Introduction to Data Mining

Pang-Ning Tan
Michael Steinbach
Vipin Kumar

Pearson

Pearson Education Limited
Edinburgh Gate
Harlow
Essex CM20 2JE
England and Associated Companies throughout the

world*Visit us on the World Wide Web at:*

www.pearsoned.co.uk

ISBN 10: 1-292-02615-4
ISBN 13: 978-1-292-02615-2

Printed in the United States of America

Contents

Introduction

Rapid advances in data collection and storage technology have enabled or ganizations to accumulate vast amounts of data. However, extracting useful information has proven extremely challenging. Often, traditional data analy sis tools and techniques cannot be used because of the massive size of a data set. Sometimes, the non-traditional nature of the data means that traditional approaches cannot be applied even if the data set is relatively small. In other situations, the questions that need to be answered cannot be addressed using existing data analysis techniques, and thus, new methods need to be devel oped.

Data mining is a technology that blends traditional data analysis methods with sophisticated algorithms for processing large volumes of data. It has also opened up exciting opportunities for exploring and analyzing new types of data and for analyzing old types of data in new ways. In this introductory chapter, we present an overview of data mining and outline the key topics to be covered in this book. We start with a description of some well-known applications that require new techniques for data analysis.

Business Point-of-sale data collection (bar code scanners, radio frequency identification (RFID), and smart card technology) have allowed retailers to collect up-to-the-minute data about customer purchases at the checkout coun ters of their stores. Retailers can utilize this information, along with other business-critical data such as Web logs from e-commerce Web sites and cus tomer service records from call centers, to help them better understand the needs of their customers and make more informed business decisions.

Data mining techniques can be used to support a wide range of business intelligence applications such as customer profiling, targeted marketing, work flow management, store layout, and fraud detection. It can also help retailers answer important business questions such as “Who are the most profitable customers?” “What products can be cross-sold or up-sold?” and “What is the revenue outlook of the company for next year?” Some of these questions mo tivated the creation of association analysis (Chapters 6 and 7), a new data analysis technique.

Medicine, Science, and Engineering Researchers in medicine, science, and engineering are rapidly accumulating data that is key to important new discoveries. For example, as an important step toward improving our under standing of the Earth's climate system, NASA has deployed a series of Earth orbiting satellites that continuously generate global observations of the land surface, oceans, and atmosphere. However, because of the size and spatio temporal nature of the data, traditional methods are often not suitable for analyzing these data sets. Techniques developed in data mining can aid Earth scientists in answering questions such as "What is the relationship between the frequency and intensity of ecosystem disturbances such as droughts and hurricanes to global warming?" "How is land surface precipitation and temper ature affected by ocean surface temperature?" and "How well can we predict the beginning and end of the growing season for a region?"

As another example, researchers in molecular biology hope to use the large amounts of genomic data currently being gathered to better understand the structure and function of genes. In the past, traditional methods in molecu lar biology allowed scientists to study only a few genes at a time in a given experiment. Recent breakthroughs in microarray technology have enabled sci entists to compare the behavior of thousands of genes under various situations. Such comparisons can help determine the function of each gene and perhaps isolate the genes responsible for certain diseases. However, the noisy and high dimensional nature of data requires new types of data analysis. In addition to analyzing gene array data, data mining can also be used to address other important biological challenges such as protein structure prediction, multiple sequence alignment, the modeling of biochemical pathways, and phylogenetics.

1.1 What Is Data Mining?

Data mining is the process of automatically discovering useful information in large data repositories. Data mining techniques are deployed to scour large databases in order to find novel and useful patterns that might otherwise remain unknown. They also provide capabilities to predict the outcome of a future observation, such as predicting whether a newly arrived customer will spend more than $100 at a department store.

Not all information discovery tasks are considered to be data mining. For example, looking up individual records using a database management system or finding particular Web pages via a query to an Internet search engine are tasks related to the area of information retrieval. Although such tasks are important and may involve the use of the sophisticated algorithms and data structures, they rely on traditional computer science techniques and obvious features of the data to create index structures for efficiently organizing and retrieving information. Nonetheless, data mining techniques have been used to enhance information retrieval systems.

