases or in a small region of a data space. It is also worth noting here that the word "pattern," as it is used in pattern recognition, has a rather different meaning for data mining. In pattern recognition it refers to the vector of measurements characterizing a particular object, which is a point in a multidimensional data space. In data mining, a pattern is simply a local model. In this book we refer to n-dimensional vectors of data as *samples*.

1.3   DATA-MINING PROCESS

Without trying to cover all possible approaches and all different views about data mining as a discipline, let us start with one possible, sufficiently broad definition of data mining:

> Data Mining is a process of discovering various models, summaries, and derived values from a given collection of data.

The word "process" is very important here. Even in some professional environ-ments, there is a belief that data mining simply consists of picking and applying a

computer-based tool to match the presented problem and automatically obtaining a solution. This is a misconception based on an artificial idealization of the world. There are several reasons why this is incorrect. One reason is that data mining is not simply a collection of isolated tools, each completely different from the other and waiting to be matched to the problem. A second reason lies in the notion of matching a problem to a technique. Only very rarely is a research question stated sufficiently precisely that a single and simple application of the method will suffice. In fact, what happens in practice is that data mining becomes an iterative process. One studies the data, examines it using some analytic technique, decides to look at it another way, perhaps modifies it, and then goes back to the beginning and applies another data-analysis tool, reaching either better or different results. This can go round and round many times; each technique is used to probe slightly different aspects of data—to ask a slightly different question of the data. What is essentially being described here is a voyage of discovery that makes modern data mining exciting. Still, data mining is not a random application of statistical, machine learning, and other methods and tools. It is not a random walk through the space of analytic techniques but a carefully planned and considered process of deciding what will be most useful, promising, and revealing.

It is important to realize that the problem of discovering or estimating dependencies from data or discovering totally new data is only one part of the general experimental procedure used by scientists, engineers, and others who apply standard steps to draw conclusions from the data. The general experimental procedure adapted to data-mining problems involves the following steps:

1. **State the problem and formulate the hypothesis**

   Most data-based modeling studies are performed in a particular application domain. Hence, domain-specific knowledge and experience are usually necessary in order to come up with a meaningful problem statement. Unfortunately, many application studies tend to focus on the data-mining technique at the expense of a clear problem statement. In this step, a modeler usually specifies a set of variables for the unknown dependency and, if possible, a general form of this dependency as an initial hypothesis. There may be several hypotheses formulated for a single problem at this stage. The first step requires the combined expertise of an application domain and a data-mining model. In practice, it usually means a close interaction between the data-mining expert and the application expert. In successful data-mining applications, this cooperation does not stop in the initial phase; it continues during the entire data-mining process.

2. **Collect the data**

   This step is concerned with how the data are generated and collected. In general, there are two distinct possibilities. The first is when the data-generation process is under the control of an expert (modeler): this approach is known as a *designed experiment*. The second possibility is when the expert cannot influence the data-generation process: this is known as the *observational approach*. An observational setting, namely, random data generation, is assumed in most data-mining

applications. Typically, the sampling distribution is completely unknown after data are collected, or it is partially and implicitly given in the data-collection procedure. It is very important, however, to understand how data collection affects its theoretical distribution, since such a priori knowledge can be very useful for modeling and, later, for the final interpretation of results. Also, it is important to make sure that the data used for estimating a model and the data used later for testing and applying a model come from the same, unknown, sampling distribution. If this is not the case, the estimated model cannot be successfully used in a final application of the results.

3. **Preprocessing the data**

In the observational setting, data are usually "collected" from the existing databases, data warehouses, and data marts. Data preprocessing usually includes at least two common tasks:

(a) *Outlier detection (and removal)*

Outliers are unusual data values that are not consistent with most observations. Commonly, outliers result from measurement errors and coding and recording errors and, sometimes, are natural, abnormal values. Such non-representative samples can seriously affect the model produced later. There are two strategies for dealing with outliers:

(i) Detect and eventually remove outliers as a part of the preprocessing phase.

(ii) Develop robust modeling methods that are insensitive to outliers.

(b) *Scaling, encoding, and selecting features*

Data preprocessing includes several steps such as variable scaling and different types of encoding. For example, one feature with the range [0, 1] and the other with the range [−100, 1000] will not have the same weight in the applied technique; they will also influence the final data-mining results differently. Therefore, it is recommended to scale them and bring both features to the same weight for further analysis. Also, application-specific encoding methods usually achieve dimensionality reduction by providing a smaller number of informative features for subsequent data modeling.

These two classes of preprocessing tasks are only illustrative examples of a large spectrum of preprocessing activities in a data-mining process.

Data-preprocessing steps should not be considered completely independent from other data-mining phases. In every iteration of the data-mining process, all activities, together, could define new and improved data sets for subsequent iterations. Generally, a good preprocessing method provides an optimal representation for a data-mining technique by incorporating a priori knowledge in the form of application-specific scaling and encoding. More about these techniques and the preprocessing phase in general will be given in Chapters 2 and 3, where we have functionally divided preprocessing and its corresponding techniques into two subphases: data preparation and data-dimensionality reduction.

4. **Estimate the model**

The selection and implementation of the appropriate data-mining technique is the main task in this phase. This process is not straightforward; usually, in practice, the implementation is based on several models, and selecting the best one is an additional task. The basic principles of learning and discovery from data are given in Chapter 4 of this book. Later, Chapters 5 through 13 explain and analyze specific techniques that are applied to perform a successful learning process from data and to develop an appropriate model.

5. **Interpret the model and draw conclusions**

In most cases, data-mining models should help in decision-making. Hence, such models need to be interpretable in order to be useful because humans are not likely to base their decisions on complex "black-box" models. Note that the goals of accuracy of the model and accuracy of its interpretation are somewhat contradictory. Usually, simple models are more interpretable, but they are also less accurate. Modern data-mining methods are expected to yield highly accurate results using high-dimensional models. The problem of interpreting these models, also very important, is considered a separate task, with specific techniques to validate the results. A user does not want hundreds of pages of numerical results. He does not understand them; he cannot summarize, interpret, and use them for successful decision-making.

Even though the focus of this book is on steps 3 and 4 in the data-mining process, we have to understand that they are just two steps in a more complex process. All phases, separately, and the entire data-mining process, as a whole, are highly iterative, as has been shown in Figure 1.2. A good understanding of the whole process is important for any successful application. No matter how powerful the data-mining method used in step 4 is, the resulting model will not be valid if the data are not collected and preprocessed correctly or if the problem formulation is not meaningful.

In 1999, several large companies including automaker Daimler-Benz, insurance provider OHRA, hardware and software manufacturer NCR Corp., and statistical software maker SPSS, Inc. formalize and standardize an approach to data-mining process. The result of their work was CRISP-DM, the CRoss-Industry Standard Process for Data Mining presented on Figure 1.3. The process was designed to be independent of any specific tool. The CRISP-DM methodology provides a structured approach in planning a data-mining project. Numerous data-mining applications showed its practicality and flexibility and its usefulness when using analytics to solve complex business issues. This model is an idealized sequence of events. In practice, many of the tasks can be performed in a different order, and it will often be necessary to backtrack to previous activities and repeat certain actions. The model does not try to capture all possible routes through the data-mining process. The reader may recognize the connection and similarities between steps of data mining presented on Figures 1.2 and 1.3.
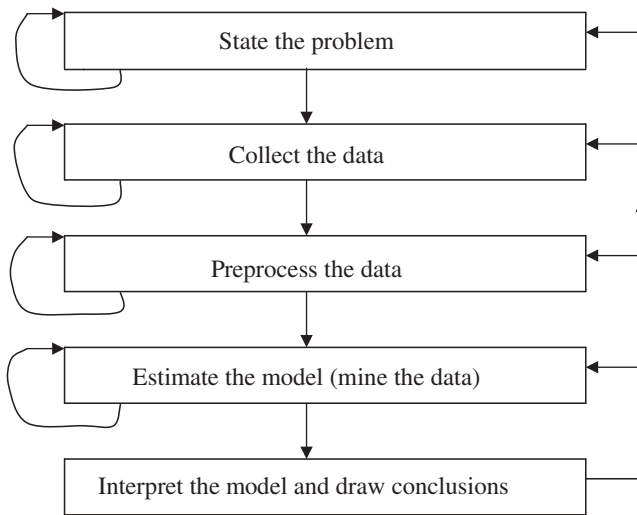
Figure 1.2. The data-mining process.
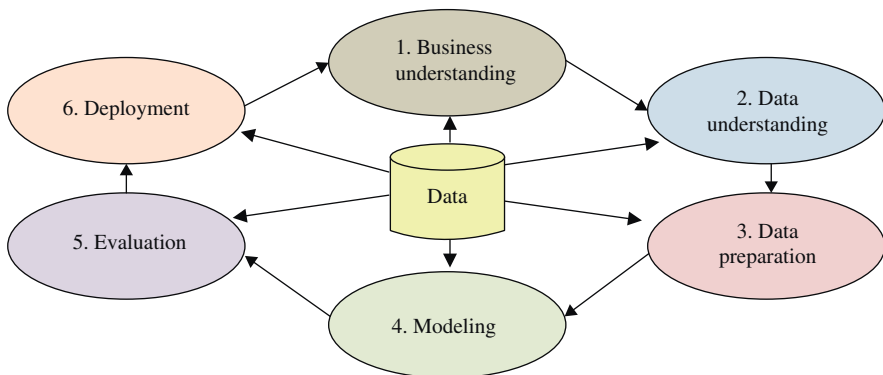
Figure 1.3. CRISP-DM conceptual model.

1.4   FROM DATA COLLECTION TO DATA PREPROCESSING

As we enter into the age of digital information, the problem of data overload looms ominously ahead. Our ability to analyze and understand massive *data sets* is far behind our ability to gather and store the data. Recent advances in computing, communications, and digital storage technologies, together with the development of high-throughput data-acquisition technologies, have made it possible to gather and store incredible volumes of data. Large databases of digital information are ubiquitous.

World internet hosts: 1981 – 2009 (Data source: ISC
https://www.isc.org/solutions/survey/history)
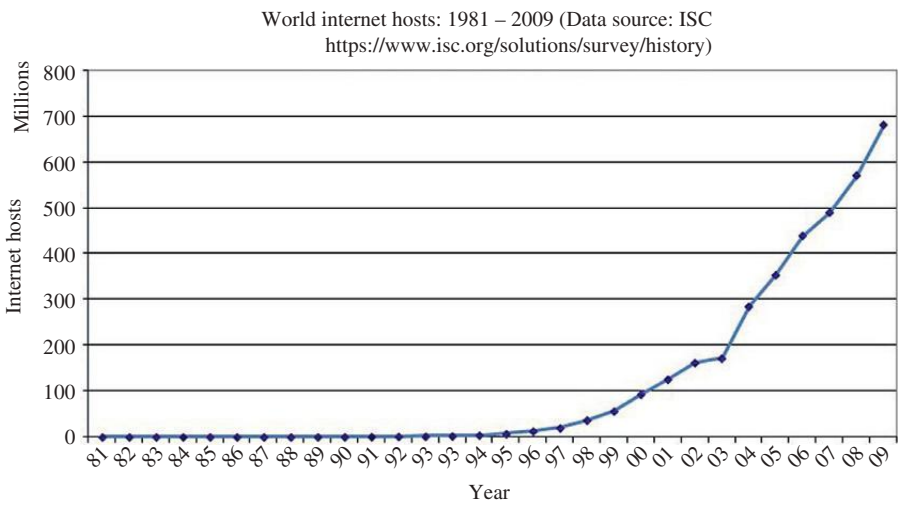
Figure 1.4. Growth of Internet hosts.

Data from the neighborhood store's checkout register, your bank's credit card authorization device, records in your doctor's office, patterns in your telephone calls, and many more applications generate streams of digital records archived in huge business databases. Complex distributed computer systems, communication networks, and power systems, for example, are equipped with sensors and measurement devices that gather and store a variety of data for use in monitoring, controlling, and improving their operations. Scientists are at the higher end of today's data-collection machinery, using data from different sources—from remote sensing platforms to microscope probing of cell details. Scientific instruments can easily generate terabytes of data in a short period of time and store them in the computer. One example is the hundreds of terabytes of DNA, protein-sequence, and gene expression data that biological science researchers have gathered at steadily increasing rates. The information age, with the expansion of the Internet, has caused an exponential growth in information sources and also in information storage units. An illustrative example is given in Figure 1.4, where we can see a dramatic increase of Internet hosts in recent years, where these numbers are directly proportional to the amount of data stored on the Internet.

It is estimated that the digital universe consumed approximately 281 exabytes in 2007, and it was already 10 times that size by 2011. (One exabyte is $\sim 10^{18}$ bytes or 1,000,000 terabytes.) Inexpensive digital and video cameras have made available huge archives of images and videos. The prevalence of radio frequency ID (RFID) tags or transponders due to their low cost and small size has resulted in the deployment of millions of sensors that transmit data regularly. E-mails, blogs, transaction data, and billions of Web pages create terabytes of new data every day.

There is a rapidly widening gap between data-collection and data-organization capabilities and the ability to analyze the data. Current hardware and database

technology allows efficient, inexpensive, and reliable data storage and access. However, whether the context is business, medicine, science, or government, the data sets themselves, in their raw form, are of little direct value. What is of value is the knowledge that can be inferred from the data and put to use. For example, the marketing database of a consumer goods company may yield knowledge of the correlation between sales of certain items and certain demographic groups. This knowledge can be used to introduce new, targeted marketing campaigns with a predictable financial return, as opposed to unfocused campaigns.

The root of the problem is that the data size and dimensionality are too large for manual analysis and interpretation or even for some semiautomatic computer-based analyses. A scientist or a business manager can work effectively with a few hundred or thousand records. Effectively mining millions of data points, each described with tens or hundreds of characteristics, is another matter. Imagine the analysis of terabytes of sky image data with thousands of photographic high-resolution images ($23,040 \times 23,040$ pixels per image) or human genome databases with billions of components. In theory, "big data" can lead to much stronger conclusions, but in practice many difficulties arise. The business community is well aware of today's information overload, and one analysis shows that:

1. 61% of managers believe that information overload is present in their own workplace,
2. 80% believe the situation will get worse,
3. more than 50% of the managers ignore data in current decision-making processes because of the information overload,
4. 84% of managers store this information for the future; it is not used for current analysis, and
5. 60% believe that the cost of gathering information outweighs its value.

What are the solutions? Work harder. Yes, but how long can you keep up, because the limits are very close. Employ an assistant. Maybe, if you can afford it. Ignore the data. But then you are not competitive in the market. The only real solution will be to replace classical data-analysis and interpretation methodologies (both manual and computer based) with a new data-mining technology.

In theory, most data-mining methods should be happy with large data sets. Large data sets have the potential to yield more valuable information. If data mining is a search through a space of possibilities, then large data sets suggest many more possibilities to enumerate and evaluate. The potential for increased enumeration and search is counterbalanced by practical limitations. Besides the computational complexity of the data-mining algorithms that work with large data sets, a more exhaustive search may also increase the risk of finding some low-probability solutions that evaluate well for the given data set, but may not meet future expectations.

In today's multimedia-based environment that has a huge Internet infrastructure, different types of data are generated and digitally stored. To prepare adequate

data-mining methods, we have to analyze the basic types and characteristics of data sets. The first step in this analysis is systematization of data with respect to their computer representation and use. Data that is usually the source for a data-mining process can be classified into structured data, semi-structured data, and unstructured data

Most business databases contain structured data consisting of well-defined fields with numeric or alphanumeric values, while scientific databases may contain all three classes. Examples of semi-structured data are electronic images of business documents, medical reports, executive summaries, and repair manuals. The majority of Web documents also fall in this category. An example of unstructured data is a video recorded by a surveillance camera in a department store. Such visual and, in general, multimedia recordings of events or processes of interest are currently gaining widespread popularity because of reduced hardware costs. This form of data generally requires extensive processing to extract and structure the information contained in it.

Structured data is often referred to as traditional data, while the semi-structured and unstructured data are lumped together as nontraditional data (also called multimedia data). Most of the current data-mining methods and commercial tools are applied to traditional data. However, the development of data-mining tools for nontraditional data, as well as interfaces for its transformation into structured formats, is progressing at a rapid rate.

The standard model of structured data for data mining is a collection of cases. Potential measurements called features are specified, and these features are uniformly measured over many cases. Usually the representation of structured data for data-mining problems is in a tabular form or in the form of a single relation (term used in relational databases), where columns are features of objects stored in a table and rows are values of these features for specific entities. A simplified graphical representation of a data set and its characteristics is given in Figure 1.5. In the data-mining literature, we usually use the terms samples or cases for rows. Many different types of features (attributes or variables)—i.e. fields—in structured data records are common in data mining. Not all of the data-mining methods are equally good at dealing with different types of features.

There are several ways of characterizing features. One way of looking at a feature—or in a formalization process, the more often used term, variable—is to see whether it is an *independent variable* or a *dependent variable*, that is, whether or not it is a variable whose values depend upon values of other variables represented in a data set. This is a model-based approach to classifying variables. All dependent variables are accepted as outputs from the system for which we are establishing a model, and independent variables are inputs to the system, as represented in Figure 1.6.

There are some additional variables that influence system behavior, but the corresponding values are not available in a data set during a modeling process. The reasons are different: from high complexity and the cost of measurements for these features to a modeler's not understanding the importance of some factors and their influences on the model. These are usually called unobserved variables, and they are the main cause of ambiguities and estimations in a model.
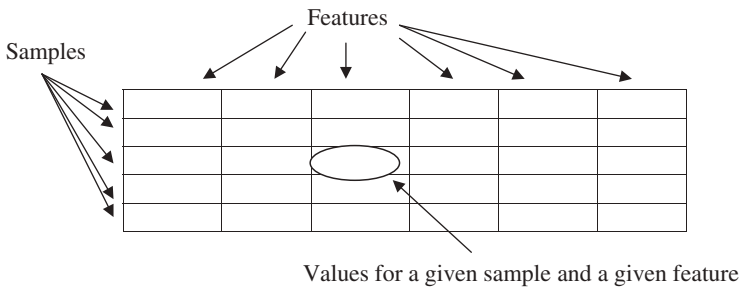
Figure 1.5. Tabular representation of a data set.
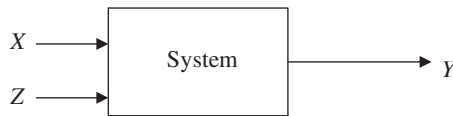
Figure 1.6. A real system, besides input (independent) variables *X* and (dependent) outputs *Y*, often has unobserved inputs *Z*.

Today' computers and corresponding software tools support processing of data sets with millions of samples and hundreds of features. Large data sets, including those with mixed data types, are a typical initial environment for application of data-mining techniques. When a large amount of data is stored in a computer, one cannot rush into data-mining techniques, because the important problem of data quality has first to be resolved. Also, it is obvious that a manual quality analysis is not possible at that stage. Therefore, it is necessary to prepare a data-quality analysis in the earliest phases of the data-mining process; usually it is a task to be undertaken in the data-preprocessing phase. The quality of data could limit the ability of end users to make informed decisions. It has a profound effect on the image of the system and determines the corresponding model that is implicitly described. Using the available data-mining techniques, it will be difficult to undertake major qualitative changes in an organization based on poor-quality data; also, to make new sound discoveries from poor-quality scientific data will be almost impossible. There are a number of indicators of data quality that have to be taken care of in the preprocessing phase of a data-mining process:

1. The data should be accurate. The analyst has to check that the name is spelled correctly, the code is in a given range, the value is complete, and so on.

2. The data should be stored according to data type. The analyst must ensure that the numerical value is not presented in character form, that integers are not in the form of real numbers, and so on.

3. The data should have integrity. Updates should not be lost because of conflicts among different users; robust backup and recovery procedures should be implemented if they are not already part of the Data Base Management System (DBMS).

4. The data should be consistent. The form and the content should be the same after integration of large data sets from different sources.
5. The data should not be redundant. In practice, redundant data should be minimized, and reasoned duplication should be controlled, or duplicated records should be eliminated.
6. The data should be timely. The time component of data should be recognized explicitly from the data or implicitly from the manner of its organization.
7. The data should be well understood. Naming standards are a necessary but not the only condition for data to be well understood. The user should know that the data corresponds to an established domain.
8. The data set should be complete. Missing data, which occurs in reality, should be minimized. Missing data could reduce the quality of a global model. On the other hand, some data-mining techniques are robust enough to support analyses of data sets with missing values.

How to work with and solve some of these problems of data quality is explained in greater detail in Chapters 2 and 3 where basic data-mining preprocessing methodologies are introduced. These processes are performed very often using data-warehousing technology, briefly explained in Section 1.5.

1.5   DATA WAREHOUSES FOR DATA MINING

Although the existence of a data warehouse is not a prerequisite for data mining, in practice, the task of data mining, especially for some large companies, is made a lot easier by having access to a data warehouse. A primary goal of a data warehouse is to increase the "intelligence" of a decision process and the knowledge of the people involved in this process. For example, the ability of product marketing executives to look at multiple dimensions of a product's sales performance—by region, by type of sales, and by customer demographics—may enable better promotional efforts, increased production, or new decisions in product inventory and distribution. It should be noted that average companies work with averages. The superstars differentiate themselves by paying attention to the details. They may need to slice and dice the data in different ways to obtain a deeper understanding of their organization and to make possible improvements. To undertake these processes, users have to know what data exists, where it is located, and how to access it.

A data warehouse means different things to different people. Some definitions are limited to data; others refer to people, processes, software, tools, and data. One of the global definitions is the following:

The data warehouse is a collection of integrated, subject-oriented databases designed to support the decision-support functions (DSF), where each unit of data is relevant to some moment in time.

Based on this definition, a data warehouse can be viewed as an organization's repository of data, set up to support strategic decision-making. The function of the

data warehouse is to store the historical data of an organization in an integrated manner that reflects the various facets of the organization and business. The data in a warehouse are never updated but used only to respond to queries from end users who are generally decision-makers. Typically, data warehouses are huge, storing billions of records. In many instances, an organization may have several local or departmental data warehouses often called data marts. A data mart is a data warehouse that has been designed to meet the needs of a specific group of users. It may be large or small, depending on the subject area.

At this early time in the evolution of data warehouses, it is not surprising to find many projects floundering because of the basic misunderstanding of what a data warehouse is. What *is* surprising is the size and scale of these projects. Many companies err by not defining exactly what a data warehouse is, the business problems it will solve, and the uses to which it will be put. Two aspects of a data warehouse are most important for a better understanding of its design process: the first is the specific types (classification) of data stored in a data warehouse, and the second is the set of transformations used to prepare the data in the final form such that it is useful for decision-making. A data warehouse includes the following categories of data, where the classification is accommodated to the time-dependent data sources:

1. Old detail data,
2. Current (new) detail data,
3. Lightly summarized data,
4. Highly summarized data, and
5. Metadata (the data directory or guide).

To prepare these five types of elementary or derived data in a data warehouse, the fundamental types of data transformation are standardized. There are four main types of transformations, and each has its own characteristics:

1. *Simple transformations*—These transformations are the building blocks of all other more complex transformations. This category includes manipulation of data that is focused on one field at a time, without taking into account its values in related fields. Examples include changing the data type of a field or replacing an encoded field value with a decoded value.
2. *Cleansing and scrubbing*—These transformations ensure consistent formatting and usage of a field or of related groups of fields. This can include a proper formatting of address information, for example. This class of transformations also includes checks for valid values in a particular field, usually checking the range or choosing from an enumerated list.
3. *Integration*—This is a process of taking operational data from one or more sources and mapping it, field by field, onto a new data structure in the data warehouse. The common identifier problem is one of the most difficult integration issues in building a data warehouse. Essentially, this situation occurs when there are multiple system sources for the same entities and there is no

clear way to identify those entities as the same. This is a challenging problem, and in many cases it cannot be solved in an automated fashion. It frequently requires sophisticated algorithms to pair up probable matches. Another complex data-integration scenario occurs when there are multiple sources for the same data element. In reality, it is common that some of these values are contradictory, and resolving a conflict is not a straightforward process. Just as difficult as having conflicting values is having no value for a data element in a warehouse. All these problems and corresponding automatic or semiautomatic solutions are always domain dependent.

4. *Aggregation and summarization*—These are methods of condensing instances of data found in the operational environment into fewer instances in the warehouse environment. Although the terms aggregation and summarization are often used interchangeably in the literature, we believe that they do have slightly different meanings in the data-warehouse context. Summarization is a simple addition of values along one or more data dimensions, e.g. adding up daily sales to produce monthly sales. Aggregation refers to the addition of different business elements into a common total; it is highly domain dependent. For example, aggregation is adding daily product sales and monthly consulting sales to get the combined monthly total.

These transformations are the main reason why we prefer a warehouse as a source of data for a data-mining process. If the data warehouse is available, the preprocessing phase in data mining is significantly reduced, sometimes even eliminated. Do not forget that this preparation of data is the most time-consuming phase. Although the implementation of a data warehouse is a complex task, described in many textbooks in great detail, in this text we are giving only the basic characteristics. A three-stage data-warehousing development process is summarized through the following basic steps:

1. *Modeling*—In simple terms, to take the time to understand business processes, the information requirements of these processes, and the decisions that are currently made within processes.

2. *Building*—To establish requirements for tools that suit the types of decision support necessary for the targeted business process; to create a data model that helps further define information requirements; and to decompose problems into data specifications and the actual data store, which will, in its final form, represent either a data mart or a more comprehensive data warehouse.

3. *Deploying*—to implement, relatively early in the overall process, the nature of the data to be warehoused and the various business intelligence tools to be employed; and to begin by training users. The deploy stage explicitly contains a time during which users explore both the repository (to understand data that are and should be available) and early versions of the actual data warehouse. This can lead to an evolution of the data warehouse, which involves adding more data, extending historical periods, or returning to the build stage to expand the scope of the data warehouse through a data model.

Data mining represents one of the major applications for data warehousing, since the sole function of a data warehouse is to provide information to end users for decision support. Unlike other query tools and application systems, the data-mining process provides an end user with the capacity to extract hidden, nontrivial information. Such information, although more difficult to extract, can provide bigger business and scientific advantages and yield higher returns on "data-warehousing and data-mining" investments.

How is data mining different from other typical applications of a data warehouse, such as structured query languages (SQL) and online analytical processing tools (OLAP), which are also applied to data warehouses? SQL is a standard relational database language that is good for queries that impose some kind of constraints on data in the database in order to extract an answer. In contrast, data-mining methods are good for queries that are exploratory in nature, trying to extract hidden, not so obvious information. SQL is useful when we know exactly what we are looking for and we can describe it formally. We will use data-mining methods when we know only vaguely what we are looking for. Therefore these two classes of data-warehousing applications are complementary.

OLAP tools and methods have become very popular in recent years as they let users analyze data in a warehouse by providing multiple views of the data, supported by advanced graphical representations. In these views, different dimensions of data correspond to different business characteristics. OLAP tools make it very easy to look at dimensional data from any angle or to slice and dice it. OLAP is part of the spectrum of decision-support tools. Traditional query and report tools describe *what* is in a database. OLAP goes further; it is used to answer *why* certain things are true. The user forms a hypothesis about a relationship and verifies it with a series of queries against the data. For example, an analyst might want to determine the factors that lead to loan defaults. He or she might initially hypothesize that people with low incomes are bad credit risks and analyze the database with OLAP to verify (or disprove) this assumption. In other words, the OLAP analyst generates a series of hypothetical patterns and relationships and uses queries against the database to verify them or disprove them. OLAP analysis is essentially a deductive process.

Although OLAP tools, like data-mining tools, provide answers that are derived from data, the similarity between them ends here. The derivation of answers from data in OLAP is analogous to calculations in a spreadsheet; because they use simple and given-in-advance calculations, OLAP tools do not learn from data, nor do they create new knowledge. They are usually special-purpose visualization tools that can help end users draw their own conclusions and decisions, based on graphically condensed data. OLAP tools are very useful for the data-mining process; they can be a part of it but they are not a substitute.

1.6   FROM BIG DATA TO DATA SCIENCE

We are living in a data tsunami era where enormous amount of data have been continually generated, each day at increasing scales. This exponential growth of potentially valuable data, compounded by the Internet, social media, cloud computing,

variety of sensors and new types of mobile devices, is often referred to as *big data*. Recent studies estimate an increase of annually created data from around 1.2 zetta-bytes in 2010 to 40 zettabytes in 2020. If this is a new concept for the reader, it means the following: 1 zettabyte $= 10^3$ exabytes $= 10^6$ petabytes. *Big data* may be primary generated through five main types of data sources:

- *Operational data* comes from traditional transactional systems, where the assumption is that it includes monitoring streaming data often coming from large amount of sensors.
- *Dark data* is large amount of data that you already own, but do not use in current decision processes; it may include emails, contracts, and variety of written reports.
- *Commercial data* is available on the market and may be purchased from some companies, specialized social media, or even governmental organizations.
- *Social data* coming from Twitter, Facebook, and other general social media and examples of the rapid growth of data are given in Table 1.1.
- *Public data* such as economic, sociodemographic, or weather data (Fig. 1.7).

Big data could be a new infrastructure for advancements of medical research, global security, logistics and transportation solutions, and identification of terrorism activities and also dealing with socio-economic and environmental issues.

Fundamentally, *Big data* means not only a large volume of data but also other features that differentiate it from the concepts of "massive data" and "very large data." The term *Big data* has gained huge popularity in recent years, but it is still poorly defined. One of the most commonly cited definitions specify big data through the four following dimensions: "volume," "variety," "velocity," and "veracity" (so-called 4V model):

1. *Volume* refers to the magnitude of data. Real-world big data applications are reported in multiple terabytes and petabytes, and tomorrow they will be in exabytes. What may be deemed and impress as *big data* today may not meet the

TABLE 1.1. Big Data on the Web

Company	Big Data
YouTube	Users upload 100 hours of new videos per minute
Facebook	More than 1.4 billion users communicating in 70+ languages
Twitter	175 million tweets per day
Google	2 million search queries/minute → processing 35 petabytes daily
Apple	47,000 applications are downloaded per minute
Instagram	Users share 40 million photos per day
LinkedIn	2.1 million groups have been created
Foursquare	571 new Web sites are launched each minute

Global data in zetabytes
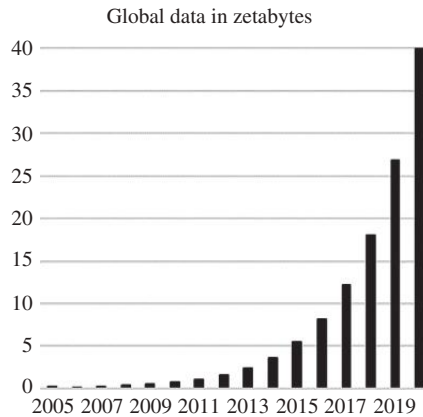
Figure 1.7. Exponential growth of global data. From: http://s3.amazonaws.com/sdieee/
1990-IEEE_meeting_Jun_2016_Final-2.pdf.

threshold in the future. Storage capacities are increasing, and new tools are developing, allowing bigger data sets to be captured and analyzed.

2. *Variety* refers to the structural heterogeneity in a data set, including the use and benefits of various types of structured, semi-structured, and unstructured data. Text, images, audio, and video are examples of unstructured data, which are dominant data types with more than 90% representation in today's digital world. These different forms and quality of data clearly indicate that heterogeneity is a natural property of big data and it is a challenge to comprehend and successfully manage such data. For instance, during the Fukushima nuclear disaster, when the public started broadcasting about radioactive materials, a wide variety of inconsistent data, using diverse and uncalibrated devices, for similar or neighboring locations was reported—all this add to the problem of increasing variety of data.

3. *Velocity* refers to the rate at which data are generated and the speed at which it should be analyzed and acted upon. Digital devices such as smartphones and variety of available and relatively cheap sensors have led to an unprecedented rate of data creation in real time. It requires new IT infrastructures and new methodologies supporting growing need for real-time analytics. Floods of digital personalized data about customers, such as their geospatial location and buying behavior and patterns, can be used in real time for many companies to monitor and improve their business models.

4. *Veracity* highlights the unreliability inherent in some sources of today's digital data. The need to deal with this imprecise and uncertain data is important facet of big data, which is requiring adjustment of tools and applied analytics methodologies. The fact that one in three business leaders does not trust the information that they use to make decisions is a strong indicator that a good

big data application needs to address veracity. Customer sentiments, analyzed through the Internet, are an example where the data is uncertain in nature, since they entail human judgment. Yet, they contain valuable information that could help businesses.

There are many businesses and scientific opportunities related to *big data*, but at the same time new threats are there too. Big data market is poised to grow to more than $50 billion in 2017, but at the same time more than 55% of big data projects failed! Heterogeneity, ubiquity, and dynamic nature of the different resources and devices for data generation, and the enormous scale of data itself, make determining, retrieving, processing, integrating, and inferring the real-world data a challenging task. For the beginning we can briefly enumerate main problems with implementations and threats to these new *big data* solutions:

(a) Data breaches and reduced security,
(b) Intrusion of user's privacy,
(c) Unfair use of data,
(d) Escalating cost of data movement,
(e) Scalability of computations, and
(f) Data quality.

Because of these serious challenges, novel approaches and techniques are required to address these big data problems.

Although it seems that *big data* makes it possible to find more useful, actionable information, the truth is that more data do not necessarily mean better analyses and more informative conclusions. Therefore, designing and deploying a big data mining system is not a trivial or straightforward task. The remaining chapters of this book will try to give some initial answers to these big data challenges.

In this introductory section we would like to introduce one more concept that is highly related to *big data*. It is the new field of *data science*. Decision-makers of all kinds, from company executives and government agencies to researchers and scientists, would like to base their decisions and actions on the available data. In response to these multidisciplinary requests, a new discipline of big data science is forming. Data scientists are professionals who are trying to gain knowledge or awareness of something not known before about data. They need business knowledge; they need to know how to deploy new technology; they have to understand statistical, machine learning, and visualization techniques; and they need to know how to interpret and present the results.

The name of data science seems to connect most strongly with areas such as databases and computer science in general, and more specific it is based on machine learning and statistics. But many different kinds of skill are necessary for the profile, and many other disciplines are involved: skillful in communication with data users; understanding the big picture of a complex system described by data; analyzing business aspects of big data application; knowing how to transform, visualize, interpret, and summarize big data; maintaining the quality of data; and taking care about security,

privacy, and legal aspect of data. Of course there are very small number of experts who are good in all these skills, and therefore, we have always to make emphasis on the importance of multidisciplinary teamwork in big data environments. Maybe the following definition of a data scientists, which insists and highlights professional persistence, gives better insight: *A data scientist is the adult version of a kid who can't stop asking "Why?"*. Data science is supporting discoveries in many human endeavors, including healthcare, manufacturing, education, cybersecurity, financial modeling, social science, policing, and marketing. It has been used to produce significant results in areas from particle physics such as Higgs Boson, and identifying and resolving sleep disorders using Fitbit data, to recommender systems for literature, theater, and shopping. As a result of these initial successes and potential, data science is rapidly becoming an applied sub-discipline of many academic areas.

Very often there is confusion between concepts of data science, big data analytics, and data mining. Based on previous interpretations of a data science discipline, data mining highlight only a segment of data scientist's tasks, but they represent very important core activities in gaining new knowledge from *big data*. Although major innovations in data-mining techniques for big data have not yet matured, we anticipate the emergence of such novel analytics in the near future. Recently, several additional terms including advanced data analytics are introduced and more often used, but with some level of approximation, we can accept them as equivalent concepts with data mining.

The sudden rise of big data has left many unprepared including corporate leaders, municipal planners, and academics. The fast evolution of big data technologies and the ready acceptance of the concept by public and private sectors left little time for the discipline to mature, leaving open questions of security, privacy, and legal aspects of big data. The security and privacy issues that accompany the work of big data mining are challenging research topics. They contain important questions how to safely store the data, how to make sure the data communication is protected, and how to prevent someone from finding out our private information. Because big data means more sensitive data is put together, it is more attractive to potential hackers: in 2012 LinkedIn was accused of leaking 6.5 million user account passwords, while later Yahoo faced network attacks, resulting in 450,000 user ID leaks. The privacy concern typically will make most people uncomfortable, especially if systems cannot guarantee that their personal information will not be accessed by the other people and organizations. The anonymous, temporary identification and encryption are the representative technologies for privacy of big data mining, but the critical factor is how to use, what to use, when to use, and why to use the collected big data.

1.7 BUSINESS ASPECTS OF DATA MINING: WHY A DATA-MINING PROJECT FAILS?

Data mining in various forms is becoming a major component of business operations. Almost every business process today involves some form of data mining. Customer relationship management, supply chain optimization, demand forecasting, assortment optimization, business intelligence, and knowledge management are just some

examples of business functions that have been impacted by data-mining techniques. Even though data mining has been successful in becoming a major component of various business and scientific processes as well as in transferring innovations from academic research into the business world, the gap between the problems that the data mining research community works on and real-world problems is still significant. Most business people (marketing managers, sales representatives, quality assurance managers, security officers, and so forth) who work in industry are only interested in data mining insofar as it helps them do their job better. They are uninterested in technical details and do not want to be concerned with integration issues; a successful data-mining application has to be integrated seamlessly into an application. Bringing an algorithm that is successful in the laboratory to an effective data-mining application with real-world data in industry or scientific community can be a very long process. Issues like cost effectiveness, manageability, maintainability, software integration, ergonomics, and business process re-engineering come into play as significant components of a potential data-mining success.

Data mining in a business environment can be defined as the effort to generate actionable models through automated analysis of a company's data. In order to be useful, data mining must have a financial justification. It must contribute to the central goals of the company by, for example, reducing costs, increasing profits, improving customer satisfaction, or improving the quality of service. The key is to find actionable information or information that can be utilized in a concrete way to improve profitability of a company. For example, credit card marketing promotions typically generate a response rate of about 1%. The praxis shows that this rate is improved significantly through data-mining analyses. In telecommunications industry a big problem is the concept of churn, when customers switch carriers. When dropped calls, mobility patterns, and variety of demographic data are recorded, and data-mining techniques are applied, churn is reduced by an estimated 61%.

Data mining does not replace skilled business analysts or scientist, but rather gives them powerful new tools and support of an interdisciplinary team to improve the job they are doing. Today, companies collect huge amounts of data about their customers, partners, products, and employees as well as their operational and financial systems. They hire professionals (either locally or outsourced) to create data-mining models that analyze collected data to help business analysts create reports and identify trends, so that they can optimize their channel operations, improve service quality, and track customer profiles, ultimately reducing costs and increasing revenue. Still, there is a semantic gap between the data miner who talks about regressions, accuracy, and ROC curves and business analysts who talk about customer retention strategies, addressable markets, profitable advertising, etc. Therefore, in all phases of a data-mining process, it is a core requirement for understanding, coordination, and successful cooperation between all team members. The best results in data mining are achieved when data-mining experts combine experience with organizational domain experts. While neither group needs to be fully proficient in the other's field, it is certainly beneficial to have a basic background across areas of focus.

Introducing a data-mining application into an organization is not essentially very different from any other software application project, and the following conditions have to be satisfied:

- There must be a well-defined problem.
- The data must be available.
- The data must be relevant, adequate, and clean.
- The problem should not be solvable by means of ordinary query or OLAP tools only.
- The results must be actionable.

A number of data-mining projects have failed in the past years because one or more of these criteria were not met.

The initial phase of a data-mining process is essential from a business perspective. It focuses on understanding the project objectives and business requirements and then converting this knowledge into a data-mining problem definition, and a preliminary plan designed to achieve the objectives. The first objective of the data miner is to understand thoroughly, from a business perspective, what the client really wants to accomplish. Often the client has many competing objectives and constraints that must be properly balanced. The data miner's goal is to uncover important factors, at the beginning, that can influence the outcome of the project. A possible consequence of neglecting this step is to expend a great deal of effort producing the right answers to the wrong questions. Data-mining projects do not fail because of poor or inaccurate tools or models. The most common pitfalls in data mining involve a lack of training, overlooking the importance of a thorough pre-project assessment, not employing the guidance of a data-mining expert, and not developing a strategic project definition adapted to what is essentially a discovery process. A lack of competent assessment, environmental preparation, and resulting strategy is precisely why the vast majority of data-mining projects fail.

The model of a data-mining process should help to plan, work through, and reduce the cost of any given project by detailing procedures to be performed in each of the phases. The model of the process should provide a complete description of all phases from problem specification to deployment of the results. Initially the team has to answer the key question: What is the ultimate purpose of mining this data, and more specifically what are the business goals? The key to success in data mining is coming up with a precise formulation of the problem the team is trying to solve. A focused statement usually results in the best payoff. The knowledge of organization's needs or scientific research objectives will guide the team in formulating the goal of a data-mining process. The prerequisite to knowledge discovery is understanding the data and the business. Without this deep understanding, no algorithm, regardless of sophistication, is going to provide results in which a final user should have confidence. Without this background a data miner will not be able to identify the problems he/she is trying to solve or even correctly interpret the results. To make the best use of data mining, we must make a clear statement of project objectives. An effective

statement of the problem will include a way of measuring the results of a knowledge discovery project. It may also include details about a cost justification. Preparatory steps in a data-mining process may also include analysis and specification of a type of data-mining task and selection of an appropriate methodology and corresponding algorithms and tools. When selecting a data-mining product, we have to be aware that they generally have different implementations of a particular algorithm even when they identify it with the same name. Implementations differences can affect operational characteristics such as memory usage and data storage, as well as performance characteristics such as speed and accuracy.

The data understanding phase starts early in the project, and it includes important and time-consuming activities that could make enormous influence on the final success of the project. "Get familiar with the data" is the phrase that requires serious analysis of data including source of data, owner, organization responsible for maintaining data, cost (if purchased), storage organization, size in records and attributes, size in bytes, security requirements, restrictions on use, and privacy requirements. Also, the data miner should identify data-quality problems and discover first insights into the data such as data types, definitions of attributes, units of measure, list or range of values, collection information, time and space characteristics, missing and invalid data, etc. Finally, we should detect interesting subsets of data in these preliminary analyses to form hypotheses for hidden information. The important characteristic of a data-mining process is the relative time spent to complete each of the steps in the process, and the data is counterintuitive as it is presented in Figure 1.8. Some authors estimate that about 20% of the effort is spent on business objective determination, about 60% on data preparation and understanding, and only about 10% for data mining and analysis.

Technical literature reports only on successful data-mining applications. To increase our understanding of data-mining techniques and their limitations, it is crucial to analyze not only successful but also unsuccessful applications. Failures or dead ends also provide valuable input for data-mining research and applications. We have
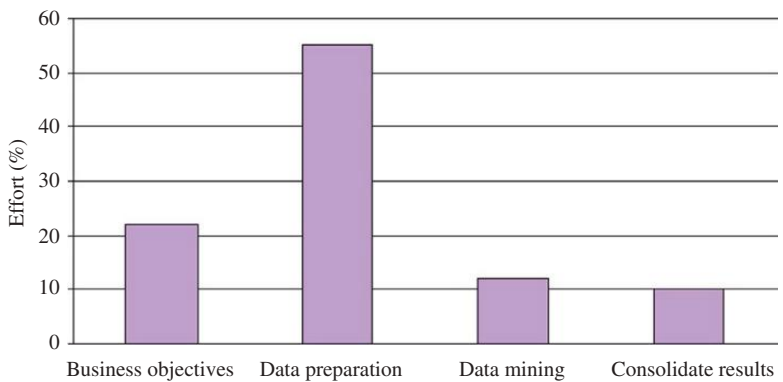
Figure 1.8. Effort in data-mining process.

to underscore the intensive conflicts that have arisen between practitioners of "digital discovery" and classical experience-driven human analysts objecting to these intrusions into their hallowed turf. One good case study is that of US economist Orley Ashenfelter, who used data-mining techniques to analyze the quality of French Bordeaux wines. Specifically he sought to relate auction prices to specific local annual weather conditions, in particular rainfall and summer temperatures. His findings were that hot and dry years produced the wines most valued by buyers. Ashenfelter's work and analytical methodology resulted in a deluge of hostile invective from established wine tasting experts and writers. There was a fear of losing a lucrative monopoly and the reality that a better informed market is more difficult to manipulate on pricing. Another interesting study is that of US baseball analyst William James, who applied analytical methods to predict which of the players would be most successful in the game, challenging the traditional approach. James's statistically driven approach to correlating early performance to mature performance in players resulted very quickly in a barrage of criticism and rejection of the approach.

There have been numerous claims that data-mining techniques have been used successfully in counterterrorism intelligence analysis, but little has surfaced to support these claims. The idea is that by analyzing the characteristics and profiles of known terrorists, it should be feasible to predict who in a sample of population might also be a terrorist. This is actually a good example of potential pitfalls in the application of such analytical techniques to practical problems, as this type of profiling generates hypotheses, for which there may be good substantiation. The risk is that overly zealous law enforcement personnel, again highly motivated for good reasons, overreact when the individual despite the profile is not a terrorist. There is enough evidence in the media, albeit sensationalized, to suggest this is a real risk. Only careful investigation can prove whether the possibility is a probability. The degree to which a data-mining process supports business goals or scientific objectives of data explorations is much more important than the algorithms and data-mining tools it uses.

1.8   ORGANIZATION OF THIS BOOK

After introducing the basic concepts of data mining in Chapter 1, the rest of the book follows the basic phases of a data-mining process. Chapters 2 and 3 explained common characteristics of raw, large, data sets, and the typical techniques of data preprocessing. The text emphasizes the importance and influence of these initial phases on the final success and quality of data-mining results. Chapter 2 provides basic techniques for transforming raw data, including data sets with missing values and with time-dependent attributes. Outlier analysis is a set of important techniques for preprocessing of messy data and is also explained in this chapter. Chapter 3 deals with reduction of large data sets and introduces efficient methods for reduction of features, values, and cases. When the data set is preprocessed and prepared for mining, a wide spectrum of data-mining techniques is available, and the selection of a technique or techniques depends on the type of application and the data characteristics. In Chapter 4, before

introducing particular data-mining methods, we present the general theoretical back-ground and formalizations applicable for all mining techniques. The essentials of the theory can be summarized with the question: How can one learn from data? The emphasis in Chapter 4 is on statistical learning theory and the different types of learning methods and learning tasks that may be derived from the theory. Also, problems of evaluation and deployment of developed models is discussed in this chapter.

Chapters 5–11 give an overview of common classes of data-mining techniques. Predictive methods are described in Chapters 5–8, while descriptive data mining is given in Chapters 9–11. Selected statistical inference methods are presented in Chapter 5, including Bayesian classifier, predictive and logistic regression, ANOVA analysis, and log-linear models. Chapter 6 summarizes the basic characteristics of the C4.5 algorithm as a representative of logic-based techniques for classification problems. Basic characteristics of the CART approach are also introduced and compared with C4.5 methodology. Chapter 7 discusses the basic components of artificial neural networks and introduces two classes: multilayer perceptrons and competitive networks as illustrative representatives of a neural-network technology. Also, introduction to very popular deep networks is given. Practical applications of a data-mining technology showed that the use of several models in predictive data mining increases the quality of results. This approach is called ensemble learning and basic principles are given in Chapter 8.

Chapter 9 explains the complexity of clustering problems and introduces agglomerative, partitional, and incremental clustering techniques. Different aspects of local modeling in large data sets are addressed in Chapter 10, and common techniques of association rule mining are presented. Web mining and text mining are becoming one of central topics for many researchers, and results of these activities are new algorithms summarized in Chapter 11. There are a number of new topics and recent trends in data mining that are emphasized in last seven years. Some of these topics such as graph mining and temporal, spatial, and distributed data mining are covered in Chapter 12. Important legal restrictions and guidelines and security and privacy aspects of data-mining applications are also introduced in this chapter. Cloud computing is an important technological support for the avalanche of *big data*, while reinforcement learning is opening the modeling approaches in big streaming data. Both topics are also introduced in Chapter 12. Most of the techniques explained in Chapters 13 and 14, about genetic algorithms and fuzzy systems, are maybe not directly applicable in mining large data sets. Recent advances in the field show that these technologies, derived from soft computing, are becoming more important in better representing and computing with data, especially as they are combined with other techniques. Finally, Chapter 15 recognizes the importance of data-mining visualization techniques, especially those for representation of large-dimensional samples, and Chapter 16 gives comprehensive bibliography.

It is our hope that we have succeeded in producing an informative and readable text supplemented with relevant examples and illustrations. All chapters in the book have a set of review problems and reading lists. The author is preparing a solution manual for instructors, who might use the book for undergraduate or graduate classes.

For an in-depth understanding of the various topics covered in this book, we recommend to the reader a selected list of references, given at the end of each chapter. Although most of these references are from various journals, magazines, and conference and workshop proceedings, it is obvious that, as data mining is becoming more mature field, there are many more books available, covering different aspects of data mining and knowledge discovery. Finally, the book has two appendices with useful background information for practical applications of data-mining technology. In Appendix A we provide an overview of most influential journals, conferences, forums, and blogs, as well as a list of commercial and publicly available data-mining tools, while Appendix B presents a number of commercially successful data-mining applications.

The reader should have some knowledge of the basic concepts and terminology associated with data structures and databases. In addition, some background in elementary statistics and machine learning may also be useful, but it is not necessarily required, as the concepts and techniques discussed within the book can be utilized without deeper knowledge of the underlying theory.

1.9   REVIEW QUESTIONS AND PROBLEMS

1. Explain why it is not possible to analyze some large data sets using classical modeling techniques.

2. Do you recognize in your business or academic environment some problems in which the solution can be obtained through classification, regression, or deviation? Give examples and explain.

3. Explain the differences between statistical and machine-learning approaches to the analysis of large data sets.

4. Why are preprocessing and dimensionality reduction important phases in successful data-mining applications?

5. Give examples of data where the time component may be recognized explicitly and other data where the time component is given implicitly in a data organization.

6. Why is it important that the data miner understand data well?

7. Give examples of structured, semi-structured, and unstructured data from everyday situations.

8. Can a set with 50,000 samples be called a large data set? Explain your answer.

9. Enumerate the tasks that a data warehouse may solve as a part of the data-mining process.

10. Many authors include OLAP tools as a standard data-mining tool. Give the arguments for and against this classification.

11. Churn is a concept originating in the telephone industry. How can the same concept apply to banking or to human resources?

12. Describe the concept of actionable information.

13. Go to the Internet and find a data-mining application. Report the decision problem involved, the type of input available, and the value contributed to the organization that used.

14. Determine whether or not each of the following activities is a data-mining task. Discuss your answer.

    (a) Dividing the customers of a company according to their age and sex.
    (b) Classifying the customers of a company according to the level of their debt.
    (c) Analyzing the total sale of a company in the next month based on current month sale.
    (d) Classifying a student database based on a department, sorted based on a student identification number.
    (e) Determining the influence of the number of new University of Louisville students on the stock market value.
    (f) Estimating the future stock price of a company using historical records.
    (g) Monitoring the heart rate of a patient with abnormalities.
    (h) Monitoring seismic waves for earthquake activities.
    (i) Extracting frequencies of a sound wave.
    (j) Predicting the outcome of tossing a pair of dice.

15. Determine which is the best approach (out of three: a–c) for problems 1–7.

    (a) *Supervised learning*
    (b) *Unsupervised clustering*
    (c) *SQL-based data query*

        1. What is the average weekly salary of all female employees under 40 years of age?
        2. Develop a profile for credit card customers likely to carry an average monthly balance of more than $1000.00.
        3. Determine the characteristics of a successful used car salesperson.
        4. What attribute similarities group customers holding one or several insurance policies?
        5. Do meaningful attribute relationships exist in a database containing information about credit card customers?
        6. Do single men play more golf than married men?
        7. Determine whether a credit card transaction is valid or fraudulent.

16. Perform a Google search on "mining text data" and "text data mining."

    (a) Do you get the same top 10 search results?
    (b) What does this tell you about the content component of the ranking heuristic used by search engines?

17. The big data concept is usually defined with four main dimensions—as 4V. If you would like to extend this definition to 5V, what would be this fifth dimension? Give detailed explanations why you would introduce this new dimension. If you do not believe in 5V, check on the Web and you will find extensions up to 8V!

18. If you are preparing to apply for the job of data scientist in one company, what would be your strengths and what are your weaknesses based on your current education and experiences (be honest, it is not official interview for the position)? Do you think you are ready for this job?

1.10   REFERENCES FOR FURTHER STUDY

1. 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.

2. Han, J., M. Kamber, J. Pel, *Data Mining: Concepts and Techniques*, 3rd edition, Morgan Kaufman, San Francisco, 2011.

   This book gives a sound understanding of data-mining principles. The primary orientation of the book is for database practitioners and professionals, with emphasis on OLAP and data warehousing. In-depth analysis of association rules and clustering algorithms is an additional strength of the book. All algorithms are presented in easily understood pseudocode, and they are suitable for use in real-world, large-scale data-mining projects, including advanced applications such as Web mining and text mining.

3. Leskovac J., A. Rajaraman, J. Ullman, *Mining of Massive Datasets*, 2nd edition, Cambridge University Press, 2014.

   At the highest level of description, this book is about data mining. However, it focuses on data mining of very large amounts of data, that is, data so large it does not fit in main memory. Because of the emphasis on size, many of our examples are about the Web or data derived from the Web. Further, the book takes an algorithmic point of view: data mining is about applying algorithms to data, rather than using data to "train" a machine-learning engine of some sort.

4. Olson D., S. Yong, *Introduction to Business Data Mining*, McGraw-Hill, Englewood Cliffs, 2007.

   Introduction to Business Data Mining was developed to introduce students, as opposed to professional practitioners or engineering students, to the fundamental concepts of data mining. Most importantly, this text shows readers how to gather

and analyze large sets of data to gain useful business understanding. The author team has had extensive experience with the quantitative analysis of business as well as with data-mining analysis. They have both taught this material and used their own graduate students to prepare the text's data-mining reports. Using real-world vignettes and their extensive knowledge of this new subject, David Olson and Yong Shi have created a text that demonstrates data-mining processes and techniques needed for business applications.

5. Pouyanfar S., et al, Multimedia Big Data Analytics: A Survey, *ACM Computing Surveys*, Vol. 51, No. 1, April 2018.

   With the proliferation of online services and mobile technologies, the world has stepped into a multimedia big data era. A vast amount of research work has been done in the multimedia area, targeting different aspects of big data analytics, such as the capture, storage, indexing, mining, and retrieval of multimedia big data. However, very few research work provides a complete survey of the whole pine line of the multimedia big data analytics, including the management and analysis of the large amount of data, the challenges and opportunities, and the promising research directions. To serve this purpose, we present this survey, which conducts a comprehensive overview of the state-of-the-art research work on multimedia big data analytics. It also aims to bridge the gap between multimedia challenges and big data solutions by providing the current big data frameworks, their applications in multimedia analyses, the strengths and limitations of the existing methods, and the potential future directions in multimedia big data analytics.

6. Paul Zikopoulos, Chris Eaton, *Understanding Big Data: Analytics for Enterprise Class Hadoop and Streaming Data*, McGraw Hill Professional, 2011.

   Big data represents a new era in data exploration and utilization, and IBM is uniquely positioned to help clients navigate this transformation. This book reveals how IBM is leveraging open-source big data technology, infused with IBM technologies, to deliver a robust, secure, highly available, enterprise-class big data platform. The three defining characteristics of big data—volume, variety, and velocity—are discussed. You will get a primer on Hadoop and how IBM is hardening it for the enterprise and learn when to leverage IBM InfoSphere BigInsights (big data at rest) and IBM InfoSphere Streams (big data in motion) technologies. Industry use cases are also included in this practical guide: (1) learn how IBM hardens Hadoop for enterprise-class scalability and reliability, (2) gain insight into IBM's unique in-motion and at-rest big data analytics platform, and (3) learn tips and tricks for big data use cases and solutions.

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 1.5 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* 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 $\neq$ 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, namely, 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 zero point is defined in the scale. The zero 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 °F does not mean a total absence of temperature. Because of the arbitrary placement of the zero point, the ratio relation does not hold true for variables measured using interval scales. For example, 80 °F does not imply twice as much heat as 40 °F. In contrast, *a ratio scale* has an absolute zero 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 its values defined, using one of two kinds of non-metric 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 is utility, where customer-type values are 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 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, e.g., 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; e.g. the difference between the students ranked 4th and 5th 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, e.g., 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, while the formalization and use of fuzzy values in a data-mining process have been 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 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*

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

Figure 2.1. Variable types with examples.
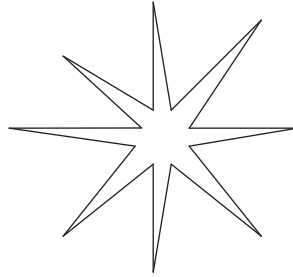
Figure 2.2. High-dimensional data looks conceptually like a porcupine.

data. On the other hand, there are attribute values that change with time and we call this type of data *dynamic* or *temporal data*. The majority of the 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. Besides, these samples are very often high dimensional, 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 counter-intuitive 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 are often the guidelines in 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 (1D) 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 1–100 are values of 1D samples, where the domain of the dimension is [0, 100], then to obtain the same density of samples in a five-dimensional 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
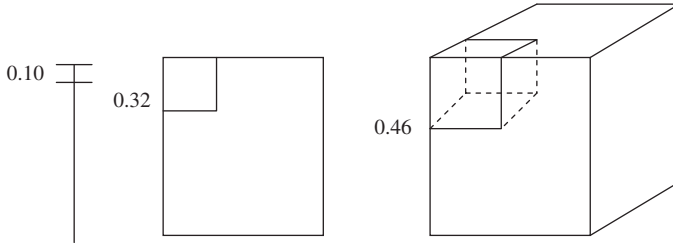
Figure 2.3. Regions enclose 10% of the samples for one-, two-, and three-dimensional space.

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

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 corresponding edge for a two-dimensional (2D) space will be $e_2(0.1) = 0.32$, for a three-dimensional space $e_3(0.1) = 0.46$, and for a 10-dimensional 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

$$D(d,n) = \frac{1}{2} \left(\frac{1}{n} \right)^{1/d}$$

For example, for a 2D space with 10,000 points, the expected distance is $D(2,10,000) = 0.005$ and for a 10-dimensional 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 in making local estimates for high-dimensional samples; we need more and more samples to establish the required data density for
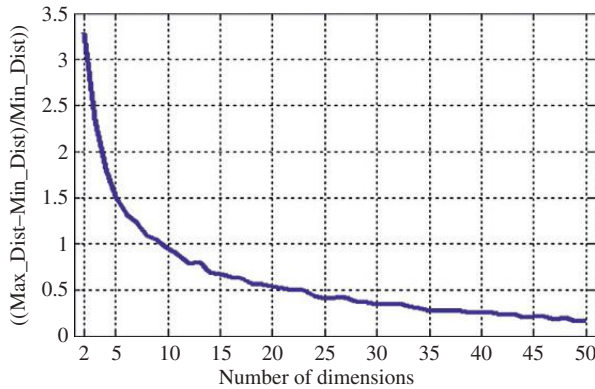
Figure 2.4. With large number of dimensions, the concept of a distance changes its meaning.

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. 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:

$$P_n = \frac{(\text{MAX}-\text{DIST}_n - \text{MIN}-\text{DIST}_n)}{\text{MIN}-\text{DIST}_n}$$

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 with high number of dimensions, parameter P_n is reaching the value of 0. That means maximum and minimum distances becoming so 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 its meaning! When dimensionality of a data set increases, data becomes increasingly sparse with mostly outliers in the space that it occupies. Therefore we have to revisit and reevaluate traditional concepts from statistics: distance, similarity, data distribution, mean, standard deviation, etc.

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 not only be able to model with the data that are presented, but 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 *misrecorded data*, and that is typical in large volumes of data. We have to have mechanisms to discover some of these "unusual" values—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, and a model that goes beyond the various sources of uncertainty and ambiguity in the data all 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 prepared 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 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. To organize 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. To prepare 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, e.g., [−1, 1] or [0, 1]. If the values are not normalized, the distance measures will overweight those features that have, on an average, larger values. There are many ways of normalizing data. Here are three simple and effective normalization techniques:

(a) *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 the scaled value

$$v'(i) = \frac{v(i)}{10^k}$$

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 0.834, and the divisor for all $v(i)$ is 1000 ($k = 3$).

(b) **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 0.15 and 0.25; 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, e.g., [0, 1], we can use the min–max formula

$$v'(i) = \frac{(v(i) - \min(v(i)))}{(\max(v(i)) - \min(v(i)))}$$

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.

(c) **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

$$v*(i) = \frac{(v(i) - \text{mean}(v))}{\text{sd}(v)}$$

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 between

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, e.g., 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 will be 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 the following:

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.
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 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 three-dimensional 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 this book 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 a 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 uni-variate time-series problem, where it is expected that the value of the variable X at a given time 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 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 eleven 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. While 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 $t(n - i), \ldots, t(n)$, 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.

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					Next 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	$t(3)$	$t(4)$	$t(5)$	$t(6)$	$t(7)$	$t(10)$

In general, the further out 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-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 $t(i+1) \geq$ 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 more 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 of 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

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

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

$$MA(i,m) = t(i) = mean(i) + error$$

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

$$EMA(i,m) = p \times t(i) + (1-p) \times EMA(i-1,m-1)$$
$$EMA(i,1) = t(i)$$

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:

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

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

A moving average 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 to 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 MA, 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.

(a)

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

(b)

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

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

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.

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 might know that the patient was still alive after 5 years, but we 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 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, or 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. 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, while detecting fraudulent credit card transactions in 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 ($k \ll n$). The problem of defining outliers is nontrivial, especially in multidimensional samples. 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 (1D), scatter plot (2D), and spin plot (3D), and they will be explained in the following chapters. Data-visualization methods that are useful in outlier detection for data with one to three dimensions are much weaker in multidimensional data because of a lack of adequate visualization methodologies for *n*-dimensional spaces. An illustrative example of a visualization of 2D samples and visual detection of outliers is given in Figures 2.6 and 2.7. The main limitations of the approach are time-consuming process and subjective nature of outlier detection.

Statistical-based outlier detection methods can be divided between *univariate methods*, proposed in earlier works in this field, and *multivariate methods* that usually form most of the current body of research. Statistical methods either assume a known underlying distribution of the observations or, at least, 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.
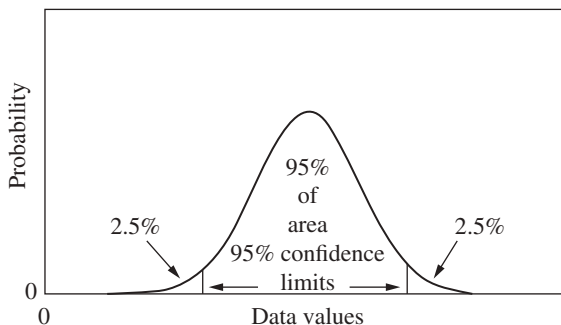
Figure 2.6. Outliers for univariate data based on mean value and standard deviation.
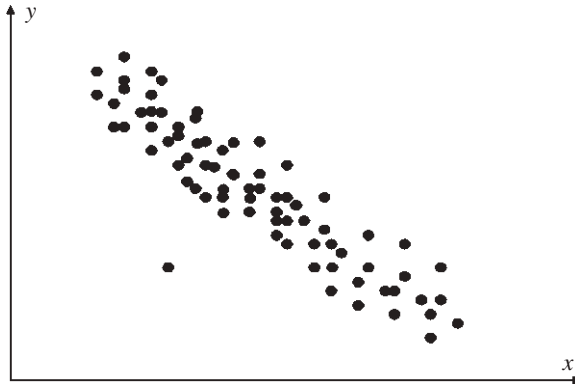
Figure 2.7. Two-dimensional data set with one outlying sample.

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 1D 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 it is represented 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 twenty different values

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

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 than zero) 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. Illustrative example is given in Figure 2.7 where analysis of each dimension separately will not give any outlier, while analysis of 2D 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. Therefore, 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 $n \gg p$), denote the sample mean vector by x_n and the sample covariance matrix by V_n, where

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

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

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

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 for outlier detection 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 2D samples S, where the requirements for outliers are the values of thresholds, $p \geq 4$ and $d \geq :$

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

The table of Euclidean 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.

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. Number of Samples p with the Distance Greater Than d from a Given Sample

Sample	p
s_1	2
s_2	1
s_3	5
s_4	2
s_5	5
s_6	3
s_7	2
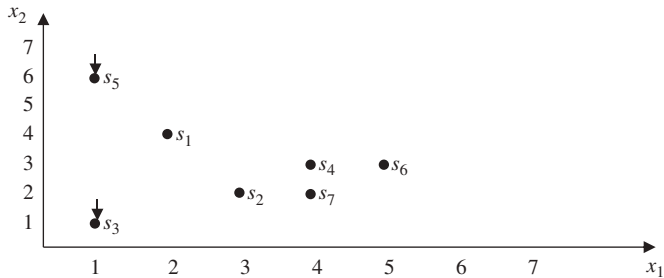
Figure 2.8. Visualization of two-dimensional data set for outlier detection.

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 2D 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.

There is a possibility for reducing complexity of the algorithm by partitioning the data into n-dimensional 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 (nondeterministic polynomial time, 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 BIRCH and DBSCAN, 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 outlier detection. For example, in 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 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.
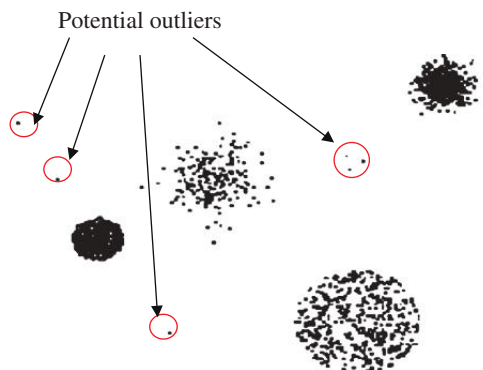
Figure 2.9. Determining outliers through clustering.

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 is 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 six-dimensional 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 six-dimensional space?

5. Derive the formula for min–max normalization of data on [−1, 1] interval.

6. Given one-dimensional 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].
  (d) Standard deviation normalization.
  (e) Compare the results of previous normalizations and discuss the advantages and disadvantages of different techniques.

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

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

Present the new data set when the rounding is performed to the precision of:
  (a) 0.1
  (b) 1

8. Given a set of four-dimensional 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-hour time-dependent data set X is collected as a training data set to predict values 3 hours 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 hours? What is the number of samples?
   (ii) the window width 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-hour moving averages (MA).
(c) Plot time-dependent variable X and its 4-h exponential moving average (EMA).

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

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

(a) Find the outliers in the set C using standard statistical parameters mean and variance.
(b) 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 three-dimensional 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 exponential moving averages (EMA) instead of moving averages (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 two-dimensional samples is given in the following table. Check if we do 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 three-dimensional 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 outlier detection*, find the outliers in the set

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

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

19. (a) Derive the formula for *min–max normalization* of data for the range $[-1, 1]$.
    (b) What will be normalized values (using this formula) for the data set X?

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

20. Every attribute in six-dimensional 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 six-dimensional 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. (Example: Age in Years. Answer is: 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 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.

22. What are the major challenges of mining a huge amount of data (such as billions of tuples) in comparison with mining a small amount of data (such as a few hundred tuple data set)?

23. Use the two methods to *normalize* the following set of data: 200, 300, 400, 600, 1000.

(a) min–max normalization by setting min = 0 and max = 1.

(b) Standard deviation normalization (often used as *z*-score).

2.8 REFERENCES FOR FURTHER STUDY

1. 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 analysis fits together when applied to real-world data-mining problems.

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

   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.

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

   The author presents several methods for outlier detection while distinguishing between univariate and multivariate techniques and parametric and 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.

4. Zaki M. J., W. Meira, *Data Mining And Analysis: Fundamental Concepts and Algorithms*, Cambridge University Press, New York, 2014.

The fundamental algorithms in data mining and analysis form the basis for the emerging field of data science, which includes automated methods to analyze patterns and models for all kinds of data, with applications ranging from scientific discovery to business intelligence and analytics. This textbook for senior undergraduate and graduate data mining courses provides a broad yet in-depth overview of data mining, integrating related concepts from machine learning and statistics. The main parts of the book include exploratory data analysis, frequent pattern mining, clustering, and classification. The book lays the basic foundations of these tasks, and it also covers cutting-edge topics like kernel methods, high-dimensional data analysis, and complex graphs and networks. It integrates concepts from related disciplines like machine learning and statistics and is also ideal for a course on data analysis. Most of the prerequisite material is covered in the text, especially on linear algebra and probability and statistics.

5. Agraval C., *Outliers Analysis*, 2nd edition, Springer, New York, 2016.

The problem of outlier detection finds applications in numerous domains, where it is desirable to determine interesting and unusual events in the underlying generating process. The core of all outlier detection methods is the creation of a probabilistic, statistical, or algorithmic model that characterizes the normal data. The deviations from this model are used to identify the outliers. A good domain-specific knowledge of the underlying data is often crucial designing simple and accurate models that do not overfit the underlying data. The problem of outlier detection becomes especially challenging when significant relationships exist among the different data points. This is the case for time-series and network data in which the patterns in the relationships among the data points (whether temporal or structural) play the key role in defining the outliers. Outlier analysis has tremendous scope for further research, especially in the area of structural and temporal analysis.

3

DATA REDUCTION

Chapter Objectives

- Identify the differences in dimensionality reduction based on features, cases, and reduction of value techniques.
- Explain the advantages of data reduction in the preprocessing phase of a data-mining process.
- Understand the basic principles of feature selection and feature composition tasks using corresponding statistical methods.
- Apply and compare entropy-based technique and principal component analysis for feature ranking.
- Understand the basic principles and implement ChiMerge and Bin-based techniques for reduction of discrete values.
- Distinguish approaches in cases where reduction is based on incremental and average samples.

For small or moderate data sets, the preprocessing steps mentioned in the previous chapter in preparation for data mining are usually enough. For really large data sets, there is an increased likelihood that an intermediate, additional step—data reduction—should be performed prior to applying the data-mining techniques. While large data sets have the potential for better mining results, there is no guarantee that they will yield better knowledge than small data sets. Given multidimensional data, a central question is whether it can be determined, prior to searching for all data-mining solutions in all dimensions, that the method has exhausted its potential for mining and discovery in a reduced data set. More commonly, a general solution may be deduced from a subset of available features or cases, and it will remain the same even when the search space is enlarged.

The main theme for simplifying the data in this step is dimension reduction, and the main question is whether some of these prepared and preprocessed data can be discarded without sacrificing the quality of results. There is one additional question about techniques for data reduction: Can the prepared data be reviewed and a subset found in a reasonable amount of time and space? If the complexity of algorithms for data reduction increases exponentially, then there is little to gain in reducing dimensions in big data. In this chapter, we will present basic and relatively efficient techniques for dimension reduction applicable to different data-mining problems.

3.1   DIMENSIONS OF LARGE DATA SETS

The choice of data representation and selection, reduction, or transformation of features is probably the most important issue that determines the quality of a data-mining solution. Besides influencing the nature of a data-mining algorithm, it can determine whether the problem is solvable at all or how powerful the resulting model of data mining is. A large number of features can make available samples of data relatively insufficient for mining. In practice, the number of features can be as many as several hundreds. If we have only a few hundred samples for analysis, dimensionality reduction is required in order for any reliable model to be mined or to be of any practical use. On the other hand, data overload, because of high dimensionality, can make some data-mining algorithms non-applicable, and the only solution is again a reduction of data dimensions. For example, a typical classification task is to separate healthy patients from cancer patients, based on their gene expression "profile." Usually fewer than 100 samples (patients' records) are available altogether for training and testing. But the number of features in the raw data ranges from 6,000 to 60,000. Some initial filtering usually brings the number of features to a few thousand; still it is a huge number and additional reduction is necessary. The three main dimensions of preprocessed data sets, usually represented in the form of flat files, are *columns* (features), *rows* (cases or samples), and *values* of the features.

Therefore, the three basic operations in a data-reduction process are delete a column, delete a row, and reduce the number of values in a column (smooth a feature). These operations attempt to preserve the character of the original data by deleting data

that are nonessential. There are other operations that reduce dimensions, but the new data are unrecognizable when compared with the original data set, and these operations are mentioned here just briefly because they are highly application dependent. One approach is the replacement of a set of initial features with a new composite feature. For example, if samples in a data set have two features, person's height and person's weight, it is possible for some applications in the medical domain to replace these two features with only one body mass index, which is proportional to the quotient of the initial two features. Final reduction of data does not reduce the quality of results; in some applications, the results of data mining are even improved.

Performing standard data-reduction operations (deleting rows, columns, or values) as a preparation for data mining, we need to know what we gain and/or lose with these activities. The overall comparison involves the following parameters for analysis:

1. *Computing time*—Simpler data, a result of the data-reduction process, can hopefully lead to a reduction in the time taken for data mining. In most cases, we cannot afford to spend too much time on the data-preprocessing phases, including a reduction of data dimensions, although the more time we spend in preparation the better the outcome.

2. *Predictive/descriptive accuracy*—This is the dominant measure for most data-mining models since it measures how well the data is summarized and generalized into the model. We generally expect that by using only relevant features, a data-mining algorithm can not only learn faster but also with higher accuracy. Irrelevant data may mislead a learning process and a final model, while redundant data may complicate the task of learning and cause unexpected data-mining results.

3. *Representation of the data-mining model*—The simplicity of representation, obtained usually with data reduction, often implies that a model can be better understood. The simplicity of the induced model and other results depends on its representation. Therefore, if the simplicity of representation improves, a relatively small decrease in accuracy may be tolerable. The need for a balanced view between accuracy and simplicity is necessary, and dimensionality reduction is one of the mechanisms for obtaining this balance.

It would be ideal if we could achieve reduced time, improved accuracy, and simplified representation at the same time, using dimensionality reduction. More often, however, we gain in some and lose in others and balance between them according to the application at hand. It is well known that no single data-reduction method can be best suited for all applications. A decision about method selection is based on available knowledge about an application (relevant data, noise data, metadata, correlated features) and required time constraints for the final solution.

Algorithms that perform all basic operations for data reduction are not simple, especially when they are applied to large data sets. Therefore, it is useful to enumerate the desired properties of these algorithms before giving their detailed descriptions.

Recommended characteristics of data-reduction algorithms that may be guidelines for designers of these techniques are as follows:

1. *Measurable quality*—The quality of approximated results using a reduced data set can be determined precisely.
2. *Recognizable quality*—The quality of approximated results can be easily determined at run time of the data-reduction algorithm, before application of any data-mining procedure.
3. *Monotonicity*—The algorithms are usually iterative, and the quality of results is a nondecreasing function of time and input data quality.
4. *Consistency*—The quality of results is correlated with computation time and input data quality.
5. *Diminishing returns*—The improvement in the solution is large in the early stages (iterations) of the computation, and it diminishes over time.
6. *Interruptability*—The algorithm can be stopped at any time and provide some answers.
7. *Preemptability*—The algorithm can be suspended and resumed with minimal overhead.

3.2   FEATURES REDUCTION

Most of the real-world data-mining applications are characterized by high-dimensional data, where not all of the features are important. For example, high-dimensional data (i.e., data sets with hundreds or even thousands of features) can contain a lot of irrelevant, noisy information that may greatly degrade the performance of data-mining process. Even state-of-art data-mining algorithms cannot overcome the presence of large number of weakly relevant and redundant features. This is usually attributed to the "curse of dimensionality" or to the fact that irrelevant features decrease the signal-to-noise ratio. In addition, many algorithms become computationally intractable when the dimensionality is high.

Data such as images, text, and multimedia are high dimensional in nature, and this dimensionality of data poses a challenge to data-mining tasks. Researchers have found that reducing the dimensionality of data results in a faster computation while maintaining reasonable accuracy. In the presence of many irrelevant features, mining algorithms tend to overfit the model. Therefore, many features can be removed without performance deterioration in the mining process.

When we are talking about data quality and improved performances of reduced data sets, we can see that this issue is not only about noisy or contaminated data (problems mainly solved in the preprocessing phase) but also about irrelevant, correlated, and redundant data. Recall that data with corresponding features are not usually collected solely for data-mining purposes. Therefore, dealing with relevant features alone can be far more effective and efficient. Basically, we want to choose features that are

relevant to our data-mining application in order to achieve maximum performance with the minimum measurement and processing effort. A feature-reduction process should result in:

1. less data so that the data-mining algorithm can learn faster,
2. higher accuracy of a data-mining process so that the model can generalize better from data,
3. simple results of a data-mining process so that they are easier to understand and use, and
4. fewer features so that in the next round of data collection, a saving can be made by removing redundant or irrelevant features.

Let us start our detailed analysis of possible column-reduction techniques, where features are eliminated from the data set based on a given criterion. To address the curse of dimensionality, dimensionality reduction techniques are proposed as a data-preprocessing step. This process identifies a suitable low-dimensional representation of original data. Reducing the dimensionality improves the computational efficiency and accuracy of the data analysis. Also, it improves comprehensibility of a data-mining model. Proposed techniques are classified as supervised and unsupervised techniques based on the type of a learning process. Supervised algorithms need a training set with the output class label information to learn the lower-dimensional representation according to some criteria. Unsupervised approaches project the original data to a new lower-dimensional space without utilizing the label (class) information. Dimensionality reduction techniques function either by transforming the existing features to a new reduced set of features or by selecting a subset of the existing features. Therefore, two standard tasks are associated with producing a reduced set of features, and they are classified as follows:

1. *Feature selection*—Based on the knowledge of the application domain and the goals of the mining effort, the human analyst may select a subset of the features found in the initial data set. The process of feature selection can be manual or supported by some automated procedures.

   Roughly speaking, feature selection methods are applied in one of the three conceptual frameworks: the filter model, the wrapper model, and embedded methods. These three basic families differ in how the learning algorithm is incorporated in evaluating and selecting features. In the filter model the selection of features is done as a preprocessing activity, without trying to optimize the performance of any specific data-mining technique directly. This is usually achieved through an (ad hoc) evaluation function using a search method in order to select a subset of features that maximizes this function. Performing an exhaustive search is usually intractable due to the large number of initial features. Therefore, different methods may apply a variety of search heuristics. Wrapper methods select features by "wrapping" the search around the selected

learning algorithm and evaluate feature subsets based on the learning performance of the data-mining technique for each candidate feature subset. The main drawback of this approach is its computational complexity. Main characteristics of both approaches are given in Figure 3.1. Finally, embedded methods incorporate feature search and the learning algorithm into a single optimization problem formulation. When the number of samples and dimensions becomes very large, the filter approach is usually a choice due to its computational efficiency and neutral bias toward any learning methodology.
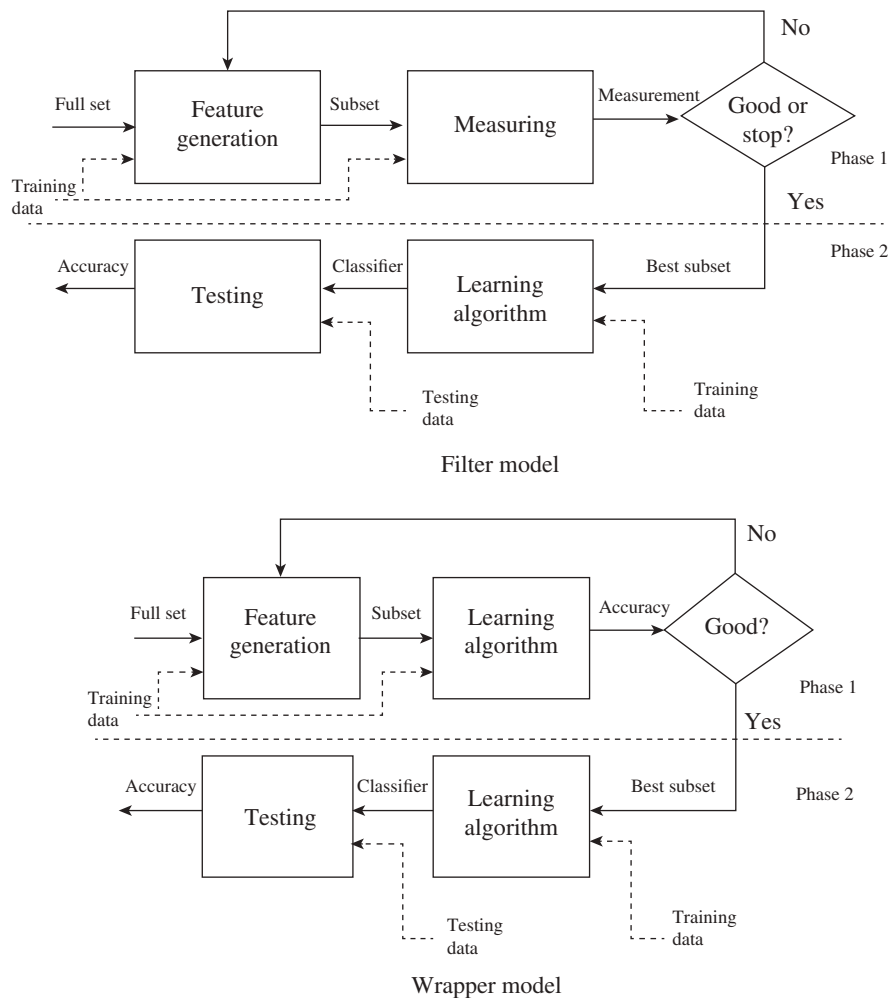
Figure 3.1. Feature selection approaches.

2. *Feature extraction/transformation*—There are transformations of data that can have a surprisingly strong impact on the results of data-mining methods. In this sense, the composition and/or transformation of features is a greater determining factor in the quality of data-mining results. In most instances, feature composition is dependent on knowledge of the application, and an interdisciplinary approach to feature composition tasks produces significant improvements in the preparation of data. Still, some general-purpose techniques, such as principal component analysis (PCA), are often used and with a high success.

Feature selection is typically preferred over extraction/transformation when one wishes to keep the original meaning of the features and wishes to determine which of those features are important. Moreover, once features are selected, only these features need to be calculated or collected, whereas, in transformation-based methods all input features are still needed to obtain the reduced dimension.

3.2.1 Feature Selection

In data mining, feature selection, also known as variable selection, feature reduction, attribute selection, or variable subset selection, is a set of techniques, which selects a subset of relevant features for building robust learning models by removing most irrelevant and redundant features from the data. The objective of feature selection is threefold: improving the performance of a data mining model, providing faster and more cost-effective learning process, and providing a better understanding of the underlying process that generates the data. Feature selection algorithms typically fall into two categories: feature ranking and subset selection. Feature ranking ranks all features by a specified metric and eliminates all features that do not achieve an adequate score. Subset selection, on the other hand, searches the set of all features for the optimal subset where features in the selected subset are not ranked. We have to be aware that different feature selection methods may give different reduced features sets.

In the feature ranking algorithm, one can expect a ranked list of features that are ordered according to a specific evaluation measure. A measure can be based on accuracy of available data, consistency, information content, distances between samples, and, finally, statistical dependencies between features. These algorithms do not tell you what the minimum set of features for further analysis is; they indicate only the relevance of a feature compared with others. Minimum subset algorithms, on the other hand, return a minimum feature subset, and no differences are made among features in the subset—all have the same ranking. The features in the subset are relevant for the mining process; the others are irrelevant. In both types of algorithms, it is important to establish a feature-evaluation scheme: the way in which the features are evaluated and then ranked or added to the selected subset.

Feature selection in general can be viewed as a search problem, with each state in the search space specifying a subset of the possible features. If, for example, a data set has three features $\{A_1, A_2, A_3\}$, and in the process of selecting features, the presence of

a feature is coded with 1 and its absence with 0, then there should be a total of 2^3 reduced feature subsets coded with $\{0, 0, 0\}$, $\{1, 0, 0\}$, $\{0, 1, 0\}$, $\{0, 0, 1\}$, $\{1, 1, 0\}$, $\{1, 0, 1\}$, $\{0, 1, 1\}$, and $\{1, 1, 1\}$. The problem of feature selection is relatively trivial if the search space is small, since we can analyze all subsets in any order and a search will get completed in a short time. However, the search space is usually not small. It is 2^N where the number of dimensions N in typical data-mining applications is large ($N > 20$). This makes the starting point and the search strategy very important. An exhaustive search of all subsets of features very often is replaced with some heuristic search procedures. Using knowledge of the problem, these procedures find near-optimal subsets of features that further improve the quality of the data-mining process.

The objective of feature selection is to find a subset of features with data-mining performances comparable with the full set of features. Given a set of features m, the number of subsets to be evaluated as candidates for column reduction is finite but still very large for iterative analysis through all cases. For practical reasons, an optimal search is not feasible, and simplifications are made to produce reasonable, acceptable, and timely results. If the reduction task is to create a subset, one possibility—the so-called bottom-up approach—starts with an empty set and fills it in by choosing the most relevant features from the initial set of features. This process is based on some heuristic criteria for a feature evaluation. The top-down approach, on the other hand, begins with a full set of original features and then removes one by one those that are shown as irrelevant based on the selected heuristic evaluation measure. Additional approximations to the optimal approach are as follows:

1. To examine only promising subsets of features where promising subsets are usually obtained heuristically. This provides enough room for exploration of competing alternatives.
2. To substitute computationally simple distance measures for the error measures. This approximation will reduce the computation time yet give satisfactory results for comparison of subset alternatives.
3. To select features based only on subsets of large amounts of data. The subsequent steps of data mining will be applied on the whole set.

The application of feature selection and reduction of data dimensionality may be used in all phases of the data-mining process for successful knowledge discovery. It has to be started in the preprocessing phase, but, on many occasions, feature selection and reduction is a part of the data-mining algorithm, even it is applied in postprocessing for better evaluation and consolidation of obtained results.

Let us return to the promising subsets of features. One possible technique for feature selection is based on comparison of *means* and *variances*. To summarize the key characteristics of the distribution of values for a given feature, it is necessary to compute the mean value and the corresponding variance. The main weakness in this approach is that the distribution for the feature is not known. If it is assumed to be a normal curve, the statistics can work out very well, but this may in fact be a poor assumption. Without knowing the shape of the distribution curve, the means and

variances are viewed as heuristics for feature selection, not exact, mathematical modeling tools.

In general, if one feature describes different classes of entities, samples of two different classes can be examined. The means of feature values are normalized by its variances and then compared. If the means are far apart, interest in a feature increases; it has potential, in terms of its use in distinguishing between two classes. If the means are indistinguishable, interest wanes in that feature. It is a heuristic, non-optimal approach to feature selection, but it is consistent with practical experience in many data-mining applications in the triage of features. Next, equations formalize the test, where A and B are sets of feature values measured for two different classes and n_1 and n_2 are the corresponding number of samples:

$$SE(A-B) = \sqrt{\left(\frac{var(A)}{n_1} + \frac{var(B)}{n_2} \right)}$$

$$TEST : \frac{|\,mean\,(A) - mean\,(B)\,|}{SE(A-B)} > threshold\text{-}value$$

The mean of a feature is compared between two classes without taking into consideration relationship to other features. In this approach to feature selection, we assumed a priori that the given feature is independent of the others. A comparison of means is a natural fit to classification problems. For the purpose of feature selection, a regression problem can be considered as a pseudo-classification problem. For k classes, k pairwise comparisons can be made, comparing each class to its complement. A feature is retained if it is significant for any of the pairwise comparisons.

We can analyze this approach in feature selection through one example. A simple data set is given in Table 3.1 with two input features X and Y and an additional output feature C that classifies samples into two classes A and B. It is necessary to decide whether the features X and Y are candidates for reduction or not. Suppose that the threshold value of the applied test is 0.5.

First, we need to compute a mean value and a variance for both classes and both features X and Y. The analyzed subsets of the feature's values are

TABLE 3.1. Data Set with Three Features

X	Y	C
0.3	0.7	A
0.2	0.9	B
0.6	0.6	A
0.5	0.5	A
0.7	0.7	B
0.4	0.9	B

$X_A = \{0.3, 0.6, 0.5\}, \quad X_B = \{0.2, 0.7, 0.4\}, \quad Y_A = \{0.7, 0.6, 0.5\}, \quad \text{and} \quad Y_B = \{0.9, 0.7, 0.9\}$

and the results of applied tests are

$$\text{SE}(X_A - X_B) = \sqrt{\left(\frac{\text{var}(X_A)}{n_1} + \frac{\text{var}(X_B)}{n_2}\right)} = \sqrt{\left(\frac{0.0233}{3} + \frac{0.06333}{3}\right)} = 0.1699$$

$$\text{SE}(Y_A - Y_B) = \sqrt{\left(\frac{\text{var}(Y_A)}{n_1} + \frac{\text{var}(Y_B)}{n_2}\right)} = \sqrt{\left(\frac{0.01}{3} + \frac{0.0133}{3}\right)} = 0.0875$$

$$\frac{|\text{mean}(X_A) - \text{mean}(X_B)|}{\text{SE}(X_A - X_B)} = \frac{|0.4667 - 0.4333|}{0.1699} = 0.1961 < 0.5$$

$$\frac{|\text{mean}(Y_A) - \text{mean}(Y_B)|}{\text{SE}(Y_A - Y_B)} = \frac{|0.6 - 0.8333|}{0.0875} = 2.6667 > 0.5$$

This analysis shows that X is a candidate for reduction because its mean values are close and, therefore, the final test is below the threshold value. On the other hand, the test for feature Y is significantly above the threshold value; this feature is not a candidate for reduction because it has the potential to be a distinguishing feature between two classes.

Similar idea for features ranking is given in the algorithm, which is based on correlation criteria. Let us consider first the prediction of a continuous outcome y. The Pearson correlation coefficient is defined as

$$\mathcal{R}(i) = \frac{\text{cov}(X_i, Y)}{\sqrt{\text{var}(X_i)\text{var}(Y)}}$$

where cov designates the covariance and var the variance. The estimate of $R(i)$ for the given data set with samples' inputs $x_{k,j}$ and outputs y_k, is defined by

$$R(i) = \frac{\sum_{k=1}^{m}(x_{k,i} - \bar{x}_i)(y_k - \bar{y})}{\sqrt{\sum_{k=1}^{m}(x_{k,i} - \bar{x}_i)^2 \sum_{k=1}^{m}(y_k - \bar{y})^2}}$$

where the bar notation stands for an average over the index k (set of all samples). Using $R(i)^2$ as a variable ranking criterion enforces a ranking according to goodness of linear fit of individual variables. Correlation criteria such as $R(i)^2$ can only detect linear dependencies between input features and target or output feature (variable). One common criticism of variable ranking is that it leads to the selection of a redundant subset. The same performance could possibly be achieved with a smaller subset of complementary variables. Still, one may wonder whether deleting presumably redundant variables can result in a performance gain.

Practical experiments show that noise reduction and consequently better model estimation may be obtained by adding features that are presumably redundant. Therefore, we have to be very careful in the preprocessing analysis. Yes, perfectly correlated variables are truly redundant in the sense that no additional information is gained by adding them. But even variables with relatively high correlation (or anticorrelation) do not guarantee absence of variables' complementarity. We can find cases where a feature looks like completely useless by itself, and it is ranked very low, but it can provide significant information to the model and performance improvement when taken with others. These features by itself may have little correlation with the output, target concept, but when it is combined with some other features; they can be strongly correlated with the target feature. Unintentional removal of these features can result in poor mining performance.

The previous simple methods test features separately. Several features may be useful when considered separately, but they may be redundant in their predictive ability. If the features are examined collectively, instead of independently, additional information can be obtained about their characteristics and mutual relations. Assuming normal distributions of values, it is possible to describe an efficient technique for selecting subsets of features. Two descriptors characterize a multivariate normal distribution:

1. M—A vector of the m feature means.
2. C—An $m \times m$ covariance matrix of the means, where $C_{i,i}$ are simply the variance of feature i and $C_{i,j}$ terms are correlations between each pair of features:

$$C_{i,j} = \frac{1}{n} \sum_{k=1}^{n} (((v(k,i) - m(i)) * (v(k,j) - m(j))))$$

where $v(k,i)$ and $v(k,j)$ are the values of features indexed with i and j; $m(i)$ and $m(j)$ are feature means; and n is the number of dimensions

These two descriptors, M and C, provide a basis for detecting redundancies in a set of features. If two classes exist in a data set, then the heuristic measure, DM, for filtering features that separate the two classes is defined as

$$DM = (M_1 - M_2)(C_1 + C_2)^{-1}(M_1 - M_2)^T$$

where M_1 and C_1 are descriptors of samples for the first class and M_2 and C_2 for the second class. Given the target of k best features, all subsets of k from m features must be evaluated to find the subset with the largest DM. With large data sets that have features, this can be a huge search space, and alternative heuristic methods should be considered. One of these methods selects and ranks features based on an entropy measure. Detailed explanations are given in Section 3.4. The other heuristic approach, explained in the following text, is based on a combined correlation and covariance analyses and ranking of all features.

Existing efficient feature selection algorithms usually rank features under assumption of feature independence. Under this framework, features are ranked as relevant mainly based on their individually high correlations with the output feature. Because of the irreducible nature of feature interactions, these algorithms cannot select interacting features. In principle, it can be shown that a feature is relevant due to two reasons: (1) it is strongly correlated with the target feature; or (2) it forms a features' subset and the subset is strongly correlated with the target. A heuristic approach is developed to analyze features of type (2) in the selection process.

In the first part of a selection process, the features are ranked in descending order based on their correlation values with output using previously defined technique. We may assume that a set of features S can be divided into subset $S1$ including relevant features and subset $S2$ containing irrelevant ones. Heuristically, critical for removal are features in $S2$ first, while features in $S1$ are more likely to remain in the final set of selected features.

In the second part, features are evaluated one by one starting from the end of the $S2$ ranked feature list. The monotonic property suggests that the backward elimination search strategy fits best in feature selection. That is, one can start with the full feature set and successively eliminating features one at a time from the bottom of the ranked list if their interaction does not contribute to better correlation with output. The criterion could be, for example, based on covariance matrix. If a feature, together with previously selected features, shows influence on the output with less than threshold value (it could be expressed through some covariance matrix factor!), the feature is removed, and otherwise it is selected. Backward elimination allows every feature to be evaluated with the features it may interact with. The algorithm repeats until all features in the $S2$ list are checked.

3.2.2   Feature Extraction

The art of data mining starts with the design of appropriate data representations. Better performance is often achieved using features derived from the original input. Building a feature representation is an opportunity to incorporate domain knowledge into the data and can be very application specific. Transforming the input set into a new, reduced set of features is called *feature extraction*. If the features extracted are carefully chosen, it is expected that the new features set will extract the relevant information from the input data in order to perform the desired data mining task using this reduced representation.

Feature transformation techniques aim to reduce the dimensionality of data to a small number of dimensions, which are linear or nonlinear combinations of the original dimensions. Therefore, we distinguish two major types of dimension reduction methods: linear and nonlinear. Linear techniques result in k new derived features instead of initial p ($k \ll p$). Components of the new feature are a linear combination of the original features:

$$s_i = w_{i,1}x_1 + \cdots + w_{i,p}x_p \quad \text{for } i = 1, \ldots, k;$$

or in a matrix form

$$s = W x$$

where $W_{k \times p}$ is the linear transformation weight matrix. Such linear techniques are simpler and easier to implement than more recent methods considering nonlinear transforms.

In general, the process reduces feature dimensions by combining features instead of by deleting them. It results in a new set of fewer features with totally new values. One well-known approach is merging features by *principal components*. The features are examined collectively, merged, and transformed into a new set of features that, it is hoped, will retain the original information content in a reduced form. The basic transformation is linear. Given p features, they can be transformed into a single new feature F' by the simple application of weights:

$$F' = \sum_{j=1}^{p} w(j) \cdot f(j)$$

Most likely, a single set of weights $w(j)$ will not be adequate transformation for complex multidimensional data set, and up to p transformations are generated. Each vector of p features combined by weights w is called a principal component, and it defines a new transformed feature. The first vector of m weights is expected to be the strongest, and the remaining vectors are ranked according to their expected usefulness in reconstructing the original data. Eliminating the bottom-ranked transformation will reduce dimensions. The complexity of computation increases significantly with the number of features. The main weakness of the method is that it makes an advance assumption to a linear model that maximizes the variance of features. Formalization of principal components analysis (PCA) and the basic steps of the corresponding algorithm for selecting features are given in Section 3.5.

Examples of additional methodologies in feature extraction include factor analysis (FA), independent component analysis (ICA), and multidimensional scaling (MDS). Probably the last one is the most popular, and it represents the basis for some new, recently developed techniques. Given n samples in a p-dimensional space and an $n \times n$ matrix of distance measures among the samples, multidimensional scaling (MDS) produces a k-dimensional ($k \ll p$) representation of the items such that the distances among the points in the new space reflect the distances in the original data. A variety of distance measures may be used in the technique, and the main characteristic for all these measures is the more similar two samples are, the smaller their distance is. Popular distance measures include the Euclidean distance ($L2$ norm), the Manhattan distance ($L1$, absolute norm), and the maximum norm, and more detail about these measures and their applications is given in Chapter 9. MDS has been typically used to transform the data into two or three dimensions, visualizing the result to uncover hidden structure in the data. A rule of thumb to determine the maximum number of k is to ensure that there are at least twice as many pairs of samples than the number of parameters to be estimated, resulting in $p \geq k + 1$. Results of the MDS technique are indeterminate with respect to translation, rotation, and reflection of data.

PCA and multidimensional scaling (MDS) are both simple methods for linear dimensionality reduction, where an alternative to MDS is FastMap, a computationally efficient algorithm. The other variant, *Isomap*, has also emerged as a powerful technique for nonlinear dimensionality reduction, and it is primary graph-based method.

Isomap is based on computing the low-dimensional representation of a high-dimensional data set that most faithfully preserves the pairwise distances between input samples as measured along geodesic distance (details about geodesic are given in Chapter 12, the section about graph mining). The algorithm can be understood as a variant of MDS in which estimates of geodesic distances are substituted for standard Euclidean distances.

The algorithm has three steps. The first step is to compute the k-nearest neighbors of each input sample and to construct a graph whose vertices represent input samples and whose (undirected) edges connect k-nearest neighbors. The edges are then assigned weights based on the Euclidean distance between nearest neighbors. The second step is to compute the pairwise distances between all nodes (i, j) along shortest paths through the graph. This can be done using well-known Dijkstra's algorithm with complexity $O(n^2 \log n + n^2 k)$. Finally, in the third step, the pairwise distances are fed as input to MDS to determine new reduced set of features.

With the size of data getting bigger and bigger, all feature selection (and reduction) methods also face a problem of oversized data because of a computer's limited resources. But do we really need so much data for selecting features as an initial process in data mining? Or can we settle for less data? We know that some portion of a huge data set can represent it reasonably well. The point is which portion and how large should it be. Instead of looking for the right portion, we can randomly select a part, P, of a data set, use that portion to find the subset of features that satisfy the evaluation criteria, and test this subset on a different part of the data. The results of this test will show whether the task has been successfully accomplished. If an inconsistency is found, we shall have to repeat the process with a slightly enlarged portion of the initial data set. What should be the initial size of the data subset P? Intuitively, we know that its size should not be too small or too large. A simple way to get out of this dilemma is to choose a percentage of data, say, 10%. The right percentage can be determined experimentally.

What are the results of a feature-reduction process, and why do we need this process for every specific application? The purposes vary, depending upon the problem on hand, but, generally, we want:

1. to *improve performances* of the model-generation process and the resulting model itself (typical criteria are speed of learning, predictive accuracy, and simplicity of the model);
2. to *reduce dimensionality* of the model without reduction of its quality through:
   (a) Elimination of irrelevant features,
   (b) Detection and elimination of redundant data and features,

(c) Identification of highly correlated features, and

(d) Extraction of independent features that determine the model; and

3. to help the user *visualize* alternative results, which have fewer dimensions, to improve decision-making.

3.3 RELIEF ALGORITHM

Relief is a feature weight-based algorithm for feature selection inspired by so-called instance-based learning. It relies on relevance evaluation of each feature given in a training data set. The main idea of *Relief* is to compute a ranking score for every feature indicating how well this feature separates neighboring samples. The authors of the Relief algorithm, Kira and Rendell, proved that the ranking score is large for relevant features and small for irrelevant ones.

The core of the *Relief* algorithm is to estimate the quality of features according to how well their values distinguish between samples close to each other. Given training data *S*, the algorithm randomly selects subset of sample size *m*, where m is a user-defined parameter. *Relief* analyzes each feature based on a selected subset of samples. For each randomly selected sample *X* from a training data set, *Relief* searches for its two nearest neighbors: one from the same class, called nearest hit *H*, and the other one from a different class, called nearest miss *M*. An example for two-dimensional data is given in Figure 3.2.
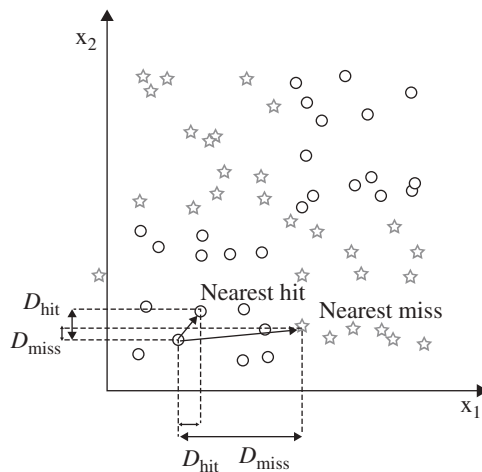
Figure 3.2. Determining nearest hit *H* and nearest miss *M* samples.

The *Relief* algorithm updates the quality score $W(A_i)$ for all features A_i depending on differences on their values for samples X, M, and H:

$$W_{new}(A_i) = W_{old}(A_i) + \frac{\left(-\text{diff}(X(A_i), H(A_i))^2 + \text{diff}(X(A_i), M(A_i))^2\right)}{m}$$

The process is repeated m times for randomly selected samples from the training data set, and the scores $W(A_i)$ are accumulated for each sample. Finally, using threshold of relevancy τ, the algorithm detects those features that are statistically relevant to the target classification, and these are the features with $W(A_i) \geq \tau$. We assume the scale of every feature is either nominal (including Boolean) or numerical (integer or real). The main steps of the *Relief* algorithm may be formalized as follows:

```
Initialize: W(Aⱼ) = 0; i = 1,.., p (p is the number of features)
For i=1 to m
  Begin
     Randomly select sample X from training data set S.
     Find nearest hit H and nearest miss M samples.

     For j=1 to p
```
$$W(A_j) = W(A_j) + \left(-\text{diff}(X(A_j), H(A_j))^2 + \text{diff}(X(A_j), M(A_j))^2\right)/m$$
```
  End

  End
Output: Subset of feature where W(Aⱼ) ≥ τ
```

For example, if the available training set is given in Table 3.2 with three features (the last one of them is classification decision) and four samples, the scoring values W for the features F_1 and F_2 may be calculated using Relief:

$$W(F_1) = 0 \frac{(+(-1+4)+(-1+9)+(-1+9)+(-1+4))}{4} = 5.5$$

$$W(F_2) = 0 \frac{(+(-1+4)+(-1+1)+(-1+4)+(-1+1))}{4} = 1.5$$

TABLE 3.2. Training Data Set for Applying Relief Algorithm

Sample	F_1	F_2	Class
1	3	4	C_1
2	2	5	C_1
3	6	7	C_2
4	5	6	C_2

In this example the number of samples is low, and therefore we use all samples ($m = n$) to estimate the features' scores. Based on the previous results, feature F_1 is much more relevant in classification than the feature F_2. If we assign the threshold value of $\tau = 5$, it is possible to eliminate feature F_2 and build the classification model only based on the feature F_1.

Relief is a simple algorithm that relies entirely on a statistical methodology. The algorithm employs few heuristics, and therefore it is efficient—its computational complexity is $O(mpn)$. With m, number of randomly selected training samples, being a user-defined constant, the time complexity becomes $O(pn)$, which makes it very scalable to data sets with both a huge number of samples n and a very high dimensionality p. When n is very large, it is assumed that $m \ll n$. It is one of the few algorithms that can evaluate features for real-world problems with large feature space and large number of samples. *Relief* is also noise tolerant and is unaffected by feature interaction, and this is especially important for hard data-mining applications. However, *Relief* does not help with removing redundant features. As long as features are deemed relevant to the class concept, they will all be selected even though many of them are highly correlated to each other.

One of the *Relief* problems is to pick a proper value of τ. Theoretically, using so-called Cebysev's inequality, τ may be estimated:

$$\tau \ll \frac{1}{\sqrt{\alpha m}}$$

While the above formula determines τ in terms of α (data-mining model accuracy) and m (the training data set size), experiments show that the score levels display clear contrast between relevant and irrelevant features so τ can easily be determined by inspection.

Relief was extended to deal with multiclass problems, noise, redundant, and missing data. Recently, additional feature selection methods based on feature weighting are proposed including *ReliefF*, *Simba*, and *I-Relief*, and they are improvements of the basic *Relief* algorithm.

3.4   ENTROPY MEASURE FOR RANKING FEATURES

A method for unsupervised feature selection or ranking based on entropy measure is a relatively simple technique; but with a large number of features its complexity increases significantly. The basic assumption is that all samples are given as vectors of a feature's values without any classification of output samples. The approach is based on the observation that removing an irrelevant feature, a redundant feature, or both from a set may not change the basic characteristics of the data set. The idea is to remove as many features as possible but yet maintain the level of distinction between the samples in the data set as if no features had been removed. The algorithm is based on a similarity measure S that is in inverse proportion to the distance D

between two n-dimensional samples. The distance measure D is small for close samples (close to zero) and large for distinct pairs (close to one). When the features are numeric, the similarity measure S of two samples can be defined as

$$S_{ij} = e^{-\alpha D ij}$$

where D_{ij} is the distance between samples x_i and x_j and α is a parameter mathematically expressed as

$$\alpha = \frac{-(\ln 0.5)}{D}$$

D is the average distance among samples in the data set. Hence, α is determined by the data. But, in a successfully implemented practical application, it was used a constant value of $\alpha = 0.5$. Normalized Euclidean distance measure is used to calculate the distance D_{ij} between two samples x_i and x_j:

$$D_{ij} = \left[\sum_{k=1}^{n} \left(\left(\frac{x_{ik} - x_{jk}}{(\max_k - \min_k)} \right)^2 \right) \right]^{1/2}$$

where n is the number of dimensions and $\max_k$ and $\min_k$ are maximum and minimum values used for normalization of the kth dimension.

All features are not numeric. The similarity for nominal variables is measured directly using Hamming distance:

$$S_{ij} = \frac{\left(\sum_{k=1}^{n} |x_{ik} = x_{jk}| \right)}{n}$$

where $|x_{ik} = x_{jk}|$ is 1 if $x_{ik} = x_{jk}$ and 0 otherwise. The total number of variables is equal to n. For mixed data, we can discretize numeric values and transform numeric features into nominal features before we apply this similarity measure. Figure 3.3a is an

(a)

Sample	F_1	F_2	F_3
R_1	A	X	1
R_2	B	Y	2
R_3	C	Y	2
R_4	B	X	1
R_5	C	Z	3

(b)

	R_1	R_2	R_3	R_4	R_5
R_1		0/3	0/3	2/3	0/3
R_2			2/3	1/3	0/3
R_3				0/3	1/3
R_4					0/3

Figure 3.3. A tabular representation of similarity measures S. (a) Data set with three categorical features. (b) A table of similarity measures Sij between samples.

example of a simple data set with three categorical features; corresponding similarities are given in Figure 3.3b.

The distribution of all similarities (distances) for a given data set is a characteristic of the organization and order of data in an n-dimensional space. This organization may be more or less ordered. Changes in the level of order in a data set are the main criteria for inclusion or exclusion of a feature from the features set; these changes may be measured by entropy.

From information theory, we know that entropy is a global measure and that it is less for ordered configurations and higher for disordered configurations. The proposed technique compares the entropy measure for a given data set before and after removal of a feature. If the two measures are close, then the reduced set of features will satisfactorily approximate the original set. For a data set of N samples, the entropy measure is

$$E = -\sum_{i=1}^{N-1} \sum_{j=i+1}^{N} \left(S_{ij} \times \log S_{ij} + \left(1 - S_{ij}\right) \times \log \left(1 - S_{ij}\right) \right)$$

where S_{ij} is the similarity between samples x_i and x_j. This measure is computed in each of the iterations as a basis for deciding the ranking of features. We rank features by gradually removing the least important feature in maintaining the order in the configurations of data. The steps of the algorithm are based on sequential backward ranking, and they have been successfully tested on several real-world applications:

1. Start with the initial full set of features F.
2. For each feature $f \in F$, remove one feature f from F and obtain a subset F_f. Find the difference between entropy for F and entropy for all F_f. In our example in Figure 3.3, we have to compare the differences $(E_F - E_{F-F1})$, $(E_F - E_{F-F2})$, and $(E_F - E_{F-F3})$.
3. Let f_k be a feature such that the difference between entropy for F and entropy for F_{fk} is minimum.
4. Update the set of features $F = F - \{f_k\}$, where "−" is a difference operation on sets. In our example, if the difference $(E_F - E_{F-F1})$ is minimum, then the reduced set of features is $\{F_2, F_3\}$. F_1 becomes the bottom of the ranked list.
5. Repeat steps 2–4 until there is only one feature in F.

A ranking process may be stopped in any iteration and may be transformed into a process of selecting features, using the additional criterion mentioned in step 4. This criterion is that the difference between entropy for F and entropy for F_f should be less than the approved threshold value to reduce feature f_k from set F. A computational complexity is the basic disadvantage of this algorithm, and its parallel implementation could overcome the problems of working with large data sets and large number of features sequentially.

3.5 PRINCIPAL COMPONENT ANALYSIS

The most popular statistical method for dimensionality reduction of a large data set is the Karhunen–Loeve (K–L) method, also called principal component analysis (PCA). In various fields, it is also known as the singular value decomposition (SVD), the Hotelling transform, and the empirical orthogonal function (EOF) method. PCA is a method of transforming the initial data set represented by vector samples into a new set of vector samples with derived dimensions. The goal of this transformation is to concentrate the information about the differences between samples into a small number of dimensions. Practical applications confirmed that PCA is the best linear dimension reduction technique in the mean-squared error sense. Being based on the covariance matrix of the features, it is a second-order method. In essence, PCA seeks to reduce the dimension of the data by finding a few orthogonal linear combinations of the original features with the largest variance. Since the variance depends on the scale of the variables, it is customary to first standardize each variable to have mean zero and standard deviation one. After the standardization, the original variables with possibly different units of measurement are all in comparable units.

More formally, the basic idea can be described as follows: a set of n-dimensional vector samples $X = \{x_1, x_2, x_3, \ldots, x_m\}$ should be transformed into another set $Y = \{y_1, y_2, \ldots, y_m\}$ of the same dimensionality, but Y have the property that most of their information content is stored in the first few dimensions. This will allow us to reduce the data set to a smaller number of dimensions with low information loss.

The transformation is based on the assumption that high information corresponds to high variance. So, if we want to reduce a set of input dimensions X to a single dimension Y, we should transform X into Y as a matrix computation:

$$Y = A \cdot X$$

choosing A such that Y has the largest variance possible for a given data set. The single dimension Y obtained in this transformation is called *the first principal component*. The first principal component is an axis in the direction of maximum variance. It minimizes the distance of the sum of squares between data points and their projections on the component axis, as it is shown in Figure 3.4 where a two-dimensional space is transformed into a one-dimensional space in which the data set has the highest variance.

In practice, it is not possible to determine matrix A directly, and therefore we compute the covariance matrix S as a first step in feature transformation. Matrix S is defined as

$$S_{n \times n} = \frac{1}{(n-1)} \left[\sum_{j=1}^{n} \left(x_j - x'\right)^{\mathrm{T}} \left(x_j - x'\right) \right]$$

where $x' = (1/n) \sum_{j=1}^{n} x_j$.
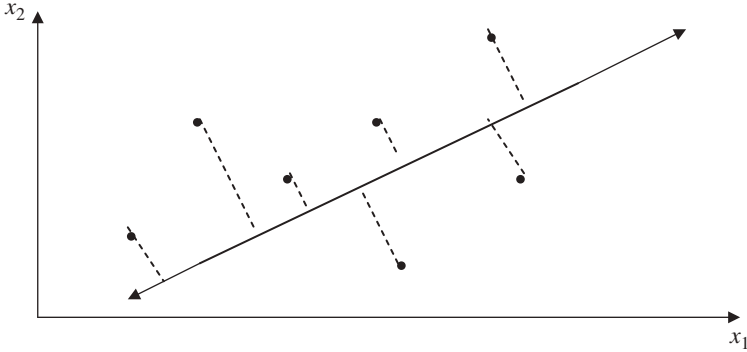
Figure 3.4. The first principal component is an axis in the direction of maximum variance.

The eigenvalues of the covariance matrix S for the given data should be calculated in the next step. Finally, the m eigenvectors corresponding to the m largest eigenvalues of S define a linear transformation from the n-dimensional space to an m-dimensional space in which the features are uncorrelated. To specify the principal components, we need the following additional explanations about the notation in matrix S:

1. The eigenvalues of $S_{n \times n}$ are $\lambda_1, \lambda_2,...,\lambda_n$, where $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n \geq 0$.
2. The eigenvectors $e_1, e_2,... e_n$ correspond to eigenvalues $\lambda_1, \lambda_2,...,\lambda_n$, and they are called the principal axes.

Principal axes are new, transformed axes of n-dimensional space, where the new variables are uncorrelated and variance for the ith component is equal to the ith eigenvalue. Because λ_i's are sorted, most of the information about the data set is concentrated in a few first principal components. The fundamental question is: how many of the principal components are needed to get a good representation of the data? In other words, what is the effective dimensionality of the data set? The easiest way to answer the question is to analyze the proportion of variance. Dividing the sum of the first m eigenvalues by the sum of all the variances (all eigenvalues), we will get the measure for the quality of representation based on the first m principal components. The result is expressed as a percentage, and if the projection accounts for over 90% of the total variance, it is considered to be good. More formally, we can express this ratio in the following way. The criterion for features selection is based on the ratio of the sum of the m largest eigenvalues of S to the trace of S. That is a fraction of the variance retained in the m-dimensional space. If the eigenvalues are labeled so that $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$, then the ratio can be written as

$$R = \frac{\left(\sum_{i=1}^{m} \lambda_i \right)}{\left(\sum_{i=1}^{n} \lambda_i \right)}$$

When the ratio R is sufficiently large (greater than the threshold value), all analyses of the subset of m features represent a good initial estimate of the n-dimensional space. This method is computationally inexpensive, but it requires characterizing data with the covariance matrix S.

We will use one example from the literature to show the advantages of PCA. The initial data set is the well-known set of Iris data, available on the Internet for data-mining experimentation. This data set has four features, so every sample is a four-dimensional vector. The correlation matrix, calculated from the Iris data after normalization of all values, is given in Table 3.3.

Based on the correlation matrix, it is a straightforward calculation of eigenvalues (in practice, usually, one of the standard statistical packages is used), and these final results for the Iris data are given in Table 3.4.

By setting a threshold value for $R* = 0.95$, we choose the first two features as the subset of features for further data-mining analysis, because

$$R = \frac{(2.91082 + 0.92122)}{(2.91082 + 0.92122 + 0.14735 + 0.02061)} = 0.958 > 0.95$$

For the Iris data, the first two principal components should be adequate description of the characteristics of the data set. The third and fourth components have small eigenvalues, and therefore, they contain very little variation; their influence on the information content of the data set is thus minimal. Additional analysis shows that, based on the reduced set of features in the Iris data, the model has the same quality using different data-mining techniques (sometimes the results were even better than with the original features).

TABLE 3.3. The Correlation Matrix for Iris Data

	Feature 1	Feature 2	Feature 3	Feature 4
Feature 1	1.0000	−0.1094	0.8718	0.8180
Feature 2	−0.1094	1.0000	−0.4205	−0.3565
Feature 3	0.8718	−0.4205	1.0000	0.9628
Feature 4	0.8180	−0.3565	0.9628	1.0000

TABLE 3.4. The Eigenvalues for Iris Data

Feature	Eigenvalue
Feature 1	2.91082
Feature 2	0.92122
Feature 3	0.14735
Feature 4	0.02061

The interpretation of the principal components can be difficult at times. Although they are uncorrelated features constructed as linear combinations of the original features, and they have some desirable properties, they do not necessarily correspond to meaningful physical quantities. In some cases, such loss of interpretability is not satisfactory to the domain scientists, and they prefer others, usually feature selection techniques.

3.6 VALUE REDUCTION

A reduction in the number of discrete values for a given feature is based on the second set of techniques in the data-reduction phase; these are the *feature-discretization techniques*. The task of feature-discretization techniques is to discretize the values of continuous features into a small number of intervals, where each interval is mapped to a discrete symbol. The benefits of these techniques are simplified data description and easy-to-understand data and final data-mining results. Also, more data-mining techniques are applicable with discrete feature values. An "old-fashioned" discretization is made manually, based on our a priori knowledge about the feature. For example, using common sense or consensus, a person's age, given at the beginning of a data-mining process as a continuous value (between 0 and 150 years), may be classified into categorical segments: child, adolescent, adult, middle age, and elderly. Cutoff points are subjectively defined (Fig. 3.5). Two main questions exist about this reduction process:

1. What are the cutoff points?
2. How does one select representatives of intervals?

Without any knowledge about a feature, a discretization is much more difficult and, in many cases, arbitrary. A reduction in feature values usually is not harmful for real-world data-mining applications, and it leads to a major decrease in computational complexity. Therefore, we will introduce, in the next two sections, several automated discretization techniques.

Within a column of a data set (set of feature values), the number of distinct values can be counted. If this number can be reduced, many data-mining methods, especially the logic-based methods explained in Chapter 6, will increase the quality of a data
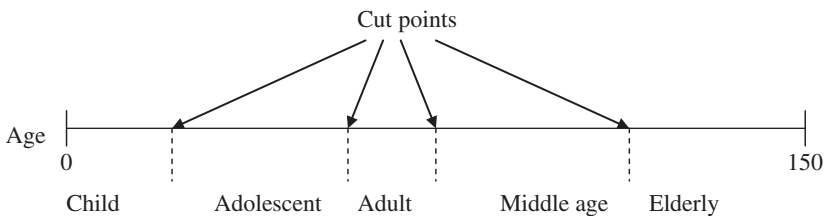
Figure 3.5. Discretization of the age feature.

analysis. Reducing the number of values by smoothing feature values does not require a complex algorithm because each feature is smoothed independently of other features and the process is performed only once, without iterations.

Suppose that a feature has a range of numeric values and that these values can be ordered from the smallest to the largest using standard greater than and less than operators. This leads naturally to the concept of *placing the values in bins*—partitioning into groups with close values. Typically, these bins have a close number of elements. All values in a bin will be merged into a single concept represented by a single value— usually either the mean or median of the bin's values. The mean or the mode is effective for a moderate or large number of bins. When the number of bins is small, the closest boundaries of each bin can be candidates for representatives in a given bin.

For example, if a set of values for a given feature f is {3, 2, 1, 5, 4, 3, 1, 7, 5, 3}, then, after sorting, these values will be organized into an ordered set:

$$\{1,1,2,3,3,3,4,5,5,7\}$$

Now, it is possible to split the total set of values into three bins with a close number of elements in each bin:

$$\{1,1,2, \quad 3,3,3, \quad 4,5,5,7\}$$
$$\text{BIN}_1 \quad \text{BIN}_2 \quad \text{BIN}_3$$

In the next step, different representatives can be selected for each bin. If the smoothing is performed based on bin modes, the new set of values for each bin will be

$$\{1,1,1, \quad 3,3,3, \quad 5,5,5,5\}$$
$$\text{BIN}_1 \quad \text{BIN}_2 \quad \text{BIN}_3$$

If the smoothing is performed based on mean values, then the new distribution for reduced set of values will be

$$\{1.33,1.33,1.33, \quad 3,3,3, \quad 5.25,5.25,5.25,5.25\}$$
$$\text{BIN}_1 \quad \text{BIN}_2 \quad \text{BIN}_3$$

and finally, if all the values in a bin are replaced by the closest of the boundary values, the new set will be

$$\{1,1,2, \quad 3,3,3, \quad 4,4,4,7\}$$
$$\text{BIN}_1 \quad \text{BIN}_2 \quad \text{BIN}_3$$

One of the main problems of this method is to find the best cutoffs for bins. In theory, a decision about cutoffs cannot be made independently of other features. Still,

heuristic decisions for every feature independently give good results in many data-mining applications. The value-reduction problem can be stated as an optimization problem in the selection of k bins. Given the number of bins k, distribute the values in the bins to minimize the average distance of a value from its bin mean or median. The distance is usually measured as the squared distance for a bin mean and as the absolute distance for a bin median. This algorithm can be computationally very complex, and a modified heuristic procedure is used to produce a near-optimal solution. The procedure consists of the following steps:

1. Sort all values for a given feature.
2. Assign approximately equal numbers of sorted adjacent values (v_i) to each bin, where the number of bins is given in advance.
3. Move a border element v_i from one bin to the next (or previous) when that reduces the global distance error (ER) (the sum of all distances from each v_i to the mean or mode of its assigned bin).

A simple example of the dynamic bin procedure for feature discretization is given next. The set of values for a feature f is {5, 1, 8, 2, 2, 9, 2, 1, 8, 6}. Split them into three bins ($k = 3$), where the bins will be represented by their modes. The computations in the first iteration of the algorithm are:

1. Sorted set of values for feature f is {1, 1, 2, 2, 2, 5, 6, 8, 8, 9}.
2. Initial bins ($k = 3$) are

$$\{1,1,2, \quad 2,2,5, \quad 6,8,8,9\}$$
$$\text{BIN}_1 \quad \text{BIN}_2 \quad \text{BIN}_3$$

3. (i) Modes for the three selected bins are {1, 2, 8}. After initial distribution, the total error, using absolute distance for modes, is

$$ER = 0+0+1+0+0+3+2+0+0+1 = 7$$

4. (iv) After moving two elements from BIN_2 into BIN_1 and one element from BIN_3 to BIN_2 in the next three iterations and obtaining smaller and smaller ER, the new and final distribution of elements will be

$$f = \{1,1,2,2,2, \quad 5,6, \quad 8,8,9\}$$
$$\text{Final bins} \Rightarrow \quad \text{BIN}_1 \quad \text{BIN}_2 \quad \text{BIN}_3$$

The corresponding modes are {2, 5, 8}, and the total minimized error ER is 4. Any additional move of elements between bins will cause an increase in the ER value. The final distribution, with medians as representatives, will be the solution for a given value-reduction problem.

Another, very simple method of smoothing feature values is *number approximation by rounding*. Rounding is a natural operation for humans; it is also natural for a

computer, and it involves very few operations. First, numbers with decimals are converted to integers prior to rounding. After rounding, the number is divided by the same constant. Formally, these steps are described with the following computations applied to the feature value X:

1. Integer division $Y = \text{int } (X/10^k)$
2. Rounding *If* $(\text{mod } (X, 10^k) \geq (10^k/2))$ *then* $Y = Y + 1$
3. Integer multiplication $X = Y \times 10^k$

where k is the number of rightmost decimal places to round. For example, the number 1453 is rounded to 1450 if $k = 1$, rounded to 1500 if $k = 2$, and rounded to 1000 if $k = 3$.

Given a number of values for a feature as an input parameter of the procedure, this simple rounding algorithm can be applied in iterations to reduce these values in large data sets. First, the feature's values are sorted so that the number of distinct values after rounding can be counted. Starting at $k = 1$, rounding is performed for all values, and the number of distinct values counted. Parameter k is increased in the next iteration until the number of values in the sorted list is reduced to less than the allowable maximum, typically in real-world applications between 50 and 100.

3.7 FEATURE DISCRETIZATION: ChiMERGE TECHNIQUE

ChiMerge is one automated discretization algorithm that analyzes the quality of multiple intervals for a given feature by using χ^2 statistics. The algorithm determines similarities between distributions of data in two adjacent intervals based on output classification of samples. If the conclusion of the χ^2 test is that the output class is independent of the feature's intervals, then the intervals should be merged; otherwise, it indicates that the difference between intervals is statistically significant and no merger will be performed.

ChiMerge algorithm consists of three basic steps for discretization:

1. Sort the data for the given feature in ascending order.
2. Define initial intervals so that every value of the feature is in a separate interval.
3. Repeat until no χ^2 of any two adjacent intervals is less than threshold value. After each merger, χ^2 tests for the remaining intervals are calculated, and two adjacent features with the χ^2 values are found. If the calculated χ^2 is less than the lowest threshold, merge these intervals. If no merge is possible, and the number of intervals is greater than the user-defined maximum, increase the threshold value.

The χ^2 test or contingency table test is used in the methodology for determining the independence of two adjacent intervals. When the data are summarized in a contingency table (its form is represented in Table 3.5), the χ^2 test is given by the formula

TABLE 3.5. A Contingency Table for 2 × 2 Categorical Data

	Class 1	Class 2	$\sum$
Interval-1	A_{11}	A_{12}	R_1
Interval-2	A_{21}	A_{22}	R_2
$\sum$	C_1	C_2	N

$$\chi^2 = \sum_{i=1}^{2} \sum_{j=1}^{k} \frac{\left(A_{ij} - E_{ij}\right)^2}{E_{ij}}$$

where k is the number of classes; A_{ij} is the number of instances in the ith interval, jth class; E_{ij} is the expected frequency of A_{ij}, which is computed as $(R_i \times C_j)/N$; R_i is the number of instances in the ith interval $= \sum A_{ij}, j = 1,\ldots,k$; C_j is the number of instances in the jth class $= \sum A_{ij}, i = 1,2$; and N is the total number of instances $= \sum R_i, i = 1,2$

If either R_i or C_j in the contingency table is 0, E_{ij} is set to a small value, for example, $E_{ij} = 0.1$. The reason for this modification is to avoid very small values in the denominator of the test. The degree of freedom parameter of the χ^2 test is for one less than the number of classes. When more than one feature has to be discretized, a threshold for the maximum number of intervals and a confidence interval for the χ^2 test should be defined separately for each feature. If the number of intervals exceeds the maximum, the algorithm ChiMerge may continue with a new, reduced value for confidence.

For a classification problem with two classes ($k = 2$), where the merging of two intervals is analyzed, the contingency table for 2 × 2 has the form given in Table 3.5. A_{11} represents the number of samples in the first interval belonging to the first class; A_{12} is the number of samples in the first interval belonging to the second class; A_{21} is the number of samples in the second interval belonging to the first class; and finally A_{22} is the number of samples in the second interval belonging to the second class.

We will analyze the ChiMerge algorithm using one relatively simple example, where the database consists of 12 two-dimensional samples with only one continuous feature (F) and an output classification feature (K). The two values, 1 and 2, for the feature K represent the two classes to which the samples belong. The initial data set, already sorted with respect to the continuous numeric feature F, is given in Table 3.6.

We can start the algorithm of a discretization by selecting the smallest χ^2 value for intervals on our sorted scale for F. We define a middle value in the given data as a splitting interval point. For the given example, interval points for feature F are 0, 2, 5, 7.5, 8.5, 10, 17, 30, 38, 42, 45.5, and 52.5. Based on this distribution of intervals, we analyze all adjacent intervals trying to find a minimum for the χ^2 test. In our example, χ^2 was the minimum for intervals [7.5, 8.5] and [8.5, 10]. Both intervals contain only one sample, and they belong to class $K = 1$. The initial contingency table is given in Table 3.7.

TABLE 3.6. Data on the Sorted Continuous Feature
F with Corresponding Classes *K*

Sample:	F	K
1	1	1
2	3	2
3	7	1
4	8	1
5	9	1
6	11	2
7	23	2
8	37	1
9	39	2
10	45	1
11	46	1
12	59	1

TABLE 3.7. Contingency Table for Intervals [7.5, 8.5] and [8.5, 10]

	$K = 1$	$K = 2$	Σ
Interval [7.5, 8.5]	$A_{11} = 1$	$A_{12} = 0$	$R_1 = 1$
Interval [8.5, 10]	$A_{21} = 1$	$A_{22} = 0$	$R_2 = 1$
Σ	$C_1 = 2$	$C_2 = 0$	$N = 2$

Based on the values given in the table, we can calculate the expected values

$$E_{11} = \frac{2}{2} = 1$$

$$E_{12} = \frac{0}{2} \approx 0.1$$

$$E_{21} = \frac{2}{2} = 1 \text{ and}$$

$$E_{22} = \frac{0}{2} \approx 0.1$$

and the corresponding $\chi 2$ test

$$\chi^2 = \frac{(1-1)^2}{1} + \frac{(0-0.1)^2}{0.1} + \frac{(1-1)^2}{1} + \frac{(0-0.1)^2}{0.1} = 0.2$$

For the degree of freedom $d = 1$, and $\chi^2 = 0.2 < 2.706$ (the threshold value from the tables for chi-square distribution for $\alpha = 0.1$), we can conclude that there are no

TABLE 3.8. Contingency Table for Intervals [0, 7.5] and [7.5, 10]

	$K = 1$	$K = 2$	$\sum$
Interval [0, 7.5]	$A_{11} = 2$	$A_{12} = 1$	$R_1 = 3$
Interval [7.5, 10]	$A_{21} = 2$	$A_{22} = 0$	$R_2 = 2$
$\sum$	$C_1 = 4$	$C_2 = 1$	$N = 5$

significant differences in relative class frequencies and that the selected intervals can be merged. The merging process will be applied in one iteration only for two adjacent intervals with a minimum χ^2 and, at the same time, with $\chi^2 <$ threshold value. The iterative process will continue with the next two adjacent intervals that have the minimum χ^2. We will just show one additional step, somewhere in the middle of the merging process, where the newly merged intervals [0, 7.5] and [7.5, 10] are analyzed. The contingency table is given as Table 3.8, and expected values are

$$E_{11} = \frac{12}{5} = 2.4$$

$$E_{12} = \frac{3}{5} = 0.6$$

$$E_{21} = \frac{8}{5} = 1.6$$

$$E_{22} = \frac{2}{5} = 0.4$$

while the $\chi 2$ test is

$$\chi^2 = \frac{(2-2.4)^2}{2.4} + \frac{(1-0.6)^2}{0.6} + \frac{(2-1.6)^2}{1.6} + \frac{(0-0.4)^2}{0.4} = 0.834$$

Selected intervals should be merged into one because, for the degree of freedom $d = 1, \chi^2 = 0.834 < 2.706$ (for $\alpha = 0.1$). In this example, with the given threshold value for χ^2, the algorithm will define a final discretization with three intervals: [0, 10], [10, 42], and [42, 60], where 60 is supposed to be the maximum value for the feature F. We can assign to these intervals coded values 1, 2, and 3 or descriptive linguistic values low, medium, and high.

Additional merging is not possible because the χ^2 test will show significant differences between intervals. For example, if we attempt to merge the intervals [0, 10] and [10, 42]—contingency table given in Table 3.9—the test results are $E_{11} = 2.78$, $E_{12} = 2.22$, $E_{21} = 2.22$, $E_{22} = 1.78$, and $\chi^2 = 2.72 > 2.706$; the conclusion is that significant differences between two intervals exist, and merging is not recommended.

TABLE 3.9. Contingency Table for Intervals [0, 10] and [10, 42]

	$K = 1$	$K = 2$	$\sum$
Interval [0, 10.0]	$A_{11} = 4$	$A_{12} = 1$	$R_1 = 5$
Interval [10.0, 42.0]	$A_{21} = 1$	$A_{22} = 3$	$R_2 = 4$
$\sum$	$C_1 = 5$	$C_2 = 4$	$N = 9$

3.8 CASE REDUCTION

Data mining can be characterized as a secondary data analysis in the sense that data miners are not involved directly with the data-collection process. That fact may sometimes explain the poor quality of raw data. Seeking the unexpected or the unforeseen, the data-mining process is not concerned with optimal ways to collect data and to select the initial set of samples; they are already given, usually in large numbers, with a high or low quality, and with or without prior knowledge of the problem at hand.

The largest and the most critical dimension in the initial data set is the number of cases or samples or, in other words, the number of rows in the tabular representation of data. Case reduction is the most complex task in data reduction. Already, in the preprocessing phase, we have elements of case reduction through the elimination of outliers and, sometimes, samples with missing values. But the main reduction process is still ahead. If the number of samples in the prepared data set can be managed by the selected data-mining techniques, then there is no technical or theoretical reason for case reduction. In real-world data-mining applications, however, with millions of samples available, that is not the case.

Let us specify two ways in which the sampling process arises in data analysis. First, sometimes the data set itself is merely a sample from a larger, unknown population, and sampling is a part of the data-collection process. Data mining is not interested in this type of sampling. Second, and more characteristic of data mining, the initial data set represents an extremely large population, and the analysis of the data is based only on a subset of samples. After the subset of data is obtained, it is used to provide some information about entire data set. It is often called estimator and its quality depends on the elements in the selected subset. A sampling process always causes a sampling error. Sampling error is inherent and unavoidable for every approach and every strategy. This error, in general, will decrease when the size of subset increases, and it will theoretically become nonexistent in the case of a complete data set. Compared to data mining with the entire data set, practical sampling possesses one or more of the following advantages: reduced cost, greater speed, greater scope, and sometimes even higher accuracy. As yet there is no known method of sampling that ensures with certainty that the estimates of the subset will be equal to the characteristics of the entire data set. Relying on sampling nearly always involves the risk of reaching incorrect conclusions. Sampling theory and the correct selection of a sampling technique can assist in reducing that risk, but not eliminating it.

There are various strategies for drawing a representative subset of samples from a data set. The size of a suitable subset is determined by taking into account the cost of computation, memory requirements, accuracy of the estimator, and other characteristics of the algorithm and data. Generally, a subset size can be determined so that the estimates for the entire data set not differ by more than a stated margin error in more than δ of the samples. By setting up a probability inequality $P(|e - e_0| \geq \varepsilon) \leq \delta$, we solve it for the subset of sample size n and for a given value ε (confidence limit) and δ (where $1 - \delta$ is the confidence level). The parameter e stands for an estimate from the subset, and it is generally a function of the subset size n, while e_0 stands for the true value obtained from entire data set. However, e_0 is usually unknown too. In this case, a practical way to determine the required size of the data subset can be done as follows: In the first step we select a small preliminary subset of samples of size m. Observations made based on this subset of data will be used to estimate e_0. After replacing e_0 in the inequality, it is solved for n. If $n \geq m$, additional $n-m$ samples are selected in the final subset for analysis. If $n \leq m$ no more samples are selected, and the preliminary subset of data is used as the final.

One possible classification of sampling methods in data mining is based on the scope of application of these methods, and the main classes are:

1. general-purpose sampling methods and
2. sampling methods for specific domains.

In this text we will introduce only some of the techniques that belong to the first class because they do not require specific knowledge about the application domain and may be used for a variety of data-mining applications.

Systematic sampling is the simplest sampling technique. For example, if we want to select 50% of a data set, we could take every other sample in a database. This approach is adequate for many applications, and it is a part of many data-mining tools. However, it can also lead to unpredicted problems when there are some regularities in the database. Therefore, the data miner has to be very careful in applying this sampling technique.

Random sampling is a method by which every sample from an initial data set has the same chance of being selected in the subset. The method has two variants: *random sampling without replacement* and *random sampling with replacement*. Random sampling without replacement is a popular technique in which n distinct samples are selected from N initial samples in the data set without repetition (a sample may not occur twice). The advantages of the approach are simplicity of the algorithm and nonexistence of any bias in a selection. In random sampling with replacement, the samples are selected from a data set such that all samples are given an equal chance of being selected, no matter how often they already have been drawn, i.e. any of the samples may be selected more than once. Random sampling is not a one-time activity in a data-mining process. It is an iterative process,

resulting in several randomly selected subsets of samples. The two basic forms of a random sampling process are as follows:

1. *Incremental sampling*—Mining incrementally larger random subsets of samples that have many real-world applications helps spot trends in error and complexity. Experience has shown that the performance of the solution may level off rapidly after some percentage of the available samples has been examined. A principal approach to case reduction is to perform a data-mining process on increasingly larger random subsets, to observe the trends in performances, and to stop when no progress is made. The subsets should take big increments in data sets, so that the expectation of improving performance with more data is reasonable. A typical pattern of incremental subsets might be 10, 20, 33, 50, 67, and 100%. These percentages are reasonable but can be adjusted based on knowledge of the application and the number of samples in the data set. The smallest subset should be substantial: typically, no fewer than 1000 samples.

2. *Average sampling*—When the solutions found from many random subsets of samples of cases are *averaged* or *voted*, the combined solution can do as well or even better than the single solution found on the full collection of data. The price of this approach is the repetitive process of data mining on smaller sets of samples and, additionally, a heuristic definition of criteria to compare the several solutions of different subsets of data. Typically, the process of voting between solutions is applied for classification problems (if three solutions are class1 and one solution is class2, then the final voted solution is class1) and averaging for regression problems (if one solution is 6, the second is 6.5, and the third 6.7, then the final averaged solution is 6.4). When the new sample is to be presented and analyzed by this methodology, an answer should be given by each solution, and a final result will be obtained by comparing and integrating these solutions with the proposed heuristics.

Two additional techniques, *stratified sampling* and *inverse sampling*, may be convenient for some data-mining applications. Stratified sampling is a technique in which the entire data set is split into nonoverlapping subsets or strata, and sampling is performed for each different strata independently of another. The combination of all the small subsets from different strata forms the final, total subset of data samples for analysis. This technique is used when the strata are relatively homogeneous and the variance of the overall estimate is smaller than that arising from a simple random sample. Inverse sampling is used when a feature in a data set occurs with rare frequency, and even a large subset of samples may not give enough information to estimate a feature value. In that case, sampling is dynamic; it starts with the small subset, and it continues until some conditions about the required number of feature values are satisfied.

For some specialized types of problems, alternative techniques can be helpful in reducing the number of cases. For example, for time-dependent data, the number of samples is determined by the frequency of sampling. The sampling period is specified based on knowledge of the application. If the sampling period is too short, most samples are repetitive and few changes occur from case to case. For some applications,

increasing the sampling period causes no harm and can even be beneficial in obtaining a good data-mining solution. Therefore, for time-series data the windows for sampling and measuring features should be optimized, and that requires additional preparation and experimentation with available data.

3.9   REVIEW QUESTIONS AND PROBLEMS

1. Explain what we gain and what we lose with dimensionality reduction in large data sets in the preprocessing phase of data mining.

2. Use one typical application of data mining in a retail industry to explain monotonicity and interruptability of data-reduction algorithms.

3. Given the data set X with three input features and one output feature representing the classification of samples:

X:	I_1	I_2	I_3	O
	2.5	1.6	5.9	0
	7.2	4.3	2.1	1
	3.4	5.8	1.6	1
	5.6	3.6	6.8	0
	4.8	7.2	3.1	1
	8.1	4.9	8.3	0
	6.3	4.8	2.4	1

(a) Rank the features using a comparison of means and variances.
(b) Rank the features using Relief algorithm. Use all samples for the algorithm ($m = 7$).

4. Given four-dimensional samples where the first two dimensions are numeric and last two are categorical:

X_1	X_2	X_3	X_4
2.7	3.4	1	A
3.1	6.2	2	A
4.5	2.8	1	B
5.3	5.8	2	B
6.6	3.1	1	A
5.0	4.1	2	B

(a) Apply a method for unsupervised feature selection based on entropy measure to reduce one dimension from the given data set.
(b) Apply Relief algorithm under the assumption that X4 is output (classification) feature.

5. (a) Perform bin-based value reduction with the best cutoffs for the following:

    (i) The feature I_3 in problem #3 using mean values as representatives for two bins.

    (ii) The feature X_2 in problem #4 using the closest boundaries for two bin representatives

  (b) Discuss the possibility of applying approximation by rounding to reduce the values of numeric attributes in problems #3 and #4.

6. Apply the ChiMerge technique to reduce the number of values for numeric attributes in problem #3.

  (a) Reduce the number of numeric values for feature I_1 and find the final, reduced number of intervals.

  (b) Reduce the number of numeric values for feature I_2 and find the final, reduced number of intervals.

  (c) Reduce the number of numeric values for feature I_3 and find the final, reduced number of intervals.

  (d) Discuss the results and benefits of dimensionality reduction obtained in (a), (b), and (c).

7. Explain the differences between averaged and voted combined solutions when random samples are used to reduce dimensionality of a large data set.

8. How can the incremental sample approach and the average sample approach be combined to reduce cases in large data sets?

9. Develop a software tool for feature ranking based on means and variances. Input data set is represented in the form of flat file with several features.

10. Develop a software tool for ranking features using entropy measure. The input data set is represented in the form of a flat file with several features.

11. Implement the ChiMerge algorithm for automated discretization of selected features in a flat input file.

12. Given the data set: $F = \{4, 2, 1, 6, 4, 3, 1, 7, 2, 2\}$. Apply two iterations of *bin method* for value reduction with best cutoffs. Initial number of bins is 3. What are the final medians of bins, and what is the total minimized error?

13. Assume you have 100 values that are all different, and use equal width discretization with 10 bins.

  (a) What is the largest number of records that could appear in one bin?

  (b) What is the smallest number of records that could appear in one bin?

  (c) If you use equal height discretization with 10 bins, what is largest number of records that can appear in one bin?

  (d) If you use equal height discretization with 10 bins, what is smallest number of records that can appear in one bin?

  (e) Now assume that the maximum value frequency is 20. What is the largest number of records that could appear in one bin with equal width discretization (10 bins)?

  (f) What about with equal height discretization (10 bins)?

14. The text shows a set of eight letters: OXYMORON.
   (a) What is the entropy in bits of the text?
   (b) What is maximum entropy for the text with eight letters?

3.10   REFERENCES FOR FURTHER STUDY

1. Liu, H., H. Motoda, eds., *Instance Selection and Construction for Data Mining*, Kluwer Academic Publishers, Boston, MA, 2001.

   Many different approaches have been used to address the data-explosion issue, such as algorithm scale-up and data reduction. Instance, sample, or tuple selection pertains to methods that select or search for a representative portion of data that can fulfill a data-mining task as if the whole data were used. This book brings researchers and practitioners together to report new developments and applications in instance-selection techniques, to share hard-learned experiences in order to avoid similar pitfalls, and to shed light on future development.

2. Liu, H., H. Motoda, *Computational Methods of Feature Selection*, CRC Press, Boston, MA, 2007.

   The book represents an excellent surveys, practical guidance, and comprehensive tutorials from leading experts. It paints a picture of the state-of-the-art techniques that can boost the capabilities of many existing data-mining tools and gives the novel developments of feature selection that have emerged in recent years, including causal feature selection and Relief. The book contains real-world case studies from a variety of areas, including text classification, Web mining, and bioinformatics.

3. Weiss, S. M., N. Indurkhya, *Predictive Data Mining: a Practical Guide*, Morgan Kaufman, San Francisco, 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. A simple presentation of topics with a large number of examples is an additional strength of the book.

4. Fodor I. K., A survey of dimension reduction techniques, *LLNL technical report*, June 2002.

   The author reviews principal component analysis and factor analysis, respectively, the two most widely used linear dimension reduction methods based on second-order statistics. However, many data sets of interest are not realizations from Gaussian distributions. For those cases, higher-order dimension reduction methods, using information not contained in the covariance matrix, are more appropriate. It includes independent component analysis and method of random projections.

5. Saul L.K., et al, Spectral Methods for Dimensionality Reduction, in B. Schööelkopf, O. Chapelle and A. Zien (Eds.), *Semisupervised learning*, MIT Press, 2005.

   Spectral methods have recently emerged as a powerful tool for nonlinear

dimensionality reduction and manifold learning. These methods are able to reveal low-dimensional structure in high-dimensional data from the top or bottom eigenvectors of specially constructed matrices. To analyze data that lies on a low-dimensional submanifold, the matrices are constructed from sparse weighted graphs whose vertices represent input patterns and whose edges indicate neighborhood relations.

6. Van der Maaten L.J.P., E.O. Postma, H.J. Van den Herik, Dimensionality Reduction: A Comparative Review, *Citeseer*, Vol. 10, February 2007, pp. 1–41.

   http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.112.5472
   &rep=rep1&type=pdf

   In recent years, a variety of nonlinear dimensionality reduction techniques have been proposed that aim to address the limitations of traditional techniques such as PCA. The paper presents a review and systematic comparison of these techniques. The performances of the nonlinear techniques are investigated on artificial and natural tasks. The results of the experiments reveal that nonlinear techniques perform well on selected artificial tasks, but do not outperform the traditional PCA on real-world tasks. The paper explains these results by identifying weaknesses of current nonlinear techniques and suggests how the performance of nonlinear dimensionality reduction techniques may be improved.

7. Ramirez-Gallego S., et al, A Survey on Data Preprocessing for Data Stream Mining: Current Status and Future Directions, *Neurocomputing*, Vol. 239, No. 24 May 2017, pp. 39–57.

   Data preprocessing and reduction have become essential techniques in current knowledge discovery scenarios, dominated by increasingly large data sets. These methods aim at reducing the complexity inherent to real-world data sets, so that they can be easily processed by current data-mining solutions. Advantages of such approaches include, among others, a faster and more precise learning process and more understandable structure of raw data. However, in the context of data preprocessing, techniques for data streams have a long road ahead of them, despite online learning is growing in importance thanks to the development of Internet and technologies for massive data collection. Throughout this survey, the authors summarize, categorize and analyze those contributions on data preprocessing that cope with streaming data. This work also takes into account the existing relationships between the different families of methods (feature and instance selection and discretization).

8. Stanton J. M., J. S. Saltz, *An Introduction to Data Science*, *SAGE Publ. Inc.*, 2018.

   This book is an easy-to-read, gentle introduction for people with a wide range of backgrounds into the world of data science. Needing no prior coding experience or a deep understanding of statistics, this book uses the R programming language and RStudio® platform to make data science welcoming and accessible for all learners. After introducing the basics of data science, the book builds on each previous concept to explain R programming from the ground up. Readers will learn essential skills in data science through demonstrations of how to use data to construct models, predict outcomes, and visualize data.

4

LEARNING FROM DATA

Chapter Objectives

- Analyze the general model of inductive learning in observational environments.
- Explain how the learning machine selects an approximating function from the set of functions it supports.
- Introduce the concepts of risk functional for regression and classification problems.
- Identify the basic concepts in statistical learning theory (SLT) and discuss the differences between inductive principles, empirical risk minimization (ERM), and structural risk minimization (SRM).
- Discuss the practical aspects of the Vapnik–Chervonenkis (VC) dimension concept as an optimal structure for inductive-learning tasks.
- Compare different inductive-learning tasks using graphical interpretation of approximating functions in a 2D space.

- Explain basic principles of support vector machines (SVMs) and semi-supervised support vector machines (S3VM).
- Specify *k*-nearest neighbor classifier: algorithm and applications.
- Introduce methods for validation of inductive-learning results.
- Introduce SMOTE algorithm for unbalanced data, and compare methods of evaluation for balanced and unbalanced data.

Many recent approaches to developing models from data have been inspired by the learning capabilities of biological systems and, in particular, those of humans. In fact, biological systems learn to cope with the unknown statistical nature of the environment in a data-driven fashion. Babies are not aware of the laws of mechanics when they learn how to walk, and most adults drive a car without knowledge of the underlying laws of physics. Humans as well as animals also have superior pattern-recognition capabilities for such tasks as identifying faces, voices, or smells. People are not born with such capabilities, but learn them through data-driven interaction with the environment.

It is possible to relate the problem of learning from data samples to the general notion of inference in classical philosophy. Every predictive-learning process consists of two main phases:

1. Learning or estimating unknown dependencies in the system from a given set of samples, and
2. Using estimated dependencies to predict new outputs for future input values of the system.

These two steps correspond to the two classical types of inference known as *induction* (progressing from particular cases—training data—to a general dependency or model) and *deduction* (progressing from a general model and given input values to particular cases of output values). These two phases are shown graphically in Figure 4.1.
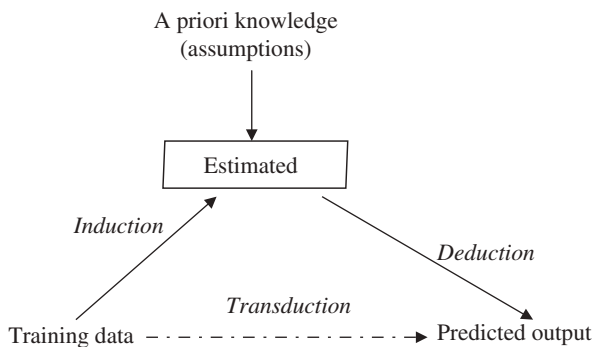
Figure 4.1. Types of inference: induction, deduction, and transduction.

A unique estimated model implies that a learned function can be applied everywhere, i.e., for all possible input values. Such global-function estimation can be overkill, because many practical problems require one to deduce estimated outputs only for a few given input values. In that case, a better approach may be to estimate the outputs of the unknown function for several points of interest directly from the training data without building a global model. Such an approach is called *transductive* inference in which a local estimation is more important than a global one. An important application of the *transductive* approach is a process of mining association rules, which has been described in detail in Chapter 8. It is very important to mention that the standard formalization of machine learning does not apply to this type of inference.

The process of inductive learning and estimating the model may be described, formalized, and implemented using different learning methods. A learning method is an algorithm (usually implemented in software) that estimates an unknown mapping (dependency) between a system's inputs and outputs from the available data set, namely, from known samples. Once such a dependency has been accurately estimated, it can be used to predict the future outputs of the system from the known input values. Learning from data has been traditionally explored in such diverse fields as statistics, engineering, and computer science. Formalization of the learning process and a precise, mathematically correct description of different inductive-learning methods were the primary tasks of disciplines such as statistical learning theory and artificial intelligence. In this chapter, we will introduce the basics of these theoretical fundamentals for inductive learning.

4.1 LEARNING MACHINE

Machine learning, as a combination of artificial intelligence and statistics, has proven to be a fruitful area of research, spawning a number of different problems and algorithms for their solution. These algorithms vary in their goals, in the available training data sets, and in the learning strategies and representation of data. All of these algorithms, however, learn by searching through an *n*-dimensional space of a given data set to find an acceptable generalization. One of the most fundamental machine-learning tasks is *inductive machine learning* where a generalization is obtained from a set of samples, and it is formalized using different techniques and models.

We can define inductive learning as the process of estimating an unknown input–output dependency or structure of a system using limited number of observations or measurements of inputs and outputs of the system. In the theory of inductive learning, all data in a learning process are organized, and for each instance of input–output pairs, we use a simple term known as a sample. The general learning scenario involves three components, represented in Figure 4.2:

1. A generator of random input vectors *X*,
2. A system that returns an output Y for a given input vector *X*, and
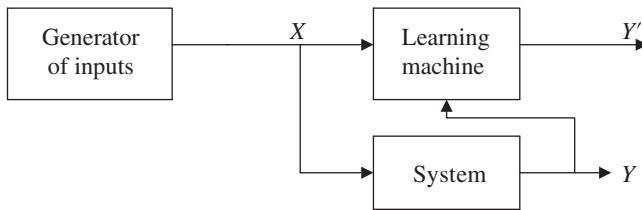
Figure 4.2. A learning machine uses observations of the system to form an approximation of its output.

   3. A learning machine, which estimates an unknown (input X, output Y') mapping of the system from the observed (input X, output Y) samples.

   This formulation is very general and describes many practical inductive-learning problems such as interpolation, regression, classification, clustering, and density estimation. The generator produces a random vector X, which is drawn independently from any distribution. In statistical terminology, this situation is called an observational setting. It differs from the designed-experiment setting, which involves creating a deterministic sampling scheme, optimal for a specific analysis according to experimental design theory. The learning machine has no control over which input values were supplied to the system, and therefore, we are talking about an observational approach in inductive machine-learning systems.

   The second component of the inductive-learning model is the system that produces an output value Y for every input vector X according to the conditional probability $p(Y/X)$, which is unknown. Note that this description includes the specific case of a deterministic system where $Y = f(X)$. Real-world systems rarely have truly random outputs; however, they often have unmeasured inputs. Statistically, the effects of these unobserved inputs on the output of the system can be characterized as random and represented with a probability distribution.

   An inductive-learning machine tries to form generalizations from particular true facts, which we call the training data set. These generalizations are formalized as a set of functions that approximate a system's behavior. This is an inherently difficult problem, and its solution requires a priori knowledge in addition to data. All inductive-learning methods use a priori knowledge in the form of the selected class of approximating functions of a learning machine. In the most general case, the learning machine is capable of implementing a set of functions $f(X, w)$, $w \in W$, where X is an input, w is a parameter of the function, and W is a set of abstract parameters used only to index the set of functions. In this formulation, the set of functions implemented by the learning machine can be any set of functions. Ideally, the choice of a set of approximating functions reflects a priori knowledge about the system and its unknown dependencies. However, in practice, because of the complex and often informal nature of a priori knowledge, specifying such approximating functions may be, in many cases, difficult or impossible.
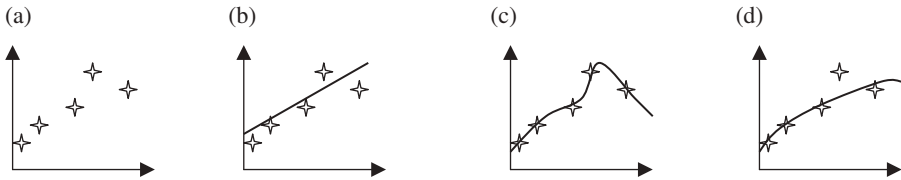
Figure 4.3. Three hypotheses for a given data set. (a) Data set. (b) Linear approximation. (c) Highly nonlinear approximation. (d) Quadratic approximation.

To explain the selection of approximating functions, we can use a graphical interpretation of the inductive-learning process. The task of inductive inference is this: given a collection of samples $(x_i, f(x_i))$, return a function $h(x)$ that approximates $f(x)$. The function $h(x)$ is often called a hypothesis. Figure 4.3 shows a simple example of this task, where the points in two-dimensional (2D) are given in Figure 4.3a, and it is necessary to find "the best" function through these points. The true $f(x)$ is unknown, so there are many choices for $h(x)$. Without more knowledge, we have no way to prefer one of three suggested solutions (Figure 4.3b, c, or d). Because there are almost always a large number of possible consistent hypotheses, all learning algorithms search through the solution space based on given criteria. For example, the criterion may be a linear approximating function that has a minimum distance from all given data points. This a priori knowledge will restrict the search space to the functions in the form given in Figure 4.3b.

There is also an important distinction between the two types of approximating functions we usually use in an inductive-learning process. Their parameters could be *linear or nonlinear*. Note that the notion of linearity is with respect to parameters rather than input variables. For example, polynomial regression in the form

$$Y = w_1 x^n + w_2 x^{n-1} + \cdots + w_0$$

is a linear method, because the parameters w_i in the function are linear (even if the function by itself is nonlinear). We will see later that some learning methods such as multilayer artificial neural networks provide an example of nonlinear parameterization, since the output of an approximating function depends nonlinearly on parameters. A typical factor in these functions is e^{-ax}, where a is a parameter and x is the input value. Selecting the approximating functions $f(X, w)$ and estimating the values of parameters w are typical steps for every inductive-learning method.

Before further formalization of a learning process, it is necessary to make a clear distinction between two concepts that are highly connected with a learning process. Let us discuss the differences between *statistical dependency and causality*. The statistical dependency between input X and output Y is expressed with the approximating functions of the learning method. The main point is that causality cannot be inferred from data analysis alone and concluded with some inductive learned model using input–output approximating functions, $Y = f(X, w)$; instead, it must be assumed or demonstrated by arguments outside the results of inductive-learning analysis.

For example, it is well known that people in Florida are on average older than in the rest of the United States. This observation may be supported by inductive-learning dependencies, but it does not imply, however, that the climate in Florida causes people to live longer. The cause is totally different; people just move there when they retire and that is possibly the cause, and maybe not the only one, of people being older in Florida than elsewhere. Similar misinterpretation could be based on the data analysis of life expectancy for a married versus a single man. Statistics show that the married man lives longer than the single man. But do not hurry with sensational causality and conclusions: that marriage is good for one's health and increases life expectancy. It can be argued that males with physical problems and/or socially deviant patterns of behavior are less likely to get married, and this could be one of possible explanations why married men live longer. Unobservable factors such as a person's health and social behavior are more likely the cause of changed life expectancy, and not the observed variable, marriage status. These illustrations should lead us to understand that inductive-learning processes build the model of dependencies but they should not automatically be interpreted as causality relations. Only experts in the domain where the data are collected may suggest additional deeper semantics of discovered dependencies.

Let us return again to the learning machine and its task of system modeling. The problem encountered by the learning machine is to select a function from the set of functions this machine supports, which best approximates the system's responses. The learning machine is limited to observing a finite number of samples n in order to make this selection. The finite number of samples, which we call a training data set, is denoted by (X_i, y_i), where $i = 1,\ldots,n$. The quality of an approximation produced by the learning machine is measured by the *loss function $L(y, f(X, w))$* where

- y is the output produced by the system,
- X is a set of inputs,
- $f(X, w)$ is the output produced by the learning machine for a selected approximating function, and
- w is the set of parameters in the approximating functions.

L measures the difference between the outputs produced by the system y_i and that produced by the learning machine $f(X_i, w)$ for every input point X_i. By convention, the loss function is nonnegative so that large positive values correspond to poor approximation and small positive values close to zero show a good approximation. The expected value of the loss is called the *risk functional $R(w)$*:

$$R(w) = \int \int L(y, f(X, w)) p(X, y) \, dX \, dy$$

where $L(y, f(X, w))$ is a loss function and $p(X, y)$ is a probability distribution of samples. The $R(w)$ value, for a selected approximating functions, is dependent only on a

set of parameters w. Inductive learning can be now defined as the process of estimating the function $f(X, w_{opt})$, which minimizes the risk functional $R(w)$ over the set of functions supported by the learning machine, using only the training data set and not knowing the probability distribution $p(X, y)$. With finite data, we cannot expect to find $f(X, w_{opt})$ exactly, so we denote $f(X, w_{opt}^*)$ as the estimate of parameters w_{opt}^* of the optimal solution w_{opt} obtained with finite training data using some learning procedure.

For common learning problems such as classification or regression, the nature of the loss function and the interpretation of risk functional are different. In a two-class classification problem, where the output of the system takes on only two symbolic values, $y = \{0, 1\}$, corresponding to the two classes, a commonly used loss function measures the classification error:

$$L(y, f(X, w)) = \begin{cases} 0 & \text{if } y = f(X, w) \\ 1 & \text{if } y \neq f(X, w) \end{cases}$$

Using this loss function, the risk functional quantifies the *probability of misclassification*. Inductive learning becomes a problem of finding the classifier function $f(X, w)$, which minimizes the probability of misclassification using only the training data set.

Regression is a process of estimating a real-value function based on a finite data set of noisy samples. A common loss function for regression is the squared error measure:

$$L(y, f(X, w)) = (y - f(X, w))^2$$

The corresponding risk functional measures the *accuracy* of the learning machine's predictions of the system output. Maximum accuracy will be obtained by minimizing the risk functional because, in that case, the approximating function will describe the best set of given samples. Classification and regression are only two of many typical learning tasks. For the other data-mining tasks, different loss functions may be selected, and they are supported with different interpretations of a risk functional.

What is a learning procedure? Or how should a learning machine use training data? The answer is given by the concept known as *inductive principle*. An inductive principle is a general prescription for obtaining an estimate $f(X, w_{opt}^*)$ in the class of approximating functions from the available finite training data. An inductive principle tells us *what* to do with the data, whereas the learning method specifies *how* to obtain an estimate. Hence a learning method or learning algorithm is a constructive implementation of an inductive principle. For a given inductive principle, there are many learning methods corresponding to a different set of functions of a learning machine. The important issue here is to choose the candidate models (approximating functions of a learning machine) of the right complexity to describe the training data.

The mathematical formulation and formalization of the learning problem explained in this section may give the unintended impression that learning algorithms do not require human intervention, but this is clearly not the case. Even though available literature is concerned with the formal description of learning methods, there is an equally important, informal part of any practical learning system. This part involves such practical and human-oriented issues as selection of the input and output variables, data encoding and representation, and incorporating a priori domain knowledge into the design of a learning system. In many cases, the user also has some influence over the generator in terms of the sampling rate or distribution. The user very often selects the most suitable set of functions for the learning machine based on his/her knowledge of the system. This part is often more critical for an overall success than the design of the learning machine itself. Therefore, all formalizations in a learning theory are useful only if we keep in mind that inductive learning is a process in which there is some overlap between activities that can be formalized and others that are not a part of formalization.

4.2   STATISTICAL LEARNING THEORY

Statistical learning theory (SLT) is relatively new, but it is perhaps one of the best currently available formalized theories for finite-sample inductive learning. It is also known as the Vapnik–Chervonenkis (VC) theory. It rigorously defines all the relevant concepts for inductive learning and provides mathematical proofs for most inductive-learning results. In contrast, other approaches such as neural networks, Bayesian inference, and decision rules are more engineering oriented, with an emphasis on practical implementation without needing strong theoretical proofs and formalizations.

STL effectively describes statistical estimation with small samples. It explicitly takes into account the sample size and provides quantitative description of the trade-off between the complexity of the model and the available information. The theory includes, as a special case, classical statistical methods developed for large samples. Understanding SLT is necessary for designing sound constructive methods of inductive learning. Many nonlinear learning procedures recently developed in neural networks, artificial intelligence, data mining, and statistics can be understood and interpreted in terms of general SLT principles. Even though SLT is quite general, it was originally developed for pattern recognition or classification problems. Therefore, the widely known practical applications of the theory are mainly for classification tasks. There is growing empirical evidence, however, of successful application of the theory to other types of learning problems.

The goal of inductive learning is to estimate unknown dependencies in a class of approximating functions using available data. The optimal estimate corresponds to the minimum expected risk functional that includes general distribution of data. This distribution is unknown, and the only available information about distribution is the finite training sample. Therefore, the only possibility is to substitute an unknown *true risk functional* with its approximation given as *empirical risk*, which is computable based

on the available data set. This approach is called *empirical risk minimization* (ERM), and it represents the basic inductive principle. Using the ERM inductive principle, one seeks to find a solution $f(X, w^*)$ that minimizes the empirical risk expressed through the training error as a substitute for the unknown true risk, which is a measure of the true error on the entire population. Depending on the chosen loss function and the chosen class of approximating functions, the ERM inductive principle can be implemented by a variety of methods defined in statistics, neural networks, automatic learning, etc. The ERM inductive principle is typically used in a learning setting where the model is given or approximated first and then its parameters are estimated from the data. This approach works well only when the number of training samples is large relative to the prespecified model complexity, expressed through the number of free parameters.

