The 4th edition substantially revises the first three editions of the book, with numerous enhancements and a reorganization of the technical contents. The core technical material, which handles different mining methodologies on general data types, is expanded and substantially enhanced. To keep the book concise and up-to-date, we have done the following major revisions: (1) Two chapters in the 3rd edition, "Getting to Know You Data" and "Data Preprocessing" are merged into one chapter "Data, Measurements and Data Preprocessing," with the "Data Visualization" section removed since the concepts are easy to understand, the methods have been covered in multiple, dedicated data visualization books, and the software tools are widely available on the web; (2) two chapters in the 3rd edition, "Data Warehousing and Online Analytical Processing" and "Data Cube Technology" are merged into one chapter, with some not well-adopted data cube computation methods and data cube extensions omitted, but with a newer concept, "Data Lakes" introduced; (3) the key data mining method chapters in the 3rd edition on pattern discovery, classification, clustering and outlier analysis are retained with contents substantially enhanced and updated; (4) a new chapter "Deep Learning" is added, with a systematic introduction to neural network and deep learning methodologies; (5) the final chapter on "Data Mining Trends and Research Frontiers" is completely rewritten with many new advanced topics on data mining introduced in comprehensive and concise way; and finally, (6) a set of appendices that briefly introduce essential mathematical background needed to understand the contents of this book.

The chapters of this new edition are described briefly as follows, with emphasis on the new material.

Chapter 1 provides an *introduction* to the multidisciplinary field of data mining. It discusses the evolutionary path of information technology, which has led to the need for data mining, and the importance of its applications. It examines various kinds of data to be mined, and presents a general classification of data mining tasks, based on the kinds of knowledge to be mined, the kinds of technologies used, and the kinds of applications that are targeted. It shows that data mining is a confluence of multiple disciplines, with broad applications. Finally, it discusses how data mining may impact society.

Chapter 2 introduces the *data, measurements and data preprocessing*. It first discusses data objects and attribute types, and then introduces typical measures for basic statistical data descriptions. It also introduces ways to measure similarity and dissimilarity for various kinds of data. Then, the chapter moves to introduce techniques for data preprocessing. In particular, it introduces the concept of data quality and methods for data cleaning and data integration. It also discusses various methods for data transformation and dimensionality reduction.

Chapter 3 provides a comprehensive introduction to *datawarehouses* and online analytical processing (*OLAP*). The chapter starts with a well-accepted definition of data warehouse, an introduction to the architecture, and the concept of data lake. Then it studies the logical design of a data warehouse as a multidimensional data model, and looks at OLAP operations and how to index OLAP data for efficient analytics. The chapter includes an in-depth treatment of the techniques of building data cube as a way of implementing a data warehouse.

Chapters 4 and 5 present methods for *mining frequent patterns, associations*, and *correlations* in large data sets. **Chapter 4** introduces fundamental concepts, such as market basket analysis, with many techniques for frequent itemset mining presented in an organized way. These range from the basic Apriori algorithm and its variations to more advanced methods that improve efficiency, including the frequent pattern growth approach, frequent pattern mining with vertical data format, and mining closed and max frequent itemsets. The chapter also discusses pattern evaluation methods and introduces measures for mining correlated patterns. **Chapter 5** is on advanced pattern mining methods. It discusses methods for pattern mining in multilevel and multidimensional space, mining quantitative association

rules, mining high-dimensional data, mining rare and negative patterns, mining compressed or approximate patterns, and constraint-based pattern mining. It then moves to advanced methods for mining sequential patterns and subgraph patterns. It also presents applications of pattern mining, including phrase mining in text data and mining copy and paste bugs in software programs.

Chapters 6 and 7 describe methods for *data classification*. Due to the importance and diversity of classification methods, the contents are partitioned into two chapters. **Chapter 6** introduces basic concepts and methods for classification, including decision tree induction, Bayes classification, *k*-nearest neighbor classifiers, and linear classifiers. It also discusses model evaluation and selection methods and methods for improving classification accuracy, including ensemble methods and how to handle imbalanced data. **Chapter 7** discusses advanced methods for classification, including feature selection, Bayesian belief networks, support vector machines, rule-based and pattern-based classification. Additional topics include classification with weak supervision, classification with rich data type, multiclass classification, distant metric learning, interpretation of classification, genetic algorithms and reinforcement learning.

Cluster analysis forms the topic of Chapters 8 and 9. **Chapter 8** introduces the basic concepts and methods for data clustering, including an overview of basic cluster analysis methods, partitioning methods, hierarchical methods, density-based and grid-based methods. It also introduces methods for the evaluation of clustering. **Chapter 9** discusses advanced methods for clustering, including probabilistic model-based clustering, clustering high-dimensional data, clustering graph and network data, and semisupervised clustering.

Chapter 10 introduces *deep learning*, which is a powerful family of techniques based on artificial neural networks with broad applications in computer vision, natural language processing, machine translation, social network analysis, and so on. We start with the basic concepts and a foundational technique called backpropagation algorithm. Then, we introduce various techniques to improve the training of deep learning models, including responsive activation functions, adaptive learning rate, dropout, pretraining, cross-entropy, and autoencoder. We also introduce several commonly used deep learning architectures, ranging from feed-forward neural networks, convolutional neural networks, recurrent neural networks, and graph neural networks.

Chapter 11 is dedicated to *outlier detection*. It introduces the basic concepts of outliers and outlier analysis and discusses various outlier detection methods from the view of degree of supervision (i.e., supervised, semisupervised, and unsupervised methods), as well as from the view of approaches (i.e., statistical methods, proximity-based methods, reconstruction-based methods, clustering-based methods, and classification-based methods). It also discusses methods for mining contextual and collective outliers, and for outlier detection in high-dimensional data.

Finally, in **Chapter 12**, we discuss *future trends* and *research frontiers* in data mining. We start with a brief coverage of mining complex data types, including text data, graphs and networks, and spatiotemporal data. After that, we introduce a few data mining applications, ranging from sentiment and opinion analysis, truth discovery and misinformation identification, information and disease propagation, to productivity and team science. The chapter then moves ahead to cover other data mining methodologies, including structuring unstructured data, data augmentation, causality analysis, network-as-a-context, and auto-ML. Finally, it discusses social impacts of data mining, including privacy-preserving data mining, human-algorithm interaction, fairness, interpretability and robustness, and data mining for social good.

Throughout the text, *italic* font is used to emphasize terms that are defined, and **bold** font is used to highlight or summarize main ideas. Sans serif font is used for reserved words. Bold italic font is used to represent multidimensional quantities.

This book has several strong features that set it apart from other textbooks on data mining. It presents a very broad yet in-depth coverage of the principles of data mining. The chapters are written to be as self-contained as possible, so they may be read in order of interest by the reader. Advanced chapters offer a larger-scale view and may be considered optional for interested readers. All of the major methods of data mining are presented. The book presents important topics in data mining regarding multidimensional OLAP analysis, which is often overlooked or minimally treated in other data mining books. The book also maintains web sites with a number of online resources to aid instructors, students, and professionals in the field. These are described further in the following.

To the instructor

This book is designed to give a broad, yet detailed overview of the data mining field. First, it can be used to teach an introductory course on data mining at an advanced undergraduate level or at the first-year graduate level. Moreover, the book also provides essential materials for an advanced graduate course on data mining.

Depending on the length of the instruction period, the background of students, and your interests, you may select subsets of chapters to teach in various sequential orderings. For example, an introductory course may cover the following chapters.

- Chapter 1: Introduction
- Chapter 2: Data, measurements, and data preprocessing
- Chapter 3: Data warehousing and online analytical processing
- Chapter 4: Pattern mining: basic concepts and methods
- Chapter 6: Classification: basic concepts
- Chapter 8: Cluster analysis: basic concepts and methods

If time permits, some materials about deep learning (Chapter 10) or outlier detection (Chapter 11) may be chosen. In each chapter, the fundamental concepts should be covered, while some sections on advanced topics can be treated optionally.

As another example, for a place where a machine learning course is offered to cover supervised learning well, a data mining course can discuss in depth on clustering. Such a course can be based on the following chapters.

- Chapter 1: Introduction
- Chapter 2: Data, measurements, and data preprocessing
- Chapter 3: Data warehousing and online analytical processing
- Chapter 4: Pattern mining: basic concepts and methods
- Chapter 8: Cluster analysis: basic concepts and methods
- Chapter 9: Cluster analysis: advanced methods
- Chapter 11: Outlier detection

An instructor teaching an advanced data mining course may find Chapter 12 particularly informative, since it discusses an extensive spectrum of new topics in data mining that are under dynamic and fast development.

Alternatively, you may choose to teach the whole book in a two-course sequence that covers all of the chapters in the book, plus, when time permits, some advanced topics such as graph and network mining. Material for such advanced topics may be selected from the companion chapters available from the book's web site, accompanied with a set of selected research papers.

Individual chapters in this book can also be used for tutorials or for special topics in related courses, such as machine learning, pattern recognition, data warehousing, and intelligent data analysis.

Each chapter ends with a set of exercises, suitable as assigned homework. The exercises are either short questions that test basic mastery of the material covered, longer questions that require analytical thinking, or implementation projects. Some exercises can also be used as research discussion topics. The bibliographic notes at the end of each chapter can be used to find the research literature that contains the origin of the concepts and methods presented, in-depth treatment of related topics, and possible extensions.

To the student

We hope that this textbook will spark your interest in the young yet fast-evolving field of data mining. We have attempted to present the material in a clear manner, with careful explanation of the topics covered. Each chapter ends with a summary describing the main points. We have included many figures and illustrations throughout the text to make the book more enjoyable and reader-friendly. Although this book was designed as a textbook, we have tried to organize it so that it will also be useful to you as a reference book or handbook, should you later decide to perform in-depth research in the related fields or pursue a career in data mining.

What do you need to know to read this book?

- You should have some knowledge of the concepts and terminology associated with statistics, database systems, and machine learning. However, we do try to provide enough background of the basics, so that if you are not so familiar with these fields or your memory is a bit rusty, you will not have trouble following the discussions in the book.
- You should have some programming experience. In particular, you should be able to read pseudocode and understand simple data structures such as multidimensional arrays and structures.

To the professional

This book was designed to cover a wide range of topics in the data mining field. As a result, it is an excellent handbook on the subject. Because each chapter is designed to be as standalone as possible, you can focus on the topics that most interest you. The book can be used by application programmers, data scientists, and information service managers who wish to learn about the key ideas of data mining on their own. The book would also be useful for technical data analysis staff in banking, insurance, medicine, and retailing industries who are interested in applying data mining solutions to their businesses. Moreover, the book may serve as a comprehensive survey of the data mining field, which may also benefit researchers who would like to advance the state-of-the-art in data mining and extend the scope of data mining applications.

The techniques and algorithms presented are of practical utility. Rather than selecting algorithms that perform well on small "toy" data sets, the algorithms described in the book are geared for the discovery of patterns and knowledge hidden in large, real data sets. Algorithms presented in the book are illustrated in pseudocode. The pseudocode is similar to the C programming language, yet is designed so that it should be easy to follow by programmers unfamiliar with C or C++. If you wish to implement any of the algorithms, you should find the translation of our pseudocode into the programming language of your choice to be a fairly straightforward task.

Book web site with resources

The book has a website with Elsevier at *https://educate.elsevier.com/book/details/9780128117606*. This website contains many supplemental materials for readers of the book or anyone else with an interest in data mining. The resources include the following:

- **Slide presentations for each chapter**. Lecture notes in Microsoft PowerPoint slides are available for each chapter.
- **Instructors' manual**. This complete set of answers to the exercises in the book is available only to instructors from the publisher's web site.
- **Figures from the book**. This may help you to make your own slides for your classroom teaching.
- **Table of Contents** of the book in PDF format.
- **Errata on the different printings of the book**. We encourage you to point out any errors in this book. Once the error is confirmed, we will update the errata list and include acknowledgment of your contribution.

Interested readers may also like to check **Authors' course teaching websites**. All the authors are university professors in their respective universities. Please check their corresponding data mining course websites which may contain their undergraduate and/or graduate course materials for introductory and/or advanced courses on data mining, including updated course/chapter slides, syllabi, homeworks, programming assignments, research projects, errata, and other related information.

Acknowledgments

Fourth edition of the book

We would like to express our sincere thanks to Micheline Kamber, the co-author of the previous editions of this book. Micheline has contributed substantially to these editions. Due to her commitment of other duties, she will not be able to join us in this new edition. We really appreciate her long-term collaborations and contributions in the past many years.

We would also like to express our grateful thanks to the previous and current members, including faculty and students, of the Data and Information Systems (DAIS) Laboratory, the Data Mining Group, IDEA Lab and iSAIL Lab at UIUC and the Data Mining Group at SFU, and many friends and colleagues, whose constant support and encouragement have made our work on this edition a rewarding experience. Our thanks extend to the students and TAs in the many data mining courses we taught at UIUC and SFU, as well as those in summer schools and beyond, who carefully went through the early drafts and the early editions of this book, identified many errors, and suggested various improvements.

We also wish to thank Steve Merken and Beth LoGiudice at Elsevier, for their enthusiasm, patience, and support during our writing of this edition of the book. We thank Gayathri S, the Project Manager, and her team members, for keeping us on schedule. We are also grateful for the invaluable feedback from all of the reviewers.

We would like to thank the US National Science Foundation (NSF), US Defense Advanced Research Projects Agency (DARPA), US Army Research Laboratory (ARL), US National Institute of Health (NIH), US Defense Threat Reduction Agency (DTRA), and Natural Science and Engineering Research Council of Canada (NSERC), as well as Microsoft Research, Google Research, IBM Research, Amazon, Adobe, LinkedIn, Yahoo!, HP Labs, PayPal, Facebook, Visa Research, and other industry research labs for their support of our research in the form of research grants, contracts, and gifts. Such research support deepens our understanding of the subjects discussed in this book.

Finally, we thank our families for their wholehearted support throughout this project.

This page intentionally left blank

About the authors

Jiawei Han is a Michael Aiken Chair Professor in the Department of Computer Science at the University of Illinois at Urbana-Champaign. He has received numerous awards for his contributions on research into knowledge discovery and data mining, including ACM SIGKDD Innovation Award (2004), IEEE Computer Society Technical Achievement Award (2005), and IEEE W. Wallace McDowell Award (2009). He is a Fellow of ACM and a Fellow of IEEE. He served as founding Editor-in-Chief of *ACM Transactions on Knowledge Discovery from Data* (2006–2011) and as an editorial board member of several journals, including *IEEE Transactions on Knowledge and Data Engineering* and *Data Mining and Knowledge Discovery*.

Jian Pei is currently Professor of Computer Science, Biostatistics and Bioinformatics, and Electrical and Computer Engineering at Duke University. He received a Ph.D. degree in computing science from Simon Fraser University in 2002 under Dr. Jiawei Han's supervision. He has published prolifically in the premier academic forums on data mining, databases, Web searching, and information retrieval and actively served the academic community. He is a fellow of the Royal Society of Canada, the Canadian Academy of Engineering, ACM, and IEEE. He received the 2017 ACM SIGKDD Innovation Award and the 2015 ACM SIGKDD Service Award.

Hanghang Tong is currently an associate professor at Department of Computer Science at University of Illinois at Urbana-Champaign. He received his Ph.D. degree from Carnegie Mellon University in 2009. He has published over 200 refereed articles. His research is recognized by several prestigious awards and thousands of citations. He is the Editor-in-Chief of SIGKDD Explorations (ACM) and an associate editor of several journals.

This page intentionally left blank

CHAPTER

1 Introduction

This book is an introduction to the young and fast-growing field of *data mining* (also known as *knowledge discovery from data*, or *KDD* for short). The book focuses on fundamental data mining concepts and techniques for discovering interesting patterns from data in various applications. In particular, we emphasize prominent techniques for developing effective, efficient, and scalable data mining tools.

This chapter is organized as follows. In Section 1.1, we learn what is data mining and why data mining is in high demand. Section 1.2 links data mining with the overall knowledge discovery process. Next, we learn about data mining from multiple aspects, such as the kinds of data that can be mined (Section 1.3), the kinds of knowledge to be mined (Section 1.4), the relationship between data mining and other disciplines (Section 1.5), and data mining applications (Section 1.6). Finally, we discuss the impact of data mining to society (Section 1.7).

1.1 What is data mining?

Necessity, who is the mother of invention.
– Plato

We live in a world where vast amounts of data are generated constantly and rapidly.

"*We are living in the information age*" is a popular saying; however, *we are actually living in the data age*. Terabytes or petabytes of data pour into our computer networks, the World Wide Web (WWW), and various kinds of devices every day from business, news agencies, society, science, engineering, medicine, and almost every other aspect of daily life. This explosive growth of available data volume is a result of the computerization of our society and the fast development of powerful computing, sensing, and data collection, storage, and publication tools.

Businesses worldwide generate gigantic data sets, including sales transactions, stock trading records, product descriptions, sales promotions, company profiles and performance, and customer feedback. Scientific and engineering practices generate high orders of petabytes of data in a continuous manner, from remote sensing, to process measuring, scientific experiments, system performance, engineering observations, and environment surveillance. Biomedical research and the health industry generate tremendous amounts of data from gene sequence machines, biomedical experiment and research reports, medical records, patient monitoring, and medical imaging. Billions of Web searches supported by search engines process tens of petabytes of data daily. Social media tools have become increasingly popular, producing a tremendous number of texts, pictures, and videos, generating various kinds of Web communities and social networks. The list of sources that generate huge amounts of data is endless.

Data Mining. https://doi.org/10.1016/B978-0-12-811760-6.00011-4

This explosively growing, widely available, and gigantic body of data makes our time truly *the data age*. Powerful and versatile tools are badly needed to automatically uncover valuable information from the tremendous amounts of data and to transform such data into organized knowledge. This necessity has led to the birth of data mining.

Essentially, ***data mining*** *is the process of discovering interesting patterns, models, and other kinds of knowledge in large data sets*. The term, *data mining*, as a vivid view of searching for *gold nuggets* from data, appeared in 1990s. However, to refer to the mining of gold from rocks or sand, we say *gold mining* instead of rock or sand mining. Analogously, data mining should have been more appropriately named "knowledge mining from data," which is unfortunately somewhat long. However, the shorter term, *knowledge mining* may not reflect the emphasis on mining from large amounts of data. Nevertheless, mining is a vivid term characterizing the process that finds a small set of precious nuggets from a great deal of raw material. Thus, such a misnomer carrying both "data" and "mining" became a popular choice. In addition, many other terms have a similar meaning to data mining—for example, *knowledge mining from data*, *KDD (i.e., Knowledge Discovery from Data)*, *pattern discovery*, *knowledge extraction*, *data archaeology*, *data analytics*, and *information harvesting*.

Data mining is a young, dynamic, and promising field. It has made and will continue to make great strides in our journey from the data age toward the coming information age.

Example 1.1. Data mining turns a large collection of data into knowledge. A search engine (e.g., Google) receives billions of queries every day. What novel and useful knowledge can a search engine learn from such a huge collection of queries collected from users over time? Interestingly, some patterns found in user search queries can disclose invaluable knowledge that cannot be obtained by reading individual data items alone. For example, Google's *Flu Trends* uses specific search terms as indicators of flu activity. It found a close relationship between the number of people who search for flu-related information and the number of people who actually have flu symptoms. A pattern emerges when all of the search queries related to flu are aggregated. Using aggregated Google search data, *Flu Trends* can estimate flu activity up to two weeks faster than what traditional systems can.[1] This example shows how data mining can turn a large collection of data into knowledge that can help meet a current global challenge. □

1.2 Data mining: an essential step in knowledge discovery

Many people treat data mining as a synonym for another popularly used term, **knowledge discovery from data**, or **KDD**, whereas others view data mining as merely an essential step in the overall process of knowledge discovery. The overall knowledge discovery process is shown in Fig. 1.1 as an iterative sequence of the following steps:

1. **Data preparation**
   a. **Data cleaning** (to remove noise and inconsistent data)
   b. **Data integration** (where multiple data sources may be combined)[2]

[1] This is reported in [GMP+09]. The *Flu Trend* reporting stopped in 2015.

[2] A popular trend in the information industry is to perform data cleaning and data integration as a preprocessing step, where the resulting data are stored in a data warehouse.