Data Mining and Knowledge Discovery

Data mining is an integral part of knowledge discovery in databases (KDD), which is the overall process of converting raw data into useful in formation, as shown in Figure 1.1. This process consists of a series of trans formation steps, from data preprocessing to postprocessing of data mining results.

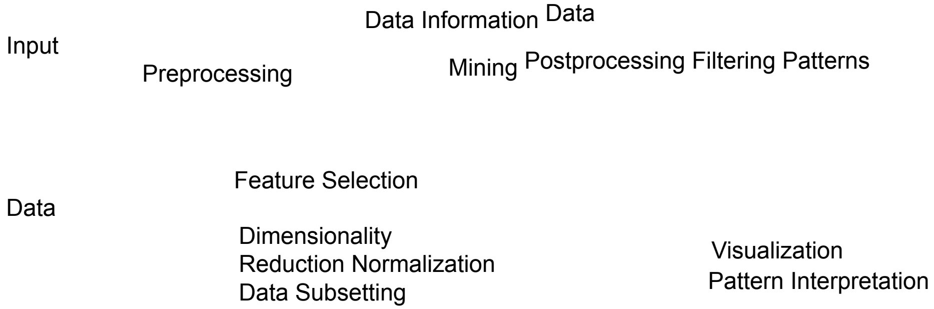


Figure 1.1. The process of knowledge discovery in databases (KDD).

The input data can be stored in a variety of formats (flat files, spread sheets, or relational tables) and may reside in a centralized data repository or be distributed across multiple sites. The purpose of preprocessing is to transform the raw input data into an appropriate format for subsequent analysis. The steps involved in data preprocessing include fusing data from multiple sources, cleaning data to remove noise and duplicate observations, and selecting records and features that are relevant to the data mining task at hand. Because of the many ways data can be collected and stored, data preprocessing is perhaps the most laborious and time-consuming step in the overall knowledge discovery process.

“Closing the loop” is the phrase often used to refer to the process of in tegrating data mining results into decision support systems. For example, in business applications, the insights offered by data mining results can be integrated with campaign management tools so that effective marketing promotions can be conducted and tested. Such integration requires a postprocessing step that ensures that only valid and useful results are incorporated into the decision support system. An example of postprocessing is visualization (see Chapter 3), which allows analysts to explore the data and the data mining results from a variety of viewpoints. Statistical measures or hypothesis testing methods can also be applied during postprocessing to eliminate spurious data mining results.

1.2 Motivating Challenges

As mentioned earlier, traditional data analysis techniques have often encountered practical difficulties in meeting the challenges posed by new data sets. The following are some of the specific challenges that motivated the development of data mining.

Scalability Because of advances in data generation and collection, data sets with sizes of gigabytes, terabytes, or even petabytes are becoming common. If data mining algorithms are to handle these massive data sets, then they must be scalable. Many data mining algorithms employ special search strategies to handle exponential search problems. Scalability may also require the implementation of novel data structures to access individual records in an efficient manner. For instance, out-of-core algorithms may be necessary when processing data sets that cannot fit into main memory. Scalability can also be improved by using sampling or developing parallel and distributed algorithms.

High Dimensionality It is now common to encounter data sets with hundreds or thousands of attributes instead of the handful common a few decades ago. In bioinformatics, progress in microarray technology has produced gene expression data involving thousands of features. Data sets with temporal or spatial components also tend to have high dimensionality. For example, consider a data set that contains measurements of temperature at various locations. If the temperature measurements are taken repeatedly for an extended period, the number of dimensions (features) increases in proportion to the number of measurements taken. Traditional data analysis techniques that were developed for low-dimensional data often do not work well for such high dimensional data. Also, for some data analysis algorithms, the computational complexity increases rapidly as the dimensionality (the number of features) increases.