A general property necessary for any inductive principle including ERM is asymptotic *consistency*, which is a requirement that the estimated model converge to the true model or the best possible estimation, as the number of training samples grows large. An important objective of the STL is to formulate the conditions under which the ERM principle is consistent. The notion of consistency is illustrated in Figure 4.4. When the number of samples increases, empirical risk also increases, while true, expected risk decreases. Both risks approach the common minimum value of the risk functional: min $R(w)$ over the set of approximating functions and for an extra large number of samples. If we take the classification problem as an example of inductive learning, the empirical risk corresponds to the probability of misclassification for the training data, and the expected risk is the probability of misclassification averaged over a large amount of data not included into a training set and with unknown distribution.

Even though it can be intuitively expected that for $n \to \infty$ the empirical risk converges to the true risk, this by itself does not imply the consistency property, which states that minimizing one risk for a given data set will also minimize the other risk. To ensure that the consistency of the ERM method is always valid and does not depend on the properties of the approximating functions, it is necessary that consistency
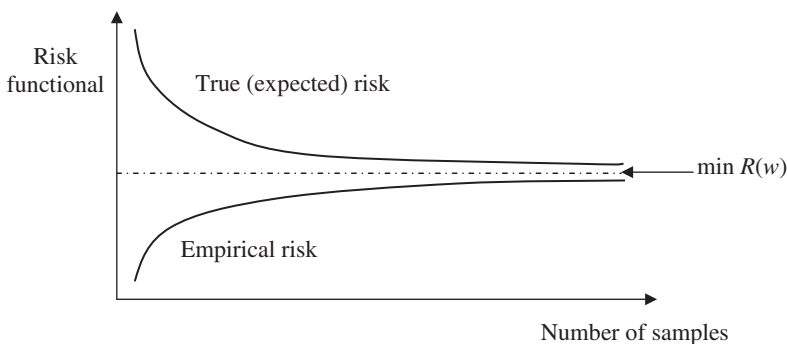
Figure 4.4. Asymptotic consistency of the ERM.
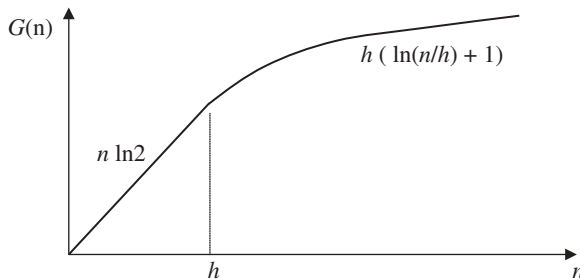
Figure 4.5. Behavior of the growth function $G(n)$.

requirement should hold for all approximating functions. This requirement is known as nontrivial consistency. From a practical point of view, conditions for consistency are at the same time prerequisites for a good generalization obtained with the realized model. Therefore, it is desirable to formulate conditions for convergence of risk functions in terms of the general properties of a set of the approximating functions.

Let us define the concept of a *growth function $G(n)$* as a function that is either linear or bounded by a logarithmic function of the number of samples n. Typical behavior of the growth function $G(n)$ is given in Figure 4.5. Every approximating function that is in the form of the growth function $G(n)$ will have a consistency property and potential for a good generalization under inductive learning, because empirical and true risk functions converge. The most important characteristic of the growth function $G(n)$ is the concept of *VC dimension*. At a point $n = h$ where the growth starts to slow down, it is a characteristic of a set of functions. If h is finite, then the $G(n)$ function does not grow linearly for enough large training data sets, and it is bounded by a logarithmic function. If $G(n)$ is only linear, then $h \rightarrow \infty$, and no valid generalization through selected approximating functions is possible. The finiteness of h provides necessary and sufficient conditions for the quick convergence of risk functions, consistency of ERM, and potentially good generalization in the inductive-learning process. These requirements place analytic constraints on the ability of the learned model to generalize, expressed through the empirical risk. All theoretical results in the STL use the VC dimension defined on the set of loss functions. But it has also been proved that the VC dimension for theoretical loss functions is equal to the VC dimension for approximating functions in typical inductive-learning tasks such as classification or regression.

The ERM inductive principle is intended for relatively large data sets, namely, when the ratio n/h is large and the empirical risk converges close to the true risk. However, if n/h is small, namely, when the ratio n/h is less than 20, then a modification of the ERM principle is necessary. The inductive principle called *structural risk minimization* (SRM) provides a formal mechanism for choosing a model with optimal complexity in finite and small data sets. According to SRM, solving a learning problem with a finite data set requires a priori specification of a structure on a set of approximating functions. For example, a set of functions S_1 is a subset of S_2, S_2 is subset of
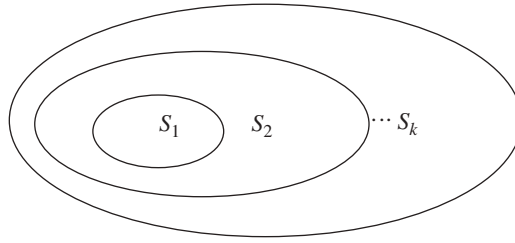
Figure 4.6. Structure of a set of approximating functions.

S_3, etc. The set of approximating functions S_1 has the lowest complexity, but the complexity increases with each new superset $S_2, S_3, \ldots, S_k$. A simplified graphical representation of the structure is given in Figure 4.6.

For a given data set, the optimal model estimation is performed through two steps:

1. Selecting an element of a structure having optimal complexity, and
2. Estimating the model based on the set of approximating functions defined in a selected element of the structure.

Through these two steps, the SRM provides a quantitative characterization of the trade-off between the complexity of approximating functions and the quality of fitting the training data. As the complexity increases (increase of the index k for S_k), the minimum empirical risk decreases, and the quality of fitting the data improves. But estimated true risk, measured through the additional testing data set, has a convex form, and in one moment it moves in a direction opposite that of the empirical risk, as shown in Figure 4.7. The SRM chooses an optimal element of the structure that yields the minimal guaranteed bound on the true risk.

In practice, to implement the SRM approach, it is necessary to be able to:

1. calculate or estimate the VC dimension for any element S_k of the structure and
2. minimize the empirical risk for each element of the structure.

For most practical inductive-learning methods that use nonlinear approximating functions, finding the VC dimension analytically is difficult, as is the nonlinear optimization of empirical risk. Therefore, rigorous application of the SRM principle cannot only be difficult but, in many cases, impossible with nonlinear approximations. This does not, however, imply that the STL is impractical. There are various heuristic procedures that are often used to implement SRM implicitly. Examples of such heuristics include early stopping rules and weight initialization, which are often used in artificial neural networks. These heuristics will be explained together with different learning methods in the following chapters. The choice of an SRM-optimization strategy suitable for a given learning problem depends on the type of approximating
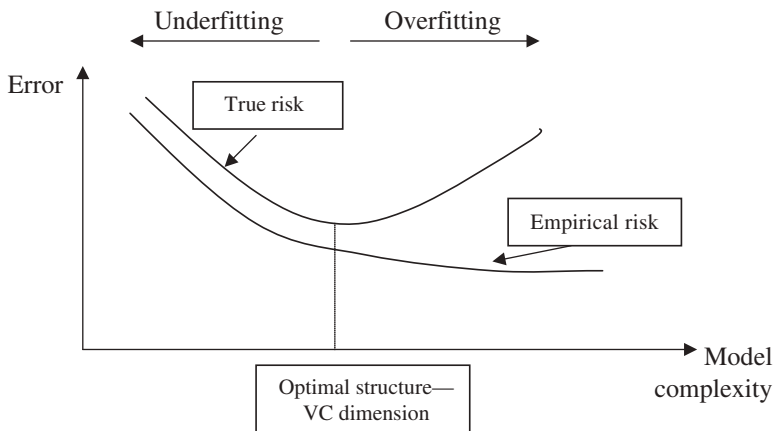
Figure 4.7. Empirical and true risk as a function of *h* (model complexity).

functions supported by the learning machine. There are three commonly used optimization approaches:

1. *Stochastic approximation (or gradient descent).* Given an initial estimate of the values for the approximating functions of parameter *w*, the optimal parameter values are found by repeatedly updating them. In each step, while computing the gradient of the risk function, the updated values of the parameters cause a small movement in the direction of the steepest descent along the risk (error) function.

2. *Iterative methods.* Parameter values *w* are estimated iteratively so that at each iteration the value of the empirical risk is decreased. In contrast to stochastic approximation, iterative methods do not use gradient estimates; instead they rely on a particular form of approximating functions with a special iterative parameter.

3. *Greedy optimization.* The greedy method is used when the set of approximating functions is a linear combination of some basic functions. Initially, only the first term of the approximating functions is used, and the corresponding parameters are optimized. Optimization corresponds to minimizing the differences between the training data set and the estimated model. This term is then held fixed, and the next term is optimized. The optimization process is repeated until values are found for all parameters w and for all terms in the approximating functions.

These typical optimization approaches and also other more specific techniques have one or more of the following problems:

1. *Sensitivity to initial conditions.* The final solution is very sensitive to the initial values of the approximation function parameters.

2. *Sensitivity to stopping rules*. Nonlinear approximating functions often have regions that are very flat, where some optimization algorithms can become "stuck" for a long time (for a large number of iterations). With poorly designed stopping rules, these regions can be interpreted falsely as local minima by the optimization algorithm.

3. *Sensitivity to multiple local minima*. Nonlinear functions may have many local minima, and optimization methods can find, at best, one of them without trying to reach global minimum. Various heuristics can be used to explore the solution space and move from a local solution toward a globally optimal solution.

Working with finite data sets, SLT reaches several conclusions that are important guidelines in a practical implementation of data-mining techniques. Let us briefly explain two of these useful principles. First, when solving a problem of inductive learning based on finite information, one should keep in mind the following general commonsense principle: Do not attempt to solve a specified problem by indirectly solving a harder general problem as an intermediate step. We are interested in solving a specific task, and we should solve it directly. Following STL results, we stress that for estimation with finite samples, it is always better to solve a specific learning problem rather than attempt a general one. Conceptually, this means that posing the problem directly will then require fewer samples for a specified level of accuracy in the solution. This point, while obvious, has not been clearly stated in most of the classical textbooks on data analysis.

Second, there is a general belief that for inductive-learning methods with finite data sets, the best performance is provided by a model of optimal complexity, where the optimization is based on the general philosophical principle known as Occam's razor. According to this principle, limiting the model complexity is more important than using true assumptions with all details. We should seek simpler models over complex ones and optimize the trade-off between model complexity and the accuracy of the model's description and fit to the training data set. Models that are too complex and fit the training data very well or too simple and fit the data poorly are both not good models because they often do not predict future data very well. Model complexity is usually controlled in accordance with Occam's razor principle by a priori knowledge.

Summarizing SLT, in order to form a unique model of a system from finite data, any inductive-learning process requires the following:

1. A wide, flexible set of *approximating functions f(X, w), w∈W*, that can be linear or nonlinear in parameters *w*.

2. *A priori knowledge* (or assumptions) used to impose constraints on a potential solution. Usually such a priori knowledge orders the functions, explicitly or implicitly, according to some measure of their flexibility to fit the data. Ideally, the choice of a set of approximating functions reflects a priori knowledge about a system and its unknown dependencies.

3. *An inductive principle*, or method of inference, specifying what has to be done. It is a general prescription for combining a priori knowledge with available training data in order to produce an estimate of an unknown dependency.

4. *A learning method*, namely, a constructive, computational implementation of an inductive principle for a given class of approximating functions. There is a general belief that for learning methods with finite samples, the best performance is provided by a model of optimum complexity, which is selected based on the general principle known as Occam's razor. According to this principle, we should seek simpler models over complex ones and optimize the model that is the trade-off between model complexity and the accuracy of fit to the training data.

4.3 TYPES OF LEARNING METHODS

There are two common types of the inductive-learning methods known:

1. Supervised learning (or learning with a teacher), and
2. Unsupervised learning (or learning without a teacher).

Supervised learning is used to estimate an unknown dependency from known input–output samples. Classification and regression are common tasks supported by this type of inductive learning. Supervised learning assumes the existence of a teacher—fitness function or some other external method of estimating the proposed model. The term "supervised" denotes that the output values for training samples are known (i.e., provided by a "teacher").

Figure 4.8a shows a block diagram that illustrates this form of learning. In conceptual terms, we may think of the teacher as having knowledge of the environment,
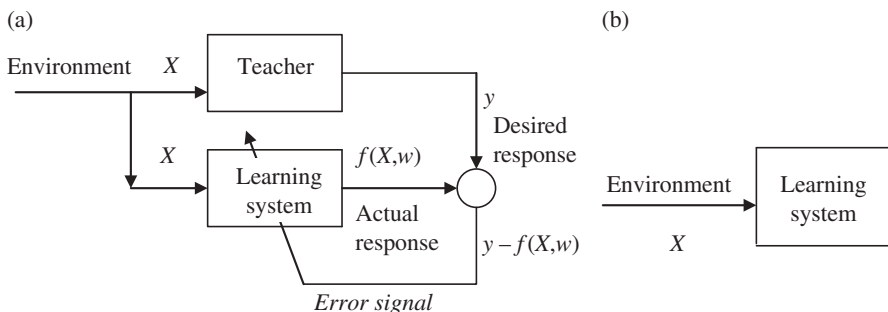
Figure 4.8. Two main types of inductive learning. (a) Supervised learning. (b) Unsupervised learning.

with that knowledge being represented by a set of input–output examples. The environment with its characteristics and model is, however, unknown to the learning system. The parameters of the learning system are adjusted under the combined influence of the training samples and the error signal. The error signal is defined as the difference between the desired response and the actual response of the learning system. Knowledge of the environment available to the teacher is transferred to the learning system through the training samples, which adjust the parameters of the learning system. It is a closed-loop feedback system, but the unknown environment is not in the loop. As a performance measure for the system, we may think in terms of the mean squared error or the sum of squared errors over the training samples. This function may be visualized as a multidimensional error surface, with the free parameters of the learning system as coordinates. Any learning operation under supervision is represented as a movement of a point on the error surface. For the system to improve the performance over time and therefore learn from the teacher, the operating point on an error surface has to move down successively toward a minimum of the surface. The minimum point may be a local minimum or a global minimum. The basic characteristics of optimization methods such as stochastic approximation, iterative approach, and greedy optimization have been given in the previous section. An adequate set of input–output samples will move the operating point toward the minimum, and a supervised learning system will be able to perform such tasks as pattern classification and function approximation. Different techniques support this kind of learning, and some of them such as logistic regression, multilayer perceptron, and decision rules and trees will be explained with more details in Chapters 5–7.

Under the *unsupervised learning* scheme, only samples with input values are given to a learning system, and there is no notion of the output during the learning process. Unsupervised learning eliminates the teacher and requires that the learner forms and evaluates the model on its own. The goal of unsupervised learning is to discover "natural" structure in the input data. In biological systems, perception is a task learned via unsupervised techniques.

The simplified schema of unsupervised or self-organized learning, without an external teacher to oversee the learning process, is indicated in Figure 4.8b. The emphasis in this learning process is on a task-independent measure of the quality of representation that is learned by the system. The free parameters w of the learning system are optimized with respect to that measure. Once the system has become tuned to the regularities of the input data, it develops the ability to form internal representations for encoding features of the input examples. This representation can be *global*, applicable to the entire input data set. These results are obtained with methodologies such as cluster analysis or some artificial neural networks, explained in Chapters 7 and 9. On the other hand, learned representation for some learning tasks can be only *local*, applicable to the specific subsets of data from the environment; association rules are a typical example of an appropriate methodology. It has been explained with more details in Chapter 10.

4.4   COMMON LEARNING TASKS

The generic inductive-learning problem can be subdivided into several common learning tasks. The fundamentals of inductive learning, along with the classification of common learning tasks, have already been given in the introductory chapter of this book. Here, we would like to analyze these tasks in detail, keeping in mind that for each of these tasks, the nature of the loss function and the output differ. However, the goal of minimizing the risk based on training data is common to all tasks. We believe that visualization of these tasks will give the reader the best feeling about the complexity of the learning problem and the techniques required for its solution.

To obtain a graphical interpretation of the learning tasks, we start with the formalization and representation of data samples that are the "infrastructure" of the learning process. Every sample used in data mining represents one entity described with several attribute–value pairs. That is one row in a tabular representation of a training data set, and it can be visualized as a point in an n-dimensional space, where n is the number of attributes (dimensions) for a given sample. This graphical interpretation of samples is illustrated in Figure 4.9, where a student with the name John represents a point in a four-dimensional space that has four additional attributes.

When we have a basic idea of the representation of each sample, the training data set can be interpreted as a set of points in the *n*-dimensional space. Visualization of data and a learning process is difficult for large number of dimensions. Therefore, we will explain and illustrate the common learning tasks in a 2D space, supposing that the basic principles are the same for a higher number of dimensions. Of course, this approach is an important simplification that we have to take care of, especially keeping in mind all the characteristics of large multidimensional data sets, explained earlier under the topic "the curse of dimensionality."

Let us start with the first and most common task in inductive learning: classification. This is a learning function that classifies a data item into one of several predefined classes. The initial training data set is given in Figure 4.10a. Samples belong

Student name	Sex	Year of name	Major	Credits
John	M	1982	CS	64
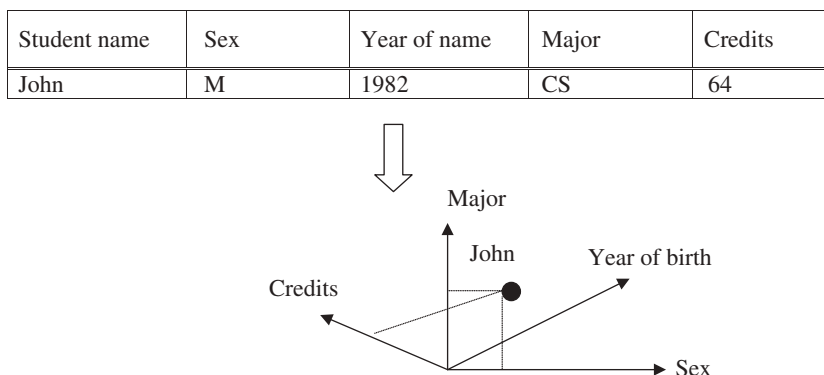
Figure 4.9. Data samples = points in an *n*-dimensional space.
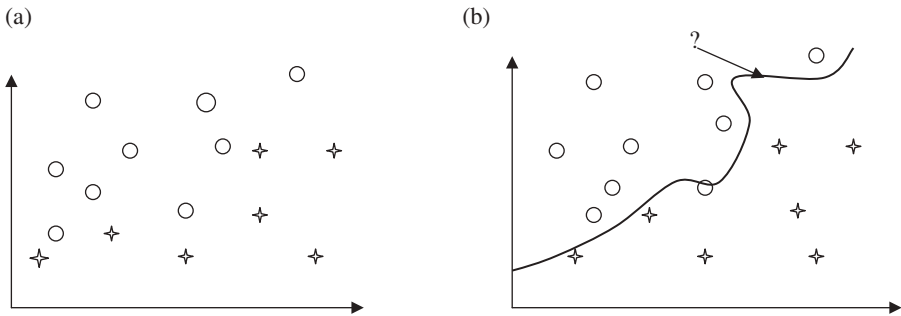
Figure 4.10. Graphical interpretation of classification. (a) Training data set. (b) Classification function.

to different classes and therefore we use different graphical symbols to visualize each class. The final result of classification in a 2D space is the curve shown in Figure 4.10b, which best separates samples into two classes. Using this function, every new sample, even without a known output (the class to which it belongs), may be classified correctly. Similarly, when the problem is specified with more than two classes, more complex functions are a result of a classification process. For an *n*-dimensional space of samples, the complexity of the solution increases exponentially, and the classification function is represented in the form of hypersurfaces in the given space.

The second learning task is *regression*. The result of the learning process in this case is a learning function, which maps a data item to a real-value prediction variable. The initial training data set is given in Figure 4.11a. The regression function in Figure 4.11b was generated based on some predefined criteria built inside a data-mining technique. Based on this function, it is possible to estimate the value of a prediction variable for each new sample. If the regression process is performed in the time domain, specific subtypes of data and inductive-learning techniques can be defined.

Clustering is the most common unsupervised learning task. It is a descriptive task in which one seeks to identify a finite set of categories or clusters to describe the data. Figure 4.12a shows the initial data, and they are grouped into clusters, as shown in Figure 4.12b, using one of the standard distance measures for samples as points in an *n*-dimensional space. All clusters are described with some general characteristics, and the final solutions differ for different clustering techniques. Based on results of the clustering process, each new sample may be assigned to one of the previously found clusters using its similarity with the cluster characteristics of the sample as a criterion.

Summarization is also a typical descriptive task where the inductive-learning process is without a teacher. It involves methods for finding a *compact description* for a set (or subset) of data. If a description is formalized, as given in Figure 4.13b, that information may simplify and therefore improve the decision-making process in a given domain.

(a)

Prediction variable

(b)

Prediction variable

New sample
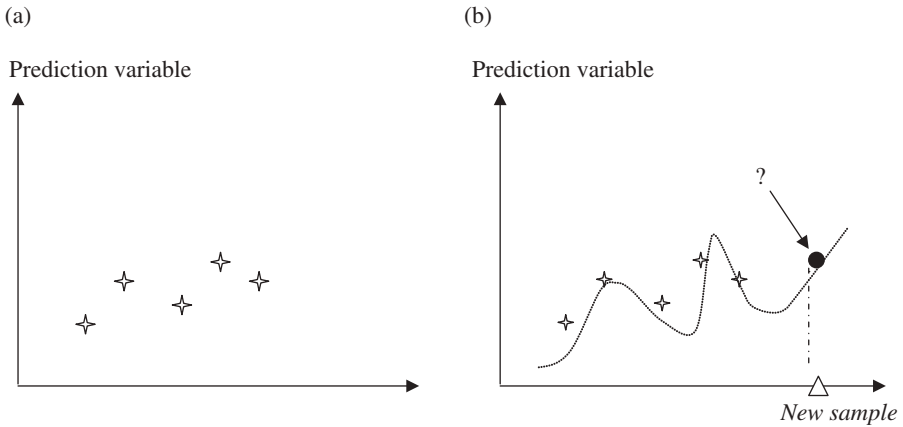
Figure 4.11. Graphical interpretation of regression. (a) Training data set. (b) Regression function.

(a)

(b)

?
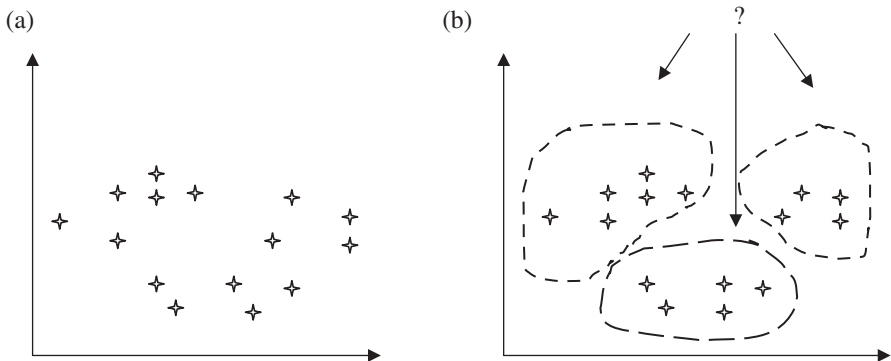
Figure 4.12. Graphical interpretation of clustering. (a) Training data set. (b) Description of clusters.

Dependency modeling is a learning task that discovers local models based on a training data set. The task consists of finding a model that describes significant dependency between features or between values in a data set covering not the entire data set, but only some specific subsets. An illustrative example is given in Figure 4.14b, where the ellipsoidal relation is found for one subset and a linear relation for the other subset of the training data. These types of modeling are especially useful in large data sets that describe very complex systems. Discovering general models based on the entire data set is, in many cases, almost impossible, because of the computational complexity of the problem at hand.

(a)

(b)
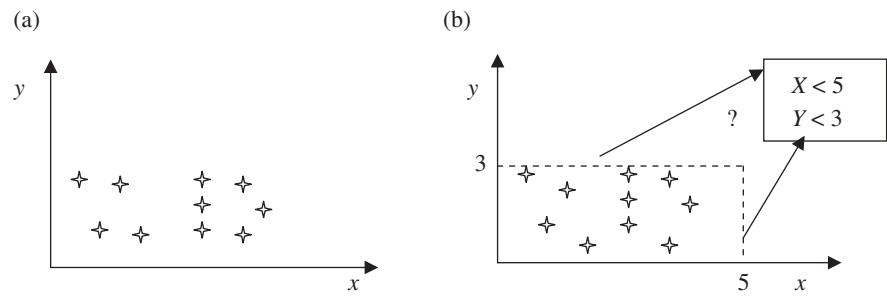
Figure 4.13. Graphical interpretation of summarization. (a) Training data set. (b) Formalized description.

(a)

(b)
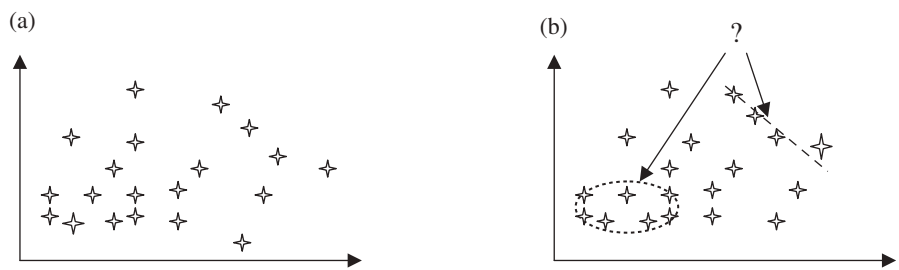
Figure 4.14. Graphical interpretation of dependency-modeling task. (a) Training data set. (b) Discovered local dependencies.

Change and deviation detection is a learning task, and we have been introduced already to some of its techniques in Chapter 2. These are the algorithms that detect outliers. In general, this task focuses on discovering the most significant changes in a large data set. Graphical illustrations of the task are given in Figure 4.15. In Figure 4.15a the task is to discover outliers in a given data set with discrete values of features. The task in Figure 4.15b is detection of time-dependent deviations for the variable in a continuous form.

The list of inductive-learning tasks is not exhausted with these six classes that are common specifications for data-mining problems. With wider and more intensive applications of the data-mining technology, new specific tasks are being developed, together with the corresponding techniques for inductive learning.

Whatever the learning task and whatever the available data-mining techniques, we have to accept that the foundation for successful data-mining processes is data-preprocessing and data-reduction methods. They transform raw and usually messy data into valuable data sets for mining using methodologies explained in Chapters 2 and 3. As a review, we will enumerate some of these techniques just to show how many alternatives the data-mining designer has in the beginning phases of the process: scaling and normalization, encoding, outliers detection and removal, feature

(a)

(b)
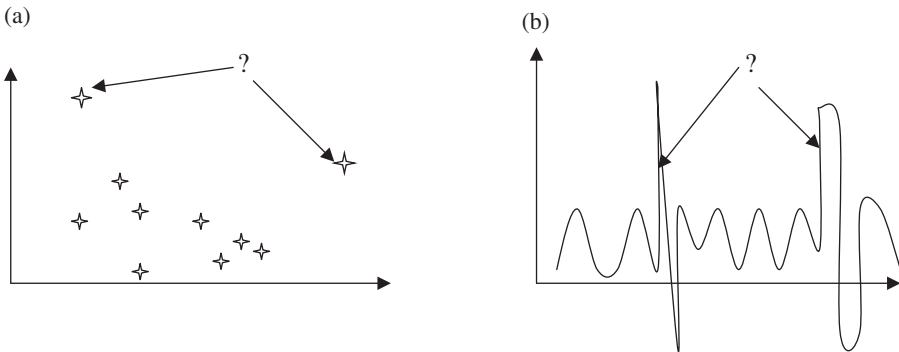
Figure 4.15. Graphical interpretation of change and detection of deviation. (a) Outliers. (b) Changes in time.

selection and composition, data cleansing and scrubbing, data smoothing, missing-data elimination, and case reduction by sampling.

When the data are preprocessed and when we know what kind of learning task is defined for our application, a list of data-mining methodologies and corresponding computer-based tools is available. Depending on the characteristics of the problem at hand and the available data set, we have to make a decision about the application of one or more of the data-mining and knowledge-discovery techniques, which include the following:

1. *Statistical methods* where the typical techniques are Bayesian inference, logistic regression, ANOVA analysis, and log-linear models.
2. *Cluster analysis*, the common techniques of which are divisible algorithms, agglomerative algorithms, partitional clustering, and incremental clustering.
3. *Decision trees and decision rules* are the set of methods of inductive learning developed mainly in artificial intelligence. Typical techniques include the CLS method, the ID3 algorithm, the C4.5 algorithm, and the corresponding pruning algorithms.
4. *Association rules* represent a set of relatively new methodologies that include algorithms such as market basket analysis, Apriori algorithm, and WWW path-traversal patterns.
5. *Artificial neural networks*, where common examples are multilayer perceptrons with backpropagation learning, Kohonen networks, or convolutional neural networks.
6. *Genetic algorithms* are very useful as a methodology for solving hard-optimization problems, and they are often a part of a data-mining algorithm.
7. *Fuzzy inference systems* are based on the theory of fuzzy sets and fuzzy logic. Fuzzy modeling and fuzzy decision-making are steps very often included in the data-mining process.

8. *N-dimensional visualization methods* are usually skipped in the literature as a standard data-mining methodology, although useful information may be discovered using these techniques and tools. Typical data-mining visualization techniques are geometric, icon-based, pixel-oriented, and hierarchical techniques.

This list of data-mining and knowledge-discovery techniques is not exhaustive, and the order does not suggest any priority in the application of these methods. Iterations and interactivity are basic characteristics of these data-mining techniques. Also, with more experience in data-mining applications, the reader will understand the importance of not relying on a single methodology. Parallel application of several techniques that cover the same inductive-learning task is a standard approach in this phase of data mining. In that case, for each iteration in a data-mining process, the results of the different techniques must additionally be evaluated and compared.

4.5   SUPPORT VECTOR MACHINES

The foundations of support vector machines (SVM) have been developed by Vladimir Vapnik and are gaining popularity due to many attractive features and promising empirical performance. The formulation embodies the SRM principle. SVMs were developed to solve the classification problem, but recently they have been extended to the domain of regression problems (for prediction of continuous variables). SVMs can be applied to regression problems by the introduction of an alternative loss function that is modified to include a distance measure. The term SVM is referring to both classification and regression methods, and the terms support vector classification (SVC) and support vector regression (SVR) may be used for more precise specification.

An SVM is a supervised learning algorithm creating learning functions from a set of labeled training data. It has a sound theoretical foundation and requires relatively small number of samples for training, and experiments showed that it is insensitive to the number of sample's dimensions. Initially, the algorithm addresses the general problem of learning to discriminate between members of two classes represented as n-dimensional vectors. The function can be a classification function (the output is binary) or the function can be a general regression function.

SVM's classification function is based on the concept of decision planes that define decision boundaries between classes of samples. A simple example is shown in Figure 4.16a where the samples belong either to class gray or black. The separating line defines a boundary on the right side of which all samples are gray and to the left of which all samples are black. Any new unclassified sample falling to the right will be classified as gray (or classified as black should it fall to the left of the separating line).

The classification problem can be restricted to consideration of the two-class problem without loss of generality. Before considering n-dimensional analysis, let us look at a simple two-dimensional example. Assume we wish to perform a classification, and our data has a categorical target variable with two categories. Also

(a)

(b)
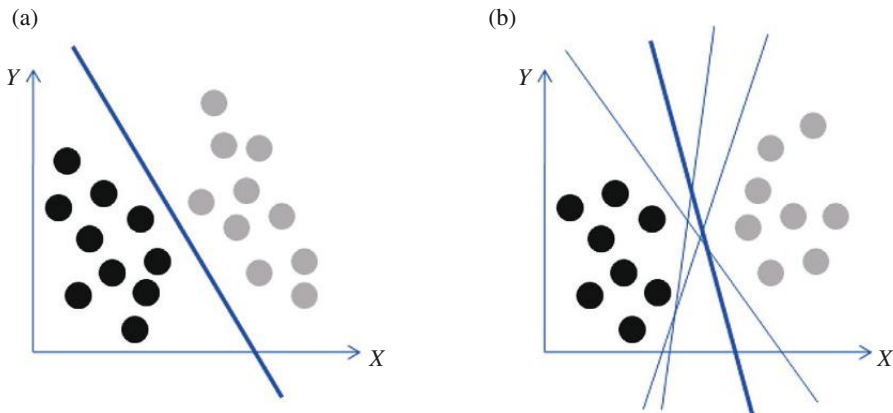
Figure 4.16. Linear separation in 2D space. (a) A decision plane in 2D space is a line. (b) How to select optimal separating line.

assume that there are two input attributes with continuous values. If we plot the data points using the value of one attribute on the X axis and the other on the Y axis, we might end up with an image such as shown in Figure 4.16b. In this problem the goal is to separate the two classes by a function that is induced from available examples. The goal is to produce a classifier that will work well on unseen examples, i.e. it generalizes well. Consider the data in Figure 4.16b. Here there are many possible linear classifiers that can separate the two classes of samples. Are all decision boundaries equally good? How to prove that selected one is the best?

The main idea is that the decision boundary should be as far away as possible from the data points of both classes. There is only one that maximizes the margin (maximizes the distance between it and the nearest data point of each class). Intuitively, the margin is defined as the amount of space or separation between the two classes as defined by the hyperplane. Geometrically, the margin corresponds to the shortest distance between the closest data points to a point on the hyperplane. STL suggests that the choice of the maximum margin hyperplane will lead to maximal generalization when predicting the classification of previously unseen examples.

Therefore a linear SVM classifier is termed the optimal separating hyperplane with the maximum margin such as the margin in Figure 4.17b. The goal of SVM modeling in n-dimensional spaces is to find the optimal hyperplane that separates classes of *n*-dimensional vectors. The split will be chosen again to have the largest distance from the hypersurface to the nearest of the positive and negative samples. Intuitively, this makes the classification correct for testing data that is near, but not identical to the training data.

Why we should maximize the margin? Skinny *margin* is more flexible and thus more complex, and the complexity is not the goal. Fat *margin* is less complex. SRM principle expresses a trade-off between training error and model complexity. It

(a)   (b)
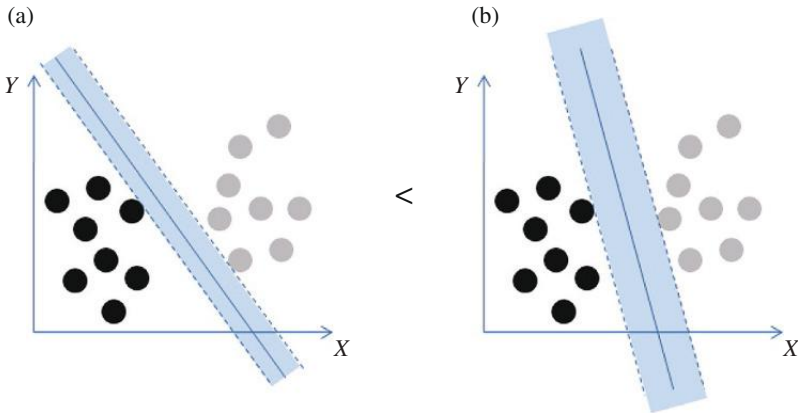
Figure 4.17. Comparison between sizes of margin of different decision boundaries. (a) Margin of decision boundary 1. (b) Margin of decision boundary 2.

(a)   (b)
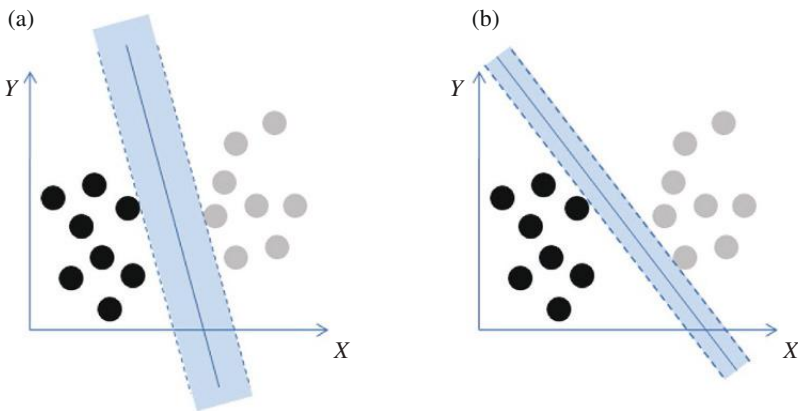
Figure 4.18. SRM principle expresses a trade-off between training error and model complexity. (a) "Fat" margin. (b) "Skinny" margin.

recommends maximum margin, such as the one in Figure 4.18, as an optimal separation criteria ensuring that SVM worst-case generalization errors are minimized.

Based on the vector equation of the line in 2D, we can define function $f(x) = w\,x + b$ as a separation model. For all points above line $f(x) > 0$, and for the points below line $f(x) < 0$. We define the sign of this function $h(x) = \text{sign}(f(x))$ as a classification function because it has the value 1 for all points above the line and the value -1 for all points below line. An example is given in Figure 4.19.

Before we continue, it is important to note that while the above examples show 2D data set, which can be conveniently represented by points in a plane, in fact we will

$$f(x_1, x_2) = x_1 + 3x_2 - 6 = 0$$
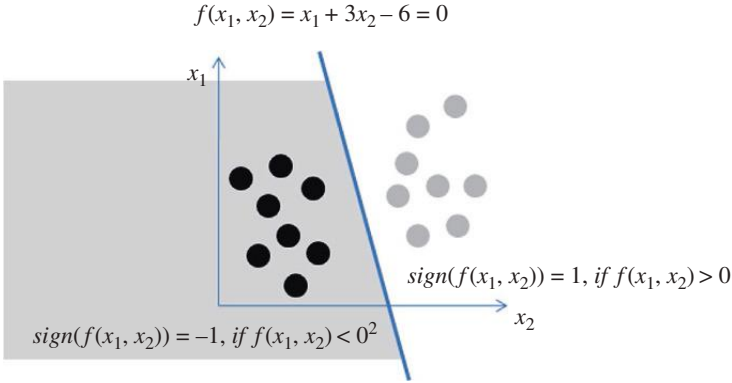
Figure 4.19. Classification function, sign (*f(x)*), on a 2D space.

typically be dealing with higher-dimensional data. The question is how to determine an optimal hyperplane in n-dimensional spaces based on given set of training samples. Consider the problem of separating the set of training vectors D belonging to two classes (coded binary with -1 and 1)

$$D = \left\{ \left(\mathbf{x}^l, \mathbf{y}^l \right), \ldots, \left(\mathbf{x}^l, \mathbf{y}^l \right) \right\}, \quad \mathbf{x} \in \Re^n, \mathbf{y} \in \{-1, 1\},$$

with a hyperplane

$$\langle \mathbf{w}, \mathbf{x} \rangle + b = 0.$$

The set of vectors is said to be optimally separated by the hyperplane if it is separated without error and the distance between the closest vectors to the hyperplane is maximal. An illustration of the separation with a graphical interpretation of main parameters w and b is given in Figure 4.20a. In this way we have parameterized the function by the weight vector w and the scalar b. The notation $\langle w, x \rangle$ denotes the inner or scalar product of vectors w and x, defined by

$$\langle \mathbf{w}, \mathbf{x} \rangle = \sum_{i=1}^{n} w_i x_i$$

In order for our hyperplane to correctly separate the two classes, we need to satisfy the following constraints:

$$\langle \mathbf{w}, \mathbf{x}^i \rangle + b > 0, \quad \text{for all } y^i = 1$$
$$\langle \mathbf{w}, \mathbf{x}^i \rangle + b < 0, \quad \text{for all } y^i = -1$$

(a)

(b)
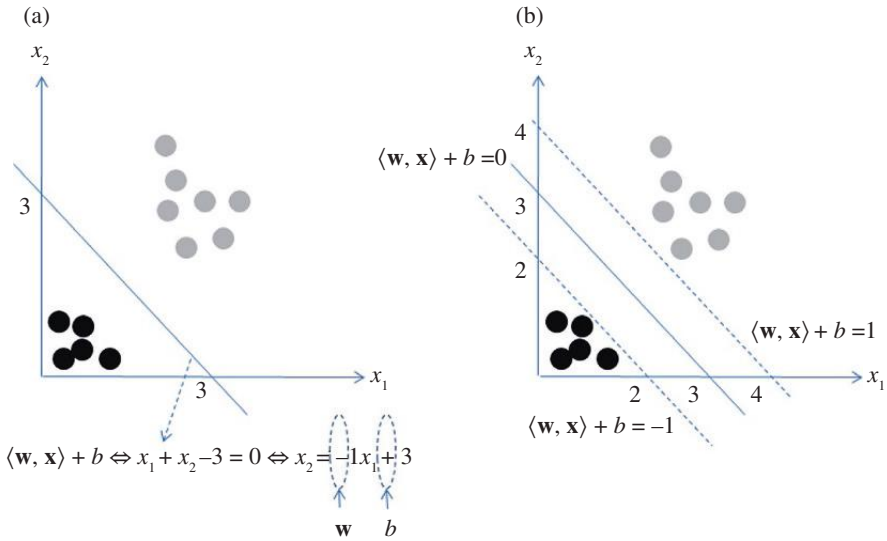
Figure 4.20. A separating hyperplane (*w*, *b*) for 2D data. (a) Parameters *w* and *b*. (b) Two parallel hyperplanes define margin.

The set of constraints that we have so far is equivalent to saying that these data must lie on the correct side (according to class label) of this decision surface. Next notice that we have also plotted as dotted lines two other hyperplanes represented in Figure 4.20b, which are the hyperplanes where the function $\langle w, x \rangle + b$ is equal to −1 (on the lower left) and +1 (on the upper right). In order to find the maximum margin hyperplane, we can see intuitively that we should keep the dotted lines parallel and equidistant to the decision surface, and maximize their distance from one another, while satisfying the constraint that the data lie on the correct side of the dotted lines associated with that class. In mathematical form, the final clause of this sentence (the constraints) can be written as

$$y^i \left(\langle \mathbf{w}, \mathbf{x}^i \rangle + b \right) \geq 1, \quad i = 1, \ldots, l.$$

The distance between these two margin hyperplanes may be formalized, because it is the parameter we want to maximize. We may obtain the distance between hyperplanes in *nD* space using equations

$$\langle \mathbf{w}, \mathbf{x}^1 \rangle + b = 1$$
$$\langle \mathbf{w}, \mathbf{x}^2 \rangle + b = -1$$

where x^1 and x^2 are any points on corresponding hyperplanes. If we subtract these equations

$$\left\langle \mathbf{w}, \left(\mathbf{x}^1 - \mathbf{x}^2\right)\right\rangle = 2$$

and representing scalar product of vectors by definition,

$$w\left(x^1 - x^2\right)\cos \gamma = 2$$

we obtain

$$\|\mathbf{w}\| \times d = 2,$$

where ‖ ‖ represents Euclidean norm or

$$d = \frac{2}{\|\mathbf{w}\|}$$

Therefore, the problem of "maximum margin" is represented as a maximum of a distance parameter d, which is a function of parameters w. Maximizing d means maximizing $1/|w|$ or minimizing $|w|$. The learning problem may be reformulated as

$$\arg\min_{w} \frac{1}{2}(\mathbf{w}\cdot\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|^2$$

subject to the constraints of linear separability. So, the final problem for optimization is

$$\arg\min_{\mathbf{w},b} \frac{1}{2}(\mathbf{w}\cdot\mathbf{w}) \text{ such that } y^i\left(\langle\mathbf{w},\mathbf{x}^i\rangle + b\right) \geq 1 \text{ for all } i = 1, 2, \ldots, l$$

The problem of optimization under constraints may be transformed using the Lagrangian $L(w,b,\alpha)$:

$$L(\mathbf{w},b,\alpha) = \frac{\|\mathbf{w}\|^2}{2} - \sum_{i=1}^{l} \alpha_i\left\{\left(\langle\mathbf{w},\mathbf{x}^i\rangle + b\right)y^i - 1\right\}$$

where α_i are the Lagrange multipliers, one for each data point. The first term is the same as the original objective function, and the second term captures the inequality constraints. The Lagrangian has to be minimized with respect to w and b:

$$\frac{\partial L}{\partial b} = 0 \quad \Rightarrow \quad \sum_{i=0}^{l} \alpha_i y^i = 0$$

$$\frac{\partial L}{\partial \mathbf{w}} = 0 \quad \Rightarrow \quad \mathbf{w}_0 = \sum_{i=0}^{l} y^i \alpha_i \mathbf{x}^i = 0$$

Substituting results of partial derivatives into L leads to the dual formulation of the optimization problem, which has to be maximized with respect to the constraints $\alpha_i \geq 0$:

$$D(\alpha) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l}\alpha_i\alpha_j y^i y^j\left(\mathbf{x}^i\cdot\mathbf{x}^j\right)$$

The dual Lagrangian $D(\alpha)$ involves only the Lagrangian multipliers α_i and the training data (there are no more parameters w and b). Finding the solution for real-world problems will usually require application of quadratic programming (QP) optimization techniques. This problem has a global optimum. The optimization approach for SVM provides an accurate implementation of the SRM inductive principle. When α_i parameters are determined, it is necessary to determine the values for w and b, and they determine final classification hyperplane. It is important to note that dual function D is function of only scalar products of sample vectors, not of vectors alone. Once the solution has been found in the form of a vector α^0, the optimal separating hyperplane is given by

$$\mathbf{w}_0 = \sum_{i\in SVs} y^i \alpha_i^0 \mathbf{x}^i$$

$$b_0 = -\frac{1}{2}\mathbf{w}_0\cdot\left[\mathbf{x}^r + \mathbf{x}^s\right]$$

where x_r and x_s are any support vectors (SVs) from each class. The classifier can then be constructed as

$$f(x) = \mathrm{sign}(\langle\mathbf{w}_0,\mathbf{x}\rangle + b_0) = \mathrm{sign}\left(\sum_{i\in SVs} y^i\alpha_i^0\left(\mathbf{x}^i\cdot\mathbf{x}\right) + b_0\right)$$

Only the points x^i that will have nonzero Lagrangian multipliers α^0 are termed *support vectors*. If the data is linearly separable, all the SVs will lie on the margin, and hence the number of SVs can be very small as it is represented in Figure 4.21. This "sparse" representation can be viewed as data compression in the construction of the classifier. The SVs are the "hard" cases; these are the training samples that are most difficult to classify correctly and that lie closest to the decision boundary.

The SVM learning algorithm is defined so that, in a typical case, the number of SVs is small compared with the total number of training examples. This property allows the SVM to classify new examples efficiently, since the majority of the training examples can be safely ignored. SVMs effectively remove the uninformative patterns from the data set by assigning them α_i weights of zero. So, if internal points that are not SVs are changed, no effect will be made on the decision boundary. The hyperplane is represented sparsely as a linear combination of "SV" points. The SVM automatically identifies a subset of these informative points and uses them to represent the solution.
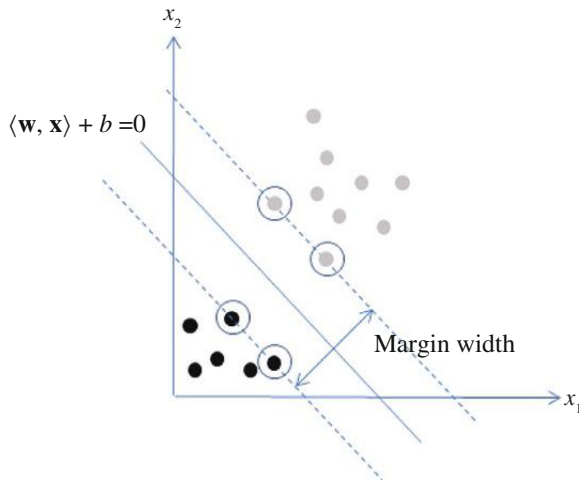
Figure 4.21. A maximal margin hyperplane with its support vectors encircled.
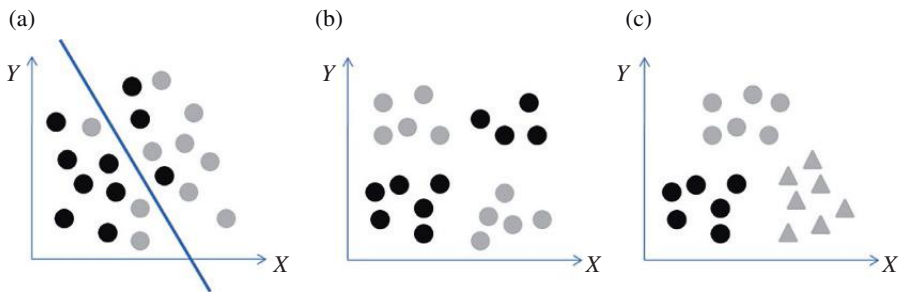
Figure 4.22. Issues for an SVM in real-world applications. (a) Subsets cannot be completely separated. (b) Nonlinear separation. (c) Three categories.

In real-world applications SVMs must deal with (1) handling the cases where subsets cannot be completely separated, (2) separating the points with nonlinear surfaces, and (3) handling classifications with more than two categories. Illustrative examples are given in Figure 4.22. What are solutions in these cases? We will start with the problem of data that are not linearly separable. The points such as shown in Figure 4.23a could be separated only by a nonlinear region. Is it possible to define linear margin where some points may be on opposite sides of hyperplanes?

Obviously, there is no hyperplane that separates all of the samples in one class from all of the samples in the other class. In this case there would be no combination of w and b that could ever satisfy the set of constraints. This situation is depicted in Figure 4.23b, where it becomes apparent that we need to *soften* the constraint that these data lay on the correct side of the +1 and −1 hyperplanes. We need to allow

(a)

(b) $\langle \mathbf{w}, \mathbf{x} \rangle + b = 0$

$\langle \mathbf{w}, \mathbf{x} \rangle + b = 1$

$\langle \mathbf{w}, \mathbf{x} \rangle + b = -1$
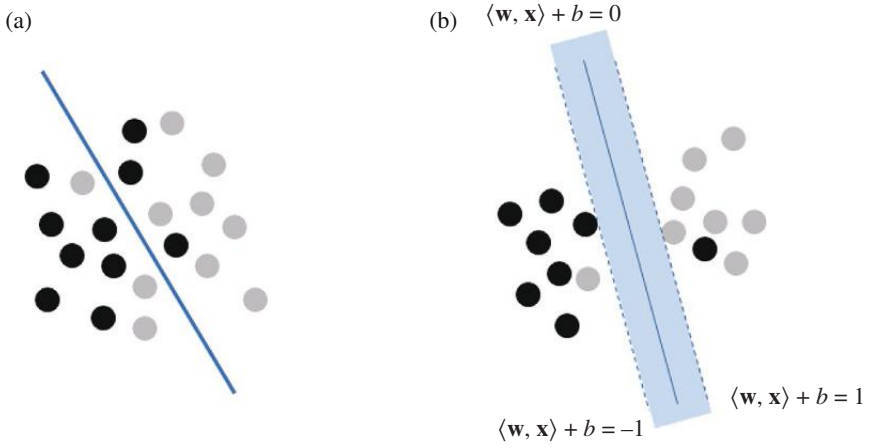
Figure 4.23. Soft margin SVM. (a) Soft separating hyperplane. (b) Errors points with their distance.

some, but not too many, data points to violate these constraints by a preferably small amount. This alternative approach turns out to be very useful not only for data sets that are not linearly separable, but also, and perhaps more importantly, in allowing improvements in generalization. We modify the optimization problem including cost of violation factor for samples that violate constraints:

$$\frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{l}\xi_i$$

under new constraints:

$$\left(\langle \mathbf{w}, \mathbf{x}^i \rangle + b\right)y^i \geq 1 - \xi_i$$

where C is a parameter representing the cost of violating the constraints and ξ_i are distances of samples that violate constraints. To allow some flexibility in separating the categories, SVM models introduce a cost parameter, C, that controls the trade-off between allowing training errors and forcing rigid margins. It creates a *soft margin*, as the one in Figure 4.24 that permits some misclassifications. If C is too small, then insufficient stress will be placed on fitting the training data. Increasing the value of C increases the cost of misclassifying points and forces the creation of a more accurate model that may not generalize well.

This SVM model is very similar case to the previous optimization problem for the linear separable data, except that there is an upper bound C on all α_i parameters. The value of C trades between how large of a margin we would prefer and how many of the training set examples violate this margin (and by how much). The process

(a)   (b)
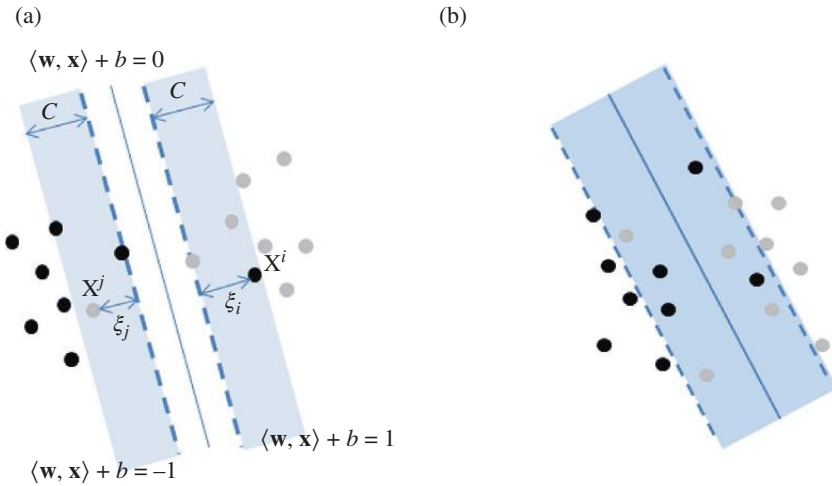
Figure 4.24. Trade-off between allowing training errors and forcing rigid margins. (a) Parameters C and ξ for a soft margin. (b) Soft classifier with a fat margin ($C > 0$).

of optimization is going through the same steps: Lagrangian, optimization of α_i parameters, determining w and b values for classification hyperplane. The dual stay the same, but with additional constraints on α parameters: $0 \leq \alpha_i \leq C$.

Most classification tasks require more complex models in order to make an optimal separation, i.e., correctly classify new test samples on the basis of the trained SVM. The reason is that the given data set requires nonlinear separation of classes. One solution to the inseparability problem is to map the data into a higher-dimensional space and define a separating hyperplane there. This higher-dimensional space is called the *feature space*, as opposed to the *input space* occupied by the training samples. With an appropriately chosen feature space of sufficient dimensionality, any consistent training set can be made linearly separable. However, translating the training set into a higher-dimensional space incurs both computational and learning costs. Representing the feature vectors corresponding to the training set can be extremely expensive in terms of memory and time. Computation in the feature space can be costly because it is high dimensional. Also, in general, there is the question which function is appropriate for transformation. Do we have to select from infinite number of potential functions?

There is one characteristic of the SVM optimization process that helps in determining the steps in the methodology. The SVM decision function for classifying points with respect to the hyperplane only involves dot products between points. Furthermore, the algorithm that finds a separating hyperplane in the feature space can be stated entirely in terms of vectors in the input space and dot products in the feature space. We are transforming training samples from one space into the other. But we are making computation only with scalar products of points in this new space. This
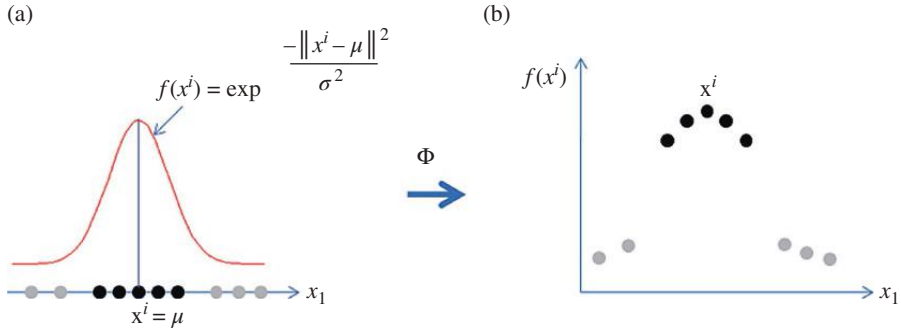
Figure 4.25. An example of a mapping Φ to a feature space in which the data become linearly separable. (a) One-dimensional input space. (b) Two-dimensional feature space.

product is very easy to compute because only a small subset of points are SVs involved in product computation. Thus, an SVM can locate a separating hyperplane in the feature space and classify points in that space without ever representing the space explicitly, simply by defining a function, called a *kernel function*. Kernel function K plays always the role of the dot product in the feature space:

$$K(x,y) = \langle \Phi(x), \Phi(y) \rangle$$

This approach avoids the computational burden of explicitly representing all transformed source data and high-dimensional feature vectors. The two most widely used kernel functions are polynomial kernel

$$K(x,y) = (\langle x,y \rangle + 1)^d$$

and Gaussian kernel

$$K(x,y) = \exp\left(\frac{-\|x-y\|^2}{\sigma^2}\right)$$

The polynomial kernel is valid for all positive integers $d \geq 1$. The Gaussian kernel is one of a group of kernel functions known as radial basis functions (RBFs). RBFs are kernel functions that depend only on the geometric distance between x and y, and the kernel is valid for all nonzero values of the kernel width σ. It is probably the most useful and commonly applied kernel function. The concept of a kernel mapping function is very powerful. It allows SVM models to perform separations even with very

complex boundaries. We can analyze the relation between a kernel function and a feature space for simplified version of quadratic kernel $k(x, y) = \langle x, y \rangle^2$ where $x, y \in R^2$:

$$\langle \mathbf{x}, \mathbf{y} \rangle^2 = (x_1 y_1 + x_2 y_2)^2$$

$$= (x_1 y_1 + x_2 y_2)(x_1 y_1 + x_2 y_2)$$

$$= (x_1^2 y_1^2 + x_2^2 y_2^2 + 2x_1 x_2 y_1 y_2)$$

$$= \left(x_1^2, x_2^2, \sqrt{2} x_1 x_2 \right) \left(y_1^2, y_2^2, \sqrt{2} y_1 y_2 \right)$$

$$= \langle \Phi(\mathbf{x}), \Phi(\mathbf{y}) \rangle$$

It defines three-dimensional (3D) feature space $\Phi(x) = \left(x_1^2, x_2^2, \sqrt{2} x_1 x_2 \right)$. Similar analysis may be performed for other kernel function. For example, through the similar process, verify that for the "full" quadratic kernel $(\langle x, y \rangle + 1)^2$, the feature space is six-dimensional.

In practical use of SVM, only the kernel function k (and not transformation function Φ) is specified. The selection of an appropriate kernel function is important, since the kernel function defines the feature space in which the training set examples will be classified. As long as the kernel function is legitimate, an SVM will operate correctly even if the designer does not know exactly what features of the training data are being used in the kernel-induced feature space. The definition of a legitimate kernel function is given by Mercer's theorem: the function must be continuous and positive definite.

Modified and enhanced SVM constructs an optimal separating hyperplane in the higher-dimensional space. In this case, the optimization problem becomes

$$D(\alpha) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \alpha_j y^i y^j K\left(\mathbf{x}^i \cdot \mathbf{x}^j \right)$$

where $K(x,y)$ is the kernel function performing the nonlinear mapping into the feature space and the constraints are unchanged. Using kernel function we will perform minimization of dual Lagrangian in the feature space, and determine all margin parameter, without representing points in this new space. Consequently, everything that has been derived concerning the linear case is also applicable for a nonlinear case by using a suitable kernel K instead of the dot product.

The approach with kernel functions gives modular SVM methodology. One module is always the same: linear learning module. It will find margin for linear separation of samples. If the problem of classification is more complex, requiring nonlinear separation, then we include new preparatory module. This module is based on kernel function, and it transforms input space into higher feature space where the same linear learning module may be applied and the final solution is nonlinear classification model. Illustrative example is given in Figure 4.26. This combination of different kernel functions with standard SVM learning algorithm for linear separation gives the flexibility to the SVM methodology for efficient application in nonlinear cases.
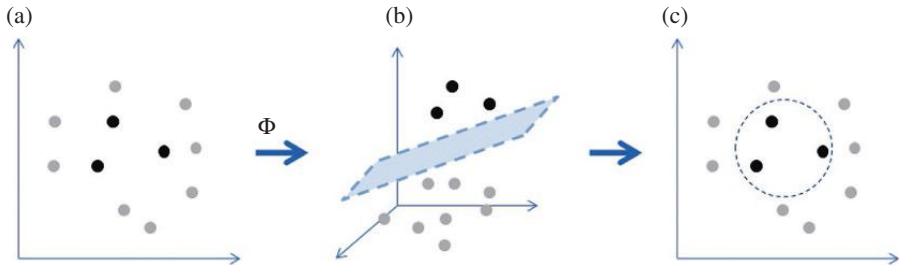
Figure 4.26. SVM performs nonlinear classification by kernel-based transformations. (a) 2D input space. (b) 3D feature space. (c) 2D input space.

The idea of using a hyperplane to separate the feature vectors into two groups works well when there are only two target categories, but how does SVM handle the case where the target variable has more than two categories? Several approaches have been suggested, but two are the most popular: (1) "one against many" where each category is split out and all of the other categories are merged and (2) "one against one" where $k(k-1)/2$ models are constructed and k is the number of categories.

A preparation process for SVM applications is enormously important for the final results, and it includes preprocessing of raw data and setting model parameters. SVM requires that each data sample be represented as a vector of real numbers. If there are categorical attributes, we first have to convert them into numeric data. Multi-attribute coding is recommended in this case. For example, a three-category attribute such as red, green, and blue can be represented with three separate attributes and corresponding codes such as (0,0,1), (0,1,0), and (1,0,0). This approach is appropriate only if the number of values in an attribute is not too large. Second, scaling values of all numerical attributes before applying SVM is very important in successful application of the technology. The main advantage is to avoid that attributes with greater numeric ranges dominate those in smaller ranges. Normalization for each attribute may be applied to the range [−1; +1] or [0; 1].

Selection of parameters for SVM is very important, and the quality of results depends on these parameters. Two most important parameters are cost C and parameter γ for Gaussian kernel. It is not known beforehand which C and σ are the best for one problem; consequently some kind of parameter search must be done. The goal is to identify good $(C; \sigma)$ so that the classifier can accurately predict unknown data (i.e., testing data). Note that it may not be required to achieve high training accuracy. Small cost C is appropriate for close to linear separable samples. If we select small C for nonlinear classification problem, it will cause underfitted learning. Large C for nonlinear problems is appropriate, but not too much because the classification margin will become very thin resulting in overfitted learning. Similar analysis is for Gaussian kernel σ parameter. Small σ will cause close to linear kernel with no significant transformation in future space and less flexible solutions. Large σ generates extremely complex nonlinear classification solution.

The experience in many real-world SVM applications suggests that, in general, RBF model is a reasonable first choice. The RBF kernel nonlinearly maps samples into a higher-dimensional space, so it, unlike the linear kernel, can handle the case when the relation between classes is highly nonlinear. The linear kernel should be treated a special case of RBF. The second reason for RBF selection is the number of hyperparameters that influences the complexity of model selection. For example, polynomial kernels have more parameters than the RBF kernel, and the tuning proves is much more complex and time consuming. However, there are some situations where the RBF kernel is not suitable, and one may just use the linear kernel with extremely good results. The question is: when to use the linear kernel as a first choice? If the number of features is large, one may not need to map data to a higher-dimensional space. Experiments showed that the non-linear mapping does not improve the SVM performance. Using the linear kernel is good enough, and C is the only tuning parameter. Many microarray data in bioinformatics and collection of electronic documents for classification are examples of this data set type. As the number of features is smaller, and the number of samples increases, SVM successfully maps data to higher-dimensional spaces using nonlinear kernels.

One of the methods for finding optimal parameter values for an SVM is a grid search. The algorithm tries values of each parameter across the specified search range using geometric steps. Grid searches are computationally expensive because the model must be evaluated at many points within the grid for each parameter. For example, if a grid search is used with 10 search intervals and an RBF kernel function is used with two parameters (C and σ), then the model must be evaluated at $10 \times 10 = 100$ grid points, i.e. 100 iterations in a parameter selection process.

At the end we should highlight main strengths of the SVM methodology. First, a training process is relatively easy with small number of parameters, and the final model is never presented with local optima, unlike some other techniques. Also, SVM methodology scales relatively well to high-dimensional data, and it represents a trade-off between classifier's complexity and accuracy. Nontraditional data structures like strings and trees can be used as input samples to SVM, and the technique is applicable not only for classification problems, but also it is accommodated for pre-diction. Weaknesses of SVMs include computational inefficiency and need to choose experimentally a "good" kernel function.

The SVM methodology is very popular in the data-mining community. Software tools that include SVM are becoming professional, user friendly, and applicable for many real-world problems where data sets are extremely large. It has been shown that SVM outperforms other techniques such as logistic regression or artificial neural networks on a wide variety of real-world problems. Some of the most successful applications of the SVM have been in image processing, in particular handwritten digit recognition and face recognition. Other interesting application areas for SVMs are in text mining and categorization of large collection of documents and in the analysis of genome sequences in bioinformatics. Furthermore, the SVM has been successfully used in a study of text and data for marketing applications. As kernel methods and max-imum margin methods including SVM are further improved and taken up by the data-mining community, they are becoming an essential tool in any data miner's toolkit.

4.6   SEMI-SUPERVISED SUPPORT VECTOR MACHINES (S3VM)

In recent decades, the ways of collecting data are more diverse, and the amount of data is growing exponentially. With the rapid development of various technologies, it becomes easy to collect large amounts of data. To build good predictive models, it is necessary to have training data sets with labels. However, because the obtained data in most of modern applications are extremely massive, it is unfeasible to invest a lot of resources in the work of their labeling. It is likely to occur that collecting unlabeled data samples is becoming cheap, but obtaining the labels costs a lot of time, effort, or money. This is the case in many application areas of machine learning, and the following examples are just a few illustrations in big data environment:

- In speech recognition, it costs almost nothing to record huge amounts of speech, but labeling it requires some human to listen to it and type a transcript.
- Billions of Web pages are directly available for automated processing, but to classify them reliably, humans have to read them.
- Protein sequences are nowadays acquired at industrial speed (by genome sequencing, computational gene finding, and automatic translation), but to resolve a 3D structure or to determine the functions of a single protein may require very significant scientific work.

Based on characteristics of training data sets, the classification of machine-learning tasks can be extended from two into three main types: unsupervised learning, supervised learning, and a new type so-called semi-supervised learning (SSL). While supervised and unsupervised learning techniques are introduced earlier, this section is explaining main ideas behind SSL. Essentially, the learning models in this case are similar to models in supervised learning with labeled samples; only this time the model is enhanced using large amount of cheap unlabeled data.

SSL represents one of the research focuses on machine learning in recent years, and it has attracted much attention in many application fields ranging from bioinformatics to Web mining. These are disciplines where it is easier to obtain unlabeled samples, while labeling requires significant effort, expertise in the field, and time consumption. Just imagine reading labeling millions of emails as a spam or no spam to create high-quality automatic classification system. SSL is a machine-learning approach that is combining unsupervised learning and supervised learning. The basic idea is in using a large number of unlabeled data to help the supervised learning method improve modeling results. More formally, in SSL, there are labeled data set $L = \{(x_1, y_1), (x_2, y_2),\ldots,(x_m, y_m)\}$ and unlabeled data set $U = \{x'_1, x'_2, \ldots, x'_n\}$, where $m \ll n$, x is a d-dimensional input vector, and y are labels. The task is to determine a function $f\colon X \rightarrow Y$, which could accurately predict a label y for each sample $x \square X$. Since unlabeled data carry less information about f function than labeled data, they are required in large amounts in order to increase prediction accuracy of the model. SSL will be mostly useful whenever there are far more unlabeled data samples than

(a)   (b)
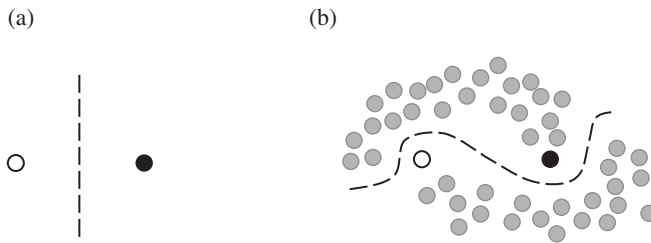
Figure 4.27. Classification model using labeled and unlabeled samples. (a) Model based on only labeled samples. (b) Model based on only labeled and unlabeled samples.

labeled ones. This big data assumption implies the need for fast and very efficient SSL algorithms.

An illustrative example of the influence of unlabeled data in SSL is given in Figure 4.27. The Figure 4.27a part shows a decision boundary we might adopt after seeing only one positive (white circle) and one negative (black circle) example. The Figure 4.27b part shows a decision boundary we might adopt if, in addition to the two labeled examples, we were given a collection of unlabeled data (gray circles). This could be viewed as performing clustering of unlabeled samples and then labeling the clusters by synchronizing these labels with the given labeled data. This labeling process enables to push the decision boundary away from high-density regions. In both cases in Figure 4.27, with or without unlabeled samples, maximum margin principle has determined the final decision boundaries.

In order to evaluate SSL model, it is necessary to make a comparison with results of a supervised algorithm that uses only labeled data. The question is, can SSL implementation have a more accurate prediction and better model by taking into account the unlabeled points? In principle, the answer is "yes." However, there are important conditions to reach this improved solution: the distribution of unlabeled samples has to be relevant for the classification problem. Using more mathematical formulation, one could say that the knowledge on $p(x)$ distribution, which is gained through the unlabeled data, has to carry useful information in the inference of $p(y|x)$. If this is not the case, SSL will no yield an improvement over supervised learning. It might even happen that using the unlabeled data degrades the prediction accuracy by misguiding the inference.

SSL include techniques such as semi-supervised support vector machines (S3VM), self-training algorithms, generative models, graph-based algorithms, and multi-view approaches. This short review gives some additional details about S3VM. S3VM is an extension of standard SVM methodology using additionally available unlabeled data. This approach implements the cluster assumption for SSL, that is, examples in data cluster have similar labels, so classes are well separated and do not cut through dense unlabeled data. The main goal of S3VM is to build classifier by using both labeled and unlabeled data. Similar to the main idea of SVM, S3VM requires the maximum margin to separate training samples, including all
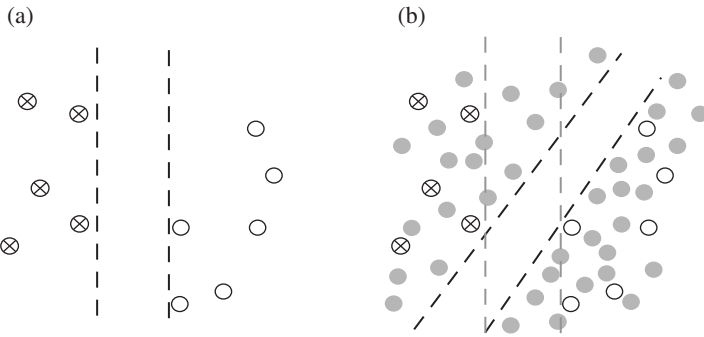
Figure 4.28. Semi-supervised model improves classification. (a) Supervised model trained on labeled samples alone. (b) Semi-supervised model use also unlabeled samples.

labeled and unlabeled data. The basic principle of S3VM is presented in Figure 4.28. If the learning process is made only based on labeled samples represented with small circles with + or −, SVM model with maximum separation is given in with dashed lines. If unlabeled samples are taken into modeling and density is accepted as a criterion for separation margin, then maximized classification margin is totally transformed into parallel full lines.

S3VM shows satisfactory results only if two assumptions about unlabeled data are satisfied:

1. *Continuity assumption*—Unlabeled samples in n-dimensional space, which are close to each other, are more likely to share the same label. This is also generally assumed in supervised learning, and it yields a preference for geometrically simple decision boundaries. In the case of SSL, the *smoothness assumption* represents extension that additionally yields a preference for classification boundaries in low-density regions so that there are fewer samples close to each other belonging different classes.
2. *Cluster assumption*—The data tend to form discrete clusters, and points in the same cluster are more likely to share the same label. Label sharing may be spread across multiple clusters. For example, if unlabeled samples are organized into X clusters, then $X–Y$ clusters may belong to one class (one label), and the rest of Y clusters will belong to the other class (the example is for two-class problems!). Cluster assumption is a special case of the smoothness assumption and gives rise to feature learning with clustering algorithms.

With the assumption that smoothness and clustering requirements are satisfied, core steps in the S3VM algorithm are to:

1. Enumerate all 2^u possible labeling combinations of unlabeled samples X_u (exponential complexity of a problem—it requires analysis of all alternatives in labeling unlabeled samples).
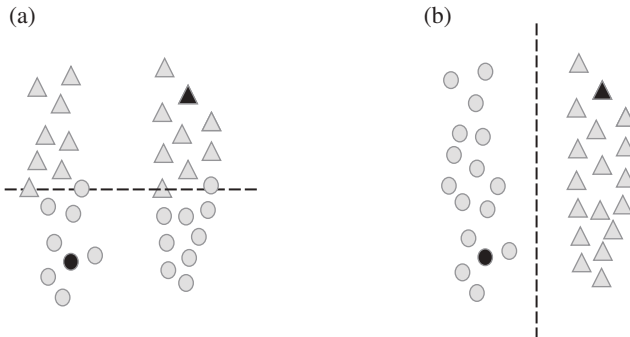
Figure 4.29. Continuity and clustering assumption determine the quality of semi-supervised learning. (a) S3VM is not the best approach. (b) S3VM is the appropriate approach.

2. Build one standard SVM for each labeling case in the previous step (and variety of X_l splitting).
3. Pick the SVM with the largest margin.

Obviously the algorithm has a problem of combinatorial explosion of alternatives in the first step! Variety of optimization methods are applied, and different S3VM implementations shows reasonable performances in practice (for example, min-cut approach in a graph of unlabeled and labeled samples). Obviously, S3VM have serious deficiency in a case of big (unlabeled) data. The methods are using extremely large time in training, and it is currently the biggest challenge in all implementations of S3VM.

In general, there is no uniform SSL solution for all applications with available both unlabeled and labeled data. Depending on available data and knowledge about the problem, we may use different techniques and approaches: from standard supervised learning (such as SVM) when there are enough labeled samples to different SSL techniques including S3VM when two assumptions in data set are satisfied. If SSL is used as a first step, verify and discuss solution by comparing with other approaches including supervised and even unsupervised learning model. Figure 4.29 gives only illustrative examples with applicable and non-applicable S3VM.

4.7   kNN: NEAREST NEIGHBOR CLASSIFIER

Unlike SVM's global classification model, k-nearest neighbor (kNN) classifier determines the decision boundary locally. For 1NN we assign each new sample to the class of its closest neighbor as it is represented in Figure 4.30a. Initially we have samples belonging to two classes (+ and −). The new sample "?" should be labeled with the class of its closest neighbor. 1NN classifier is not very robust methodology. The classification decision of each test sample relies on the class of a single training sample,

(a)   (b)
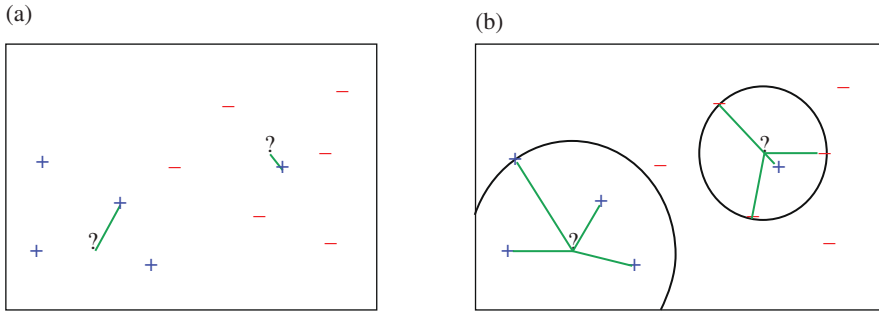
Figure 4.30. *k*-Nearest neighbor classifier. (a) *k* = 1. (b) *k* = 4

which may be incorrectly labeled or atypical. For larger k, kNN will assign new sample to the majority class of its k closest neighbors where k is a parameter of the methodology. An example for $k = 4$ is given in Figure 4.30b. kNN classifier for $k > 1$ is more robust. Larger k values help reduce the effects of noisy points within the training data set.

The rationale of kNN classification is that we expect a test sample X to have the same label as the training sample located in the local region surrounding X. KNN classifiers are lazy learners, that is, models are not built explicitly unlike SVM and the other classification models given in the following chapters. Training a kNN classifier simply consists of determining k. In fact, if we preselect a value for k and do not preprocess given samples, then kNN approach requires no training at all. kNN simply memorizes all samples in the training set and then compares the test sample to them. For this reason, kNN is also called *memory-based learning* or *instance-based learning*. It is usually desirable to have as much training data as possible in machine learning. But in kNN, large training sets come with a severe efficiency penalty in classification of testing samples.

Building the kNN model is computationally cheap (just store the training data), but classifying unknown sample is relatively expensive since it requires the computation of the kNN of the testing sample to be labeled. This, in general, requires computing the distance of the unlabeled object to all the objects in the labeled set, which can be expensive particularly for large training sets. Among the various methods of supervised learning, the nearest neighbor classifier achieves consistently high performance, without a priori assumptions about the distributions from which the training examples are drawn. The reader may have noticed the similarity between the problem of finding nearest neighbors for a test sample and ad hoc retrieval methodologies. In standard information retrieval systems such as digital libraries or Web search, we search for the documents (samples) with the highest similarity to the query document represented by a set of keywords. Problems are similar, and often the proposed solutions are applicable in both disciplines.

Decision boundaries in 1NN are concatenated segments of the *Voronoi diagram* as shown in Figure 4.31. The Voronoi diagram decomposes space into Voronoi cells,
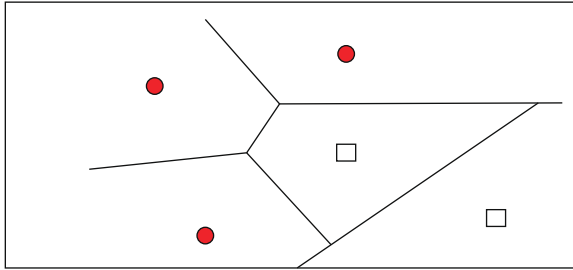
Figure 4.31. Voronoi diagram in 2D space.

where each cell consists of all points that are closer to the sample than to other samples. Assume that we have X training samples in 2D space. The diagram then partitions the 2D plane into $|X|$ convex polygons, each containing its corresponding sample (and no other), where a convex polygon is a convex region in 2D space bounded by lines. For general $k > 1$ case, consider the region in the space for which the set of kNN is the same. This again is a convex polygon and the space is partitioned into convex polygons, within each of which the set of kNN is invariant.

The parameter k in kNN is often chosen based on experience or knowledge about the classification problem at hand. It is desirable for k to be odd to make ties less likely. $k = 3$ and $k = 5$ are common choices, but much larger values up to 100 are also used. An alternative way of setting the parameter is to select k through the iterations of testing process and select k that gives best results on testing set.

Time complexity of the algorithm is linear in the size of the training set as we need to compute the distance of each training sample from the new test sample. Of course, the computing time goes up as k goes up, but the advantage is that higher values of k provide smoothing of the classification surface that reduces vulnerability to noise in the training data. At the same time high value for k may destroy the locality of the estimation since farther samples are taken into account, and large k increases the computational burden. In practical applications, typically, k is in units or tens rather than in hundreds or thousands. The nearest neighbor classifier is quite simple algorithm, but very computationally intensive especially in the testing phase. The choice of the distance measure is another important consideration. It is well known that the Euclidean distance measure become less discriminating as the number of attributes increases, and in some cases it may be better to use cosine or other measures rather than Euclidean distance.

Testing time of the algorithm is independent of the number of classes, and kNN therefore has a potential advantage for classification problems with multiple classes. For the example in Figure 4.32, we have three classes ($\omega_1, \omega_2, \omega_3$) represented by a set of training samples, and the goal is to find a class label for the testing sample x_u. In this case, we use the Euclidean distance and a value of $k = 5$ neighbors as the threshold. Of the five closest neighbors, four belong to ω_1 class and one belongs to ω_3 class, so x_u is assigned to ω_1 as the predominant class in the neighborhood.
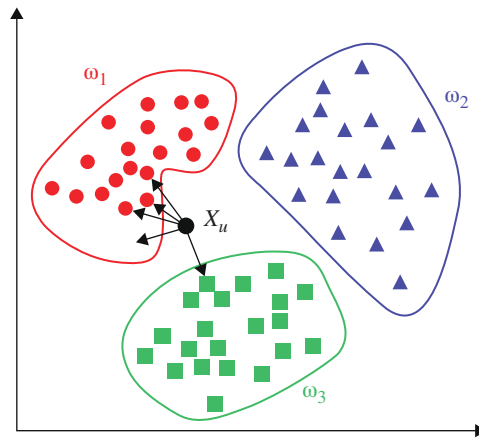
Figure 4.32. Nearest neighbor classifier for $k = 5$.

In summary, kNN classifier only requires a parameter k, a set of labeled training samples, and a metric measure for determining distances in n-dimensional space. kNN classification process is usually based on the following steps:

- Determine parameter k—number of nearest neighbors.
- Calculate the distance between each testing sample and all the training samples.
- Sort the distance and determine nearest neighbors based on the kth threshold.
- Determine the category (class) for each of the nearest neighbors.
- Use simple majority of the category of nearest neighbors as the prediction value of the testing sample classification.

There are many techniques available for improving the performance and speed of a nearest neighbor classification. One solution is to choose a subset of the training data for classification. The idea of the *condensed nearest neighbor (CNN)* is to select the smallest subset Z of training data X such that when Z is used instead of X, error in classification of new testing samples does not increase. 1NN is used as the nonparametric estimator for classification. It approximates the classification function in a piecewise linear manner. Only the samples that define the classifier need to be kept. Other samples, inside regions, need not to be stored because they belong to the same class. An example of CNN classifier in 2D space is given in Figure 4.33. Greedy CNN algorithm is defined with the following steps:

1. Start with empty set Z.
2. Passing samples from X one by one in a random order, and check whether they can be classified correctly by instances in Z.
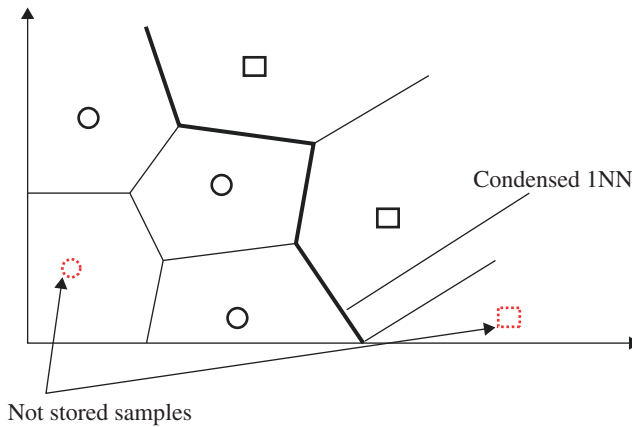
Figure 4.33. CNN classifier in 2D space.
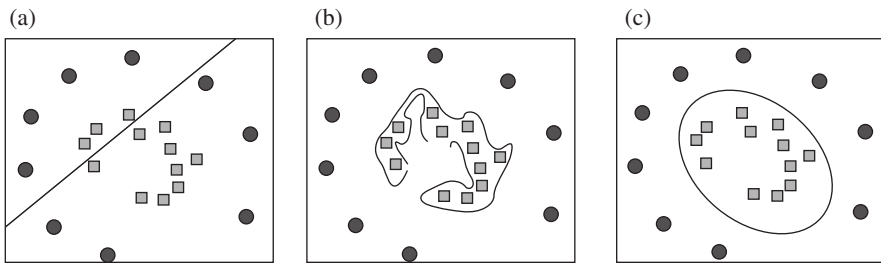
Figure 4.34. Trade-off between model complexity and the amount of data. (a) Too simple model. (b) Too complex model. (c) Appropriate model.

3. If a sample is misclassified, it is added to Z; if it is correctly classified, Z is unchanged.
4. Repeat a few times over training data set until Z is unchanged. Algorithm does not guarantee minimum subset for Z.

kNN methodology is relatively simple and could be applicable in many real-world problems. Still, there are some methodological problems such as scalability, "curse of dimensionality," influence of irrelevant attributes, weight factors in the distance measure, weight factors for votes of k neighbors, etc.

4.8   MODEL SELECTION VS. GENERALIZATION

We assume that the empirical data is given according to an unknown probability distribution. The question arises as to whether a finite set of empirical data includes sufficient information such that the underlying regularities can be learned and

represented with a corresponding model. A positive answer to this question is a necessary condition for the success of any learning algorithm. A negative answer would yield the consequence that a learning system may remember the empirical data perfectly and the system may have an unpredictable behavior with unseen data.

We may start discussion about appropriate model selection with an easy problem of learning a Boolean function from examples. Assume that both inputs and outputs are binary. For d inputs, there are 2^d different samples, and there are 2^{2d} possible Boolean functions as outputs. Each function is potential hypothesis h_i. In a case of two inputs x_1 and x_2, there are $2^4 = 16$ different hypotheses as it is given in Table 4.1.

Each training (learning) sample with two inputs and one output value (x_1, x_2, o) removes half the hypotheses (h_i). For example, sample $(0, 0, 0)$ removes h_9 to h_{16} because these hypotheses have output value 1 for the input pair $(0, 0)$. This is one way of interpreting learning: we start with all possible hypotheses, and as we see more training samples, we remove non-consistent hypotheses. After seeing N samples, there remain $2^{2d - N}$ possible hypotheses, i.e. Boolean functions as a model for a given data set. In reality, where all inputs are usually not binary but with k different values $(k \gg)$, and also data are high dimensional $(d \gg)$, then $k^d \gg N$. The number of samples for real-world data is significantly lower than the number of hypotheses (or the number of potential models). Therefore, data set by itself is not sufficient to find a unique solution–model. There is still huge number of hypotheses. We have to make some extra assumptions to reach a unique solution with the given data (N samples). We call these assumptions inductive bias (principle) of the learning algorithm. It influences a model selection. The main question is how well a model trained on training data set predicts the right output for new samples (not available in training data!), and it represents essential requirement for model generalization. For best generalization, we should match the complexity of the hypothesis class with the complexity of the function underlying training data. We made in the learning process a trade-off between the complexity of the hypothesis, the amount of data, and the generalization error of new samples. Therefore, building data-mining model is not straightforward procedure, but a very sensitive process requiring in many cases feedback information for multiple mining iterations.

TABLE 4.1. Boolean Functions as Hypotheses

Inputs		Hypotheses			
x_1	x_2	h_1	h_2	...	h_{16}
0	0	0	0		1
0	1	0	0		1
1	0	0	0		1
1	1	0	1		1

In the final phase of the data-mining process, when the model is obtained using one or more inductive-learning techniques, one important question still exists. How does one verify and validate the model? At the outset, let us differentiate between *validation* and *verification*.

Model *validation* is substantiating that the model, within its domain of applicability, behaves with satisfactory accuracy consistent with the objectives defined by the users. In other words, in model validation, we substantiate that the data has transformed into the model and that it has sufficient accuracy in representing the observed system. Model validation deals with building the *right* model, the model that corresponds to the system. Model *verification* is substantiating that the model is transformed from the data as intended into new representations with sufficient accuracy. Model verification deals with building the model *right*, the model that corresponds correctly to the data.

Model validity is a necessary but insufficient condition for the credibility and acceptability of data-mining results. If, for example, the initial objectives are incorrectly identified or the data set is improperly specified, the data-mining results expressed through the model will not be useful; however, we may still find the model valid. We can claim that we conducted an "excellent" data-mining process, but the decision-makers will not accept our results and we cannot do anything about it. Therefore, we have always to keep in mind, as it has been said, that a problem correctly formulated is a problem half-solved. Albert Einstein once indicated that the correct formulation and preparation of a problem was even more crucial than its solution. The ultimate goal of a data-mining process should not be just to produce a model for a problem at hand, but to provide one that is sufficiently credible and accepted and implemented by the decision-makers.

The data-mining results are validated and verified by the testing process. Model testing is demonstrating that inaccuracies exist or revealing the existence of errors in the model. We subject the model to test data or test cases to see if it functions properly. "Test failed" implies the failure of the model, not the test. Some tests are devised to evaluate the behavioral accuracy of the model (i.e., validity), and some tests are intended to judge the accuracy of data transformation into the model (i.e., verification).

The objective of a model obtained through the data-mining process is to classify/predict new instances correctly. The commonly used measure of a model's quality is predictive accuracy. Since new instances are supposed not to be seen by the model in its learning phase, we need to estimate its predictive accuracy using the true error rate. The true error rate is statistically defined as the error rate of the model on an asymptotically large number of new cases that converge to the actual population distribution. In practice, the true error rate of a data-mining model must be estimated from all the available samples, which are usually split into training and testing sets. The model is first designed using training samples, and then it is evaluated based on its performance on the test samples. In order for this error estimate to be reliable in predicting future model performance, not only should the training and the testing sets be sufficiently large, but they must also be independent. This requirement of independent training and test samples is still often overlooked in practice.

How should the available samples be split to form training and test sets? If the training set is small, then the resulting model will not be very robust and will have low generalization ability. On the other hand, if the test set is small, then the confidence in the estimated error rate will be low. Various methods are used to estimate the error rate. They differ in how they utilize the available samples as training and test sets. If the number of available samples is extremely large (say, 1 million), then all these methods are likely to lead to the same estimate of the error rate. If the number of samples is smaller, then the designer of the data-mining experiments has to be very careful in splitting data. There are no good guidelines available on how to divide the samples into subsets. No matter how the data are split, it should be clear that different random splits, even with the specified size of training and testing sets, would result in different error estimates.

Let us discuss different techniques, usually called *resampling methods*, for splitting data sets into training and test samples. The main advantage of using the resampling approach over the analytical approach for estimating and selecting models is that the former does not depend on assumptions about the statistical distribution of the data or specific properties of approximating functions. The main disadvantages of resampling techniques are their high computational effort and the variation in estimates depending on the resampling strategy.

The basic approach in model estimation is first to prepare or to learn a model using a portion of the training data set and then to use the remaining samples to estimate the prediction risk for this model. The first portion of the data is called a learning set, and the second portion is a validation set, also called a testing set. This *naïve strategy* is based on the assumption that the learning set and the validation set are chosen as representatives of the same unknown distribution of data. This is usually true for large data sets, but the strategy has an obvious disadvantage for smaller data sets. With a smaller number of samples, the specific method of splitting the data starts to have an impact on the accuracy of the model. The various methods of resampling are used for smaller data sets, and they differ according to the strategies used to divide the initial data set. We will give a brief description of the resampling methods that are common in today's data-mining practice, and a designer of a data-mining system will have to make a selection based on the characteristics of the data and the problem:

1. *Resubstitution method*—This is the simplest method. All the available data are used for training as well as for testing. In other words, the training and testing sets are the same. Estimation of the error rate for this "data distribution" is optimistically biased (estimated error is often smaller than could be expected in real applications of the model), and therefore the method is very seldom used in real-world data-mining applications. This is especially the case when the ratio of sample size to dimensionality is small.

2. *Holdout method*—Half the data, or sometimes two thirds of the data, is used for training, and the remaining data is used for testing. Training and testing sets are independent, and the error estimation is pessimistic. Different

    partitioning will give different estimates. A repetition of the process, with different training and testing sets randomly selected, and integration of the error results into one standard parameter, will improve the estimate of the model.

3. *Leave-one-out method*—A model is designed using $(n - 1)$ samples for training and evaluated on the one remaining sample. This is repeated n times with different training sets of size $(n - 1)$. This approach has large computational requirements because n different models have to be designed and compared.

4. *Rotation method (n-fold cross-validation)*—This approach is a compromise between holdout and leave-one-out methods. It divides the available samples into P disjoint subsets, where $1 \leq P \leq n$. $(P - 1)$ subsets are used for training and the remaining subset for testing. This is the most popular method in practice, especially for problems where the number of samples is relatively small.

5. *Bootstrap method*—This method resamples the available data with replacements to generate a number of "fake" data sets of the same size as the given data set. The number of these new sets is typically several hundreds. These new training sets can be used to define so-called bootstrap estimates of the error rate. Experimental results have shown that the bootstrap estimates can outperform the cross-validation estimates. This method is especially useful in small data set situations.

4.9   MODEL ESTIMATION

A model realized through the data-mining process using different inductive-learning techniques might be estimated using the standard error rate parameter as a measure of its performance. This value expresses an approximation of the true error rate, a parameter defined in STL. The error rate is computed using a testing data set obtained through one of applied resampling techniques. In addition to the accuracy measured by the error rate, data-mining models can be compared with respect to their speed, robustness, scalability, and interpretability, and all these parameters may have an influence on the final verification and validation of the model. In the short overview that follows, we will illustrate primary the characteristics of the error rate parameter for classification tasks; similar approaches and analyses are possible for other common data-mining tasks.

    The computation of error rate is based on counting of errors in a testing process. These errors are, for a classification problem, simply defined as misclassification (wrongly classified samples). If all errors are of equal importance, an error rate R is the number of errors E divided by the number of samples S in the testing set:

$$R = \frac{E}{S}$$

    The accuracy AC of a model is a part of the testing data set that is classified correctly, and it is computed as one minus the error rate:

$$AC = 1 - R = \frac{(S-E)}{S}$$

For standard classification problems, there can be as many as $m^2 - m$ types of errors, where m is the number of classes.

Two tools commonly used to assess the performance of different classification models are the *confusion matrix* and the *lift chart*. A confusion matrix, sometimes called a classification matrix, is used to assess the prediction accuracy of a model. It measures whether a model is confused or not, that is, whether the model is making mistakes in its predictions. The format of a confusion matrix for a two-class case with classes yes and no is shown in Table 4.2.

If there are only two classes (positive and negative samples, symbolically represented with T and F or with 1 and 0), we can have only two types of errors:

1. It is expected to be T, but it is classified as F: these are false negative errors (C: False−).
2. It is expected to be F, but it is classified as T: these are false positive errors (B: False+).

If there are more than two classes, the types of errors can be summarized in a confusion matrix, as shown in Table 4.3. For the number of classes $m = 3$, there are six types of errors ($m^2 - m = 3^2 - 3 = 6$), and they are represented in bold type in Table 4.3. Every class contains 30 samples in this example, and the total is 90 testing samples.

TABLE 4.2. Confusion Matrix for Two-Class Classification Model

Predicted Class	Actual Class	
	Class 1	Class 2
Class 1	**A**: True +	**B**: False +
Class 2	**C**: False −	**D**: True −

TABLE 4.3. Confusion Matrix for Three Classes

Classification Model	True Class			Total
	0	1	2	
0	28	**1**	**4**	33
1	**2**	28	**2**	32
2	**0**	**1**	24	25
Total	*30*	*30*	*30*	*90*

TABLE 4.4. Evaluation Metrics for Confusion Matrix 2×2

Evaluation Metrics	Computation Using Confusion Matrix
True positive rate (TP)	$TP = A/(A + C)$
False positive rate (FP)	$FP = B/(B + D)$
Sensitivity (SE)	$SE = TP$
Specificity (SP)	$SP = 1\text{-}FP$
Accuracy (AC)	$AC = (A + D)/(A + B + C + D)$
Recall (R)	$R = A/(A + C)$
Precision (P)	$P = A/(A + B)$
F-measure (F)	$F = 2PR/(P+R)$

The error rate for this example is

$$R = \frac{E}{S} = \frac{10}{90} = 0.11$$

and the corresponding accuracy is

$$AC = 1 - R = 1 - 0.11 = 0.89 \ (\text{or as a percentage} : A = 89\%)$$

Accuracy is not always the best measure of the quality of the classification model. It is especially true for the real-world problems where the distribution of classes is unbalanced, for example, if the problem is classification of healthy person from these with the disease. In many cases the medical database for training and testing will contain mostly healthy person (99%) and only small percentage of people with disease (about 1%). In that case, no matter how good the accuracy of a model is estimated to be, there is no guarantee that it reflects the real world. Therefore, we need other measures for model quality. In practice, several measures are developed, and some of best known are presented in Table 4.4. Computation of these measures is based on parameters A, B, C, and D for the confusion matrix in Table 4.2. Selection of the appropriate measure depends on the application domain, and for example, in medical field, the most often used are measures: sensitivity and specificity.

Previous measures are primarily developed for classification problems where the output of the model is expected to be a categorical variable. If the output is numerical, several additional prediction accuracy measures for regression problems are defined. It is assumed that the prediction error for each sample e_i is defined as the difference between its actual output Y_a value and predicted Y_p value:

$$e_i = Y_{ai} - Y_{pi}.$$

Based on this standard error measure for each sample, several predictive accuracy measures are defined for the entire data set:

1. *MAE* (mean absolute error) = $1/n \sum |e_i|$
   where *n* is number of samples in the data set. It represents average error through all available samples for building the model.
2. *MAPE* (mean absolute percentage error) = $100\% * 1/n \sum |e_i/Y_{ai}|$
   This measure gives in percentages how the prediction deviate in average form the actual value.
3. *SSE* (sum of squared errors) = $\sum e_i^2$
   This measure may become very large, especially if the number of samples is large. ———
4. *RMSE* (root mean squared error) = $\sqrt{SSE/n}$
   RMSE is the standard deviation of the residuals (prediction errors), where residuals are a measure of how far are samples from the regression model. It is a measure of how spread out these residuals are. In other words, it tells us how concentrated the data is around the model. This measure is most often used in real-world applications.

So far we have considered that every error is equally bad. In many data-mining applications where the result is classification model, the assumption that all errors have the same weight is unacceptable. So, the differences between various errors should be recorded, and the final measure of the error rate will take into account these differences. When different types of errors are associated with different weights, we need to multiply every error type with the given weight factor c_{ij}. If the error elements in the confusion matrix are e_{ij}, then the total cost function C (which replaces the number of errors in the accuracy computation) can be calculated as

$$C = \sum_{i=1}^{m} \sum_{j=1}^{m} c_{ij}\, e_{ij}$$

In many data-mining applications, it is not adequate to characterize the performance of a model by a single number that measures the overall error rate. More complex and global measures are necessary to describe the quality of the model. A lift chart, sometimes called a cumulative gains chart, is additional measure of classification model performance. It shows how classification results are changed by applying the model to different segments of a testing data set. This change ratio, which is hopefully the increase in response rate, is called the "lift." A lift chart indicates which subset of the data set contains the greatest possible proportion of positive responses or accurate classification. The higher the lift curve is from the baseline, the better the performance of the model since the baseline represents the null model, which is no model at all. To explain a lift chart, suppose a two-class prediction where the outcomes

(a)

(b)
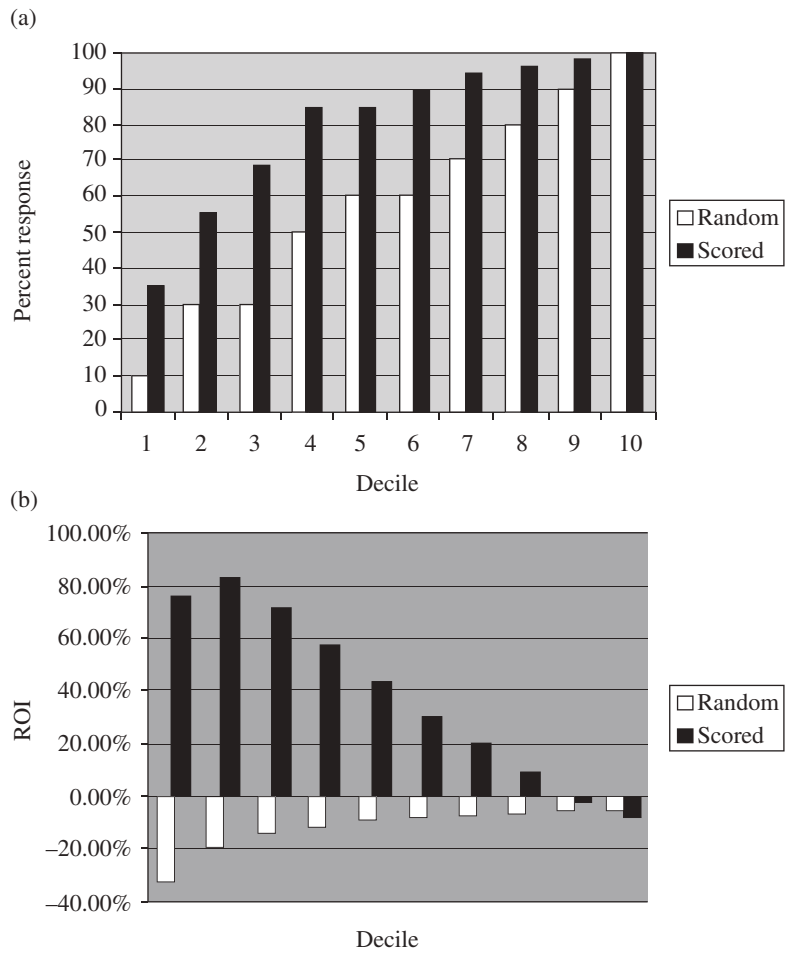
Figure 4.35. Assessing the performances of data-mining model. (a) Lift chart. (b) ROI chart.

were yes (a positive response) or no (a negative response). To create a lift chart, instances in the testing data set are sorted in descending probability order according to the predicted probability of a positive response. When the data is plotted, we can see a graphical depiction of the various probabilities as it is represented with the black histogram in Figure 4.35a.

Sorted test samples are divided into *deciles* where each decile is a group of samples containing 10% of data set. The lift at the specific decile is the ratio between the percentage of correctly classified samples (positive response) in that decile and the percentage of the same class in the entire test population. Cumulative lift calculates lift value for all samples up to particular decile, and it is present as cumulative lift chart through deciles. The chart is evidence of the quality of the predictive model: how

much is better than random guessing or how much it is above the random cumulative chart presented with the white histogram in Figure 4.35a. The baseline, represented as the white histogram on the figure, indicates the expected result if no model were used at all. The closer the cumulative gain is to the top left corner of the chart, the better the model is performing. Note that the best model is not the one with the highest lift when it is being built with the training data. It is the model that performs the best on unseen future data.

The lift chart is a big help in evaluating the usefulness of a model. It shows how responses are changed in percentiles of testing samples population by applying the data-mining model. For example, in Figure 4.35a, instead of a 10% response rate when a random 10% of the population is treated, the response rate of a top selected 10% of the population is over 35%. The lift value is 3.5 in this case.

Another important component of interpretation is to assess financial benefits of the model. Again, a discovered model may be interesting and relatively accurate, but acting on it may cost more than the revenue or savings it generates. The return on investment (ROI) chart, given in Figure 4.35b, is a good example of how attaching values to a response and costs to a program can provide additional guidance to decision-making. Here, ROI is defined as ratio of profit to cost. Note that beyond the eighth decile (80%), or 80% of testing population, the ROI of the scored model becomes negative. It is at a maximum for this example at the second decile (20% of a samples population).

We can explain the interpretation and practical use of lift and ROI charts on a simple example of a company who wants to advertise their products. Suppose they have a large database of addresses for sending advertising materials. The question is: will they send these materials to everyone in a database? What are the alternatives? How to obtain the maximum profit from this advertising campaign? If the company has additional data about "potential" costumers in their database, they may build the predictive (classification) model about the behavior of customers and their responses to the advertisement. In estimation of the classification model, lift chart is telling the company what are potential improvements in advertising results. What are benefits if they use the model and based on the model select only the most promising (responsive) subset of database instead of sending ads to everyone? If the results of the campaign are presented in Figure 4.35a, the interpretation may be the following. If the company is sending the advertising materials to the top 10% of customers selected by the model, the expected response will be 3.5 times greater than sending the same ads to randomly selected 10% of customers. On the other hand, sending ads involves some cost, and receiving response and buying the product results in additional profit for the company. If the expenses and profits are included in the model, ROI chart shows the level of profit obtained with the predictive model. From Figure 4.35b it is obvious that the profit will be negative if ads are sent to all customers in the database. If it is sent only to 10% of top customers selected by the model, ROI will be about 70%. This simple example may be translated to large number of different data-mining applications.

While lift and ROI charts are popular in a business community for evaluation of data-mining models, scientific community "likes" a receiver operating characteristic

(ROC) curve better. What is the basic idea behind an ROC curve construction? Consider a classification problem where all samples have to be labeled with one of two possible classes. A typical example is a diagnostic process in medicine, where it is necessary to classify the patient as being with or without disease. For these types of problems, two different yet related error rates are of interest. The false acceptance rate (FAR) is the ratio of the number of test cases that are incorrectly "accepted" by a given model to the total number of cases. For example, in medical diagnostics, these are the cases in which the patient is wrongly predicted as having a disease. On the other hand, the false reject rate (FRR) is the ratio of the number of test cases that are incorrectly "rejected" by a given model to the total number of cases. In the previous medical example, these are the cases of test patients who are wrongly classified as healthy.

For the most of the available data-mining methodologies, a classification model can be tuned by setting an appropriate threshold value to operate at a desired value of FAR. If we try to decrease the FAR parameter of the model, however, it would increase the FRR and vice versa. To analyze both characteristics at the same time, a new parameter was developed, the ROC curve. It is a plot of FAR versus FRR for different threshold values in the model. This curve permits one to assess the performance of the model at various operating points (thresholds in a decision process using the available model) and the performance of the model as a whole (using as a parameter the area below the ROC curve). The ROC curve is especially useful for a comparison of the performances of two models obtained by using different data-mining methodologies. The typical shape of an ROC curve is given in Figure 4.36 where the axes are *sensitivity* (FAR) and *1-specificity* (1-FRR).

How to construct an ROC curve in practical data-mining applications? Of course, many data-mining tools have a module for automatic computation and visual representation of an ROC curve. What if this tool is not available? At the beginning we are assuming that we have a table with actual classes and predicted classes for all training samples. Usually, predicted values as an output from the model are not computed as 0 or 1 (for two-class problem) but as real values on interval [0, 1]. When we select a threshold value, we may assume that all predicted values above the threshold are 1, and all values below the threshold are 0. Based on this approximation we may compare actual class with predicted class by constructing a confusion matrix, and compute
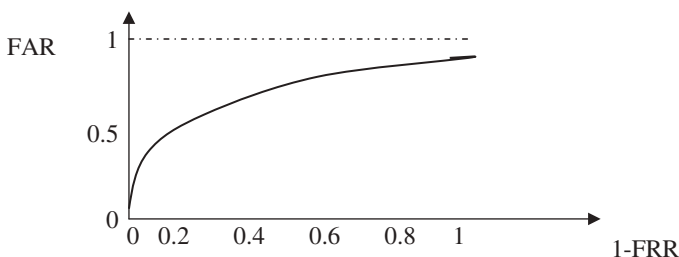
Figure 4.36. The ROC curve shows the trade-off between sensitivity and 1-specificity values.

(a)

Actual outcome

	1	0
Predicted outcome 1	8	3
0	0	9

Sensitivity: $\dfrac{8}{8+0}$ = 100%

Specificity: $\dfrac{9}{9+3}$ = 75%

(b)

Actual outcome

	1	0
Predicted outcome 1	6	2
0	2	10

Sensitivity: $\dfrac{6}{6+2}$ = 75%

Specificity: $\dfrac{10}{10+2}$ = 83%
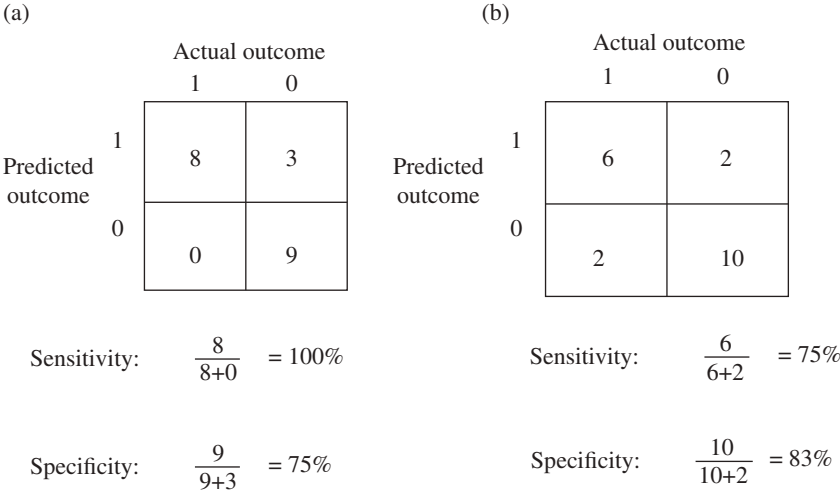
Figure 4.37. Computing points on an ROC curve. (a) The threshold = 0.5. (b) The threshold = 0.8.

sensitivity and specificity of the model for the given threshold value. In general, we can compute sensitivity and specificity for large number of different threshold values. Every pair of values (sensitivity, 1-specificity) represents one discrete point on a final ROC curve. Examples are given in Figure 4.37. Typically, for graphical presentation, we are selecting systematically threshold values, for example, starting from 0 and increasing by 0.05 until 1, and in that case we have 21 different threshold values. That will generate enough points to reconstruct an ROC curve.

When we are comparing two classification algorithms, we may compare the measures as accuracy or F-measure and conclude that one model is giving better results than the other. Also, we may compare lift charts, ROI charts, or ROC curves, and if one curve is above the other, we may conclude that corresponding model is more appropriate. But in both cases we may not conclude that there are significant differences between models or more important that one model shows better performances than the other with statistical significance. There are some simple tests that could verify these differences. The first one is McNemar's test. After testing models of both classifiers, we are creating a specific contingency table based on classification results on testing data for both models. Components of the contingency table are explained in Table 4.5.

After computing the components of the contingency table, we may apply the $\chi2$ statistic with one degree of freedom for the following expression:

$$\frac{\left[(|e_{01}-e_{10}|-1)^2\right]}{(e_{01}+e_{10})} \sim \chi2$$

TABLE 4.5. Contingency Table for McNemar's Test

e_{00}: Number of samples misclassified by both classifiers	e_{01}: Number of samples misclassified by classifier 1, but not classifier 2
e_{10}: Number of samples misclassified by classifier 2, but not classifier 1	e_{11}: Number of samples correctly classified by both classifier s

McNemar's test rejects the hypothesis that the two algorithms have the same error at the significance level α, if previous value is greater than $\chi^2_{\alpha,1}$. For example, for $\alpha = 0.05$, $\chi^2_{0.05,1} = 3.84$.

The other test is applied if we compare two classification models that are tested with k-fold cross-validation process. The test starts with the results of k-fold cross-validation obtained from k training/validation set pairs. We compare the error percentages in two classification algorithms based on errors in k validation sets, which are recorded for two models as: p_i^1 and p_i^2, $i = 1,\ldots,K$.

The difference in error rates on fold i is $P_i = p_i^1 - p_i^2$. Then, we can compute

$$m = \frac{\left[\sum_{i=1}^{k} P_i \right]}{K} \quad \text{and} \quad S^2 = \frac{\left[\sum_{i=1}^{k} (P_i - m)^2 \right]}{(K-1)}$$

We have a statistic that is t distributed with k-1 degrees of freedom and the following test:

$$\frac{\left(\sqrt{K} \times m \right)}{S} \sim t_{K-1}$$

Thus, the k-fold cross-validation paired t-test rejects the hypothesis that two algorithms have the same error rate at significance level α, if previous value is outside interval $(-t_{\alpha/2,K-1}, \ t_{\alpha/2,K-1})$. For example, the threshold values could be for $\alpha = 0.05$ and $K = 10$ or 30: $t_{0.025,9} = 2.26$, and $t_{0.025,29} = 2.05$.

Over time, all systems evolve. Thus, from time to time, the model will have to be retested, retrained and possibly completely rebuilt. Charts of the residual differences between forecasted and observed values are an excellent way to monitor model results.

4.10   IMBALANCED DATA CLASSIFICATION

Most data-mining algorithms are working the best when the number of samples in each class is approximately equal. But in a case of real-world classification problems, a scenario where the number of observations belonging to one class is significantly lower than those belonging to the other classes is not so rare at all. These are so-called

classification problems with imbalanced data, and by convention, we call the classes having more samples the *majority classes* while the ones having fewer samples the *minority classes*.

Imbalanced data sets exist in many application domains, such as recognition in advance of unreliable telecommunication customers, detection of oil spills using satellite radar images, detection of fraudulent telephone calls, filtering spam email, identifying rare diseases in medical diagnostics, and predicting natural disaster like earthquakes. For example, if the system is classifying credit card transactions, most transactions are legitimate, and only a few of them are fraudulent. What could be an available balance between majority and minority class depends on an application. In practical applications, the ratio of the small to the large classes can be drastic such as 1–100, while some applications for fraud detection reported imbalance of 1–100,000. For example, in a case of utilities fraud detection, usually for every 1000 observations, it may be recognized about 20 fraudulent observations; we may assume that the minority event rate is about 2%. Similar case is in large medical record databases used to build classification model for some rare diseases. Minority samples rate may be below 1% because of large number of patients who do not have that rare disease. In such cases with imbalanced classes, standard classifiers tend to be overwhelmed by the large majority classes and ignore the small ones. Furthermore, in medical diagnostics of a certain rare disease such as cancer, cancer is regarded as the positive, minority class, and non-cancer as negative, majority class. In this kind of applications, missing a cancer using classification model is much more serious error than the false positive error predicting cancer for healthy person. Therefore, data-mining professionals have addressed two additional issues of a class imbalance:

(a) Assign distinct costs to majority and minority training samples, and more often
(b) Rebalance the original data set, either by oversampling the minority class or undersampling the majority class.

With undersampling we randomly select a subset of samples from the majority class to match the number of samples coming from each class. The main disadvantage of undersampling is that we lose potentially relevant information given in the left-out samples. With oversampling, we randomly duplicate or combine samples from minority class to match the amount of samples in each class. While we are avoiding losing information with this approach, we also run the risk of overfitting our model by repeating some samples.

One of the most popular oversampling techniques is Synthetic Minority Oversampling Technique (SMOTE). Its main idea is to form new minority class samples by interpolating between several minority class samples that lie close together. New synthetic samples are generated in the following way:

• Randomly select a sample from a minority class, and find its nearest neighbor belonging also to the minority class.

- Take the difference between n-dimensional feature vector representing sample under consideration and a vector for its nearest neighbor.
- Multiply this difference by a random number between 0 and 1, and add it to the feature vector of a minority sample under consideration. This causes the selection of a random point along the line segment between specific samples.
- Repeat the previous steps until a balance between minority and majority classes is obtained.

This approach effectively forces the decision region of the minority class to become more general. Illustrative example of a SMOTE approach is given in Figure 4.38. Minority class samples are presented with small circles, while majority class samples in 2D space are given with X symbol. Minority class consists of only six 2D samples, and it is oversampled with additional five new synthetic samples on lines between initial members of a minority class and their nearest neighbors.

The explanation why synthetic minority oversampling improves performance while minority oversampling with replacement does not is obvious from Figure 4.38. Consider the effect on the decision regions in n-dimensional feature space when minority oversampling is done by replication (sampling with replacement), instead of introduction of synthetic samples. With replication, the decision region, which results in a classification model for the minority class, is becoming smaller and more specific as the minority samples in the region are replicated. This is the opposite of the desired effect of a model generality. On the other hand, SMOTE method of synthetic oversampling enables the classifier to build larger decision regions that contain not only original but also nearby minority class points.

Usually SMOTE approach assumes a combination of SMOTE oversampling algorithm for minority class, combined with undersampling of a majority class by randomly removing some of these samples, and making a classification problem more balanced. Of course, it is not required that the balance expect exactly 50–50%
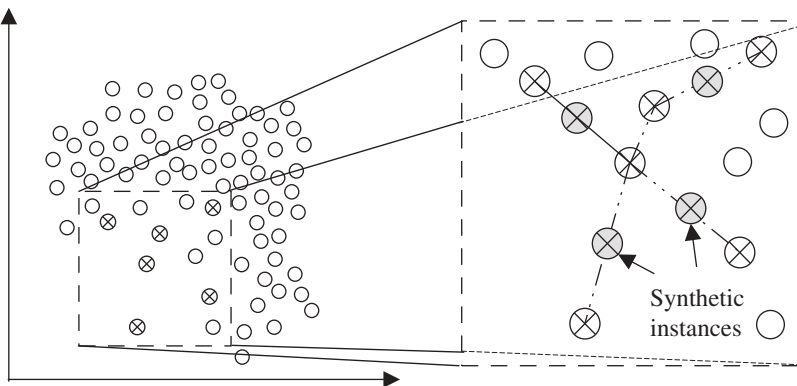
Figure 4.38. Generation of synthetic sampling SMOTE approach.

representation of classes. Final effect on the SMOTE-based classification model is combination of positive aspects of both undersampling and oversampling processes: better generalization of a minority class and no loss of useful information-related majority class. The main problem with SMOTE is weak effectiveness for high-dimensional data, but some improved version of the algorithm solved satisfactory these problems.

While the initial SMOTE approach does not handle data sets with nominal features, new version was developed to handle mixed data sets of continuous and nominal features. This approach is called Synthetic Minority Over-sampling Technique-Nominal Continuous [SMOTE-NC]. The main generalization is based on median computation. Compute in the first step the median of standard deviations of all continuous features for the minority class. If the nominal features differ between a sample and its potential nearest neighbors, then this median is included in the Euclidean distance computation. A median is used to penalize the difference of each nominal feature. For example, it is necessary to determine the distance between two six-dimensional samples, F_1 and F_2, where the first three dimensions are numerical while last three are nominal:

$$F_1 = \{1,2,3,A,B,C\}$$
$$F_2 = \{4,6,5,A,D,E\}$$

Initially, value Med is determined representing the median of the standard deviations of all continuous features of the minority class (first three features). Then, the Euclidean distance between the samples F_1 and F_2 is determined as

$$\text{Distance}(F_1, F_2) = \text{SQRT}\left((1-4)^2 + (2-6)^2 + (3-5)^2 + 0 + \text{Med}^2 + \text{Med}^2\right)$$

The median term Med is included in the distance twice for two nominal features with different values in samples F_1 and F_2. For the first nominal feature, because the values are the same (both samples have value A), the distance is equal to 0.

Traditionally, accuracy is the most commonly used measure for the quality of the classification model. However, for classification with the class imbalance problem, accuracy is no longer a proper measure since the rare class has very little impact on accuracy as compared with the majority class. For example, in a problem where a rare class is represented by only 1% of the training data, a simple classification strategy can be to predict the majority class label for every sample. This simplified approach will achieve a high accuracy of 99%. However, this measurement is meaningless to some applications where the learning is concentrated on the identification of the rare minority cases. The nature of some applications requires a fairly high rate of correct detection in the minority class allowing as a balance a small error rate in the majority class.

Because the accuracy is not meaningful for imbalanced data classification problems, two additional measures are commonly used as a replacement: *precision* (also

called positive predictive value) and *recall* (also called sensitivity). These two measures may be replaced with the single one *F*-measure (as it is given in Table 4.4):

$$\text{F-measure} = \frac{2 \times \text{precision} \times \text{recall}}{(\text{precision} + \text{recall})}$$

F-measure represents the harmonic mean of precision and recall, and it tends to be closer to the smaller of the two values. As a consequence, high *F*-measure value ensures that both recall and precision are reasonably high.

When the performance of both classes is concerned in imbalanced applications, both true positive rate (TPrate) and true negative rate (TNrat*e*) are expected to be high simultaneously. Additional measure is suggested for these cases. It is the *geometric mean* or *G-mean* measure defined as

$$\text{G-mean} = (\text{TPrate} \times \text{TNrate})^{1/2} = (\text{sensitivity} \times \text{specificity})^{1/2}$$

G-mean measures the balanced performance of a learning algorithm between majority and minority classes. The final decision in a model selection should consider a combination of different measures instead of relying on a single measure. To minimize imbalance-biased estimates of performance, it is recommended to report both the obtained metric values for selected measures and the degree of imbalance in the data.

4.11   90% ACCURACY … NOW WHAT?

Often forgotten in texts on data mining is a discussion of the deployment process. Any data-mining student may produce a model with relatively high accuracy over some small data set using the tools available. However, an experienced data miner sees beyond the creation of a model during the planning stages. There needs to be a plan created to evaluate how useful a data-mining model is to a business and how the model will be rolled out. In a business setting the value of a data-mining model is not simply the accuracy, but how that model can impact the bottom line of a company. For example, in fraud detection, algorithm A may achieve an accuracy of 90%, while algorithm *B* achieves 85% on training data. However, an evaluation of the business impact of each may reveal that algorithm A would likely underperform algorithm *B* because of larger number of very expensive false negative cases. Additional financial evaluation may recommend algorithm *B* for the final deployment because with this solutions company saves more money. A careful analysis of the business impacts of data-mining decisions gives much greater insight of a data-mining model.

In this section two case studies are summarized. The first case study details the deployment of a data-mining model that improved the efficiency of employees in finding fraudulent claims at an insurance company in Chile. The second case study

involves a system deployed in hospitals to aid in counting compliance with industry standards in caring for individuals with cardiovascular disease (CVD).

4.11.1 Insurance Fraud Detection

In 2005, insurance company Banmedica S.A. of Chile received 800 digital medical claims per day. The process of identifying fraud was entirely manual. Those responsible for identifying fraud had to look one by one at medical claims to find fraudulent cases. Instead it was hoped that data-mining techniques would aid in a more efficient discovery of fraudulent claims.

The first step in the data-mining process required that the data-mining experts gain a better understanding of the processing of medical claims. After several meetings with medical experts, the data-mining experts were able to better understand the business process as it related to fraud detection. They were able to determine the current criteria used in manually discriminating between claims that were approved, rejected, and modified. A number of known fraud cases were discussed and the behavioral patterns that revealed these documented fraud cases.

Next, two data sets were supplied. The first data set contained 169 documented cases of fraud. Each fraudulent case took place over an extended period of time showing that time was an important factor in these decisions as cases developed. The second data set contained 500,000 medical claims with labels supplied by the business of "approved," "rejected," or "reduced."

Both data sets were analyzed in detail. The smaller data set of known fraud cases revealed that these fraudulent cases all involved on a small number of medical professionals, affiliates, and employers. From the original paper, "19 employers and 6 doctors were implicated with 152 medical claims." The labels of the larger data set were revealed to be not sufficiently accurate for data mining. Contradictory data points were found. A lack of standards in recording these medical claims with a large number of missing values contributed to the poorly labeled data set. Instead of the larger 500,000 point data set, the authors were "forced" to rebuild a subset of this data. This required manual labeling of the subset.

The manual labeling would require a much smaller set of data points to be used from the original 500,000. To cope with a smaller set of data points, the problem was split into four smaller problems, namely, identifying fraudulent medical claims, affiliates, medical professionals, and employers. Individual data sets were constructed for each of these four subtasks ranging in size from 2838 samples in the medical claims task to 394 samples in the employer subtask. For each subtask a manual selection of features was performed. This involved selecting only one feature from highly correlated features, replacing categorical features with numerical features, and designing new features that "summarize temporal behavior over an extended time span." The original 125 features were paired down to between 12 and 25 features depending on the subtask. Additionally, the output of all other subtasks became inputs to each subtask, thus providing feedback to each subtask. Lastly, 2% of outliers were removed and features were normalized.

When modeling the data, it was found that initially the accuracy of a single neural network on these data sets could vary by as much as 8.4%. Instead of a single neural network for a particular data set, a committee of neural networks was used. Each data set was also divided into a training set, a validation set, and a testing set to avoid overfitting the data. At this point it was also decided that each of the four models would be retrained monthly to keep up with the ever-evolving process of fraud.

Neural networks and committees of neural networks output scores rather than an absolute fraud classification. It was necessary that the output be thresholded. The threshold was decided after accounting for personnel costs, false alarm costs, and the cost of not detecting a particular instance of fraud. All of these factors figured into an ROC curve to decide upon acceptable false and true positive rates. When the medical claims model using the input of the other three subtasks scored a medical claim above the chosen threshold, then a classification of fraud is given to that claim. The system was tested on a historical data set of 8819 employers that contains 418 instances of fraud. After this historical data set was split into training, validation, and test set, the results showed the system identified 73.4% of the true fraudsters and had a false positive rate of 6.9%.

The completed system was then run each night, giving each new medical claim a fraud probability. The claims are then reviewed being sorted by the given probabilities. There were previously very few documented cases of fraud. After implementation there were approximately 75 rejected claims per month. These newly found cases of fraud accounted for nearly 10% of the raw overall costs to the company! Additionally, the culture of fraud detection changed. A taxonomy of the types of fraud was created, and further improvements were made on the manual revision process. The savings covered the operational costs and increased the quality of health coverage.

Overall this project was a big success. The authors spent a lot of time first understanding the problem and second analyzing the data in detail before the data was modeled. The final models produced were analyzed in terms of real business costs. In the end the results showed that the costs of the project were justified and Banmedica S.A. greatly benefited from the final system.

4.11.2   Improving Cardiac Care

CVD leads to nearly one million deaths per year in the United States or 38% of all deaths in the United States. Additionally, in 2005 the estimated cost of CVD was $394 billion compared with an estimated $190 billion on all cancers combined. CVD is a real problem that appears to be growing in the number of lives claimed and the percent of the population that will be directly affected by this disease. Certainly we can gain a better understanding of this disease. There already exist guidelines for the care of patients with CVD that were created by panels of experts. With the current load on the medical system, doctors are able to only spend a short amount of time with each patient. With the large number of guidelines that exists, it is not reasonable to expect that doctors will follow every guideline on every patient. Ideally a system would aid a doctor in following the given guidelines without adding additional overheads.

This case study outlines the use and deployment of a system called REMIND meant to both find patients at need within the system and enable a better tracking of when patients are being cared for according to guidelines. Currently two main types of records are kept for each patient, financial and clinical. The financial records are used for billing. These records use standardized codes (ICD-9, for example) for doctor assessments and drugs prescribed. This standardization makes it straightforward for computer systems to extract information from these records and to be used by data-mining processes. However, it has been found that these codes are accurate only 60–80% of the time for various reasons. One reason is that when these codes are used for billing, though two conditions are nearly identical in symptoms and prescriptions, the amount of money that will be paid out by an insurer may be very different. The other form of records kept is clinical records. Clinical records are made up of unstructured text and allow for the transfer of knowledge about a patient's condition and treatments from one doctor to another. These records are much more accurate, but are not in a form that is easily used by automated computer systems.

It is not possible that with great demands on the time of doctors and nurses, additional data may be recorded specifically for this system. Instead, the REMIND system combines data from all available systems. This includes extracting knowledge from the unstructured clinical records. The REMIND system combines all available sources of data available and then using redundancy in data the most likely state of the patient is found. For example, to determine a patient is diabetic one, may use any of the following pieces of data: billing code 250.xx for diabetes, a free text dictation identifying a diabetic diagnosis, a blood sugar value >300, a treatment of insulin or oral antidiabetic, or a common diabetic complication. The likelihood of the patient having diabetes increases as more relevant information is found. The REMIND system uses extracted information from all possible data sources, combined in a Bayesian network. The various outputs of the network are used along with temporal information to find the most probable sequence of states with a predefined disease progression modeled as a Markov model. The probabilities and structure of the Bayesian network are provided as domain knowledge provided beforehand by experts and tunable per deployment. The domain knowledge in the REMIND system is fairly simple as stated by the author of the system. Additionally, by using a large amount of redundancy, the system performs well for a variety of probability settings and temporal settings for the disease progression. However, before a wide distribution of the REMIND system, a careful tuning of all the parameters must take place.

One example deployment was to the South Carolina Heart Center where the goal was to identify among 61,027 patients those at risk of sudden cardiac death (SCD). Patients who have previously suffered a myocardial infarction (heart attack, abbreviated as MI) are at the highest risk of SCD. A study was performed on the efficacy of implantable cardioverter defibrillators (ICDs). It was found that patients, with a prior MI and low ventricular function, had their 20-month mortality rate drop from 19.8 to 14.2%. This implantation is now a standard recommendation. Previous to the REMIND system, one had two options to find who would require the implantation of an ICD. The first option is to manually review the records of all patients to identify

those who were eligible for an ICD. This would be extremely time consuming considering the large number of records. The other approach would be to evaluate the need for an ICD during regular checkups. However, not all patients come in for regular checkups, and there would be a high chance that not every patient would be carefully considered for the need of an ICD. The REMIND system was given access to billing and demographics databases and transcribed free text including histories, physical reports, physician progress notes, and lab reports. From this data the REMIND system processed all records on a single laptop in 5 hours and found 383 patients who qualified for an ICD.

To check the validity of the 383 found patients, 383 randomly chosen patients were mixed with the 383 found previously. Then 150 patients were chosen from the 766 patients' sample. An electrophysiologist manually reviewed the 150 patients, being blinded to the selection made by the REMIND system. The REMIND system concurred with the manual analysis in 94% (141/150) of the patients. The sensitivity was 99% (69/70) and the specificity was 90% (72/80). Thus it was shown that the REMIND system could fairly accurately identify at risk patients in a large database. An expert was required to verify the results of the system. Additionally, all of the patients found would be reviewed by a physician before implantation would occur.

From the previous cases we see that a great deal of time was required from experts to prepare data for mining, and careful analysis of a model application needed to take place after deployment. Although applied data-mining techniques (neural and Bayesian networks) will be explained in the following chapters, the emphasis of these stories is on complexity of a data-mining process, and especially deployment phase, in real-world applications. The system developed for Banmedica was measured after analysis in terms of fraudulent cases found and the amount of money saved. If these numbers were not in favor of the system, then it would have been rolled back. In the case of the REMIND system, the results of the system-wide search had to be manually analyzed for accuracy. It was not enough that the rules were good, but the actual patients found needed to be reviewed.

4.12 REVIEW QUESTIONS AND PROBLEMS

1. Explain the differences between the basic types of inferences: induction, deduction, and transduction.

2. Why do we use the observational approach in most data-mining tasks?

3. Discuss situations in which we would use the interpolated functions given in Figures 4.3b, c, and d as "the best" data-mining model.

4. Which of the functions have linear parameters and are nonlinear? Explain why.
   (a) $y = ax^5 + b$
   (b) $y = a/x$
   (c) $y = a\,e^x$
   (d) $y = e^{a\,x}$

5. Explain the difference between interpolation of loss function for classification problems and for regression problems.

6. Is it possible that empirical risk becomes higher than expected risk? Explain.

7. Why is it so difficult to estimate the VC dimension for real-world data-mining applications?

8. What will be the practical benefit of determining the VC dimension in real-world data-mining applications?

9. Classify the common learning tasks explained in Section 4.4 as supervised or unsupervised learning tasks. Explain your classification.

10. Analyze the differences between validation and verification of inductive-based models.

11. In which situations would you recommend the leave-one-out method for validation of data-mining results?

12. Develop a program for generating "fake" data sets using the bootstrap method.

13. Develop a program for plotting an ROC curve based on a table of FAR–FRR results.

14. Develop an algorithm for computing the area below the ROC curve (that is a very important parameter in the evaluation of inductive-learning results for classification problems).

15. The testing data set (inputs: A, B, and C, output: class) is given together with testing results of the classification (predicted output). Find and plot two points on the ROC curve for the threshold values of 0.5 and 0.8.

A	B	C	Class (output)	Predicted output
10	2	A	1	*0.97*
20	1	B	0	*0.61*
30	3	A	1	*0.77*
40	2	B	1	*0.91*
15	1	B	0	*0.12*

16. Machine-learning techniques differ from statistical techniques in that machine-learning methods:
    (a) Typically assume an underlying distribution for the data.
    (b) Are better able to deal with missing and noisy data.
    (c) Are not able to explain their behavior.
    (d) Have trouble with large-sized data sets.

17. Explain the difference between sensitivity and specificity.

18. When do you need to use a separate validation set in addition to train and test sets?

19. In this question we will consider learning problems where each instance x is some integer in the set $X = \{1, 2, \ldots, 127\}$ and where each hypothesis $h \in H$ is an interval of the form $a \leq x \leq b$, where a and b can be any integers between 1 and 127 (inclusive), so long as $a \leq b$. A hypothesis $a \leq x \leq b$ labels instance x positive if x falls into the interval defined by a and b and labels the instance negative otherwise. Assume throughout this question that the teacher is only interested in teaching concepts that can be represented by some hypothesis in H.

    (a) How many distinct hypotheses are there in H?
    (b) Suppose the teacher is trying to teach the specific target concept $32 \leq x \leq 84$. What is the minimum number of training examples the teacher must present to guarantee that any consistent learner will learn this concept exactly?

20. Is it true that the SVM learning algorithm is guaranteed to find the globally optimal hypothesis with respect to its object function? Discuss your answer.

21. A marketing company working for a charity has developed two different models that predict the likelihood that donors will respond to a mailshot asking them to make a special extra donation. The prediction scores generated for a test set for these two models are shown in the table below.

    (a) Using a classification threshold of 0.5 and assuming that true is the positive target level, construct a confusion matrix for each of the models.
    (b) Generate a cumulative gain chart for each model.
    (c) Find the values for the McNamara's test that compare two models.

ID	Target	Model 1 Score	Model 2 Score	ID	Target	Model 1 Score	Model 2 Score
1	False	0.1026	0.2089	16	True	0.7165	0.4569
2	False	0.2937	0.0080	17	True	0.7677	0.8086
3	True	0.5120	0.8378	18	False	0.4468	0.1458
4	True	0.8645	0.7160	19	False	0.2176	0.5809
5	False	0.1987	0.1891	20	False	0.9800	0.5783
6	True	0.7600	0.9398	21	True	0.6562	0.7843
7	True	0.7519	0.9800	22	True	0.9693	0.9521
8	True	0.2994	0.8578	23	False	0.0275	0.0377
9	False	0.0552	0.1560	24	True	0.7047	0.4708
10	False	0.9231	0.5600	25	False	0.3711	0.2846
11	True	0.7563	0.9062	26	False	0.4440	0.1100
12	True	0.5664	0.7301	27	True	0.5440	0.3562
13	True	0.2872	0.8764	28	True	0.5713	0.9200
14	True	0.9326	0.9274	29	False	0.3757	0.0895
15	False	0.0651	0.2992	30	True	0.8224	0.8614

22. A nearest neighbor approach is best used:
   (a) With large-sized data sets.
   (b) When irrelevant attributes have been removed from the data.
   (c) When a generalized model of the data is desirable.
   (d) When an explanation of what has been found is of primary importance.

   Select only one choice and give additional explanations.

23. Given desired class C and population P, lift is defined as:
   (a) The probability of class C given population P divided by the probability of C given a sample taken from the population.
   (b) The probability of population P given a sample taken from P.
   (c) The probability of class C given a sample taken from population P.
   (d) The probability of class C given a sample taken from population P divided by the probability of C within the entire population P.

24. When do you need to use a separate validation set in addition to train and test sets?

25. Show that accuracy is a function of *sensitivity* and *specificity*.

4.13   REFERENCES FOR FURTHER STUDY

1. Engel, A., C. Van den Broeck, *Statistical Mechanics of Learning*, Cambridge University Press, Cambridge, UK, 2001.

   The subject of this book is the contribution of machine learning over the last decade by researchers applying the techniques of statistical mechanics. The authors provide a coherent account of various important concepts and techniques that are currently only found scattered in papers. They include many examples and exercises, making this a book that can be used with courses, or for self-teaching, or as a handy reference.

2. Cherkassky, V., F. Mulier, *Learning from Data: Concepts, Theory and Methods*, 2nd edition, John Wiley, New York, 2007.

   The book provides a unified treatment of the principles and methods for learning dependencies from data. It establishes a general conceptual framework in which various learning methods from statistics, machine learning, and other disciplines can be applied—showing that a few fundamental principles underlie most new methods being proposed today. An additional strength of this primarily theoretical book is the large number of case studies and examples that simplify and make understandable concepts in statistical learning theory.

3. Berthold, M. and D. J. Hand, eds., *Intelligent Data Analysis – An Introduction*, Springer, Berlin, Germany, 2007.

   The book is a detailed introductory presentation of the key classes of intelligent data-analysis methods including all common data-mining techniques. The first half of the

book is devoted to the discussion of classical statistical issues, ranging from basic concepts of probability and inference to advanced multivariate analyses and Bayesian methods. The second part of the book covers theoretical explanations of data-mining techniques that have their roots in disciplines other than statistics. Numerous illustrations and examples enhance the reader's knowledge about theory and practical evaluations of data-mining techniques.

4. Alpaydin A., *Introduction to Machine Learning*, 2nd edition, MIT Press, 2010.

   The goal of machine learning is to program computers to use example data or past experience to solve a given problem. Many successful applications of machine learning exist already, including systems that analyze past sales data to predict customer behavior, optimize robot behavior so that a task can be completed using minimum resources, and extract knowledge from bioinformatics data. *Introduction to Machine Learning* is a comprehensive textbook on the subject, covering a broad array of topics not usually included in introductory machine-learning texts. In order to present a unified treatment of machine-learning problems and solutions, it discusses many methods from different fields, including statistics, pattern recognition, neural networks, artificial intelligence, signal processing, control, and data mining. All learning algorithms are explained so that the student can easily move from the equations in the book to a computer program.

5. Haibo He, Yunqian Ma, *Imbalanced Learning: Foundations, Algorithms, and Applications*, 1st edition, John Wiley & Sons, Inc., 2013.

   Imbalanced learning focuses on how an intelligent system can learn when it is provided with imbalanced data. Solving imbalanced learning problems is critical in numerous data-intensive networked systems, including surveillance, security, Internet, finance, biomedical, defense, and more. Due to the inherent complex characteristics of imbalanced data sets, learning from such data requires new understandings, principles, algorithms, and tools to transform vast amounts of raw data efficiently into information and knowledge representation. The first comprehensive look at this new branch of machine learning, this book offers a critical review of the problem of imbalanced learning, covering the state of the art in techniques, principles, and real-world applications.

6. Aggarwal C., *Data Mining*, Springer, 2015.

   This textbook explores the different aspects of data mining from the fundamentals to the complex data types and their applications, capturing the wide diversity of problem domains for data-mining issues. It goes beyond the traditional focus on data-mining problems to introduce advanced data types such as text, time series, discrete sequences, spatial data, graph data, and social networks. The chapters of this book fall into one of three categories:

   - Fundamental chapters: Data mining has four main problems, which correspond to clustering, classification, association pattern mining, and outlier analysis. These chapters comprehensively discuss a wide variety of methods for these problems.

- Domain chapters: These chapters discuss the specific methods used for different domains of data such as text data, time-series data, sequence data, graph data, and spatial data.
- Application chapters: These chapters study important applications such as stream mining, Web mining, ranking, recommendations, social networks, and privacy preservation. The domain chapters also have an applied flavor.

<div align="right">

5

</div>

STATISTICAL METHODS

Chapter Objectives

- Explain methods of statistical inference commonly used in data-mining applications.
- Identify different statistical parameters for assessing differences in data sets.
- Describe the components and the basic principles of naïve Bayesian classifier and the logistic regression method.
- Introduce log-linear models using correspondence analysis of contingency tables.
- Discuss the concepts of ANOVA analysis and linear discriminant analysis of multidimensional samples.

Statistics is the science of collecting and organizing data and drawing conclusions from data sets. The organization and description of the general characteristics of data sets is the subject area of *descriptive statistics*. How to draw conclusions from data is

the subject of *statistical inference*. In this chapter, the emphasis is on the basic principles of statistical inference; other related topics will be described briefly, only enough to understand the basic concepts.

Statistical data analysis is the most well-established set of methodologies for data mining. Historically, the first computer-based applications of data analysis were developed with the support of statisticians. Ranging from one-dimensional data analysis to multivariate data analysis, statistics offered a variety of methods for data mining, including different types of regression and discriminant analysis. In this short overview of statistical methods that support the data-mining process, we will not cover all approaches and methodologies; a selection has been made of the techniques used most often in real-world data-mining applications.

5.1   STATISTICAL INFERENCE

The totality of the observations with which we are concerned in statistical analysis, whether their number is finite or infinite, constitutes what we call a *population*. The term refers to anything of statistical interest, whether it is a group of people, objects, or events. The number of observations in the population is defined as the size of the population. In general, populations may be finite or infinite, but some finite populations are so large that, in theory, we assume them to be infinite.

In the field of statistical inference, we are interested in arriving at conclusions concerning a population when it is impossible or impractical to observe the entire set of observations that make up the population. For example, in attempting to determine the average length of the life of a certain brand of light bulbs, it would be practically impossible to test all such bulbs. Therefore, we must depend on a subset of observations from the population for most statistical-analysis applications. In statistics, a subset of a population is called a *sample*, and it describes a finite data set of *n*-dimensional vectors. Throughout this book, we will call this subset of population just simply *data set* to eliminate confusion between the two definitions of sample: one, explained earlier, denoting the description of a single entity in the population, and the other, given here, referring to the subset of a population. From a given data set, we build a statistical model of the population that will help us to make inferences concerning that same population. If our inferences from the data set are to be valid, we must obtain samples that are representative of the population. Very often, we are tempted to choose a data set by selecting the most convenient members of the population. But such an approach may lead to erroneous inferences concerning the population. Any sampling procedure that produces inferences that consistently overestimate or underestimate some characteristics of the population is said to be biased. To eliminate any possibility of bias in the sampling procedure, it is desirable to choose a random data set in the sense that the observations are made independently and at random. The main purpose of selecting random samples is to elicit information about unknown population parameters.

The relation between data sets and the system they describe may be used for inductive reasoning: from observed data to knowledge of a (partially) unknown system. Statistical inference is the main form of reasoning relevant to data analysis. The theory of statistical inference consists of those methods by which one makes inferences or generalizations about a population. These methods may be categorized into two major areas: *estimation* and *tests of hypotheses*.

In *estimation*, one wants to come up with a plausible value or a range of plausible values for the unknown parameters of the system. The goal is to gain information from a data set T in order to estimate one or more parameters w belonging to the model of the real-world system $f(X, w)$. A data set T is described by the ordered n-tuples of values for variables: $X = \{X_1, X_2,\ldots,X_n\}$ (attributes of entities in population):

$$T = \{(x_{11},\ldots,x_{1n}), (x_{21},\ldots,x_{2n}),\ldots,(x_{m1},\ldots,x_{mn})\}$$

It can be organized in a tabular form as a set of samples with its corresponding feature values. Once the parameters of the model are estimated, we can use them to make predictions about the random variable Y from the initial set of attributes $Y \in X$, based on other variables or sets of variables $X^* = X - Y$. If Y is numeric, we speak about *regression*, and if it takes its values from a discrete, unordered data set, we speak about *classification*.

Once we have obtained estimates for the model parameters w from some dataset T, we may use the resulting model (analytically given as a function $f(X^*, w)$) to make predictions about Y when we know the corresponding value of the vector X^*. The difference between the prediction $f(X^*, w)$ and the real value Y is called the prediction error. It should preferably take values close to zero. A natural quality measure of a model $f(X^*, w)$, as a predictor of Y, is the expected mean squared error for the entire data set T:

$$E_T\left[(Y - f(X^*, w))^2\right]$$

In *statistical testing*, on the other hand, one has to decide whether a hypothesis concerning the value of the population characteristic should be accepted or rejected in the light of an analysis of the data set. A statistical hypothesis is an assertion or conjecture concerning one or more populations. The truth or falsity of a statistical hypothesis can never be known with absolute certainty, unless we examine the entire population. This, of course, would be impractical in most situations, sometimes even impossible. Instead, we test a hypothesis on a randomly selected data set. Evidence from the data set that is inconsistent with the stated hypothesis leads to a rejection of the hypothesis, whereas evidence supporting the hypothesis leads to its acceptance, or more precisely, it implies that the data do not contain sufficient evidence to refuse it. The structure of hypothesis testing is formulated with the use of the term *null hypothesis*. This refers to any hypothesis that we wish to test and is denoted by H_0. H_0 is only rejected if the given data set, on the basis of the applied statistical tests, contains strong

evidence that the hypothesis is not true. The rejection of H_0 leads to the acceptance of an alternative hypothesis about the population.

In this chapter, some statistical estimation and hypothesis-testing methods are described in great detail. These methods have been selected primarily based on the applicability of the technique in a data-mining process on a large data set.

5.2   ASSESSING DIFFERENCES IN DATA SETS

For many data-mining tasks, it would be useful to learn the more general characteristics about the given data set, regarding both central tendency and data dispersion. These simple parameters of data sets are obvious descriptors for assessing differences between different data sets. Typical measures of central tendency include *mean*, *median*, and *mode*, while measures of data dispersion include *variance* and *standard deviation*.

The most common and effective numeric measure of the center of the data set is the *mean* value (also called the arithmetic mean). For the set of n numeric values x_1, $x_2,\ldots,x_n$, for the given feature X, the mean is

$$\text{mean} = \frac{1}{n} \sum_{i=1}^{n} x_i$$

and it is a built-in function (like all other descriptive statistical measures) in most modern statistical software tools. For each numeric feature in the n-dimensional set of samples, it is possible to calculate the mean value as a central tendency characteristic for this feature. Sometimes, each value x_i in a set may be associated with a weight w_i, which reflects the frequency of occurrence, significance, or importance attached to the value. In this case, the weighted arithmetic mean or the weighted average value is

$$\text{mean} = \frac{\sum_{i=1}^{n} w_i x_i}{\sum_{i=1}^{n} w_i}$$

Although the mean is the most useful quantity that we use to describe a set of data, it is not the only one. For skewed data sets, a better measure of the center of data is the *median*. It is the middle value of the ordered set of feature values if the set consists of an odd number of elements, and it is the average of the middle two values if the number of elements in the set is even. If $x_1, x_2,\ldots,x_n$ represents a data set of size n, arranged in increasing order of magnitude, then the median is defined by

$$\text{median} = \begin{cases} x_{(n+1)/2} & \text{if } n \text{ is odd} \\ \dfrac{\left(x_{n/2} + x_{(n/2)+1}\right)}{2} & \text{if } n \text{ is even} \end{cases}$$

Another measure of the central tendency of a data set is the *mode*. The mode for the set of data is the value that occurs most frequently in the set. While mean and

median are characteristics of primarily numeric data sets, the mode may be applied also to categorical data, but it has to be interpreted carefully because the data are not ordered. It is possible for the greatest frequency to correspond to several different values in a data set. That results in more than one mode for a given data set. Therefore, we classify data sets as unimodal (with only one mode) and multimodal (with two or more modes). Multimodal data sets may be precisely defined as bimodal, trimodal, etc. For unimodal frequency curves that are moderately asymmetrical, we have the following useful empirical relation for numeric data sets:

$$\text{mean} - \text{mode} \leq 3 \times (\text{mean} - \text{median})$$

that may be used for an analysis of data set distribution and the estimation of one central tendency measure based on the other two.

As an example, let us analyze these three measures on the simple data set T that has the following numeric values:

$$T = \{3, 5, 2, 9, 0, 7, 3, 6\}$$

After a sorting process the same data set is given as

$$T = \{0, 2, 3, 3, 5, 6, 7, 9\}$$

The corresponding descriptive statistical measures for central tendency are

$$\text{mean}_T = \frac{(0+2+3+3+5+6+7+9)}{8} = 4.375$$

$$\text{median}_T = \frac{(3+5)}{2} = 4$$

$$\text{mode}_T = 3$$

The degree to which numeric data tend to spread is called dispersion of the data, and the most common measures of dispersion are the *standard deviation* σ and the *variance* σ^2. The variance of n numeric values $x_1, x_2, \ldots, x_n$ is

$$\sigma^2 = \left(\frac{1}{(n-1)}\right) \sum_{i=1}^{n} (x_i - \text{mean})^2$$

The standard deviation σ is the square root of the variance σ^2. The basic properties of the standard deviation σ as a measure of spread are as follows:

1. σ measures spread about the *mean* and should be used only when the *mean* is chosen as a measure of the center.
2. $\sigma = 0$ only when there is no spread in the data, i.e., when all measurements have the same value. Otherwise $\sigma > 0$.

For the data set given in our example, *variance* σ^2 and *standard deviation* σ are

$$\sigma^2 = \frac{1}{8} \cdot \sum_{i=1}^{8} (x_i - 4.375)^2$$

$$\sigma^2 = 8.5532$$

$$\sigma = 2.9246$$

In many statistical software tools, a popularly used visualization tool of descriptive statistical measures for central tendency and dispersion is a *boxplot* that is typically determined by the mean value, variance, and sometimes max and min values of the data set. In our example, the minimal and maximal values in the T set are $\min_T = 0$ and $\max_T = 9$. Graphical representation of statistical descriptors for the data set T has the form of a boxplot, given in Figure 5.1.

Analyzing large data sets requires proper understanding of the data in advance. This would help domain experts to influence the data-mining process and to properly evaluate the results of a data-mining application. Central tendency measures for a data set are valuable only for some specific distributions of data values. Therefore it is important to know characteristics of a distribution for a data set we are analyzing. The distribution of values in a data set is described according to the spread of its values. Usually, this is best done using a histogram representation; an example is given in Figure 5.2. In addition to quantifying the distribution of values for each feature, it is also important to know the global character of the distributions and all specifics. Knowing that data set has classic bell curve empowers researchers to use a broad range of traditional statistical techniques for data assessment. But in many practical cases the distributions are skewed or multimodal, and traditional interpretation of concepts such as mean value or standard deviation does not have a sense.

Part of the assessment process is determining relations between features in a data set. Simple visualization through the scatter plots gives initial estimation of these relations. Figure 5.3 shows part of the integrated scatter plot where it compared each pair
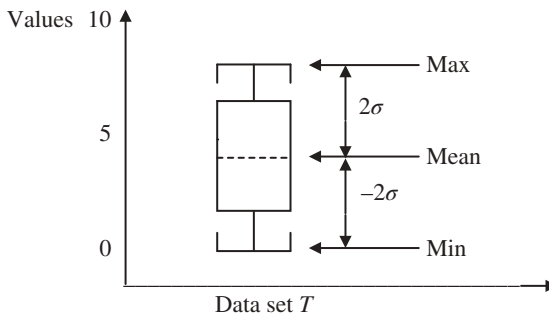
Figure 5.1. A boxplot representation of the data set T based on mean value, variance, and min and max values.
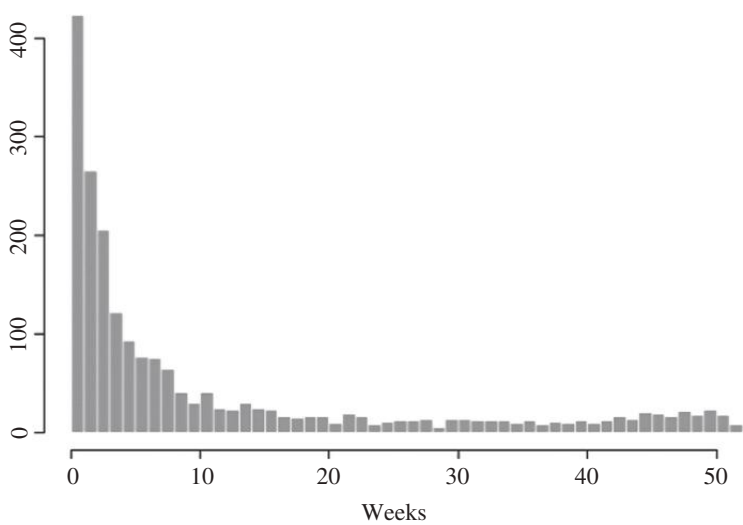
Figure 5.2. Displaying single feature distribution.
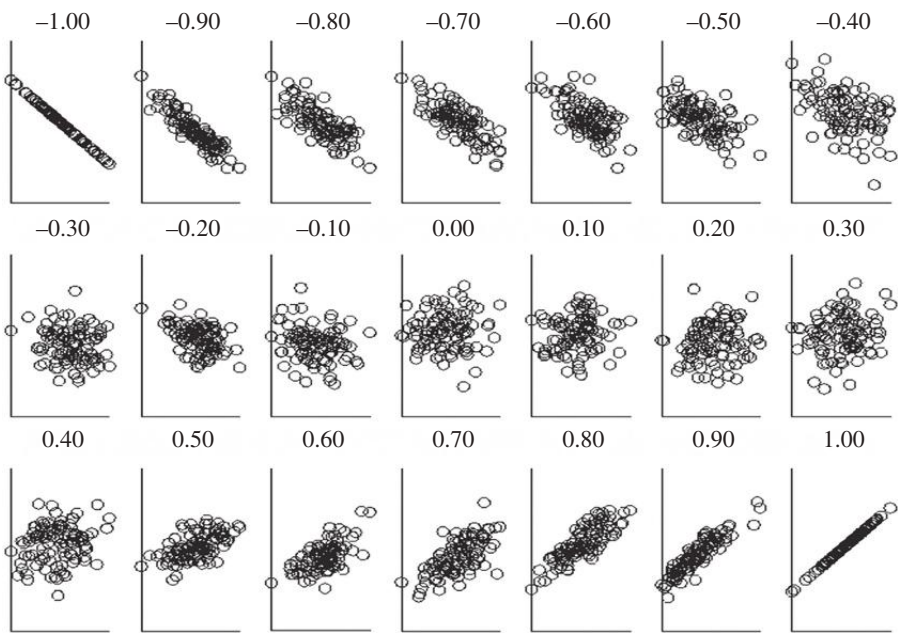
Figure 5.3. Scatter plots showing the correlation between features from –1 to 1.

of features. This visualization technique is available in most integrated data-mining tools. Quantification of these relations is given through the correlation factor.

These visualization processes are part of data understanding phase, and they are essential in better preparation for data mining. This human interpretation helps to obtain a general view of the data. It is also possible to identify abnormal or interesting characteristics, such as anomalies.

5.3   BAYESIAN INFERENCE

It is not hard to imagine situations in which the data are not the only available source of information about the population or about the system to be modeled. The Bayesian method provides a principled way to incorporate this external information into the data-analysis process. This process starts with an already given probability distribution for the analyzed data set. As this distribution is given before any data is considered, it is called a *prior distribution*. The new data set updates this prior distribution into a *posterior distribution*. The basic tool for this updating is the Bayes theorem.

The Bayes theorem represents a theoretical background for a statistical approach to inductive-inferencing classification problems. We will explain first the basic concepts defined in the Bayes theorem and then use this theorem in the explanation of the naïve Bayesian classification process, or the *simple Bayesian classifier*.

Let X be a data sample whose class label is unknown. Let H be some hypothesis, such that the data sample X belongs to a specific class C. We want to determine $P(H/X)$, the probability that the hypothesis H holds given the observed data sample X. $P(H/X)$ is the posterior probability representing our confidence in the hypothesis after X is given. In contrast, $P(H)$ is the prior probability of H for any sample, regardless of how the data in the sample looks. The posterior probability $P(H/X)$ is based on more information than the prior probability $P(H)$. The Bayesian theorem provides a way of calculating the posterior probability $P(H/X)$ using probabilities $P(H)$, $P(X)$, and $P(X/H)$. The basic relation is

$$P(H/X) = [P(X/H) \times P(H)]/P(X)$$

Suppose now that there are a set of m samples $S = \{S_1, S_2, \ldots, S_m\}$ (the training data set) where every sample S_i is represented as an *n*-dimensional vector $\{x_1, x_2, \ldots, x_n\}$. Values x_i correspond to attributes $A_1, A_2, \ldots, A_n$, respectively. Also, there are k classes $C_1, C_2, \ldots, C_k$, and every sample belongs to one of these classes. Given an additional data sample X (its class is unknown), it is possible to predict the class for X using the highest conditional probability $P(C_i/X)$, where $i = 1, \ldots, k$. That is the basic idea of naïve Bayesian classifier. These probabilities are computed using the Bayes theorem:

$$P(C_i/X) = [P(X/C_i) \times P(C_i)]/P(X)$$

As $P(X)$ is constant for all classes, only the product $P(X/C_i) \cdot P(C_i)$ needs to be maximized. We compute the prior probabilities of the class as

$$P(C_i) = \text{number of training samples of class } C_i / m$$

where m is total number of training samples.

Because the computation of $P(X/C_i)$ is extremely complex, especially for large data sets, the naïve assumption of conditional independence between attributes is made. Using this assumption, we can express $P(X/C_i)$ as a product:

$$P(X/C_i) = \prod_{t=1}^{n} P(x_t/C_i)$$

where x_t are values for attributes in the sample X. The probabilities $P(x_t/C_i)$ can be estimated from the training data set.

A simple example will show that the naïve Bayesian classification is a computationally simple process even for large training data sets. Given a training data set of seven four-dimensional samples (Table 5.1), it is necessary to predict classification of the new sample $X = \{1, 2, 2, \text{class} = ?\}$. For each sample, A_1, A_2, and A_3 are input dimensions and C is the output classification.

In our example, we need to maximize the product $P(X/C_i) \cdot P(C_i)$ for $i = 1,2$ because there are only two classes. First, we compute prior probabilities $P(C_i)$ of the class:

$$P(C = 1) = 4/7 = 0.5714$$
$$P(C = 2) = 3/7 = 0.4286$$

TABLE 5.1. Training Data Set for a Classification Using Naïve Bayesian Classifier

Sample	Attribute1	Attribute2	Attribute3	Class
	A_1	A_2	A_3	C
1	1	2	1	1
2	0	0	1	1
3	2	1	2	2
4	1	2	1	2
5	0	1	2	1
6	2	2	2	2
7	1	0	1	1

Second, we compute conditional probabilities $P(x_i/C_i)$ for every attribute value given in the new sample $X = \{1, 2, 2, C=?\}$ (or more precisely, $X = \{A_1 = 1, A_2 = 2, A_3 = 2, C = ?\}$) using training data sets:

$$P(A_1 = 1/C = 1) = 2/4 = 0.50$$
$$P(A_1 = 1/C = 2) = 1/3 = 0.33$$
$$P(A_2 = 2/C = 1) = 1/4 = 0.25$$
$$P(A_2 = 2/C = 2) = 2/3 = 0.66$$
$$P(A_3 = 2/C = 1) = 1/4 = 0.25$$
$$P(A_3 = 2/C = 2) = 2/3 = 0.66$$

Under the assumption of conditional independence of attributes, the conditional probabilities $P(X/C_i)$ will be

$$P(X/C = 1) = P(A_1 = 1/C = 1) \times P(A_2 = 2/C = 1) \times P(A_3 = 2/C = 1)$$
$$= 0.50 \cdot 0.25 \cdot 0.25 = 0.03125$$
$$P(X/C = 2) = P(A_1 = 1/C = 2) \times P(A_2 = 2/C = 2) \times P(A_3 = 2/C = 2)$$
$$= 0.33 \cdot 0.66 \cdot 0.66 = 0.14375$$

Finally, multiplying these conditional probabilities with corresponding priori probabilities, we can obtain values proportional ($\approx$) to $P(C_i/X)$ and find their maximum:

$$P(C_1/X) \approx P(X/C = 1) \times P(C = 1) = 0.03125 \times 0.5714 = 0.0179$$
$$P(C_2/X) \approx P(X/C = 2) \times P(C = 2) = 0.14375 \times 0.4286 = 0.0616$$
$$\Downarrow$$
$$P(C_2/X) = \text{Max}\{P(C_1/X), P(C_2/X)\} = \text{Max}\{0.0179, 0.0616\}$$

Based on the previous two values that are the final results of the naive Bayesian classifier, we can predict that the new sample X belongs to the class $C = 2$. The product of probabilities for this class $P(X/C = 2) \cdot P(C = 2)$ is higher, and therefore $P(C = 2/X)$ is higher because it is directly proportional to the computed probability product.

In theory, the Bayesian classifier has the minimum error rate compared with all other classifiers developed in data mining. In practice, however, this is not always the case because of inaccuracies in the assumptions of attributes and class-conditional independence.

5.4   PREDICTIVE REGRESSION

The prediction of continuous values can be modeled by a statistical technique called *regression*. The objective of regression analysis is to determine the best model that can relate the output variable to various input variables. More formally, regression analysis is the process of determining how a variable Y is related to one or more other variables $x_1, x_2, \ldots, x_n$. Y is usually called the response output, or dependent variable, and x_i-s are inputs, regressors, explanatory variables, or independent variables. Common reasons for performing regression analysis include the following:

1. The output is expensive to measure but the inputs are not, and so a cheap prediction of the output is sought.
2. The values of the inputs are known before the output is known, and a working prediction of the output is required.
3. Controlling the input values, we can predict the behavior of corresponding outputs.
4. There might be a causal link between some of the inputs and the output, and we want to identify the links.

Before explaining regression technique in details, let us explain main differences between two concepts: interpolation and regression. In both cases training data set $X = \{x^t, r^t\}_{t=1,N}$ is given where x^t are input features and output value $r^t \in R$:

- If there is *no noise* in the data set, the task is *interpolation*. We would like to find a function $f(x)$ that passes through all these training points such that we have $r^t = f(x^t)$. In polynomial interpolation, given N points, we found that $(N - 1)$ degree polynomial we can use to predict exact output r for any input x.
- In *regression*, *there is noise ε* added to the output of the unknown function f: $r^t = f(x^t) + \varepsilon$. The explanation for noise is that there are extra hidden variables z^t that we cannot observe. We would like to approximate the output $r^t = f(x^t, z^t)$ by our model $g(x^t)$, not only for present training data but for data in future. We are minimizing empirical error: $E(g/x) = 1/N \sum (r^t - g(x^t))^2$ for $t = 1$ to N.

Generalized linear regression models are currently the most frequently applied statistical techniques. They are used to describe the relationship between the trend of one variable and the values taken by several other variables. Modeling this type of relationship is often called linear regression. Fitting models is not the only task in statistical modeling. We often want to select one of several possible models as being the most appropriate. An objective method for choosing between different models is called analysis of variance, and it is described in Section 5.5.