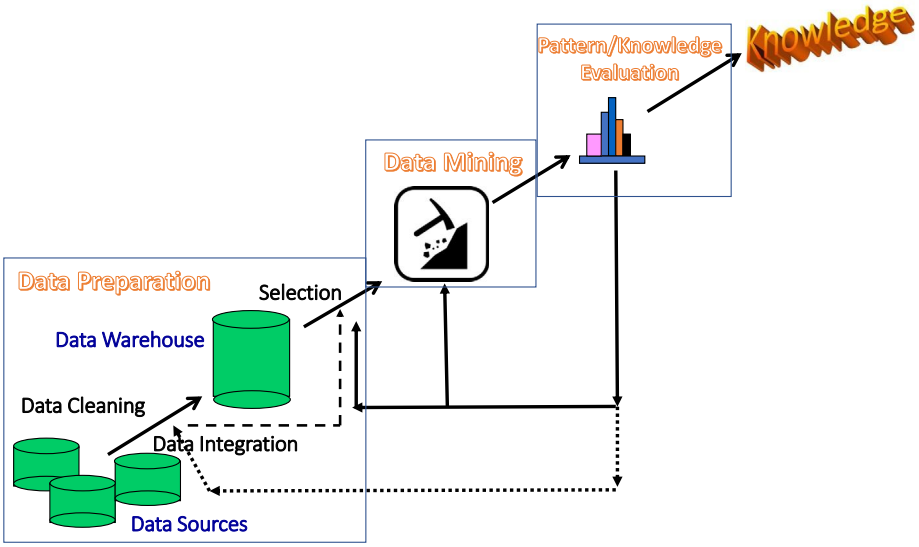


FIGURE 1.1

Data mining: An essential step in the process of knowledge discovery.

c. **Data transformation** (where data are transformed and consolidated into forms appropriate for mining by performing summary or aggregation operations)[3]
d. **Data selection** (where data relevant to the analysis task are retrieved from the database)

2. **Data mining** (an essential process where intelligent methods are applied to extract patterns or construct models)
3. **Pattern/model evaluation** (to identify the truly interesting patterns or models representing knowledge based on *interestingness measures*—see Section 1.4.7)
4. **Knowledge presentation** (where visualization and knowledge representation techniques are used to present mined knowledge to users)

Steps 1(a) through 1(d) are different forms of data preprocessing, where data are prepared for mining. The data mining step may interact with a user or a knowledge base. The interesting patterns are presented to the user and may be stored as new knowledge in the knowledge base.

The preceding view shows data mining as one step in the knowledge discovery process, albeit an essential one because it uncovers hidden patterns or models for evaluation. However, in industry, in media, and in the research milieu, the term *data mining* is often used to refer to the entire knowledge discovery process (perhaps because the term is shorter than *knowledge discovery from data*). Therefore, we adopt a broad view of data mining functionality: *Data mining is the* process *of discovering inter-*

[3] Data transformation and consolidation are often performed before the data selection process, particularly in the case of data warehousing. *Data reduction* may also be performed to obtain a smaller representation of the original data without sacrificing its integrity.

esting patterns and knowledge from large *amounts of data*. The data sources can include databases, data warehouses, the Web, other information repositories, or data that are streamed into the system dynamically.

1.3 Diversity of data types for data mining

As a general technology, data mining can be applied to any kind of data as long as the data are meaningful for a target application. However, different kinds of data may need rather different data mining methodologies, from simple to rather sophisticated, making data mining a rich and diverse field.

Structured vs. unstructured data

Based on whether data have clear structures, we can categorize data as *structured* vs. *unstructured data*.

Data stored in *relational databases*, *data cubes*, *data matrices*, and many *data warehouses* have uniform, record- or table-like structures, defined by their data dictionaries, with a fixed set of attributes (or fields, columns), each with a fixed set of value ranges and semantic meaning. These data sets are typical examples of highly structured data. In many real applications, such strict structural requirement can be relaxed in multiple ways to accommodate *semistructured* nature of the data, such as to allow a data object to contain a set value, a small set of heterogeneous typed values, or nested structures or to allow the structure of objects or subobjects to be defined flexibly and dynamically (e.g., XML structures).

There are many data sets that may not be as structured as relational tables or data matrices. However, they do have certain structures with clearly defined semantic meaning. For example, a *transactional data set* may contain a large set of transactions each containing a set of items. A *sequence data set* may contain a large set of sequences each containing an ordered set of elements that can in turn contain a set of items. Many application data sets, such as shopping transaction data, time-series data, gene or protein data, or Weblog data, belong to this category.

A more sophisticated type of semistructured data set is *graph or network data*, where a set of nodes are connected by a set of edges (also called links); and each node/link may have its own semantic description or substructures.

Each of such categories of structured and semistructured data sets may have special kinds of patterns or knowledge to be mined and many dedicated data mining methods, such as sequential pattern mining, graph pattern mining, and information network mining methods, have been developed to analyze such data sets.

Beyond such structured or semistructured data, there are also large amounts of unstructured data, such as text data and multimedia (e.g., audio, image, video) data. Although some studies treat them as one-dimensional or multidimensional byte streams, they do carry a lot of interesting semantics. Domain-specific methods have been developed to analyze such data in the fields of natural language understanding, text mining, computer vision, and pattern recognition. Moreover, recent advances on deep learning have made tremendous progress on processing text, image, and video data. Nevertheless, mining hidden structures from unstructured data may greatly help understand and make good use of such data.

The real-world data can often be a mixture of structured data, semistructured data, and unstructured data. For example, an online shopping website may host information for a large set of products, which

can be essentially structured data stored in a relational database, with a fixed set of fields on product name, price, specifications, and so on. However, some fields may essentially be text, image, and video data, such as product introduction, expert or user reviews, product images, and advertisement videos. Data mining methods are often developed for mining some particular type of data, and their results can be integrated and coordinated to serve the overall goal.

Data associated with different applications

Different applications may generate or need to handle very different data sets and require rather different data analysis methods. Thus when categorizing data sets for data mining, we should take specific applications into consideration.

Take sequence data as an example. *Biological sequences* such as DNA or protein sequences may have very different semantic meaning from *shopping transaction sequences* or *Web click streams*, calling for rather different sequence mining methods. A special kind of sequence data is time-series data where a *time-series* may contain an ordered set of numerical values with equal time interval, which is also rather different from shopping transaction sequences, which may not have fixed time gaps (a customer may shop at anytime she likes).

Data in some applications can be associated with spatial information, time information, or both, forming *spatial, temporal, and spatiotemporal data*, respectively. Special data mining methods, such as spatial data mining, temporal data mining, spatiotemporal data mining, or trajectory pattern mining, should be developed for mining such data sets as well.

For graph and network data, different applications may also need rather different data mining methods. For example, social networks (e.g., Facebook or LinkedIn data), computer communication networks, biological networks, and information networks (e.g., authors linking with keywords) may carry rather different semantics and require different mining methods.

Even for the same data set, finding different kinds of patterns or knowledge may require different data mining methods. For example, for the same set of software (source) programs, finding plagiarized subprogram modules or finding copy-and-paste bugs may need rather different data mining techniques.

Rich data types and diverse application requirements call for very diverse data mining methods. Thus data mining is a rich and fascinating research domain, with lots of new methods waiting to be studied and developed.

Stored vs. streaming data

Usually, data mining handles finite, stored data sets, such as those stored in various kinds of large data repositories. However, in some applications such as video surveillance or remote sensing, data may stream in dynamically and constantly, as infinite *data streams*. Mining stream data will require rather different methods than stored data, which may form another interesting theme in our study.

1.4 Mining various kinds of knowledge

Different kinds of patterns and knowledge can be uncovered via data mining. In general, data mining tasks can be put into two categories: **descriptive data mining** and **predictive data mining**. Descriptive mining characterizes properties of the interested set of data, whereas predictive mining performs induction on the data set in order to make predictions.

In this section, we introduce different data mining tasks. These include multidimensional data summarization (Section 1.4.1); the mining of frequent patterns, associations, and correlations (Section 1.4.2); classification and regression (Section 1.4.3); cluster analysis (Section 1.4.4); and outlier analysis (Section 1.4.6). Different data mining functionalities generate different kinds of results that are often called patterns, models, or knowledge. In Section 1.4.7, we will also introduce the interestingness of a pattern or a model. In many cases, only interesting patterns or models will be considered as *knowledge*.

1.4.1 Multidimensional data summarization

It is often tedious for a user to go over the details of a large set of data. Thus it is desirable to automatically summarize an interested set of data and compare it with the contrasting sets at some high levels. Such summaritive description of an *interested set of data* is called **data summarization**. Data summarization can often be conducted in a multidimensional space. If the multidimensional space is well defined and frequently used, such as product category, producer, location, or time, massive amounts of data can be aggregated in the form of **data cubes** to facilitate user's drill-down or roll-up of the summarization space with mouse clicking. The output of such multidimensional summarization can be presented in various forms, such as **pie charts**, **bar charts**, **curves**, **multidimensional data cubes**, and **multidimensional tables**, including crosstabs.

For structured data, multidimensional aggregation methods have been developed to facilitate such precomputation or online computation of multidimensional aggregations using data cube technology, which will be discussed in Chapter 3. For unstructured data, such as text, this task becomes challenging. We will give a brief discussion of such research frontiers in our last chapter.

1.4.2 Mining frequent patterns, associations, and correlations

Frequent patterns, as the name suggests, are patterns that occur frequently in data. There are many kinds of frequent patterns, including frequent itemsets, frequent subsequences (also known as sequential patterns), and frequent substructures. A *frequent itemset* typically refers to a set of items that often appear together in a transactional data set—for example, milk and bread, which are frequently bought together in grocery stores by many customers. A frequently occurring subsequence, such as the pattern that customers, tend to purchase first a laptop, followed by a computer bag, and then other accessories, is a (*frequent*) *sequential pattern*. A substructure can refer to different structural forms (e.g., graphs, trees, or lattices) that may be combined with itemsets or subsequences. If a substructure occurs frequently, it is called a (*frequent*) *structured pattern*. Mining frequent patterns leads to the discovery of interesting associations and correlations within data.

Example 1.2. Association analysis. Suppose that, a webstore manager wants to know which items are frequently purchased together (i.e., in the same transaction). An example of such a rule, mined from the transactional database, is

$$buys(X, \text{“}computer\text{”}) \Rightarrow buys(X, \text{“}webcam\text{”}) \quad [support = 1\%, confidence = 50\%],$$

where X is a variable representing a customer. A **confidence**, or certainty, of 50% means that if a customer buys a computer, there is a 50% chance that she will buy webcam as well. A 1% **support** means

that 1% of all the transactions under analysis show that computer and webcam are purchased together. This association rule involves a single attribute or predicate (i.e., *buys*) that repeats. Association rules that contain a single predicate are referred to as **single-dimensional association rules**. Dropping the predicate notation, the rule can be written simply as "*computer* $\Rightarrow$ *webcam* [1%, 50%]."

Suppose, mining the same database generates another association rule:

$$age(X, \text{“20..29”}) \wedge income(X, \text{“40K..49K”}) \Rightarrow buys(X, \text{“laptop”})$$
$$[support = 0.5\%, confidence = 60\%].$$

The rule indicates that of all its customers under study, 0.5% are 20 to 29 years old with an income of $40,000 to $49,000 and have purchased a laptop (computer). There is a 60% probability that a customer in this age and income group will purchase a laptop. Note that this is an association involving more than one attribute or predicate (i.e., *age, income*, and *buys*). Adopting the terminology used in multidimensional databases, where each attribute is referred to as a dimension, the above rule can be referred to as a **multidimensional association rule**. □

Typically, association rules are discarded as uninteresting if they do not satisfy both a **minimum support threshold** and a **minimum confidence threshold**. Additional analysis can be performed to uncover interesting statistical **correlations** between associated attribute–value pairs.

Frequent itemset mining is a fundamental form of frequent pattern mining. Mining frequent itemsets, associations, and correlations will be discussed in Chapter 4. Mining diverse kinds of frequent pattern, as well as mining sequential patterns and structured patterns, will be covered in Chapter 5.

1.4.3 Classification and regression for predictive analysis

Classification is the process of finding a **model** (or function) that describes and distinguishes data classes or concepts. The model is derived based on the analysis of a set of **training data** (i.e., data objects for which the class labels are known). The model is used to predict the class labels of objects for which the class labels are unknown.

Depending on the classification methods, a derived model can be in various forms, such as a set of *classification rules* (i.e., *IF-THEN rules*), a *decision tree*, a *mathematical formula*, or a learned *neural network* (Fig. 1.2). A **decision tree** is a flowchart-like tree structure, where each node denotes a test on an attribute value, each branch represents an outcome of the test, and tree leaves represent classes or class distributions. Decision trees can easily be converted to classification rules. A **neural network**, when used for classification, is typically a collection of neuron-like processing units with weighted connections between the units. There are many other methods for constructing classification models, such as naïve Bayesian classification, support vector machines, and k-nearest-neighbor classification.

Whereas classification predicts categorical (discrete, unordered) labels, **regression** models continuous-valued functions. That is, regression is used to predict missing or unavailable *numerical data values* rather than (discrete) class labels. The term *prediction* refers to both numeric prediction and class label prediction. **Regression analysis** is a statistical methodology that is most often used for numeric prediction, although other methods exist as well. Regression also encompasses the identification of distribution *trends* based on the available data.

Classification and regression may need to be preceded by **feature selection** or **relevance analysis**, which attempts to identify attributes (often called *features*) that are significantly relevant to the clas-

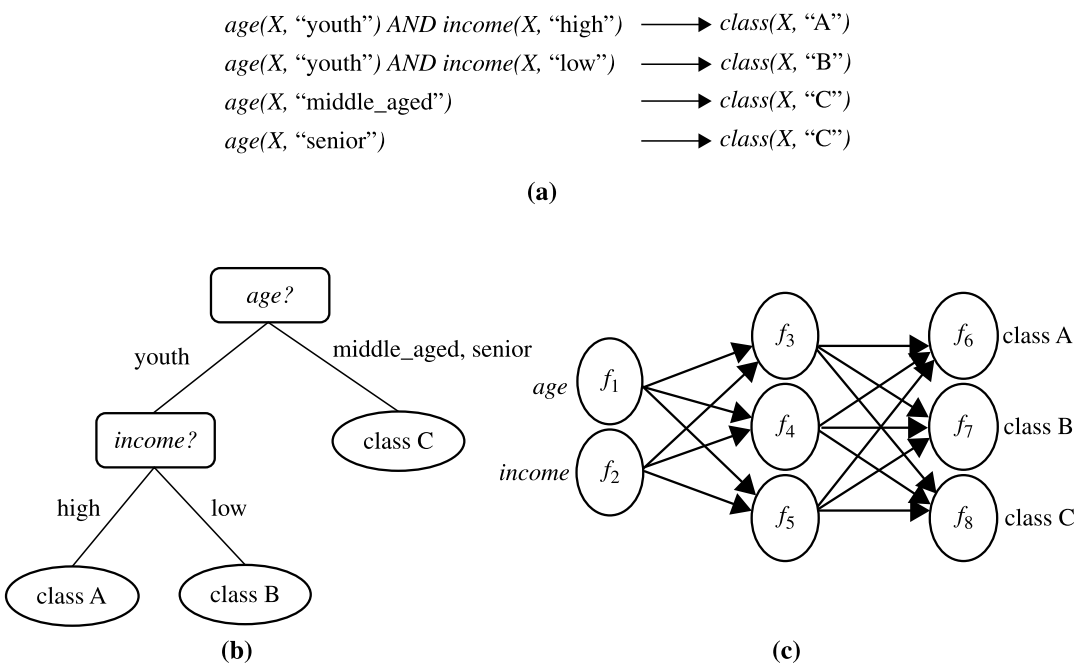


FIGURE 1.2

A classification model can be represented in various forms: (a) IF-THEN rules, (b) a decision tree, or (c) a neural network.

sification and regression process. Such attributes will be selected for the classification and regression process. Other attributes, which are irrelevant, can then be excluded from consideration.

Example 1.3. Classification and regression. Suppose a webstore sales manager wants to classify a large set of items in the store, based on three kinds of responses to a sales campaign: *good response*, *mild response*, and *no response*. You want to derive a model for each of these three classes based on the descriptive features of the items, such as *price*, *brand*, *place_made*, *type*, and *category*. The resulting classification should maximally distinguish each class from the others, presenting an organized picture of the data set.

Suppose that the resulting classification is expressed as a decision tree. The decision tree, for instance, may identify *price* as being the first important factor that best distinguishes the three classes. Other features that help further distinguish objects of each class from one another include *brand* and *place_made*. Such a decision tree may help the manager understand the impact of the given sales campaign and design a more effective campaign in the future.

Suppose instead, that rather than predicting categorical response labels for each store item, you would like to predict the amount of revenue that each item will generate during an upcoming sale, based on the previous sales data. This is an example of regression analysis because the regression model constructed will predict a continuous function (or ordered value.) □

Chapters 6 and 7 discuss classification in further detail. Regression analysis is covered lightly in these chapters since it is typically introduced in statistics courses. Sources for further information are given in the bibliographic notes.

1.4.4 Cluster analysis

Unlike classification and regression, which analyze class-labeled (training) data sets, **cluster analysis** (also called **clustering**) groups data objects without consulting class labels. In many cases, class-labeled data may simply not exist at the beginning. Clustering can be used to generate class labels for a group of data. The objects are clustered or grouped based on the principle of *maximizing the intraclass similarity and minimizing the interclass similarity*. That is, clusters of objects are formed so that objects within a cluster have high similarity in comparison to one another, but are rather dissimilar to objects in other clusters. Each cluster so formed can be viewed as a class of objects, from which rules can be derived. Clustering can also facilitate **taxonomy formation**, that is, the organization of observations into a hierarchy of classes that group similar events together.

Example 1.4. Cluster analysis. Cluster analysis can be performed on the webstore customer data to identify homogeneous subpopulations of customers. These clusters may represent individual target groups for marketing. Fig. 1.3 shows a 2-D plot of customers with respect to customer locations in a city. Three clusters of data points are evident. □

Cluster analysis forms the topic of Chapters 8 and 9.

1.4.5 Deep learning

For many data mining tasks, such as classification and clustering, a key step often lies in finding "good features," which is a vector representation of each input data tuple. For example, in order to predict

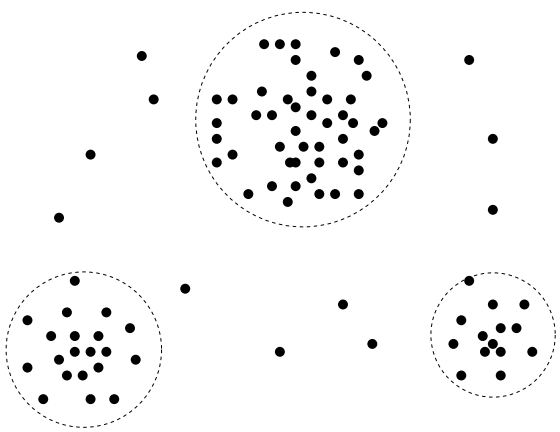


FIGURE 1.3

A 2-D plot of customer data with respect to customer locations in a city, showing three data clusters.

whether a regional disease outbreak will occur, one might have collected a large number of features from the health surveillance data, including the number of daily positive cases, number of daily tests, number of daily hospitalization, etc. Traditionally, this step (called feature engineering) often heavily relies on domain knowledge. Deep learning techniques provide an automatic way for feature engineering, which is capable of generating semantically meaningful features (e.g., weekly positive rate) from the initial input features. The generated features often significantly improve the mining performance (e.g., classification accuracy).

Deep learning is based on *neural networks*. A neural network is a set of connected input-output units where each connection has a weight associated with it. During the learning phase, the network learns by adjusting the weights to be able to predict the correct target values (e.g., class labels) of the input tuples. The core algorithm to learn such weights is called *backpropagation*, which searches for a set of weights and bias values that can model the data to minimize the loss function between the network's prediction and the actual target output of data tuples. Various forms (called architectures) of neural networks have been developed, including feed-forward neural networks, convolutional neural networks, recurrent neural networks, graph neural networks, and many more.

Deep learning has broad applications in computer vision, natural language processing, machine translation, social network analysis, and so on. It has been used in a variety of data mining tasks, including classification, clustering, outlier detection, and reinforcement learning.

Deep learning is the topic of Chapter 10.

1.4.6 Outlier analysis

A data set may contain objects that do not comply with the general behavior or model of the data. These data objects are **outliers**. Many data mining methods discard outliers as noise or exceptions. However, in some applications (e.g., fraud detection) the rare events can be more interesting than the more regularly occurring ones. The analysis of outlier data is referred to as **outlier analysis** or **anomaly mining**.

Outliers may be detected using statistical tests that assume a distribution or probability model for the data, or using distance measures where objects that are remote from any other cluster are considered outliers. Rather than using statistical or distance measures, density-based methods may identify outliers in a local region, although they look normal from a global statistical distribution view.

Example 1.5. Outlier analysis. Outlier analysis may uncover fraudulent usage of credit cards by detecting purchases of unusually large amounts for a given account number in comparison to regular charges incurred by the same account. Outlier values may also be detected with respect to the locations and types of purchase, or the purchase frequency. □

Outlier analysis is discussed in Chapter 11.