Heterogeneous and Complex Data Traditional data analysis methods often deal with data sets containing attributes of the same type, either contin uous or categorical. As the role of data mining in business, science, medicine, and other fields has grown, so has the need for techniques that can handle heterogeneous attributes. Recent years have also seen the emergence of more complex data objects. Examples of such non-traditional types of data include collections of Web pages containing semi-structured text and hyperlinks; DNA data with sequential and three-dimensional structure; and climate data that consists of time series measurements (temperature, pressure, etc.) at various locations on the Earth's surface. Techniques developed for mining such com plex objects should take into consideration relationships in the data, such as temporal and spatial autocorrelation, graph connectivity, and parent-child re lationships between the elements in semi-structured text and XML documents.

Data Ownership and Distribution Sometimes, the data needed for an analysis is not stored in one location or owned by one organization. Instead, the data is geographically distributed among resources belonging to multiple entities. This requires the development of distributed data mining techniques. Among the key challenges faced by distributed data mining algorithms in clude (1) how to reduce the amount of communication needed to perform the distributed computation, (2) how to effectively consolidate the data mining results obtained from multiple sources, and (3) how to address data security issues.

Non-traditional Analysis The traditional statistical approach is based on a hypothesize-and-test paradigm. In other words, a hypothesis is proposed, an experiment is designed to gather the data, and then the data is analyzed with respect to the hypothesis. Unfortunately, this process is extremely labor intensive. Current data analysis tasks often require the generation and evalu ation of thousands of hypotheses, and consequently, the development of some data mining techniques has been motivated by the desire to automate the process of hypothesis generation and evaluation. Furthermore, the data sets analyzed in data mining are typically not the result of a carefully designed
experiment and often represent opportunistic samples of the data, rather

than random samples. Also, the data sets frequently involve non-traditional types of data and data distributions.

1.3 The Origins of Data Mining

Brought together by the goal of meeting the challenges of the previous section, researchers from different disciplines began to focus on developing more efficient and scalable tools that could handle diverse types of data. This work, which culminated in the field of data mining, built upon the methodology and algorithms that researchers had previously used. In particular, data mining draws upon ideas, such as (1) sampling, estimation, and hypothesis testing from statistics and (2) search algorithms, modeling techniques, and learning theories from artificial intelligence, pattern recognition, and machine learning. Data mining has also been quick to adopt ideas from other areas, including optimization, evolutionary computing, information theory, signal processing, visualization, and information retrieval.

A number of other areas also play key supporting roles. In particular, database systems are needed to provide support for efficient storage, indexing, and query processing. Techniques from high performance (parallel) computing are often important in addressing the massive size of some data sets. Distributed techniques can also help address the issue of size and are essential when the data cannot be gathered in one location.

Figure 1.2 shows the relationship of data mining to other areas.

AI,
Machine
and
Statistics
Pattern
Data Mining
Recognition
Learning,

Database Technology, Parallel Computing, Distributed Computing

Figure 1.2. Data mining as a confluence of many disciplines.

1.4 Data Mining Tasks

Data mining tasks are generally divided into two major categories:

Predictive tasks. The objective of these tasks is to predict the value of a par ticular attribute based on the values of other attributes. The attribute to be predicted is commonly known as the target or dependent vari able, while the attributes used for making the prediction are known as the explanatory or independent variables.

Descriptive tasks. Here, the objective is to derive patterns (correlations, trends, clusters, trajectories, and anomalies) that summarize the un derlying relationships in data. Descriptive data mining tasks are often exploratory in nature and frequently require postprocessing techniques to validate and explain the results.

Figure 1.3 illustrates four of the core data mining tasks that are described in the remainder of this book.