The relationship that fits a set of data is characterized by a prediction model called a *regression equation*. The most widely used form of the regression model is the general linear model formally written as

$$Y = \alpha + \beta_1 \cdot X_1 + \beta_2 \cdot X_2 + \beta_3 \cdot X_3 + \cdots + \beta_n \cdot X_n$$

Applying this equation to each of the given samples, we obtain a new set of equations:

$$y_j = \alpha + \beta_1 \cdot x_{1j} + \beta_2 \cdot x_{2j} + \beta_3 \cdot x_{3j} + \cdots + \beta_n \cdot x_{nj} + \varepsilon_j \quad j = 1, \ldots, m$$

where ε_j's are errors of regression for each of m given samples. The linear model is called linear because the expected value of y_j is a linear function: the weighted sum of input values.

Linear regression with one input variable is the simplest form of regression. It models a random variable Y (called a response variable) as a linear function of another random variable X (called a predictor variable). Given n samples or data points of the form (x_1, y_1), (x_2, y_2),…,(x_n, y_n), where $x_i \in X$ and $y_i \in Y$, linear regression can be expressed as

$$Y = \alpha + \beta \cdot X$$

where α and β are regression coefficients. With the assumption that the variance of Y is a constant, these coefficients can be solved by the method of least squares, which minimizes the error between the actual data points and the estimated line. The residual sum of squares is often called the sum of squares of the errors about the regression line, and it is denoted by SSE:

$$\text{SSE} = \sum_{i=1}^{n} e_i^2 = \sum_{i=1}^{n} (y_i - y_i')^2 = \sum_{i=1}^{n} (y_i - \alpha - \beta x_i)^2$$

where y_i is the real output value given in the data set and y'_i is a response value obtained from the model. Differentiating SSE with respect to α and β, we have

$$\frac{\partial (\text{SEE})}{\partial \alpha} = -2 \sum_{i=1}^{n} (y_i - \alpha - \beta x_i)$$

$$\frac{\partial (\text{SSE})}{\partial \beta} = -2 \sum_{i=1}^{n} ((y_i - \alpha - \beta x_i) \cdot x_i)$$

Setting the partial derivatives equal to zero (minimization of the total error) and rearranging the terms, we obtain the equations

$$n\,\alpha + \beta \sum_{i=1}^{n} x_i = \sum_{i=1}^{n} y_i$$

$$\alpha \sum_{i=1}^{n} x_i + \beta \sum_{i=1}^{n} x_i^2 = \sum_{i=1}^{n} x_i y_i$$

which may be solved simultaneously to yield computing formulas for α and β. Using standard relations for the mean values, regression coefficients for this simple case of optimization are

$$\beta = \frac{\left[\sum_{i=1}^{n}(x_i - \text{mean}_x)\cdot(y_i - \text{mean}_y)\right]}{\left[\sum_{i=1}^{n}(x_i - \text{mean}_x)^2\right]}$$

$$\alpha = \text{mean}_y - \beta \cdot \text{mean}_x$$

where mean_x and mean_y are the mean values for random variables X and Y given in a training data set. It is important to remember that our values of α and β, based on a given data set, are only estimates of the true parameters for the entire population. The equation $y = \alpha + \beta x$ may be used to predict the mean response y_0 for the given input x_0, which is not necessarily from the initial set of samples.

For example, if the sample data set is given in the form of a table (Table 5.2) and we are analyzing the linear regression between two variables (predictor variable A and response variable B), then the linear regression can be expressed as

$$B = \alpha + \beta \cdot A$$

where α and β coefficients can be calculated based on previous formulas (using $\text{mean}_A = 5$ and $\text{mean}_B = 6$), and they have the values

TABLE 5.2. A Database for the Application of Regression Methods

A	B
1	3
8	9
11	11
4	5
3	2

$$\alpha = 1.03$$
$$\beta = 0.92$$

The optimal regression line is

$$B = 1.03 + 0.92 \cdot A$$

The initial data set and the regression line are graphically represented in Figure 5.4 as a set of points and a corresponding line.

Multiple regression is an extension of linear regression and involves more than one predictor variable. The response variable Y is modeled as a linear function of several predictor variables. For example, if the predictor attributes are X_1, X_2, and X_3, then the multiple linear regression is expressed as

$$Y = \alpha + \beta_1 \cdot X_1 + \beta_2 \cdot X_2 + \beta_3 \cdot X_3$$

where $\alpha, \beta_1, \beta_2, \beta_3$ are coefficients that are found by using the method of least squares. For a linear regression model with more than two input variables, it is useful to analyze the process of determining β parameters through a matrix calculation:

$$Y = \beta \cdot X$$

where $\beta = \{\beta_0, \beta_1, \ldots, \beta_n\}$, $\beta_0 = \alpha$, and X and Y are input and output matrices for a given training data set. The residual sum of the squares of errors SSE will also have the matrix representation

$$\text{SSE} = (Y - \beta \cdot X)' \cdot (Y - \beta \cdot X)$$
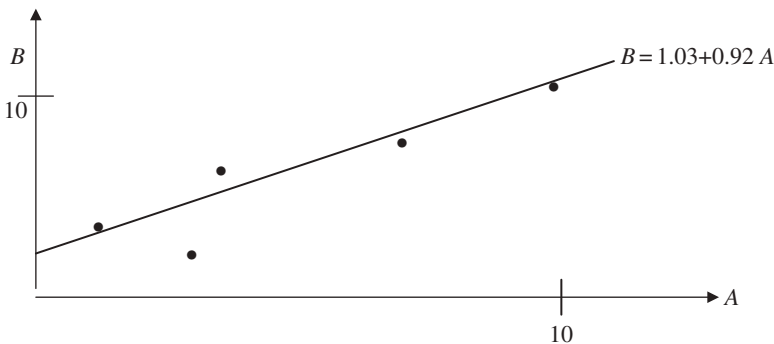
Figure 5.4. Linear regression for the data set given in Table 5.2.

and after optimization

$$\frac{\partial(\text{SSE})}{\partial \beta} = 0 \Rightarrow (X' \cdot X)\beta = X' \cdot Y$$

the final β vector satisfies the matrix equation

$$\beta = (X' \cdot X)^{-1}(X' \cdot Y)$$

where β is the vector of estimated coefficients in a linear regression. Matrices X and Y have the same dimensions as the training data set. Therefore, an optimal solution for β vector is relatively easy to find in problems with several hundred training samples. For real-world data-mining problems, the number of samples may increase to several millions. In these situations, because of the extreme dimensions of matrices and the exponentially increased complexity of the algorithm, it is necessary to find modifications and/or approximations in the algorithm or to use totally different regression methods.

There is a large class of regression problems, initially nonlinear, that can be converted into the form of the general linear model. For example, a polynomial relationship such as

$$Y = \alpha + \beta_1 \cdot X_1 + \beta_2 \cdot X_2 + \beta_3 \cdot X_1 X_3 + \beta_4 \cdot X_2 X_3$$

can be converted to the linear form by setting new variables $X_4 = X_1 \cdot X_3$ and $X_5 = X_2 \cdot X_3$. Also, polynomial regression can be modeled by adding polynomial terms to the basic linear model. For example, a cubic polynomial curve has the form

$$Y = \alpha + \beta_1 \cdot X + \beta_2 \cdot X^2 + \beta_3 \cdot X^3$$

By applying transformation to the predictor variables ($X_1 = X$, $X_2 = X^2$, and $X_3 = X^3$), it is possible to linearize the model and transform it into a multiple-regression problem, which can be solved by the method of least squares. It should be noted that the term linear in the general linear model applies to the dependent variable being a linear function of the unknown parameters. Thus, a general linear model might also include some higher-order terms of independent variables, e.g. terms such as X_1^2, $e^{\beta X}$, $X_1 \cdot X_2$, $1/X$, or X_2^3. The basis is, however, to select the proper transformation of input variables or their combinations. Some useful transformations for linearization of the regression model are given in Table 5.3.

TABLE 5.3. Some Useful Transformations to Linearize Regression

Function	Proper Transformation	Form of Simple Linear Regression
Exponential: $Y = \alpha\, e^{\beta x}$	$Y^* = \ln Y$	Regress Y^* against x
Power: $Y = \alpha\, x^{\beta}$	$Y^* = \log Y$; $x^* = \log x$	Regress Y^* against x^*
Reciprocal: $Y = \alpha + \beta(1/x)$	$x^* = 1/x$	Regress Y against x^*
Hyperbolic: $Y = x/(\alpha + \beta x)$	$Y^* = 1/Y$; $x^* = 1/x$	Regress Y^* against x^*

The major effort, on the part of a user, in applying multiple-regression techniques lies in identifying the *relevant* independent variables from the initial set and in selecting the regression model using only relevant variables. Two general approaches are common for this task:

1. *Sequential search approach*—which consists primarily of building a regression model with an initial set of variables and then selectively adding or deleting variables until some overall criterion is satisfied or optimized.
2. *Combinatorial approach*—which is, in essence, a brute-force approach, where the search is performed across all possible combinations of independent variables to determine the best regression model.

Irrespective of whether the sequential or combinatorial approach is used, the maximum benefit to model building occurs from a proper understanding of the application domain.

Additional postprocessing steps may estimate the quality of the linear regression model. Correlation analysis attempts to measure the strength of a relationship between two variables (in our case this relationship is expressed through the linear regression equation). One parameter, which shows this strength of linear association between two variables by means of a single number, is called a *correlation coefficient r*. Its computation requires some intermediate results in a regression analysis:

$$r = \beta \cdot \sqrt{\left(\frac{S_{xx}}{S_{yy}}\right)} = \sqrt{\frac{S_{xy}}{(S_{xx} \cdot S_{yy})}}$$

where

$$S_{xx} = \sum_{i=1}^{n} (x_i - \text{mean}_x)^2$$

$$S_{yy} = \sum_{i=1}^{n} (y_i - \text{mean}_y)^2$$

$$S_{xy} = \sum_{i=1}^{n} (x_i - \text{mean}_x)(y_i - \text{mean}_y)$$

The value of r is between –1 and 1. Negative values for r correspond to regression lines with negative slopes and a positive r shows a positive slope. We must be very careful in interpreting the r value. For example, values of r equal to 0.3 and 0.6 only mean that we have two positive correlations, the second somewhat stronger than the first. It is wrong to conclude that $r = 0.6$ indicates a linear relationship twice as strong as that indicated by the value $r = 0.3$.

For our simple example of linear regression given at the beginning of this section, the model obtained was $B = 0.8 + 1.04A$. We may estimate the quality of the model using the correlation coefficient r as a measure. Based on the available data in Figure 4.3, we obtained intermediate results

$$S_{AA} = 62$$
$$S_{BB} = 60$$
$$S_{AB} = 52$$

and the final correlation coefficient

$$r = \frac{52}{\sqrt{62 \cdot 60}} = 0.85$$

A correlation coefficient $r = 0.85$ indicates a good linear relationship between two variables. Additional interpretation is possible. Because $r^2 = 0.72$, we can say that approximately 72% of the variations in the values of B is accounted for by a linear relationship with A.

5.5 ANALYSIS OF VARIANCE

Often the problem of analyzing the quality of the estimated regression line and the influence of the independent variables on the final regression is handled through an *analysis-of-variance* (ANOVA) approach. This is a procedure where the total variation in the dependent variable is subdivided into meaningful components that are then observed and treated in a systematic fashion. The ANOVA is a powerful tool that is used in many data-mining applications.

ANOVA is a primarily a method of identifying which of the β''s in a linear regression model are nonzero. Suppose that the β parameters have already been estimated by the least-square error algorithm. Then the residuals are differences between the observed output values and the fitted values:

$$R_i = y_i - f(x_i)$$

The size of the residuals, for all m samples in a data set, is related to the size of variance σ^2, and it can be estimated by

$$S^2 = \frac{\left[\sum_{i=1}^{m} (y_i - f(x_i))^2 \right]}{(m - (n-1))}$$

assuming that the model is not overparameterized. The numerator is called the residual sum, while the denominator is called the residual degree of freedom (d.f.).

The key fact about S^2 is that it allows us to compare different linear models. If the fitted model is adequate, then S^2 is a good estimate of σ^2. If the fitted model includes redundant terms (some β's are really zero), S^2 is still good and close to σ^2. Only if the fitted model does not include one or more of the inputs that it ought to will S^2 tend to be significantly larger than the true value of σ^2. These criteria are basic decision steps in the ANOVA algorithm, in which we analyze the influence of input variables on a final model. First, we start with all inputs and compute S^2 for this model. Then, we omit inputs from the model one by one. If we omit a useful input, the estimate S^2 will significantly increase, but if we omit a redundant input, the estimate should not change much. Note that omitting one of the inputs from the model is equivalent to forcing the corresponding β to the zero. In principle, in each iteration, we compare two S^2 values and analyze the differences between them. For this purpose, we introduce an F-ratio or F-statistic test in the form

$$F = \frac{S^2_{\text{new}}}{S^2_{\text{old}}}$$

If the new model (after removing one or more inputs) is adequate, then F will be close to 1; a value of F significantly larger than one will signal that the model is not adequate. Using this iterative ANOVA approach, we can identify which inputs are related to the output and which are not. The ANOVA procedure is only valid if the models being compared are nested; in other words, one model is special case of the other.

Suppose that the data set has three input variables x_1, x_2, and x_3 and one output Y. In preparation for the use of the linear regression method, it is necessary to estimate the simplest model, in the sense of the number of required inputs. Suppose that after applying the ANOVA methodology, the results given in Table 5.4 are obtained.

The results of ANOVA analysis show that the input attribute x_3 does not have an influence on the output estimation because the F-ratio value is close to 1:

$$F_{21} = \frac{S_2}{S_1} = \frac{3.98}{3.56} = 1.12$$

TABLE 5.4. ANOVA Analysis for a Data Set with Three Inputs x_1, x_2, and x_3

Case	Set of Inputs	S_i^2	F
1	x_1, x_2, x_3	3.56	
2	x_1, x_2	3.98	$F_{21} = 1.12$
3	x_1, x_3	6.22	$F_{31} = 1.75$
4	x_2, x_3	8.34	$F_{41} = 2.34$
5	x_1	9.02	$F_{52} = 2.27$
6	x_2	9.89	$F_{62} = 2.48$

In all other cases, the subsets of inputs increase the F-ratio significantly, and therefore, there is no possibility of reducing the number of input dimensions further without influencing the quality of the model. The final linear regression model for this example will be

$$Y = \alpha + \beta_1 \cdot x_1 + \beta_2 \cdot x_2$$

Multivariate analysis of variance (MANOVA) is a generalization of the previously explained ANOVA analysis, and it concerns data-analysis problems in which the output is a vector rather than a single value. One way to analyze this sort of data would be to model each element of the output separately, but this ignores the possible relationship between different outputs. In other words, the analysis would be based on the assumption that outputs are not related. MANOVA is a form of analysis that *does* allow correlation between outputs. Given the set of input and output variables, we might be able to analyze the available data set using a multivariate linear model:

$$Y_j = \alpha + \beta_1 \cdot x_{1j} + \beta_2 \cdot x_{2j} + \beta_3 \cdot x_{3j} + \cdots + \beta_n \cdot x_{nj} + \varepsilon_j \quad j = 1, 2, \ldots, m$$

where n is the number of input dimensions, m is the number of samples, Y_j is a vector with dimensions $c \times 1$, and c is the number of outputs. This multivariate model can be fitted in exactly the same way as a linear model using least-square estimation. One way to do this fitting would be to fit a linear model to each of c dimensions of the output, one at a time. The corresponding residuals for each dimension will be $(y_j - y'_j)$ where y_j is the exact value for a given dimension and y'_j is the estimated value.

The analogue of the residual sum of squares for the univariate linear model is the matrix of the residual sums of squares for the multivariate linear model. This matrix R is defined as

$$R = \sum_{j=1}^{m} \left(y_j - y'_j \right) \left(y_j - y'_j \right)^{\mathrm{T}}$$

The matrix R has the residual sum of squares for each of the c dimensions stored on its leading diagonal. The off-diagonal elements are the residual sums of cross products for pairs of dimensions. If we wish to compare two nested linear models to determine whether certain β's are equal to zero, then we can construct an extra sum of squares matrix and apply a method similar to ANOVA—MANOVA. While we had an F-statistic in the ANOVA methodology, MANOVA is based on matrix R with four commonly used test statistics: Roy's greatest root, the Lawley–Hotelling trace, the Pillai trace, and Wilks' lambda. Computational details of these tests are not explained in the book, but most textbooks on statistics will explain these; also, most standard statistical packages that support MANOVA analysis support all four statistical tests and explain which one to use under what circumstances.

Classical multivariate analysis also includes the method of principal component analysis, where the set of vector samples is transformed into a new set with a reduced number of dimensions. This method has been explained in Chapter 3

when we were talking about data reduction and data transformation as preprocessing phases for data mining.

5.6   LOGISTIC REGRESSION

Linear regression is used to model continuous-value functions. Generalized regression models represent the theoretical foundation on which the linear regression approach can be applied to model categorical response variables. A common type of a generalized linear model is *logistic regression*. Logistic regression models the probability of some event occurring as a linear function of a set of predictor variables.

Rather than predicting the value of the dependent variable, the logistic regression method tries to estimate the probability p that the dependent variable will have a given value. For example, in place of predicting whether a customer has a good or bad credit rating, the logistic regression approach tries to estimate the probability of a good credit rating. The actual state of the dependent variable is determined by looking at the estimated probability. If the estimated probability is greater than 0.50, then the prediction is closer to *YES* (a good credit rating); otherwise the output is closer to *NO* (a bad credit rating is more probable). Therefore, in logistic regression, the probability p is called the success probability.

We use logistic regression only when the output variable of the model is defined as a categorical binary. On the other hand, there is no special reason why any of the inputs should not also be quantitative, and, therefore, logistic regression supports a more general input data set. Suppose that output Y has two possible categorical values coded as 0 and 1. Based on the available data we can compute the probabilities for both values for the given input sample: $P(y_j = 0) = 1 - p_j$ and $P(y_j = 1) = p_j$. The model that we will fit these probabilities is accommodated linear regression:

$$\log \left(\frac{p_j}{(1 - p_j)} \right) = \alpha + \beta_1 \cdot X_{1j} + \beta_2 \cdot X_{2j} + \beta_3 \cdot X_{3j} + \cdots + \beta_n \cdot X_{nj}$$

This equation is known as the *linear logistic model*. The function $\log(p_j/(1 - p_j))$ is often written as logit(p). The main reason for using the logit form of output is to prevent the predicting probabilities from becoming values out of required range [0, 1]. Suppose that the estimated model, based on a training data set and using the linear regression procedure, is given with a linear equation

$$\text{logit}(p) = 1.5 - 0.6 \cdot x_1 + 0.4 \cdot x_2 - 0.3 \cdot x_3$$

and also suppose that the new sample for classification has input values $\{x_1, x_2, x_3\} = \{1, 0, 1\}$. Using the linear logistic model, it is possible to estimate the probability of the output value 1, $(p(Y = 1))$ for this sample. First, calculate the corresponding logit(p):

$$\text{logit}(p) = 1.5 - 0.6 \cdot 1 + 0.4 \cdot 0 - 0.3 \cdot 1 = 0.6$$

and then the probability of the output value 1 for the given inputs:

$$\log \left(\frac{p}{(1-p)} \right) = 0.6$$

$$p = \frac{e^{0.6}}{(1 + e^{0.6})} = 0.65$$

Based on the final value for probability p, we may conclude that output value $Y = 1$ is more probable than the other categorical value $Y = 0$. Even this simple example shows that logistic regression is a very simple but powerful classification tool in data-mining applications. With one set of data (training set), it is possible to establish the logistic regression model, and with other set of data (testing set), we may analyze the quality of the model in predicting categorical values. The results of logistic regression may be compared with other data-mining methodologies for classification tasks such as decision rules, neural networks, and Bayesian classifier.

5.7 LOG-LINEAR MODELS

Log-linear modeling is a way of analyzing the relationship between categorical (or quantitative) variables. The log-linear model approximates discrete multidimensional probability distributions. It is a type of generalized linear model where the output Y_i is assumed to have a Poisson distribution, with expected value μ_j. The natural logarithm of μ_j is assumed to be the linear function of inputs:

$$\log \left(\mu_j \right) = \alpha + \beta_1 \cdot X_{1j} + \beta_2 \cdot X_{2j} + \beta_3 \cdot X_{3j} + \cdots + \beta_n \cdot X_{nj}$$

Since all the variables of interest are categorical variables, we use a table to represent them, a frequency table that represents the global distribution of data. The aim in log-linear modeling is to identify associations between categorical variables. Association corresponds to the interaction terms in the model, so our problem becomes a problem of finding out which of all β's are 0 in the model. A similar problem can be stated in ANOVA analysis. If there is an interaction between the variables in a log-linear mode, it implies that the variables involved in the interaction are not independent but related, and the corresponding β is not equal to zero. There is no need for one of the categorical variables to be considered as an output in this analysis. If the output is specified, then instead of the log-linear models, we can use logistic regression for analysis. Therefore, we will next explain log-linear analysis when a data set is defined without output variables. All given variables are categorical, and we want to analyze the possible associations between them. That is the task for *correspondence analysis*.

TABLE 5.5. A 2 × 2 Contingency Table for 1100 Samples Surveying Attitudes About Abortion

		Support		Total
		Yes	No	
Sex	Female	309	191	500
	Male	319	281	600
	Total	628	472	1100

Correspondence analysis represents the set of categorical data for analysis within incidence matrices, also called *contingency tables*. The result of an analysis of the contingency table answers the question: "Is there a relationship between analyzed attributes or not?" An example of 2 × 2 contingency table, with cumulative totals, is shown in Table 5.5. The table is a result of a survey to examine the relative attitude of males and females about abortion. The total set of samples is 1100, and every sample consists of two categorical attributes with corresponding values. For the attribute *sex*, the possible values are male and female, and for attribute s*upport*, the values are yes and no. Cumulative results for all the samples are represented in four elements of the contingency table.

Are there any differences in the extent of support for abortion between the male and the female populations? This question may be translated into: "What is the level of dependency (if any) between the two given attributes: sex and support?" If an association exists, then there are statistically significant differences in opinion between the male and the female populations; otherwise both populations have a similar opinion.

Having seen that log-linear modeling is concerned with association of categorical variables, we might attempt to find some quantity (measure) based on this model using data in the contingency table. But we do not do this. Instead, we define the algorithm for feature association based on a comparison of two contingency tables:

1. The first step in the analysis is to transform given contingency table into similar table with *expected* values. These expected values are calculated under assumption that the variables are independent.
2. In the second step, we compare these two matrices using the squared distance measure and the chi-squared test as criteria of association for two categorical variables.

The computational process for these two steps is very simple for a 2 × 2 contingency table. The process is also applicable for increased dimensions of a contingency table (analysis of categorical variables with more than two values, with matrices such as 3 × 4 or 6 × 9).

Let us introduce the notation. Denote the contingency table as $X_{m \times n}$. The row totals for the table are

$$X_{j+} = \sum_{i=1}^{n} X_{ji}$$

and they are valid for every row ($j = 1, \ldots, m$). Similarly, we can define the column totals as

$$X_{+i} = \sum_{j=1}^{m} X_{ji}$$

The grand total is defined as a sum of row totals:

$$X_{++} = \sum_{j=1}^{m} X_{j+}$$

or as a sum of column totals:

$$X_{++} = \sum_{i=1}^{n} X_{+i}$$

Using these totals we can calculate the contingency table of expected values under the assumption that there is no association between the row variable and the column variable. The expected values are

$$E_{ji} = \frac{\left(X_{j+} \cdot X_{+i}\right)}{X_{++}} \quad \text{for } j = 1, \ldots, m, \quad i = 1, \ldots, n$$

and they are computed for every position in the contingency table. The final result of this first step will be a totally new table that consists only of expected values, and the two tables will have the same dimensions.

For our example in Table 5.5, all sums (columns, rows, and grand total) are represented already in the contingency table. Based on these values, we can construct the contingency table of expected values. The expected value on the intersection of the first row and the first column will be

$$E_{11} = \frac{\left(X_{1+} \cdot X_{+1}\right)}{X_{++}} = \frac{500 \cdot 628}{1100} = 285.5$$

Similarly, we can compute the other expected values, and the final contingency table with expected values will be as given in Table 5.6.

TABLE 5.6. 2×2 Contingency Table of Expected Values for the Data Given in Table 5.5

		Support		Total
		Yes	No	
Sex	Female	285.5	214.5	500
	Male	342.5	257.5	600
	Total	628	472	1100

The next step in the analysis of categorical attribute dependency is application of the chi-squared test of association. The initial hypothesis H_0 is the assumption that the two attributes are unrelated, and it is tested by Pearson's chi-squared formula:

$$\chi^2 = \sum_{j=1}^{m} \sum_{i=1}^{n} \left(\frac{(X_{ji} - E_{ji})^2}{E_{ji}} \right)$$

The greater the value of χ^2, the greater the evidence against the hypothesis H_0. For our example, comparing the tables in Tables 5.5 and 5.6, the test gives the following result:

$$\chi^2 = 8.2816$$

with the d.f. for an $m \times n$ dimensional table computed as

$$\text{d.f.(degrees of freedom)} = (m-1) \cdot (n-1) = (2-1)(2-1) = 1$$

In general, the hypothesis H_0 is rejected at the level of significance α if

$$\chi^2 \geq T(\alpha)$$

where $T(\alpha)$ is the threshold value from the χ^2 distribution table usually given in textbooks on statistics. For our example, selecting $\alpha = 0.05$, we obtain the threshold (from the χ^2 tables in most of statistical books)

$$T(0.05) = \chi^2(1 - \alpha, \text{d.f.}) = \chi^2(0.95, 1) = 3.84.$$

A simple comparison shows that

$$\chi^2 = 8.2816 \geq T(0.05) = 3.84$$

and therefore, we can conclude that hypothesis H_0 is rejected; the attributes analyzed in the survey have a high level of dependency. In other words, the attitude about abortion shows differences between the male and the female populations.

The same procedure may be generalized and applied to contingency tables where the categorical attributes have more than two values. The next example shows how the previously explained procedure can be applied without modifications to the

TABLE 5.7. Contingency Tables for Categorical Attributes with Three Values

(a) *A 3× 3 Contingency Table of Observed Values*

		Attribute1			Totals
		Low	Med.	High	
Attribute2	Excell.	21	11	4	36
	Good	3	2	2	7
	Poor	7	1	1	9
Totals		31	14	7	52

(b) *A 3 × 3 Contingency Table of Expected Values under H_0*

		Attribute1			Totals
		Low	Med.	High	
Attribute2	Excell.	21.5	9.7	4.8	36
	Good	4.2	1.9	0.9	7
	Poor	5.4	2.4	1.2	9
Totals		31	14	7	52

contingency table 3×3. The initial table given in Table 5.7a is compared with the table of estimated values that is given in Table 5.7b, and the corresponding test is calculated as $\chi^2 = 3.229$. Note that in this case d.f. parameter has the value

$$\text{d.f.} = (n-1)(m-1) = (3-1) \cdot (3-1) = 4$$

We have to be very careful about drawing additional conclusions and further analyzing the given data set. It is quite obvious that the sample size is not large. The number of observations in many cells of the table is small. This is a serious problem, and additional statistical analysis is necessary to check if the sample is a good representation of the total population or not. We do not cover this analysis here because in most real-world data-mining problems, the data set is enough large to eliminate the possibility of occurrence of these deficiencies.

That was one level of generalization for an analysis of contingency tables with categorical data. The other direction of generalization is inclusion into analysis of more than two categorical attributes. The methods for three- and high-dimensional contingency-table analysis are described in many books on advanced statistics; they explain the procedure of discovered dependencies between several attributes that are analyzed simultaneously.

5.8   LINEAR DISCRIMINANT ANALYSIS

Linear discriminant analysis (LDA) is concerned with classification problems where the dependent variable is categorical (nominal or ordinal) and the independent variables are metric. The objective of LDA is to construct a discriminant function that
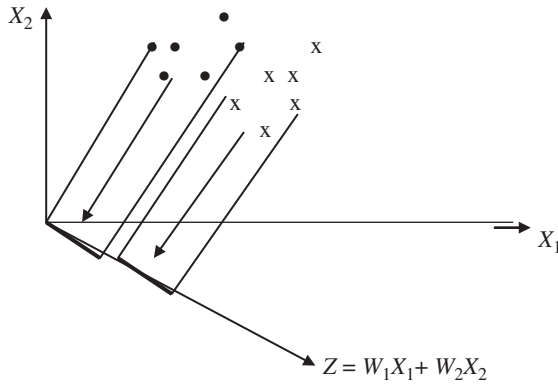
Figure 5.5. Geometric interpretation of the discriminant score.

yields different scores when computed with data from different output classes. A linear discriminant function has the following form:

$$z = w_1 x_1 + w_2 x_2 + \cdots + w_k x_k$$

where $x_1, x_2, \ldots, x_k$ are independent variables. The quantity z is called the discriminant score and $w_1, w_2, \ldots, w_k$ are called weights. A geometric interpretation of the discriminant score is shown in Figure 5.5. As the figure shows, the discriminant score for a data sample represents its projection onto a line defined by the set of weight parameters.

The construction of a discriminant function z involves finding a set of weight values w_i that maximizes the ratio of the *between-class* to the *within-class* variance of the discriminant score for a preclassified set of samples. Once constructed, the discriminant function z is used to predict the class of a new nonclassified sample. Cutting scores serve as the criteria against which each individual discriminant score is judged. The choice of cutting scores depends upon a distribution of samples in classes. Letting z_a and z_b be the mean discriminant scores of preclassified samples from class A and B, respectively, the optimal choice for the cutting score $z_{\text{cut-ab}}$ is given as

$$z_{\text{cut-ab}} = \frac{(z_a + z_b)}{2}$$

when the two classes of samples are of equal size and are distributed with uniform variance. A new sample will be classified to one or another class depending on its score $z > z_{\text{cut-ab}}$ or $z < z_{\text{cut-ab}}$. A weighted average of mean discriminant scores is used as an optimal cutting score when the set of samples for each of the classes is not of equal size:
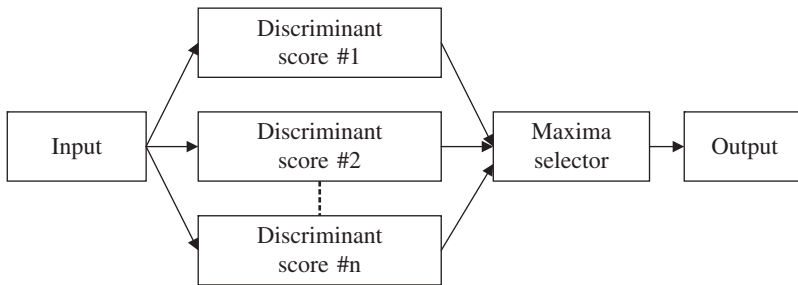
Figure 5.6. Classification process in multiple discriminant analysis.

$$z_{\text{cut-ab}} = \frac{(n_a \cdot z_a + n_b \cdot z_b)}{(n_a + n_b)}$$

The quantities n_a and n_b represent the number of samples in each class. Although a single discriminant function z with several discriminant cuts can separate samples into several classes, *multiple discriminant analysis* is used for more complex problems. The term multiple discriminant analysis is used in situations when separate discriminant functions are constructed for each class. The classification rule in such situations takes the following form: decide in favor of the class whose discriminant score is the highest. This is illustrated in Figure 5.6.

5.9 REVIEW QUESTIONS AND PROBLEMS

1. What are the differences between statistical testing and estimation as basic areas in statistical inference theory?

2. A data set for analysis includes only one attribute X:

   $X = \{7, 12, 5, 18, 5, 9, 13, 12, 19, 7, 12, 12, 13, 3, 4, 5, 13, 8, 7, 6\}$.

   (a) What is the mean of the data set X?
   (b) What is the median?
   (c) What is the mode, and what is the modality of the data set X?
   (d) Find the standard deviation for X.
   (e) Give a graphical summarization of the data set X using a boxplot representation.
   (f) Find outliers in the data set X. Discuss the results.

3. For the training set given in Table 5.1, predict the classification of the following samples using simple Bayesian classifier.

   (a) $\{2, 1, 1\}$
   (b) $\{0, 1, 1\}$

4. Given a data set with two dimensions X and Y:

X	Y
1	5
4	2.75
3	3
5	2.5

(a) Use a linear regression method to calculate the parameters α and β where $y = \alpha + \beta x$.

(b) Estimate the quality of the model obtained in (a) using the correlation coefficient r.

(c) Use an appropriate nonlinear transformation (one of those represented in Table 5.3) to improve regression results. What is the equation for a new, improved, and nonlinear model? Discuss a reduction of the correlation coefficient value.

5. A logit function, obtained through logistic regression, has the form

$$\text{Logit}(p) = 1.2 - 1.3 \times 1 + 0.6 \times 2 + 0.4 \times 3$$

Find the probability of output values 0 and 1 for the following samples.

(a) $\{1, -1, -1\}$

(b) $\{-1, 1, 0\}$

(c) $\{0, 0, 0\}$

6. Analyze the dependency between categorical attributes X and Y if the data set is summarized in a 2×3 contingency table.

		Y	
		T	F
X	A	128	7
	B	66	30
	C	42	55

7. Implement the algorithm for a boxplot representation of each numeric attribute in an input flat file.

8. What are the basic principles in the construction of a discriminant function applied in a linear discriminant analysis?

9. Implement the algorithm to analyze a dependency between categorical variables using two-dimensional contingency tables.

10. Find $EMA(4, 4)$ for the data set $\{27, 27, 18, 9\}$ if: (a) $p = 1/3$, and (b) $p = 3/4$. Discuss the solutions!

11. With Bayes classifier, missing data items are:

    (a) Treated as equal compares.
    (b) Treated as unequal compares.
    (c) Replaced with a default value.
    (d) Ignored.

    Determine what is the true statement.

12. The table below contains counts and ratios for a set of data instances to be used for supervised Bayesian learning. The output attribute is sex with possible values *male* and *female*. Consider an individual who has said *no* to the life insurance promotion, *yes* to the magazine promotion, and *yes* to the watch promotion and has credit card insurance. Use the values in the table together with Bayes classifier to determine the probability that this individual is *male*.

	Magazine Promotion		Watch Promotion		Life Insurance Promotion		Credit Card Insurance	
	Male	Female	Male	Female	Male	Female	Male	Female
Yes	4	3	2	2	2	3	2	1
No	2	1	4	2	4	1	4	3

13. The probability that a person owns a sports car given that they subscribe to at least one automotive magazine is 40%. We also know that 3% of the adult population subscribes to at least one automotive magazine. Finally, the probability of a person owning a sports car given that they do not subscribe to at least one automotive magazine is 30%. Use this information together with Bayes theorem to compute the probability that a person subscribes to at least one automotive magazine given that they own a sports car.

14. Suppose the fraction of undergraduate students who smoke is 15% and the fraction of graduate students who smoke is 23%. If one-fifth of the college students are graduate students and the rest are undergraduates, what is the probability that a student who smokes is a graduate student?

15. Given a 2×2 contingency table for X and Y attributes:

		X	
		x_1	x_2
Y	y_1	7	4
	y_2	2	8

    (a) Find a contingency table with *expected values*.
    (b) If the threshold value for χ^2 test is 8.28, determine whether two attributes X and Y are dependent or not.

16. The logit function, obtained through logistic regression, has the form

$$\text{Logit}(p) = 1.2 - 1.3 \times 1 + 0.6 \times 2 + 0.4 \times 3$$

Find the probability of output values 0 and 1 for the sample $\{1, -1, -1\}$.

17. Given:

- P(Good Movie | Includes Tom Cruise) = 0.01
- P(Good Movie | Tom Cruise absent) = 0.1
- P(Tom Cruise in a randomly chosen movie) = 0.01

What is P(Tom Cruise is in the movie | Not a Good Movie)?

18. You have the following training set with three Boolean input x, y and z and a Boolean output U. Suppose you have to predict U using a naive Bayesian classifier.

x	y	z	U
1	0	0	0
0	1	1	0
0	0	1	0
1	0	0	1
0	0	1	1
0	1	0	1
1	1	0	1

(a) After learning is complete, what would be the predicted probability $P(U = 0|x = 0, y = 1, z = 0)$?
(b) Using the probabilities obtained during the Bayesian classifier training, what would be the predicted probability $P(U = 0|x = 0)$?

19. Three people flip a fair coin. What is the probability that exactly two of them will get heads?

5.10   REFERENCES FOR FURTHER STUDY

1. Berthold, M., D. J. Hand, eds., *Intelligent Data Analysis – An Introduction*, Springer, Berlin, 2007.

The book is a detailed introductory presentation of the key classes of intelligent data-analysis methods including all common data-mining techniques. The first half of the book is devoted to the discussion of classical statistical issues, ranging from basic concepts of probability and inference to advanced multivariate analyses and

Bayesian methods. The second part of the book covers theoretical explanations of data-mining techniques with their roots in disciplines other than statistics. Numerous illustrations and examples will enhance a reader's knowledge about the theory and practical evaluations of data-mining techniques.

2. Cherkassky, V., F. Mulier, *Learning from Data: Concepts, Theory and Methods*, 2nd edition, John Wiley, New York, 2007.

The book provides a unified treatment of the principles and methods for learning dependencies from data. It establishes a general conceptual framework in which various learning methods from statistics, machine learning, and other disciplines can be applied—showing that a few fundamental principles underlie most new methods being proposed today. An additional strength of this primary theoretical book is a large number of case studies and examples that simplifies and makes understandable statistical learning theory concepts.

3. 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.

4. Nisbet R., J. Elder, G. Miner, *Handbook of Statistical Analysis and Data Mining Applications*, Elsevier Inc., Amsterdam, 2009.

The book is a comprehensive professional reference book that guides business analysts, scientists, engineers, and researchers (both academic and industrial) through all stages of data analysis, model building, and implementation. The handbook helps one discern the technical and business problem, understand the strengths and weaknesses of modern data-mining algorithms, and employ the right statistical methods for practical application. Use this book to address massive and complex datasets with novel statistical approaches and be able to objectively evaluate analyses and solutions. It has clear, intuitive explanations of the principles and tools for solving problems using modern analytic techniques, and discusses their application to real problems, in ways accessible and beneficial to practitioners across industries—from science and engineering to medicine, academia, and commerce. This handbook brings together, in a single resource, all the information a beginner will need to understand the tools and issues in data mining to build successful data-mining solutions.

5. Alexander von Eye, Eun-Young Mun, *Log-Linear Modeling: Concepts, Interpretation, and Application*, John Wiley & Sons, Inc., New York, 2013.

The book begins with basic coverage of categorical data and goes on to describe the basics of hierarchical log-linear models as well as decomposing effects in cross-classifications and goodness-of-fit tests. Additional topics include:

- The generalized linear model (GLM) along with popular methods of coding such as effect coding and dummy coding.
- Parameter interpretation and how to ensure that the parameters reflect the hypotheses being studied.
- Symmetry, rater agreement, homogeneity of association, logistic regression, and reduced design models

6

DECISION TREES AND DECISION RULES

Chapter Objectives

- Analyze the characteristics of a logic-based approach to classification problems.
- Describe the differences between decision-tree and decision-rule representations in a final classification model.
- Explain in depth the C4.5 algorithm for generating decision trees and decision rules.
- Identify the required changes in the C4.5 algorithm when missing values exist in training or testing data set.
- Introduce basic characteristics of CART algorithm and Gini index.
- Know when and how to use pruning techniques to reduce the complexity of decision trees and decision rules.
- Summarize the limitations of representing a classification model by decision trees and decision rules.

Decision trees and decision rules are data-mining methodologies applied in many real-world applications as a powerful solution to classification problems. Therefore, at the beginning, let us briefly summarize the basic principles of classification. In general, classification is a process of learning a function that maps a data item into one of several predefined classes. Every classification based on inductive-learning algorithms is given as input a set of samples that consist of vectors of attribute values (also called feature vectors) and a corresponding class. The goal of learning is to create a classification model, known as a *classifier*, which will predict, with the values of its available input attributes, the class for some entity (a given sample). In other words, classification is the process of assigning a discrete label value (class) to an unlabeled record, and a classifier is a model (a result of classification) that predicts one attribute—class of a sample—when the other attributes are given. In doing so, samples are divided into predefined groups. For example, a simple classification might group customer billing records into two specific classes: those who pay their bills within 30 days and those who takes longer than 30 days to pay. Different classification methodologies are applied today in almost every discipline where the task of classification, because of the large amount of data, requires automation of the process. Examples of classification methods used as a part of data-mining applications include classifying trends in financial market and identifying objects in large image databases.

A more formalized approach to classification problems is given through its graphical interpretation. A data set with n features may be thought of as a collection of discrete points (one per example) in an *n*-dimensional space. A classification rule is a hypercube that contains one or more of these points. When there is more than one cube for a given class, all the cubes are OR-ed to provide a complete classification for the class, such as the example of two 2D classes in Figure 6.1. Within a cube the conditions for each part are AND-ed. The size of a cube indicates its generality, i.e., the larger the cube, the more vertices it contains and potentially covers more sample points.

In a classification model, the connection between classes and other properties of the samples can be defined by something as simple as a flowchart or as complex and unstructured as a procedure manual. Data-mining methodologies restrict discussion to
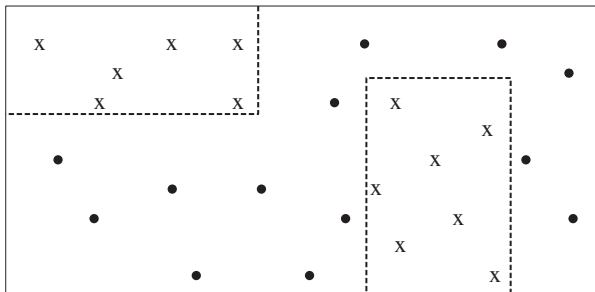

Figure 6.1. Classification of samples in a 2D space.

formalized, "executable" models of classification, and there are two very different ways in which they can be constructed. On the one hand, the model might be obtained by interviewing the relevant expert or experts, and most knowledge-based systems have been built this way despite the well-known difficulties attendant on taking this approach. Alternatively, numerous recorded classifications might be examined, and a model constructed inductively by generalizing from specific examples that are of primary interest for data-mining applications.

The statistical approach to classification explained in Chapter 5 gives one type of model for classification problems: summarizing the statistical characteristics of the set of samples. The other approach is based on logic. Instead of using math operations like addition and multiplication, the logical model is based on expressions that are evaluated as true or false by applying Boolean and comparative operators to the feature values. These methods of modeling give accurate classification results compared with other nonlogical methods, and they have superior explanatory characteristics. Decision trees and decision rules are typical data-mining techniques that belong to a class of methodologies that give the output in the form of logical models.

6.1 DECISION TREES

A particularly efficient method for producing classifiers from data is to generate a decision tree. The decision-tree representation is the most widely used logic method. There is a large number of decision-tree induction algorithms described primarily in the machine-learning and applied-statistics literature. They are supervised learning methods that construct decision trees from a set of input–output samples. It is an efficient nonparametric method for classification and regression. A decision tree is a hierarchical model for supervised learning where the *local region* is identified in a sequence of recursive *splits* through decision nodes with test function. A decision tree is also a nonparametric model in the sense that we *do not assume any parametric form* for the class density.

A typical decision-tree learning system adopts a top-down strategy that searches for a solution in a part of the search space. It guarantees that a simple, but not necessarily the simplest, tree will be found. A decision tree consists of *nodes* where attributes are tested. In a univariate tree, for each internal node, the test uses only one of the attributes for testing. The outgoing *branches* of a node correspond to all the possible outcomes of the test at the node. A simple decision tree for classification of samples with two input attributes X and Y is given in Figure 6.2. All samples with feature values $X > 1$ and $Y = B$ belong to Class2, while the samples with values $X < 1$ belong to Class1, whatever the value for feature Y. The samples, at a nonleaf node in the tree structure, are thus partitioned along the branches, and each child node gets its corresponding subset of samples. Decision trees that use univariate splits have a simple representational form, making it relatively easy for the user to understand the inferred model; at the same time, they represent a restriction on the expressiveness of the model. In general, any restriction on a particular tree representation can significantly
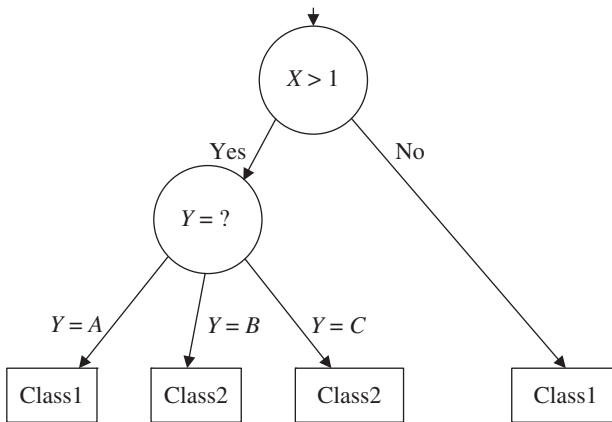
Figure 6.2. A simple decision tree with the tests on attributes *X* and *Y*.

restrict the functional form and thus the approximation power of the model. A well-known tree-growing algorithm for generating decision trees based on univariate splits is Quinlan's *ID3* with an extended version called *C4.5*. Greedy search methods, which involve growing and pruning decision-tree structures, are typically employed in these algorithms to explore the exponential space of possible models.

The ID3 algorithm starts with all the training samples at the root node of the tree. An attribute is selected to partition these samples. For each value of the attribute, a branch is created, and the corresponding subset of samples that have the attribute value specified by the branch is moved to the newly created child node. The algorithm is applied recursively to each child node until all samples at a node are of one class. Every path to the leaf in the decision tree represents a classification rule. Note that the critical decision in such a top-down decision-tree-generation algorithm is the choice of attribute at a node. Attribute selection in ID3 and C4.5 algorithms is based on minimizing an information entropy measure applied to the examples at a node. The approach based on information entropy insists on minimizing the number of tests that will allow a sample to classify in a database. The attribute selection part of ID3 is based on the assumption that the complexity of the decision tree is strongly related to the amount of information conveyed by the value of the given attribute. An information-based heuristic selects the attribute providing the highest information gain, i.e., the attribute that minimizes the information needed in the resulting subtree to classify the sample. An extension of ID3 is the C4.5 algorithm, which extends the domain of classification from categorical attributes to numeric ones. The measure favors attributes that result in partitioning the data into subsets that have low class entropy, i.e., when the majority of examples in it belong to a single class. The algorithm basically chooses the attribute that provides the maximum degree of discrimination between classes locally. More details about basic principles and implementation of these algorithms will be given in the following sections.

To apply some of the methods, which are based on the inductive-learning approach, several key requirements have to be satisfied:

1. *Attribute-value description*—The data to be analyzed must be in a flat-file form—all information about one object or example must be expressible in terms of a fixed collection of properties or attributes. Each attribute may have either discrete or numeric values, but the attributes used to describe samples must not vary from one case to another. This restriction rules out domains in which samples have an inherently variable structure.

2. *Predefined classes*—The categories to which samples are to be assigned must have been established beforehand. In the terminology of machine learning, this is supervised learning.

3. *Discrete classes*—The classes must be sharply delineated: a case either does or does not belong to a particular class. It is expected that there will be far more samples than classes.

4. *Sufficient data*—Inductive generalization given in the form of decision tree proceeds by identifying patterns in data. The approach is valid if enough number of robust patterns can be distinguished from chance coincidences. As this differentiation usually depends on statistical tests, there must be sufficient number of samples to allow these tests to be effective. The amount of data required is affected by factors such as the number of properties and classes and the complexity of the classification model. As these factors increase, more data will be needed to construct a reliable model.

5. *"Logical" classification models*—These methods construct only such classifiers that can be expressed as decision trees or decision rules. These forms essentially restrict the description of a class to a logical expression whose primitives are statements about the values of particular attributes. Some applications require weighted attributes or their arithmetic combinations for a reliable description of classes. In these situations logical models become very complex, and, in general, they are not effective.

6.2 *C4.5 ALGORITHM*: GENERATING A DECISION TREE

The most important part of the C4.5 algorithm is the process of generating an initial decision tree from the set of training samples. As a result, the algorithm generates a classifier in the form of a decision tree; a structure with two types of nodes: *a leaf*, indicating a class, or *a decision node* that specifies some test to be carried out on a single-attribute value, with one branch and subtree for each possible outcome of the test.

A decision tree can be used to classify a new sample by starting at the root of the tree and moving through it until a leaf is encountered. At each nonleaf decision node, the features' outcome for the test at the node is determined and
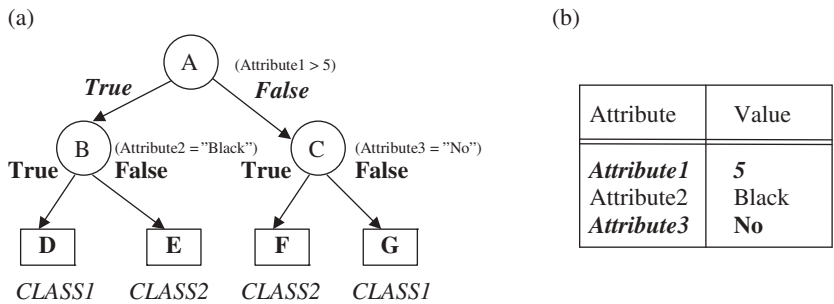
(a)

(b)

Attribute	Value
Attribute1	**5**
Attribute2	Black
Attribute3	**No**

Figure 6.3. Classification of a new sample based on the decision-tree model. (a) Decision tree. (b) An example for classification.

attention shifts to the root of the selected subtree. For example, if the classification model of the problem is given with the decision tree in Figure 6.3a, and the sample for classification in Figure 6.3b, then the algorithm will create the path through the nodes **A**, **C**, and **F** (leaf node) until it makes the final classification decision: *CLASS2*.

The skeleton of the C4.5 algorithm is based on Hunt's *CLS* method for constructing a decision tree from a set T of training samples. Let the classes be denoted as $\{C_1, C_2,\ldots,C_k\}$. There are three possibilities for the content of the set T:

1. T contains one or more samples, all belonging to a single class C_j. The decision tree for T is a leaf-identifying class C_j.
2. T contains no samples. The decision tree is again a leaf, but the class to be associated with the leaf must be determined from information other than T, such as the overall majority class in T. The C4.5 algorithm uses as a criterion the most frequent class at the parent of the given node.
3. T contains samples that belong to a mixture of classes. In this situation, the idea is to refine T into subsets of samples that are heading toward a single-class collection of samples. Based on single attribute, an appropriate test that has one or more mutually exclusive outcomes $\{O_1, O_2,\ldots,O_n\}$ is chosen. T is partitioned into subsets $T_1, T_2,\ldots,T_n$ where T_i contains all the samples in T that have outcome O_i of the chosen test. The decision tree for T consists of a decision node identifying the test and one branch for each possible outcome (examples of this type of nodes are nodes A, B, and C in the decision tree in Figure 6.3a).

The same tree-building procedure is applied recursively to each subset of training samples, so that the ith branch leads to the decision tree constructed from the subset T_i of training samples. The successive division of the set of training samples proceeds until all the subsets consist of samples belonging to a single class.

The tree-building process is not uniquely defined. For different tests, even for a different order of their application, different trees will be generated. Ideally, we would like to choose a test at each stage of sample-set splitting so that the final tree is small. Since we are looking for a compact decision tree that is consistent with the training set, why not explore all possible trees and select the simplest? Unfortunately, the problem of finding the smallest decision tree consistent with a training data set is NP-complete. Enumeration and analysis of all possible trees will cause a combinatorial explosion for any real-world problem. For example, for a small database with five attributes and only twenty training examples, the possible number of decision trees is greater than 10^6, depending on the number of different values for every attribute. Therefore, most decision-tree construction methods are non-backtracking, greedy algorithms. Once a test has been selected using some heuristics to maximize the measure of progress and the current set of training cases has been partitioned, the consequences of alternative choices are not explored. The measure of progress is a local measure, and the gain criterion for a test selection is based on the information available for a given step of data splitting.

Suppose we have the task of selecting a possible test with n outcomes (*n* values for a given feature) that partitions the set *T* of training samples into subsets $T_1, T_2,\ldots, T_n$. The only information available for guidance is the distribution of classes in *T* and its subsets T_i. If *S* is any set of samples, let freq(C_i, *S*) stand for the number of samples in *S* that belong to class C_i (out of k possible classes), and let $|S|$ denote the number of samples in the set *S*.

The original ID3 algorithm used a criterion called *gain* to select the attribute to be tested that is based on the information theory concept: *entropy*. The following relation gives the computation of the entropy of the set *T* (bits are units):

$$\text{Info}(T) = -\sum_{i=1}^{k}\left(\left(\frac{\text{freq}(C_i,T)}{|T|}\right)\cdot\log_2\left(\frac{\text{freq}(C_i,T)}{|T|}\right)\right)$$

Now consider a similar measurement after *T* has been partitioned in accordance with *n* outcomes of one attribute test *X*. The expected information requirement can be found as the weighted sum of entropies over the subsets:

$$\text{Info}_x(T) = \sum_{i=1}^{n}\left(\left(\frac{|T_i|}{|T|}\right)\cdot\text{Info}(T_i)\right)$$

The quantity

$$\text{Gain}(X) = \text{Info}(T) - \text{Info}_x(T)$$

measures the information that is gained by partitioning *T* in accordance with the test *X*. The gain criterion selects a test *X* to maximize Gain(*X*), i.e., this criterion will select an attribute with the highest info-gain.

TABLE 6.1. A Simple Flat Database of Examples for Training

Database T:			
Attribute1	Attribute2	Attribute3	Class
A	70	True	CLASS1
A	90	True	CLASS2
A	85	False	CLASS2
A	95	False	CLASS2
A	70	False	CLASS1
B	90	True	CLASS1
B	78	False	CLASS1
B	65	True	CLASS1
B	75	False	CLASS1
C	80	True	CLASS2
C	70	True	CLASS2
C	80	False	CLASS1
C	80	False	CLASS1
C	96	False	CLASS1

Let us analyze the application of these measures and the creation of a decision tree for one simple example. Suppose that the database T is given in a flat form in which each out of fourteen examples (cases) is described by three input attributes and belongs to one of two given classes: CLASS1 or CLASS2. The database is given in tabular form in Table 6.1.

Nine samples belong to CLASS1 and five samples to CLASS2, so the entropy before splitting is

$$\text{Info}(T) = -\frac{9}{14}\log_2\left(\frac{9}{14}\right) - \frac{5}{14}\log_2\left(\frac{5}{14}\right) = 0.940 \text{ bits}$$

After using Attribute1 to divide the initial set of samples T into three subsets (test x_1 represents the selection one of three values A, B, or C), the resulting information is given by

$$\text{Info}_{x1}(T) = \frac{5}{14}\left(-\frac{2}{5}\log_2\left(\frac{2}{5}\right) - \frac{3}{5}\log_2\left(\frac{3}{5}\right)\right)$$

$$+ \frac{4}{14}\left(-\frac{4}{4}\log_2\left(\frac{4}{4}\right) - \frac{0}{4}\log_2\left(\frac{0}{4}\right)\right)$$

$$+ \frac{5}{14}\left(-\frac{3}{5}\log_2\left(\frac{3}{5}\right) - \frac{2}{5}\log_2\left(\frac{5}{5}\right)\right)$$

$$= 0.694 \text{ bits}$$

The information gained by this test x_1 is

$$\text{Gain}(x_1) = 0.940 - 0.694 = 0.246 \text{ bits}$$

If the test and splitting is based on Attribute3 (test x_2 represents the selection one of two values True or False), a similar computation will give new results:

$$\text{Info}_{x2}(T) = \frac{6}{14}\left(-\frac{3}{6}\log_2\left(\frac{3}{6}\right) - \frac{3}{6}\log_2\left(\frac{3}{6}\right)\right)$$

$$+ \frac{8}{14}\left(-\frac{6}{8}\log_2\left(\frac{6}{8}\right) - \frac{2}{8}\log_2\left(\frac{2}{8}\right)\right)$$

$$= 0.892 \text{ bits}$$

and corresponding gain is

$$\text{Gain}(x_2) = 0.940 - 0.892 = 0.048 \text{ bits}$$

Based on the gain criterion, the decision-tree algorithm will select test x_1 as an initial test for splitting the database T because this gain is higher. To find the optimal test, it will be necessary to analyze a test on Attribute2, which is a numeric feature with continuous values. In general, C4.5 contains mechanisms for proposing three types of tests:

1. The "standard" test on a discrete attribute, with one outcome and one branch for each possible value of that attribute (in our example these are both tests x_1 for Attribute1 and x_2 for Attribute3).
2. If attribute Y has continuous numeric values, a binary test with outcomes $Y \le Z$ and $Y > Z$ could be defined, by comparing its value against a threshold value Z.
3. A more complex test also based also on a discrete attribute, in which the possible values are allocated to a variable number of groups with one outcome and branch for each group.

While we have already explained standard test for categorical attributes, additional explanations are necessary about a procedure for establishing tests on attributes with numeric values. It might seem that tests on continuous attributes would be difficult to formulate, since they contain an arbitrary threshold for splitting all values into two intervals. But there is an algorithm for the computation of optimal threshold value Z. The training samples are first sorted on the values of the attribute Y being considered. There are only a finite number of these values, so let us denote them in sorted order as $\{v_1, v_2, \ldots, v_m\}$. Any threshold value lying between v_i and v_{i+1} will have the same effect as dividing the cases into those whose value of the attribute Y lies in $\{v_1, v_2, \ldots, v_i\}$ and those whose value is in $\{v_{i+1}, v_{i+2}, \ldots, v_m\}$. There are thus only $m - 1$ possible splits on Y, all of which should be examined systematically to obtain an optimal split. It is usual to choose the midpoint of each interval, $(v_i + v_{i+1})/2$, as the

representative threshold. The algorithm C4.5 differs in choosing as the threshold a smaller value v_i for every interval $\{v_i, v_{i+1}\}$, rather than the midpoint itself. This ensures that the threshold values appearing in either the final decision tree or rules or both actually occur in the database.

To illustrate this threshold-finding process, we could analyze, for our example of database T, the possibilities of Attribute2 splitting. After a sorting process, the set of values for Attribute2 is {65, 70, 75, 78, 80, 85, 90, 95, 96}, and the set of potential threshold values Z is {65, 70, 75, 78, 80, 85, 90, 95}. Out of these eight values, the optimal Z (with the highest information gain) should be selected. For our example, the optimal Z value is Z = 80, and the corresponding process of information-gain computation for the test x_3 (Attribute 2 ≤ 80 or Attribute 2 > 80) is the following:

$$\text{Info}_{x3}(T) = \frac{9}{14}\left(-\frac{7}{9}\log_2\left(\frac{7}{9}\right) - \frac{2}{9}\log_2\left(\frac{2}{9}\right)\right)$$

$$+ \frac{5}{14}\left(-\frac{2}{5}\log_2\left(\frac{2}{5}\right) - \frac{3}{5}\log_2\left(\frac{3}{5}\right)\right)$$

$$= 0.837\,\text{bits}$$

$$\text{Gain}(x_3) = 0.940 - 0.837 = 0.103\,\text{bits}$$

Now, if we compare the information gain for the three attributes in our example, we can see that Attribute1 still gives the highest gain of 0.246 bits, and therefore this attribute will be selected for the first splitting in the construction of a decision tree. The root node will have the test for the values of Attribute1, and three branches will be created, one for each of the attribute values. This initial tree with the corresponding subsets of samples in the children nodes is represented in Figure 6.4.
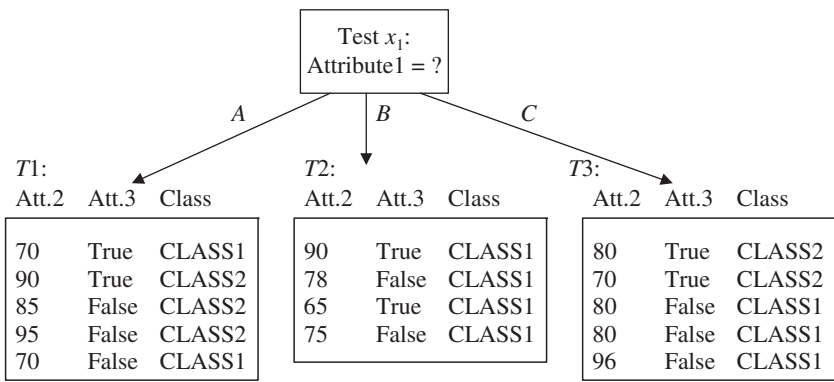

Figure 6.4. Initial decision tree and subset cases for a database in Table 6.1.

After initial splitting, every child node has several samples from the database, and the entire process of test selection and optimization will be repeated for every child node. Because the child node for test x_1: Attribute1 = B has four cases and all of them are in CLASS1, this node will be the leaf node, and no additional tests are necessary for this branch of the tree.

For the remaining child node where we have five cases in subset T_1, tests on the remaining attributes can be performed; an optimal test (with maximum information gain) will be test x_4 with two alternatives: Attribute 2 ≤ 70 or Attribute 2 > 70:

$$\text{Info}(T_1) = -\frac{2}{5}\log_2\left(\frac{2}{5}\right) - \frac{3}{5}\log_2\left(\frac{3}{5}\right) = 0.97\,\text{bits}$$

Using Attribute 2 to divide T_1 into two subsets (test x_4 represents the selection of one of two intervals), the resulting information is given by

$$\text{Info}_{x4}(T_1) = \frac{2}{5}\left(-\frac{2}{2}\log_2\left(\frac{2}{2}\right) - \frac{0}{2}\log_2\left(\frac{0}{2}\right)\right)$$

$$+ \frac{3}{5}\left(-\frac{0}{3}\log_2\left(\frac{0}{3}\right) - \frac{3}{3}\log_2\left(\frac{3}{3}\right)\right)$$

$$= 0\,\text{bits}$$

The information gained by this test is maximal:

$$\text{Gain}(x_4) = 0.97 - 0 = 0.97\,\text{bits}$$

and two branches will create the final leaf nodes because the subsets of cases in each of the branches belong to the same class.

A similar computation will be carried out for the third child of the root node. For the subset T_3 of the database T, the selected optimal test x_5 is the test on Attribute 3 values. Branches of the tree, Attribute 3 = True and Attribute 3 = False, will create uniform subsets of cases, which belong to the same class. The final decision tree for database T is represented in Figure 6.5.

Alternatively, a decision tree can be presented in the form of an executable code (or pseudocode) with *if-then* constructions for branching into a tree structure. The transformation of a decision tree from one representation to the other is very simple and straightforward. The final decision tree for our example is given in pseudocode in Figure 6.6.

While the gain criterion has had some good results in the construction of compact decision trees, it also has one serious deficiency: a strong bias in favor of tests with many outcomes. A solution was found in some kinds of normalization. By analogy with the definition of Info(S), an additional parameter was specified:
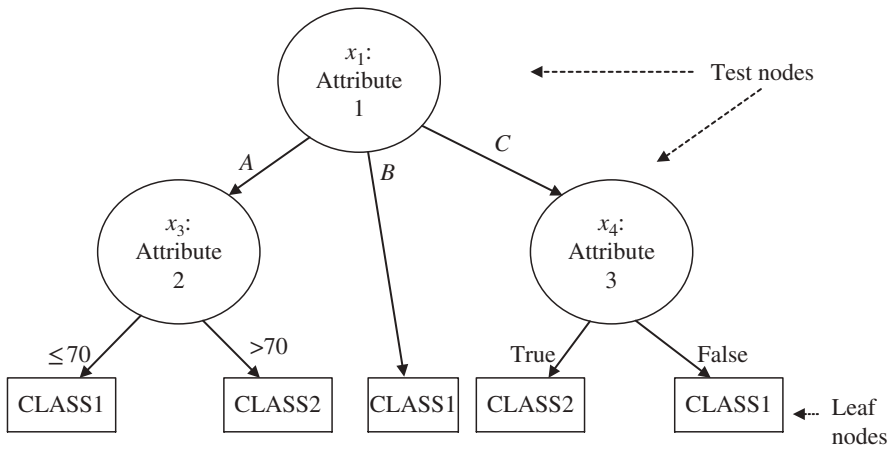
Figure 6.5. A final decision tree for database *T* given in Table 6.1.

```
If        Attribute1 = A
          Then
                    If        Attribute2 <= 70
                              Then
                                        Classification = CLASS1;
                              Else
                                        Classification = CLASS2;
Elseif    Attribute1 = B
          Then
                                        Classification = CLASS1;
Elseif    Attribute1 = C
          Then
                    If        Attribute3 = True
                              Then
                                        Classification = CLASS2;
                              Else
                                        Classification = CLASS1.
```
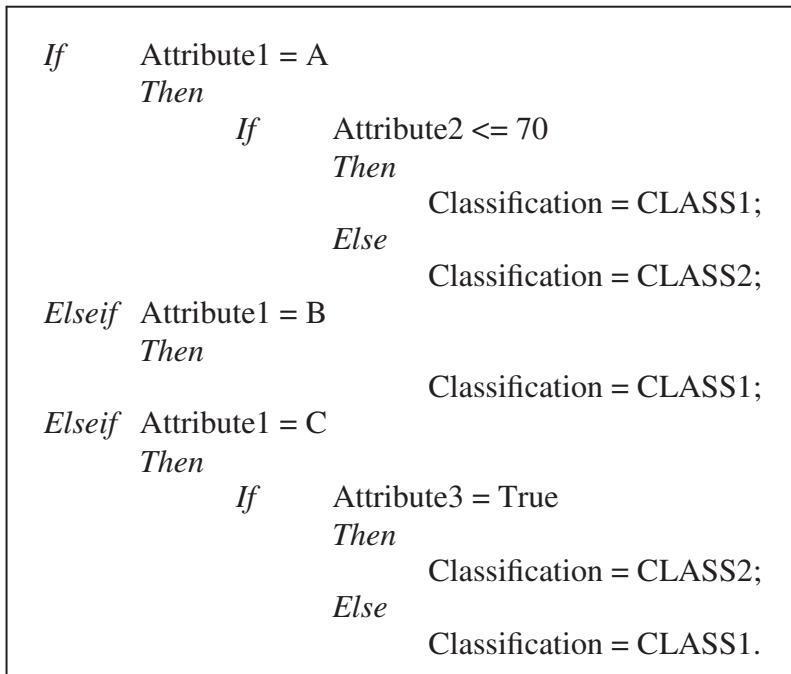
Figure 6.6. A decision tree in the form of pseudocode for the database *T* given in Table 6.1.

$$\text{Split-info}(X) = -\sum_{i=1}^{n}\left(\left(\frac{|T_i|}{|T|}\right)\log_2\left(\frac{|T_i|}{|T|}\right)\right)$$

This represented the potential information generated by dividing set T into n subsets T_i. Now, a new gain measure could be defined:

$$\text{Gain-ratio}(X) = \frac{\text{gain}(X)}{\text{Split-info}(X)}$$

This new gain measure expresses the proportion of information generated by the split that is useful, i.e. that appears helpful in classification. The gain-ratio criterion also selects a test that maximizes the ratio given earlier. This criterion is robust and typically gives a consistently better choice of a test than the previous gain criterion. A computation of the gain-ratio test can be illustrated for our example. To find the gain-ratio measure for the test x_1, an additional parameter Split-info(x_1) is calculated:

$$\text{Split-info}(x_1) = -\frac{5}{14}\log_2\left(\frac{5}{14}\right) - \frac{4}{14}\log_2\left(\frac{4}{14}\right) - \frac{5}{14}\log_2\left(\frac{5}{14}\right) = 1.577 \text{ bits}$$

$$\text{Gain-ratio}(x_1) = \frac{0.246}{1.557} = 0.156$$

A similar procedure should be performed for other tests in the decision tree. Instead of gain measure, the maximal gain ratio will be the criterion for attribute selection, along with a test to split samples into subsets. The final decision tree created using this new criterion for splitting a set of samples will be the most compact.

6.3   UNKNOWN ATTRIBUTE VALUES

The previous version of the C4.5 algorithm is based on the assumption that all values for all attributes are determined. But in a data set, often some attribute values for some samples can be missing—such incompleteness is typical in real-world applications. This might occur because the value is not relevant to a particular sample, or it was not recorded when the data was collected, or an error was made by the person the entering data into a database. To solve the problem of missing values, there are two choices:

1. Discard all samples in a database with missing data.
2. Define a new algorithm or modify an existing algorithm that will work with missing data.

The first solution is simple but unacceptable when large amounts of missing values exist in a set of samples. To address the second alternative, several questions must be answered:

1. How does one compare two samples with different numbers of unknown values?
2. Training samples with unknown values cannot be associated with a particular value of the test, and so they cannot be assigned to any subsets of cases. How should these samples be treated in the partitioning?
3. In a testing phase of classification, how does one treat a missing value if the test is on the attribute with the missing value?

All these and many other questions arise with any attempt to find a solution for missing data. Several classification algorithms that work with missing data are usually based on filling in a missing value with the most probable value or on looking at the probability distribution of all values for the given attribute. None of these approaches is uniformly superior.

In C4.5, it is an accepted principle that samples with unknown values are distributed probabilistically according to the relative frequency of known values. Let $Info(T)$ and $Info_x(T)$ be calculated as before, except that only samples with known values of attributes are taken into account. Then the gain parameter can reasonably be corrected with a factor F, which represents the probability that a given attribute is known (F = number of samples in the database with a known value for a given attribute/total number of samples in a data set). The new gain criterion will have the form

$$Gain(x) = F(Info(T) - Info_x(T))$$

Similarly, Split-info(x) can be altered by regarding the samples with unknown values as an additional group in splitting. If the test x has n outcomes, its Split-info(x) is computed as if the test divided the data set into $n + 1$ subsets. This modification has a direct influence on the final value of the modified criterion Gain-ratio(x).

Let us explain the modifications of the C4.5 decision-tree methodology applied on one example. The database is similar to previous one (Table 6.1), only there is now one value missing for Attribute1 denoted by "?" as presented in Table 6.2.

The computation of the gain parameter for Attribute1 is similar as before; only the missing value corrects some of the previous steps. Eight out of the thirteen cases with values for Attribute1 belong to CLASS1, and five cases to CLASS2, so the entropy before splitting is

$$Info(T) = -\frac{8}{13}\log_2\left(\frac{8}{13}\right) - \frac{5}{13}\log_2\left(\frac{5}{13}\right) = 0.961 \text{ bits}$$

TABLE 6.2. A Simple Flat Database of Examples with One Missing Value

Database T:			
Attribute1	Attribute2	Attribute3	Class
A	70	True	CLASS1
A	90	True	CLASS2
A	85	False	CLASS2
A	95	False	CLASS2
A	70	False	CLASS1
?	90	True	CLASS1
B	78	False	CLASS1
B	65	True	CLASS1
B	75	False	CLASS1
C	80	True	CLASS2
C	70	True	CLASS2
C	80	False	CLASS1
C	80	False	CLASS1
C	96	False	CLASS1

After using Attribute1 to divide T into three subsets (test x_1 represents the selection one of three values A, B, or C), the resulting information is given by

$$\text{Info}_{x1}(T) = \frac{5}{13}\left(-\frac{2}{5}\log_2\left(\frac{2}{5}\right) - \frac{3}{5}\log_2\left(\frac{3}{5}\right)\right)$$

$$+ \frac{3}{13}\left(-\frac{3}{3}\log_2\left(\frac{3}{3}\right) - \frac{0}{3}\log_2\left(\frac{0}{3}\right)\right)$$

$$+ \frac{5}{13}\left(-\frac{3}{5}\log_2\left(\frac{3}{5}\right) - \frac{2}{5}\log_2\left(\frac{5}{5}\right)\right)$$

$$= 0.747\,\text{bits}$$

The information gained by this test is now corrected with the factor F ($F = 13/14$ for our example):

$$\text{Gain}(x_1) = \frac{13}{14}(0.961 - 0.747) = 0.199\,\text{bits}$$

The gain for this test is slightly lower than the previous value of 0.216 bits. The split information, however, is still determined from the entire training set and is larger, since there is an extra category for unknown values:

$$\text{Split-info}(x_1) = - \left(\frac{5}{14} \log \left(\frac{5}{14} \right) + \frac{3}{14} \log \left(\frac{3}{14} \right) + \frac{5}{14} \log \left(\frac{5}{14} \right) + \frac{1}{14} \log \left(\frac{1}{14} \right) \right)$$

$$= 1.8$$

Additionally, the concept of partitioning must be generalized. With every sample a new parameter, probability, is associated. When a case with known value is assigned from T to subset T_i, the probability of it belonging to T_i is 1, and in all other subsets is 0. When a value is not known, only a weaker probabilistic statement can be made. C4.5 therefore associates with each sample (having a missing value) in each subset T_i and weight w, representing the probability that the case belongs to each subset. To make the solution more general, it is necessary to take into account that the probabilities of samples before splitting are not always equal to one (in subsequent iterations of the decision-tree construction). Therefore, new parameter w_{new} for missing values after splitting is equal to the old parameter w_{old} before splitting multiplied by the probability that the sample belongs to each subset $P(T_i)$, or more formally

$$w_{\text{new}} = w_{\text{old}} \cdot P(T_i)$$

For example, the record with the missing value, given in the database in Table 6.2, will be represented in all three subsets after the splitting set T into subsets T_i using test x_1 on Attribute1. New weights w_i will be equal to probabilities 5/13, 3/13, and 5/13, because the initial (old) value for w is equal to one. The new subsets are given in Figure 6.7. $|T_i|$ can be now reinterpreted in C4.5 not as a number of elements in a set T_i, but as a sum of all weights w for the given set T_i. From Figure 6.7, the new values are computed as $|T_1| = 5 + 5/13$, $|T_2| = 3 + 3/13$, and $|T_3| = 5 + 5/13$.

If these subsets are partitioned further by the tests on Attribute2 and Attribute3, the final decision tree for a data set with missing values has the form shown in Figure 6.8.

The decision tree in Figure 6.8 has much the same structure as before (Fig. 6.6), but because of the ambiguity in final classification, every decision is attached with two

T_1: (Attribute1 = A)

Att.2	Att.3	Class	w
70	True	CLASS1	1
90	True	CLASS2	1
85	False	CLASS2	1
95	False	CLASS2	1
70	False	CLASS1	1
90	*True*	*CLASS1*	*5/13*

T_2: (Attribute1 = B)

Att.2	Att.3	Class	w
90	*True*	*CLASS1*	*3/13*
78	False	CLASS1	1
65	True	CLASS1	1
75	False	CLASS1	1

T_3: (Attribute1 = C)

Att.2	Att.3	Class	w
80	True	CLASS2	1
70	True	CLASS2	1
80	False	CLASS1	1
80	False	CLASS1	1
96	False	CLASS1	1
90	*True*	*CLASS1*	*5/13*

Figure 6.7. Results of test x_1 are subsets T_i (initial set T is with missing value).

```
If      Attribute1 = A
              Then
                          If       Attribute2 <= 70
                                   Then
                                               Classification = CLASS1      (2.0 / 0);
                                   Else
                                               Classification = CLASS2      (3.4 / 0.4);
              Elseif  Attribute1 = B
                   Then
                                               Classification = CLASS1      (3.2 / 0);
              Elseif  Attribute1 = C
                   Then
                          If       Attribute3 = True
                                   Then
                                               Classification = CLASS2      (2.4 / 0.4);
                                   Else
                                               Classification = CLASS1      (3.0 / 0).
```
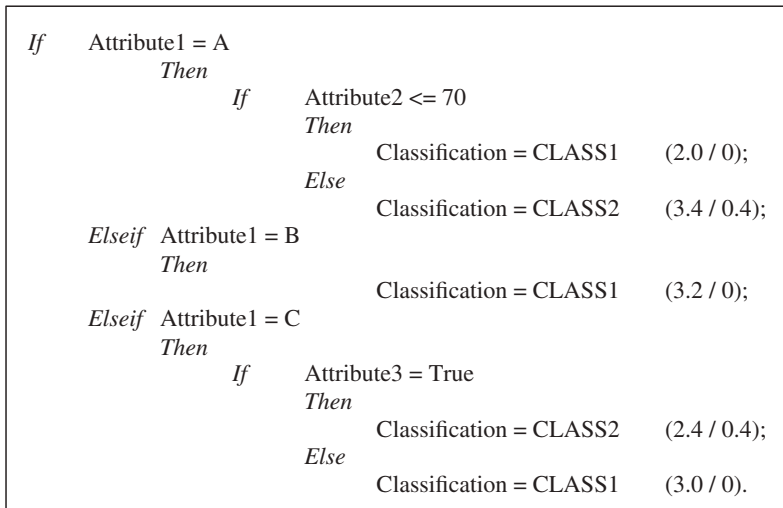
Figure 6.8. Decision tree for the database *T* with missing values.

parameters in a form ($|T_i|/E$). $|T_i|$ is the sum of the fractional samples that reach the leaf, and E is the number of samples that belong to classes other than the nominated class.

For example, (3.4/0.4) means that 3.4 (or $3 + 5/13$) fractional training samples reached the leaf, of which 0.4 (or 5/13) did not belong to the class assigned to the leaf. It is possible to express the $|T_i|$ and E parameters in percentages:

$3/3.4 \cdot 100\% = 88\%$ of cases at a given leaf would be classified as CLASS2.

$0.4/3.4 \cdot 100\% = 12\%$ of cases at a given leaf would be classified as CLASS1.

A similar approach is taken in C4.5 when the decision tree is used to classify a sample previously not present in a database; that is the *testing phase*. If all attribute values are known, then the process is straightforward. Starting with a root node in a decision tree, tests on attribute values will determine traversal through the tree, and at the end, the algorithm will finish in one of leaf nodes that uniquely defines the class of a testing example (or with probabilities, if the training set had missing values). If the value for a relevant testing attribute is unknown, the outcome of the test cannot be determined. Then the system explores all possible outcomes from the test and combines the resulting classification arithmetically. Since there can be multiple paths from the root of a tree or subtree to the leaves, a classification is a class distribution rather than a single class. When the total class distribution for the tested case has been established, the class with the highest probability is assigned as the predicted class.

6.4   PRUNING DECISION TREES

Discarding one or more subtrees and replacing them with leaves simplify a decision tree, and that is the main task in decision-tree pruning. In replacing the subtree with a leaf, the algorithm expects to lower the *predicted error rate* and increase the quality of a classification model. But computation of error rate is not simple. An error rate based only on a training data set does not provide a suitable estimate. One possibility to estimate the predicted error rate is to use a new, additional set of test samples if they are available or to use the cross-validation techniques explained in Chapter 4. This technique divides initially available samples into equal-sized blocks, and, for each block, the tree is constructed from all samples except this block and tested with a given block of samples. With the available training and testing samples, the basic idea of decision-tree pruning is to remove parts of the tree (subtrees) that do not contribute to the classification accuracy of unseen testing samples, producing a less complex and thus more comprehensible tree. There are two ways in which the recursive-partitioning method can be modified:

1. Deciding not to divide a set of samples any further under some conditions. The stopping criterion is usually based on some statistical tests, such as the χ^2 test: *If there are no significant differences in classification accuracy before and after division, then represent a current node as a leaf.* The decision is made in advance, before splitting, and therefore this approach is called *prepruning*.
2. Removing retrospectively some of the tree structure using selected accuracy criteria. The decision in this process of *postpruning* is made after the tree has been built.

C4.5 follows the *postpruning* approach, but it uses a specific technique to estimate the predicted error rate. This method is called *pessimistic pruning*. For every node in a tree, the estimation of the upper confidence limit U_{cf} is computed using the statistical tables for binomial distribution (given in most textbooks on statistics). Parameter U_{cf} is a function of $|T_i|$ and E for a given node. C4.5 uses the default confidence level of 25% and compares $U_{25\%}$ ($|T_i|/E$) for a given node T_i with a weighted confidence of its leaves. Weights are the total number of cases for every leaf. If the predicted error for a root node in a subtree is less than weighted sum of $U_{25\%}$ for the leaves (predicted error for the subtree), then a subtree will be replaced with its root node, which becomes a new leaf in a pruned tree.

Let us illustrate this procedure with one simple example. A subtree of a decision tree is given in Figure 6.9, where the root node is the test x_1 on three possible values $\{1, 2, 3\}$ of the attribute A. The children of the root node are leaves denoted with corresponding classes and ($|T_i|/E$) parameters. The question is to estimate the possibility of pruning the subtree and replacing it with its root node as a new, generalized leaf node.
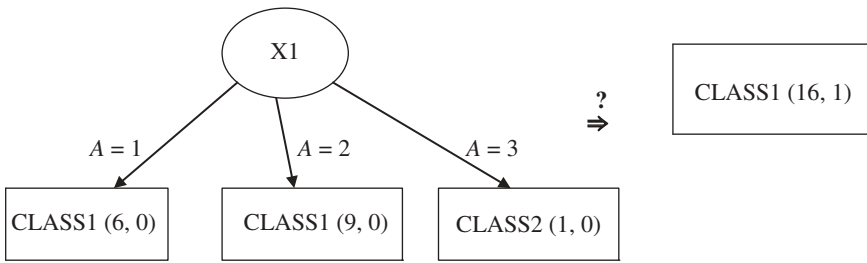
Figure 6.9. Pruning a subtree by replacing it with one leaf node.

To analyze the possibility of replacing the subtree with a leaf node, it is necessary to compute a predicted error PE for the initial tree and for a replaced node. Using default confidence of 25%, the upper confidence limits for all nodes are collected from statistical tables: $U_{25\%}(6,0) = 0.206$, $U_{25\%}(9,0) = 0.143$, $U_{25\%}(1,0) = 0.750$, and $U_{25\%}(16,1) = 0.157$. Using these values, the predicted errors for the initial tree and the replaced node are

$$PE_{tree} = 6 \cdot 0.206 + 9 \cdot 0.143 + 1 \cdot 0.750 = 3.257$$

$$PE_{node} = 16 \cdot 0.157 = 2.512$$

Since the existing subtree has a higher value of predicted error than the replaced node, it is recommended that the decision tree be pruned and the subtree replaced with the new leaf node.

6.5 *C4.5 ALGORITHM*: GENERATING DECISION RULES

Even though the pruned trees are more compact than the originals, they can still be very complex. Large decision trees are difficult to understand because each node has a specific context established by the outcomes of tests at antecedent nodes. To make a decision-tree model more readable, a path to each leaf can be transformed into an IF-THEN production rule. The IF part consists of all tests on a path, and the THEN part is a final classification. Rules in this form are called *decision rules*, and a collection of decision rules for all leaf nodes would classify samples exactly as the tree does. As a consequence of their tree origin, the IF parts of the rules would be mutually exclusive and exhaustive, so the order of the rules would not matter. An example of the transformation of a decision tree into a set of decision rules is given in Figure 6.10, where the two given attributes, *A* and *B*, may have two possible values, 1 and 2, and the final classification is into one of two classes.
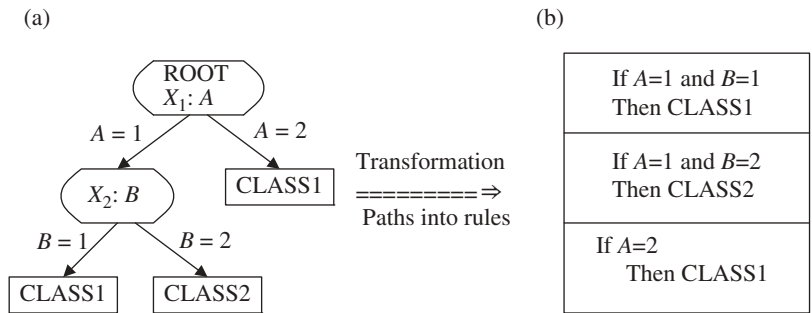
(a)   (b)

| If $A=1$ and $B=1$ Then CLASS1 |
| If $A=1$ and $B=2$ Then CLASS2 |
| If $A=2$ Then CLASS1 |

ROOT $X_1: A$

$A = 1$ $A = 2$

$X_2: B$ CLASS1

Transformation ========⇒ Paths into rules

$B = 1$ $B = 2$

CLASS1   CLASS2

Figure 6.10. Transformation of a decision tree into decision rules. (a) Decision tree. (b) Decision rules.

For our trained decision tree in Figure 6.8, the corresponding five decision rules will be:

If   Attribute1 = A and Attribute2 <= 70   *Then* Classification = CLASS1 (2.0/0);
If   Attribute1 = A and Attribute2 > 70   *Then* Classification = CLASS2 (3.4/0.4);
If   Attribute1 = B   *Then* Classification = CLASS1 (3.2 / 0);
If   Attribute1 = C and Attribute3 = True   *Then* Classification = CLASS2 (2.4 / 0);
If   Attribute1 = C and Attribute3 = False   *Then* Classification = CLASS1 (3.0 / 0).

Rewriting the tree to a collection of rules, one for each leaf in the tree, would not result in a simplified model. The number of decision rules in the classification model can be extremely large, and pruning of rules can improve readability of the model. In some cases, the antecedents of individual rules may contain irrelevant conditions. The rules can be generalized by deleting these superfluous conditions without affecting rule set accuracy. What are criteria for deletion of rule conditions? Let rule R be

$$\text{If } A \text{ Then Class-}C$$

and a more general rule R' could be

$$\text{If } A' \text{ then Class-}C$$

where A' is obtained by deleting one condition X from A ($A = A' \cup X$). The evidence for the importance of condition X must be found in the training samples. Each sample in

the database that satisfies the condition A' either satisfies or does not satisfy the extended conditions A. Also, each of these cases does or does not belong to the designated Class-C. The results can be organized into a contingency 2×2 table:

	Class-C	Other classes
Satisfies condition X	Y_1	E_1
Does not satisfy condition X	Y_2	E_2

There are $Y_1 + E_1$ cases that are covered by the original rule R, where R misclassifies E_1 of them since they belong to classes other than C. Similarly, $Y_1 + Y_2 + E_1 + E_2$ is the total number of cases covered by rule R', and $E_1 + E_2$ are errors. The criterion for the elimination of condition X from the rule is based on a pessimistic estimate of the accuracy of rules R and R'. The estimate of the error rate of rule R can be set to $U_{cf}(Y_1 + E_1, E_1)$, and that of rule R' to $U_{cf}(Y_1 + Y_2 + E_1 + E_2, E_1 + E_2)$. If the pessimistic error rate of rule R' is no greater than that of the original rule R, then it makes sense to delete condition X. Of course, more than one condition may have to be deleted when a rule is generalized. Rather than looking at all possible subsets of conditions that could be deleted, the C4.5 system performs greedy elimination: at each step, a condition with the lowest pessimistic error is eliminated. As with all greedy searches, there is no guarantee that minimization in every step will lead to a global minimum.

If, for example, the contingency table for a given rule R is given in Table 6.3, then the corresponding error rates are as follows:

1. For initially given rule R:
$$U_{cf}(Y_1 + E_1, E_1) = U_{cf}(9, 1) = 0.183$$

2. For a general rule R' without condition X:
$$U_{cf}(Y_1 + Y_2 + E_1 + E_2, E_1 + E_2) = U_{cf}(16, 1) = 0.157$$

Because the estimated error rate of the rule R' is lower than the estimated error rate for the initial rule R, a rule set pruning could be done by simplifying the decision rule R and replacing it with R'.

One complication caused by a rule's generalization is that the rules are no more mutually exclusive and exhaustive. There will be the cases that satisfy the conditions of more than one rule, or of no rules. The conflict resolution schema adopted in C4.5

TABLE 6.3. Contingency Table for the Rule R

	Class-C	Other Classes
Satisfies condition X	8	1
Does not satisfy condition X	7	0

(detailed explanations have not been given in this book) selects one rule when there is "multiple-rule satisfaction." When no other rule covers a sample, the solution is a *default rule* or a *default class*. One reasonable choice for the default class would be the class that appears most frequently in the training set. C.4.5 uses a modified strategy and simply chooses as the default class the one that contains the most training samples not covered by any rule.

The other possibility of reducing the complexity of decision rules and decision trees is a process of grouping attribute values for categorical data. A large number of values cause a large space of data. There is concern that useful patterns may not be detectable because of the insufficiency of training data or that patterns will be detected but the model will be extremely complex. To reduce the number of attribute values, it is necessary to define appropriate groups. The number of possible splitting is large: for n values, there exist $2^{n-1} - 1$ nontrivial binary partitions. Even if the values are ordered, there are $n - 1$ "cut values" for binary splitting. A simple example, which shows the advantages of grouping categorical values in decision-rule reduction, is given in Figure 6.11.

C4.5 increases the number of grouping combinations because it does not include only binary categorical data, but also *n*-ary partitions. The process is iterative, starting with an initial distribution where every value represents a separate group and then, for each new iteration, analyzing the possibility of merging the two previous groups into one. Merging is accepted if the information-gain ratio (explained earlier) is nondecreasing. A final result may be two or more groups that will simplify the classification model based on decision trees and decision rules.

C4.5 was superseded by a commercial system C5.0. The changes include:

- a variant of boosting technique, which constructs an ensemble of classifiers that are then voted to give a final classification, and
- new data types such as dates, work with "not applicable" values, concept of variable misclassification costs, and mechanisms to prefilter attributes.

C5.0 greatly improves scalability of both decision trees and rule sets and enables successful applications with large real-world data sets. In practice, these data can be translated into more complicated decision trees, which can include dozens of levels and hundreds of variables.
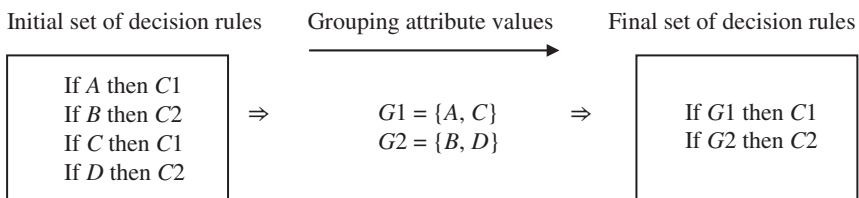

Figure 6.11. Grouping attribute values can reduce decision-rule set.

6.6   CART ALGORITHM AND GINI INDEX

CART is an acronym for *C*lassification *A*nd *R*egression *T*rees. The basic methodology of divide and conquer described in C4.5 is also used in CART. The main differences are in the tree structure, the splitting criteria, the pruning method, and the way missing values are handled.

   CART constructs trees that have only binary splits. This restriction simplifies the splitting criterion because there need not be a penalty for multiway splits. Furthermore, if the label is binary, the binary split restriction allows CART to optimally partition categorical attributes (minimizing any concave splitting criteria) to two subsets of values in the number of attribute values. The restriction has its disadvantages, however, because the tree may be less interpretable with multiple splits occurring on the same attribute at adjacent levels.

   CART uses the *Gini* diversity index as a splitting criterion instead of information-based criteria for C4.5. The CART authors favor the Gini criterion over information gain because the Gini can be extended to include symmetrized costs, and it is computed more rapidly than information gain. The Gini index is used to select the feature at each internal node of the decision tree. We define the Gini index for a data set S as follows:

$$\text{Gini}(S) = 1 - \sum_{i=0}^{c-1} p_i^2$$

where

- c is the number of predefined classes
- C_i are classes for $i = 1,\ldots,c-1$
- s_i is the number of samples belonging to class C_i
- $p_i = s_i/S$ is a relative frequency of class C_i in the set

   This metric indicates the partition purity of the data set S. For branch prediction where we have two classes, the Gini index lies within [0, 0.5]. If all the data in S belong to the same class, *Gini S* equals the minimum value 0, which means that S is pure. If *Gini S* equals 0.5, all observations in S are equally distributed among two classes. This decreases as a split favoring one class: for instance, a 70/30 distribution produces Gini index of 0.42. If we have more than two classes, the maximum possible value for index increases; for instance, the worst possible diversity for three classes is a 33% split, and it produce a *Gini* value of 0.67. The Gini coefficient, which range from 0 to 1 (for extremely large number of classes), is multiplied by 100 to range between 0 and 100 in some commercial tools.

   The quality of a split on a feature into k subsets S_i is then computed as the weighted sum of the Gini indices of the resulting subsets:

$$\text{Gini}_{\text{split}} = \sum_{i=0}^{k-1} \left(\frac{n_i}{n}\right) \text{Gini}(S_i)$$

where

- n_i is the number of samples in subset S_i after splitting
- n is the total number of samples in the given node

Thus $Gini_{split}$ is calculated for all possible features, and the feature with minimum $Gini_{split}$ is selected as split point. The procedure is repetitive as in C4.5. We may compare the results of CART and C4.5 attribute selection by applying entropy index (C4.5) and Gini index (CART) for splitting Attribute1 in Table 6.1. While we already have results for C4.5 (Gain-ratio for Attribute1), $Gini_{split}$ index for the same attribute may be calculated as

$$\text{Gini}_{\text{split}} = \sum_{i=0}^{2} \frac{n_i}{n} \text{Gini}(S_i) = \frac{5}{14} \times \left(1 - \left(\frac{2}{5} \right)^2 - \left(\frac{3}{5} \right)^2 \right)$$

$$+ \frac{4}{14} \times \left(1 - \left(\frac{0}{4} \right)^2 - \left(\frac{4}{4} \right)^2 \right) + \frac{5}{14} \times \left(1 - \left(\frac{2}{5} \right)^2 - \left(\frac{3}{5} \right)^2 \right)$$

$$= 0.34$$

Interpretation of the absolute value for $Gini_{split}$ index is not important. It is important that relatively this value is lower for attribute $A1$ than for other two attributes in Table 6.1. The reader may check easily this claim. Therefore, based on Gini index Attribute1 is selected for splitting. It is the same result obtained in C4.5 algorithm using entropy criterion. Although the results are the same in this example, for many other data sets, there could be (usually small) differences in results between these two approaches.

CART also supports the *twoing* splitting criterion, which can be used for multi-class problems. At each node, the classes are separated into two superclasses containing disjoint and mutually exhaustive classes. A splitting criterion for a two-class problem is used to find the attribute and the two superclasses that optimize the two-class criterion. The approach gives "strategic" splits in the sense that several classes that are similar are grouped together. Although twoing splitting rule allows us to build more balanced trees, this algorithm works slower than Gini rule. For example, if the total number of classes is equal to K, then we will have $2K - 1$ possible grouping into two classes. It can be seen that there is a small difference between trees constructed using Gini and trees constructed via twoing rule. The difference can be seen mainly at the bottom of the tree where the variables are less significant in comparison with top of the tree.

Pruning technique used in CART is called minimal cost complexity pruning, while C4.5 uses binomial confidence limits. The proposed approach assumes that the bias in the resubstitution error of a tree increases linearly with the number of leaf nodes. The cost assigned to a subtree is the sum of two terms: the resubstitution error

and the number of leaves times a complexity parameter α. It can be shown that, for every α value, there exists a unique smallest tree minimizing cost of the tree. Note that, although α runs through a continuum of values, there are at most a finite number of possible subtrees for analysis and pruning.

Unlike C4.5, CART does not penalize the splitting criterion during the tree construction if examples have unknown values for the attribute used in the split. The criterion uses only those instances for which the value is known. CART finds several surrogate splits that can be used instead of the original split. During classification, the first surrogate split based on a known attribute value is used. The surrogates cannot be chosen based on the original splitting criterion because the subtree at each node is constructed based on the original split selected. The surrogate splits are therefore chosen to maximize a measure of predictive association with the original split. This procedure works well if there are attributes that are highly correlated with the chosen attribute.

As its name implies, CART also supports building regression trees. Regression trees are somewhat simpler than classification trees because the growing and pruning criteria used in CART are the same. The regression tree structure is similar to a classification tree, except that each leaf predicts a real number. The resubstitution estimate for pruning the tree is the mean squared error.

Among main advantages of CART method is its robustness to outliers and noisy data. Usually the splitting algorithm will isolate outliers in an individual node or nodes. Also, an important practical property of CART is that the structure of its classification or regression trees is invariant with respect to monotone transformations of independent variables. One can replace any variable with its logarithm or square root value; the structure of the tree will not change. One of CART disadvantages is that the system may have unstable decision trees. Insignificant modification of learning samples, such as eliminating several observations, could lead to radical changes in decision tree: with significant increase or decrease of tree complexity being changes in splitting variables and values.

C4.5 and CART are two popular algorithms for decision-tree induction; however, their corresponding splitting criteria, information gain and the Gini index, are considered to be skew sensitive. In other words they are not applicable, or minimum they are not successfully applied, in cases when classes are not equally distributed in training and testing data sets. It becomes important to design a decision-tree splitting criterion that captures the divergence in distributions without being dominated by the class priors. One of the proposed solutions is the *Hellinger distance* as a decision-tree splitting criterion. Resent experimental results show that this distance measure is skew insensitive.

For application as a decision-tree splitting criterion, we assume a countable space, so all continuous features are discretized into p partitions or bins. Assuming a two-class problem (class + and class −), let X+ be samples belonging to class + and X− are samples with class −. Then, we are essentially interested in calculating the "distance" in the normalized frequencies distributions aggregated over all the partitions of the two class distributions X+ and X−. The Hellinger distance between X+ and X− is
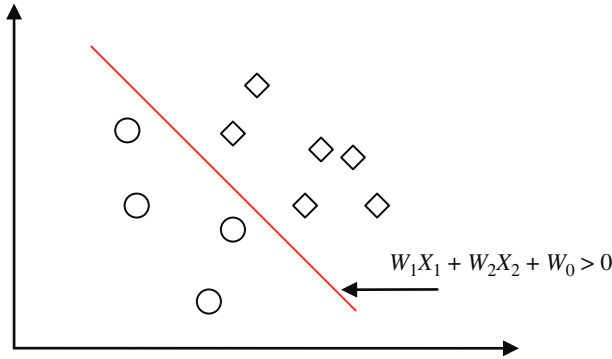
Figure 6.12. Multivariate decision node.

$$d_{\mathrm{H}}(X_+, X_-) = \sqrt{\sum_{j=1}^{p} \left(\sqrt{\frac{|X_{+j}|}{|X_+|}} - \sqrt{\frac{|X_{-j}|}{|X_-|}} \right)^2}$$

This formulation is strongly skew insensitive. Experiments with real-world data show that the proposed measure may be successfully applied for cases where $X+ \ll X-$. It essentially captures the divergence between the feature value distributions given the two different classes.

Recent developments in the field extend technology toward multivariate trees. In a multivariate tree, at a decision node, all input dimensions can be used for testing (for example, $w_1 x_1 + w_2 x_2 + w_0 > 0$ as it is presented in Figure 6.12). It is a hyperplane with an arbitrary orientation. This is 2^d ($^N{}_d$) possible hyperplanes and exhaustive search is not practical.

With linear multivariate nodes, we can use hyperplanes for better approximation using fewer nodes. A disadvantage of the technique is that multivariate nodes are mode difficult to interpret. Also, more complex nodes require more data. The earliest version of multivariate trees is implemented in CART algorithm, which fine-tunes the weights w_i one by one to decrease impurity. CART also has a preprocessing stage to decrease dimensionality through subset input selection (and therefore reduction of node complexity).

6.7   LIMITATIONS OF DECISION TREES AND DECISION RULES

Decision-rule- and decision-tree-based models are relatively simple and readable, and their generation is very fast. Unlike many statistical approaches, a logical approach does not depend on assumptions about distribution of attribute values or independence

of attributes. Also, this method tends to be more robust across tasks than most other statistical methods. But there are also some disadvantages and limitations of a logical approach, and a data-mining analyst has to be aware of it because the selection of an appropriate methodology is a key step to the success of a data-mining process.

If data samples are represented graphically in an n-dimensional space, where N is the number of attributes, then a logical classifier (decision trees or decision rules) divides the space into regions. Each region is labeled with a corresponding class. An unseen testing sample is then classified by determining the region into which the given point falls. Decision trees are constructed by successive refinement, splitting existing regions into smaller ones that contain highly concentrated points of one class. The number of training cases needed to construct a good classifier is proportional to the number of regions. More complex classifications require more regions that are described with more rules and a tree with higher complexity. All that will require an additional number of training samples to obtain a successful classification.

A graphical representation of decision rules is given by orthogonal hyperplanes in an n-dimensional space. The regions for classification are hyperrectangles in the same space. If the problem at hand is such that the classification hyperplanes are not orthogonal, but are defined through a linear (or nonlinear) combination of attributes, such as the example in Figure 6.13, then that increases the complexity of a rule-based model. A logical approach based on decision rules tries to approximate nonorthogonal and, sometimes, nonlinear classification with hyperrectangles; classification becomes extremely complex with large number of rules and a still larger error.

A possible solution to this problem is an additional iteration of the data-mining process: returning to the beginning of preprocessing phases, it is necessary to transform input features into new dimensions that are linear (or nonlinear) combinations of initial inputs. This transformation is based on some domain heuristics, and it requires emphasis with additional effort in data preparation; the reward is a simpler classification model with a lower error rate.

The other types of classification problems, where decision rules are not the appropriate tool for modeling, have classification criteria in the form: a given class is
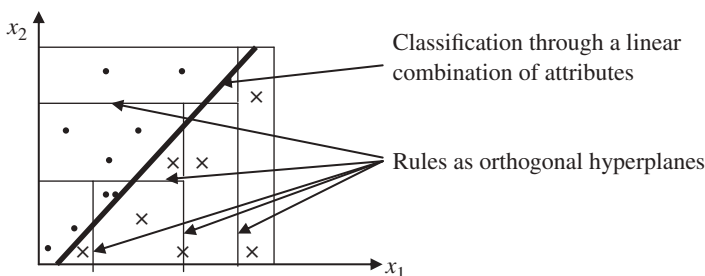

Figure 6.13. Approximation of nonorthogonal classification with hyperrectangles.

supported if n out of m conditions are present. To represent this classifier with rules, it would be necessary to define $\binom{m}{n}$ regions only for one class. Medical diagnostic decisions are a typical example of this kind of classification. If 4 out of 11 symptoms support diagnosis of a given disease, then the corresponding classifier will generate 330 regions in an 11-dimensional space for positive diagnosis only. That corresponds to 330 decision rules. Therefore a data-mining analyst has to be very careful in applying the orthogonal-classification methodology of decision rules for this type of nonlinear problems.

Finally, introducing new attributes rather than removing old ones can avoid the sometimes-intensive fragmentation of the n-dimensional space by additional rules. Let us analyze a simple example. A classification problem is described by nine binary inputs $\{A_1, A_2,\ldots,A_9\}$, and the output class C is specified by the logical relation

$$(A_1 \vee A_2 \vee A_3) \wedge (A_4 \vee A_5 \vee A_6) \wedge (A_7 \vee A_8 \vee A_9) \to C$$

The above expression can be rewritten in a conjunctive form:

$$((A_1 \wedge A_4 \wedge A_7) \vee (A_1 \wedge A_5 \wedge A_7) \vee \ldots) \to C$$

and it will have 27 factors with only $\wedge$ operations. Every one of these factors is a region in a nine-dimensional space and corresponds to one rule. Taking into account regions for negative examples, there exist about 50 leaves in the decision tree (and the same number of rules) describing class C. If new attributes are introduced

$$B_1 = A_1 \vee A2 \vee A_3,$$
$$B_2 = A_4 \vee A_5 \vee A_6, \quad \text{and}$$
$$B_3 = A_7 \vee A_8 \vee A_9$$

the description of class C will be simplified into the logical rule

$$B_1 \wedge B_2 \wedge B_3 \to C$$

It is possible to specify the correct classification using a decision tree with only four leaves. In a new three-dimensional space (B_1, B_2, B_3), there will be only four decision regions. This kind of simplification via constructive induction (development of new attributes in the preprocessing phase) can be applied also in a case n-of-m attributes' decision. If none of previous transformations are found appropriate, the only way to deal with the increased fragmentation of an n-dimensional space is to bring more data to bear on the problem.

6.8   REVIEW QUESTIONS AND PROBLEMS

1. Explain the differences between the statistical and logical approaches in the con-
struction of a classification model.

2. What are the new features of C4.5 algorithm comparing with original Quinlan's
ID3 algorithm for decision-tree generation?

3. Given a data set X with three-dimesional categorical samples:

X:	Attribute1	Attribute2	Class
	T	1	$C2$
	T	2	C_1
	F	1	C_2
	F	2	C_2

Construct a decision tree using the computation steps given in the C4.5 algorithm.

4. Given a training data set Y:

Y:	A	B	C	Class
	15	1	A	C_1
	20	3	B	C_2
	25	2	A	C_1
	30	4	A	C_1
	35	2	B	C_2
	25	4	A	C_1
	15	2	B	C_2
	20	3	B	C_2

(a) Find the best threshold (for the maximal gain) for attribute A.
(b) Find the best threshold (for the maximal gain) for attribute B.
(c) Find a decision tree for data set Y.
(d) If the testing set is

A	B	C	Class
10	2	A	C_2
20	1	B	C_1
30	3	A	C_2
40	2	B	C_2
15	1	B	C_1

what is the percentage of correct classifications using the decision tree developed in c).
   (e) Derive decision rules from the decision tree.

5. Use the C4.5 algorithm to build a decision tree for classifying the following objects:

Class	Size	Color	Shape
A	Small	Yellow	Round
A	Big	Yellow	Round
A	Big	Red	Round
A	Small	Red	Round
B	Small	Black	Round
B	Big	Black	Cube
B	Big	Yellow	Cube
B	Big	Black	Round
B	Small	Yellow	Cube

6. Given a training data set $Y*$ with missing values ($-$):

$Y*$:	A	B	C	Class
	15	1	A	C_1
	20	3	B	C_2
	25	2	A	C_1
	—	4	A	C_1
	35	2	—	C_2
	25	4	A	C_1
	15	2	B	C_2
	20	3	B	C_2

(a) Apply a modified C4.5 algorithm to construct a decision tree with the (T_i/E) parameters explained in Section 6.3.
(b) Analyze the possibility of pruning the decision tree obtained in (a).
(c) Generate decision rules for the solution in (a). Is it necessary to generate a default rule for this rule-based model?

7. Why is postpruning in C4.5 defined as pessimistic pruning?

8. Suppose that two decision rules are generated with C4.5:

$$\text{Rule1}: (X > 3) \wedge (Y \geq 2) \rightarrow \text{Class1} \ (9.6/0.4)$$
$$\text{Rule2}: (X > 3) \wedge (Y < 2) \rightarrow \text{Class2} \ (2.4/2.0).$$

Analyze if it is possible to generalize these rules into one using confidence limit $U_{25\%}$ for the binomial distribution.

9. Discuss the complexity of the algorithm for optimal splitting of numeric attributes into more than two intervals.

10. In real-world data-mining applications, a final model consists of extremely large number of decision rules. Discuss the potential actions and analyses you should perform to reduce the complexity of the model.

11. Search the Web to find the basic characteristics of publicly available or commercial software tools for generating decision rules and decision trees. Document the results of your search.

12. Consider a binary classification problem (output attribute Value = {Low, High}) with the following set of input attributes and attribute values:
    - Air Conditioner = {Working, Broken}
    - Engine = {Good, Bad}
    - Mileage = {High, Medium, Low}
    - Rust = {Yes, No}

    Suppose a rule-based classifier produces the following rule set:

    Mileage = High −→ Value = Low
    Mileage = Low −→ Value = High
    Air Conditioner = Working and Engine = Good −→ Value = High
    Air Conditioner = Working and Engine = Bad −→ Value = Low
    Air Conditioner = Broken −→ Value = Low

    (a) Are the rules mutually exclusive? Explain your answer.
    (b) Is the rule set exhaustive (covering each possible case)? Explain your answer.
    (c) Is ordering needed for this set of rules? Explain your answer.
    (d) Do you need a default class for the rule set? Explain your answer.

13. Of the following algorithms:
    1. C4.5
    2. K-nearest neighbor
    3. Naïve Bayes
    4. Linear regression
        (a) Which are fast in training but slow in classification?
        (b) Which one produces classification rules?
        (c) Which one requires discretization of continuous attributes before application?
        (d) Which model is the most complex?

14. (a) How much information is involved in choosing one of eight items, assuming that they have an equal frequency?
    (b) One of 16 items?

15. The following data set will be used to learn a decision tree for predicting whether a mushroom is edible or not based on its shape, color, and odor.

Shape	Color	Odor	Edible
C	B	1	Yes
D	B	1	Yes
D	W	1	Yes
D	W	2	Yes
C	B	2	Yes
D	B	2	No
D	G	2	No
C	U	2	No
C	B	3	No
C	W	3	No
D	W	3	No

(a) What is entropy $H(\text{Edible}|\text{Odor} = 1 \text{ or Odor} = 3)$?
(b) Which attribute would the C4.5 algorithm choose to use for the root of the tree?
(c) Draw the full decision tree that would be learned for this data (no pruning).
(d) Suppose we have a validation set as follows. What will be the training set error and validation set error of the tree? Express your answer as the number of examples that would be misclassified.

Shape	Color	Odor	Edible
C	B	2	No
D	B	2	No
C	W	2	Yes

16. Suppose we have three binary input attributes (A, B, and C), class attribute as the output, and four training examples. We are interested in finding a minimum-depth decision tree consistent with the training data. Find the tree using C4.5, and show it will not find the decision tree with the minimum depth.

A	B	C	Class
1	1	0	0
1	0	1	1
0	1	1	1
0	0	1	0

17. Consider the rules *Age > 40 ⇒ Donor* and *Age ≤ 50 ⇒¬Donor*.

    (a) Are these two rules mutually exclusive?

    (b) Are these two rules exhaustive?

18. Given a decision tree, you have the option of (a) converting the decision tree to rules and then pruning the resulting rules or (b) pruning the decision tree and then converting the pruned tree to rules. What advantage does (a) have over (b)?

6.9   REFERENCES FOR FURTHER STUDY

1. Quinlan, J. R., *C4.5: Programs for Machine Learning*, Morgan Kaufmann, San Mateo, CA, 1992.

The book outlines the C4.5 algorithm step by step, with detailed explanations and many illustrative examples. The second part of the book is taken up by the source listing of the C program that makes up the C4.5 system. The explanations in the book are intended to give a broad-brush view of C4.5 inductive learning with many small heuristics, leaving the detailed discussion to the code itself.

2. Mitchell, T., *Machine Learning*, McGraw Hill, New York, NY, 1997.

This is one of the most comprehensive books on machine learning. Systematic explanations of all methods and a large number of examples for all topics are the main strengths of the book. Inductive machine-learning techniques are only a part of the book, but for a deeper understanding of current data-mining technology and newly developed techniques, it is very useful to get a global overview of all approaches in machine learning.

3. Russell, S., P. Norvig, *Artificial Intelligence: A Modern Approach*, Prentice Hall, Upper Saddle River, NJ, 1995.

The book gives a unified presentation of the artificial intelligence field using an agent-based approach. Equal emphasis is given to theory and practice. An understanding of the basic concepts in AI, including an approach to inductive machine learning, is obtained through layered explanations and agent-based implementations of algorithms.

4. Dzeroski S., N. Lavrac, eds., *Relational Data Mining*, Springer-Verlag, Berlin, Germany, 2001.

Relational data mining has its roots in inductive-logic programming, an area in the intersection of machine learning and programming languages. The book provides a thorough overview of different techniques and strategies used in knowledge discovery from multirelational data. The chapters describe a broad selection of practical, inductive-logic programming approaches to relational data mining and give a good overview of several interesting applications.

5. Kralj Novak P., Lavrac N., Webb G. L., Supervised Descriptive Rule Discovery: A Unifying Survey of Contrast Set, Emerging Pattern and Subgroup Mining, *Journal of Machine Learning Research*, Vol. 10, 2009, pp. 377–403.

This paper gives a survey of contrast set mining (CSM), emerging pattern mining (EPM), and subgroup discovery (SD) in a unifying framework named *supervised descriptive rule discovery*. While all these research areas aim at discovering patterns in the form of rules induced from labeled data, they use different terminology and task definitions, claim to have different goals, claim to use different rule learning heuristics, and use different means for selecting subsets of induced patterns. This paper contributes a novel understanding of these subareas of data mining by presenting a unified terminology, by explaining the apparent differences between the learning tasks as variants of a unique supervised descriptive rule discovery task, and by exploring the apparent differences between the approaches.

6. Maimon Oded Z, Rokach Lior, *Data Mining With Decision Trees: Theory And Applications*, 2nd edition, World Scientific, 2014.

Decision trees have become one of the most powerful and popular approaches in knowledge discovery and data mining; it is the science of exploring large and complex bodies of data in order to discover useful patterns. Decision-tree learning continues to evolve over time. Existing methods are constantly being improved and new methods introduced. This second edition is dedicated entirely to the field of decision trees in data mining to cover all aspects of this important technique, as well as improved or new methods and techniques developed after the publication of our first edition. In this new edition, all chapters have been revised and new topics brought in. New topics include cost-sensitive active learning, learning with uncertain and imbalanced data, using decision trees beyond classification tasks, privacy preserving decision-tree learning, lessons learned from comparative studies, and learning decision trees for big data. A walkthrough guide to existing open-source data-mining software is also included in this edition. This book invites readers to explore the many benefits in data mining that decision trees offer.

7

ARTIFICIAL NEURAL NETWORKS

Chapter Objectives

- Identify basic components of artificial neural networks and their properties and capabilities.
- Describe common learning tasks such as pattern association, pattern recognition, approximation, control, and filtering that are performed by artificial neural networks.
- Compare different artificial neural-network architecture such as feedforward and recurrent networks, and discuss their applications
- Explain the learning process at the level of an artificial neuron and its extension for multiplayer feedforward neural networks.
- Compare the learning processes and the learning tasks of competitive networks and feedforward networks.
- Presents basic principles of Kohonen maps and their applications

- Discuss the requirements for good generalizations with artificial neural networks, based on heuristic parameter tuning.
- Introduce basic principles of deep learning and deep neural networks.
- Analyze main components of convolutional neural networks (CNNs).

Work on artificial neural networks (ANNs) has been motivated by the recognition that the human brain computes in an entirely different way from the conventional digital computer. It was a great challenge for many researchers in different disciplines to model the brain's computational processes. The brain is a highly complex, nonlinear, and parallel information-processing system. It has the capability to organize its components and to perform certain computations with a higher quality and many times faster than the fastest computer in existence today. Examples of these processes are pattern recognition, perception, and motor control. ANNs have been studied for more than four decades since Rosenblatt first applied the *single-layer perceptrons* to pattern-classification learning in the late 1950s.

An ANN is an abstract computational model of the human brain. The human brain has an estimated 10^{11} tiny units called neurons. These neurons are interconnected with an estimated 10^{15} links. Similar to the brain, an ANN is composed of artificial neurons (or processing units) and interconnections. When we view such a network as a graph, neurons can be represented as nodes (or vertices) and interconnections as edges. Although the term artificial neural network is most commonly used, other names include "neural network," parallel distributed processing (PDP) system, connectionist model, and distributed adaptive system. ANNs are also referred to in the literature as neurocomputers.

A neural network, as the name indicates, is a network structure consisting of a number of nodes connected through directional links. Each node represents a processing unit, and the links between nodes specify the causal relationship between connected nodes. All nodes are adaptive, which means that the outputs of these nodes depend on modifiable parameters pertaining to these nodes. Although there are several definitions and several approaches to the ANN concept, we may accept the following definition, which views the ANN as a formalized adaptive machine:

> *An artificial neural network is a massive parallel distributed processor made up of simple processing units. It has the ability to learn experiential knowledge expressed through interunit connection strengths, and can make such knowledge available for use.*

It is apparent that an ANN derives its computing power through, first, its massive parallel distributed structure and, second, its ability to learn and therefore to generalize. Generalization refers to the ANN producing reasonable outputs for new inputs not encountered during a learning process. The use of ANNs offers several useful properties and capabilities:

1. *Nonlinearity*: An artificial neuron as a basic unit can be a linear or nonlinear processing element, but the entire ANN is highly nonlinear. It is a special kind

of nonlinearity in the sense that it is distributed throughout the network. This characteristic is especially important, for ANN models the inherently nonlinear real-world mechanisms responsible for generating data for learning.

2. *Learning from examples*: An ANN modifies its interconnection weights by applying a set of training or learning samples. The final effects of a learning process are tuned parameters of a network (the parameters are distributed through the main components of the established model), and they represent implicitly stored knowledge for the problem at hand.

3. *Adaptivity*: An ANN has a built-in capability to adapt its interconnection weights to changes in the surrounding environment. In particular, an ANN trained to operate in a specific environment can be easily retrained to deal with changes in its environmental conditions. Moreover, when it is operating in a nonstationary environment, an ANN can be designed to adopt its parameters in real time.

4. *Evidential response*: In the context of data classification, an ANN can be designed to provide information not only about which particular class to select for a given sample but also about confidence in the decision made. This later information may be used to reject ambiguous data, should they arise, and therefore improve the classification performance or performances of the other tasks modeled by the network.

5. *Fault tolerance*: An ANN has the potential to be inherently fault tolerant or capable of robust computation. Its performances do not degrade significantly under adverse operating conditions such as disconnection of neurons and noisy or missing data. There is some empirical evidence for robust computation, but usually it is uncontrolled.

6. *Uniformity of analysis and design*: Basically, ANNs enjoy universality as information processors. The same principles, notation, and the same steps in methodology are used in all domains involving application of ANNs.

To explain a classification of different types of ANNs and their basic principles, it is necessary to introduce an elementary component of every ANN. This simple processing unit is called an artificial neuron.

7.1 MODEL OF AN ARTIFICIAL NEURON

An artificial neuron is an information-processing unit that is fundamental to the operation of an ANN. The block diagram (Fig. 7.1), which is a model of an artificial neuron, shows that it consists of three basic elements:

1. *A set of connecting links* from different inputs x_i (or synapses), each of which is characterized by a weight or strength w_{ki}. The first index refers to the neuron in question, and the second index refers to the input of the synapse to which
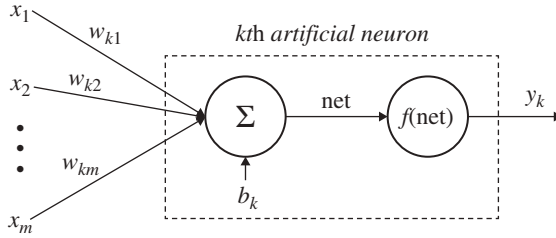
Figure 7.1. Model of an artificial neuron.

the weight refers. In general, the weights of an artificial neuron may lie in a range that includes negative as well as positive values.

2. *An adder* for summing the input signals x_i weighted by the respective synaptic strengths w_{ki}. The operation described here constitutes a linear combiner.

3. *An activation function f* for limiting the amplitude of the output y_k of a neuron.

The model of the neuron given in Figure 7.1 also includes an externally applied bias, denoted by b_k. The bias has the effect of increasing or lowering the net input of the activation function, depending on whether it is positive or negative.

In mathematical terms, an artificial neuron is an abstract model of a natural neuron, and its processing capabilities are formalized using the following notation. First, there are several inputs x_i, $i = 1,\ldots,m$. Each input x_i is multiplied by the corresponding weight w_{ki} where k is the index of a given neuron in an ANN. The weights simulate the biological synaptic strengths in a natural neuron. The weighted sum of products $x_i\, w_{ki}$ for $i = 1,\ldots,m$ is usually denoted as *net* in the ANN literature:

$$\mathrm{net}_k = x_1 w_{k1} + x_2 w_{k2} + \cdots + x_m w_{km} + b_k$$

Using adopted notation for $w_{k0} = b_k$ and default input $x_0 = 1$, a new uniform version of net summation will be

$$\mathrm{net}_k = x_0 w_{k0} + x_1 w_{k1} + x_2 w_{k2} + \cdots + x_m w_{km} = \sum_{i=0}^{m} x_i w_{ki}$$

The same sum can be expressed in vector notation as a scalar product of two m-dimensional vectors:

$$\mathrm{net}_k = X \cdot W$$

where

$$X = \{x_0, x_1, x_2, \ldots, x_m\}$$
$$W = \{w_{k0}, w_{k1}, w_{k2}, \ldots, w_{km}\}$$

Finally, an artificial neuron computes the output y_k as a certain function of net_k value:

$$y_k = f(net_k)$$

The function f is called the activation function. Various forms of activation functions can be defined. Some commonly used activation functions are given in Table 7.1.

Now, when we introduced the basic components of an artificial neuron and its functionality, we can analyze all the processing phases in a single neuron. For example, for the neuron with three inputs and one output, the corresponding input values, weight factors, and bias are given in Figure 7.2a. It is necessary to find the output y for different activation functions such as symmetrical hard limit, saturating linear, and log-sigmoid.

1. Symmetrical hard limit

$$net = 0.5 \cdot 0.3 + 0.5 \cdot 0.2 + 0.2 \cdot 0.5 + (-0.2) \cdot 1 = 0.15$$
$$y = f(net) = f(0.15) = 1$$

2. Saturating linear

$$net = 0.15 \, (\text{computation is the same as for case 1})$$
$$y = f(net) = f(0.15) = 0.15$$

3. Log-sigmoid

$$net = 0.15 \, (\text{computation is the same as for case 1})$$
$$y = f(net) = f(0.15) = \frac{1}{(1 + e^{-0.15})} = 0.54$$

The basic principles of computation for one node may be extended for an ANN with several nodes even if they are in different layers, as given in Figure 7.2b. Suppose that for the given configuration of three nodes, all bias values are equal to 0 and activation functions for all nodes are symmetric saturating linear. What is the final output y_3 from the node 3?
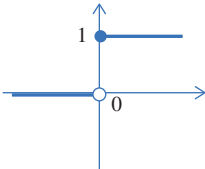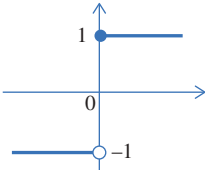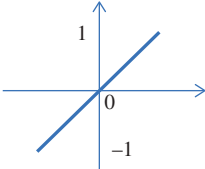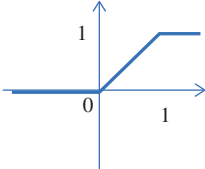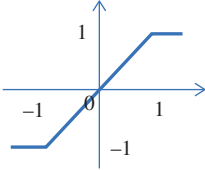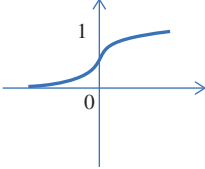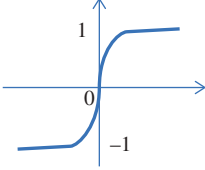
The processing of input data is layered. In the first step, the neural network performs the computation for nodes 1 and 2 that are in the first layer:

$$net_1 = 1 \cdot 0.2 + 0.5 \cdot 0.5 = 0.45 \Rightarrow y_1 = f(0.45) = 0.45$$

$$net_2 = 1 \cdot (-0.6) + 0.5 \cdot (-1) = -1.1 \Rightarrow y_2 = f(-1.1) = -1$$

Outputs y_1 and y_2 from the first layer nodes are inputs for node 3 in the second layer:

TABLE 7.1. A Neuron's Common Activation Functions

| Activation Function	Input–Output Relation	Graph
Hard limit	$y = \begin{cases} 1 & \text{if } net \geq 0 \\ 0 & \text{if } net < 0 \end{cases}$	
Symmetrical hard limit	$y = \begin{cases} 1 & \text{if } net \geq 0 \\ -1 & \text{if } net < 0 \end{cases}$	
Linear	$y = net$	
Saturating linear	$y = \begin{cases} 1 & \text{if } net > 1 \\ net & \text{if } 0 \leq net \leq 1 \\ 0 & \text{if } net < 0 \end{cases}$	
Symmetric saturating linear	$y = \begin{cases} 1 & \text{if } net > 1 \\ net & \text{if } -1 \leq net \leq 1 \\ -1 & \text{if } net < -1 \end{cases}$	
Log-sigmoid	$y = \dfrac{1}{1 + e^{-net}}$	
Hyperbolic tangent sigmoid	$y = \dfrac{e^{net} - e^{-net}}{e^{net} + e^{-net}}$	

(a)

(b)
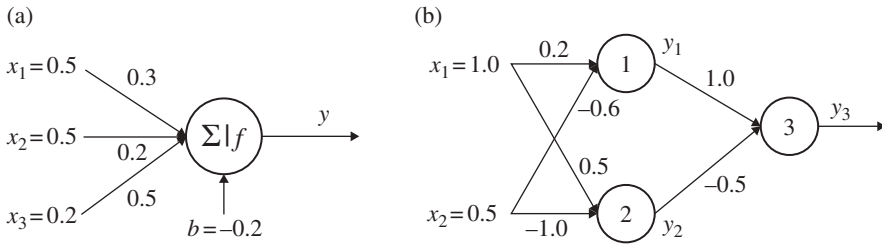
Figure 7.2. Examples of artificial neurons and their interconnections. (a) A single node. (b) Three interconnected nodes.

$$\text{net}_3 = y_1 \cdot 1 + y_2 \cdot (-0.5) = 0.45 \cdot 1 + (-1) \cdot (-0.5) = 0.95 \Rightarrow y_3 = f(0.95) = 0.95$$

As we can see from the previous examples, the processing steps at the node level are very simple. In highly connected networks of artificial neurons, computational tasks are multiplied with an increase in the number of nodes. The complexity of processing depends on the ANN architecture.

7.2 ARCHITECTURES OF ARTIFICIAL NEURAL NETWORKS

The architecture of an ANN is defined by the characteristics of a node and the characteristics of the node's connectivity in the network. The basic characteristics of a single node have been given in a previous section, and in this section the parameters of connectivity will be introduced. Typically, network architecture is specified by the number of inputs to the network, the number of outputs, the total number of elementary nodes that are usually equal processing elements for the entire network, and their organization and interconnections. Neural networks are generally classified into two categories on the basis of the type of interconnections: *feedforward* and *recurrent*.

The network is *feedforward* if the processing propagates from the input side to the output side unanimously, without any loops or feedbacks. In a layered representation of the feedforward neural network, there are no links between nodes in the same layer; outputs of nodes in a specific layer are always connected as inputs to nodes in succeeding layers. This representation is preferred because of its modularity, i.e., nodes in the same layer have the same functionality or generate the same level of abstraction about input vectors. If there is a feedback link that forms a circular path in a network (usually with a delay element as a synchronization component), then the network is *recurrent*. Examples of ANNs belonging to both classes are given in Figure 7.3.

Although many neural-network models have been proposed in both classes, the multilayer feedforward network with a backpropagation-learning mechanism is the most widely used model in terms of practical applications. Why multilayered networks? A simple example will show the basic differences in application requirements between single-layer and multilayer networks.
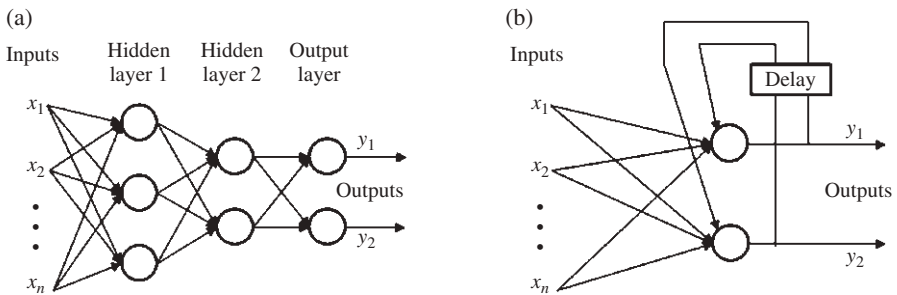
Figure 7.3. Typical architectures of artificial neural networks. (a) Feedforward network. (b) Recurrent network.
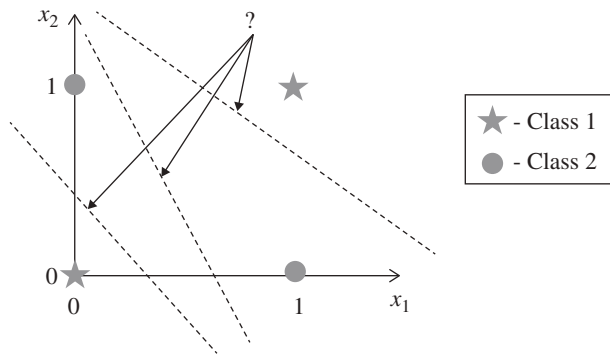

Figure 7.4. XOR problem.

The simplest and well-known classification problem, very often used as an illustration in the neural-network literature, is the exclusive-OR (XOR) problem. The task is to classify a binary input vector X to class 0 if the vector has an even number of 1's or otherwise assign it to class 1. The XOR problem is not linearly separable; this can easily be observed from the plot in Figure 7.4 for a two-dimensional (2D) input vector $X = \{x_1, x_2\}$. There is no possibility of obtaining a single linear separation of points that belong to different classes. In other words, we cannot use a single-layer network to construct a straight line (in general, it is a linear hyperplane in an n-dimensional space) to partition the 2D input space into two regions, each containing data points of only the same class. It is possible to solve the problem with a two-layer network, as illustrated in Figure 7.5, in which one possible solution for the connection weights and thresholds is indicated. This network generates a nonlinear separation of points in a 2D space.

The basic conclusion from this example is that single-layer ANNs are a convenient modeling tool only for relatively simple problems that are based on linear models. For most real-world problems, where models are highly nonlinear, multilayer networks are better and maybe the only solution.
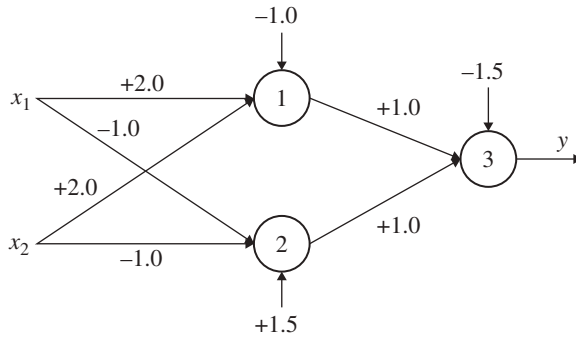
Figure 7.5. XOR solution: The two-layer ANN with the hard-limit activation function.

7.3 LEARNING PROCESS

A major task for an ANN is to learn a model of the world (environment) in which it is embedded and to maintain the model sufficiently consistent with the real world so as to achieve the specified goals of the concerned application. The learning process is based on data samples from the real world, and here lies a fundamental difference between the design of an ANN and a classical information-processing system. In the latter case, we usually proceed by first formulating a mathematical model of environmental observations, validating the model with real data, and then building (programming) the system on the basis of the model. In contrast, the design of an ANN is based directly on real-life data, with the data set being permitted to "speak for itself." Thus, an ANN not only provides the implicit model formed through the learning process but also performs the information-processing function of interest.

The property that is of primary significance for an ANN is the ability of the network to learn from its environment based on real-life examples and to improve its performance through that learning process. An ANN learns about its environment through an interactive process of adjustments applied to its connection weights. Ideally, the network becomes more knowledgeable about its environment after each iteration in the learning process. It is very difficult to agree on a precise definition of the term learning. In the context of ANNs, one possible definition of inductive learning is:

Learning is a process by which the free parameters of a neural network are adapted through a process of stimulation by the environment in which the network is embedded. The type of learning is determined by the manner in which the parameters change.

A prescribed set of well-defined rules for the solution of a learning problem is called a learning algorithm. Basically, learning algorithms differ from each other in the way in which the adjustment of the weights is formulated. Another factor to be considered in the learning process is the manner in which ANN architecture (nodes and connections) is built.

To illustrate one of the learning rules, consider the simple case of a neuron k, shown in Figure 7.1, constituting the only computational node of the network. Neuron k is driven by input vector $X(n)$, where n denotes discrete time, or, more precisely, the time step of the iterative process involved in adjusting the input weights w_{ki}. Every data sample for ANN training (learning) consists of the input vector $X(n)$ and the corresponding output $d(n)$.

	Inputs	Output
Sample$_k$	$x_{k1}, x_{k2},...,x_{km}$	d_k

Processing the input vector $X(n)$, a neuron k produces the output that is denoted by $y_k(n)$:

$$y_k = f\left(\sum_{i=1}^{m} x_i w_{ki}\right)$$

It represents the only output of this simple network, and it is compared to a desired response or target output $d_k(n)$ given in the sample. An error $e_k(n)$ produced at the output is by definition

$$e_k(n) = d_k(n) - y_k(n)$$

The error signal produced actuates a control mechanism of the learning algorithm, the purpose of which is to apply a sequence of corrective adjustments to the input weights of a neuron. The corrective adjustments are designed to make the output signal $y_k(n)$ come closer to the desired response $d_k(n)$ in a step-by-step manner. This objective is achieved by minimizing a cost function $E(n)$, which is the instantaneous value of error energy, defined for this simple example in terms of the error $e_k(n)$:

$$E(n) = \frac{1}{2} e_k^2(n)$$

The learning process based on a minimization of the cost function is referred to as *error-correction learning*. In particular, minimization of $E(n)$ leads to a learning rule commonly referred to as the *delta rule* or Widrow–Hoff rule. Let $w_{kj}(n)$ denote the value of the weight factor for neuron k excited by input $x_j(n)$ at time step n. According to the delta rule, the adjustment $\Delta w_{kj}(n)$ is defined by

$$\Delta w_{kj}(n) = \eta \cdot e_k(n) \cdot x_j(n)$$

where η is a positive constant that determines the rate of learning. Therefore, the delta rule may be stated as follows: the adjustment made to a weight factor of an input

neuron connection is proportional to the product of the error signal and the input value of the connection in question.

Having computed the adjustment $\Delta w_{kj}(n)$, the updated value of synaptic weight is determined by

$$w_{kj}(n+1) = w_{kj}(n) + \Delta w_{kj}(n)$$

In effect, $w_{kj}(n)$ and $w_{kj}(n+1)$ may be viewed as the old and new values of synaptic weight w_{kj}, respectively. From Figure 7.6 we recognize that error-correction learning is an example of a closed-loop feedback system. Control theory explains that the stability of such a system is determined by those parameters that constitute the feedback loop. One of those parameters of particular interest is the learning rate η. This parameter has to be carefully selected to ensure that the stability of convergence of the iterative-learning process is achieved. Therefore, in practice, this parameter plays a key role in determining the performance of error-correction learning.

Let us analyze one simple example of the learning process performed on a single artificial neuron in Figure 7.7a, with a set of the three training (or learning) examples given in Figure 7.7b.

The process of adjusting the weight factors for a given neuron will be performed with the learning rate $\eta = 0.1$. The bias value for the neuron is equal 0, and the
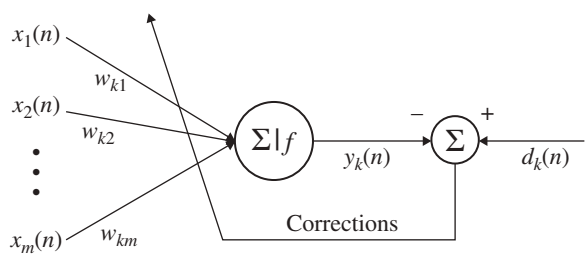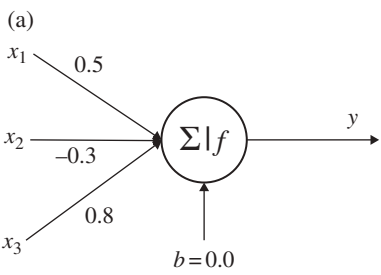

Figure 7.6. Error-correction learning performed through weight adjustments.

(a)

(b)

n (sample)	x_1	x_2	x_3	d
1	1.0	1	0.5	0.7
2	−1.0	0.7	−0.5	0.2
3	0.3	0.3	−0.3	0.5

Figure 7.7. Initialization of the error-correction learning process for a single neuron. (a) Artificial neuron with the feedback. (b) Training data set for a leraning process.

activation function is linear. The first iteration of a learning process, and only for the first training example, is performed with the following steps:

$$\text{net}(1) = 0.5 \cdot 1 + (-0.3) \cdot 1 + 0.8 \cdot 0.5 = 0.6$$

$$\Downarrow$$

$$y(1) = f(\text{net}(1)) = f(0.6) = 0.6$$

$$\Downarrow$$

$$e(1) = d(1) - y(1) = 0.7 - 0.6 = 0.1$$

$$\Downarrow$$

$$\Delta w_1(1) = 0.1 \cdot 0.1 \cdot 1 = 0.01 \Rightarrow w_1(2) = w_1(1) + \Delta w_1(1) = 0.5 + 0.01 = 0.51$$

$$\Delta w_2(1) = 0.1 \cdot 0.1 \cdot 1 = 0.01 \Rightarrow w_2(2) = w_2(1) + \Delta w_2(1) = -0.3 + 0.01 = -0.29$$

$$\Delta w_3(1) = 0.1 \cdot 0.1 \cdot 0.5 = 0.005 \Rightarrow w_3(2) = w_3(1) + \Delta w_3(1) = 0.8 + 0.005 = 0.805$$

Similarly, it is possible to continue with the second and third examples ($n = 2$ and $n = 3$). The results of the learning corrections Δw together with new weight factors w are given in Table 7.2.

Error-correction learning can be applied on much more complex ANN architecture, and its implementation is discussed in Section 7.5, where the basic principles of multilayer feedforward ANNs with backpropagation are introduced. This example only shows how weight factors change with every training (learning) sample. We gave the results only for the first iteration. The weight-correction process will continue with either new training samples or use the same data samples in the next iterations. When to finish the iterative process is defined by a special parameter or set of parameters

TABLE 7.2. Adjustment of Weight Factors with Training Examples in Figure 7.7b

Parameter	$n = 2$	$n = 3$
x_1	−1	0.3
x_2	0.7	0.3
x_3	−0.5	−0.3
y	−1.1555	−0.18
d	0.2	0.5
e	1.3555	0.68
$\Delta w_1(n)$	−0.14	0.02
$\Delta w_2(n)$	0.098	0.02
$\Delta w_3(n)$	−0.07	−0.02
$w_1(n+1)$	0.37	0.39
$w_2(n+1)$	−0.19	−0.17
$w_3(n+1)$	0.735	0.715

called *stopping criteria*. A learning algorithm may have different stopping criteria, such as the maximum number of iterations, or the threshold level of the weight factor may change in two consecutive iterations. This parameter of learning is very important for final learning results, and it will be discussed in later sections.

7.4 LEARNING TASKS USING ANNS

The choice of a particular learning algorithm is influenced by the learning task that an ANN is required to perform. We identify six basic learning tasks that apply to the use of different ANNs. These tasks are subtypes of general learning tasks introduced in Chapter 4.

7.4.1 Pattern Association

Association has been known to be a prominent feature of human memory since Aristotle, and all models of cognition use association in one form or the other as the basic operation. Association takes one of two forms: *autoassociation* or *heteroassociation*. In *autoassociation*, an ANN is required to store a set of patterns by repeatedly presenting them to the network. The network is subsequently presented with a partial description or a distorted, noisy version of an original pattern, and the task is to retrieve and recall that particular pattern. *Heteroassociation* differs from *autoassociation* in that an arbitrary set of input patterns is paired with another arbitrary set of output patterns. *Autoassociation* involves the use of unsupervised learning, whereas *heteroassociation* learning is supervised. For both, *autoassociation* and *heteroassociation*, there are two main phases in the application of an ANN for pattern-association problems:

1. The storage phase, which refers to the training of the network in accordance with given patterns, and
2. The recall phase, which involves the retrieval of a memorized pattern in response to the presentation of a noisy or distorted version of a key pattern to the network.

7.4.2 Pattern Recognition

Pattern recognition is also a task that is performed much better by humans than by the most powerful computers. We receive data from the world around us via our senses and are able to recognize the source of the data. We are often able to do so almost immediately and with practically no effort. Humans perform pattern recognition through a learning process, so it is with ANNs.

Pattern recognition is formally defined as the process whereby a received pattern is assigned to one of a prescribed number of classes. An ANN performs pattern

recognition by first undergoing a training session, during which the network is repeatedly presented a set of input patterns along with the category to which each particular pattern belongs. Later, in a testing phase, a new pattern is presented to the network that it has not seen before, but belongs to the same population of patterns used during training. The network is able to identify the class of that particular pattern because of the information it has extracted from the training data. Graphically, patterns are represented by points in a multidimensional space. The entire space, which we call decision space, is divided into regions, each one of which is associated with a class. The decision boundaries are determined by the training process, and they are tested if a new unclassified pattern is presented to the network. In essence, pattern recognition represents a standard classification task.

7.4.3   Function Approximation

Consider a nonlinear input–output mapping described by the functional relationship

$$Y = f(X)$$

where the vector X is the input and Y is the output. The vector-value function f is assumed to be unknown. We are given the set of labeled examples $\{X_i, Y_i\}$, and we have to design an ANN that approximates the unknown function f with a function F that is very close to original function. Formally,

$$|F(X_i) - f(X_i)| < \varepsilon \quad \text{for all } X_i \text{ from the training set}$$

where ε is a small positive number. Provided that the size of the training set is large enough and the network is equipped with an adequate number of free parameters, the approximation error ε can be made small enough for the task. The approximation problem described here is a perfect candidate for supervised learning.

7.4.4   Control

Control is another learning task that can be done by an ANN. Control is applied to a process or a critical part in a system, which has to be maintained in a controlled condition. Consider the control system with feedback shown in Figure 7.8.
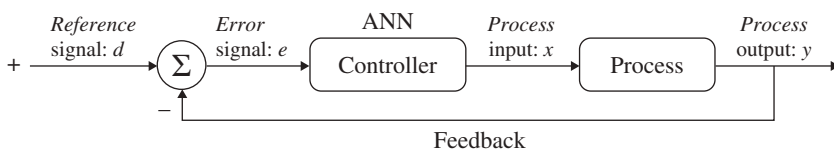

Figure 7.8. Block diagram of ANN-based feedback control system.

The system involves the use of feedback to control the output y on the level of a reference signal d supplied from the external source. A controller of the system can be realized in an ANN technology. The error signal e, which is the difference between the process output y and the reference value d, is applied to an ANN-based controller for the purpose of adjusting its free parameters. The primary objective of the controller is to supply appropriate inputs x to the process to make its output y track the reference signal d. It can be trained through:

1. *Indirect learning*—Using actual input–output measurements on the process, an ANN model of a control is first constructed offline. When the training is finished, the ANN controller may be included into the real-time loop.
2. *Direct learning*—The training phase is online, with real-time data, and the ANN controller is enabled to learn the adjustments to its free parameters directly from the process.

7.4.5 Filtering

The term filter often refers to a device or algorithm used to extract information about a particular quantity from a set of noisy data. Working with series of data in time domain, frequent domain, or other domains, we may use an ANN as a filter to perform two basic information-processing tasks:

1. *Filtering*—This task refers to the extraction of information about a particular quantity at discrete time n by using data measured up to and including time n.
2. *Smoothing*—This task differs from filtering in that data need not be available only at time n; data measured later than time n can also be used to obtain the required information. This means that in smoothing there is a delay in producing the result at discrete time n.

7.4.6 Prediction

The task of prediction is to forecast data in the future. The aim is to derive information about what the quantity of interest will be like at some time $n + n_0$ in the future, for $n_0 > 0$, by using data measured up to and including time n. Prediction may be viewed as a form of model building in the sense that the smaller we make the prediction error, the better the network serves as a model of the underlying physical process responsible for generating the data. The block diagram of an ANN for a prediction task is given in Figure 7.9.

7.5 MULTILAYER PERCEPTRONS

Multilayer feedforward networks are one of the most important and most popular classes of ANNs in real-world applications. Typically, the network consists of a set of inputs that constitute the input layer of the network, one or more hidden layers
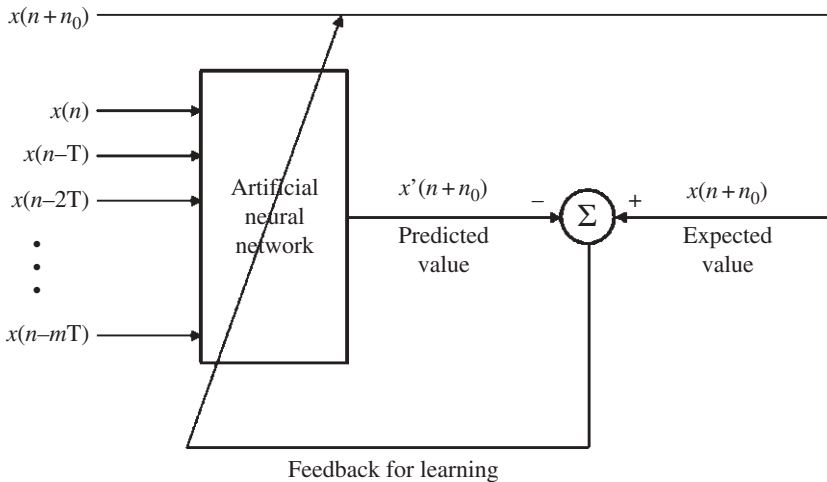
Figure 7.9. Block diagram of an ANN-based prediction.

of computational nodes, and finally an output layer of computational nodes. The processing is in a forward direction on a layer-by-layer basis. This type of ANNs are commonly referred to as multilayer perceptrons (MLPs), which represent a generalization of the simple perceptron, a network with a single layer, considered earlier in this chapter.

An MLP has three distinctive characteristics:

1. The model of each neuron in the network includes usually a nonlinear activation function, sigmoidal or hyperbolic.
2. The network contains one or more layers of hidden neurons that are not a part of the input or output of the network. These hidden nodes enable the network to learn complex and highly nonlinear tasks by extracting progressively more meaningful features from the input patterns.
3. The network exhibits a high degree of connectivity from one layer to the next one.

Figure 7.10 shows the architectural graph of an MLP with two hidden layers of nodes for processing and an output layer. The network shown here is fully connected. This means that the neuron in any layer of the network is connected to all the nodes (neurons) in the previous layer. Data flow through the network progresses in a forward direction from left to right and on a layer-by-layer basis.

MLPs have been applied successfully to solve some difficult and diverse problems by training the network in a supervised manner with a highly popular algorithm known as the *error backpropagation algorithm*. This algorithm is based on the error-correction learning rule, and it may be viewed as its generalization. Basically, error
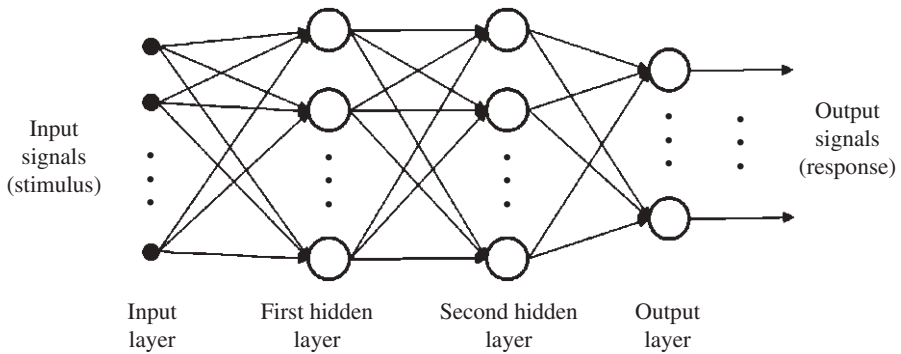
Figure 7.10. A graph of a multilayer-perceptron architecture with two hidden layers.

backpropagation learning consists of two phases performed through the different layers of the network: a forward pass and a backward pass.

In the forward pass, a training sample (input data vector) is applied to the input nodes of the network, and its effect propagates through the network layer by layer. Finally, a set of outputs is produced as the actual response of the network. During the forward phase, the synaptic weights of the network are all fixed. During the backward phase, on the other hand, the weights are all adjusted in accordance with an error-correction rule. Specifically, the actual response of the network is subtracted from a desired (target) response, which is a part of the training sample, to produce an error signal. This error signal is than propagated backward through the network against the direction of synaptic connections. The synaptic weights are adjusted to make the actual response of the network closer to the desired response.

Formalization of the backpropagation algorithm starts with the assumption that an error signal exists at the output of a neuron j at iteration n (i.e. presentation of the nth training sample). This error is defined by

$$e_j(n) = d_j(n) - y_j(n)$$

We define the instantaneous value of the error energy for neuron j as $1/2\,e_j^2(n)$. The total error energy for the entire network is obtained by summing instantaneous values over all neurons in the output layer. These are the only "visible" neurons for which the error signal can be calculated directly. We may thus write

$$E(n) = \tfrac{1}{2} \sum_{j \in C} e_j^2(n),$$

where the set C includes all neurons in the output layer of the network. Let N denote the total number of samples contained in the training set. The average squared error energy is obtained by summing $E(n)$ over all n and then normalizing it with respect to size N, as shown by

$$E_{av} = \frac{1}{N} \sum_{n=1}^{N} E(n)$$

The average error energy E_{av} is a function of all the free parameters of the network. For a given training set, E_{av} represents the cost function as a measure of learning performances. The objective of the learning process is to adjust the free parameters of the network to minimize E_{av}. To do this minimization, the weights are updated on a sample-by-sample basis for one iteration, i.e., one complete presentation of the entire training set of a network has been dealt with.

To obtain the minimization of the function E_{av}, we have to use two additional relations for node-level processing, which have been explained earlier in this chapter:

$$v_j(n) = \sum_{i=1}^{m} w_{ji}(n) x_i(n)$$

and

$$y_j(n) = \varphi(v_j(n))$$

where m is the number of inputs for jth neuron. Also, we use the symbol v as a shorthand notation of the previously defined variable net. The backpropagation algorithm applies a correction $\Delta w_{ji}(n)$ to the synaptic weight $w_{ji}(n)$, which is proportional to the partial derivative $\delta E(n)/\delta w_{ji}(n)$. Using the chain rule for derivation, this partial derivative can be expressed as

$$\frac{\partial E(n)}{\partial w_{ji}(n)} = \frac{\partial E(n)}{\partial e_j(n)} \cdot \frac{\partial e_j(n)}{\partial y_j(n)} \cdot \frac{\partial y_j(n)}{\partial v_j(n)} \cdot \frac{\partial v_j(n)}{\partial w_{ji}(n)}$$

The partial derivative $\delta E(n)/\delta w_{ji}(n)$ represents a sensitive factor, determining the direction of search in weight space. Knowing that the next relations

$$\frac{\partial E(n)}{\partial e_j(n)} = e_j(n) \quad \left(\text{from } E(n) = \tfrac{1}{2} \sum e_j^2(n)\right)$$

$$\frac{\partial e_j(n)}{\partial y_j(n)} = -1 \quad \left(\text{from } e_j(n) = d_j(n) - y_j(n)\right)$$

$$\frac{\partial y_j(n)}{\partial v_j(n)} = \varphi'(v_j(n)) \quad \left(\text{from } y_j(n) = \varphi(v_j(n))\right)$$

$$\frac{\partial v_j(n)}{\partial w_{ji}(n)} = x_i(n) \quad \left(\text{from } \sum w_{ji}(n) x_i(n)\right)$$

are valid, we can express the partial derivative $\partial E(n)/\partial w_{ji}(n)$ in the form

$$\frac{\partial E(n)}{\partial w_{ji}(n)} = -e_j(n) \cdot \varphi'\left(v_j(n)\right) \cdot x_i(n)$$

The correction $\Delta w_{ji}(n)$ applied to $w_{ji}(n)$ is defined by the delta rule

$$\Delta w_{ji}(n) = -\frac{\eta \cdot \partial E(n)}{\partial w_{ji}(n)} = \eta \cdot e_j(n) \cdot \varphi'\left(v_j(n)\right) \cdot x_i(n)$$

where η is the learning-rate parameter of the backpropagation algorithm. The use of the minus sign accounts for gradient descent in weight space, i.e., a direction for weight change that reduces the value $E(n)$. Asking for $\varphi'(v_j(n))$ in the learning process is the best explanation for why we prefer continuous functions such as log-sigmoid and hyperbolic as a standard activation function at the node level. Using the notation $\delta_j(n) = e_j(n) \cdot \varphi'_j(v_j(n))$, where $\delta_j(n)$ is the *local gradient*, the final equation for $w_{ji}(n)$ corrections is

$$\Delta w_{ji}(n) = \eta \cdot \delta_j(n) \cdot x_i(n)$$

The local gradient $\delta_j(n)$ points to the required changes in synaptic weights. According to its definition, the local gradient $\delta_j(n)$ for output neuron j is equal to the product of the corresponding error signal $e_j(n)$ for that neuron and the derivative $\varphi'(v_j(n))$ of the associated activation function.

Derivative $\varphi'(v_j(n))$ can be easily computed for a standard activation function, where differentiation is the only requirement for the function. If the activation function is sigmoid, it means that in the form

$$y_j(n) = \varphi\left(v_j(n)\right) = \frac{1}{\left(1 + e^{(-vj(n))}\right)}$$

the first derivative is

$$\varphi'\left(v_j(n)\right) = \frac{e^{(-vj(n))}}{\left(1 + e^{(-vj(n))}\right)^2} = y_j(n)\left(1 - y_j(n)\right)$$

and a final weight correction is

$$\Delta w_{ji}(n) = \eta \cdot e_j(n) \cdot y_j(n)\left(1 - y_j(n)\right) \cdot x_i(n)$$

The final correction $\Delta w_{ji}(n)$ is proportional to the learning rate η, the error value at this node is $e_j(n)$, and the corresponding input and output values are $x_i(n)$ and $y_j(n)$.

Therefore, the process of computation of $\Delta w_{ji}(n)$ for a given sample n is relatively simple and straightforward.

If the activation function is a hyperbolic tangent, a similar computation will give the final value for the first derivative $\varphi'(v_j(n))$:

$$\varphi'\left(v_j(n)\right) = \left(1 - y_j(n)\right) \cdot \left(1 + y_j(n)\right)$$

and

$$\Delta w_{ji}(n) = \eta \cdot e_j(n) \cdot \left(1 - y_j(n)\right) \cdot \left(1 + y_j(n)\right) \cdot x_i(n)$$

Again, the practical computation of $\Delta w_{ji}(n)$ is very simple because the local-gradient derivatives depend only on the output value of the node $y_j(n)$.

In general, we may identify two different cases of computation for $\Delta w_{ji}(n)$, depending on where in the network neuron j is located. In the first case, neuron j is an output node. This case is simple to handle because each output node of the network is supplied with a desired response, making it a straightforward matter to calculate the associated error signal. All previously developed relations are valid for output nodes without any modifications.

In the second case, neuron j is a hidden node. Even though hidden neurons are not directly accessible, they share responsibility for any error made at the output of the network. We may redefine the local gradient $\delta_j(n)$ for a hidden neuron j as the product of the associated derivative $\varphi'(v_j(n))$ and the weighted sum of the local gradients computed for the neurons in the next layer (hidden or output) that are connected to neuron j:

$$\delta_j(n) = \varphi'\left(v_j(n)\right) \sum_k \delta_k(n) \cdot w_{kj}(n), \quad k \in D$$

where D denotes the set of all nodes on the next layer that are connected to the node j. Going backward, all $\delta_k(n)$ for the nodes in the next layer are known before computation of the local gradient $\delta_j(n)$ for a given node on a layer closer to the inputs.

Let us analyze once more the application of the backpropagation-learning algorithm with two distinct passes of computation that are distinguished for each training example. In the first pass, which is referred to as the forward pass, the function signals of the network are computed on a neuron-by-neuron basis, starting with the nodes on first hidden layer (the input layer is without computational nodes), then the second, etc., until the computation is finished with final output layer of nodes. In this pass, based on given input values of each learning sample, a network computes the corresponding output. Synaptic weights remain unaltered during this pass.

The second, backward pass, on the other hand, starts at the output layer, passing the error signal (the difference between the computed and the desired output value) leftward through the network, layer by layer, and recursively computing the local

gradients δ for each neuron. This recursive process permits the synaptic weights of the network to undergo changes in accordance with the delta rule. For the neuron located at the output layer, δ is equal to the error signal of that neuron multiplied by the first derivative of its nonlinearity represented in the activation function. Based on local gradients δ, it is straightforward to compute Δw for each connection to the output nodes. Given the δ values for all neurons in the output layer, we use them in the previous layer before (usually the hidden layer) to compute modified local gradients for the nodes that are not the final and again to correct Δw for input connections for this layer. The backward procedure is repeated until all layers are covered and all weight factors in the network are modified. Then, the backpropagation algorithm continues with a new training sample. When there are no more training samples, the first iteration of the learning process finishes. With the same samples, it is possible to go through a second, third, and sometimes hundreds of iterations until error energy E_{av} for the given iteration is small enough to stop the algorithm.

The backpropagation algorithm provides an "approximation" to the trajectory in weight space computed by the method of steepest descent. The smaller we make the learning rate parameter η, the smaller the changes to the synaptic weights in the network will be from one iteration to the next, and the smoother will be the trajectory in weight space. This improvement, however, is attained at the cost of a slower rate of learning. If, on the other hand, we make η too large in order to speed up the learning process, the resulting large changes in the synaptic weights can cause that network to become unstable, and the solution will become oscillatory about a minimal point never reaching it.

A simple method of increasing the rate of learning yet avoiding the danger of instability is to modify the delta rule by including a *momentum term*:

$$\Delta w_{ji}(n) = \eta \cdot \delta_j(n) \cdot x_i(n) + \alpha \cdot \Delta w_{ji}(n-1)$$

where α is usually a positive number called momentum constant and $\Delta w_{ji}(n-1)$ is the correction of the weight factor for a previous $(n-1)$th sample. α, in practice, is usually set to the value between 0.1 and 1. The addition of the momentum term smoothes the weight updating and tends to resist erratic weight changes because of gradient noise or high spatial frequencies in the error surface. However, the use of momentum terms does not always seem to speed up training; it is more or less application dependent. The momentum factor represents a method of averaging; rather than averaging derivatives, momentum averages the weight changes themselves. The idea behind momentum is apparent from its name, including some kind of inertia in weight corrections. The inclusion of the momentum term in the backpropagation algorithm has a stabilizing effect in cases where corrections in weight factors have a high oscillation and sign changes. The momentum term may also have the benefit of preventing the learning process from terminating in a shallow local minimum on the error surface.

Reflecting practical approaches to the problem of determining the optimal architecture of the network for a given task, the question about values for three parameters,

namely, the number of hidden nodes (including the number of hidden layers), learning rate η, and momentum rate α, becomes very important. Usually the optimal architecture is determined experimentally, but some practical guidelines exist. If several networks with different numbers of hidden nodes give close results with respect to error criteria after the training, then the best network architecture is the one with smallest number of hidden nodes. Practically, that means starting the training process with networks that have a small number of hidden nodes, increasing this number, and then analyzing the resulting error in each case. If the error does not improve with the increasing number of hidden nodes, the latest analyzed network configuration can be selected as optimal. Optimal learning and momentum constants are also determined experimentally, but experience shows that the solution should be found with η about 0.1 and α about 0.5.

When the ANN is first set up, the initial weight factors must be given. The goal in choosing these values is to begin the learning process as fast as possible. The appropriate method is to take the initial weights as very small evenly distributed random numbers. That will cause the output values to be in a midrange regardless of the values of its inputs, and the learning process will converge much faster with every new iteration.

In backpropagation learning, we typically use the algorithm to compute the synaptic weights by using as many training samples as possible. The hope is that the neural network so designed will generalize the best. A network is said to generalize well when the input–output mapping computed by the network is correct for test data never used earlier in creating or training the network. In the MLP, if the number of hidden units is less than the number of inputs, the first layer performs a dimensionality reduction. Each hidden unit may be interpreted as defining a template. By analyzing these templates we can extract knowledge from a trained ANN. In this interpretation weights are defining relative importance in the templates. But the largest number of training samples and the largest number of learning iterations using these samples do not necessarily lead to the best generalization. Additional problems occur during the learning process, and they are briefly described through the following analysis.

The learning process using an ANN may be viewed as a curve-fitting problem. Such a viewpoint then permits us to look on generalization not as a theoretical property of neural networks but as the effect of a good nonlinear interpolation of the input data. An ANN that is designed to generalize well will produce a correct input–output mapping, even when the input is slightly different from the samples used to train the network, as illustrated in Figure 7.11a. When, however, an ANN learns from too many input–output samples, the network may end up memorizing the training data. Such a phenomenon is referred to as *overfitting* or *overtraining*. This problem has already been described in Chapter 4. When the network is overtrained, it loses the ability to generalize between similar patterns. A smoothness of input–output mapping, on the other hand, is closely related to the generalization abilities of an ANN. The essence is to select, based on training data, the simplest function for generalization; that means the smoothest function that approximates the mapping for a given error criterion. Smoothness is natural in many applications, depending on the scale of
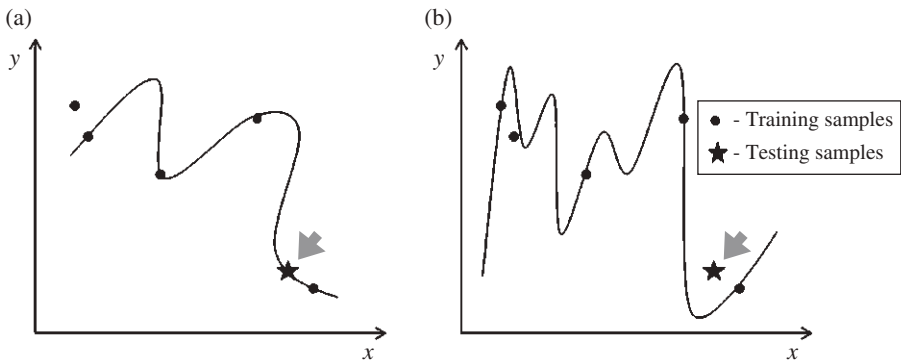
Figure 7.11. Generalization as a problem of curve fitting. (a) A fitting curve with good generalization. (b) Overfitted curve.

the phenomenon being studied. It is therefore important to seek a smooth nonlinear mapping so that the network is able to classify novel patterns correctly with respect to the training patterns. In Figure 7.11a and b, a fitting curve with a good generalization and an overfitted curve are represented for the same set of training data.

To overcome the problem of overfitting, some additional practical recommendations may be introduced for the design and application of ANN in general and MLPs in particular. In ANNs, as in all modeling problems, we want to use the simplest network that can adequately represent the training data set. Do not use a bigger network when a smaller network will work! An alternative to using the simplest network is to stop the training before the network overfits. Also, one very important constraint is that the number of network parameters should be limited. For a network to be able to generalize, it should have fewer parameters (significantly) than there are data points in the training set. ANN generalization is extremely poor if there is a large input space with very few training samples.

Interpretability of data-mining models including ANNs, or the understanding of the way inputs relate to an output in a model, is a desirable property in applied data-mining research because the intent of such studies is to gain knowledge about the underlying reasoning mechanisms. Interpretation may also be used to validate results that are inconsistent or contradictory to common understanding of issues involved, and it may also indicate problems with data or models.