1.4.7 Are all mining results interesting?

Data mining has the potential to generate a lot of results. A question can be, *"Are all of the mining results interesting?"*

This is a great question. Each type of data mining functions has its own measures on the evaluation of the mining quality. Nevertheless, there are some shared philosophy and principles.

Take pattern mining as an example. Pattern mining may generate thousands or even millions of patterns, or rules. You may wonder, *"What makes a pattern interesting? Can a data mining system generate all of the interesting patterns? Or, can the system generate only the interesting ones?"*

To answer the first question, a pattern is **interesting** if it is (1) *easily understood* by humans, (2) *valid* on new or test data with some degree of *certainty*, (3) potentially *useful*, and (4) *novel*. A pattern is also interesting if it validates a hypothesis that the user *sought to confirm*.

Several **objective measures of pattern interestingness** exist. These are based on the structure of discovered patterns and the statistics underlying them. An objective measure for association rules of the form $X \Rightarrow Y$ is rule **support**, representing the percentage of transactions from a transaction database that the given rule satisfies. This is taken to be the probability $P(X \cup Y)$, where $X \cup Y$ indicates that a transaction contains both X and Y, that is, the union of itemsets X and Y. Another objective measure for association rules is **confidence**, which assesses the degree of certainty of the detected association. This is taken to be the conditional probability $P(Y|X)$, that is, the probability that a transaction containing X also contains Y. More formally, support and confidence are defined as

$$support(X \Rightarrow Y) = P(X \cup Y),$$
$$confidence(X \Rightarrow Y) = P(Y|X).$$

In general, each interestingness measure is associated with a threshold, which may be controlled by the user. For example, rules that do not satisfy a confidence threshold of, say, 50% can be considered uninteresting. Rules below the threshold likely reflect noise, exceptions, or minority cases and are probably of less value.

There are also other objective measures. For example, one may like set of items to be strongly correlated in an association rule. We will discuss such measures in the corresponding chapter.

Although objective measures help identify interesting patterns, they are often insufficient unless combined with subjective measures that reflect a particular user's needs and interests. For example, patterns describing the characteristics of customers who shop frequently online should be interesting to the marketing manager, but may be of little interest to other analysts studying the same database for patterns on employee performance. Furthermore, many patterns that are interesting by objective standards may represent common sense and, therefore, are actually uninteresting.

Subjective interestingness measures are based on user beliefs in the data. These measures find patterns interesting if the patterns are **unexpected** (contradicting a user's belief) or offer strategic information on which the user can act. In the latter case, such patterns are referred to as **actionable**. For example, patterns like "a large earthquake often follows a cluster of small quakes" may be highly actionable if users can act on the information to save lives. Patterns that are **expected** can be interesting if they confirm a hypothesis that the user wishes to validate or they resemble a user's hunch.

The second question—"*Can a data mining system generate* all *of the interesting patterns?*"—refers to the **completeness** of a data mining algorithm. It is often unrealistic and inefficient for a pattern mining system to generate all possible patterns since there could be a very large number of them. However, one may also worry whether one may miss some important ones if the system stops short. To solve this dilemma, user-provided constraints and interestingness measures should be used to focus the search. With well-defined interesting measures and user-provided constraints, it is quite realistic to ensure the completeness of pattern mining. The methods involved are examined in detail in Chapter 4.

Finally, the third question—*"Can a data mining system generate only interesting patterns?"*—is an optimization problem in data mining. It is highly desirable for a data mining system to generate only

interesting patterns. This would be efficient for both the data mining system and the user because the system may spend much less time to generate much fewer but interesting patterns, whereas the user will not need to sift through a large number of patterns to identify the truly interesting ones. Constraint-based pattern mining described in Chapter 5 is a good example in this direction.

Methods to assess the quality or interestingness of data mining results, and how to use them to improve data mining efficiency, are discussed throughout the book.

1.5 Data mining: confluence of multiple disciplines

As a discipline that studies efficient and effective methods for uncovering patterns and knowledge from various kinds of massive data sets for many applications, data mining naturally serves a confluence of multiple disciplines including machine learning, statistics, pattern recognition, natural language processing, database technology, visualization and human computer interaction (HCI), algorithms, high-performance computing, social sciences, and many application domains (Fig. 1.4). The interdisciplinary nature of data mining research and development contributes significantly to the success of data mining and its extensive applications. On the other hand, data mining is not only nurtured from the knowledge and development of these disciplines, the dedicated research, development, and applications of data mining on various kinds of big data may have substantially impacted the development of these disciplines in recent years as well. In this section, we discuss several disciplines that strongly impact and actively interact with the research, development, and applications of data mining.

1.5.1 Statistics and data mining

Statistics studies the collection, analysis, interpretation or explanation, and presentation of data. Data mining has an inherent connection with statistics.

A **statistical model** is a set of mathematical functions that describe the behavior of the objects in a target class in terms of random variables and their associated probability distributions. Statistical

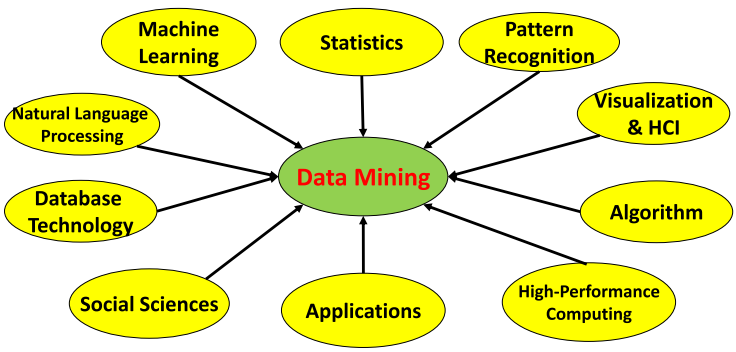


FIGURE 1.4

Data mining: Confluence of multiple disciplines.

models are widely used to model data and data classes. For example, in data mining tasks such as data characterization and classification, statistical models of target classes can be built. In other words, such statistical models can be the outcome of a data mining task. Alternatively, data mining tasks can be built on top of statistical models. For example, we can use statistics to model noise and missing data values. Then, when mining patterns in a large data set, the data mining process can use the model to help identify and handle noisy or missing values in the data.

Statistics research develops tools for prediction and forecasting using data and statistical models. Statistical methods can be used to summarize or describe a collection of data. Basic **statistical descriptions** of data are introduced in Chapter 2. Statistics is useful for mining various patterns from data and for understanding the underlying mechanisms generating and affecting the patterns. **Inferential statistics** (or **predictive statistics**) models data in a way that accounts for randomness and uncertainty in the observations and is used to draw inferences about the process or population under investigation.

Statistical methods can also be used to verify data mining results. For example, after a classification or prediction model is mined, the model should be verified by statistical hypothesis testing. A **statistical hypothesis test** (sometimes called *confirmatory data analysis*) makes statistical decisions using experimental data. A result is called *statistically significant* if it is unlikely to have occurred by chance. If the classification or prediction model holds, then the descriptive statistics of the model increases the soundness of the model.

Applying statistical methods in data mining is far from trivial. Often, a serious challenge is how to scale up a statistical method over a large data set. Many statistical methods have high complexity in computation. When such methods are applied on large data sets that are also distributed on multiple logical or physical sites, algorithms should be carefully designed and tuned to reduce the computational cost. This challenge becomes even tougher for online applications, such as online query suggestions in search engines, where data mining is required to continuously handle fast, real-time data streams.

Data mining research has developed many scalable and effective solutions for the analysis of massive data sets and data streams. Moreover, different kinds of data sets and different applications may require rather different analysis methods. Effective solutions have been proposed and tested, which leads to many new, scalable data mining-based statistical analysis methods.

1.5.2 Machine learning and data mining

Machine learning investigates how computers can learn (or improve their performance) based on data. Machine learning is a fast-growing discipline, with many new methodologies and applications developed in recent years, from support vector machines to probabilistic graphical models and deep learning, which we will cover in this book.

In general, machine learning addresses two classical problems: *supervised learning* and *unsupervised learning*.

- **Supervised learning**: A classic example of supervised learning is classification. The supervision in the learning comes from the labeled examples in the training data set. For example, to automatically recognize handwritten postal codes on mails, the learning system takes a set of handwritten postal code images and their corresponding machine-readable translations as the training examples, and learns (i.e., computes) a classification model.
- **Unsupervised learning**: A classic example of unsupervised learning is clustering. The learning process is unsupervised since the input examples are not class-labeled. Typically, we may use clustering

to discover groups within the data. For example, an unsupervised learning method can take, as input, a set of images of handwritten digits. Suppose that it finds 10 clusters of data. These clusters may hopefully correspond to the 10 distinct digits of 0 to 9, respectively. However, since the training data are not labeled, the learned model cannot tell us the semantic meaning of the clusters found.

As to these two basic problems, data mining and machine learning do share many similarities. However, data mining differs from machine learning in several major aspects. First, even on similar tasks like classification and clustering, data mining often works on very large data sets, or even on infinite data streams, scalability can be an important concern, and many efficient and highly scalable data mining algorithms or stream mining algorithms have to be developed to accomplish such tasks.

Second, in many data mining problems, the data sets are usually large, but the training data can still be rather small since it is expensive for experts to provide quality labels for many examples. Therefore, data mining has to put a lot of effort on developing *weakly supervised methods*. These include methodologies like *semisupervised learning* with a small set of labeled data but a large set of unlabeled data (with the idea sketched in Fig. 1.5), *integration or ensemble of multiple weak models* obtained from nonexperts (e.g., those obtained by crowd-sourcing), *distant supervision*, such as using popularly available and general (but distantly relevant to the problem to be solved) knowledge-bases (e.g., wikipedia, DBPedia), *actively learning* by carefully selecting examples to ask human experts, or *transfer learning* by integrating models learned from similar problem domains. Data mining has been extending such weakly supervised methods for constructing quality classification models on large data sets with a very limited set of high quality training data.

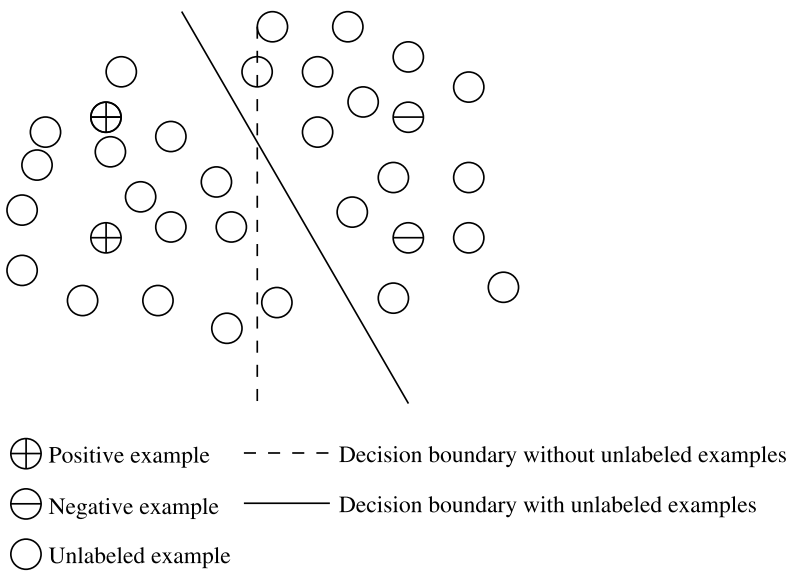


FIGURE 1.5

Semisupervised learning.

Third, machine learning methods may not be able to handle many kinds of knowledge discovery problems on big data. On the other hand, data mining, developing effective solutions for concrete application problems, goes deep in the problem domain, and expands far beyond the scope covered by machine learning. For example, many application problems, such as business transaction data analysis, software program execution sequence analysis, and chemical and biological structural analysis, need effective methods for mining frequent patterns, sequential patterns, and structured patterns. Data mining research has generated many scalable, effective, and diverse mining methods for such tasks. As another example, the analysis of large-scale social and information networks poses many challenging problems that may not fit the typical scope of many machine learning methods due to the information interaction across links and nodes in such networks. Data mining has developed a lot of interesting solutions to such problems.

From this point of view, data mining and machine learning are two different but closely related disciplines. Data mining dives deep into concrete, data-intensive application domains, does not confine itself to a single problem-solving methodology, and develops concrete (sometimes rather novel), effective and scalable solutions for many challenging application problems. It is a young, broad, and promising research discipline for many researchers and practitioners to study and work on.

1.5.3 Database technology and data mining

Database system research focuses on the creation, maintenance, and use of databases for organizations and end-users. Particularly, database system researchers have established well-recognized principles in data models, query languages, query processing and optimization, data storage, and indexing methods. Database technology is well known for its scalability in processing very large, relatively structured data sets.

Many data mining tasks need to handle large data sets or even real-time, fast streaming data. Data mining can make good use of scalable database technologies to achieve high efficiency and scalability on large data sets. Moreover, data mining tasks can be used to extend the capability of existing database systems to satisfy users' sophisticated data analysis requirements.

Recent database systems have built systematic data analysis capabilities on database data using data warehousing and data mining facilities. A **data warehouse** integrates data originated from multiple sources and various timeframes. It consolidates data in multidimensional space to form partially materialized data cubes. The data cube model not only facilitates online analytical processing (OLAP) in multidimensional databases but also promotes *multidimensional data mining*, which will be further discussed in future chapters.

1.5.4 Data mining and data science

With the tremendous amount of data in almost every discipline and various kinds of applications, big data and data science have become buzzwords in recent years. **Big data** generally refers to huge amounts of structured and unstructured data of various forms, and **data science** is an interdisciplinary field that uses scientific methods, processes, algorithms and systems to extract knowledge and insights from massive data of various forms. Clearly, data mining plays an essential role in data science.

For most people, data science is a concept that unifies statistics, machine learning, data mining, and their related methods in order to understand and analyze massive data. It employs techniques and theories drawn from many fields within the context of mathematics, statistics, information science, and

computer science. For many industry people, the term "data science" often refers to business analytics, business intelligence, predictive modeling, or any meaningful use of data, and is being taken as a glamorized term to re-brand statistics, data mining, machine learning, or any kind of data analytics. So far, there exists no consensus on a definition or suitable curriculum contents in data science degree programs of many universities. Nonetheless, most universities take basic knowledge generated in statistics, machine learning, data mining, database, and human computer interaction as the core curriculum in data science education.

In 1990s, the late Turing award winner Jim Gray envisioned data science as the "fourth paradigm" of science (i.e., from empirical to theoretical, computational, and now data-driven) and asserted that "everything about science is changing because of the impact of information technology" and the emergence of massive data. So there is no wonder that data science, big data, and data mining are closely interrelated and represent an inevitable trend in science and technology developments.

1.5.5 Data mining and other disciplines

Besides statistics, machine learning, and database technology, data mining has close relationships with many other disciplines as well.

The majority of the real-world data are unstructured, in the form of natural language text, images, or audio-video data. Therefore, natural language processing, computer vision, pattern recognition, audio-video signal processing, and information retrieval will offer critical help at handling such data. Actually, handling any special kinds of data will need a lot of domain knowledge to be integrated into the data mining algorithm design. For example, mining biomedical data will need the integration of knowledge from biological sciences, medical sciences, and bioinformatics. Mining geospatial data will need much knowledge and techniques from geography and geospatial data sciences. Mining software bugs in large software programs will need to integrate software engineering with data mining. Mining social media and social networks will need knowledge and skills from social sciences and network sciences. Such examples can go on and on since data mining will penetrate almost every application domain.

One major challenge in data mining is efficiency and scalability since we often have to handle huge amounts of data with critical time and resource constraints. Data mining is critically connected with efficient algorithm design such as low-complexity, incremental, and streaming data mining algorithms. It often needs to explore high performance computation, parallel computation, and distributed computation, with advanced hardware and cloud computing or cluster computing environment.

Data mining is also closely tied with human-computer interaction. Users need to interact with a data mining system or process in an effective way, telling the system what to mine, how to incorporate background knowledge, how to mine, and how to present the mining results in an easy-to-understand (e.g., by interpretation and visualization) and easy-to-interact way (e.g., with friendly graphic user interface and interactive mining).

Actually, nowadays, there are not only many interactive data mining systems but also many more data mining functions hidden in various kinds of application programs. It is unrealistic to expect everyone in our society to understand and master data mining techniques. It is also forbidden for industries to expose their large data sets. Many systems have data mining functions built within so that people can perform data mining or use data mining results simply by mouse clicking. For example, intelligent search engines and online retails perform such **invisible data mining** by collecting their data and user's search or purchase history, incorporating data mining into their components to improve their performance, functionality, and user satisfaction. When your grandma shops online, she may be surprised

when receiving some smart recommendations. This could likely be resulted from such invisible data mining.

1.6 Data mining and applications

Where there are data, there are data mining applications

As a highly application-driven discipline, data mining has seen great successes in many applications. It is impossible to enumerate all applications where data mining plays a critical role. Presentations of data mining in knowledge-intensive application domains, such as bioinformatics and software engineering, require more in-depth treatment and are beyond the scope of this book. To demonstrate the importance of applications of data mining, we briefly discuss a few highly successful and popular application examples of data mining: *business intelligence*; *search engines*; *social media and social networks*; and *biology, medical science, and health care*.

Business intelligence

It is critical for businesses to acquire a better understanding of the commercial context of their organization, such as their customers, the market, supply and resources, and competitors. **Business intelligence** (**BI**) technologies provide historical, current, and predictive views of business operations. Examples include reporting, online analytical processing, business performance management, competitive intelligence, benchmarking, and predictive analytics.

"How important is data mining in business intelligence?" Without data mining, many businesses may not be able to perform effective market analysis, compare customer feedback on similar products, discover the strengths and weaknesses of their competitors, retain highly valuable customers, and make smart business decisions.

Clearly, data mining is the core of business intelligence. Online analytical processing tools in business intelligence rely on data warehousing and multidimensional data mining. Classification and prediction techniques are the core of predictive analytics in business intelligence, for which there are many applications in analyzing markets, supplies, and sales. Moreover, clustering plays a central role in customer relationship management, which groups customers based on their similarities. Using multidimensional summarization techniques, we can better understand features of each customer group and develop customized customer reward programs.

Web search engines

A **Web search engine** is a specialized computer server that searches for information on the Web. The search results of a user query are often returned as a list (sometimes called *hits*). The hits may consist of web pages, images, and other types of files. Some search engines also search and return data available in public databases or open directories. Search engines differ from **web directories** in that web directories are maintained by human editors, whereas search engines operate algorithmically or by a mixture of algorithmic and human input.

Search engines pose grand challenges to data mining. First, they have to handle a huge and ever-growing amount of data. Typically, such data cannot be processed using one or a few machines. Instead, search engines often need to use *computer clouds*, which consist of thousands or even hundreds of thou-

sands of computers that collaboratively mine the huge amount of data. Scaling up data mining methods over computer clouds and large distributed data sets is an area of active research and development.

Second, Web search engines often have to deal with online data. A search engine may be able to afford constructing a model offline on huge datasets. To do this, it may construct a query classifier that assigns a search query to predefined categories based on the query topic (i.e., whether the search query "apple" is meant to retrieve information about a fruit or a brand of computers). Even if a model is constructed offline, the adaptation of the model online must be fast enough to answer user queries in real time.

Another challenge is maintaining and incrementally updating a model on fast-growing data streams. For example, a query classifier may need to be incrementally maintained continuously since new queries keep emerging and predefined categories and the data distribution may change. Most of the existing model training methods are offline and static and thus cannot be used in such a scenario.

Third, Web search engines often have to deal with queries that are asked only a very small number of times. Suppose a search engine wants to provide *context-aware* query recommendations. That is, when a user poses a query, the search engine tries to infer the context of the query using the user's profile and his query history in order to return more customized answers within a small fraction of a second. However, although the total number of queries asked can be huge, many queries may be asked only once or a few times. Such severely skewed data are challenging for many data mining and machine learning methods.

Social media and social networks

The prevalence of social media and social networks has fundamentally changed our life and the way we exchange information and socialize nowadays. With tremendous amounts of social media and social network data available, it is critical to analyze such data to extract actionable patterns and trends from social media and social network data.

Social media mining is to sift through massive amounts of social media data (e.g., on social media usage, online social behaviors, connections between individuals, online shopping behavior, content exchange, etc.) in order to discern patterns and trends. These patterns and trends have been used for social event detection, public health monitoring and surveillance, sentiment analysis in social media, recommendation in social media, information provenance, social media trustability analysis, and social spammer detection.

Social network mining is to investigate social network structures and the information associated with such networks through the use of networks and graph theory and data mining methods. The social network structures are characterized in terms of nodes (individual actors, people, or things within the network) and the ties, edges, or links (relationships or interactions) that connect them. Examples of social structures commonly visualized through social network analysis include social media networks, memes spread, friendship and acquaintance networks, collaboration graphs, kinship, disease transmission, and sexual relationships. These networks are often visualized through sociograms in which nodes are represented as points and ties are represented as lines.