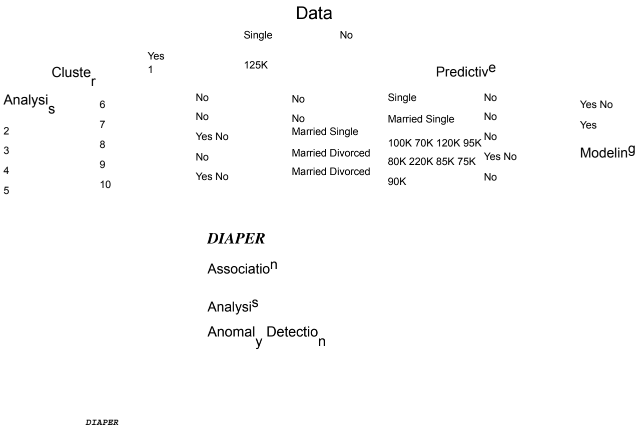


Figure 1.3. Four of the core data mining tasks.

Predictive modeling refers to the task of building a model for the target variable as a function of the explanatory variables. There are two types of predictive modeling tasks: classification, which is used for discrete target variables, and regression, which is used for continuous target variables. For example, predicting whether a Web user will make a purchase at an online bookstore is a classification task because the target variable is binary-valued. On the other hand, forecasting the future price of a stock is a regression task because price is a continuous-valued attribute. The goal of both tasks is to learn a model that minimizes the error between the predicted and true values of the target variable. Predictive modeling can be used to identify customers that will respond to a marketing campaign, predict disturbances in the Earth's ecosystem, or judge whether a patient has a particular disease based on the results of medical tests.

Example 1.1 (Predicting the Type of a Flower). Consider the task of predicting a species of flower based on the characteristics of the flower. In particular, consider classifying an Iris flower as to whether it belongs to one of the following three Iris species: Setosa, Versicolour, or Virginica. To per form this task, we need a data set containing the characteristics of various flowers of these three species. A data set with this type of information is the well-known Iris data set from the UCI Machine Learning Repository at http://www.ics.uci.edu/~mlearn. In addition to the species of a flower, this data set contains four other attributes: sepal width, sepal length, petal length, and petal width. (The Iris data set and its attributes are described further in Section 3.1.) Figure 1.4 shows a plot of petal width versus petal length for the 150 flowers in the Iris data set. Petal width is broken into the categories *low*, *medium*, and *high*, which correspond to the intervals [0, 0.75), [0.75, 1.75), [1.75, ∞), respectively. Also, petal length is broken into categories *low*, *medium*, and *high*, which correspond to the intervals [0, 2.5), [2.5, 5), [5, ∞), respectively. Based on these categories of petal width and length, the following rules can be derived:

Petal width low and petal length low implies Setosa.
Petal width medium and petal length medium implies Versicolour.
Petal width high and petal length high implies Virginica.

While these rules do not classify all the flowers, they do a good (but not perfect) job of classifying most of the flowers. Note that flowers from the Setosa species are well separated from the Versicolour and Virginica species with respect to petal width and length, but the latter two species overlap somewhat with respect to these attributes.

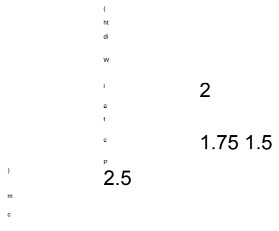


Figure 1.4. Petal width versus petal length for 150 Iris flowers.

Association analysis is used to discover patterns that describe strongly as sociated features in the data. The discovered patterns are typically represented in the form of implication rules or feature subsets. Because of the exponential size of its search space, the goal of association analysis is to extract the most interesting patterns in an efficient manner. Useful applications of association analysis include finding groups of genes that have related functionality, identi fying Web pages that are accessed together, or understanding the relationships between different elements of Earth's climate system.

Example 1.2 (Market Basket Analysis). The transactions shown in Ta ble 1.1 illustrate point-of-sale data collected at the checkout counters of a grocery store. Association analysis can be applied to find items that are fre quently bought together by customers. For example, we may discover the rule *{*Diapers*}* −→ *{*Milk*}*, which suggests that customers who buy diapers also tend to buy milk. This type of rule can be used to identify potential cross-selling opportunities among related items.