While ANNs have been intensively studied and successfully used in classification and regression problems, their interpretability still remains vague. They suffer from the shortcoming of being "black boxes," i.e., without explicit mechanisms for determining why an ANN makes a particular decision. That is, one provides the input values for an ANN and obtains an output value, but generally no information is provided regarding how those outputs were obtained, how the input values correlate to the output value, and what is the meaning of large number of weight factors in the network. ANNs' acceptability as valid data-mining methods for business and research requires that beyond providing excellent predictions, they provide meaningful insight

that can be understood by variety of users: clinicians, policy makers, business planners, academicians, and laypersons. Human understanding and acceptance is greatly enhanced if the input–output relations are explicit, and end users would gain more confidence in the prediction produced.

Interpretation of trained ANNs can be considered in two forms: broad and detailed. The aim of a broad interpretation is to characterize how important an input neuron is for predictive ability of the model. This type of interpretation allows us to rank input features in order of importance. The broad interpretation is essentially a sensitivity analysis of the neural network. The methodology does not indicate the sign or direction of the effect of each input. Thus we cannot draw conclusions regarding the nature of the correlation between input descriptors and network output, but only we are concluding about the level of influence.

The goal of a detailed interpretation of an ANN is to extract the structure–property trends from an ANN model. For example, each of hidden neurons corresponds to the number of piecewise hyperplanes that are components available for approximating the target function. These hyperplanes act as the basic building blocks for constructing an explicit ANN model. To obtain a more comprehensible system that approximates the behavior of the ANN, we require the model with less complexity and at the same time maybe scarifying accuracy of results. The knowledge hidden in a complex structure of an ANN may be uncovered using a variety of methodologies that allow mapping an ANN into a rule-based system. Many authors have focused their activities on compiling the knowledge captured in the topology and weight matrix of a neural network into a symbolic form: some of them into sets of ordinary if-then rules, others into formulas from propositional logic or from non-monotonic logics, or most often into sets of fuzzy rules. These transformations make explicit the knowledge implicitly captured by the trained neural network, and it allows the human specialist to understand how the neural network generates a particular result. It is important to emphasize that any method of rule extraction from ANN is valuable only to the degree to which the extracted rules are meaningful and comprehensible to a human expert.

It is proven that the best interpretation of trained ANNs with continuous activation functions is in a form of fuzzy rule-based systems. In this way, a more comprehensible description of the action of the ANN is achieved. Multilayer feedforward ANNs are seen as additive fuzzy rule-based systems. In these systems, the outputs of each rule are weighted by the activation degree of the rule, and then they are added for an integrated representation of an ANN model. The main disadvantage of most approximation techniques of neural networks by fuzzy rules is the exponential increase of required number of rules for a good approximation. Fuzzy rules that express the input–output mapping of the ANNs are extracted using different approaches described in numerous references. If the reader is interested for more details about methodologies, the starting points may be recommended references at the end of this chapter and also introductory concepts about fuzzy systems given in Chapter 14.

7.6   COMPETITIVE NETWORKS AND COMPETITIVE LEARNING

Competitive neural networks belong to a class of recurrent networks, and they are based on algorithms of unsupervised learning, such as the competitive algorithm explained in this section. In competitive learning, the output neurons of a neural network compete among themselves to become active (to be "fired"). Whereas in MLPs several output neurons may be active simultaneously, in competitive learning, only a single output neuron is active at any one time. There are three basic elements necessary to build a network with a competitive-learning rule, a standard technique for this type of ANNs:

1. A *set of neurons* that have the same structure and that are connected with initially randomly selected weights. Therefore, the neurons respond differently to a given set of input samples.
2. A *limit value* that is determined on the strength of each neuron.
3. A *mechanism that permits the neurons to compete* for the right to respond to a given subset of inputs, such that only one output neuron is active at a time. The neuron that wins the competition is called winner-take-all neuron.

In the simplest form of competitive learning, an ANN has a single layer of output neurons, each of which is fully connected to the input nodes. The network may include feedback connections among the neurons, as indicated in Figure 7.12. In the network architecture described herein, the feedback connections perform *lateral inhibition*, with each neuron tending to inhibit the neuron to which it is laterally connected. In contrast, the feedforward synaptic connections in the network of Figure 7.12 are all *excitatory*.
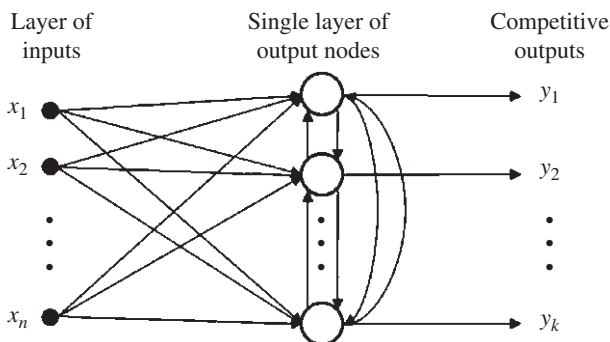

Figure 7.12. A graph of a simple competitive network architecture.

For a neuron k to be the winning neuron, its net value net_k for a specified input sample $X = \{x_1, x_2,\ldots,x_n\}$ must be the largest among all the neurons in the network. The output signal y_k of the winning neuron k is set equal to one; the outputs of all other neurons that lose the competition are set equal to zero. We thus write

$$y_k = \begin{cases} 1 & \text{if } net_k > net_j \text{ for all } j, j \neq k \\ 0 & \text{otherwise} \end{cases}$$

where the induced local value net_k represents the combined action of all the forward and feedback inputs to neuron k.

Let w_{kj} denote the synaptic weights connecting input node j to neuron k. A neuron then learns by shifting synaptic weights from its inactive input nodes to its active input nodes. If a particular neuron wins the competition, each input node of that neuron relinquishes some proportion of its synaptic weight, and the weight relinquished is then distributed among the active input nodes. According to the standard competitive-learning rule, the change Δw_{kj} applied to synaptic weight w_{kj} is defined by

$$\Delta w_{ki} = \begin{cases} \eta\,(x_j - w_{kj}) & \text{if neuron } k \text{ wins the competition} \\ 0 & \text{if neuron } k \text{ loses the competition} \end{cases}$$

where η is the learning-rate parameter. The rule has the overall effect of moving the synaptic weights of the winning neuron toward the input pattern X. We may use the geometric analogy represented in Figure 7.13 to illustrate the essence of competitive learning.

Each output neuron discovers a cluster of input samples by moving its synaptic weights to the center of gravity of the discovered cluster. Figure 7.13 illustrates the
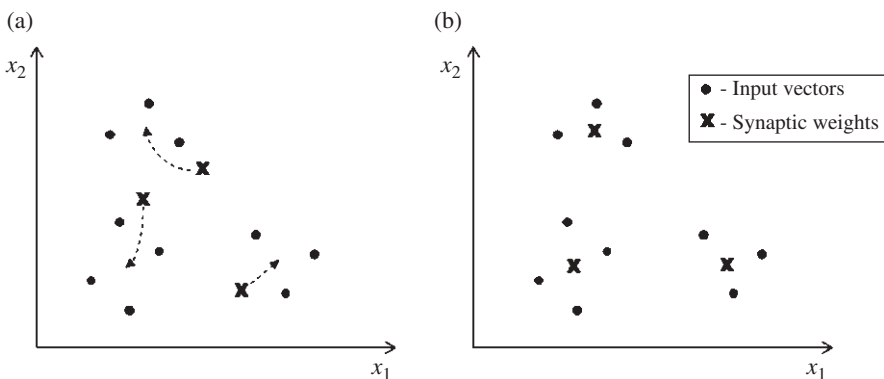

Figure 7.13. Geometric interpretation of competitive learning. (a) Initial state of the network. (b) Final state of the network.

ability of a neural network to perform clustering through competitive learning. During the competitive-learning process, similar samples are grouped by the network and represented by a single artificial neuron at the output. This grouping, based on data correlation, is done automatically. For this function to be performed in a stable way, however, the input samples must fall into sufficiently distinct groups. Otherwise, the network may be unstable.

Competitive (or winner-take-all) neural networks are often used to cluster input data where the number of output clusters is given in advance. Well-known examples of ANNs used for clustering based on unsupervised inductive learning include Kohonen's learning vector quantization (LVQ), self-organizing map (SOM), and networks based on adaptive-resonance theory models. Since the competitive network discussed in this chapter is very closely related to the Hamming networks, it is worth reviewing the key concepts associated with this general and very important class of ANNs. The Hamming network consists of two layers. The first layer is a standard feedforward layer, and it performs a correlation between the input vector and the preprocessed output vector. The second layer performs a competition to determine which of the preprocessed output vectors is closest to the input vector. The index of the second-layer neuron with a stable positive output (the winner of the competition) is the index of the prototype vector that best matches the input.

Competitive learning makes efficient adaptive classification, but it suffers from a few methodological problems. The first problem is that the choice of learning rate η forces a trade-off between speed of learning and the stability of the final weight factors. A learning rate near zero results in slow learning. Once a weight vector reaches the center of a cluster, however, it will tend to stay close to the center. In contrast, a learning rate near 1 results in fast but unstable learning. A more serious stability problem occurs when clusters are close together, which causes weight vectors also to become close, and the learning process switches its values and corresponding classes with each new example. Problems with the stability of competitive learning may occur also when a neuron's initial weight vector is located so far from any input vector that it never wins the competition, and therefore it never learns. Finally, a competitive-learning process always has as many clusters as it has output neurons. This may not be acceptable for some applications, especially when the number of clusters is not known or if it is difficult to estimate it in advance.

The following example will trace the steps in the computation and learning process of competitive networks. Suppose that there is a competitive network with three inputs and three outputs. The task is to group a set of three-dimensional (3D) input samples into three clusters. The network is fully connected; there are connections between all inputs and outputs, and there are also lateral connections between output nodes. Only local feedback weights are equal to zero, and these connections are not represented in the final architecture of the network. Output nodes are based on a linear activation function with the bias value for all nodes equal to zero. The weight factors for all connections are given in Figure 7.14, and we assume that the network is already trained with some previous samples.
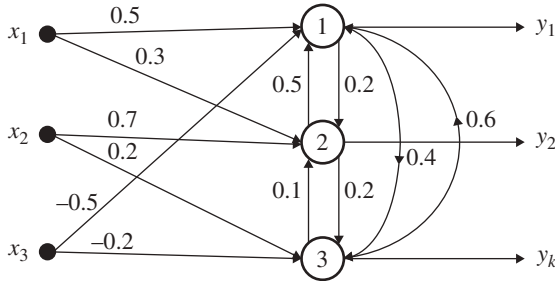
Figure 7.14. Example of a competitive neural network.

Suppose that the new sample vector X has components

$$X = \{x_1, x_2, x_3\} = \{1, 0, 1\}$$

In the first, forward phase, the temporary outputs for competition are computed through their excitatory connections, and their values are

$$net_1{}^* = 0.5 \cdot x_1 + (-0.5) \cdot x_3 = 0.5 \cdot 1 - 0.5 \cdot 1 = 0$$

$$net_2{}^* = 0.3 \cdot x_1 + 0.7 \cdot x_2 = 0.3 \cdot 1 + 0.7 \cdot 0 = 0.3$$

$$net_3{}^* = 0.2 \cdot x_2 + (-0.2) \cdot x_3 = 0.2 \cdot 0 - 0.2 \cdot 1 = -0.2$$

and after including lateral inhibitory connections:

$$net_1 = net_1{}^* + 0.5 \cdot 0.3 + 0.6 \cdot (-0.2) = 0.03$$

$$net_2 = net_2{}^* + 0.2 \cdot 0 + 0.1 \cdot (-0.2) = 0.28 \,(\text{maximum!!})$$

$$net_3 = net_3{}^* + 0.4 \cdot 0 + 0.2 \cdot 0.3 = -0.14$$

Competition between outputs shows that the highest output value is net_2, and it is the winner. So the final outputs from the network for a given sample will be

$$Y = \{y_1, y_2, y_3\} = \{0, 1, 0\}$$

Based on the same sample, in the second phase of competitive learning, the procedure for a weight factor's correction (only for the winning node y_2) starts. The results of the adaptation of the network, based on learning rate $\eta = 0.2$, are new weight factors:

$$w_{12} = 0.3 + 0.2\,(1 - 0.3) = 0.44$$

$$w_{22} = 0.7 + 0.2\,(0 - 0.7) = 0.56$$

$$w_{32} = 0.0 + 0.2\,(1 - 0.0) = 0.20$$

The other weight factors in the network remain unchanged because their output nodes were not the winners in the competition for this sample. New weights are the results of a competitive-learning process only for one sample. The process repeats iteratively for large training data sets.

7.7   SELF-ORGANIZING MAPS

Self-organizing maps (SOM), often called Kohonen maps, are a data visualization technique introduced by the University of Helsinki Professor Teuvo Kohonen, The main idea of the SOMs is to project the n-dimensional input data into some representation that could be better understood visually, for example, in a 2D image map. The SOM algorithm is a heuristic model used not only to visualize but also to explore linear and nonlinear relationships in high-dimensional data sets. SOMs were first used in the 1980s for speech recognition problems, but later they become very popular and often used methodology for variety of clustering and classification-based applications.

The problem that data visualization attempts to solve is that humans simply cannot visualize high-dimensional data, and SOM techniques are created to help us visualize and understand characteristics of this dimensional data. The SOM's output emphasizes on the salient features of the data and subsequently leads to the automatic formation of clusters of similar data items. SOMs are interpreted as unsupervised neural networks (without teacher), and they are solving clustering problem by visualization of clusters. As a result of a learning process, SOM is used as an important visualization and data-reduction aid as it gives a complete picture of the data; similar data items are transformed in lower dimension but still automatically grouped together.

The way SOMs perform dimensions reduction is by producing an output map of usually one or two dimensions. which plot the similarities of the data by grouping similar data items on the map. Through these transformations. SOMs accomplish two goals: they reduce dimensions and display similarities. Topological relationships in input samples are preserved, while complex multidimensional data can be represented in a lower-dimensional space.

The basic SOM can be visualized as a neural network whose nodes become specifically tuned to various input sample patterns or classes of patterns in an orderly fashion. Nodes with weighted connections are sometimes referred to as neurons since SOMs are actually a specific type of ANNs. SOM is represented as a single-layer feedforward network where the output neurons are arranged usually in a 2D topological grid. The output grid can be either rectangular or hexagonal. In the first case each neuron excluding borders and corners has four nearest neighbors, while in the second there are six. The hexagonal map requires more calculations, but final visualization provides more smoothed result. Attached to every output neuron, there is a weight vector with the same dimensionality as the input space. Each node i has a corresponding weight vector $w_i = \{w_{i1}, w_{i2}, \ldots, w_{id}\}$ where d is a dimension of the input feature space.
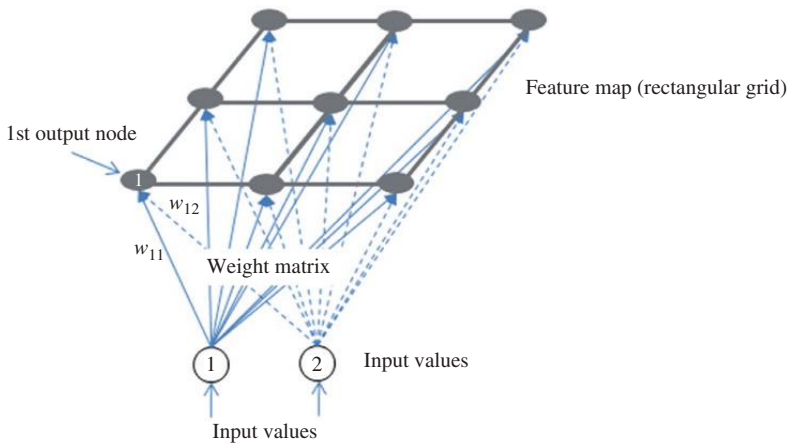
Figure 7.15. SOM with 2D input and 3 × 3 output.

The structure of the SOM outputs may be one-dimensional (1D) array or 2D matrix, but also may be more complex structures in 3D such as cylinder or toroid. Figure 7.15 shows an example of a simple SOM architecture with all interconnections between inputs and outputs.

An important part of an SOM technique is the data. These are the samples used for SOM learning. The learning process is competitive and unsupervised, meaning that no teacher is needed to define the correct output for a given input. Competitive learning is used for training the SOM, i.e. output neurons compete among themselves to share the input data samples. The winning neuron, with weights w_w, is a neuron that is the "closest" to the input example x among all other m neurons in the defined metric:

$$d(x, w_w) = \arg\min_{1 \leq j \leq m} d(x, w_j)$$

In the basic version of SOMs, only one output node (winner) at a time is activated corresponding to each input. The winner-take-all approach reduces the distance between winner's node weight vector and the current input sample, making the node closer to be "representative" for the sample. The SOM learning procedure is an iterative process that finds the parameters of the network (weights w) in order to organize the data into clusters keeping topological structure. Thus the algorithm finds an appropriate projection of high-dimensional data into a low-dimensional space.

The first step of the SOM learning is the initialization of the neurons' weights where two methods are widely used. Initial weights can be taken as

1. the coordinates of randomly selected m points from the data set (usually normalized between 0 and 1), or

2. small random values sampled evenly from the input data subspace spanned by the two largest principal component eigenvectors.

The second method can increase the speed of training but may lead to a local minima and miss some nonlinear structures in data.

The learning process is performed after initialization where training data set is submitted to the SOM one by one sample sequentially and usually in several iterations. Each output with its connections, often called a cell, is a node containing a template against which input samples are matched. All output nodes are compared with the same input sample in parallel, and SOM computes the distances between each cell and the input. All cells compete so that only the cell with the closest match between the input and its template produces an active output. Each node therefore acts like a separate decoder or pattern detector for the same input sample, and the winning node is commonly known as the best matching unit (BMU).

When the winning node is determined for a given input sample, the learning phase adapts the weight vectors in the SOM. Adaptation of the weight vectors for each output occurs through a similar process to competitive learning except that subsets of nodes are adapted at each learning step in order to produce topologically ordered maps. The "winning" node BMU aligns its own weight vector with the training input and hence becomes sensitive to it and will provide maximum response if it is shown to the network again after training. Nodes in the neighborhood set of the "winning" node must also be modified in a similar way to create regions of nodes that will respond to related samples. Nodes outside the neighborhood set remain unchanged. Figure 7.16a gives an example of 2D matrix outputs for SOM. For the given BMU the neighborhood is defined as a 3×3 matrix of nodes surrounding BMU.

Every node within the BMU's neighborhood (including the BMU) has its weight vector adjusted according to the following equation in the iterative training process:

$$w_i(t+1) = w_i(t) + h_i(t)[x(t) - w_i(t)]$$

where $h_i(t)$ is a so-called neighborhood function. It is defined as a function of time t or more precisely a training iteration, and it specifies the neighborhood area of the ith neuron. It has been found experimentally that in order to achieve global ordering of the map, the neighborhood set around the winning node should initially be large to quickly produce a rough mapping. With increased number of iterations through the training set data, the neighborhood should be reduced to force more localized adaptation of the network. This is done so the input samples can first move to an area of SOM where they will probably be, and then they will more precisely determine the position. This process is similar to coarse adjustment followed by fine-tuning (Fig. 7.17). The radius of the neighborhood of the BMU is therefore dynamic. To do this SOM can use, for example, the *exponential decay function*, which is reducing the radius dynamically with each of new iterations. The graphical interpretation of the function is given in Figure 7.16b.
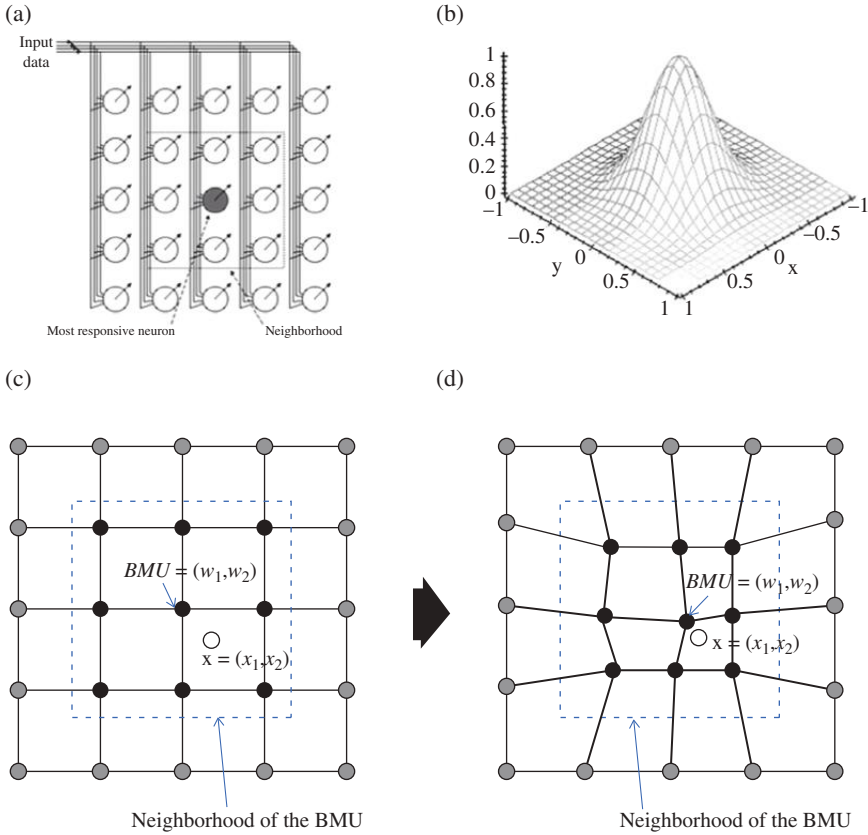
(a)

Input
data

Most responsive neuron   Neighborhood

(b)

(c)

$BMU = (w_1, w_2)$

$x = (x_1, x_2)$

Neighborhood of the BMU

(d)

$BMU = (w_1, w_2)$

$x = (x_1, x_2)$

Neighborhood of the BMU

Figure 7.16. Characteristics of SOM learning process. (a) SOM: BMU and neighborhood. (b) The radius of the neighborhood diminishes with each sample and iteration. (c) BEFORE learning, rectangular grid of SOM. (d) AFTER, learning, rectangular grid of SOM.

The simplest neighborhood function, which refers to a neighborhood set of nodes around the BMU node i, is a monotonically decreasing Gaussian function:

$$h_i(t) = a(t)\exp\left(\frac{-d(i,w)}{2\sigma^2(t)}\right)$$

where $\alpha(t)$ is a learning rate ($0 < \alpha(t) < 1$), the width of the kernel $\sigma(t)$ is a monotonically decreasing function of time as well, and t is the current time step (iteration of the loop). While the process will adapt all weight vectors within the current neighborhood region, including those of the winning neuron, those outside this neighborhood are left unchanged. The initial radius is set high, some value near the width or height of the map. As a result, at the early stage of training when the neighborhood is broad
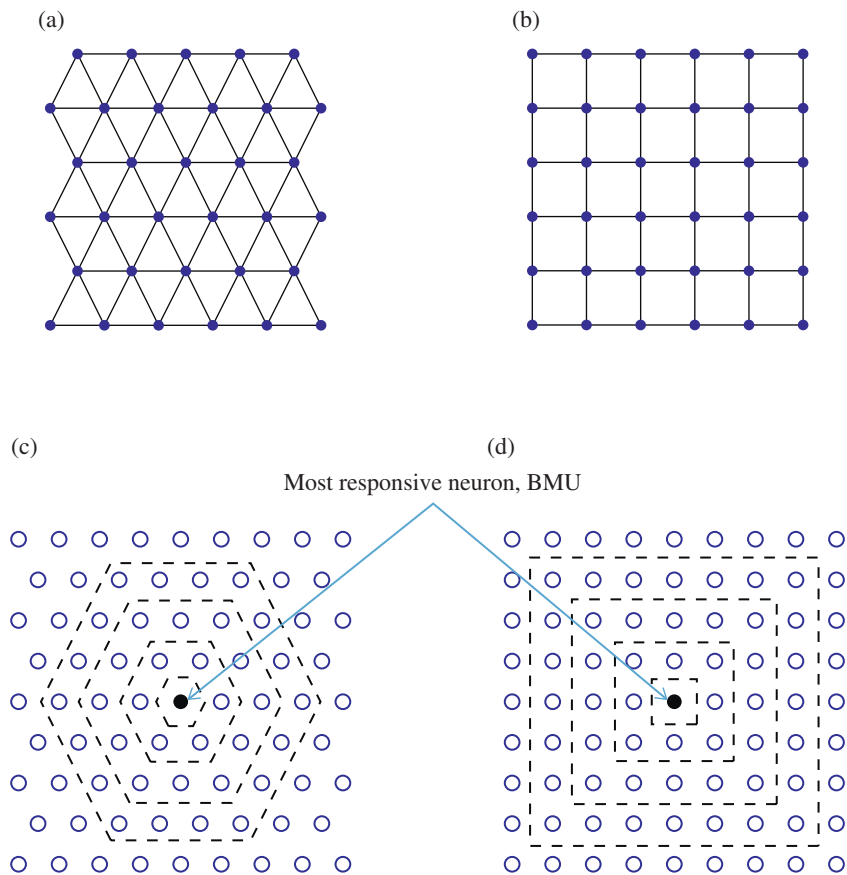
Figure 7.17. Coarse adjustment followed by fine-tuning! (a) Hexagonal grid. (b) Rectangular grid. (c) Neighborhood in a hexagonal grid. (d) Neighborhood in a retangular grid.

and covers almost all the neurons, self-organization takes place at the global scale. As the iterations continue, the base goes toward the center, so there are fewer neighbors as time progresses. At the end of training, the neighborhood shrinks to zero, and only BMU neuron updates its weights. The network will generalize through the process to organize similar vectors, which it has not previously seen, spatially close at the SOM outputs.

Apart from reducing the neighborhood, it has also been found that quicker convergence of the SOM algorithm is obtained if the adaptation rate of nodes in the network is reduced over time. Initially the adaptation rate should be high to produce coarse clustering of nodes. Once this coarse representation has been produced, however, the adaptation rate is reduced so that smaller changes to the weight vectors are

made at each node and regions of the map become fine-tuned to the input training vectors. Therefore, every node within the BMU's neighborhood including the BMU has its weight vector adjusted through the learning process. The previous equation for weight factor correction $h_i(t)$ may include an exponential decrease of "winner's influence" introducing $\alpha(t)$ also as a monotonically decreasing function.

The number of output neurons in an SOM (i.e., map size) is important to detect the deviation of the data. If the map size is too small, it might not explain some important differences that should be detected between input samples. Conversely, if the map size is too big, the differences are too small. In practical applications, if there is no additional heuristics, the number of output neurons in an SOM can be selected using iterations with different SOM architectures.

Main advantages of SOM technology are the following: presented results are very easy to understand and interpret, technology is very simple for implementation, and most importantly it works well in many practical problems. Of course, there are also some disadvantages. SOMs are computationally expensive, they are also very sensitive to measure of similarity, and finally, they are not applicable for real-world data sets with missing values. There are several possible improvements in implementations of SOMs. To reduce the number of iterations in a learning process, good initialization of weight factors is essential. *Principal components of input data* can make computation of the SOM orders of magnitude faster. Also, practical experience shows that hexagonal grids give output results with a better quality. Finally selection of distance measure is important as in any clustering algorithm. Euclidean distance is almost standard, but that does not mean that it is always the best. For an improved quality (isotropy) of the display, it is advisable to select the grid of the SOM units as *hexagonal*.

SOMs have been used in large spectrum of applications such as automatic speech recognition, clinical data analysis, monitoring of the condition of industrial plants and processes, classification from satellite images, analysis of genetic information, analysis of electrical signals from the brain, and retrieval from large document collections. Illustrative examples are given in Figure 7.18.

7.8   DEEP LEARNING

Deep learning is a subfield of machine learning that initiated enormous interests in the research community in the last few years including related fields: speech recognition, computer vision, language processing, and information retrieval. The primary reasons are some very successful and attractive applications, and they made a lot of publicity such as new solutions for self-driving cars, recommender systems for sentiment analysis, and, most recently, the successes with *AlphaGo* solution winning the best Go players in the world. Three additional important reasons and trends in computer science are supporting recent applications of deep learning: (1) the drastically increased chip processing abilities (e.g., general-purpose graphics processing unit [GPGPU]) and new computer architectures, (2) significantly increased size of data used for
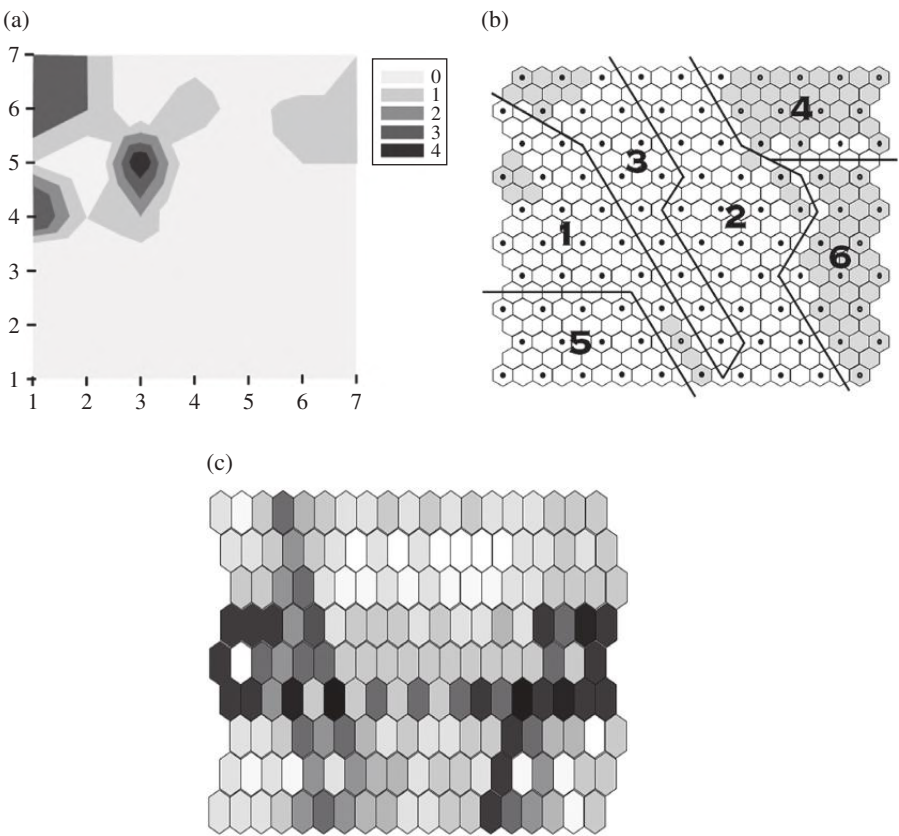
(a)

(b)

(c)

Figure 7.18. SOM applications. (a) Drugs binding to human cytochrome. (b) Interest rate classification. (c) Analysis of books buying behavior.

analyses and modeling, and (3) recent advances in data-mining and machine-learning techniques and signal/information-processing research achievements. Deep learning methodologies are becoming more mature and showing promises for even bigger advances in the future.

To explain the basic principles of deep learning, it is necessary to start with main approaches implemented in traditional machine-learning techniques. They are performing training/learning process by repeating the same steps thousands or even millions of times using available samples in iterations again and again. All these repeating activities are translated into tuning process of model parameters. Eventually, this process converges to the good enough model, applicable for many real-world applications. But, in many cases, because of user selection of not appropriate input parameters, the model gets stuck in so-called local minima, and the solution is not applicable. For decades, construction of a data-mining solution, with the core represented by

machine-learning algorithm, required careful engineering and considerable domain expertise to design a future extractor. This extractor should transform the raw data included in an input features vector (such as pixel values in an image or letters in the longer natural language text) into a suitable internal representation. Based on these new, internal, and usually more complex features, the learning subsystem could detect the best and highly applicable input–output learning model.

The performances of machine-learning methods are heavily dependent on the choice of data representation not only for selected input features but also for internal derived feature representation. For that reason, much of the actual effort in deploying machine-learning algorithms goes into the design of preprocessing pipelines and data transformation. It is expected that these processes will result in a representation of data that could support effective machine learning. Such feature engineering is very important but labor intensive, and it highlights the main weakness of current machine-learning algorithms: it is the inability of these algorithms to extract and organize automatically the discriminative information from the data. Feature engineering is a way to take all advantages of human ingenuity and prior knowledge to compensate for that weakness. Sometimes this engineering process is successful, but in many cases it does not include all complexity of inner features and their structures. Therefore, traditional machine-learning algorithms represent shallow learning approaches, usually with the depth of one or two layers. That means these algorithms have only one or two steps in input data transformation to determine the output. This class of shallow methodologies include ANNs described in the previous sections, support vector machines, decision rules, and logistic regression. The real breakthrough in deep learning was to realize that it is practical to go beyond the shallow one and two hidden layers in the network learning models, and that is opening up the exploration and practical implementations of much more expressive models.

To expand the scope and easy applicability of machine-learning techniques, it would be highly desirable to make learning algorithms less dependent on feature engineering. Deep learning is trying to solve the problem of appropriate set of input feature selection. The main idea is that the best features for the model are mostly not determined in advanced or given by some expert in the field; they should be determined through the machine-learning process. Depending on features' complexity and a level of abstraction, deep learning process allows to discover the features "naturally," through different layers of machine learning. Usually, at the input, the data set consists only of raw data, and through the learning process without any additional domain/ expert knowledge, in a layer-by-layer network, important features are discovered. Because the process of automatic feature detection in the network layers is a core of the approach, deep learning may be interpreted as a general-purpose framework for representation learning. In this context, a representation learning represents a set of methods that allows that a machine, supported by raw data, has ability to automatically discover the representations needed for prediction or classification tasks. Deep learning covers variety of learning methods with multiple levels of representation usually in a form of a multilayer network structure. These derived features are obtained by composing simple nonlinear modules that transform the representation

of data at one level into a representation at a higher, slightly more abstract level. It is important that the same deep learning architecture can be trained to perform different tasks in completely different application domains.

Deep learning is not only about finding good representative features but also about discovering a hidden structure of features based on large amount of high-dimensional raw data. The approach gives the best results when the input features are locally structured through their common spatial or temporal characteristics. Most recent examples are successful applications of deep learning in analysis of images, audio signals, or natural language processing where 1D and 2D organization of raw data is showing space structure in images or time structure in signals. For example, in the case of image processing, where input is given as an image with 1000×1000 pixels, it is possible to analyze and extract features from small segments of the image represented by neighboring 20×20 pixels. These initial and local features may be combined, in the following layers of the network, to determine more complex and more global features. Image processing is starting with raw data in the form of large number of pixels, and then the next layer defines the local features such as edges and corners, while in the following layers some more complex motifs as a combination of local features may be defined. Following deeper structure of the network, these motifs are combined into parts of the objects, and finally complete objects may be recognized. Similar hierarchy in feature discovery is applicable in a field of text recognition, where the features discovered on different layers are characters $\rightarrow$ words $\rightarrow$ phrases $\rightarrow$ clause $\rightarrow$ sentence $\rightarrow$ story. This idea of iterative feature discovery, from local characteristics toward global, is presented in Figure 7.19.

The description of the deep learning, which highlights this complex structure of discovered features, is often given in a form: "Deep learning is a sub-field of machine learning that is based on learning several levels of representations, corresponding to a hierarchy of features or concepts, where higher-level concepts are defined from lower-level ones, and the same lower-level concepts can help to define many higher-level concepts." A central idea, referred to as greedy layer-wise unsupervised pre-training, was to learn a hierarchy of features one level at a time using unsupervised feature learning to learn a new transformation at each level to be composed with the previously learned transformations. Finally, the previous set of layers with unsupervised learning, which automatically determined the best set of features describing given data
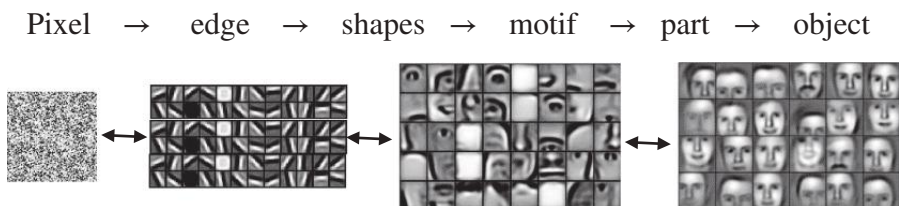
Pixel $\rightarrow$ edge $\rightarrow$ shapes $\rightarrow$ motif $\rightarrow$ part $\rightarrow$ object

Figure 7.19. Multilayer transformation of raw input features through deep learning (https://dl.acm.org/citation.cfm?id=1553453).

set, could be combined with traditional layer of supervised learning in the final phase. This architecture represents a deep learning supervised predictor or deep learning classifier, depending on what kind of final output is generated. The main advantages of deep architectures are as follows:

1. Deep architectures promote the reuse of features, which is at the heart of the theoretical advantages behind deep learning.
2. Deep architectures can lead to progressively more abstract features at higher layers of representations that are constructed in terms of less abstract ones.

One of the approaches for unsupervised feature selection at each layer of deep learning is using auto-encoder approach. Auto-encoder is the architecture that is trying to transform input samples into low-dimensional samples. But the requirement is that the set of features in transformed samples is selected as a set of generic features, which enables complete reconstruction of each sample in the training data set. Usually, auto-encoder process is applied several times through several layers of deep network. Basically, the process of unsupervised learning is trying to learn the features that describe the best what comes as a main characteristics from the previous layer.

The main idea behind the use of auto-encoders, to build richer feature sets that are by definition more compact than the input, follows the argument made earlier regarding the human brain striving to create such compact representations for efficient reasoning. Auto-encoders consist of an encoder and a decoder. This represents itself as three layers of neurons, with an input and output layer, as well as a hidden layer in the middle, as it is presented in Figure 7.20.

After an input vector x is entered into the auto-encoder, a hidden vector y is created by the hidden layer. This hidden vector represents the new encoding representation of the data based on new features. On the output layer, the hidden vector is used to attempt to reconstruct the input vector, x'. To train the auto-encoder, an error function is defined using the output and input vectors, typically using squared error. This
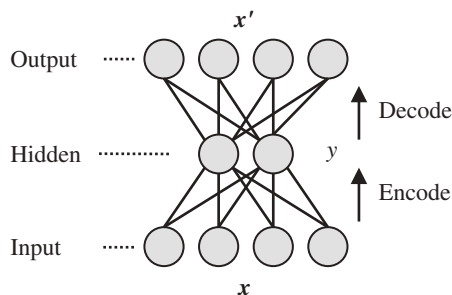

Figure 7.20. Auto-encoder consists of two main tasks: encode and decode.

concept can be extended to multiple layers, where each subsequent layer "encodes" the previous layer using significantly fewer neurons.

An auto-encoder is trained with an absolutely standard weight-adjustment algorithm to reproduce the input. By making this happen with fewer hidden nodes than the inputs, this forces the "hidden layer" units to become good feature detectors. At the same time, it satisfied a request for reduction of dimensionality of the data, important especially in a case of Big Data. For example, if the nodes' activation function in the hidden layer (with k-nodes) is linear, then the auto-encoder is essentially copying the method of PCA and mapping the variables onto the k-principle axis. However, if the activation function is nonlinear, then this allows the auto-encoder to capture much more complex multimodal behavior in the input data.

This simple reduction of nodes at each layer, along with unsupervised learning, has led to phenomenal automated feature engineering and has dramatically outperformed the past 30 years of human feature engineering in many machine-learning tasks. With multiple nonlinear layers, say, a depth of 5–20, a deep learning system can implement extremely intricate functions of its inputs that are simultaneously sensitive to very precise details. The architecture of the deep learning network, presented in Figure 7.21, consists of stacked auto-encoder with three layers. Deep network architecture in general has turned out to be very good at discovering natural intrinsic structures in high-dimensional data and is therefore applicable to many domains of science, business, and government.
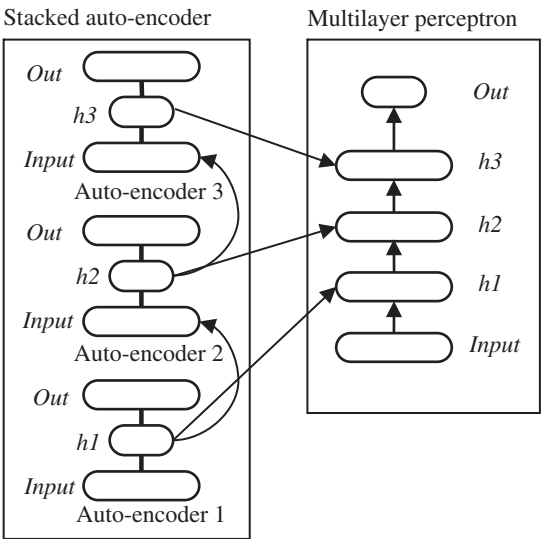

Figure 7.21. Structure of a deep network using a stacked auto-encoder.

7.9   CONVOLUTIONAL NEURAL NETWORKS (CNNs)

Convolutional neural networks (CNNs) have played an important role in the history of deep learning. CNNs are proposed in 1989 by LeCun, but main advances are realized in last few years. They are a key example of a successful application of insights obtained by studying the brain to machine-learning applications. CNNs represent some of the first deep models to perform well, long before arbitrary deep models were considered viable. Convolutional networks were also some of the first neural networks to solve important commercial applications and remain at the forefront of commercial use of deep learning today. They are designed to process very efficiently data that come in the form of multidimensional arrays: 1D for signals and sequences, including natural language; 2D for images or audio spectrograms; and 3D for video or volumetric images. There are four key ideas behind CNNs that make these applications efficient and successful: (1) local connections, (2) shared weights, (3) pooling, and (4) the use of many layers.

In traditional ANNs every output unit interacts with every input, and these connections are represented by matrix multiplication of inputs with corresponding parameters represented as weight factors in network connections. These networks had only one or two layers, but with full connectivity. On the other hand, CNNs have sparse interaction, because they are based on *local connectivity*. For example, if the input is picture with 1000×1000 pixels, in traditional ANNs, all these pixels will be connected with each node on the next layer, which means $10^6 \times 10^6 = 10^{12}$ connections if both layers have the same number of nodes. In the case of CNN, only local segments of the image will be connected with the next layer. If the local segment is 20×20 pixels, it means total of 400 connections to each of the nodes on the next layer, and that represents much smaller total connectivity. The main ideas about local connections are presented in Figure 7.22 with the simplified version of network of five inputs and five nodes on the next layer. In the case of traditional shallow ANNs that are fully connected, total number of connections is $5 \times 5 = 25$. For the CNN with local connectivity of three neighboring inputs connected to the node on the next layer, total number of connections is $3 \times 3 + 2 \times 2 = 13$ (ending nodes in the next layer have only two local connections with inputs).
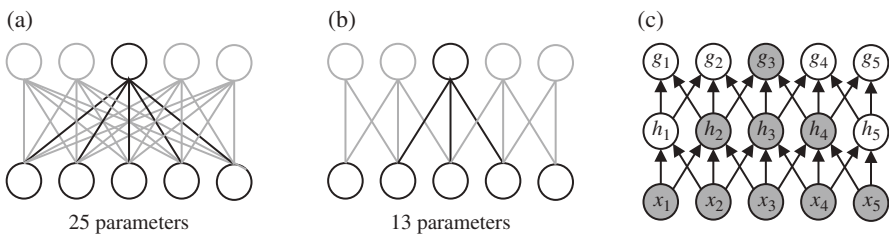

Figure 7.22. Connectivity in CNNs is sparse. (a) Global connectivity. (b) Local connectivity. (c) Multilayer connectivity.

   While connections in CNN are sparse, still units in deeper layers can be indirectly connected to almost all inputs. In Figure 7.22c three consecutive inputs are always connected to the node at the next level representing local connectivity. Going one layer deeper, middle node on the second layer is indirectly connected to all inputs in the presented network. The CNN connectivity from one layer to the next one may be sparse and often not direct. But applying multilayer networks, indirect connectivity of deeper nodes to the most of the inputs is obtained.

   Parameter sharing refers to using the same parameter for more than one function in a model. In a traditional neural net, each element of the weight matrix is used exactly once when computing the output of a layer. Every weight factor is multiplied by the corresponding input and then never revisited in the entire network. In other words, the number of parameters is equal the number of connections, or number of weight factors in the network. In a convolutional neural net, each member, which is a part of local connections set, is used at every position of the input (except perhaps some of the boundary pixels, depending on the design decisions regarding the boundary). The parameter sharing used by the convolution operation means that rather than learning a separate set of parameters for every location, we learn only one set of parameters applied to all nodes on the next layer of the network. An illustrative example is given in Figure 7.23. Instead of 13 global parameters on Figure 7.23a (13 local connections from input layer to the next layer), CNN with three-node local connectivity defines only three main parameters, namely, "*left connection*," "*central connection*," and "*right connection*," to the next layer presented in Figure 7.23b.

   Local connectivity with parameter sharing is basis for defining convolution kernels in CNNs: small local matrices that are extracting local features from the previous layer. Selection of appropriate kernels represents semiautomatic feature engineering in the CNNs. For example, when CNNs are used for image processing, specifically designed kernels may detect horizontal lines, vertical lines, small circles, and specific corners, all of them as higher-level features of the image. Simple illustrative examples of two 3×3 kernels for edge detection and sharpening the image are given in Figure 7.24. These elementary image features extracted on initial network's layers may be combined further to more complex advanced features representing specific
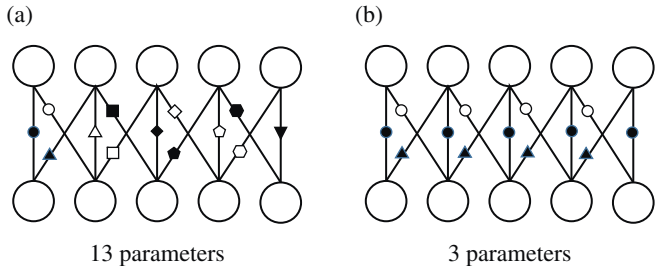

Figure 7.23. Sharing parameters of CNNs. (a) Global parameter. (b) Shared parameter.

(a)   (b)

$$\begin{vmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{vmatrix} \qquad \begin{vmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{vmatrix}$$

Figure 7.24. Simple 3×3 image kernels. (a) Edge detection kernel. (b) Sharpening the image kernel.

parts of the objects searched in the image. For example, it could be eyes, lips, nose, or other face characteristics if the analysis is looking for humans in the image.

In the next stage, CNNs are using use a *pooling function* to modify additionally the output of the previous layer. A pooling function replaces the output of the network at a certain location with a summary statistic of the nearby outputs. Pooling over spatial regions produces invariance to translation, which means that if we translate the input by a small amount, the values of most of the pooled outputs do not change. Invariance to local translation can be a very useful property because the network cares more about whether some feature is present than exactly where it is. For example, when determining whether an image contains a face, CNN does not need to know the location of the eyes with pixel-perfect accuracy, but the system just need to determine that there is an eye on the left side of the face and the other eye on the right side of the face.

A pooling layer takes each feature from the convolutional layer and prepares a condensed feature set as a new output. For instance, each unit in the pooling layer may summarize a region of $n \times n$ nodes in the previous layer. As a concrete example, one common procedure for pooling is known as *max pooling*. In max pooling, a pooling unit simply outputs the maximum activation in the specified input region. Two simple examples of max pooling, for three-node local connectivity, are given in Figure 7.25.

Max pooling may be interpreted as a way for the network to ask whether a given feature is found anywhere in a region of the image. This approach throws away the exact positional information. The intuition is that once a feature has been found, its exact location is not as important; only maybe its rough location is useful relative to other features of a given sample. A big benefit of pooling in general is that there are fewer pooled features, and this helps in reducing the total number of parameters in the following layers.

The final architecture of the CNN consists of several layers, often more than 10, where convolution and pooling operations and corresponding layers are repeating iteratively one after the other. Many software packages today are including CNNs as the standard deep learning methodology such as GoogLeNet, VGGNet, or ResNet. CNNs are almost standard models today for every image-related analysis and recognition problems. It is also successfully applied to recommender systems, natural language processing, and more. CNN is also computationally very efficient architecture. Using convolution and pooling operations together with parameter sharing, this architecture enables CNN models to run on any device, making them universally attractive. CNNs
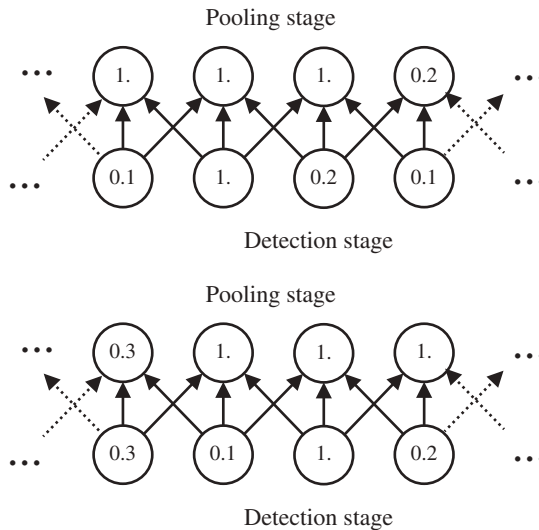
Figure 7.25. Max-pooling layer in CNNs.

are only one of often used deep architectures, and there are several others developed recently such as deep belief networks, deep Boltzmann machines, and so on. All of them are able to handle and decode complex data structures that have ability to work with multiple nonlinear features.

7.10 REVIEW QUESTIONS AND PROBLEMS

1. Explain the fundamental differences between the design of an artificial neural network and "classical" information-processing systems.

2. Why is the fault-tolerant property one of the most important characteristics and capabilities of artificial neural networks?

3. What are the basic components of the neuron's model?

4. Why are continuous functions such as log-sigmoid or hyperbolic tangent common activation functions in real-world applications of artificial neural networks?

5. Discuss the differences between feedforward and recurrent neural networks.

6. Given a two-input neuron with the following parameters, namely, bias $b = 1.2$, weight factors $W = [w_1, w_2] = [3, 2]$, and input vector $X = [-5, 6]^T$, calculate the neuron's output for the following activation functions.
   (a) A symmetrical hard limit.
   (b) A log-sigmoid.
   (c) A hyperbolic tangent.

7. Consider a two-input neuron with the following weight factors W and input vector X:

$$W = [3,2] \quad X = [-5,7]^T$$

We would like to have an output of 0.5.

(a) Is there a transfer function from Table 9.1 that will do the job if the bias is zero?
(b) Is there a bias that will do the job if the linear-transfer function is used?
(c) What is the bias that will do the job with a log-sigmoid activation function?

8. Consider a classification problem defined with the set of three-dimensional samples X, where two dimensions are inputs and the third one is the output.

X:	I_1	I_2	O
	−1	1	1
	0	0	1
	1	−1	1
	1	0	0
	0	1	0

(a) Draw a graph of the data points X labeled according to their classes. Is the problem of classification solvable with a single-neuron perceptron? Explain the answer.
(b) Draw a diagram of the perceptron you would use to solve the problem. Define the initial values for all network parameters.
(c) Apply single iteration of the delta-learning algorithm. What is the final vector of weight factors?

9. The one-neuron network is trained to classify input–output samples.

| I_1	I_2	O
1	0	1
1	1	−1
0	1	1

Show that this problem cannot be solved unless the network uses a bias.

10. Consider the classification problem based on the set of samples X.

X:	I_1	I_2	O
	−1	1	1
	−1	−1	1
	0	0	0
	1	0	0

(a) Draw a graph of the data points labeled according to their classification. Is the problem solvable with one artificial neuron? If yes, graph the decision boundaries.

(b) Design a single-neuron perceptron to solve this problem. Determine the final weight factors as a weight vector orthogonal to the decision boundary.

(c) Test your solution with all four samples.

(d) Using your network classify the following samples: (–2, 0), (1, 1), (0, 1), and (–1, –2).

(e) Which of the samples in (d) will always be classified the same way, and for which samples classification may vary depending on the solution?

11. Implement the program that performs the computation (and learning) of a single-layer perceptron.

12. For the given competitive network:
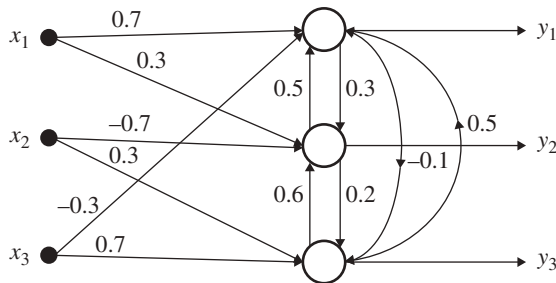
(a) Find the output vector [Y1, Y2, Y3] if the input sample is [X1, X2, X3] = [1, –1, –1].

(b) What are the new weight factors in the network?

13. Search the Web to find the basic characteristics of publicly available or commercial software tools that are based on artificial neural networks. Document the results of your search. Which of them are for learning with a teacher, and which are support learning without a teacher?

14. For a neural network, which one of these structural assumptions is the one that most affects the trade-off between underfitting (i.e., a high bias model) and overfitting (i.e. a high variance model):

(a) The number of hidden nodes.

(b) The learning rate.

(c) The initial choice of weights.

(d) The use of a constant-term unit input.

15. Is it true that the VC dimension of a perceptron is smaller than the VC dimension of a simple linear SVM? Discuss your answer.

16. Which type of artificial neural-network architecture does not contain a hidden layer? Why?

(a) Backpropagation.

(b) Perceptron.

(c) Self-organizing map.

(d) Convolutional networks.

(e) Several previous types.

7.11   REFERENCES FOR FURTHER STUDY

1. Haykin, S., *Neural Networks and Learning Machines*, 3rd edition, Prentice Hall, Upper Saddle River, NJ, 2009.

   Fluid and authoritative, this well-organized book represents the first comprehensive treatment of neural networks from an engineering perspective, providing extensive state-of-the-art coverage that will expose readers to the myriad facets of neural networks and help them appreciate the technology's origin, capabilities, and potential applications. The book examines all the important aspects of this emerging technology, covering the learning process, backpropagation, radial basis functions, recurrent networks, self-organizing systems, modular networks, temporal processing, neurodynamics, and VLSI implementation. This also integrates computer experiments throughout to demonstrate how neural networks are designed and perform in practice. Chapter objectives, problems, worked examples, a bibliography, photographs, illustrations, and a thorough glossary all reinforce concepts throughout. New chapters delve into such areas as support vector machines, and reinforcement learning/neurodynamic programming, plus readers will find an entire chapter of case studies to illustrate the real-life practical applications of neural networks. A highly detailed bibliography is included for easy reference. It is the book for professional engineers and research scientists.

2. Heaton J., *Introduction to Neural Network with Java*, Heaton Research, St. Louis, 2005.

   Introduction to Neural Networks with Java introduces the Java programmer to the world of neural networks and artificial intelligence (AI). Neural-network architectures such as the feedforward backpropagation, Hopfield, and Kohonen networks are discussed. Additional AI topics such as genetic algorithms and simulated annealing, are also introduced. Practical examples are given for each neural network. Examples include the traveling salesman problem, handwriting recognition, fuzzy logic, and learning mathematical functions. All Java source code can be downloaded online. In addition to showing the programmer how to construct these neural networks, the book discusses the Java Object Oriented Neural Engine (JOONE). JOONE is a free open-source Java neural engine.

3. Principe J. C., R. Mikkulainen, *Advances in Self-Organizing Maps*, Series: Lecture Notes in Computer Science, Vol. 5629, Springer, 2009.

   This book constitutes the refereed proceedings of the seventh International Workshop on Advances in Self-Organizing Maps, WSOM 2009, held in St. Augustine, Florida, in June 2009. The 41 revised full papers presented were carefully reviewed and selected from numerous submissions. The papers deal with topics in the use of SOM in many areas of social sciences, economics, computational biology, engineering, time-series analysis, data visualization, and theoretical computer science.

4. Goodfellow I., Y. Bengio, A. Courville, *Deep Learning*, MIT Press, Cambridge, MA, 2016.

   Deep learning is a form of machine learning that enables computers to learn from experience and understand the world in terms of a hierarchy of concepts. Because the computer gathers knowledge from experience, there is no need for a human computer operator to formally specify all the knowledge that the computer needs. The hierarchy of concepts allows the computer to learn complicated concepts by building them out of simpler ones; a graph of these hierarchies would be many layers deep. This book introduces a broad range of topics in deep learning. The text offers mathematical and conceptual background, covering relevant concepts in linear algebra, probability theory and information theory, numerical computation, and machine learning. It describes deep learning techniques used by practitioners in industry, including deep feedforward networks, regularization, optimization algorithms, convolutional networks, sequence modeling, and practical methodology; and it surveys such applications as natural language processing, speech recognition, computer vision, online recommendation systems, bioinformatics, and video games. Finally, the book offers research perspectives, covering such theoretical topics as linear factor models, auto-encoders, representation learning, structured probabilistic models, Monte Carlo methods, the partition function, approximate inference, and deep generative models.

5. Fandango A., *Mastering TensorFlow 1.x: Advanced machine learning and deep learning concepts using TensorFlow 1.x and Keras*, Packt Publishing, 2018.

   TensorFlow is the most popular numerical computation library built from the ground up for distributed, cloud, and mobile environments. This represents the data as tensors and the computation as graphs. This book is a comprehensive guide that lets you explore the advanced features of TensorFlow 1.x. Gain insight into TensorFlow Core, Keras, TF Estimators, TFLearn, TF Slim, Pretty Tensor, and Sonnet. Leverage the power of TensorFlow and Keras to build deep learning models using concepts such as transfer learning, generative adversarial networks, and deep reinforcement learning. Throughout the book, you will obtain hands-on experience with varied data sets, such as MNIST, CIFAR-10, PTB, text8, and COCO-Images.

8

ENSEMBLE LEARNING

Chapter Objectives

- Explain a basic characteristics of ensemble learning methodologies.
- Distinguish between different implementations of combination schemes for different learners.
- Compare bagging and boosting approaches.
- Explain main characteristics of random forest algorithm.
- Introduce AdaBoost algorithm and its advantages.

One of primary goals of data mining is to predict an "unknown" value of a new sample from observed samples. Such a prediction is achieved by two sequential phases as shown in Figure 8.1:

(a) Training phase—Producing a predictive model from training samples using one of available supervised learning algorithms.
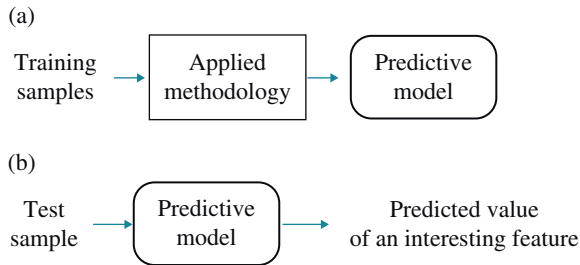
(a)

Training → Applied → Predictive
samples   methodology   model

(b)

Test → Predictive → Predicted value
sample   model   of an interesting feature

Figure 8.1. (a) Training phase and (b) testing phase for a predictive model.

(b) Testing phase—Evaluating the generated predictive model using test sam-
ples that are not used in the training phase.

Numerous applications of a data-mining process showed validity of so-called
"no-free-lunch theorem." It states that there is no single learning algorithm that is
the best and most accurate in all applications. Each algorithm determines a certain
model that comes with a set of assumptions. Sometimes these assumptions hold,
sometimes not, and therefore no single algorithm "wins" all the time!

In order to improve accuracy of a predictive model, the promising approach
called the ensemble learning is introduced. The idea is to combine results from various
predictive models generated using the training samples. Key motivation behind the
proposed approach is to reduce the error rate. An initial assumption is that it will
become much more unlikely that the ensemble will misclassify a new sample compar-
ing with a single predictive model. When combing multiple, independent, and diverse
"decision-makers," each of which is at least more accurate than random guessing, cor-
rect decisions should be reinforced. The idea may be demonstrated with some simple
decision process where single human performances are compared with human ensem-
bles. For example, given the question "How many jelly beans is in the jar?," group
average will outperform individual estimates or in TV series "Who Wants to be a Mil-
lionaire?" where audience (ensemble) vote is support for the candidate who is not sure
in the answer.

This idea is proven theoretically by Hansen and group through the statement: If N
classifiers make *independent* errors and they have the error probability $e < 0.5$, then it
can be shown that the error of an ensemble E is monotonically decreasing function of
N. Clearly, performances quickly decrease for dependent classifiers.

8.1 ENSEMBLE LEARNING METHODOLOGIES

The ensemble learning methodology consists also of two sequential phases (1) train-
ing phase and (2) testing phase. However, in the training phase, the ensemble method
generates several different predictive models from training samples as it is presented
in Figure 8.2a. For predicting an unknown value of a test sample, the ensemble
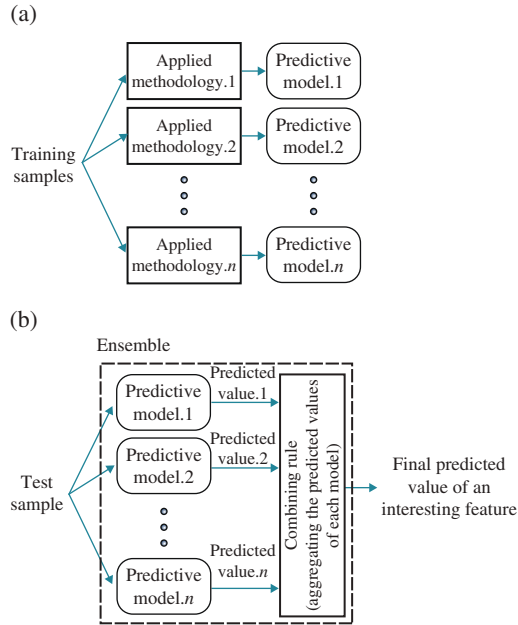
Figure 8.2. (a) Training phase and (b) testing phase for building an ensemble.

method aggregates outputs of each predictive model (Fig. 8.2b). An integrated predictive model generated by an ensemble approach consists of several predictive models (predictive model.1, predictive model.2,…, predictive model.n) and a combining rule as shown in Figure 8.2b. We will refer to such a predictive model as an *ensemble*. The field of ensemble learning is still relatively new, and several names are used as synonyms depending also which predictive task is performed including: combination of multiple classifiers, classifier fusion, mixture of experts, or consensus aggregation.

To perform better than a single predictive model, an ensemble should consist of predictive models that are independent of each other, i.e. their errors are uncorrelated, and each of them has accuracy >0.5. The outcome of each predictive model is aggregated to determine the output value of a test sample. We may analyze all steps of ensemble prediction for a classification task. For example, we may analyze a classification task where the ensemble consists of 15 classifiers, each of which classifies test samples into one of two categorical values. The ensemble decides the categorical value based on dominant frequency of classifiers' outputs. If 15 predictive models are different from each other, and each model has the identical error rate ($\varepsilon = 0.3$), the ensemble will make a wrong prediction only if more than half of the predictive models misclassify a test sample. Therefore, the error rate of the ensemble is

$$\varepsilon_{\text{ensemble}} = \sum_{i=8}^{15} \binom{15}{i} \varepsilon^i (1-\varepsilon)^{15-i} = 0.05$$

which is considerably lower than the 0.3 error rate of a single classifier. The sum is starting with 8, and it means that 8 or more models misclassified a test sample, while 7 or fewer models classified the sample correctly.

Figure 8.3a shows error rates of an ensemble, which consists of 15 predictive models ($n = 15$). The x-axis represents an error rate (ε) of a single classifier. The diagonal line represents the case in which all models in the ensemble are identical. The solid line represents error rates of an ensemble in which predictive models are different and independent from each other. An ensemble has a significantly lower error rate
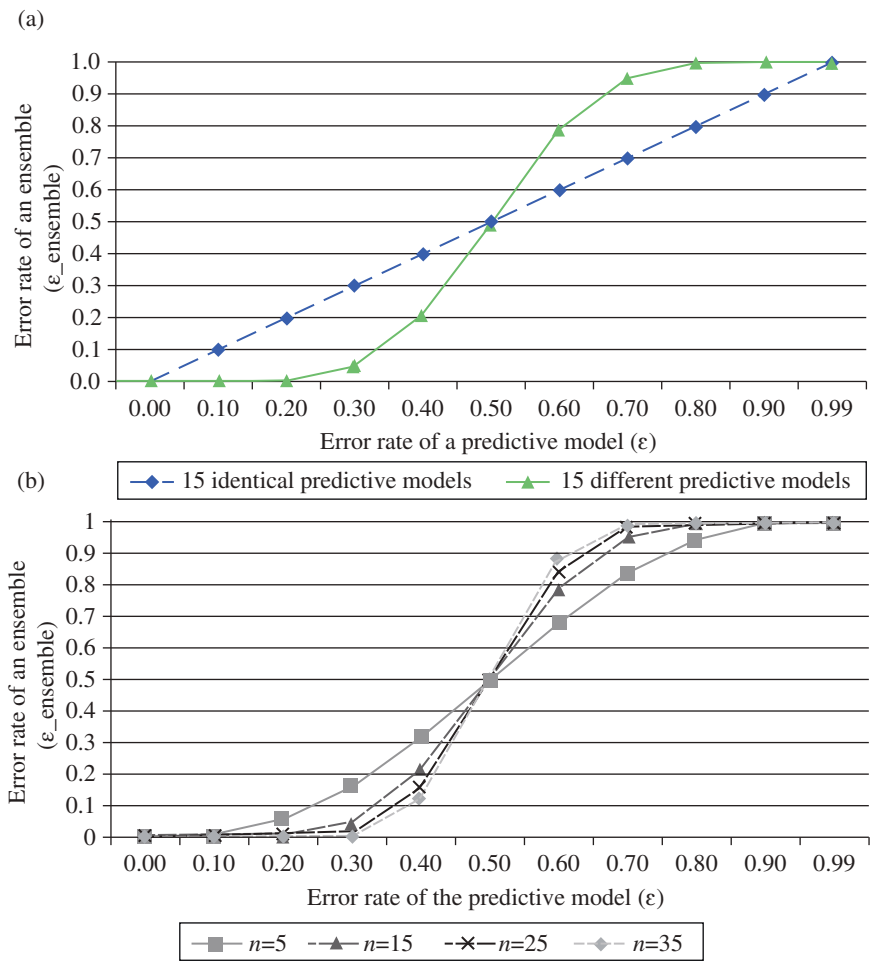
(a)

(b)

Figure 8.3. Changes in error rates of an ensemble. (a) Identical predictive model vs. different predictive models in an ensemble. (b) The different number of predictive models in an ensemble.

than a single predictive model only when an error rate (ε) of the members of the ensemble is lower than 0.5.

We can also analyze the effect of the number of predictive models in an ensemble. Figure 8.3b shows error-rate curves for ensembles that consist of 5, 15, 25, and 35 predictive models, respectively. Observe that when an error rate of a predictive model is lower than 0.5, the larger the number of predictive models is, the lower an error rate of an ensemble is. For example, when each predictive model of an ensemble has error rate of 0.4, error rates of each ensemble ($n = 5$, $n = 15$, $n = 25$, and $n = 35$) is calculated as 0.317, 0.213, 0.153, and 0.114, respectively. However, this decrease in the error rate for an ensemble is becoming less significant if the number of classifiers is very large or when error rate of each classifier becomes relatively small.

The basic questions in creating an ensemble learner are: How to generate base learners, and how to combine the outputs from base learners? Diverse and independent learners can be generated by:

(a) Using different learning algorithms for different learning models such as support vector machines, decision trees, and neural networks.

(b) Using different hyperparameters in the same algorithm to tune different models (for example, different number of hidden nodes in ANNs).

(c) Using different input representations, such as using different subsets of input features in a data set.

(d) Using different training subsets of input data to generate different models usually using the same learning methodology.

Stacked generalization (or stacking) is a methodology that could be classified in the first group (a). Unlike other well-known techniques, stacking may be (and normally is) used to combine models of different types. One way of combining multiple models is specific by introducing the concept of a meta-learner. The learning procedure is as follows:

1. Split the training set into two disjoint sets.
2. Train several base learners on the first part.
3. Test the base learners on the second part.
4. Using the predictions from (3) as the inputs, and the correct responses as the outputs, train a higher-level learner.

Note that steps (1)–(3) are the same as cross-validation, but instead of using a winner-takes-all approach, the base learners are combined, possibly nonlinearly. Although an attractive idea, it is less theoretically analyzed and less widely used than bagging and boosting, two most recognized ensemble learning methodologies. Similar situation is with second group of methodologies (b): although very simple approach, it is not used or analyzed intensively. Maybe the main reason is that applying the same methodology with different parameters does not guarantee independence of models.

Class (c) methodologies are based on manual or automatic feature selection/
extraction that can be used for generating diverse classifiers using different feature
sets. For example, subsets related to different sensors or subsets of features computed
with different algorithms may be used. To form training data sets, different subsets of
input features are chosen, and then each training sample with the selected input fea-
tures becomes an element of training data sets. In Figure 8.4, there are five training
samples {S1, S2, S3, S4, S5} with four features {F1, F2, F3, F4}. When the training
data set 1 is generated, three features {F1, F2, F4} is randomly selected from input
features {F1, F2, F3, F4}, and all training samples with those features form the first
training set. Similar process is performed for the other training sets. The main require-
ment is that classifiers use different subsets of features that are complementary.

The *random subspace method* (RSM) is a relatively recent method of ensemble
learning, which is based on theory of stochastic discrimination. Learning machines are
trained on randomly chosen subspaces of the original input space, and the outputs of
the models are then combined. Illustrative example for movies classification is given
in Figure 8.5. RSM works well for large feature sets with redundant features. Random
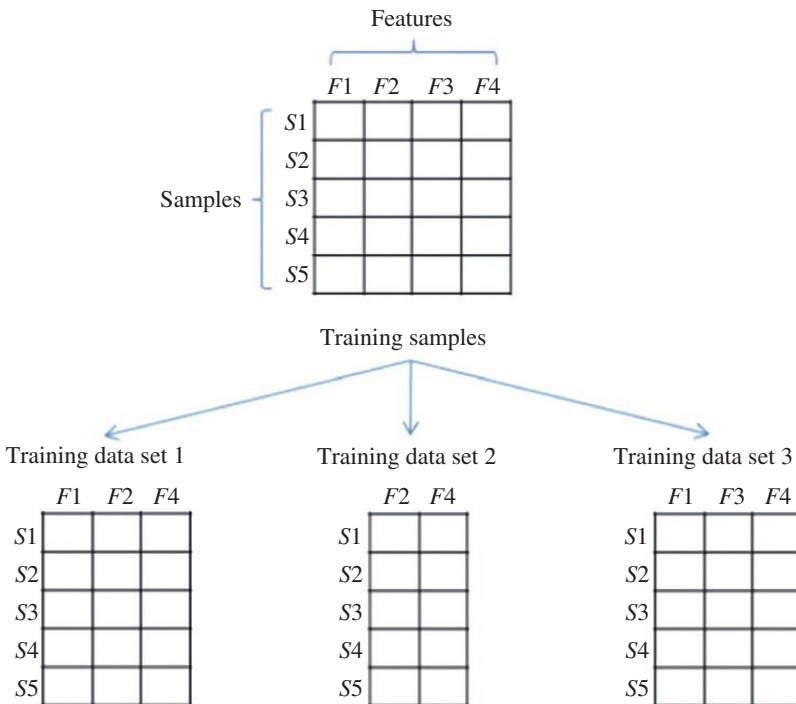forest methodology, which not only utilizes such an approach but also has some

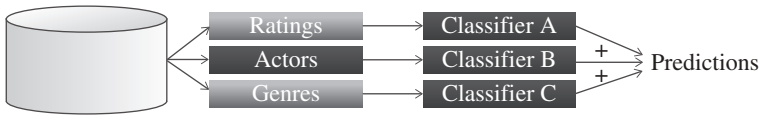**Figure 8.4.** Features' selection for ensemble classifiers methodology.

Figure 8.5. RSM approach in ensemble classifier for movie classification.

components of the bagging schema, is implemented in many commercial data-mining tools, and it is very often used in real-world data-mining applications.

Methodologies based on different training subsets of input samples (d) are the most popular approaches in ensemble learning, and corresponding techniques such as bagging and boosting are widely applied in different tools. But, before detailed explanations of these techniques, it is necessary to explain one additional and final step in ensemble learning, and that is combining of outcomes for different learners.

8.2   COMBINATION SCHEMES FOR MULTIPLE LEARNERS

Combination schemes include the following:

- *Global approach* is through learners' fusion where all learners produce an output, and these outputs are combined by voting, averaging, or stacking. This represents integration (fusion) functions where for each pattern, all the classifiers contribute to the final decision.
- *Local approach* is based on learner selection where one or more learners responsible for generating the output are selected based on their closeness to the sample. Selection function is applied where for each pattern, just one classifier, or a subset, is responsible for the final decision.
- *Multistage combination* uses a serial approach where the next learner is trained with or tested on only instances where previous learners were inaccurate.

Voting is the simplest way of combining classifiers on a global level and representing the result as a linear combination of outputs d_j for n learners:

$$y_i = \sum_{j=1}^{n} w_j d_j \quad \text{where } w_j \geq 0 \quad \text{and} \quad \sum_{j=1}^{n} w_j = 1$$

Result of combination could be different depending on w_j. Alternatives for combinations are *simple sum (*equal weights*)*, *weighted sum*, *median*, *minimum*, *maximum*, and *product of d_{ij}*. Voting schemes can be seen as approximations under a Bayesian framework where weights w_j approximate prior model probabilities.

Rank-level fusion method is applied for some classifiers, which provide class "scores," or some sort of class probabilities. In general, if $\Omega = \{c_1,...,c_k\}$ is the set

of classes, each of these classifiers can provide an "ordered" (ranked) list of class labels. For example, if probabilities of output classes are 0.10, 0.75, and 0.20, corresponding ranks for the classes will be 1, 3, and 2, respectively. The highest rank is given to the class with the highest probability. Let us check an example, where the number of classifiers is $N = 3$ and the number of classes $k = 4$, $\Omega = \{a, b, c, d\}$. For a given sample, the ranked outputs of the three classifiers are as follows:

Rank value	Classifier 1	Classifier 2	Classifier 3
4	c	a	b
3	b	b	a
2	d	d	c
1	a	c	d

In this case, final selection of the output class will be determined by accumulation of scores for each class:

$$r_a = r_a{}^{(1)} + r_a{}^{(2)} + r_a{}^{(3)} = 1 + 4 + 3 = 8$$
$$r_b = r_b{}^{(1)} + r_b{}^{(2)} + r_b{}^{(3)} = 3 + 3 + 4 = 10$$
$$r_c = r_c{}^{(1)} + r_c{}^{(2)} + r_c{}^{(3)} = 4 + 1 + 2 = 7$$
$$r_d = r_d{}^{(1)} + r_d{}^{(2)} + r_d{}^{(3)} = 2 + 3 + 1 = 5$$

The winner class is b because it has the maximum overall rank.

Additional methodology in combining the results of multiple learners, *dynamic classifier selection* (DCS) algorithm, which is representing a local approach, assumes the following steps:

1. Find the *k*-nearest training samples to the test input.
2. Look at the accuracies of the base classifiers on these samples.
3. Choose one (or top *N*) classifiers that best performs on these samples.
4. Combine decisions for selected classifiers.

8.3 BAGGING AND BOOSTING

Bagging and boosting are well-known procedures with solid theoretical background. They belong to the class (d) of ensemble methodologies, and essentially they are based on resampling of a training data set.

Bagging, a name derived from bootstrap aggregation, was the first effective method of ensemble learning and is one of the simplest methods. It was originally designed for classification and is usually applied to decision tree models, but it can be used with any type of model for classification or regression. The method uses

multiple versions of a training set by using the bootstrap, i.e. sampling with replace-
ment. Each of these data sets is used to train a different model. The outputs of the
models are combined by averaging (in the case of regression) or voting (in the case
of classification) to create a single output.

In the bagging methodology a training data set for a predictive model consists of
samples taken with replacement from initial set of samples according to a sampling
distribution. The sampling distribution determines how likely it is that a sample will
be selected. For example, when the sampling distribution is predefined as the uniform
distribution, all N training samples have the same probability, $1/N$, of being selected.
In the same training data set, because of replacement sampling, some training samples
may appear multiple times, while any training samples may not appear even once. In
Figure 8.6, there are five training samples $\{S1, S2, S3, S4, S5\}$ with four features $\{F1,
F2, F3, F4\}$. Suppose that three training data sets are formed by samples that are ran-
domly selected with replacement from the training samples according to the uniform
distribution. Each training sample has 1/5 probability of being selected as an element
of a training data set. In the training data set 1, $S2$ and $S4$ appear twice, while $S1$ and $S3$
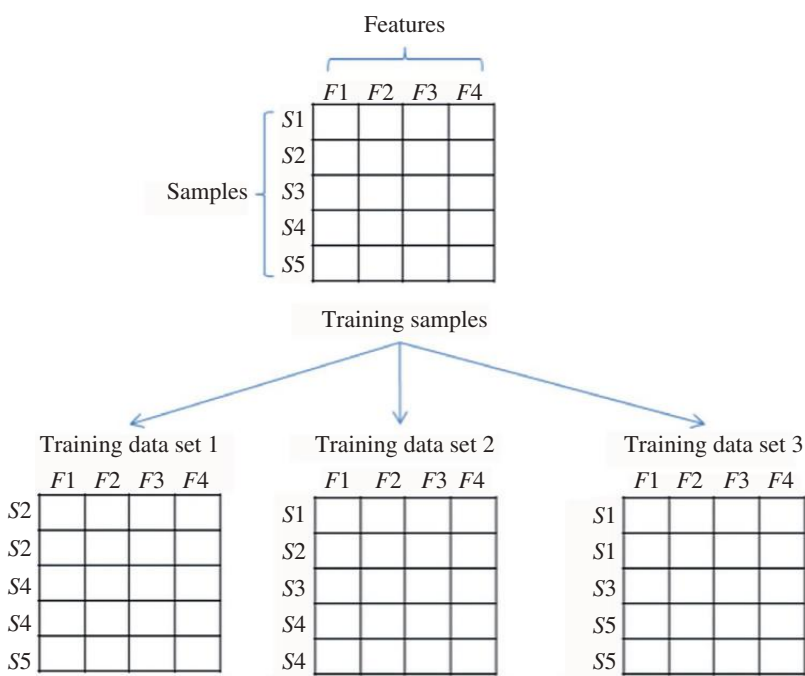do not appear.

Figure 8.6. Bagging methodology distributes samples taken with replacement from initial
set of samples.

Bagging is only effective when using unstable nonlinear models where small changes in training data lead to significantly different classifiers and large changes in accuracy. It decreases error by decreasing the variance in the results of unstable learners.

Random forests differ in two ways from this general bagging scheme. First, *Random forest* is an ensemble of classifiers, which uses a large number of individual, unpruned decision trees in the ensemble. Second, it uses a modified tree learning algorithm that selects, at each candidate split in the learning process, a random subset of the features. This process is sometimes called "feature bagging". The reason for doing this is the correlation of the trees in an ordinary bootstrap sample. If one or a few features are very strong predictors for the output class, these features will be selected in many of the trees, causing them to become correlated. The *random forest* algorithm brings extra randomness into the model, when it is growing the trees. Instead of searching for the best feature while splitting a node, it searches for the best feature among a random subset of features. This process creates a wide diversity, which generally results in a better model. Typically, for a classification problem with p features in data set, $\sqrt{p}$ (rounded down) features are used in each split. For regression problems the recommendation is $p/3$ (rounded down) selected features with a minimum node size of five samples as the default.

Boosting is the most widely used ensemble method and one of the most powerful learning ideas introduced in ensemble learning community. Originally designed for classification, it can also be extended to regression. The algorithm first creates a "weak" classifier, that is, it suffices that its accuracy on the training set is slightly better than random guessing. Samples are given initial weights, and usually it starts with uniform weighting. For the following iterations, the samples are reweighted to focus the system on samples that are not correctly classified with recently learned classifier. During each step of learning, (1) increase weights of the samples that are not correctly learned by the weak learner, and (2) decrease weights of the samples that are correctly learned by the weak learner. Final classification is based on weighted vote of weak classifiers generated in iterations.

8.4 ADABOOST

The original boosting algorithm combined three weak learners to generate a strong, high quality learner. *AdaBoost*, short for "adaptive boosting," is the most popular boosting algorithm. *AdaBoost* combine "weak" learners into a highly accurate classifier to solve difficult highly nonlinear problems. Instead of sampling as in a bagging approach, *AdaBoost* reweigh samples! It uses the same training set over and over again (thus it need not be large), and it may keep adding weak learners until target training error is reached (Fig. 8.7).

Given a training data set $\{(x_1, y_1),\dots,(x_m, y_m)\}$ where $x_i \in X$ and $y_i \in \{-1, +1\}$. When a weak classifier is trained with the data, for each input sample x_i, the classifier will give classification $h(x_i)$ (where $h(x_i) \in \{-1, +1\}$). With these assumptions the main steps of *AdaBoost* algorithm are presented in Figure 8.8.
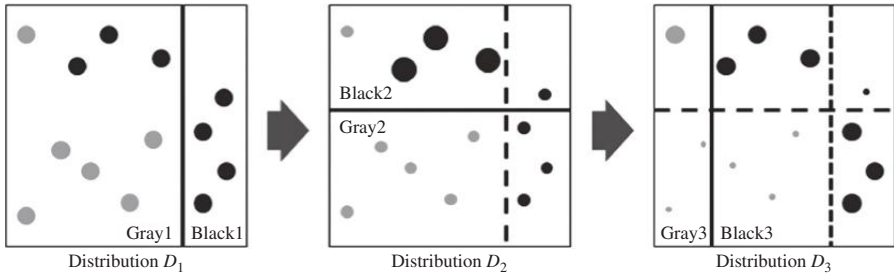
Figure 8.7. AdaBoost iterations.

- Initialize distribution over the training set $D_1(i)=1/m$
- For $t=1,...,T$:
    1. Train *weak learner* using distribution D_t.
    2. Choose a weight (or confidence value) $\alpha_t \in R$.
    3. Update the distribution over the training set:

    $$D_{t+1}(i) = \frac{D_t(i)e^{-\alpha_t y_i h_t(x_i)}}{Z_t} \qquad (2)$$

    Where Z_t is a normalization factor chosen so that D_{t+1} will be a distribution
- Final vote $H(x)$ is a weighted sum:

$$H(x)=\text{sign}(f(x))=\text{sign}\left(\sum_{t=1}^{T}\alpha_t h_t(x)\right)$$

Figure 8.8. AdaBoost algorithm.

Simplicity and easy to implement are the main reasons why *AdaBoost* is so popular. It can be combined with any classifiers including neural networks, decision trees, or nearest neighbor classifiers. The algorithm requires almost no parameters to tune and still is very effective even for the most complex classification problems, but at the same time it could be sensitive to noise and outliers.

Ensemble learning approach showed all advantages in one very famous application, Netflix \$1 million competition. The Netflix Prize required to substantially improve the accuracy of predictions about how much someone is going to love a movie based on their previous movies' preferences. Users' rating for movies was 1–5 stars, and therefore the problem was classification task with five classes. Most of top ranked competitors have used some variation of ensemble learning, showing its advantages in practice. Top competitor, *BellKor* team, explains ideas behind its success: "Our final solution consists of blending 107 individual predictors. Predictive accuracy is substantially improved when blending multiple predictors. Our experience is that most efforts should be concentrated in deriving substantially different approaches, rather than refining a single technique. Consequently, our solution is an ensemble of many methods."

–	No progress prize candidates yet	–	–
Progress prize -RMSE <= 0.8625			
1	Bellkor	0.8705	8.50
Progress prize 2007 - RMSE = 0.8712 - Winning team: korbell			
2	Korbell	0.8712	8.43
3	When gravity and dinosaurs unite	0.8717	8.38
4	Gravity	0.8743	8.10
5	Basho	0.8746	8.07

Figure 8.9. Top competitors in 2007/2008 for Netflix award.

8.5   REVIEW QUESTIONS AND PROBLEMS

1. Explain the basic idea of the ensemble learning, and discuss why the ensemble mechanism is able to improve prediction accuracy of a model.

2. Designing an ensemble model, there are factors that directly affect accuracy of the ensemble. Explain those factors and approaches to each of them.

3. Bagging and boosting approach are very famous ensemble approach. Both of them generate a single predictive model from each different training set. Discuss the differences between bagging and boosting approach, and explain advantages and disadvantages of each of them.

4. Propose the efficient boosting approach for a large data set.

5. In the bagging methodology, a subset is formed by samples that are randomly selected with replacement from training samples. On average, a subset contains approximately what percentage of training samples?

6. In Figure 8.7, draw a picture of the next distribution D_4.

7. In Eq. (2) of the AdaBoost algorithm (Fig. 8.8), $e^{\alpha_t y_i h_t(x_i)}$ replaces the term of $e^{-\alpha_t y_i h_t(x_i)}$. Explain how this change influences the AdaBoost algorithm and the reason.

8. Consider the following data set, where there are 10 samples with 1 dimension and 2 classes:
   Training samples:

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}
f_1	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
Class	1	1	1	−1	1	−1	1	−1	−1	−1

1. Determine ALL of the best one-level binary decision trees?
   (For example, IF $f_1 \leq 0.35$ THEN Class is 1, and IF $f_1 > 0.35$ THEN Class is -1. The accuracy of that tree is 80%.)

2. We have the following five training data sets randomly selected from the above training samples. Apply the bagging procedure using those training data sets.
   (A) Construct the best one-level binary decision tree from each training data set.
   (B) Predict the training samples using each constructed one-level binary decision tree.
   (C) Combine outputs predicted by each decision tree using voting method
   (D) What accuracy does the bagging provide?

Training data set 1:

	x_1	x_2	x_3	x_4	x_5	x_8	x_9	x_{10}	x_{10}	x_{10}
f_1	0.1	0.2	0.3	0.4	0.5	0.8	0.9	1.0	1.0	1.0
Class	1	1	1	-1	1	-1	-1	-1	-1	-1

Training data set 2:

	x_1	x_1	x_2	x_4	x_4	x_5	x_5	x_7	x_8	x_9
f_1	0.1	0.1	0.2	0.4	0.4	0.5	0.5	0.7	0.8	0.9
Class	1	1	1	-1	-1	1	1	1	-1	-1

Training data set 3:

	x_2	x_4	x_5	x_6	x_7	x_7	x_7	x_8	x_9	x_{10}
f_1	0.2	0.4	0.5	0.6	0.7	0.7	0.7	0.8	0.9	1.0
Class	1	-1	1	-1	1	1	1	-1	-1	-1

Training data set 4:

	x_1	x_2	x_5	x_5	x_5	x_7	x_7	x_8	x_9	x_{10}
f_1	0.1	0.2	0.5	0.5	0.5	0.7	0.7	0.8	0.9	1.0
Class	1	1	1	1	1	1	1	-1	-1	-1

Training data set 5:

	x_1	x_1	x_1	x_1	x_3	x_3	x_8	x_8	x_9	x_9
f_1	0.1	0.1	0.1	0.1	0.3	0.3	0.8	0.8	0.9	0.9
Class	1	1	1	-1	1	-1	1	-1	-1	-1

3. Applying the AdaBoost algorithm (Fig. 8.8) to the above training samples, we generate the following initial one-level binary decision tree from that samples:

IF $f_1 \leq 0.35$ THEN Class is 1
IF $f_1 > 0.35$ THEN Class is -1

To generate the next decision tree, what is the probability (D_2 in Fig. 8.8) that each sample is selected to the training data set? (α_t is defined as an accuracy rate of the initial decision tree on training samples)

9. For classifying a new sample into four classes, $C1$, $C2$, $C3$, and $C4$, we have an ensemble, which consists of three different classifiers: *Classifier 1*, *Classifiers 2*, and *Classifier 3*. Each of them has 0.9, 0.6, and 0.6 accuracy rate on training samples, respectively. When the new sample, X, is given, the outputs of the three classifiers are as follows:

Class label	Classifier 1	Classifier 2	Classifier 3
C1	0.9	0.3	0.0
C2	0.0	0.4	0.9
C3	0.1	0.2	0.0
C4	0.0	0.1	0.1

Each number in the above table describes the probability that a classifier predicts the class of a new sample as a corresponding class. For example, the probability that *Classifier* 1 predicts the class of X as $C1$ is 0.9.

When the ensemble combines predictions of each of them, as a combination method,

(a) If the *simple sum* is used, which class is X classified as and why?
(b) If the *weight sum* is used, which class is X classified as and why?
(c) If the *rank-level fusion* is used, which class is X classified as and why?

10. Suppose you have a drug discovery data set, which has 1,950 samples and 100,000 features. You must classify chemical compounds represented by structural molecular features as active or inactive using ensemble learning. In order to generate diverse and independent classifiers for an ensemble, which ensemble methodology do you choose? Explain the reason for selecting that methodology?

11. Which of the following is a fundamental difference between bagging and boosting?

(a) Bagging is used for supervised learning. Boosting is used with unsupervised clustering.
(b) Bagging gives varying weights to training instances. Boosting gives equal weight to all training instances.
(c) Bagging does not take the performance of previously built models into account when building a new model. With boosting each new model is built based upon the results of previous models.

    (d) With boosting, each model has an equal weight in the classification of new instances. With bagging, individual models are given varying weights.

12. Ensemble of classifiers contain 11 independent models, all of which have an error rate of 0.2. (Hint: Understanding how to use the binomial distribution will be useful in answering this question.)

    (a) What is the total error rate of the ensemble?
    (b) If the error rate is 0.49, what is the total error rate of the ensemble?

8.6 REFERENCES FOR FURTHER STUDY

1. Thomas G. Dieterich, Ensemble Methods in Machine Learning,in *Lecture Notes in Computer Science on Multiple Classifier Systems*, Vol. 1857, Springer, Berlin/Heidelberg, 2000.

   Ensemble methods are learning algorithms that construct a set of classifiers and then classify new data points by taking a (weighted) vote of their predictions. The original ensemble method is Bayesian averaging, but more recent algorithms include error-correcting output coding, bagging, and boosting. This paper reviews these methods and explains why ensembles can often perform better than any single classifier. Some previous studies comparing ensemble methods are reviewed, and some new experiments are presented to uncover the reasons that AdaBoost does not overfit rapidly.

2. Gavin Brown, Ensemble Learning, in: *Encyclopedia of Machine Learning*, C. Sammut & G.I. Webb (Eds.), Springer Press, Berlin, 2010.

   Ensemble learning refers to the procedures employed to train multiple learning machines and combine their outputs, treating them as a "committee" of decision-makers. The principle is that the committee decision, with individual predictions combined appropriately, should have better overall accuracy, on average, than any individual committee member. Numerous empirical and theoretical studies have demonstrated that ensemble models very often attain higher accuracy than single models. Ensemble methods constitute some of the most robust and accurate learning algorithms of the past decade. A multitude of heuristics has been developed for randomizing the ensemble parameters, to generate diverse models. It is arguable that this line of investigation is nowadays rather oversubscribed, and the more interesting research is now in methods for nonstandard data.

3. Kuncheva L. I., *Combining Pattern Classifiers: Methods and Algorithms*, Wiley Press, New Jersey, 2004.

   Covering pattern classification methods, *Combining Classifiers: Ideas and Methods* focuses on the important and widely studied issue of how to combine several classifiers together in order to achieve improved recognition performance. It is one of the first books to provide unified, coherent, and expansive coverage of the topic and as such will be welcomed by those involved in the area. With case studies that

bring the text alive and demonstrate "real-world" applications, it is destined to become essential reading.

4. Gérard Biau G., E. Scornet, A Random Forest Guided Tour, arXiv:1511.05741, November 2015.

The random forest algorithm, proposed by L. Breiman in 2001, has been extremely successful as a general-purpose classification and regression method. The approach, which combines several randomized decision trees and aggregates their predictions by averaging, has shown excellent performance in settings where the number of variables is much larger than the number of observations. Moreover, it is versatile enough to be applied to large-scale problems, is easily adapted to various ad hoc learning tasks, and returns measures of variable importance. The present article reviews the most recent theoretical and methodological developments for random forests. Emphasis is placed on the mathematical forces driving the algorithm, with special attention given to the selection of parameters, the resampling mechanism, and variable importance measures. This review is intended to provide nonexperts easy access to the main ideas.

9

CLUSTER ANALYSIS

Chapter Objectives

- Distinguish between different representations of clusters and different measures of similarities.
- Compare basic characteristics of agglomerative- and partitional-clustering algorithms.
- Implement agglomerative algorithms using single-link or complete-link measures of similarity.
- Derive the K-means method for partitional clustering and analysis of its complexity.
- Explain the implementation of incremental-clustering algorithms and its advantages and disadvantages.
- Introduce concepts of density clustering and algorithms DBSCAN and BIRCH.
- Discuss why validation of clustering results is difficult problem.

Cluster analysis is a set of methodologies for automatic classification of samples into a number of groups using a measure of association so that the samples in one group are similar and samples belonging to different groups are not similar. The input for a system of cluster analysis is a set of samples and a measure of similarity (or dissimilarity) between two samples. The output from cluster analysis is a number of groups (clusters) that form a partition, or a structure of partitions, of the data set. One additional result of cluster analysis is a generalized description of every cluster, and this is especially important for a deeper analysis of the data set's characteristics.

9.1 CLUSTERING CONCEPTS

Organizing data into sensible groupings is one of the most fundamental approaches of understanding and learning. Cluster analysis is the formal study of methods and algorithms for natural grouping, or clustering, of objects according to measured or perceived intrinsic characteristics or similarity. Samples for clustering are represented as a vector of measurements, or more formally as a point in a multidimensional space. Samples within a valid cluster are more similar to each other than they are to a sample belonging to a different cluster. Clustering methodology is particularly appropriate for the exploration of interrelationships among samples to make a preliminary assessment of the sample structure. Human's performances are competitive with automatic-clustering procedures in one, two, or three dimensions, but most real problems involve clustering in higher dimensions. It is very difficult for humans to intuitively interpret data embedded in a high-dimensional space.

Table 9.1 shows a simple example of clustering information for nine customers, distributed across three clusters. Two features describe customers: the first feature is the number of items the customers bought, and the second feature shows the price they paid for each.

TABLE 9.1. Sample Set of Clusters Consisting of Similar Objects

	# of Items	Price
Cluster 1	2	1700
	3	2000
	4	2300
Cluster 2	10	1800
	12	2100
	11	2500
Cluster 3	2	100
	3	200
	3	350

Customers in Cluster 1 purchase few high-priced items, customers in Cluster 2 purchase many high-priced items, and customers in Cluster 3 purchase few low-priced items. Even this simple example and interpretation of a cluster's characteristics shows that clustering analysis (in some references also called unsupervised classification) refers to situations in which the objective is to construct decision boundaries (classification surfaces) based on unlabeled training data set. The samples in these data sets have only input dimensions, and the learning process is classified as unsupervised.

Clustering is a very difficult problem because data can reveal clusters with different shapes and sizes in an n-dimensional data space. To compound the problem further, the number of clusters in the data often depends on the resolution (fine vs. coarse) with which we view the data. The next example illustrates these problems through the process of clustering points in the Euclidean two-dimensional (2D) space. Figure 9.1a shows a set of points (samples in a 2D space) scattered on a 2D plane. Let us analyze the problem of dividing the points into a number of groups. The number of groups N is not given beforehand. Figure 9.1b shows the natural clusters G_1, G_2, and G_3 bordered by broken curves. Since the number of clusters is not given, we have another partition of four clusters in Figure 9.1c that is as natural as the groups in Figure 9.1b. This kind of arbitrariness for the number of clusters is a major problem in clustering.

Note that the above clusters can be recognized by sight. For a set of points in a higher-dimensional Euclidean space, we cannot recognize clusters visually. Accordingly, we need an objective criterion for clustering. To describe this criterion, we have to introduce a more formalized approach in describing the basic concepts and the clustering process.

An input to a cluster analysis can be described as an ordered pair (X, s), or (X, d), where X is a set of objects' descriptions represented with samples and s and d are measures for similarity or dissimilarity (distance) between samples, respectively. Output from the clustering system is a partition $\Lambda = \{G_1, G_2,...,G_N\}$ where $G_k, k = 1,...,N$ is a crisp subset of X such that
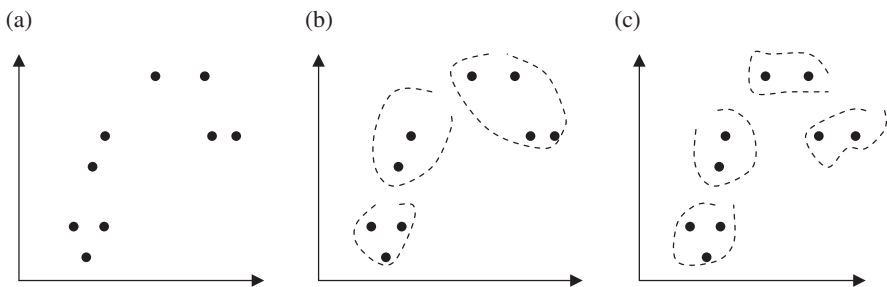

Figure 9.1. Cluster analysis of points in a 2D space. (a) Initial data. (b) Three clusters of data. (c) Four clusters of data.

$$G_1 \cup G_2 \cup \cdots \cup G_N = X, \quad \text{and}$$

$$G_i \cap G_j = \emptyset, \quad i \neq j$$

The members G_1, G_2,...,G_N of Λ are called clusters. Every cluster may be described with some characteristics. In discovery-based clustering, both the cluster (a separate set of points in X) and its descriptions or characterizations are generated as a result of a clustering procedure. There are several schemata for a formal description of discovered clusters:

1. Represent a cluster of points in an n-dimensional space (samples) by their centroid or by a set of distant (border) points in a cluster.
2. Represent a cluster graphically using nodes in a clustering tree.
3. Represent clusters by using logical expression on sample attributes.

Figure 9.2 illustrates these ideas. Using the centroid to represent a cluster is the most popular schema. It works well when the clusters are compact or isotropic. When the clusters are elongated or non-isotropic, however, this schema fails to represent them properly.

The availability of a vast collection of clustering algorithms in the literature and also in different software environments can easily confound a user attempting to select an approach suitable for the problem at hand. It is important to mention that there is no clustering technique that is universally applicable in uncovering the variety of structures present in multidimensional data sets. The user's understanding of the problem and the corresponding data types will be the best criteria to select the appropriate method. Most clustering algorithms are based on the following two popular approaches:

1. Hierarchical clustering.
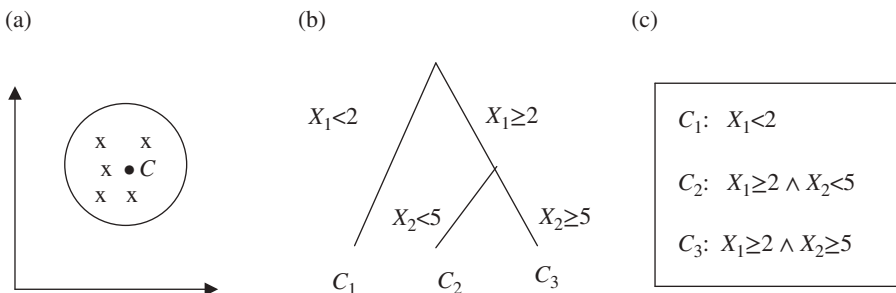2. Iterative square-error partitional clustering.

Figure 9.2. Different schemata for cluster representation. (a) Centroid. (b) Clustering tree. (c) Logical expressions.

Hierarchical techniques organize data in a nested sequence of groups, which can be displayed in the form of a dendrogram or a tree structure. Square-error partitional algorithms attempt to obtain that partition that minimizes the within-cluster scatter or maximizes the between-cluster scatter. These methods are nonhierarchical because all resulting clusters are groups of samples at the same level of partition. To guarantee that an optimum solution has been obtained, one has to examine all possible partitions of the N samples with n dimensions into K clusters (for a given K), but that retrieval process is not computationally feasible. Notice that the number of all possible partitions of a set of N objects into K clusters is given by

$$\frac{1}{K!}\sum_{j=1}^{K} \left({}^{K}j\right) j^{N}$$

So various heuristics are used to reduce the search space, but then there is no guarantee that the optimal solution will be found.

Hierarchical methods that produce a nested series of partitions are explained in Section 9.3, while partitional methods that produce only one level of data grouping are given with more details in Section 9.4. The next section introduces different measures of similarity between samples; these measures are the core component of every clustering algorithm.

9.2 SIMILARITY MEASURES

To formalize the concept of a similarity measure, the following terms and notation are used throughout this chapter. A sample x (or feature vector, observation) is a single data vector used by the clustering algorithm in a space of samples X. In many other texts, the term pattern is used. We do not use this term because of a collision in meaning with patterns as in pattern-association analysis, where the term has a totally different meaning. Most data samples for clustering take the form of finite-dimensional vectors, and it is unnecessary to distinguish between an object or a sample x_i and the corresponding vector. Accordingly, we assume that each sample $x_i \in X$, $i = 1,\ldots,n$, is represented by a vector $x_i = \{x_{i1}, x_{i2},\ldots, x_{im}\}$. The value m is the number of dimensions (features) of samples, while n is the total number of samples prepared for a clustering process that belongs to the sample domain X.

A sample can describe either a physical object (a chair) or an abstract object (a style of writing). Samples, represented conventionally as multidimensional vectors, have each dimension as a single feature. These features can be either quantitative or qualitative descriptions of the object. If the individual scalar component x_{ij} of a sample x_i is a feature or attribute value, then each component $x_{ij}, j = 1,\ldots,m$, is an element of a domain P_j, where P_j could belong to different types of data such as binary ($P_j = \{0,1\}$), integer ($P_j \subseteq Z$), real number ($P_j \subseteq R$), or a categorical set of symbols. In the last case, for example, P_j may be a set of colors: $P_j = \{$white, black, red, blue,

green}. If weight and color are two features used to describe samples, then the sample (20, black) is the representation of a black object with 20 units of weight. The first feature is quantitative and the second one is qualitative. In general, both feature types can be further subdivided, and details of this taxonomy are already given in Chapter 1.

Quantitative features can be subdivided as:

1. *continuous values* (e.g., real numbers where $P_j \subseteq R$),
2. *discrete values* (e.g., binary numbers $P_j = \{0,1\}$ or integers $P_j \subseteq Z$), and
3. *interval values* (e.g., $P_j = \{x_{ij} \leq 20,\ 20 < x_{ij} < 40,\ x_{ij} \geq 40\}$.

Qualitative features can be:

1. *nominal or unordered* (e.g., color is "blue" or "red") and
2. *ordinal* (e.g., military rank with values "general," "colonel," etc.).

Since similarity is fundamental to the definition of a cluster, a measure of the similarity between two patterns drawn from the same feature space is essential to most clustering algorithms. This measure must be chosen very carefully because the quality of a clustering process depends on this decision. It is most common to calculate, instead of the similarity measure, the dissimilarity between two samples using a distance measure defined on the feature space. A distance measure may be a metric or a quasi-metric on the sample space, and it is used to quantify the dissimilarity of samples.

The word "similarity" in clustering means that the value of $s(x, x')$ is large when x and x' are two similar samples; the value of $s(x, x')$ is small when x and x' are not similar. Moreover, a similarity measure s is symmetric:

$$s(x, x') = s(x', x), \quad \forall x, x' \in X$$

For most clustering techniques, we say that a similarity measure is normalized:

$$0 \leq s(x, x') \leq 1, \quad \forall x, x' \in X$$

Very often a measure of dissimilarity is used instead of a similarity measure. A dissimilarity measure is denoted by $d(x, x')$, $\forall x, x' \in X$. Dissimilarity is frequently called a distance. A distance $d(x, x')$ is small when x and x' are similar; if x and x' are not similar, $d(x, x')$ is large. We assume without loss of generality that

$$d(x, x') \geq 0, \quad \forall x, x' \in X$$

Distance measure is also symmetric:

$$d(x, x') = d(x', x), \quad \forall x, x' \in X$$

and if it is accepted as a *metric distance measure*, then a triangular inequality is required:

$$d(x,x'') \le d(x,x'') + d(x',x''), \quad \forall x,x',x'' \in X$$

The most well-known metric distance measure is the Euclidean distance in an *m*-dimensional feature space:

$$d_2(x_i,x_j) = \left(\sum_{k=1}^{m} (x_{ik} - x_{jk})^2 \right)^{1/2}$$

Another metric that is frequently used is called the L_1 metric or city block distance:

$$d_1(x_i,x_j) = \sum_{k=1}^{m} |x_{ik} - x_{jk}|$$

and finally, the Minkowski metric includes the Euclidean distance and the city block distance as special cases:

$$d_p(x_i,x_j) = \left(\sum_{k=1}^{m} (x_{ik} - x_{jk})^p \right)^{1/p}$$

It is obvious that when $p = 1$, then d coincides with L_1 distance and when $p = 2$, d is identical with the Euclidean metric. For example, for four-dimensional vectors $x_1 = \{1, 0, 1, 0\}$ and $x_2 = \{2, 1, -3, -1\}$, these distance measures are $d_1 = 1 + 1 + 4 + 1 =$, $d_2 = (1 + 1 + 16 + 1)^{1/2} = 4.36$, and $d_3 = (1 + 1 + 64 + 1)^{1/3} = 4.06$.

The Euclidean *n*-dimensional space model offers not only the Euclidean distance but also other measures of similarity. One of them is called the cosine correlation:

$$s_{\cos}(x_i,x_j) = \frac{\left[\sum_{k=1}^{m} (x_{ik} \cdot x_{jk}) \right]}{\left[\sum_{k=1}^{m} x_{ik}^2 \cdot \sum_{k=1}^{m} x_{jk}^2 \right]^{1/2}}$$

It is easy to see that

$$s_{\cos}(x_i,x_j) = 1 \Leftrightarrow \forall i, j \text{ and } \lambda > 0 \text{ where } x_i = \lambda \cdot x_j$$

$$s_{\cos}(x_i,x_j) = -1 \Leftrightarrow \forall i, j \text{ and } \lambda < 0 \text{ where } x_i = \lambda \cdot x_j$$

For the previously given vectors x_1 and x_2, the corresponding cosine measure of similarity is $s_{\cos}(x_1, x_2) = (2 + 0 - 3 + 0)/(2^{1/2} \cdot 15^{1/2}) = -0.18$.

TABLE 9.2. The 2 × 2 Contingency Table

		x_j	
	1	1	0
x_j	1	a	b
	0	c	d

Computing distances or measures of similarity between samples that have some or all features that are noncontinuous is problematic, since the different types of features are not comparable and one standard measure is not applicable. In practice, different distance measures are used for different features of heterogeneous samples. Let us explain one possible distance measure for binary data. Assume that each sample is represented by the n-dimensional vector x_i, which has components with binary values ($v_{ij} \in \{0,1\}$). A conventional method for obtaining a distance measure between two samples x_i and x_j represented with binary features is to use the 2 × 2 contingency table for samples x_i and x_j, as shown in Table 9.2.

The meaning of the table parameters a, b, c, and d, which are given in Table 9.2, is as follows:

1. a is the number of binary attributes of samples x_i and x_j such that $x_{ik} = x_{jk} = 1$.
2. b is the number of binary attributes of samples x_i and x_j such that $x_{ik} = 1$ and $x_{jk} = 0$.
3. c is the number of binary attributes of samples x_i and x_j such that $x_{ik} = 0$ and $x_{jk} = 1$.
4. d is the number of binary attributes of samples x_i and x_j such that $x_{ik} = x_{jk} = 0$.

For example, if x_i and x_j are eight-dimensional vectors with binary feature values

$$x_i = \{0,0,1,1,0,1,0,1\}$$
$$x_j = \{0,1,1,0,0,1,0,0\}$$

then the values of the parameters a to d in the contingency table are

$$a = 2, \quad b = 2, \quad c = 1, \quad \text{and} \quad d = 3.$$

Several similarity measures for samples with binary features are proposed using the values in the 2 × 2 contingency table. Some of them are:

1. Simple matching coefficient (SMC)

$$s_{smc}(x_i, x_j) = \frac{(a+d)}{(a+b+c+d)}$$

2. Jaccard's coefficient

$$s_{\mathrm{jc}}\left(x_i, x_j\right) = \frac{a}{(a+b+c)}$$

3. Rao's coefficient

$$s_{\mathrm{rc}}\left(x_i, x_j\right) = \frac{a}{(a+b+c+d)}$$

For the previously given eight-dimensional samples x_i and x_j, these measures of similarity will be $s_{\mathrm{smc}}(x_i, x_j) = 5/8$, $s_{\mathrm{jc}}(x_i, x_j) = 2/5$, and $s_{\mathrm{rc}}(x_i, x_j) = 2/8$.

How to measure distances between values when categorical data is not binary? The simplest way to find similarity between two categorical attributes is to assign a similarity of 1 if the values are identical and a similarity of 0 if the values are not identical. For two multivariate categorical data points, the similarity between them will be directly proportional to the number of attributes in which they match. This simple measure is also known as the *overlap measure* in the literature. One obvious drawback of the overlap measure is that it does not distinguish between the different values taken by an attribute. All matches, as well as mismatches, are treated as equal.

This observation has motivated researchers to come up with data-driven similarity measures for categorical attributes. Such measures take into account the frequency distribution of different attribute values in a given data set to define similarity between two categorical attribute values. Intuitively, the use of additional information would lead to better performance. There are two main characteristics of categorical data that are included in new measures of similarity (distance):

1. Number of values taken by each attribute, n_k. One attribute might take several hundred possible values, while another attribute might take very few values.
2. Distribution $f_k(x)$, which refers to the distribution of frequency of values taken by an attribute in the given data set.

Almost all similarity measures assign a similarity value between two d-dimensional samples X and Y belonging to the data set D as follows:

$$S(X,Y) = \sum_{k=1}^{d} w_k S_k(X_k, Y_k)$$

where $S_k(X_k, Y_k)$ is the per-attribute similarity between two values for the categorical attribute A_k. The quantity w_k denotes the weight assigned to the attribute A_k. To understand how different measures calculate the per-attribute similarity, $S_k(X_k; Y_k)$, consider a categorical attribute A_k, which takes one of the values in the set $\{a, b, c, d\}$. The per-attribute similarity computation is equivalent to constructing the (symmetric) matrix shown in Table 9.3.

TABLE 9.3. Similarity Matrix for a Single Categorical
Attribute

	a	b	c	d
a	S(a,a)	S(a,b)	S(a,c)	S(a,d)
b		S(b,b)	S(b,c)	S(b,d)
c			S(c,c)	S(c,d)
d				S(d,d)

TABLE 9.4. Goodall3 Similarity Measure for Categorical
Attributes

Measure	$S_k(X_k, Y_k)$	$w_k, k = 1,\ldots,d$
Goodall3	$1 - p_k{}^2(X_k)$ if $X_k = Y_k$	$1/d$
	0 otherwise	

Essentially, in determining the similarity between two values, any categorical measure is filling the entries of this matrix. For example, the overlap measure sets the diagonal entries to 1 and the off-diagonal entries to 0, i.e., the similarity is 1 if the values match and 0 if the values mismatch. Additionally, measures may use the following information in computing a similarity value (all the measures in this paper use only this information):

- $f(a)$, $f(b)$, $f(c)$, and $f(d)$, the frequencies of the values in the data set.
- N, the size of the data set.
- n, the number of values taken by the attribute (4 in the case above).

We will present, as an illustrative example, only one additional measure of similarity for categorical data, *Goodall3*, because it shows good performances in average for variety of experiments with different data sets. That does not mean that some other measures such as Eskin, Lin, Smirnov, or Burnaby will not be more appropriate for a specific data set. *The Goodall3 measure*, given in Table 9.4, assigns a high similarity if the matching values are infrequent regardless of the frequencies of the other values. The range of $S_k(X_k; Y_k)$ for matches in the *Goodall3* measure is $[0, 1 - 2/N(N-1)]$, with the minimum value being attained if X_k is the only value for attribute A_k and maximum value being attained if X_k occurs only twice.

There are some advanced distance measures applicable to categorical, but also to numerical data, that take into account the effect of the surrounding or neighboring

points in the n-dimensional spaces of samples. These surrounding points are called contexts. The similarity between two points, x_i and x_j, with the given context, is measured using the *mutual neighbor distance* (MND), which is defined as

$$\mathrm{MND}\left(x_i, x_j\right) = \mathrm{NN}\left(x_i, x_j\right) + \mathrm{NN}\left(x_j, x_i\right)$$

where NN(x_i, x_j) is the neighbor number of x_j with respect to x_i. If x_i is the closest point to x_j, then NN(x_i, x_j) is equal to 1, if it is the second closest point, NN(x_i, x_j) is equal to 2, etc. Figures 9.3 and 9.4 give an example of the computation and basic characteristics of the MND measure.

Points in Figures 9.3 and 9.4, denoted by *A, B, C, D, E*, and *F*, are 2D samples with features x_1 and x_2. In Figure 9.3, the nearest neighbor of *A* is *B* using Euclidean distance, and *B*'s nearest neighbor is *A*. So,

$$\mathrm{NN}(A,B) = \mathrm{NN}(B,A) = 1 \Rightarrow \mathrm{MND}(A,B) = 2$$

If we compute the distance between points *B* and *C*, the results will be

$$\mathrm{NN}(B,C) = 1, \mathrm{NN}(C,B) = 2 \Rightarrow \mathrm{MND}(B,C) = 3$$
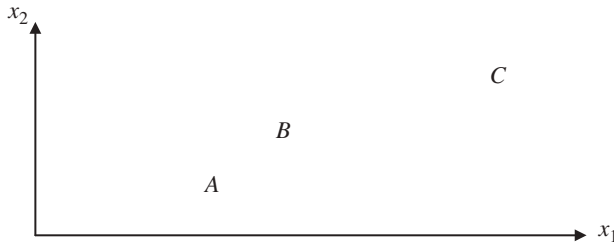
Figure 9.3. *A* and *B* are more similar than *B* and *C* using the MND measure.
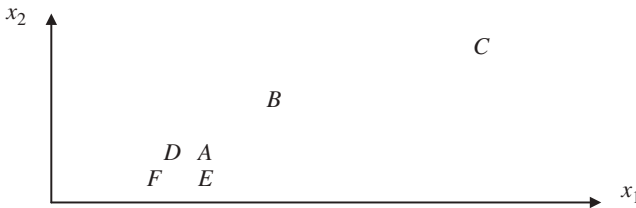
Figure 9.4. After changes in the context, *B* and *C* are more similar than A and B using the MND measure.

Figure 9.4 was obtained from Figure 9.3 by adding three new points D, E, and F (samples in the data set). Now, because the context has changed, the distances between the same points A, B, and C have also changed:

$$NN(A,B) = 1, \quad NN(B,A) = 4 \Rightarrow MND(A,B) = 5$$

$$NN(B,C) = 1, \quad NN(C,B) = 2 \Rightarrow MND(B,C) = 3$$

The MND between A and B has increased by introducing additional points close to A, even though A and B have not moved. B and C points become more similar than points A and B. The MND measure is not a metric because it does not satisfy the triangle inequality. Despite this, MND has been successfully applied in several real-world clustering tasks.

In general, based on a distance measure between samples, it is possible to define a distance measure between clusters (set of samples). These measures are an essential part in estimating the quality of a clustering process, and therefore they are part of clustering algorithms. The widely used measures for distance between clusters C_i and C_j are:

1. $D_{\min}(C_i, C_j) = \min|p_i - p_j|$, where $p_i \in C_i$ and $p_j \in C_j$.
2. $D_{\mean}(C_i, C_j) = |m_i - m_j|$, where m_i and m_j are centroids of C_i and C_j.
3. $D_{\avg}(C_i, C_j) = 1/(n_i \, n_j) \sum \sum |p_i - p_j|$, where $p_i \in C_i$ and $p_j \in C_j$ and n_i and n_j are the numbers of samples in clusters C_i and C_j.
4. $D_{\max}(C_i, C_j) = \max|p_i - p_j|$, where $p_i \in C_i$ and $p_j \in C_j$.

9.3   AGGLOMERATIVE HIERARCHICAL CLUSTERING

In hierarchical cluster analysis, we do not specify the number of clusters as a part of the input. Namely, the input to a system is (X, s), where X is a set of samples and s is a measure of similarity. An output from a system is a hierarchy of clusters. Most procedures for hierarchical clustering are not based on the concept of optimization, and the goal is to find some approximate, suboptimal solution using iterations for improvement of partitions until convergence. Algorithms of hierarchical cluster analysis are divided into two categories: divisible algorithms and agglomerative algorithms. A *divisible algorithm* starts from the entire set of samples X and divides it into a partition of subsets, then divides each subset into smaller sets, and so on. Thus, a divisible algorithm generates a sequence of partitions that is ordered from a coarser one to a finer one. An *agglomerative algorithm* first regards each object as an initial cluster. The clusters are merged into a coarser partition, and the merging process proceeds until the trivial partition is obtained: all objects are in one large cluster. This process of clustering is a bottom-up process, where partitions are from a finer one to a coarser one. In general, agglomerative algorithms are more frequently used in

real-world applications than divisible methods, and therefore we will explain the agglomerative approach in greater detail.

Most agglomerative hierarchical clustering algorithms are variants of the *single-link* or *complete-link* algorithms. These two basic algorithms differ only in the way they characterize the similarity between a pair of clusters. In the single-link method, the distance between two clusters is the *minimum* of the distances between all pairs of samples drawn from the two clusters (one element from the first cluster, the other from the second). In the complete-link algorithm, the distance between two clusters is the *maximum* of all distances between all pairs drawn from the two clusters. A graphical illustration of these two distance measures is given in Figure 9.5.

In either case, two clusters are merged to form a larger cluster based on minimum-distance criteria. Although the single-link algorithm is computationally more simple, from a practical viewpoint, it has been observed that the complete-link algorithm produces more useful hierarchies in most applications.

As explained earlier, the only difference between the single-link and complete-link approaches is in the distance computation. For both, the basic steps of the agglomerative-clustering algorithm are the same. These steps are as follows:

1. Place each sample in its own cluster. Construct the list of inter-cluster distances for all distinct unordered pairs of samples, and sort this list in ascending order.
2. Step through the sorted list of distances, forming for each distinct threshold value d_k a graph of the samples where pairs of samples closer than d_k are connected into a new cluster by a graph edge. If all the samples are members of a connected graph, stop. Otherwise, repeat this step.
3. The output of the algorithm is a nested hierarchy of graphs, which can be cut at the desired dissimilarity level forming a partition (clusters) identified by simple connected components in the corresponding subgraph.

Let us consider five points $\{x_1, x_2, x_3, x_4, x_5\}$ with the following coordinates as a 2D sample for clustering:
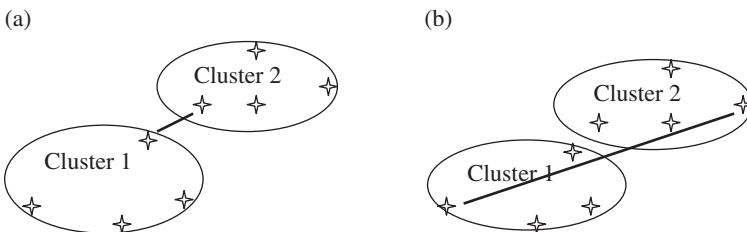

Figure 9.5. Distances for a single-link and a complete-link clustering algorithm. (a) Single-link distance. (b) Complete-link distance.
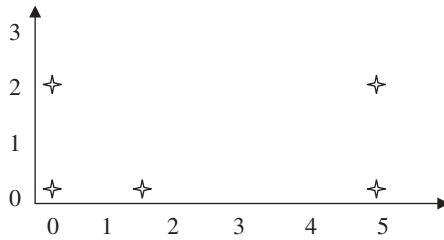
Figure 9.6. Five two-dimensional samples for clustering.

$$x_1 = (0,2), \quad x_2 = (0,0), \quad x_3 = (1.5,0), \quad x_4 = (5,0), \quad \text{and} \quad x_5 = (5,2).$$

For this example, we selected 2D points because it is easier to graphically represent these points and to trace all the steps in the clustering algorithm. The points are represented graphically in Figure 9.6.

The distances between these points using the Euclidean measure are

$$d(x_1,x_2) = 2, \quad d(x_1,x_3) = 2.5, \quad d(x_1,x_4) = 5.39, \quad d(x_1,x_5) = 5$$

$$d(x_2,x_3) = 1.5, \quad d(x_2,x_4) = 5, \quad d(x_2,x_5) = 5.29$$

$$d(x_3,x_4) = 3.5, \quad d(x_3,x_5) = 4.03$$

$$d(x_4,x_5) = 2$$

The distances between points as clusters in the first iteration are the same for both single-link and complete-link clustering. Further computation for these two algorithms is different. Using agglomerative single-link clustering, the following steps are performed to create a cluster and to represent the cluster structure as a dendrogram.

First, x_2 and x_3 samples are merged, and a cluster $\{x_2, x_3\}$ is generated with a minimum distance equal to 1.5. Second, x_4 and x_5 are merged into a new cluster $\{x_4, x_5\}$ with a higher merging level of 2.0. At the same time, the minimum single-link distance between clusters $\{x_2, x_3\}$ and $\{x_1\}$ is also 2.0. So, these two clusters merge at the same level of similarity as x_4 and x_5. Finally, the two clusters $\{x_1, x_2, x_3\}$ and $\{x_4, x_5\}$ are merged at the highest level with a minimum single-link distance of 3.5. The resulting dendrogram is shown in Figure 9.7.

The cluster hierarchy created by using an agglomerative complete-link clustering algorithm is different compared with the single-link solution. First, x_2 and x_3 are merged, and a cluster $\{x_2, x_3\}$ is generated with the minimum distance equal to 1.5. Also, in the second step, x_4 and x_5 are merged into a new cluster $\{x_4, x_5\}$ with a higher merging level of 2.0. Minimal single-link distance is between clusters $\{x_2, x_3\}$, and $\{x_1\}$ is now 2.5, so these two clusters merge after the previous two steps. Finally, the two clusters $\{x_1, x_2, x_3\}$ and $\{x_4, x_5\}$ are merged at the highest level with a minimal complete-link distance of 5.4. The resulting dendrogram is shown in Figure 9.8.
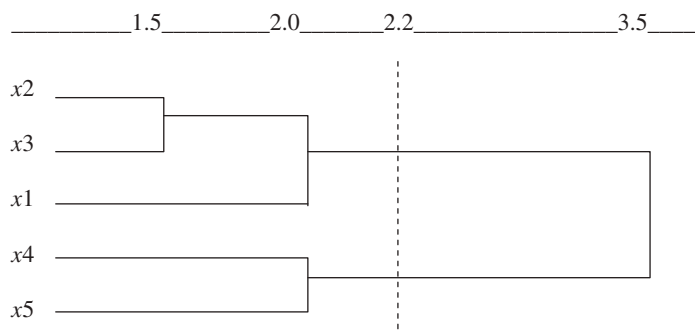
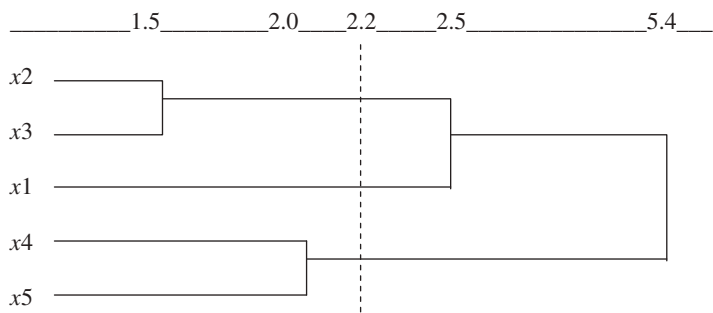Figure 9.7. Dendrogram by single-link method for the data set in Figure 9.6.

Figure 9.8. Dendrogram by complete-link method for the data set in Figure 9.6.

Selecting, for example, a threshold measure of similarity $s = 2.2$, we can recognize from the dendrograms in Figures 9.7 and 9.8 that the final clusters for single-link and complete-link algorithms are not the same. A single-link algorithm creates only two clusters, namely, $\{x_1, x_2, x_3\}$ and $\{x_4, x_5\}$, while a complete-link algorithm creates three clusters, namely, $\{x_1\}$, $\{x_2, x_3\}$, and $\{x_4, x_5\}$.

Unlike traditional agglomerative methods, *Chameleon* is a clustering algorithm that tries to improve the clustering quality by using a more elaborate criterion when merging two clusters. Two clusters will be merged if the interconnectivity and closeness of the merged clusters are very similar to the interconnectivity and closeness of the two individual clusters before merging.

To form the initial subclusters, Chameleon first creates a graph $G = (V, E)$, where each node $v \in V$ represents a data sample and a weighted edge $e(v_i, v_j)$ exists between two nodes v_i and v_j if v_j is one of the k-nearest neighbors of v_i. The weight of each edge in G represents the closeness between two samples, i.e., an edge will weigh more if the two data samples are closer to each other. *Chameleon* then uses a graph-partition algorithm to recursively partition G into many small, unconnected subgraphs by doing a min-cut on G at each level of recursion. Here, a min-cut on a graph G refers to a

partitioning of G into two parts of close, equal size such that the total weight of the edges being cut is minimized. Each subgraph is then treated as an initial subcluster, and the algorithm is repeated until a certain criterion is reached.

In the second phase, the algorithm goes bottom up. *Chameleon* determines the similarity between each pair of elementary clusters C_i and C_j according to *their relative interconnectivity* RI(C_i, C_j) and their *relative closeness* RC(C_i, C_j). Given that the interconnectivity of a cluster is defined as the total weight of edges that are removed when a min-cut is performed, the relative interconnectivity RI(C_i, C_j) is defined as the ratio of the interconnectivity of the merged cluster C_i and C_j to the average interconnectivity of C_i and C_j. Similarly, the relative closeness RC(C_i, C_j) is defined as the ratio of the closeness of the merged cluster of C_i and C_j to the average internal closeness of C_i and C_j. Here the closeness of a cluster refers to the average weight of the edges that are removed when a min-cut is performed on the cluster.

The similarity function is then computed as a product: RC(C_i, C_j) $*$ RI(C_i, C_j)$^\alpha$ where α is a parameter between 0 and 1. A value of 1 for α will give equal weight to both measures, while decreasing α will place more emphasis on RI(C_i, C_j). Chameleon can automatically adapt to the internal characteristics of the clusters, and it is effective in discovering arbitrarily shaped clusters of varying density. However, the algorithm is ineffective for high-dimensional data having $O(n^2)$ time complexity for n samples.

9.4   PARTITIONAL CLUSTERING

Every partitional-clustering algorithm obtains a single partition of the data instead of the clustering structure, such as a dendrogram, produced by a hierarchical technique. Partitional methods have the advantage in applications involving large data sets for which the construction of a dendrogram is computationally very complex. The partitional techniques usually produce clusters by optimizing a criterion function defined either locally (on a subset of samples) or globally (defined over all of the samples). Thus we say that a clustering criterion can be either global or local. A global criterion, such as the Euclidean square-error measure, represents each cluster by a prototype or centroid and assigns the samples to clusters according to the most similar prototypes. A local criterion, such as the minimal MND, forms clusters by utilizing the local structure or context in the data. Therefore, identifying high-density regions in the data space is a basic criterion for forming clusters.

The most commonly used partitional-clustering strategy is based on the square-error criterion. The general objective is to obtain the partition that, for a fixed number of clusters, minimizes the total square error. Suppose that the given set of N samples in an n-dimensional space has somehow been partitioned into K clusters $\{C_1, C_2,\ldots,C_k\}$. Each C_k has n_k samples and each sample is in exactly one cluster so that $\Sigma n_k = N$, where $k = 1,\ldots,K$. The mean vector M_k of cluster C_k is defined as the *centroid* of the cluster or

$$M_k = \left(\frac{1}{n_k}\right) \sum_{i=1}^{n_k} x_{ik}$$

where x_{ik} is the ith sample belonging to cluster C_k. The square error for cluster C_k is the sum of the squared Euclidean distances between each sample in C_k and its centroid. This error is also called the *within-cluster variation*:

$$e_k^2 = \sum_{i=1}^{n_k} (x_{ik} - M_k)^2$$

The square error for the entire clustering space containing K clusters is the sum of the within-cluster variations:

$$E_k^2 = \sum_{k=1}^{K} e_k^2$$

The objective of a square-error clustering method is to find a partition containing K clusters that minimize E_k^2 for a given K.

The *K-means partitional-clustering algorithm* is the simplest and most commonly used algorithm employing a square-error criterion. It starts with a random initial partition and keeps reassigning the samples to clusters, based on the similarity between samples and clusters, until a convergence criterion is met. Typically, this criterion is met when there is no reassignment of any sample from one cluster to another that will cause a decrease of the total squared error. K-means algorithm is popular because it is easy to implement, and its time and space complexity is relatively small. A major problem with this algorithm is that it is sensitive to the selection of the initial partition and may converge to a local minimum of the criterion function if the initial partition is not properly chosen.

The simple K-means partitional-clustering algorithm is computationally efficient and gives surprisingly good results if the clusters are compact, hyperspherical in shape, and well separated in the feature space. The basic steps of the K-means algorithm are as follows:

1. Select an initial partition with K clusters containing randomly chosen samples, and compute the centroids of the clusters.
2. Generate a new partition by assigning each sample to the closest cluster center.
3. Compute new cluster centers as the centroids of the clusters.
4. Repeat steps 2 and 3 until an optimum value of the criterion function is found (or until the cluster membership stabilizes).

Let us analyze the steps of the K-means algorithm on the simple data set given in Figure 9.6. Suppose that the required number of clusters is two and initially, clusters are formed from random distribution of samples: $C_1 = \{x_1, x_2, x_4\}$ and $C_2 = \{x_3, x_5\}$. The centroids for these two clusters are

$$M_1 = \left\{ \frac{(0+0+5)}{3}, \frac{(2+0+0)}{3} \right\} = \{1.66, 0.66\}$$

$$M_2 = \left\{ \frac{(1.5+5)}{2}, \frac{(0+2)}{2} \right\} = \{3.25,\ 1.00\}$$

Within-cluster variations, after initial random distribution of samples, are

$$e_1^2 = \left[(0-1.66)^2 + (2-0.66)^2 \right] + \left[(0-1.66)^2 + (0-0.66)^2 \right]$$

$$+ \left[(5-1.66)^2 + (0-0.66)^2 \right] = 19.36$$

$$e_2^2 = \left[(1.5-3.25)^2 + (0-1)^2 \right] + \left[(5-3.25)^2 + (2-1)^2 \right] = 8.12$$

and the total square error is

$$E^2 = e_1^2 + e_2^2 = 19.36 + 8.12 = 27.48$$

When we reassign all samples, depending on a minimum distance from centroids M_1 and M_2, the new redistribution of samples inside clusters will be

$$d(M_1, x_1) = \left(1.66^2 + 1.34^2 \right)^{1/2} = 2.1 \quad \text{and} \quad d(M_2, x_1) = 3.40 \Rightarrow x_1 \in C_1$$

$$d(M_1, x_2) = 1.79 \quad \text{and} \quad d(M_2, x_2) = 3.40 \Rightarrow x_2 \in C_1$$

$$d(M_1, x_3) = 0.83 \quad \text{and} \quad d(M_2, x_3) = 2.01 \Rightarrow x_3 \in C_1$$

$$d(M_1, x_4) = 3.41 \quad \text{and} \quad d(M_2, x_4) = 2.01 \Rightarrow x_4 \in C_2$$

$$d(M_1, x_5) = 3.60 \quad \text{and} \quad d(M_2, x_5) = 2.01 \Rightarrow x_5 \in C_2$$

New clusters $C_1 = \{x_1, x_2, x_3\}$ and $C_2 = \{x_4, x_5\}$ have new centroids

$$M_1 = \{0.5,\ 0.67\}$$
$$M_2 = \{5.0,\ 1.0\}$$

The corresponding within-cluster variations and the total square error are

$$e_1^2 = 4.17$$
$$e_2^2 = 2.00$$
$$E^2 = 6.17$$

We can see that after the first iteration, the total square error is significantly reduced (from the value 27.48 to 6.17). In this simple example, the first iteration was at the same time the final one because if we analyze the distances between the

new centroids and the samples, the latter will all be assigned to the same clusters. There is no reassignment and therefore the algorithm halts.

In summary, the K-means algorithm and its equivalent in an artificial neural networks domain—the Kohonen net—have been applied for clustering on large data sets. The reasons behind the popularity of the K-means algorithm are as follows:

1. Its time complexity is $O(n \times k \times l)$, where n is the number of samples, k is the number of clusters, and l is the number of iterations taken by the algorithm to converge. Typically, k and l are fixed in advance, and so the algorithm has linear time complexity in the size of the data set.
2. Its space complexity is $O(k + n)$, and if it is possible to store all the data in the primary memory, access time to all elements is very fast, and the algorithm is very efficient.
3. It is an order-independent algorithm. For a given initial distribution of clusters, it generates the same partition of the data at the end of the partitioning process irrespective of the order in which the samples are presented to the algorithm.

A big frustration in using iterative partitional-clustering programs is the lack of guidelines available for choosing K-number of clusters apart from the ambiguity about the best direction for initial partition, updating the partition, adjusting the number of clusters, and the stopping criterion. The K-means algorithm is very sensitive to noise and outlier data points, because a small number of such data can substantially influence the mean value. Unlike the K-means, the K-*medoids* method, instead of taking the mean value of the samples, uses the most centrally located object (medoids) in a cluster to be the cluster representative. Because of this, the K-medoids method is less sensitive to noise and outliers. *Fuzzy c-means*, proposed by Dunn and later improved, is an extension of K-means algorithm where each data point can be a member of multiple clusters with a membership value expressed through fuzzy sets. Despite its drawbacks, k-means remains the most widely used partitional-clustering algorithm in practice. The algorithm is simple, easily understandable, and reasonably scalable and can be easily modified to deal with streaming data.

9.5 INCREMENTAL CLUSTERING

There are more and more applications where it is necessary to cluster a large collection of data. The definition of "large" has varied with changes in technology. In the 1960s, "large" meant several thousand samples for clustering. Now, there are applications where millions of samples of high dimensionality have to be clustered. The algorithms discussed above work on large data sets, where it is possible to accommodate the entire data set in the main memory. However, there are applications where the entire data set cannot be stored in the main memory because of its size. There are currently three possible approaches to solve this problem:

1. The data set can be stored in a secondary memory, and subsets of this data are clustered independently, followed by a merging step to yield a clustering of the entire set. We call this approach the divide-and-conquer approach.
2. An incremental-clustering algorithm can be employed. Here, data are stored in the secondary memory, and data items are transferred to the main memory one at a time for clustering. Only the cluster representations are stored permanently in the main memory to alleviate space limitations.
3. A parallel implementation of a clustering algorithm may be used where the advantages of parallel computers increase the efficiency of the divide-and-conquer approach.

An incremental-clustering approach is most popular, and we will explain its basic principles. The following are the global steps of the incremental-clustering algorithm:

1. Assign the first data item to the first cluster.
2. Consider the next data item. Either assign this item to one of the existing clusters or assign it to a new cluster. This assignment is done based on some criterion, e.g., the distance between the new item and the existing cluster centroids. In that case, after every addition of a new item to an existing cluster, recompute a new value for the centroid.
3. Repeat step 2 until all the data samples are clustered.

The space requirements of the incremental algorithm are very small, necessary only for the centroids of the clusters. Typically, these algorithms are noniterative, and therefore their time requirements are also small. But even if we introduce iterations into the incremental-clustering algorithm, computational complexity and corresponding time requirements do not increase significantly. On the other hand, there is one obvious weakness of incremental algorithms that we have to be aware of. Most incremental algorithms do not satisfy one of the most important characteristics of a clustering process: order independence. An algorithm is order independent if it generates the same partition for any order in which the data set is presented. Incremental algorithms are very sensitive to the order of samples, and for different orders, they generate totally different partitions.

Let us analyze the incremental-clustering algorithm with the set of samples given in Figure 9.6. Suppose that the order of samples is x_1, x_2, x_3, x_4, x_5 and the threshold level of similarity between clusters is $\delta = 3$.

1. The first sample x_1 will become the first cluster $C_1 = \{x_1\}$. The coordinates of x_1 will be the coordinates of the centroid $M_1 = \{0, 2\}$.
2. Start analysis of the other samples.
   (a) Second sample x_2 is compared with M_1, and the distance d is determined:

   $$d(x_2, M_1) = \left(0^2 + 2^2\right)^{1/2} = 2.0 < 3$$

Therefore, x_2 belongs to the cluster C_1. The new centroid will be

$$M_1 = \{0, 1\}$$

(b) The third sample x_3 is compared with the centroid M_1 (still the only centroid!):

$$d(x_3, M_1) = \left(1.5^2 + 1^2\right)^{\frac{1}{2}} = 1.8 < 3$$

$$x_3 \in C_1 \Rightarrow C_1 = \{x_1, x_2, x_3\} \Rightarrow M_1 = \{0.5, 0.66\}$$

(c) The fourth sample x_4 is compared with the centroid M_1:

$$d(x_4, M_1) = \left(4.5^2 + 0.66^2\right)^{1/2} = 4.55 > 3$$

Because the distance of the sample from the given centroid M_1 is larger than the threshold value δ, this sample will create its own cluster $C_2 = \{x_4\}$ with the corresponding centroid $M_2 = \{5, 0\}$.

(d) The fifth sample x_5 is compared with both cluster centroids:

$$d(x_5, M_1) = \left(4.5^2 + 1.44^2\right)^{\frac{1}{2}} = 4.72 > 3$$

$$d(x_5, M_2) = \left(0^2 + 2^2\right)^{1/2} = 2 < 3$$

The sample is closer to the centroid M_2, and its distance is less than the threshold value δ. Therefore, sample x_5 is added to the second cluster C_2:

$$C_2 = \{x_4, x_5\} \Rightarrow M_2 = \{5, 1\}$$

3. All samples are analyzed, and a final clustering solution of two clusters is obtained:

$$C_1 = \{x_1, x_2, x_3\} \quad \text{and} \quad C_2 = \{x_4, x_5\}$$

The reader may check that the result of the incremental-clustering process will not be the same if the order of the samples is different. Usually, this algorithm is not iterative (although it could be extended!), and the clusters generated after all the samples have been analyzed in one iteration are the final clusters. If the iterative approach is used, the centroids of the clusters computed in the previous iteration are used as a basis for the partitioning of samples in the next iteration.

For most partitional-clustering algorithms, including the iterative approach, a summarized representation of the cluster is given through its *clustering feature (CF)* vector. This vector of parameters is given for every cluster as a triple, consisting of the number of points (samples) of the cluster, the centroid of the cluster, and the radius of the cluster. The cluster's radius is defined as the square root of the average mean squared distance from the centroid to the points in the cluster (averaged within-cluster variation). When a new point is added or removed from a cluster, the new CF

can be computed from the old CF. It is very important that we do not need the set of points in the cluster to compute a new CF value.

If samples are with categorical data, then we do not have a method to calculate centroids as representatives of the clusters. In that case, an additional algorithm called *k-nearest neighbor* may be used to estimate distances (or similarities) between samples and existing clusters. The basic steps of the algorithm are:

1. to compute the distances between the new sample and all previous samples, already classified into clusters,
2. to sort the distances in increasing order and select K samples with the smallest distance values, and
3. to apply the voting principle. A new sample will be added (classified) to the largest cluster out of K selected samples.

For example, given six six-dimensional categorical samples

$$X_1 = \{A, B, A, B, C, B\}$$
$$X_2 = \{A, A, A, B, A, B\}$$
$$X_3 = \{B, B, A, B, A, B\}$$
$$X_4 = \{B, C, A, B, B, A\}$$
$$X_5 = \{B, A, B, A, C, A\}$$
$$X_6 = \{A, C, B, A, B, B\}$$

they are gathered into two clusters $C_1 = \{X_1, X_2, X_3\}$ and $C_2 = \{X_4, X_5, X_6\}$. How does one to classify the new sample $Y = \{A, C, A, B, C, A\}$?

To apply the *k*-nearest neighbor algorithm, it is necessary, as the first step, to find all distances between the new sample and the other samples already clustered. Using the SMC measure, we can find similarities instead of distances between samples.

Similarities with elements in C_1	Similarities with elements in C_2
SMC(Y, X_1) = 4/6 = 0.66	SMC(Y, X_4) = 4/6 = 0.66
SMC(Y, X_2) = 3/6 = 0.50	SMC(Y, X_5) = 2/6 = 0.33
SMC(Y, X_3) = 2/6 = 0.33	SMC(Y, X_6) = 2/6 = 0.33

Using the 1-nearest neighbor rule ($K = 1$), the new sample cannot be classified because there are two samples (X_1 and X_4) with the same highest similarity (smallest distances), and one of them is in the class C_1 and the other in the class C_2. On the other hand, using the 3-nearest neighbor rule ($K = 3$) and selecting the three largest similarities in the set, we can see that two samples (X_1 and X_2) belong to the class C_1 and only one sample belongs to the class C_2. Therefore, using a simple voting system, we can classify the new sample Y into the C_1 class.

9.6 DBSCAN ALGORITHM

Density-based approach in clustering assumes that clusters are regarded as dense regions of objects in the data space that are separated by regions of low object density (noise). These regions may have an arbitrary shape. Crucial concepts of this approach are density and connectivity both measured in terms of local distribution of nearest neighbors. The algorithm Density-Based Spatial Clustering of Applications with Noise (DBSCAN) targeting low-dimensional data is the major representative in this category of density-based clustering algorithms. The main reason why DBSCAN recognizes the clusters is that within each cluster, we have a typical density of points that is considerably higher than outside of the cluster. Furthermore, the points' density within the areas of noise is lower than the density in any of the clusters.

DBSCAN is based on two main concepts: *density reachability* and *density connectability*. Both concepts depend on two input parameters of the DBSCAN clustering: the size of epsilon neighborhood (ε) and the minimum points in a cluster (m). The key idea of the DBSCAN algorithm is that for each point of a cluster, the neighborhood of a given radius ε has to contain at least a minimum number of points m; that is, the density in the neighborhood has to exceed some predefined threshold. For example, in Figure 9.9, point p has only two points in the neighborhood ε, while point q has eight. Obviously, the density around q is higher than around p.

Density reachability defines whether two close points belong to the same cluster. Point p_1 is density reachable from p_2 if two conditions are satisfied: (1) the points are close enough to each other: distance$(p_1, p_2) < \varepsilon$, and (2) there are enough of points in ε neighborhood of p_2: distance$(r, p_2) > m$, where r are some database points. In the example represented in Figure 9.9, point p is reachable from point q. *Density connectivity* is the next building step of DBSCAN. Points p_0 and p_n are *density connected* if there is a sequence of *density reachable* points $(p_0, p_1, p_2,\ldots)$ from p_0 to p_n such that p_{i+1} is *density reachable* from p_i. These ideas are translated into DBSCAN *cluster* as a set of all density connected points.

The clustering process is based on the classification of the points in the data set as *core points*, *border points*, and *noise points* (examples are given in Fig. 9.10):

- A point is a *core point* if it has more than a specified number of points (m) within neighborhood ε. These are points that are at the interior of a cluster.
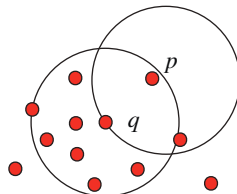

Figure 9.9. Neighborhood (ε) for points p and q.
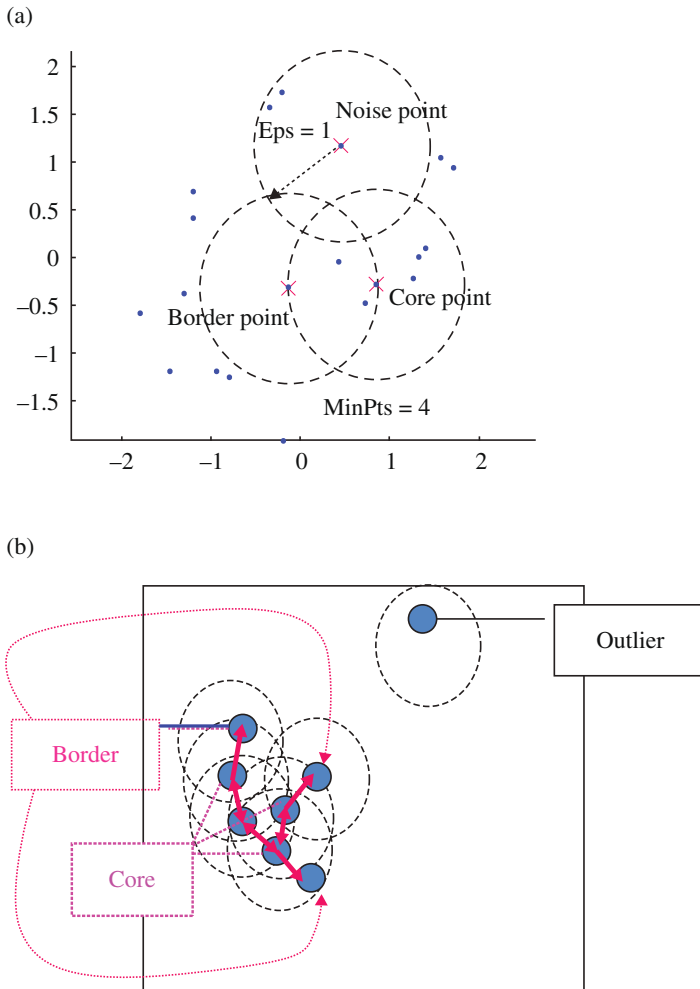
(a)

(b)

Figure 9.10. Examples of core, border, and noise points. (a) ε and m determine the type of the point. (b) Core points build dense regions.

- A *border point* has fewer than m points within its neighborhood ε, but it is in the neighbor of a core point.
- A *noise point* is any point that is not a core point or a border point.

Ideally, we would have to know the appropriate parameters ε and m of each cluster. But there is no easy way to get this information in advance for all clusters of the database. Therefore, DBSCAN uses global values for ε and m, i.e. the same values for all clusters. Also, numerous experiments indicate that DBSCAN clusters for $m > 4$ do

not significantly differ from the case $m = 4$, while the algorithm needs considerably more computations. Therefore, in practice, we may eliminate the parameter m by setting it to 4 for low-dimensional databases. The main steps of DBSCAN algorithm are:

- Arbitrary select a point p.
- Retrieve all points density reachable from p with respect to ε and m.
- If p is a core point, a new cluster is formed or existing cluster is extended.
- If p is a border point, no points are density reachable from p, and DBSCAN visits the next point of the database.
- Continue the process with other points in database until all of the points have been processed.
- Since global values for ε and m are used, DBSCAN may merge two clusters into one cluster if two clusters of different density are "close" to each other. They are close if the distance between clusters is lower than ε.

Examples of clusters obtained by DBSCAN algorithm are illustrated in Figure 9.11. Obviously, DBSCAN finds all clusters properly, independent of the size, shape, and location of clusters to each other.

The main advantages of the DBSCAN clustering algorithm are as follows:

1. DBSCAN does not require the number of clusters a priori, as opposed to K-means and some other popular clustering algorithms.
2. DBSCAN can find arbitrarily shaped clusters.
3. DBSCAN has a notion of noise and eliminate outliers from clusters.
4. DBSCAN requires just two parameters and is mostly insensitive to the ordering of the points in the database.

DBSCAN also has some disadvantages. The complexity of the algorithm is still very high, although with some indexing structures it reaches $O(n \times \log n)$. Finding neighbors is an operation based on distance, generally the Euclidean distance, and
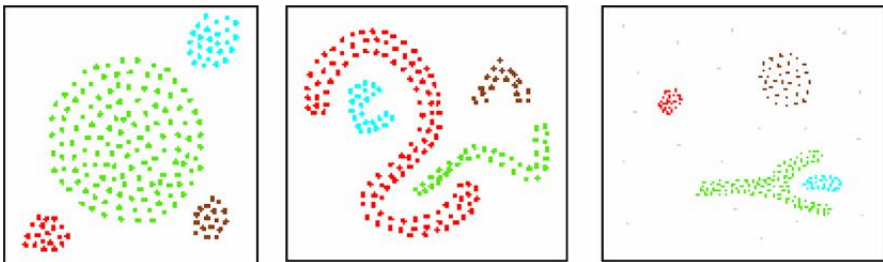
Figure 9.11. DBSCAN builds clusters of different shapes.

the algorithm may find the curse of dimensionality problem for high-dimensional data sets. Therefore, most applications of the algorithm are for low-dimensional real-world data.

9.7   BIRCH ALGORITHM

Balanced and Iterative Reducing and Clustering using Hierarchies (BIRCH) is an efficient clustering technique for data in Euclidean vector spaces. The algorithm can efficiently cluster data with a single pass, and also it can deal effectively with outliers. BIRCH is based on the notion of a *CF* and a *CF tree*.

CF is a small representation of an underlying cluster that consists of one or many samples. BIRCH builds on the idea that samples that are close enough should always be considered as a group. CFs provide this level of abstraction with corresponding summarization of samples in a cluster. The idea is that a cluster of data samples can be represented by a triple of numbers (N, LS, SS), where N is the number of samples in the cluster, LS is the linear sum of the data points (vectors representing samples), and SS is the sum of squares of the data points. More formally, the components of vectors LS and SS are computed for every attribute X of data samples in a cluster:

$$\mathrm{LS}(X) = \sum_{i=1}^{N} X_i \quad \text{and} \quad \mathrm{SS}(X) = \sum_{i=1}^{N} X_i^2$$

In Figure 9.12 five 2D samples are representing the cluster, and their CF summary is given with components: $N = 5$, LS = (16, 30), and SS = (54, 190). These are common statistical quantities, and a number of different cluster characteristics and intercluster distance measures can be derived from them. For example, we can compute the centroid for the cluster based on its CF representation, without revisiting original samples. Coordinates of the centroid are obtained by dividing the components of the LS vector by N. In our example the centroid will have the coordinates (3.2, 6.0). The reader may check on the graphical interpretation of the data (Fig. 9.12) that the position of the centroid is correct. The obtained summaries are then used instead of the original data for further clustering or manipulations with clusters. For example, if $CF_1 = (N_1, LS_1, SS_1)$ and $CF_2 = (N_2, LS_2, SS_2)$ are the CF entries of two disjoint clusters, then the CF entry of the cluster formed by merging the two clusters is

$$CF = CF_1 + CF_2 = (N_1 + N_2, LS_1 + LS_2, SS_1 + SS_2)$$

This simple equation show us how simple is procedure for merging clusters based on their simplified CF descriptions. That allows efficient incremental merging of clusters even for the streaming data!

BIRCH uses a hierarchical data structure called CF tree for partitioning the incoming data points in an incremental and dynamic way. A CF tree is a height-balanced tree

CF = (5, (16, 30),(54, 190))
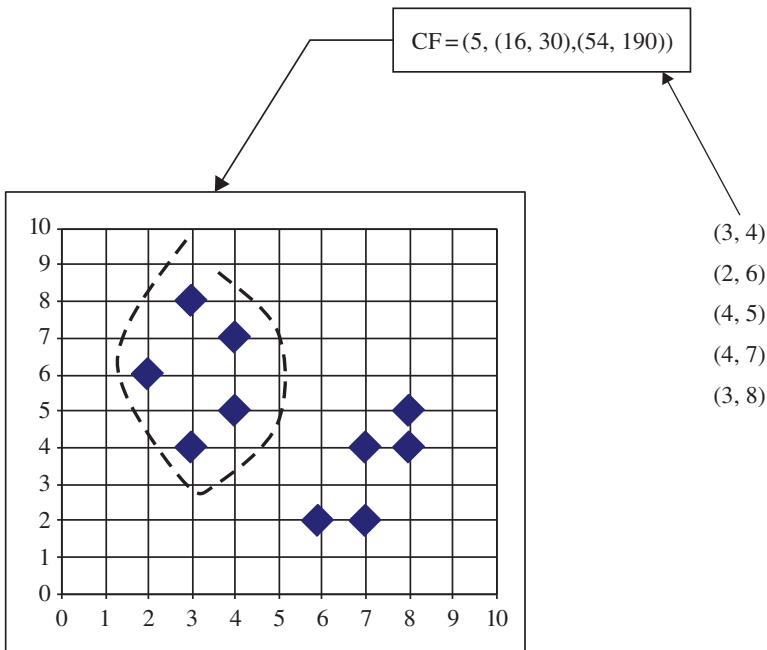

(3, 4)
(2, 6)
(4, 5)
(4, 7)
(3, 8)

Figure 9.12. CF representation and visualization for a 2D cluster.

usually stored in a central memory. This allows fast lookups even when large data sets have been read. It is based on two parameters for nodes. CF nodes can have at maximum B children for nonleaf nodes and a maximum of L entries for leaf nodes. Also, T is the threshold for the maximum diameter of an entry in the cluster. The CF tree size is a function of T. The bigger T is, the smaller the tree will be.

A CF tree is built as the data samples are scanned (Fig. 9.13). At every level of the tree, a new data sample is inserted to the closest node. Upon reaching a leaf, the sample is inserted to the closest CF entry, as long as it is not overcrowded (diameter of the cluster $D > T$ after the insert). Otherwise, a new CF entry is constructed, and the sample is inserted. Finally, all CF statistics are updated for all nodes from the root to the leaf to represent the changes made to the tree. Since the maximum number of children per node (branching factor) is limited, one or several splits can happen. Building CF tree is only one, but the most important, phase in the BIRCH algorithm. In general, BIRCH employs four different phases during the clustering process:

1. Phase 1: Scan *all data and build an initial in-memory CF tree*.
   It linearly scans all samples and inserts them in the CF tree as described earlier.
2. Phase 2: Condense *the tree to a desirable size by building a smaller CF tree*.
   This can involve removing outliers and further merging of clusters.
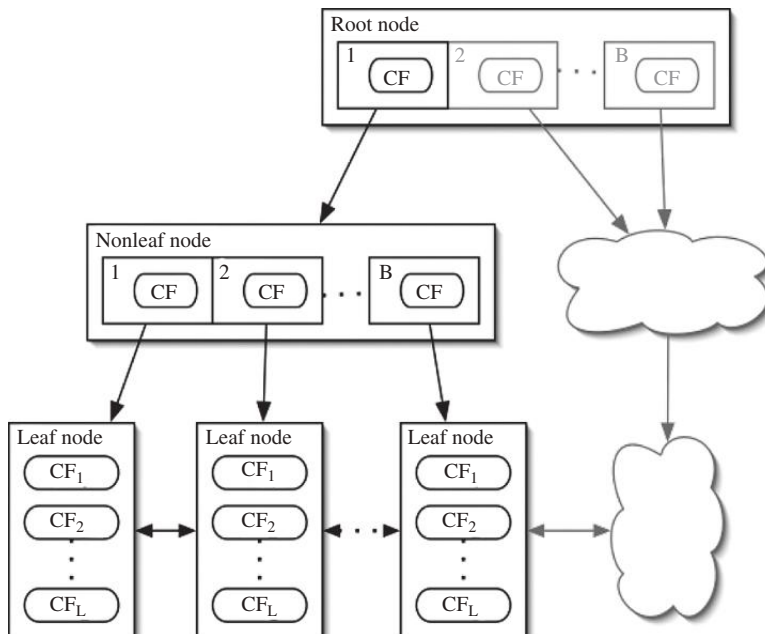
Figure 9.13. CF tree structure.

3. Phase 3: Global *clustering*.

   Employ a global clustering algorithm using the CF tree's leaves as input. CFs
   allow for effective clustering because the CF tree is very densely compressed
   in the central memory at this point. The fact that a CF tree is balanced allows
   the log-efficient search.

4. Phase 4: Cluster *refining*.

   This is optional, and it requires more passes over the data to refine the
   results. All clusters are now stored in memory. If desired, the actual data
   points can be associated with the generated clusters by reading all points
   from disk again.

BIRCH performs faster than most of existing algorithms on large data sets. The
algorithm can typically find a good clustering with a single scan of the data and
improve the quality further with a few additional scans (phases 3 and 4). Basic algo-
rithm condenses metric data in the first pass using spherical summaries, and this part
can be an incremental implementation. Additional passes cluster CFs to detect non-
spherical clusters, and the algorithm approximates density function. There are several
extensions of the algorithm trying to include non-metric data and make applicability
of the approach much wider.

9.8   CLUSTERING VALIDATION

How is the output of a clustering algorithm evaluated? What characterizes a "good" clustering result and a "poor" one? All clustering algorithms will, when presented with data, produce clusters regardless of whether the data contain clusters or not. Therefore, the first step in evaluation is actually an assessment of the data domain rather than the clustering algorithm itself. Data that we do not expect to form clusters should not be processed by any clustering algorithm. If the data does contain clusters, some clustering algorithms may obtain a "better" solution than others. Cluster validity is the second step when we expect to have our data clusters. A clustering structure is valid if it cannot reasonably have occurred by chance or as an artifact of a clustering algorithm. Applying some of the available cluster methodologies, we assess the outputs. This analysis uses a specific criterion of optimality that usually contains knowledge about the application domain and therefore is subjective. There are three types of validation studies for clustering algorithms. An *external* assessment of validity compares the discovered structure to an a priori structure. An *internal* examination of validity tries to determine if the discovered structure is intrinsically appropriate for the data. Both assessments are subjective and domain dependent. A *relative* test, as a third approach, compares the two structures obtained either from different cluster methodologies or by using the same methodology but with different clustering parameters, such as the order of input samples. This test measures their relative merit, but we still need to resolve the question how to select the structures for comparison.

Theory and practical applications both show that all approaches in the validation of clustering results have a subjective component. Hence, little in the way of "gold standards" exists in clustering evaluation. Recent studies in cluster analysis suggest that a user of a clustering algorithm should keep always the following issues in mind:

1. Every clustering algorithm will find clusters in a given data set whether they exist or not; the data should, therefore, be subjected to tests for clustering tendency before applying a clustering algorithm, followed by a validation of the clusters generated by the algorithm.
2. There is no best clustering algorithm; therefore a user is advised to try several algorithms on a given data set.

It is important to remember that cluster analysis is an exploratory tool; the outputs of clustering algorithms only suggest or sometimes confirm hypotheses, but never prove any hypothesis about natural organization of data. Still, there are some standard clustering validation measures that should be used in practical applications.

Clustering validation has long been recognized as one of the vital issues essential to the success of clustering applications. Numerical measures, also referred as criteria or indices, which are applied to judge various aspects of cluster validity, may be classified into two main categories:

- *Internal measures:* Used to measure the goodness of a clustering structure *without* use of any external information or when external information is not available at all. Internal validation measures only rely on information in the data.
- *External measures:* Used to measure the extent to which cluster labels match externally supplied class labels. For example, it is measured how a clustering partition presents clusters defined by important background characteristics representing a basis for the partition but not available in the data set or when the true clusters in the data are known in advance and samples are class-labeled as a "ground truth."

Most of internal measures for clustering validation are based on two parameters: cohesion and separation. Cluster cohesion measures how close are samples inside each cluster, as it is presented in Figure 9.14a. The most straightforward way to formalize that all objects within a cluster should be similar to each other; it is expected that clusters are highly homogeneous. There are numerous measures that estimate the cluster compactness based on distance, such as maximum or average pairwise distance and maximum or average center-based distance. For example, cohesion is measured as the within-cluster sum of squared errors (WSS):

$$\text{WSS} = \sum_{k=1,N} \sum_{x_i \& C} \left(x_i - m_j \right)^2$$

where N is the number of clusters, C is set of centroids x_i determined by the clustering algorithm, and m_j are samples, members of corresponding cluster where x_i is a centroid.

Separation measures how distinct or well separated a cluster is from other clusters, and it is expressed through distances between samples in different clusters as it is given in Figure 9.14b. For example, the pairwise distances between cluster centers or the pairwise minimum distances between objects in different clusters are widely used as measures of separation. Separation as it is usually interpreted cannot be measured by averaging all between-cluster dissimilarities, because it refers to what goes on "between" the clusters. The smallest between-cluster distances may have more weight
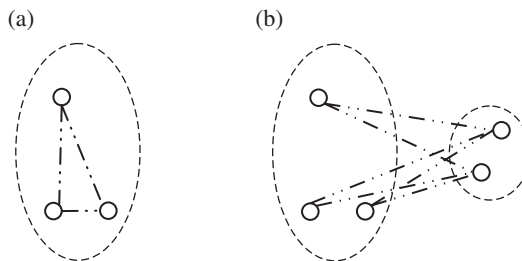

(a)   (b)

Figure 9.14. Components of the internal validation of the clustering. (a) Cluster cohesion. (b) Cluster separation.

in the final separation measure than the distance between pairs of farthest clusters. One possible interpretation of the separation measure is given by

$$BSS = \sum_{x_i \,\&\, C} C_i \,(x - x_i)^2$$

where x_i are centroids of the clusters, x is centroid of the entire data set, and C_i is the size of ith cluster. In this case separation is expressed as a weight sum of distances between centroids of all generated clusters and centroid of the entire data set. Sometimes in this measure it is necessary to include the weight representing the size of clusters.

One of the approaches in evaluating clustering quality is silhouette method. The method of silhouette combines cohesion and separation into one parameter. The silhouette value s_i is determined for each sample in the clustered data set:

$$s_i = \frac{(b_i - a_i)}{\max(b_i, a_i)}$$

where a_i is the distance between ith sample and its centroid (cluster center), while b_i is the distance between ith sample and the next closest centroid (Fig. 9.15).

It is obvious that $b_i > a_i$ for each sample. So, the value of s_i is positive number, and we expect values for s_i greater of 0.5 for most of samples if the clustering is successful. Taking the average silhouette value S over all n samples

$$S = \frac{1}{N} \times \sum_{i=1}^{N} s_i$$

it is obtained a good combined internal measure S (cohesion + separation) for the quality of a clustering process. The largest overall average silhouette indicates the best clustering if several clustering algorithms are compared. Also, the appropriate number of cluster may be determined through experiments using maximum overall average silhouette criterion.
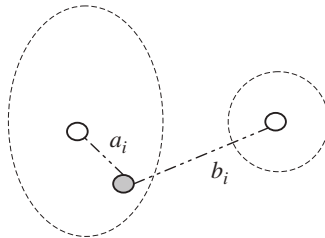

Figure 9.15. Components of the silhouette coefficient.