Social network mining has been used to detect hidden communities, uncover the evolution and dynamics of social networks, compute network measures (e.g., centrality, transitivity, reciprocity, balance, status, and similarity), analyze how information propagates in social media sites, measure and model node/substructure influence and homophily, and conduct location-based social network analysis.

Social media mining and social network mining are important applications of data mining.

Biology, medical science, and health care

Biology, medical science and health care have also been generating massive data at exponential scale. Biomedical data take many forms, from "omics" to imaging, mobile health, and electronic health records. With the availability of more efficient digital collection methods, biomedical scientists and clinicians now find themselves confronting ever larger sets of data and trying to devise creative ways to sift through this mountain of data and make sense of it. Indeed, data that used to be considered large now seems small as the amount of data now being collected in a single day by an investigator can surpass what might have been generated over his/her career even a decade ago. This deluge of biomedical information requires new thinking about how data can be managed and analyzed to further scientific understanding and for improving healthcare.

Biomedical data mining involves many challenging data mining tasks, including mining massive genomic and proteomic sequence data, mining frequent subgraph patterns for classifying biological data, mining regulatory networks, characterization and prediction of protein-protein interactions, classification and predictive analysis of medical images, biological text mining, biological information network construction from biotext data, mining electronic health records, and mining biomedical networks.

1.7 Data mining and society

With data mining penetrating our everyday lives, it is important to study the impact of data mining on society. How can we use data mining technology to benefit society? How can we guard against its misuse? The improper disclosure or use of data and the potential violation of individual privacy and data protection rights are areas of concern that need to be addressed.

Data mining will help scientific discovery, business management, economy recovery, and security protection (e.g., the real-time discovery of intruders and cyberattacks). However, it also poses the risk of unintentionally disclosing some confidential business or government information and disclosing an individual's personal information. Studies on data security in data mining and privacy-preserving data publishing and data mining are important, ongoing research theme. The philosophy is to observe data sensitivity and preserve data security and people's privacy while performing successful data mining.

These issues and many additional ones relating to the research, development, and application of data mining will be discussed throughout the book.

1.8 Summary

- *Necessity is the mother of invention.* With the mounting growth of data in every application, data mining meets the imminent need for effective, scalable, and flexible data analysis in our society. Data mining can be considered as a natural evolution of information technology and a confluence of several related disciplines and application domains.
- **Data mining** is the process of discovering interesting patterns and knowledge from massive amounts of data. As a *knowledge discovery process*, it typically involves data cleaning, data integration, data selection, data transformation, pattern and model discovery, pattern or model evaluation, and knowledge presentation.

- A pattern or model is *interesting* if it is valid on test data with some degree of certainty, novel, potentially useful (e.g., can be acted on or validates a hunch about which the user was curious), and easily understood by humans. Interesting patterns represent knowledge. Measures of **pattern interestingness**, either *objective* or *subjective*, can be used to guide the discovery process.
- Data mining can be conducted on any kind of **data** as long as the data are meaningful for a target application, such as structured data (e.g., relational database, transaction data) and unstructured data (e.g., text and multimedia data), as well as data associated with different applications. Data can also be categorized as stored vs. stream data, whereas the latter may need to explore special stream mining algorithms.
- **Data mining functionalities** are used to specify the kinds of patterns or **knowledge** to be found in data mining tasks. The functionalities include characterization and discrimination; the mining of frequent patterns, associations, and correlations; classification and regression; deep learning; cluster analysis; and outlier detection. As new types of data, new applications, and new analysis demands continue to emerge, there is no doubt we will see more and more novel data mining tasks in the future.
- Data mining, is a confluence of multiple disciplines but it has its unique research focus, dedicated to many advanced applications. We study the close relationships of data mining with statistics, machine learning, database technology, and many other disciplines.
- Data mining has many successful **applications**, such as business intelligence, Web search, bioinformatics, health informatics, finance, digital libraries, and digital governments.
- Data mining may already have its strong impact on the society and the study of such impact, such as how to ensure the effectiveness of data mining and in the meantime ensure the data privacy and security, has become an important issue in research.

1.9 Exercises

1.1. What is *data mining*? In your answer, address the following:

- **a.** Is it a simple transformation or application of technology developed from *databases*, *statistics*, *machine learning*, and *pattern recognition*?
- **b.** Someone believes that data mining is an inevitable result of the evolution of information technology. If you are a database researcher, show data mining is resulted from a nature evolution of database technology. What about if you are a machine learner researcher, or a statistician?
- **c.** Describe the steps involved in data mining when viewed as a process of knowledge discovery.

1.2. Define each of the following *data mining functionalities*: association and correlation analysis, classification, regression, clustering, deep learning, and outlier analysis. Give examples of each data mining functionality, using a real-life database that you are familiar with.

1.3. Present an example where data mining is crucial to the success of a business. What *data mining functionalities* does this business need (e.g., think of the kinds of patterns that could be mined)? Can such patterns be generated alternatively by data query processing or simple statistical analysis?

1.4. Explain the difference and similarity between correlation analysis and classification, between classification and clustering, and between classification and regression.

1.5. Based on your observations, describe another possible kind of knowledge that needs to be discovered by data mining methods but has not been listed in this chapter. Does it require a mining methodology that is quite different from those outlined in this chapter?

1.6. *Outliers* are often discarded as noise. However, one person's garbage could be another's treasure. For example, exceptions in credit card transactions can help us detect the fraudulent use of credit cards. Using fraud detection as an example, propose two methods that can be used to detect outliers and discuss which one is more reliable.

1.7. What are the major challenges of mining a huge amount of data (e.g., billions of tuples) in comparison with mining a small amount of data (e.g., data set of a few hundred tuples)?

1.8. Outline the major research challenges of data mining in one specific application domain, such as stream/sensor data analysis, spatiotemporal data analysis, or bioinformatics.

1.10 Bibliographic notes

The book *Knowledge Discovery in Databases*, edited by Piatetsky-Shapiro and Frawley [PSF91], is an early collection of research papers on knowledge discovery from data. The book *Advances in Knowledge Discovery and Data Mining*, edited by Fayyad, Piatetsky-Shapiro, Smyth, and Uthurusamy [FPSSe96], is another early collection of research results on knowledge discovery and data mining. There have been many data mining textbook or research books published since then. Some popular ones include *Data Mining: Practical Machine Learning Tools and Techniques* (4th ed.) by Witten, Frank, Hall and Pal [WFHP16]; *Data Mining: Concepts and Techniques* (3rd ed.) by Han and Kamber and Pei [HKP11], *Introduction to Data Mining* (2nd ed.) by Tan, Steinbach, Karpatne, and Kumar [TSKK18]; *Data Mining: The Textbook* [Agg15b]; *Data Mining and Machine Learning: Fundamental Concepts and Algorithms* (2nd ed.) by Zaki and Meira [ZJ20]; *Mining of Massive Datasets* (3rd ed.) by Leskovec, Rajaraman and Ullman [ZJ20]; *The Elements of Statistical Learning* (2nd ed.) by Hastie, Tibshirani, and Friedman [HTF09]; *Data Mining Techniques: For Marketing, Sales, and Customer Relationship Management* (3rd ed.) by Linoff and Berry [LB11]; *Principles of Data Mining (Adaptive Computation and Machine Learning)* by Hand, Mannila, and Smyth [HMS01]; *Mining the Web: Discovering Knowledge from Hypertext Data* by Chakrabarti [Cha03]; *Web Data Mining: Exploring Hyperlinks, Contents, and Usage Data* by Liu [Liu06]; and *Data Mining: Multimedia, Soft Computing, and Bioinformatics* by Mitra and Acharya [MA03].

There are also numerous books that contain collections of papers or chapters on particular aspects of knowledge discovery, such as cluster analysis, outlier detection, classification, association mining, and mining particular kinds of data, such as mining text data, multimedia data, relational data, geospatial data, social and information network data, and social media data. However, this list has gone very long over the years and we will not list them individually. There are numerous tutorial notes on data mining in major data mining, database, machine learning, statistics, and Web technology conferences.

KDNuggets is a regular electronic newsletter containing information relevant to knowledge discovery and data mining, moderated by Piatetsky-Shapiro since 1991. The Internet site *KDNuggets* (*https://www.kdnuggets.com*) contains a good collection of KDD-related information.

The data mining community started its first international conference on knowledge discovery and data mining in 1995. The conference evolved from the four international workshops on knowledge discovery in databases, held from 1989 to 1994. ACM-SIGKDD, a Special Interest Group on Knowl-

edge Discovery in Databases was set up under ACM in 1998 and has been organizing the international conferences on knowledge discovery and data mining since 1999. IEEE Computer Science Society has organized its annual data mining conference, International Conference on Data Mining (ICDM), since 2001. SIAM (Society on Industrial and Applied Mathematics) has organized its annual data mining conference, SIAM Data Mining Conference (SDM), since 2002. A dedicated journal, *Data Mining and Knowledge Discovery*, published by Springer, has been available since 1997. An ACM journal, *ACM Transactions on Knowledge Discovery from Data*, published its first volume in 2007.

ACM-SIGKDD also publishes a bi-annual newsletter, *SIGKDD Explorations*. There are a few other international or regional conferences on data mining, such as the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECMLPKDD), the Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD), and the International Conference on Web Search and Data Mining (WSDM).

Research in data mining has also been popularly published in many textbooks, research books, conferences, and journals on data mining, database, statistics, machine learning, and data visualization.

CHAPTER

Data, measurements, and data preprocessing

2

To conduct successful data mining, the first important thing is to get familiar with your data. You may want to know the following: What are the types of *attributes* or fields that make up your data? What kind of values does each attribute have? How are the values distributed? How can we measure the similarity of some data objects with respect to others? Gaining such insights into the data will help with the subsequent analysis. Moreover, real-world data are typically noisy, enormous in volume (often several gigabytes or more), and may originate from a hodgepodge of heterogeneous sources. How can we measure the quality of data? How can we clean and integrate data from multiple heterogeneous sources? How can we normalize, compress, or transform the data? How can we reduce the dimensionality of data to help subsequent analysis? These are the tasks of this chapter.

We begin in Section 2.1 by studying the various attribute types. These include nominal attributes, binary attributes, ordinal attributes, and numeric attributes. Basic *statistical descriptions* can be used to learn more about each attribute's values, as described in Section 2.2. Given a *temperature* attribute, for example, we can determine its *mean* (average value), *median* (middle value), and *mode* (most common value). These are *measures of central tendency*, which give us an idea of the "middle" or center of a distribution. Knowing such basic statistics regarding each attribute makes it easier to fill in missing values, smooth noisy values, and spot outliers during data preprocessing. Knowledge of the attributes and attribute values can also help in fixing inconsistencies incurred during data integration. Plotting the measures of central tendency shows us if the data are symmetric or skewed. Quantile plots, histograms, and scatter plots are other graphic displays of basic statistical descriptions. These can all be useful during data preprocessing and can provide insight into areas for mining.

We may also want to examine how *similar* (or *dissimilar*) data objects are. For example, suppose we have a database where the data objects are patients, described by their symptoms. We may want to find the similarity or dissimilarity between individual patients. Such information can allow us to find clusters of like patients within the data set. The similarity (or dissimilarity) between objects may also be used to detect outliers in the data, or to perform nearest-neighbor classification. There are many measures for assessing similarity and dissimilarity. In general, such measures are referred to as *proximity measures*. Think of the proximity of two objects as a function of the *distance* between their attribute values, although proximity can also be calculated based on probabilities rather than actual distance. Measures of data proximity are described in Section 2.3.

Finally, we will discuss data preprocessing, which is to address today's real-world challenges: data sets are highly susceptible to noisy, missing, and inconsistent data due to their typically huge size and their likely origin from multiple, heterogeneous sources. Low-quality data will lead to low-quality mining results. Huge efforts need to be paid to preprocess the data to enhance the quality of data for effective mining. Section 2.4 is on *data cleaning* and *data integration*. The former is to remove noise and correct inconsistencies in data, whereas the latter is to merge data from multiple sources into a

Data Mining. https://doi.org/10.1016/B978-0-12-811760-6.00012-6

coherent data store such as a data warehouse. Section 2.5 is on *data transformation*, which transforms or consolidates data into forms appropriate for mining. That is, it can make the resulting mining process be more efficient, and the patterns found be easier to understand. Various strategies for data transformation have been developed. For example, *data normalization* scales the attribute data to fall within a smaller range, like 0.0 to 1.0; *data discretization* replaces the raw values of a numeric attribute by interval labels or conceptual labels; and *data reduction* techniques (e.g., *compression* and *sampling*) transform the input data to a reduced representation and can improve the accuracy and efficiency of mining algorithms involving distance measurements. Last, Section 2.6 is on *dimensionality reduction*, which is the process of reducing the number of random variables or attributes under consideration. Please note that various data preprocessing techniques are not mutually exclusive; they may work together. For example, data cleaning can involve transformations to correct wrong data, such as by transforming all entries for a *date* field to a common format.

2.1 Data types

Data sets are made up of data objects. A **data object** represents an entity—in a sales database, the objects may be customers, store items, and sales; in a medical database, the objects may be patients; in a university database, the objects may be students, professors, and courses. Data objects are typically described by attributes. Data objects can also be referred to as *samples, examples, instances, data points*, or *objects*. If the data objects are stored in a database, they are *data tuples*. That is, the rows of a database correspond to the data objects, and the columns correspond to the attributes. In this section, we define attributes and look at the various attribute types.

What is an attribute? An **attribute** is a data field, representing a characteristic or feature of a data object. The nouns *attribute*, *dimension*, *feature*, and *variable* are often used interchangeably in the literature. The term *dimension* is commonly used in data warehousing. Machine learning literature tends to use the term *feature*, whereas statisticians prefer the term *variable*. Data mining and database professionals commonly use the term *attribute*, and we do here as well. Attributes describing a customer object can include, for example, *customer_ID*, *name*, and *address*. Observed values for a given attribute are known as *observations*. A set of attributes used to describe a given object is called an *attribute vector* (or *feature vector*). The distribution of data involving one attribute (or variable) is called *univariate*. A *bivariate* distribution involves two attributes, and so on.

The **type** of an attribute is determined by the set of possible values—nominal, binary, ordinal, or numeric—the attribute can have. In the following subsections, we introduce each type.

2.1.1 Nominal attributes

Nominal means "relating to names." The values of a **nominal attribute** are symbols or *names of things*. Each value represents some kind of category, code, or state, and so nominal attributes are also referred to as **categorical**. The values do not have any meaningful order. In computer science, the values are also known as *enumerations*.

Example 2.1. Nominal attributes. Suppose that *hair_color* and *marital_status* are two attributes describing *person* objects. In our application, possible values for *hair_color* are *black*, *brown*, *blond*, *red*, *auburn*, *gray*, and *white*. The attribute *marital_status* can take on the values *single, married, divorced*,

and *widowed*. Both *hair_color* and *marital_status* are nominal attributes. Another example of a nominal attribute is *occupation*, with the values *teacher, dentist, programmer, farmer*, and so on. □

Although we said that the values of a nominal attribute are symbols or "names of things," it is possible to represent such symbols or "names" with numbers. With *hair_color*, for instance, we can assign a code of 0 for *black*, 1 for *brown*, and so on. Another example is *customer_ID*, with possible values that are all numeric. However, in such cases, the numbers are not intended to be used quantitatively. That is, mathematical operations on values of nominal attributes are not meaningful. It makes no sense to subtract one customer ID number from another, unlike, say, subtracting an age value from another (where *age* is a numeric attribute). Even though a nominal attribute may have integers as values, it is not considered a numeric attribute because the integers are not meant to be used quantitatively. We will say more on numeric attributes in Section 2.1.4.

Because nominal attribute values do not have any meaningful order about them and are not quantitative, it makes no sense to find the mean (average) value or median (middle) value for such an attribute, given a set of objects. One thing that is of interest, however, is the attribute's most commonly occurring value. This value, known as the *mode*, is one of the measures of central tendency. You will learn about measures of central tendency in Section 2.2.

2.1.2 Binary attributes

A **binary attribute** is a nominal attribute with only two categories or states: 0 or 1, where 0 typically means that the attribute is absent, and 1 means that it is present. Binary attributes are referred to as **Boolean** if the two states correspond to *true* and *false*.

Example 2.2. Binary attributes. Given the attribute *smoker* describing a *patient* object, 1 indicates that the patient smokes, whereas 0 indicates that the patient does not. Similarly, suppose the patient undergoes a medical test that has two possible outcomes. The attribute *medical_test* is binary, where a value of 1 means the result of the test for the patient is positive, whereas 0 means the result is negative. □

A binary attribute is **symmetric** if both of its states are equally valuable and carry the same weight; that is, there is no preference on which outcome should be coded as 0 or 1. One such example could be the attribute *gender* having the states *male* and *female*.

A binary attribute is **asymmetric** if the outcomes of the states are not equally important, such as the *positive* and *negative* outcomes of a medical test for HIV. By convention, we code the most important outcome, which is usually the rarer one, by 1 (e.g., *HIV positive*) and the other by 0 (e.g., *HIV negative*).

Computing similarities between objects involving symmetric and asymmetric binary attributes will be discussed in a later section of this chapter.

2.1.3 Ordinal attributes

An **ordinal attribute** is an attribute with possible values that have a meaningful order or *ranking* among them, but the magnitude between successive values is not known.

Example 2.3. Ordinal attributes. Suppose that *drink_size* corresponds to the size of drinks available at a fast-food restaurant. This nominal attribute has three possible values: *small, medium*, and *large*.

The values have a meaningful sequence (which corresponds to increasing drink size); however, we cannot tell from the values *how much* bigger, say, a large is than a medium. Other examples of ordinal attributes include *grade* (e.g., *A+, A, A−, B+*, and so on) and *professional_rank*. Professional ranks can be enumerated in a sequential order: for example, *assistant*, *associate*, and *full* for professors, and *private, private second class, private first class, specialist, corporal, sergeant, ...* for army ranks.

Ordinal attributes are useful for registering subjective assessments of qualities that cannot be measured objectively; thus ordinal attributes are often used in surveys for ratings. In one survey, participants were asked to rate how satisfied they were as customers. Customer satisfaction had the following ordinal categories: *1: very dissatisfied, 2: dissatisfied, 3: neutral, 4: satisfied*, and *5: very satisfied.* □

Ordinal attributes may also be obtained from the discretization of numeric quantities by splitting the value range into a finite number of ordered categories as described in a later section on data reduction.

The central tendency of an ordinal attribute can be represented by its mode and its median (the middle value in an ordered sequence), but the mean cannot be defined.

Note that nominal, binary, and ordinal attributes are *qualitative*. That is, they *describe* a feature of an object without giving an actual size or quantity. The values of such qualitative attributes are typically words representing categories. If integers are used, they represent computer codes for the categories, as opposed to measurable quantities (e.g., 0 for *small* drink size, 1 for *medium*, and 2 for *large*). In the following subsection we look at numeric attributes, which provide *quantitative* measurements of an object.

2.1.4 Numeric attributes

A **numeric attribute** is *quantitative*; that is, it is a measurable quantity, represented in integer or real values. Numeric attributes can be *interval-scaled* or *ratio-scaled*.

Interval-scaled attributes

Interval-scaled attributes are measured on a scale of equal-size units. The values of interval-scaled attributes have order and can be positive, 0, or negative. Thus, in addition to providing a ranking of values, such attributes allow us to compare and quantify the *difference* between values.

Example 2.4. Interval-scaled attributes. A *temperature* attribute is interval-scaled. Suppose that we have the outdoor *temperature* values for a number of different days, where each day is an object. By ordering the values, we obtain a ranking of the objects with respect to *temperature*. In addition, we can quantify the difference between values. For example, a temperature of 20 °C is five degrees higher than a temperature of 15 °C. Calendar dates are another example. For instance, the years 2012 and 2020 are eight years apart. □

Temperatures in Celsius and Fahrenheit do not have a true zero-point, that is, neither 0 °C nor 0 °F indicates "no temperature." (On the Celsius scale, for example, the unit of measurement is 1/100 of the difference between the melting temperature and the boiling temperature of water in atmospheric pressure.) Although we can compute the *difference* between temperature values, we cannot talk of one temperature value as being a *multiple* of another. Without a true zero, we cannot say, for instance, that 10 °C is twice as warm as 5 °C. That is, we cannot speak of the values in terms of ratios. Similarly, there is no true zero-point for calendar dates. (The year 0 does not correspond to the beginning of time.) This brings us to ratio-scaled attributes, for which a true zero-point exists.

Because interval-scaled attributes are numeric, we can compute their mean value, in addition to the median and mode measures of central tendency.