The Rand index, developed by William Rand in 1971, is one of the mostly used external measures for clustering. It is comparing results of clustering algorithm with a priori given labels (classes) of all samples in data set. This index measures the number of pairwise agreements between the set of discovered clusters K and a set of class labels C, given by

$$R = \frac{(a+d)}{(a+b+c+d)}$$

where

- a denotes the number of pairs of data points with the same label in C and assigned to the same cluster in K
- b denotes the number of pairs with the same label, but in different clusters
- c denotes the number of pairs in the same cluster, but with different class labels
- d denotes the number of pairs with a different label in C that were assigned to a different cluster in K

The index results in $0 \leq R \leq 1$, where a value of 1 indicates that C and K are identical. A high value for this index generally indicates a high level of agreement between a clustering and the natural classes. Rand index may be also used to compare similarity of results between two clustering algorithms. In that case the results of second clustering algorithm replace ground truth of labels C.

Purity P is an additional external evaluation measure for the clustering. The local purity p_i is determined for each cluster, and it represents the fraction of data samples assigned to the majority label in each cluster. Total purity P is accumulated local purities for each cluster divided by total number of samples in data set:

$$P = \sum_{i=1,N} \frac{p_i}{M}$$

where p_i are purity values for each cluster, N is the number of clusters, and M is the total number of samples in data set. For example, if the clustering algorithm resulted in four clusters, c_1 to c_4, and also the samples have externally determined one out of four labels (dog, cat, mouse, or fox), the following matrix shows how much the clusters are covered with external labels.

	c_1	c_2	c_3	c_4
Dog	0	4	**9**	4
Cat	**10**	0	1	0
Mouse	0	10	5	**12**
Fox	1	**13**	5	10

Purity p_1 for cluster c_1 is equal 10 because label cat is majority label with 10 samples in this cluster. In this case, total purity P, for applied clustering results, is

$$P = \frac{(10 + 13 + 9 + 12)}{84} = \frac{44}{84} = 0.52.$$

One class of clustering algorithms may be very useful in the applications where the border regions between clusters in N-dimensional spaces are not clearly defined. For these kinds of data sets, several fuzzy clustering algorithms are developed. One example is fuzzy c-means clustering algorithm, which works by assigning membership to each data point corresponding to each cluster center on the basis of distance between the cluster center and the data point. The more the data point is closer to the cluster center, the more is its membership toward the particular cluster center. Each data point can have membership to multiple clusters. By relaxing the definition of membership coefficients from strictly 1 or 0, these values can range from 1 to 0.

Figure 9.16a shows how the specific point P may belong to two different clusters with centroids C_A and C_B. Assume that Euclidean distance is used as inversely proportional measure of the membership function, where $m_B = 0.6$ is greater than $m_A = 0.4$, but the point P based on these values still belongs to both clusters. Clearly, summation of membership of each data point should be equal to one. Figure 9.16b shows the data samples on X axes, where different membership values are assigned for each single point depending on distances from centroids of the clusters. If the threshold value for membership is selected, such as $m = 0.3$, crisp clusters A and B may be separated.

Fuzzy c-means clustering algorithm gives best result for overlapped data set in N-dimensional spaces, and multiple experiments showed that it gives comparatively better results than k-means algorithm. Unlike k-means where data point must exclusively belong to one cluster center, here data point is assigned through the membership to belong to more than one cluster.
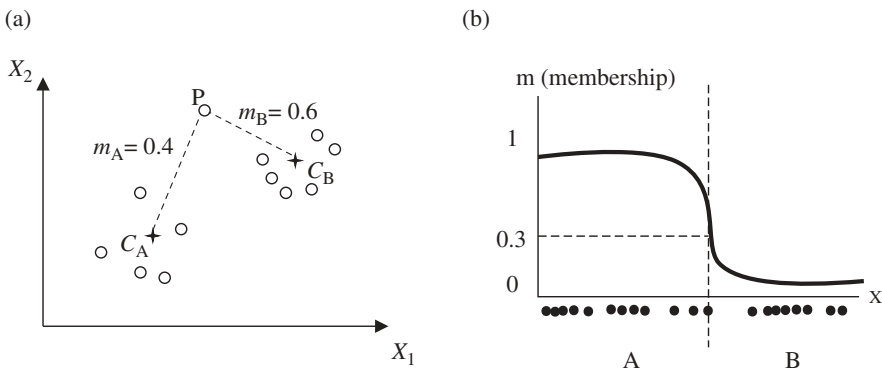

Figure 9.16. Membership function for samples in fuzzy clustering. (a) Sum of membership coefficients is equal to 1. (b) Threshold of 0.3 determines the clusters A and B.

9.9 REVIEW QUESTIONS AND PROBLEMS

1. Why is the validation of a clustering process highly subjective?

2. What increases the complexity of clustering algorithms?

3. (a) Using MND distance, distribute the input samples given as 2D points $A(2, 2)$, $B(4, 4)$, and $C(7, 7)$ into two clusters.

    (b) What will be the distribution of samples in the clusters if samples $D(1, 1)$, $E(2, 0)$, and $F(0, 0)$ are added?

4. Given five-dimensional numeric samples $A = (1, 0, 2, 5, 3)$ and $B = (2, 1, 0, 3, -1)$, find:

    (a) The Euclidean distance between points.
    (b) The city block distance.
    (c) The Minkowski distance for $p = 3$.
    (d) The cosine-correlation distance.

5. Given six-dimensional categorical samples $C = (A, B, A, B, A, A)$ and $D = (B, B, A, B, B, A)$, find:

    (a) A simple matching coefficient (SMC) of the similarity between samples.
    (b) Jaccard's coefficient.
    (c) Rao's coefficient.

6. Given a set of five-dimensional categorical samples:

$$A = (1, 0, 1, 1, 0)$$
$$B = (1, 1, 0, 1, 0)$$
$$C = (0, 0, 1, 1, 0)$$
$$D = (0, 1, 0, 1, 0)$$
$$E = (1, 0, 1, 0, 1)$$
$$F = (0, 1, 1, 0, 0)$$

    (a) Apply agglomerative hierarchical clustering using:
       (i) Single-link similarity measure based on Rao's coefficient.
      (ii) Complete-link similarity measure based on simple matching coefficient SMC.
    (b) Plot the dendrograms for the solutions to part (i) and (ii) of (a).

7. Given the samples $X_1 = \{1, 0\}$, $X_2 = \{0, 1\}$, $X_3 = \{2, 1\}$, and $X_4 = \{3, 3\}$, suppose that the samples are randomly clustered into two clusters $C_1 = \{X_1, X_3\}$ and $C_2 = \{X_2, X_4\}$.

    (a) Apply one iteration of the K-means partitional-clustering algorithm, and find a new distribution of samples in clusters. What are the new centroids? How can you prove that the new distribution of samples is better than the initial one?

(b) What is the change in the total square error?

(c) Apply the second iteration of the K-means algorithm and discuss the changes in clusters.

8. For the samples in Problem #7, apply iterative clustering with the threshold value for cluster radius $T = 2$. What is the number of clusters and samples distribution after the first iteration?

9. Suppose that the samples in Problem #6 are distributed into two clusters:

$$C1 = \{A, B, E\} \quad \text{and} \quad C2 = \{C, D, F\}.$$

Using k-nearest neighbor algorithm, find the classification for the following samples:

(a) $Y = \{1, 1, 0, 1, 1\}$ using $K = 1$.

(b) $Y = \{1, 1, 0, 1, 1\}$ using $K = 3$.

(c) $Z = \{0, 1, 0, 0, 0\}$ using $K = 1$.

(d) $Z = \{0, 1, 0, 0, 0\}$ using $K = 5$.

10. Implement the hierarchical agglomerative algorithm for samples with categorical values using the SMC measure of similarity.

11. Implement the partitional K-means clustering algorithm. Input samples are given in the form of a flat file.

12. Implement the incremental-clustering algorithm with iterations. Input samples are given in the form of a flat file.

13. Given the similarity matrix between five samples:

(a) Use the similarity matrix in the Table to perform *complete-link* hierarchical clustering. Show your results by drawing a dendrogram. The dendrogram should clearly show the order in which the points are merged.

(b) How many clusters exist if the threshold similarity value is 0.5. Give the elements of each cluster.

(c) If DBSCAN algorithm is applied with threshold similarity of 0.6 and MinPts ≥ 2 (required density), what are *core*, *border*, and *noise* points in the set of points p_i given in the Table. Explain.

Similarity matrix for Exercise 13.

	$p1$	$p2$	$p3$	$p4$	$p5$
$p1$	1.00	0.10	0.41	0.55	0.35
$p2$	0.10	1.00	0.64	0.47	0.98
$p3$	0.41	0.64	1.00	0.44	0.85
$p4$	0.55	0.47	0.44	1.00	0.76
$p5$	0.35	0.98	0.85	0.76	1.00

14. Given the points $x_1 = \{1, 0\}$, $x_2 = \{0,1\}$, $x_3=\{2, 1\}$, and $x_4 = \{3, 3\}$. Suppose that these points are randomly clustered into two clusters: $C_1 = \{x_1, x_3\}$ and $C_2 = \{x_2, x_4\}$. Apply one iteration of *K-means partitional-clustering algorithm* and find new distribution of elements in clusters. What is the change in a *total square error*?

15. Answer True/False to the following statements. Discuss your answer if necessary.

     1. Running K-means with different initial seeds is likely to produce different results
     2. Initial cluster centers have to be data points.
     3. Clustering stops when cluster centers are moved to the mean of clusters.
     4. k-means can be less sensitive to outliers if standard deviation is used instead of the average.
     5. k-means can be less sensitive to outliers if median is used instead of the average.

16. Identify the clusters in the figure below using the center-, contiguity-, and density-based clustering. Assume center-based clustering means *K-means*, contiguity-based means single-link hierarchical, and density-based means *DBSCAN*. Also indicate the number of clusters for each case, and give a brief indication of your reasoning. Note that darkness or the number of dots indicates density.

(a)   (b)   (c)   (d)
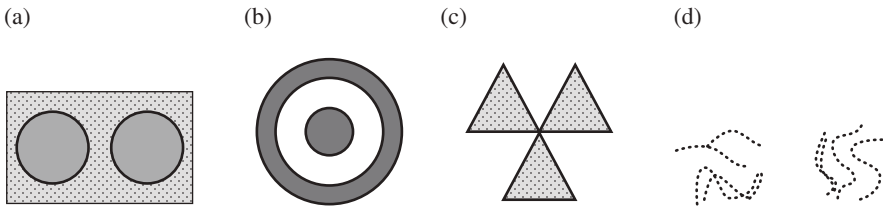
17. Derive the mathematical relationship between cosine similarity and Euclidean distance when each data object has an L2 (Euclidean) length of 1.

18. Given a similarity measure with values in the interval $[0, 1]$, describe two ways to transform this similarity value into a *dissimilarity* value in the interval $[0, \infty]$.

19. Distances between samples (A, B, C, D, and E) are given in a graphical form.
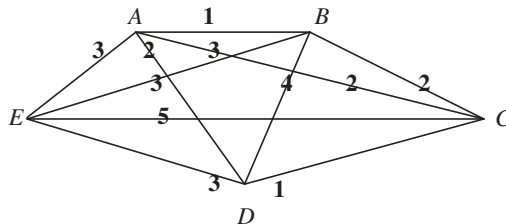
Determine single-link and complete-link dendrograms for the set of samples.

20. There is a set S consisting of six points in the plane shown as below: $a = (0, 0)$, $b = (8, 0)$, $c = (16, 0)$, $d = (0, 6)$, $e = (8, 6)$, $f = (16, 6)$. Now we run the k-means algorithm on those points with $k = 3$. The algorithm uses the Euclidean distance metric (i.e. the straight-line distance between two points) to assign each point to its nearest centroid. Also we define the following:

- *3-starting configuration* is a subset of three starting points from S that form the initial centroids, e.g. $\{a, b, c\}$.
- *3-partition* is a partition of S into k non-empty subsets, e.g. $\{a, b, e\}, \{c, d\}, \{f\}$ is a 3-partition.

(a) How many 3-starting configurations are there?
(b) Fill in the last two columns of the following table.

3-Partition	An Example of 3-Starting Configuration That Can Arrive at the 3-Partition After 0 or More Iterations of k-Means	Number of Unique 3-Starting Configurations
$\{a, b\}$ $\{d, e\}$ $\{c, f\}$		
$\{a\}$ $\{d\}$ $\{b, c, e, f\}$		
$\{a, b, d\}$ $\{c\}$ $\{e, f\}$		
$\{a, b\}$ $\{d\}$ $\{c, e, f\}$		

21. On the space of nonnegative integers, which of the following functions are distance measures? Explain.

(a) $\max(x, y)$ = the larger of x and y.
(b) $\text{diff}(x, y) = |x - y|$ (the magnitude of the difference between x and y).
(c) $\text{sum}(x, y) = x + y$.

22. Find the *edit distances* between the following pairs of strings:

(a) *abcdef* and *bdaefc*.
(b) *abccdabc* and *acbdcab*.
(c) *abcdef* and *baedfc*.

23. Consider the following three vectors u, v, and w in a six-dimensional space:

$$u = [1, 0.25, 0, 0, 0.5, 0]$$
$$v = [0.75, 0, 0, 0.2, 0.4, 0]$$
$$w = [0, 0.1, 0.75, 0, 0, 1]$$

Suppose $\cos(x,y)$ denotes the similarity of vectors x and y under the cosine similarity measure. Compute all three pairwise similarities among u, v, and w.

24. In a certain high-dimensional space, points A and B are in one cluster, and points C, D, and E are in another cluster. Points in the same cluster may be assumed "close," while points in different clusters are "distant." Assuming the "curse of dimensionality" applies in this case, we expect certain angles between the lines from one of these points X to two other points Y and Z to be approximately right angles. Identify which of the following angles we would assume *not* to be approximately a right angle.

   (a) Select which of the following angles are *not* right angles: ACB, ACD, ADB, CAD, CEB, CBE.
   (b) What is the general rule to select not right angles?

25. Given five vectors in a ten-dimensional space:

$$1111000000, 0100100101, 0000011110, 0111111111, 1011111111$$

Compute the cosines' distances of the angles between each pair of these vectors. Note that, conveniently, each vector has either 4 or 9 1's. Then, identify one of the cosines of these angles from the list of fractions below.

   (a) Select which of the following values are possible cosine distances: 1/3, ¼, ½, 5/6, 2/3, ¾.
   (b) What is the general rule to select the value as possible cosine measure?

26. Suppose our data set consists of the perfect squares 1, 4, 9, 16, 25, 36, 49, and 64, which are points in one dimension. Perform a hierarchical clustering on these points as follows. Initially, each point is in a cluster by itself. At each step, merge the two clusters with the closest centroids, and continue until only two clusters remain. What are centroids of these two clusters?

27. Perform a hierarchical clustering of the following six points:

$$A(0,0), B(10,10), C(21,21), D(33, 33), E(5,27), \text{ and } F(28,6)$$

   (a) Using the *single-link* proximity measure (the distance between clusters is the shortest distance between any pair of points, one from each cluster).
   (b) Using the *complete-link* proximity measure (the distance between two clusters is the largest distance between any two points, one from each cluster).

28. Suppose that the data-mining task is to cluster the following eight points (representing location) into three clusters:

$$A1(2;10); A2(2;5); A3(8;4); B1(5;8); B2(7;5); B3(6;4); C1(1;2); C2(4;9)$$

The distance function is Euclidean distance. Suppose initially we assign $A1$, $B1$, and $C1$ as the center of each cluster, respectively. Use the *k-means* algorithm to determine:

   (a) The three cluster centers after the first round of execution.
   (b) The final three clusters.
   (c) Calculate purity of three clusters if labels of samples are already given: A, B, and C.

9.10 REFERENCES FOR FURTHER STUDY

1. Jain, A. K., M. N. Murty, P. J. Flynn. Data Clustering: A Review, *ACM Computing Surveys*, Vol. 31, No. 3, (September 1999), pp. 264–323.

   Although there are several excellent books on clustering algorithms, this review paper will give the reader enough details about the state-of-the-art techniques in data clustering, with an emphasis on large data set problems. The paper presents the taxonomy of clustering techniques and identifies crosscutting themes, recent advances, and some important applications. For readers interested in practical implementation of some clustering methods, the paper offers useful advice and a large spectrum of references.

2. 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.

3. Miyamoto S., *Fuzzy Sets in Information Retrieval and Cluster Analysis*, Kluwer Academic Publishers, Dordrecht, Germany, 1990.

   This book offers an in-depth presentation and analysis of some clustering algorithms and reviews the possibilities of combining these techniques with fuzzy representation of data. Information retrieval, which, with the development of advanced Web-mining techniques, is becoming more important in the data-mining community, is also explained in the book.

4. Han, J., M. Kamber, *Data Mining: Concepts and Techniques*, 3rd edition, Morgan Kaufmann, San Francisco, 2011.

   This book gives a sound understanding of data-mining principles. The primary orientation of the book is for database practitioners and professionals with emphasis on OLAP and data warehousing. In-depth analysis of association rules and clustering algorithms is the additional strength of the book. All algorithms are presented in easily understood pseudocode, and they are suitable for use in real-world, large-scale data-mining projects including advanced applications such as Web mining and text mining.

5. Filippone M., F. Camastra, F. Masulli, S. Rovetta, A Survey of Kernel and Spectral Methods for Clustering, *Pattern Recognition*, Vol. 41, 2008, pp. 176–190.

   Clustering algorithms are a useful tool to explore data structures and have been employed in many disciplines. The focus of this paper is the partitioning clustering problem with a special interest in two recent approaches: kernel and spectral methods. The aim of this paper is to present a survey of kernel and spectral clustering methods, two approaches able to produce nonlinear separating hypersurfaces

between clusters. The presented kernel clustering methods are the kernel version of many classical clustering algorithms, e.g. K-means, SOM, and neural gas. Spectral clustering arise from concepts in spectral graph theory, and the clustering problem is configured as a graph cut problem where an appropriate objective function has to be optimized.

6. Slawomir Wierzchoń, Mieczyslaw Kłopotek, *Modern Algorithms of Cluster Analysis*, Springer, Berlin, 2018.

The book explains feature-based, graph-based, and spectral clustering methods and discusses their formal similarities and differences. Understanding the related formal concepts is particularly vital in the epoch of big data; due to the volume and characteristics of the data, it is no longer feasible to predominantly rely on merely viewing the data when facing a clustering problem. The book addresses grid-based methods, sampling methods, parallelization via MapReduce, usage of tree structures, random projections, and various heuristic approaches, especially those used for community detection.

10

ASSOCIATION RULES

Chapter Objectives

- Explain the local modeling character of association rule techniques.
- Analyze the basic characteristics of large transactional databases.
- Describe the Apriori algorithm and explain all its phases through illustrative examples.
- Compare the frequent pattern growth method with the Apriori algorithm.
- Outline the solution for association rule generation from frequent itemsets.
- Explain discovery of multidimensional associations.
- Introduce extension of FP-growth methodology for classification problems.

When we talk about machine-learning methods applied in data mining, we may classified them as parametric or nonparametric methods. In a case of *parametric methods*, which are used for density estimation, classification, or regression, we assume that a

final model is valid over the entire input space. In regression, for example, when we derive a linear model, we apply it for all future inputs. In classification, we assume that all samples (training but also new testing) are drawn from the same density distribution. Models in these cases are global models valid for the entire n-dimensional space of samples. The advantage of a parametric method is that it reduces the problem of modeling with only a small number of parameters. Its main disadvantage is that initial assumptions do not hold in many real-world problems causing a large error. In *nonparametric estimation*, all we assume is that similar inputs have similar outputs. Methods do not assume any a priori density or parametric form. There is no single global model. Local models are estimated as they are occurred, affected only by nearby training samples (Figure 10.1).

Association rule discovery is one of the major techniques of data mining, and it is perhaps the most common form of local-pattern discovery in unsupervised learning systems. It is a form of data mining that most closely resembles the process that most people think about when they try to understand the data-mining process, namely, "mining" for gold through a vast database. The gold in this case would be a rule that is interesting, which tells you something about your database that you did not already know and probably were not able to explicitly articulate. These methodologies retrieve all possible interesting patterns in the database. This is strength in the sense that it leaves "no stone unturned." But it can be viewed also as a weakness because the user can easily become overwhelmed with a large amount of new information and an analysis of their usability is difficult and time consuming.

Besides the standard methodologies such as the *Apriori* technique for association rule mining, we will explain some extensions such as *FP-tree* and *CMAR* algorithms. All these methodologies show how important and applicable is problem of market-basket analysis and corresponding methodologies for discovery of association's rules in data.
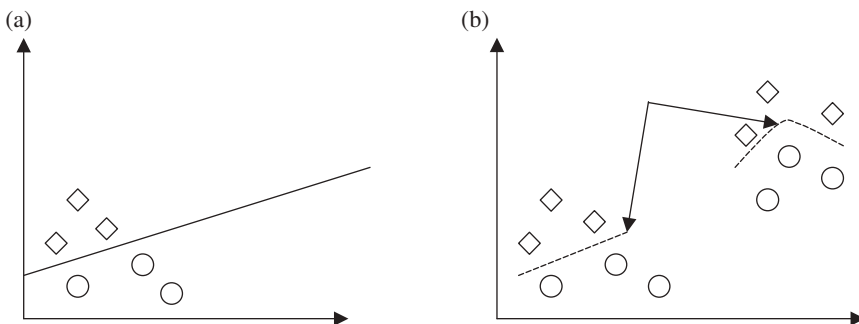

Figure 10.1. Parametric vs. nonparametric methods. (a) Parametric methods build global models. (b) Nonparametric methods result in local modeling

10.1 MARKET-BASKET ANALYSIS

A market basket is a collection of items purchased by a customer in a single transaction, which is a well-defined business activity. For example, customer's visits to a grocery store or an online purchase from a virtual store on the Web are typical customer transactions. Retailers accumulate huge collections of transactions by recording business activities over time. One common analysis run against a transaction database is to find sets of items, or *itemset*, that appear together in many transactions. A business can use knowledge of these patterns to improve the placement of these items in the store or the layout of mail-order catalog pages and Web pages. An itemset containing i items is called an *i-itemset*. The percentage of transactions that contain an itemset is called the itemset's *support*. For an itemset to be interesting, its support must be higher than a user-specified minimum. Such itemsets are said to be frequent.

Why is finding frequent itemsets a nontrivial problem? First, the number of customer transactions can be very large and usually will not fit in a central memory of a computer. Second, the potential number of frequent itemsets is exponential to the number of different items, although the actual number of frequent itemsets can be much smaller. Therefore, we want algorithms that are scalable (their complexity should increase linearly, not exponentially, with the number of transactions) and that examine as few infrequent itemsets as possible. Before we explain some of the more efficient algorithms, let us try to describe the problem more formally and develop its mathematical model.

From a database of sales transactions, we want to discover the important associations among items such that the presence of some items in a transaction will imply the presence of other items in the same transactions. Let $I = \{i_1, i_2, \ldots, i_m\}$ be a set of literals, called items. Let database DB be a set of transactions, where each transaction T is a set of items such that $T \subseteq I$. Note that the quantities of the items bought in a transaction are not considered, meaning that each item is a binary variable indicating whether an item was bought or not. Each transaction is associated with an identifier called a transaction identifier (TID). An example of the model for such a transaction database is given in Table 10.1.

Let X be a set of items. A transaction T is said to contain X if and only if $X \subseteq T$. An association rule implies the form $X \Rightarrow Y$, where $X \subseteq I$, $Y \subseteq I$, and $X \cap Y = \emptyset$. The rule $X \Rightarrow Y$

TABLE 10.1. A Model of a Simple Transaction Database

Database DB:	
TID	Items
001	*A C D*
002	*B C E*
003	*A B C E*
004	*B E*

holds in the transaction set DB with *confidence c* if *c%* of the transactions in *D* that contain *X* also contain *Y*. The rule *X⇒Y* has *support s* in the transaction set *D* if *s%* of the transactions in DB contain *X∪Y*. Confidence denotes the strength of implication, and support indicates the frequency of the patterns occurring in the rule. It is often desirable to pay attention to only those rules that may have a reasonably large support. Such rules with high confidence and strong support are referred to as *strong rules*. The task of mining association rules is essentially to discover strong association rules in large databases. The problem of mining association rules may be decomposed into two phases:

1. Discover the large itemsets, i.e., the sets of items that have transaction support s above a predetermined minimum threshold.
2. Use the large itemsets to generate the association rules for the database that have confidence c above a predetermined minimum threshold.

The overall performance of mining association rules is determined primarily by the first step. After the large itemsets are identified, the corresponding association rules can be derived in a straightforward manner. Efficient counting of large itemsets is thus the focus of most mining algorithms, and many efficient solutions have been designed to address previous criteria. The *Apriori* algorithm provided one early solution to the problem, and it will be explained in greater detail in this chapter. Other subsequent algorithms built upon the *Apriori* algorithm represent refinements of a basic solution, and they are explained in a wide spectrum of articles including texts recommended in Section 10.2.

10.2   ALGORITHM *APRIORI*

The algorithm *Apriori* computes the frequent itemsets in the database through several iterations. Iteration *i* computes all frequent *i*-itemsets (itemsets with *i* elements). Each iteration has two steps: *candidate generation* and *candidate counting and selection*.

In the first phase of the first iteration, the generated set of candidate itemsets contains all 1-itemsets (i.e., all items in the database). In the counting phase, the algorithm counts their support searching again through the whole database. Finally, only 1-itemsets (items) with s above required threshold will be selected as frequent. Thus, after the first iteration, all frequent 1-itemsets will be known.

What are the itemsets generated in the second iteration? In other words, how does one generate 2-itemset candidates? Basically, all pairs of items are candidates. Based on knowledge about infrequent itemsets obtained from previous iterations, the *Apriori* algorithm reduces the set of candidate itemsets by pruning—*a priori*—those candidate itemsets that cannot be frequent. The pruning is based on the observation that if an itemset is frequent, all its subsets could be frequent as well. Therefore, before entering the candidate-counting step, the algorithm discards every candidate itemset that has an infrequent subset.

(a)

1-Itemsets $\mathbf{C_1}$
{A}
{C}
{D}
{B}
{E}

(b1)

1-Itemsets	Count	s[%]
{A}	2	50
{C}	3	75
{D}	1	25
{B}	3	75
{E}	3	75

(b2)

Large 1-itemsets $\mathbf{L_1}$	Count	s[%]
{A}	2	50
{C}	3	75
{B}	3	75
{E}	3	75

Figure 10.2. First iteration of the Apriori algorithm for database DB. (a) Generate phase. (b1) Count phase. (b2) Select phase.

Consider the database in Table 10.1. Assume that the minimum support $s = 50\%$; so an itemset is frequent if it is contained in at least 50% of the transactions—in our example, in two out of every four transactions in the database. In each iteration, the *Apriori* algorithm constructs a candidate set of large itemsets, counts the number of occurrences of each candidate, and then determines large itemsets based on the predetermined minimum support $s = 50\%$.

In the first step of the first iteration, all single items are candidates. *Apriori* simply scans all the transactions in a database DB and generates a list of candidates. In the next step, the algorithm counts the occurrences of each candidate and based on threshold s selects frequent itemsets. All these steps are given in Figure 10.2. Five 1-itemsets are generated in C_1, and, of these, only four are selected as large in L_1 because their support is greater than or equal to 2, or $s \geq 50\%$.

To discover the set of large 2-itemsets, because any subset of a large itemset could also have minimum support, the *Apriori* algorithm uses $L_1{}^*L_1$ to generate the candidates. The operation $*$ is defined in general as

$$L_k{}^* L_k = \{X \cup Y \text{ where } X, Y \in L_k, |X \cap Y| = k - 1\}$$

For $k = 1$ the operation represents a simple concatenation. Therefore, C_2 consists of 2-itemsets generated by the operation $|L_1| \cdot (|L_1| - 1)/2$ as candidates in the second iteration. In our example, this number is $4 \cdot 3/2 = 6$. Scanning the database DB with this list, the algorithm counts the support for every candidate and in the end selects a large 2-itemsets L_2 for which $s \geq 50\%$. All these steps and the corresponding results of the second iteration are given in Figure 10.3.

The set of candidate itemsets C_3 is generated from L_2 using the previously defined operation $L_2{}^*L_2$. Practically, from L_2, two large 2-itemsets with the same first item, such as {B, C} and {B, E}, are identified first. Then, *Apriori* tests whether the 2-itemset {C, E}, which consists of the second items in the sets {B, C} and {B, E}, constitutes a large 2-itemset or not. Because {C, E} is a large itemset by itself, we know that all the subsets of {B, C, E} are large and then {B, C, E} becomes a candidate 3-itemset. There is no other candidate 3-itemset from L_2 in our database

(a)

2-Itemsets C_2
$\{A, B\}$
$\{A, C\}$
$\{A, E\}$
$\{B, C\}$
$\{B, E\}$
$\{C, E\}$

(b1)

2-Itemsets	Count	s[%]
$\{A, B\}$	1	25
$\{A, C\}$	2	50
$\{A, E\}$	1	25
$\{B, C\}$	2	50
$\{B, E\}$	3	75
$\{C, E\}$	2	50

(b2)

Large 2-Itemsets L_2	Count	s[%]
$\{A, C\}$	2	50
$\{B, C\}$	2	50
$\{B, E\}$	3	75
$\{C, E\}$	2	50

Figure 10.3. Second iteration of the Apriori algorithm for database DB. (a) Generate phase. (b1) Count phase. (b2) Select phase.

(a)

3-Itemsets C_3
$\{B, C, E\}$

(b1)

3-Itemsets	Count	s[%]
$\{B, C, E\}$	2	50

(b2)

Large 3-Itemsets L_3	Count	s[%]
$\{B, C, E\}$	2	50

Figure 10.4. Third iteration of the Apriori algorithm for database DB. (a) Generate phase. (b1) Count phase. (b2) Select phase.

DB. *Apriori* then scans all the transactions and discovers the large 3-itemsets L_3, as shown in Figure 10.4.

In our example, because there is no 4-itemset candidate to be constituted from L_3, *Apriori* ends the iterative process.

Apriori counts not only the support of all frequent itemsets but also the support of those infrequent candidate itemsets that could not be eliminated during the pruning phase. The set of all candidate itemsets that are infrequent but whose support is counted by *Apriori* is called the *negative border*. Thus, an itemset is in the negative border if it is infrequent, but all its subsets are frequent. In our example, analyzing Figures 8.1 and 8.2, we can see that the negative border consists of itemsets $\{D\}$, $\{A, B\}$, and $\{A, E\}$. The negative border is especially important for some improvements in the *Apriori* algorithm such as increased efficiency in generation of large itemsets.

10.3 FROM FREQUENT ITEMSETS TO ASSOCIATION RULES

The second phase in discovering association rules based on all frequent *i*-itemsets, which have been found in the first phase using the *Apriori* or some other similar algorithm, is relatively simple and straightforward. For a rule that implies $\{x_1, x_2, x_3\} \rightarrow x_4$, it is necessary that both itemsets $\{x_1, x_2, x_3, x_4\}$ and $\{x_1, x_2, x_3\}$ are frequent. Then, the confidence c of the rule is computed as the quotient of supports for the itemsets $c = s$

$(x_1, x_2, x_3, x_4)/s(x_1, x_2, x_3)$. Strong association rules are rules with a confidence value c above a given threshold.

For our example of database DB in Table 10.1, if we want to check whether the association rule $\{B, C\} \rightarrow E$ is a strong rule, first we select the corresponding supports from tables L_2 and L_3:

$$s(B,C) = 2, \quad s(B,C,E) = 2$$

and using these supports, we compute the confidence of the rule:

$$c(\{B,C\} \rightarrow E) = \frac{s(B,C,E)}{s(B,C)} = \frac{2}{2} = 1 \text{ (or 100\%)}$$

Whatever the selected threshold for strong association rules (for example, $c_T = 0.8$ or 80%), this rule will pass because its confidence is maximal, i.e. if a transaction contains items B and C, it will also contain item E. Other rules are also possible for our database DB, such as $A \rightarrow C$ because $c(A \rightarrow C) = s(A, C)/s(A) = 1$, and both itemsets $\{A\}$ and $\{A, C\}$ are frequent based on the *Apriori* algorithm. Therefore, in this phase, it is necessary only to systematically analyze all possible association rules that could be generated from the frequent itemsets and select as strong association rules those that have a confidence value above a given threshold.

Notice that not all the discovered strong association rules (i.e., passing the required support s and required confidence c) are interesting enough to be presented and used. For example, consider the following case of mining the survey results in a school of 5000 students. A retailer of breakfast cereal surveys the activities that the students engage in every morning. The data show that 60% of the students (i.e., 3000 students) play basketball, 75% of the students (i.e., 3750 students) eat cereal, and 40% of them (i.e., 2000 students) play basketball and also eat cereal. Suppose that a data-mining program for discovering association rules is run on the following settings: the minimal support is 2000 ($s = 0.4$) and the minimal confidence is 60% ($c = 0.6$). The following association rule will be produced: "(play basketball) $\rightarrow$ (eat cereal)," since this rule contains the minimal student support and the corresponding confidence $c = 2000/3000 = 0.66$ is larger than the threshold value. However, the above association rule is misleading since the overall percentage of students eating cereal is 75%, larger than 66%. That is, playing basketball and eating cereal are in fact negatively associated. Being involved in one itemset decreases the likelihood of being involved in the other. Without fully understanding this aspect, one could make wrong business or scientific decisions from the association rules derived.

To filter out such misleading associations, one may define that an association rule $A \rightarrow B$ is *interesting* if its confidence exceeds a certain measure. The simple argument we used in the example above suggests that the right heuristic to measure association should be

$$\frac{s(A,B)}{s(A)} - s(B) > d$$

or alternatively

$$s(A,B) - s(A) \cdot s(B) > k$$

where d or k are suitable constants. The expressions above essentially represent tests of statistical independence. Clearly, the factor of statistical dependence among analyzed itemsets has to be taken into consideration to determine the usefulness of association rules. In our simple example with students, this test fails for the discovered association rule:

$$s(A,B) - s(A) \cdot s(B) = 0.4 - 0.6 \cdot 0.75 = -0.05 < 0$$

and, therefore, despite high values for parameters s and c, the rule is not interesting. In this case, it is even misleading.

10.4   IMPROVING THE EFFICIENCY OF THE *APRIORI* ALGORITHM

Since the amount of the processed data in mining frequent itemsets tends to be huge, it is important to devise efficient algorithms to mine such data. Our basic *Apriori* algorithm scans the database several times, depending on the size of the largest frequent itemset. Since Apriori algorithm was first introduced and as experience was accumulated, there have been many attempts to devise more efficient algorithms of frequent itemset mining including approaches such as hash-based technique, partitioning, sampling, and using vertical data format. Several refinements have been proposed that focus on reducing the number of database scans, the number of candidate itemsets counted in each scan, or both.

   Partition-based Apriori is an algorithm that requires only two scans of the transaction database. The database is divided into disjoint partitions, each small enough to fit into available memory. In a first scan, the algorithm reads each partition and computes locally frequent itemsets on each partition. In the second scan, the algorithm counts the support of all locally frequent itemsets toward the complete database. If an itemset is frequent with respect to the complete database, it must be frequent in at least one partition. That is the heuristics used in the algorithm. Therefore, the second scan through the database counts itemset's frequency only for a union of all locally frequent itemsets. This second scan directly determines all frequent itemsets in the database as a subset of previously define union.

   In some applications, the transaction database has to be mined frequently to capture customer behavior. In such applications, the efficiency of data mining could be a more important factor than the complete accuracy of the results. In addition, in some applications the problem domain may be vaguely defined. Missing some marginal cases that have confidence and support levels at the borderline may have little effect on the quality of the solution to the original problem. Allowing imprecise results can in fact significantly improve the efficiency of the applied mining algorithm.

As the database size increases, *sampling* appears to be an attractive approach to data mining. A sampling-based algorithm typically requires two scans of the database. The algorithm first takes a sample from the database and generates a set of candidate itemsets that are highly likely to be frequent in the complete database. In a subsequent scan over the database, the algorithm counts these itemsets' exact support and the support of their negative border. If no itemset in the negative border is frequent, then the algorithm has discovered all frequent itemsets. Otherwise, some superset of an itemset in the negative border could be frequent, but its support has not yet been counted. The sampling algorithm generates and counts all such potentially frequent itemsets in subsequent database scans.

Because it is costly to find frequent itemsets in large databases, *incremental updating* techniques should be developed to maintain the discovered frequent itemsets (and corresponding association rules) so as to avoid mining the whole updated database again. Updates on the database may not only invalidate some existing frequent itemsets but also turn some new itemsets into frequent ones. Therefore, the problem of maintaining previously discovered frequent itemsets in large and dynamic databases is nontrivial. The idea is to reuse the information of the old frequent itemsets and to integrate the support information of the new frequent itemsets in order to substantially reduce the pool of candidates to be reexamined.

In many applications, interesting associations among data items often occur at a relatively high *concept level*. For example, one possible hierarchy of food components is presented in Figure 10.5, where M (milk) and B (bread), as concepts in the hierarchy, may have several elementary subconcepts. The lowest-level elements in the hierarchy ($M_1, M_2, \ldots, B_1, B_2, \ldots$) are types of milk and bread defined with its bar code in the store. The purchase patterns in a transaction database may not show any substantial regularities at the elementary data level, such as at the bar-code level ($M_1, M_2, M_3, B_1, B_2, \ldots$), but may show some interesting regularities at some high concept level(s), such as milk M and bread B.

Consider the class hierarchy in Figure 10.5. It could be difficult to find high support for purchase patterns at the primitive-concept level, such as chocolate milk and wheat bread. However, it would be easy to find in many databases that more than 80% of customers who purchase milk may also purchase bread. Therefore, it is important to
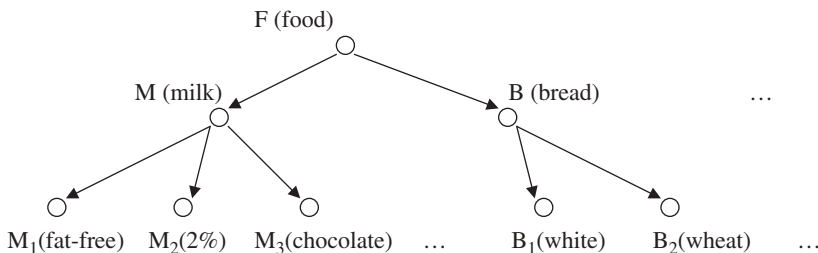
Figure 10.5. An example of concept hierarchy for mining multiple-level frequent itemsets.

mine frequent itemsets at a generalized abstraction level or at multiple-concept levels; these requirements are supported by the *Apriori* generalized-data structure.

One extension of the *Apriori* algorithm considers an *is-a* hierarchy on database items, where information about multiple abstraction levels already exist in the database organization. An *is-a* hierarchy defines which items are a specialization or generalization of other items. The extended problem is to compute frequent itemsets that include items from different hierarchy levels. The presence of a hierarchy modifies the notation of when an item is contained in a transaction. In addition to the items listed explicitly, the transaction contains their ancestors in the taxonomy. This allows the detection of relationships involving higher hierarchy levels, since an itemset's support can increase if an item is replaced by one of its ancestors.

10.5 FREQUENT PATTERN GROWTH METHOD

Let us define one of the most important problems with scalability of the *Apriori* algorithm. To generate one frequent pattern of length 100, such as $\{a_1, a_2,…,a_{100}\}$, the number of candidates that has to be generated will be at least

$$\sum_{i=1}^{100} \binom{100}{i} = 2^{100} - 1 \approx 10^{30}$$

and it will require hundreds of database scans. The complexity of the computation increases exponentially! That is only one of the many factors that influence the development of several new algorithms for association rule mining.

Frequent pattern growth (FP-growth) method is an efficient way of mining frequent itemsets in large databases. The algorithm mines frequent itemsets without the time-consuming candidate-generation process that is essential for *Apriori*. When the database is large, FP-growth first performs a database projection of the frequent items; it then switches to mining the main memory by constructing a compact data structure called the FP-tree. For an explanation of the algorithm, we will use the transactional database in Table 10.2 and the minimum support threshold of 3.

TABLE 10.2. The Transactional Database *T*

TID	Itemset
01	*f, a, c, d, g, i, m, p*
02	*a, b, c, f, l, m, o*
03	*b, f, h, j, o*
04	*b, c, k, s, p*
05	*a, f, c, e, l, p, m, n*

First, a scan of the database T derives a list L of frequent items occurring three or more than three times in the database. These are the items (with their supports):

$$L = \{(f,4),(c,4),(a,3),(b,3),(m,3),(p,3)\}$$

The items in L are listed in descending order of frequency. This ordering is important since each path of the FP-tree will follow this order.

Second, the root of the tree, labeled ROOT, is created. The database T is scanned a second time. The scan of the first transaction leads to the construction of the first branch of the FP-tree: $\{(f,1), (c,1), (a,1), (m,1), (p,1)\}$. Only those items that are in the list of frequent items L are selected. The indices for nodes in the branch (all are 1) represent the cumulative number of samples at this node in the tree, and of course, after the first sample, all are 1. The order of the nodes is not as in the sample but as in the list of frequent items L. For the second transaction, because it shares items f, c, and a, it shares the prefix $\{f, c, a\}$ with the previous branch and extends to the new branch $\{(f, 2), (c, 2), (a, 2), (m, 1), (p, 1)\}$, increasing the indices for the common prefix by one. The new intermediate version of the FP-tree, after two samples from the database, is given in Figure 10.6a. The remaining transactions can be inserted similarly, and the final FP-tree is given in Figure 10.6b.

To facilitate tree traversal, an *item header table* is built, in which each item in the list L connects nodes in the FP-tree with the same values through node links. All f nodes are connected in one list, all c nodes in the other, etc. For simplicity of representation only the list for b nodes is given in Figure 10.6b. Using the compact-tree structure, the FP-growth algorithm mines the complete set of frequent itemsets.

According to the list L of frequent items, the complete set of frequent itemsets can be divided into subsets (six for our example) without overlap: (1) frequent itemsets having item p (the end of list L); (2) the itemsets having item m but not p; (3) the frequent itemsets with b and without both m and p; (4) the frequent itemsets with a and without both b, m and p; (5) the frequent itemsets with c and without both a, b, m and p; (6) the large itemsets only with f. This classification is valid for our example, but the same principles can be applied for other databases and other L lists.
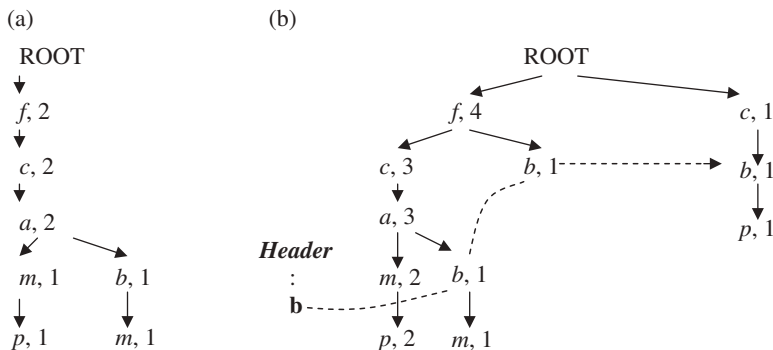
Figure 10.6. FP-tree for the database T in Table 10.2. (a) FP-tree after two samples. (b) Final FP-tree.

Based on node-link connection, we collect all the transactions that p participates in by starting from the header table of p and following p's node links. In our example, two paths will be selected in the FP-tree: $\{(f,4), (c,3), (a,3), (m,2), (p,2)\}$ and $\{(c,1), (b,1), (p,1)\}$, where samples with a frequent item p are $\{(f,2), (c,2), (a,2), (m,2), (p,2)$ and $\{(c,1), (b,1), (p,1)\}$. The given threshold value (3) satisfies only the frequent itemsets $\{(c,3), (p,3)\}$ or the simplified $\{c, p\}$. All other itemsets with p are below the threshold value.

The next subsets of frequent itemsets are those with m and without p. The FP-tree recognizes the paths $\{(f,4), (c,3), (a,3), (m,2)\}$ and $\{(f,4), (c,3), (a,3), (b,1), (m,1)\}$ or the corresponding accumulated samples $\{(f,2), (c,2), (a,2), (m,2)\}$ and $\{(f,1), (c,1), (a,1), (b,1), (m,1)\}$. Analyzing the samples we discover the frequent itemset $\{(f,3), (c,3), (a,3), (m,3)\}$ or, simplified, $\{f, c, a, m\}$.

Repeating the same process for subsets 3–6 in our example, additional frequent itemsets could be mined. These are itemsets $\{f, c, a\}$ and $\{f, c\}$, but they are already subsets of the frequent itemset $\{f, c, a, m\}$. Therefore, the final solution in the FP-growth method is the set of frequent itemsets, which is, in our example, $\{\{c, p\}, \{f, c, a, m\}\}$.

Experiments have shown that the FP-growth algorithm is faster than the *Apriori* algorithm by about one order of magnitude. Several optimization techniques are added to the FP-growth algorithm, and there exists its versions for mining sequences and patterns under constraints.

10.6   ASSOCIATIVE-CLASSIFICATION METHOD

Classification based on multiple association rules (CMAR) is a classification method adopted from the frequent pattern growth (FP-growth) method for generation of frequent itemsets. The main reason we included CMAR methodology in this chapter is its FP-growth roots, but there is the possibility of comparing CMAR accuracy and efficiency with the C4.5 methodology.

Suppose data samples are given with n attributes $(A_1, A_2, \ldots, A_n)$. Attributes can be categorical or continuous. For a continuous attribute, we assume that its values are discretized into intervals in the preprocessing phase. A training data set T is a set of samples such that for each sample there exists a class label associated with it. Let $C = \{c_1, c_2, \ldots, c_m\}$ be a finite set of *class labels*.

In general, a pattern $P = \{a_1, a_2, \ldots, a_k\}$ is a set of attribute values for different attributes $(1 \leq k \leq n)$. A sample is said to match the pattern P if it has all the attribute values given in the pattern. For rule $R: P \rightarrow c$, the number of data samples matching pattern P and having class label c is called the *support* of rule R, denoted sup(R). The ratio of the number of samples matching pattern P and having class label c versus the total number of samples matching pattern P is called the *confidence* of R, denoted as conf(R). The associative-classification method (CMAR) consists of two phases:

1. Rule generation or training and
2. Classification or testing.

TABLE 10.3. Training Database T for the CMAR Algorithm

ID	A	B	C	D	Class
01	a_1	b_1	c_1	d_1	A
02	a_1	b_2	c_1	d_2	B
03	a_2	b_3	c_2	d_3	A
04	a_1	b_2	c_3	d_3	C
05	a_1	b_2	c_1	d_3	C

In the first rule generation phase, CMAR computes the complete set of rules in the form R: $P \rightarrow c$, such that sup(R) and conf(R) pass the given thresholds. For a given support threshold and confidence threshold, the associative-classification method finds the complete set of class-association rules (CAR) passing the thresholds. In a testing phase, when a new (unclassified) sample comes, the classifier, represented by a set of association rules, selects the rule that matches the sample and has the highest confidence and uses it to predict the classification of the new sample.

We will illustrate the basic steps of the algorithm through one simple example. Suppose that for a given training data set T, as shown in Table 10.3, the support threshold is 2 and the confidence threshold is 70%.

First, CMAR scans the training data set and finds the set of attribute values occurring beyond the threshold support (at least twice in our database). One simple approach is to sort each attribute and to find all frequent values. For our database T, this is a set $F = \{a_1, b_2, c_1, d_3\}$ and it is called a frequent item set. All other attribute values fail the support threshold. Then, CMAR sorts attribute values in F, in support-descending order, that is, F-list $= (a_1, b_2, c_1, d_3)$.

Now, CMAR scans the training data set again to construct an FP-tree. The FP-tree is a prefix tree with respect to the F-list. For each sample in a training data set, attribute values appearing in the F-list are extracted and sorted according to the order in F-list. For example, for the first sample in database T, (a_1, c_1) are extracted and inserted in the tree as the leftmost branch in the tree. The class label A of the sample and the corresponding counter are attached to the last node in the path.

Samples in the training data set share prefixes. For example, the second sample carries attribute values (a_1, b_2, c_1) in the F-list and shares a common prefix a_1 with the first sample. An additional branch from the node a_1 will be inserted in the tree with new nodes b_2 and c_1. A new class label B with the count equal to 1 is also inserted at the end of the new path. The final FP-tree for the database T is given in Figure 10.7a.

After analyzing all the samples and constructing an FP-tree, the set of class-association rules can be generated dividing all rules into subsets without overlap. In our example it will be four subsets: (1) the rules having d_3 value, (2) the rules having c_1 but no d_3, (3) the rules having b_2 but neither d_3 nor c_1, and (4) the rules having only a_1. CMAR find these subsets one by one.

To find the subset of rules having d_3, CMAR traverses nodes having the attribute value d_3 and looks "upward" the FP-tree to collect d_3-projected samples. In our example, there are three samples represented in the FP-tree, and they are (a_1, b_2, c_1, d_3):
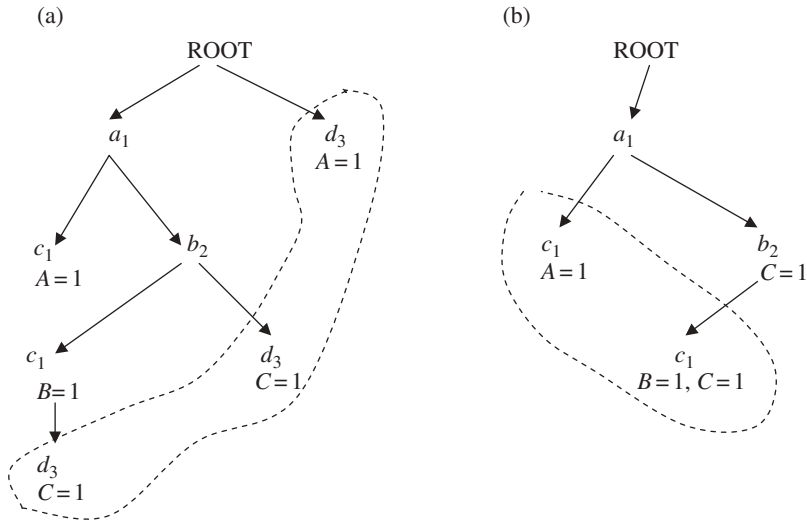
(a)

ROOT

(b)

ROOT

Figure 10.7. FP-tree for the database in Table 10.3. (a) Nonmerged FP-tree. (b) FP-tree after merging d_3 nodes

C, (a_1, b_2, d_3):C, and (d_3):A. The problem of finding all frequent patterns in the training set can be reduced to mining frequent patterns in the d_3-projected database. In our example, in the d_3-projected database, since the pattern (a_1, b_2, d_3) occurs twice, its support is equal to the required threshold value 2. Also, the rule based on this frequent pattern, $(a_1, b_2, d_3) \rightarrow C$, has a confidence 100% (above the threshold value), and that is the only rule generated in the given projection of the database.

After a search for rules having d_3 value, all the nodes of d_3 and their corresponding class labels are merged into their parent nodes of the FP-tree. The FP-tree is shrunk as shown in Figure 10.7b. The remaining set of rules can be mined similarly repeating the previous procedures for a c_1-projected database, then for the b_2-projected database, and finally for the a_1-projected database. In this analysis, (a_1, c_1) is a frequent pattern with support 3, but all rules are with confidence less than threshold value. The same conclusions can be drawn for pattern (a_1, b_2) and for (a_1). Therefore, the only association rule generated through the training process with the database T is $(a_1, b_2, d_3) \rightarrow C$ with support equal to 2 and 100% confidence.

When a set of rules is selected for classification, CMAR is ready to classify new samples. For the new sample, CMAR collects the subset of rules matching the sample from the total set of rules. Trivially, if all the rules have the same class, CMAR simply assigns that label to the new sample. If the rules are not consistent in the class label, CMAR divides the rules into groups according to the class label and yields the label of the "strongest" group. To compare the strength of groups, it is necessary to measure the "combined effect" of each group. Intuitively, if the rules in a group are highly positively correlated and have good support, the group should have a strong effect. CMAR uses the strongest rule in the group as its representative, i.e., the rule with

highest χ^2 test value (adopted for this algorithm for a simplified computation). Preliminary experiments have shown that CMAR outperforms the C4.5 algorithm in terms of average accuracy, efficiency, and scalability.

10.7 MULTIDIMENSIONAL ASSOCIATION RULE MINING

A multidimensional transactional database DB has the schema

$$(ID, A_1, A_2, \ldots, A_n, \text{items})$$

where ID is a unique identification of each transaction, A_i are structured attributes in the database, and items are sets of items connected with the given transaction. The information in each tuple $t = (id, a_1, a_2, \ldots, a_n, \text{items-}t)$ can be partitioned into two: dimensional part $(a_1, a_2, \ldots, a_n)$ and itemset part (items-t). It is common sense to divide the mining process into two steps: first mine patterns about dimensional information and then find frequent itemsets from the projected subdatabase, or vice versa. Without any preferences in the methodology, we will illustrate the first approach using the multidimensional database DB in Table 10.4.

One can first find the frequent multidimensional value combinations and then find the corresponding frequent itemsets of a database. Suppose that the threshold value for our database DB in Table 10.4 is set to 2. Then, the combination of attribute values that occurs two or more than two times is frequent, and it is called a multidimensional pattern or MD-pattern. For mining MD-patterns, a modified bottom-up computation (BUC) algorithm can be used; it is an efficient "iceberg cube" computing algorithm. The basic steps of the BUC algorithm are as follows:

1. First, sort all tuples in the database in alphabetical order of values in the first dimension (A_1), because the values for A_1 are categorical. The only MD-pattern found for this dimension is $(a, {}^*, {}^*)$ because only the value a occurs two times; the other values b and c occur only once, and they are not part of the MD-patterns. Value * for the other two dimensions shows that they are not relevant in this first step, and they could have any combination of allowed values.

TABLE 10.4. Multidimensional Transactional Database DB

ID	A_1	A_2	A_3	Items
01	a	1	m	x, y, z
02	b	2	n	z, w
03	a	2	m	x, z, w
04	c	3	p	x, w

Select tuples in a database with found MD-pattern (or patterns). In our database, these are the samples with ID values 01 and 03. Sort the reduced database again with respect to the second dimension (A_2), where the values are 1 and 2. Since no pattern occurs twice, there are no MD-patterns for exact A_1 and A_2 values. Therefore, one can ignore the second dimension A_2 (this dimension does not reduce the database further). All selected tuples are used in the next phase.

Selected tuples in the database are sorted in alphabetic order of values for the third dimension (in our example A_3 with categorical values). A subgroup $(a, *, m)$ is contained in two tuples and it is an MD-pattern. Since there are no more dimensions in our example, the search continues with the second step.

2. Repeat the processes in step 1; only start not with the first but with the second dimension (first dimension is not analyzed at all in this iteration). In the following iterations, reduce the search process further for one additional dimension at the beginning. Continue with other dimensions.

In our example in the second iteration, starting with attribute A_2, MD-pattern $(*, 2, *)$ will be found. Including dimension A_3, there are no additional MD-patterns. The third and last iteration in our example starts with the A_3 dimension, and the corresponding pattern is $(*, *, m)$.

In summary, the modified BUC algorithm defines a set of MD-patterns with the corresponding projections of a database. The processing tree for our example of database DB is shown in Figure 10.8. Similar trees will be generated for a larger number of dimensions.

When all MD-patterns are found, the next step in the analysis of multidimensional transactional database is the mining of frequent itemsets in the MD-projected database for each MD-pattern. An alternative approach is based on finding frequent itemsets first and then the corresponding MD-patterns.
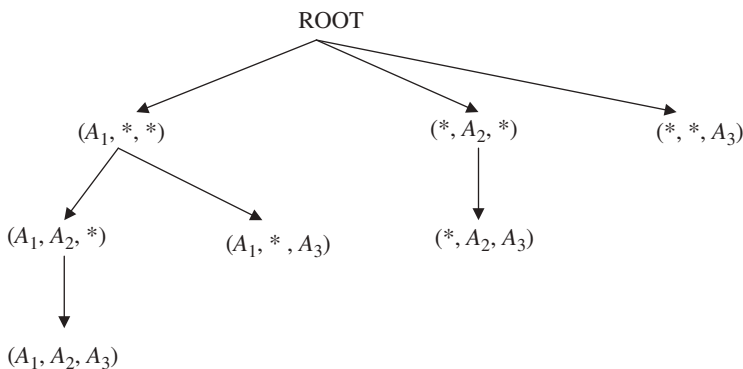
Figure 10.8. A processing tree using the BUC algorithm for the database in Table 10.4.

10.8   REVIEW QUESTIONS AND PROBLEMS

1. What is the essential difference between association rules and decision rules (described in Chapter 6)?

2. What are the typical industries in which market-basket analysis plays an important role in the strategic decision-making processes?

3. What are the common values for support and confidence parameters in the *Apriori* algorithm? Explain using the retail industry as an example.

4. Why is the process of discovering association rules relatively simple compared to generating large itemsets in transactional databases?

5. Given a simple transactional database X:

X:	TID	Items
	T01	A, B, C, D
	T02	A, C, D, F
	T03	C, D, E, G, A
	T04	A, D, F, B
	T05	B, C, G
	T06	D, F, G
	T07	A, B, G
	T08	C, D, F, G

Using the threshold values support = 25% and confidence = 60%, find:
   (a) All large itemsets in database X.
   (b) Strong association rules for database X.
   (c) Analyze misleading associations for the rule set obtained in (b).

6. Given a transactional database Y:

Y:	TID	Items
	T01	A1, B1, C2
	T02	A2, C1, D1
	T03	B2, C2, E2
	T04	B1, C1, E1
	T05	A3, C3, E2
	T06	C1, D2, E2

Using the threshold values for support s = 30% and confidence c = 60%:
   (a) Find all large itemsets in database Y.

(b) If itemsets are organized in a hierarchy so that $A = \{A1, A2, A3\}$, $B = \{B1, B2\}$, $C = \{C1, C2, C3\}$, $D = \{D1, D2\}$, and $E = \{E1, E2\}$, find large itemsets that are defined on conceptual level including a hierarchy of items.

(c) Find strong association rules for large itemsets in (b).

7. Implement the *Apriori* algorithm, and discover large itemsets in transactional database.

8. Search the Web to find the basic characteristics of publicly available or commercial software tools for association rule discovery. Document the results of your search.

9. Given a simple transactional database, find FP-tree for this database if:

(a) Support threshold is 5.
(b) Support threshold is 3.

TID	Items
1	*a b c d*
2	*a c d f*
3	*c d e g a*
4	*a d f b*
5	*b c g*
6	*d f g*
7	*a b g*
8	*c d f g*

10. Given a simple transaction database:

TID	Items
1	*X Z V*
2	*X Y U*
3	*Y Z V*
4	*Z V W*

Using two iterations of the *Apriori* algorithm, find large 2-itemsets if required support is $s \geq 50\%$. Show all steps of the algorithm.

11. Given a frequent itemset *A, B, C, D,* and *E*, how many possible association rules exist?

12. What are the frequent itemsets with a minimum support of 3 for the given set of transactions?

TID	Items
101	*A,B,C,D,E*
102	*A,C,D*
103	*D,E*
104	*B,C,E*
105	*A,B,D,E*
106	*A,B*
107	*B,D,E*
108	*A,B,D*
109	*A,D*
110	*D,E*

13. The *conviction* is a measure for an analysis of a quality of association rules. The formula for *conviction* CV in terms of probabilities is given as

$$\text{CV}(A \rightarrow B) = \frac{[P(A) \cdot P(B')]}{P(A,B')}$$

or, in terms of support and confidence of an association rule,

$$\text{CV}(A \rightarrow B) = \frac{[1 - \sup(B)]}{[1 - \text{conf}(A \rightarrow B)]}.$$

What are basic characteristics of the *conviction* measure? Explain the meaning of some characteristic values.

14. Consider the data set given in the table below.

Customer Id	Transaction Id	Items
418	234145	{*X, Z*}
345	543789	{*U, V, W, X, Y, Z*}
323	965157	{*U, W, Y*}
418	489651	{*V, X, Z*}
567	748965	{*U, Y*}
567	325687	{*W, X, Y*}
323	147895	{*X, Y, Z*}
635	617851	{*U, Z*}
345	824697	{*V, Y*}
635	102458	{*V, W, X*}

(a) Compute the support for item sets {*Y*}, {*X, Z*}, and {*X, Y, Z*} by treating each transaction ID as a market basket.

(b) Use the results from part (a) to compute the confidence for rules $XZ{\rightarrow}Y$ and $Y{\rightarrow}XZ$.

(c) Repeat part (a) by treating each customer ID as market basket. Each item should be treated as a binary variable (1 if an item appears in at least one transaction bought by the customer and 0 otherwise).

(d) Use the results from part (c) to compute the confidence for rules $XZ{\rightarrow}Y$ and $Y{\rightarrow}XZ$.

(e) Find FP-tree for this database if support threshold is 5.

15. A collection of market-basket data has 100,000 frequent items and 1,000,000 infrequent items. Each pair of frequent items appears 100 times, each pair consisting of one frequent and one infrequent item appears 10 times, and each pair of infrequent items appears once. Answer each of the following questions. Your answers only have to be correct to within 1%, and for convenience, you may optionally use scientific notation, e.g., 3.14×10^8 instead of 314,000,000.

(a) What is the total number of pair occurrences? That is, what is the sum of the counts of all pairs?

(b) We did not state the support threshold, but the given information lets us put bounds on the support threshold s. What are the tightest upper and lower bounds on s?

16. Assume that we have a data set containing information about 200 individuals. One hundred of these individuals have purchased life insurance. A supervised data-mining session has discovered the following rule:

IF age < 30 & credit card insurance = yes

THEN life insurance = yes

(*Rule Accuracy = 70%, Rule Coverage = 63%*)

How many individuals in the class *life insurance = no* have credit card insurance and are less than 30 years old?

17. Assume that the numbers 1 through 7 are items.

(a) Which of the following five association rules has a confidence that is certain to be at least as great as the confidence of the rule 12=>34567 and no greater than the confidence of the rule 1234=>5?

Association rules : $134 = >257, 124 = >357, 134 = >567, 123 = >457, 124 = >356$

(b) Explain the general characteristics of the rules that are satisfying required constraints.

18. Consider the transaction database in the table below:

TID	Items
1	*a, b, c, d*
2	*b, c, e, f*
3	*a, d, e, f*
4	*a, e, f*
5	*b, d, f*

(a) Determine the absolute support of itemsets {*a, e, f*} and {*d, f*}. Convert the absolute support to the relative support.

(b) Show the prefix-based enumeration tree of frequent itemsets, for the data set at an absolute minimum support level of 2. Assume a lexicographic ordering of *a, b, c, d, e, f*.

10.9 REFERENCES FOR FURTHER STUDY

1. Adamo, J., *Data Mining for Association Rules and Sequential Patterns*, Springer, Berlin, 2001.

   This book presents a collection of algorithms for data mining on the lattice structure of the feature space. Given the computational complexity and time requirements of mining association rules and sequential patterns, the design of efficient algorithms is critical. Most algorithms provided in the book are designed for both sequential and parallel execution, and they support sophisticated data mining of large-scale transactional databases.

2. Han, J., M. Kamber, *Data Mining: Concepts and Techniques*, 3rd edition, Morgan Kaufmann, San Francisco, 2011.

   This book gives a sound understanding of data-mining principles. The primary orientation of the book is for database practitioners and professionals with emphasis on OLAP and data warehousing. In-depth analysis of association rules and clustering algorithms is the additional strength of the book. All algorithms are presented in easily understood pseudocode, and they are suitable for use in real-world, large-scale data-mining projects including advanced applications such as Web mining and text mining.

3. Goethals B., Frequent Set Mining, in *Data Mining and Knowledge Discovery Handbook*, Springer, 2005, pp. 377–397.

   Frequent sets lie at the basis of many data-mining algorithms. As a result, hundreds of algorithms have been proposed in order to solve the frequent set mining problem. In this chapter, we attempt to survey the most successful algorithms and techniques that try to solve this problem efficiently. During the first ten years after the proposal of the frequent set mining problem, several hundreds of scientific papers were written on the topic, and it seems that this trend is keeping its pace.

4. Cheng J., Y. Ke, W. Ng, A Survey on Algorithms for Mining Frequent Itemsets over Data Streams, *Knowledge and Information Systems*, Vol. 16, No. 1, July, 2008, pp. 1–27.

   The increasing prominence of data streams arising in a wide range of advanced applications such as fraud detection and trend learning has led to the study of online mining of frequent itemsets (FIs). Unlike mining static databases, mining data streams poses many new challenges. In addition to the one-scan nature, the unbounded memory requirement, and the high data arrival rate of data streams,

the combinatorial explosion of itemsets exacerbates the mining task. The high complexity of the FI mining problem hinders the application of the stream mining techniques. We recognize that a critical review of existing techniques is needed in order to design and develop efficient mining algorithms and data structures that are able to match the processing rate of the mining with the high arrival rate of data streams. Within a unifying set of notations and terminologies, we describe in this paper the efforts and main techniques for mining data streams and present a comprehensive survey of a number of the state-of-the-art algorithms on mining frequent itemsets over data streams.

5. Kumar P., *Pattern Discovery Using Sequence Data Mining: Applications and Studies*, IGI Global, 2011.

Sequential data from Web server logs, online transaction logs, and performance measurements is collected each day. This sequential data is a valuable source of information, as it allows individuals to search for a particular value or event and also facilitates analysis of the frequency of certain events or sets of related events. Finding patterns in sequences is of utmost importance in many areas of science, engineering, and business scenarios. The book provides a comprehensive view of sequence-mining techniques and presents current research and case studies in pattern discovery in sequential data by researchers and practitioners. This research identifies industry applications introduced by various sequence-mining approaches.

11

WEB MINING AND TEXT MINING

Chapter Objectives

- Explain the specifics of Web mining.
- Introduce a classification of basic Web mining subtasks.
- Illustrate the possibilities of Web mining using HITS, LOGSOM, and path-traversal algorithms.
- Describe query-independent ranking of Web pages and main characteristics of PageRank algorithm.
- Formalize a text-mining framework specifying the refining and distillation phases.
- Outline latent semantic indexing methodology.

11.1   WEB MINING

In a distributed information environment, documents or objects are usually linked together to facilitate interactive access. Examples for such information-providing environments include the World Wide Web (WWW) and online services such as America Online, where users, when seeking information of interest, travel from one object to another via facilities such as hyperlinks and URL addresses. The Web is an ever growing body of hypertext and multimedia documents. As of 2008, Google had discovered 1 trillion Web pages. The Internet Archive, which makes regular copies of many publicly available Web pages and media files, was three petabytes in size as of March 2009. Several billions of pages are added each day to that number. As the information offered in the Web grows daily, obtaining that information becomes more and more tedious. The main difficulty lies in the semistructured or unstructured Web content that is not easy to regulate and where enforcing a structure or standards is difficult. A set of Web pages lacks a unifying structure and shows far more authoring styles and content variation than that seen in traditional print document collections. This level of complexity makes an "off-the-shelf" database-management and information-retrieval (IR) solution very complex and almost impossible to use. New methods and tools are necessary. Web mining may be defined as the use of data-mining techniques to automatically discover and extract information from Web documents and services. It refers to the overall process of discovery, not just to the application of standard data-mining tools. Some authors suggest decomposing Web mining task into four subtasks:

1. *Resource finding*—This is the process of retrieving data, which is either online or offline, from the multimedia sources on the Web, such as news articles, forums, blogs, and the text content of HTML documents obtained by removing the HTML tags.
2. *Information selection and preprocessing*—This is the process by which different kinds of original data retrieved in the previous subtask is transformed. These transformations could be either a kind of preprocessing such as removing stop words, stemming, etc. or a preprocessing aimed at obtaining the desired representation, such as finding phrases in the training corpus, representing the text in the first-order logic form, etc.
3. *Generalization*—Generalization is the process of automatically discovering general patterns within individual Web sites as well as across multiple sites. Different general-purpose machine-learning techniques, data-mining techniques, and specific Web-oriented methods are used.
4. *Analysis*—This is a task in which validation and/or interpretation of the mined patterns is performed.

There are three factors affecting the way a user perceives and evaluates Web sites through the data-mining process: (1) Web-page content, (2) Web-page

design, and (3) overall site design including structure. The first factor concerns the goods, services, or data offered by the site. The other factors concern the way in which the site makes content accessible and understandable to its users. We distinguish between the design of individual pages and the overall site design, because a site is not simply a collection of pages; it is a network of related pages. The users will not engage in exploring it unless they find its structure simple and intuitive. Clearly, understanding user-access patterns in such an environment will not only help improve the system design (e.g., providing efficient access between highly correlated objects, better authoring design for WWW pages, etc.) but also be able to lead to better marketing decisions. Commercial results will be improved by putting advertisements in proper places, better customer/user classification, and understanding user requirements better through behavioral analysis.

No longer are companies interested in Web sites that simply direct traffic and process orders. Now they want to maximize their profits. They want to understand customer preferences and customize sales pitches to individual users. By evaluating a user's purchasing and browsing patterns, e-vendors want to serve up (in real-time) customized menus of attractive offers e-buyers cannot resist. Gathering and aggregating customer information into e-business intelligence is an important task for any company with Web-based activities. E-businesses expect big profits from improved decision-making, and therefore e-vendors line up for data-mining solutions.

Borrowing from marketing theory, we measure the efficiency of a Web page by its contribution to the success of the site. For an online shop, it is the ratio of visitors that purchased a product after visiting this page to the total number of visitors that accessed the page. For a promotional site, the efficiency of the page can be measured as the ratio of visitors that clicked on an advertisement after visiting the page. The pages with low efficiency should be redesigned to better serve the purposes of the site. Navigation-pattern discovery should help in restructuring a site by inserting links and redesigning pages and ultimately accommodating user needs and expectations.

To deal with problems of Web-page quality, Web-site structure, and their use, two families of Web tools emerge. The first includes tools that accompany the users in their navigation, learn from their behavior, make suggestions as they browse, and, occasionally, customize the user profile. These tools are usually connected to or built into parts of different search engines. The second family of tools analyzes the activities of users offline. Their goal is to provide insight in the semantics of a Web site's structure by discovering how this structure is actually utilized. In other words, knowledge of the navigational behavior of users is used to predict future trends. New data-mining techniques are behind these tools, where Web log files are analyzed and information uncovered. In the next four sections, we will illustrate Web mining with four techniques that are representative of a large spectrum of Web mining methodologies developed recently.

11.2   WEB CONTENT, STRUCTURE, AND USAGE MINING

One possible categorization of Web mining is based on which part of the Web one mines. There are three main areas of Web mining: Web content mining, Web structure mining, and Web usage mining. Each area is classified by the type of data used in the mining process. Web content mining uses Web page content as the data source for the mining process. This could include text, images, video, or any other type of content on Web pages. Web structure mining focuses on the link structure of Web pages. Web usage mining does not use data from the Web itself but takes as input data recorded from the interaction of users using the Internet.

The most common use of Web content mining is in the process of search. There are many different solutions that take as input Web-page text or images with the intent of helping users find information that is of interest to them. For example, crawlers are currently used by search engines to extract Web content into the indices that allow immediate feedback from searches. The same crawlers can be altered in such a way that rather than seeking to download all reachable content on the Internet, they can be focused on a particular topic or area of interest.

To create a focused crawler, a classifier is usually trained on a number of documents selected by the user to inform the crawler as to the type of content to search for. The crawler will then identify pages of interest as it finds them and follow any links on that page. If those links lead to pages that are classified as not being of interest to the user, then the links on that page will not be used further by the crawler.

Web content mining can also be seen directly in the search process. All major search engines currently use a list like structure to display search results. The list is ordered by a ranking algorithm behind the scenes. An alternative view of search results that has been attempted is to provide the users with clusters of Web pages as results rather than individual Web pages. Often a hierarchical clustering is performed, which will give multiple topic levels.

As an example consider the Web site Clusty.com that provides a clustered view of search results. If one keyword were to enter [jaguar] as a search onto this Web site, one sees both a listing of topics and a list of search results side by side, as shown in Figure 11.1. This specific query is ambiguous, and the topics returned show that ambiguity. Some of the topics returned include cars, Onca and Panthery (animal kingdom), and Jacksonville (American football team). Each of these topics can be expanded to show all of the documents returned for this query in a given topic.

Web structure mining considers the relationships between Web pages. Most Web pages include one or more hyperlinks. These hyperlinks are assumed in structure mining to provide an endorsement by the linking page of the page linked. This assumption underlies PageRank and HITS that will be explained later in this section.

Web structure mining is mainly used in the IR process. PageRank algorithm may have directly contributed to the early success of Google. Certainly the analysis of the structure of the Internet and the interlinking of pages currently contributes to the ranking of documents in most major search engines.

Figure 11.1. Example query from Clusty.com.

Web structure mining is also used to aid in Web content mining processes. Often classification tasks will consider features from both the content of the Web page and may consider the structure of the Web pages. One of the more common features in Web mining tasks taken from structure mining is the use of anchor text. Anchor text refers to the text displayed to users on an HTML hyperlink. Oftentimes the anchor text provides summary keywords not found on the original Web page. The anchor text is often as brief as search engine queries. Additionally, if links are endorsements of Web pages, then the anchor text offers keyword specific endorsements.

Web usage mining refers to the mining of information about the interaction of users with Web sites. This information may come from server logs, logs recorded by the client's browser, registration form information, etc. Many usage questions exist such as the following: How the link structure of the Web site differs from how users may prefer to traverse the page? Where are the inefficiencies in the e-commerce process of a Web site? What segments exist in our customer base?

There are some key terms in Web usage mining that require defining. A "visitor" to a Web site may refer to a person or program that retrieves a Web page from a server. A "session" refers to all page views that took place during a single visit to a Web site. Sessions are often defined by comparing page views and determining the maximum allowable time between page views before a new session is defined. Thirty minutes is a standard setting.

Web usage mining data often requires a number of preprocessing steps before meaningful data mining can be performed. For example, server logs often include a number of computer visitors that could be search engine crawlers or any other

computer program that may visit Web sites. Sometimes these "robots" identify themselves to the server passing a parameter called "user agent" to the server that uniquely identifies them as robots. Some Web-page requests do not make it to the Web server for recording, but instead a request may be filled by a cache used to reduce latency.

Servers record information on a granularity level that is often not useful for mining. For a single Web-page view, a server may record the browsers request for the HTML page, a number of requests for images included on that page, the Cascading Style Sheets (CSS) of a page, and perhaps some JavaScript libraries used by that Web page. Often there will need to be a process to combine all of these requests into a single record. Some logging solutions sidestep this issue by using JavaScript embedded into the Web page to make a single request per page view to a logging server. However, this approach has the distinct disadvantage of not recording data for users that have disabled JavaScript in their browser.

Web usage mining takes advantage of many of the data-mining approaches available. Classification may be used to identify characteristics unique to users that make large purchases. Clustering may be used to segment the Web user population. For example, one may identify three types of behavior occurring on a university class Web site. These three behavior patterns could be described as users cramming for a test, users working on projects, and users consistently downloading lecture notes from home for study. Association mining may identify two or more pages often viewed together during the same session, but those are not directly linked on a Web site. Sequence analysis may offer opportunities to predict user-navigation patterns and therefore allow for within site targeted advertisements. More on Web usage mining will be shown through the LOGSOM algorithm and through the section "Mining Path-Traversal Patterns."

11.3 HITS AND LOGSOM ALGORITHMS

To date, index-based search engines for the Web have been the primary tool with which users search for information. Experienced Web surfers can make effective use of such engines for tasks that can be solved by searching with tightly constrained keywords and phrases. These search engines are, however, unsuited for a wide range of less precise tasks. How does one select a subset of documents with the most value from the millions that a search engine has prepared for us? To distill a large Web-search topic to a size that makes sense to a human user, we need a means of identifying the topic's most authoritative Web pages. The notion of authority adds a crucial dimension to the concept of relevance: we wish to locate not only a set of relevant pages but also those that are of the highest quality.

It is important that the Web consists not only of pages but also hyperlinks that connect one page to another. This hyperlink structure contains an enormous amount of information that can help to automatically infer notions of authority. Specifically,

the creation of a hyperlink by the author of a Web page represents an implicit endorsement of the page being pointed to. By mining the collective judgment contained in the set of such endorsements, we can gain a richer understanding of the relevance and quality of the Web's contents. It is necessary for this process to uncover two important types of pages: *authorities*, which provide the best source of information on a given topic, and *hubs*, which provide a collection of links to authorities.

Hub pages appear in a variety of forms, ranging from professionally assembled resource lists on commercial sites to lists of recommended links on individual home pages. These pages need not themselves be prominent, and working with hyperlink information in hubs can cause much difficulty. Although many links represent some kind of endorsement, some of the links are created for reasons that have nothing to do with conferring authority. Typical examples are navigation and paid advertisement hyperlinks. A hub's distinguishing feature is that they are potent conferrers of authority on a focused topic. We can define a *good hub* if it is a page that points to many good authorities. At the same time, a good authority page is a page pointed to by many good hubs. This mutually reinforcing relationship between hubs and authorities serves as the central idea applied in the *hyperlink-induced topic search (HITS) algorithm* that searches for good hubs and authorities. The two main steps of the HITS algorithm are:

1. *Sampling component*, which constructs a focused collection of Web pages likely to be rich in relevant information, and
2. *Weight-propagation component*, which determines the estimates of hubs and authorities by an iterative procedure and obtains the subset of the most relevant and authoritative Web pages.

In the sampling phase, we view the Web as a directed graph of pages. The HITS algorithm starts by constructing the subgraph in which we will search for hubs and authorities. Our goal is a subgraph rich in relevant, authoritative pages. To construct such a subgraph, we first use query terms to collect a root set of pages from an index-based search engine. Since many of these pages are relevant to the search topic, we expect that at least some of them are authorities or that they have links to most of the prominent authorities. We therefore expand the root set into a base set by including all the pages that the root set pages link to up to a designated cutoff size. This base set V typically contains from 1000 to 5000 pages with corresponding links, and it is a final result of the first phase of HITS.

In the weight-propagation phase, we extract good hubs and authorities from the base set V by giving a concrete numeric interpretation to all of them. We associate a nonnegative authority weight a_p and a nonnegative hub weight h_p with each page $p \in V$. We are interested only in the relative values of these weights; therefore normalization is applied so that their total sum remains bounded. Since we do not impose any prior estimates, we set all a and h values to a uniform constant initially. The final weights are unaffected by this initialization.

We now update the authority and hub weights as follows. If a page is pointed to by many good hubs, we would like to increase its authority weight. Thus, we update the value of a_p for the page p to be the sum of h_q over all pages q that link to p:

$$a_p = \Sigma h_q, \quad \forall q \text{ such that } q \rightarrow p$$

where the notation $q \rightarrow p$ indicates that page q links to page p. In a strictly dual fashion, if a page points to many good authorities, we increase its hub weight:

$$h_p = \Sigma a_q, \quad \forall q \text{ such that } p \rightarrow q$$

There is a more compact way to write these updates. Let us number the pages $\{1,2,\ldots,n\}$ and define their adjacency matrix A to be $n \times n$ matrix whose (i, j)th element is equal to 1 if page i links to page j and 0 otherwise. All pages at the beginning of the computation are both hubs and authorities, and, therefore, we can represent them as vectors:

$$a = \{a_1, a_2, \ldots, a_n\} \quad \text{and}$$
$$h = \{h_1, h_2, \ldots, h_n\}$$

Our update rules for authorities and hubs can be written as

$$a = A^T h$$
$$h = A a$$

or, substituting one into another relation,

$$a = A^T h = A^T A a = \left(A^T A \right) a$$
$$h = A a = A A^T h = \left(A A^T \right) h$$

These are relations for iterative computation of vectors a and h. Linear algebra tells us that this sequence of iterations, when normalized, converges to the principal eigenvector of $A^T A$. This says that the hub and authority weights we compute are truly an intrinsic feature of the linked pages collected, not an artifact of our choice of initial weights. Intuitively, the pages with large weights represent a very dense pattern of linkage, from pages of large hub weights to pages of large authority weights. Finally, HITS produces a short list consisting of the pages with the largest hub weights and the pages with the largest authority weights for the given search topic. Several extensions and improvements of the HITS algorithm are available in the literature. Here we will illustrate the basic steps of the algorithm using a simple example.

Suppose that a search engine has selected six relevant documents based on our query, and we want to select the most important authority and hub in the available set.

(a)

(b)

$$A = \begin{bmatrix} 0\,0\,0\,1\,1\,1 \\ 0\,0\,0\,1\,1\,0 \\ 0\,0\,0\,0\,1\,0 \\ 0\,0\,0\,0\,0\,0 \\ 0\,0\,0\,0\,0\,0 \\ 0\,0\,1\,0\,0\,0 \end{bmatrix}$$

$a = \{0.1, 0.1, 0.1, 0.1, 0.1, 0.1\}$
$h = \{0.1, 0.1, 0.1, 0.1, 0.1, 0.1\}$
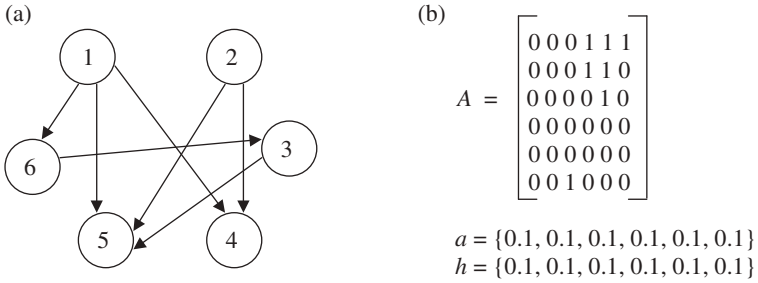
Figure 11.2. Initialization of the HITS algorithm. (a) Subgraph of the linked pages. (b) Adjacency matrix A and weight vectors for the given graph.

The selected documents are linked into a directed subgraph, and the structure is given in Figure 11.2a, while corresponding adjacency matrix A and initial weight vectors a and h are given in Figure 11.2b.

The first iteration of the HITS algorithm will give the changes in the a and h vectors:

$$a = \begin{bmatrix} 0\,0\,0\,0\,0\,0 \\ 0\,0\,0\,0\,0\,0 \\ 0\,0\,0\,0\,0\,1 \\ 1\,1\,0\,0\,0\,0 \\ 1\,1\,1\,0\,0\,0 \\ 1\,0\,0\,0\,0\,0 \end{bmatrix} \cdot \begin{bmatrix} 0\,0\,0\,1\,1\,1 \\ 0\,0\,0\,1\,1\,0 \\ 0\,0\,0\,0\,1\,0 \\ 0\,0\,0\,0\,0\,0 \\ 0\,0\,0\,0\,0\,0 \\ 0\,0\,1\,0\,0\,0 \end{bmatrix} \cdot \begin{bmatrix} 0.1 \\ 0.1 \\ 0.1 \\ 0.1 \\ 0.1 \\ 0.1 \end{bmatrix}$$

$$= [0\ 0\ 0.1\ 0.5\ 0.6\ 0.3]$$

$$h = \begin{bmatrix} 0\,0\,0\,1\,1\,1 \\ 0\,0\,0\,1\,1\,0 \\ 0\,0\,0\,0\,1\,0 \\ 0\,0\,0\,0\,0\,0 \\ 0\,0\,0\,0\,0\,0 \\ 0\,0\,1\,0\,0\,0 \end{bmatrix} \cdot \begin{bmatrix} 0\,0\,0\,0\,0\,0 \\ 0\,0\,0\,0\,0\,0 \\ 0\,0\,0\,0\,0\,1 \\ 1\,1\,0\,0\,0\,0 \\ 1\,1\,1\,0\,0\,0 \\ 1\,0\,0\,0\,0\,0 \end{bmatrix} \cdot \begin{bmatrix} 0.1 \\ 0.1 \\ 0.1 \\ 0.1 \\ 0.1 \\ 0.1 \end{bmatrix}$$

$$= [0.6\ \ 0.5\ \ 0.3\ \ 0\ \ 0\ \ 0.1]$$

Even this single iteration of the HITS algorithm shows that, in the given set of documents, document 5 has the most authority and document 1 is the best hub.

Additional iterations will correct the weight factors for both vectors, but the obtained ranking of authorities and hubs will stay unchanged for this example.

The continuous growth in the size and use of the Internet creates difficulties in the search for information. Resource discovery is frustrating and inefficient when simple keyword searches can convey hundreds of thousands of documents as results. Many of them are irrelevant pages, some of them may have been moved, and some abandoned. While the first Web mining algorithm HITS is primarily based on static information describing the Web-site structure, the second one LOGSOM uses dynamic information about a user's behavior. LOGSOM is a sophisticated method, which organizes the layout of the information in a user-interpretable graphic form. The LOGSOM system uses self-organizing maps (SOM) to organize Web pages into a two-dimensional (2D) table, according to users' navigation patterns. The system organizes Web pages according to the interest of Web users by keeping track of their navigation paths.

The SOM technique is used as the most appropriate technique for the problem of Web-page organization because of its strength not only in grouping data points into clusters but also in graphically representing the relationship among clusters. The system starts with a Web log file indicating the date, time, and address of the requested Web pages as well as the IP address of the user's machine. The data are grouped into meaningful transactions or sessions, where a transaction is defined by a set of user-requested Web pages. We assume that there is a finite set of unique URLs

$$U = \{\text{url}_1, \text{url}_2, \ldots, \text{url}_n\}$$

and a finite set of m user transactions

$$T = \{t_1, t_2, \ldots, t_m\}$$

Transactions are represented as a vector with binary values u_i:

$$t = [u_1, u_2, \ldots, u_n]$$

where

$$u_i = \begin{cases} 1 & \text{if } \text{url}_i \in t \\ 0 & \text{otherwise} \end{cases}$$

Preprocessed log files can be represented as a binary matrix. One example is given in Table 11.1.

Since the dimensions of a table $(n \times m)$ for real-world applications would be very large, especially as input data to self-organizing maps, a reduction is necessary. By using the k-means clustering algorithm, it is possible to cluster transactions into pre-specified number k ($k \ll m$) of transaction groups. An example of a transformed table

TABLE 11.1. Transactions Described by a Set of URLs

	url_1	url_2	…	url_n
t_1	0	1		1
t_2	1	1		0
…				
t_m	0	0		0

TABLE 11.2. Representing URLs as Vectors of Transaction Group Activity

	Transaction Groups			
	1	2	…	k
url_1	15	0		2
url_2	2	1		10
…				
url_n	0	1		2

with new reduced data set is represented in Table 11.2, where the elements in the rows represent the total number of times a group accessed a particular URL (the form of the table and values are only one illustration, and they are not directly connected with the values in Table 11.1).

The new reduced table is the input for SOM processing. Details about application of SOM as a clustering technique and the settings of their parameters are given in the previous chapter. We will explain only the final results and their interpretation in terms of Web-page analysis. Each URL will be mapped onto an SOM based on its similarity with other URLs in terms of user usage or, more precisely, according to users' navigation patterns (transaction group "weights" in Table 11.2). Supposing that the SOM is 2D map with $p \times p$ nodes, where $p \times p \geq n$, then a typical result of SOM processing is given in Table 11.3. The dimensions and values in the table are not the results of any computation with values in Tables 11.1 and 11.2, but a typical illustration of the SOM's final presentation.

The SOM organizes Web pages into similar classes based on users' navigation patterns. The blank nodes in the table show that there are no corresponding URLs, while the numbered nodes indicate the number of URLs contained within each node (or within each class). The distance on the map indicates the similarity of the Web pages measured by the user-navigation patterns. For example, the number 54 in the last row shows that 54 Web pages are grouped in the same class because they have been accessed by similar types of people as indicated by their transaction patterns. Similarity here is measured not by similarity of content but by similarity of usage. Therefore, the organization of the web documents in this graphical representation is based solely on the users' navigation behavior.

TABLE 11.3. A Typical SOM Generated by the Description of URLs

	1	2	3	…	p
1	2		1		15
2	3	1	10	…	
…					
p		54		…	11

What are the possible applications of the LOGSOM methodology? The ability to identify which Web pages are being accessed by a company's potential customers gives the company information to make improved decisions. If one Web page within a node successfully refers clients to the desired information or desired page, the other pages in the same node are likely to be successful as well. Instead of subjectively deciding where to place an Internet advertisement, the company can now decide objectively, supported directly by the user-navigation patterns.

11.4 MINING PATH-TRAVERSAL PATTERNS

Before improving a company's Web site, we need a way of evaluating its current usage. Ideally, we would like to evaluate a site based on the data automatically recorded on it. Each site is electronically administered by a Web server, which logs all activities that take place in it in a file called a Web server log. All traces left by the Web users are stored in this log. Therefore, from these log files we can extract information that indirectly reflects the site's quality by applying data-mining techniques. We can mine data to optimize the performance of a Web server, to discover which products are being purchased together, or to identify whether the site is being used as expected. The concrete specification of the problem guides us through different data-mining techniques applied to the same Web server log.

While the LOGSOM methodology is concentrated on similarity of Web pages, other techniques emphasize the similarity of a user's paths through the Web. Capturing user-access patterns in a Web environment is referred to as *mining path-traversal patterns*. It represents an additional class of data-mining techniques, which is showing great promise. Note that because users travel along information paths to search for the desired information, some objects or documents are visited because of their location rather than their content. This feature of the traversal pattern unavoidably increases the difficulty of extracting meaningful information from a sequence of traversal data and explains the reason why current Web usage analyses are mainly able to provide statistical information for traveling points, but not for traveling paths. However, as these information-providing services become increasingly popular, there is a growing demand for capturing user-traveling behavior to improve the quality of such services.

We first focus on the theory behind the navigational patterns of users in the Web. It is necessary to formalize known facts about navigation: that not all pages across a

path are of equal importance and that the users tend to revisit pages previously accessed. To achieve a data-mining task, we define a navigation pattern in the Web as a generalized notion of a sequence, the materialization of which is the directed acyclic graph. A sequence is an ordered list of items, in our case Web pages, ordered by time of access. The log file L is a multiset of recorded sequences. It is not a simple set, because a sequence may appear more than once.

When we want to observe sequence s as a concatenation of the consecutive subsequences x and y, we use the notation

$$s = xy$$

The function *length*(s) returns the number of elements in the sequence s. The function *prefix*(s, i) returns the subsequence comprising the first i elements of s. If $s' = prefix(s,i)$, we say that s' is a prefix of s and is denoted as $s' \leq s$. Analysis of log files shows that Web users tend to move backward and revisit pages with a high frequency. Therefore, a log file may contain duplicates. Such revisits may be part of a guided tour or may indicate disorientation. In the first case, their existence is precious as information and should be retained. To model cycles in a sequence, we label each element of the sequence with its occurrence number within the sequence, thus distinguishing between the first, second, third, and other occurrences of the same page.

Moreover, some sequences may have common prefixes. If we merge all common prefixes together, we transform parts of the log file into a tree structure, each node of which is annotated with the number of sequences having the same prefix up to and including this node. The tree contains the same information as the initial log file. Hence, when we look for frequent sequences, we can scan the tree instead of the original log multiset. On the tree, a prefix shared among k sequences appears and gets tested only once.

Sequence mining can be explained as follows: Given a collection of sequences ordered in time, where each sequence contains a set of Web pages, the goal is to discover sequences of maximal length that appear more frequently than a given percentage threshold over the whole collection. A frequent sequence is maximal if all sequences containing it have a lower frequency. This definition of the sequence-mining problem implies that the items constituting a frequent sequence need not necessarily occur adjacent to each other. They just appear in the same order. This property is desirable when we study the behavior of Web users because we want to record their intents, not their errors and disorientations.

Many of these sequences even those with the highest frequencies could be of a trivial nature. In general, only the designer of the site can say what is trivial and what is not. The designer has to read all patterns discovered by the mining process and discard unimportant ones. It would be much more efficient to automatically test data-mining results against the expectations of the designer. However, we can hardly expect a site designer to write down all combinations of Web pages that are considered typical; expectations are formed in the human mind in much more abstract terms. Extraction of informative and useful maximal sequences continues to be a challenge for researchers.

Although there are several techniques proposed in the literature, we will explain one of the proposed solutions for mining-traversal patterns that consists of two steps:

(a) In a first step, an algorithm is developed to convert the original sequence of log data into a set of *traversal subsequences*. Each traversal subsequence represents a maximum forward reference from the starting point of a user access. It should be noted that this step of conversion would filter out the effect of backward references, which are mainly made for ease of traveling. The new reduced set of user-defined forward paths enables us to concentrate on mining meaningful user-access sequences.

(b) The second step consists of a separate algorithm for determining the frequent-traversal patterns, termed *large reference sequences*. A large reference sequence is a sequence that appears a sufficient number of times in the log database. In the final phase, the algorithm forms the *maximal references* obtained from large reference sequences. A maximal large sequence is a large reference sequence that is not contained in any other maximal reference sequence.

For example, suppose the traversal log of a given user contains the following path (to keep it simple, Web pages are represented by letters):

$$\text{Path} = \{ABCDCBEGHGWAOUOV\}$$

The path is transformed into the tree structure shown in Figure 11.3. The set of maximum forward references MRF found in the step (a) after elimination of backward references is

$$\text{MFR} = \{ABCD, ABEGH, ABEGW, AOU, AOV\}$$
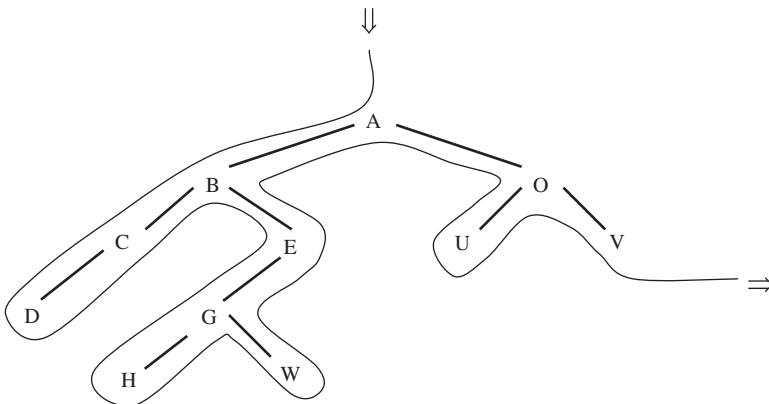
Figure 11.3. An example of traversal patterns.

When maximum forward references have been obtained for all users, the problem of finding frequent-traversal patterns is mapped into one of finding frequently occurring consecutive subsequences among all maximum forward references. In our example, if the threshold value is 0.4 (or 40%), large reference sequences (LRS) with lengths 2, 3, and 4 are

$$LRS = \{AB, BE, EG, AO, ABE, BEG, ABEG\}$$

Finally, with large reference sequences determined, *maximal reference sequences* can be obtained through the process of selection. The resulting set for our example is

$$MRS = \{ABEG, AO\}$$

In general, these sequences, obtained from large log files, correspond to a frequently accessed pattern in an information-providing service.

The problem of finding large reference sequences is very similar to that of finding frequent itemsets (occurring in a sufficient number of transactions) in association rule mining. However, they are different from each other in that a reference sequence in the mining-traversal patterns has to be references in a given order, whereas a large itemset in mining association rules is just a combination of items in a transaction. The corresponding algorithms are different because they perform operations on different data structures: lists in the first case and sets in the second. As the popularity of Internet applications explodes, it is expected that one of the most important data-mining issues for years to come will be the problem on how to effectively discover knowledge on the Web.

11.5 PageRank ALGORITHM

PageRank was originally published by Sergey Brin and Larry Page, the co-creators of Google. It likely contributed to the early success of Google. PageRank provides a global ranking of nodes in a graph. For search engines it provides a query-independent authority ranking of all Web pages. PageRank has similar goals of finding authoritative Web pages to that of the HITS algorithm. The main assumption behind the PageRank algorithm is that every link from page *a* to page *b* is a vote by page *a* for page *b*. Not all votes are equal. Votes are weighted by the PageRank score of the originating node.

PageRank is based on the random surfer model. If some surfer were to randomly select a starting Web page and at each time step the surfer were to randomly select a link on the current Web page, then PageRank could be seen as the probability that this random surfer is on any given page. Some Web pages do not contain any hyperlinks. In this model it is assumed that the random surfer selects a random Web page when exiting pages with no hyperlinks. Additionally, there is some chance that the random surfer will stop following links and restart the process.

Computationally the PageRank (Pr) of page u can be computed as follows:

$$\Pr(u) = \frac{1-d}{N} + d\left(\sum_{v\in\mathrm{In}(u)} \frac{\Pr(v)}{|\mathrm{Out}(v)|}\right)$$

where d is a dampening factor $0 \le d \le 1$ usually set to 0.85 and N refers to the total number of nodes in the graph. The function In(u) returns the set of nodes with edges pointing into node u. $|\mathrm{Out}(v)|$ returns the number of nodes with edges pointing from v to that node. For example, if the Web-page connections are those given in Figure 11.4, and the current node under consideration were node B, then the following values would hold through all iterations: $N = 3$, In$(B) = \{A,C\}$, $|\mathrm{Out}(A)| = |\{B,C\}| = 2$, and $|\mathrm{Out}(C)| = |\{B\}| = 1$. The values for Pr$(A)$, Pr$(B)$, and Pr$(C)$ would vary depending on the calculations from the previous iterations. The result is a recursive definition of PageRank. To calculate the PageRank of a given node, one must calculate the PageRank of all nodes with edges pointing into that given node.

Often PageRank is calculated using an iterative approach where all nodes are given an initial value for Pr of 1/N. Then during a single iteration, we calculate what the PageRank of each node would be according to the current values of all nodes linking to that node. This process is repeated until the change between iterations is below some predetermined threshold or a maximum number of iterations are achieved. Let us consider an example graph with three nodes as follows:

Initially, the Pr values are as follows:

$$\Pr(A) = \frac{1}{N} = 0.333$$

$$\Pr(B) = 0.333$$

$$\Pr(C) = 0.333$$

The first iteration corrects initial values as follows:

$$\Pr(A) = \frac{(1-0.85)}{3} + 0.85\left(\frac{\Pr(B)}{1}\right) = \frac{0.15}{3} + 0.85(0.333) = 0.333$$

$$\Pr(B) = \frac{(1-0.85)}{3} + 0.85\left(\frac{\Pr(A)}{2} + \frac{\Pr(C)}{1}\right) = \frac{0.15}{3} + 0.85\left(\frac{0.333}{2} + 0.333\right) = 0.475$$
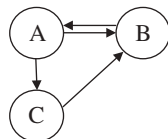
Figure 11.4. First example used to demonstrate PageRank.

$$Pr(C) = \frac{(1-0.85)}{3} + 0.85 \left(\frac{Pr(A)}{2} \right) = \frac{0.15}{3} + 0.85 \left(\frac{0.333}{2} \right) = 0.192$$

The second iteration then shows the passing of these values through the graph:

$$Pr(A) = \frac{(1-0.85)}{3} + 0.85 \left(\frac{Pr(B)}{1} \right) = \frac{0.15}{3} + 0.85(0.475) = 0.454$$

$$Pr(B) = \frac{(1-0.85)}{3} + 0.85 \left(\frac{Pr(A)}{2} + \frac{Pr(C)}{1} \right) = \frac{0.15}{3} + 0.85 \left(\frac{0.454}{2} + 0.192 \right) = 0.406$$

$$Pr(C) = \frac{(1-0.85)}{3} + 0.85 \left(\frac{Pr(A)}{2} \right) = \frac{0.15}{3} + 0.85 \left(\frac{0.454}{2} \right) = 0.243$$

If we carry this same procedure out to 100 iterations, we achieve the following results:

$$Pr(A) = 0.388$$
$$Pr(B) = 0.397$$
$$Pr(C) = 0.215$$

Additional iterations produce the same results. From this we can see a stable ordering emerge. B has the largest PageRank value having two in-links, more than any other. However, page A is not far behind, since page B has only a single link to page A without dividing its PageRank value among a number of outbound links.

Next we consider the example used for the HITS algorithm, which is applied on the graph in Figure 11.5a. The PageRank of this graph with a dampening factor of 0.85 is given in Figure 11.5b after running 100 iterations. A reader may, for the practice, check the results after the first iteration or implement PageRank algorithm and check final results in Figure 11.5b.

(a)

(b)

Node	PageRank
1	0.095
2	0.095
3	0.198
4	0.162
5	0.330
6	0.121
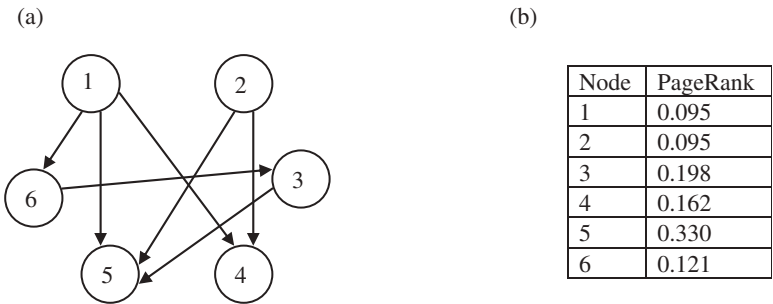
Figure 11.5. An example graph and scoring for PageRank also used with HITS. (a) Graph of linked pages. (b) Calculated PageRank for given graph.

From the values given in Figure 11.5b, we can interpret that node 5 has the highest PageRank by far and also has the highest in-degree or edges pointing in to it. Surprising perhaps is that node 3 has the next highest score, having a score higher than node 4, which has more in-edges. The reason is that node 6 has only one out-edge pointing to node 3, while the edges pointing to node 4 are each one of multiple out edges. Lastly, as expected the lowest ranked edges are those with no in-edges, nodes 1 and 2.

One of the main contributions of Google's founders is implementation experimental evaluation of PageRank algorithm. They included database of Web sites with 161 millions of links, and the algorithm converges in 45 iterations. Repeated experiments with 322 millions of links converged in 52 iterations. These experiments were evidence that PageRank converge in $\log(n)$ time where n is number of links, and it is applicable for growing Web. Of course, initial version of the PageRank algorithm, explained in a simplified form in this text, had numerous modifications to evolve into current commercial version implemented in Google search engine.

11.6 RECOMMENDER SYSTEMS

The explosive growth of Internet has resulted in a phenomenon known as information abundance, and it demands new techniques that can assist us to discover resources of interest among the enormous options we are presented with. All of this paved a way for the introduction of recommender systems (RS). RSs help users or groups of users deal with information overload by proposing to them items suited to their interests. The history of RSs started in the late 1990s, but main advances are obtained with the Netflix competition for movie RSs that attracted over 41,000 participating teams and turned RS into a hot topic among researchers.

RSs exploit various sources of information: about users and their demographics, about products and their features, and about user interactions with the products. Data infrastructure for RS may be either explicit with different scales of rating and satisfaction or implicit: product purchased, book read, song heard, or Web-site content clicked. Research activities in RSs has become very active recently and successfully been used in many industry sectors to recommend items such as Netflix movies, Amazon products, jobs to Facebook users, books, songs, news, friends, restaurants, food, apparels, vehicles, banners, or content on a social site. The basic principle of recommendations is that significant dependencies exist between user and item-centric activity, and discovery of these dependencies is available based on large amount of historic data. The basic models for RSs work with two categories of data:

(a) The user-item interactions, such as user rating movies or user buying behavior, and

(b) Attribute information about the users and items, including textual profiles or relevant keywords.

RS algorithms basically perform information filtering and can be classified into two main types:

(a) Collaborative filtering and
(b) Content-based filtering.

While collaborative filtering usually uses user-item interaction data, content-based filtering is using additional attribute information about users and items to develop the model. The term "collaborative filtering" refers to the use of ratings from multiple users in a collaborative way to predict missing ratings. These dependencies can be *learned* in a data-driven manner from the ratings user-item matrix, and the resulting model is used to make predictions for target users. The larger the number of rated items that are available, the easier it is to make robust predictions. The main challenge in designing collaborative filtering methods is that the underlying ratings matrices are highly sparse. Consider an example of a movie application in which users specify ratings indicating their like or dislike of specific movies. Most users would have viewed only a small fraction of the large universe of available movies; the matrix will be with the large number of "empty ratings."

In content-based RSs, the descriptive attributes of items are used to make recommendations. The term "content" refers to these descriptions. The ratings and buying behavior of users are combined with the content information available about items. For example, consider a situation where John has rated highly the movie *Terminator*, but we do not have access to the ratings of other users for this movie. Therefore, collaborative filtering methods are ruled out. However, the item description of *Terminator* movie contains similar genre keywords as other science fiction movies, such as *Alien* and *Predator*. In such cases, these movies can be recommended to John based on similarity of movie attributes.

From the perspective of the user, recommendations can help improve overall user satisfaction with the product presented at the Web site. At the merchant end, the recommendation process can provide insights into the needs of the user and help customize the user experience further. While a product recommendation directly increases the profit of the merchant by facilitating product sales, an increase in the number of social connections improves the experience of a user at a social network. This, in turn, encourages the growth of the social network. Social networks are heavily dependent on the growth of the network to increase their advertising revenues. Therefore, the recommendation of potential friends or specific links enables better growth and connectivity of the network.

11.7   TEXT MINING

Enormous amounts of knowledge reside today in text documents that are stored either within organizations or are freely available. Text databases are rapidly growing because of the increasing amounts of information available in electronic form, such

as electronic publications, digital libraries, e-mail, and the World Wide Web. Data stored in most text databases are semistructured, and special data-mining techniques, called text mining, have been developed for discovering new information from large collections of textual data.

In general, there are two key technologies that make online text mining possible. One is *Internet search* capabilities and the other is the *text-analysis* methodology. *Internet search* has been around for a few years. With the explosion of Web sites in the past decade, numerous search engines are designed to help users find content appeared practically overnight. Yahoo, AltaVista, and Excite are three of the earliest, while Google and Bing are most popular in recent years. Search engines operate by indexing the content in a particular Web site and allowing users to search these indices. With the new generation of Internet-search tools, users can gain relevant information by processing a smaller amount of links, pages, and indices.

Text analysis, as a field, has been around longer than Internet search. It has been a part of the efforts to make computers understand natural languages, and it is commonly thought of as a problem for artificial intelligence. Text analysis can be used anywhere where there is a large amount of text that need to be analyzed. Although automatic processing of documents using different techniques does not allow the depth of analysis that a human can bring to the task, it can be used to extract key points in the text, categorize documents, and generate summaries in a situation when a large number of documents makes manual analysis impossible.

To understand the details of text documents, you can either search for keywords, or you can try to categorize the semantic content of the document itself. When identifying keywords in text documents, you are looking at defining specific details or elements within documents that can be used to show connections or relationships with other documents. In the IR domain, documents have been traditionally represented in the vector space model. Documents are tokenized using simple syntactic rules (such as white-space delimiters in English), and tokens are transformed to canonical form (e.g., "reading" to "read," "is," "was," and "are" to "be"). Each canonical token represents an axis in a Euclidean space. Documents are vectors in this n-dimensional space. If a token t called term occurs n times in document d, then the tth coordinate of d is simply n. One may choose to normalize the length of the document to 1, using the L_1, L_2, or L_∞ norms:

$$\|d_1\| = \sum_t n(d,t), \qquad \|d_2\| = \sqrt{\sum_t n(d,t)^2}, \qquad \|d_\infty\| = \max_t n(d, t)$$

where $n(d,t)$ is the number of occurrences of a term t in a document d. These representations do not capture the fact that some terms, also called keywords (like "algorithm"), are more important than others (like "the" and "is") in determining document content. If t occurs in n_t out of N documents, n_t/N gives sense of rarity and, hence, the importance of the term. The inverse document frequency $IDF = 1 + \log(n_t/N)$ is used to stretch the axes of the vector space differentially. Thus the tth coordinate of document d may be represented with the value $(n(d,t)/\|d_1\|) \times IDF(t)$ in the weighted vector space model. In spite of being extremely crude

and not capturing any aspect of language or semantics, this model often performs well for its intended purpose. Also, in spite of minor variations, all these models of text regard documents as multisets of terms, without paying attention to ordering between terms. Therefore, they are collectively called bag-of-words models. Very often, the outputs from these keyword approaches can be expressed as relational data sets that may be then analyzed using one of the standard data-mining techniques.

Hypertext documents, usually represented as basic components on the Web, are a special type of text-based documents that have hyperlinks in addition to text. They are modeled with varying levels of details, depending on the application. In the simplest model, hypertext can be regarded as directed graph (D, L) where D is the set of nodes representing documents or Web pages and L is the set of links. Crude models may not need to include the text models at the node level, when the emphasis is on documents' links. More refined models will characterize some sort of joint distribution between the term distribution of a node and those in a certain neighborhood of the document in the graph.

Content-based analysis and partition of documents is a more complicated problem. Some progress has been made along these lines, and new text-mining techniques have been defined, but no standards or common theoretical background has been established in the domain. Generally, you can think of text categorization as comparing a document to other documents or to some predefined set of terms or definitions. The results of these comparisons can be presented visually within a semantic landscape in which similar documents are placed together in the semantic space and dissimilar documents are placed further apart. For example, indirect evidence often lets us build semantic connections between documents that may not even share the same terms. For example, "car" and "auto" terms co-occurring in a set of documents may lead us to believe that these terms are related. This may help us to relate documents with these terms as similar. Depending on the particular algorithm used to generate the landscape, the resulting topographic map can depict the strengths of similarities among documents in terms of Euclidean distance. This idea is analogous to the type of approach used to construct Kohonen feature maps. Given the semantic landscape, you may then extrapolate concepts represented by documents.

The automatic analysis of text information can be used for several different general purposes:

1. To provide an overview of the contents of a large document collection and organize them in the most efficient way,
2. To identify hidden structures between documents or groups of documents,
3. To increase the efficiency and effectiveness of a search process to find similar or related information, and
4. To detect duplicate information or documents in an archive.

Text mining is an emerging set of functionalities that are primarily built on text-analysis technology. Text is the most common vehicle for the formal exchange of information. The motivation for trying to automatically extract, organize, and use

information from it is compelling, even if success is only partial. While traditional, commercial text-retrieval systems are based on inverted text indices composed of statistics such as word occurrence per document, text mining must provide values beyond the retrieval of text indices such as keywords. Text mining is about looking for semantic patterns in text, and it may be defined as the process of analyzing text to extract interesting, nontrivial information that is useful for particular purposes.

As the most natural form of storing information is text, text mining is believed to have a commercial potential even higher than that of traditional data mining with structured data. In fact, recent studies indicate that 80% of a company's information is contained in text documents. Text mining, however, is also a much more complex task than traditional data mining as it involves dealing with unstructured text data that are inherently ambiguous. Text mining is a multidisciplinary field involving IR, text analysis, information extraction, natural language processing, clustering, categorization, visualization, machine learning, and other methodologies already included in the data-mining "menu"; even some additional specific techniques developed lately and applied on semistructured data can be included in this field. Market research, business intelligence gathering, e-mail management, claim analysis, E-procurement, and automated help desk are only a few of the possible applications where text mining can be deployed successfully. The text-mining process, which is graphically represented in Figure 11.6, consists of two phases:

- *Text refining* that transforms free-form text documents into a chosen intermediate form (IF) and
- *Knowledge distillation* that deduces patterns or knowledge from an intermediate form.

An (IF) can be semistructured such as the conceptual-graph representation or structured such as the relational data representation. Intermediate forms with varying
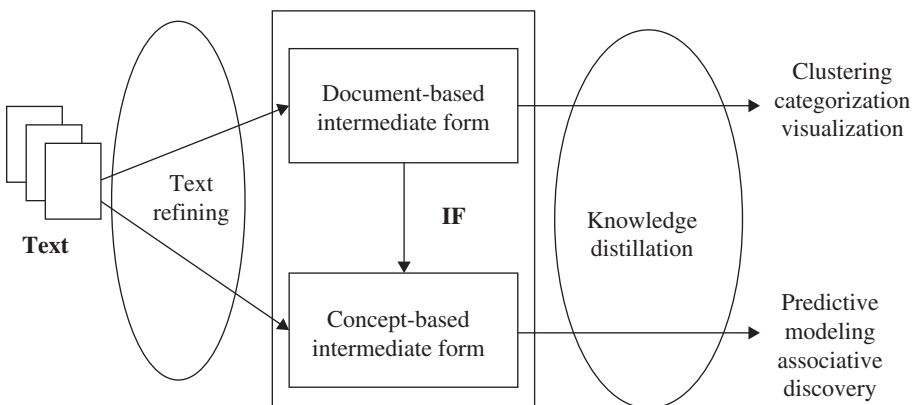
Figure 11.6. A text-mining framework.

degrees of complexity are suitable for different mining purposes. They can be classi-fied as *document based*, wherein each entity represents a document, or *concept based*, wherein each entity represents an object or concept of interests in a specific domain. Mining a document-based IF deduces patterns and relationships across documents. Document clustering, visualization, and categorization are examples of mining from document-based IFs.

For a fine-grained, domain-specific, knowledge-discovery task, it is necessary to perform a semantic analysis and derive a sufficiently rich representation to capture the relationship between objects or concepts described in the document. Mining a concept-based IF derives patterns and relationships across objects and concepts. These semantic analysis methods are computationally expensive, and it is a challenge to make them more efficient and scalable for very large text corpora. Text-mining operations such as predictive modeling and association discovery fall in this category. A document-based IF can be transformed into a concept-based IF by realigning or extracting the relevant information according to the objects of interests in a specific domain. It follows that a document-based IF is usually domain independent and a con-cept-based is a domain-dependent representation.

Text-refining and knowledge-distillation functions as well as the intermediate form adopted are the basis for classifying different text-mining tools and their corre-sponding techniques. One group of techniques, and recently available commercial products, focuses on document organization, visualization, and navigation. Another group focuses on text-analysis functions, IR, categorization, and summarization.

An important and large subclass of these text-mining tools and techniques is based on document visualization. The general approach here is to organize documents based on their similarities and present the groups or clusters of the documents as 2D or 3D graphics. IBM's Intelligent Miner and SAS Enterprise Miner are probably the most comprehensive commercial text-mining products. They offer a set of text-analysis tools that include tools for feature extraction, clustering, summarization, and categorization; it also incorporates a text search engine. More examples of text-mining tools are given in Appendix A.

Domain knowledge, not used and analyzed by any currently available text-mining tool, could play an important role in the text-mining process. Specifically, domain knowledge can be used as early as in the text-refining stage to improve parsing efficiency and derive a more compact intermediate form. Domain knowledge could also play a part in knowledge distillation to improve learning efficiency. All these ideas are still in their infancy, and we expect that the next generation of text-mining techniques and tools will improve the quality of information and knowledge discovery from text.

11.8 LATENT SEMANTIC ANALYSIS

Latent semantic analysis (LSA) is a method that was originally developed to improve the accuracy and effectiveness of IR techniques by focusing on semantic meaning of words across a series of usage contexts, as opposed to using simple string-matching

operations. LSA is a way of partitioning free text using a statistical model of word usage that is similar to eigenvector decomposition and factor analysis. Rather than focusing on superficial features such as word frequency, this approach provides a quantitative measure of semantic similarities among documents based on a word's context.

Two major shortcomings to the use of term counts are synonyms and polysemes. Synonyms refer to different words that have the same or similar meanings but are entirely different words. In the case of the vector approach, no match would be found between a query using the term "altruistic" and a document using the word "benevolent" though the meanings are quite similar. On the other hand polysemes are words that have multiple meanings. The term "bank" could mean a financial system, to rely upon, or a type of basketball shot. All of these lead to very different types of documents, which can be problematic for document comparisons.

LSA attempts to solve these problems, not with extensive dictionaries and natural language processing engines, but by using mathematical patterns within the data itself to uncover these relationships. We do this by reducing the number of dimensions used to represent a document using a mathematical matrix operation called singular value decomposition.

Let us take a look at an example data set. This very simple data set consists of five documents. We will show the dimension reduction steps of LSA on the first four documents ($d_{1\ldots4}$), which will make up our training data. Then we will make distance comparisons to a fifth document (d_5) in our test set, using the nearest neighbor classification approach. Initial documents' set is:

- d_1: A bank will protect your money.
- d_2: A guard will protect a bank.
- d_3: Your bank shot is money.
- d_4: A bank shot is lucky.
- d_5: Bank guard.

From the text data we derive a vector representation of our documents using only term counts. This vector representation could be thought of as a matrix with rows representing terms and columns representing documents. This representation may be seen in Figure 11.7.

The first step of LSA is to decompose the matrix representing our original data set of four documents, matrix A, using SVD as follows: $A = USV^T$. The calculation of singular value decomposition (SVD) is beyond the scope of this text, but there are several computing packages that will perform this operation for you (such as R or MATLAB packages). The matrices resulting from this decomposition can be seen in Figure 11.8. The $\mathbf{U}$ and $\mathbf{V^T}$ matrices provided a vector of weights for terms and documents, respectively. Considering $\mathbf{V^T}$, this matrix gives a new four-dimensional

$$
\mathbf{A} = \begin{array}{l}
\text{Terms} \\
\text{a} \\
\text{bank} \\
\text{guard} \\
\text{is} \\
\text{lucky} \\
\text{money} \\
\text{protect} \\
\text{shot} \\
\text{will} \\
\text{your}
\end{array}
\begin{array}{cccc}
\text{d1} & \text{d2} & \text{d3} & \text{d4} \\
1 & 2 & 0 & 1 \\
1 & 1 & 1 & 1 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 1 \\
0 & 0 & 0 & 1 \\
1 & 0 & 1 & 0 \\
1 & 1 & 0 & 0 \\
0 & 0 & 1 & 1 \\
1 & 1 & 0 & 0 \\
1 & 0 & 1 & 0
\end{array}
\qquad
\mathbf{d5} = \begin{array}{c}
\text{d5} \\
0 \\
1 \\
1 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0
\end{array}
$$

Figure 11.7. Initial term counts.

$$
\mathbf{U} = \begin{bmatrix}
-0.575 & 0.359 & 0.333 & 0.072 \\
-0.504 & -0.187 & 0.032 & -0.044 \\
-0.162 & 0.245 & 0.137 & -0.698 \\
-0.197 & -0.473 & 0.237 & -0.188 \\
-0.105 & -0.172 & 0.401 & 0.626 \\
-0.236 & -0.260 & -0.506 & 0.029 \\
-0.307 & 0.286 & -0.205 & 0.144 \\
-0.197 & -0.473 & 0.237 & -0.188 \\
-0.307 & 0.286 & -0.205 & 0.144 \\
-0.236 & -0.260 & -0.506 & 0.029
\end{bmatrix}
,\ \mathbf{S} = \begin{bmatrix}
3.869 & 0.000 & 0.000 & 0.000 \\
0.000 & 2.344 & 0.000 & 0.000 \\
0.000 & 0.000 & 1.758 & 0.000 \\
0.000 & 0.000 & 0.000 & 0.667
\end{bmatrix}
,\ \mathbf{V^T} = \begin{bmatrix}
-0.560 & -0.628 & -0.354 & -0.408 \\
0.095 & 0.575 & -0.705 & -0.404 \\
-0.602 & 0.241 & -0.288 & 0.705 \\
0.562 & -0.465 & -0.542 & 0.417
\end{bmatrix}
$$

Figure 11.8. Singular value decomposition of initial data.

representation to each document where each document is described with corresponding column. Each of the new dimensions is derived from the original 10 word count dimensions. For example, document 1 is now represented as follows: $d_1 = -0.56x_1 + 0.095x_2 - 0.602x_3 + 0.562x_4$$d_1 = -0.56x_1 + 0.095x_2 - 0.602x_3 + 0.562x_4$, where each x_n represents one of the newly derived dimensions. The **S** matrix is a diagonal matrix of eigenvalues for each principal component direction.

With this initial step, we have already reduced the number of dimensions representing each document from 10 to 4. We now show how one would go about further reducing the number of dimensions. When the data set contains many documents, the previous step is not enough to reduce the number of dimensions meaningfully. To perform further reduction, we first observe that matrix **S** provides us with a diagonal matrix of eigenvalues in descending order as follows: $\lambda_1,\ldots,\lambda_4 = \{3.869, 2.344, 1.758, 0.667\}$. We will be able to capture most of the variability of the data by retaining only the first k eigenvalues rather than all n terms (in our matrix S, $n = 4$). If for example $k = 2$, then we retain $(\lambda_1^2 + \lambda_2^2)/\sum_{i=1}^{4}\lambda_i^2 = 0.853$ or 85% of the variability when we move from the four new dimensions per document down to only two. The rank 2 approximations can be seen in Figure 11.9. This rank 2 approximation is achieved

$$U \approx U_k = \begin{vmatrix} -0.575 & 0.359 \\ -0.504 & -0.187 \\ -0.162 & 0.245 \\ -0.197 & -0.473 \\ -0.105 & -0.172 \\ -0.236 & -0.260 \\ -0.307 & 0.286 \\ -0.197 & -0.473 \\ -0.307 & 0.286 \\ -0.236 & -0.260 \end{vmatrix}, S \approx S_k = \begin{vmatrix} 3.869 & 0.000 \\ 0.000 & 2.344 \end{vmatrix}, V^T \approx V_k^T = \begin{vmatrix} -0.560 & -0.628 & -0.354 & -0.408 \\ 0.095 & 0.575 & -0.705 & -0.404 \end{vmatrix}$$

Figure 11.9. Rank 2 approximation of the singular value decomposition.

by selecting the first k columns from **U**, the upper left $k \times k$ matrix from **S**, and the top k rows from $\mathbf{V^T}$ as is shown in Figure 11.9.

The rank k approximation to $\mathbf{V^T}$ gives us our dimensionally reduced data set where each document is now described with only two dimensions. $\mathbf{V^T}$ can be thought of as a transformation of the original document matrix and can be used for any of a number of text-mining tasks such as classification and clustering. Improved results may be obtained using the newly derived dimensions, comparing against the data-mining tasks using the original word counts.

In most text-mining cases, all training documents (in our case 4) would have been included in matrix A and been transformed together. Document five (d_5) was intentionally left out of these calculations to demonstrate a technique called "folding in," which allows for a small number of documents to be transformed into the new reduced space and compared with training documents' database. Matrices from our previous SVD calculations are used for this transformation. This is done using the following modified formula: $V'^T = A'^T U_k S_k^{-1}$. This equation is a rearranging of terms from the initial SVD equation replacing the original data set, matrix A, with the term counts for document five (d_5) as matrix A'. The resulting multiplication can be seen in Figure 11.10. The result is the document d_5 represented in the reduced 2D space, $d_5 = [-0.172, 0.025]$.

To visualize the transformed data, we have plotted our example documents, $d_{1\ldots5}$, in Figure 11.11. We now perform the nearest neighbor classification algorithm. If the task is to disambiguate the term "bank" and the possible classes are "financial institution" and "basketball shot," then a review of the original text reveals that documents d_1, d_2, and d_5 are of class "financial institution" and documents d_3 and d_4 are of the class "basketball shot." To classify the test document d_5, we compare d_5 with other document. The closest neighbor will determine the class of d_5. Ideally, d_5 should be nearest to d_1 and d_2 and furthest from d_3 and d_4 as pointed earlier. Table 11.4 shows these calculations using the Euclidean distance metric. The assessment for which document is closest is made based on both the original 10 dimensions and the reduced set of 2 dimensions. Table 11.5 shows the same comparison using cosine similarity to compare documents. Cosine similarity is often used in text-mining tasks for document comparisons.

$$V^T = A^T U_k S_k^{-1}$$

$$V^T = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} -0.575 & 0.359 \\ -0.504 & -0.187 \\ -0.162 & 0.245 \\ -0.197 & -0.473 \\ -0.105 & -0.172 \\ -0.236 & -0.260 \\ -0.307 & 0.286 \\ -0.197 & -0.473 \\ -0.307 & 0.286 \\ -0.236 & -0.260 \end{bmatrix} \begin{bmatrix} 0.258 & 0.000 \\ 0.000 & 0.427 \end{bmatrix} = \begin{bmatrix} -0.172 & 0.025 \end{bmatrix}$$

Figure 11.10. SVD calculations to produce 2D approximation for d_5.

Figure 11.11. 2D plot of documents and query using LSA.

Euclidean distance as shown in Table 11.4 when using 10 dimensions ranks d_3 and d_4 above d_1 and d_2. By the formulation of Euclidean distance when two documents both share a term or both do not share a term, the result is a distance of zero for that dimension. After applying the LSA transformation, the Euclidean distance ranks d_1 above d_3 and d_4. However, document d_2 only is ranked above d_3 and not d_4.

Cosine similarity calculates the cosine of the angle between two vectors representing points in n-dimensional space. The result is a similarity score between 0

TABLE 11.4. Use of Euclidean Distance to Find Nearest Neighbor to d_5 in Both 2D and 10D (The Smallest Distance Ranks First)

Comparison	10D (Original)		2D (LSA)	
	Dist.	Rank	Dist.	Rank
d_1–d_5	2.449	3–4	0.394	1
d_2–d_5	2.449	3–4	0.715	3
d_3–d_5	2.236	1–2	0.752	4
d_4–d_5	2.236	1–2	0.489	2

TABLE 11.5. Use of Cosine Similarity to Find Most Similar Document to d_5 in Both 2D and 10D (The Largest Similarity Ranks First)

Comparison	10D (Original)		2D (LSA)	
	Sim.	Rank	Sim.	Rank
d_1–d_5	0.289	4	0.999	1
d_2–d_5	0.500	1	0.826	2
d_3–d_5	0.316	2–3	0.317	4
d_4–d_5	0.316	2–3	0.603	3

and 1, where a 1 shows highest similarity and 0 shows no similarity. For document vector comparisons, no additional strength of similarity is added by two vectors containing a zero for a specific term. This is very beneficial characteristics for textual applications. When we consider Table 11.5 we see that without the transformation, in ten-dimensional space, d_2 is ranked above d_3 and d_4 for containing both terms had in d_5. However, d_1 is ranked below d_3 and d_4 for having more terms than these other sentences. After the LSA transformation, d_1 and d_2 are ranked above d_3 and d_4, providing the expected ranking. In this simple example the best results occurred when we first transformed the data using LSA and then used cosine similarity to measure the similarity between initial, training documents in a database and a new document for comparison. Using the nearest neighbor classifier, d_5 would be classified correctly as "financial institution" document using cosine similarity for both the 10D and 2D case or using Euclidean distance for the 2D case. Euclidean distance in the 10D case would have classified d_5 incorrectly. If we used k-nearest neighbor with $k = 3$, then Euclidean distance in the 10D case would have also incorrectly classified d_5. Clearly, the LSA transformation affects the results of document comparisons, even in this very simple example. Results are better because LSA enable better representation of documents' semantics.

11.9 REVIEW QUESTIONS AND PROBLEMS

1. Give specific examples of where Web content mining, Web structure mining, and Web usage mining would be valuable. Discuss the benefits.

2. Given a table of linked Web pages:

Page	Linked to Page
A	B, D, E, F
B	C, D, E
C	B, E, F
D	A, F, E
E	B, C, F
F	A, B

   (a) Find authorities using two iterations of the HITS algorithm.
   (b) Find hubs using two iterations of the HITS algorithm.
   (c) Find the PageRank scores for each page after one iteration using 0.1 as the dampening factor.
   (d) Explain the HITS and PageRank authority rankings obtained in a and c.

3. For the traversal log: {X, Y, Z, W, Y, A, B, C, D, Y, C, D, E, F, D, E, X, Y, A, B, M, N},

   (a) Find maximal forward references.
   (b) Find large reference sequences if the threshold value is 0.3 (or 30%).
   (c) Find maximal reference sequences.

4. Given the following text documents and assumed decomposition:

Document	Text
A	Web content mining
B	Web structure mining
C	Web usage mining
D	Text mining

$$
USV^T=
\begin{array}{cccccccccccc}
-0.60 & 0.43 & 0.00 & 0.00 & 2.75 & 0.00 & 0.00 & 0.00 & -0.55 & -0.55 & -0.55 & -0.30 \\
-0.20 & 0.14 & 0.00 & 0.82 & 0.00 & 1.21 & 0.00 & 0.00 & 0.17 & 0.17 & 0.17 & -0.95 \\
-0.71 & -0.36 & 0.00 & 0.00 & 0.00 & 0.00 & 1.00 & 0.00 & 0.00 & -0.71 & 0.71 & 0.00 \\
-0.20 & 0.14 & --.71 & -0.41 & -0.00 & 0.00 & 0.00 & 1.00 & 0.82 & -0.41 & -0.41 & 0.00 \\
-0.20 & 0.14 & 0.71 & -0.41 \\
-0.11 & -0.79 & 0.00 & 0.00 \\
\end{array}
$$

   (a) Create matrix A by using term counts from the original documents.
   (b) Obtain rank 1, 2, and 3 approximations to the document representations.
   (c) Calculate the variability preserved by rank 1, 2, and 3 approximations.
   (d) Manually cluster documents A, B, C, and D into two clusters.

5. Given a table of linked Web pages and a dampening factor of 0.15:

Page	Linked to Page
A	F
B	F
C	F
D	F
E	A, F
F	E

(a) Find the PageRank scores for each page after one iteration.

(b) Find the PageRank scores after 100 iterations recording the absolute difference between scores per iteration (be sure to use some programming or scripting language to obtain these scores).

(c) Explain the scores and rankings computed previously in parts (a) and (b). How quickly would you say that the scores converged? Explain.

6. Why is the text-refining task very important in a text-mining process? What are results of text refining?

7. Implement the HITS algorithm, and discover authorities and hubs if the input is the table of linked pages.

8. Implement the PageRank algorithm and discover central nodes in a table of linked pages.

9. Develop a software tool for discovering maximal reference sequences in a Web log file.

10. Search the Web to find the basic characteristics of publicly available or commercial software tools for association rule discovery. Document the results of your search.

11. Apply LSA to 20 Web pages of your choosing and compare the clusters obtained using the original term counts as attributes against the attributes derived using LSA. Comment on the successes and shortcomings of this approach.

12. What are the two main steps in mining-*traversal patterns* using log data?

13. The XYZ Corporation maintains a set of five Web pages: {A, B, C, D, and E}. The following sessions (listed in timestamp order) have been created:

$$S1 = [A, B, C], S2 = [A, C], S3 = [B, C, E], \text{ and } S4 = [A, C, D, C, E].$$

Suppose that support threshold is 30%. Find all *large sequences* (after building the tree!).

14. Suppose a Web graph is undirected, i.e. page i points to page j if and only if page j points to page i. Are the following statements true or false? Justify your answers briefly.

    (a) The hubbiness and authority vectors are identical; i.e for each page, its hubbiness is equal to its authority.
    (b) The matrix M that we use to compute PageRank is symmetric; i.e. $M[i; j] = M[j; i]$ for all i and j.

15. Consider three Web pages A, B, and C with the following links:

    Suppose we compute PageRank with $d = 0.7$ and we introduce the additional constraint that the sum of the PageRanks of the three pages must be normalized to 3. Compute the PageRanks model for ranks defined as a, b, and c of the three pages A, B, and C, only in the first iteration.

16. Imagine that you have been given a data set of 1,000 documents that have been classified as being about entertainment or education. There are 700 entertainment documents in the data set and 300 education documents in the data set. The tables below give the number of documents from each topic that a selection of words occurred in.

Word-document counts for the *entertainment* dataset					
fun	is	machine	christmas	family	learning
415	695	35	0	400	70

Word-document counts for the *education* dataset					
fun	is	machine	christmas	family	learning
200	295	120	0	10	105

    What target level will a naive Bayes model predict for the following query document: "*machine learning is fun*" in both data sets?

17. Compute the cosine measure, using the raw frequencies of words, between the following two sentences:

    (a) "*The sly fox jumped over the lazy dog.*"
    (b) "*The dog jumped at the intruder.*"

11.10 REFERENCES FOR FURTHER STUDY

1. Han, J., M. Kamber, *Data Mining: Concepts and Techniques*, 3rd edition, Morgan Kaufmann, San Francisco, 2011.

   This book gives a sound understanding of data-mining principles. The primary orientation of the book is for database practitioners and professionals with emphasis on OLAP and data warehousing. In-depth analysis of association rules and clustering algorithms is the additional strength of the book. All algorithms are presented in easily understood pseudocode, and they are suitable for use in real-world, large-scale data-mining projects including advanced applications such as Web mining and text mining.

2. Chang, G., M. J. Haeley, J. A. M. McHugh, J. T. L. Wang, *Mining the World Wide Web: An Information Search Approach*, Kluwer Academic Publishers, Boston, MA, 2001.

   This book is an effort to bridge the gap between information search and data mining on the Web. The first part of the book focuses on information retrieval on the Web. The ability to find relevant documents on the Web is essential to the process of Web mining. The cleaner the set of Web documents and data, the better the knowledge that can be extracted from it. In the second part of the book, basic concepts and techniques on text mining, Web mining, and Web crawling are introduced. A case study, in the last part of the book, focuses on a search engine prototype called EnviroDaemon.

3. Akerkar, R., P. Lingras, *Building an Intelligent Web: Theory and Practice*, Jones and Bartlett Publishers, Sudbury, MA, 2008.

   This provides a number of techniques used in Web mining. Code is provided along with illustrative examples showing how to perform Web content mining, Web structure mining, and Web usage mining.

4. Zhang Q., R. S. Segall, Review of Data, Text and Web Mining Software, *Kybernetes*, Vol. 39, No. 4, 2010, pp. 625–655.

   The paper reviews and compares selected software for data mining, text mining (TM), and web mining that are not available as free open-source software. The software for data mining are SAS® Enterprise Miner™, Megaputer PolyAnalyst® 5.0, NeuralWare Predict®, and BioDiscovery GeneSight®. The software for TM are Compare Suite, SAS® Text Miner, TextAnalyst, VisualText, Megaputer PolyAnalyst® 5.0, and WordStat. The software for Web mining are Megaputer PolyAnalyst®, SPSS Clementine®, ClickTracks, and QL2. The paper discusses and compares the existing features, characteristics, and algorithms of selected software for data mining, TM, and web mining, respectively.

5. Struhl Steven, *Practical Text Analytics: Interpreting Text and Unstructured Data for Business Intelligence*, Kogan Page Limited, July 2015.

   Bridging the gap between the marketer who must put text analytics to use and data analysis experts, *Practical Text Analytics* is an accessible guide to the many

advances in text analytics. It explains the different approaches and methods and their uses, strengths, and weaknesses, in a way that is relevant to marketing professionals. Each chapter includes illustrations and charts, hints and tips, pointers on the tools and techniques, definitions, and case studies/examples. Consultant and researcher Steven Struhl presents the process of text analysis in ways that will help marketers clarify and organize the confusing array of methods, frame the right questions, and apply the results successfully to find meaning in any unstructured data and develop effective new marketing strategies.

6. Aggarwal C. C., C. Zhai, *Mining Text Data*, Springer, Heidelberg, 2012.

Text-mining applications have experienced tremendous advances because of Web 2.0 and social networking applications. Recent advances in hardware and software technology have led to a number of unique scenarios where text-mining algorithms are learned. *Mining Text Data* introduces an important niche in the text analytics field and is an edited volume contributed by leading international researchers and practitioners focused on social networks and data mining. This book contains a wide swath in topics across social networks and data mining. There is a special focus on Text Embedded with Heterogeneous and Multimedia Data, which makes the mining process much more challenging.

7. Aggarwal C. C., *Recommender Systems*, Springer, Heidelberg, 2016.

This book comprehensively covers the topic of recommender systems, which provide personalized recommendations of products or services to users based on their previous searches or purchases. Recommender system methods have been adapted to diverse applications including query log mining, social networking, news recommendations, and computational advertising. This book synthesizes both fundamental and advanced topics of a research area that has now reached maturity. The chapters of this book are organized into three categories: (1) algorithms and evaluation, (2) recommendations in specific domains and contexts, and (3) advanced topics and applications.

12

ADVANCES IN DATA MINING

Chapter Objectives

- Analyze the characteristics of graph-mining algorithms and introduce some illustrative examples.
- Identify the required changes in data-mining algorithm when temporal and spatial components are introduced.
- Introduce basic characteristics of distributed-data-mining algorithms and specific modifications for distributed DBSCAN clustering.
- Describe the differences between causality and correlation.
- Introduce basic principles in Bayesian network modeling.
- Know when and how to include privacy protection in data-mining process.
- Summarize social and legal aspects of data-mining applications.
- Highlights the concepts of cloud computing, Hadoop framework, and Map/Reduce programming paradigm.
- Explain basic principles of reinforcement learning and insight Q-learning methodology.

Current technological progress permits the storage and access of large amounts of data at virtually no cost. These developments have created unprecedented opportunities for large-scale data-driven discoveries, as well as the potential for fundamental gains in scientific and business understanding. The popularity of the Internet and the Web makes it imperative that the data-mining framework is extended to include distributed, time- and space-dependent information and tools. New complex and distributed systems are supported by enhanced multimedia data sources such as images and signals and advanced data structures such as graphs. In this environment, data-mining applications have new social and legal challenges, and privacy preserving is one of the priority tasks.

12.1 GRAPH MINING

Traditional data-mining tasks such as association rule mining, market basket analysis, and cluster analysis commonly attempt to find patterns in a data set characterized by a collection of independent instances of a single relation. This is consistent with the classical statistical inference problem of trying to identify a model given a random sample from a common underlying distribution. An emerging challenge for data mining is the problem of mining richly structured data sets, where the objects are linked in some way. Many real-world data sets describe a variety of entity types linked via multiple types of relations. These links provide additional context that can be helpful for many data-mining tasks. Yet multi-relational data violates the traditional assumption of independent, identically distributed data instances that provides the basis for many statistical machine-learning algorithms. Naively applying traditional statistical inference procedures, which assume that samples are independent, may lead in many applications to inappropriate conclusions. Care must be taken that potential correlations due to links between samples are handled appropriately. In fact, record linkage is knowledge that should be exploited. Clearly, this is information that can be used to improve the predictive accuracy of the learned models: attributes of linked objects are often correlated and links are more likely to exist between objects that have some commonality. Relationships between objects represent a rich source of information and ultimately knowledge. Therefore, new approaches are needed that can exploit the dependencies across the attribute and link structure. Certainly, as a general data structure, graph can meet demands of modeling complicated relations among data.

   Graph-based data mining represents a collection of techniques for mining the relational aspects of data represented as a graph. It has the task of finding novel, useful, and understandable graph-theoretic patterns in a graph representation of data. Graph mining has become an important topic of research recently because of numerous applications to a wide variety of data-mining problems in computational biology, chemical data analysis, drug discovery, and communication networking. Some examples of graph represented data are presented in Figure 12.1. Traditional data-mining

Figure 12.1. Graph representation of data. (a) Chemical compound. (b) Social network (c) Genome co-expression network.

and management algorithms such as clustering, classification, frequent pattern mining, and indexing have now been extended to the graph scenario. While the field of graph mining has been a relatively recent development in the data-mining community, it has been studied under different names by other groups of researchers. This is because research on graphs has a long history in mathematics, but most notably important results are obtained by sociologists in the field of a social network analysis. However, there are important differences, and the primary one is that of network size. Social networks are, in general, small with the larger studies considering a few hundred nodes. On the other hand, graph-mining data sets in new application domains may typically consist of hundreds of thousands of nodes and millions of edges.

Many data sets of interest today are best described as a linked collection of inter-related objects. These may represent homogeneous networks, in which there is a single-object type and single-link type, or richer heterogeneous networks, in which there may be multiple-object and multiple-link types and possibly other semantic information. Examples of homogeneous networks include single mode social networks, such as people connected by friendship links, or the WWW, a collection of linked Web pages. Examples of heterogeneous networks include those in medical domains describing patients, diseases, treatments, and contacts or in bibliographic domains describing publications, authors, and venues. Graph-mining techniques explicitly consider these links when building predictive or descriptive models of the linked data.

The requirement of different applications with graph-based data sets is not very uniform. Thus, graph models and mining algorithms that work well in one domain may not work well in another. For example, chemical data is often represented as graphs in which the nodes correspond to atoms and the links correspond to bonds between the atoms. The individual graphs are quite small, though there are significant repetitions among the different nodes. Biological data is modeled in a similar way as chemical data. However, the individual graphs are typically much larger. Protein interaction networks link proteins that must work together to perform some particular biological function. A single biological network could easily contain thousands of nodes. In the case of computer networks and the Web, the number of nodes in the

underlying graph may be massive. Computer networks consist of routers/computers representing nodes and the links between them. Since the number of nodes is massive, this can lead to a very large number of distinct edges. Social networks may be modeled with large graphs, which are defined by people who appear as nodes, and links, which correspond to communications or relationships between these different people. The links in the social network can be used to determine relevant communities, members with particular expertise sets, and the flow of information in the social network. For example, the problem of community detection in social networks is related to the problem of node clustering of very large graphs. In this case, we wish to determine dense clusters of nodes based on the underlying linkage structure. It is clear that the design of a particular mining algorithm depends upon the application domain at hand.

Before introducing some illustrative examples of graph-mining techniques, it will summarize some basic concepts from graph theory. Graph theory provides a vocabulary that can be used to label and denote many structural properties in data. Also, graph theory gives us mathematical operations and ideas with which many of these properties can be quantified and measured.

A graph $G = G(N, L)$ consists of two sets of information: a set of nodes $N = \{n_1, n_2,…,n_k\}$ and a set of links between pairs of nodes $L = \{l_1, l_2,…,l_m\}$. A graph with nodes and without links is called an empty graph, while the graph with only one node is a trivial graph. Two nodes, n_i and n_j, are *adjacent* if there is a link between them. A graph $G(N, L)$ can be presented as a diagram in Figure 12.2a in which points depict nodes and lines between two points are links. A graph $G'(N', L')$ is a subgraph of $G(N, L)$ if $N' \subseteq N$ and $L' \subseteq L$.

An induced subgraph of a graph G has a subset of the nodes of G and the same links between pairs of nodes as in G. For example, the subgraph (b) in Figure 12.3 is an induced subgraph of the graph (a), but the subgraph (c) is a general subgraph but not an induced subgraph of G, since the incoming link L_1 of the node labeled as N_2 in (a) is not retained in (c) while the node labeled as N_2 is included.

The degree of a node, denoted by $d(n_i)$, is the number of links connected to the given node. Equivalently, the degree of a node is the number of nodes adjacent to it. For example, in Figure 12.2a, the degree $d(B) = 3$. The degree of a node ranges from a minimum of 0, if no nodes are adjacent to a given node, to a maximum $k - 1$, if a given

Figure 12.2. (a) Undirected graph. (b) Directed Graph. (c) Weighted Graph.

Figure 12.3. (a) Graph. (b) Induced Subgraph. (c) General Subgraph (but not Induced).

node is adjacent to all other nodes in the graph. The degrees are very easy to compute, and yet they can be very informative for many applications. For some applications, it is useful to summarize the degrees of all nodes in the graph. The mean nodal degree is a statistic that reports the average degree of the nodes in the graph:

$$d_{av} = \frac{\sum_{i-1}^{k} d(n_i)}{k}$$

For the graph in Figure 12.2a, $d_{av} = (1 + 3 + 2 + 2)/4 = 2$. One might also be interested in the variability of the nodal degrees. If all the degrees of all nodes are equal, the graph is said to be d-regular, and its variability is equal to 0. If the nodes differ in degrees, the variance is calculated as a measure for variability:

$$SD^2 = \sum_{i=1}^{k} \frac{(d(n_i) - d_{av})^2}{k}$$

The maximum number of links in the graph is defined by the number of nodes. Since there are k nodes in the graph, and if we exclude the loops as links, there are $k(k-1)/2$ possible links that could be presented in the graph. If all links are present, then all nodes are adjacent, and the graph is said to be complete. Consider now what proportion of these links is actually present. The density of a graph is the proportion of actual links in the graph to maximum possible number of links. The density of a graph goes from 0 when there are no links in a graph to 1 when all possible links are presented.

We can also analyze the paths between a pair of nodes, and they are represented by multiple links. We define a path from $s \in N$ to $t \in N$ as an alternating sequence of nodes and links, beginning with node s and ending with node t, such that each link connects its preceding with its succeeding node. It is likely that there are several paths between given pair of nodes and that these paths are differ in lengths (number of links included in the path). A shortest path between two nodes is referred as a *geodesic*. The geodesic distance d_G or simply the distance between two nodes is defined as the length of a geodesic between them, and it represents the length of the shortest path. By definition, $d_G(s; s) = 0$ for every node $s \in N$, and $d_G(s; t) = d_G(t; s)$ for each pair of nodes s;

$t \in V$. For example, the paths between nodes A and D in Figure 12.2a are A–B–D and A–B–C–D. The shortest one is A–B–D, and therefore the distance $d(A,D) = 2$. If there is no path between two nodes, then the distance is infinite (or undefined). The *diameter* of a connected graph is the length of the largest geodesic between any pairs of nodes. It represents the largest nodal eccentricity. The diameter of a graph can range from 1, if the graph is complete, to a maximum of $k - 1$. If a graph is not connected, its diameter is infinite. For the graph in Figure 12.2a, the diameter is 2.

These basic definitions are introduced for non-labeled and non-directed graphs such the one in Figure 12.2a. A *directed graph*, or digraph $G(N, L)$ consists of a set of nodes N and set of directed links L. Each link is defined by an order pair of nodes (sender, receiver). Since each link is directed, there are $k(k - 1)$ possible links in the graph. A *labeled graph* is a graph in which each link carries some value. Therefore, a labeled graph G consists of three set of information: $G(N,L,V)$, where the new component $V = \{v_1, v_2,\ldots,v_t\}$ is a set of values attached to links. An example of a directed graph is given in Figure 12.2b, while the graph in Figure 12.2c is a labeled graph. Different applications use different types of graphs in modeling linked data. In this chapter a primary focus is on undirected and unlabeled graphs, although the reader still has to be aware that there are numerous graph-mining algorithms for directed and/or labeled graphs.

Besides a graphical representation, each graph may be presented in a form of the *incidence matrix I(G)* where nodes are indexing the rows and links are indexing columns. The matrix entry in the position (i,j) has value a if node n_i is incident with a the link l_j. The other matrix representation of a graph (in a case of undirected and unlabeled graphs) is $k \times k$ adjacency matrix where both rows and columns are defined by nodes. The graph-structured data can be transformed without much computational effort into an *adjacency matrix*, which is a well-known representation of a graph in mathematical graph theory. On the intersection (i,j) is the value of 1 if the nodes n_i and n_j are connected by a link; otherwise it is 0 value (Figure 12.4b).

If a graph has labeled links, the following conversion of the graph to a new graph that has labels at its nodes only is applied. This transformation reduces the ambiguity in the adjacency matrix representation. Given a node pair (u, v) and a directed or undirected link $\{u, v\}$ between the nodes, i.e., *node(u)−link({u, v})−node(v)* where *node()* and *link()* stand for the labels of a node and a link, respectively. The *link()* label information can be removed and transformed into a new *node(u, v)*, and the following triangular graph may be deduced where the original information of the node pair and the link between them is preserved.

(a)

$$\begin{array}{c c c c c} & l_1 & l_2 & l_3 & l_4 \\ n_1 & \begin{bmatrix} a & b & 0 & 0 \\ n_2 & 0 & b & 0 & 0 \\ n_3 & 0 & 0 & b & 0 \\ n_4 & a & 0 & b & c \\ n_5 & 0 & 0 & 0 & c \end{bmatrix} \end{array}$$

(b)

$$\begin{array}{c c c c c} & n_1 & n_2 & n_3 & n_4 \\ n_1 & \begin{bmatrix} 0 & 0 & 0 & 1 \\ n_2 & 0 & 0 & 1 & 0 \\ n_3 & 0 & 1 & 0 & 1 \\ n_4 & 1 & 0 & 1 & 0 \end{bmatrix} \end{array}$$

Figure 12.4. Matrix representations of graphs. (a) Incidence matrix: nodes × links. (b) Adjacency matrix: nodes × nodes.

This operation on each link in the original graph can convert the graph into another graph representation having no labels on the links while preserving the topological information of the original graph.

Accordingly, the adjacency matrix of Figure 12.5a is converted to that of Figure 12.5b as follows:

$$\begin{array}{c c c c} & N_1 & N_1 & N_2 \\ N_1 & \begin{bmatrix} L_2 & 0 & 0 \\ N_1 & 0 & 0 & L_1 \\ N_2 & L_1 & 0 & 0 \end{bmatrix} \end{array} \Rightarrow \begin{array}{c c c c c c c} & N_1 & N_1 & N_2 & L_1 & L_1 & L_2 \\ N_1 & \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 \\ N_1 & 0 & 0 & 1 & 0 & 1 & 0 \\ N_2 & 1 & 0 & 0 & 1 & 0 & 0 \\ L_1 & 1 & 0 & 0 & 0 & 0 & 0 \\ L_1 & 0 & 0 & 1 & 0 & 0 & 0 \\ L_2 & 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

The aforementioned explanation was for directed graphs. But the identical pre-processing may be applied to undirected graphs. The difference from the case of directed graphs is that the adjacency matrix becomes diagonally symmetric. The adjacency matrix of Figure 12.6a is represented at Figure 12.6b.

For the sake of the efficiency in memory consumption but also for more efficient computations with graphs, we define a code representation of an adjacency matrix as follows. In case of an undirected graph, the code of an adjacency matrix X_k, i.e., *code* (X_k), is represented by a binary number. It is obtained by scanning the upper triangular elements of the matrix along each column until diagonal elements (for undirected graph). For example, the code of the adjacency matrix in Figure 12.6b is given by

$$\text{Code}(X_k) = 0101100101$$

Variety of operations is defined in graph theory, and corresponding algorithms are developed to perform efficiently these operations. The algorithms are working

(a)

(b)

Figure 12.5. Preprocessing of labeled links and self-looped nodes in a graph. (a) Original graph. (b) Graph with preprocessed labeled links.

(a)

(b)

$$
\begin{array}{c|ccccc}
 & N_1 & N_2 & N_3 & N_3 & N_4 \\
\hline
N_1 & 0 & 0 & 1 & 1 & 0 \\
N_2 & 0 & 0 & 0 & 1 & 1 \\
N_3 & 1 & 0 & 0 & 0 & 0 \\
N_3 & 1 & 1 & 0 & 0 & 1 \\
N_4 & 0 & 1 & 0 & 1 & 0
\end{array}
$$

Figure 12.6. Undirected graph representations. (a) Original Graph. (b) Adjacency Matrix.

with graphical, matrix, or code representations of graphs. One of very important operations is join operation of two graphs and forming new, more complex graph. This operation is used in many graph-mining algorithms including frequent patterns mining in graphs. The join operation is demonstrated through the example depicted in Figure 12.7. The examples of the adjacency matrices X_4, Y_4, and join result Z_5 are given in (d) representing the graphs at (a), (b), and (c).

Graph analysis includes a number of parameters that describe important characteristics of a graph, and they are used as fundamental concepts in developing graph-mining algorithms. Over the years, graph-mining researchers have introduced a large number of *centrality indices*, measures of the importance of the nodes in a graph according to one criterion or another.

Perhaps the simplest centrality measure is *degree*, which is the number of links for a given node. Degree is a measure in some sense of the "popularity" of a node in the presented graph. Nodes with high degree are considered to be more central. However, this weights a node only by its immediate neighbors and not by, for example, its 2-hop and 3-hop neighbors. A more sophisticated centrality measure is *closeness*, which is the mean geodesic (i.e. shortest path) distance between a vertex and all other vertices reachable from it. Examples of computations for both measures are given in

(a)   (b)   (c)

(d)

$$X_4 = \begin{array}{c} \\ N_1 \\ N_2 \\ N_3 \\ N_3 \end{array} \begin{array}{cccc} N_1 & N_2 & N_3 & N_3 \\ \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix} \end{array}$$

$$Y_4 = \begin{array}{c} \\ N_1 \\ N_2 \\ N_3 \\ N_3 \end{array} \begin{array}{cccc} N_1 & N_2 & N_3 & N_3 \\ \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{bmatrix} \end{array}$$

$$Z_5 = \begin{array}{c} \\ N_1 \\ N_2 \\ N_3 \\ N_3 \\ N_3 \end{array} \begin{array}{ccccc} N_1 & N_2 & N_3 & N_3 & N_3 \\ \begin{bmatrix} 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & ? \\ 1 & 1 & 0 & ? & 0 \end{bmatrix} \end{array}$$

Figure 12.7. An example of join operation between graphs (a) and (b). (d) Matrix representation of a Join operation.

Figure 12.8. Closeness can be regarded as a measure of how long it will take information to spread from a given node to others in the graph. Nodes that have low distance to all other nodes in the graph have high closeness centrality.

Another important class of centrality measures is the class of *betweenness* measures. Betweenness is a measure of the extent to which a vertex lies on the paths between others. The simplest and most widely used betweenness measure is *shortest-path betweenness*, usually called simply *betweenness*. The betweenness of a node i is defined to be the fraction of shortest paths between any pairs of vertices in a graph that pass through node i. This is, in some sense, a measure of the influence a node has over the spread of connections through the network. Nodes with high betweenness values occur on larger number of shortest paths and are presumably more important than nodes with low betweenness. The parameter is costly to compute especially when the graphs are complex with large number of nodes and links. Currently, the fastest known algorithms require $O(n^3)$ time complexity and $O(n^2)$ space complexity, where n is the number of nodes in the graph.

Illustrative examples of centrality measures and their interpretations are given in Figure 12.9. Node X has importance because it bridges the structural hole between the two clusters of interconnected nodes. It has the highest betweenness measure comparing to all other nodes in the graph. Such nodes get lots of brokerage opportunities and can control the flow in the paths between subgraphs. On the other hand, node Y is in the middle of a dense web of nodes, which provides easy, short path access to neighboring nodes; thus Y also has a good central position in the subgraph. This characteristic of the node Y is described with the highest degree measure.

	Degree	Closeness
N_1	6	15/9 = 1.66
N_2	5	14/9 = 1.55
N_3	4	17/9 = 1.88

Figure 12.8. Degree and closeness parameters of the graph.

Figure 12.9. Different types of node's importance in a graph.

Vertex-betweenness index reflects the amount of control exerted by a given vertex over the interactions between the other vertices in the graph. The other approach to measure betweenness is to concentrate on links in the graph instead of nodes. Edge-betweenness centrality is related to the frequency of an edge that places on the shortest paths between all pairs of vertices. The betweenness centrality of an edge in a network is given by the sum of the edge-betweenness values for all pairs of nodes in the graph going through the given edge. The edges with highest betweenness values are most likely to lie between subgraphs, rather than inside a subgraph. Consequently, successively removing edges with the highest edge betweenness will eventually isolate subgraphs consisting of nodes that share connections only with other nodes in the same subgraph. This gives edge-betweenness index a central role in graph clustering algorithms where it is necessary to separate large graph into smaller highly connected subgraphs. Edge (also called link) betweenness centrality is traditionally determined in two steps:

1. Compute the length and number of shortest paths between all pairs of nodes through the link.
2. Sum all link dependencies.

(a)   (b)

Figure 12.10. Preparatory phase for link betweenness computation. (a) Initial graph. (b) Transformed graph with the root in node **A.**

The overall betweenness centrality of a link *v* is obtained by summing up its partial betweenness values for this link calculated using the graph transformation on breadth first strategy from each node. For example, if at the beginning it is given a graph in Figure 12.10a, it is necessary to find link betweenness measures for all links in the graph. In the first step, we are building "modified" graph, which is starting with node *A*, and specify all links in the graph, layer by layer: first neighbors, second neighbors, etc. The resulting graph is given in Figure 12.10b. This graph is starting point for *partial* betweenness computation. The total betweenness will be sum of partial scores obtained for transformed graphs with root nodes *A* to *K*. The process with each transformed graph in (b) consists of:

- forward phase, and
- backward phase.

It will be illustrated with activities on the graph in Figure 12.10b. In the forward phase, the count of shortest paths from *A* to all other nodes of the network is determined. The computation is performed iteratively, layer by layer. For example, the number of shortest paths from the initial node *A* to the node *I* is computed based on the number of shortest paths from node *A* to the nodes *F* and *G*.

The result of the completed forward phase is given in Figure 12.11a. Each node is labeled with the number of shortest paths from the root node *A*. For example, the node *J* has three shortest paths, two of them through the node *H* (ADHJ and AEHJ) and one through the node *G* (ADGJ).

Backward phase is starting from the bottom of the layered graph structure, in our example from the node *K*. If there are multiple paths from the given node up, count betweenness measure of each link fractionally. The proportion is determined by the number of shortest paths to these nodes on the previous layer. What amount we are

(a)

(b)

Figure 12.11. Computation of the partial link betweenness measure. (a) Forward phase. (b) Backward phase.

splitting between these links? The amount is defined as (*1 + sum of all betweenness measures entering into the node from below*). For example, from the node *K*, there are two paths toward nodes *I* and *J*, and because both nodes have the same number of shortest paths (3), the amount we are splitting is $1 + 0 = 1$, and the partial betweenness measures for links IK and JK are 0.5. In a similar way we may compute betweenness measures for node *G*. Total betweenness value for splitting is $1 + 0.5 + 0.5 = 2$. There is only one node up; it is *D* and link betweenness measure for GD is 2.

When we computed betweenness measures for all links in the graph, the procedure should be repeated for other nodes in the graph as the root nodes, until it is explored each node of the network. Finally, all partial link scores determined for different graphs should be added to determine final link betweenness score.

Graph-mining applications are far more challenging to implement because of the additional constraints that arise from the structural nature of the underlying graph. The problem of frequent pattern mining has been widely studied in the context of mining transactional data. Recently, the techniques for frequent pattern mining have also been extended to the case of graph data. This algorithm attempts to find interesting or commonly occurring subgraphs in a set of graphs. Discovery of these patterns may be the sole purpose of the systems, or the discovered patterns may be used for graph classification or graph summarization. The main difference in the case of graphs is that the process of determining support is quite different. The problem can be defined in different ways depending upon the application domain. In the first case, we have a group of graphs, and we wish to determine all patterns that support a fraction of the corresponding graphs. In the second case, we have a single large graph, and we wish to determine all patterns that are supported at least a certain number of times in this large graph. In both cases, we need to account for the isomorphism issue in determining

whether one graph is supported by another. However, the problem of defining the support is much more challenging, if overlaps are allowed between different embeddings. Frequently occurring subgraphs in a large graph or a set of graphs could represent important motifs in the real-world data.

Apriori-style algorithms can be extended to the case of discovering frequent subgraphs in graph data by using a similar level-wise strategy of generating $(k+1)$-candidates from k-patterns. Various measures to mine substructures' frequencies in graphs are used similarly to conventional data mining. The selection of the measures depends on the objective and the constraints of the mining approach. The most popular measure in the graph-based data mining is a "support" parameter whose definition is identical with that of market basket analysis. Given a graph data set D, the support of the subgraph G_s, sup(G_s), is defined as

$$\text{sup}(G_s) = \frac{\text{number of graphs including } G_s \text{ in } D}{\text{total number of graphs in } D}$$

By specifying a "minimum support" value, subgraphs G_s whose support values are above threshold are mined as candidates or components of candidates for maximum frequent subgraphs. The main difference in an apriori implementation is that we need to define the join process of two subgraphs a little differently. Two graphs of size k can be joined if they have a structure of size $(k-1)$ in common. The *size of this structure* could be defined in terms of either nodes or edges. The algorithm may starts by finding all frequent single- and double-link subgraphs. Then, in each iteration, it generates candidate subgraphs by expanding the subgraphs found in the previous iteration by one edge. The algorithm checks how many times the candidate subgraph with the extension occurs within an entire graph or set of graphs. The candidates, whose frequency is below a user-defined level, are pruned. The algorithm returns all subgraphs occurring more frequently than the given threshold. A naïve approach of a subgraph extension from $k-1$ size to size k is computationally very expensive as it is illustrated in Figure 12.12. Therefore, the candidate generation of frequent induced subgraph is done with some constraints. Two frequent graphs are joined only when the following conditions are satisfied to generate a candidate of frequent graph of size $k+1$. Let X_k and Y_k be adjacency matrices of two frequent graphs $G(X_k)$ and $G(Y_k)$ of size k. If both $G(X_k)$ and $G(Y_k)$ have equal elements of the matrices except for the elements of the kth row and the kth column, then they may be joined to generate Z_{k+1} as an adjacency matrix for a candidate graph of size $k+1$:

$$X_k = \begin{pmatrix} X_{k-1} & x_1 \\ x_2^T & 0 \end{pmatrix}, \quad Y_k = \begin{pmatrix} X_{k-1} & y_1 \\ y_2^T & 0 \end{pmatrix}, \quad Z_{k+1} = \begin{pmatrix} X_{k-1} & x_1 & y_1 \\ x_2^T & 0 & z_{k,k+1} \\ y_2^T & z_{k+1,k} & 0 \end{pmatrix}$$

In this matrix representations, X_{k-1} is the common adjacency matrix representing the graph whose size is $k-1$, while x_i and y_i ($i=1, 2$) are $(k-1) \times 1$ column vectors.

6 edges

7 edges

22 new graphs

Figure 12.12. Free extensions in graphs.

These column vectors are representing the differences between two graphs prepared for join operation.

The process suffers from computational intractability when the graph size becomes too large. One problem is subgraph *isomorphism*, with NP complexity, as a core step in a graph matching. Also, all frequent patterns may not be equally relevant in the case of graphs. In particular, patterns that are highly connected, which means dense subgraphs, are much more relevant. This additional analysis requires more computations. One possible application of discovering frequent subgraphs is summarized representation of larger, complex graphs. After extracting common subgraphs, it is possible to simplify large graphs by condensing these subgraphs into new nodes. Illustrative example is given in Figure 12.13 where subgraph of four nodes is replaced in the set of graphs with a single node. The resulting graphs represent summarized representation of the initial graph set.

In recent years, significant attention has focused on studying the structural properties of networks such as the World Wide Web (WWW), online social networks, communication networks, citation networks, and biological networks. Across these large networks, an important characteristic is that they can be characterized by the nature of the underlying graphs and subgraphs, and clustering is often used technique for miming these large networks. The problem of graph clustering arises in two different contexts: a single large graph or large set of smaller graphs. In the first case, we wish to determine dense node clusters in a *single large graph* minimizing the intercluster similarity for a fixed number of clusters. This problem arises in the context of a number of applications such as graph partitioning and the minimum cut problem. The determination of dense regions in the graph is a critical problem from the perspective of a number of different applications in social networks and Web page summarization.

Figure 12.13. Graph summarization through graph compression.

Top-down clustering algorithms are closely related to the concept of *centrality analysis* in graphs where central nodes are typically key members in the network that are well connected to other members of the community. Centrality analysis can also be used in order to determine the central points in information flows. Thus, it is clear that the same kind of structural analysis algorithm can lead to different kinds of insights in graphs. For example, if the criterion for separating graph into subgraphs is a maximum measure of link betweenness, then graph in Figure 12.14a may be transformed into six subgraphs as it is presented in Figure 12.14b. In this case the maximum betweenness of 49 was for the link (7, 8), and elimination of this link defines two clusters on the highest level of hierarchy. The next value of betweenness 33 was found for links: (3,7), (8, 9), (6, 7), and (8, 12). After elimination of these links on the second level of hierarchy, the graph is decomposed into six dense subgraphs of clustered nodes.

The second case of cluster analysis assumes multiple graphs, each of which may possibly be of modest size. These large number of graphs need to be clustered based on their underlying structural behavior. The problem is challenging because of the need to match the structures of the underlying graphs and use these structures for clustering purposes. The main idea is that we wish to cluster graphs as objects, and the distance between graphs is defined based on a structural similarity function such as the edit distance. This clustering approach makes it an ideal technique for applications in areas such as scientific data exploration, information retrieval, computational biology, Web log analysis, forensics analysis, blog analysis, and many others.

Link analysis is an important field that received a lot of attention recently when advances in information technology (IT) enabled mining of extremely large networks. The basic data structure is still a graph; only the emphasis in analysis is on links and their characteristics: labeled or unlabeled and directed or undirected. There is an inherent ambiguity with respect to the term "link" that occurs in many circumstances, but

Figure 12.14. Graph clustering using betweenness measure. (a) Initial graph. (b) Subgraphs, after elimination of links with maximum betweenness.

especially in discussions with people whose background and research interests are in the database community. In the database community, especially the subcommunity that uses the well-known entity-relationship (ER) model, a "link" is a connection between two records in two different tables. This usage of the term "link" in the database community differs from that in the intelligence community and in the artificial intelligence (AI) research community. Their interpretation of a "link" typically refers to some real-world connection between two entities. Probably the most famous example of exploiting link structure in the graph is the use of links to improve information retrieval results. Both the well-known page rank measure and hubs and authority scores are based on the link structure of the Web. Link analysis techniques are used in law enforcement, intelligence analysis, fraud detection, and related domains. It is sometimes described using the metaphor of "connecting the dots" because link diagrams, showing the connections between people, places, events, and things, represent invaluable tools in these domains.

12.2 TEMPORAL DATA MINING

Time is one of the essential natures of data. Many real-life data describes the property or status of some object at a particular time instant. Today time-series data are being generated at an unprecedented speed from almost every application domain, e.g., daily fluctuations of stock market, traces of dynamic processes and scientific experiments, medical and biological experimental observations, various readings obtained from sensor networks, Web logs, computer network traffic, position updates of moving objects in location-based services, etc. Time series or, more generally, temporal sequences appear naturally in a variety of different domains, from engineering to scientific research, finance, and medicine. In engineering matters, they usually arise with either sensor-based monitoring, such as telecommunication control, or log-based systems monitoring. In scientific research, they appear, for example, in spatial missions

or in the genetics domain. In healthcare, temporal sequences are a reality for decades, with data originated by complex data acquisition systems like ECGs or even with simple ones like measuring the patient temperature or treatment effectiveness. For example, a supermarket transaction database records the items purchased by customers at some time point. In this database, every transaction has a time stamp in which the transaction is conducted. In a telecommunication database, every signal is also associated with a time. The price of a stock at the stock market database is not a constant, but changes with time as well.

Temporal databases capture attributes whose values change with time. Temporal data mining is concerned with data mining of these large data sets. Samples related with the temporal information present in this type of databases need to be treated differently from static samples. The accommodation of time into mining techniques provides a window into the temporal arrangement of events and, thus, an ability to suggest cause and effect that are overlooked when the temporal component is ignored or treated as a simple numeric attribute. Moreover, temporal data mining has the ability to mine the behavioral aspects of objects as opposed to simply mining rules that describe their states at a point in time. Temporal data mining is an important extension as it has the capability of mining activity rather than just states and, thus, inferring relationships of contextual and temporal proximity, some of which may also indicate a cause–effect association.

Temporal data mining is concerned with data mining of large sequential data sets. By sequential data, we mean data that is ordered with respect to some index. For example, time series constitute a popular class of sequential data, where records are indexed by time. Other examples of sequential data could be text, gene sequences, protein sequences, Web logs, lists of moves in a chess game, etc. Here, although there is no notion of time as such, the ordering among the records is very important and is central to the data description/modeling. Sequential data include:

1. *Temporal sequences* representing ordered series of nominal symbols from particular alphabet (examples are huge number of relatively short sequences in Web log files or relatively small number of extremely long gene expression sequences). This category includes ordered but not time-stamped collections of samples. The sequence relationships include before, after, meet, and overlap.

2. *Time series* representing time-stamped series of continuous, real-valued elements (examples are relatively small number of long sequences of multiple sensor data or monitoring recordings from digital medical devices). Typically, most of the existing work on time series assume that time is discrete. Formally, a time-series data is defined as a sequence of pairs $T = [(p_1, t_1), (p_2, t_2),\ldots, (p_n, t_n)]$ where $t_1 < t_2 < \cdots <t_n$. Each p_i is a data point in a d-dimensional data space, and each t_i is the time stamp at which p_i occurs. If the sampling rate of time series is constant, one can omit the time stamps and consider series as sequences of d-dimensional data points. Such a sequence is called the raw representation of the time series.

(a)

(b)

(c)

(d)

Figure 12.15. Traditional statistical analyses of time series. (a) Trend. (b) Cycles. (c) Seasonal. (d) Outliers.

Traditional analyses of temporal data require statistical approach because of noise in raw data, missing values, or incorrect recordings. They include (1) long-term trend estimation; (2) cyclic variations, e.g., business cycles; (3) seasonal patterns; and (4) irregular movements representing outliers. Examples are given in Figure 12.15. The discovery of relations in temporal data requires more emphasis in a data-mining process on the following three steps: (1) the representation and modeling of the data sequence in a suitable form, (2) the definition of similarity measures between sequences, and (3) the application of variety of new models and representations to the actual mining problems.

12.2.1 Temporal Data Representation

The representation of temporal data is especially important when dealing with large volumes of data, since direct manipulation of continuous, high-dimensional data in an efficient way is extremely difficult. There are a few ways this problem can be addressed:

Original data or with minimal preprocessing Use data as it is without or with minimal preprocessing. We preserve the characteristics of each data point when a model is built. The main disadvantage of this process is that it is extremely inefficient to build data-mining models with millions of records of temporal data, all of them with different values.

Windowing and piecewise approximations There is well-known psychological evidence that human eye segments smooth curves into piecewise straight lines. Based on this theory, there are a number of algorithms for segmenting a curve that represents a time series. Figure 12.16 shows a simple example where it replaces the original nonlinear function with several piecewise linear functions. As shown in the figure, the initial real-valued elements (time series) are partitioned into several segments. Finding the required number of segments that best represent the original sequence is not trivial. An easy approach is to predefine the number of segments. A more realistic approach may be to define them when a change point is detected in the original sequence. Another technique based on the same idea segments a sequence by iteratively merging two similar segments. Segments to be merged are selected based on the squared error minimization criteria. Even though these methods have the advantage of ability to reduce the impact of noise in the original sequence, when it comes to real-world applications (for example, sequence matching), differences in amplitudes (scaling) and time axis distortions are not addressed easily.

To overcome these drawbacks, *Piecewise Aggregate Approximation* (PAA) technique was introduced. It approximates the original data by segmenting the sequences into same length sections and recording the mean value of these sections. A time series **C** of length n is represented as $C = \{c_1, c_2, \ldots, c_n\}$. The task is to represent C as C' in a w-dimensional space ($w < n$) by mean values of c_is in w equal-sized segments. The ith element of C' is calculated as a mean of all value in the segment:

$$C'_i = \sum_{j=w \times (i-1)+1}^{w \times i} c^i_j, \quad 1 \leq i \leq \text{the number of segments}$$

For example, if the original sequence is $C = \{-2, -4, -3, -1, 0, 1, 2, 1, 1, 0\}$, where $n = |C| = 10$, and we decide to represent C in two sections of the same length, then

$$C' = \{\text{mean}(-2, -4, -3, -1, 0), \text{mean}(1, 2, 1, 1, 0)\}$$
$$C' = \{-2, 1\}$$

Usually PAA is visualized as a linear combination of box basis functions as it is illustrated in Figure 12.16b where a continuous function is replaced with 10 discrete averages for each interval.

A modified PAA algorithm, which is called *Symbolic Aggregate approXimation (SAX)*, is proposed with the assumption that the original normalized time series, C, have a Gaussian distribution of PAA values. SAX defines "breakpoints" in the Gaussian curve that will produce equal-sized areas below the curve. Formally, breakpoints are a sorted list of numbers $B = \beta_1, \beta_2, \beta_3, \ldots, \beta_{a-1}$ such that the areas under the Gaussian curve from β_i to β_{i+1} are equal $1/a$ and they are constant. α is a parameter of the methodology representing the number of intervals. These breakpoints can be

(a)

(b)

Figure 12.16. Simplified representation of temporal data. (a) Piecewise Linear Approximation. (b) Piecewise Aggregated Approximation.

determined in a statistical table. For example, Figure 12.17 gives the breakpoints for α values from 3 to 10.

Once the breakpoints are defined as well as the corresponding coding symbols for each interval, the sequence is discretized as follows:

1. Obtain the PAA averages of the time series.
2. All PAA averages in the given interval (β_i, β_{i+1}) are coded with a specific symbol for this interval. For example, if $\alpha = 3$, all PAA averages less than

(a)

$$\text{Gaussian function} = \exp\left(\frac{-(x-b)^2}{2 \times c^2}\right)$$

$\beta_1 = -0.43$ $\beta_2 = 0.43$

(b)

β_1 \ a	3	4	5	6	7	8	9	10
β_1	−0.43	−0.67	−0.84	−0.97	−1.07	−1.15	−1.22	−1.28
β_2	0.43	0	−0.25	−0.43	−0.57	−0.67	−0.76	−0.84
β_3		0.67	0.25	0	−0.18	−0.32	−0.43	−0.52
β_4			0.84	0.43	0.18	0	−0.14	−0.25
β_5				0.97	0.57	0.32	0.14	0
β_6					1.07	0.67	0.43	0.25
β_7						1.15	0.76	0.52
β_8							1.22	0.84
β_9								1.28

Figure 12.17. SAX lookup table. (a) Breakpoints in the Gaussian curve ($b = 0$, $c_2 = 0.2$) when $\alpha = 3$. (b) SAX lookup table.

the smallest breakpoint (−0.43) are mapped to "*a*." All PAA coefficients less than the second breakpoint but greater than first breakpoint (−0.43, 0.43) are mapped to "*b*," and all average values larger than the second breakpoint (0.43) are mapped to "*c*." This process is illustrated in Figure 12.18.

It is assumed that the symbols "*a*," "*b*," and "*c*" are approximately equiprobable symbols in the representation of a time series. The original sequence is then represented as a concatenation of these symbols, which is known as a "word." For example, the mapping from PAA (C') to a word C'' is represented as $C'' = (bcccccbaaaaabbccccbb)C' = (aabcbbbc)$. The main advantage of the SAX method is that 100 different discrete numerical values in an initial

Figure 12.18. Transformation of PAA to SAX.

discrete time series C are first reduced to 20 different (average) values using PAA and then they are transformed into only three different categorical values using SAX:

$$C = \{c_1, c_2, \ldots, c_{100}\} \rightarrow C' = \{c'_1, c'_2, \ldots, c'_{20}\} \rightarrow C'' = f\{a, b, c\}$$

The proposed approach is intuitive and simple, yet powerful methodology in a simplified representation of a large number of different values in time series. The method is fast to compute and support different distance measures. Therefore, it is applicable as a data-reduction step for different data-mining techniques.

Transformation based representations The main idea behind transformation-based techniques is to map the original data into a point of a more manageable domain. One of the widely used methods is the *discrete Fourier transformation* (DFT). It transforms a sequence in the time domain to a point in the frequency domain. This is done by selecting the top-K frequencies and representing each sequence as a point in the K-dimensional space. One important property that is worth noting is that Fourier coefficients do not change under the shift operation. One problem with DFT is that is misses the important feature of time localization. To avoid this problem, *piecewise Fourier transform* was proposed, but the size of the *pieces* introduces new problems. Large *pieces* reduce the power of multi-resolution, while modeling of low frequencies with small *pieces* does not always give expected representations. The *discrete wavelet transformation* (DWT) has been introduced to overcome the difficulties in DFT. DWT transformation technique, analogously to fast Fourier transformation, turns a discrete vector of function values with the length N into a vector of N wavelet coefficients. Wavelet transformation is a linear operation and it is usually implemented as a recursion. The advantage of using DWT is its ability of multi-resolution representation of signals. It has

the time–frequency localization property. Therefore signals represented by wavelet transformations bear more information than that of DFT.

In some applications we need to retrieve objects from a database of certain shapes. Trends can often be reflected by specifying shapes of interest such as steep peaks or upward and downward changes. For example, in a stock market database, we may want to retrieve stocks whose closing price contains a *head and shoulder* pattern, and we should be able to represent and recognize these shapes. Pattern discovery can be driven by a template-based mining language in which the analyst specifies the shapes that should be looked for. *Shape definition language* (SDL) was proposed to translate the initial sequence with real-valued elements occurring in historical data into a sequence of symbols from a given alphabet. SDL is capable of describing variety of queries about the shapes found in the database. It allows the user to create their own language with complex patterns in terms of primitives. More interestingly, it performs well on approximate matching where the user cares only about the overall shape of the sequence but not on specific details. The first step in the representation process is defining the alphabet of symbols and then translating the initial sequence to a sequence of symbols. The translation is done by considering sample-to-sample transitions and then assigning a symbol of the described alphabet to each transition.

A significantly different approach is to convert a sequence into discrete representation by using *clustering*. A sliding window of width w is used to generate subsequences from the original sequence. These subsequences are then clustered, considering the pattern similarity between subsequences, using a suitable clustering method, for example, k-nearest neighbor method. A different symbol is then assigned to each cluster. The discrete version of the time series is obtained by using cluster identities corresponding to the subsequence. For example, the original time sequence is defined with integer values given in time (1, 2, 3, 2, 3, 4, 3, 4, 3, 4, 5, 4, 5) as it is represented in Figure 12.19a. The window width is defined by three consecutive values, and samples of primitives are collected through the time series. After simplified clustering, the final set of three "frequent" primitive shapes, representing cluster centroids, is given in Figure 12.19b. Assigning symbolic representation for these shapes, namely, a_1, a_2, and a_3, the final symbolic representation of the series will be (a_3, a_2, a_1, a_1, a_3, a_2).

12.2.2   Similarity Measures Between Sequences

The individual elements of the sequences may be vectors of real numbers (e.g. in applications involving speech or audio signals), or they may be symbolic data (e.g. in applications involving gene sequences). After representing each sequence in a suitable form, it is necessary to define a similarity measure between sequences in order to determine if they match. Given two sequences T_1 and T_2, we need to define an appropriate similarity function Sim, which calculates closeness of the two sequences, denoted by $\text{Sim}(T_1, T_2)$. Usually the similarity measure is expressed in terms of inverse distance measure, and for various types of sequences and applications, we

Figure 12.19. Clustering approach in primitive shapes discovery. (a) Time series. (b) Primitive shapes after clustering.

have numerous distance measures. An important issue in measuring similarity between two sequences is the ability to deal with outlying points, noise in the data, amplitude differences causing *scaling problems*, and the existence of gaps and other time distortion problems. The most straightforward distance measure for time series is the Euclidean distance and its variants, based on the common Lp norms. It is used in time-domain continuous representations by viewing each subsequence with n discrete values as a point in R_n. Besides being relatively straightforward and intuitive, Euclidean distance and its variants have several other advantages. The complexity of evaluating these measures is linear, and they are easy to implement and indexable with any access method and, in addition, are parameter-free. Furthermore, the Euclidean distance is surprisingly competitive with other more complex approaches, especially if the size of the training set/database is relatively large. However, since the mapping between the points of two time series is fixed, these distance measures are very sensitive to noise and misalignments in time and are unable to handle local time shifting, i.e., similar segments that are out of phase.

   When a sequence is represented as a sequence of discrete symbols of an alphabet, the similarity between two sequences is, most of the times, achieved by comparing each element of one sequence with the correspondent one in the other sequence. The best known such distance is the longest common subsequence (LCS) similarity, utilizing the search for the LCS in both sequences we are comparing, and normalized

with the length of the longer sequence. For example, if two sequences X and Y are given as $X = \{10, 5, 6, 9, 22, 15, 4, 2\}$ and $Y = \{6, 5, 10, 22, 15, 4, 2, 6, 8\}$, then the longest common sequence is

$$\text{LCS}(X,Y) = \{22, 15, 4, 2\}$$

and normalized similarity measure is

$$\text{LCS-Similarity}\,(X,Y) = \frac{\text{LCS}(X,Y)}{\max\{|X|,|Y|\}} = \frac{4}{9}$$

In order to deal with noise, scaling, approximate values, and translation problems, a simple improvement consists in determining the pairs of sequence portions that agree in both sequences after some linear transformation is applied. It consists in determining if there is a linear function f, such that one sequence can be approximately mapped into the other. In most applications involving determination of similarity between pairs of sequences, the sequences would be of different lengths. In such cases, it is not possible to blindly accumulate distances between corresponding elements of the sequences. This brings us to the second aspect of sequence matching, namely, sequence alignment. Essentially we need to properly insert "gaps" in the two sequences or decide which should be corresponding elements in the two sequences. There are many situations in which such symbolic sequence matching problems find applications. For example, many biological sequences such as genes, proteins, etc. can be regarded as sequences over a finite alphabet. When two such sequences are similar, it is expected that the corresponding biological entities have similar functions because of related biochemical mechanisms. The approach includes a similarity measure for sequences based on the concept of the edit distance for strings of discrete symbols. This distance reflects the amount of work needed to transform a sequence to another and is able to deal with different sequence length and gap existence. Typical edit operations are insert, delete, and replace, and they may be included in the measure with the same or with different weights (costs) in the transformation process. The distance between two strings is defined as the least sum of edit operation costs that needs to be performed to transform one string into another. For example, if two sequences are given: $X = \{a, b, c, b, d, a, b, c\}$ and $Y = \{b, b, b, d, b\}$, the following operations are applied to transform X into Y: delete(a), replace(c,b), delete(a), delete (c). The total number of operations in this case is 4, and it represents non-normalized distance measure between two sequences.

12.2.3   Temporal Data Modeling

A *model* is a global, high-level, and often abstract representation for the data. Typically, models are specified by a collection of model parameters that can be estimated from a given data set. It is possible to classify models as predictive or descriptive depending on the task they are performing. In contrast to the (global) model structure,

a *temporal pattern* is a local model that makes a specific statement about a few data samples in time. Spikes, for example, are patterns in a real-valued time series that may be of interest. Similarly, in symbolic sequences, regular expressions represent well-defined patterns. In bioinformatics, genes are known to appear as local patterns interspersed between chunks of noncoding DNA. Matching and discovery of such patterns are very useful in many applications not only in bioinformatics. Due to their readily interpretable structure, patterns play a particularly dominant role in data mining. There have been many techniques used to model global or local temporal events. We will introduce only some of most popular modeling techniques.

Finite state machine (FSM) has a set of states and set of transitions. A state may have transitions to other states that are caused by fulfilling some conditions within the state. An FSM must have an initial state, usually shown drawn with an arrow, and it is a state that provides a starting point of the model. Inputs to the states, in our case representing symbols in a sequence, act as triggers for the transition from one state to another state. An accept state, which is also known as final state, is usually represented by a double circle in a graph representation. The machine reaches the final state when it has performed the procedure successfully or in our case recognized a sequence pattern. An FSM can be represented using a state transition table or state transition diagram. Figure 12.20a and b shows both of these representations for a modeling recognition of binary number with even number of ones. FSM does not work very well when the transitions are not precise and does not scale well when the set of symbols for sequence representation is large.

Markov model (MM) extends the basic idea behind FSM. Both FSM and MM are directed graphs. As with FSR, MM always has a current state. Start and end nodes are drawn for illustrative purposes and need not be present. Unlike in FSM transitions are not associated with specific input values. Arcs carry a probability value for transition from one state to the other. For example, the probability that transition from state "Start" to "S1" is 0.4, and the probability staying in the "Start" state is 0.6. The sum of the probability values coming out of each node should be 1. MM shows only transitions with probability greater than 0. If a transition is not shown, it is assumed to have a probability of 0. The probabilities are combined to determine the final probability of the pattern produced by the MM. For example, with the MM shown in Figure 12.21, the probability that the MM takes the horizontal path from starting node to $S2$ is $0.4 \times 0.7 = 0.28$.

(a)

Condition	Current state	
	S1	S2
Input 0	S1	S2
Input 1	S2	S1

Figure 12.20. Finite state machine. (a) State transition table. (b) State transition diagram.

Figure 12.21. A simple Markov model.

Figure 12.22. Markov model vs. hidden Markov model. (a) 1-coin model. (b) 2-coin model.

MM is derived based on the memoryless assumption. It states that given the current state of the system, the future evolution of the system is independent of its history. MM have been used widely in speech recognition and natural language processing.

Hidden Markov model (HMM) is an extension to MM. Similar to MM, HMM consists of set of states and transition probabilities. In a regular MM, the states are visible to the observer, and the state transition probabilities are the only parameters. In HMM, each state is associated with a state probability distribution. For example, assume that we were given a sequence of events in a coin toss: O = (HTTHTHH), where H = Head and T = Tail. But additional information is necessary. What is not given is the sequence generated with one or two coins. According to the above definitions, Figure 12.22 shows two possible models. Figure 12.22a assumes that only one coin was tossed. We can model this system as an MM with two-state model, where Head and Tail are these two states with the same initial probabilities. The probability of the sequence O is $P(O) = 0.5 \times 0.7 \times 0.3 \times 0.7 \times 0.3 \times 0.7 \times 0.7 = 0.0108$.

Another possibility to explain the observed sequence is shown in Figure 12.22b. There are again two states in this model, and each state corresponds to a separate biased coin being tossed. Each state has its own probability distribution of Heads and Tails, and therefore the model is represented as an HMM. Obviously, in this model, we have several "paths" to determine the probability of the sequence. In other words, we can start with tossing one or other coin and continue with this selection. In all these cases composite probability will be different. In this situation we may search for the maximum probability of the sequence O in the HMM. HMM may be formalized as a directed graph with V vertices and A arcs. Set $V = \{v_1, v_2,\ldots,v_n\}$ represents states, and matrix $A = \{a_{ij}\}$ represents transition probability distribution, where a_{ij} is the transitional probability from state i to state j. Given a set of possible observations

$O = \{o_1, o_2,...,o_m\}$ for each state v_i, the probability of seeing each observation in the sequence is given by $B_i = \{o_{i1}, o_{i2},...,o_{im}\}$. The initial state distribution is represented as σ, which determines the starting state at time $t = 0$.

12.2.4 Mining Sequences

Temporal data-mining tasks include prediction, classification, clustering, search and retrieval, and pattern discovery. The first four have been investigated extensively in traditional time-series analysis, pattern recognition, and information retrieval. We will concentrate in this text on illustrative examples of algorithms for pattern discovery in large databases, which are of more recent origin and showing wide applicability. The problem of pattern discovery is to find and evaluate all "interesting" patterns in the data. There are many ways of defining what constitutes a pattern in the data, and we shall discuss some generic approaches. There is no universal notion for interestingness of a pattern either. However, one concept that is found very useful in data mining is that of frequent patterns. A frequent pattern is one that occurs many times in the data. Much of data-mining literature is concerned with formulating useful pattern structures and developing efficient algorithms for discovering all patterns that occur frequently in the data.

A pattern is a local structure in the database. In the sequential patterns framework, we are given a collection of sequences, and the task is to discover sequences of items called sequential patterns that occur in sufficiently many of those sequences. In the frequent episodes analysis, the data set may be given in a single long sequence or in a large set of shorter sequences. An *event sequence* is denoted by $\{(E_1, t_1),$ $(E_2, t_2), ... (E_n, t_n)\}$, where E_i takes values from a finite set of event types E and t_i is an integer denoting the time stamp of the ith event. The sequence is ordered with respect to the time stamps so that $t_i \leq t_{i+1}$ for all $i = 1, 2,...,n$. The following is an example event sequence S with 10 events in it:

$$S = \{(A,2),(B,3),(A,7),(C,8),(B,9),(D,11),(C,12),(A,13),(B,14),(C,15)\}$$

An episode is a partially ordered set of events. When the order among the events of an episode is total, it is called a *serial* episode, and when there is no order at all, the episode is called a *parallel* episode. For example, $(A \rightarrow B \rightarrow C)$ is a 3-node serial episode. The arrows in our notation serve to emphasize the total order. In contrast, parallel episodes are somewhat similar to itemsets, and so, we can denote a 3-node parallel episode with event types A, B, and C as (ABC).

An episode is said to *occur* in an event sequence if there exist events in the sequence occurring with exactly the same order as that prescribed in the episode. For example, in the example, the events $(A, 2)$, $(B, 3)$, and $(C, 8)$ constitute an occurrence of the serial episode $(A \rightarrow B \rightarrow C)$, while the events *(A, 7)*, *(B, 3)*, and *(C, 8)* do not, because for this serial episode to occur, A must occur before B and C. Both these sets of events, however, are valid occurrences of the parallel episode *(ABC)*, since there are no restrictions with regard to the order in which the events must occur

for parallel episodes. Let α and β be two episodes. β is said to be a *sub-episode* of α if all the event types in β appear in α as well and if the partial order among the event types of β is the same as that for the corresponding event types in α. For example, $(A \rightarrow C)$ is a 2-node sub-episode of the serial episode $(A \rightarrow B \rightarrow C)$, while $(B \rightarrow A)$ is not. In the case of parallel episodes, this order constraint is not required.

The sequential pattern mining framework may extend the frequent itemset idea described in the chapter on association rules with temporal order. The database D *of itemsets* is considering no longer just some unordered collection of transactions. Now, each transaction in D carries a time stamp as well as a customer ID. Each transaction as earlier is simply a collection of items. The transactions associated with a single customer can be regarded as a sequence of itemsets ordered by time, and D would have one such transaction sequence corresponding to each customer. Consider an example database with five customers whose corresponding transaction sequences are as follows:

Customer ID	Transaction Sequence
1	({A,B}{A,C,D}{B,E})
2	({D,G} {A,B,E,H})
3	({A}{B,D}{A,B,E,F}{G,H})
4	({A}{F})
5	({A,D} {B,E,G,H} {F})

Each customer's transaction sequence is enclosed in angular braces, while the items bought in a single transaction are enclosed in round braces. For example, customer 3 made 4 visits to the supermarket. In his/her first visit, he/she bought only item A; in the second visit, items are B and D; and so on.

The temporal patterns of interest are sequences of itemsets. A sequence S of itemsets is denoted by $\{s_1 s_2 \cdot s_n\}$, where s_j is an itemset. Since S has n itemsets, it is called an n-sequence. A sequence $A = \{a_1 a_2 \cdot a_n\}$ is said to be *contained in* another sequence $B = \{b_1 b_2 \cdot b_m\}$ if there exist integers $i_1 < i_2 < \cdot < i_n$ such that $a_1 \subseteq b_{i1}, a_2 \subseteq b_{i2}, \ldots, a_n \subseteq b_{in}$. That is, an n-sequence A is contained in a sequence B if there exists an n-length subsequence in b, in which each itemset contains the corresponding itemsets of a. For example, the sequence $\{(A)(BC)\}$ is contained in $\{(AB) (F) (BCE) (DE)\}$ but not in $\{(BC)(AB)(C)(DEF)\}$. Further, a sequence is said to be *maximal* in a set of sequences if it is not contained in any other sequence. In the set of example customer transaction sequences listed above, all are maximal (with respect to the given set of sequences) except the sequence of customer 4, which is contained in transaction sequences of customers 3 and 5.

The Apriori algorithm described earlier can be used to find frequent sequences, except that there is a small difference in the definition of support. Earlier, the support of an itemset was defined as the fraction of *all* transactions that contained the itemset. Now, the *support* for any arbitrary sequence A is the fraction of customer transaction sequences in the database D that contain A. For our example database, the sequence

$\{(D)(GH)\}$ has a support of 0.4, since it is contained in two out of the five transaction sequences (namely, that of customer 3 and customer 5). The user specifies a minimum support threshold. Any sequence of itemsets with support greater than or equal to the threshold value is called a *large* sequence. If a sequence A is large and maximal, then it is regarded as a *sequential pattern*. The process of frequent episode discovery is an Apriori-style iterative algorithm that starts with discovering frequent 1-element sequences. These are then combined to form candidate 2-element sequences, and then by counting their frequencies, 2-element frequent sequences are obtained. This process is continued till frequent sequences of all lengths are found. The task of a sequence mining is to systematically discover all sequential patterns in database D.

Counting frequencies of parallel itemsets is straightforward and described in traditional algorithms for frequent itemset detection. Counting serial itemsets, on the other hand, requires more computational resources. For example, unlike for parallel itemsets, we need finite state automata to recognize serial episodes. More specifically, an appropriate l-state automaton can be used to recognize occurrences of an l-node serial sequence. For example, for the sequence $(A \rightarrow B \rightarrow A \rightarrow A)$, there would be a 5-state automaton given in Figure 12.23. It transits from its first state on seeing an event of type A and then waits for an event of type B to transit to its next state and so on. We need such automata for each episode whose frequency is being counted.

While we described the framework using an example of mining a database of customer transaction sequences for temporal buying patterns, this concept of sequential patterns is quite general and can be used in many other situations as well. Indeed, the problem of motif discovery in a database of protein sequences can also be easily addressed in this framework. Another example is Web navigation mining. Here the database contains a sequence of Web sites that a user navigates through in each browsing session. Sequential pattern mining can be used to discover sequences of Web sites that are frequently visited. Temporal associations are particularly appropriate as candidates for causal rules' analysis in temporally related medical data, such as in the histories of patients' medical visits. Patients are associated with both static properties, such as gender, and temporal properties, such as age, symptoms, or current medical treatments. Adapting this method to deal with temporal information leads to some different approaches. A possible extension is a new meaning for a typical association rule: $X\Rightarrow Y$. It states now that if X occurs, then Y will occur within time T. Stating a rule in this new form allows for controlling the impact of the occurrence of one event to the other event occurrence, within a specific time interval. In the case of the sequential pattern framework, some generalizations are proposed to incorporate minimum and maximum time gap constraints between successive elements of a sequential pattern.

Mining continuous data streams is a new research topic related to temporal data mining that has recently received significant attention. The term "data stream" pertains to data arriving over time in a nearly continuous fashion. It is often fast-changing stream with huge number multidimensional data. Data are collected close to their source, such as sensors data, so they are usually with a low level of abstraction. In streaming data-mining applications, the data is often available for mining only once,

Figure 12.23. FSA for the sequence A → B → A→ A.

Figure 12.24. Multidimensional streams.

as it flows by. That cause several challenging problems including how to aggregate the data, how to obtain scalability of traditional analyses in massive, heterogeneous, non-stationary data environment, and how to incorporate incremental learning into a data-mining process. Linear, single-scan algorithms are still rare in commercial data-mining tools, but also still challenged in a research community. Many applications, such as network monitoring, telecommunication applications, stock market analysis, bio-surveillance systems, and distribute sensors depend critically on the efficient processing and analysis of data streams. For example, a frequent itemset mining algorithm over data stream is developed. It is based on an incremental algorithm to maintain the FP stream, which is a tree data structure to represent the frequent itemsets and their dynamics in time (Figure 12.24).

Ubiquitous data mining (UDM) is an additional new field that defines a process of performing analysis of data on mobile, embedded, and ubiquitous devices. It represents the next generation of data-mining systems that will support the intelligent and time-critical information needs of mobile users and will facilitate "anytime, anywhere" data mining. It is the next natural step in the world of ubiquitous computing. The underlying focus of UDM systems is to perform computationally intensive mining techniques in mobile environments that are constrained by limited computational resources and varying network characteristics. Additional technical challenges are how to minimize energy consumption of the mobile device during the data-mining process, how to present results on relatively small screens, and how to transfer data-mining results over a wireless network with a limited bandwidth.

12.3 SPATIAL DATA MINING

Spatial data mining (SDM) is the process of discovering interesting and previously unknown but potentially useful information from large spatial data sets. Spatial data carries topological and/or distance information, and it is often organized in databases by spatial indexing structures and accessed by spatial access methods. The applications covered by SDM include geomarketing, environmental studies, risk analysis, remote sensing, geographical information systems (GIS), computer cartography, environmental planning, and so on. For example, in geomarketing, a store can establish its trade area, i.e. the spatial extent of its customers, and then analyze the profile of those customers on the basis of both their properties and the area where they live. Simple illustrations of SDM results are given in Figure 12.25, where (a) shows that a fire is often located close to a dry tree and a bird is often seen in the neighborhood of a house, while (b) emphasizes a significant trend that can be observed for the city of Munich where the average rent decreases quite regularly when moving away from the city. One of the main reasons for developing large number of SDM applications is enormous amount of special data that is collected recently at relatively low price. High spatial and spectral resolution remote sensing systems and other environmental monitoring devices gather vast amounts of geo-referenced digital imagery, video, and sound. The complexity of spatial data and intrinsic spatial relationships limits the usefulness of conventional data-mining techniques for extracting spatial patterns.

One of the fundamental assumptions of data-mining analysis is that the data samples are independently generated. However, in the analysis of spatial data, the assumption about the independence of samples is generally false. In fact, spatial data tends to be highly self-correlated. Extracting interesting and useful patterns from spatial data sets is more difficult than extracting corresponding patterns from traditional numeric and categorical data due to the complexity of spatial data types, spatial relationships, and spatial autocorrelation. The spatial attributes of a spatial object most often include information related to spatial locations, e.g., longitude, latitude, and elevation, as well as shape. Relationships among nonspatial objects are explicit in data inputs, e.g. arithmetic relation, ordering, is instance of, subclass of, and membership of. In contrast, relationships among spatial objects are often implicit, such as overlap, intersect, close,

Figure 12.25. Illustrative examples of spatial-data-mining results. (a) Example of co-location spatial data mining (Shekhar and Chawla, 2003). (b) Average rent for the communities of Bavaria (Ester et al., 1997).

and behind. Proximity can be defined in highly general terms, including distance, direction, and/or topology. Also, spatial heterogeneity or the non-stationarity of the observed variables with respect to location is often evident since many space processes are local. Omitting the fact that nearby items tend to be more similar than items situated more apart causes inconsistent results in the spatial data analysis. In summary, specific features of spatial data that preclude the use of general purpose data-mining algorithms are (1) rich data types (e.g., extended spatial objects), (2) implicit spatial relationships among the variables, (3) observations that are not independent, and (4) spatial autocorrelation among the features (Figure 12.26).

One possible way to deal with implicit spatial relationships is to materialize the relationships into traditional data input columns and then apply classical data-mining techniques. However, this approach can result in loss of information. Another way to capture implicit spatial relationships is to develop models or techniques to incorporate spatial information into the SDM process. A concept within statistics devoted to the analysis of spatial relations is called spatial autocorrelation. Knowledge-discovery techniques that ignore spatial autocorrelation typically perform poorly in the presence of spatial data.

The spatial relationship among locations in a spatial framework is often modeled via a contiguity matrix. A simple contiguity matrix may represent a neighborhood relationship defined using adjacency. Figure 12.27a shows a gridded spatial framework with four locations, A, B, C, and D. A binary matrix representation of a four-neighborhood relationship is shown in Figure 12.27b. The row-normalized representation of this matrix is called a contiguity matrix, as shown in Figure 12.27c. The essential idea is to specify the pairs of locations that influence each other along with the relative intensity of interaction.

	Traditional Data Mining	Spatial Data Mining
Input	Simple types Explicit relationship	Complex types Implicit relationships
Statistical foundation	Independence of samples	Spatial autocorrelation
Output	Set-based interest measures, e.g. classification accuracy	Spatial interest measures, e.g. spatial accuracy

Figure 12.26. Main differences between traditional data mining and spatial data mining.

(a)   (b)   (c)

$$\begin{array}{c c} & \begin{array}{c c c c} A & B & C & D \end{array} \\ \begin{array}{c} A \\ B \\ C \\ D \end{array} & \left[\begin{array}{c c c c} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{array} \right] \end{array}$$

$$\begin{array}{c c} & \begin{array}{c c c c} A & B & C & D \end{array} \\ \begin{array}{c} A \\ B \\ C \\ D \end{array} & \left[\begin{array}{c c c c} 0 & 0.5 & 0.5 & 0 \\ 0.5 & 0 & 0 & 0.5 \\ 0.5 & 0 & 0 & 0.5 \\ 0 & 0.5 & 0.5 & 0 \end{array} \right] \end{array}$$

Figure 12.27. Spatial framework and its four-neighborhood contiguity matrix. (a) Spatial framework. (b) Binary matrix representation. (c) Normalized contiguity matrix.

SDM consists of extracting knowledge, spatial relationships, and any other properties that are not explicitly stored in the database. SDM is used to find implicit regularities and relations between spatial data and nonspatial data. In effect, a spatial database constitutes a spatial continuum in which properties concerning a particular place are generally linked and explained in terms of the properties of its neighborhood. In this section, we introduce as illustrations of SDM characteristics two important and often used techniques: (1) spatial autoregressive modeling (SAR) and (2) spatial outliers' detection using variogram-cloud technique.

1. *The spatial autoregressive model* is a classification technique that decomposes a classifier into two parts, spatial autoregression and logistic transformation. Spatial dependencies are modeled using the framework of logistic regression analysis. If the spatially dependent values y_i are related to each other, then the traditional regression equation can be modified as

$$y = \rho W y + X \beta + \varepsilon$$

where W is the neighborhood relationship contiguity matrix and ρ is a parameter that reflects the strength of the spatial dependencies between the elements of the dependent variable. After the correction term $\rho W y$ is introduced, the components of the residual error vector ε are then assumed to be generated from independent and identical standard normal distributions. As in the case of classical regression, the proposed equation has to be transformed via the

logistic function for binary dependent variables, and we refer to this equation as the Spatial AutoRegressive model (SAR). Notice that when $\rho = 0$, this equation collapses to the classical regression model. If the spatial autocorrelation coefficient is statistically significant, then SAR will quantify the presence of spatial autocorrelation in the classification model. It will indicate the extent to which variations in the dependent variable (y) are influenced by the average of neighboring observation values.

2. *A spatial outlier* is a spatially referenced object whose nonspatial attribute values differ significantly from those of other spatially referenced objects in its spatial neighborhood. This kind of outliers shows a local instability in values of nonspatial attributes. They represent spatially referenced objects whose nonspatial attributes are extreme relative to its neighbors, even though the attributes may not be significantly different from the entire population. For example, a new house in an old neighborhood of a growing metropolitan area is a spatial outlier based on the nonspatial attribute house age.

A variogram-cloud technique displays data points related by neighborhood relationships. For each pair of samples, the square root of the absolute difference between attribute values at the locations versus the Euclidean distance between the locations is plotted. In data sets exhibiting strong spatial dependence, the variance in the attribute differences will increase with increasing distance between locations. Locations that are near to one another, but with large attribute differences, might indicate a spatial outlier, even though the values at both locations may appear to be reasonable when examining the data set nonspatially. For example, the spatial data set is represented with six five-dimensional samples given in Figure 12.28a. Traditional nonspatial analysis will not discover any outliers, especially because the number of samples is relatively small. However, after applying a variogram-cloud technique, assuming that first two attributes are *X–Y* spatial coordinates and the other three are characteristics of samples, the conclusion could be significantly changed. Figure 12.29 shows the variogram cloud for this data set. This plot has some pairs of points that are out of main dense region of common distances.

(a)

Sample	X-C	Y-C	AT-1	AT-2	AT-3
S1	1	2	8	4	1
S2	3	4	2	6	4
S3	2	1	4	2	4
S4	5	3	3	2	5
S5	2	2	7	4	2
S6	1	1	6	5	1

(b)

Samples Compared	Spatial Distance	Distance of Samples
S3 – S6	1.0	4.69
S3 – S5	1.0	4.12
S1 – S3	1.41	5.39

Figure 12.28. An example of variogram-cloud graph. (a) Spatial data set. (b) Critical sample's relations in a variogram cloud.

Figure 12.29. A variogram-cloud technique discovers an outlier.

Computation of spatial distances and distances of samples, as a part of a variogram technique, shows that there is a sample spatially relatively close to a group of other samples (small space distances) but with very high distances in other nonspatial attributes. This is the sample S3, which is spatially close to samples S1, S5, and S6. Coordinates of these samples and corresponding distances are given in Figure 12.28b selecting S3 as a candidate for an outlier. Visualization of these and other relations between samples through a variogram shows the same results.

12.4   DISTRIBUTED DATA MINING

The emergence of tremendous data sets creates a growing need for analyzing them across geographical lines using distributed systems. These developments have created unprecedented opportunities for large-scale data-driven knowledge discovery, as well as the potential for fundamental gains in scientific and business understanding. Implementations of data-mining techniques on high-performance distributed computing platforms are moving away from centralized computing models for both technical and organizational reasons. In some cases, centralization is hard because it requires these multi-terabyte data sets to be transmitted over very long distances. In others, centralization violates privacy legislation, exposes business secrets, or poses other social challenges. Common examples of such challenges arise in medicine where relevant data might be spread among multiple parties, in commercial organizations such as drug companies or hospitals, government bodies such as the US Food and Drug Administration, and nongovernment organizations such as charities and public-health

organizations. Each organization is bound by regulatory restrictions, such as privacy legislation, or corporate requirements on proprietary information that could give competitors a commercial advantage. Consequently, a need exists for developing algorithms, tools, services, and infrastructure that let us mine data distributed across organizations while preserving privacy.

This shift toward intrinsically distributed, complex environments has prompted a range of new data-mining challenges. The added dimension of distributed data significantly increases the data-mining process's complexity. Advances in computing and communication over wired and wireless networks have resulted in many pervasive distributed computing environments. Many of these environments deal with different distributed sources of voluminous data, multiple compute nodes, and distributed user community. Analyzing and monitoring these distributed data sources require a new data-mining technology designed for distributed applications. The field of distributed data mining (DDM) deals with these problems—mining distributed data by paying careful attention to the distributed resources. In addition to data being distributed, the advent of the Internet has led to increasingly complex data, including natural language text, images, time series, sensor data, multi-relational and object data types, and so on. To further complicate matters, systems with distributed streaming data need incremental or online mining tools that require complete process whenever a change is made to the underlying data. Data-mining techniques involving in such complex environment must encounter great dynamics due to changes in the system, and it can affect the overall performance of the system. Providing support for all these features in DDM systems requires novel solutions.

The Web architecture, with layered protocols and services, provides a sound framework for supporting DDM. New framework embraces the growing trend of merging computation with communication. DDM accepts the fact that data may be inherently distributed among different loosely coupled sites often with heterogeneous data and connected by a network. It offers techniques to discover new knowledge through distributed data analysis and modeling using minimal communication of data. Also, interactions in a distributed system need to be implemented in a reliable, stable, and scalable way. Ultimately, systems must be able to hide this technological complexity from users.

Today, the goods that are able to be transacted through e-services are not restricted only for real entities such as electronics, furniture, or plane tickets. The Internet and WWW evolve to include also resources such as software, computation abilities, or useful data sets. These new resources are potentially able to be sold or rented to clients as services for Internet users. Data mining is emerging as intuitively suitable for being delivered as an e-service because the approach reduces the high cost of setting up and maintaining infrastructure of supporting technologies. To efficiently and effectively deliver data mining as a service in the WWW, Web service technologies are introduced to provide layers of abstractions and standards above existing software systems. These layers are capable of bridging any operating system, hardware platform, or programming language, just as the Web is doing. The natural extension for these services is grid computing. The grid is a distributed computing

infrastructure that enables coordinated resource sharing within dynamic organizations consisting of individuals, institutions, and resources. The main aim of grid computing is to give organizations and application developers the ability to create distributed computing environments that can utilize computing resources on demand. Grid computing can leverage the computing power of a large numbers of server computers, desktop PCs, clusters, and other kind of hardware. Therefore, it can help increase efficiencies and reduce the cost of computing networks by decreasing data processing time and optimizing resources and distributing workloads. Grid allows users to achieve much faster results on large operations and at lower costs. Recent development and applications show that the grid technology represents a critical infrastructure for high-performance DDM and knowledge discovery. This technology is particularly suitable for applications that typically deal with very large amount of distributed data such as retail transactions, scientific simulation, or telecommunication data that cannot be analyzed on traditional machines in acceptable times. As the grid is becoming a well-accepted computing infrastructure in science and industry, it provides more general data-mining services, algorithms, and applications. This framework helps analysts, scientists, organizations, and professionals to leverage grid capacity in supporting high-performance distributed computing for solving their data-mining problem in a distributed way. The creation of so-called knowledge grids on top of data and computational grids is the condition for meeting the challenges posed by the increasing demand for power and abstractions coming from complex data-mining scenarios in business, science, and engineering.

It is not only that DDM infrastructure is changing by offering new approaches through Web services together with grid technology. Basic data-mining algorithms also need changes in a distributed environment. Most off-the-shelf data-mining systems are designed to work as a monolithic centralized application. They normally download the relevant data to a centralized location and then perform the data-mining operations. This centralized approach does not work well in many of the emerging distributed, ubiquitous, possibly privacy-sensitive data-mining applications. A primary goal of DDM algorithms is to achieve the same or similar data-mining result as a centralized solution, without moving data from their original locations. Distributed approach assumes that local computation is done on each of the sites, and either a central site communicates with each distributed site to compute the global model, or a peer-to-peer architecture is used. In the latter case, individual nodes perform most of the tasks by communicating with neighboring nodes by message passing over an asynchronous network. Illustrative examples are networks of independent and intelligent sensors that are connected to each other in an ad hoc fashion. Some features of a distributed mining scenario are as follows:

- The system consists of multiple independent sites of data and computation.
- Sites exchange their results by communicating with other sites often through message passing.
- Communication between the sites is expensive and often represents a bottleneck.

- Sites have resource constraints, e.g. battery power in distributed sensor systems.
- Sites have privacy and/or security concerns.
- The system should have the ability to efficiently scale up because distributed systems may consist today of millions of nodes.
- The system should have the ability to function correctly in the presence of local site failures and also missing or incorrect data.

Obviously, the emphasis in DDM algorithms is on the local computation and communication. Local algorithms for DDM can be broadly classified under two categories:

- *Exact local algorithms:* Those algorithms guarantee to always terminate with precisely the same result that would have to be found by a centralized algorithm. Exact local algorithms are obviously more desirable, but are more difficult to develop and in some cases seemingly not possible.
- *Approximate local algorithms:* Those algorithms cannot make accuracy guaranteed by centralized solutions. They make a balance between quality of solution and system's responses.

Selection of a type of local algorithm depends on the data-mining problem and application domain including the amount of data and its dynamics. In general, approximate approaches are used in cases when the balance between accuracy and efficiency is important, and communications between sites represent a bottleneck. We will illustrate this balance between local computation and communication with a simple approximate algorithm useful in many data-mining applications. For example, if we want to compare the data vectors observed at different sites, the centralized approach will collect these vectors to the central computer and then compare the vectors using whatever metric is appropriate for the domain. DDM technology offers more efficient solutions for the problem using a simple randomized technique.

Vectors $a = (a_1, a_2,...,a_m)$ and $b = (b_1, b_2,...,b_m)$ are given at two distributed sites A and B, respectively. We want to approximate the Euclidean distance between them using a small number of messages and reduced data transfer between sites A and B. Centralized solution requires that one vector is transferred on the other site; that is, m components of one vector are transferred. How to obtain the same result with less than m data transfer? Note that the problem of computing the Euclidean distance between a pair of vectors a and b can be represented as the problem of computing the inner products as follows:

$$d^2(a,b) = (a \bullet a) + (b \bullet b) - 2(a \bullet b)$$

where $(a \bullet b)$ represents scalar product between vectors a and b defined as $\sum a_i b_i$ and $(a \bullet a)$ is special case of the scalar product representing square of the magnitude of the

vector a. The reader can easily check the previous relation. If, for example, the vectors a and b are $a = (1,2,3)$ and $b = (2,1,2)$, then the Euclidean distance may be calculated as $d^2 = 14 + 9 - 2 \times 10 = 3$. While products $(a \cdot a)$ and $(b \cdot b)$ can be computed locally and each result is a single value, the core challenge is to develop an algorithm for distributed inner product computation $(a \cdot b)$. A simple, communication-efficient randomized technique for computing this inner product between two vectors observed at two different sites may consist of the following steps:

1. Vectors a and b are given on two sites A and B, respectively. Site A sends to the site B a random number generator seed. *[That is only one passed message.]*

2. Both sites A and B cooperatively generate random matrix R with dimensions $k \times m$, where $k \ll m$. Each entry in the matrix R is generated independently and identically from some fixed distribution with mean zero and finite variance.

3. Based on the matrix R, sites A and B compute their own local matrix products:

$$\hat{a} = R\,a \text{ and } \hat{b} = R\,b$$

Dimensions of new local vectors $\hat{a}$ and $\hat{b}$ are k, and that means significantly lower than initial lengths of m.

4. Site A sends the resulting vector $\hat{a}$ to the site B. *[That represents k passed messages.]*

5. Site B computes approximate inner product $(a \cdot b) = (\hat{a}^T \cdot \hat{b})/k$.

So, instead of sending an m-dimensional vector to the other site, the algorithm sends only a $(k + 1)$-dimensional vector where $k \ll m$ (k is a user-defined parameter). The inner product of vectors can still be estimated accurately with lower communication load.

In the DDM literature, one of two assumptions is commonly adopted as to how data is distributed across sites: (1) homogeneously or horizontally partitioned or (2) heterogeneously or vertically partitioned. Both viewpoints assume that the data tables at each distributed site are partitions of a single global table. It is important to stress that the global table viewpoint is strictly conceptual. It is not necessarily assumed that such a table was physically realized and partitioned to form the tables at each site. In the homogeneous case, the global table is horizontally partitioned. The tables at each site are subsets of the global table; they have exactly the same attributes. Figures 12.30a illustrate the homogeneously distributed case using an example from weather data where both tables use the same three attributes. In the heterogeneous case, the table is vertically partitioned, where each site contains a subset of columns. That means sites do not have the same attributes. However, samples at each site are assumed to contain a unique identifier to facilitate matching, and Figure 12.30b illustrates this case. The tables at distributed sites have different attributes and samples are linked through a unique identifier, Patient ID.

DDM technology support different data-mining tasks including classification, prediction, clustering, market basket analysis, and outliers' detection. A solution for each of these tasks may be implemented with variety of DDM algorithms. For

(a)

Site 1

City	Humidity	Temperature
Louisville	85%	83°F
Cincinnati	77%	81°F
Nashville	89%	85°F

Site 2

City	Humidity	Temperature
Seattle	67%	73°F
Miami	77%	91°F
Huston	56%	95°F

(b)

Site 1

Patient ID	Temperature	Hearth rate
1	97°F	75
2	98.3°F	68
3	99.9°F	72
4	101°F	80

Site 2

Patient ID	Red Cells	White Cells
1	4.2×10^3	4×10^9
2	6.2×10^3	7.2×10^9
3	6.7×10^3	6.1×10^9
4	5.1×10^3	4.8×10^9

Figure 12.30. (a) Horizontally vs. (b) vertically partitioned data.

example, distributed Apriori has several versions for frequent itemset generation in distributed transactional database. They usually require multiple synchronizations and communication steps. Most of these implementations assume that platforms are homogeneous, and therefore the data sets are partitioned evenly among the sites. However, in practice, both the data sets and the processing platforms are more likely to be heterogeneous running multiple and different systems and tools. This leads to unbalanced data set distributions and workloads causing additional problems in implementation.

One of recent trends is online mining technology used for monitoring in distributed sensor networks. This is because deployments of large-scale distributed sensor networks are now possible owing to hardware advances and increasing software support. Online data mining, also called *data stream mining*, is concerned with extracting patterns, detecting outliers, or developing dynamic models of system's behavior from continuous data streams such as those generated by sensor networks. Because of the massive amount of data and the speed of which the data are generated, many data-mining applications in sensor networks require in-network processing such as aggregation to reduce sample size and the communication overhead. Online data mining in sensor networks offers many additional challenges, including:

- limited communication bandwidth,
- constraints on local computing resources,
- limited power supply,
- need for fault tolerance, and
- asynchronous nature of the network.

Obviously, data-mining systems have evolved in short period of time from stand-alone programs characterized by single algorithms with little support for the entire knowledge-discovery process to integrated systems incorporating several mining

algorithms, multiple users, communications, and various and heterogeneous data formats and distributed data sources. Although many DDM algorithms are developed and deployed in variety of applications, the trend will be illustrated in this book with only one example of a distributed clustering algorithm.

12.4.1 Distributed DBSCAN Clustering

Distributed clustering assumes that samples to be clustered reside on different sites. Instead of transmitting all samples to a central site where we can apply one of standard clustering algorithms to analyze the data locally, the data are clustered independently on the distributed local sites. Then, in a subsequent step, the central site is trying to establish a global clustering model based on the downloaded local models, i.e. summarized representatives of local data. Distributed clustering is carried out on two different levels, i.e. the local level and the global level (Figure 12.31). On the local level, all sites carry out a clustering independently from each other. Communication with central site and determining global model should reflect an optimum trade-off between complexity and accuracy of the algorithm.

Local models consist of a set of representatives for each locally found cluster. A representative is a good approximation for samples residing on the corresponding local site. The local model is transferred to a central site, where the local models are merged in order to form a global model. The representation of local models should be enough simple so there will be no overload in communications. At the same time, local model should be enough informative to support high quality of approximate

Figure 12.31. System architecture for distributed clustering.

global clustering. The global model is created by analyzing and integrating local representatives. The resulting global clustering is sent back, at the end of the process, to all local sites.

This global distributed framework may be more precisely specified when we implement specific clustering algorithm. The density-based clustering algorithm DBSCAN is a good candidate, because it is robust to outliers, is easy to implement, supports clusters of different shapes, and allows incremental online implementation. Main steps of the algorithm are explained in Chapter 9, and the same process is applied locally. To find local clusters, DBSCAN starts with an arbitrary core object p, which is not yet clustered and retrieves all objects density-reachable from p. The retrieval of density-reachable objects is performed in iterations until all local samples are analyzed. After having clustered the data locally, we need a small number of representatives that will describe the local clustering result accurately. For determining suitable representatives of the clusters, the concept of *specific core points* is introduced.

Let C be a local cluster with respect to the given DBSCAN parameters ε and *MinPts*. Furthermore, let $Cor_C \subseteq C$ be the set of core points belonging to this cluster. Then $Scor_C \subseteq C$ is called a *complete set of specific core points of C* iff the following conditions are true:

- $Scor_C \subseteq Cor_C$
- $\forall s_i, s_j \subseteq Scor_C: s_i \notin Neighborhood_\varepsilon (s_j)$
- $\forall c \in Cor_C, \exists s \in Scor_C: c \in Neighborhood_\varepsilon (s)$

$Scor_C$ set of points consists of very small number of *specific core points* that describes the cluster *C*. For example, given in Figure 12.32a, sites 2 and 3 have only one *specific core point*, while site 1, because of the cluster shape, has two *specific core points*. To further simplify the representation of local clusters, the number of specific core points, $|Scor_C| = K$, is used as input parameters for a further local "clustering step" with an adapted version of *K-means*. For each cluster *C* found by *DBSCAN*, *k-means* use $Scor_C$ points as starting points. The result is $K = |Scor_C|$ subclusters and centroids within *C*.

Each local model $LocalModel_k$ consists of a set of m_k pairs: a representative *r* (complete specific core point) and a ε radius value. The number *m* of pairs transmitted from each site *k* is determined by the number *n* of clusters C_i found on site *k*. Each of these pairs (r, ε_r) represents subset of samples that are all located in corresponding local cluster. Obviously, we have to check whether it is possible to merge two or more of these clusters, found on different sites, together. That is the main task of a global modeling part. To find such a global model, the algorithm continues with the density-based clustering algorithm DBSCAN again, but only for collected representatives from local models. Because of characteristics of these representative points, the parameter $MinPts_{global}$ is set to 2, and radius ε_{global} value should be set generally close to $2\varepsilon_{local}$.

(a)

(b)

(c)

Figure 12.32. Distributed DBSCAN clustering (Januzaj 2003). (a) Local clusters. (b) Local representatives. (c) Global model with $\varepsilon_{\text{global}} = 2\varepsilon_{\text{local}}$.

In Figure 12.32, an example of distributed DBSCAN for $\varepsilon_{\text{global}} = 2\varepsilon_{\text{local}}$ is depicted. In Figure 12.32a the independently detected clusters on sites 1, 2, and 3 are represented. The cluster on site 1 is represented using K-means by two representatives, R_1 and R_2, whereas the clusters on site 2 and site 3 are only represented by one representative as shown in Figure 12.32b. Figure 12.32c illustrates that all four local clusters from the different sites are merged together to one large cluster. This integration is obtained by using an $\varepsilon_{\text{global}}$ parameter equal to $2\varepsilon_{\text{local}}$. Figure 12.32c also makes clear that an $\varepsilon_{\text{global}} = \varepsilon_{\text{local}}$ is insufficient to detect this global cluster. When the final global model is obtained, the model is distributed to local sites. This model makes corrections comparing previously found local models. For example, in the local clustering, some points may be left as outliers, but with the global model they may be integrated into modified clusters.

12.5 CORRELATION DOES NOT IMPLY CAUSALITY!

An associational concept is any relationship that can be defined in terms of a frequency-based joint distribution of observed variables, while a causal concept is any relationship that cannot be defined from the distribution alone. Even simple examples show that the associational criterion is neither necessary nor sufficient for causality confirmation. For example, data mining might determine that males with income between $50,000 and $65,000 who subscribe to certain magazines are likely purchasers of a product you want to sell. While you can take advantage of this pattern, say, by aiming your marketing at people who fit the pattern, you should not assume that any of these factors (income, type of magazine) *cause* them to buy your product. The predictive relationships found via data mining are not necessarily *causes* of an action or behavior.

The research questions that motivate many studies in the health, social, and behavioral sciences are not statistical but causal in nature. For example, what is the efficacy of a given drug in a given population, or what fraction of past crimes could have been avoided by a given policy? The central target of such studies is determining of cause–effect relationships among variables of interests, for example, treatments–diseases or policies–crime, as preconditions–outcomes relationships. In order to express causal assumptions mathematically, certain extensions are required in the standard mathematical language of statistics, and these extensions are not generally emphasized in the mainstream literature and education.

The aim of standard statistical analysis, typified by regression and other estimation techniques, is to infer parameters of a distribution from samples drawn of that distribution. With the help of such parameters, one can infer associations among variables or estimate the likelihood of past and future events. These tasks are managed well by standard statistical analysis so long as experimental conditions remain the same. Causal analysis goes one step further; its aim is to infer aspects of the data-generation process. Associations characterize static conditions, while causal analysis deals with changing conditions. There is nothing in the joint distribution of symptoms and diseases to tell us that curing the former would or would not cure the latter.

Drawing analogy to visual perception, the information contained in a probability function is analogous to a geometrical description of a three-dimensional object; it is sufficient for predicting how that object will be viewed from any angle outside the object, but it is insufficient for predicting how the object will be deformed if manipulated and squeezed by external forces. The additional information needed for making predictions such as the object's resilience or elasticity is analogous to the information that causal assumptions provide. These considerations imply that the slogan "correlation does not imply causation" can be translated into a useful principle: one cannot substantiate causal claims from associations alone, even at the population level. Behind every causal conclusion, there must lie some causal assumption that is not testable in observational studies.

Any mathematical approach to causal analysis must acquire new notation for expressing causal assumptions and causal claims. To illustrate, the syntax of probability calculus does not permit us to express the simple fact that "symptoms do not cause diseases," let alone draw mathematical conclusions from such facts. All we can say is that two events are dependent—meaning that if we find one, we can expect to encounter the other, but we cannot distinguish statistical dependence, quantified by the conditional probability $P(disease/symptom)$ from causal dependence, for which we have no expression in standard probability calculus. Symbolic representation for the relation "symptoms cause disease" is distinct from the symbolic representation of "symptoms are associated with disease."

The need to adopt a new notation, foreign to the province of probability theory, has been traumatic to most persons trained in statistics, partly because the adaptation of a new language is difficult in general and partly because statisticians—this author included—have been accustomed to assuming that all phenomena, processes, thoughts, and modes of inference can be captured in the powerful language of probability theory. Causality formalization requires new mathematical machinery for cause–effect analysis and a formal foundation for counterfactual analysis including concepts such as "path diagrams," "controlled distributions," causal structures, and causal models.

12.5.1   Bayesian Networks

One of the powerful aspects of graphical models is that a specific graph can make probabilistic statements for a broad class of distributions. In order to motivate the use of directed graphs to describe probability distributions, consider first an arbitrary joint distribution $p(a, b, c)$ over three variables, a, b, and c. By application of the product rule of probability, we can write the joint distribution in the form

$$p(a,b,c) = p(c \mid a,b)p(a,b) = p(c \mid a,b)p(b \mid a)p(a)$$

We now represent the right-hand side of the equation in terms of a simple graphical model as follows: First, we introduce a node for each of the random variables a, b, and c and associate each node with the corresponding conditional distribution on the right-hand side of the equation. Then, for each conditional distribution, we add directed links (arrows) to the graph from the nodes corresponding to the variables on which the distribution is conditioned. Thus for the factor $p(c \mid a, b)$, there will be links from nodes a and b to node c, whereas for the factor $p(a)$ there will be no incoming links as it is presented in Figure 12.33a. If there is a link going from a node a to a node b, then we say that node a is the *parent* of node b, and we say that node b is the *child* of node a.

For given K variables, we can again represent a joint probability distribution as a directed graph having K nodes, one for each conditional distribution, with each node having incoming links from all lower numbered nodes. We say that this graph is *fully connected* because there is a link between every pair of nodes. Consider now the graph shown in Figure 12.33b, which is not a fully connected graph because, for instance,

(a)

(b)

Figure 12.33. A directed graphical model representing the joint probability distribution over a set of variables. (a) Fully connected. (b) Partially connected.

there is no link from x_1 to x_2 or from x_3 to x_7. We may transform this graph to the corresponding representation of the joint probability distribution written in terms of the product of a set of conditional distributions, one for each node in the graph. The joint distribution of all seven variables is given by

$$p(x_1, x_2, \ldots, x_7) = p(x_1)p(x_2)p(x_3)p(x_4 \mid x_1, x_2, x_3)p(x_5 \mid x_1, x_3)p(x_6 \mid x_4)p(x_7 \mid x_4, x_5)$$

Any joint distribution can be represented by a corresponding graphical model. It is the *absence* of links in the graph that conveys interesting information about the properties of the class of distributions that the graph represents. We can interpret such models as expressing the processes by which the observed data arose, and in many situations we may draw conclusions about new samples from a given probability distribution. The directed graphs that we are considering are subject to an important restriction, namely, that there must be no *directed cycles*; in other words, there are no closed paths within the graph such that we can move from node to node along links following the direction of the arrows and end up back at the starting node. Such graphs are also called *directed acyclic graphs* (*DAGs*).

An important concept for probability distributions over multiple variables is that of *conditional independence*. Consider three variables a, b, and c, and suppose that the conditional distribution of a, given b and c, is such that it does not depend on the value of b so that

$$p(a \mid b, c) = p(a \mid c)$$

We say that a is conditionally independent of b given c. This can be extended in a slightly different way if we consider the joint distribution of a and b conditioned on c, which we can write in the form

$$p(a, b \mid c) = p(a \mid b, c)p(b \mid c) = p(a \mid c)p(b \mid c)$$

(a)   (b)   (c)

Figure 12.34. Joint probability distributions show different dependencies between variables *a*, *b*, and *c*.

The joint distribution of *a* and *b*, conditioned on *c*, may be factorized into the product of the marginal distribution of a and the marginal distribution of *b* (again both conditioned on *c*). This says that the variables a and b are statistically independent, given *c*. This independence may be presented in a graphical form in Figure 12.34a. The other typical joint distributions may be graphically interpreted. The distribution for Figure 12.34b represents the case

$$p(b,c \mid a) = p(c \mid a)p(b \mid c)$$

while for Figure 12.34c the probability $p(c \mid a, b)$ is under the assumption that variables *a* and *b* are independent $p(a, b) = p(a)\,p(b)$.

In general, graphical models may capture the *causal* processes by which the observed data was generated. For this reason, such models are often called *generative* models. We could make previous models in Figure 12.33 generative by introducing a suitable prior distribution $p(x)$ for all input variables (these are variables—nodes without input links). For the case in Figure 12.33a, this is a variable *a*, and for the case in Figure 12.33b, these are variables x_1, x_2, and x_3. In practice, producing synthetic observations from a generative model can prove informative in understanding the form of the probability distribution represented by that model.

This preliminary analysis about joint probability distributions brings us to the concept of Bayesian networks (BN), which are also called in the literature belief networks or probabilistic networks. The nodes in a BN represent variables of interest (e.g., the temperature of a device, the gender of a patient, the price of a product, the occurrence of an event), and the links represent dependencies among the variables. Each node has *states*, or a set of probable values for each variable. For example, the weather could be cloudy or sunny, an enemy battalion could be near or far, symptoms of a disease are present or not present, and the garbage disposal is working or not working. Nodes are connected with an arrow to show causality, also indicating the direction of influence. These arrows are called *edges*. The dependencies are quantified by conditional probabilities for each node given its parents in the network. Figure 12.35 presents some BN architectures, initially without probability distributions. In general, we can formally describe a BN as a graph in which the following holds:

Figure 12.35. Two examples of Bayesian network architectures.

1. A set of random variables makes up the nodes of the network.
2. A set of directed links connects pairs of nodes. The intuitive meaning of an arrow from node X to node Y is that X has a *direct influence* on Y.
3. Each node has a *conditional probability table* (CPT) that quantifies the effects that the parents have on the node. The parents of a node X are all those nodes that have arrows pointing to X.
4. The graph has no directed cycles (hence a DAG).

Each node in the BN corresponds to a random variable X and has a probability distribution of the variable $P(X)$. If there is a directed arc from node X to node Y, this indicates that X has a direct influence on Y. The influence is specified by the conditional probability $P(Y|X)$. Nodes and arcs define a *structure* of the BN. Probabilities are *parameters* of the structure.

We turn now to the problem of inference in graphical models, in which some of the nodes in a graph are clamped to observed values, and we wish to compute the posterior distributions of one or more subsets of other nodes. The network supports the computation of the probabilities of any subset of variables given evidence about any other subset. We can exploit the graphical structure both to find efficient algorithms for inference and to make the structure of those algorithms transparent. Specifically, many inference-based algorithms can be expressed in terms of the propagation of local *probabilities* around the graph. A BN can be considered as a probabilistic graph in which the probabilistic knowledge is represented by the topology of the network and the conditional probabilities at each node. The main purpose of building knowledge on probabilities is to use it for inference, i.e., computing the answer for particular cases about the domain.

For example, we may assume that rain causes the grass to get wet. Causal graph in Figure 12.36 explains cause–effect relation between these variables including corresponding probabilities. If $P(\text{Rain}) = P(R) = 0.4$ is given, that also means $P(\neg R) = 0.6$. Also, note that the sum of presented conditional probabilities is not equal to 1. If you analyze the relations between probabilities, $P(W | R) + P(\neg W | R) = 1$, and also $P(W | \neg R) + P(\neg W | \neg R) = 1$, not the sum of given probabilities. In these expressions

$$p(W\,|\,R)=0.9$$
$$p(W\,|\,\neg R)=0.2$$

$$\text{Rain} \longrightarrow \text{Wet grass}$$

$$p(R)=0.4$$

Figure 12.36. Simple causal graph.

R means "Rain," and W means "Wet grass." Based on the given BN, we may check what is the probability of "Wet grass":

$$P(W)=P(W\,|\,R)\,P(R)+P(W\,|\,\neg R)\,P(\neg R)=0.9\times 0.4+0.2\times 0.6=0.48\,(\text{or}\,48\%)$$

Bayesian rule allow us to invert the dependencies and obtain probabilities of parents in the graph based on probabilities of children. That could be useful in many applications, such as determining probability of a diagnosis based on symptoms. For example, based on the BN in Figure 12.36, we may determine conditional probability $P(\text{Rain}\,|\,\text{Wet grass}) = P(R|W)$. We know that

$$P(R,W)=P(W\,|\,R)\,P(R)=P(R\,|\,W)\,P(W)$$

and therefore

$$P(R\,|\,W)=\frac{[P(W\,|\,R)\,P(R)]}{P(W)}=\frac{[P(W\,|\,R)\,P(R)]}{[P(W\,|\,R)\,P(R)+P(W\,|\,\neg R)\,P(\neg R)]]}$$

$$=0.9^*0.4/(0.9^*0.4+0.2^*0.6)=0.75$$

Let us include now more complex problem and more complex BN represented in Figure 12.37. In this case we have three nodes, and they are connected serially, often called head-to-tail connections of three events. Now additional event "Cloudy" with yes and no values is included as a variable at the beginning of the network. *R* node blocks a path from *C* to *W*, which separates them. If *R* node is removed, there is no path from *C* to W. Therefore, relation between conditional probabilities in the graph is given as $P(C, R, W) = P(C)^* P(R|C)^* P(W|R)$.

In our case, based on the BN in Figure 12.37, it is possible to determine and use "forward" and "backward" conditional probabilities as we represented in the previous BN. We are starting with

$$P(W\,|\,C)=P(W\,|\,R)^*P(R\,|\,C)+P(W\,|\,\neg R)^*P(\neg R\,|\,C)=0.9^*0.8+0.2^*0.2=0.76$$

Then, we may use Bayesian rule for inverted conditional probabilities:

$$P(C\,|\,W)=\frac{[P(W\,|\,C)^*P(C)]}{P(W)}=0.65\quad(P(W)\text{ requires detailed computation!!})$$
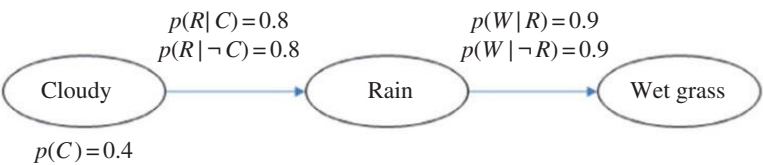
Figure 12.37. An extended causal graph.
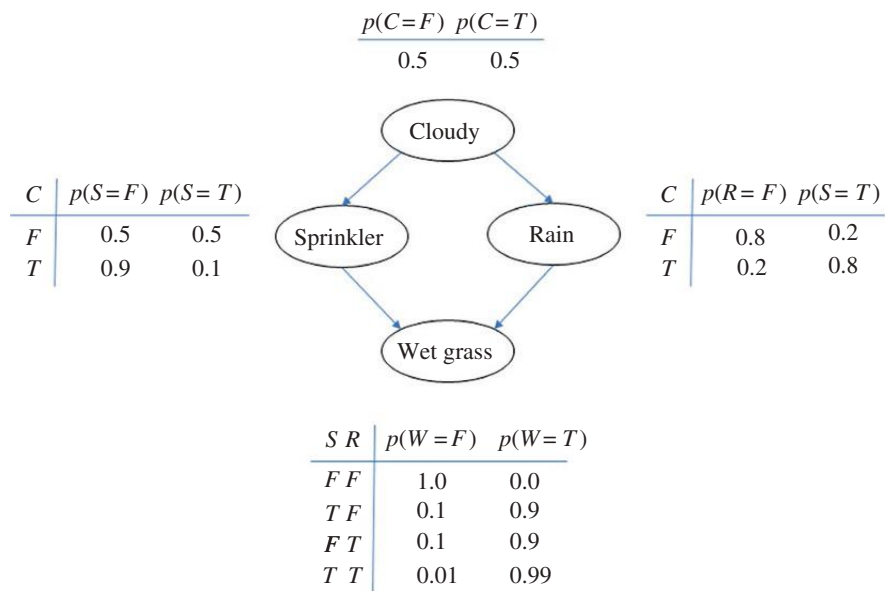
Figure 12.38. Four-node architecture of a Bayesian network.

More complex connections may be analyzed in BN. Figure 12.38 shows the graph structure and the assumed input parameters.

The parameters of a graphical model are represented by the conditional probability distributions in a form of CPTs for each node given its parents. The simplest form of a formalized distribution, a CPT, is suitable when the nodes are discrete valued. All nodes in Figure 12.38 are represented with discrete set of states and corresponding CPTs. For example, "Sprinkler" node (S) may be "on" and "off," and it is represented in the table with T and F values. Sprinkler CPT is generated including input discrete values for node "Cloudy," C. Many algorithms for BN analysis may be expressed in terms of the propagation of probabilities through the graph.

All probabilistic models, no matter how refined and accurate, Bayesian included, describe a distribution over possible observed events, but say nothing about what will happen if a certain intervention occurs. For example, what if I turn on the sprinkler? What effect does that have on the season or on the connection between wetness and

slipperiness? A causal network is a BN with the added property that the parents of each node are its direct causes. In such a network, the result of an intervention is obvious: the sprinkler node is set to "on," and the causal link between the season and the sprinkler is removed. All other causal links and conditional probabilities remain intact. This added property endows the causal network with the capability of representing and responding to external or spontaneous changes. For example, to represent a disabled sprinkler in the story of Figure 12.38, we simply delete from the network all links incident to the node Sprinkler. To represent the policy of turning the sprinkler off if it rains, we simply add a link between Rain and Sprinkle. Such changes would require much greater remodeling efforts if the network were not constructed along the causal direction. This remodeling flexibility may well be cited as the ingredient that manages novel situations instantaneously, without requiring training or adaptation of the model.

12.6   PRIVACY, SECURITY, AND LEGAL ASPECTS OF DATA MINING

An important lesson of the Industrial Revolution was that the introduction of new technologies can have a profound effect on our ethical principles. In today's Information Revolution we strive to adapt our ethics to diverse concepts in cyberspace. The recent emergence of very large databases, and their associated data-mining tools, presents us with yet another set of ethical challenges to consider. The rapid dissemination of data-mining technologies calls for an urgent examination of their social impact. It should be clear that data mining itself is not socially problematic. Ethical challenges arise when it is executed over data of a personal nature. For example, the mining of manufacturing data is unlikely to lead to any consequences of a personally objectionable nature. However, mining clickstreams of data obtained from Web users initiate a variety of ethical and social dilemmas. Perhaps the most significant of these is the invasion of privacy, but that is not the only one.

Thanks to the proliferation of digital technologies and networks such as the Internet and tremendous advances in the capacity of storage devices and parallel decreases in their cost and physical size, many private records are linked and shared more widely and stored far longer than ever before, often without the individual consumer's knowledge or consent. As more everyday activities move online, digital records contain more detailed information about individuals' behavior. Merchants' record data are no longer only on what individuals buy and how they pay for their purchases. Instead, those data include every detail of what we look at, the books we read, the movies we watch, the music we listen to, the games we play, and the places we visit. The robustness of these records is difficult to overestimate and is not limited to settings involving commercial transactions. More and more computers track every moment of most employees' days. Email and voice mail are stored digitally; even the content of telephone conversations may be recorded. Digital time clocks and entry keys record physical movements. Computers store work product, text messages, and Internet browsing records—often in keystroke-by-keystroke detail, they monitor employee behavior.

The ubiquitous nature of data collection, analysis, and observation is not limited only to the workplace. Digital devices for paying tolls, computer diagnostic equipment in car engines, and global positioning services that are increasingly common on passenger vehicles record every mile driven. Cellular telephones and personal digital assistants record not only call and appointment information but also location and transmit this information to service providers. ISPs record online activities, digital cable and satellite record what we watch and when, alarm systems record when we enter and leave our homes, and all of these data are held by third parties. Information on our browsing habits is available to both the employer and the ISP. If an employee buys an airline ticket through an online travel service, such as Travelocity or Expedia, the information concerning that transaction will be available to the employer, the ISP, the travel service, the airline, and the provider of the payment mechanism, at a minimum.

All indications are that this is just the beginning. Broadband Internet access into homes has not only increased the personal activities we now engage in online but also created new and successful markets for remote computer backup and online photo, email, and music storage services. With Voice over IP telephone service, digital phone calls are becoming indistinguishable from digital documents: both can be stored and accessed remotely. Global positioning technologies are appearing in more and more products, and radio-frequency identification tags are beginning to be used to identify high-end consumer goods, pets, and even people.

Many individuals are unaware of the extent of the personal data stored, analyzed, and used by governments' institutions, private corporations, and research labs. It is usually only when things go wrong that individuals exercise their rights to obtain this data and seek to eliminate or correct it. For many of those, whose records are accessed through data mining, we do not know it is happening and may never find out because nothing incriminating is signaled. But we still know that data mining allows companies to accumulate and analyze vast amounts of information about us, sufficient perhaps to create, with the help of data mining, what some have called personality or psychological "mosaics" of the subjects. One result of the entry into the information age is that faceless bureaucrats (in company, in government, everywhere) will be able to compile dossiers on anyone and everyone, for any reason or for no reason at all. The possibility, even if slim, that this information could somehow be used to our detriment or simply revealed to others can create a chilling effect on all these activities.

Data and the information derived from that data using data mining is an extremely valuable resource for any organization. Every data-mining professional is aware of this, but few are concentrated on the impact that data mining could have on privacy and the laws surrounding the privacy of personal data. Recent survey showed that data-mining professionals "prefer to focus on the advantages of web-data mining instead of discussing the possible dangers." These professionals argued that Web data mining does not threaten privacy. One might wonder why professionals are not aware of or concerned over the possible misuse of their work and the possible harm it might cause to individuals and society. Part of the reason some professionals are not concerned over the possible misuse of their work and the possible harm it might cause

might lie in the explanations that "they are primary technical professionals and somebody else should take care about these social and legal aspects." But sensible regulations of data mining depend on understanding of its many variants and its potential harms. Therefore, technical professional has to be a part of the team, often leading, which will try to solve privacy challenges.

The key ethical issues in mining personal data are that people are generally:

1. not aware that their personal information is being gathered,
2. do not know to what use the data will be made, and/or
3. have not consented to such collecting or data use.

In order to alleviate concerns about data privacy, a number of techniques have recently been proposed in order to perform the data-mining tasks in a privacy-preserving way. These techniques for performing privacy-preserving data mining are drawn from a wide array of related topics such as cryptography and information hiding. Most privacy-preserving data-mining methods apply a transformation that reduces the effectiveness of the underlying data when it is applied to data-mining methods or algorithms. In fact, there is a natural trade-off between privacy and accuracy, though this trade-off is affected by the particular algorithm that is used for privacy preservation. The key directions in the field of privacy-preserving data mining include:

- *Privacy-preserving data publishing:* These techniques tend to study different transformation methods associated with privacy. They concentrate on how the perturbed data can be used in conjunction with classical data-mining methods.
- *Changing the results of data-mining applications to preserve privacy:* These techniques are concentrated on privacy of data-mining results where some results are modified in order to preserve the privacy. A classic example of such techniques are association rule hiding methods, in which some of the association rules are suppressed in order to preserve privacy.
- *Cryptographic methods for distributed privacy:* If the data are distributed across multiple sites, a variety of cryptographic protocols may be used in order to communicate among the different sites so that secure function computation is possible without revealing sensitive information.

Recent research trends propose that issues of privacy protection, currently viewed in terms of data *access*, be reconceptualized in terms of data *use*. From a technology perspective, this requires supplementing legal and technical mechanisms for access control with new mechanisms for *transparency* and *accountability* of data used in data-mining process. Current technical solutions of the impact of data mining on privacy have generally focused on limiting access to data at the point of collection or storage. Most effort has been put into the application of cryptographic and statistical techniques to construct finely tuned access-limiting mechanisms.

Even if privacy-preserving data-mining techniques prove to be practical, they are unlikely to provide sufficient public assurance that data-mining inferences conform to legal restrictions. While privacy-preserving data-mining techniques are certainly necessary in some contexts, they are not sufficient privacy protection without the transparency and accountability.

In the long run, access restriction alone is not enough neither to protect privacy nor to ensure reliable conclusions, and the best example of these challenges is Web and Web mining technology. As we leave the well-bounded world of enterprise databases and enter the open, unbounded world of the Web, data users need a new class of tools to verify that the results they see are based on data that is from trustworthy sources and is used according to agreed-upon institutional and legal requirements. The implications of data mining on digital social networks such as Facebook, WhatsApp, or Twitter may be enormous. Unless it is part of a public record designed for consumption by everyone or describes an activity observed by strangers, the stored information is rarely known outside our families, much less outside our social networks. An expectation that such information and potential derivatives will remain "private" on Internet is not anymore reasonable assumption from the social network perspective. One of the major contributors to these controversies is the absence of clear legal standards. Thirty years ago, the lack of relevant law was understandable: the technologies were new, their capacity was largely unknown, and the types of legal issues they might raise were novel. Today, it is inexplicable and threatens to undermine both privacy and security. Hence, we must develop technical, legal, and policy foundations for transparency and accountability of large-scale mining across distributed heterogeneous data sources. *Policy awareness* is a property of the Semantic Web still in development that should provide users with accessible and understandable views of the policies associated with resources.

The following issues related to privacy concerns may assist in individual privacy protection during a data-mining process and should be a part of the best data-mining practices:

- *Whether there is a clear description of a program's collection of personal information, including how the collected information will serve the program's purpose*. In other words, be transparent early on about a data-mining project's purpose. Clearly state up front the business benefits that will be achieved by data mining. Provide notice of the combining of the information from different sources. Companies, like Walmart or Kroger, store much of their business and customer data in large warehouses. Their customers are not told the extent of the information that is accumulated on them, how long it will be kept, nor the uses to which the data will be put, or other users with which data will be shared.

- *Whether information collected for one purpose will then be used for additional secondary purposes in the future*. Ensure that any new purpose of a project is consistent with the project's original purpose. Maintain oversight of data-mining project and create audit requirements.

- *Whether privacy protections are built in to systems in the early developmental stage*. Build in privacy considerations up front, and bring in all stakeholders at the beginning, including privacy advocates to get input from them. Ensure the accuracy of data entry.
- What type of action will be taken on the basis of information discovered through a data-mining process? Where appropriate, anonymize personal information. Limit the actions that may be taken as a result of unverified findings from data mining.
- *Whether there is an adequate feedback system for individuals to review and correct their personal information that is collected and maintained in order to avoid "false positives" in a data-mining program*. Determine whether an individual should have a choice in the collection of information. Provide notice to individuals about use of their personal information. Create a system where individuals can ensure that any incorrect personal information can be corrected.
- *Whether there are proper disposal procedures for collected and derived personal information that has been determined to be irrelevant*.

Some observers suggest that the privacy issues presented by data mining will be resolved by technologies, not by law or policy. But even the best technological solutions will still require a legal framework in which to operate, and the absence of that framework may not only slow their development and deployment, but make them entirely unworkable. Although there is no explicit right to privacy of personal data in the Constitution, legislation and court decisions on privacy are usually based on parts of the First, Fourth, Fifth, and Fourteenth Amendments. Except for healthcare and financial organizations and data collected from children, there is no law that governs the collection and use of personal data by commercial enterprises. Therefore, it is essentially up to each organization to decide how they will use the personal data they have accumulated on their customers. In early March 2005, hackers stole the personal information of 32,000 people from the databases of LexisNexis. The stolen data included Social Security numbers and financial information. Although the CEO of LexisNexis claimed that the information they collect is governed by the US Fair Credit Reporting Act, members of Congress disagreed. As a result of this and other large-scale identity thefts in recent years, Congress is considering new laws explaining what personal data a company can collect and share. For example, Congress is considering a law to prohibit almost all sales of Social Security numbers.

At the same time, especially since 9/11, government agencies have been eager to experiment with the data-mining process as a way of nabbing criminals and terrorists. Although details of their operation often remain unknown, a number of such programs have come to light since 2001. The DOJ, through the FBI, has been collecting telephone logs, banking records, and other personal information regarding thousands of Americans not only in connection with counterterrorism efforts but also in furtherance of ordinary law enforcement. A 2004 report by the Government Accountability Office

(GAO) found 42 federal departments—including every cabinet-level agency that responded to the GAO's survey—engaged in, or were planning to engage in, 122 data-mining efforts involving personal information (U.S. General Accounting Office, *Data Mining: Federal Efforts Cover a Wide Range of Uses* [GAO-04-548], May 2004, pp. 27–64). Recently US government recognized that sensible regulation of data mining depends on understanding its many variants and its potential harms, and many of these data-mining programs are reevaluated. In the United Kingdom, the problem is being addressed more comprehensively by the Foundation for Information Policy Research, an independent organization examining the interaction between IT and society with goals: to identify technical developments with significant social impact, commission research into public policy alternatives, and promote public understanding and dialogue between technologists and policy makers in the United Kingdom and Europe. It combines IT researchers with people interested in social impacts and uses a strong media presence to disseminate its arguments and educate the public.

There is one additional legal challenge related specifically to data mining. Today's privacy laws and guidelines, if they exist, protect data that is explicit, confidential, and exchanged between databases. However, there is no legal or normative protection for data that is implicit, nonconfidential, and not exchanged. Data mining can reveal sensitive information that is derived from nonsensitive data and metadata through the inference process. Information gathered in data mining is usually implicit patterns, models, or outliers in the data, and questionable is the application of privacy regulations primary written for traditional explicit data.

In addition to data privacy issues, data mining raises other social concerns. For example, some researchers argue that data mining and the use of consumer profiles in some companies can actually exclude groups of customers from full participation in the marketplace and limit their access to information.

As data mining increasingly affects decisions in domains protected by anti-discrimination law, there is much interest in algorithmically measuring and ensuring fairness in the field. What does it mean for a data-mining model to be fair or nondiscriminatory? The answer is not simple, but the basic idea is that the models were trained with data that reflect society's biases and algorithms amplify these biases. Although reliance on data and quantitative measures can help quantify and eliminate some of existing biases, latest research has warned that data-mining algorithms can also introduce totally new biases or perpetuate existing ones.

Recently social study research and numerous media, including highly respected *Nature* magazine, have pointed out plenty of anecdotal evidence that decision-making by data-mining algorithms may unintentionally discriminate people. For example, if one naively trains a model to filter resumes and find the most qualified candidates for certain jobs based on prior hires data, even if the algorithm is explicitly instructed to ignore "protected attributes" like race or gender, the results can turn out to be race or gender biased. It turns out that race and gender are correlated with other "unprotected" information like names, which the naive algorithm can use. When Harvard Professor Latanya Sweeney put her name into a search engine, she was delivered an ad saying

"Latanya Sweeney, Arrested?" and offered a background check for a fee. The result of that background check was that Dr. Sweeney had no arrest record, and an initial triggering of this ad was obviously deeply unfair and discriminatory to Dr. Sweeney. Searching on people whose first names indicated they were more likely to be black, like Latanya, was much more likely to produce this "Arrested? ad." If potential employers put Dr. Sweeney's name into a search engine, they might write her off immediately upon seeing the ad.

A vetted methodology for avoiding discrimination against protected attributes in data mining is lacking. An initial idea was based on naïve approach that require that the data-mining algorithm should ignore all protected attributes such as race, color, religion, gender, disability, or family status. However, this idea of "fairness through unawareness" is ineffective, and sometimes it is a source of discrimination by themselves. There is an entirely new field emerging at the intersection of computer science, law, and social study. In the long run it is necessary to develop fundamental scientific understanding for ensuring transparency and accountability of using data mining in the society. Within this goal guaranteeing fairness of algorithms is one of the key issues. Current approaches to fair data mining are typically focused on interventions at the data preparation, model learning, or including tuning postprocessing stages. While fair data-mining algorithms have already been proposed, the insight of underlying mechanisms how such discrimination happens from the computational perspective is not yet scientifically understood. We need to develop theoretical understanding how algorithms may become discriminatory and establish fundamental machine-learning principles for prevention. The state-of-the-art fair data mining should advance from heuristic repairing to proactive and theoretically supported prevention.

Good privacy and fairness protection not only can help to build support for data mining and other tools to enhance security, but it can also contribute to making those tools more effective. Still, as the data-mining and machine-learning industry makes significant progress in performing various tasks and actions in the everyday life, questions are raised regarding ethics, responsibilities, and human engagement in these real-time applications. Instead of highly publicized and discussed with confronting opinions application of self-driving cars, let us present an illustrative example of a smaller, locally developed application. They are also very important, although they are not always under enough society attention. These new applications often open the questions of a balance between attractiveness and usefulness of collected data and intelligent decisions on one hand and privacy, legality, and even fairness issues on the other hand.

An illustrative example is a high school in Hangzhou City, Zhejiang Province, located on the eastern coast of China, which has employed facial recognition technology to monitor students' attentiveness in a classroom. Three cameras at the front of the classroom scan students' faces every 30 seconds, analyzing their facial expressions to detect different moods—surprised, sad, antipathy, angry, happy, afraid, or neutral. The recordings are stored and averaged during each class. The system, called "smart classroom behavior management system," also analyzes students' actions during the classes, categorized into reading, listening, writing, standing up, raising hands, or

leaning on the desk. As the result of this monitoring system, the teacher's screen displays a list of student names deemed "not paying attention." Similar systems, based on facial recognition technology in schools, were used to serve meals to students at school cafeterias or pay for items at local stores.

As technology designers, we should provide information infrastructure that helps society to be more certain that data-mining power is used only in legally approved ways and that the data that may give rise to consequences for individuals is based on inferences that are derived from accurate, approved, fair, and legally available data. Future data-mining solutions reconciling any social issues must not only be applicable to the ever-changing technological environment but also flexible with regard to specific contexts and disputes.

12.7 CLOUD COMPUTING BASED ON HADOOP AND MAP/REDUCE

Modern information societies are defined by vast repositories of data, both public and private where new generation of applications must be able to scale up to adjust big data frameworks. Current technological advances in storage, communication, and computations have enabled cost-effective capture of big data in a timely manner. Big data find a solution for more computing power in cloud computing platforms. As the National Institute of Standards and Technology defined, "Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction." The growth of cloud computing enables businesses to turn into new service-based IT solutions overcoming challenges of big investments in IT infrastructure. The cloud may be considered as a marketplace, where the storage and computing capabilities of a cloud can be leased as necessary.

Cloud computing offers enterprises and users high scalability, high availability, and high reliability. It can improve resource utilization efficiency and can reduce the cost of business information construction, investment, and maintenance. As the public cloud services from Amazon, Google, Microsoft, and other specialized companies become more sophisticated and better developed, more and more companies are migrating toward the cloud computing platform.

The core of cloud computing model is a new parallel computing architecture, often called cluster computing. It is organized as large number of computing nodes (standard processing modules with processor, central memory, and disk), which are stored together on racks, usually 8–64 on each rack. The nodes on a single rack are connected by a gigabit Ethernet network. There can be many racks multiple nodes, and racks are connected by another level of network through a special high speed network or switch. Figure 12.39 highlights main components of the architecture of a large-scale parallel computing system.
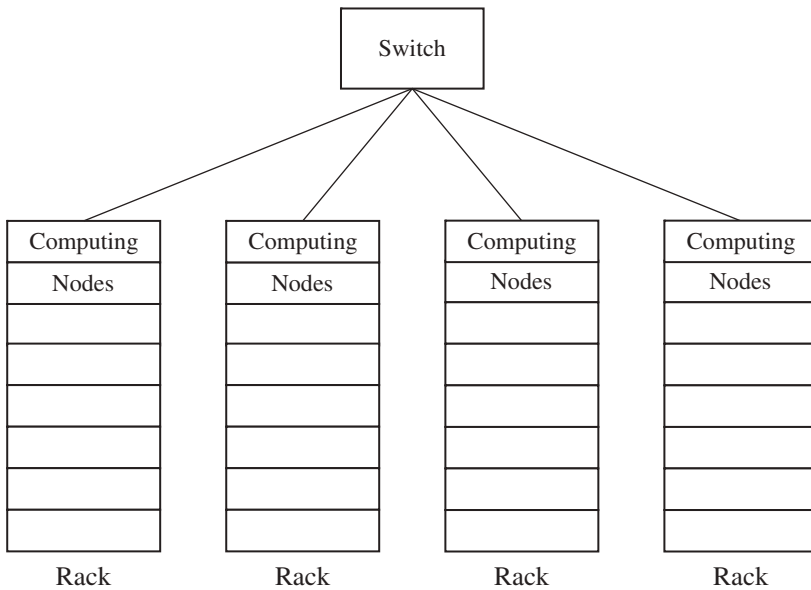
Figure 12.39. Computing nodes are organized into racks, and racks are interconnected by a switch.

These new computing facilities have given rise to a new generation of programming systems, which take advantage of the power of parallelism and at the same time avoid the reliability problems that arise when the computing hardware consists of thousands of independent components. Series of Google products has opened the door to the proposed massive data storage, and cloud computing becomes the de facto standard in a big data field, with Google as a technology leader. While Google's technology was not open source, Yahoo was first to offer an open-source software framework designed for supporting data-intensive applications called Hadoop.

Instead of relying on expensive proprietary hardware to store and process the data, Hadoop enables distributed processing of large amounts of data on large clusters of commodity servers. Hadoop has many advantages comparing other previously developed architectures, and it is particularly suitable for big data management and analysis because of:

- *High scalability*: Hadoop allows hardware infrastructure to be scaled up and down with no need to change data formats. The system will automatically redistribute data and computation jobs to accommodate hardware changes.
- *Cost efficiency*: Hadoop brings massively parallel computation to commodity servers, leading to a sizeable decrease in cost per terabyte of storage, which makes massively parallel computation affordable for the ever-growing volume of big data.

- *Fault tolerance*: Missing data and computation failures are common in big data analytics. Hadoop can recover the data and computation failures caused by node breakdown or network congestion.

The foundation of Hadoop is Hadoop Distributed File System (HDFS). This module is responsible for storing and retrieving data. It is designed and optimized to deal with large amount of data. Files are composed of chunks of about 64 megabytes, and each chunk is replicated several times on different compute nodes or racks enabling HDFS protection from failures mechanisms. HDFS is the only mandatory module of the Hadoop framework, and all other components are optional. Second important component that represents core module of Hadoop and that is essential for big data applications on the cloud is MapReduce.

MapReduce, a powerful programming framework, enables the automatic paralleling and distribution of computation applications on large clusters of commodity machines. It gives a style of solving specific problems when big data is available. One of the most significant advantages of MapReduce is that it provides abstractions that hide many implementation details from the programmer. MapReduce can be viewed as the first breakthrough in the new abstractions that allow us to organize computations, not over individual machines, but over entire clusters. Additionally, MapReduce can also provide great fault tolerance ability, which is important for working with the large data sets. The core idea of MapReduce is to divide massive data into small chunks firstly and then deal with these chunks in parallel and in a distributed manner to generate intermediate results. By aggregating all the intermediate results, the final result is derived.

MapReduce codifies this generic "recipe" for processing large data sets into two stages. In the first so-called Map function, a user-specified computation is applied over all input records in a data set. It takes a collection of input objects and turns each into zero or more key–value pairs. Key values are not necessarily unique. These Map operations occur in parallel and yield intermediate output that is then aggregated by the second function called Reduce. Reduce tasks are also parallel, and they combine the elements on each key–value list by applying the function written by the user. The results produced by all the Reduce tasks form the aggregated output of the map–reduce process. This MapReduce programming model has achieved massive scalability in real-world applications across hundreds or thousands of servers within a Hadoop cluster.

An illustrative example of a MapReduce process could explain all advantages of the framework. Suppose that there is a large collection of text documents in a database and the task is to count the number of times each distinct word is occurring in the document corpus. Figure 12.40 illustrates the basic steps in MapReduce process to perform the required task. To simplify explanations, we selected to process the database with only three documents, and assumption is that there are three servers for Map functions and two servers for Reduce functions. Of course, scalability of the solution enables to extend the problem to millions of documents with hundreds or even thousands of Map and Reduce servers in the cloud, while the example only captures the essence of MapReduce framework.

(a)   (b)   (c)   (d)

Map input   Map output   Reduce input   Reduce output

Web page 1
The weather is good

Worker 1
(the, 1)
(weather, 1)
(is, 1)
(good, 1)

Worker 1
(the, 1)

Worker 1
(the, 1)

Worker 2
(is, 1)
(is, 1)
(is, 1)

Worker 2
(is, 3)

Web page 2
Today is good

Worker 2
(today, 1)
(is, 1)
(good, 1)

Worker 3
(weather, 1)
(weather, 1)

Worker 3
(weather, 2)

Web page 3
Good weather is good

Worker 3
(good, 1)
(weather, 1)
(is, 1)
(good, 1)

Worker 4
(today, 1)

Worker 4
(today, 1)

Worker 5
(good, 1)
(good, 1)
(good, 1)
(good, 1)

Worker 5
(good, 4)

Figure 12.40. MapReduce process for word count problem. Based on Jeff Dean & Sanjay Ghemawat slides [Google inc.].

In a simple example, because there are only three documents and three Map servers, each document (at Figure 12.40a) is assigned to a separate server, and the Map function is decomposing the document text into set of distinct words (Figure 12.40b). The final result of the Map phase is an integrated list of key–value pairs for all documents. In our case, it is a list of all words with its count in each document (Figure 12.40c). Results of a Map phase are now distributed between Reduce servers. Sorted list of words is distributed between two Reduce servers in our example. Reduce phase accumulates the words that are repeating in different documents, and the Reduce aggregation produces final word count results (Figure 12.40d). Final output is written in the distributed file system, one file per reducer, where each file will contain roughly the same number of words.

Although a two-stage MapReduce processing structure may appear to be very restrictive, many interesting algorithms can be expressed quite concisely, especially if one decomposes complex algorithms into a sequence of MapReduce jobs. An illustrative example is Google's PageRank algorithm. It is important to realize that also many algorithms cannot be easily expressed as a single MapReduce job. Consider what would happen if the task is to find mean value of large set of numerical values.

The Mapper would compute the mean of an arbitrary subset of values associated with the same key, and the Reducer would compute the mean of those (mean) values. As a concrete example, we know that

$$\text{Mean}(1; 2; 3; 4; 5) \neq \text{Mean}(\text{Mean}(1; 2); \text{Mean}(3; 4; 5))$$

and it means that simple implementation of MapReduce framework for this problem is not applicable.

The availability of cloud-based solutions has dramatically lowered the cost of storage and enables a new business models that support on-demand, pay-for-use, and scalable IT services over the Internet. Clouds provide services at different levels of resource use:

- *Software as a Service (SaaS)* model enables a software provider to license a software application to be used and purchased on demand. Application can be accessed through networks from various clients (Web browser, mobile phone, etc.) The application requires no client installation, just a browser or other client device and network connectivity.
- *Platform as a Service (PaaS)* model provides developers a service that can be used as a complete software development lifecycle management, from planning and design, through building application and deployment, to finally testing and maintenance.
- *Infrastructure as a Service (IaaS)* focuses on enabling technologies. The cloud consumers directly use infrastructure components (storage, network, firewalls, etc.) that are enabled by the cloud provider.

The trend of making everything-as-a-service has affected many disciplines including machine learning and data mining. *Machine Learning as a Service (MLaaS)* is an integrate framework of automated and semiautomated cloud platforms that cover most infrastructure required during data-mining process including data preprocessing, model training, model evaluation, and application of the machine-learning model in a real-world environments. Obtained prediction results can be bridged with other IT infrastructure in the company. Information governance, security, and system management support each processing phase to ensure regulation and policies for all data used or derived in data-mining process. Compliance is tracked to ensure controls about delivering expected results.

Amazon Machine Learning services, Microsoft Azure Machine Learning, and Google Cloud AI are three initial and leading frameworks that are offering cloud MLaaS services that allow for fast model developing and deployment with little data-mining experience. Tools for classification, regression, clustering, anomaly detection, ranking, and recommender systems are available in all three frameworks with different levels of automation and graphical interfaces. These platforms include additionally APIs, which could perform text, speech and image recognition, topics

extraction, voice verification, sentiment analysis, etc. Besides fully integrated plat-forms, you can use high-level APIs for performing specific machine-learning tasks in the cloud environment. These are the services with easy trained models that you can feed your data into, and get results. APIs do not require machine-learning exper-tise at all. Currently, the APIs from main vendors include applications such as text recognition, text translation, textual analysis, sentiment analysis, facial recognition, image annotation, and video recognition.

12.8 REINFORCEMENT LEARNING

Reinforcement learning (RL) is a class of machine-learning algorithms that aims at allowing an *agent* to learn how to behave in an *environment*. RL agent has the ability to act, and this interaction with the environment enables agent to learn by *trial and error*. Each action in the environment influences the agent's future state. Success of an agent's action is measured by a scalar reward signal, which represents the only feedback from the environment. In general, the main goal of the RL process is selec-tion of appropriate actions that will maximize reward in a multiple agent–environment interaction. The agent has to explore the environment by performing actions and ana-lyzing the consequences. The effects on the environments are obtained through rewards for each agent's action, and that feedback is a core of a machine-learning process. The *goal* of the agent is to perform set of *actions* that will maximize the reward signal in the long run.

Model-free RL algorithms do not rely on the availability of a perfect model. Instead, they rely on *interaction* with the environment and learning step by step through these interactions. Because the model is unknown, the learner has to try out different actions to see their results and to evaluate potential rewards. Each step in the agent–environment interaction generates a *learning sample*. These samples are used to bring some value in accordance to the immediate reward and also to estimate the value of the next state.

RL may be formalized as a framework defining the interaction between a learning agent and its environment in terms of (1) states, (2) actions, and (3) rewards. The main components and their interactions are given in Figure 12.41. The framework is a
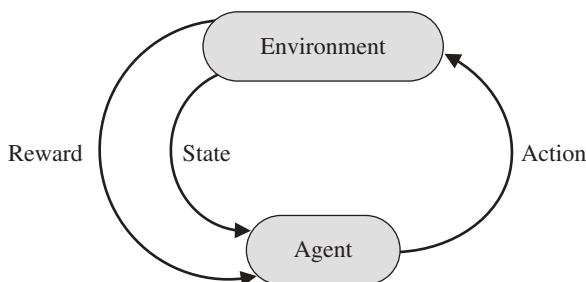
Figure 12.41. Main components in the reinforcement learning process.

simplified representation of essential features and main dynamics of the RL process. The agent, to be involved in the learning process, must have a single goal or alternative goals relating to the state of the environment, and it must be able to learn from its own experience. Beyond the agent and the environment, additional three elements participate in an RL activities: *a policy*, *a reward* function, and *a value function*.

A policy defines the learning agent's way of behaving at a given time. It is a mapping function from perceived states of the environment to actions to be taken when the agent is in those states. A *reward function* highlights the main goal in an RL problem; it specifies a reward for an agent being in a given state or doing some specific action in a state. The environment sends to the RL agent a single number as a reward for every step in each learning sample. While the reward signal indicates what is good in an immediate sense, a *value function* specifies what is good, as a solution in the long-run learning process. The agent's main objective is to maximize the total reward it receives over multiple learning samples. One of the core RL challenges, which is not essential in other kinds of machine learning, is the trade-off between exploration and exploitation processes. The agent has to *exploit* what it already knows in order to obtain the highest reward, but it also has to *explore* unknown paths using new samples in order to make better action selections in the future.

One of the basic and most popular RL methods is the *Q*-learning algorithm. The basic idea in *Q*-learning is to incrementally estimate *Q*-values for all new actions, based on feedback expressed through rewards, and the previous agent's *Q*-value function. *Q*-learning is exploration insensitive. It means that the learning process will converge to the optimal policy regardless of the exploration policy being followed. The only required assumption is that each state–action pair is visited an infinite or in practice enough large number of times.

Basic ideas and components of a *Q*-learning algorithm will be illustrated through a relatively simple example. Suppose that there is an environment that represents a building with five rooms connected by doors. The layout of the framework is given in Figure 12.42a, where the rooms are numerated 0 to 4, while number 5 is representing the outside space. Notice that the doors from rooms 1 and 4 lead into the "room" 5 (outside), while the other doors are between regular rooms. We may call each room, including outside, a "state" of the environment, and the agent's movement from one room to another should be called an "action." In the graph formalization on Figure 12.42b, a "state" is depicted as a node in a graph, while "actions" are represented by the arrows between states. The structure of the building is represented with the corresponding graph where actions are represented between two states only if there are direct doors between corresponding rooms.

For illustration of a *Q*-learning process, an agent may be initially located in any room, and it should learn how to move from that room to reach the goal—outside the building space 5. To set this "room" 5 as a goal, it is necessary to associate a reward value to each door or each action in the graph. Assume that the doors from rooms 1 and 4 leading immediately to the goal state 5 have an instant highest reward of 100. It includes "room" 5 feedback loop to itself with the same reward of 100. In *Q*-learning,

(a)   (b)
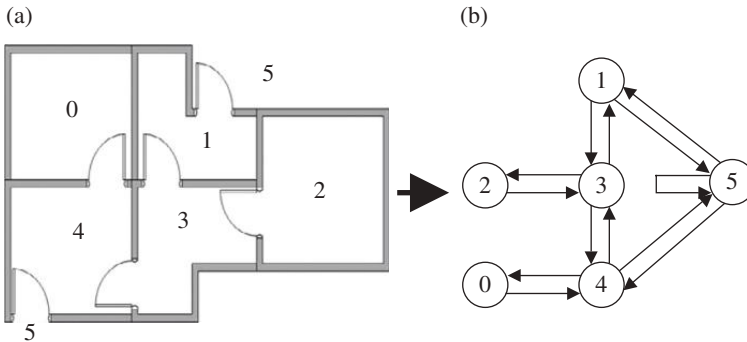
Figure 12.42. A framework for illustration of *Q*-learning process. (a) Five-room building. (b) Graph representation of the building. From: http://mnemstudio.org/path-finding-q-learning-tutorial.htm

(a)   (b)

$$R = \begin{array}{c} \\ \text{State} \\ \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{bmatrix} -1 & -1 & -1 & -1 & 0 & -1 \\ -1 & -1 & -1 & 0 & -1 & 100 \\ -1 & -1 & -1 & 0 & -1 & -1 \\ -1 & 0 & 0 & -1 & 0 & -1 \\ 0 & -1 & -1 & 0 & -1 & 100 \\ -1 & 0 & -1 & -1 & 0 & 100 \end{bmatrix}$$
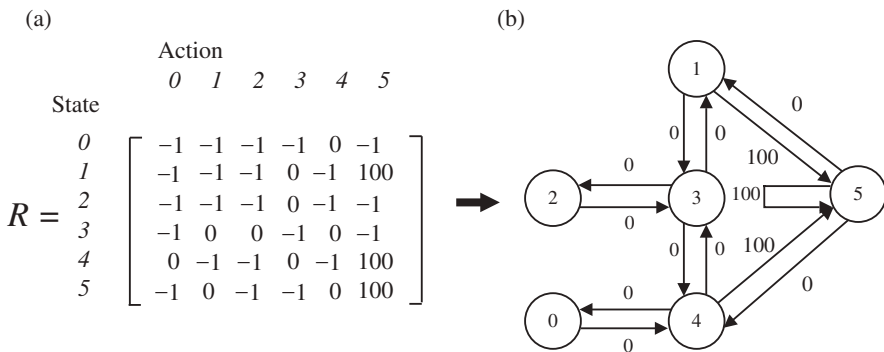
Figure 12.43. Reward function *R* for the five-room building environment. (a) Tabular representation of *R* matrix. (b) Graph representation of *R* matrix.

the goal is to reach the state with the highest reward so that if the agent arrives at the goal, it should remain there forever.

In the reward matrix *R* in Figure 12.43a, the value (-1) shows that there is no link (door) between corresponding rooms (states). The reward value of 0 is assigned for all existing doors, which are not leading to the goal, outside state 5. The reward matrix may be translated into the graph in Figure 12.43b where only existing links/actions with corresponding awards 0 or 100 are specified.

The agent starts activities of learning, and at the beginning the assumption is that the agent knows nothing. Therefore, the matrix *Q* is initialized to zero. The main transition rule of *Q*-learning is represented by two simple formulas:

$$Q_{\text{LEARNED}}(\text{state}, \text{action}) = R(\text{state}, \text{action}) + \gamma^{*}\text{Max}[Q_{\text{OLD}}(\text{next state}, \text{all actions})]$$

$$Q_{\text{NEW}}(\text{state}, \text{action}) = (1 - \alpha) Q_{\text{OLD}}(\text{state}, \text{action}) + \alpha Q_{\text{LEARNED}}(\text{state}, \text{action})$$

where α is the learning rate $(0 \leq \alpha \leq 1)$ and γ is the discount factor $(0 \leq \gamma \leq 1)$.

These formulas connect calculation of a new Q-values in the matrix based on initial R matrix of rewards and current exploration of the agent in the current state and with the current action. The agent will learn without teacher, based on experiences with every new action. The exploration process is analyzed through episodes, where each episode consists of an agent moving from the initial state to the goal state, and it is equivalent to one training session. Each time the agent arrives at the goal state, the learning process goes to the next episode. More training results applied in the Q methodology gives a more optimized matrix Q.

Let us start with the simple learning process assuming that the learning parameter α, discount parameter $\gamma = 0.8$, and the initial state is randomly selected as room 1. In this case $Q_{NEW} = Q_{LEARNED}$. There are two possible actions for the current state 1: (1) go to state 3, or (2) go to state 5. The agent does not have any initial knowledge, and it selects randomly to go to room 5 as the action. In that case correction in the Q matrix will be calculated as follows:

$$Q_{NEW}(1,5) = R(1,5) + 0.8^* \text{Max}[Q(5,1), Q(5,4), Q(5,5)] = 100 + 0.8^*0 = 100$$

Initial $R(1,5)$ value is taken from R matrix, while all alternative Q-values for actions from state 5 toward states 1, 4, and 5 are equal 0 in the current Q matrix. Because the agent arrived in the goal state 5, this episode is finished, and the optimized Q matrix has only one new value $Q(1,5) = 100$. All the other values stay the same with values 0.

For the next episode, the agent starts with a randomly chosen initial state 3. Based on the matrix R, there are three possible actions: go to state 1, 2, or 4. Without additional knowledge, the agent uses a random selection of state 1 as the action. The correction in the Q matrix will be calculated as

$$Q_{NEW}(3,1) = R(3,1) + 0.8^* \text{Max}[Q(1,3), Q(1,5)] = 0 + 0.8^* \text{Max}(0, 100) = 80$$

This time the values for $R(3,1)$ and $Q(1, 3)$ are both 0, while $Q(1,5) = 100$ is already learned value from the previous episode. The matrix Q becomes the matrix with two corrected values, as it is given on the Figure 12.44a. With each new episode new values or modified previous values will be introduced in the Q matrix. With larger number of learning episodes and often changes in the matrix Q, the values in the matrix may become very high. The matrix Q can be normalized and converted to percentages by dividing all nonzero entries by the highest number in the matrix. If the reader practices calculation of Q matrix after several randomly selected episodes, the obtained result will be similar to the normalized matrix on Figure 12.44b.

Once the matrix Q gets close enough to a state of convergence, the agent has learned the most optimal paths to the goal state for the given environment. The state of convergence may be recognized when the matrix Q is not changing significantly with any additional episodes. The agent learns enough, and it behaves in the environment by tracing the best sequences of states; it is essentially a simple process, which is following the actions with the highest Q-values at each state.

RL approach become widely well-known with *AlphaGo* system. The ancient Chinese game of Go has challenged AI researchers for many decades. Methods that

(a)

$$Q = \begin{matrix} & 0 & 1 & 2 & 3 & 4 & 5 \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{matrix} \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 100 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 80 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

(b)

$$\rightarrow Q = \begin{matrix} & 0 & 1 & 2 & 3 & 4 & 5 \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{matrix} \begin{bmatrix} 0 & 0 & 0 & 0 & 80 & 0 \\ 0 & 0 & 0 & 64 & 0 & 100 \\ 0 & 0 & 0 & 64 & 0 & 0 \\ 0 & 80 & 51 & 0 & 80 & 0 \\ 64 & 0 & 0 & 64 & 0 & 100 \\ 0 & 80 & 0 & 0 & 80 & 100 \end{bmatrix}$$

Figure 12.44. Dynamic changes of Q matrix with each new episode. (a) Q matrix after two episodes. (b) Q matrix already learned the problem.

achieve human-level skills in other games, such as poker and chess, have not been successful in producing strong Go programs. Recently, a Google DeepMind team developed a program called *AlphaGo* that broke this barrier in the game of Go learning by combining deep learning neural networks and RL. The system beats the best human players in the world in high-profile matches. But the RL approach shows the ability of building more advanced learning systems, where application's areas are much wider than games. The promising and highly publicized application of RL technology is humanlike behavior of automatic agents in self-driving cars. Today's driverless vehicles are becoming more and more sophisticated; still they are often falter in complex situations that still involve interacting with human drivers.

A number of industrial-robot makers are testing the RL approach as a way to train their machines to perform new tasks without manual programming. Applications of RL in high-dimensional control problems, including robotics, have been the subject of research in academia and industry, while startups are beginning to use RL to build new generation of products for industrial robotics. Google worked with DeepMind group in using RL to make its data centers more energy efficient. It is difficult to figure out how all the elements in a data center will affect energy usage, but an RL algorithm can learn from collated data and experiment in simulation to suggest, say, how and when to operate the cooling systems.

In the financial sector there are several solution leveraging RL for evaluating trading strategies. It is turning out that RL represents a robust methodology for training systems to optimize financial objectives. It has primary applications in stock market trading, where Q-learning algorithm is able to learn an optimal trading strategy with one simple instruction: maximize the value of user's portfolio. Also, personalizing Web services such as the delivery of news articles or advertisements is one approach to increasing users' satisfaction with a Web site or to increase the yield of a marketing campaign. This is a natural domain for RL. An RL system can improve a recommendation policy by making adjustments in response to user feedback. One way to obtain user feedback is by means of Web site satisfaction surveys, but for acquiring feedback in real time, it is common to monitor user clicks as indicators of interest in a link.

In spite of all initial successes with RL, there are several warnings from some experienced users and researchers that the approach, to be successful, requires a huge amount of data. While some solutions may be obtained with large number of simulations if available, the researchers are aware that RL becomes less effective when dealing with highly complicated problems with high-dimensional state and action spaces. Some recognize potential solutions in combining different machine-learning approaches such as integration of RL with deep learning techniques.

12.9   REVIEW QUESTIONS AND PROBLEMS

1. What are the benefits in modeling social networks with a graph structure? What kind of graphs you would you in this case?

2. For the given undirected graph *G*:
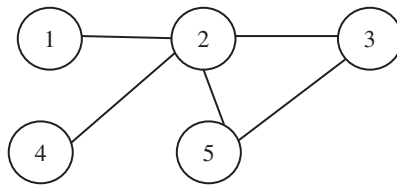
   (a) Compute degree and variability parameters of the graph.
   (b) Find adjacency matrix for the graph *G*.
   (c) Determine binary code(*G*) for the graph.
   (d) Find closeness parameter or each node of the graph.
   (e) What is the betweenness measure for the node #2?

3. For the graph given in Problem #2, find partial *betweenness centrality* using modified graph starting with node #5.

4. Give real-world examples for traditional analyses of temporal data (i.e. trends, cycles, seasonal patterns, outliers).

5. Given the temporal sequence $S = \{1\ 2\ 3\ 2\ 4\ 6\ 7\ 5\ 3\ 1\ 0\ 2\}$.

   (a) Find PAA for four sections of the sequence.
   (b) Determine SAX values for solution in (a) if (i) $\alpha = 3$ and (ii) $\alpha = 4$.
   (c) Find PAA for three sections of the sequence.
   (d) Determine SAX values for solution in (c) if (i) $\alpha = 3$ and (ii) $\alpha = 4$.

6. Given the sequence $S = \{A\ B\ C\ B\ A\ A\ B\ A\ B\ C\ B\ A\ B\ A\ B\ B\ C\ B\ A\ C\ C\}$.

   (a) Find the longest subsequence with frequency ≥ 3.
   (b) Construct finite state automaton (FSA) for the subsequence found in (a).

7. Find normalized contiguity matrix for the table of US cities:

Minneapolis	Chicago	New York
Nashville	Louisville	Charlotte

Make assumption that only neighboring cities (vertical and horizontal) in the table are close.

8. For the Bayesian network in Figure 12.38, determine:

(a) $P(C, R, W)$

(b) $P(\rceil C, \rceil S, W)$

9. Review the latest articles on privacy-preserving data mining that are available on the Internet. Discuss the trends in the field.

10. What are the largest sources of unintended personal data on the Internet? How to increase awareness of Web users of their personal data that are available on the Web for variety of data-mining activities?

11. Discuss an implementation of transparency and accountability mechanisms in a data-mining process. Illustrate your ideas with examples of real-world data-mining applications.

12. Give examples of data-mining applications where you would use distributed-data-mining approach. Explain the reasons.

13. Develop and explain all steps of MapReduce process for inverted file index. Inputs are text document. The output should be a list of words used in the documents, where each word is connected with the list of keys, representing each document containing the word.

14. Find the set of 2-shingles for the "documents" (or simple text) in (a) and (b):

(a) ABRACADABRA

(b) BRICABRAC

(c) How many 2-shingles two texts have in common?

15. For the following graph:

```
      C -- D -- E
     /           \
    A             B
     \           /
      F -- G – H
```
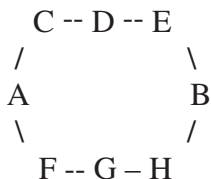
compute the betweenness of every edge.

16. The following is a description of the causal relationship between storms, the behavior of burglars and cats, and house alarms:

Stormy nights are rare. Burglary is also rare, and if it is a stormy night, burglars are likely to stay at home (burglars do not like going out in storms). Cats do not like storms either, and if there is a storm, they like to go inside. The alarm on your house is designed to be triggered if a burglar breaks into your house, but sometimes it can be set off by your cat coming into the house, and sometimes it might not be triggered even if a burglar breaks in (it could be faulty or the burglar might be very good).

Define the topology of a Bayesian network that encodes these causal relationships.

17. Consider the time series $(-3, -1, 0, 3, 5, 7, *)$. Here, a missing entry at the end is denoted by $*$.

    (a) What would be the estimated value of the missing entry using linear interpolation on a window of size 3?

    (b) What would be the estimated value if the fourth value in the time series is missing and calculation is based on the same window size?

12.10 REFERENCES FOR FURTHER STUDY

1. Aggarwal C. C., Yu P. S., *Privacy-Preserving Data Mining: Models and Algorithms*, Springer, Berlin, 2008.

   The book proposes a number of techniques to perform the data-mining tasks in a privacy-preserving way. These techniques generally fall into the following categories: data modification techniques, cryptographic methods and protocols for data sharing, statistical techniques for disclosure and inference control, query auditing methods, randomization, and perturbation-based techniques. This edited volume contains surveys by distinguished researchers in the privacy field. Each survey includes the key research content as well as future research directions. *Privacy-Preserving Data Mining: Models and Algorithms* is designed for researchers, professors, and advanced-level students in computer science and is also suitable for industry practitioners.

2. Mitsa T., *Temporal data Mining*, Chapmann & Hall / CRC Press, 2010.

   From basic data-mining concepts to state-of-the-art advances, *Temporal Data Mining* covers the theory of this subject as well as its application in a variety of fields. It discusses the incorporation of temporality in databases as well as temporal data representation, similarity computation, data classification, clustering, pattern discovery, and prediction. The book also explores the use of temporal data mining in medicine and biomedical informatics, business and industrial applications, Web usage mining, and spatiotemporal data mining. Along with various state-of-the-art

algorithms, each chapter includes detailed references and short descriptions of relevant algorithms and techniques described in other references. In the appendices, the author explains how data mining fits the overall goal of an organization and how these data can be interpreted for the purpose of characterizing a population. She also provides programs written in the Java language that implement some of the algorithms presented in the first chapter.

3. Zeitouni K., A Survey of Spatial Data Mining Methods: Databases and Statistics Point of View, *In "Data warehousing and web engineering"*, Becker S., editor, IRM Press, 2002.

   This chapter reviews the data-mining methods that are combined with geographic information systems (GIS) for carrying out spatial analysis of geographic data. We will first look at data-mining functions as applied to such data and then highlight their specificity compared with their application to classical data. We will go on to describe the research that is currently going on in this area, pointing out that there are two approaches: the first comes from learning on spatial databases, while the second is based on spatial statistics. We will conclude by discussing the main differences between these two approaches and the elements they have in common.

4. Chakrabarti D., Faloutsos C., Graph Mining: Laws, Generators, and Algorithms, *ACM Computing Surveys*, Vol. 38, March 2006, pp. 1–69.

   How does the Web look? How could we tell an abnormal social network from a normal one? These and similar questions are important in many fields where the data can intuitively be cast as a graph; examples range from computer networks to sociology to biology and many more. Indeed, any M: N relation in database terminology can be represented as a graph. A lot of these questions boil down to the following: "How can we generate synthetic but realistic graphs?" To answer this, we must first understand what patterns are common in real-world graphs and can thus be considered a mark of normality/realism. This survey gives an overview of the incredible variety of work that has been done on these problems. One of our main contributions is the integration of points of view from physics, mathematics, sociology, and computer science. Further, we briefly describe recent advances on some related and interesting graph problems.

5. Pearl J., *Causality: Models, Reasoning and Inference*, 2nd edition, Cambridge University Press, 2009.

   This book fulfills a long-standing need for a rigorous yet accessible treatise on the mathematics of causal inference. Judea Pearl has done a masterful job of describing the most important approaches and displaying their underlying logical unity. The book deserves to be read by all scientists who use non-experimental data to study causation and would serve well as a graduate or advanced undergraduate course text. The book should prove invaluable to researchers in artificial intelligence, statistics, economics, epidemiology, and philosophy and, indeed, all those interested in the fundamental notion of causality. It may well prove to be one of the most influential books of the next decade.

6. Gosavi A., *Simulation-Based Optimization: Parametric Optimization Techniques and Reinforcement Learning*, 2nd edition, Springer, New York, NY, 2014.

   The book introduces the evolving area of static and dynamic simulation-based optimization. Covered in detail are model-free optimization techniques—especially designed for those discrete-event stochastic systems that can be simulated but whose analytical models are difficult to find in closed mathematical forms. It includes an in-depth consideration of dynamic simulation optimization via temporal differences and reinforcement learning: Q-learning, SARSA, and R-SMART algorithms, and policy search, via API, Q-P-learning, actor-critics, and learning automata. A special examination is given to neural-network-based function approximation for reinforcement learning, semi-Markov decision processes (SMDPs), finite-horizon problems, two time scales, case studies for industrial tasks, computer codes (placed online), and convergence proofs, via Banach fixed point theory and ordinary differential equations. The book is themed around three areas in separate sets of chapters—static simulation optimization, reinforcement learning, and convergence analysis.

7. Talia D., *Data Analysis in the Cloud: Models, Techniques and Applications*, Elsevier Inc., 2015.

   The book introduces and discusses models, methods, techniques, and systems to analyze the large number of digital data sources available on the Internet using the computing and storage facilities of the cloud. Coverage includes scalable data-mining and knowledge-discovery techniques together with cloud computing concepts, models, and systems. Specific sections focus on map–reduce and NoSQL models. The book also includes techniques for conducting high-performance distributed analysis of large data on clouds. Finally, the book examines research trends such as big data pervasive computing, data-intensive exascale computing, and massive social network analysis.

13

GENETIC ALGORITHMS

Chapter Objectives

- Identify effective algorithms for approximate solutions of optimization problems described with large data sets.
- Compare basic principles and concepts of natural evolution and simulated evolution expressed through genetic algorithms.
- Describe the main steps of a genetic algorithm with illustrative examples.
- Explain standard and nonstandard genetic operators such as a mechanism for improving solutions.
- Discuss a schema concept with don't care values and its application to approximate optimization.
- Apply a genetic algorithm to the traveling salesman problem and optimization of classification rules as examples of hard optimizations.

There is a large class of interesting problems for which no reasonably fast algorithms have been developed. Many of these problems are optimization problems that arise frequently in applications. The fundamental approach to optimization is to formulate a single standard of measurement—a cost function—that summarizes the performance or value of a decision and iteratively improves this performance by selecting from among the available alternatives. Most classical methods of optimization generate a deterministic sequence of trial solutions based on the gradient or higher-order statistics of the cost function. In general, any abstract task to be accomplished can be thought of as solving a problem, which can be perceived as a search through a space of potential solutions. Since we are looking for "the best" solution, we can view this task as an optimization process. For small data spaces, classical exhaustive search methods usually suffice; for large spaces, special techniques must be employed. Under regular conditions, the techniques can be shown to generate sequences that asymptotically converge to optimal solutions, and in certain cases they converge exponentially fast. But the methods often fail to perform adequately when random perturbations are imposed on the function that is optimized. Further, locally optimal solutions often prove insufficient in real-world situations. Despite such problems, which we call *hard-optimization problems*, it is often possible to find an effective algorithm whose solution is approximately optimal. One of the approaches is based on genetic algorithms (GAs), which are developed on the principles of natural evolution.

Natural evolution is a population-based optimization process. Simulating this process on a computer results in stochastic-optimization techniques that can often outperform classical methods of optimization when applied to difficult real-world problems. The problems that the biological species have solved are typified by chaos, chance, temporality, and nonlinear interactivity. These are the characteristics of the problems that have proved to be especially intractable to classical methods of optimization. Therefore, the main avenue of research in simulated evolution is a *genetic algorithm*, which is a new iterative optimization method that emphasizes some facets of natural evolution. GAs approximate an optimal solution to the problem at hand; they are by nature stochastic algorithms whose search methods model some natural phenomena such as genetic inheritance and the Darwinian strife for survival.

13.1   FUNDAMENTALS OF GENETIC ALGORITHMS

GAs are derivative-free stochastic-optimization methods based loosely on the concepts of natural selection and evolutionary processes. They were first proposed and investigated by John Holland at the University of Michigan in 1975. The basic idea of GAs was revealed by a number of biologists when they used computers to perform simulations of natural genetic systems. In these systems, one or more chromosomes combine to form the total genetic prescription for the construction and operation of some organism. The chromosomes are composed of genes, which may take a number of values called allele values. The position of a gene (its locus) is identified separately

from the gene's function. Thus, we can talk of a particular gene, e.g., an animal's eye-color gene with its locus at position 10 and its allele value as blue eyes.

Before going into details of the applications of GAs in the following sections, let us understand its basic principles and components. GAs encode each point in a parameter or solution space into a binary-bit string called a chromosome. These points in an n-dimensional space do not represent samples in the terms that we defined them at the beginning of this book. While samples in other data-mining methodologies are data sets given in advance for training and testing, sets of n-dimensional points in GAs are a part of a GA, and they are produced iteratively in the optimization process. Each point or binary string represents a potential solution to the problem that is to be solved. In GAs, the decision variables of an optimization problem are coded by a structure of one or more strings, which are analogous to chromosomes in natural genetic systems. The coding strings are composed of features that are analogous to genes. Features are located in different positions in the string, where each feature has its own position (locus) and a definite allele value, which complies with the proposed coding method. The string structures in the chromosomes go through different operations similar to the natural evolution process to produce better alternative solutions. The quality of new chromosomes is estimated based on the "fitness" value, which can be considered as the objective function for the optimization problem. The basic relations between concepts in natural evolution and GAs are given in Table 13.1. Instead of single a point, GAs usually keep a set of points as a population, which is then evolved repeatedly toward a better overall fitness value. In each generation, the GA constructs a new population using genetic operators such as crossover and mutation. Members with higher fitness values are more likely to survive and participate in mating or *crossover operations*.

As a general-purpose optimization tool, GAs are moving out of academia and finding significant applications in many other venues. Typical situations where GAs are particularly useful are in difficult optimization cases for which analytical methods do not work well. GAs have been quite successfully applied to optimization problems like wire routing, scheduling, adaptive control, game playing, transportation problems, traveling salesman problems (TSP), database query optimization, machine learning, etc. During the last decades, the significance of optimization has grown even further because many important large-scale combinatorial-optimization problems and

TABLE 13.1. Basic Concepts in Genetic Algorithms

Concept in Natural Evolution	Concept in Genetic Algorithms
Chromosome	String
Gene	Features in the string
Locus	Position in the string
Allele	Position value (usually 0 or 1)
Genotype	String structure
Phenotype	Set of characteristics (features)

highly constrained engineering problems can only be solved approximately. GAs aim at such complex problems. They belong to the class of probabilistic algorithms, yet they are very different from random algorithms as they combine elements of directed and stochastic search. Another important property of genetics-based search methods is that they maintain a population of potential solutions while all other methods process a single point of the search space. Because of these characteristics, GAs are more robust than existing directed-search methods.

Gas are popular because they do not depend on functional derivatives and they have the following characteristics:

1. GAs are parallel-search procedures that can be implemented on parallel-processing machines for massively speeding up their operations.
2. GAs are applicable to both continuous- and discrete-optimization problems.
3. GAs are stochastic and less likely to get trapped in local minima, which inevitably are present in any practical optimization application.
4. GAs' flexibility facilitates both structure and parameter identification in complex models.

The GA theory provides some explanation why, for a given problem formulation, we may obtain convergence to the sought optimal point. Unfortunately, practical applications do not always follow the theory, the main reason being:

1. The coding of the problem often moves the GA to operate in a different space than that of the problem itself.
2. There are practical limits on the hypothetically unlimited number of iterations (generations in the GA).
3. There is a limit on the hypothetically unlimited population size.

One of the implications of these observations is the inability of GAs, under certain conditions, to find the optimal solution or even an approximation to the optimal solution; such failures are usually caused by premature convergence to a local optimum. Do not forget that this problem is common not only for the other optimization algorithms but also for the other data-mining techniques.

13.2   OPTIMIZATION USING GENETIC ALGORITHMS

Let us note first that without any loss of generality, we can assume that all optimization problems can be analyzed as maximization problems only. If the optimization problem is to minimize a function $f(x)$, this is equivalent to maximizing a function $g(x) = -f(x)$. Moreover, we may assume that the objective function $f(x)$ takes positive values in its domain. Otherwise, we can translate the function for some positive constant C so that it will be always positive; i.e.,

$$\max f^*(x) = \max \{f(x) + C\}$$

If each variable x_i, with real values, is coded as a binary string of length m, then the relation between the initial value and the coded information is

$$x_i = a + \text{decimal (binary-string}_i) \left\{ \frac{[b-a]}{[2^m - 1]} \right\}$$

where the variable x_i can take the values from a domain $D_i = [a, b]$ and m is the smallest integer such that the binary code has the required precision. For example, the value for variable x given on the domain $[10, 20]$ is a binary-coded string with the length equal to 3 and the code 100. While the range of codes is between 000 and 111, the question is: what is the real value of the coded variable x? For this example $m = 3$ and the corresponding precision is

$$\frac{[b-a]}{[2^m - 1]} = \frac{(20 - 10)}{(2^3 - 1)} = \frac{10}{7} = 1.42$$

and that is the difference between two successive x_i values that could be tested as candidates for extreme. Finally, the attribute with the code 100 has a decimal value:

$$x = 10 + \text{decimal}(100) \cdot 1.42 = 10 + 4 \cdot 1.42 = 15.68$$

Each chromosome as a potential solution is represented by a concatenation of binary codes for all features in the problem to be optimized. Its total length m is a sum of the features' code lengths m_i:

$$m = \sum_{i=1}^{k} m_i$$

where k is the number of features or input variables for the problem at hand. When we introduce these basic principles of a code construction, it is possible to explain the main steps of a GA.

13.2.1 Encoding Schemes and Initialization

A GA starts with designing a representation of a solution for the given problem. A solution here means any value that is a candidate for a correct solution that can be evaluated. For example, suppose we want to maximize function $y = 5 - (x - 1)^2$. Then, $x = 2$ is a potential solution, $x = 2.5$ is another solution, and $x = 3$ is the correct solution of the problem that maximizes y. The representation of each solution for a GA is up to the designer. It depends on what each solution looks like and which solution form will be convenient for applying a GA. The most common representation of a solution is as a

string of characters, i.e., a string of codes for feature representation, where the characters belong to a fixed alphabet. The larger the alphabet, the more the information that can be represented by each character in the string. Therefore, fewer elements in a string are necessary to encode specific amounts of information. However, in most real-world applications, GAs usually use a binary-coding schema.

The encoding process transforms points in a feature space into bit string representation. For instance, a point (11, 6, 9) in a three-dimensional feature space, with ranges [0, 15] for each dimension, can be represented as a concatenated binary string:

$$(11,6,9) \Rightarrow (101101101001)$$

in which each feature's decimal value is encoded as a gene composed of four bits using a binary coding.

Other encoding schemes, such as Gray coding, can also be used, and, when necessary, arrangements can be made for encoding negative, floating-point, or discrete-value numbers. Encoding schemes provide a way of translating problem-specific knowledge directly into the GA framework. This process plays a key role in determining GAs' performances. Moreover, genetic operators can and should be designed along with the encoding scheme used for a specific application.

A set of all features values encoded into a bit string represents one chromosome. In GAs we are manipulating not a single chromosome but a set of chromosomes called a population. To initialize a population, we can simply set some *pop-size* number of chromosomes randomly. The size of the population is also one of the most important choices faced by any user of GAs and may be critical in many applications: will we reach the approximate solution at all, and if yes, how fast? If the population size is too small, the GA may converge too quickly and maybe to a solution that is only the local optimum; if it is too large, the GA may waste computational resources and the waiting time for an improvement might be too long.

13.2.2   Fitness Evaluation

The next step, after creating a population, is to calculate the fitness value of each member in the population because each chromosome is a candidate for an optimal solution. For a maximization problem, the fitness value f_i of the ith member is usually the objective function evaluated at this member (or the point in parameter space). The fitness of a solution is a measure that can be used to compare solutions to determine which is better. The fitness values may be determined from complex analytical formulas and simulation models or by referring to observations from experiments or real-life problem settings. GAs will work correctly if fitness values are determined appropriately keeping in mind that a selection of the objective function is highly subjective and problem dependent.

We usually need fitness values that are positive, so some kind of scaling and/or translation of data may become necessary if the objective function is not strictly positive. Another approach is to use the rankings of members in a population as their

fitness values. The advantage of this approach is that the objective function does not need to be accurate, as long as it can provide the correct ranking information.

13.2.3   Selection

In this phase, we have to create a new population from the current generation. The selection operation determines which parent chromosomes participate in producing offspring for the next generation. Usually, members are selected for mating with a selection probability proportional to their fitness values. The most common way to implement this method is to set the selection probability p equal to

$$p_i = \frac{f_i}{\sum_{k=1}^{n} f_k}$$

where n is the population size and f_i is a fitness value for the ith chromosome. The effect of this selection method is to allow members with above-average values to reproduce and replace members with below-average fitness values.

For the selection process (selection of a new population with respect to the probability distribution based on fitness values), a roulette wheel with slots sized according to fitness for each chromosome is used. We construct such a roulette wheel as follows:

1. Calculate the fitness value $f(v_i)$ for each chromosome v_i.
2. Find the total fitness of the population:

$$F = \sum_{i=1}^{\text{pop-size}} f(v_i)$$

3. Calculate the probability of a selection p_i for each chromosome v_i:

$$p_i = \frac{f(v_i)}{F}$$

4. Calculate a cumulative probability q_i after each chromosome v_i is included:

$$q_i = \sum_{j=1}^{i} p_i$$

where q increases from 0 to maximum 1. Value 1 shows that all chromosomes from the population are included into a cumulative probability.

The selection process is based on spinning the roulette wheel pop-size times. Each time we select a single chromosome for a new population. An implementation could repeat steps 1 and 2 *pop-size* times:

1. Generate a random number r from the range [0, 1].
2. If $r < q_1$, then select the first chromosome v_1; otherwise select the ith chromosome v_i such that $q_{i-1} < r \le q_i$.

Obviously, some chromosomes would be selected more than once. That is in accordance with the theory. GA performs a multidirectional search by maintaining a population of potential solutions and encourages good solutions. The population undergoes a simulated evolution—in each generation the relatively "good" solutions reproduce, while the relatively "bad" solutions die. To distinguish between different solutions, we use an objective or evaluation function, which plays the role of an environment.

13.2.4 Crossover

The strength of GAs arises from the structured information exchange of crossover combinations of highly fit individuals. So what we need is a crossover-like operator that would exploit important similarities between chromosomes. The probability of crossover PC is the parameter that will define the expected number of chromosomes—PC · *pop-size*—which undergo the crossover operation. We define the chromosomes for crossover in a current population using the following iterative procedure. Steps 1 and 2 have to be repeated for all chromosomes:

1. Generate a random number r from the range [0, 1].
2. If $r <$ PC, select the given chromosome for crossover.

If PC is set to 1, all chromosomes in the population will be included into the crossover operation; if PC = 0.5, only half of the population will perform crossover, and the other half will be included into a new population directly without changes.

To exploit the potential of the current gene pool, we use crossover operators to generate new chromosomes that will retain the good features from the previous generation. Crossover is usually applied to selected pairs of parents.

One-point crossover is the most basic crossover operator, where a crossover point on the genetic code is selected at random and two parent chromosomes are interchanged at this point. In *two-point crossover*, two points are selected, and a part of chromosome string between these two points is then swapped to generate two children of the new generation. Examples of one- and two-point crossover are shown in Figure 13.1.

We can define an *n*-point crossover similarly, where the parts of strings between points 1 and 2, 3 and 4, and finally $n - 1$ and n are swapped. The effect of crossover is similar to that of mating in the natural evolutionary process in which parents pass segments of their own chromosomes on to their children. Therefore, some children are able to outperform their parents if they get "good" genes or genetic traits from their parents.

13.2.5 Mutation

Crossover exploits existing gene potentials, but if the population does not contain all the encoded information needed to solve a particular problem, no amount of gene mixing can produce a satisfactory solution. For this reason, a mutation operator capable of

(a)

Selected point for one-point crossover (after the fifth position in the string)

```
⇓
┌─────────────────┐                      ┌─────────────────┐
│  1 0 0 1 1 1 1 0 │                      │  1 0 0 1 0 0 1 0 │
│                  │        ⇒             │                  │
│  1 0 1 1 0 0 1 0 │                      │  1 0 1 1 1 1 1 0 │
└─────────────────┘                      └─────────────────┘
```

(b)

Selected points for two-point crossover (after the second and fifth positions in the strings)

```
⇓     ⇓
┌─────────────────┐                      ┌─────────────────┐
│  1 0 0 1 1 1 1 0 │                      │  1 0 1 1 0 1 1 0 │
│                  │        ⇒             │                  │
│  1 0 1 1 0 0 1 0 │                      │  1 0 0 1 1 0 1 0 │
└─────────────────┘                      └─────────────────┘
```
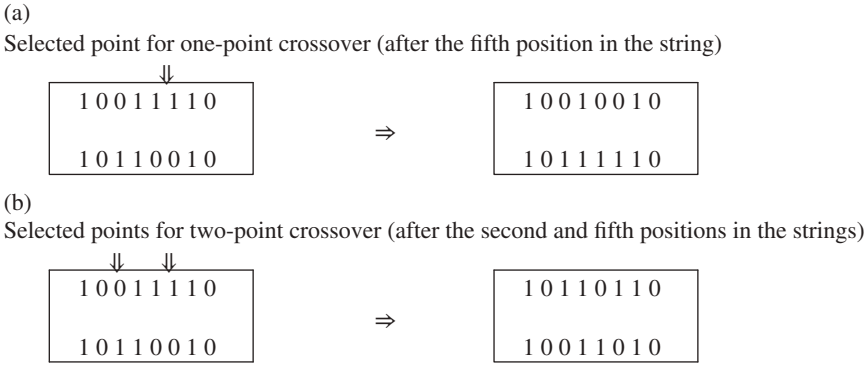
Figure 13.1. Crossover operators. (a) One-point crossover. (b)Two-point crossover.

Randomly selected mutated bit   Chromosome after mutation
(in the sixth position)

```
         ⇓                                            ⇓
┌─────────────────┐                      ┌─────────────────┐
│ 1 0 0 1 1 1 1 0 │        ⇒             │ 1 0 0 1 1 0 1 0 │
└─────────────────┘                      └─────────────────┘
```
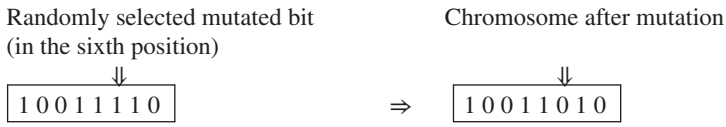
Figure 13.2. Mutation operator.

spontaneously generating new chromosomes is included. The most common way of implementing mutation is to flip a bit with a probability equal to a very low given mutation rate (MR). A mutation operator can prevent any single bit from converging to a value through the entire population, and, more importantly, it can prevent the population from converging and stagnating at any local optima. The MR is usually kept low so good chromosomes obtained from crossover are not lost. If the MR is high (for example, above 0.1), GA performance will approach that of a primitive random search. Figure 13.2 provides an example of mutation.

In the natural evolutionary process, selection, crossover, and mutation all occur simultaneously to generate offspring. Here we split them into consecutive phases to facilitate implementation of and experimentation with GAs. Note that this section only gives a general description of the basics of GAs. Detailed implementations of GAs vary considerably, but the main phases and the iterative process remain.

At the end of this section, we can summarize that the major components of GAs include encoding schemas, fitness evaluation, parent selection, and application of crossover operators and mutation operators. These phases are performed iteratively, as represented in Figure 13.3.

It is relatively easy to keep track of the best individual chromosomes in the evolution process. It is customary in GA implementations to store "the best ever" individual at a separate location. In that way, the algorithm would report the best value found during the whole process, just in the final population.
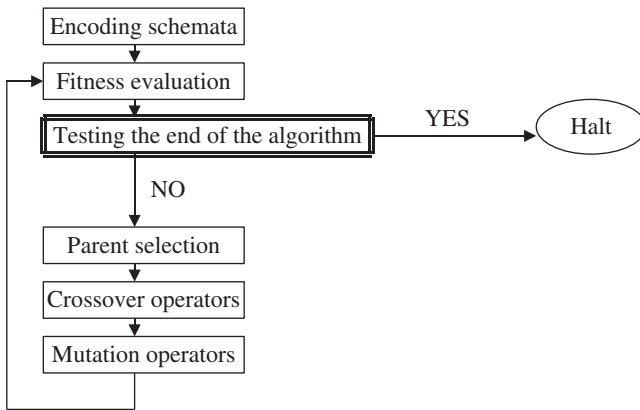
<u>Figure 13.3.</u> Major phases of a genetic algorithm.

Optimization under constraints is also the class of problems for which GAs are an appropriate solution. The constraint-handling techniques for GAs can be grouped into different categories. One typical way of dealing with GA candidates that violate the constraints is to generate potential solutions without considering the constraints and then to penalize them by decreasing the "goodness" of the evaluation function. In other words, a constrained problem is transformed to an unconstrained one by associating a penalty with all constraint violations. These penalties are included in the function evaluation, and there are different kinds of implementations. Some penalty functions assign a constant as a penalty measure. Other penalty functions depend on the degree of violation: the larger the violation, the greater the penalty. The growth of the penalty function can be logarithmic, linear, quadratic, exponential, etc., depending upon the size of the violation. Several implementations of GAs' optimization under constraints are given in the texts recommended for further study (Section 13.9).

13.3   A SIMPLE ILLUSTRATION OF A GENETIC ALGORITHM

To apply a GA for a particular problem, we have to define or to select the following five components:

1. A genetic representation or encoding schema for potential solutions to the problem.
2. A way to create an initial population of potential solutions.
3. An evaluation function that plays the role of the environment, rating solutions in terms of their "fitness."
4. Genetic operators that alter the composition of offspring.
5. Values for the various parameters that the GA uses (population size, rate of applied operators, etc.).

We discuss the main features of GAs by presenting a simple example. Suppose that the problem is the optimization of a simple function of one variable. The function is defined as

$$f(x) = x^2$$

The task is to find x from the range [0, 31] that maximizes the function $f(x)$. We selected this problem because it is relatively easy to analyze optimization of the function $f(x)$ analytically, to compare the results of the analytic optimization with a GA, and to find the approximate optimal solution.

13.3.1 Representation

The first step in the GA is to represent the solution alternative (a value for the input feature) in a coded-string format. Typically, the string is a series of features with their values; each feature's value can be coded with one from a set of discrete values called allele set. The allele set is defined according to the needs of the problem, and finding the appropriate coding method is a part of the art of using GAs. The coding method must be minimal but completely expressive. We will use a binary vector as a chromosome to represent real values of the single variable x. The length of the vector depends on the required precision, which, in this example, is selected as 1. Therefore, we need a minimum five-bit code (string) to accommodate the range with required precision:

$$\frac{(b-a)}{(2^m - 1)} \leq \text{Required precision}$$

$$\frac{(31-0)}{(2^m - 1)} \leq 1$$

$$2^m \geq 32$$

$$m \geq 5$$

For this example, the mapping from a real number to a binary code is defined by the relation (because $a = 0$)

$$\text{Code} = \text{binary}\left(x_{\text{decimal}}\right)$$

Opposite mapping from the binary code to the real value of the argument is also unique:

$$x = \text{decimal}\left(\text{Code}_{\text{binary}}\right)$$

and it will be used only for checking the intermediate results of optimization. For example, if we want to transform the value $x = 11$ into a binary string, the corresponding code will be 01011. On the other hand, code 11001 represents decimal value $x = 25$.

13.3.2   Initial Population

The initialization process is very simple: we randomly create a population of chromosomes (binary codes) with the given length. Suppose that we decide that the parameter for the number of strings in the population is equal to four. Then one possible randomly selected population of chromosomes is

$$CR_1 = 01101$$
$$CR_2 = 11000$$
$$CR_3 = 01000$$
$$CR_4 = 10011$$

13.3.3   Evaluation

The evaluation function for binary vectors representing chromosomes is equivalent to the initial function $f(x)$ where the given chromosome represents the binary code for the real value x. As noted earlier, the evaluation function plays the role of the environment, rating potential solutions in terms of their fitness. For our example, four chromosomes CR_1 to CR_4 correspond to values for input variable x:

$$x_1(CR_1) = 13$$
$$x_2(CR_2) = 24$$
$$x_3(CR_3) = 8$$
$$x_4(CR_4) = 19.$$

Consequently, the evaluation function would rate them as follows:

$$f(x_1) = 169$$
$$f(x_2) = 576$$
$$f(x_3) = 64$$
$$f(x_4) = 361$$

The results of evaluating the chromosomes initially generated may be given in a tabular form, and they are represented in Table 13.2. The expected reproduction column shows "the evaluated quality" of chromosomes in the initial population. Chromosomes CR_2 and CR_4 are more likely to be reproduced in the next generation than CR_1 and CR_3.

13.3.4 Alternation

In the alternation phase, the new population is selected based on the population evaluated in the previous iteration. Clearly, the chromosome CR_4 in our example is the best of the four chromosomes, since its evaluation returns the highest value. In the alternation phase, an individual may be selected depending on its objective-function value or fitness value. For maximization problems, the higher the individual's fitness, the more probable that it can be selected for the next generation. There are different schemes that can be used in the selection process. In the simple GA we proposed earlier, the roulette wheel selection technique, an individual is selected randomly depending on a computed probability of selection for each individual. The probability of selection is computed by dividing the individual's fitness value by the sum of fitness values of the corresponding population, and these values are represented in column 5 in Table 13.2.

In the next step we design the roulette wheel, which is, for our problem, graphically represented in Figure 13.4.

TABLE 13.2. Evaluation of the Initial Population

CR_i	Code	x	$f(x)$	$f(x)/\sum f(x)$	Expected Reproduction: $f(x)/f_{av}$
1	01101	13	169	0.14	0.58
2	11000	24	576	0.49	1.97
3	01000	8	64	0.06	0.22
4	10011	19	361	0.31	1.23
$\sum$			1170	1.00	4.00
Average			293	0.25	1.00
Max			576	0.49	1.97
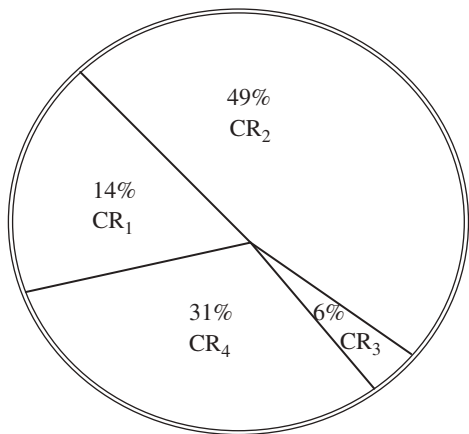
Figure 13.4. Roulette wheel for selection of the next population.

Using the roulette wheel, we can select chromosomes for the next population. Suppose that the randomly selected chromosomes for the next generation are CR_1, CR_2, CR_2, CR_4 (the selection is in accordance with the expected reproduction—column 6 in Table 13.2). In the next step, these four chromosomes undergo the genetic operations: crossover and mutation.

13.3.5 Genetic Operators

Crossover is not necessarily applied to all pairs of selected individuals. A choice is made depending on a specified probability called crossover probability (PC), which is typically between 0.5 and 1. If crossover is not applied (PC = 0), the offspring are simply a duplication of the parents. For the process of crossover, it is necessary to determine the percentage of the population that will perform the crossover. For our particular problem, we use the following parameters of the GA:

1. Population size, *pop-size* = 4 (the parameter was already used).
2. Probability of crossover, PC = 1.
3. Probability of mutation, PM = 0.001 (the parameter will be used in a mutation operation).

A value of 1 for the probability of crossover translates into a 100% crossover—all chromosomes will be included in the crossover operation.

The second set of parameters in this phase of a GA is the random selection of parents for crossover and positions in the strings where the crossover will be performed. Suppose that these are randomly selected pairs, CR_1–CR_2 and CR_2–CR_4, and crossover is after the third position in the strings for both pairs. Then the selected strings

First pair:

$$CR_1 = 01101$$

$$CR_2 = 11000$$

⇑

Second pair:

$$CR_2 = 11000$$

$$CR_4 = 10011$$

⇑

will become, after crossover, a new population:

$$CR_1' = 01100$$
$$CR_2' = 11001$$
$$CR_3' = 11011$$
$$CR_4' = 10000$$

The second operator that can be applied in every iteration of a GA is mutation. For our example, the mutation operator has a probability of 0.1%, which means that on the 1000 transformed bits, a mutation will be performed only once. Because we transformed only 20 bits (one population of 4×5 bits is transformed into another), the probability that a mutation will occur is very small. Therefore, we can assume that the strings CR'_1 to CR'_4 will stay unchanged with respect to a mutation operation in this first iteration. It is expected that only one bit will be changed for every 50 iterations.

That was the final processing step in the first iteration of the GA. The results, in the form of the new population CR'_1 to CR'_4, are used in the next iteration, which starts again with an evaluation process.

13.3.6 Evaluation (Second Iteration)

The process of evaluation is repeated in the new population. These results are given in Table 13.3.

The process of optimization with additional iterations of the GA can be continued in accordance with Figure 13.3. We will stop here with a presentation of computational steps for our example and give some additional analyses of results useful for a deeper understanding of a GA.

Although the search techniques used in the GA are based on many random parameters, they are capable of achieving a better solution by exploiting the best alternatives in each population. A comparison of sums and average and max values from Tables 13.2 and 13.3

$$\sum\nolimits_1 = 1170 \implies \sum\nolimits_2 = 1754$$
$$\text{Average}_1 = 293 \implies \text{Average}_2 = 439$$
$$\text{Max}_1 = 576 \implies \text{Max}_2 = 729$$

shows that the new second population is approaching closer to the maximum of the function $f(x)$. The best result obtained from the evaluation of chromosomes in the first two iterations is the chromosome $CR3' = 11011$ and it corresponds to the feature's value $x = 27$ (theoretically it is known that the maximum of $f(x)$ is for $x = 31$, where

TABLE 13.3. Evaluation of the Second Generation of Chromosomes

CR_i	Code	x	$f(x)$	$f(x)/\sum f(x)$	Expected Reproduction: $f(x)$/fav
1	01100	12	144	0.08	0.32
2	11001	25	625	0.36	1.44
3	11011	27	729	0.42	1.68
4	10000	16	256	0.14	0.56
$\sum$			1754	1.00	4.00
Average			439	0.25	1.00
Max			729	0.42	1.68

$f(x)$ reaches the value 961). This increase will not be obtained in each GA iteration, but on average, the final population is much closer to a solution after a large number of iterations. The number of iterations is one possible stopping criterion for the GA algorithm. The other possibilities for stopping the GA are when the difference between the sums in two successive iterations is less than the given threshold, when a suitable fitness value is achieved, or when the computation time is limited.

13.4   SCHEMATA

The theoretical foundations of GAs rely on a binary string representation of solutions and on the notation of a *schema*—a template allowing exploration of similarities among chromosomes. To introduce the concept of a schema, we have to first formalize some related terms. The search space Ω is the complete set of possible chromosomes or strings. In a fixed-length string l, where each bit (gene) can take on a value in the alphabet A of size k, the resulting size of the search space is k^l. For example, in binary-coded strings where the length of the string is 8, the size of the search space is $2^8 = 256$. A string in the population S is denoted by a vector $x \in \Omega$. So, in the previously described example, x would be an element of $\{0, 1\}^8$. A schema is a similarity template that defines a subset of strings with fixed values in certain positions.

A schema is built by introducing a *don't care* symbol (*) into the alphabet of genes. Each position in the scheme can take on the values of the alphabet (fixed positions) or a "don't care" symbol. In the binary case, for example, the schemata of the length l are defined as $H \in \{0, 1, *\}^l$. A schema represents all the strings that match it on all positions other than "*." In other words, a schema defines a subset of the search space or a hyperplane partition of this search space. For example, let us consider the strings and schemata of the length 10. The schema

$$(^*111100100)$$

matches two strings

$$\{(0111100100), (1111100100)\},$$

and the schema

$$(^*1^*1100100)$$

matches four strings

$$\{(0101100100),(0111100100),(1101100100),(1111100100)\}.$$

Of course, the schema

$$(1001110001)$$

represents one string only, and the schema

$$(^{**********})$$

represents all strings of length 10. In general, the total number of possible schemata is $(k+1)^l$, where k is the number of symbols in the alphabet and l is the length of the string. In the binary example of coding strings, with a length of 10, it is $(2+1)^{10} = 3^{10} = 59049$ different strings. It is clear that every binary schema matches exactly 2^r strings, where r is the number of *don't care* symbols in a schema template. On the other hand, each string of length m is matched by 2^m different schemata.

We can graphically illustrate the representation of different schemata for five-bit codes used to optimize the function $f(x) = x^2$ on interval [0, 31]. Every schema represents a subspace in the 2D space of the problem. For example, the schema 1**** reduces the search space of the solutions on the subspace given in Figure 13.5a, and the schema 1*0** has a corresponding search space in Figure 13.5b.

Different schemata have different characteristics. There are three important schema properties: *order* (*O*), *length* (*L*), and *fitness* (*F*). The *order* of the schema *S* denoted by *O*(*S*) is the number of 0 and 1 positions, i.e., fixed positions presented in the schema. A computation of the parameter is very simple: it is the length of the
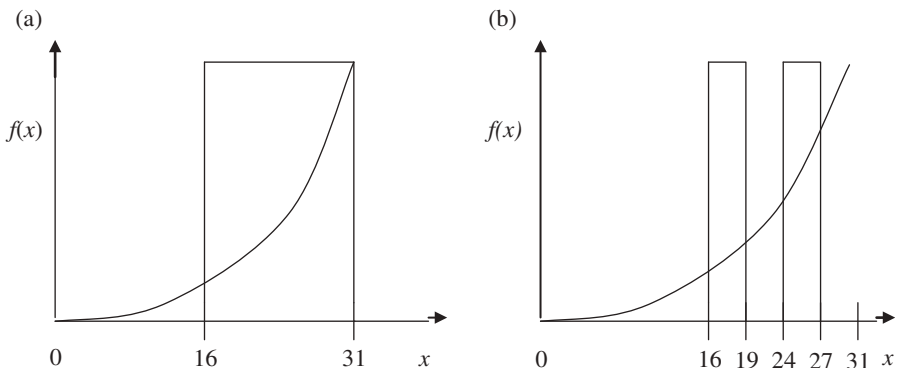
Figure 13.5. $f(x) = x^2$: search spaces for different schemata. (a) Schema 1****. (b) Schema 1*0**.

template minus the number of *don't care* symbols. For example, the following three schemata, each of length 10

$$S_1 = (***001*110)$$
$$S_2 = (*****0**0*)$$
$$S_3 = (11101**001)$$

have the following orders:

$$O(S_1) = 10-4 = 6, \quad O(S_2) = 10-8 = 2, \quad O(S_3) = 10-2 = 8$$

The schema S_3 is the most specific one and the schema S_2 is the most general one. The notation of the order of a schema is useful in calculating survival probabilities of the schema for mutations.

The *length* of the schema S, denoted by $L(S)$, is the distance between the first and the last fixed-string positions. It defines the compactness of information contained in a schema. For example, the values of this parameter for the given schemata S_1 to S_3 are

$$L(S_1) = 10-4 = 6, \quad L(S_2) = 9-6 = 3, \quad L(S_3) = 10-1 = 9.$$

Note that the schema with a single fixed position has a length of zero. The length L of a schema is a useful parameter in calculating the survival probabilities of the schema for crossover.

Another property of a schema S is its *fitness* $F(S, t)$ at time t (i.e., for the given population). It is defined as the average fitness of all strings in the population matched by the schema S. Assume there are p strings $\{v_1, v_2,...,v_p\}$ in a population matched by a schema S at the time t. Then

$$F(S,t) = \frac{\left[\sum_{i=1}^{p} f(v_i)\right]}{p}$$

The fundamental theorem of schema construction given in this book without proof explains that the *short* (high O), *low-order* (low L), and *above-average* schemata (high F) receive exponentially increasing number of strings in the next generations of a GA. An immediate result of this theorem is that GAs explore the search space by short, low-order schemata that, subsequently, are used for information exchange during crossover and mutation operations. Therefore, a GA seeks near-optimal performance through the analysis of these schemata, called the *building blocks*. Note, however, that the building-block approach is just a question of empirical results without any proof, and these rules for some real-world problems are easily violated.

13.5   TRAVELING SALESMAN PROBLEM

In this section, we explain how a GA can be used to approach the TSP. Simply stated, the traveling salesman must visit every city in his territory exactly once and then return to the starting point. Given the cost of travel between all the cities, how should he plan his itinerary at minimum total cost for the entire tour? The TSP is a problem in combinatorial optimization and arises in numerous applications. There are several branch-and-bound algorithms, approximate algorithms, and heuristic search algorithms that approach this problem. During the last few years, there have been several attempts to approximate the TSP solution using GAs.

The TSP description is based on a graph representation of data. The problem could be formalized as follows: given an undirected weighted graph, find the shortest route, i.e., a shortest path in which every vertex is visited exactly once, except that the initial and terminal vertices are the same. Figure 13.6 shows an example of such a graph and its optimal solution. A, B, C, etc., are the cities that were visited, and the numbers associated with the edges are the cost of travel between the cities.

It is natural to represent each solution of the problem, even if it is not optimal, as a permutation of the cities. The terminal city can be omitted in the representation since it should always be the same as the initial city. For the computation of the total distance of each tour, the terminal city must be counted.

By representing each solution as a permutation of the cities, each city will be visited exactly once. Not every permutation, however, represents a valid solution, since some cities are not directly connected (e.g. A and E in Figure 10.6). One practical approach is to assign an artificially large distance between cities that are not directly connected. In this way, invalid solutions that contain consecutive nonadjacent cities will disappear, and all solutions will be allowed.

Our objective here is to minimize the total distance of each tour. We can select different fitness functions that will reflect this objective. For example, if the total distance is s, then $f(s)$ could be a simple $f(s) = s$ if we minimize the fitness function; alternatives for the maximization of a fitness function are $f(s) = 1/s$, $f(s) = 1/s^2$, $f(s) = K - s$, where K is a positive constant that makes $f(s) \geq 0$. There is no general formula to design the best fitness function. But when we do not adequately reflect the goodness

Optimal solution: A -----$>$B------$>$C-------E------$>$D-----$>$(A)
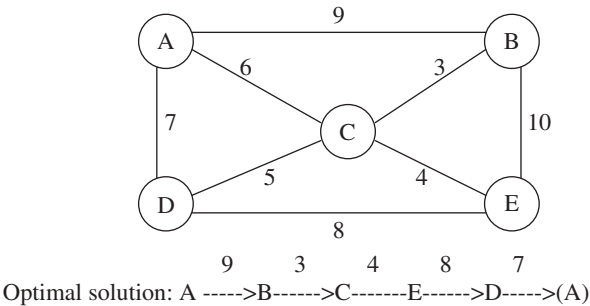
Figure 13.6. Graphical representation of the traveling salesman problem with a corresponding optimal solution.

of the solution in the fitness function, finding an optimal or near-optimal solution will not be effective.