Ratio-scaled attributes

A **ratio-scaled attribute** is a numeric attribute with an inherent zero-point. That is, if a measurement is ratio-scaled, we can speak of a value as being a multiple (or ratio) of another value. In addition, the values are ordered, and we can also compute the difference between values, as well as the mean, median, and mode.

Example 2.5. Ratio-scaled attributes. Unlike temperatures in Celsius and Fahrenheit, the Kelvin (K) temperature scale has what is considered a true zero-point ($0\,\mathrm{K} = -273.15\,^{\circ}\mathrm{C}$): It is the point at which all thermal motion ceases in the classical description of thermodynamics. Other examples of ratio-scaled attributes include *count* attributes such as *years_of_experience* (e.g., the objects are employees) and *number_of_words* (e.g., the objects are documents). Additional examples include attributes to measure weight, height, and speed, and monetary quantities (e.g., you are 100 times richer with \$100 than with \$1). □

2.1.5 Discrete vs. continuous attributes

In our presentation, we have organized attributes into nominal, binary, ordinal, and numeric types. There are many ways to organize attribute types. The types are not mutually exclusive.

Classification algorithms developed from the field of machine learning often consider attributes as being either *discrete* or *continuous*. Each type may be processed differently. A **discrete attribute** has a finite or countably infinite set of values, which may or may not be represented as integers. The attributes *hair_color*, *smoker*, *medical_test*, and *drink_size* each have a finite number of values, and so are discrete. Note that discrete attributes may have numeric values, such as 0 and 1 for binary attributes or, the values 0 to 110 for the attribute *age*. An attribute is *countably infinite* if the set of possible values is infinite but the values can be put in a one-to-one correspondence with natural numbers. For example, the attribute *customer_ID* is countably infinite. The number of customers can grow to infinity, but in reality, the actual set of values is countable (where the values can be put in one-to-one correspondence with the set of integers). Zip codes are another example.

If an attribute is not discrete, it is **continuous**. The terms *numeric attribute* and *continuous attribute* are often used interchangeably in the literature. (This can be confusing because, in the classic sense, continuous values are real numbers, whereas numeric values can be either integers or real numbers.) In practice, real values are represented using a finite number of digits. Continuous attributes are typically represented as floating-point variables.

2.2 Statistics of data

For data preprocessing to be successful, it is essential to have an overall picture of your data. Basic statistical descriptions can be used to identify properties of the data and highlight which data values should be treated as noise or outliers.

This section discusses three areas of basic statistical descriptions. We start with *measures of central tendency* (Section 2.2.1), which measure the location of the middle or center of a data distribution.

Intuitively speaking, given an attribute, where do most of its values fall? In particular, we discuss the mean, median, mode, and midrange.

In addition to assessing the central tendency of our data set, we also would like to have an idea of the *dispersion of the data*. That is, how are the data spread out? The most common data dispersion measures are the *range*, *quartiles* (e.g., Q_1, which is the first quartile, i.e., the 25th percentile), and *interquartile range*; the *five-number summary* and *boxplots*; and the *variance* and *standard deviation* of the data. These measures are useful for identifying outliers and are described in Section 2.2.2.

To facilitate the description of relations among multiple variables, the concepts of *co-variance* and *correlation coefficient* for numerical data and χ^2 *correlation test* for nominal data are introduced in Section 2.2.3.

Finally, we can use many graphic displays of basic statistical descriptions to visually inspect our data (Section 2.2.4). Most statistical or graphical data presentation software packages include bar charts, pie charts, and line graphs. Other popular displays of data summaries and distributions include *quantile plots*, *quantile-quantile plots*, *histograms*, and *scatter plots*.

2.2.1 Measuring the central tendency

In this section, we look at various ways to measure the central tendency of data. Suppose that we have some attribute X, like *salary*, which has been recorded for a set of objects. Let $x_1, x_2, \ldots, x_N$ be the set of N observed values or *observations* for X. Here, these values may also be referred to as the data set (for X). If we were to plot the observations for *salary*, where would most of the values fall? This gives us an idea of the central tendency of the data. Measures of central tendency include the mean, median, mode, and midrange.

The most common and effective numeric measure of the "center" of a set of data is the *(arithmetic) mean*. Let $x_1, x_2, \ldots, x_N$ be a set of N values or *observations*, such as for some numeric attribute X, like *salary*. The **mean** of this set of values is

$$\bar{x} = \frac{\sum_{i=1}^{N} x_i}{N} = \frac{x_1 + x_2 + \cdots + x_N}{N}. \tag{2.1}$$

This corresponds to the built-in aggregate function, *average* (`avg()` in SQL), provided in relational database systems.

Example 2.6. Mean. Suppose we have the following values for *salary* (in thousands of dollars), shown in ascending order: 30, 36, 47, 50, 52, 52, 56, 60, 63, 70, 70, 110. Using Eq. (2.1), we have

$$\begin{aligned}\bar{x} &= \frac{30 + 36 + 47 + 50 + 52 + 52 + 56 + 60 + 63 + 70 + 70 + 110}{12} \\ &= \frac{696}{12} = 58.\end{aligned}$$

Thus, the mean salary is $58,000. □

Sometimes, each value x_i in a set may be associated with a weight w_i for $i = 1, \ldots, N$. The weights reflect the significance, importance, or occurrence frequency attached to their respective values. In this

case, we can compute

$$\bar{x} = \frac{\sum_{i=1}^{N} w_i x_i}{\sum_{i=1}^{N} w_i} = \frac{w_1 x_1 + w_2 x_2 + \cdots + w_N x_N}{w_1 + w_2 + \cdots + w_N}. \tag{2.2}$$

This is called the **weighted arithmetic mean** or the **weighted average**.

Although the mean is the single most useful quantity for describing a data set, it is not always the best way of measuring the center of the data. A major problem with the mean is its sensitivity to extreme (e.g., outlier) values. Even a small number of extreme values can corrupt the mean. For example, the mean salary at a company may be substantially pushed up by that of a few highly paid managers. Similarly, the mean score of a class in an exam could be pulled down quite a bit by a few very low scores. To offset the effect caused by a small number of extreme values, we can instead use the **trimmed mean**, which is the mean obtained after chopping off values at the high and low extremes. For example, we can sort the values observed for *salary* and remove the top and bottom 2% before computing the mean. We should avoid trimming too large a portion (such as 20%) at both ends, as this can result in the loss of valuable information.

For skewed (asymmetric) data, a better measure of the center of data is the **median**, which is the middle value in a set of ordered data values. It is the value that separates the higher half of a data set from the lower half.

In probability and statistics, the median generally applies to numeric data; however, we may extend the concept to ordinal data. Suppose that a given data set of N values for an attribute X is sorted in ascending order. If N is odd, then the median is the *middle value* of the ordered set. If N is even, then the median is not unique; it is the two middlemost values and any value in between. If X is a numeric attribute in this case, by convention, the median is taken as the average of the two middlemost values.

Example 2.7. Median. Let's find the median of the data from Example 2.6. The data are already sorted in ascending order. There is an even number of observations (i.e., 12); therefore, the median is not unique. It can be any value within the two middlemost values of 52 and 56 (that is, within the sixth and seventh values in the list). By convention, we assign the average of the two middlemost values as the median; that is, $\frac{52+56}{2} = \frac{108}{2} = 54$. Thus, the median is \$54,000.

Suppose that we had only the first 11 values in the list. Given an odd number of values, the median is the middlemost value. This is the sixth value in this list, which has a value of \$52,000. □

The median is expensive to compute when we have a large number of observations. For numeric attributes, however, we can easily *approximate* the value. Assume that data are grouped in intervals according to their x_i data values and that the frequency (i.e., number of data values) of each interval is known. For example, employees may be grouped according to their annual salary in intervals such as \$10,001–20,000, \$20,001–50,000, and so on. (A similar, concrete example can be seen in the data table of Exercise 2.3.) Let the interval that contains the median frequency be the *median interval*. We can approximate the median of the entire data set (e.g., the median salary) by interpolation using the

formula

$$median \approx L_1 + \left(\frac{N/2 - \left(\sum freq\right)_l}{freq_{median}} \right) \times width, \tag{2.3}$$

where L_1 is the lower boundary of the median interval, N is the number of values in the entire data set, $\left(\sum freq\right)_l$ is the sum of the frequencies of all of the intervals that are lower than the median interval, $freq_{median}$ is the frequency of the median interval, and *width* is the width of the median interval.

Mode is another measure of central tendency. The **mode** for a set of data is the value that occurs most frequently compared to all neighboring values in the set. Therefore, it can be determined for qualitative and quantitative attributes. It is possible for the greatest frequency to correspond to several different values, which results in more than one mode. Data sets with one, two, or three modes are respectively called **unimodal**, **bimodal**, and **trimodal**. In general, a data set with two or more modes is **multimodal**.

Example 2.8. Mode. The data from Example 2.6 are bimodal. The two modes are \$52,000 and \$70,000. □

For unimodal numeric data that are moderately skewed (asymmetrical), we have the following empirical relation:

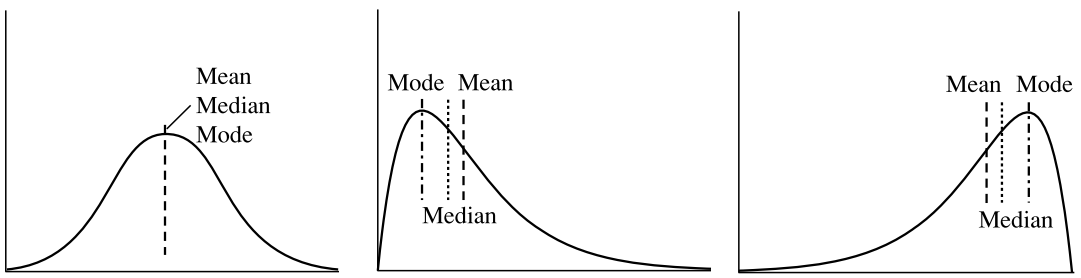
$$mean - mode \approx 3 \times (mean - median). \tag{2.4}$$

This implies that the mode for unimodal frequency curves that are moderately skewed can easily be approximated if the mean and median values are known.

The **midrange** can also be used to assess the central tendency of a numeric data set. It is the average of the largest and smallest values in the set. This measure is easy to compute using the SQL aggregate functions, `max()` and `min()`.

Example 2.9. Midrange. The midrange of the data of Example 2.6 is $\frac{30{,}000+110{,}000}{2} = \$70{,}000$. □

In a unimodal frequency curve with perfect **symmetric** data distribution, the mean, median, and mode are all at the same center value, as shown in Fig. 2.1a.



(a) Symmetric data **(b)** Positively skewed data **(c)** Negatively skewed data

FIGURE 2.1

Mean, median, and mode of symmetric vs. positively and negatively skewed data.

Data in most real applications are not symmetric. They may instead be either **positively skewed**, where the mode occurs at a value that is smaller than the median (Fig. 2.1b), or **negatively skewed**, where the mode occurs at a value greater than the median (Fig. 2.1c).

2.2.2 Measuring the dispersion of data

We now look at measures to assess the dispersion or spread of numeric data. The measures include range, quantiles, quartiles, percentiles, and the interquartile range. The five-number summary, which can be displayed as a boxplot, is useful in identifying outliers. Variance and standard deviation also indicate the spread of a data distribution.

Range, quartiles, and interquartile range

To start off, let's study the *range*, *quantiles*, *quartiles*, *percentiles*, and the *interquartile range* as measures of data dispersion.

Let $x_1, x_2, \ldots, x_N$ be a set of observations for some numeric attribute, X. The **range** of the set is the difference between the largest (max()) and smallest (min()) values.

Suppose that the data for attribute X are sorted in ascending numeric order. Imagine that we can pick certain data points so as to split the data distribution into equal-size consecutive sets, as in Fig. 2.2. These data points are called *quantiles*. **Quantiles** are points taken at regular intervals of a data distribution, dividing it into essentially equal-size consecutive sets. (We say "essentially" because there may not be data values of X that divide the data into exactly equal-size subsets. For readability, we will refer to them as equal.) The kth q-*quantile* for a given data distribution is the value x such that at most k/q of the data values are less than x and at most $(q-k)/q$ of the data values are more than x, where k is an integer such that $0 < k < q$. There are $q-1$ q-quantiles.

The 2-quantile is the data point dividing the lower and upper halves of the data distribution. It corresponds to the median. The 4-quantiles are the three data points that split the data distribution into four equal parts; each part represents one-fourth of the data distribution. They are more commonly referred to as **quartiles**. The 100-quantiles are more commonly referred to as **percentiles**; they divide

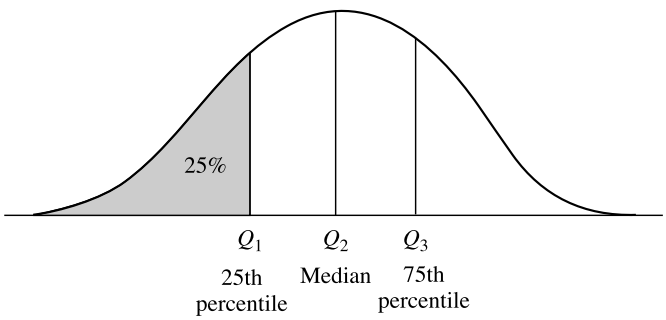


FIGURE 2.2

A plot of the data distribution for some attribute X. The quantiles plotted are quartiles. The three quartiles divide the distribution into four equal-size consecutive subsets. The second quartile corresponds to the median.

the data distribution into 100 equal-size consecutive sets. The median, quartiles, and percentiles are the most widely used forms of quantiles.

The quartiles give an indication of a distribution's center, spread, and shape. The **first quartile**, denoted by Q_1, is the 25th percentile. It cuts off the lowest 25% of the data. The **third quartile**, denoted by Q_3, is the 75th percentile—it cuts off the lowest 75% (or highest 25%) of the data. The second quartile is the 50th percentile. As the median, it gives the center of the data distribution.

The distance between the first and third quartiles is a simple measure of spread that gives the range covered by the middle half of the data. This distance is called the **interquartile range** (**IQR**) and is defined as

$$IQR = Q_3 - Q_1. \tag{2.5}$$

Example 2.10. Interquartile range. The quartiles are the three values that split the sorted data set into four equal parts. The data of Example 2.6 contain 12 observations, already sorted in ascending order. Since there are even number of elements on this list, the median of the list should be the mean of the center two elements, that is (\$52,000 + \$56,000)/2 = \$54,000. Then the first quartile should be the mean of the 3rd and 4th elements, that is, (\$47,000 + \$50,000)/2 = \$48,500, whereas the 3rd quartile should be the mean of the 9th and 10th elements, that is, (\$63,000 + \$70,000)/2 = \$66,500. Thus the interquartile range is IQR = \$66,500 − \$48,500 = \$18,000. □

Five-number summary, boxplots, and outliers

No single numeric measure of spread (e.g., IQR) is very useful for describing skewed distributions. Have a look at the symmetric and skewed data distributions of Fig. 2.1. In the symmetric distribution, the median (and other measures of central tendency) splits the data into equal-size halves. This does not occur for skewed distributions. Therefore it is more informative to also provide the two quartiles Q_1 and Q_3, along with the median. A common rule of thumb for identifying suspected **outliers** is to single out values falling at least $1.5 \times IQR$ above the third quartile or below the first quartile.

Because Q_1, the median, and Q_3 together contain no information about the endpoints (e.g., tails) of the data, a fuller summary of the shape of a distribution can be obtained by providing the lowest and highest data values as well. This is known as the *five-number summary*. The **five-number summary** of a distribution consists of the median (Q_2), the quartiles Q_1 and Q_3, and the smallest and largest individual observations, written in the order of *Minimum*, Q_1, *Median*, Q_3, *Maximum*.

Boxplots are a popular way of visualizing a distribution. A boxplot incorporates the five-number summary as follows:

- Typically, the ends of the box are at the quartiles so that the box length is the interquartile range.
- The median is marked by a line within the box.
- Two lines (called *whiskers*) outside the box extend to the smallest (*Minimum*) and largest (*Maximum*) observations.

When dealing with a moderate number of observations, it is worthwhile to plot potential outliers individually. To do this in a boxplot, the whiskers are extended to the extreme low and high observations *only if* these values are less than $1.5 \times IQR$ beyond the quartiles. Otherwise, the whiskers terminate at the most extreme observations occurring within $1.5 \times IQR$ of the quartiles. The remaining cases are plotted individually. Boxplots can be used in the comparisons of several sets of compatible data.

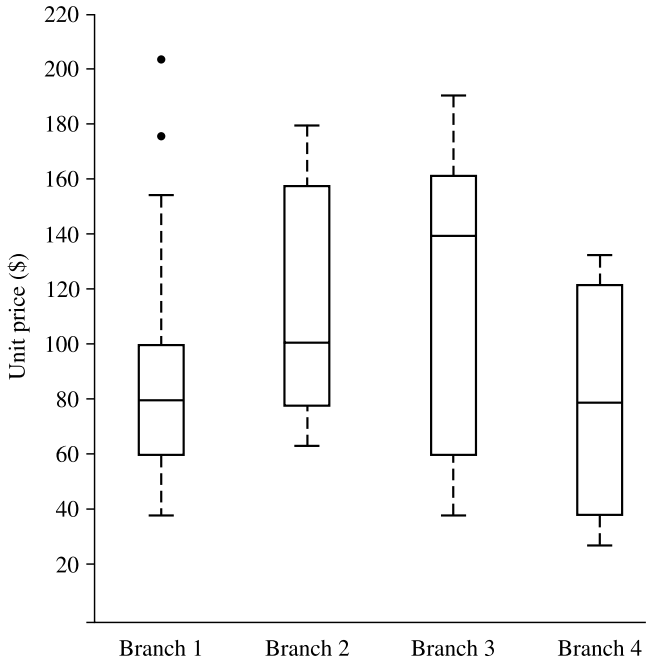


FIGURE 2.3

Boxplot for the unit price data for items sold at four branches of an online store during a given time period.

Example 2.11. Boxplot. Fig. 2.3 shows boxplots for unit price data for items sold at four branches of an online store during a given time period. For branch 1, we see that the median price of items sold is \$80, Q_1 is \$60, and Q_3 is \$100. Notice that two outlying observations for this branch were plotted individually, as their values of 175 and 202 are more than 1.5 times the *IQR* here of 40. □

Variance and standard deviation

Variance and standard deviation are measures of data dispersion. They indicate how spread out a data distribution is. A low standard deviation means that the data observations tend to be very close to the mean, whereas a high standard deviation indicates that the data are spread out over a large range of values.

The **variance** of N observations, $x_1, x_2, \ldots, x_N$ (when N is large), for a numeric attribute X is

$$\sigma^2 = \frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})^2 = \left(\frac{1}{N}\sum_{i=1}^{N}x_i^2\right) - \bar{x}^2, \tag{2.6}$$

where $\bar{x}$ is the mean value of the observations, as defined in Eq. (2.1). The **standard deviation**, σ, of the observations is the square root of the variance, σ^2.

Example 2.12. Variance and standard deviation. In Example 2.6, we found $\bar{x} = \$58{,}000$ using Eq. (2.1) for the mean. To determine the variance and standard deviation of the data from that example, we set $N = 12$ and use Eq. (2.6) to obtain

$$\begin{aligned}\sigma^2 &= \frac{1}{12}(30^2 + 36^2 + 47^2 \ldots + 110^2) - 58^2 \\ &\approx 379.17 \\ \sigma &\approx \sqrt{379.17} \approx 19.47.\end{aligned}$$

□

The basic properties of the standard deviation, σ, as a measure of spread are as follows:

- σ measures spread about the mean and should be considered only when the mean is chosen as the measure of center.
- $\sigma = 0$ only when there is no spread, that is, when all observations have the same value. Otherwise, $\sigma > 0$.

Importantly, an observation is unlikely to be more than several standard deviations away from the mean. Mathematically, using Chebyshev's inequality, it can be shown that at least $\left(1 - \frac{1}{k^2}\right) \times 100\%$ of the observations are no more than k standard deviations from the mean. Therefore, the standard deviation is a good indicator of the spread of a data set.

The computation of the variance and standard deviation is scalable in large data sets.

2.2.3 Covariance and correlation analysis

Covariance of numeric data

In probability theory and statistics, correlation and covariance are two similar measures for assessing how much two attributes change together. Consider two numeric attributes A and B and a set of n real-valued observations $\{(a_1, b_1), \ldots, (a_n, b_n)\}$. The mean values of A and B, respectively, are also known as the **expected values** on A and B, that is,

$$E(A) = \bar{A} = \frac{\sum_{i=1}^{n} a_i}{n}$$

and

$$E(B) = \bar{B} = \frac{\sum_{i=1}^{n} b_i}{n}.$$

The **covariance** between A and B is defined as

$$Cov(A, B) = E((A - \bar{A})(B - \bar{B})) = \frac{\sum_{i=1}^{n}(a_i - \bar{A})(b_i - \bar{B})}{n}. \tag{2.7}$$

Mathematically, it can also be shown that

$$Cov(A, B) = E(A \cdot B) - \bar{A}\bar{B}. \tag{2.8}$$

This equation may simplify calculations.