When dealing with permutations as solutions, simple crossover operations will result in invalid solutions. For example, for the problem in Figure 10.6, a crossover of two solutions after the third positions in the strings

$$\Downarrow$$
$$A\ D\ E\ B\ C$$
$$A\ E\ C\ D\ B$$
$$\Uparrow$$

will produce new strings

$$A\ D\ E\ D\ B$$
$$A\ E\ C\ B\ C$$

which are invalid solutions because they do not represent permutations of initial elements in the strings. To avoid this problem, a modified crossover operation is introduced that directly operates on permutations and still gives permutations. This is a *partially matched crossover* (PMX) operation. It can be used not only for the TSP but also for any other problems that involve permutations in a solution's representation. We illustrate the effects of the PMX operation by an example. Assume that two solutions are given as permutations of the same symbols, and suppose that the PMX is a two-point operation. Selecting two strings and two random crossing points is the first step in the process of applying the PMX operation:

$$\Downarrow\ \Downarrow$$
$$A\ D\ E\ B\ C$$
$$A\ E\ C\ D\ B$$
$$\Uparrow\ \Uparrow$$

The substrings between crossing points are called matching sections. In our example, we have two elements in the matching sections: E B for the first string and C D for the second one. The crossover operation requires an exchange of the symbols E with C, denoted as an ordered pair (E, C), and B with D, represented as (B, D). The next step in the PMX operation is to permute each of these two-element permutations in each string. In other words, it is necessary to exchange the places for pairs (E, C) and (B, D) in both strings. The result of (E, C) changes in the first string is A D C B E, and after second pair (B, D) has been changed, the final version of the first string is A B C D E. The second string after application of the same permutations will become A C E D B first and then A C E B D finally. If we analyze the two strings obtained by PMX operation

$$\Downarrow \quad \Downarrow$$

$$A\ B\ C\ D\ E$$

$$A\ C\ E\ B\ D$$

$$\Uparrow \quad \Uparrow$$
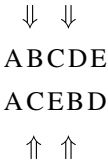
we can see that middle parts of the strings were really exchanged as in a standard crossover operation. On the other hand, the two new strings remain as valid permutations, since the symbols are actually permuted within each string.

The other steps of a GA applied on the TSP are unchanged. A GA based on the above operator outperforms a random search for TSP, but still leaves much room for improvement. Typical results from the algorithm, when applied to 100 randomly generated cities, gave, after 20,000 generations, a value of the whole tour 9.4% above minimum.

13.6 MACHINE LEARNING USING GENETIC ALGORITHMS

Optimization problems are one of the most common application categories of GAs. In general, an optimization problem attempts to determine a solution that maximizes the profit in an organization or minimizes the cost of production by determining values for selected features of the production process. Another typical area where GAs are applied is the discovery of input-to-output mapping for a given, usually complex, system, which is the type of problem that all machine-learning algorithms are trying to solve.

The basic idea of input-to-output mapping is to come up with an appropriate form of a function or a model, which is typically simpler than the mapping given originally—usually represented through a set of input–output samples. We believe that a function best describes this mapping. Measures of the term "best" depend on the specific application. Common measures are accuracy of the function, its robustness, and its computational efficiency. Generally, determining a function that satisfies all these criteria is not necessarily an easy task; therefore, a GA determines a "good" function, which can then be used successfully in applications such as pattern classification, control, and prediction. The process of mapping may be automated, and this automation, using GA technology, represents another approach to the formation of a model for inductive machine learning.

In the previous chapters of the book, we described different algorithms for machine learning. Developing a new model (input-to-output abstract relation) based on some given set of samples is an idea that can also be implemented in the domain of GAs. There are several approaches to GA-based learning. We will explain the principles of the technique that is based on schemata and the possibilities of its application to classification problems.

Let us consider a simple database that will be used as an example throughout this section. Suppose that the training or learning data set is described with a set of attributes where each attribute has its categorical range: a set of possible values. These attributes are given in Table 13.4.

TABLE 13.4. Attributes A_i with Possible Values for a Given Data Set s

Attributes	Values
A_1	x, y, z
A_2	x, y, z
A_3	y, n
A_4	m, n, p
A_5	r, s, t, u
A_6	y, n

The description of a classification model with two classes of samples, C_1 and C_2, can be represented in an *if-then* form where on the left side is a Boolean expression of the input features' values and on the right side its corresponding class:

$$((A_1 = x) \wedge (A_5 = s)) \vee ((A_1 = y) \wedge (A_4 = n)) \Rightarrow C_1$$

$$((A_3 = y) \wedge (A_4 = n)) \vee (A_1 = x) \Rightarrow C_2$$

These classification rules or classifiers can be represented in a more general way: as some strings over a given alphabet. For our data set with six inputs and one output, each classifier has the form

$$(p_1, p_2, p_3, p_4, p_5, p_6) : d$$

where p_i denotes the value of the ith attribute ($1 \leq i \leq 6$) for the domains described in Table 13.4 and d is one of two classes. To describe the classification rules in a given form, it is necessary to include the "don't care" symbol "*" into the set of values for each attribute. For example, the new set of values for attribute A_1 is $\{x, y, z, *\}$. Similar extensions are given for other attributes. The rules that were given earlier for classes C_1 and C_2 can be decomposed to the segments under conjunction (AND logical operation) and expressed as

$$(x^{***}s^*) : C_1$$
$$(y^{**}n^{**}) : C_1$$
$$(^{**}yn^{**}) : C_2$$
$$(x^{*****}) : C_2$$

To simplify the example, we assume that the system has to classify into only two classes: C_1 and *not* C_1. Any system can be easily generalized to handle multiple

classes (multiple classification). For a simple classification with a single rule C_1, we can accept only two values for d: $d = 1$ (member of the class C_1) and $d = 0$ (not a member of the class C_1).

Let us assume that at some stage of the learning process, there is a small and randomly generated population of classifiers Q in the system, where each classifier is given with its strength s:

$$Q_1 \quad (***ms^*):1, \quad s_1 = 12.3$$
$$Q_2 \quad (**y**n):0, \quad s_2 = 10.1$$
$$Q_3 \quad (xy****):1, \quad s_3 = 8.7$$
$$Q_4 \quad (*z****):0, \quad s_4 = 2.3$$

Strengths s_i are parameters that are computed based on the available training data set. They show the fitness of a rule to the training data set, and they are proportional to the percentage of the data set supported by the rule.

The basic iterative steps of a GA, including corresponding operators, are applied here to optimize the set of rules with respect to the fitness function of the rules to training data set. The operators used in this learning technique are, again, mutation and crossover. However, some modifications are necessary for mutation. Let us consider the first attribute A_1 with its domain $\{x, y, z, ^* \}$. Thus, when mutation is called, we would change the mutated character (code) to one of the other three values that have equal probability. The strength of the offspring is usually the same as that of its parents. For example, if mutation is the rule Q_3 on the randomly selected position 2, replacing the value y with a randomly selected value *, the new mutated classifier will be

$$Q_{3M} \quad (x*****):1, \quad s_{3M} = 8.7$$

The crossover does not require any modification. We take advantage of the fact that all classifiers are of equal length. Therefore, to crossover two selected parents, say, Q_1 and Q_2

$$\Downarrow$$
$$Q_1 \quad (***ms^*):1$$
$$Q_2 \quad (**y**n):0$$

we generate a random crossover-position point. Suppose that we cross over after the third character in the string as marked, and then the offspring are

$$Q_{1c} \quad (*****n):0,$$
$$Q_{2c} \quad (**yms^*):1$$

The strength of the offspring is an average (possibly weighted) of the parents' strengths. Now the system is ready to continue its learning process: starting another cycle and accepting further positive and negative samples from the training set and modifying the strengths of classifiers as a measure of fitness. We can note that the training data set is included in the learning process through evaluation of schema's strengths for each iteration. We expect that the population of classifiers converges to some rules with very high strengths.

One of the possible implementations of the previous ideas is the *GIL system*, which moves the GA closer to the symbolic level—mainly by defining specialized operators that manipulate binary strings. Previous symbolic classifiers are translated into binary strings, where for each attribute a binary string of fixed length is generated. The length is equal to the number of possible values for the given attribute. In the string, the required value is set to 1, and all others are set to 0. For example, if attribute A_1 has the value z, it is represented with the binary string 001 (0s are for values x, y). If the value of some attribute is *, that means that all values are possible, so it is represented with value 1 in all positions of the binary string.

For our previous example, with six attributes and a total number of seventeen different values for all attributes, the classifier symbolically represented as

$$(x^{***}r^*) \vee (y^{**}n^{**}) : 1$$

can be transformed into the binary representation

$$(100 \,|\, 111 \,|\, 11 \,|\, 111 \,|\, 1000 \,|\, 11 \vee 010 \,|\, 111 \,|\, 11 \,|\, 010 \,|\, 1111 \,|\, 11)$$

 where bars separate bitsets for each attribute. The operators of the GIL system are modeled on inductive reasoning, which includes various inductive operators such as RuleExchange, RuleCopy, RuleGeneralization, RuleSpecialization, RuleSplit, SelectorDrop, ReferenceChange, ReferenceExtension, etc. We discuss some of them in turn.

13.6.1   RuleExchange

The RuleExchange operator is similar to a crossover of the classical GA, as it exchanges selected complex between two parent chromosomes. For example, two parents (two rules)

$$\Downarrow$$
$$(100 \,|\, 111 \,|\, 11 \,|\, 111 \,|\, 1000 \,|\, 11 \vee 010 \,|\, 111 \,|\, 11 \,|\, 010 \,|\, 1111 \,|\, 11) \quad \text{and}$$

$$(111 \,|\, 001 \,|\, 01 \,|\, 111 \,|\, 1111 \,|\, 01 \vee 110 \,|\, 100 \,|\, 10 \,|\, 010 \,|\, 0011 \,|\, 01)$$
$$\Uparrow$$

may produce the following offspring (new rules):

$$(100\,|\,111\,|\,11\,|\,111\,|\,1000\,|\,11 \lor 110\,|\,100\,|\,10\,|\,010\,|\,0011\,|\,01)\;\; \text{and}$$

$$(111\,|\,001\,|\,01\,|\,111\,|\,1111\,|\,01 \lor 010\,|\,111\,|\,11\,|\,010\,|\,1111\,|\,11).$$

13.6.2 RuleGeneralization

This unary operator generalizes a random subset of complexes. For example, for a parent

$$(100|111|11|111|1000\,|\,11 \lor 110|100|10|010|0011\,|\,01 \lor 010\,|\,111\,|\,11\,|\,010\,|\,1111\,|\,11)$$

and the second and third complexes selected for generalization, the bits are *OR*ed, and the following offspring is produced:

$$(100\,|\,111\,|\,11\,|\,111\,|\,1000\,|\,11 \lor 110\,|\,111\,|\,11\,|\,010\,|\,1111\,|\,11)$$

13.6.3 RuleSpecialization

This unary operator specializes a random subset of complexes. For example, for a parent

$$(100\,|\,111\,|\,11\,|\,111\,|\,1000\,|\,11 \lor 110\,|\,100\,|\,10\,|\,010\,|\,0011\,|\,01 \lor 010\,|\,111\,|\,11\,|\,010\,|\,1111\,|\,11)$$

and the second and third complexes selected for specialization, the bits are *AND*ed, and the following offspring is produced:

$$(100\,|\,111\,|\,11\,|\,111\,|\,1000\,|\,11 \lor 010\,|\,100\,|\,10\,|\,010\,|\,0011\,|\,01)$$

13.6.4 RuleSplit

This operator acts on a single complex, splitting it into a number of complexes. For example, a parent

$$(100 \mid 111 \mid 11 \mid 111 \mid 1000 \mid 11)$$

$$= = =$$

$$\Uparrow$$

may produce the following offspring (the operator splits the second selector):

$$(100 \mid 011 \mid 11 \mid 111 \mid 1000 \mid 11 \vee 100 \mid 100 \mid 11 \mid 111 \mid 1000 \mid 11)$$

The GIL system is a complex inductive-learning system based on GA principles. It requires a number of parameters such as the probabilities of applying each operator. The process is iterative. At each iteration, all chromosomes are evaluated with respect to their completeness, consistency, and fitness criteria, and a new population is formed with those chromosomes that are better and more likely to appear. The operators are applied to the new population, and the cycle is repeated.

13.7 GENETIC ALGORITHMS FOR CLUSTERING

Much effort has been undertaken toward applying GAs to provide better solutions than those found by traditional clustering algorithms. The emphasis was on appropriate encoding schemes, specific genetic operators, and corresponding fitness functions. Several encoding schemes have been proposed specifically for data clustering, and main three types are *binary*, *integer*, and *real encoding*.

Binary encoding solution is usually represented as a binary string of length N, where N is the number of data set samples. Each position of the binary string corresponds to a particular sample. The value of the ith gene is 1 if the ith sample is a prototype of a cluster and zero otherwise. For example, the data set s in Table 13.5

TABLE 13.5. Three Clusters are Defined for a Given Data Set s

Samples	Feature 1	Feature 2	Cluster
1	1	1	C_1
2	1	2	C_1
3	2	1	C_1
4	2	2	C_1
5	10	1	C_2
6	10	2	C_2
7	11	1	C_2
8	11	2	C_2
9	5	5	C_3
10	5	6	C_3

TABLE 13.6. Binary Encoded Data Set *s* given in Table 13.5

1	1	1	1	0	0	0	0	0	0
0	0	0	0	1	1	1	1	0	0
0	0	0	0	0	0	0	0	1	1

can be encoded by means of the string [0100001010], in which samples 2, 7, and 9 are prototypes for clusters C_1, C_2, and C_3. Total number of 1s in the string is equal to a priori defined number of clusters. Clearly, such an encoding scheme leads to a *medoid-based representation* in which the cluster prototypes coincide with representative samples from the data set. There is an alternative way to represent a data partition using a *binary encoding*. The matrix of $k \times N$ dimensions is used in which the rows represent clusters and the columns represent samples. In this case, if the *j*th sample belongs to the *i*th cluster, then 1 is assigned to (i,j) genotype, whereas the other elements of the same column receive 0. For example, using this representation, the data set in Table 13.5 would be encoded as 3×10 matrix in Table 13.6.

Integer encoding uses a vector of N integer positions where N is the number of data set samples. Each position corresponds to a particular sample, i.e., the *i*th position (gene) represents the *i*th data sample. Assuming that there is k clusters, each gene has a value over the alphabet $\{1, 2, 3,…,k\}$. These values define the cluster labels. For example, the integer vector [1111222233] represents the clusters depicted in Table 13.5. Another way of representing a partition by means of an *integer encoding* scheme involves using an array of only k elements to provide a *medoid-based representation* of the data set. In this case, each array element represents the index of the sample x_i, $i = 1, 2,…,N$ (with respect to the order the samples appear in the data set) corresponding to the prototype of a given cluster. As an example, the array [1 6 10] may represent a partition in which 1, 6, and 10 are indices of the cluster prototypes (medoids) for the data set in Table 13.5. Integer encoding is usually more computationally efficient than the *binary encoding* schemes.

Real encoding is the third encoding scheme where the genotypes are made up of real numbers that represent the coordinates of the cluster centroids. In an *n*-dimensional space, the first n positions represent the n coordinates of the first centroid, the next n positions represent the coordinates of the second centroid, and so forth. To illustrate this, the genotype [1.5 1.5 10.5 1.5 5.0 5.5] encodes the three centroids, (1.5, 1.5), (10.5, 1.5), and (5.0, 5.5), of clusters C_1, C_2, and C_3 in Table 13.5, respectively.

Second important decision, in applying GAs for clustering, is a selection of appropriate *genetic operators*. A number of crossover and mutation operators are proposed trying to solve an important problem of the *context insensitivity* in GAs. It means when traditional genetic operators are employed in clustering problems, they usually just manipulate gene values without taking into account their connections with
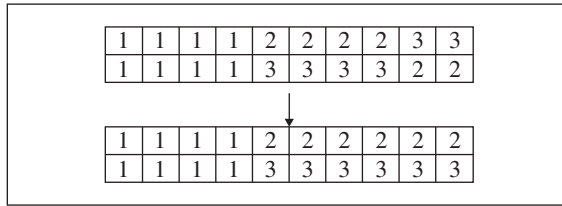
Figure 13.7. Equal parents produce different offspring through crossover.

other genes. For example, crossover operation presented in Figure 13.7 shows how two parents representing the same solution to the clustering problem (different labeling but the same integer encoding) produce the resulting offspring representing clustering solutions different from the ones encoded into their parents. Moreover, assuming that the number of clusters has been fixed in advance as $k = 3$, invalid solutions with only two clusters have been derived. Therefore, it is necessary to develop specially designed genetic operators for clustering problems. For example, the crossover operator should be repeatedly applied, or randomly scrambling mutation was performed, until a valid child has been found.

Different clustering validity criteria are adapted as *fitness functions* to evolve data partitions in clustering problems. They depend primary on encoding schemes but also on a selected set of genetic operators. We will illustrate in this text only one example of a clustering fitness function when *real, centroid-based* encoding scheme is used. Fitness function f minimizes the sum of squared Euclidean distances of the samples from their respective cluster means. Formally, the fitness function f $(C_1, C_2,\ldots,C_k)$ is

$$f(C_1, C_2, \ldots, C_k) = \sum_{j=1}^{k} \sum_{x_i \in C_j} |x_i - z_j|^2$$

where $\{C_1, C_2,\ldots,C_k\}$ is the set of k clusters encoded into the genotype, x_i is a sample in a data set, and z_j is the centroid of cluster C_j. It is important to stress that this criterion is valid only if the number of clusters k is given in advance, and it minimizes the intra-cluster distances and maximizes the inter-cluster distances as well. In general, fitness functions are based on the distance between samples and either clusters' centroids or medoids. Although these types of functions are widely used, usually they are biased toward the discovery of spherical clusters, which clearly will be inappropriate in many real-world applications. Other approaches are possible including density-based fitness function. In practice, the success of a GA to solve a clustering problem is highly dependent upon how it has been designed in terms of encoding scheme, operators, fitness function, selection procedure, and initial population generation.

13.8   REVIEW QUESTIONS AND PROBLEMS

1. Given a binary string that represents a concatenation of four attribute values:

$$\{2,5,4,7\} = \{010101100111\}$$

   Use this example to explain the basic concepts of a genetic algorithm and their equivalents in natural evolution.

2. If we want to optimize a function $f(x)$ using a genetic algorithm, where the precision requirement for x is six decimal places and the range is $[-1, 2]$, what will be the length of a binary vector (chromosome)?

3. If $v_1 = (0\ 0\ 1\ 1\ 0\ 0\ 1\ 1)$ and $v_2 = (0\ 1\ 0\ 1\ 0\ 1\ 0\ 1)$ are two chromosomes, and suppose that the crossover point is randomly selected after the fifth gene, what are the two resulting offspring?

4. Given the schema (* 1 * 0 0), what are the strings that match with it?

5. What is the number of strings that match with the schema (* * * * * * * *)?

6. The function $f(x) = -x^2 + 16x$ is defined on interval $[0, 63]$. Use two iterations of a genetic algorithm to establish the approximate solution for a maximum of $f(x)$.

7. For the function $f(x)$ given in problem #6, compare three schemata

$$S_1 = (*1*1**)$$
$$S_2 = (*10*1*)$$
$$S_3 = (**1***)$$

   with respect to order (O), length (L), and fitness (F).

8. Given a parent chromosome $(1\ 1\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 1)$, what is the potential offspring (give examples) if the mutation probability is:
   (a) $p_m = 1.0$
   (b) $p_m = 0.5$
   (c) $p_m = 0.2$
   (d) $p_m = 0.1$
   (e) $p_m = 0.0001$

9. Explain the basic principles of the building-block hypothesis and its potential applications.

10. Perform a partially matched crossover (PMC) operation for two strings $S1$ and $S2$, in which two randomly selected crossing points are given.

$$S1 = \{\,A\,C\,B\,D\,F\,G\,E\,\}$$
$$S2 = \{\,B\,D\,C\,F\,E\,G\,A\,\}$$
$$\uparrow \quad \uparrow$$

11. Search the Web to find the basic characteristics of publicly available or commercial software tools that are based on genetic algorithms. Document the results of your search.

13.9 REFERENCES FOR FURTHER STUDY

1. Michalewicz, Z., *Genetic Algorithms + Data Structures = Evolution Programs*, Springer, Berlin, Germany, 1999.

   This textbook explains the field of genetic algorithms in simple terms and discusses the efficiency of its methods on many interesting test cases. The importance of these techniques is their applicability to many hard-optimization problems specified with large amounts of discrete data, such as the traveling salesman problem, scheduling, partitioning, and control.

2. Fogel, D. B., ed., *Evolutionary Computation*, IEEE Press, New York, 1998.

   The book provides a collection of 30 landmark papers, and it spans the entire history of evolutionary computation—from today's researches back to its very origins more than 40 years ago.

3. Goldberg, D. E., *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison Wesley, Reading, MA, 1989.

   This book represents one of the first comprehensive texts on genetic algorithms. It introduces in a very systematic way most of the techniques that are, with small modifications and improvements, part of today's approximate-optimization solutions.

4. Hruschka E., R. Campello, A. Freitas, A. Carvalho, A Survey of Evolutionary Algorithms for Clustering, *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, Vol. 39, No. 2, March 2009, pp. 133–155.

   This paper presents a survey of evolutionary algorithms designed for clustering tasks. It tries to reflect the profile of this area by focusing more on those subjects that have been given more importance in the literature. In this context, most of the paper is devoted to partitional algorithms that look for hard clustering of data, though overlapping (i.e., soft and fuzzy) approaches are also covered in the paper. The paper ends by addressing some important issues and open questions that can be subject of future research.

5. Sheppard C., *Genetic Algorithms with Python*, Clinton Sheppard, 2018.

   Get a hands-on introduction to machine learning with genetic algorithms using Python. Genetic algorithms are one of the tools you can use to apply machine

learning to finding good, sometimes even optimal, solutions to problems that have billions of potential solutions. This book gives you experience making genetic algorithms work for you using easy-to-follow example projects that you can fall back upon when learning to use other machine-learning tools and techniques. The step-by-step tutorials build your skills from Hello World! to optimizing one genetic algorithm with another and finally genetic programming, thus preparing you to apply genetic algorithms to problems in your own field of expertise.

14

FUZZY SETS AND FUZZY LOGIC

Chapter Objectives

- Explain the concept of fuzzy sets with formal interpretation in continuous and discrete domains.
- Analyze characteristics of fuzzy sets and fuzzy set operations.
- Describe the extension principle as a basic mechanism for fuzzy inferences.
- Discuss the importance of linguistic imprecision and computing with them in decision-making processes.
- Construct methods for multifactorial evaluation and extraction of a fuzzy rule-based model from large, numeric data sets.
- Understand why fuzzy computing and fuzzy systems are an important part of data-mining technology.

In the previous chapters, a number of different methodologies for the analysis of large data sets have been discussed. Most of the approaches presented, however, assume that the data is precise. That is, they assume that we deal with exact measurements for further analysis. Historically, as reflected in classical mathematics, we commonly seek a precise and crisp description of things or events. This precision is accomplished by expressing phenomena in numeric or categorical values. But in most, if not all, real-world scenarios, we will never have totally precise values. There is always going to be a degree of uncertainty. However, classical mathematics can encounter substantial difficulties because of this fuzziness. In many real-world situations, we may say that fuzziness is reality whereas crispness or precision is simplification and idealization. The polarity between fuzziness and precision is quite a striking contradiction in the development of modern information-processing systems. One effective means of resolving the contradiction is the fuzzy set theory, a bridge between high precision and the high complexity of fuzziness.

14.1   FUZZY SETS

Fuzzy concepts derive from fuzzy phenomena that commonly occur in the real world. For example, rain is a common natural phenomenon that is difficult to describe precisely since it can rain with varying intensity anywhere from a light shower to a torrential downpour. Since the word rain does not adequately or precisely describe the wide variations in the amount and intensity of any rain event, "rain" is considered a fuzzy phenomenon.

Often, the concepts formed in the human brain for perceiving, recognizing, and categorizing natural phenomena are also fuzzy. The boundaries of these concepts are vague. Therefore, the judging and reasoning that emerges from them are also fuzzy. For instance, "rain" might be classified as "light rain," "moderate rain," and "heavy rain" in order to describe the degree of raining. Unfortunately, it is difficult to say when the rain is light, moderate, or heavy, because the boundaries are undefined. The concepts of "light," "moderate," and "heavy" are prime examples of fuzzy concepts themselves. To explain the principles of fuzzy sets, we will start with the basics in classical set theory.

The notion of a set occurs frequently as we tend to organize, summarize, and generalize knowledge about objects. We can even speculate that the fundamental nature of any human being is to organize, arrange, and systematically classify information about the diversity of any environment. The encapsulation of objects into a collection whose members all share some general features naturally implies the notion of a set. Sets are used often and almost unconsciously; we talk about a set of even numbers, positive temperatures, personal computers, fruits, and the like. For example, a classical set A of real numbers greater than 6 is a set with a crisp boundary, and it can be expressed as

$$A = \{x \mid x > 6\}$$

where there is a clear, unambiguous boundary 6 such that if x is greater than this number, then x belongs to the set A; otherwise x does not belong to the set. Although classical sets have suitable applications and have proven to be an important tool for mathematics and computer science, they do not reflect the nature of human concepts and thoughts, which tend to be abstract and imprecise. As an illustration, mathematically we can express a set of tall persons as a collection of persons whose height is more than 6 ft; this is the set denoted by previous equation, if we let A = "tall person" and x = "height." Yet, this is an unnatural and inadequate way of representing our usual concept of "tall person." The dichotomous nature of the classical set would classify a person 6.001 ft tall as a tall person, but not a person 5.999 ft tall. This distinction is intuitively unreasonable. The flaw comes from the sharp transition between inclusions and exclusions in a set.

In contrast to a classical set, a fuzzy set, as the name implies, is a set without a crisp boundary. That is, the transition from "belongs to a set" to "does not belong to a set" is gradual, and this smooth transition is characterized by membership functions that give sets flexibility in modeling commonly used linguistic expressions such as "the water is hot" or "the temperature is high." Let us introduce some basic definitions and their formalizations concerning fuzzy sets.

Let X be a space of objects and x be a generic element of X. A classical set A, $A \subseteq X$, is defined as a collection of elements or objects $x \in X$ such that each x can either belong or not belong to the set A. By defining a *characteristic function* for each element x in X, we can represent a classical set A by a set of ordered pairs $(x, 0)$ or $(x, 1)$, which indicates $x \notin A$ or $x \in A$, respectively.

Unlike the aforementioned conventional set, a fuzzy set expresses the degree to which an element belongs to a set. The characteristic function of a fuzzy set is allowed to have values between 0 and 1, which denotes the degree of membership of an element in a given set. If X is a collection of objects denoted generically by x, then a fuzzy set A in X is defined as a set of ordered pairs:

$$A = \{(x, m_A(x)) \mid x \in X\}$$

where $\mu_A(x)$ is called the membership function (or MF for short) for the fuzzy set A. The membership function maps each element of X to a membership grade (or membership value) between 0 and 1.

Obviously, the definition of a fuzzy set is a simple extension of the definition of a classical set in which the characteristic function is permitted to have any value between 0 and 1. If the value of the membership function $\mu_A(x)$ is restricted to either 0 or 1, then A is reduced to a classic set, and $\mu_A(x)$ is the characteristic function of A. For clarity, we shall also refer to classical sets as ordinary sets, crisp sets, nonfuzzy sets, or, just, sets.

Usually X is referred to as the universe of discourse, or, simply, the universe, and it may consist of discrete (ordered or nonordered) objects or continuous space. This can be clarified by the following examples. Let X = {San Francisco, Boston, Los

Angeles} be the set of cities one may choose to live in. The fuzzy set C = "desirable city to live in" may be described as follows:

$$\text{Let } C = \{ (\text{San Francisco}, 0.9), (\text{Boston}, 0.8), (\text{Los Angeles}, 0.6) \}$$

The universe of discourse X is discrete, and it contains nonordered objects: three big cities in the United States. As one can see, the membership grades listed above are quite subjective; anyone can come up with three different but legitimate values to reflect his or her preference.

In the next example, let $X = \{0, 1, 2, 3, 4, 5, 6\}$ be a set of the number of children a family may choose to have. Then the fuzzy set A = "sensible number of children in a family" may be described as follows:

$$A = \{(0, 0.1), (1, 0.3), (2, 0.7), (3, 1), (4, 0.7), (5, 0.3), (6, 0.1)\}$$

or, in the notation that we will use through this chapter,

$$A = 0.1/0 + 0.3/1 + 0.7/2 + 1.0/3 + 0.7/4 + 0.3/5 + 0.1/6$$

Here we have a discrete-order universe X; the membership function for the fuzzy set A is shown in Figure 14.1a. Again, the membership grades of this fuzzy set are obviously subjective measures.

Finally, let $X = R^+$ be the set of possible ages for human beings. Then the fuzzy set B = "about 50-years old" may be expressed as

$$B = \{(x, \mu_B(x)) \mid x \in X\}$$

where

$$\mu_B(x) = \frac{1}{\left(1 + ((x-50)/10)^4\right)}$$
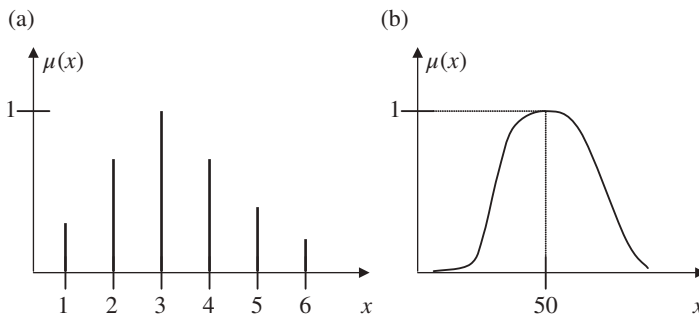
Figure 14.1. Discrete and continuous representation of membership functions for given fuzzy sets. (a) A = "sensible number of children". (b) B = "about 50 years old."
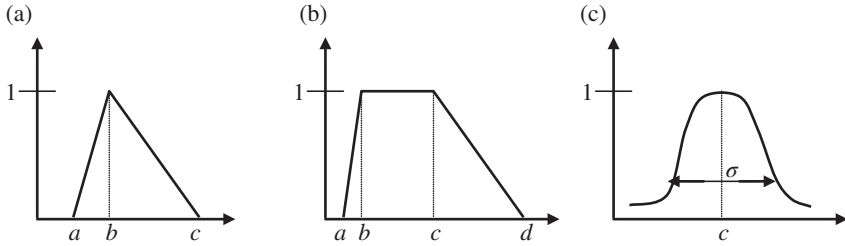
Figure 14.2. Most commonly used shapes for membership functions. (a) Triangular. (b) Trapezoidal. (c) Gaussian.

This is illustrated in Figure 14.1b.

As mentioned earlier, a fuzzy set is completely characterized by its membership function (MF). Since many fuzzy sets in use have a universe of discourse X consisting of the real line R, it would be impractical to list all the pairs defining a membership function. A more convenient and concise way to define a membership function is to express it as a mathematical formula. Several classes of parametrized membership functions are introduced, and in real-world applications of fuzzy sets, the shape of membership functions is usually restricted to a certain class of functions that can be specified with only a few parameters. The most well known are triangular, trapezoidal, and Gaussian; Figure 14.2 shows these commonly used shapes for membership functions.

A triangular membership function is specified by three parameters $\{a, b, c\}$ as follows:

$$\mu(x) = \text{triangle}\,(x,\, a,\, b,\, c) = \begin{cases} 0 & \text{for } x \leq a \\ \dfrac{(x-a)}{(b-a)} & \text{for } a \leq x \leq b \\ \dfrac{(c-x)}{(c-b)} & \text{for } b \leq x \leq c \\ 0 & \text{for } c \leq x \end{cases}$$

The parameters $\{a, b, c\}$, with $a < b < c$, determine the x coordinates of the three corners of the underlying triangular membership function.

A trapezoidal membership function is specified by four parameters $\{a, b, c, d\}$ as follows:

$$\mu(x) = \text{trapezoid}\,(x,\, a,\, b,\, c,\, d) = \begin{cases} 0 & \text{for } x \leq a \\ \dfrac{(x-a)}{(b-a)} & \text{for } a \leq x \leq b \\ 1 & \text{for } b \leq x \leq c \\ \dfrac{(d-x)}{(d-c)} & \text{for } c \leq x \leq d \\ 0 & \text{for } d \leq x \end{cases}$$

(a)   (b)   (c)
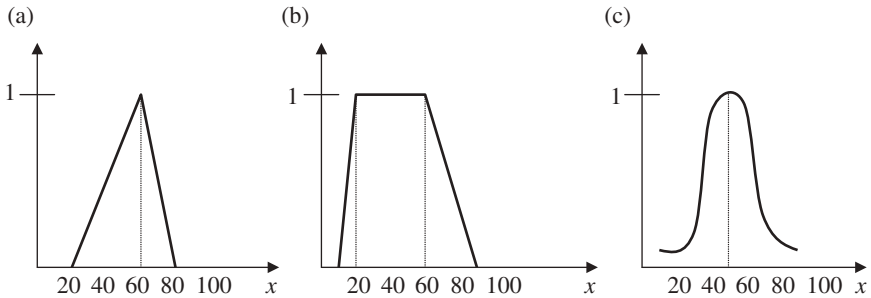
Figure 14.3. Examples of parametrized membership functions. (a) Triangle(x, 20, 60, 80). (b) Trapezoid(x, 10, 20, 60, 90). (c) Gaussian(x, 50, 20).

The parameters $\{a, b, c, d\}$, with $a < b \le c < d$, determine the x coordinates of the four corners of the underlying trapezoidal membership function. A triangular membership function can be seen as a special case of the trapezoidal form where $b = c$.

Finally, a Gaussian membership function is specified by two parameters $\{c, \sigma\}$:

$$\mu(x) = \text{gaussian}(x, c, \sigma) = e^{-1/2\,((x-c)/\sigma)2}$$

A Gaussian membership function is determined completely by c and σ; c represents the membership-function center, and σ determines the membership-function width. Figure 14.3 illustrates the three classes of parametrized membership functions.

From the preceding examples, it is obvious that the construction of a fuzzy set depends on two things: the identification of a suitable universe of discourse and the specification of an appropriate membership function. The specification of membership function is subjective, which means that the membership functions for the same concept (say, "sensible number of children in a family") when specified by different persons may vary considerably. This subjectivity comes from individual differences in perceiving or expressing abstract concepts and has little to do with randomness. Therefore, the *subjectivity* and *nonrandomness* of fuzzy sets is the primary difference between the study of fuzzy sets and the probability theory, which deals with the objective treatment of random phenomena.

There are several parameters and characteristics of membership function that are used very often in some fuzzy set operations and fuzzy set inference systems. We will define only some of them that are, in our opinion, the most important:

1. *Support*—The *support* of a fuzzy set A is the set of all points x in the universe of discourse X such that $\mu_A(x) > 0$:

$$\text{Support}(A) = \{x \mid \mu_A(x) > 0\}$$

2. *Core*—The core of a fuzzy set A is the set of all points x in X such that $\mu_A(x) = 1$:

$$\text{Core}(A) = \{x \mid \mu_A(x) = 1\}$$

3. *Normalization*—A fuzzy set A is *normal* if its core in nonempty. In other words, we can always find a point $x \in \underline{X}$ such that $\mu_A(x) = 1$.

4. *Cardinality*—Given a fuzzy set A in a finite universe X, its cardinality, denoted by Card(A), is defined as

$$\text{Card}(A) = \sum \mu_A(x) \quad \text{where } x \in X$$

Often, Card(X) is referred to as the scalar cardinality or the count of A. For example, the fuzzy set $A = 0.1/1 + 0.3/2 + 0.6/3 + 1.0/4 + 0.4/5$ in universe $X = \{1, 2, 3, 4, 5, 6\}$ has a cardinality Card(A) = 2.4.

5. *α-cut*—The *α-cut* or *α-level set* of a fuzzy set A is a crisp set defined by

$$A_\alpha = \{x \mid \mu_A(x) \ge \alpha\}$$

6. *Fuzzy* number—Fuzzy numbers are a special type of fuzzy sets restricting the possible types of membership functions:

   (a) The membership function must be normalized (i.e., the core is nonempty) and singular. This results in precisely one point, which lies inside the core, modeling the typical value of the fuzzy number. This point is called the modal value.

   (b) The membership function has to monotonically increase left of the core and monotonically decrease on the right. This ensures that only one peak and, therefore, only one typical value exists. The spread of the support (i.e., the nonzero area of the fuzzy set) describes the degree of imprecision expressed by the fuzzy number.

A graphical illustration for some of these basic concepts is given in Figure 14.4.
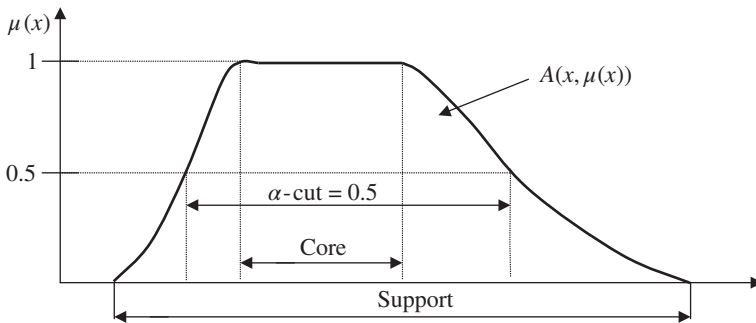
Figure 14.4. Core, support, and α-cut for fuzzy set *A*.

14.2   FUZZY SET OPERATIONS

Union, intersections, and complement are the most basic operations in classic sets. Corresponding to the ordinary set operations, fuzzy sets too have operations, which were initially defined by Zadeh, the founder of the fuzzy set theory.

The *union* of two fuzzy sets A and B is a fuzzy set C, written as $C = A \cup B$ or $C = A$ *OR B*, whose membership function $\mu_C(x)$ is related to those of A and B by

$$\mu_C(x) = \max\left(\mu_A(x), \mu_B(x)\right) = \mu_A(x) \vee \mu_B(x), \quad \forall x \in X$$

As pointed out by Zadeh, a more intuitive but equivalent definition of the union of two fuzzy sets A and B is the "smallest" fuzzy set containing both A and B. Alternatively, if D is any fuzzy set that contains both A and B, then it also contains $A \cup B$.

The *intersection* of fuzzy sets can be defined analogously. The intersection of two fuzzy sets A and B is a fuzzy set C, written as $C = A \cap B$ or $C = A$ *AND B*, whose membership function is related to those of A and B by

$$\mu_C(x) = \min\left(\mu_A(x), \mu_B(x)\right) = \mu_A(x) \wedge \mu_B(x), \quad \forall x \in X$$

As in the case of the union of sets, it is obvious that the intersection of A and B is the "largest" fuzzy set that is contained in both A and B. This reduces to the ordinary intersection operation if both A and B are nonfuzzy.

The *complement* of a fuzzy set A, denoted by A', is defined by the membership function as

$$\mu_{A'}(x) = 1 - \mu_A(x), \quad \forall x \in X$$

Figure 14.5 demonstrates these three basic operations: Figure 14.5a illustrates two fuzzy sets A and B; Figure 14.5b is the complement of A; Figure 14.5c is the union of A and B; and Figure 14.5d is the intersection of A and B.

Let A and B be fuzzy sets in X and Y domains, respectively. The Cartesian product of A and B, denoted by $A \times B$, is a fuzzy set in the product space $X \times Y$ with a membership function:

$$\mu_{A \times B}(x, y) = \min\left(\mu_A(x), \mu_B(y)\right) = \mu_A(x) \wedge \mu_B(y), \quad \forall x \in X \text{ and } \forall y \in Y$$

Numeric computations based on these simple fuzzy operations are illustrated through one simple example with a discrete universe of discourse S. Let $S = \{1, 2, 3, 4, 5\}$ and assume that fuzzy sets A and B are given by

$$A = 0/1 + 0.5/2 + 0.8/3 + 1.0/4 + 0.2/5$$
$$B = 0.9/1 + 0.4/2 + 0.3/3 + 0.1/4 + 0/5$$
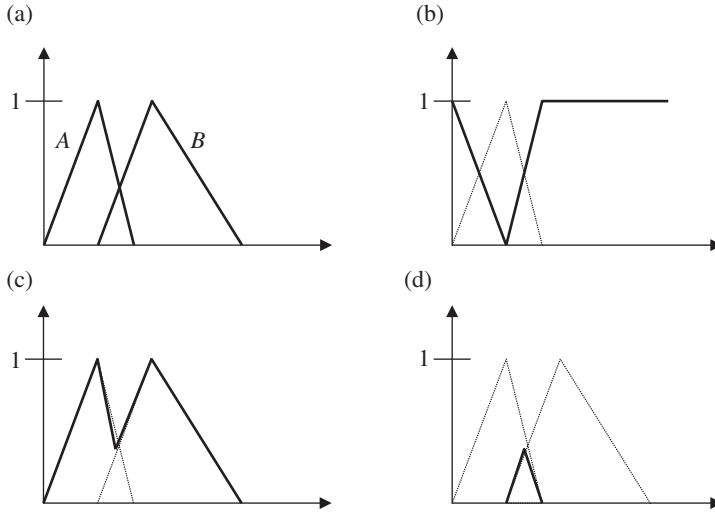
Figure 14.5. Basic operations on fuzzy sets. (a) Fuzzy sets A and B. (b) $C = A'$. (c) $C = A \cup B$. (d) $C = A \cap B$.

Then,

$$A \cup B = 0.9/1 + 0.5/2 + 0.8/3 + 1.0/4 + 0.2/5$$
$$A \cap B = 0/1 + 0.4/2 + 0.3/3 + 0.1/4 + 0/5$$
$$A^C = 1/1 + 0.5/2 + 0.2/3 + 0/4 + 0.8/5$$

and the Cartesian product of fuzzy sets A and B is

$$A \times B = 0/(1,1) + 0/(1,2) + 0/(1,3) + 0/(1,4) + 0/(1,5)$$
$$+ 0.5/(2,1) + 0.4/(2,2) + 0.3/(2,3) + 0.1/(2,4) + 0/(2,5)$$
$$+ 0.8/(3,1) + 0.4/(3,2) + 0.3/(3,3) + 0.1/(3,4) + 0/(3,5)$$
$$+ 0.9/(4,1) + 0.4/(4,2) + 0.3/(4,3) + 0.1/(4,4) + 0/(4,5)$$
$$+ 0.2/(5,1) + 0.2/(5,2) + 0.2/(5,3) + 0.1/(5,4) + 0/(5,5)$$

Fuzzy sets, as defined by membership function, can be compared in different ways. Although the primary intention of comparing is to express the extent to which two fuzzy numbers match, it is almost impossible to come up with a single method. Instead, we can enumerate several classes of methods available today for satisfying

this objective. One class, distance measures, considers a distance function between membership functions of fuzzy sets A and B and treats it as an indicator of their closeness. Comparing fuzzy sets via distance measures does not place the matching procedure in the set-theory perspective. In general, the distance between A and B, defined in the same universe of discourse X, where $X \in R$, can be defined using the Minkowski distance

$$D(A, B) = \left\{ \sum |A(x) - B(x)|^p \right\}^{1/p}, \quad x \in X$$

where $p \geq 1$. Several specific cases are typically encountered in applications:

1. Hamming distance for $p = 1$,
2. Euclidean distance for $p = 2$, and
3. Tchebychev distance for $p = \infty$.

For example, the distance between given fuzzy sets A and B, based on Euclidian measure, is

$$D(A, B) = \sqrt{(0 - 0.9)^2 + (0.5 - 0.4)^2 + (0.8 - 0.3)^2 + (1 - 0.1)^2 + (0.2 - 0)^2} = 1.39$$

For continuous universes of discourse, summation is replaced by integration. The more similar the two fuzzy sets, the lower the distance function between them. Sometimes, it is more convenient to normalize the distance function and denote it $d_n(A, B)$ and use this version to express similarity as a straight complement, $1 - d_n(A, B)$.

The other approach to comparing fuzzy sets is the use of possibility and necessity measures. The possibility measure of fuzzy set A with respect to fuzzy set B, denoted by $Pos(A, B)$, is defined as

$$Pos(A, B) = \max[\min(A(x), B(x))], \quad x \in X$$

The necessity measure of A with respect to B, $Nec(A, B)$, is defined as

$$Nec(A, B) = \min[\max(A(x), 1 - B(x))], \quad x \in X$$

For the given fuzzy sets A and B, these alternative measures for fuzzy set comparison are

$$Pos(A, B) = \max[\min\{(0, 0.5, 0.8, 1.0, 0.2), (0.9, 0.4, 0.3, 0.1, 0)\}]$$

$$= \max[0, 0.4, 0.3, 0.1, 0] = 0.4$$

$$Nec(A, B) = \min[\max\{(0, 0.5, 0.8, 1.0, 0.2), (0.1, 0.6, 0.7, 0.9, 1.0)\}]$$

$$= \min[0.1, 0.6, 0.8, 1.0, 1.0] = 0.1$$
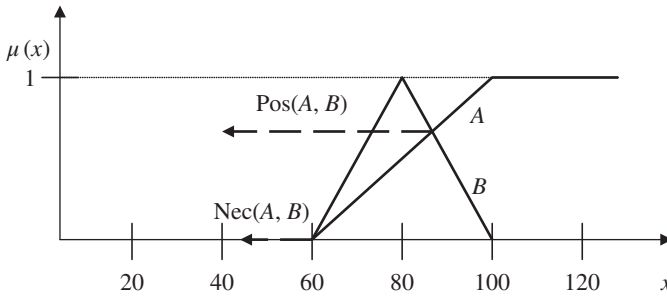
Figure 14.6. Comparison of fuzzy sets representing linguistic terms A = high speed and B = speed around 80 km/h.

An interesting interpretation arises from these measures. The possibility measure quantifies the extent to which A and B overlap. By virtue of the definition introduced, the measure is symmetric. On the other hand, the necessity measure describes the degree to which B is included in A. As seen from the definition, the measure is asymmetrical. A visualization of these two measures is given in Figure 14.6.

A number of simple yet useful operations may be performed on fuzzy sets. These are one-argument mappings, because they apply to a single membership function:

1. *Normalization*—This operation converts a subnormal, nonempty fuzzy set into a normalized version by dividing the original membership function by the height of A:

$$\text{Norm} A(x) = \{(x, \mu_A(x)/\text{hgt}(x) = \mu_A(x)/\max \mu_A(x)) \quad \text{where } x \in X\}$$

2. *Concentration*—When fuzzy sets are concentrated, their membership functions take on relatively smaller values. That is, the membership function becomes more concentrated around points with higher membership grades as, for instance, being raised to power two:

$$\text{Con} A(x) = \{(x, \mu_A^2(x)) \text{ where } x \in X\}$$

3. *Dilation*—Dilation has the opposite effect from concentration and is produced by modifying the membership function through exponential transformation, where the exponent is less than 1:

$$\text{Dil} A(x) = \{(x, \mu_A^{1/2}(x)) \text{ where } x \in X\}$$

The basic effects of the previous three operations are illustrated in Figure 14.7.

In practice, when the universe of discourse X is a continuous space (the real axis R or its subset), we usually partition X into several fuzzy sets whose membership

(a)   (b)   (c)
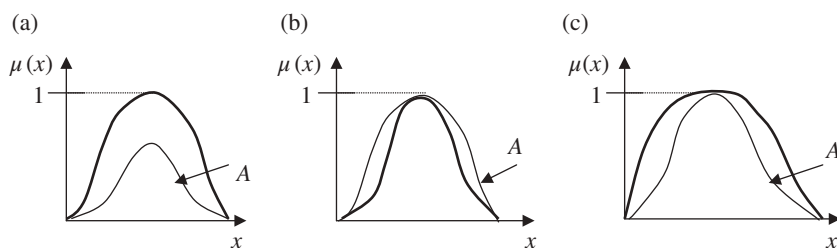
Figure 14.7. Simple unary fuzzy operations. (a) Normalization. (b) Concentration. (c) Dilation.
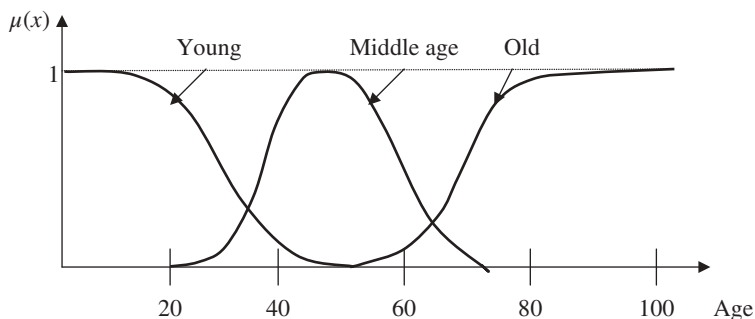
Figure 14.8. Typical membership functions for linguistic values "young," "middle aged," and "old."

functions cover X in a more or less uniform manner. These fuzzy sets, which usually carry names that conform to adjectives appearing in our daily linguistic usage, such as "large," "medium," or "small," are called *linguistic values* or linguistic labels. Thus, the universe of discourse X is often called the linguistic variable. Let us give some simple examples.

Suppose that X = "age." Then we can define fuzzy sets "young," "middle aged," and "old" that are characterized by MFs $\mu_{young}(x)$, $\mu_{middleaged}(x)$, and $\mu_{old}(x)$, respectively. Just as a variable can assume various values, a linguistic variable "age" can assume different linguistic values, such as "young," "middle aged," and "old" in this case. If "age" assumes the value of "young," then we have the expression "age is young," and so also for the other values. Typical membership functions for these linguistic values are displayed in Figure 14.8, where the universe of discourse X is totally covered by the membership functions and their smooth and gradual transition from one to another. Unary fuzzy operations, concentration and dilation, may be interpreted as linguistic modifiers "very" and "more or less," respectively.

A linguistic variable is characterized by a quintuple $(x, T(x), X, G, M)$ in which x is the name of the variable; $T(x)$ is the term set of x, the set of its linguistic values; X is the universe of discourse; G is a syntactic rule, which generates the terms in $T(x)$; and M is a semantic rule, which associates with each linguistic value A its meaning $M(A)$, where $M(A)$ denotes a membership function for a fuzzy set in X. For example, if age is interpreted as a linguistic variable, then the term set $T(age)$ could be

$$T(\text{age}) = \left\{ \begin{array}{l} \text{very young, young, not very young, not young, ...middle aged, not} \\ \text{middle aged, not old, more-or-less old, old, very old} \end{array} \right\}$$

where each term in $T(age)$ is characterized by a fuzzy set of a universe of discourse $X = [0,100]$. The syntactic rule refers to the way the linguistic values in the term set T(age) are generated, and the semantic rule defines the membership function of each linguistic value of the term set $T(age)$, such as the linguistic values in Figure 14.8.

14.3 EXTENSION PRINCIPLE AND FUZZY RELATIONS

As in the set theory, we can define several generic relations between two fuzzy sets, such as *equality* and *inclusion*. We say that two fuzzy sets, A and B, defined in the same space X are equal if and only if (iff) their membership functions are identical:

$$A = B \quad \text{iff} \quad \mu_A(x) = \mu_B(x), \quad \forall x \in X$$

Analogously, we shall define the notion of *containment*, which plays a central role in both ordinary and fuzzy sets. This definition of containment is, of course, a natural extension of the case for ordinary sets. Fuzzy set A is *contained* in fuzzy set B (or, equivalently, A is a subset of B) if and only if $\mu_A(x) \leq \mu_B(x)$ for all x. In symbols,

$$A \subseteq B \Leftrightarrow \mu_A(x) \leq \mu_B(x), \quad \forall x \in X$$

Figure 14.9 illustrates the concept of $A \subseteq B$.
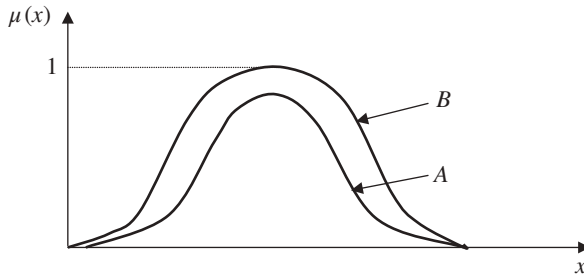
Figure 14.9. The concept of $A \subseteq B$ where A and B are fuzzy sets.

When the fuzzy sets A and B are defined in a finite universe X, and the requirement that for each x in X, $\mu_A(x) \leq \mu_B(x)$ is relaxed, we may define the degree of subsethood DS as

$$DS(A, B) = \left(\frac{1}{\text{Card}(A)} \right) \left\{ \text{Card}(A) - \sum \max[0, A(x) - B(x)] \right\}, \quad x \in X$$

DS(A, B) provides a normalized measure of the degree to which the inequality $\mu_A(x) \leq \mu_B(x)$ is violated.

Now we have enough background to explain one of the most important concepts in formalization of a fuzzy-reasoning process. The *extension principle* is a basic transformation of the fuzzy set theory that provides a general procedure for extending the crisp domains of mathematical expressions to fuzzy domains. This procedure generalizes a common point-to-point mapping of a function f between fuzzy sets. The *extension principle* plays a fundamental role in translating set-based concepts into their fuzzy counterparts. Essentially, the extension principle is used to transform fuzzy sets via functions. Let X and Y be two sets, and F is a mapping from X to Y:

$$F : X \rightarrow Y$$

Let A be a fuzzy set in X. The extension principle states that the image of A under this mapping is a fuzzy set $B = f(A)$ in Y such that for each $y \in Y$:

$$\mu_B(y) = \max \mu_A(x), \quad \text{subject to } x \in X \text{ and } y = f(x)$$

The basic idea is illustrated in Figure 14.10. The extension principle easily generalizes to functions of many variables as follows. Let X_i, $i = 1, \ldots, n$, and Y be universes of
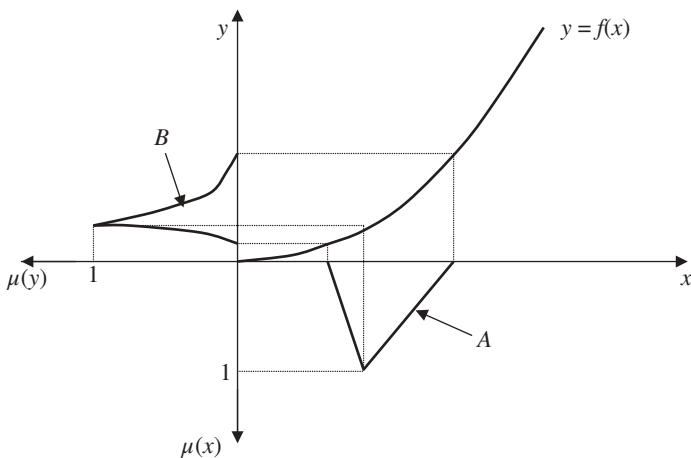
Figure 14.10. The idea of the extension principle.

discourse, and $X = X_1 \times X_2 \times \cdots \times X_n$ constitute the Cartesian product of the X_is. Consider fuzzy sets A_i in X_i, $i = 1,\ldots,n$ and a mapping $y=f(x)$, where the input is an n-dimensional vector $x = (x_1, x_2,\ldots,x_n)$ and $x \in X$. Fuzzy sets $A_1, A_2,\ldots,A_n$ are then transformed via f, producing the fuzzy set $B = f(A_1, A_2,\ldots,A_n)$ in Y such that for each $y \in Y$:

$$\mu_B(y) = \max_x \{ \min [\mu_{A1}(x_1), \mu_{A2}(x_2), \ldots, \mu_{An}(x_n)] \}$$

subject to $x \in X$ and $y = f(x)$. Actually, in the expression above, the min operator is just a choice within a family of operators called triangular norms.

More specifically, suppose that f is a function from X to Y where X and Y are discrete universes of discourse and A is a fuzzy set on X defined as

$$A = \mu_A(x_1)/x_1 + \mu_A(x_2)/x_2 + \mu_A(x_3)/x_3 + \cdots + \mu_A(x_n)/x_n$$

Then the extension principle states that the image of fuzzy set A under the mapping f can be expressed as a fuzzy set B:

$$B = f(A) = \mu_A(x_1)/y_1 + \mu_A(x_2)/y_2 + \mu_A(x_3)/y_3 + \cdots + \mu_A(x_n)/y_n$$

where $y_i = f(x_i)$, $i = 1,\ldots,n$. In other words, the fuzzy set B can be defined through the mapped values x_i using the function f.

Let us analyze the extension principle using one example. Suppose that $X = \{1, 2, 3, 4\}$ and $Y = \{1, 2, 3, 4, 5, 6\}$ are two universes of discourse and the function for transformation is $y = x + 2$. For a given fuzzy set $A = 0.1/1 + 0.2/2 + 0.7/3 + 1.0/4$ in X, it is necessary to find a corresponding fuzzy set $B(y)$ in Y using the extension principle through function $B = f(A)$. In this case, the process of computation is straightforward, and a final, transformed fuzzy set is $B = 0.1/3 + 0.2/4 + 0.7/5 + 1.0/6$.

Another problem will show that the computational process is not always a one-step process. Suppose that A is given as

$$A = 0.1/-2 + 0.4/-1 + 0.8/0 + 0.9/1 + 0.3/2$$

and the function f is

$$f(x) = x^2 - 3$$

Upon applying the extension principle, we have

$$B = 0.1/1 + 0.4/-2 + 0.8/-3 + 0.9/-2 + 0.31$$
$$= 0.8/-3 + (0.4 \vee 0.9)/-2 + (0.1 \vee 0.3)/1$$
$$= 0.8/-3 + 0.9/-2 + 0.3/1$$

where $\vee$ represents the max function. For a fuzzy set with a continuous universe of discourse X, an analogous procedure applies.

Besides being useful in the application of the extension principle, some of the unary and binary fuzzy relations are also very important in a fuzzy-reasoning process.

Binary fuzzy relations are fuzzy sets in $X \times Y$ that map each element in $X \times Y$ to a membership grade between 0 and 1. Let X and Y be two universes of discourse. Then

$$R = \{((x,y), \mu_R(x,y)) \mid (x,y) \in X \times Y\}$$

is a binary fuzzy relation in $X \times Y$. Note that $\mu_R(x, y)$ is in fact a two-dimensional (2D) membership function. For example, let $X = Y = R^+$ (the positive real axis); the fuzzy relation is given as $R =$ "y is much greater than x." The membership function of the fuzzy relation can be subjectively defined as

$$\mu_R(x,y) = \begin{cases} \dfrac{(y-x)}{(x+y+2)}, & \text{if } y > x \\ 0 & \text{if } y \le x \end{cases}$$

If X and Y are a finite set of discrete values such as $X = \{3, 4, 5\}$ and $Y = \{3, 4, 5, 6, 7\}$, then it is convenient to express the fuzzy relation R as a relation matrix:

$$R = \begin{bmatrix} 0 & 0.111 & 0.200 & 0.273 & 0.333 \\ 0 & 0 & 0.091 & 0.167 & 0.231 \\ 0 & 0 & 0 & 0.077 & 0.143 \end{bmatrix}$$

where the element at row i and column j is equal to the membership grade between the ith element of X and the jth element of Y.

Common examples of binary fuzzy relations are as follows:

1. x is close to y (x and y are numbers).
2. x depends on y (x and y are categorical data).
3. x and y look alike.
4. If x is large, then y is small.

Fuzzy relations in different product spaces can be combined through a composition operation. Different composition operations have been suggested for fuzzy relations; the best known is the max–min composition proposed by Zadeh. Let R_1 and R_2 be two fuzzy relations defined on $X \times Y$ and $Y \times Z$, respectively. The max–min composition of R_1 and R_2 is a fuzzy set defined by

$$R_1 \circ R_2 = \left\{ [(x, z), \max_y \min (\mu_{R1}(x, y), \mu_{R2}(y, z))] \mid x \in X, y \in Y, z \in Z \right\}$$

or, equivalently,

$$R_1 \circ R_2 = \vee_y [(\mu_{R1}(x, y) \wedge \mu_{R2}(y, z)]$$

with the understanding that $\vee$ and $\wedge$ represent max and min, respectively.

When R_1 and R_2 are expressed as relation matrices, the calculation of $R_1 \circ R_2$ is similar to the matrix-multiplication process, except that $\times$ and $+$ operations are replaced by $\vee$ and $\wedge$, respectively.

The following example demonstrates how to apply the max–min composition on two relations and how to interpret the resulting fuzzy relation $R_1 \circ R_2$. Let $R_1 =$ "x is relevant to y" and $R_2 =$ "y is relevant to z" be two fuzzy relations defined on $X \times Y$ and $Y \times Z$, where $X = \{1, 2, 3\}$, $Y = \{\alpha, \beta, \gamma, \delta\}$, and $Z = \{a, b\}$. Assume that R_1 and R_2 can be expressed as the following relation matrices of μ values:

$$R_1 = \begin{bmatrix} 0.1 & 0.3 & 0.5 & 0.7 \\ 0.4 & 0.2 & 0.8 & 0.9 \\ 0.6 & 0.8 & 0.3 & 0.2 \end{bmatrix}$$

$$R_2 = \begin{bmatrix} 0.9 & 0.1 \\ 0.2 & 0.3 \\ 0.5 & 0.6 \\ 0.7 & 0.2 \end{bmatrix}$$

Fuzzy relation $R_1 \circ R_2$ can be interpreted as a derived relation "x is relevant to z" based on relations R_1 and R_2. We will make a detailed max–min composition only for one element in a resulting fuzzy relation: $(x, z) = (2, a)$:

$$\mu_{R1 \circ R2}(2, a) = \max(0.4 \wedge 0.9, 0.2 \wedge 0.2, 0.8 \wedge 0.5, 0.9 \wedge 0.7)$$

$$= \max(0.4, 0.2, 0.5, 0.7)$$

$$= 0.7$$

Analogously, we can compute the other elements, and the final fuzzy matrix $R_1 \circ R_2$ will be

$$R_1 \circ R_2 = \begin{bmatrix} 0.7 & 0.5 \\ 0.7 & 0.6 \\ 0.6 & 0.3 \end{bmatrix}$$

14.4 FUZZY LOGIC AND FUZZY INFERENCE SYSTEMS

Fuzzy logic enables us to handle uncertainty in a very intuitive and natural manner. In addition to making it possible to formalize imprecise data, it also enables us to do arithmetic and Boolean operations using fuzzy sets. Finally, it describes the inference

systems based on fuzzy rules. Fuzzy rules and fuzzy-reasoning processes, which are the most important modeling tools based on the fuzzy set theory, are the backbone of any fuzzy inference system. Typically, a fuzzy rule has the general format of a conditional proposition. A *fuzzy If-then rule*, also known as *fuzzy implication*, assumes the form

<p style="text-align:center">If x is A, then y is B</p>

where *A* and *B* are linguistic values defined by fuzzy sets on the universes of discourse *X* and *Y*, respectively. Often, "*x* is *A*" is called the antecedent or premise, while "*y* is *B*' is called the consequence or conclusion. Examples of fuzzy If-then rules are widespread in our daily linguistic expressions, such as the following:

1. If pressure is high, then volume is small.
2. If the road is slippery, then driving is dangerous.
3. If a tomato is red, then it is ripe.
4. If the speed is high, then apply the brake a little.

Before we can employ fuzzy If-then rules to model and analyze a fuzzy-reasoning process, we have to formalize the meaning of the expression "If *x* is *A* then *y* is *B*," sometimes abbreviated in a formal presentation as $A \rightarrow B$. In essence, the expression describes a relation between two variables *x* and *y*; this suggests that a fuzzy If-then rule be defined as a binary fuzzy relation *R* on the product space $X \times Y$. *R* can be viewed as a fuzzy set with a 2D membership function:

$$\mu_R(x, y) = f(\mu_A(x), \mu_B(y))$$

If we interpret $A \rightarrow B$ as *A entails B*, still it can be formalized in several different ways. One formula that could be applied based on a standard logical interpretation is

$$R = A \rightarrow B = A' \cup B.$$

Note that this is only one of several possible interpretations for fuzzy implication. The accepted meaning of $A \rightarrow B$ represents the basis for an explanation of the fuzzy-reasoning process using If-then fuzzy rules.

Fuzzy reasoning, also known as approximate reasoning, is an inference procedure that derives its conclusions from a set of fuzzy rules and known facts (they also can be fuzzy sets). The basic rule of inference in a traditional two-valued logic is *modus ponens*, according to which we can infer the truth of a proposition B from the truth of A and the implication $A \rightarrow B$. However, in much of human reasoning, *modus ponens* is employed in an approximate manner. For example, if we have the rule "if the tomato is red, then it is ripe" and we know that "the tomato is more or less red," then we may infer that "the tomato is more or less ripe." This type of approximate reasoning can be formalized as

Fact :	x is A'
Rule :	If x is A, then y is B
Conclusion :	y is B'

where A' is close to A and B' is close to B. When A, A', B, and B' are fuzzy sets of an approximate universe, the foregoing inference procedure is called approximate reasoning or fuzzy reasoning; it is also called *generalized modus ponens*, since it has *modus ponens* as a special case.

Using the composition rule of inference, we can formulate the inference procedure of fuzzy reasoning. Let A, A', and B be fuzzy sets on X, X, and Y domains, respectively. Assume that the fuzzy implication $A \to B$ is expressed as a fuzzy relation R on $X \times Y$. Then the fuzzy set B' induced by A' and $A \to B$ is defined by

$$\mu_{B'}(y) = \max_x \min\left[\mu_{A'}(x), \mu_R(x, y)\right]$$

$$= \vee_x\left[\mu_{A'}(x) \wedge \mu_R(x, y)\right]$$

Some typical characteristics of the fuzzy-reasoning process and some conclusions useful for this type of reasoning are as follows:

1. $\forall A, \forall A' \to B' \supseteq B \left(\text{or}\, \mu_{B'}(y) \geq \mu_B(y)\right)$
2. If $A' \subseteq A \left(\text{or}\, \mu_A(x) \geq \mu_{A'}(x)\right) \to B' = B$

Let us analyze the computational steps of a fuzzy-reasoning process for one simple example. Given the fact $A' =$ "x is above average height" and the fuzzy rule "if x is high, then his/her weight is also high," we can formalize this as a fuzzy implication $A \to B$. We can use a discrete representation of the initially given fuzzy sets A, A', and B (based on subjective heuristics):

A':	x	$\mu(x)$	A:	x	$\mu(x)$	B:	y	$\mu(y)$
	5′6″	0.3		5′6″	0		120	0
	5′9″	1.0		5′9″	0.2		150	0.2
	6′	0.4		6′	0.8		180	0.5
	6′3″	0		6′3″	1.0		210	1.0

$\mu_R(x, y)$ can be computed in several different ways, such as

$$\mu_R(x,y) = \begin{cases} 1 & \text{for } \mu_A(x) \leq \mu_B(y) \\ \mu_B(y) & \text{otherwise} \end{cases}$$

or as the Lukasiewicz-norm

$$\mu_R(x,y) = \{1 \wedge (1 - \mu_A(x) + \mu_B(y))\}$$

Both definitions lead to a very different interpretation of fuzzy implication. Applying the first relation for $\mu_R(x, y)$ on the numeric representation for our sets A and B, the 2D membership function will be

$$\mu_R(x,y) = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 0.2 & 0.5 & 1 \\ 0 & 0.2 & 0.5 & 1 \end{bmatrix}$$

Now, using the basic relation for inference procedure, we obtain

$$\mu_{B'}(y) = \max_x \min\left[\mu_{A'}(x), \mu_R(x,y)\right]$$

$$= \max_x \min\left[0.3 \ 1 \ 0.4 \ 0\right] \cdot \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 0.2 & 0.5 & 1 \\ 0 & 0.2 & 0.5 & 1 \end{bmatrix}$$

$$= \max_x \begin{bmatrix} 0.3 & 0.3 & 0.3 & 0.3 \\ 0 & 1 & 1 & 1 \\ 0 & 0.2 & 0.4 & 0.4 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$= \begin{bmatrix} 0.3 & 1 & 1 & 1 \end{bmatrix}$$

The resulting fuzzy set B' can be represented in the form of a table:

B'	y	$\mu(y)$
	120	0.3
	150	1.0
	180	1.0
	210	1.0

Or it can be interpreted approximately in linguistic terms: "x's weight is more-or-less high." A graphical comparison of membership functions for fuzzy sets A, A', B, and B' is given in Figure 14.11.

To use fuzzy sets in approximate reasoning (a set of linguistic values with numeric representations of membership functions), the main tasks for the designer of a system are to:

1. represent any fuzzy data, given as a linguistic value, in terms of the code-book A,
2. use these coded values for different communication and processing steps, and
3. at the end of approximate reasoning, transform the computed results back into its original (linguistic) format using the same codebook A.

These three fundamental tasks are commonly referred to as encoding, transmission and processing, and decoding (the terms have been borrowed from communication theory). The encoding activities occur at the transmitter while the decoding take place at the receiver. Figure 14.12 illustrates encoding and decoding with the use of
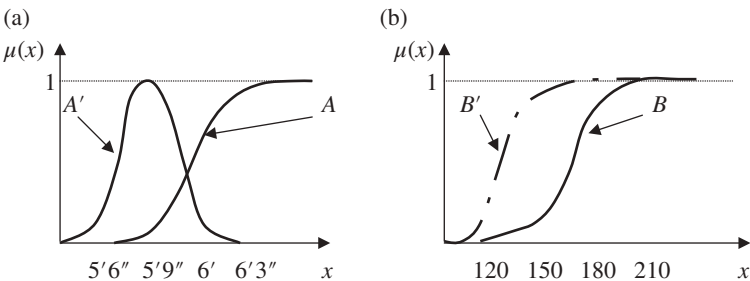
Figure 14.11. Comparison of approximate reasoning result B' with initially given fuzzy sets A', A, and B and the fuzzy rule $A \rightarrow B$. (a) Fuzzy sets A and A'. (b) Fuzzy sets B and B' (conclusion).
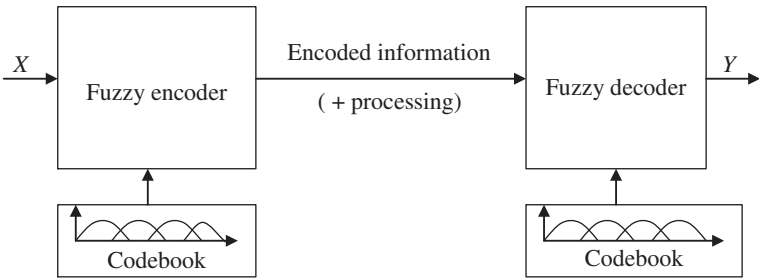
Figure 14.12. Fuzzy communication channel with fuzzy encoding and decoding.

the codebook A. Any input information, whatever its nature, is encoded (represented) in terms of the elements of the codebook. In this internal format, encoded information is sent with or without processing across the channel. Using the same codebook, the output message is decoded at the receiver.

Fuzzy set literature has traditionally used the terms fuzzification and defuzzification to denote encoding and decoding, respectively. These are, unfortunately, quite misleading and meaningless terms because they mask the very nature of the processing that takes place in fuzzy reasoning. They neither address any design criteria nor introduce any measures aimed at characterizing the quality of encoding and decoding information completed by the fuzzy channel.

The next two sections are examples of the application of fuzzy logic and fuzzy reasoning to decision-making processes, where the available data sets are ambiguous. These applications include multifactorial evaluation and extraction of fuzzy rules-based models from large numeric data sets.

14.5 MULTIFACTORIAL EVALUATION

Multifactorial evaluation is a good example of the application of the fuzzy set theory to decision-making processes. Its purpose is to provide a synthetic evaluation of an object relative to an objective in a fuzzy decision environment that has many factors. Let $U = \{u_1, u_2, \ldots, u_n\}$ be a set of objects for evaluation, let $F = \{f_1, f_2, \ldots, f_m\}$ be the set of basic factors in the evaluation process, and let $E = \{e_1, e_2, \ldots, e_p\}$ be a set of descriptive grades or qualitative classes used in the evaluation. For every object $u \in U$, there is a single-factor evaluation matrix $R(u)$ with dimensions $m \times p$, which is usually the result of a survey. This matrix may be interpreted and used as a 2D membership function for fuzzy relation $F \times E$.

With the preceding three elements, F, E, and R, the evaluation result $D(u)$ for a given object $u \in U$ can be derived using the basic fuzzy-processing procedure: the product of fuzzy relations through max–min composition. This has been shown in Figure 14.13. An additional input to the process is the weight vector $W(u)$ for evaluation factors, which can be viewed as a fuzzy set for a given input u. A detailed explanation of the computational steps in the multifactorial-evaluation process will be given through two examples.
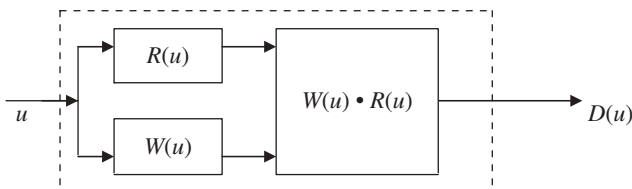
Figure 14.13. Multifactorial-evaluation model.

14.5.1   A Cloth-Selection Problem

Assume that the basic factors of interest in the selection of cloth consist of f_1 = style, f_2 = quality, and f_3 = price, i.e., $F = \{f_1, f_2, f_3\}$. The verbal grades used for the selection are e_1 = best, e_2 = good, e_3 = fair, and e_4 = poor, i.e., $E = \{e_1, e_2, e_3, e_4\}$. For a particular piece of cloth u, the single-factor evaluation may be carried out by professionals or customers by a survey. For example, if the survey results of the "style" factor f_1 are 60% for the best, 20% for the good, 10% for the fair, and 10% for the poor, then the single-factor evaluation vector $R_1(u)$ is

$$R_1(u) = \{0.6, 0.2, 0.1, 0.1\}$$

Similarly, we can obtain the following single-factor evaluation vectors for f_2 and f_3:

$$R_2(u) = \{0.1, 0.5, 0.3, 0.1\}$$
$$R_3(u) = \{0.1, 0.3, 0.4, 0.2\}$$

Based on single-factor evaluations, we can build the following evaluation matrix:

$$R(u) = \begin{bmatrix} R_1(u) \\ R_2(u) \\ R_3(u) \end{bmatrix} = \begin{bmatrix} 0.6 & 0.2 & 0.1 & 0.1 \\ 0.1 & 0.5 & 0.3 & 0.1 \\ 0.1 & 0.3 & 0.4 & 0.2 \end{bmatrix}$$

If a customer's weight vector with respect to the three factors is

$$W(u) = \{0.4, 0.4, 0.2\}$$

then it is possible to apply the multifactorial-evaluation model to compute the evaluation for a piece of cloth u. "Multiplication" of matrices $W(u)$ and $R(u)$ is based on the max–min composition of fuzzy relations, where the resulting evaluation is in the form of a fuzzy set $D(u) = [d_1, d_2, d_3, d_4]$:

$$D(u) = W(u) \cdot R(u) = \begin{bmatrix} 0.4 & 0.4 & 0.2 \end{bmatrix} \cdot \begin{bmatrix} 0.6 & 0.2 & 0.1 & 0.1 \\ 0.1 & 0.5 & 0.3 & 0.1 \\ 0.1 & 0.3 & 0.4 & 0.2 \end{bmatrix}$$

$$= \begin{bmatrix} 0.4 & 0.4 & 0.3 & 0.2 \end{bmatrix}$$

where, for example, d_1 is calculated through the following steps:

$$d_1 = (w_1 \wedge r_{11}) \vee (w_2 \wedge r_{21}) \vee (w_3 \wedge r_{31})$$
$$= (0.4 \wedge 0.6) \vee (0.4 \wedge 0.1) \vee (0.2 \wedge 0.1)$$
$$= 0.4 \vee 0.1 \vee 0.1$$
$$= 0.4$$

The values for d_2, d_3, and d_4 are found similarly, where $\wedge$ and $\vee$ represent the operations min and max, respectively. Because the largest components of $D(u)$ are $d_1 = 0.4$ and $d_2 = 0.4$ at the same time, the analyzed piece of cloth receives a rating somewhere between "best" and "good."

14.5.2 A Problem of Evaluating Teaching

Assume that the basic factors that influence students' evaluation of teaching are f_1 = clarity and understandability, f_2 = proficiency in teaching, f_3 = liveliness and stimulation, and f_4 = writing neatness or clarity, i.e., $F = \{f_1, f_2, f_3, f_4\}$. Let $E = \{e_1, e_2, e_3, e_4\}$ = {excellent, very good, good, poor} be the verbal grade set. We evaluate a teacher u. By selecting an appropriate group of students and faculty, we can have them respond with their ratings on each factor and then obtain the single-factor evaluation. As in the previous example, we can combine the single-factor evaluation into an evaluation matrix. Suppose that the final matrix $R(u)$ is

$$R(u) = \begin{bmatrix} 0.7 & 0.2 & 0.1 & 0.0 \\ 0.6 & 0.3 & 0.1 & 0.0 \\ 0.2 & 0.6 & 0.1 & 0.1 \\ 0.1 & 0.1 & 0.6 & 0.2 \end{bmatrix}$$

For a specific weight vector $W(u) = \{0.2, 0.3, 0.4, 0.1\}$, describing the importance of the teaching-evaluation factor f_i and using the multifactorial-evaluation model, it is easy to find

$$D(u) = W(u) \cdot R(u) = [0.2 \quad 0.3 \quad 0.4 \quad 0.1] \cdot \begin{bmatrix} 0.7 & 0.2 & 0.1 & 0.0 \\ 0.6 & 0.3 & 0.1 & 0.0 \\ 0.2 & 0.6 & 0.1 & 0.1 \\ 0.1 & 0.1 & 0.6 & 0.2 \end{bmatrix}$$

$$= [0.2 \quad 0.4 \quad 0.1 \quad 0.1]$$

Analyzing the evaluation results $D(u)$, because $d_2 = 0.4$ is a maximum, we may conclude that teacher u should be rated as "very good."

14.6 EXTRACTING FUZZY MODELS FROM DATA

In the context of different data-mining analyses, it is of great interest to see how fuzzy models can automatically be derived from a data set. Besides prediction, classification, and all other data-mining tasks, understandability is of prime concern, because the resulting fuzzy model should offer an insight into the underlying system. To achieve this goal, different approaches exist. Let us explain a common technique that constructs grid-based rule sets using a global granulation of the input and output spaces.

Grid-based rule sets model each input variable usually through a small set of linguistic values. The resulting rule base uses all or a subset of all possible combinations of these linguistic values for each variable, resulting in a global granulation of the feature space into rectangular regions. Figure 14.14 illustrates this approach in two dimensions: with three linguistic values (low, medium, high) for the first dimension x_1 and two linguistic values for the second dimension x_2 (young, old).

Extracting grid-based fuzzy models from data is straightforward when the input granulation is fixed, i.e., the antecedents of all rules are predefined. Then, only a matching consequent for each rule needs to be found. This approach, with fixed grids, is usually called the *Mamdani model*. After predefinition of the granulation of all input variables and also the output variable, one sweeps through the entire data set and determines the closest example to the geometrical center of each rule,
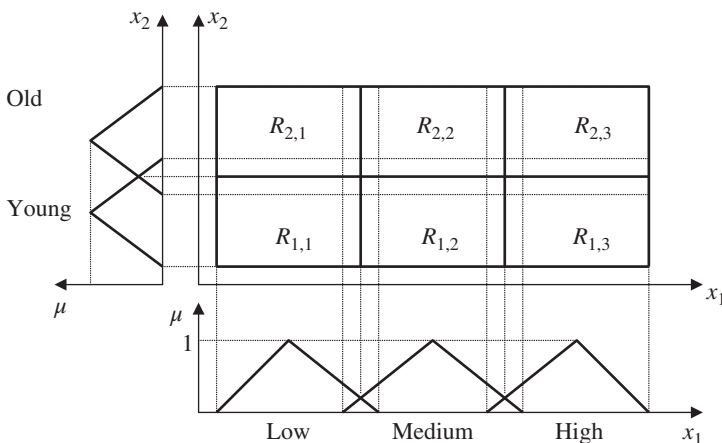
Figure 14.14. A global granulation for a two-dimensional space using three membership functions for x_1 and two for x_2.

assigning the closest fuzzy value output to the corresponding rule. Using graphical interpretation in a 2D space, the global steps of the procedure are illustrated through an example in which only one input x and one output dimension y exist. The formal analytical specification, even with more than one input/output dimension, is very easy to establish:

1. *Granulate the input and output space*. Divide each variable x_i into n_i equidistant, triangular membership functions. In our example, both input x and output y are granulated using the same four linguistic values: low, below average, above average, and high. A representation of the input–output granulated space is given in Figure 14.15.

2. *Analyze the entire data set in the granulated space*. First, enter a data set in the granulated space, and then find the points that lie closest to the centers of the granulated regions. Mark these points and the centers of the region. In our example, after entering all discrete data, the selected center points (closest to the data) are additionally marked with **x,** as in Figure 14.16.

3. *Generate fuzzy rules from given data*. Data representative directly selects the regions in a granulated space. These regions may be described with the corresponding fuzzy rules. In our example, four regions are selected, one for each fuzzy input linguistic value, and they are represented in Figure 14.17 with a corresponding crisp approximation (a thick line through the middle of the regions). These regions are the graphical representation of fuzzy rules. The same rules may be expressed linguistically as a set of IF-THEN constructions:
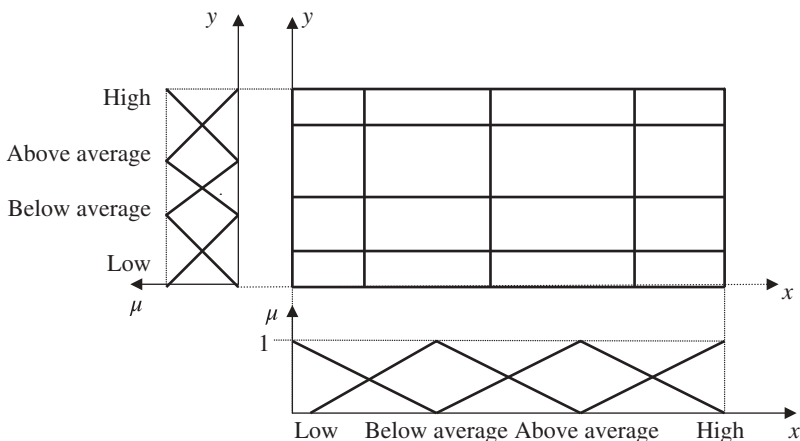
Figure 14.15. Granulation of a two-dimensional I/O space.
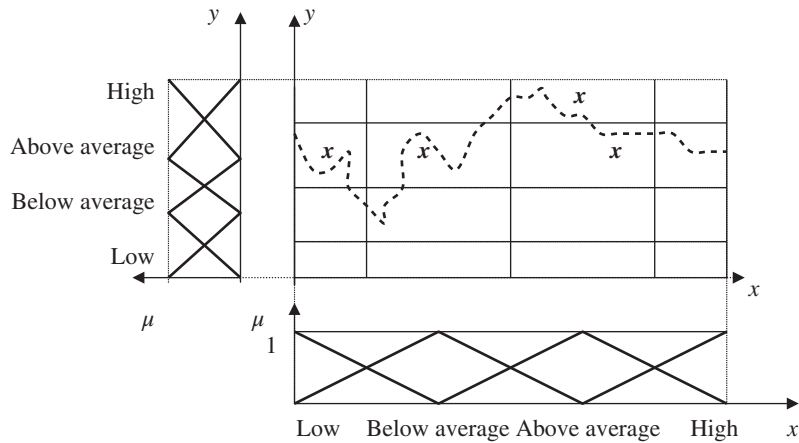
Figure 14.16. Selection of characteristic points in a granulated space.
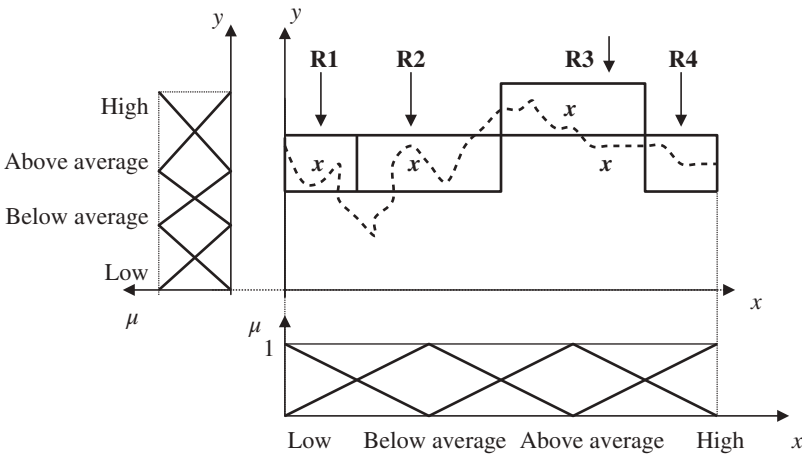
Figure 14.17. Graphical representation of generated fuzzy rules and the resulting crisp approximation.

R_1 :   IF x is *small*   THEN y is *above average*.

R_2 :   IF x is *below average* THEN y is *above average*.

R_3 :   IF x is *above average* THEN y is *high*.

R_4 :   IF x is *high*   THEN y is *above average*.

Note how the generated model misses the extremes that lie far from the existing rule centers. This behavior occurs because only one pattern per rule is used to

determine the outcome of this rule. Even a combined approach would very much depend on the predefined granulation. If the function to be modeled has a high variance inside one rule, the resulting fuzzy rule model will fail to model this behavior.

For practical applications it is obvious, however, that using such a predefined, fixed grid results in a fuzzy model that will either not fit the underlying functions very well or consist of a large number of rules because of small granulation. Therefore, new approaches have been introduced that automatically determine the granulations of both input and output variables based on a given data set. We will explain the basic steps for one of these algorithms using the same data set from the previous example and the graphical representation of applied procedures:

1. Initially, only one membership function is used to model each of the input variables as well as the output variable, resulting in one large rule covering the entire feature space. Subsequently, new membership functions are introduced at points of maximum error (the maximum distance between data points and the obtained crisp approximation). Figure 14.18 illustrates this first step in which the crisp approximation is represented with a thick line and the selected point of maximal error with a triangle.

2. For the selected point of maximum error, new triangular fuzzy values for both input and output variables are introduced. Processes of granulation, determining fuzzy rules in the form of space regions, and crisp approximation are repeated for a space, with additional input and output fuzzy values for the second step—that means two fuzzy values for both input and output variables. The final results of the second step, for our example, are presented in Figure 14.19.
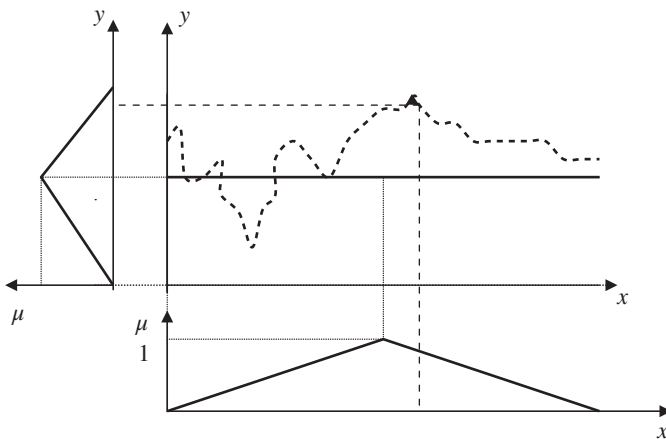
Figure 14.18. The first step in automatically determining fuzzy granulation.
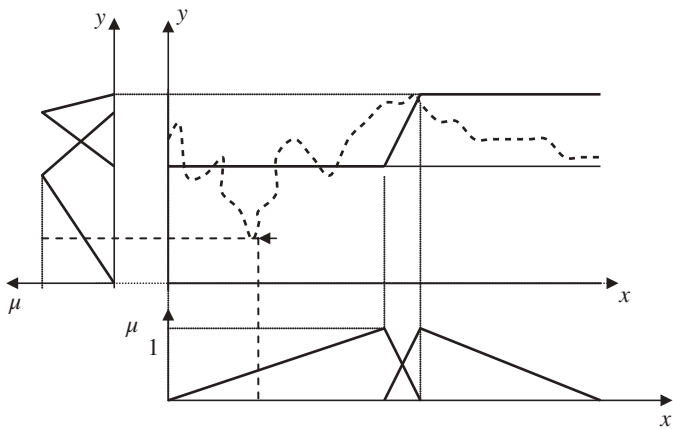
Figure 14.19. The second step (first iteration) in automatically determining granulation.
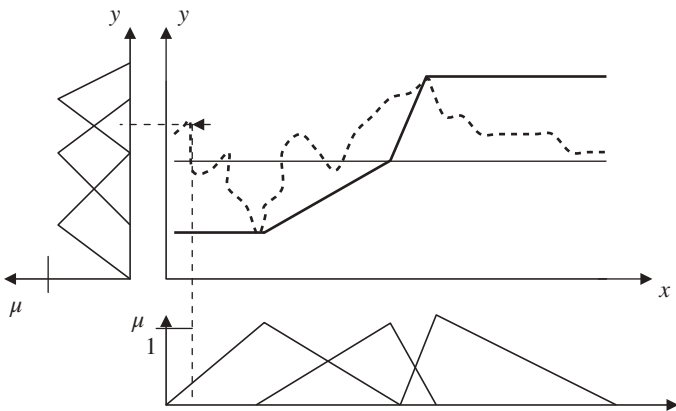
Figure 14.20. The second step (second iteration) in automatically determining granulation.

3. Step 2 is repeated until a maximum number of divisions (fuzzy values) is reached, or the approximation error remains below a certain threshold value. Figures 14.20 and 14.21 demonstrate two additional iterations of the algo-rithm for a data set. Here granulation was stopped after a maximum of four membership functions was generated for each variable. Obviously this algo-rithm is able to model extremes much better than the previous one with a fixed granulation. At the same time, it has a strong tendency to favor extremes and to concentrate on outliers. The final set of fuzzy rules, using dynamically created fuzzy values A_x to D_x and A_y to D_y for input and output variables, is
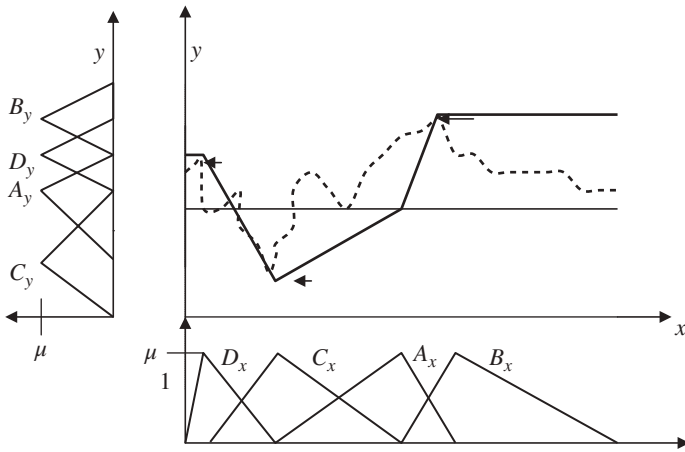
Figure 14.21. The second step (third iteration) in automatically determining granulation.

$$
\begin{array}{lll}
R_1: & \text{IF } x \text{ is } A_x & \text{THEN } y \text{ is } A_y. \\
R_2: & \text{IF } x \text{ is } B_x & \text{THEN } y \text{ is } B_y. \\
R_3: & \text{IF } x \text{ is } C_x & \text{THEN } y \text{ is } C_y. \\
R_4: & \text{IF } x \text{ is } D_x & \text{THEN } y \text{ is } D_y.
\end{array}
$$

14.7 DATA MINING AND FUZZY SETS

There is a growing indisputable role of fuzzy set technology in the realm of data mining. In a data-mining process, discovered models, learned concepts, or patterns of interest are often vague and have non-sharp boundaries. Unfortunately, the representation of graduality is often foiled in data-mining applications, especially in connection with the learning of predictive models. For example, the fact that neural networks are often used as data-mining methods, although their learning result (weight matrices of numbers) is hardly interpretable, shows that in contrast to the standard definition, the goal of understandable models is often neglected. In fact, one should recognize that graduality is not only advantageous for expressing concepts and patterns but also for modeling the qualifying properties and relations. Of course, correctness, completeness, and efficiency are important in data-mining models, but in order to manage systems that are more and more complex, there is a constantly growing demand to keep the solutions conceptually simple and understandable. Modern technologies are accepted more readily, if the methods applied and models derived are easy to understand, and the results can be checked against human intuition.

The complexity of the learning task, obviously, leads to a problem: when learning from information, one must choose between mostly quantitative methods that achieve

good performances and qualitative models that explain to a user what is going on in the complex system. Fuzzy set theory has the potential to produce models that are more comprehensible, less complex, and more robust. Fuzzy information granulation appears to be appropriate approach for trading off accuracy against complexity and understandability of data-mining models. Also, fuzzy set theory in conjunction with possibility theory can contribute considerably to the modeling and processing of various forms of uncertain and incomplete information available in large real-world systems.

The tools and technologies that have been developed in fuzzy set theory have the potential to support all of the steps that comprise a process of knowledge discovery. Fuzzy methods appear to be particularly useful for data-preprocessing and data-postprocessing phases of a data-mining process. In particular, it has already been used in the data selection phase, e.g., for modeling vague data in terms of fuzzy sets, to "condense" several crisp observations into a single fuzzy one, or to create fuzzy summaries of the data.

Standard methods of data mining can be extended to include fuzzy set representation in a rather generic way. Achieving focus is important in data mining because there are too many attributes and values to be considered and can result in combinatorial explosion. Most unsupervised data-mining approaches try to achieve focus by recognizing the most interesting structures and their features even if there is still some level of ambiguity. For example, in standard clustering, each sample is assigned to one cluster in a unique way. Consequently, the individual clusters are separated by sharp boundaries. In practice, such boundaries are often not very natural or even counterintuitive. Rather, the boundary of single clusters and the transition between different clusters are usually "smooth" rather than abrupt. This is the main motivation underlying fuzzy extensions to clustering algorithms. In fuzzy clustering an object may belong to different clusters at the same time, at least to some extent, and the degree to which it belongs to a particular cluster is expressed in terms of a membership degree.

The most frequent application of fuzzy set theory in data mining is related the adaptation of rule-based predictive models. This is hardly surprising, since rule-based models have always been a cornerstone of fuzzy systems and a central aspect of research in the field. Set of fuzzy rules can represent both classification and regression models. Instead of dividing quantitative attributes into fixed intervals, they employ linguistic terms to represent the revealed regularities. Therefore, no user-supplied thresholds are required, and quantitative values can be directly inferred from the rules. The linguistic representation leads to the discovery of natural and more understandable rules.

Decision-tree induction includes well-known algorithms such as ID3, C4.5, C5.0, and CART. Fuzzy variants of decision-tree induction have been developed for quite a while and seem to remain a topic of interest even today. In fact, these approaches provide a typical example for the "fuzzification" of standard predictive methods. In the case of decision trees, it is primarily the "crisp" thresholds used for defining splitting attributes, such as size > 181 at inner nodes. Such thresholds lead to hard decision

boundaries in the input space, which means that a slight variation of an attribute (e.g. size = 182 instead of size = 181) can entail a completely different classification of a sample. Usually, a decision in favor of one particular class label has to be made, even if the sample under consideration seems to have partial membership in several classes simultaneously. Moreover, the learning process becomes unstable in the sense that a slight variation of the training samples can change the induced decision tree drastically. In order to make the decision boundaries "soft," an obvious idea is to apply fuzzy predicates at the nodes of a decision tree, such as size = *LARGE*, where *LARGE* is a fuzzy set. In that case the sample is not assigned to exactly one successor node in a unique way, but perhaps to several successors with a certain degree. Also, for fuzzy classification solutions, the consequent of single rules is usually a class assignment represented with a singleton fuzzy set. Evaluating a rule-based model thus becomes trivial and simply amounts to "maximum matching," that is, searching the maximally supporting rule for each class.

Especially important trend, in the field of fuzzy systems, are *hybrid methods* that combine fuzzy set theory with other methodologies such as neural networks. In the *neurofuzzy* methods the main idea is to encode a fuzzy system in a neural network and to apply standard approaches like backpropagation in order to train such a network. This way, *neurofuzzy* systems combine the representational advantages of fuzzy systems with the flexibility and adaptivity of neural networks. Interpretations of fuzzy membership include similarity, preference, and uncertainty. A primary motivation was to provide an interface between a numerical scale and a symbolic scale, which is usually composed of linguistic terms. Thus, fuzzy sets have the capability to interface quantitative data with qualitative knowledge structures expressed in terms of natural language. In general, due to their closeness to human reasoning, solutions obtained using fuzzy approaches are easy to understand and to apply. This provides the user with comprehensive information and often data summarization for grasping the essence of discovery from a large amount of information in a complex system.

14.8 REVIEW QUESTIONS AND PROBLEMS

1. Find some examples of fuzzy variables in daily life.

2. Show graphically and explain why the law of contradiction is violated in the fuzzy set theory.

3. The membership function of a fuzzy set is defined as

$$\mu_A(x) = \begin{cases} 1 & \text{for } 0 < x < 20 \\ \dfrac{(50 - x)}{30} & \text{for } 20 \le x < 50 \\ 0 & \text{for } x \ge 50 \end{cases}$$

(a) What will be linguistic description of the fuzzy set A if x is the variable "age" in years?

(b) Give an analytical description for $\mu_B(x)$ if B is a fuzzy set "age is close to 60 years."

4. Assume you were told that the room temperature is around $70\,°F$. How you would represent this information:

(a) by a set notation.

(b) by a fuzzy set notation.

5. Consider the fuzzy sets A, B, and C defined on the interval $x = [0, 10]$ with corresponding μ functions:

$$\mu_A(x) = \frac{x}{(x+2)} \quad \mu_B(x) = 2^{-x} \quad \mu_C(x) = \begin{cases} \dfrac{x^2}{24} & \text{for } x \in [0, 4.89] \\ 1 & \text{otherwise} \end{cases}$$

Determine analytically and graphically:

(a) A' and B'.

(b) $A \cup C$ and $A \cup B$.

(c) $A \cap C$ and $A \cap B$.

(d) $A \cup B \cup C$.

(e) $A \cap C'$.

(f) Calculate the α-cuts for A, B, and C if $\alpha = 0.2$, $\alpha = 0.5$, and $\alpha = 1$.

6. Consider two fuzzy sets with triangular membership functions $A(x, 1, 2, 3)$ and $B(x, 2, 2, 4)$. Find their intersection and union graphically, and express them analytically using the min and max operators.

7. If $X = \{3, 4, 5\}$ and $Y = \{3, 4, 5, 6, 7\}$, and the binary fuzzy relation $R =$ "Y is much greater than X" is defined by the analytical membership function

$$\mu_R(X, Y) = \begin{cases} \dfrac{(Y-X)}{(X+Y+2)} & \text{if } Y > X \\ 0 & \text{if } Y \leq X \end{cases}$$

what will be corresponding relation matrix of R (for all discrete X and Y values)?

8. Apply the extension principle to the fuzzy set

$$A = 0.1/-2 + 0.4/-1 + 0.8/0 + 0.9/1 + 0.3/2$$

where the mapping function $f(x) = x^2 - 3$.

(a) What is the resulting image B where $B = f(A)$?

(b) Sketch this transformation graphically.

9. Assume that the proposition "if x is A then y is B" is given where A and B are fuzzy sets:

$$A = 0.5/x_1 + 1/x_2 + 0.6/x_3$$
$$B = 1/y_1 + 0.4/y_2$$

Given a fact expressed by the proposition "x is $A*$," where

$$A^* = 0.6/x_1 + 0.9/x_2 + 0.7/x_3$$

derive the conclusion in the form "y is $B*$" using the generalized *modus ponens* inference rule.

10. Solve problem #9 by using

$$A = 0.6/x_1 + 1/x_2 + 0.9/x_3$$
$$B = 0.6/y_1 + 1/y_2$$
$$A* = 0.5/x_1 + 0.9/x_2 + 1/x_3$$

11. The test scores for the three students are given in the following table:

	Math	Physics	Chemistry	Language
Henry	66	91	95	83
Lucy	91	88	80	73
John	80	88	80	78

Find the best student using multifactorial evaluation, if the weight factors for the subjects are given as the vector $W = [0.3, 0.2, 0.1, 0.4]$.

12. Search the Web to find basic characteristics of publicly available or commercial software tools that are based on fuzzy sets and fuzzy logic. Make a report of your search.

14.9   REFERENCES FOR FURTHER STUDY

1. Pedrycz, W. and F. Gomide, *An Introduction to Fuzzy Sets: Analysis and Design*, MIT Press, Cambridge, 1998.

The book provides a highly readable, comprehensive, self-contained, updated, and well-organized presentation of the fuzzy set technology. Both theoretical and practical aspects of the subject are given a coherent and balanced treatment. The reader is introduced to the main computational models, such as fuzzy modeling and rule-based computation, and to the frontiers of the field at the confluence of fuzzy set technology with other major methodologies of soft computing.

2. Chen Y., Wang T., Wang B., and Li Z., A survey of fuzzy decision-tree classifier, *Fuzzy Information and Engineering*, Vol. 1, No. 2, June 2009, pp. 149–159.

Decision-tree algorithm provides one of the most popular methodologies for symbolic knowledge acquisition. The resulting knowledge, a symbolic decision tree along with a simple inference mechanism, has been praised for comprehensibility. The most comprehensible decision trees have been designed for perfect symbolic data. Over the years, additional methodologies have been investigated and proposed to deal with continuous or multivalued data and with missing or noisy features. Recently, with the growing popularity of fuzzy representation, some researchers have proposed to utilize fuzzy representation in decision trees to deal with similar situations. This paper presents a survey of current methods for fuzzy decision-tree (FDT) design and the various existing issues. After considering potential advantages of FDT classifiers over traditional decision-tree classifiers, we discuss the subjects of FDT including attribute selection criteria, inference for decision assignment, and stopping criteria.

3. Cox E., *Fuzzy Modeling and Genetic Algorithms for Data Mining and Exploration*, Morgan Kaufmann, 2005.

Fuzzy Modeling and Genetic Algorithms for Data Mining and Exploration is a handbook for analysts, engineers, and managers involved in developing data-mining models in business and government. As you will discover, fuzzy systems are extraordinarily valuable tools for representing and manipulating all kinds of data, and genetic algorithms and evolutionary programming techniques drawn from biology provide the most effective means for designing and tuning these systems. You do not need a background in fuzzy modeling or genetic algorithms to benefit, for this book provides it, along with detailed instruction in methods that you can immediately put to work in your own projects. The author provides many diverse examples and also an extended example in which evolutionary strategies are used to create a complex scheduling system.

4. Laurent A., Lesot M. ed., *Scalable Fuzzy Algorithms for Data Management and Analysis, Methods and Design*, IGI Global, 2010.

The book presents innovative, cutting-edge fuzzy techniques that highlight the relevance of fuzziness for huge data sets in the perspective of scalability issues, from both a theoretical and experimental point of view. It covers a wide scope of research areas including data representation, structuring and querying, and information retrieval and data mining. It encompasses different forms of databases, including data warehouses, data cubes, tabular or relational data, and many

applications among which are music warehouses, video mining, bioinformatics, semantic Web, and data streams.

5. Chen, G., Liu, F., Shojafar, M., ed., *Fuzzy System and Data Mining*, IOS Press, April 2016.

Fuzzy logic is widely used in machine control. The term "fuzzy" refers to the fact that the logic involved can deal with concepts that cannot be expressed as either "true" or "false," but rather as "partially true." Fuzzy set theory is very suitable for modeling the uncertain duration in process simulation, as well as defining the fuzzy goals and fuzzy constraints of decision-making. It has many applications in industry, engineering and social sciences. This book presents the proceedings of the 2015 International Conference on Fuzzy System and Data Mining (FSDM), held in Shanghai, China. The application domain covers geography, biology, economics, medicine, the energy industry, social science, logistics, transport, industrial and production engineering, and computer science. The papers presented at the conference focus on topics such as system diagnosis, rule induction, process simulation/control, and decision-making.

15

VISUALIZATION METHODS

Chapter Objectives

- Recognize the importance of a visual-perception analysis in humans to discover appropriate data-visualization techniques.
- Distinguish between scientific-visualization and information-visualization techniques.
- Understand the basic characteristics of geometric, icon-based, pixel-oriented, and hierarchical techniques in visualization of large data sets.
- Explain the methods of parallel coordinates and radial visualization for n-dimensional data sets.
- Analyze the requirements for advanced visualization systems in data mining.

How are humans capable of recognizing hundreds of faces? What is our "channel capacity" when dealing with the visual or any other of our senses? How many distinct visual icons and orientations can humans accurately perceive? It is important to factor

all these cognitive limitations when designing a visualization technique that avoids delivering ambiguous or misleading information. Categorization lays the foundation for a well-known cognitive technique: the "chunking" phenomena. How many chunks can you hang onto? That varies among people, but the typical range forms "the magical number seven, plus or minus two." The process of reorganizing large amounts of data into fewer chunks with more bits of information per chunk is known in cognitive science as "recoding." We expand our comprehension abilities by reformatting problems into multiple dimensions or sequences of chunks or by redefining the problem in a way that invokes relative judgment, followed by a second focus of attention.

15.1 PERCEPTION AND VISUALIZATION

Perception is our chief means of knowing and understanding the world; images are the mental pictures produced by this understanding. In perception as well as art, a meaningful whole is created by the relationship of the parts to each other. Our ability to see patterns in things and pull together parts into a meaningful whole is the key to perception and thought. As we view our environment, we are actually performing the enormously complex task of deriving meaning out of essentially separate and disparate sensory elements. The eye, unlike the camera, is not a mechanism for capturing images so much as it is a complex processing unit that detects changes, forms, and features and selectively prepares data for the brain to interpret. The image we perceive is a mental one, the result of gleaning what remains constant while the eye scans. As we survey our three-dimensional (3D) ambient environment, properties such as contour, texture, and regularity allow us to discriminate objects and see them as constants.

Human beings do not normally think in terms of data; they are inspired by and think in terms of images (mental pictures of a given situation), and they assimilate information more quickly and effectively as visual images than as textual or tabular forms. Human vision is still the most powerful means of sifting out irrelevant information and detecting significant patterns. The effectiveness of this process is based on a picture's submodalities (shape, color, luminance, motion, vectors, texture). They depict abstract information as a visual grammar that integrates different aspects of represented information. Visually presenting abstract information, using graphical metaphors in an immersive 2D or 3D environment, increases one's ability to assimilate many dimensions of the data in a broad and immediately comprehensible form. It converts aspects of information into experiences our senses and mind can comprehend, analyze, and act upon.

We have heard the phrase "Seeing is believing" many times, though merely seeing is not enough. When you understand what you see, seeing becomes believing. Recently, scientists discovered that seeing and understanding together enable humans to discover new knowledge with deeper insight from large amounts of data. The approach integrates the human mind's exploratory abilities with the enormous processing power of computers to form a powerful visualization environment that capitalizes on the best of both worlds. A computer-based visualization technique has to

incorporate the computer less as a tool and more as a communication medium. The power of visualization to exploit human perception offers both a challenge and an opportunity. The challenge is to avoid visualizing incorrect patterns, leading to incorrect decisions and actions. The opportunity is to use knowledge about human perception when designing visualizations. Visualization creates a feedback loop between perceptual stimuli and the user's cognition.

Visual data-mining technology builds on visual and analytical processes developed in various disciplines including scientific visualization, computer graphics, data mining, statistics, and machine learning with custom extensions that handle very large multidimensional data sets interactively. The methodologies are based on both functionality that characterizes structures and displays data and human capabilities that perceives patterns, exceptions, trends, and relationships.

15.2 SCIENTIFIC VISUALIZATION AND INFORMATION VISUALIZATION

Visualization is defined in the dictionary as "a mental image." In the field of computer graphics, the term has a much more specific meaning. Technically, visualization concerns itself with the display of behavior and, particularly, with making complex states of behavior comprehensible to the human eye. Computer visualization, in particular, is about using computer graphics and other techniques to think about more cases, more variables, and more relations. The goal is to think clearly, appropriately, with insight, and to act with conviction. Unlike presentations, visualizations are typically interactive and very often animated.

Because of the high rate of technological progress, the amount of data stored in databases increases rapidly. This proves true for traditional relational databases and complex 2D and 3D multimedia databases that store images, computer-aided design (CAD) drawings, geographic information, and molecular biology structure. Many of the applications mentioned rely on very large databases consisting of millions of data objects with several tens to a few hundred dimensions. When confronted with the complexity of data, users face tough problems: Where do I start? What looks interesting here? Have I missed anything? What are the other ways to derive the answer? Is there other data available? People think iteratively and ask ad hoc questions of complex data while looking for insights.

Computation, based on these large data sets and databases, creates content. Visualization makes computation and its content accessible to humans. Therefore, visual data mining uses visualization to augment the data-mining process. Some data-mining techniques and algorithms are difficult for decision-makers to understand and use. Visualization can make the data and the mining results more accessible, allowing comparison and verification of results. Visualization can also be used to steer the data-mining algorithm.

It is useful to develop taxonomy for data visualization, not only because it brings order to disjointed techniques but also because it clarifies and interprets ideas and

purposes behind these techniques. Taxonomy may trigger the imagination to combine existing techniques or discover a totally new technique.

Visualization techniques can be classified in a number of ways. They can be classified as to whether their focus is geometric or symbolic; whether the stimulus is 2D, 3D, or *n*-D; or whether the display is static or dynamic. Many visualization tasks involve detection of differences in data rather than a measurement of absolute values. It is the well-known Weber's law that states that the likelihood of detection is proportional to the relative change, not the absolute change, of a graphical attribute. In general, visualizations can be used to explore data, to confirm a hypothesis, or to manipulate a view.

In *exploratory visualizations*, the user does not necessarily know what he/she is looking for. This creates a dynamic scenario in which interaction is critical. The user is searching for structures or trends and is attempting to arrive at some hypothesis. In *confirmatory visualizations*, the user has a hypothesis that needs only to be tested. This scenario is more stable and predictable. System parameters are often predetermined, and visualization tools are necessary for the user to confirm or refute the hypothesis. In *manipulative (production) visualizations*, the user has a validated hypothesis and so knows exactly what is to be presented. Therefore, he focuses on refining the visualization to optimize the presentation. This type is the most stable and predictable of all visualizations.

The accepted taxonomy in this book is primarily based on different approaches in visualization caused by different types of source data. Visualization techniques are divided roughly into two classes, depending on whether physical data is involved. These two classes are *scientific visualization* and *information visualization*.

Scientific visualization focuses primarily on physical data such as the human body, the earth, molecules, and so on. Scientific visualization also deals with multidimensional data, but most of the data sets used in this field use the spatial attributes of the data for visualization purposes, e.g., computer-aided tomography (CAT) and computer-aided design (CAD). Also, many of the geographical information systems (GIS) use either the Cartesian coordinate system or some modified geographical coordinates to achieve a reasonable visualization of the data.

Information visualization focuses on abstract, nonphysical data such as text, hierarchies, and statistical data. Data-mining techniques are primarily oriented toward information visualization. The challenge for nonphysical data is in designing a visual representation of multidimensional samples (where the number of dimensions is greater than three). Multidimensional information visualizations present data that is not primarily plenary or spatial. One-, two-, and three-dimensional data include temporal information-visualization schemes that can be viewed as a subset of multidimensional information visualization. One approach is to map the nonphysical data to a virtual object such as a cone tree, which can be manipulated as if it were a physical object. Another approach is to map the nonphysical data to the graphical properties of points, lines, and areas.

Using historical developments as criteria, we can divide information-visualization techniques (IVT) into two broad categories: traditional IVT and novel IVT. Traditional methods of 2D and 3D graphics offer an opportunity for information

visualization, even though these techniques are more often used for presentation of physical data in scientific visualization. Traditional visual metaphors are used for a single or a small number of dimensions, and they include:

1. bar charts that show aggregations and frequencies,
2. histograms that show the distribution of variables values,
3. line charts for understanding trends in order,
4. pie charts for visualizing fractions of a total, and
5. scatter plots for bivariate analysis.

Color coding is one of the most common traditional IVT methods for displaying a one-dimensional set of values where each value is represented by a different color. This representation becomes a continuous tonal variation of color when real numbers are the values of a dimension. Normally, a color spectrum from blue to red is chosen, representing a natural variation from "cool" to "hot," in other words from the smallest to the highest values.

With the development of large data warehouses, data cubes became very popular information-visualization techniques. A *data cube*, the raw-data structure in a multidimensional database, organizes information along a sequence of categories. The categorizing variables are called dimensions. The data, called measures, are stored in cells along given dimensions. The cube dimensions are organized into hierarchies and usually include a dimension representing time. The hierarchical levels for the dimension time may be year, quarter, month, day, and hour. Similar hierarchies could be defined for other dimensions given in a data warehouse. Multidimensional databases in modern data warehouses automatically aggregate measures across hierarchical dimensions. They support hierarchical navigation; expand and collapse dimensions; enable drill-down, drill-up, or drill-across; and facilitate comparisons through time. In a transaction information in the database, the cube dimensions might be product, store, department, customer number, region, month, or year. The dimensions are predefined indices in a cube cell, and the measures in a cell are roll-ups or aggregations over the transactions. They are usually sums but may include functions such as average, standard deviation, and percentage.

For example, the values for the dimensions in a database may be:

1. Region: north, south, east, west.
2. Product: shoes, shirts.
3. Month: January, February, March,…,December.

Then, the cell corresponding to [north, shirt, February] is the total sales of shirts for the northern region for the month of February.

Novel information-visualization techniques can simultaneously represent large data sets with many dimensions on one screen. Widely accepted classification of these new techniques is:

1. geometric projection techniques,
2. icon-based techniques,
3. pixel-oriented techniques, and
4. hierarchical techniques.

Geometric projection techniques aim to find interesting projections of multidimensional data sets. We will present some illustrative examples of these techniques.

The scatter-plot matrix technique is an approach that is very often available in new data-mining software tools. A grid of 2D scatter plots is the standard means of extending a standard 2D scatter plot to higher dimensions. If you have 10-dimensional data, a 10×10 array of scatter plots is used to provide a visualization of each dimension versus every other dimension. This is useful for looking at all possible two-way interactions or correlations between dimensions. Positive and negative correlations, but only between two dimensions, can be seen easily. The standard display quickly becomes inadequate for extremely large numbers of dimensions, and user interactions of zooming and panning are needed to interpret the scatter plots effectively.

The survey plot is a simple technique of extending an n-dimensional point (sample) in a line graph. Each dimension of the sample is represented on a separate axis in which the dimension's value is a proportional line from the center of the axis. The principles of representation are given in Figure 15.1.

This visualization of *n*-dimensional data allows you to see correlations between any two variables, especially when the data is sorted according to a particular dimension. When color is used for different classes of samples, you can sometimes use a sort to see which dimensions are best at classifying data samples. This technique was evaluated with different machine-learning data sets, and it showed the ability to present exact IF–THEN rules in a set of samples.

The *Andrews's curves* technique plots each n-dimensional sample as a curved line. This is an approach similar to a Fourier transformation of a data point. This technique uses the function $f(t)$ in the time domain t to transform the *n*-dimensional point
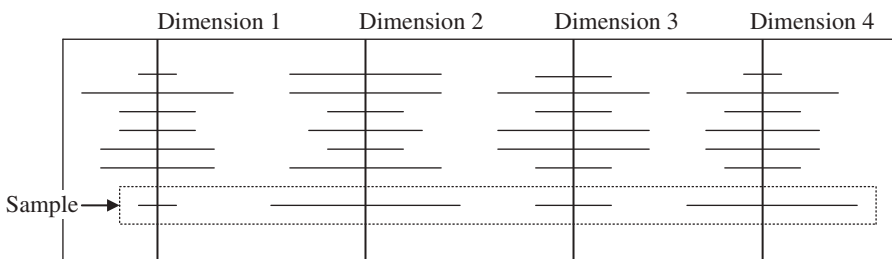
Figure 15.1. A four-dimensional survey plot.

$X = (x_1, x_2, x_3,\ldots, x_n)$ into a continuous plot. The function is usually plotted in the interval $-\pi \le t \le \pi$. An example of the transforming function $f(t)$ is

$$f(t) = \frac{x_1}{1.41} + x_2 \sin(t) + x_3 \cos(t) + x_4 \sin(2t) + x_5 \cos(2t) + \cdots$$

One advantage of this visualization is that it can represent many dimensions; the disadvantage, however, is the computational time required to display each n-dimensional point for large data sets.

The class of geometric projection techniques includes also techniques of exploratory statistics such as principal component analysis, factor analysis, and multidimensional scaling. Parallel-coordinate-visualization technique and radial-visualization technique belong in this category of visualizations, and they are explained in the next sections.

Another class of techniques for visual data mining is the *icon-based techniques* or iconic display techniques. The idea is to map each multidimensional data item to an icon. An example is the stick figure technique. It maps two dimensions to the display dimensions, and the remaining dimensions are mapped to the angles and/or limb lengths of the stick figure icon. This technique limits the number of dimensions that can be visualized. A variety of special symbols have been invented to convey simultaneously the variations on several dimensions for the same sample. In 2D displays, these include Chernoff's faces, glyphs, stars, and color mapping. Glyphs represent samples as complex symbols whose features are functions of data. We think of glyphs as location-independent representations of samples. For successful use of glyphs, however, some sort of suggestive layout is often essential, because comparison of glyph shapes is what this type of rendering primarily does. If glyphs are used to enhance a scatter plot, the scatter plot takes over the layout functions. Figure 15.2 shows how the other icon-based technique, called a *star display*, is applied to quality of life measures for various states. Seven dimensions represent seven equidistant radiuses for a circle: one circle for each sample. Every dimension is normalized on interval
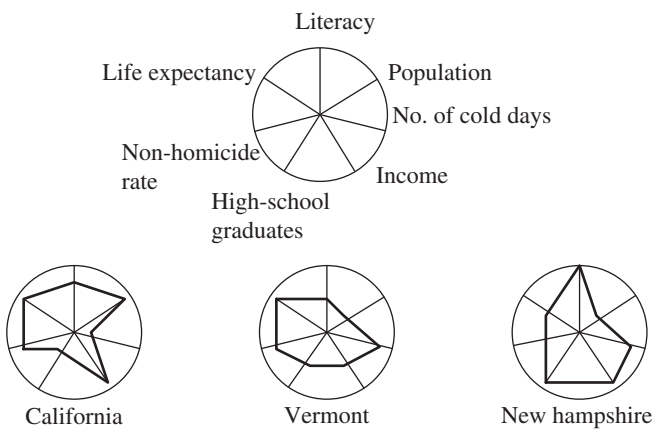
Figure 15.2. A star display for data on seven quality-of-life measures for three states.

[0, 1], where the value 0 is in the center of the circle and the value 1 is at the end of the corresponding radius. This representation is convenient for a relatively large number of dimensions but for a very small number of samples. It is usually used for comparative analyses of samples, and it may be included as a part of more complex visualizations.

The other approach is an icon-based, shape-coding technique that visualizes an arbitrary number of dimensions. The icon used in this approach maps each dimension to a small array of pixels and arranges the pixel arrays of each data item into a square or a rectangle. The pixels corresponding to each of the dimensions are mapped to gray scale or color according to the dimension's data value. The small squares or rectangles corresponding to the data items or samples are then arranged successively in a line-by-line fashion.

The third class of visualization techniques for multidimensional data aims to map each data value to a colored pixel and present the data values belonging to each attribute in separate windows. Since the *pixel-oriented techniques* use only one pixel per data value, the techniques allow a visualization of the largest amount of data that is possible on current displays (up to about 1,000,000 data values). If one pixel represents one data value, the main question is how to arrange the pixels on the screen. These techniques use different arrangements for different purposes. Finally, the *hierarchical techniques* of visualization subdivide the k-dimensional space and present the subspaces in a hierarchical fashion. For example, the lowest levels are 2D subspaces. A common example of hierarchical techniques is dimensional-stacking representation.

Dimensional stacking is a recursive-visualization technique for displaying high-dimensional data. Each dimension is discretized into a small number of bins, and the display area is broken into a grid of subimages. The number of subimages is based on the number of bins associated with the two "outer" dimensions that are user-specified. The subimages are decomposed further based on the number of bins for two more dimensions. This decomposition process continues recursively until all dimensions have been assigned.

Some of the novel visual metaphors that combine data-visualization techniques are already built into advanced visualization tools, and they include the following:

1. *Parabox* that combines boxes, parallel coordinates, and bubble plots for visualizing *n*-dimensional data. It handles both continuous and categorical data. The reason for combining box and parallel-coordinate plots involves their relative strengths. Box plots work well for showing distribution summaries. The strength of parallel coordinates is their ability to display high-dimensional outliers, individual cases with exceptional values. Details about this class of visualization techniques are given in Section 15.3.

2. *Data constellations*, a component for visualizing large graphs with thousands of nodes and links. Two tables parametrize Data Constellations, one corresponding to nodes and another to links. Different layout algorithms dynamically position the nodes so that patterns emerge (a visual interpretation of outliers, clusters, etc.).

3. *Data sheet*, a dynamic scrollable-text visualization that bridges the gap between text and graphics. The user can adjust the zoom factor, progressively displaying smaller and smaller fonts, eventually switching to a one-pixel representation. This process is called smashing.
4. *Timetable*, a technique for showing thousands of time-stamped events.
5. *Multiscape* landscape visualization that encodes information using 3D "skyscrapers" on a 2D landscape.

An example of one of these novel visual representations is given in Figure 15.3, where a large graph is visualized using the Data Constellations technique with one possible graph-layout algorithm.

For most basic visualization techniques that endeavor to show each item in a data set, such as scatterplots or parallel coordinates, a massive number of items will overload the visualization, resulting in clutter that both causes scalability problems as well as hinders the user's understanding of its structure and contents. New visualization techniques have been proposed to overcome data overload problem and to introduce abstractions that reduce the amount of items to display either in data space or in visual space. The approach is based on coupling aggregation in data space with a corresponding visual representation of the aggregation as a visual entity in the graphical space. This visual aggregate can convey additional information about the underlying contents, such as an average value, minima and maxima, or even its data distribution.

Drawing visual representations of abstractions performed in data space allow for creating simplified versions of visualization while still retaining the general overview. By dynamically changing the abstraction parameters, the user can also retrieve details-on-demand. There are several algorithms to perform data aggregations in a visualization process. For example, given a set of data items, *hierarchical aggregation* is based on iteratively building a tree of aggregates either bottom up or top down. Each aggregate item consists of one or more children that are either the original data items (leaves) or aggregate items (nodes). The root of the tree is an aggregate item that represents the entire data set. One of the main visual aggregations for scatterplots involves
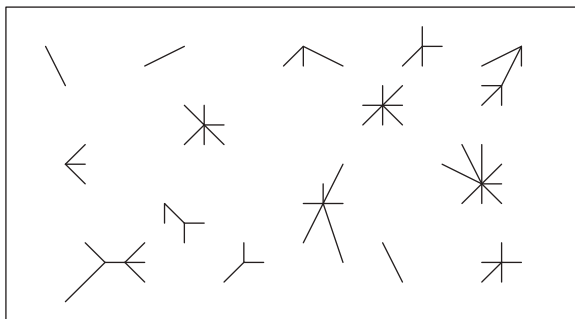
Figure 15.3. Data constellations as a novel visual metaphor.
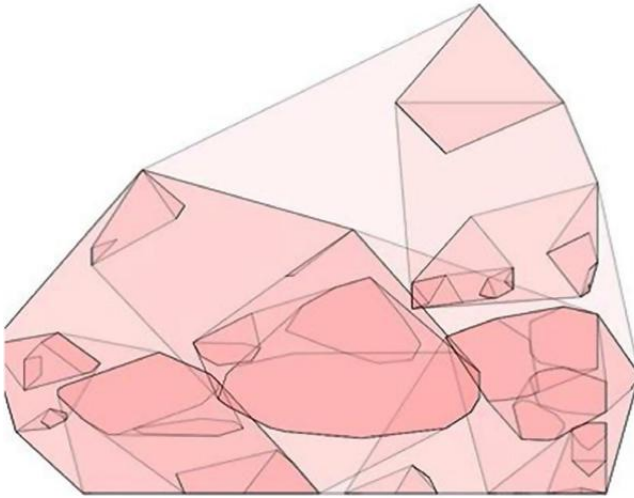
Figure 15.4. Convex hull aggregation (Elmquist 2010).

hierarchical aggregations of data into hulls, as it is represented in Figure 15.4. Hulls are variations and extensions of rectangular boxes as aggregates. They show enhanced displayed dimensions by using 2D or 3D convex hulls instead of axis-aligned boxes as a constrained visual metric. Clearly, the benefit of a data aggregate hierarchy and corresponding visual aggregates is that the resulting visualization can be adapted to the requirements of the human user as well as the technical limitations of the visualization platform.

15.3 PARALLEL COORDINATES

Geometric projection techniques include the parallel-coordinate-visualization technique, one of the most frequently used modern visualization tools. The basic idea is to map the k-dimensional space onto the two-display dimensions by using k equidistant axes parallel to one of the display axes. The axes correspond to the dimensions and are linearly scaled from the minimum to the maximum value of the corresponding dimension. Each data item is presented as a polygonal line, intersecting each of the axes at the point that corresponds to the value of the considered dimension.

Suppose that a set of six-dimensional samples, given in Table 15.1, is a small relational database. To visualize this data, it is necessary to determine the maximum and minimum values for each dimension. If we accept that these values are determined automatically based on a stored database, then graphical representation of data is given in Figure 15.5.

The *anchored-visualization perspective* focuses on displaying data with an arbitrary number of dimensions, say, between four and twenty, using and combining

TABLE 15.1. Database with Six Numeric Attributes

Sample#	Dimensions					
	A	B	C	D	E	F
1	1	5	10	3	3	5
2	3	1	3	1	2	2
3	2	2	1	2	4	2
4	4	2	1	3	1	2
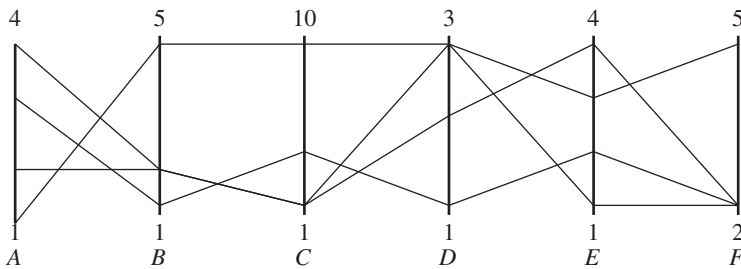
Figure 15.5. Graphical representation of six-dimensional samples from the database given in Table 15.1 using a parallel-coordinate-visualization technique.

multidimensional-visualization techniques such as weighted Parabox, bubble plots, and parallel coordinates. These methods handle both continuous and categorical data. The reason for combining them involves their relative strengths. Box plots work well for showing distribution summaries. Parallel coordinates' strength is their ability to display high-dimensional outliers, individual cases with exceptional values. Bubble plots are used for categorical data, and the size of the circles inside the bubbles shows the number of samples and their respective value. The dimensions are organized along a series of parallel axes, as with parallel-coordinate plots. Lines are drawn between the bubble and the box plots connecting the dimensions of each available sample. Combining these techniques results in a visual component that excels the visual representations created using separate methodologies.

An example of multidimensional anchored visualization, based on a simple and small data set, is given in Table 15.2. The total number of dimensions is five, two of them are categorical and three are numeric. Categorical dimensions are represented by bubble plots (one bubble for every value) and numeric dimensions by boxes. The circle inside the bubbles shows visually the percentage that the given value represents in a database. Lines inside the boxes represent mean value and standard deviation for a given numeric dimension. The resulting representation in Figure 15.6 shows all six five-dimensional samples as connecting lines. Although the database given in Table 15.2 is small, still, by using anchored representation, we can see that one sample is an outlier for both numeric and categorical dimensions.

TABLE 15.2. The Database for Visualization

Sample #	Dimensions				
	A	*B*	*C*	*D*	*E*
1	Low	Low	2	4	3
2	Med.	Med.	4	2	1
3	High	Med.	7	5	9
4	Med.	Low	1	3	5
5	Low	Low	3	1	2
6	Low	Med.	4	3	2
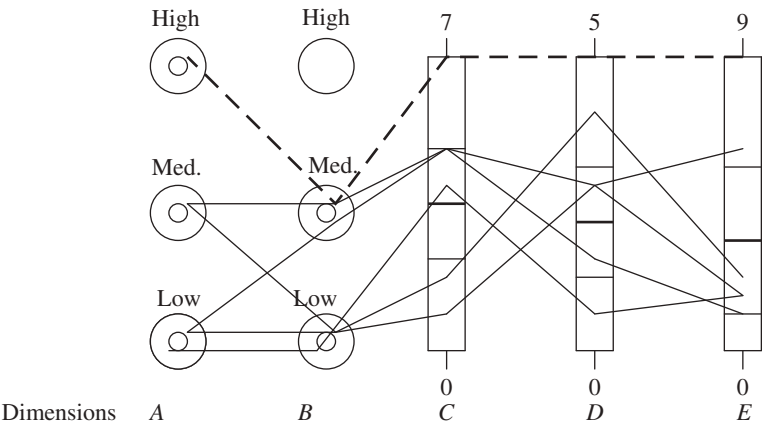
Figure 15.6. Parabox visualization of a database given in Table 15.2.

The circular-coordinate method is a simple variation of parallel coordinates, in which the axes radiate from the center of a circle and extend to the perimeter. The line segments are longer on the outer part of the circle where higher data values are typically mapped, whereas inner-dimensional values toward the center of the circle are more cluttered. This visualization is actually a star and glyph visualization of the data superimposed on one another. Because of the asymmetry of lower (inner) data values from higher ones, certain patterns may be easier to detect with this visualization.

15.4 RADIAL VISUALIZATION

Radial visualization is a technique for representation of multidimensional data where the number of dimensions is significantly greater than 3. Data dimensions are laid out as points equally spaced around the perimeter of a circle. For example, in the case of an eight-dimensional space, the distribution of dimensions will be given as in Figure 15.7.
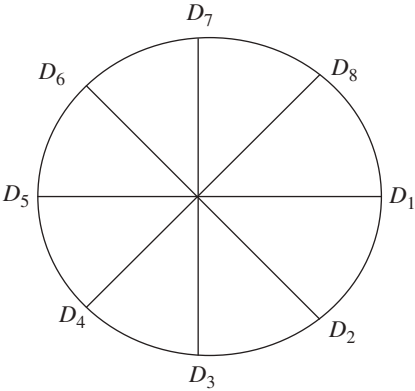
Figure 15.7. Radial visualization for an eight-dimensional space.
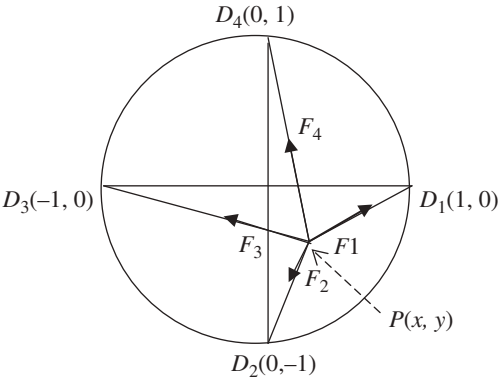
Figure 15.8. Sum of the spring forces for the given point P is equal to 0.

A model of springs is used for point representation. One end of n springs (one spring for each of n dimensions) is attached to n perimeter points. The other end of the springs is attached to a data point. Spring constants can be used to represent values of dimensions for a given point. The spring constant K_i equals the value of the ith coordinate of the given n-dimensional point where $i = 1,\ldots,n$. Values for all dimensions are normalized to the interval between 0 and 1. Each data point is then displayed in 2D under condition that the sum of the spring forces is equal to 0. The radial visualization of a four-dimensional point $P(K_1, K_2, K_3, K_4)$ with the corresponding spring force is given in Figure 15.8.

Using basic laws from physics, we can establish a relation between coordinates in an n-dimensional space and in 2D presentation. For our example of 4D representation given in Figure 15.8, point P is under the influence of four forces F_1, F_2, F_3, and F_4.

Knowing that every one of these forces can be expressed as a product of a spring constant and a distance or in a vector form

$$F = K \cdot d$$

it is possible to calculate this force for a given point. For example, force F_1 in Figure 15.8 is a product of a spring constant K_1 and a distance vector between points $P(x, y)$ and $D_1(1,0)$:

$$F_1 = K_1 \cdot [(x-1)i + yj]$$

The same analysis will give expressions for F_2, F_3, and F_4. Using the basic relation between forces

$$F_1 + F_2 + F_3 + F_4 = 0$$

we will obtain

$$K_1[(x-1)i + yj] + K_2[xi + (y+1)j] + K_3[(x+1)i + yj] + K_4[xi + (y-1)j] = 0$$

Both the i and j components of the previous vector have to be equal to 0, and therefore

$$K_1(x-1) + K_2x + K_3(x+1) + K_4x = 0$$
$$K_1y + K_2(y+1) + K_3y + K_4(y-1) = 0$$

or

$$x = \frac{(K_1 - K_3)}{(K_1 + K_2 + K_3 + K_4)}$$

$$y = \frac{(K_4 - K_2)}{(K_1 + K_2 + K_3 + K_4)}$$

These are the basic relations for representing a four-dimensional point $P^*(K_1, K_2, K_3, K_4)$ in a 2D space $P(x, y)$ using the radial-visualization technique. Similar procedures may be performed to get transformations for other n-dimensional spaces.

We can analyze the behavior of n-dimensional points after transformation and representation with two dimensions. For example, if all n coordinates have the same value, the data point will lie exactly in the center of the circle. In our four-dimensional space, if the initial point is $P_1^*(0.6, 0.6, 0.6, 0.6)$, then using relations for x and y its presentation will be $P_1(0, 0)$. If the n-dimensional point is a unit vector for one dimension, then the projected point will lie exactly at the fixed point on the edge of the circle (where the spring for that dimension is fixed). Point $P_2^*(0, 0, 1, 0)$ will be represented

as $P_2(-1, 0)$. Radial visualization represents a nonlinear transformation of the data, which preserves certain symmetries. This technique emphasizes the relations between dimensional values, not between separate absolute values. Some additional features of radial visualization include the following:

1. Points with approximately equal coordinate values will lie close to the center of the representational circle. For example, $P_3^*(0.5, 0.6, 0.4, 0.5)$ will have 2D coordinates $P_3(0.05, -0.05)$.
2. Points that have one or two coordinate values greater than the others lie closer to the origins of those dimensions. For example, $P_4^*(0.1, 0.8, 0.6, -0.1)$ will have a 2D representation $P_4(-0.36, -0.64)$. The point is in a third quadrant closer to D2 and D3, points where the spring is fixed for the second and third dimensions.
3. An n-dimensional line will map to the line or in a special case to the point. For example, points $P_5^*(0.3, 0.3, 0.3, 0.3)$, $P_6^*(0.6, 0.6, 0.6, 0.6)$, and $P_7^*(0.9, 0.9, 0.9, 0.9)$ are on a line in a four-dimensional space, and all three of them will be transformed into the same 2D point $P_{567}(0, 0)$.
4. A sphere will map to an ellipse.
5. An n-dimensional plane maps to a bounded polygon.

The Gradviz method is a simple extension of a radial visualization that places the dimensional anchors on a rectangular grid instead of the perimeter of a circle. The spring forces work the same way. Dimensional labeling for *Gradviz* is difficult, but the number of dimensions that can be displayed increases significantly in comparison with the Radviz technique. For example, in a typical Radviz display, fifty seems to be a reasonable limit to the points around a circle. However, in a grid layout supported by the Gradviz technique, you can easily fit 50×50 grid points or dimensions into the same area.

15.5 VISUALIZATION USING SELF-ORGANIZING MAPS

Self-organizing map (SOM) is often seen as a promising technique for exploratory analyses through visualization of high-dimensional data. It visualizes a data structure of a high-dimensional data space usually as a 2D or 3D geometrical picture. SOMs are, in effect, a nonlinear form of principal component analysis and share similar goals to multidimensional scaling. PCA is much faster to compute, but it has disadvantage comparing SOMs of not retaining the topology of the higher-dimensional space.

The topology of the data set in its n-dimensional space is captured by the SOM and reflected in the ordering of its output nodes. This is an important feature of the SOM that allows the data to be projected onto a lower-dimensional space while roughly preserving the order of the data in its original space. Resultant SOMs are then visualized using graphical representations. SOM algorithm may use different

data-visualization techniques including a cell or *U*-matrix visualization (a distance matrix visualization), projections (mesh visualization), visualization of component planes (in a multiples linked view), and 2D and 3D surface plot of distance matrices. These representations use visual variables (size, value, texture, color, shape, orientation) added to the position property of the map elements. This allows exploration of relationships between samples. A coordinate system enables to determine distance and direction, from which other relationships (size, shape, density, arrangement, etc.) may be derived. Multiple levels of detail allow exploration at various scales, creating the potential for hierarchical grouping of items, regionalization, and other types of generalizations. Graphical representations in SOMs are used to represent uncovered structure and patterns that may be hidden in the data set and to support understanding and knowledge construction. An illustrative example is given in Figure 15.9 where linear or nonlinear relationships are detected by the SOM.

For years there has been visualization of primary numeric data using pie charts, colored graphs, graphs over time, multidimensional analysis, Pareto charts, and so forth. The counterpart to numeric data is unstructured textual data. Textual data is found in many places but nowhere more prominently than on the Web. Unstructured electronic data includes emails, email attachments, pdf files, spread sheets, Power-Point files, text files, document files, and many more. In this new environment the end user faces massive amounts often millions of unstructured documents. The end user cannot read them all, and especially there is no way he/she could manually organize or summarize them. Unstructured data runs less formal part of the organization, while structured data runs the formal part of the organization. It is a good assumption,
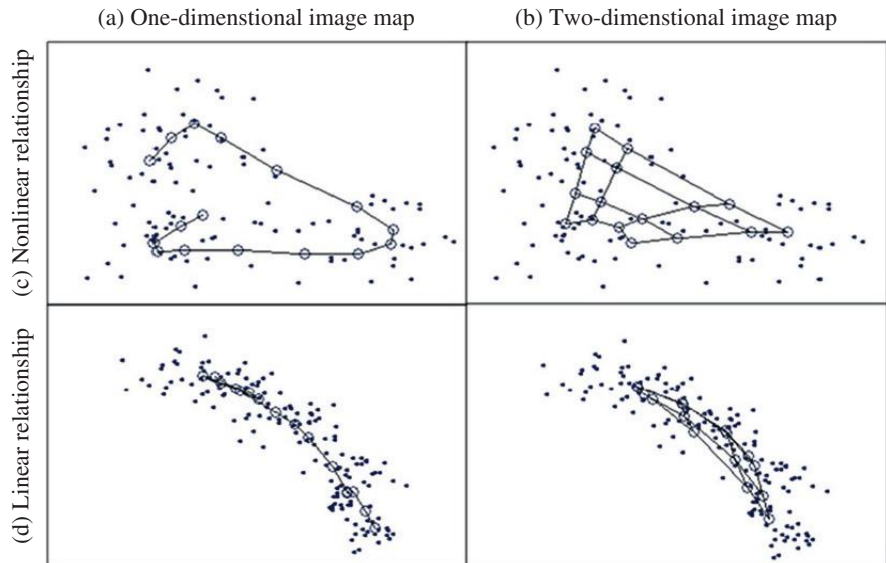
Figure 15.9. Output maps generated by the SOM detect relationships in data.

confirmed in many real-world applications, that as many business decisions are made in the unstructured environment as in the structured environment.

The SOM is one efficient solution for the problems of unstructured visualization of documents and unstructured data. With a properly constructed SOM, you can analyze literally millions of unstructured documents that can be merged into a single SOM. The SOM deals not only with individual unstructured documents but also relationships between documents. The SOMs may show text that is correlated to other text. For example, in the medical field, working with medical patient records, this ability to correlate is very attractive. The SOM also allows the analyst to see the larger picture as well as drilling down to the detailed picture. The SOM goes down to the individual stemmed text level, and that is as accurate as textual processing can become. All these characteristics have resulted in the growing popularity of SOM's visualizations in order to assist visual inspection of complex high-dimensional data. For the end user the flexibility of the SOM algorithm is defined through a number of parameters. For appropriate configuration of the network, and tuning the visualization output, user-defined parameters include grid dimensions (2D, 3D), grid shape (rectangle, hexagon), number of output nodes, neighborhood function, neighborhood size, learning rate function, initial weights in the network, way of learning and number of iterations, and order of input samples.

15.6   VISUALIZATION SYSTEMS FOR DATA MINING

Many organizations, particularly within the business community, have made significant investments in collecting, storing, and converting business information into results that can be used. Unfortunately, typical implementations of business "intelligence software" have proven to be too complex for most users except for their core reporting and charting capabilities. Users' demands for multidimensional analysis, finer data granularity, and multiple data sources, simultaneously, all at Internet speed, require too much specialist intervention for broad utilization. The result is a report explosion in which literally hundreds of predefined reports are generated and pushed throughout the organization. Every report produces another. Presentations get more complex. Data is exploding. The best opportunities and the most important decisions are often the hardest to see. This is in direct conflict with the needs of frontline decision-makers and knowledge workers who are demanding to be included in the analytical process.

Presenting information visually, in an environment that encourages the exploration of linked events, leads to deeper insights and more results that can be acted upon. Over the past decade, research on information visualization has focused on developing specific visualization techniques. An essential task for the next period is to integrate these techniques into a larger system that supports work with information in an interactive way, through the three basic components: *foraging the data*, *thinking about data*, and *acting on data*.

The vision of a visual data-mining system stems from the following principles: simplicity, visibility, user autonomy, reliability, reusability, availability, and security. A visual data-mining system must be syntactically simple to be useful. Simple does not mean trivial or nonpowerful. Simple to learn means use of intuitive and friendly input mechanisms as well as instinctive and easy-to-interpret output knowledge. Simple to apply means an effective discourse between humans and information. Simple to retrieve or recall means a customized data structure that facilitates fast and reliable searches. Simple to execute means a minimum number of steps needed to achieve the results. In short, simple means the smallest, functionally sufficient system possible.

A genuinely visual data-mining system must not impose knowledge on its users, but instead guide them through the mining process to draw conclusions. Users should study the visual abstractions and gain insight instead of accepting an automated decision. A key capability in visual analysis, called visibility, is the ability to focus on particular regions of interest. There are two aspects of visibility: excluding and restoring data. The exclude process eliminates the unwanted data items from the display so that only the selected set is visible. The restore process brings all data back, making them visible again.

A reliable data-mining system must provide for estimated error or accuracy of the projected information in each step of the mining process. This error information can compensate for the deficiency that an imprecise analysis of data visualization can cause. A reusable, visual data-mining system must be adaptable to a variety of environments to reduce the customization effort, provide assured performance, and improve system portability. A practical, visual data-mining system must be generally and widely available. The quest for new knowledge or deeper insights into existing knowledge cannot be planned. It requires that the knowledge received from one domain adapt to another domain through physical means or electronic connections. A complete, visual data-mining system must include security measures to protect the data, the newly discovered knowledge, and the user's identity because of various social issues.

Through data visualization we want to understand or get an overview of the whole or a part of the n-dimensional data, analyzing also some specific cases. Visualization of multidimensional data helps decision-makers to:

1. slice information into multiple dimensions and present information at various levels of granularity,
2. view trends and develop historical tracers to show operations over time,
3. produce pointers to synergies across multiple dimensions,
4. provide exception analysis and identify isolated (needle in the haystack) opportunities,
5. monitor adversarial capabilities and developments,
6. create indicators of duplicative efforts, and
7. conduct What-If Analysis and Cross-Analysis of variables in a data set.

Visualization tools transform raw experimental or simulated data into a form suitable for human understanding. Representations can take on many different forms, depending on the nature of the original data and the information that is to be extracted. However, the visualization process that should be supported by modern visualization-software tools can generally be subdivided into three main stages: data preprocessing, visualization mapping, and rendering. Through these three steps the tool has to answer the questions: What should be shown in a plot? How should one work with individual plots? How should multiple plots be organized?

Data preprocessing involves such diverse operations as interpolating irregular data, filtering and smoothing raw data, and deriving functions for measured or simulated quantities. Visualization mapping is the most crucial stage of the process, involving design and adequate representation of the filtered data, which efficiently conveys the relevant and meaningful information. Finally, the representation is often rendered to communicate information to the human user.

Data visualization is essential for understanding the concept of multidimensional spaces. It allows the user to explore the data in different ways and at different levels of abstraction to find the right level of details. Therefore, techniques are most useful if they are highly interactive, permit direct manipulation, and include a rapid response time. The analyst must be able to navigate the data, change its grain (resolution), and alter its representation (symbols, colors, etc.).

Broadly speaking, the problems addressed by current information-visualization tools and requirements for a new generation fall into the following classes:

1. *Presentation graphics*—These generally consist of bars, pies, and line charts that are easily populated with static data and drop into printed reports or presentations. The next generation of presentation graphics enriches the static displays with a 3D or projected *n*-dimensional information landscape. The user can then navigate through the landscape and animate it to display time-oriented information.

2. *Visual interfaces for information access*—They are focused on enabling users to navigate through complex information spaces to locate and retrieve information. Supported user tasks involve searching, backtracking, and history logging. User-interface techniques attempt to preserve user context and support smooth transitions between locations.

3. *Full visual discovery and analysis*—These systems combine the insights communicated by presentation graphics with an ability to probe, drill down, filter, and manipulate the display to answer the "why" question as well as the "what" question. The difference between answering a "what" and a "why" question involves an interactive operation. Therefore, in addition to the visualization technique, effective data exploration requires using some *interaction* and *distortion* techniques. The *interaction techniques* let the user directly interact with the visualization. Examples of interaction techniques include interactive mapping, projection, filtering, zooming, and interactive linking and brushing. These techniques allow dynamic changes in the visualizations according to the

exploration objectives, but they also make it possible to relate and combine multiple, independent visualizations. Note that connecting multiple visualizations by linking and brushing, e.g., provides more information than considering the component visualizations independently. The *distortion techniques* help in the interactive exploration process by providing a means for focusing while preserving an overview of the data. Distortion techniques show portions of the data with a high level of detail, while other parts are shown with a much lower level of detail.

Three tasks are fundamental to data exploration with these new visualization tools:

1. *Finding Gestalt*—Local and global linearities and nonlinearities, discontinuities, clusters, outliers, unusual groups, and so on are examples of gestalt features that can be of interest. Focusing through individual views is the basic requirement to obtain a qualitative exploration of data using visualization. Focusing determines what gestalt of the data is seen. The meaning of focusing depends very much on the type of visualization technique chosen.
2. *Posing queries*—This is a natural task after the initial gestalt features have been found, and the user requires query identification and characterization technique. Queries can concern individual cases as well as subsets of cases. The goal is essentially to find intelligible parts of the data. In graphical data analysis it is natural to pose queries graphically. For example, familiar brushing techniques such as coloring or otherwise highlighting a subset of data means issuing a query about this subset. It is desirable that the view where the query is posed and the view that present the response are linked. Ideally, responses to queries should be instantaneous.
3. *Making comparisons*—Two types of comparisons are frequently made in practice. The first one is a comparison of variables or projections, and the second one is a comparison of subsets of data. In the first case, one compares views "from different angles"; in the second, comparison is based on views "of different slices" of the data. In either case, it is likely that a large number of plots are generated, and therefore it is a challenge to organize the plots in such a way that meaningful comparisons are possible.

Visualization has been used routinely in data mining as a presentation tool to generate initial views, navigate data with complicated structures, and convey the results of an analysis. Generally, the analytical methods themselves do not involve visualization. The loosely coupled relationships between visualization and analytical data-mining techniques represent the majority of today's state of the art in visual data mining. The process-sandwich strategy, which interlaces analytical processes with graphical visualization, penalizes both procedures with the other's deficiencies and limitations. For example, because an analytical process cannot analyze multimedia data, we have to give up the strength of visualization to study movies and music in a visual data-mining environment. A stronger strategy lies in tightly coupling the

visualization and analytical processes into one data-mining tool. Letting human visualization participate in the decision-making in analytical processes remains a major challenge. Certain mathematical steps within an analytical procedure may be substituted by human decisions based on visualization to allow the same procedure to analyze a broader scope of information. Visualization supports humans in dealing with decisions that can no longer be automated.

For example, visualization techniques can be used for efficient process of "visual clustering." The algorithm is based on finding a set of projections $P = [P_1, P_2, \ldots, P_k]$ useful for separating the initial data into clusters. Each projection represents the histogram information of the point density in the projected space. The most important information about a projection is whether it contains well-separated clusters. Note that well-separated clusters in one projection could result from more than one cluster in the original space. Figure 15.10 shows an illustration of these projections. You can see that the axes' parallel projections do not preserve well the information necessary for clustering. Additional projections A and B, in Figure 15.10, define three clusters in the initial data set.

Visual techniques that preserve some characteristics of the data set can be invaluable for obtaining good separators in a clustering process. In contrast to dimension reduction approaches such as principal component analyses, this visual approach does not require that a single projection preserves all clusters. In the projections, some clusters may overlap and therefore not be distinguishable, such as projection A in Figure 15.10. The algorithm only needs projections that separate the data set into at least two subsets without dividing any clusters. The subsets may then be refined using other projections and possibly partitioned further based on separators in other projections. Based on the visual representation of the projections, it is possible to find clusters with unexpected characteristics (shapes, dependencies) that
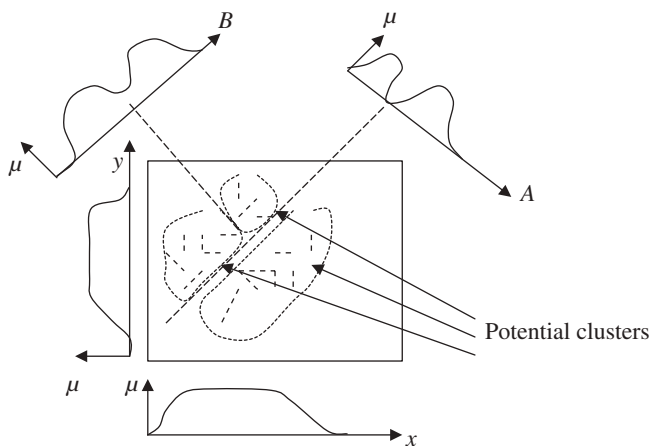
Figure 15.10. An example of the need for general projections, which are not parallel to axes, to improve clustering process.

would be very difficult or impossible to find by tuning the parameter settings of automatic-clustering algorithms.

In general, model visualization and exploratory data analysis (EDA) are data-mining tasks in which visualization techniques have played a major role. Model visualization is the process of using visual techniques to make the discovered knowledge understandable and interpretable by humans. Techniques range from simple scatter plots and histograms to sophisticated multidimensional visualizations and animations. These visualization techniques are being used not only to convey mining results more understandable to end users but also to help them understand how the algorithm works. EDA, on the other hand, is the interactive exploration of usually graphical representations of a data set without heavy dependence on preconceived assumptions and models, thus attempting to identify interesting and previously unknown patterns. Visual data exploration techniques are designed to take advantage of the powerful visual capabilities of human beings. They can support users in formulating hypotheses about the data that may be useful in further stages of the mining process.

15.7 REVIEW QUESTIONS AND PROBLEMS

1. Explain the power of n-dimensional visualization as a data-mining technique. What are the phases of data mining supported by data visualization?

2. What are fundamental experiences in human perception we would build into effective visualization tools?

3. Discuss the differences between scientific visualization and information visualization.

4. The following is the data set X:

X:	Year	A	B
	1996	7	100
	1997	5	150
	1998	7	120
	1999	9	150
	2000	5	130
	2001	7	150

Although the following visualization techniques are not explained with enough details in this book, use your knowledge from earlier studies of statistics and other courses to create 2D presentations:

(a) Show a bar chart for the variable A.
(b) Show a histogram for the variable B.
(c) Show a line chart for the variable B.

(d) Show a pie chart for the variable A.

(e) Show a scatter plot for A and B variables.

5. Explain the concept of a data cube and where it is used for visualization of large data sets.

6. Use examples to discuss the differences between icon-based and pixel-oriented visualization techniques.

7. Given seven-dimensional samples:

x_1	x_2	x_3	x_4	x_5	x_6	x_7
A	1	25	7	T	1	5
B	3	27	3	T	2	9
A	5	29	5	T	1	7
A	2	21	9	F	3	2
B	5	30	7	F	1	7

(a) Make a graphical representation of samples using the parallel-coordinates technique.

(b) Are there any outliers in the given data set?

8. Derive formulas for radial visualization of

(a) Three-dimensional samples.

(b) Eight-dimensional samples.

(c) Using the formulas derived in (a), represent samples (2, 8, 3) and (8, 0, 0).

(d) Using the formulas derived in (b), represent samples (2, 8, 3, 0, 7, 0, 0, 0) and (8, 8, 0, 0, 0, 0, 0, 0).

9. Implement a software tool supporting a radial-visualization technique.

10. Explain the requirements for full visual discovery in advanced visualization tools.

11. Search the Web to find the basic characteristics of publicly available or commercial software tools for visualization of n-dimensional samples. Document the results of your search.

15.8 REFERENCES FOR FURTHER STUDY

1. Fayyad, V., G. G. Grinstein, A. Wierse, *Information Visualization in Data Mining and Knowledge Discovery*, Morgan Kaufmann, San Diego, CA, 2002.

   Leading researchers from the fields of data mining, data visualization, and statistics present findings organized around topics introduced in two recent international knowledge-discovery and data-mining workshops. The book introduces the concepts and components of visualization, details current efforts to include

visualization and user interaction in data-mining, and explores the potential for further synthesis of data-mining algorithms and data-visualization techniques.

2. Spence, R., *Information Visualization*, Addison Wesley, Harlow, England, 2001.

   This is the first fully integrated book on the emerging discipline of information visualization. Its emphasis is on real-world examples and applications of computer-generated interactive information visualization. The author also explains how these methods for visualizing information support rapid learning and accurate decision-making.

3. Draper G. M., L. Y. Livnat, R. F. Riesenfeld, A Survey of Radial Methods for Information Visualization, *IEEE Transaction on Visualization and Computer Graphics*, Vol. 15, No. 5, September/October 2009, pp. 759–776.

   Radial visualization, or the practice of displaying data in a circular or elliptical pattern, is an increasingly common technique in information-visualization research. In spite of its prevalence, little work has been done to study this visualization paradigm as a methodology in its own right. We provide a historical review of radial visualization, tracing it to its roots in centuries-old statistical graphics. We then identify the types of problem domains to which modern radial-visualization techniques have been applied. A taxonomy for radial visualization is proposed in the form of seven design patterns encompassing nearly all recent works in this area. From an analysis of these patterns, we distill a series of design considerations that system builders can use to create new visualizations that address aspects of the design space that have not yet been explored. It is hoped that our taxonomy will provide a framework for facilitating discourse among researchers and stimulate the development of additional theories and systems involving radial visualization as a distinct design metaphor.

4. Ferreira de Oliveira M. C., H. Levkowitz, From Visual Data Exploration to Visual Data Mining: A Survey, *IEEE Transactions On Visualization And Computer Graphics*, Vol. 9, No. 3, July–September 2003, pp. 378–394.

   The authors survey work on the different uses of graphical mapping and interaction techniques for visual data mining of large data sets represented as table data. Basic terminology related to data mining, data sets, and visualization is introduced. Previous work on information visualization is reviewed in light of different categorizations of techniques and systems. The role of interaction techniques is discussed, in addition to work addressing the question of selecting and evaluating visualization techniques. We review some representative work on the use of information-visualization techniques in the context of mining data. This includes both visual data exploration and visually expressing the outcome of specific mining algorithms. We also review recent innovative approaches that attempt to integrate visualization into the DM/KDD process, using it to enhance user interaction and comprehension.

5. Tufte E. R., *Beautiful Evidence*, Graphic Press, LLC, 2nd edition, January 2007.

   Beautiful Evidence is a masterpiece from a pioneer in the field of data visualization. It is not often an iconoclast comes along, trashes the old ways, and replaces

them with an irresistible new interpretation. By teasing out the sublime from the seemingly mundane world of charts, graphs, and tables, Tufte has proven to a generation of graphic designers that great thinking begets great presentation. In *Beautiful Evidence*, his fourth work on analytical design, Tufte digs more deeply into art and science to reveal very old connections between truth and beauty—all the way from Galileo to Google.

6. Segall, Richard S., Jeffrey S. Cook, *Handbook of Research on Big Data Storage and Visualization Techniques*, IGI Global, 2018.

The digital age has presented an exponential growth in the amount of data available to individuals looking to draw conclusions based on given or collected information across industries. Challenges associated with the analysis, security, sharing, storage, and visualization of large and complex data sets continue to plague data scientists and analysts alike as traditional data processing applications struggle to adequately manage big data. The handbook is a critical scholarly resource that explores big data analytics and technologies and their role in developing a broad understanding of issues pertaining to the use of big data in multidisciplinary fields. Featuring coverage on a broad range of topics, such as architecture patterns, programing systems, and computational energy, this publication is geared toward professionals, researchers, and students seeking current research and application topics on the subject.

APPENDIX A
INFORMATION ON
DATA MINING

This summary of some recognized journals, conferences, blog sites, data-mining tools, and data sets is being provided to help readers to communicate with other users of data-mining technology and to receive information about trends and new applications in the field. It could be especially useful for students who are starting to work in data mining and trying to find appropriate information or solve current class-oriented tasks. This list is not intended to endorse any specific Web site, and the reader has to be aware that this is only small sample of possible resources on the Internet.

A.1 DATA-MINING JOURNALS

1. Data Mining and Knowledge Discovery (DMKD)

https://link.springer.com/journal/10618

Data Mining and Knowledge Discovery is a premier technical publication in the KDD field, providing a resource collecting relevant common methods and techniques and a forum for unifying the diverse constituent research communities. The journal publishes original technical papers in both the research and practice of DMKD, surveys and tutorials of important areas and techniques, and detailed descriptions of significant applications. The scope of *Data Mining and Knowledge Discovery* includes: (1) theory and foundational issues including data and knowledge representation, uncertainty management, algorithmic complexity, and statistics over massive data sets; (2) data-mining methods such as classification, clustering, probabilistic modeling,

prediction and estimation, dependency analysis, search, and optimization; (3) algorithms for spatial, textual, and multimedia data mining, scalability to large databases, parallel and distributed data-mining techniques, and automated discovery agents; (4) knowledge-discovery process including data preprocessing, evaluating, consolidating, and explaining discovered knowledge, data and knowledge visualization, and interactive data exploration and discovery; and (5) application issues such as application case studies, data-mining systems and tools, details of successes and failures of KDD, resource/knowledge discovery on the Web, and privacy and security issues.

2. IEEE Transactions on Knowledge and Data Engineering (TKDE)

https://ieeexplore.ieee.org/xpl/RecentIssue.jsp?punumber=69

The *IEEE Transactions on Knowledge and Data Engineering* is an archival journal published monthly. The information published in this Transactions is designed to inform researchers, developers, managers, strategic planners, users, and others interested in state-of-the-art and state-of-the-practice activities in the knowledge and data engineering area. We are interested in well-defined theoretical results and empirical studies that have potential impact on the acquisition, management, storage, and graceful degeneration of knowledge and data, as well as in provision of knowledge and data services. Specific topics include, but are not limited to (1) artificial intelligence techniques, including speech, voice, graphics, images, and documents; (2) knowledge and data engineering tools and techniques; (3) parallel and distributed processing; (4) real-time distributed; (5) system architectures, integration, and modeling; (6) database design, modeling, and management; (7) query design and implementation languages; (8) distributed database control; (9) algorithms for data and knowledge management; (10) performance evaluation of algorithms and systems; (11) data communications aspects; m) system applications and experience; (12) knowledge-based and expert systems; and (13) integrity, security, and fault tolerance.

3. Knowledge and Information Systems (KAIS)

http://www.cs.uvm.edu/~kais/

Knowledge and Information Systems (KAIS) is a peer-reviewed archival journal published by Springer. It provides an international forum for researchers and professionals to share their knowledge and report new advances on all topics related to knowledge systems and advanced information systems. The journal focuses on knowledge systems and advanced information systems, including their theoretical foundations, infrastructure, enabling technologies, and emerging applications. In addition to archival papers, the journal also publishes significant ongoing research in the form of short papers and very short papers on "visions and directions."

4. IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)

http://computer.org/tpami/

IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI) is a scholarly archival journal published monthly. Its editorial board strives to present

most important research results in areas within TPAMI's scope. This includes all traditional areas of computer vision and image understanding, all traditional areas of pattern analysis and recognition, and selected areas of machine intelligence. Areas such as machine learning, search techniques, document and handwriting analysis, medical image analysis, video and image sequence analysis, content-based retrieval of image and video, face and gesture recognition, and relevant specialized hardware and/or software architectures are also covered.

5. Machine Learning

https://link.springer.com/journal/10994

Machine Learning is an international forum for research on computational approaches to learning. The journal publishes articles reporting substantive results on a wide range of learning methods applied to a variety of learning problems. It features papers that describe research on problems and methods, applications research, and issues of research methodology, and papers making claims about learning problems or methods provide solid support via empirical studies, theoretical analysis, or comparison to psychological phenomena. Application papers show how to apply learning methods to solve important application problems. Research methodology papers improve how machine-learning research is conducted. All papers describe the supporting evidence in ways that can be verified or replicated by other researchers. The papers also detail the learning component clearly and discuss assumptions regarding knowledge representation and the performance task.

6. Journal of Machine Learning Research (JMLR)

http://www.jmlr.org/

The Journal of Machine Learning Research provides an international forum for the electronic and paper publication of high-quality scholarly articles in all areas of machine learning. All published papers are freely available online. JMLR has a commitment to rigorous yet rapid reviewing. JMLR provides a venue for papers on machine learning featuring new algorithms with empirical, theoretical, psychological, or biological justification; experimental and/or theoretical studies yielding new insight into the design and behavior of learning in intelligent systems; accounts of applications of existing techniques that shed light on the strengths and weaknesses of the methods; formalization of new learning tasks (e.g. in the context of new applications) and of methods for assessing performance on those tasks; development of new analytical frameworks that advance theoretical studies of practical learning methods; computational models of data from natural learning systems at the behavioral or neural level; or extremely well-written surveys of existing work.

7. ACM Transactions on Knowledge Discovery from Data (TKDD)

https://tkdd.acm.org/index.cfm

The ACM Transactions on Knowledge Discovery from Data addresses a full range of research in the knowledge discovery and analysis of diverse forms of data. Such subjects include scalable and effective algorithms for data mining and data

warehousing, mining data streams, mining multi-media data, mining high-dimensional data, mining text, Web, and semi-structured data, mining spatial and temporal data, data mining for community generation, social network analysis, and graph structured data, security and privacy issues in data mining, visual, interactive and online data mining, pre-processing and post-processing for data mining, robust and scalable statistical methods, data-mining languages, foundations of data mining, KDD framework and process, and novel applications and infrastructures exploiting data-mining technology.