For two attributes A and B that tend to change together, if a value a_i of A is larger than $\bar{A}$ (the expected value of A), then the corresponding value of b_i of attribute B is likely to be larger than $\bar{B}$ (the expected value of B). Therefore the covariance between A and B is *positive*. On the other hand, if one of the attributes tends to be above its expected value when the other attribute is below its expected value, then the covariance of A and B is *negative*.

If A and B are *independent* (i.e., they do not have correlation), then $E(A \cdot B) = E(A) \cdot E(B)$. Therefore the covariance is $Cov(A, B) = E(A \cdot B) - \bar{A}\bar{B} = E(A) \cdot E(B) - \bar{A}\bar{B} = 0$. However, the converse is not true. Some pairs of random variables (attributes) may have a covariance of 0 but are not independent. Only under some additional assumptions (e.g., the data follow multivariate normal distributions) does a covariance of 0 imply independence.

Example 2.13. Covariance analysis of numeric attributes. Consider Table 2.1, which presents a simplified example of stock prices observed at five time points for *AllElectronics* and *HighTech*, a high-tech company. If the stocks are affected by the same industry trends, will their prices rise or fall together?

$$E(AllElectronics) = \frac{6+5+4+3+2}{5} = \frac{20}{5} = \$4$$

and

$$E(HighTech) = \frac{20+10+14+5+5}{5} = \frac{54}{5} = \$10.80.$$

Thus, using Eq. (2.7), we compute

$$\begin{aligned} Cov(AllElectroncis, HighTech) &= \frac{6 \times 20 + 5 \times 10 + 4 \times 14 + 3 \times 5 + 2 \times 5}{5} - 4 \times 10.80 \\ &= 50.2 - 43.2 = 7. \end{aligned}$$

Therefore, given the positive covariance we can say that stock prices for both companies rise together. □

Variance is a special case of covariance, where the two attributes are identical (i.e., the covariance of an attribute with itself).

Table 2.1 Stock prices for *AllElectronics* and *HighTech*.

Time point	AllElectronics	HighTech
t1	6	20
t2	5	10
t3	4	14
t4	3	5
t5	2	5

Correlation coefficient for numeric data

For numeric attributes, we can evaluate the correlation between two attributes, A and B, by computing the **correlation coefficient** (also known as **Pearson's product moment coefficient**, named after its inventer, Karl Pearson). This is

$$r_{A,B} = \frac{\sum_{i=1}^{n}(a_i - \bar{A})(b_i - \bar{B})}{n\sigma_A\sigma_B} = \frac{\sum_{i=1}^{n}(a_i b_i) - n\bar{A}\bar{B}}{n\sigma_A\sigma_B}, \tag{2.9}$$

where n is the number of tuples, a_i and b_i are the respective values of A and B in tuple i, $\bar{A}$ and $\bar{B}$ are the respective mean values of A and B, σ_A and σ_B are the respective standard deviations of A and B (as defined in Section 2.2.2), and $\Sigma(a_i b_i)$ is the sum of the AB cross-product (i.e., for each tuple, the value for A is multiplied by the value for B in that tuple). Note that $-1 \leq r_{A,B} \leq +1$. If $r_{A,B}$ is greater than 0, then A and B are *positively correlated*, meaning that the values of A increase as the values of B increase. The higher the value, the stronger the correlation (i.e., the more each attribute implies the other). Hence, a higher value may indicate that A (or B) may be removed as a redundancy.

If the resulting value is equal to 0, then A and B are *independent*, and there is no correlation between them. If the resulting value is less than 0, then A and B are *negatively correlated*, where the values of one attribute increase as the values of the other attribute decrease. This means that each attribute discourages the other. Scatter plots can also be used to view correlations between attributes (Section 2.2.3). For example, Fig. 2.8's scatter plots, respectively, show positively correlated data and negatively correlated data, whereas Fig. 2.9 displays uncorrelated data.

Note that correlation does not imply causality. That is, if A and B are correlated, this does not necessarily imply that A causes B or that B causes A. For example, in analyzing a demographic database, we may find that attributes representing the number of hospitals and the number of car thefts in a region are correlated. This does not mean that one causes the other. Both are actually causally linked to a third attribute, namely, *population*.

χ^2 correlation test for nominal data

For nominal data, a correlation relationship between two attributes, A and B, can be discovered by a χ^2 (**chi-square**) test. Suppose A has c distinct values, namely, $a_1, a_2, \ldots a_c$, and B has r distinct values, namely, $b_1, b_2, \ldots b_r$. The data tuples described by A and B can be shown as a **contingency table**, with the c values of A making up the columns and the r values of B making up the rows. Let (A_i, B_j) denote the joint event that attribute A takes on value a_i and attribute B takes on value b_j, that is, where $(A = a_i, B = b_j)$. Each and every possible (A_i, B_j) joint event has its own cell (or slot) in the table. The χ^2 value (also known as the *Pearson* χ^2 *statistic*) is computed as

$$\chi^2 = \sum_{i=1}^{c}\sum_{j=1}^{r}\frac{(o_{ij} - e_{ij})^2}{e_{ij}}, \tag{2.10}$$

where o_{ij} is the *observed frequency* (i.e., actual count) of the joint event (A_i, B_j) and e_{ij} is the *expected frequency* of (A_i, B_j), which can be computed as

$$e_{ij} = \frac{count(A = a_i) \times count(B = b_j)}{n}, \tag{2.11}$$

where n is the number of data tuples, $count(A = a_i)$ is the number of tuples having value a_i for A, and $count(B = b_j)$ is the number of tuples having value b_j for B. The sum in Eq. (2.10) is computed over all of the $r \times c$ cells. Note that the cells that contribute the most to the χ^2 value are those for which the actual count is very different from that expected.

The χ^2 statistic tests the hypothesis that A and B are *independent*, that is, there is no correlation between them. The test is based on a significance level, with $(r-1) \times (c-1)$ degrees of freedom. We illustrate the use of this statistic in Example 2.14. If the hypothesis can be rejected, then we say that A and B are statistically correlated.

Example 2.14. Correlation analysis of nominal attributes using χ^2. Suppose that a group of 1500 people was surveyed. The gender of each person was noted. Each person was polled as to whether his or her preferred type of reading material was fiction or nonfiction. Thus, we have two attributes, *gender* and *preferred_reading*. The observed frequency (or count) of each possible joint event is summarized in the contingency table shown in Table 2.2, where the numbers in parentheses are the expected frequencies. The expected frequencies are calculated based on the data distribution for both attributes using Eq. (2.11).

Using Eq. (2.11), we can verify the expected frequencies for each cell. For example, the expected frequency for the cell (*male, fiction*) is

$$e_{11} = \frac{count(male) \times count(fiction)}{n} = \frac{300 \times 450}{1500} = 90,$$

and so on. Notice that in any row, the sum of the expected frequencies must equal the total observed frequency for that row, and the sum of the expected frequencies in any column must also equal the total observed frequency for that column.

Using Eq. (2.10) for χ^2 computation, we get

$$\begin{aligned}\chi^2 &= \frac{(250-90)^2}{90} + \frac{(50-210)^2}{210} + \frac{(200-360)^2}{360} + \frac{(1000-840)^2}{840} \\ &= 284.44 + 121.90 + 71.11 + 30.48 = 507.93.\end{aligned}$$

For this 2×2 table, the degrees of freedom are $(2-1) \times (2-1) = 1$. For 1 degree of freedom, the χ^2 value needed to reject the hypothesis at the 0.001 significance level is 10.828 (taken from the table of upper percentage points of the χ^2 distribution, typically available from any textbook on statistics). Since our computed value is above this, we can reject the hypothesis that *gender* and *preferred_reading*

Table 2.2 Example 2.1's 2 × 2 contingency table data.

	Male	Female	Total
fiction	250 (90)	200 (360)	450
non_fiction	50 (210)	1000 (840)	1050
Total	300	1200	1500

Note: Are gender and preferred_reading correlated?

are independent and conclude that the two attributes are (strongly) correlated for the given group of people. □

2.2.4 Graphic displays of basic statistics of data

In this section, we study graphic displays of basic statistical descriptions. These include *quantile plots*, *quantile-quantile plots*, *histograms*, and *scatter plots*. Such graphs are helpful for the visual inspection of data, which is useful for data preprocessing. The first three of these show univariate distributions (i.e., data for one attribute), whereas scatter plots show bivariate distributions (i.e., involving two attributes).

Quantile plot

A **quantile plot** is a simple and effective way to have a first look at a univariate data distribution. First, it displays all of the data for the given attribute (allowing a user to assess both the overall behavior and unusual occurrences). Second, it plots quantile information (see Section 2.2.2). Let x_i, for $i = 1$ to N, be the data sorted in ascending order so that x_1 is the smallest observation and x_N is the largest for some ordinal or numeric attribute X. Each observation, x_i, is paired with a percentage, f_i, which indicates that approximately $f_i \times 100\%$ of the data are below the value, x_i. We say "approximately" because there may not be a value with exactly a fraction, f_i, of the data below x_i. Note that the 0.25 quantile corresponds to quartile Q_1, the 0.50 quantile is the median, and the 0.75 quantile is Q_3.

Let

$$f_i = \frac{i - 0.5}{N}. \tag{2.12}$$

These numbers increase in equal steps of $1/N$, ranging from $\frac{1}{2N}$ (which is slightly above 0) to $1 - \frac{1}{2N}$ (which is slightly below 1). On a quantile plot, x_i is graphed against f_i. This allows us to compare different distributions based on their quantiles. For example, given the quantile plots of sales data for two different time periods, we can compare their Q_1, median, Q_3, and other f_i values at a glance.

Example 2.15. Quantile plot. Fig. 2.4 shows a quantile plot for the *unit price* data of Table 2.3. □

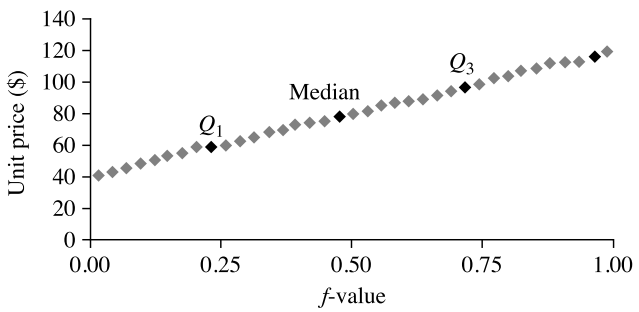


FIGURE 2.4

A quantile plot for the unit price data of Table 2.3.

Table 2.3 A set of unit price data for items sold at a branch of the online store.

Unit price ($)	Count of items sold
40	275
43	300
47	250
⋮	⋮
74	360
75	515
78	540
⋮	⋮
115	320
117	270
120	350

Quantile-quantile plot

A **quantile-quantile plot**, or **q-q plot**, graphs the quantiles of one univariate distribution against the corresponding quantiles of another. It is a powerful visualization tool in that it allows the user to view whether there is a shift in going from one distribution to another.

Suppose that we have two sets of observations for the attribute or variable *unit price*, taken from two different branch locations. Let $x_1, \ldots, x_N$ be the data from the first branch, and $y_1, \ldots, y_M$ be the data from the second, where each data set is sorted in ascending order. If $M = N$ (i.e., the number of points in each set is the same), then we simply plot y_i against x_i, where y_i and x_i are both $(i - 0.5)/N$ quantiles of their respective data sets. If $M < N$ (i.e., the second branch has fewer observations than the first), there can be only M points on the q-q plot. Here, y_i is the $(i - 0.5)/M$ quantile of the y data, which is plotted against the $(i - 0.5)/M$ quantile of the x data. This computation typically involves interpolation.

Example 2.16. Quantile-quantile plot. Fig. 2.5 shows a quantile-quantile plot for *unit price* data of items sold at two branches of the online store during a given time period. Each point corresponds to the same quantile for each data set and shows the unit price of items sold at branch 1 vs. branch 2 for that quantile. (To aid comparison, the straight line represents the case where, for each given quantile, the unit price at each branch is the same. The darker points correspond to the data for Q_1, the median, and Q_3, respectively.)

We see, for example, that at Q_1, the unit price of items sold at branch 1 was slightly less than that at branch 2. In other words, 25% of items sold at branch 1 were less than or equal to $60, whereas 25% of items sold at branch 2 were less than or equal to $64. At the 50th percentile (marked by the median, which is also Q_2), we see that 50% of items sold at branch 1 were less than $78, whereas 50% of items at branch 2 were less than $85. In general, we note that there is a shift in the distribution of branch 1 with respect to branch 2 in that the unit prices of items sold at branch 1 tend to be lower than those at branch 2. □

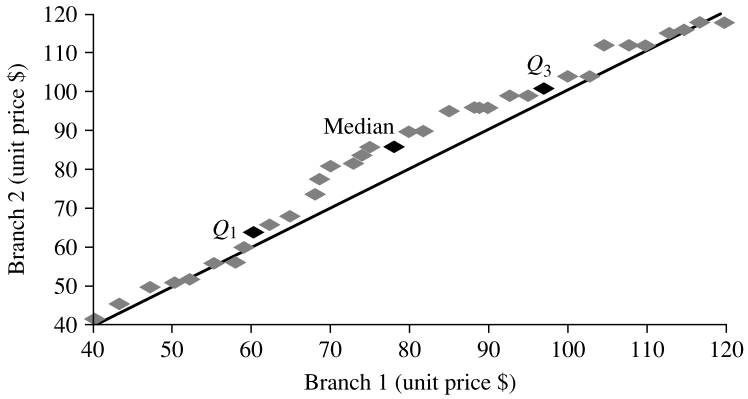


FIGURE 2.5

A q-q plot for unit price data from two branches of the online store.

Histograms

Histograms (or **frequency histograms**) are at least a century old and are widely used. "Histos" means pole or mast, and "gram" means chart, so a histogram is a chart of poles. Plotting histograms is a graphical method for summarizing the distribution of a given attribute, X. According to the number of poles desired in the chart, the range of values for X is partitioned into a set of disjoint consecutive subranges. The subranges, referred to as *buckets* or *bins*, are disjoint subsets of the data distribution for X. The range of a bucket is known as the **width**. Typically, the buckets are of equal width. For example, a *price* attribute with a value range of \$1–\$200 (rounded up to the nearest dollar) can be partitioned into subranges 1–20, 21–40, 41–60, and so on. For each subrange, a bar is drawn with a height that represents the total count of items observed within the subrange.

Please note that histogram is different from another popularly used graph representation called **bar chart**. Bar chart uses a set of bars (often separated with space) with X representing a set of categorical data, such as *automobile_model* or *item_type*, and the height of the bar (column) indicates the size of the group defined by the categories. On the other hand, histogram plots quantitative data with a range of X values grouped into bins or intervals. Histograms are used to show distributions (along X axis) while bar charts are used to compare categories. It is always appropriate to talk about the skewness of a histogram; that is, the tendency of the observations to fall more on the low end or the high end of the X axis. However, bar chart's X axis does not have a low end or a high end; because the labels on the X axis are categorical—not quantitative. Thus, bars can be reordered in bar charts but not in histograms.

Example 2.17. Histogram. Fig. 2.6 shows a histogram for a data set on research award distribution for a region, where buckets (or bins) are defined by equal-width ranges representing \$1000 increments, and the frequency is the number of research awards in the corresponding buckets. □

Although histograms are widely used, they may not be as effective as the quantile plot, q-q plot, and boxplot methods in comparing groups of univariate observations.

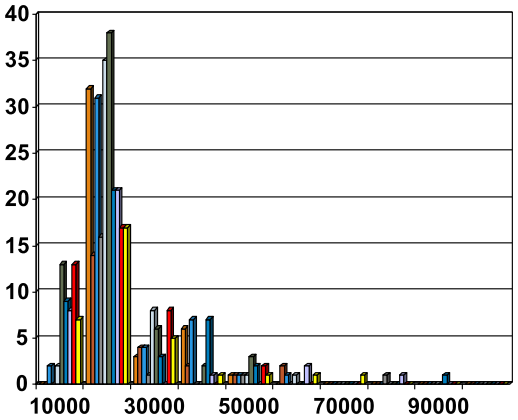


FIGURE 2.6

A histogram on research award distribution for a region.

Scatter plots and data correlation

A **scatter plot** is one of the most effective graphical methods for determining whether there appears to be a relationship, pattern, or trend between two numeric attributes. To construct a scatter plot, each pair of values is treated as a pair of coordinates in an algebraic sense and plotted as points in the plane. Fig. 2.7 shows a scatter plot for the set of data in Table 2.3.

The scatter plot is a useful method for providing a first look at bivariate data to see clusters of points and outliers, or to explore the possibility of correlation relationships. Two attributes, X and Y, are **correlated** if the knowledge of one attribute enables to predict the other with some accuracy. Correlations can be positive, negative, or null (uncorrelated). Fig. 2.8 shows examples of positive and negative correlations between two attributes.

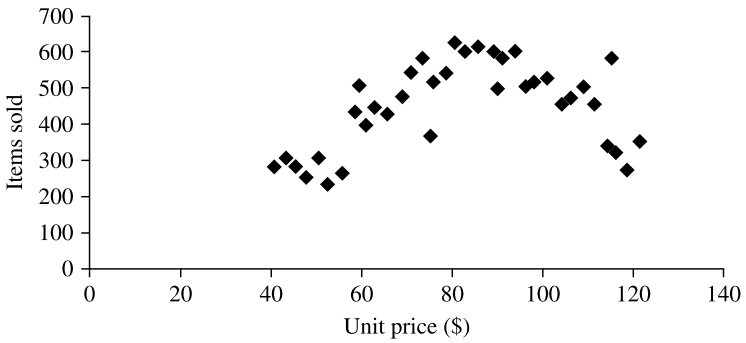


FIGURE 2.7

A scatter plot for Table 2.3 data set.

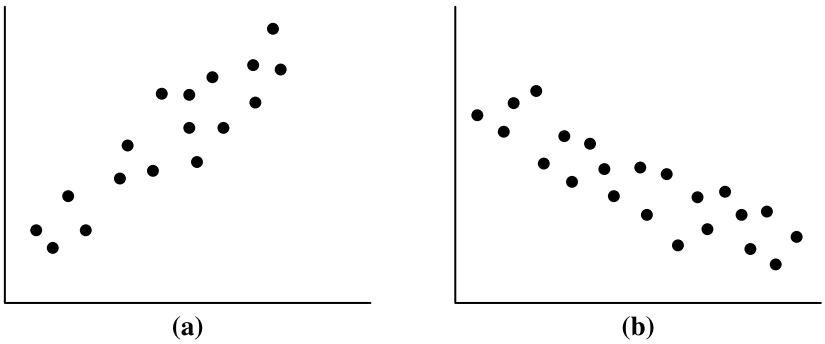


FIGURE 2.8

Scatter plots can be used to find (a) positive or (b) negative correlations between attributes.

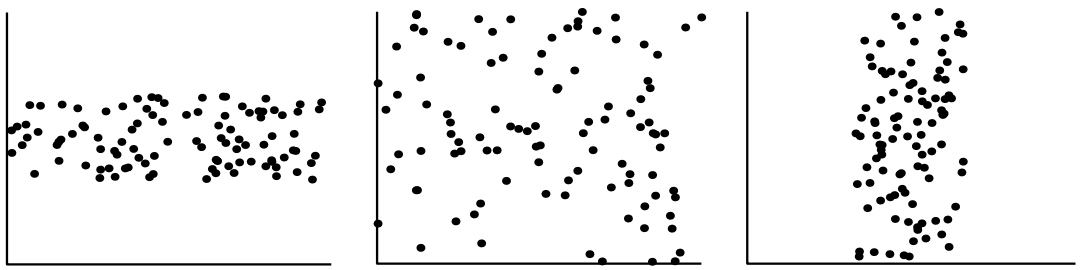


FIGURE 2.9

Three cases where there is no observed correlation between the two plotted attributes in each of the data sets.

If the plotted points pattern slopes from lower left to upper right, this means that the values of X increase as the values of Y increase, suggesting a *positive correlation* (Fig. 2.8a). If the pattern of plotted points slopes from upper left to lower right, the values of X increase as the values of Y decrease, suggesting a *negative correlation* (Fig. 2.8b). A line of best fit can be drawn to study the correlation between the variables. Statistical tests for correlation are introduced in Appendix A.

Fig. 2.9 shows three cases for which there is no correlation relationship between the two attributes in each of the given data sets. Scatter plots can also be extended to n attributes, resulting in a *scatter-plot matrix*.

In summary, basic data descriptions (e.g., measures of central tendency and measures of dispersion) and graphic statistical displays (e.g., quantile plots, histograms, and scatter plots) provide valuable insight into the overall behavior of your data. By helping to identify noise and outliers, they are especially useful for data cleaning.

2.3 Similarity and distance measures

In data mining applications, such as clustering, outlier analysis, and nearest-neighbor classification, we need ways to assess how alike or unalike objects are in comparison to one another. For example, a store may want to search for clusters of *customer* objects, resulting in groups of customers with similar characteristics (e.g., similar income, area of residence, and age). Such information can then be used for marketing. A **cluster** is a collection of data objects such that the objects within a cluster are *similar* to one another and *dissimilar* to the objects in other clusters. Outlier analysis also employs clustering-based techniques to identify potential outliers as objects that are highly dissimilar to others. Knowledge of object similarities can also be used in nearest-neighbor classification schemes where a given object (e.g., a *patient*) is assigned a class label (relating to, say, a *diagnosis*) based on its similarity toward other objects in the model.

This section presents similarity and dissimilarity measures, which are referred to as measures of *proximity*. Similarity and dissimilarity are related. A similarity measure for two objects, i and j, will typically return value 0 if the objects are completely unalike. The higher the similarity value, the greater the similarity between objects. (Typically, a value of 1 indicates complete similarity, that is, the objects are identical.) A dissimilarity measure works the opposite way. It returns a value of 0 if the objects are the same (and therefore, far from being dissimilar). The higher the dissimilarity value, the more dissimilar the two objects are.

In Section 2.3.1 we present two data structures that are commonly used in the above types of applications: the *data matrix* (used to store the data objects) and the *dissimilarity matrix* (used to store dissimilarity values for pairs of objects). We also switch to a different notation for data objects than previously used in this chapter since now we are dealing with objects described by more than one attribute. We then discuss how object dissimilarity can be computed for objects described by *nominal* attributes (Section 2.3.2), by *binary* attributes (Section 2.3.3), by *numeric* attributes (Section 2.3.4), by *ordinal* attributes (Section 2.3.5), or by combinations of these attribute types (Section 2.3.6). Section 2.3.7 provides similarity measures for very long and sparse data vectors, such as term-frequency vectors representing documents in information retrieval. Finally, Section 2.3.8 discusses how to measure the difference between two probability distributions over the same variable x, and introduces a measure, called the *Kullback-Leibler divergence*, or simply, the *KL divergence*, which has been popularly used in the data mining literature.

Knowing how to compute dissimilarity is useful in studying attributes and will also be referenced in later topics on clustering (Chapters 8 and 9), outlier analysis (Chapter 11), and nearest-neighbor classification (Chapter 6).

2.3.1 Data matrix vs. dissimilarity matrix

In Section 2.2, we looked at ways of studying the central tendency, dispersion, and spread of observed values for some attribute X. Our objects there were one-dimensional, that is, described by a single attribute. In this section, we talk about objects described by *multiple* attributes. Therefore we need a change in notation. Suppose that we have n objects (e.g., persons, items, or courses) described by p attributes (also called *measurements* or *features*, such as age, height, weight, or gender). The objects are $x_1 = (x_{11}, x_{12}, \ldots, x_{1p})$, $x_2 = (x_{21}, x_{22}, \ldots, x_{2p})$, and so on, where x_{ij} is the value for object x_i of the jth attribute. For brevity, we hereafter refer to object x_i as object i. The objects may be tuples in a relational database and are also referred to as *data samples* or *feature vectors*.

Main memory-based clustering and nearest-neighbor algorithms typically operate on either of the following two data structures:

- **Data matrix** (or *object-by-attribute structure*): This structure stores the n data objects in the form of a relational table or an n-by-p matrix (n objects $\times p$ attributes):

$$\begin{bmatrix} x_{11} & \cdots & x_{1f} & \cdots & x_{1p} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ x_{i1} & \cdots & x_{if} & \cdots & x_{ip} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ x_{n1} & \cdots & x_{nf} & \cdots & x_{np} \end{bmatrix}. \tag{2.13}$$

  Each row corresponds to an object. As part of our notation, we may use f to index through the p attributes.
- **Dissimilarity matrix** (or *object-by-object structure*): This structure stores a collection of proximities that are available for all pairs of n objects. It is often represented by an n-by-n table:

$$\begin{bmatrix} 0 & & & & \\ d(2,1) & 0 & & & \\ d(3,1) & d(3,2) & 0 & & \\ \vdots & \vdots & \vdots & & \\ d(n,1) & d(n,2) & \cdots & \cdots & 0 \end{bmatrix}, \tag{2.14}$$

  where $d(i, j)$ is the measured **dissimilarity** or "difference" between objects i and j. In general, $d(i, j)$ is a nonnegative number that is close to 0 when objects i and j are highly similar or "near" each other, and becomes larger the more they differ. Note that $d(i, i) = 0$; that is, the difference between an object and itself is 0. Furthermore, $d(i, j) = d(j, i)$. (For readability, we do not show the $d(j, i)$ entries since the matrix is symmetric.) Measures of dissimilarity are discussed throughout the remainder of this chapter.

Measures of similarity can often be expressed as a function of measures of dissimilarity. For example, for nominal data,

$$sim(i, j) = 1 - d(i, j), \tag{2.15}$$

where $sim(i, j)$ is the similarity between objects i and j. Throughout the rest of this chapter, we will also comment on measures of similarity.

A data matrix is made up of two entities or "things," namely rows (for objects) and columns (for attributes). Therefore, the data matrix is often called a **two-mode** matrix. The dissimilarity matrix contains one kind of entity (dissimilarities) and so is called a **one-mode** matrix. Many clustering and nearest-neighbor algorithms operate on a dissimilarity matrix. Data in the form of a data matrix can be transformed into a dissimilarity matrix before applying such algorithms.

2.3.2 Proximity measures for nominal attributes

A nominal attribute can take on two or more states (Section 2.1.1). For example, *map_color* is a nominal attribute that may have, say, five states: *red*, *yellow*, *green*, *pink*, and *blue*.

Let the number of states of a nominal attribute be M. The states can be denoted by letters, symbols, or a set of integers, such as $1, 2, \ldots, M$. Notice that such integers are used just for data handling and do not represent any specific ordering.

"How is dissimilarity computed between objects described by nominal attributes?" The dissimilarity between two objects i and j can be computed based on the ratio of mismatches:

$$d(i, j) = \frac{p - m}{p}, \tag{2.16}$$

where m is the number of *matches* (i.e., the number of attributes for which i and j are in the same state), and p is the total number of attributes describing the objects. Weights can be assigned to increase the effect of m or to assign greater weight to the matches in attributes having a larger number of states.

Example 2.18. Dissimilarity between nominal attributes. Suppose that we have the sample data of Table 2.4, except that only the *object-identifier* and the attribute *test-1* are available, where *test-1* is nominal. (We will use *test-2* and *test-3* in later examples.) Let's compute the dissimilarity matrix Eq. (2.14), that is,

$$\begin{bmatrix} 0 & & & \\ d(2,1) & 0 & & \\ d(3,1) & d(3,2) & 0 & \\ d(4,1) & d(4,2) & d(4,3) & 0 \end{bmatrix}.$$

Since here we have one nominal attribute, *test-1*, we set $p = 1$ in Eq. (2.16) so that $d(i, j)$ evaluates to 0 if objects i and j match, and 1 if the objects differ. Thus, we get

$$\begin{bmatrix} 0 & & & \\ 1 & 0 & & \\ 1 & 1 & 0 & \\ 0 & 1 & 1 & 0 \end{bmatrix}.$$

From this, we see that all objects are dissimilar except objects 1 and 4 (i.e., $d(4, 1) = 0$). □

Alternatively, similarity can be computed as

$$sim(i, j) = 1 - d(i, j) = \frac{m}{p}. \tag{2.17}$$

Table 2.4 A sample data table containing attributes of mixed types.

Object Identifier	Test-1 (nominal)	Test-2 (ordinal)	Test-3 (numeric)
1	code A	excellent	45
2	code B	fair	22
3	code C	good	64
4	code A	excellent	28

Proximity between objects described by nominal attributes can be computed using an alternative encoding scheme. Nominal attributes can be encoded using asymmetric binary attributes by creating a new binary attribute for each of the M states. For an object with a given state value, the binary attribute representing that state is set to 1, whereas the remaining binary attributes are set to 0. For example, to encode the nominal attribute *map_color*, a binary attribute can be created for each of the five colors previously listed. For an object having the color *yellow*, the *yellow* attribute is set to 1, whereas the remaining four attributes are set to 0. Proximity measures for this form of encoding can be calculated using the methods discussed in the next subsection.

2.3.3 Proximity measures for binary attributes

Let's look at dissimilarity and similarity measures for objects described by either *symmetric* or *asymmetric binary attributes*.

Recall that a binary attribute has only one of two states, 0 and 1, where 0 means that the attribute is absent, and 1 means that it is present (Section 2.1.2). Given the attribute *smoker* describing a patient, for instance, 1 indicates that the patient smokes, whereas 0 indicates that the patient does not. Treating binary attributes as if they are other numeric attributes can be misleading. Therefore methods specific to binary data are necessary for computing dissimilarity.

"So, how can we compute the dissimilarity between two binary attributes?" One approach involves computing a dissimilarity matrix from the given binary data. If all binary attributes are thought of as having the same weight, we have the 2×2 contingency table of Table 2.5, where q is the number of attributes that equal 1 for both objects i and j, r is the number of attributes that equal 1 for object i but equal 0 for object j, s is the number of attributes that equal 0 for object i but equal 1 for object j, and t is the number of attributes that equal 0 for both objects i and j. The total number of attributes is p, where $p = q + r + s + t$.

Recall that for symmetric binary attributes, each state is equally valuable. Dissimilarity that is based on symmetric binary attributes is called **symmetric binary dissimilarity**. If objects i and j are described by symmetric binary attributes, then the dissimilarity between i and j is

$$d(i, j) = \frac{r + s}{q + r + s + t}. \tag{2.18}$$

For asymmetric binary attributes, the two states are not equally important, such as the *positive* (1) and *negative* (0) outcomes of a disease test. Given two asymmetric binary attributes, the agreement of two 1s (a positive match) is then considered more significant than that of two 0s (a negative match). Therefore such binary attributes are often considered "monary" (having one state). The dissimilarity

Table 2.5 Contingency table for binary attributes.

		Object j		
		1	0	sum
Object i	1	q	r	$q + r$
	0	s	t	$s + t$
	sum	$q + s$	$r + t$	p

based on these attributes is called **asymmetric binary dissimilarity**, where the number of negative matches, t, is considered unimportant and is thus ignored in the following computation:

$$d(i, j) = \frac{r + s}{q + r + s}. \tag{2.19}$$

Complementarily, we can measure the difference between two binary attributes based on the notion of similarity instead of dissimilarity. For example, the **asymmetric binary similarity** between the objects i and j can be computed as

$$sim(i, j) = \frac{q}{q + r + s} = 1 - d(i, j). \tag{2.20}$$

The coefficient $sim(i, j)$ of Eq. (2.20) is called the **Jaccard coefficient** and is popularly referenced in the literature.

When both symmetric and asymmetric binary attributes occur in the same data set, the approach for mixed attributes described in Section 2.3.6 can be applied.

Example 2.19. Dissimilarity between binary attributes. Suppose that a patient record table (Table 2.6) contains the attributes *name, gender, fever, cough, test-1, test-2, test-3*, and *test-4*, where *name* is an object identifier, *gender* is a symmetric binary attribute, and the remaining attributes are asymmetric binary. □

For asymmetric binary attribute values, let the values Y (*yes*) and P (*positive*) be set to 1, and the value N (*no* or *negative*) be set to 0. Suppose that the distance between objects (patients) is computed based only on the asymmetric binary attributes. According to Eq. (2.19), the distance between each pair of the three patients—Jack, Mary, and Jim—is

$$d(Jack, Jim) = \frac{1 + 1}{1 + 1 + 1} = 0.67,$$
$$d(Jack, Mary) = \frac{0 + 1}{2 + 0 + 1} = 0.33,$$
$$d(Jim, Mary) = \frac{1 + 2}{1 + 1 + 2} = 0.75.$$

These measurements suggest that Jim and Mary are unlikely to have a similar disease because they have the highest dissimilarity value among the three pairs. Of the three patients, Jack and Mary are the most likely to have a similar disease.

Table 2.6 Relational table where patients are described by binary attributes.

Name	Gender	Fever	Cough	Test-1	Test-2	Test-3	Test-4
Jack	M	Y	N	P	N	N	N
Jim	M	Y	Y	N	N	N	N
Mary	F	Y	N	P	N	P	N
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

2.3.4 Dissimilarity of numeric data: Minkowski distance

In this section, we describe distance measures that are commonly used for computing the dissimilarity of objects described by numeric attributes. These measures include the *Euclidean, Manhattan*, and *Minkowski distances*.

In some cases, the data are normalized before applying distance calculations. This involves transforming the data to fall within a smaller or common range, such as $[-1.0, 1.0]$ or $[0.0, 1.0]$. Consider a *height* attribute, for example, which could be measured in either meters or inches. In general, expressing an attribute in smaller units will lead to a larger range for that attribute and thus tend to give such attributes greater effect or "weight." Normalizing the data attempts to give all attributes an equal weight. It may or may not be useful in a particular application. Methods for normalizing data are discussed in detail in Section 2.5 on data transformation.

The most popular distance measure is **Euclidean distance** (i.e., straight line or "as the crow flies"). Let $i = (x_{i1}, x_{i2}, \ldots, x_{ip})$ and $j = (x_{j1}, x_{j2}, \ldots, x_{jp})$ be two objects described by p numeric attributes. The Euclidean distance between objects i and j is defined as

$$d(i, j) = \sqrt{(x_{i1} - x_{j1})^2 + (x_{i2} - x_{j2})^2 + \cdots + (x_{ip} - x_{jp})^2}. \tag{2.21}$$

Another well-known measure is the **Manhattan (or city block) distance**, named so because it is the distance in blocks between any two points in a city (such as 2 blocks down and 3 blocks over for a total of 5 blocks). It is defined as

$$d(i, j) = |x_{i1} - x_{j1}| + |x_{i2} - x_{j2}| + \cdots + |x_{ip} - x_{jp}|. \tag{2.22}$$

Both the Euclidean and the Manhattan distance satisfy the following mathematical properties:

Nonnegativity: $d(i, j) \geq 0$: Distance is a nonnegative number.
Identity of indiscernibles: $d(i, i) = 0$: The distance of an object to itself is 0.
Symmetry: $d(i, j) = d(j, i)$: Distance is a symmetric function.
Triangle inequality: $d(i, j) \leq d(i, k) + d(k, j)$: Going directly from object i to object j in space is no more than making a detour over any other object k.

A measure that satisfies these conditions is known as **metric**. Please note that the nonnegativity property is implied by the other three properties.

Example 2.20. Euclidean distance and Manhattan distance. Let $x_1 = (1, 2)$ and $x_2 = (3, 5)$ represent two objects as shown in Fig. 2.10. The Euclidean distance between the two is $\sqrt{2^2 + 3^2} = 3.61$. The Manhattan distance between the two is $2 + 3 = 5$. □

Minkowski distance is a generalization of the Euclidean and Manhattan distances. It is defined as

$$d(i, j) = \sqrt[h]{|x_{i1} - x_{j1}|^h + |x_{i2} - x_{j2}|^h + \cdots + |x_{ip} - x_{jp}|^h}, \tag{2.23}$$

where h is a real number such that $h \geq 1$. (Such a distance is also called L_p **norm** in some literature, where the symbol p refers to our notation of h. We have kept p as the number of attributes to be consistent with the rest of this chapter.) It represents the Manhattan distance when $h = 1$ (i.e., L_1 norm) and Euclidean distance when $h = 2$ (i.e., L_2 norm).

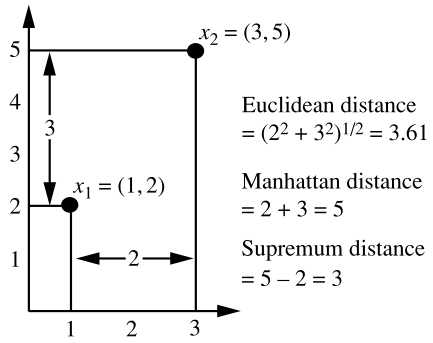


FIGURE 2.10

Euclidean, Manhattan, and supremum distances between two objects.

The **supremum distance** (also referred to as L_{max}, L_∞ **norm**, and the **Chebyshev distance**) is a generalization of the Minkowski distance for $h \to \infty$. To compute it, we find the attribute f that gives the maximum difference in values between the two objects. This difference is the supremum distance, defined more formally as:

$$d(i, j) = \lim_{h \to \infty} \left(\sum_{f=1}^{p} |x_{if} - x_{jf}|^h \right)^{\frac{1}{h}} = \max_{f}^{p} |x_{if} - x_{jf}|. \tag{2.24}$$

The L_∞ norm is also known as the *uniform norm*.

Example 2.21. Supremum distance. Let's use the same two objects, $x_1 = (1, 2)$ and $x_2 = (3, 5)$, as in Fig. 2.10. The second attribute gives the greatest difference between the values for the objects. That is, $max\{(|3 - 1|, |5 - 2|\} = 3$. This is the supremum distance between the two objects. □

If each attribute is assigned a weight according to its perceived importance, the **weighted Euclidean distance** can be computed as

$$d(i, j) = \sqrt{w_1 |x_{i1} - x_{j1}|^2 + w_2 |x_{i2} - x_{j2}|^2 + \cdots + w_m |x_{ip} - x_{jp}|^2}. \tag{2.25}$$

Weighting can also be applied to other distance measures as well.

2.3.5 Proximity measures for ordinal attributes

The values of an ordinal attribute have a meaningful order or ranking about them, yet the magnitude between successive values is unknown (Section 2.1.3). An example includes the sequence *small*, *medium*, *large* for a *size* attribute. Ordinal attributes may also be obtained from the discretization of numeric attributes by splitting the value range into a finite number of categories. These categories are organized into ranks. That is, the range of a numeric attribute can be mapped to an ordinal attribute f having M_f

states. For example, the range of the interval-scaled attribute *temperature* (in Celsius) can be organized into the following states: −30 to −10, −10 to 10, and 10 to 30, representing the categories *cold temperature*, *moderate temperature*, and *warm temperature*, respectively. Let M_f represent the number of possible states that an ordinal attribute can have. These ordered states define the ranking $1, \ldots, M_f$.

"How are ordinal attributes handled?" The treatment of ordinal attributes is quite similar to that of numeric attributes when computing dissimilarity between objects. Suppose that f is an attribute from a set of ordinal attributes describing n objects. The dissimilarity computation with respect to f involves the following steps:

1. The value of f for the ith object is x_{if}, and f has M_f ordered states, representing the ranking $1, \ldots, M_f$. Replace each x_{if} by its corresponding rank, $r_{if} \in \{1, \ldots, M_f\}$.
2. Since each ordinal attribute can have a different number of states, it is often necessary to map the range of each attribute onto [0.0, 1.0] so that each attribute has equal weight. We perform such data normalization by replacing the rank r_{if} of the ith object in the fth attribute by

$$z_{if} = \frac{r_{if} - 1}{M_f - 1}. \tag{2.26}$$

3. Dissimilarity can then be computed using any of the distance measures described in Section 2.3.4 for numeric attributes, using z_{if} to represent the f value for the ith object.

Example 2.22. Dissimilarity between ordinal attributes. Suppose that we have the sample data shown earlier in Table 2.4, except that this time only the *object-identifier* and the continuous ordinal attribute, *test-2*, are available. There are three states for *test-2*: *fair*, *good*, and *excellent*, that is, $M_f = 3$. For step 1, if we replace each value for *test-2* by its rank, the four objects are assigned the ranks 3, 1, 2, and 3, respectively. Step 2 normalizes the ranking by mapping rank 1 to 0.0, rank 2 to 0.5, and rank 3 to 1.0. For step 3, we can use, say, the Euclidean distance defined in Eq. (2.21), which results in the following dissimilarity matrix:

$$\begin{bmatrix} 0 & & & \\ 1.0 & 0 & & \\ 0.5 & 0.5 & 0 & \\ 0 & 1.0 & 0.5 & 0 \end{bmatrix}.$$

Therefore objects 1 and 2 are the most dissimilar, as are objects 2 and 4 (i.e., $d(2, 1) = 1.0$ and $d(4, 2) = 1.0$). This makes intuitive sense since objects 1 and 4 are both *excellent*. Object 2 is *fair*, which is at the opposite end of the range of values for *test-2*. □

Similarity values for ordinal attributes can be interpreted from dissimilarity as $sim(i, j) = 1 - d(i, j)$.