8. Journal of Intelligent Information Systems (JIIS)

https://link.springer.com/journal/10844

The Journal of Intelligent Information Systems: Integrating Artificial Intelligence and Database Technologies (JIIS) fosters and presents research and development results focused on the integration of artificial intelligence and database technologies to create next-generation information systems—Intelligent Information Systems. JIIS provides a forum wherein academics, researchers, and practitioners may publish high-quality, original and state-of-the-art papers describing theoretical aspects, systems architectures, analysis and design tools and techniques, and implementation experiences in intelligent information systems. Articles published in JIIS include research papers, invited papers, meeting, workshop and conference announcements and reports, survey and tutorial articles, and book reviews. Topics include foundations and principles of data, information, and knowledge models; methodologies for IIS analysis, design, implementation, validation, maintenance, and evolution; and more.

9. Statistical Analysis and Data Mining

https://onlinelibrary.wiley.com/journal/19321872

The *Statistical Analysis and Data Mining* addresses the broad area of data analysis, including data-mining algorithms, statistical approaches, and practical applications. Topics include problems involving massive and complex data sets, solutions using innovative data-mining algorithms and/or novel statistical approaches, and the objective evaluation of analyses and solutions. Of special interest are articles that describe analytical techniques and discuss their application to real problems in such a way that they are accessible and beneficial to domain experts across science, engineering, and commerce.

10. Intelligent Data Analysis

http://www.iospress.nl/html/1088467x.php

Intelligent Data Analysis provides a forum for the examination of issues related to the research and applications of artificial intelligence techniques in data analysis across a variety of disciplines. These techniques include (but are not limited to) all areas of data visualization, data preprocessing (fusion, editing, transformation, filtering, sampling), data engineering, database mining techniques, tools and applications, use of domain knowledge in data analysis, evolutionary algorithms, machine

learning, neural nets, fuzzy logic, statistical pattern recognition, knowledge filtering, and postprocessing. In particular, we prefer papers that discuss development of new AI-related data-analysis architectures, methodologies, and techniques and their applications to various domains. Papers published in this journal are geared heavily toward applications, with an anticipated split of 70% of the papers published being applications-oriented research and the remaining 30% containing more theoretical research.

11. Expert Systems With Applications

https://www.journals.elsevier.com/expert-systems-with-applications

Expert Systems With Applications is a refereed international journal whose focus is on exchanging information relating to expert and intelligent systems applied in industry, government, and universities worldwide. The thrust of the journal is to publish papers dealing with the design, development, testing, implementation, and/or management of expert and intelligent systems and also to provide practical guidelines in the development and management of these systems. The journal will publish papers in expert and intelligent systems technology and application in the areas of, but not limited to, finance, accounting, engineering, marketing, auditing, law, procurement and contracting, project management, risk assessment, information management, information retrieval, crisis management, stock trading, strategic management, network management, telecommunications, space education, intelligent front ends, intelligent database-management systems, medicine, chemistry, human resources management, human capital, business, production management, archaeology, economics, energy, and defense. Papers in multi-agent systems, knowledge management, neural networks, knowledge discovery, data and text mining, multimedia mining, and genetic algorithms will also be published in the journal.

12. Computational Statistics & Data Analysis (CSDA)

https://www.journals.elsevier.com/computational-statistics-and-data-analysis

Computational Statistics & Data Analysis (CSDA), an Official Publication of the network Computational and Methodological Statistics (CMStatistics) and of the International Association for Statistical Computing (IASC), is an international journal dedicated to the dissemination of methodological research and applications in the areas of computational statistics and data analysis. The journal consists of three refereed sections that are divided into the following subject areas: I) computational statistics, II) statistical methodology for data analysis and statistical methodology, and III) special applications.

13. Neurocomputing

https://www.journals.elsevier.com/neurocomputing/

Neurocomputing publishes articles describing recent fundamental contributions in the field of neurocomputing. Neurocomputing theory, practice, and applications are the essential topics being covered. Neurocomputing welcomes theoretical

contributions aimed at winning further understanding of neural networks and learning systems, including, but not restricted to, architectures, learning methods, analysis of network dynamics, theories of learning, self-organization, biological neural-network modeling, sensorimotor transformations, and interdisciplinary topics with artificial intelligence, artificial life, cognitive science, computational learning theory, fuzzy logic, genetic algorithms, information theory, machine learning, neurobiology, and pattern recognition.

14. Information Sciences

https://www.journals.elsevier.com/information-sciences/

The journal is designed to serve researchers, developers, managers, strategic planners, graduate students, and others interested in state-of-the art research activities in information, knowledge engineering, and intelligent systems. Readers are assumed to have a common interest in information science, but with diverse backgrounds in fields such as engineering, mathematics, statistics, physics, computer science, cell biology, molecular biology, management science, cognitive science, neurobiology, behavioral sciences, and biochemistry. The journal publishes high-quality, refereed articles. It emphasizes a balanced coverage of both theory and practice. It fully acknowledges and vividly promotes a breadth of the discipline of information sciences.

15. ACM Transactions on Intelligent Systems and Technology (TIST)

https://tist.acm.org/index.cfm

ACM *Transactions on Intelligent Systems and Technology* is a scholarly journal that publishes the highest-quality papers on intelligent systems, applicable algorithms and technology with a multidisciplinary perspective. An intelligent system is one that uses artificial intelligence (AI) techniques to offer important services (e.g. as a component of a larger system) to allow integrated systems to perceive, reason, learn, and act intelligently in the real world.

A.2   DATA-MINING CONFERENCES

1. SIAM International Conference on Data Mining (SDM)

http://www.siam.org/meetings/

This conference provides a venue for researchers who are addressing extracting knowledge from large data sets that requires the use of sophisticated, high-performance, and principled analysis techniques and algorithms, based on sound theoretical and statistical foundations. It also provides an ideal setting for graduate students and others new to the field to learn about cutting-edge research by hearing outstanding invited speakers and attending presentations and tutorials (included with conference registration). A set of focused workshops are also held in the conference.

The proceedings of the conference are published in archival form and are also made available on the SIAM Web site.

2. The ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)

http://www.kdd.org/

The annual ACM SIGKDD conference is the premier international forum for data-mining researchers and practitioners from academia, industry, and government to share their ideas, research results, and experiences. It features keynote presentations, oral paper presentations, poster sessions, workshops, tutorials, panels, exhibits, and demonstrations. Authors can submit their original work either to SIGKDD Research track or SIGKDD Industry/Government track. The research track accepts papers on all aspects of knowledge discovery and data mining overlapping with topics from machine learning, statistics, databases, and pattern recognition. Papers are expected to describe innovative ideas and solutions that are rigorously evaluated and well presented. The Industrial/Government track highlights challenges, lessons, concerns, and research issues arising out of deploying applications of KDD technology. The focus is on promoting the exchange of ideas between researchers and practitioners of data mining.

3. IEEE International Conference on Data Mining (ICDM)

http://www.cs.uvm.edu/~icdm/

The IEEE International Conference on Data Mining (ICDM) has established itself as the world's premier research conference in data mining. The conference provides a leading forum for presentation of original research results, as well as exchange and dissemination of innovative, practical development experiences. The conference covers all aspects of data mining, including algorithms, software and systems, and applications. In addition, ICDM draws researchers and application developers from a wide range of data-mining-related areas such as statistics, machine learning, pattern recognition, databases and data warehousing, data visualization, knowledge-based systems, and high-performance computing. By promoting novel, high-quality research findings and innovative solutions to challenging data-mining problems, the conference seeks to continuously advance the state-of-the-art in data mining. Besides the technical program, the conference will feature workshops, tutorials, panels, and the ICDM data-mining contest.

4. International Conference on Machine Learning and Applications (ICMLA)

http://www.icmla-conference.org/

The aim of the conference is to bring researchers working in the areas of machine learning and applications together. The conference will cover both theoretical and experimental research results. Submission of machine-learning papers describing machine-learning applications in fields like medicine, biology, industry, manufacturing, security, education, virtual environments, game playing, and problem solving is strongly encouraged.

5. The World Congress in Computer Science Computer Engineering and Applied Computing (WORLDCOMP)

http://www.world-academy-of-science.org/

   WORLDCOMP is the largest annual gathering of researchers in computer science, computer engineering, and applied computing. It assembles a spectrum of affiliated research conferences, workshops, and symposiums into a coordinated research meeting held in a common place at a common time. This model facilitates communication among researchers in different fields of computer science and computer engineering. The WORLDCOMP is composed of more than 20 major conferences. Each conference will have its own proceedings. All conference proceedings/books are considered for inclusion in major database indexes that are designed to provide easy access to the current literature of the sciences (database examples: DBLP, ISI Thomson Scientific, IEE INSPEC).

6. IADIS European Conference on Data Mining (ECDM)

http://www.datamining-conf.org/

   The European Conference on Data Mining (ECDM) is aimed to gather researchers and application developers from a wide range of data-mining-related areas such as statistics, computational intelligence, pattern recognition, databases, and visualization. ECDM is aimed to advance the state of the art in data-mining field and its various real-world applications. ECDM will provide opportunities for technical collaboration among data-mining and machine-learning researchers around the globe.

7. Neural Information Processing Systems (NIPS) Conference

http://nips.cc/

   The Neural Information Processing Systems (NIPS) Foundation is a nonprofit corporation whose purpose is to foster the exchange of research on neural information processing systems in their biological, technological, mathematical, and theoretical aspects. Neural information processing is a field that benefits from a combined view of biological, physical, mathematical, and computational sciences.

   The primary focus of the NIPS Foundation is the presentation of a continuing series of professional meetings known as the Neural Information Processing Systems conference, held over the years at various locations in the United States and Canada.

   The NIPS conference features a single track program, with contributions from a large number of intellectual communities. Presentation topics include algorithms and architectures, applications, brain imaging, cognitive science and artificial intelligence, control and reinforcement learning, emerging technologies, learning theory, neuroscience, speech and signal processing, and visual processing.

8. European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD)

http://www.ecmlpkdd.org/

   The European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD) is one of the leading academic

conferences on machine learning and knowledge discovery, held in Europe every year. ECML PKDD is a merger of two European conferences, European Conference on Machine Learning (ECML) and European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD). In 2008 the conferences were merged into one conference, and the division into traditional ECML topics and traditional PKDD topics was removed.

9. Association for the Advancement of Artificial Intelligence (AAAI) Conference

http://www.aaai.org/

Founded in 1979, the Association for the Advancement of Artificial Intelligence (AAAI) (formerly the American Association for Artificial Intelligence) is a nonprofit scientific society devoted to advancing the scientific understanding of the mechanisms underlying thought and intelligent behavior and their embodiment in machines. AAAI also aims to increase public understanding of artificial intelligence, improve the teaching and training of AI practitioners, and provide guidance for research planners and funders concerning the importance and potential of current AI developments and future directions.Major AAAI activities include organizing and sponsoring conferences, symposia, and workshops; publishing a quarterly magazine for all members; publishing books, proceedings, and reports; and awarding grants, scholarships, and other honors. The purpose of the AAAI conference is to promote research in AI and scientific exchange among AI researchers, practitioners, scientists, and engineers in related disciplines.

10. International Conference on Very Large Data Base (VLDB)

http://www.vldb.org/

VLDB Endowment Inc. is a nonprofit organization incorporated in the United States for the sole purpose of promoting and exchanging scholarly work in databases and related fields throughout the world. Since 1992, the Endowment has started to publish a quarterly journal, the VLDB Journal, for disseminating archival research results, which has become one of the most successful journals in the database area. The VLDB Journal is published in collaboration with Springer-Verlag. On various activities, the Endowment closely cooperates with ACM SIGMOD.VLDB conference is a premier annual international forum for data management and database researchers, vendors, practitioners, application developers, and users. The conference features research talks, tutorials, demonstrations, and workshops. It covers current issues in data management, database, and information systems research. Data management and databases remain among the main technological cornerstones of emerging applications of the twenty-first century.

11. ACM International Conference on Web Search and Data Mining (WSDM)

http://www.wsdm-conference.org/

WSDM (pronounced "wisdom") is one of the premier conferences on Web-inspired research involving search and data mining. WSDM is a highly selective conference that includes invited talks, as well as refereed full papers. WSDM

publishes original, high-quality papers related to search and data mining on the Web and the Social Web, with an emphasis on practical yet principled novel models of search and data mining, algorithm design and analysis, economic implications, and in-depth experimental analysis of accuracy and performance.

12. IEEE International Conference on Big Data

http://cci.drexel.edu/bigdata/bigdata2018/index.html

In recent years, "big data" has become a new ubiquitous term. Big data is transforming science, engineering, medicine, healthcare, finance, business, and ultimately our society itself. The IEEE Big Data conference series started in 2013 has established itself as the top tier research conference in big data. It provides a leading forum for disseminating the latest results in big data research, development, and applications.

13. International Conference on Artificial Intelligence and Statistics (AISTATS)

https://www.aistats.org/

AISTATS is an interdisciplinary gathering of researchers at the intersection of computer science, artificial intelligence, machine learning, statistics, and related areas. Since its inception in 1985, the primary goal of AISTATS has been to broaden research in these fields by promoting the exchange of ideas among them. The Society for Artificial Intelligence and Statistics is a nonprofit organization, incorporated in New Jersey (USA), dedicated to facilitating interactions between researchers in AI and statistics. The Society has a governing board, but no general membership. The primary responsibilities of the Society are to maintain the AI-Stats home page on WWW, maintain the AI-Stats electronic mailing list, and organize the biennial International Workshops on Artificial Intelligence and Statistics.

14. ACM Conference on Recommender Systems (RecSys)

https://recsys.acm.org/

The ACM Recommender Systems (RecSys) conference is the premier international forum for the presentation of new research results, systems, and techniques in the broad field of recommender systems. Recommendation is a particular form of information filtering that exploits past behaviors and user similarities to generate a list of information items that is personally tailored to an end user's preferences. As RecSys brings together the main international research groups working on recommender systems, along with many of the world's leading e-commerce companies, it has become the most important annual conference for the presentation and discussion of recommender systems research.

A.3   DATA-MINING FORUMS/BLOGS

1. KDnuggets Forums

http://www.kdnuggets.com/phpBB/index.php

Good resource for sharing experience and asking questions.

2. Data Mining and Predictive Analytics

http://abbottanalytics.blogspot.com/

    The posts on this blog cover topics related to data mining and predictive analytics from the perspectives of both research and industry.

3. AI, Data Mining, Machine Learning and Other Things

http://blog.markus-breitenbach.com/

    This blog discusses machine learning with emphasis on AI and statistics.

4. Data Miners Blog

http://blog.data-miners.com/

    The posts on this blog provide industry-oriented reflections on topics from data analysis and visualization.

5. Data Mining Research

http://www.dataminingblog.com/

    This blog provides a venue for exchanging ideas and comments about data-mining techniques and applications.

6. Machine Learning (Theory)

http://hunch.net/

    A blog dedicated to the various aspects of machine-learning theory and applications.

7. Forrester Big Data Blog

https://go.forrester.com/blogs/category/big-data/

    An aggregation of blogs from company contributors focusing on big data topics.

8. IBM Big Data Hub Blogs

http://www.ibmbigdatahub.com/blogs

    Blogs from IBM thought leaders.

9. Big on Data

http://www.zdnet.com/blog/big-data/

    Andrew Brust, Tony Baer and George Anadiotis cover big data technologies including Hadoop, NoSQL, Data Warehousing, BI, and Predictive Analytics.

10. Deep Data Mining

http://www.deep-data-mining.com/
    Mostly focused on technical aspect of data mining, by Jay Zhou.

11. Insight Data Science Blog

https://blog.insightdatascience.com/
    Blog on latest trends and topics in data science by alumnus of Insight Data Science Fellows Program.

12. Machine Learning Mastery

https://machinelearningmastery.com/blog/
    By Jason Brownlee, on programming and machine learning.

13. Statisfaction

https://statisfaction.wordpress.com/
    A blog by jointly written by PhD students and post-docs from Paris (U. Paris-Dauphine, CREST). Mainly tips and tricks useful in everyday jobs, links to various interesting pages, articles, seminars, etc.

14. The Practical Quant

http://practicalquant.blogspot.com/
    By Ben Lorica, O'Reilly Media Chief Data Scientist, on OLAP analytics, big data, data applications, etc.

15. What's the Big Data

https://whatsthebigdata.com/
    By Gil Press. Gil covers the Big Data space and also writes a column on Big Data and Business in Forbes.

A.4   DATA SETS

This section describes a number of freely available data sets ready for use in data-mining algorithms. We selected few examples for students who are starting to learn data mining, and they would like to practice traditional data-mining tasks. A majority of these data sets are hosted on the UCI Machine Learning Repository. For more data sets look up this repository at http://archive.ics.uci.edu/ml/index.html. *Two additional*

*resources are Stanford SNAP Web data repository (*http://snap.stanford.edu/data/index.html*) and*

*KDD Cup data sets (*http://www.kdd.org/kdd-cup*).*

Classification

Iris Data Set. http://archive.ics.uci.edu/ml/datasets/Iris

The Iris Data Set is a small data set often used in machine learning and data mining. It includes 150 data points each representing measurements of 3 different kinds of iris. The task is to learn to classify iris based on 4 measurements. This data set was used by R. A. Fisher in 1936 as an example for discriminant analysis.

Adult Data Set. http://archive.ics.uci.edu/ml/datasets/Adult

The Adult Data Set contains 48,842 samples extracted from the US Census. The task is to classify individuals as having an income that does or does not exceed $50k/yr. based on factors such as age, education, race, sex, and native country.

Breast Cancer Wisconsin (Diagnostic) Data Set. http://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Diagnostic%29

This data set consists of a number of measurements taken over a "digitized image of a fine needle aspirate (FNA) of a breast mass." There are 569 samples. The task is to classify each data point as benign or malignant.

Bank Marketing Data Set. https://archive.ics.uci.edu/ml/datasets/Bank+Marketing

The data is related with direct-marketing campaigns of a Portuguese banking institution. The marketing campaigns were based on phone calls. Often, more than one contact to the same client was required, in order to access if the product (bank term deposit) would be ("yes") or not ("no") subscribed. The classification goal is to predict if the client will subscribe (yes/no) a term deposit (variable *y*).

Electricity Market (Data Stream Classification). https://sourceforge.net/projects/moa-datastream/files/Datasets/Classification/elecNormNew.arff.zip/download/

This data records the rise and fall of electric price over 24-hour period due to supply and demand. This data set contains 45,312 instances. The task is to predict the change of the price relative to a moving average of the last 24 hours.

Spam Detection (Data Stream Classification). http://www.liaad.up.pt/kdus/downloads/spam-dataset/

This data set represent gradual concept drift with 9324 samples. The labels are legitimate or spam. The ratio between the two classes is 80 : 20

Forrest Cover (Data Stream Classification). https://archive.ics.uci.edu/ml/datasets/Covertype

Predicting forest cover type from cartographic variables only (no remotely sensed data). The actual forest cover type for a given observation (30 × 30 m cell) was determined from US Forest Service (USFS) Region 2 Resource Information System (RIS) data. Independent variables were derived from data originally obtained from US Geological Survey (USGS) and USFS data. Data is in raw form (not scaled) and contains binary (0 or 1) columns of data for qualitative independent variables (wilderness areas and soil types).

Clustering

Bag of Words Data Set. http://archive.ics.uci.edu/ml/datasets/Bag+of+Words
Word counts have been extracted from five document sources: Enron Emails, NIPS full papers, KOS blog entries, NYTimes news articles, and Pubmed abstracts. The task is to cluster the documents used in this data set based on the word counts found. One may compare the output clusters with the sources from which each document came.

US Census Data (1990) Data Set. http://archive.ics.uci.edu/ml/datasets/US+Census+Data+%281990%29
This data set is a 1% sample from the 1990 Public Use Microdata Samples (PUMS). It contains 2,458,285 records and 68 attributes.

Individual Household Electric Power Consumption Data Set. https://archive.ics.uci.edu/ml/datasets/Individual+household+electric+power+consumption
This archive contains 2,075,259 measurements gathered between December 2006 and November 2010 (47 months). It records energy use from three electric meters of the house.

Gas Sensor Dataset at Different Concentrations Data Set. https://archive.ics.uci.edu/ml/datasets/Gas+Sensor+Array+Drift+Dataset+at+Different+Concentrations
This data set contains 13,910 measurements from 16 chemical sensors exposed to 6 gases at different concentration levels. This data set is an extension of the Gas Sensor Array Drift Dataset ([Web Link]), providing now the information about the concentration level at which the sensors were exposed for each measurement.

Regression

Auto MPG Data Set. http://archive.ics.uci.edu/ml/datasets/Auto+MPG
This data set provides a number of attributes of cars that can be used to attempt to predict the "city-cycle fuel consumption in miles per gallon." There are 398 data points and 8 attributes.

Computer Hardware Data Set. http://archive.ics.uci.edu/ml/datasets/Computer+Hardware
This data set provides a number of CPU attributes that can be used to predict relative CPU performance. It contains 209 data points and 10 attributes.

Web Mining

Anonymous Microsoft Web Data. http://archive.ics.uci.edu/ml/datasets/
Anonymous+Microsoft+Web+Data

This data set contains page visits for a number of anonymous users who visited
www.microsoft.com. The task is to predict future categories of pages a user will visit
based on the Web pages previously visited.

KDD Cup 2000. http://www.kdd.org/kdd-cup/view/kdd-cup-2000

This Web site contains five tasks used in a data-mining competition run yearly
called KDD Cup. KDD Cup 2000 uses clickstream and purchase data obtained from
Gazelle.com. Gazelle.com sold *legwear* and *legcare* products and closed their online
store that same year. This Web site provides links to papers and posters of the winners
of the various tasks and outlines their effective methods. Additionally the description
of the tasks provides great insight into original approaches to using data mining with
clickstream data.

Web Page. http://lib.stat.cmu.edu/datasets/bankresearch.zip
Contains 11,000 Web sites from 11 categories.

Text Mining

Reuters-21578 Text Categorization Collection. http://kdd.ics.uci.edu/
databases/reuters21578/reuters21578.html

This is a collection of news articles that appeared on Reuters newswire in 1987.
All of the news articles have been categorized. The categorization provides opportu-
nities to test text classification or clustering methodologies.

20 Newsgroups. http://people.csail.mit.edu/jrennie/20Newsgroups/

The 20 Newsgroups data set contains 20,000 newsgroup documents. These
documents are divided nearly evenly among 20 different newsgroups. Similar to
the Reuters collection, this data set provides opportunities for text classification
and clustering.

Time Series

Dodgers Loop Sensor Data Set. http://archive.ics.uci.edu/ml/datasets/
Dodgers+Loop+Sensor

This data set provides the number of cars counted by a sensor every 5 minutes
over 25 weeks. The sensor was for the Glendale on ramp for the 101 North freeway
in Los Angeles. The goal of this data was to "predict the presence of a baseball game at
Dodgers stadium."

Balloon. http://lib.stat.cmu.edu/datasets/balloon
Data set consisting 2001 readings of radiation from a balloon. The data contains
trend and outliers.

Data for Association Rule Mining

KDD CUP 2009. http://www.kdd.org/kdd-cup/view/kdd-cup-2009

Data from French telecom company Orange to predict the propensity of customers to switch provider (churn), buy new products or services (appetency), or buy upgrades or add-ons proposed to them to make the sale more profitable (upselling).

Connect-4 Data Set. https://archive.ics.uci.edu/ml/datasets/Connect-4

This database contains all legal 8-ply positions in the game of connect-4 in which neither player has won yet and in which the next move is not forced. *x* is the first player; o the second. The outcome class is the game theoretical value for the first player.

A.5   COMERCIALLY AND PUBLICLY AVAILABLE TOOLS

This summary of some publicly available commercial data-mining products is being provided to help readers better understand what software tools can be found on the market and what their features are. It is not intended to endorse or critique any specific product. Potential users will need to decide for themselves the suitability of each product for their specific applications and data-mining environments. This is primarily intended as a starting point from which users can obtain more information. There is a constant stream of new products appearing in the market and hence this list is by no means comprehensive. Because these changes are very frequent, the author suggests the following web site for information about the latest tools and their performances: http://www.kdnuggets.com.

1. **Free Software**
DataLab

   – Publisher: Epina Software Labs (http://datalab.epina.at/en_home.html)
   – DataLab, a complete and powerful data-mining tool with a unique data exploration process, with a focus on marketing and interoperability with SAS. There is a public version for students.

 DBMiner

   – Publisher: Simon Fraser University (http://ddm.cs.sfu.ca)
   – DBMiner is a publicly available tool for data mining. It is a multiple-strategy tool, and it supports methodologies such as clustering, association rules, summarization, and visualization. DBMiner uses Microsoft SQL Server 7.0 Plato and runs on different Windows platforms.

GenIQ Model

   – Publisher: DM STAT-1 Consulting (www.geniqmodel.com)

– GenIQ Model uses machine learning for regression task, automatically performs variable selection and new variable construction, and then specifies the model equation to "optimize the decile table."

NETMAP

– Publisher: http://sourceforge.net/projects/netmap
– NETMAP is a general purpose, information-visualization tool. It is most effective for large, qualitative, text-based data sets. It runs on Unix workstations.

RapidMiner

– Publisher: Rapid-I (http://rapid-i.com)
– Rapid-I provides software, solutions, and services in the fields of predictive analytics, data mining, and text mining. The company concentrates on automatic intelligent analyses on a large-scale base, i.e. for large amounts of structured data like database systems and unstructured data like texts. The open-source data-mining specialist Rapid-I enables other companies to use leading-edge technologies for data mining and business intelligence. The discovery and leverage of unused business intelligence from existing data enables better informed decisions and allows for process optimization.

SIPNA

– Publisher: http://eric.univ-lyon2.fr/~ricco/sipina.html
– Sipina-W is publicly available software that includes different traditional data-mining techniques such as CART, Elisee, ID3, C4.5, and some new methods for generating decision trees.

SNNS

– Publisher: University of Stuttart (http://www.ra.cs.uni-tuebingen.de/SNNS/)
– SNNS is a publicly available software. It is a simulation environment for research on and application of artificial neural networks. The environment is available on Unix and Windows platforms.

TiMBL

– Publisher: http://ilk.uvt.nl/timbl/
– TiMBL is a publicly available software. It includes several memory-based learning techniques for discrete data. A representation of the training set is explicitly stored in memory, and new cases are classified by extrapolation from the most similar cases.

TOOLDIAG

– Publisher: http://sites.google.com/site/tooldiag/Home

– TOOLDIAG is a publicly available tool for data mining. It consists of several programs in C for statistical pattern recognition of multivariate numeric data. The tool is primary oriented toward classification problems.

Weka

– Publisher: University of Waikato (http://www.cs.waikato.ac.nz/ml/)
– Weka is a software environment that integrates several machine-learning tools within a common framework and a uniform GUI. Classification and summarization are the main data-mining tasks supported by the Weka system.

Orange

– Publisher: https://orange.biolab.si/
– Orange is an open-source software for both novice and expert. It supports interactive visualization, visual programming, and add-ons for extendibility.

KNIME

– Publisher: https://www.knime.com
– KNIME is an open-source software that has more than 2000 modules, hundreds of examples, and a vast range of integrated tools. KINME supports scripting integration, big data, machine learning, complex data types, and more.

OpenStat

– Publisher: http://openstat.info/OpenStatMain.htm
– OpenStat contains a large variety of parametric, nonparametric, multivariate, measurement, statistical process control, financial, and other procedures. One can also simulate a variety of data for tests, theoretical distributions, multivariate data, etc. You will want to explore all of these options once you acquire the program.

2. **Commercial Software WITH Trial Version**
Alice d'Isoft

– Vendor: Isoft (www.alice-soft.com)
– ISoft provides a complete range of tools and services dedicated to analytical CRM, behavioral analysis, data modeling and analysis, data mining, and data morphing.

ANGOSS' Suite

– Vendor: Angoss Software Corp. (www.datawatch.com/in-action/angoss/)
– ANGOSS' Suite consists of KnowledgeSTUDIO® and KnowledgeSEEKER®. KnowledgeSTUDIO® is an advanced data mining and predictive analytics suite for all phases of the model development and deployment cycle—profiling, exploration, modeling, implementation, scoring, validation, monitoring, and building

scorecards—all in a high-performance visual environment. KnowledgeSTUDIO is widely used by marketing, sales and risk analysts providing business users and expert analysts alike with a powerful, scalable, and complete data-mining solution. KnowledgeSEEKER® is a single-strategy desktop or client/server tool relying on a tree-based methodology for data mining. It provides a nice GUI for model building and letting the user explore data. It also allows users to export the discovered data model as text, SQL query, or Prolog program. It runs on Windows and Unix platforms and accepts data from a variety of sources.

BayesiaLab

– Vendor: Bayesia (www.bayesia.com)
– BayesiaLab, a complete and powerful data mining tool based on Bayesian networks, including data preparation, missing values imputation, data and variable clustering, and unsupervised and supervised learning.

DataEngine

– Vendor: MIT GmbH (www.dataengine.de)
– DataEngine is a multiple-strategy data-mining tool for data modeling, combining conventional data-analysis methods with fuzzy technology, neural networks, and advanced statistical techniques. It works on the Windows platform.

Evolver^{TM}

– Vendor: Palisade Corp. (www.palisade.com)
– Evolver is a single-strategy tool. It uses genetic algorithm technology to solve complex optimization problems. This tool runs on all Windows platforms, and it is based on data stored in Microsoft Excel tables.

GhostMiner System

– Vendor: FQS Poland (https://www.g6g-softwaredirectory.com/ai/data-mining/20154-FQS-Poland-Fujitsu-GhostMiner.php)
– GhostMiner, complete data-mining suite, including k-nearest neighbors, neural nets, decision tree, neurofuzzy, SVM, PCA, clustering, and visualization.

NeuroSolutions

– Vendor: NeuroDimension Inc. (www.neurosolutions.com)
– NeuroSolutions combines a modular, icon-based network design interface with an implementation of advanced learning procedures, such as recurrent backpropagation and backpropagation through time, and it solves data-mining problems such as classification, prediction, and function approximation. Some other notable features include C++ source code generation, customized components through DLLs, a comprehensive macro language, and Visual Basic accessibility through OLE Automation. The tool runs on all Windows platforms.

Oracle Data Mining

– Vendor: Oracle (www.oracle.com)
– Oracle Data Mining (ODM)—an option to Oracle Database 11g Enterprise Edition—enables customers to produce actionable predictive information and build integrated business intelligence applications. Using data-mining functionality embedded in Oracle Database 11g, customers can find patterns and insights hidden in their data. Application developers can quickly automate the discovery and distribution of new business intelligence—predictions, patterns, and discoveries—throughout their organization.

Optimus RP

– Vendor: Golden Helix Inc. (www.goldenhelix.com)
– Optimus RP uses formal inference-based recursive modeling (recursive partitioning based on dynamic programming) to find complex relationships in data and to build highly accurate predictive and segmentation models.

Partek Software

– Vendor: Partek Inc. (www.partek.com)
– Partek Software is a multiple-strategy data-mining product. It is based on several methodologies including statistical techniques, neural networks, fuzzy logic, genetic algorithms, and data visualization. It runs on Unix platforms.

Rialto™

– Vendor: Exeura (http://www.exeura.eu/en/products/rialto/)
– Exeura Rialto™ provides comprehensive support for the entire data mining and analytics lifecycle at an affordable price in a single, easy-to-use tool.

Salford Predictive Miner

– Vendor: Salford Systems (http://salford-systems.com)
– Salford Predictive Miner (SPM) includes CART®, MARS, TreeNet, and Random Forests, and powerful new automation and modeling capabilities. CART® is a robust, easy-to-use decision tree that automatically sifts large, complex databases, searching for and isolating significant patterns and relationships. Multivariate adaptive regression splines (MARS) focuses on the development and deployment of accurate and easy-to-understand regression models. TreeNet demonstrates remarkable performance for both regression and classification and can work with varying sizes of data sets, from small to huge, while readily managing a large number of columns. Random Forests is best suited for the analysis of complex data structures embedded in small to moderate data sets containing typically less than 10,000 rows but allowing for more than 1 million

columns. Random Forests has therefore been enthusiastically endorsed by many biomedical and pharmaceutical researchers.

Synapse

- Vendor: Peltarion (www.peltarion.com)
- Synapse, a development environment for neural networks and other adaptive systems, supporting the entire development cycle from data import and preprocessing via model construction and training to evaluation and deployment, allows deployment as .NET components.

SOMine

- Vendor: Viscovery (www.viscovery.net)
- This single-strategy data-mining tool is based on self-organizing maps and is uniquely capable of visualizing multidimensional data. SOMine supports clustering, classification, and visualization processes. It works on all Windows platforms.

TIBCO Spotfire® Professional

- Vendor: TIBCO Software Inc (https://www.tibco.com/)
- TIBCO Spotfire® Professional makes it easy to build and deploy reusable analytic applications over the Web or perform pure ad hoc analytics, driven on the fly by your own knowledge, intuition, and desire to answer the next question. Spotfire analytics does all this by letting you interactively query, visualize, aggregate, filter, and drill into data sets of virtually any size. Ultimately you will reach faster insights with Spotfire and bring clarity to business issues or opportunities in a way that gets all the decision-makers on the same page quickly.

Alteryx

- Publisher: https://www.alteryx.com/
- Alteryx is a leading software vendor for self-servicing machine learning. Perform machine-learning task by drag and drop, and then share the results through organization in a matter of hours.

Neural Designer

- Publisher: https://www.neuraldesigner.com/
- Neural Designer simplifies the task of building application using neural networks.

Analance

- Publisher: https://analance.ducenit.com/
- Self-service data analytics software that is easy to use and supports guided workflows. Support interactive result analysis and visualization.

Microsoft Azure Machine Learning Studio

– Publisher: https://azure.microsoft.com/en-us/services/machine-learning-studio/
– Machine Learning Studio is a browser-based, easy-to-use machine-learning platform. Drag and drop actions to perform actions such as preprocessing, model training, and performance testing.

IBM Watson Machine Learning

– Publisher: https://www.ibm.com/cloud/machine-learning
– You can use your own data to train your model using the IBM Watson Machine Learning platform.

3. **Commercial Software WITHOUT Trial Version**

AdvancedMiner

– Vendor: StatConsulting (http://algolytics.com/)
– AdvancedMiner is a platform for data mining and analysis, featuring modeling interface (OOP script, latest GUI design, advanced visualization) and grid computing.

Affinium Model

– Vendor: Unica Corp. (https://www-01.ibm.com/support/docview.wss?uid=swg27027009&aid=1)
– Affinium Model (from Unica) includes valuator, profiler, response modeler, and cross-seller. Unica provides innovative marketing solutions that turn your passion for marketing into business success. Our unique interactive marketing approach incorporates customer and Web analytics, centralized decision, cross channel execution, and integrated marketing operations. More than 1000 organizations worldwide depend on Unica.

IBM SPSS Modeler Professional

– Vendor: SPSS Inc., an IBM company
   (https://www.ibm.com/analytics/data-science/predictive-analytics/spss-statistical-software)
– IBM SPSS Modeler Professional has optimization techniques for large data sets, including boosting and bagging, which improve model stability and accuracy. It also enhanced visualization for key algorithms, including neural net and decision tree. In particular, new interactive visualization for key algorithms and ensemble models is offered in order to make results easier to understand and communicate.

DataDetective

– Vendor: Sentient Information Systems (www.sentient.nl)
– DataDetective, the powerful yet easy-to-use data-mining platform and the crime analysis software of choice for the Dutch police.

DeltaMaster

- Vendor: Bissantz & Company GmbH (www.bissantz.com)
- Delta Miner is a multiple-strategy tool supporting clustering, summarization, deviation detection, and visualization processes. A common application is the analysis of financial controlling data. It runs on Windows platforms and it integrates new search techniques and "business intelligence" methodologies into an OLAP front end.

EWA Systems

- Vendor: EWA Systems Inc. (http://www.ewa-gsi.com/)
- EWA Systems provide enterprise analytics solutions: math and statistics libraries, data mining, text mining, optimization, visualization, and rule engine software are all available from one coordinated source. EWA Systems' ability to tackle such a broad range of analytical solutions means our clients gain efficiencies in purchasing software that fits together modularly, as well as incurring decreased consulting costs. Our tools have been deployed worldwide in industries as diverse as financial analysis, e-commerce, manufacturing, and education where their outstanding performance and quality is unrivaled. Whether you are using a single PC or a supercomputer, EWA Systems has the numerical software capabilities to fit your need.

FastStats^{TM}

- Vendor: APTECO Limited (www.apteco.com)
- FastStats Suite, marketing analysis products, including data mining, customer profiling, and campaign management.

IBM Intelligent Miner

- Vendor: IBM (www.ibm.com)
- DB2 Data Warehouse Edition (DWE) is a suite of products that combines the strength of DB2 Universal Database™ (DB2 UDB) with the powerful business intelligence infrastructure from IBM®. DB2 Data Warehouse Edition provides a comprehensive business intelligence platform with the tools that your enterprise and partners need to deploy and build next-generation analytic solutions.

KnowledgeMiner

- Vendor: KnowledgeMiner Software (www.knowledgeminer.com)
- KnowledgeMiner, a self-organizing modeling tool that uses GMDH neural nets and artificial intelligence to easily extract knowledge from data (MacOS).

MATLAB NN Toolbox

- Vendor: MathWorks Inc. (www.mathworks.com)

– A MATLAB extension implements an engineering environment for research in neural networks and its design, simulation, and application. It offers various network architectures and different learning strategies. Classification and function approximations are typical data-mining problems that can be solved using this tool. It runs on Windows, Mac, and Unix platforms.

Predictive Data Mining Suite

– Vendor: Predictive Dynamix (www.predx.com)
– Predictive Data Mining Suite integrates graphical and statistical data analysis with modeling algorithms including neural networks, clustering, fuzzy systems, and genetic algorithms.

Enterprise Miner

– Vendor: SAS Institute Inc. (www.sas.com)
– SAS (Enterprise Miner) represents one of the most comprehensive sets of integrated tools for data mining. It also offers a variety of data manipulation and transformation features. In addition to statistical methods, the SAS data-mining solution employs neural networks, decision trees, and SAS Webhound that analyzes Website traffic. It runs on Windows and Unix platforms, and it provides a user-friendly GUI front end to the SEMMA (Sample, Explore, Modify, Model, Assess).

SPAD

– Vendor: Coheris (www.coheris.fr)
– SPAD provides powerful exploratory analyses and data-mining tools, including PCA, clustering, interactive decision trees, discriminant analyses, neural networks, text mining, and more, all via user-friendly GUI.

Viscovery Data Mining Suite

– Vendor: Viscovery (www.viscovery.net)
– The Viscovery® Data Mining Suite offers a selection of software for predictive analytics and data mining designed to comprehensively address the needs of business and technical users. Workflows support the generation of high-performance predictive models that may be integrated in real-time and updated automatically. The Viscovery Data Mining Suite comprises the modules—Profiler, Predictor, Scheduler, Decision Maker, One(2)One Engine—for the realization of predictive analytics and data-mining applications.

Warehouse Miner

– Vendor: Teradata Corp. (www.teradata.com)
– Warehouse Miner provides different statistical analyses, decision-tree methods, and regression methodologies for in-place mining on a Teradata database-management system.

A.6   WEB SITE LINKS

General Web Sites

Web Site	Description
www.ics.uci.edu	A comprehensive machine-learning site. Popular for its large repository of standard data sets and machine-learning programs for experimental evaluation
www.cs.cmu.edu/Groups/AI/html	This address collects files, programs, and publications of interest to the artificial intelligence research community
https://research.reading.ac.uk/dsai/	An online resource to AI programs, software, data sets, bibliographies, and links
http://archive.ics.uci.edu/ml/	Repositories focusing on the scientific study of machine learning
www.kdnuggets.com	This site contains information about data-mining activities and pointers to past and current research. It maintains a guide to commercial and public-domain tools for data mining. It also provides links to companies supporting software, consulting, and data-mining services
https://www.webopedia.com/	This site provides news, articles, and other useful sites in data-mining applications.
www.research.microsoft.com	*Journal of Data Mining & Knowledge Discovery:* The journal consolidates papers in both the research and practice of knowledge discovery, surveys of implementation techniques and application papers.
http://www.kdd.org/	Web site for machine-learning and data-mining conference KDD. Also hosts previous KDD Cup Datasets
https://snap.stanford.edu/data/	Stanford Large Network Dataset Collection

Web Sites for Data-Mining Software Tools

Web Site	Data-Mining Tool
http://algolytics.com/products/advancedminer/	AdvancedMiner
https://www-01.ibm.com/support/docview.wss?uid=swg27027009	Affinium Model (sold to IBM)
www.dazsi.com	AgentBase/Marketeer
https://isoft.fr/en/isoft-welcome/	Alice d'Isoft
https://www.alteryx.com/	Alteryx
https://analance.ducenit.com/	Analance
www.openchannelsoftware.com	Autoclass III
www.bayesia.com	BayesiaLab
www.kmi.open.ac.uk/projects/bkd/	Bayesian Knowledge Discoverer

Web Site	Data-Mining Tool
http://salford-systems.com/cart.php	CART
www.spss.com/clementine	Clementine (IBM)
http://www.oracle.com/technetwork/apps-tech/darwin-097216.html	Darwin
www.sentient.nl/?dden	DataDetective
http://www.datamind.biz/	DataMind
http://www.datasage.com/ds/f?p=111:1::::::	Datasage
www.bissantz.de	Delta Miner
www.pilotsw.com	Discovery
www.palisade.com/	Evolver
www.apteco.com	FastStats Suite
www.urbanscience.com	GainSmarts
www.geniqmodel.com/	GenIQ Model
www.goldenhelix.com	Golden Helix Optimus RP
https://www.ibm.com/cloud/machine-learning	IBM Watson Machine Learning
www.software.ibm.com	Intelligent Miner
www.acknosoft.com	KATE Tools
www.ncr.com	Knowledge Discovery Workbench
www.dialogis.de	Kepler
https://www.knime.com/	KNIME Machine Learning Software
www.dialogis.de	KnowledgeMiner
https://www.datawatch.com/in-action/angoss/	KnowledgeSeeker Datawatch
www.mathworks.com/products/neuralnet	MATLAB neural-network toolbox
https://azure.microsoft.com/en-us/services/machine-learning-studio/	Microsoft Azure Machine Learning Studio
www.neurosolutions.com	Neuro Net
www.neuralware.com/	NeuralWorks Professional II/PLUS
https://www.neuraldesigner.com/	Neural Designer
www.wardsystems.com/	NeuroShell2/NeuroWindows
http://openstat.info/OpenStatMain.htm	OpenStat
https://orange.biolab.si/	Orange Data Mining Software
www.predx.com	Predictive Data Mining Suite
www.rapid-i.com	RapidMiner
www.sas.com	SAS Enterprise Miner
https://www.ibm.com/products/cognos-analytics	IBM Cognos Analytics
www.spss.com	SPSS (IBM)
http://statlab.yale.edu/	STATlab
https://www.fernuni-hagen.de/BWLOR/spirit/index.php	SPIRIT
www.mitgmbh.de	WINROSA
www.wizsoft.com	WizWhy

Data-Mining Vendors

Data-Mining Vendor	Address	Web Site
Datawatch	1820 E. Big Beaver Rd. Troy MI 48083 United States	https://www.datawatch.com/in-action/angoss/
Business Objects, Inc. (sold to SAP)	20813 Stevens Creek Blvd., Suite 100, Cupertino, CA 95014, USA	https://www.sap.com/products/analytics/business-intelligence-bi.html
Cognos Corp. (Sold to IBM)	67 S. Bedford St., Suite 200, W. Burlington, MA 01803, USA	https://www.ibm.com/products/cognos-analytics
IBM Corp.	Old Orchard Road, Armonk, NY 10504, USA	www.ibm.com
Integral Solutions Ltd.	Berk House, Basing View, Basingstoke, Hampshire RG21 4RG, UK	www.isl.co.uk
ISoft	Route de l'Orme Les Algorithmes Bâtiment Euclide 91190 Saint-Aubin France	https://isoft.fr/en/isoft-welcome/
NeuralWare Inc.	NeuralWare, 409 Elk Street, Suite 200, Carnegie, PA 15106-2627 USA	www.neuralware.com
Pilot Software, Inc. (Sold to SAP)	One Canal Park, Cambridge, MA 02141, USA	www.pilotsw.com
SPSS, Inc. (Sold to IBM)	444 N. Michigan Ave., Chicago, IL 60611-3962, USA	https://www.ibm.com/analytics/data-science/predictive-analytics/spss-statistical-software
SAS Institute Inc.	SAS Campus Dr., Cary, NC 27513-2414, USA	www.sas.com
Sisense	1359 Broadway, 4th Floor, New York, NY, 10018, USA	https://www.sisense.com
Maxus Systems International Inc.	318 Town Line Rd, Mendon, VT 05701, USA	www.maxussystems.com
Visualize, Inc.	452 Bonnie Briar, Suite 100,Birmingham, MI 48009	https://visualize.com/
Data Description, Inc.	PO Box 4555, Ithaca, NY 14850, USA	www.datadesk.com
i2 Ltd. Sold to IBM	Breaks House Mill Court, Great Shelford, Cambridge, CB2, SLD, UK	https://www-01.ibm.com/software/uk/industry/i2software/

(*Continued*)

Data-Mining Vendor	Address	Web Site
Advanced Visual Systems, Inc.	2 Burlington Woods Drive, Suite 100,Burlington, MA 01803	www.avs.com
Imagix Corp.	6025 White Oak Lane, San Luis Obispo, CA 93401, USA	www.imagix.com
Helsinki University of Technology (merged into Aalto University)	Neural Networks Research Center, P. O. Box 1000, FIN-02015 HUT, Finland	https://www.aalto.fi/en/aalto-university/history
IBM Haifa Research Laboratory	Matam, Haifa 31905, Israel	http://www.research.ibm.com/labs/haifa/
Infospace, Inc.	1501 Main Street, Suite 201, Venice, CA 90291, USA	http://www.infospace.com/
GR-FX Pty Limited	P. O. Box 2121, Clovelly, NSW, 2031 Australia	https://www.zoominfo.com/c/gr-fx-pty-limited/48648893
Analytic Technologies	Analytic Technologies, P.O. Box 910359, Lexington, KY 40513, USA	analytictech.com
Artificial Intelligence Software SpA	Via Carlo Esterle, 9-20132 Milano, Italy	www.iunet.it/ais
General Dynamics	3150 Fairview Park Drive Falls Church, VA 22042, USA	https://www.gdit.com/
Quadstone Ltd.	16 Chester Street, Edinburgh, EH3 7RA, Scotland	www.quadstone.co.uk
Perspecta, Inc	15052 Conference Center Drive, Chantilly, VA 20151, USA	www.perspecta.com
Dynamic Diagrams	12 Bassett Street, Providence, RI 02903, USA	https://www.dynamicdiagrams.com/
NetScout Systems, Inc.	4 Technology Park Drive, Westford, MA 01886, USA	https://www.netscout.com /
MapInfo Corp.	1 Global View, Troy, NY 12180, USA	https://www.pitneybowes.com/us/location-intelligence/geographic-information-systems/mapinfo-pro.html

(*Continued*)

Data-Mining Vendor	Address	Web Site
Information Builders, Inc.	Two Penn Plaza, New York, NY 10121-2898	www.informationbuilders.com
Prism Solutions, Inc.	7455 Arroyo Crossing Pkwy, Suite 220,Las Vegas, Nevada 89113	www.prismsol.com/
Oracle Corp.	500 Oracle Parkway, Redwood Shores, CA 94086 USA	www.oracle.com/index.html
Microsoft Corporation	One Microsoft Way, Redmond, WA 98052, USA	www.microsoft.com/en-us/
Computer Associates International, Inc.	One Computer Associates Plaza Islandia, NY 11788-7000, USA	www.ca.com/us.html

APPENDIX B
DATA-MINING APPLICATIONS

Many businesses and scientific communities are currently employing data-mining technology. Their number continues to grow, as more and more data-mining success stories become known. Here we present a small collection of real-life examples of data-mining implementations from the business and scientific world. We also present some pitfalls of data mining to make readers aware that this process needs to be applied with care and knowledge (both about the application domain and about the methodology) to obtain useful results.

In the previous chapters of this book, we have studied the principles and methods of data mining. Since data mining is a young discipline with wide and diverse applications, there is a still a serious gap between the general principles of data mining and the domain-specific knowledge required to apply it effectively. In this appendix, we examine a few application domains illustrated by the results of data-mining systems that have been implemented.

B.1 DATA MINING FOR FINANCIAL DATA ANALYSIS

Most banks and financial institutions offer a wide variety of banking services such as checking, savings, business and individual customer transactions, investment services, credits, and loans. Financial data, collected in the banking and financial industry, are often relatively complete and reliable and of a high quality, which facilitates systematic data analysis and data mining to improve a company's competitiveness.

In the banking industry, data mining is used heavily in the areas of modeling and predicting credit fraud, in evaluating risk, in performing trend analyses, in analyzing profitability, and in helping with direct-marketing campaigns. In the financial

markets, neural networks have been used in forecasting stock prices, options trading, rating bonds, portfolio management, commodity-price prediction, and mergers and acquisitions analyses; it has also been used in forecasting financial disasters. Daiwa Securities, NEC Corporation, Carl & Associates, LBS Capital Management, Walkrich Investment Advisors, and O'Sullivan Brothers Investments are only a few of the financial companies who use neural-network technology for data mining. A wide range of successful business applications has been reported, although the retrieval of technical details is not always easy. The number of investment companies and banks that mine data is far more extensive than the list mentioned earlier, but you will not often find them willing to be referenced. Usually, they have policies not to discuss it. Therefore, finding articles about banking companies who use data mining is not an easy task, unless you look at the SEC reports of some of the data-mining companies who sell their tools and services. There, you will find customers such as Bank of America, First USA Bank, Wells Fargo Bank, and U.S. Bancorp.

The widespread use of data mining in banking has not been unnoticed. For example, fraud costs industries billions of dollars, so it is not surprising to see that systems have been developed to combat fraudulent activities in such areas as credit card, stock market, and other financial transactions. Fraud is an extremely serious problem for credit card companies. For example, Visa and MasterCard lost over $700 million in 1 year from fraud. A neural-network-based credit card fraud-detection system implemented in Capital One has been able to cut the company's losses from fraud by more than 50%. Several successful data-mining systems are explained here to support the importance of data-mining technology in financial institutions.

The term "robo-advisor" was essentially not known concept just 5 years ago, but it is now commonplace in the financial landscape. The term is a little bit misleading because it does not involve robots at all. Rather, robo-advisors, developed by companies such as *Betterment* and *Wealthfront*, are smart algorithms built to calibrate a financial portfolio to the goals and risk tolerance of each specific user. Users enter their goals, for example, retiring at age 65 with $250,000.00 in savings, and also age, income, and current financial assets. The intelligent advisor algorithm then spreads investments across asset classes and financial instruments in order to reach the user's goals. The system calibrates to changes in the user's goals and to real-time changes in the market, aiming always to find the best fit for the user's original goals. Robo-advisors have gained significant traction with millennial consumers who do not need a physical advisor to feel comfortable investing and who are less able to validate the fees paid to human advisors.

Additional trend of big data applications, which started with financial industry, is spreading through many other domains as a blockchain technology. A blockchain is essentially a distributed database of records for all transactions or digital events that have been executed and shared among participating parties. Each transaction in the public database is verified by consensus of a majority of the participants in the system. Once entered, information can never be erased. The blockchain contains a certain and verifiable record of every single transaction ever made. Bitcoin, the decentralized

peer-to-peer digital currency, is the most popular example that uses blockchain technology. The digital currency bitcoin is highly controversial, but the underlying blockchain technology has worked flawlessly and found wide range of applications in both financial and nonfinancial world.

The main hypothesis is that the blockchain establishes a system of creating a distributed consensus in the digital online world. This allows participating entities to know for certain that a digital event happened by creating an irrefutable record in a public ledger. It enables development of a democratic open and scalable digital economy from a centralized one. There are tremendous application opportunities in this disruptive technology, and the revolution in this space has just begun.

Due to the growing role of social responsibility and security on the Internet, the blockchain technologies are becoming increasingly relevant. In a system using blockchain, it is nearly impossible to forge any digital transactions, so the credibility of such systems will surely strengthen. As the initial hype around blockchain in the financial services' industry will slow down, we will see many more potential use cases for the government, healthcare, manufacturing, and other industries. For example, blockchain strongly influences the intellectual property management and opens new insights in protection from copyright infringement:

US Treasury Department

Worth particular mention is a system developed by the Financial Crimes Enforcement Network (FINCEN) of the US Treasury Department called "FAIS." FAIS detects potential money-laundering activities from a large number of big cash transactions. The Bank Secrecy Act of 1971 required the reporting of all cash transactions greater than $10,000, and these transactions, of about 14 million a year, are the basis for detecting suspicious financial activities. By combining user expertise with the system's rule-based reasoner, visualization facilities, and association-analysis module, FIAS uncovers previously unknown and potentially high-value leads for possible investigation. The reports generated by the FIAS application have helped FINCEN uncover more than 400 cases of money-laundering activities, involving more than $1 billion in potentially laundered funds. In addition, FAIS is reported to be able to discover criminal activities that law enforcement in the field would otherwise miss, e.g. connections in cases involving nearly 300 individuals, more than 80 front operations, and thousands of cash transactions.

Mellon Bank, USA

Mellon Bank has used the data on existing credit card customers to characterize their behavior, and they try to predict what they will do next. Using IBM Intelligent Miner, Mellon developed a credit card-attrition model to predict which customers will stop using Mellon's credit card in the next few months. Based on the prediction results, the bank can take marketing actions to retain these customers' loyalty.

Capital One Financial Group

Financial companies are one of the biggest users of data-mining technology. One such user is Capital One Financial Corp., one of the nation's largest credit card issuers. It offers 3000 financial products, including secured, joint, co-branded, and college-student cards. Using data-mining techniques, the company tries to help market and sell the most appropriate financial product to 150 million potential prospects residing in its over 2-terabyte Oracle-based data warehouse. Even after a customer has signed up, Capital One continues to use data mining for tracking the ongoing profitability and other characteristics of each of its customers. The use of data mining and other strategies has helped Capital One expand from $1 billion to $12.8 billion in managed loans over 8 years. An additional successful data-mining application at Capital One is fraud detection.

American Express

Another example of data mining is at American Express, where data warehousing and data mining are being used to cut spending. American Express has created a single Microsoft SQL Server database by merging its worldwide purchasing system, corporate purchasing card, and corporate card databases. This allows American Express to find exceptions and patterns to target for cost cutting. One of the main applications is loan application screening. American Express used statistical methods to divide loan applications into three categories: those that should definitely be accepted, those that should definitely be rejected, and those that required a human expert to judge. The human experts could correctly predict if an applicant would, or would not, default on the loan in only about 50% of the cases. Machine learning produced rules that were much more accurate—correctly predicting default in 70% of the cases—and that were immediately put into use.

MetLife, Inc.

MetLife's intelligent text analyzer has been developed to help automate the underwriting of 260,000 life insurance applications received by the company every year. Automation is difficult because the applications include many freeform text fields. The use of keywords or simple parsing techniques to understand the text fields has proven to be inadequate, while the application of full semantic natural-language processing was perceived to be too complex and unnecessary. As a compromise solution, the "information-extraction" approach was used in which the input text is skimmed for specific information relevant to the particular application. The system currently processes 20,000 life insurance applications a month, and it is reported that 89% of the text fields processed by the system exceed the established confidence-level threshold.

Bank of America (USA)

Bank of America is one of the world's largest financial institutions. With approximately 59 million consumer and small business relationships, 6,000 retail banking

offices, and more than 18,000 ATMs, Bank of America is among the world's leading wealth management companies and is a global leader in corporate and investment banking and trading across a broad range of asset classes. Bank of America identi- fied savings of $4.8 million in 2 years (a 400% return on investment) from use of a credit risk management system provided by SAS institute consultants and based on statistical and data-mining analytics ["Predicting Returns from the Use of Data Min- ing to Support CRM," http://insight.nau.edu/WhitePapers.asp]. They has also devel- oped profiles of most valuable accounts, with relationship managers being assigned to the top 10% of the bank's customers in order to identify opportunities to sell them additional services ["Using Data Mining on the Road to Successful BI, Part 3", Information Management Special Reports, October 2004]. To retain deposits, the Global Wealth and Investment Management division has used KXEN Analytic Framework in identifying clients likely to move assets and then creating offers con- ducive to retention ["KXEN Analytic Framework", Information Management Mag- azine, July/Aug 2009].

B.2 DATA MINING FOR THE TELECOMUNICATION INDUSTRY

The telecommunication industry has quickly evolved from offering local and long- distance telephone services to providing many other comprehensive communication services including voice, fax, pager, cellular phone, images, e-mail, computer and Web data transmission, and other data traffic. The integration of telecommunications, computer networks, Internet, and numerous other means of communication and com- puting is underway. The U.S. Telecommunications Act of 1996 allowed Regional Bell Operating Companies to enter the long-distance market as well as offer "cable-like" services. The European Liberalization of Telecommunications Services has been effective from the beginning of 1998. Besides deregulation, there has been a sale by the FCC of airwaves to companies pioneering new ways to communicate. The cellular industry is rapidly taking on a life of its own. With all this deregulation of the telecommunication industry, the market is expanding rapidly and becoming highly competitive.

The hypercompetitive nature of the industry has created a need to understand cus- tomers, to keep them, and to model effective ways to market new products. This cre- ates a great demand for data mining to help understand the new business involved, identify telecommunication patterns, catch fraudulent activities, make better use of resources, and improve the quality of services. In general, the telecommunications industry is interested in answering some strategic questions through data-mining applications such as the following:

- How does one retain customers and keep them loyal as competitors offer special offers and reduced rates?
- Which customers are most likely to churn?
- What characteristics indicate high-risk investments, such as investing in new fiber optic lines?

- How does one predict whether customers will buy additional products like cellular services, call waiting, or basic services?
- What characteristics differentiate our products from those of our competitors?

Companies like AT&T, AirTouch Communications, and AMS Mobile Communication Industry Group have announced the use of data mining to improve their marketing activities. There are several companies including Lightbridge and Verizon that use data-mining technology to look at cellular fraud for the telecommunications industry. Another trend has been to use advanced visualization techniques to model and analyze wireless-telecommunication networks.

The tendencies in the communication technologies indicates that the text communication became the socially acceptable form of personal interaction. People increasingly prefer chatting rather than personal contacts or even making phone calls. The idea of chatbots appeared first in the 1960s. But only after more than half a century passed we can confirm that the world is ready for their implementation into the real life. A chatbot is a complex computer program that conducts a conversation in natural language via written text or generated voice, understands the intent of the user, and sends an automatic response based on business rules and data of the organization for which the chatbot is developed.

All the technology leaders, including Microsoft, Facebook, Google, Amazon, IBM, Apple, and Samsung, have created open platforms and interfaces for the chatbot acceptance by society. Siri was introduced in 2010, IBM Watsons started in 2011, and the pilot version of the Bixby Samsung voice assistant appeared in smartphones in 2012. Alexa has been learning to answer the questions since 2014, and the Google Assistant has gained its modern shape in 2016. The excitement about chatbots is not weakening. More than 2 billion of business-related messages are sent through Facebook Messenger chats.

Part of the reason behind this success is the ease of use and the range of services that chatbots comes pre-loaded with. From streaming music on Spotify and ordering a taxi on Uber to seeking medical advice from WebMD, they does it all through a simple conversation. The customer-service chatbot provides a solution to a two additional dimensions: (1) *scalability* of solution that enables personalized interactions usually not supported at scale and (2) *speed* enabling customers to expect instant services. Chatbots are more and more involved in our daily lives: our experiences—from conversations to entertainment to shopping—will be delivered by someone who really knows and understands user preferences. This someone will be able to preempt user needs, moods, likes, and dislikes. This someone is becoming a friend, a confidant, sometimes doctor, or a legal advisor.

While the new trends in communications are target of variety of IT and other companies, selected examples of data-mining applications in the telecommunication industry follow:

Cablevision Systems, Inc.

Cablevision Systems Inc., a cable TV provider from New York, was concerned about its competitiveness after deregulation allowed telecom companies into the cable industry. As a consequence, it decided that it needed a central data repository so that its marketing people could have faster and more accurate access to data. Using data mining, the marketing people at Cablevision were able to identify nine primary customer segments among the company's 2.8 million customers. This included customers in the segment that are likely to "switch" to another provider. Cablevision also focused on those segments most likely to buy its offerings for new services. The company has used data-mining to compare the profiles of two sets of targeted customers—those who bought new services and those who did not. This has led the company to make some changes in its messages to customers, which, in turn, has led to a thirty percent increase in targeted customers signing up for new services

Worldcom

Worldcom is another company that has found great value in data mining. By mining databases of its customer-service and telemarketing data, Worldcom has discovered new ways to sell voice and data services. For example, it has found that people who buy two or more services were likely to be relatively loyal customers. It also found that people were willing to buy packages of products such as long-distance, cellular-phone, Internet, and other services. Consequently, Worldcom started to offer more such packages.

BBC TV

TV-program schedulers would like to know the likely audience for a proposed program and the best time to show it. The data for audience prediction are fairly complex. Factors, which determine the audience share gained by a particular program, include not only the characteristics of the program itself and the time at which it is shown but also the nature of the competing programs in other channels. Using Clementine, Integral Solutions Limited developed a system to predict television audiences for the BBC. The prediction accuracy was reported to be the same as that achieved by the best performance of BBC's planners.

Bell Atlantic

Bell Atlantic developed telephone technician dispatch system. When a customer reports a telephone problem to Bell Atlantic, the company must decide what type of technician to dispatch to resolve the issue. Starting in 1991, this decision was made using a handcrafted expert system, but in 1999 it was replaced by another set of rules

created with machine learning. The learned rules save Bell Atlantic more than 10 million dollars per year because they make fewer erroneous decisions. In addition, the original expert system had reached a stage in its evolution where it could not be maintained cost effectively. Because the learned system was built by training it on examples, it is easy to maintain and to adapt to regional differences and changing cost structures.

B.3   DATA MINING FOR THE RETAIL INDUSTRY

Slim margins have pushed retailers into data warehousing earlier than other industries. Retailers have seen improved decision-support processes, leading directly to improved efficiency in inventory management and financial forecasting. The early adoption of data warehousing by retailers has allowed them a better opportunity to take advantage of data mining. The retail industry is a major application area for data mining since it collects huge amounts of data on sales, customer-shopping history, goods transportation, consumption patterns, service records, and so on. The quantity of data collected continues to expand rapidly, especially due to the increasing availability and popularity of business conducted on the Web, or e-commerce. Walmart's push to use radio frequency identification (RFID) tags for supply chain optimization is a great story that illustrates the dawn of the big data era in retail industry. RFID is a great example of machine-generated data that could be collected, organized, and analyzed. Today, the world has become much more instrumented and interconnected thanks to many new technologies, including RFID tagging. Important examples of a combination of RFID technology producing big data together with data mining include tracking products at the skid level or the stock-keeping unit (SKU) level. A variety sources and types of retail data provide a rich source for data mining. Today, many stores also have Web sites where customers can make purchases online, but at the same time producing really big data for analysis of customers' satisfaction and other characteristics of retailer-customer relation.

Retail data mining can help identify customer-buying behaviors, discover customer-shopping patterns and trends, improve the quality of customer services, achieve better customer retention and satisfaction, enhance goods consumption, design more effective goods transportation and distribution policies, and, in general, reduce the cost of business and increase profitability. In the forefront of applications that have been adopted by the retail industry are direct-marketing applications. The direct-mailing industry is an area where data mining is widely used. Almost every type of retailer uses direct marketing, including catalogers, consumer retail chains, grocers, publishers, B2B marketers, and packaged goods manufacturers. The claim could be made that every Fortune 500 company has used some level of data mining in their direct-marketing campaigns. Large retail chains and groceries stores use vast amounts of sale data that is "information rich." Direct marketers are mainly concerned about customer segmentation, which is essentially a clustering or classification problem.

As the volume of customer communications through Internet grows exponentially and consumers' attention spans shrink by the day, delivering individually relevant content and experiences has become a marketing imperative for all organizations. Machine-learning personalization in marketing provides a more scalable way to achieve unique experiences for individuals, rather than segments of people or global population. It allows the company to utilize algorithms that will deliver one-to-one experiences, typically in the form of recommendations for products or content. With next-generation platforms, machine-learning personalization can also be applied to recommending categories, brands, and offers, as well as dynamically modifying site navigation, search results, and list sorting. Popularized by household names like Amazon and Netflix, algorithms are not just for giant e-commerce companies. They can be utilized by marketers from companies of any size.

Retailers are interested in creating data-mining models to answer questions such as the following:

- What are the best types of advertisements to reach certain segments of customers?
- What is the optimal timing at which to send mailers?
- What is the latest product trend?
- What types of products can be sold together?
- How does one retain profitable customers?
- What are the significant customer segments that buy products?

Data mining helps to model and identify the traits of profitable customers, and it also helps to reveal the "hidden relationship" in data that standard-query processes have not found. IBM has used data mining for several retailers to analyze shopping patterns within stores based on point-of-sale (POS) information. For example, one retail company with $2 billion in revenue, 300,000 UPC codes, and 129 stores in 15 states found some interesting results: "…we found that people who were coming into the shop gravitated to the left-hand side of the store for promotional items, and they were not necessarily shopping the whole store." Such information is used to change promotional activities and provide a better understanding of how to lay out a store in order to optimize sales. Additional real-world examples of data-mining systems in retail industry follow:

Safeway, UK

> Grocery chains have been another big user of data-mining technology. Safeway is one such grocery chain with more than $10 billion in sales. It uses Intelligent Miner from IBM to continually extract business knowledge from its product-transaction data. For example, the data-mining system found that the top-spending 25% customers very often purchased a particular cheese product ranked below 200 in sales. Normally, without the data-mining results, the product would have been discontinued. But the extracted rule showed that discontinuation would disappoint the best

customers, and Safeway continues to order this cheese, though it is ranked low in sales. Thanks to data mining, Safeway is also able to generate customized mailing to its customers by applying the sequence-discovery function of Intelligent Miner, allowing the company to maintain its competitive edge.

RS Components, UK

RS Components, a UK-based distributor of technical products such as electronic and electrical components and instrumentation, has used the IBM Intelligent Miner to develop a system to do cross-selling (suggested related products on the phone when customers ask for one set of products) and in warehouse product allocation. The company had one warehouse in Corby before 1995 and decided to open another in the Midlands to expand its business. The problem was how to split the products into these two warehouses so that the number of partial orders and split shipments could be minimized. Remarkably, the percentage of split orders is just about 6% after using the patterns found by the system, much better than expected.

Kroger Co. (USA)

The Kroger is the largest grocery store chain in the United States. 40% of all US households have one of Kroger's loyalty cards. The Kroger is trying to drive loyalty for life with their customers. In particular, their customers are rewarded with offers on what they buy instead of trying to be sold something else. In other words, each of them could receive coupons different from each other, not the same coupons. In order to match the best customers with the right coupons, the Kroger analyses customers' behavior using the data-mining techniques. For instance, one recent mailing was customized to 95% of the intended recipients. Such business strategy for looking at customers to win customers for life makes the Kroger beat their largest competitor, Walmart, for the last six years largely.

Korea Customs Service (South Korea)

The Korea Customs Service (KCS) is a government agency established to secure national revenues by controlling imports and exports for the economic development of South Korea and to protect domestic industry through contraband control. It is responsible for the customs clearance of imported goods as well as tax collection at the customs border. For detecting illegal cargo, they implemented a system using SAS for fraud detection, based on its widespread use and trustworthy reputation in the data-mining field. This system enabled more specific and accurate sorting of illegal cargo. For instance, the number of potentially illegal factors increased from 77 to 163. As a result, the detection rate for important items, as well as the total rate, increased by more than 20%[https://unctad.org/meetings/en/Presentation/dtl_eWeek2018p78_KeunhooLee_en.pdf].

Bookmark.com (USA)

> *Bookmark.com **is** an AI-powered Web-site building platform, which uses machine learning to build custom Web sites. Bookmark's AI technology, called the Artificial Intelligence Design Assistant (AiDA), learns each user's unique needs from a few nuggets of client information such as name, location, and type of business. Using the information provided, AiDA crawls competitor Web sites along with any information about a client's business or public found across Google, Facebook, and other social channels. AiDA then determines which components, colors, and layouts would be most optimal and relevant for each Web site. Machine learning helps AiDA improve with each new Web site it builds. In addition to using machine learning to create personalized and engaging Web sites, Bookmark is also looking to implement AI into their shopper service efforts. The idea is to use machine learning to provide their shoppers with quality, personalized support that speaks expressly to their individual experiences with Bookmark's platform.*

B.4   DATA MINING IN HEALTHCARE AND BIOMEDICAL RESEARCH

With the amount of information and issues in the healthcare industry, not to mention the pharmaceutical industry and biomedical research, opportunities for data-mining applications are extremely widespread, and benefits from the results are enormous. Storing patients' records in electronic format and the development in medical information systems cause a large amount of clinical data to be available online. Regularities, trends, and surprising events extracted from these data by data-mining methods are important in assisting clinicians to make informed decisions, thereby improving health services.

Clinicians evaluate a patient's condition over time. The analysis of large quantities of time-stamped data will provide doctors with important information regarding the progress of the disease. Therefore, systems capable of performing temporal abstraction and reasoning become crucial in this context. Although the use of temporal-reasoning methods requires an intensive knowledge-acquisition effort, data mining has been used in many successful medical applications, including data validation in intensive care, the monitoring of children's growth, analysis of diabetic patient's data, the monitoring of heart-transplant patients, and intelligent anesthesia monitoring.

Data mining has been used extensively in the medical industry. Data visualization and artificial neural networks are especially important areas of data mining applicable in the medical field. For example, NeuroMedical Systems used neural networks to perform a pap smear diagnostic aid. Vysis Company uses neural networks to perform protein analyses for drug development. The University of Rochester Cancer Center and the Oxford Transplant Center use KnowledgeSeeker, a decision-tree-based technology, to help with their research in oncology.

The past decade has seen an explosive growth in biomedical research, ranging from the development of new pharmaceuticals and advances in cancer therapies to

the identification and study of the human genome. The logic behind investigating the genetic causes of diseases is that once the molecular bases of diseases are known, precisely targeted medical interventions for diagnostics, prevention, and treatment of the disease themselves can be developed. Much of the work occurs in the context of the development of new pharmaceutical products that can be used to fight a host of diseases ranging from various cancers to degenerative disorders such as Alzheimer's disease.

A great deal of biomedical research has focused on DNA data analysis, and the results have led to the discovery of genetic causes for many diseases and disabilities. An important focus in genome research is the study of DNA sequences since such sequences form the foundation of the genetic codes of all living organisms. What is DNA? Deoxyribonucleic acid (DNA) forms the foundation for all living organisms. DNA contains the instructions that tell cells how to behave and is the primary mechanism that permits us to transfer our genes to our offspring. DNA is built in sequences that form the foundations of our genetic codes and that are critical for understanding how our genes behave. Each gene comprises a series of building blocks called nucleotides. When these nucleotides are combined, they form long, twisted, and paired DNA sequences or chains. Unraveling these sequences has become a challenge since the 1950s when the structure of the DNA was first understood. If we understand DNA sequences, theoretically, we will be able to identify and predict faults, weaknesses, or other factors in our genes that can affect our lives. Getting a better grasp of DNA sequences could potentially lead to improved procedures to treat cancer, birth defects, and other pathological processes. Data-mining technologies are only one weapon in the arsenal used to understand these types of data, and the use of visualization and classification techniques is playing a crucial role in these activities.

It is estimated that humans have around 100,000 genes, each one having DNA that encodes a unique protein specialized for a function or a set of functions. Genes controlling production of hemoglobin, regulation of insulin, and susceptibility to Huntington's chorea are among those that have been isolated in recent years. There are seemingly endless varieties of ways in which nucleotides can be ordered and sequenced to form distinct genes. Any one gene might comprise a sequence containing hundreds of thousands of individual nucleotides arranged in a particular order. Furthermore, the process of DNA sequencing used to extract genetic information from cells and tissues usually produces only fragments of genes. It has been difficult to tell using traditional methods where these fragments fit into the overall complete sequence from which they are drawn. Genetic scientists face the difficult task of trying to interpret these sequences and form hypotheses about which genes they might belong to and the disease processes that they may control. The task of identifying good candidate gene sequences for further research and development is like finding a needle in a haystack. There can be hundreds of candidates for any given disease being studied. Therefore, companies must decide which sequences are the most promising ones to pursue for further development. How do they determine which ones would make good therapeutic targets? Historically, this has been a process based largely on trial and error. For every lead that eventually turns into a successful pharmaceutical intervention that

is effective in clinical settings, there are dozens of others that do not produce the anticipated results. This is a research area that is crying out for innovations that can help to make these analytical processes more efficient. Since pattern analysis, data visualization, and similarity search techniques have been developed in data mining, this field has become a powerful infrastructure for further research and discovery in DNA sequences. We will describe one attempt to innovate the process of mapping human genomes that has been undertaken by Incyte Pharmaceuticals, Inc. in cooperation with Silicon Graphics:

Incyte Pharmaceuticals, Inc.

Incyte Pharmaceuticals is a publicly held company founded in 1991, and it is involved in high-throughput DNA sequencing and development of software, databases, and other products to support the analysis of genetic information. The first component of their activities is a large database called LiveSeq that contains more than 3 million human-gene sequences and expression records. Clients of the company buy a subscription to the database and receive monthly updates that include all of the new sequences identified since the last update. All of these sequences can be considered as candidate genes that might be important for future genome mapping. This information has been derived from DNA sequencing and bioanalysis of gene fragments extracted from cell and tissue samples. The tissue libraries contain different types of tissues including normal and diseased tissues, which are very important for comparison and analyses.

To help impose a conceptual structure of the massive amount of information contained in LifeSeq, the data have been coded and linked to several levels. Therefore, DNA sequences can be grouped into many different categories, depending on the level of generalization. LifeSeq has been organized to permit comparisons of classes of sequence information within a hypothesis-testing mode. For example, a researcher could compare gene sequences isolated from diseased and nondiseased tissue from an organ. One of the most important tools that is provided in LifeSeq is a measure of similarity among sequences that are derived from specific sources. If there is a difference between two tissue groups for any available sequences, this might indicate that these sequences should be explored more fully. Sequences occurring more frequently in the diseased sample might reflect genetic factors in the disease process. On the other hand, sequences occurring more frequently in the nondiseased sample might indicate mechanisms that protect the body from the disease.

Although it has proved invaluable to the company and their clients in its current incarnation, additional features are being planned and implemented to extend the LifeSeq functionality into research areas such as:

- *Identifying co-occurring gene sequences.*
- *Tying genes to disease stage.*
- *Using LifeSeq to predict molecular toxicology.*

Although the LifeSeq database is an invaluable research resource, queries to the database often produce very large data sets that are difficult to analyze in text format. For this reason, Incyte developed the LifeSeq 3D application that provides visualization of data sets and also allows users to cluster or classify and display information about genes. The 3D version has been developed using the Silicon Graphics MineSet tool. This version has customized functions that let researchers explore data from LifeSeq and discover novel genes within the context of targeted protein functions and tissue types.

Maine Medical Center (USA)

Maine Medical Center—a teaching hospital and the major community hospital for the Portland, Maine, area—has been named to the U.S. News and World Report Best Hospitals list twice in orthopedics and heart care. In order to improve quality of patient care in measurable ways, Maine Medical Center has used scorecards as key performance indicators. Using SAS, the hospital creates balanced scorecards that measure everything from staff handwashing compliance to whether a congestive heart patient is actually offered a flu vaccination. 100 percent of heart failure patients are getting quality care as benchmarked by national organizations, and a medication error reduction process has improved 35%.

> *https://www.sunjournal.com/2010/01/07/medical-group-launches-prevention-campaign/.*

> *On November 2009, the Central Maine Medical Group (CMMG) announced the launch of a prevention and screening campaign called "Saving Lives Through Evidence-Based Medicine." The new initiative is employed to redesign the ways that it works as a team of providers to make certain that each of our patients undergoes the necessary screening tests identified by the current medical literature using data-mining techniques. In particular, data-mining process identifies someone at risk for an undetected health problem [http://www.cmmc.org/news.taf].*

B.5   DATA MINING IN SCIENCE AND ENGINEERING

Enormous amounts of data have been generated in science and engineering, e.g. in cosmology, molecular biology, and chemical engineering. In cosmology, advanced computational tools are needed to help astronomers understand the origin of large-scale cosmological structures as well as the formation and evolution of their astrophysical components (galaxies, quasars, and clusters). Over three terabytes of image data have been collected by the Digital Palomar Observatory Sky Survey, which contain on the order of two billion sky objects. It has been a challenging task for astronomers to catalog the entire data set, i.e. a record of the sky location of each object and its corresponding classification such as a star or a galaxy. The Sky Image Cataloging and Analysis Tool (SKICAT) has been developed to automate this task. The SKICAT system integrates methods from machine learning, image processing, classification, and databases, and it is reported to be able to classify objects, replacing visual classification, with high accuracy.

In molecular biology, recent technological advances are applied in such areas as molecular genetics, protein sequencing, and macromolecular structure determination as was mentioned earlier. Artificial neural networks and some advanced statistical methods have shown particular promise in these applications. In chemical engineering, advanced models have been used to describe the interaction among various chemical processes, and also new tools have been developed to obtain a visualization of these structures and processes. Let us have a brief look at a few important cases of data-mining applications in engineering problems. Pavilion Technologies' Process Insights, an application-development tool that combines neural networks, fuzzy logic, and statistical methods, has been successfully used by Eastman Kodak and other companies to develop chemical manufacturing and control applications to reduce waste, improve product quality, and increase plant throughput. Historical process data is used to build a predictive model of plant behavior, and this model is then used to change the control set points in the plant for optimization.

DataEngine is another data-mining tool that has been used in a wide range of engineering applications, especially in the process industry. The basic components of the tool are neural networks, fuzzy logic, and advanced graphical user interfaces. The tool has been applied to process analysis in the chemical, steel, and rubber industries, resulting in a saving in input materials and improvements in quality and productivity. Successful data-mining applications in some industrial complexes and engineering environments follow:

Boeing

To improve its manufacturing process, Boeing has successfully applied machine-learning algorithms to the discovery of informative and useful rules from its plant data. In particular, it has been found that it is more beneficial to seek concise predictive rules that cover small subsets of the data, rather than generate general decision trees. A variety of rules were extracted to predict such events as when a manufactured part is likely to fail inspection or when a delay will occur at a particular machine. These rules have been found to facilitate the identification of relatively rare but potentially important anomalies.

R.R. Donnelly

This is an interesting application of data-mining technology in printing press control. During rotogravure printing, grooves sometimes develop on the printing cylinder, ruining the final product. This phenomenon is known as banding. The printing company R.R. Donnelly hired a consultant for advice on how to reduce its banding problems and at the same time used machine learning to create rules for determining the process parameters (e.g. the viscosity of the ink) to reduce banding. The learned rules were superior to the consultant's advice in that they were more specific to the plant where the training data was collected and they filled gaps in the consultant's advice and thus were more complete. In fact, one learned rule contradicted the consultant's advice and proved to be correct. The learned rules have been in everyday use in the

Donnelly plant in Gallatin, Tennessee, for over a decade and have reduced the number of banding occurrences from 538 to 26.

Southern California Gas Company

The Southern California Gas Company is using SAS software as a strategic marketing tool. The company maintains a data mart called the Customer Marketing Information Database that contains internal billing and order data along with external demographic data. According to the company, it has saved hundreds of thousands of dollars by identifying and discarding ineffective marketing practices.

WebWatcher

Despite the best effort of Web designers, we all have had the experience of not being able to find a certain Web page we want. A bad design for a commercial Web site obviously means the loss of customers. One challenge for the data-mining community has been the creation of "adaptive Web sites," Web sites that automatically improve their organization and presentation by learning from user-access patterns. One early attempt is WebWatcher, an operational tour guide for the WWW. It learns to predict what links users will follow on a particular page, highlight the links along the way, and learn from experience to improve its advice-giving skills. The prediction is based on many previous access patterns and the current user's stated interests. It has also been reported that Microsoft is to include in its electronic-commerce system a feature called Intelligent Cross-Sell that can be used to analyze the activity of shoppers on a Web site and automatically adapt the site to that user's preferences.

AbitibiBowater Inc. (Canada)

AbitibiBowater Inc. is a pulp and paper manufacturer headquartered in Montreal, Quebec, Canada. The pulp and paper, a key component of the forest products industry, is a major contributor to Canada's economy. In addition to market pulp, the sector produces newsprint, specialty papers, paperboard, building board, and other paper products. It is the largest industrial energy consumer, representing 23% of industrial energy consumption in Canada. AbitibiBowater Inc. used data-mining techniques to detect a period of high performance and reduce energy consumption in the papermaking process, so that they recognized that lower temporary consumption is caused by the reduced set point for chip preheating and cleaning of the heating tower on the reject refiners. AbitibiBowater Inc. was able to reproduce the process conditions required to maintain steam recovery. This has saved AbitibiBowater 200 gigajoules[1] daily—the equivalent of $600,000 a year [Head Up CIPEC(Canadian

---

[1] A gigajoule (GJ) is a metric term used for measuring energy use. For example, 1 GJ is equivalent to the amount of energy available from either 277.8 kWh of electricity, or 26.1 m^3 of natural gas, or 25.8 l of heating oil.

Industry Program for Energy Conservation) new letter: August 15, 2009 Vol. XIII, No.15].

eHarmony

The eHarmony dating service, which rather than matching prospective partners on the basis of their stated preferences, uses statistical analysis to match prospective partners, based on a 29 parameter model derived from 5000 successful marriages. Its competitors such as Perfectmatch use different models, such as the Jungian Myers–Briggs personality typing technique to parameterize individuals entered into their database. It is worth observing that while the process of matching partners may amount to little more than data retrieval using some complex set of rules, the process of determining what these rules need to be involves often complex knowledge discovery and mining techniques.

The Maintenance of Military Platforms

Another area where data-mining techniques offer promising gains in efficiency is in the maintenance of military platforms. Good and analytically based maintenance programs, with the Amberley Ageing Aircraft Program for the F-111 as a good example, systematically analyze component failure statistics to identify components with wear out or other failure rate problems. They can then be removed from the fleet by replacement with new or reengineered and thus more reliable components. This type of analysis is a simple rule-based approach, where the rule is simply the frequency of faults in specific components.

B.6   PITFALLS OF DATA MINING

Despite the above and many other success stories often presented by vendors and consultants to show the benefits that data mining provides, this technology has several pitfalls. When used improperly, data mining can generate lots of "garbage." As one professor from MIT pointed out: "Given enough time, enough attempts, and enough imagination, almost any set of data can be teased out of any conclusion." David J. Lainweber, managing director of First Quadrant Corp. in Pasadena, California, gives an example of the pitfalls of data mining. Working with a United Nations data set, he found that historically, butter production in Bangladesh is the single best predictor of the Standard & Poor's 500 stock index. This example is similar to another absurd correlation that is heard yearly around Super Bowl time—a win by the NFC team implies a rise in stock prices. Peter Coy, *Businessweek*'s associate economics editor, warns of four pitfalls in data mining:

1. It is tempting to develop a theory to fit an oddity found in the data.
2. One can find evidence to support any preconception if you let the computer churn long enough.

3. A finding makes more sense if there is a plausible theory for it. But a beguiling story can disguise weaknesses in the data.
4. The more factors or features in a data set the computer considers, the more likely the program will find a relationship, valid or not.

It is crucial to realize that data mining can involve a great deal of planning and preparation. Just having a large amount of data alone is no guarantee of the success of a data-mining project. In the words of one senior product manager from Oracle, "Be prepared to generate a lot of garbage until you hit something that is actionable and meaningful for your business."

> This appendix is certainly not an inclusive list of all data-mining activities, but it does provide examples of how data-mining technology is employed today. We expect that new generations of data-mining tools and methodologies will increase and extend the spectrum of application domains.

BIBLIOGRAPHY

CHAPTER 1

Acharjya D. P., et al., A Survey on Big Data Analytics: Challenges Open Research Issues and Tools, *(IJACSA) International Journal of Advanced Computer Science and Applications*, Vol. 7, No. 2, 2016, pp. 511–518.

Adriaans P., D. Zantinge, *Data Mining*, Addison-Wesley Publ. Co., New York, 1996.

Agosta L., *The Essential Guide to Data Warehousing*, Prentice Hall, Inc., Upper Saddle River, 2000.

An A., C. Chun, N. Shan, N. Cercone, W. Ziarko, Applying Knowledge Discovery to Predict Watter-Supply Consumption, *IEEE Expert*, July/August 1997, pp. 72–78.

Barquin R., H. Edelstein, *Building, Using, and Managing the Data Warehouse*, Prentice Hall, Inc., Upper Saddle River, 1997.

Bello-Orgaz G., et al., Social Big Data: Recent Achievements and New Challenges, *Information Fusion*, Vol. 28, March 2016, pp. 45–59.

Ben Hitt, Eric King, How to Prepare for Data Mining, http://www.b-eye-network.com/channels/1415/view/10880, July 2009.

Berson A., S. Smith, K. Thearling, *Building Data Mining Applications for CRM*, McGraw-Hill, 2000.

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

Brachman R. J., T. Khabaza, W. Kloesgen, G. S. Shapiro, E. Simoudis, Mining Business Databases, *CACM*, Vol. 39, No. 11, November 1996, pp. 42–48.

Braganza A., et al., Resource Management in Big Data Initiatives: Processes and Dynamic Capabilities, *Journal of Business Research*, Vol. 70, January 2017, pp. 328–337

De Ville B., Managing the Data Mining Project, *Microsoft Data Mining*, 2001, pp. 93–116.

Djoko S., D. J. Cook, L. B. Holder, An Empirical study of Domain Knowledge and its Benefits to Substructure Discovery, *IEEE Transactions on Knowledge and Data Engineering*, Vol. 9, No. 4, July/August 1997, pp. 575–585.

Emani C. K., Cullot N., Nicolle C., Understandable Big Data: A Survey, *Computer Science Review*, Vol. 17, August 2015, pp. 70–81.

Fayyad U., G. P. Shapiro, P. Smyth, The KDD Process for Extracting Useful Knowledge from Volumes of Data, *CACM*, Vol. 39, No. 11, November 1966a, pp. 27–34.

Fayyad U. M., G. Piatetsky-Shapiro, P. Smith, R. Uthurusamy, eds., *Advances in Knowledge Discovery and Data Mining*, AAAI Press/MIT Press, Cambridge, 1996b.

Fayyad U., G. P. Shapiro, P. Smyth, From Data Mining to Knowledge Discovery in Databases, *AI Magazine*, Fall 1996c, pp. 37–53.

Foreman J., *Data Smart: Using Data Science to Transform Information into Insight*, John Wiley, 2014.

Friedland L., Accessing the Data Warehouse: Designing Tools to Facilitate Business Understanding, *Interactions*, January–February 1998, pp. 25–36.

Gandomi A., et al., Beyond the Hype: Big Data Concepts, Methods, and Analytics, *International Journal of Information Management* Vol. 35, 2015, pp. 137–144

Ganti V., J. Gehrke, R. Ramakrishnan, Mining Very Large Databases, *Computer*, Vol. 32, No. 8, August 1999, pp. 38–45.

Groth R., *Data Mining: A Hands-On Approach for Business Professionals*, Prentice hall, Inc., Upper Saddle River, 1998.

Han J., M. Kamber, *Data Mining: Concepts and Techniques*, 2nd edition, Morgan Kaufmann, 2006.

Kantardzic M., Editorial on Big Data, Transactions on Machine Learning and Data Mining, *IBAI Publishing*, Vol. 9, No.1, 2016, pp. 1–2.

Kaudel A., Last M., Bunke H., eds., *Data Mining and Computational Intelligence*, Physica-Verlag, Heidelberg, Germany, 2001.

Khan N., et al., Big Data: Survey, Technologies, Opportunities, and Challenges, *Scientific World Journal* 2014, Hindawi Publ. Co., Article ID 712826, 18 pages.

Kriegel H.P., et al., Future Trends in Data Mining, *Data Mining and Knowledge Discovery*, Vol. 15, 2007, pp. 87–97.

Lavrac N., et al., Introduction: Lessons Learned from Data Mining Applications and Collaborative Problem Solving, *Machine Learning*, Vol. 57, 2004, pp. 13–34.

Jure Leskovac, Anand Rajaraman, Jeffrey Ullman, *Mining of Massive Datasets*, 2nd edition, Cambridge University Press, 2014.

Maxus Systems International, *What is Data Mining, Internal Documentation*, http://www.maxussystems.com/datamining.html.

David L. Olson Data Mining in Business Services, *Service Business*, Springer Berlin/Heidelberg, Vol. 1, No. 3, September 2007, pp. 181–193.

Paul Zikopoulos, Chris Eaton, *Understanding Big Data: Analytics for Enterprise Class Hadoop and Streaming Data*, McGraw Hill Professional, 2011.

Pouyanfar S., et al., Multimedia Big Data Analytics: A Survey, *ACM Computing Surveys*, Vol. 51, No. 1, April 2018.

Pyle D., *Getting the Initial Model: Basic Practices of Data Mining, Business Modeling and Data Mining*, 2003, pp. 361–425

Qui J., et al., A Survey of Machine Learning for Big Data Processing, *EURASIP Journal on Advances in Signal Processing* Vol. 2016, 2016, p. 67.

Ramakrishnan N., Grama A. Y., Data Mining: From Serendipity to Science, *Computer*, Vol. 32, No. 8, August 1999, pp. 34–37.

Shapiro G. P., The Data-Mining Industry Coming of Age, *IEEE Intelligent Systems*, November/December 1999, pp. 32–33.

Sivarajah U., et al., Critical Analysis of Big Data Challenges and Analytical Methods, *Journal of Business Research*, Vol. 70, 2017, pp. 263–286.

Skourletopoulos G., et al., Big Data and Cloud Computing: A Survey of the State-of-the-Art and Research Challenges, In C.X. Mavromoustakis et al. (eds.), *Advances in Mobile Cloud Computing and Big Data in the 5G Era*, Springer, 2017.

Thomsen E., *OLAP Solution: Building Multidimensional Information System*, John Wiley, New York, 1997.

Thuraisingham B., *Data Mining: Technologies, Techniques, Tools, and Trends*, CRC Press LLC, Boca Raton, FL, 1999.

Tsur S., Data Mining in the Bioinformatics Domain, *Proceedings of the 26th YLDB Conference*, Cairo, Egypt, 2000, pp. 711–714.

Two Crows Corp., *Introduction to Data Mining and Knowledge Discovery*, Two Crows Corporation, 2005.

Waltz D., Hong S. J., Data Mining: A Long Term Dream, *IEEE Intelligent Systems*, November/December, 1999, pp. 30–34.

Zaki M. J., Meira W., *Data Mining and Analysis: Fundamental Concepts and Algorithms*, Cambridge University Press, 2014.

CHAPTER 2

Adriaans P., D. Zantinge, *Data Mining*, Addison-Wesley Publ. Co., New York, 1996.

Agraval C., *Outliers Analysis*, 2nd edition, Springer, 2016.

Anand S. S., D. A. Bell, J. G. Hughes, The Role of Domain Knowledge in Data Mining, *Proceedings of the CIKM'95 Conference*, Baltimore, 1995, pp. 37–43.

Barquin R., H. Edelstein, *Building, Using, and Managing the Data Warehouse*, Prentice Hall, Inc., Upper Saddle River, 1997.

Ben Hitt, Eric King, *How to Prepare for Data Mining*, http://www.b-eye-network.com/channels/1415/view/10880, July 2009.

Berson A., S. Smith, K. Thearling, *Building Data Mining Applications for CRM*, McGraw-Hill, 2000.

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

Boriah S., Chandola V., Kumar V., Similarity Measures for Categorical Data: A Comparative Evaluation, *SIAM Conference*, 2008, pp. 243–254.

Brachman R. J., T. Khabaza, W. Kloesgen, G. S. Shapiro, E. Simoudis, Mining Business Databases, *CACM*, Vol. 39, No. 11, November 1996, pp. 42–48.

Chen C. H., L. F. Pau, P. S. P. Wang, *Handbook of Pattern Recognition & Computer Vision*, World Scientific Publ. Co., Singapore, 1993.

Clark W. A. V., Deurloo M. C., *Categorical Modeling/Automatic Interaction Detection, Encyclopedia of Social Measurement*, 2005, pp. 251–258.

Dwinnell W., Data Cleansing: An Automated Approach, *PC AI*, March/April 2001, pp. 21–23.

Fayyad U., D. Haussier, P. Stolorz, Mining Scientific Data, *CACM*, Vol. 39, No. 11, November 1966a, pp. 51–57.

Fayyad U. M., G. Piatetsky-Shapiro, P. Smith, R. Uthurusamy, eds., *Advances in Knowledge Discovery and Data Mining*, AAAI Press/MIT Press, Cambridge, 1996b.

Ganti V., J. Gehrke, R. Ramakrishnan, Mining Very Large Databases, *Computer*, Vol. 32, No. 8, August 1999, pp. 38–45.

García S., J. Luengo, J.A. Sáez, V. López, F. Herrera, A Survey of Discretization Techniques: Taxonomy and Empirical Analysis in Supervised Learning, *IEEE Transactions on Knowledge and Data Engineering*, Vol. 25, No. 4, 2013, pp. 734–750.

Groth R., *Data Mining: A Hands-On Approach for Business Professionals*, Prentice hall, Inc., Upper Saddle River, 1998.

Han J., M. Kamber, *Data Mining: Concepts and Techniques*, 2nd edition, Morgan Kaufmann,2006.

Hariharakrishnan J., et al., Survey of Pre-processing Techniques for Mining Big Data, *International Conference on Computer, Communication and Signal Processing (ICCCSP)*, 2017.

Liu H., H. Motoda, eds., *Feature Extraction, Construction and Selection: A Data Mining Perspective*, Kluwer Academic Publishers, Boston, MA, 1998.

Liu H., H. Motoda, *Feature Selection for Knowledge Discovery and Data Mining*, Second Printing, Kluwer Academic Publishers, Boston, 2000.

Pass S., Discovering Value in a Mountain of Data, *OR/MS Today*, October 1997, pp. 24–28.

Pyle D., *Data Preparation for Data Mining*, Morgan Kaufmann Publ. Inc., New York, 1999.

Ramirez-Gallego S., et al., A Survey on Data Preprocessing for Data Stream Mining: Current Status and Future Directions, *Neurocomputing*, Vol. 239, May 2017 pp. 39–57.

Refaat M., Treatment of Missing Values, *Data Preparation for Data Mining Using SAS*, 2007, pp. 171–206.

Tan P.-N., Steinbach M., Kumar V., *Introduction to Data Mining*, Pearson Addison-Wesley, 2006.

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

Westphal C., T. Blaxton, *Data Mining Solutions: Methods and Tools for Solving Real-World Problems*, John Wiley & Sons, Inc., New York, 1998.

Witten I. H., Frank E., *Data Mining: Practical Machine Learning Tools and Techniques*, 2nd edition, Elsevier Inc., 2005.

CHAPTER 3

Adriaans P., D. Zantinge, *Data Mining*, Addison-Wesley Publ. Co., New York, 1996.

Berson A., S. Smith, K. Thearling, *Building Data Mining Applications for CRM*, McGraw-Hill, 2000.

Bolón-Canedo V., N. Sánchez-Marono, A. Alonso-Betanzos Recent Advances and Emerging Challenges of Feature Selection in the Context of Big Data. *Knowledge-Based Systems*, Vol. 86, 2015, pp. 33–45.

Brachman R. J., T. Khabaza, W. Kloesgen, G. S. Shapiro, E. Simoudis, Mining Business Databases, *CACM*, Vol. 39, No. 11, November 1996, pp. 42–48.

Chen C. H., L. F. Pau, P. S. P. Wang, *Handbook of Pattern Recognition & Computer Vision*, World Scientific Publ. Co., Singapore, 1993.

Clark W. A.V., M. C. Deurloo, *Categorical Modeling/Automatic Interaction Detection, Encyclopedia of Social Measurement*, 2005, pp. 251–258.

Dwinnell W., Data Cleansing: An Automated Approach, *PC AI*, March/April 2001, pp. 21–23.

Eddy W. F., Large Data Sets in Statistical Computing, in *International Encyclopedia of the Social & Behavioral Sciences*, 2004, pp. 8382–8386.

Fayyad U. M., G. Piatetsky-Shapiro, P. Smith, R. Uthurusamy, eds., *Advances in Knowledge Discovery and Data Mining*, AAAI Press/MIT Press, Cambridge, 1996.

Gibaja E., S. Ventura, A Tutorial on Multilabel Learning. *ACM Computer Survey*, Vol. 47, No. 3, 2015, p. 52.

Groth R., *Data Mining: A Hands-On Approach for Business Professionals*, Prentice hall, Inc., Upper Saddle River, 1998.

Han J., M. Kamber, *Data Mining: Concepts and Techniques*, 2nd edition, Morgan Kaufmann, 2006.

Hashem I. A. T., I. Yaqoob, N. B. Anuar, S. Mokhtar, A. Gani, S. U. Khan, The Rise of "Big Data" on Cloud Computing: Review and Open Research Issues. *Information Systems*, Vol. 47, 2015, pp. 98–115.

Hodge V.J., S. O'Keefe, J. Austin, Hadoop Neural Network for Parallel and Distributed Feature Selection. *Neural Networks*, 2016, doi: http://dx.doi.org/10.1016/j.neunet.2015.08.011

Jain A., R. P. W. Duin, J. Mao, Statistical Pattern Recognition, *IEEE Transaction on Pattern Analysis and Machine Intelligence*, Vol. 22, No. 1, January 2000, pp. 4–37.

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

Kil D. H., F. B. Shin, *Pattern Recognition and Prediction with Applications to Signal Characterization*, AIP Press, Woodburg, NY, 1996.

Liu H., H. Motoda, eds., *Feature Extraction, Construction and Selection: A Data Mining Perspective*, Kluwer Academic Publishers, Boston, MA, 1998.

Liu H., H. Motoda, *Feature Selection for Knowledge Discovery and Data Mining*, Second Printing, Kluwer Academic Publishers, Boston, 2000.

Liu H., H. Motoda, eds., *Instance Selection and Construction for Data Mining*, Kluwer Academic Publishers, Boston, MA, 2001.

Maimon O., M. Last, *Knowledge Discovery and Data Mining: The Info-Fuzzy Network (IFN) Methodology*, Kluwer Academic Publishers, Boston, MA, 2001.

Pyle D., *Data Preparation for Data Mining*, Morgan Kaufmann Publ. Inc., New York, 1999.

Tan P-N, M. Steinbach, V. Kumar, *Introduction to Data Mining*, Pearson Addison-Wesley, 2006.

Wang S., W. Pedrycz, Q. Zhu, W. Zhu, Unsupervised Feature Selection via Maximum Projection and Minimum Redundancy, *Knowledge-Based Systems*, Vol. 75, 2015, pp. 19–29.

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

Westphal C., T. Blaxton, *Data Mining Solutions: Methods and Tools for Solving Real-World Problems*, John Wiley & Sons, Inc., New York, 1998.

Witten I. H., E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques*, 2nd edition, Elsevier Inc., 2005.

Yang Q., Wu X., 10 Challenging Problems in Data Mining Research, *International Journal of Information Technology & Decision Making*, Vol. 5, No. 4, 2006, pp. 597–604.

CHAPTER 4

Alpaydin Ethem, *Introduction to Machine Learning*, 2nd edition, The MIT Press, 2010.

Arabmakki E., Kantardzic M., SOM-Based Partial Labeling of Imbalanced Data Stream, *Neurocomputing*, Vol. 262, No. 1, November 2017, pp. 120–133.

Berbaum K. S., D. D. Dorfman, E. A. Franken Jr., Measuring Observer Performance by ROC Analysis: Indications and Complications, *Investigative Radiology*, Vol. 2A, March 1989, pp. 228–233.

Berthold M., D. J. Hand, eds., *Intelligent Data Analysis – An Introduction*, Springer, Berlin, 1999.

Bow S., *Pattern Recognition and Image Preprocessing*, Marcel Dekker Inc., New York, 1992.

Chapella O., et al., *Semi-Supervised Learning*, The MIT Press, Cambridge, 2006.

Chavla N. V., et al., SMOTE: Synthetic Minority Over-sampling Technique, *Journal of Artificial Intelligence Research*, Vol. 16, 2002, pp. 321–357.

Cherkassky V., F. Mulier, *Learning from Data: Concepts, Theory and Methods*, John Wiley & Sons, Inc., New York, 1998.

Diettrich T. G., Machine-Learning Research: Four Current Directions, *AI Magazine*, Winter 1997, pp. 97–136.

Ding S., X. Zhang, An Overview on Semi-supervised Support Vector Machines, *Neural Computing & Applications*, Vol. 26, No. 8, Springer, November 2015.

Ding S., et al., An Overview on Semi-supervised Support Vector Machine, *Neural Computing and Applications*, Vol. 26, No. 8, November 2015.

Engel A., C. Van den Broeck, *Statistical Mechanics of Learning*, Cambridge University Press, Cambridge, England, 2001.

Ghahramani, Z., Probabilistic Machine Learning and Artificial Intelligence, *Nature*, Vol. 521, 2015, 452–459.

Gu Q., et al., Data Mining on Imbalanced Data Sets, *2008 International Conference on Advanced Computer Theory and Engineering*, Thailand, 2008.

Gunopulos D., R. Khardon, H. Mannila, H. Toivonen, Data Mining, Hypergraph Traversals, and Machine Learning, *Proceedings of PODS'97 Conference*, Tucson, 1997, pp. 209–216.

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

He Haibo, Yunqian Ma, *Imbalanced Learning: Foundations, Algorithms, and Applications*, 1st edition, John Wiley & Sons, Inc., 2013.

Hearst M., Support Vector Machines, *IEEE Intelligent Systems*, July/August 1998, pp. 18–28.

Hilderman R. J., H. J. Hamilton, *Knowledge Discovery and Measures of Interest*, Kluwer Academic Publishers, Boston, MA, 2001.

Hirji K. K., Exploring Data Mining Implementation, *CACM*, Vol. 44, No. 7, July 2001, pp. 87–93.

Hsu C., C. Chang, C. Lin, A Practical Guide to Support Vector Classification, http://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf, 2009.

Jackson Joyce, Data Mining: A Conceptual Overview, *Communications of the Association for Information Systems*, Vol. 8, 2002, pp. 267–296.

Jiang Z., S. Zhang, J. Zeng, A Hybrid Generative/Discriminative Method for Semi-supervised Classification. *Knowledge-Based Systems*, Vol. 37, pp. 137–145.

Jordan M. I., T. M. Mitchell, Machine Learning: Trends, Perspectives, and Prospects, *Science*, Vol. 349, No. 6245, July 2015.

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

Kitts B., G. Melli, K. Rexer, eds., Data Mining Case Studies, *Proceedings of the First International Workshop on Data Mining Case Studies*, 2005.

Kukar M., Quality Assessment of Individual Classifications in Machine Learning and Data Mining, *Knowledge Information Systems*, Vol. 9, No. 3, 2006, pp. 364–384.

Lavrac N., et al., Introduction: Lessons Learned from Data Mining Applications and Collaborative Problem Solving, *Machine Learning*, Vol. 57, 2004, pp. 13–34.

Leondes C. T., *Knowledge-Based Systems: Techniques and Applications*, Academic Press, San Diego, 2000.

Luger G. F., Stubblefield W. A., *Artificial Intelligence: Structures and Strategies for Complex Problem Solving*, Addison Wesley Longman, Inc., Harlow, England, 1998.

Metz C. E., B. A. Herman, C. A. Roe, Statistical Comparison of Two ROC-curve Estimates Obtained from Partially-paired Datasets, *Medical Decision Making*, Vol. 18, No. 1, January-March 1998, pp. 110–124.

Mitchell T. M., Does Machine Learning Really Work?, *AI Magazine*, Fall 1997, pp. 11–20.

Mitchell T., *Machine Learning*, McGraw Hill, New York, NY, 1997.

Nisbet R., Elder J., Miner G., Classification, in *Handbook of Statistical Analysis and Data Mining Applications*, 2009a, pp. 235–258.

Nisbet R., Elder J., Miner G., Model Evaluation and Enhancement, in *Handbook of Statistical Analysis and Data Mining Applications*, 2009b, pp. 285–312.

Ortega, P., Figueroa, C., Ruz, G., A Medical Claim Fraud/Abuse Detection System based on Data Mining: A Case Study in Chile, *DMIN Conference*, 2006.

Platt J., Probabilistic Outputs for Support Vector Machines and Comparisons to Regularized Likelihood Methods, in "*Advances in Large Margin Classifiers*", A. Smola, P. Bartlett, B. Scholkopf, D. Schuurmans, eds., MIT Press, 1999.

Poole D., A. Mackworth, R. Goebel, *Computational Intelligence: A Logical Approach*, Oxford University Press, Inc., New York, 1998.

Pyle D., Getting the Initial Model: Basic Practices of Data Mining, *Business Modeling and Data Mining*, 2003, pp. 361–425.

Rao R., Improved Cardiac Care via Automated Mining of Medical Patient Records, *Proceedings of the First International Workshop on Data Mining Case Studies*, 2005.

Shahriari B., K. Swersky, A. Wang R. P. Adams, N. de Freitas, Taking the Human Out of the Loop: A Review of Bayesian Optimization. *Proceedings of the IEEE*, Vol. 104, No. 1, January 2016, pp. 148–175.

Thrun S., C. Faloutsos, Automated Learning and Discovery, *AI Magazine*, Fall 1999, pp. 78–82.

Wu X., et al., Top 10 Algorithms in Data Mining, *Knowledge Information Systems*, Vol. 14, 2008, pp. 1–37.

Xie Y., An Introduction to Support Vector Machine and Implementation in R, http://yihui. name/cv/images/SVM_Report_Yihui.pdf, May, 2007.

Zhong-Hui W., W. Li, Y. Cai, X Xu, An Empirical Comparison Of Ensemble Classification Algorithms With Support Vector Machines, *Proceedings of the Third International Conference on Machine Laming and Cybernetics*, Shanghai, August 2004.

Zweig M., G. Campbell, Receiver_Operating Characteristic (ROC) Plots: A Fundamental Evaluation Tool in Clinical Medicine, *Clinical Chemistry*, Vol. 39, No. 4, 1993, pp. 561–576.

CHAPTER 5

Alexander von Eye, Eun-Young Mun, *Log-Linear Modeling: Concepts, Interpretation, and Application*, John Wiley & Sons, Inc., 2013.

Bow S., *Pattern Recognition and Image Preprocessing*, Marcel Dekker, Inc, New York, 1992.

Brandt S., *Data Analysis: Statistical and Computational Methods for Scientists and Engineers*, 3rd edition, Springer, New York, 1999.

Cherkassky V., F. Mulier, *Learning from Data: Concepts, Theory and Methods*, John Wiley & Sons, Inc., New York, 1998.

Christensen R., *Log-Linear Models*, Springer-Verlag, New York, 1990.

Deng A., et al., Statistical Inference in Two-Stage Online Controlled Experiments with Treatment Selection and Validation, *International World Wide Web Conference Committee (IW3C2), WWW'14*, Seoul, Korea, April 2014.

Eddy W. F., Large Data Sets in Statistical Computing, *International Encyclopedia of the Social & Behavioral Sciences*, 2004, pp. 8382–8386.

Ezawa K. J., S. W. Norton, Constructing Bayesian Network to Predict Uncollectible Telecommunications Accounts, *IEEE Expert: Intelligent Systems & Their Applications*, Vol. 11, No. 5, October 1996, pp. 45–51.

Golden B., E. Condon, S. Lee, E. Wasil, Pre-Processing for Visualization using Principal Component Analysis, *Proceedings of the ANNEC'2000 Conference*, St. Louis, 2000, pp. 429–436.

Gose E., R. Johnsonbaugh, S. Jost, *Pattern Recognition and Image Analysis*, Prentice Hall, Inc., Upper Saddle River, NJ, 1996.

Han J., M. Kamber, *Data Mining: Concepts and Techniques*, 2nd edition, Morgan Kaufmann, 2006.

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

Jackson J., Data Mining: A Conceptual Overview, *Communications of the Association for Information Systems*, Vol. 8, 2002, pp. 267–296.

Jain A., R. P. W. Duin, J. Mao, Statistical Pattern Recognition, *IEEE Transaction on Pattern Analysis and Machine Intelligence*, Vol. 22, No. 1, January 2000, pp. 4–37.

Jurafsky D., J. H. Martin, Logistic Regression, book chapter in *Speech and Language Processing*, 3rd edition, Prentice Hall, 2017.

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

McCullagh P., J. A. Nelder, *Generalized Linear Models*, 2nd edition, Chapman&Hall, London, 1994.

Metz C. E., B. A. Herman, C. A. Roe, Statistical Comparison of Two ROC-curve Estimates Obtained from Partially-paired Datasets, *Medical Decision Making*, Vol. 18, No. 1, January–March 1998, pp. 110–124.

Nisbet R., J. Elder, G. Miner, Model Evaluation and Enhancement, *Handbook of Statistical Analysis and Data Mining Applications*, 2009, pp. 285–312.

Norusis M. J., *SPSS 7.5: Guide to Data Analysis*, Prentice-Hall, Inc., Upper Saddle River, New Jersey, 1997.

Reid Nancy, David R. Cox, On Some Principles of Statistical Inference, *International Statistical Review*, Vol. 83, No. 2, 2015, pp. 293–308.

Smith M., *Neural Networks for Statistical Modeling*, Van Nostrand Reinhold Publ., New York, 1993.

Trueblood R. P., J. N. Lovett, *Data Mining and Statistical Analysis using SQL*, Apress, Berkeley, CA, 2001.

Walpore R. E., R. H. Myers, *Probability and Statistics for Engineers and Scientists*, 4th edition MacMillan Publishing Company, New York, 1989.

Witten I. H., E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*, Morgan Kaufmannn Publ., Inc., New York, 1999.

Xie J., Z. Qiu, The Effect of Imbalanced Data Sets on LDA: A Theoretical and Empirical Analysis, *Pattern Recognition*, Volume 40, No. 2, February 2007, pp. 557–562.

Yang Q, X. Wu, 10 Challenging Problems in Data Mining Research, *International Journal of Information Technology & Decision Making*, Vol. 5, No. 4, 2006, pp. 597–604.

Zuev, K. M., Statistical Inference, *ACM Lecture Notes*, arXiv:1603.04929v1, March 2016.

CHAPTER 6

Alpaydin A, *Introduction to Machine Learning*, 2nd edition, The MIT Press, 2010.

Cieslak D. A., N.V. Chawla, Learning Decision Trees for Unbalanced Data, *European Conference on Principles and Practice of Knowledge Discovery in Databases (ECML/PKDD)*, Antwerp, Belgium, 2008.

Clementine, http://www.isl.co.uk/clem.html

Darlington J., Y. Guo, J. Sutiwaraphun, H. W. To, Parallel Induction Algorithms for Data Mining, *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining KDD'97*, 1997, pp. 35–43.

Diettrich T. G., Machine-Learning Research: Four Current Directions, *AI Magazine*, Winter 1997, pp. 97–136.

Dzeroski S., N. Lavrac, eds., *Relational Data Mining*, Springer, Berlin, Germany, 2001.

Finn P., S. Muggleton, D. Page, A. Srinivasan, Pharmacophore Discovery using the Inductive Logic Programming System Prolog, *Machine Learning*, Special Issue on Applications and Knowledge Discovery, Vol. 33, No. 1, 1998, pp. 13–47.

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

John G. H., Stock Selection Using Rule Induction, *IEEE Expert: Intelligent Systems & Their Applications*, Vol. 11, No. 5, October 1996, pp. 52–58.

King R. D., M. Quali, A. T. Strong, A. Aly, A. Elmaghraby, M. Kantardzic, D. Page, Is it Better to Combine Predictions?, *Protein Engineering*, Vol. 13, No. 1, 2000, pp. 15–19.

Leondes C. T., *Knowledge-Based Systems: Techniques and Applications*, Academic Press, San Diego, 2000.

Li W., J. Han, J. Pei, CMAR: Accurate and Efficient Classification Based on Multiple Class-Association Rules, *Proceedings on 2001 International Conference on Data Mining (ICDM'01)*, San Jose, CA, November 2001.

Luger G. F., W. A. Stubblefield, *Artificial Intelligence: Structures and Strategies for Complex Problem Solving*, Addison Wesley Longman, Inc., Harlow, England, 1998.

Maimon O., M. Last, *Knowledge Discovery and Data Mining: The Info-Fuzzy Network (IFN) Methodology*, Kluwer Academic Publishers, Boston, MA, 2001.

Maimon Oded Z., Rokach Lior, *Data Mining With Decision Trees: Theory And Applications*, 2nd edition, World Scientific, 2014.

McCarthy J., Phenomenal Data Mining, *CACM*, Vol. 43, No. 8, August 2000, pp. 75–79.

Mitchell T. M., Does Machine Learning Really Work?, *AI Magazine*, Fall 1997, pp. 11–20.

Mitchell T., *Machine Learning*, McGraw Hill, New York, NY, 1997.

Nisbet R., Elder J., Miner G., Classification, in *Handbook of Statistical Analysis and Data Mining Applications*, 2009, pp. 235–258.

Piramuthu S., Input Data for Decision Trees, *Expert Systems with Applications*, Vol. 34, No. 2, February 2008, pp. 1220–1226.

Poole D., A. Mackworth, R. Goebel, *Computational Intelligence: A Logical Approach*, Oxford University Press, Inc., New York, 1998.

Quinlan J. R., *C4.5: Programs for Machine Learning*, Morgan Kaufmann Publ. Inc., San Mateo, CA, 1992.

Russell S., P. Norvig, *Artificial Intelligence: A Modern Approach*, Prentice Hall, Upper Saddle River, NJ, 1995.

Therneau T. M., et al., An Introduction to Recursive Partitioning Using the RPART Routines, *Mayo Foundation*, February 2018.

Thrun S., C. Faloutsos, Automated Learning and Discovery, *AI Magazine*, Fall 1999, pp. 78–82.

Wang Y., et al., Unifying the Split Criteria of Decision Trees Using Tsallis Entropy, arXiv:1511.08136v5, August 2016.

Witten I. H., E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques*, 2nd edition, Elsevier Inc., 2005.

Xindong Wu, et al. Top 10 Algorithms in Data Mining, *Knowledge Information Systems*, Vol. 14, 2008, pp. 1–37.

CHAPTER 7

Banjanovic-Mehmedovic Lejla, Amel Hajdarevic, Mehmed Kantardzic, Fahrudin Mehmedovic, Izet Dzananovic, Neural Network-based Data-driven Modelling of Anomaly Detection in Thermal Power Plant, *Automatika – Journal for Control, Measurement, Electronics, Computing and Communications*, Taylor & Francis, Vol. 58, No. 1, July 2017, pp. 69–79.

Bengio S., O. Vinyals, N. Jaitly, and N. Shazeer Scheduled sampling forsequence prediction with recurrent neural networks, *Technical Report*, arXiv:1506.03099, 2015.

Bengio Y., A. Courville, P. Vincent, Representation Learning: A Review and New Perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, Vol. 35, No. 8, 2013, pp. 1798–1828.

Benitez J. M., J. L. Castro, I. Requena, Are Artificial neural networks Black Boxes?, *IEEE Transactions on Neural Networks*, Vol. 8, No. 5, September 1997, pp. 1156–1164.

Berthold M., D. J. Hand, eds., *Intelligent Data Analysis – An Introduction*, Springer, Berlin, 1999.

Burda Y., R. Grosse, R. Salakhutdinov, Importance weighted autoencoders, arXiv preprint arXiv:1509.00519, 2015.

Chen, T., M. Li, Y. Li, M. Lin, N. Wang, M. Wang, T. Xiao, B. Xu, C. Zhang, and Z. Zhang, MXNet: A flexible and efficient machine learning library for heterogeneous distributed systems, arXiv preprint arXiv:1512.01274, 2015.

Cherkassky V., F. Mulier, *Learning from Data: Concepts, Theory and Methods*, John Wiley & Sons, Inc., New York, 1998.

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

Deng L., D. Yu, *Deep Learning: Methods and Applications*, Now Publishers, 2014.

Dreyfus G., *Neural Networks: Methodology and Applications*, Springer, 2005.

Embrechts M. J., Neural Network for Data Mining, in "*Intelligent Engineering Systems through Artificial Neural Networks*", P. Chen, B. R. Fernandez, J. Gosh, eds., ASME Press, 1995, pp. 771–778.

Engel A., C. Van den Broeck, *Statistical Mechanics of Learning*, Cambridge University Press, Cambridge, England, 2001.

Fandango A., *Mastering TensorFlow 1.x: Advanced Machine Learning and Deep Learning Concepts Using TensorFlow 1.x and Keras*, Packt Publishing, 2018.

Fayyad U. M., G. Piatetsky-Shapiro, P. Smith, R. Uthurusamy, eds., *Advances in Knowledge Discovery and Data Mining*, AAAI Press/MIT Press, Cambridge, 1996.

Finn P., S. Muggleton, D. Page, A. Srinivasan, Pharmacophore Discovery using the Inductive Logic Programming System Prolog, *Machine Learning*, Special Issue on Applications and Knowledge Discovery, Vol. 33, No. 1, 1998, pp. 13–47.

Fu L., *Neural Networks in Computer Intelligence*, Mc Graw-Hill Inc., New York, 1994.

Fu L., An Expert Network for DNA Sequence Analysis, *IEEE Intelligent Systems*, January/February 1999, pp. 65–71.

Goodfellow I., Y. Bengio, A. Courville, *Deep Learning*, MIT Press, November 2016.

Hagan M. T., H. B. Demuth, M. Beale, *Neural Network Design*, PWS Publishing Co., Boston, 1996.

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

Haykin S., *Neural Networks: A Comprehensive Foundation*, Prentice Hall, Upper Saddle River, 1999.

Haykin S., *Neural Networks and Learning Machines*, 3rd edition, Pearson Education Co., 2009.

Heaton J., *Introduction to Neural Networks with Java*, Heaton Research Inc, 2005.

Jang J. R., C. Sun, Neuro-Fuzzy Modeling and Control, *Proceedings of the IEEE*, Vol. 83, No. 3, March 1995, pp. 378–406.

Jang J. -S. R., C. -T. Sun, E. Mizutani, *Neuro-Fuzzy and Soft Computing: A Computational Approach to Learning and Machine Intelligence*, Prentice Hall, Inc., Upper Saddle River, 1997.

Jin H., H. Shum, K. Leung, M. Wong, Expanding Self-Organizing Map for Data Visualization and Cluster Analysis, *Information Sciences*, Vol. 163, No. 1–3, June 2004, pp. 157–173.

Kanevski M., *Advanced Mapping of Environmental Data/Geostatistics, Machine Learning and Bayesian Maximum Entropy*, EPFL Press, Lausanne, 2008a.

Kanevski M., Classification of Interest Rate Curves Using Self-Organizing Maps, February 2008b, http://arxiv.org/PS_cache/arxiv/pdf/0709/0709.4401v1.pdf

Kantardzic M., A. A. Aly, A. S. Elmaghraby, Visualization of Neural-Network Gaps Based on Error Analysis, *IEEE Transaction on Neural Networks*, Vol. 10, No. 2, March 1999, pp. 419–426.

Kaudel A., M. Last, H. Bunke, eds., *Data Mining and Computational Intelligence*, Physica-Verlag, Heidelberg, Germany, 2001.

King R. D., M. Quali, A. T. Strong, A. Aly, A. Elmaghraby, M. Kantardzic, D. Page, Is It Better to Combine Predictions?, *Protein Engineering*, Vol. 13, No. 1, 2000, pp. 15–19.

Kukar M., Quality Assessment of Individual Classifications in Machine Learning and Data Mining, *Knowledge Information Systems*, Vol. 9, No. 3, 2006, pp. 364–384.

Munakata T., *Fundamentals of the new Artificial Intelligence: Beyond Traditional Paradigm*, Springer, New York, 1998.

Pal S. K., S. Mitra, *Neuro-Fuzzy Pattern Recognition: Methods in Soft Computing*, John Wiley & Sons, Inc., New York, 1999.

Petlenkov A., et al., Application of Self Organizing Kohonen Map to Detection of Surgeon Motions During Endoscopic Surgery, *Proceedings of the 2008 IEEE World Congress on Computational Intelligence (WCCI2008)*, Hong Kong, 2008.

Rocha M., P. Cortez, J. Neves, Evolution of Neural Networks for Classification and Regression, *Neurocomputing*, Vol. 70, No. 16–18, October 2007, pp. 2809–2816.

Smith M., *Neural Networks for Statistical Modeling*, Van Nostrand Reinhold Publ., New York, 1993.

Van Rooij A. J. F., L. C. Jain, R. P. Johnson, *Neural Network Training Using Genetic Algorithms*, World Scientific Publ. Co., Singapore, 1996.

Yann LeCun, Yoshua Bengio, Geoffrey Hinton, Deep Learning, *Nature*, Vol. 521, May 2015, pp. 436–444.

Zurada J. M., *Introduction to Artificial Neural Systems*, West Publishing Co., St. Paul, 1992.

CHAPTER 8

Gavin Brown, Ensemble Learning, In *Encyclopedia of Machine Learning*, C. Sammut & G.I. Webb (Eds.), Springer Press, 2010.

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

Cortes C., et al., Ensemble Methods for Structured Prediction, Proceedings of the 31. *International Conference on Machine Learning, JMLR: W&CP*, Vol. 32. Beijing, China, 2014.

Dietterich T. G., Ensemble Methods in Machine Learning, In *Lecture Notes in Computer Science on Multiple Classifier Systems*, Vol. 1857, Springer Berlin/Heidelberg, 2000.

Gérard Biau G., E. Scornet, A Random Forest Guided Tour, arXiv:1511.05741, November 2015.

Kuncheva L. I., *Combining Pattern Classifiers: Methods and Algorithms*, Wiley Press, 2004.

Narassiguin A., et al., An Extensive Empirical Comparison of Ensemble Learning Methods for Binary Classification, *Pattern Analysis and Applications*, Vol. 19, No. 4, November 2016, pp. 1093–1128.

Özyer T., R. Alhajj, K. Barker, Intrusion Detection by Integrating Boosting Genetic Fuzzy Classifier and Data Mining Criteria for Rule Pre-screening, *Journal of Network and Computer Applications*, Vol. 30, No. 1, January 2007, pp. 99–113.

Roadknight C., et al., An ensemble of machine learning and anti-learning methods for predicting tumour patient survival rates, *IEEE International Conference on Data Science and Advanced Analytics (DSAA),* 2015.

F. Roli, *Mini Tutorial on Multiple Classifier Systems, School on the Analysis of Patterns*, Cagliari, Italy, 2009.

Settles B., Active Learning Literature Survey, *Computer Sciences Technical Report 1648*, University of Wisconsin–Madison, January 2010.

Sewell M., *Ensemble Learning*, University College London, August 2008, http://machine-learning.martinsewell.com/ensembles/ensemble-learning.pdf

Stamatatos E., G. Widmar, Automatic Identification of Music Performers with Learning Ensembles, *Artificial Intelligence*, Vol. 165, No. 1, June 2005, pp. 37–56.

Zhong-Hui W., W. Li, Y. Cai, X. Xu, An Empirical Comparison of Ensemble Classification Algorithms with Support Vector Machines, *Proceedings of the Third International Conference on Machine Laming and Cybernetics*, Shanghai, August 2004.

CHAPTER 9

Anderson C., D. Lee, N. Dean, Spatial Clustering of Average Risks and Risk Trends in Bayesian Disease Mapping, *Biometrical Journal*, Vol. 59, No. 1, 2017, pp. 41–56.

Bouveyron C., B. Hammer, T. Villmann, Recent Developments in Clustering Algorithms, *ESANN 2012 Proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, Bruges, Belgium, April 2012.

Bow S., *Pattern Recognition and Image Preprocessing*, Marcel Dekker, Inc, New York, 1992.

Chen C. H., L. F. Pau, P. S. P. Wang, *Handbook of Pattern Recognition & Computer Vision*, World Scientific Publ. Co., Singapore, 1993.

Dzeroski S., N. Lavrac, eds., *Relational Data Mining*, Springer, Berlin, Germany, 2001.

Gose E., R. Johnsonbaugh, S. Jost, *Pattern Recognition and Image Analysis*, Prentice Hall, Inc., Upper Saddle River, NJ, 1996.

Han J., et al., Spatial Clustering Methods in Data Mining: A Survey, in "*Geographic Data Mining and Knowledge Discovery*", Miller H., and Han J., eds., Taylor and Francis Publ. Inc., 2001.

Han J., M. Kamber, *Data Mining: Concepts and Techniques*, 2nd edition, Elsevier Inc., 2006.

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

Hennig C., Cluster Validation.by Measurement of Clustering Characteristics Relevant to the User, arXiv:703.09282, March 2017.

Jain A. K., M. N. Murty, P. J. Flynn, Data Clustering: A Review, *ACM Computing Surveys*, Vol. 31, No. 3, September 1999, pp. 264–323.

Jain A.K., Data Clustering: 50 Years Beyond K-Means, *Pattern Recognition Letters*, 2009.

Jin H., H. Shum, K. Leung, M. Wong, Expanding Self-Organizing Map for Data Visualization and Cluster Analysis, *Information Sciences*, Vol. 163, No. 1–3, June 2004, pp. 157–173.

Karypis G., E. Han, V. Kumar, Chameleon: Hierarchical Clustering Using Dynamic modeling, *Computer*, Vol. 32, No. 8, August 1999, pp. 68–75.

Lee I., J. Yang, Common Clustering Algorithms, Comprehensive Chemometrics, 2009, Chapter 2.27, pp. 577–618.

Moore S. K., Understanding the Human Genoma, *Spectrum*, Vol. 37, No. 11, November 2000, pp. 33–35.

Munakata T., *Fundamentals of the New Artificial Intelligence: Beyond Traditional Paradigm*, Springer, New York, 1998.

Norusis M. J., *SPSS 7.5: Guide to Data Analysis*, Prentice-Hall, Inc., Upper Saddle River, New Jersey, 1997.

Poole D., A. Mackworth, R. Goebel, *Computational Intelligence: A Logical Approach*, Oxford University Press, Inc., New York, 1998.

Shirkhorshidi A., et al., Big Data Clustering: A Review, *International Conference on Computational Science and Its Applications ICCSA*, 2014, pp. 707–720.

Shyam Boriah, Varun Chandola, Vipin Kumar, Similarity Measures for Categorical Data: A Comparative Evaluation, *SIAM Conference*, 2008, pp. 243–254.

Slawomir Wierzchoń, Mieczyslaw Kłopotek, *Modern Algorithms of Cluster Analysis*, Springer, 2018.

Tan P.-N., M. Steinbach, V. Kumar, *Introduction to Data Mining*, Pearson Addison-Wesley, 2006.

Westphal C., T. Blaxton, *Data Mining Solutions: Methods and Tools for Solving Real-World Problems*, John Wiley & Sons, Inc., New York, 1998.

Witten I. H., E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*, Morgan Kaufmannn Publ., Inc., New York, 1999.

CHAPTER 10

Adamo J., *Data Mining for Association Rules and Sequential Patterns*, Springer, New York, 2001.

Beyer K., R. Ramakrishnan, Bottom-up Computation of Sparse and Iceberg Cubes, *Proceedings of 1999 ACM-SIGMOD International Conference on Management of Data (SIGMOD'99)*, Philadelphia, PA, June, 1999, pp. 359–370.

Bollacker K. D., S. Lawrence, C. L. Giles, Discovering Relevant Scientific Literature on the Web, *IEEE Intelligent Systems*, March/April 2000, pp. 42–47.

Chakrabarti S., Data Mining for Hypertext: A Tutorial Survey, *SIGKDD Explorations*, Vol. 1, No. 2, January 2000, pp. 1–11.

Chakrabarti S., et al. Mining the Web's Link Structure, *Computer*, Vol. 32, No. 8, August 1999, pp. 60–67.

Chang G., M. J. Haeley, J. A. M. McHugh, J. T. L. Wang, *Mining the World Wide Web: An Information Search Approach*, Kluwer Academic Publishers, Boston, MA, 2001.

Chen M., J. Park, P. S. Yu, Efficient Data Mining for Path Traversal Patterns, *IEEE Transaction on Knowledge and Data Engineering*, Vol. 10, No. 2, March/April 1998, pp. 209–214.

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

Clementine, http://www.isl.co.uk/clem.html

Cromp R. F., Campbell W. J., Data Mining of Multidimensional Remotely Sansad Images, *Proceedings of the CIKM'93 Conference*, Washington, DC, 1993, pp. 471–480.

Darlington J., Guo Y., Sutiwaraphun J., To H. W., Parallel Induction Algorithms for Data Mining, *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining KDD'97*, 1997, pp. 35–43.

Fayyad U. M., G. Piatetsky-Shapiro, P. Smith, R. Uthurusamy, eds., *Advances in Knowledge Discovery and Data Mining*, AAAI Press/MIT Press, Cambridge, 1996.

Fukada T., Y. Morimoto, S. Morishita, T. Tokuyama, Data Mining Using Two-Dimensional Optimized Association Rules: Scheme, Algorithms, and Visualization, *Proceedings of SIGMOD'96 Conference*, Montreal, 1996, pp. 13–23.

Han J., Towards On-Line Analytical Mining in Large Databases, *SIGMOD Record*, Vol. 27, No. 1, 1998, pp. 97–107.

Han J., M. Kamber, *Data Mining: Concepts and Techniques*, 2nd edition, Elsevier Inc., 2006.

Han J., J. Pei, Mining Frequent Patterns by Pattern-Growth: Methodology and Implications, *SIGKDD Explorations*, Vol.2, No. 2, December 2000, pp. 14–20.

Han E., G. Karypis, V. Kumar, Scalable Parallel Data Mining for Association Rules, *Proceedings of the SIGMOD'97 Conference*, Tucson, 1997a, pp. 277–288.

Han J., K. Koperski, N. Stefanovic, GeoMiner: A System Prototype for Spatial Data Mining, *Proceedings of the SIGMOD'97 Conference*, Arizona, 1997b, pp. 553–556.

Han J., S. Nishio, H. Kawano, W. Wang, Generalization-Based Data Mining in Object-Oriented Databases Using an Object Cube Model, *Proceedings of the CASCON'97 Conference*, Toronto, November 1997c, pp. 221–252.

Hedberg S. R., Data Mining Takes Off at the Speed of the Web, *IEEE Intelligent Systems*, November/December 1999, pp. 35–37.

Hilderman R. J., H. J. Hamilton, *Knowledge Discovery and Measures of Interest*, Kluwer Academic Publishers, Boston, MA, 2001.

Kasif S., Datascope: Mining Biological Sequences, *IEEE Intelligent Systems*, November/December 1999, pp. 38–43.

Kosala R., H. Blockeel, Web Mining Research: A Survey, *SIGKDD Explorations*, Vol. 2, No. 1, July 2000, pp. 1–15.

Kowalski G. J., M. T. Maybury, *Information Storage and Retrieval Systems: Theory and Implementation*, Kluwer Academic Publishers, Boston, 2000.

Liu B., W. Hsu, L. Mun, H. Lee, Finding Interesting Patterns Using User Expectations, *IEEE Transactions on Knowledge and Data Engineering*, Vol. 11, No. 6, November/December 1999, pp. 817–825.

McCarthy J., Phenomenal Data Mining, *CACM*, Vol. 43, No. 8, August 2000, pp. 75–79.

Moore S. K., Understanding the Human Genoma, *Spectrum*, Vol. 37, No. 11, November 2000, pp. 33–35.

Mulvenna M. D., et al., eds., Personalization on the Net Using Web Mining, A Collection of Articles, *CACM*, Vol. 43, No. 8, August 2000.

Narvekar M., et al, An Optimized Algorithm for Association Rule Mining Using FP Tree, *Procedia Computer Science*, Vol. 45, 2015, pp. 101–110.

Ng R. T., L. V. S. Lakshmanan, J. Han, A. Pang, Exploratory Mining and Optimization of Constrained Association Queries, *Technical Report*, University of British Columbia and Concordia University, October 1997.

Park J. S., M. Chen, P. S. Yu, Efficient Parallel Data Mining for Association Rules, *Proceedings of the CIKM'95 Conference*, Baltimore, 1995, pp. 31–36.

Pinto H., J. Han, J. Pei, K. Wang, Q. Chen, and U. Dayal, Multi-dimensional Sequential Pattern Mining, *Proceedings of the 2001 International Conference On Information and Knowledge Management (CIKM'01)*, Atlanta, GA, November 2001.

Pradhan G. N., Association Rule Mining in Multiple, Multidimensional Time Series Medical Data, *Journal of Healthcare Informatics Research*, Vol. 1, 2017, pp. 92–118.

Salzberg S. L., Gene Discovery in DNA Sequences, *IEEE Intelligent Systems*, November/December 1999, pp. 44–48.

Shaheen M., An Algorithm of Association Rule Mining for Microbial Energy, Prospection, *Scientific Reports* | 7:|46108 | DOI: 10.1038/srep46108, April 2017.

Spiliopoulou M., The Laborious Way from Data Mining to Web Log Mining, *Computer Systems in Science & Engineering*, Vol. 2, 1999, pp. 113–125.

Thuraisingham B., *Managing and Mining Multimedia Databases*, CRC Press LLC, Boca Raton, FL, 2001.

Wang Y., An Algorithm for Mining of Association Rules for the Information Communication Network Alarms Based on Swarm Intelligence, *Mathematical Problems in Engineering*, Vol. 2014, Article ID 894205, 14 pages, January 2014.

Witten I. H., E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*, Morgan Kaufmannn Publ., Inc., New York, 1999.

Wu X., et al, Top 10 Algorithms in Data Mining, *Knowledge and Information Systems*, Vol. 14, 2008, pp. 1–37.

Yang Q, X. Wu, 10 Challenging Problems in Data Mining Research, *International Journal of Information Technology & Decision Making*, Vol. 5, No. 4, 2006, pp. 597–604.

CHAPTER 11

Adedoyin-Olowe M., et al., A Survey of Data Mining Techniques for Social Network Analysis, *Journal of Data Mining & Digital Humanities*, June 2014.

Aggarwal C. C., C. Zhai, *Mining Text Data*, Springer, 2012.

Akerkar, R., P. Lingras, *Building an Intelligent Web: Theory and Practice*, Jones and Bartlett Publishers, Sudbury, MA, 2008.

Allahyari M., A Brief Survey of Text Mining: Classification, Clustering and Extraction Techniques, arXiv:1707.02919v2 [cs.CL], July 2017.

Chang, G., M. J. Haeley, J. A. M. McHugh, J. T. L. Wang, *Mining the World Wide Web: An Information Search Approach*, Kluwer Academic Publishers, Boston, MA, 2001.

Fan F., L. Wallace, S. Rich, Z. Zhang, Tapping the power of text mining, *Communications of ACM*, Vol. 49, No. 9, 2006, 76–82.

Garcia, E., *SVD and LSI Tutorial 4: Latent Semantic Indexing (LSI) How-to Calculations*, Mi Islita, 2006, http://www.miislita.com/information-retrieval-tutorial/svd-lsi-tutorial-4-lsi-how-to-calculations.html.

Gerrikagoitia J. K., New trends of Intelligent E-Marketing based on Web Mining for e-shops, *International Conference on Strategic Innovative Marketing, IC-SIM 2014*, Madrid, Spain, September 2014.

Han, J. and M. Kamber, *Data Mining: Concepts and Techniques*, 2nd edition, Morgan Kaufmann, San Francisco, 2006.

Jackson P., I. Moulinier, *Natural Language Processing for Online Applications: Text Retrieval, Extraction and Categorization*, John Benjamins Publ. Co., Amsterdam, 2007.

Jurafsky D., J. H. Martin, *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*, 3rd edition, Prentice Hall, 2017.

Langville AN, C.D. Meyer, *Google's PageRank and Beyond: The Science of Search Engine Rankings*, Princeton University Press, Princeton, 2006.

Liu B., *Web Data Mining: Exploring Hyperlinks, Contents and Usage Data*, Springer, Heidelberg, 2007.

Mulvenna, M. D. et al., eds., Personalization on the Net using Web Mining, *CACM*, Vol. 43, No. 8, 2000.

Nisbet R., J. Elder, G. Miner, Advanced Algorithms for Data Mining, in *Handbook of Statistical Analysis and Data Mining Applications*, 2009, pp. 151–172.

Qingyu Zhang, Richard S. Segall, Review of Data, Text and Web Mining Software, *Kybernetes*, Vol. 39, No. 4, 2010, pp. 625 – 655.

Sirmakessis S., *Text Mining and its Applications*, Springer-Verlag, Berlin, 2003.

Steven Struhl, *Practical Text Analytics: Interpreting Text and Unstructured Data for Business Intelligence*, Kogan Page Limited, July 2015.

Tyagi N., Web Structure Mining Algorithms: A Survey, *Big Data Analytics*, October 2017, pp. 305–317.

Zhang Y. et al, *Computational Web Intelligence: Intelligent Technology for Web Applications*, World Scientific Publ. Co., Singapore, 2004.

Zhang X., Edwards J., Harding J., Personalised online sales using web usage data mining, *Computers in Industry*, Vol. 58, No. 8–9, December 2007, pp. 772–782.

CHAPTER 12

Antunes, C., A. Oliveira, Temporal Data Mining: An Overview, *Proceedings of Workshop on Temporal Data Mining (KDD'01)*, 2001, 1–13.

Arulkumaran K., et al., A Brief Survey of Deep Reinforcement Learning, *IEEE Signal Processing Magazine*, Vol. 34, No. 6, November 2017, pp. 26–38.

Bar-Or A., R. Wolff, A. Schuster, and D. Keren, Decision Tree Induction in High Dimensional, Hierarchically Distributed Databases, *Proceedings of 2005 SIAM International Conference on Data Mining (SDM'05)*, Newport Beach, CA, April 2005.

Basak J. and R. Kothari, A Classification Paradigm for Distributed Vertically Partitioned Data, *Neural Computation*, Vol. 16, No. 7, July 2004, pp.1525–1544.

Bhaduri K., R. Wolff, C. Giannella, H. Kargupta, Distributed Decision-tree Induction in Peer-to-peer Systems, *Statistical Analysis and Data Mining*, Vol. 1, No. 2, 2008, pp. 85–103.

Bishop C. M., *Pattern Recognition and Machine Learning*, Springer, 2006.

Branch J., B. Szymanski, R. Wolff, C. Gianella, H. Kargupta, In-network Outlier Detection in Wireless Sensor Networks, *Proceedings of the 26th International Conference on Distributed Computing Systems (ICDCS)*, July 2006, pp. 102–111.

Cannataro M., D. Talia, The Knowledge Grid, *Communications of the ACM*, Vol. 46, No. 1, January 2003, pp. 89–93.

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

Congiusta A., D. Talia, P. Trunfio, Service-oriented Middleware for Distributed Data Mining on the Grid, *Journal of Parallel and Distributed Computing*, Vol. 68, No. 1, January 2008, pp. 3–15.

Copp C., Data Mining and Knowledge Discovery Techniques, *Defence Today*, NCW 101, 2008, http://www.ausairpower.net/NCW-101-17.pdf.

Crosby M., BlockChain Technology: Beyond Bitcoin, *Applied Innovation Review*, No. #2, June 2016.

Datta S., K. Bhaduri, C. Giannella, R. Wolff, H. Kargupta, Distributed data mining in peer-to-peer networks, *IEEE Internet Computing*, Vol. 10, No. 4, 2006, pp. 18–26.

Ester M., H.-P. Kriegel and J. Sander, Spatial Data Mining: A Database Approach, *Proceedings of 5th International Symposium on Advances in Spatial Databases*, 1997, pp. 47–66.

Fuchs Erich, Thiemo Gruber, Jiri Nitschke, Bernhard Sick, On-line motif detection in time series with Swift Motif, *Pattern Recognition*, Vol. 42, 2009, pp. 3015–3031.

Faloutsos C., Mining Time Series Data, *Tutorial ICML 2003*, Washington DC, USA, August 2003.

Gorodetsky V., O. Karsaeyv, V. Samoilov, Software Tool for Agent Based Distributed Data Mining, *International Conference on Integration of Knowledge Intensive Multi-Agent Systems (KIMAS)*, Boston, MA, October 2003.

Gosavi A., *Simulation-Based Optimization: Parametric Optimization Techniques and Reinforcement Learning*, 2nd edition, Springer, New York, NY, 2014.

Gosavi A., *A Tutorial for Reinforcement Learning*, Missouri University of Science and Technology, February 2017.

Guo H., Hsu W., A Survey of Algorithms for Real-Time Bayesian Network Inference, *AAAI-02/KDD-02/UAI-02 Workshop on Real-Time Decision Support and Diagnosis*, 2002.

Hammouda K., M. Kamel, HP2PC: Scalable Hierarchically-Distributed Peer-to-Peer Clustering, *Proceedings of the 2007 SIAM International Conference on Data Mining (SDM '07)*, Philadelphia, PA, 2007.

John McCullock, Q Learning: Step-By-Step Tutorial, http://mnemstudio.org/path-finding-q-learning-tutorial.htm.

Kang U., L. Akoglu, D. H. Chau, Big Graph Mining for the Web and Social Media: Algorithms, Anomaly Detection, and Applications, *WSDM*, 2014a, pp. 677–678.

Kang U., B. Meeder, E. E. Papalexakis, C. Faloutsos Heigen: Spectral Analysis for Billion-scale Graphs, *IEEE Transactions on Knowledge & Data Engineering*, Vol. 26, 2014b, pp. 350–362.

Keogh E., Data Mining and Machine Learning in Time Series Databases, *Tutorial ECML/PKDD 2003*, Cavtat-Dubrovnik (Croatia), September 2003.

Khan S., et al., Cloud-based Big Data Analytics – A Survey of Current Research and Future Directions, In V.B. Aggarwal, V. Bhatnagar, D.K. Mishra (eds.), *Big Data Analytics*, AISC Vol. 654, Springer, Singapore, 2015.

Kholod I., et al., Distributed Data Mining Framework for Cloud Service, *IT CoNvergence PRActice (INPRA)*, Vol. 3, No. 4, December 2015, pp. 19–33.

Koperski K., et al., Spatial Data Mining: Progress and Challenges, *SIGMOD'96 Workshop on Research Issues on Data Mining and Knowledge Discovery*, 1996.

Kotecha J. H., V. Ramachandran, and A. M. Sayeed, Distributed Multitarget Classification in Wireless Sensor Networks, *IEEE Journal of Selected Areas in Communications*, Vol. 23, No. 4, April 2005, pp. 703–713,.

Kriegel H.P., et al., Future Trends in Data Mining, *Data Mining and Knowledge Discovery*, Vol. 15, 2007, pp. 87–97.

Kumar A., Kantardzic M., Madden S., Guest Editors, Introduction: Distributed Data Mining— Framework and Implementations, *IEEE Internet Computing*, Vol. 10, No. 4, July/August 2006, pp. 15–17.

Lavrac N., et al., Introduction: Lessons Learned from Data Mining Applications and Collaborative Problem Solving, *Machine Learning*, Vol. 57, 2004, pp. 13–34.

Li S., T. Wu, and W. M. Pottenger, Distributed Higher Order Association Rule Mining Using Information Extracted from Textual Data, *SIGKDD Exploration*, Vol. 7, No. 1, 2005, pp. 26–35.

Li T., S. Zhu, and M. Ogihara, Algorithms for Clustering High Dimensional and Distributed Data, *Intelligent Data Analysis Journal*, Vol. 7, No. 4, 2003.

Lin J., C. Dyer, *Data-Intensive Text Processing with MapReduce*, Morgan & Claypool, Synthesis Lectures on Human Language Technologies, 2010.

Liu K., H. Kargupta, J. Ryan, Random Projection-Based Multiplicative Data Perturbation for Privacy Preserving Distributed Data Mining, *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, Vol. 18, No. 1, January 2006, pp. 92–106.

Martijn van Otterlo, Marco Wiering, Reinforcement Learning and Markov Decision Processes, in *"Adaptation, Learning, and Optimization" book series*, Vol. 12, Springer, 2012, pp. 3–42.

Miller H. J., Geographic Data Mining and Knowledge Discovery, in "*Handbook of Geographic Information Science*", John Wilson, A. Stewart Fotheringham, eds., Blackwell Publishing, 2008.

Neagu I., et al., A Holistic Analysis of Cloud Based Big Data Mining, *International Journal of Knowledge, Innovation and Entrepreneurship*, Vol. 2, No. 2, 2014, pp. 56–64.

Nisbet R., J. Elder, G. Miner, Advanced Algorithms for Data Mining, in *Handbook of Statistical Analysis and Data Mining Applications*, 2009, pp. 151–172.

J. Pearl, *Causality*, Cambridge University Press, New York, NY, 2000.

Pearl J., Statistics and Causal Inference: A Review, *Sociedad de Estad´istica e Investigaci´on Operativa Test*, Vol. 12, No. 2, 2003, pp. 281–345.

Petre R., Data mining in Cloud Computing, *Database Systems Journal*, Vol. III, no. 3, 2012, pp. 67–71.

John F. Roddick, Myra Spiliopoulou, A Survey of Temporal Knowledge Discovery Paradigms and Methods, *IEEE Transactions On Knowledge And Data Engineering*, Vol. 14, No. 4, July/August 2002.

Sethi, Tegjyot Singh, Mehmed Kantardzic. Handling Adversarial Concept Drift in Streaming Data, *Expert Systems with Applications*, Vol. 97, March 2018, pp.18–40.

Singh Sethi, T., M. M. Kantardzic, H. Hu, A Grid Density Based Framework for Classifying Streaming Data in the Presence of Concept Drift, *Journal of Intelligent Information Systems*, February 2016, Vol. 46, No. 1, pp 179–211.

Shekhar S., S. Chawla, Introduction to Spatial Data Mining, in *Spatial Databases: A Tour*, Prentice Hall, 2003.

Shekhar S., P. Zhang, Y. Huang, R. Vatsavai, Trends in Spatial Data Mining, as a book chapter in *Data Mining: Next Generation Challenges and Future Directions*. H. Kargupta, A. Joshi, K. Sivakumar and Y. Yesha (editors), AAAI/MIT Press, 2004.

Srivatsan Laxman and P. S. Sastry, A Survey of Temporal Data Mining, *Sadhana*, Vol. 31, Part 2, April 2006, pp. 173–198.

Stuart J. Russell, Peter Norvig, *Artificial Intelligence*, Pearson Education, Upper Saddle River, NJ, 2003.

Suton R. S., A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd edition, The MIT Press, Cambridge, 2015.

Talia D., Clouds for Scalable Big Data Analytics, *Computer*, IEEE, May 2013, pp. 898–101.

Talia D., *Data Analysis in the Cloud: Models, Techniques and Applications*, Elsevier Inc., 2015.

Varghese B., et al., Next Generation Cloud Computing: New Trends and Research Directions, *Future Generation Computer Systems*, Vol. 79, 2018, pp. 849–861.

Wasserman S., Faust K., *Social Network Analysis: Methods and Applications*, Cambridge University Press, 1994.

Wu Q., Nageswara S. V. Rao, Jacob Barhen, S. Sitharama Iyengar, Vijay K. Vaishnavi, Hairong Qi, Krishnendu Chakrabarty, On Computing Mobile Agent Routes for Data Fusion in Distributed Sensor Networks, *IEEE Transactions on Knowledge and Data Engineering*, Vol. 16, June 2004, pp. 740–753.

Xu X., N. Yuruk, Z. Feng, and T. Schweiger, SCAN: A Structural Clustering Algorithm for Networks, *Proceedings of the 13th International Conference on Knowledge Discovery and Data Mining (KDD '07)*, New York, NY, 2007, pp. 824–833.

Yang Q., Wu X., 10 Challenging Problems in Data Mining Research, *International Journal of Information Technology & Decision Making*, Vol. 5, No. 4, 2006, pp. 597–604.

Yu H. and Ee-Chien Chang, Distributed Multivariate Regression Based on Influential Observations, *Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Washington, DC, August 2003.

Zaki M., Y. Pan, Introduction: Recent Development in Parallel and Distributed Data Mining, *Distributed and Parallel Databases*, Vol. 11, No. 2, 2002.

CHAPTER 13

Cano, A., et al., A Classification Module for Genetic Programming Algorithms in JCLEC, *Journal of Machine Learning Research*, Vol.16, 2015, pp. 491–494.

Cox E., *Fuzzy Modeling and Genetic Algorithms for Data Mining and Exploration*, Morgan Kaufmann, 2005.

Dehuri S., et al., Genetic Algorithms for Multi-Criterion Classification and Clustering in Data Mining, *International Journal of Computing & Information Sciences*, Vol.4, No. 3, December 2006.

Fogel D., An Introduction to Simulated Evolutionary Optimization, *IEEE Transactions on Neural networks*, Vol. 5, No. 1, January 1994, pp. 3–14.

Fogel D. B., ed., *Evolutionary Computation*, IEEE Press, New York, 1998.

Fogel D. B., *Evolutionary Computing, Spectrum*, Vol. 37, No. 2, February 2000, pp. 26–32.

Freitas A., A Survey of Evolutionary Algorithms for Data Mining and Knowledge Discovery, in *Advances in evolutionary computing: theory and applications*, Springer Verlag, New York, 2003.

Goldenberg D. E., *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison Wesley, Reading, MA, 1989.

Hruschka E., R. Campello, A. Freitas, A. Carvalho, A Survey of Evolutionary Algorithms for Clustering, *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, Vol. 39, No. 2, March 2009, pp. 133–155.

Kaudel A., M. Last, H. Bunke, eds., *Data Mining and Computational Intelligence*, Physica-Verlag, Heidelberg, Germany, 2001.

Michalewicz Z., *Genetic Algorithms + Data Structures = Evolution Programs*, Springer, Berlin, Germany, 1999.

Munakata T., *Fundamentals of the New Artificial Intelligence: Beyond Traditional Paradigm*, Springer, New York, 1998.

Navet N., S. Chen, Financial Data Mining with Genetic Programming: A Survey and Look Forward, *The 56th Session of the International Statistical Institute (ISI2007)*, Lisbon, August 2007.

Qiong, Gu., et al., An Improved SMOTE Algorithm Based on Genetic Algorithm for Imbalanced Data Classification, *Journal of Digital Information Management*, Vol. 14, No. 2, April 2016, pp. 92–103.

Salleb-Aouissi A., C. Christel Vrain, C. Nortet, QuantMiner: A Genetic Algorithm for Mining Quantitative Association Rules, *Proceedings of the IJCAI-07*, 2007, pp. 1035–1040.

Shah S. C., A. Kusiak, Data mining and genetic algorithm based gene/SNP selection, *Artificial Intelligence in Medicine*, Vol. 31, No. 3, July 2004, pp. 183–196.

Sheppard, C., *Genetic Algorithms with Python*, Clinton Sheppard, 2018.

Van Rooij A. J. F., L. C. Jain, R. P. Johnson, *Neural Network Training Using Genetic Algorithms*, World Scientific Publ. Co., Singapore, 1996.

Vinh, N. N., et al., Incremental Spatial Clustering in Data Mining Using Genetic Algorithm and R-Tree, *Asia-Pacific Conference on Simulated Evolution and Learning – SEAL*, 2012, pp. 270–279.

CHAPTER 14

Chen S., A Fuzzy Reasoning Approach for Rule-Based Systems Based on Fuzzy Logic, *IEEE Transaction on System, Man, and Cybernetics*, Vol. 26, No. 5, October 1996, pp. 769–778.

Chen C. H., L. F. Pau, P. S. P. Wang, *Handbook of Pattern Recognition & Computer Vision*, World Scientific Publ. Co., Singapore, 1993.

Chen Y., T. Wang, B. Wang, and Z. Li, A Survey of Fuzzy Decision Tree Classifier, *Fuzzy Information and Engineering*, Vol. 1, No. 2, June 2009, pp. 149–159.

Chen, G., F. Liu, M. Shojafar, *Fuzzy System and Data Mining*, IOS Press, April 2016.

Cox E., *Fuzzy Modeling and Genetic Algorithms for Data Mining and Exploration*, Morgan Kaufmann, 2005.

Hüllermeier E., Fuzzy Sets in Machine Learning and Data Mining, *Applied Soft Computing*, January 2008.

Jang J. R., C. Sun, Neuro-Fuzzy Modeling and Control, *Proceedings of the IEEE*, Vol. 83, No. 3, March 1995, pp. 378–406.

Jang J., C. Sun, E. Mizutani, *Neuro-Fuzzy and Soft Computing: A Computational Approach to Learning and Machine Intelligence*, Prentice Hall, Inc., Upper Saddle River, 1997.

Kaudel A., M. Last, H. Bunke, eds., *Data Mining and Computational Intelligence*, Physica-Verlag, Heidelberg, Germany, 2001.

Klir G. J., B. Yuan, *Fuzzy Sets and Fuzzy Logic: Theory and Applications*, Prentice Hall Inc., Upper Saddle River, NJ, 1995.

Koczy L. T., K. Hirota, Size Reduction by Interpolation in Fuzzy Rule Bases, *IEEE Transaction on System, Man, and Cybernetics*, Vol. 27, No. 1, February 1997, pp. 14–25.

Kruse R., A. Klose, Recent Advances in Exploratory Data Analysis with Neuro-fuzzy Methods, *Soft Computing*, Vol. 8, No. 6, May 2004.

Laurent A., M. Lesot, eds., *Scalable Fuzzy Algorithms for Data Management and Analysis, Methods and Design*, IGI Global, 2010.

Lee E. S., H. Shih, *Fuzzy and Multi-level Decision Making: An Interactive Computational Approach*, Springer, London, 2001.

Li H. X., V. C. Yen, *Fuzzy Sets and Fuzzy Decision-Making*, CRC Press, Inc., Boca Raton, 1995.

Lin T. Y., N. Cerone, *Rough Sets and Data Mining*, Kluwer Academic Publishers, Inc., Boston, 1997.

Maimon O., M. Last, *Knowledge Discovery and Data Mining: The Info-Fuzzy Network (IFN) Methodology*, Kluwer Academic Publishers, Boston, MA, 2001.

Mendel J., Fuzzy Logic Systems for Engineering: A Tutorial, *Proceedings of the IEEE*, Vol. 83, No. 3, March 1995, pp. 345–377.

Miyamoto S., *Fuzzy Sets in Information Retrieval and Cluster Analysis*, Cluver Academic Publishers, Dodrecht, 1990.

Munakata T., *Fundamentals of the new Artificial Intelligence: Beyond Traditional Paradigm*, Springer, New York, 1998.

Nguyen, T., et al., Medical data classification using interval type-2 fuzzy logic system and wavelets, *Journal of Applied Soft Computing*, Vol. 30, No. C, May 2015, pp. 812–822.

Özyer T., R. Alhajj, K. Barker, Intrusion Detection by Integrating Boosting Genetic Fuzzy Classifier and Data Mining Criteria for Rule Pre-screening, *Journal of Network and Computer Applications*, Vol. 30, No. 1, January 2007, pp. 99–113.

Pal S. K., Mitra S., *Neuro-Fuzzy Pattern Recognition: Methods in Soft Computing*, John Wiley & Sons, Inc., New York, 1999.

Pedrycz W., F. Gomide, *An Introduction to Fuzzy Sets: Analysis and Design*, The MIT Press, Cambridge, 1998.

Pedrycz W., J. Waletzky, Fuzzy Clustering with Partial Supervision, *IEEE Transaction on System, Man, and Cybernetics*, Vol. 27, No. 5, October 1997, pp. 787–795.

Taneja S., A New Approach for Data Classification using Fuzzy Logic, *2016 6th International Conference - Cloud System and Big Data Engineering*, Noida, India, January 2016.

Taufik, A., et al., Land Cover Classification of Landsat 8 Satellite Data Based on Fuzzy Logic Approach, *IOP Conference Series: Earth and Environmental Science*, Vol. 37, Conference 1, June 2017.

Yager R. R., Targeted E-commerce Marketing Using Fuzzy Intelligent Agents, *IEEE Intelligent Systems*, November/December 2000, pp. 42–45.

Yeung D. S., E. C. C. Tsang, A Comparatice Study on Similarity-Based Fuzzy Reasoning Methods, *IEEE Transaction on System, Man, and Cybernetics*, Vol. 27, No. 2, April 1997, pp. 216–227.

Zadeh L. A., Fuzzy Logic = Computing with Words, *IEEE Transactions on Fuzzy Systems*, Vol. 4, No. 2, May 1996, pp. 103–111.

Zadeh L. A., Knowledge Representation in Fuzzy Logic, *IEEE Transactions on Knowledge and Data Engineering*, Vol. 1, No. 1, March 1989, pp. 89–99.

CHAPTER 15

Barry A. M. S., *Visual Intelligence*, State University, New York Press, New York, 1997.

Bikakis, N., Big Data Visualization Tools, in "*Encyclopedia of Big Data Technologies*", Sakr Sherif, Zomaya Albert, Eds., Springer, 2018.

Bohlen M., 3D Visual Data Mining – Goals and Experiences, *Computational Statistics & Data Analysis*, Volume 43, No. 4, August 2003, pp. 445–469.

Buja A., D. Cook, D. F. Swayne, *Interactive High-Dimensional Data Visualization*, http://www.research.att.com/andreas/xgobi/heidel.

Chen C., R. J. Paul, Visualizing a Knowledge Domain's Intellectual Structure, *Computer*, Vol. 36, No. 3, March 2001, pp. 65–72.

Draper G. M., L. Y. Livnat, R. F. Riesenfeld, A Survey of Radial Methods for Information Visualization, *IEEE Transaction on Visualization and Computer Graphics*, Vol. 15, No. 5, September/October 2009, pp. 759–776.

Eick S. G., Visual Discovery and Analysis, *IEEE Transaction on Visualization and Computer Graphics*, Vol. 6, No. 1, January-March 2000a, pp. 44–57.

Eick S. G., Visualizing Multi-dimensional Data, *Computer Graphics*, February 2000b, pp. 61–67.

Elmqvist N., J. Fekete, Hierarchical Aggregation for Information Visualization: Overview, Techniques and Design Guidelines, *IEEE Transactions on Visualization and Computer Graphics*, Vol. 16, No. 3, May/June 2010, pp. 439–454.

Estrin D., et al. Network Visualization with Nam, the VINT Network Animator, *Computer*, Vol. 33, No. 11, November 2000, pp. 63–68.

Faloutsos C., K. Lin, FastMap: A Fast Algorithm for Indexing, Data-Mining and Visualization of Traditional and Multimedia Datasets, *Proceedings of SIGMOD'95 Conference*, San Jose, 1995, pp. 163–174.

Fayyad U., G. Georges Grinstein, A. Wierse, *Information Visualization in Data Mining and Knowledge Discovery*, Morgan Kaufmann; 1st edition, 2001.

Fayyad U. M., G. G. Grinstein, A. Wierse, *Information Visualization in Data Mining and Knowledge Discovery*, Academic Press, San Diego, 2002a.

Fayyad U., G. G. Grinstein, A. Wierse, eds., *Information Visualization in Data Mining and Knowledge Discovery*, Morgan Kaufmann Publishers, San Francisco, CA, 2002b.

Ferreira de Oliveira M. C., H. Levkowitz, From Visual Data Exploration to Visual Data Mining: A Survey, *IEEE Transactions on Visualization and Computer Graphics*, Vol. 9, No. 3, July-September 2003, pp. 378–394.

Gallaghar R. S., *Computer Visualization: Graphics Techniques for Scientific and Engineering Analysis*, CRC Press, Inc., Boca Raton, 1995.

Hinneburg A., Keim D. A., Wawryniuk M., HD-Eye: Visual Mining of High-Dimensional Data, *IEEE Computer Graphics and Applications*, September/October 1999, pp. 22–31.

Hofman P., *Radviz*, http:www.cs.uml.edu/phoffman/viz.

Hofmann, M., A. Chisholm, *Text Mining and Visualization: Case Studies Using Open-Source Tools*, CRC Press, 2016.

Huang, M., Huang, W., *Innovative Approaches of Data Visualization and Visual Analytics*, IGI Global, 2013.

IBM, *Parallel Visual Explorer at Work in the Money Market*, http://wwws.ibm.com/news/950203/pve-03html.

Inselberg A., B. Dimsdale, Visualizing Multi-Variate Relations with Parallel Coordinates, *Proceedings of the Third International conference on Human-Computer Interaction*, New York, 1989, pp. 460–467.

Mackinlay J. D., Opportunities for Information Visualization, *IEEE Computer Graphics and Applications*, January/February 2000, pp. 22–23.

Masseglia F., P. Poncelet, T. Teisseire, *Successes and New Directions in Data Mining*, Idea Group Inc., 2007.

Plaisant C., The Challenge of Information Visualization Evaluation, *IEEE Proceedings of the Advanced Visual Interfaces*, Gallipoli, Italy, 2004, pp. 109–116.

Pu P., G. Melissargos, Visualizing Resource Allocation Tasks, *IEEE Computer Graphics and Applications*, July/August 1997, pp. 6–9.

Roth S. F., M. C. Chuah, S. Kerpedjiev, J. A. Kolojejchick, P. Lukas, Towards an Information Visualization Workspace: Combining Multiple Means of Expressions, *Human-Computer Interaction Journal*, July 1997, pp 61–70.

Segall, Richard S., Jeffrey S. Cook, *Handbook of Research on Big Data Storage and Visualization Techniques*, IGI Global, 2018.

Spence R., *Information Visualization*, Addison Wesley, Harlow, England, 2001.

Tergan S., T. Keller, *Knowledge and Information Visualization: Searching for Synergies*, Springer, 2005.

Thomsen E., *OLAP Solution: Building Multidimensional Information System*, John Wiley, New York, 1997.

Tufte E. R., *Beautiful Evidence*, 2nd edition, Graphic Press, LLC, January 2007.

Two Crows Corp., *Introduction to Data Mining and Knowledge Discovery*, Two Crows Corporation, 2005.

Wong P. C., Visual Data Mining, *IEEE Computer Graphics and Applications*, September/October 1999, pp. 20–21.

INDEX