2.3.6 Dissimilarity for attributes of mixed types

Sections 2.3.2 through 2.3.5 discussed how to compute the dissimilarity between objects described by attributes of the same type, where these types may be either *nominal*, *symmetric binary*, *asymmetric binary*, *numeric*, or *ordinal*. However, in many real databases, objects are described by a *mixture* of attribute types. In general, a database can contain all of these attribute types.

"So, how can we compute the dissimilarity between objects of mixed attribute types?" One approach is to group each type of attributes together, performing separate data mining (e.g., clustering) analysis for each type. This is feasible if these analyses derive compatible results. However, in real applications, it is unlikely that a separate analysis per attribute type will generate compatible results.

A more preferable approach is to process all attribute types together, performing a single analysis. One such technique combines the different attributes into a single dissimilarity matrix, bringing all of the meaningful attributes onto a common scale of the interval [0.0, 1.0].

Suppose that the data set contains p attributes of mixed types. The dissimilarity $d(i, j)$ between objects i and j is defined as

$$d(i, j) = \frac{\sum_{f=1}^{p} \delta_{ij}^{(f)} d_{ij}^{(f)}}{\sum_{f=1}^{p} \delta_{ij}^{(f)}}, \tag{2.27}$$

where the indicator $\delta_{ij}^{(f)} = 0$ if either (1) x_{if} or x_{jf} is missing (i.e., there is no measurement of attribute f for object i or object j), or (2) $x_{if} = x_{jf} = 0$ and attribute f is asymmetric binary; otherwise, $\delta_{ij}^{(f)} = 1$. The contribution of attribute f to the dissimilarity between i and j (i.e., $d_{ij}^{(f)}$) is computed dependent on its type:

- If f is numeric: $d_{ij}^{(f)} = \frac{|x_{if} - x_{jf}|}{max_f - min_f}$, where max_f and min_f are the maximum and minimum values of attribute f, respectively;
- If f is nominal or binary: $d_{ij}^{(f)} = 0$ if $x_{if} = x_{jf}$; otherwise, $d_{ij}^{(f)} = 1$; and
- If f is ordinal: compute the ranks r_{if} and $z_{if} = \frac{r_{if} - 1}{M_f - 1}$, and treat z_{if} as numeric.

These steps are identical to what we have already seen for each of the individual attribute types. The only difference is for numeric attributes, where we normalize so that the values map to the interval [0.0, 1.0]. Thus the dissimilarity between objects can be computed even when the attributes describing the objects are of different types.

Example 2.23. Dissimilarity between attributes of mixed types. Let's compute a dissimilarity matrix for the objects in Table 2.4. Now we will consider *all* of the attributes, which are of different types. In Examples 2.18 and 2.22, we worked out the dissimilarity matrices for each of the individual attributes. The procedures we followed for *test-1* (which is nominal) and *test-2* (which is ordinal) are the same as outlined earlier for processing attributes of mixed types. Therefore we can use the dissimilarity matrices obtained for *test-1* and *test-2* later when we compute Eq. (2.27). First, however, we need to compute the dissimilarity matrix for the third attribute, *test-3* (which is numeric). That is, we must compute $d_{ij}^{(3)}$. Following the case for numeric attributes, we let $max_h x_h = 64$ and $min_h x_h = 22$. The difference between the two is used in Eq. (2.27) to normalize the values of the dissimilarity matrix. The resulting dissimilarity matrix for *test-3* is

$$\begin{bmatrix} 0 & & & \\ 0.55 & 0 & & \\ 0.45 & 1.00 & 0 & \\ 0.40 & 0.14 & 0.86 & 0 \end{bmatrix}.$$

We can now use the dissimilarity matrices for the three attributes in our computation of Eq. (2.27). The indicator $\delta_{ij}^{(f)} = 1$ for each of the three attributes, f. We get, for example, $d(3, 1) = \frac{1(1)+1(0.50)+1(0.45)}{3} = 0.65$. The resulting dissimilarity matrix obtained for the data described by the three attributes of mixed types is:

$$\begin{bmatrix} 0 & & & \\ 0.85 & 0 & & \\ 0.65 & 0.83 & 0 & \\ 0.13 & 0.71 & 0.79 & 0 \end{bmatrix}.$$

From Table 2.4, we can intuitively guess that objects 1 and 4 are the most similar, based on their values for *test-1* and *test-2*. This is confirmed by the dissimilarity matrix, where $d(4, 1)$ is the lowest value for any pair of different objects. Similarly, the matrix indicates that objects 1 and 2 are the least similar. □

2.3.7 Cosine similarity

Cosine similarity measures the similarity between two vectors of an inner product space. It is measured by the cosine of the angle between two vectors and determines whether two vectors are pointing in roughly the same direction. It is often used to measure document similarity in text analysis.

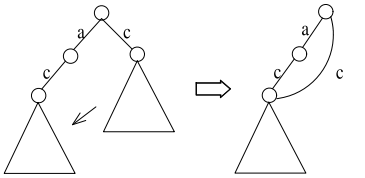
A document can be represented by thousands of attributes, each recording the frequency of a particular word (such as a keyword) or phrase in the document. Thus each document is an object represented by what is called a *term-frequency vector*. For example, in Table 2.7, we see that *Document1* contains five instances of the word *team*, whereas *hockey* occurs three times. The word *coach* is absent from the entire document, as indicated by a count value of 0. Such data can be highly asymmetric.

Term-frequency vectors are typically very long and **sparse** (i.e., they have many 0 values). Applications using such structures include information retrieval, text document clustering, and biological data analysis. The traditional distance measures that we have studied in this chapter do not work well for such sparse numeric data. For example, two term-frequency vectors may have many 0 values in common, meaning that the corresponding documents do not share many words, but this does not make them similar. We need a measure that will focus on the words that the two documents *do* have in common, and the occurrence frequency of such words. In other words, we need a measure for numeric data that ignores zero-matches.

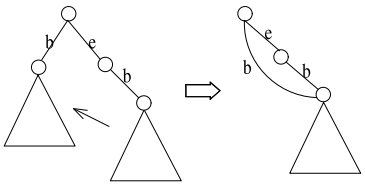
Cosine similarity is a measure of similarity that can be used to compare documents or, say, give a ranking of documents with respect to a given vector of query words. Let $\boldsymbol{x}$ and $\boldsymbol{y}$ be two vectors for

Table 2.7 Document vector or term-frequency vector.

Document	Team	Coach	Hockey	Baseball	Soccer	Penalty	Score	Win	Loss	Season
Document1	5	0	3	0	2	0	0	2	0	0
Document2	3	0	2	0	1	1	0	1	0	1
Document3	0	7	0	2	1	0	0	3	0	0
Document4	0	1	0	0	1	2	2	0	3	0



(a) backward subpattern



(b) backward superpattern

FIGURE 5.10

The pruning of a backward subpattern or a backward superpattern.

redundancy. Similarly, CloSpan will search for *backward superpatterns* for pruning to avoid redundant mining. More concretely, it will stop growing a prefix-based projected databases $S|_\beta$ if it is of the same size as that of the prefix-based projected database $S|_\alpha$ and α and β have substring/superstring relationships.

This is based on a property of sequence databases, called **equivalence of projected databases**, stated as follows: *Two projected sequence databases,* $S|_\alpha = S|_\beta$,[5] $\alpha \sqsubseteq \beta$ *(i.e.,* α *is a subsequence of* β*), are equivalent if and only if the total number of items in* $S|_\alpha$ *is equal to the total number of items in* $S|_\beta$.

Let's examine one such example.

Example 5.17. CloSpan: Pruning redundant projected database. Given a small sequence database, S, shown in Fig. 5.11, with $min\_sup = 2$. The prefix project sequence database of the prefix $\langle af \rangle$ is $(\langle acg \rangle, \langle egb(ac) \rangle, \langle ea \rangle)$ with 12 symbols (including parentheses), and the projected sequence database of the prefix $\langle f \rangle$ is of the same size. Clearly, the two projected databases should be identical and there is no need to mine the latter, the $\langle f \rangle$-projected sequence database. This is understandable since for any sequence s, if its projections on $\langle af \rangle$ and $\langle f \rangle$ respectively are not identical, the latter must contain more symbols than the former (e.g., it may contain only $\langle f \rangle$ but not $\langle a \ldots f \rangle$ or has $\langle f \rangle$ in front of $\langle a \ldots f \rangle$). However, now, the two sizes are equal. This implies that their projected databases must be identical. Such backward subpattern pruning and backward superpattern pruning can reduce the search space substantially. □

Empirical results show that CloSpan often derives a much smaller set of sequential patterns in a shorter time than PrefixSpan, which mines the complete set of sequential patterns.

[5] In $S|_\alpha$, a sequence database S is projected with respect to sequence (e.g., prefix) α. The notation $S|_\beta$ can be similarly defined.

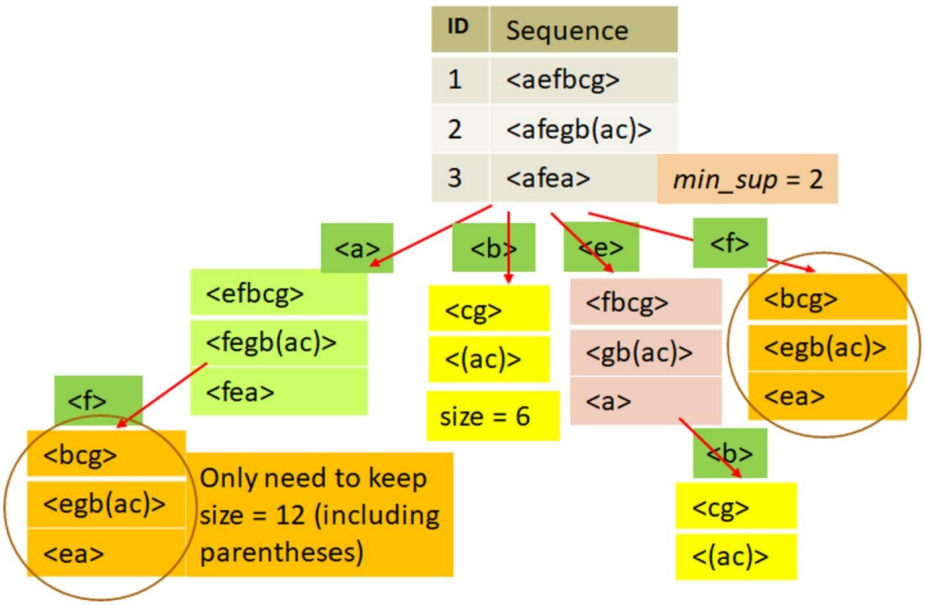


FIGURE 5.11

The pruning of a backward subpattern or a backward superpattern.

Mining multidimensional, multilevel sequential patterns

Sequence identifiers (representing individual customers, for example) and sequence items (such as products bought) are often associated with additional pieces of information. Sequential pattern mining may take advantage of such additional information to discover interesting patterns in multidimensional, multilevel information space. Take customer shopping transactions, for instance. In a sequence database for such data, the additional information associated with sequence IDs could include customer residential area, group, and profession. Information associated with items could include item category, brand, model type, model number, place manufactured, and manufacture date. Mining *multidimensional, multilevel* sequential patterns is the discovery of interesting patterns in such a broad dimensional space, at different levels of detail.

Example 5.18. Multidimensional, multilevel sequential patters. The discovery that "*Retired customers who purchase a smart home thermostat are likely to purchase a video doorbell within a month*" and that "*Young adults who purchase a laptop are likely to buy laser printer within 90 days*" are examples of multidimensional, multilevel sequential patterns. By grouping customers into *"retired customers"* and *"young adults"* according to the values in the age dimension, and by generalizing items to, say, *"smart thermostat"* rather than a specific model, the patterns mined here are associated with certain dimensions and are at a higher level of abstraction. □

"Can a typical sequential pattern algorithm such as PrefixSpan be extended to efficiently mine multidimensional, multilevel sequential patterns?" One suggested modification is to associate the mul-

tidimensional, multilevel information with the *sequence_ID* and *item_ID*, respectively, which the mining method can take into consideration when finding frequent subsequences. For example, $(Chicago, middle\_aged, business)$ can be associated with *sequence_ID_1002* (for a given customer), whereas $(laserprinter, HP, LaserJetPro, G3Q47A, USA, 2020)$ can be associated with *item_ID_543005* in the sequence. A sequential pattern mining algorithm will use such information in the mining process to find sequential patterns associated with multidimensional, multilevel information.

5.4.3 Constraint-based mining of sequential patterns

As shown in our study of frequent-pattern mining, mining that is performed without user-specified constraints may generate numerous patterns that are of no interest. Such unfocused mining can reduce both the efficiency and usability of frequent-pattern mining. Thus we promote **constraint-based mining**, which incorporates user-specified constraints to reduce the search space and derive only patterns that are of interest to the user.

Constraints can be expressed in many forms. They may specify desired relationships between attributes, attribute values, or aggregates within the resulting patterns mined. Regular expressions can also be used as constraints in the form of "pattern templates," which specify the desired form of the patterns to be mined. The general concepts introduced for constraint-based frequent pattern mining apply to constraint-based sequential pattern mining as well. The key idea to note is that these kinds of constraints can be used *during* the mining process to confine the search space, thereby improving (1) the efficiency of the mining, and (2) the interestingness of the resulting patterns found. This idea is also referred to as *"pushing the constraints deep into the mining process."*

We now examine some typical examples of constraints for sequential pattern mining.

First, constraints can be related to the **duration**, T, of a sequence. The duration can be user-specified, related to a particular time period, such as within the last 6 months. Sequential pattern mining can then be confined to the data within the specified duration, T. Constraints related to a specific duration, can be considered as *succinct* constraints. A constraint is **succinct** if we can enumerate all and only those sequences that are guaranteed to satisfy the constraint, even before support counting begins. In this case, we can push the data selection process deep into the mining process, and select sequences in the desired period before mining begins to reduce the search space.

Second, a user may confine the maximal or minimal length of the sequential patterns to be mined. The maximal or minimal length of sequential patterns can be treated as *antimonotonic* or *monotonic* constraints, respectively. For example, the constraint $L \leq 10$ is *antimonotonic* since, if a sequential pattern violates this constraint, further mining following it will always violate the constraint. Similarly, data antimonotonicity and its search space pruning rules can be established correspondingly for sequential pattern mining as well.

Third, in sequential pattern mining, a constraint can be related to an **event folding window**, w. A set of events occurring within a specified period of time can be viewed as occurring together. If w is set to 0 (i.e., no event sequence folding), sequential patterns are found where each event occurs at a distinct time instant, such as "*a customer bought a laptop, then a digital camera, and then a laser printer*" will be considered as a length-3 sequence, even if all these happen within the same day. However, if w is set to be weekly based, then these transactions are considered as occurring within the same period, and such sequences are "folded" into a set in the analysis. On the extreme, if w is set to be as long as the whole duration, T, sequential pattern mining is degenerated into sequence-insensitive frequent pattern mining.

Fourth, a desired time **gap** between events in the discovered patterns may be specified as a constraint. For example, $min\_gap \leq gap \leq max\_gap$ is to find patterns that are separated by at least min_gap but at most max_gap. A pattern like "*If a person rents movie A, it is likely she will rent movie B not within 6 days but within 30 days*" implies $6 < gap \leq 30$ (days). It is straightforward to push gap constraints into the sequential pattern mining process. With minor modifications to the mining process, it can handle constraints with approximate gaps as well.

Finally, a user can specify constraints on the kinds of sequential patterns by providing "pattern templates" in the form of regular expressions. Here we discuss mining *serial episodes* and *parallel episodes* using *regular expressions*. A **serial episode** is a set of events that occurs in total order, whereas a **parallel episode** is a set of events whose occurrence ordering is trivial. Consider the following example.

Example 5.19. Specifying serial episodes and parallel episodes with regular expressions. Let the notation (E, t) represent *event type* E at *time* t. Consider the data $(A, 1)$, $(C, 2)$, and $(B, 5)$ with an event folding window width of $w = 2$, where the serial episode $A \rightarrow B$ and the parallel episode A & C both occur in the data. The user can specify constraints in the form of a regular expression, such as $\{A|B\}C * \{D|E\}$, which indicates that the user would like to find patterns where event A and B first occur (but they are parallel in that their relative ordering is unimportant), followed by one or a set of events C, followed by the events D and E (where D can occur either before or after E). Other events can occur in between those specified in the regular expression. □

A regular expression constraint may be neither antimonotonic nor monotonic. In such cases, we cannot use it to prune the search space in the same ways as described above. However, by modifying the PrefixSpan-based pattern-growth approach, such constraints can be handled in an elegant manner. Let's examine one such example.

Example 5.20. Constraint-based sequential pattern mining with a regular expression constraint. Suppose that our task is to mine sequential patterns, again using the sequence database, S, of Table 5.4. This time, however, we are particularly interested in patterns that match the regular expression constraint, $C = \langle a * \{bb|(bc)d|dd\}\rangle$, with minimum support.

This constraint cannot be pushed deep into the mining process. Nonetheless, it can easily be integrated with the pattern-growth mining process as follows. First, only the $\langle a\rangle$-projected database, $S|_{\langle a\rangle}$, needs to be mined since the regular expression constraint C starts with a. Retain only the sequences in $S|_{\langle a\rangle}$ that contain items within the set $\{b, c, d\}$. Second, the remaining mining can proceed from the suffix. This is essentially the *Suffix-Span* algorithm, which is symmetric to PrefixSpan in that it grows suffixes from the end of the sequence forward. The growth should match the suffix as the constraint, $\langle\{bb|(bc)d|dd\}\rangle$. For the projected databases that match these suffixes, we can grow sequential patterns either in prefix- or suffix-expansion manner to find all of the remaining sequential patterns. □

Thus we have seen several ways in which constraints can be used to improve the efficiency and usability of sequential pattern mining.

5.5 Mining subgraph patterns

Graphs become increasingly important in modeling complicated structures, such as circuits, images, workflows, XML documents, webpages, chemical compounds, protein structures, biological networks,

social networks, information networks, knowledge graphs, and the Web. Many graph search algorithms have been developed in chemical informatics, computer vision, video indexing, Web search, and text retrieval. With the increasing demand on the analysis of large amounts of structured data, graph mining has become an active and important theme in data mining.

Among the various kinds of graph patterns, *frequent substructures* or *subgraphs* are the very basic patterns that can be discovered in a collection of graphs. They are useful for characterizing graph sets, discriminating different groups of graphs, classifying and clustering graphs, building graph indices, and facilitating similarity search in graph databases. Recent studies have developed several graph mining methods and applied them to the discovery of interesting patterns in various applications. For example, there have been reports on the discovery of active chemical structures in HIV-screening data sets by contrasting the support of frequent graphs between different classes. There have been studies on the use of frequent structures as features to classify chemical compounds, on the frequent graph mining technique to study protein structural families, on the detection of considerably large frequent subpathways in metabolic networks, and on the use of frequent graph patterns for graph indexing and similarity search in graph databases. Although graph mining may include mining frequent subgraph patterns, graph classification, clustering, and other analysis tasks, in this section we focus on mining frequent subgraphs. We look at various methods, their extensions, and applications.

5.5.1 Methods for mining frequent subgraphs

Before presenting graph mining methods, it is necessary to first introduce some preliminary concepts relating to frequent graph mining.

We denote the **vertex set** of a graph g by $V(g)$ and the **edge set** by $E(g)$. A label function, L, maps a vertex or an edge to a label. A graph g is a **subgraph** of another graph g' if there exists a subgraph isomorphism from g to g'. Given a labeled graph data set, $D = \{G_1, G_2, \ldots, G_n\}$, we define $support(g)$ (or $frequency(g)$) as the percentage (or number) of graphs in a graph database (i.e., a collection of graphs) D where g is a subgraph. A **frequent graph** is a graph whose support is no less than a minimum support threshold, min_sup.

Example 5.21. Frequent subgraph. Fig. 5.12 shows a sample set of chemical structures. Fig. 5.13 depicts two of the frequent subgraphs in this data set, given a minimum support of 66.6%. □

"How can we discover frequent substructures?" The discovery of frequent substructures usually consists of two steps. In the first step, we generate frequent substructure candidates. The frequency of each candidate is checked in the second step. Most studies on frequent substructure discovery focus on

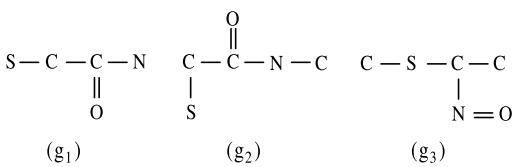


FIGURE 5.12

A sample graph data set.