Cluster analysis seeks to find groups of closely related observations so that observations that belong to the same cluster are more similar to each other than observations that belong to other clusters. Clustering has been used to group sets of related customers, find areas of the ocean that have a significant impact on the Earth’s climate, and compress data.

Table 1.1. Market basket data.

Transaction ID	Items
1	{Bread, Butter, Diapers, Milk}
2	{Coffee, Sugar, Cookies, Salmon}
3	{Bread, Butter, Coffee, Diapers, Milk, Eggs}
4	{Bread, Butter, Salmon, Chicken}
5	{Eggs, Bread, Butter}
6	{Salmon, Diapers, Milk}
7	{Bread, Tea, Sugar, Eggs}
8	{Coffee, Sugar, Chicken, Eggs}
9	{Bread, Diapers, Milk, Salt}
10	{Tea, Eggs, Cookies, Diapers, Milk}

Example 1.3 (Document Clustering). The collection of news articles shown in Table 1.2 can be grouped based on their respective topics. Each article is represented as a set of word-frequency pairs (*w, c*), where *w* is a word and *c* is the number of times the word appears in the article. There are two natural clusters in the data set. The first cluster consists of the first four articles, which correspond to news about the economy, while the second cluster contains the last four articles, which correspond to news about health care. A good clustering algorithm should be able to identify these two clusters based on the similarity between words that appear in the articles.

Table 1.2. Collection of news articles.

Article	Words
1	dollar: 1, industry: 4, country: 2, loan: 3, deal: 2, government: 2
2	machinery: 2, labor: 3, market: 4, industry: 2, work: 3, country: 1
3	job: 5, inflation: 3, rise: 2, jobless: 2, market: 3, country: 2, index: 3
4	domestic: 3, forecast: 2, gain: 1, market: 2, sale: 3, price: 2
5	patient: 4, symptom: 2, drug: 3, health: 2, clinic: 2, doctor: 2
6	pharmaceutical: 2, company: 3, drug: 2, vaccine: 1, flu: 3
7	death: 2, cancer: 4, drug: 3, public: 4, health: 3, director: 2
8	medical: 2, cost: 3, increase: 2, patient: 2, health: 3, care: 1

Anomaly detection is the task of identifying observations whose characteristics are significantly different from the rest of the data. Such observations are known as anomalies or outliers. The goal of an anomaly detection algorithm is to discover the real anomalies and avoid falsely labeling normal objects as anomalous. In other words, a good anomaly detector must have a high detection rate and a low false alarm rate. Applications of anomaly detection include the detection of fraud, network intrusions, unusual patterns of disease, and ecosystem disturbances.

Example 1.4 (Credit Card Fraud Detection). A credit card company records the transactions made by every credit card holder, along with personal information such as credit limit, age, annual income, and address. Since the number of fraudulent cases is relatively small compared to the number of legitimate transactions, anomaly detection techniques can be applied to build a profile of legitimate transactions for the users. When a new transaction arrives, it is compared against the profile of the user. If the characteristics of the transaction are very different from the previously created profile, then the transaction is flagged as potentially fraudulent.

1.5 Scope and Organization of the Book

This book introduces the major principles and techniques used in data mining from an algorithmic perspective. A study of these principles and techniques is essential for developing a better understanding of how data mining technology can be applied to various kinds of data. This book also serves as a starting point for readers who are interested in doing research in this field.

We begin the technical discussion of this book with a chapter on data (Chapter 2), which discusses the basic types of data, data quality, prepro cessing techniques, and measures of similarity and dissimilarity. Although this material can be covered quickly, it provides an essential foundation for data analysis. Chapter 3, on data exploration, discusses summary statistics, visualization techniques, and On-Line Analytical Processing (OLAP). These techniques provide the means for quickly gaining insight into a data set.

Chapters 4 and 5 cover classification. Chapter 4 provides a foundation by discussing decision tree classifiers and several issues that are important to all classification: overfitting, performance evaluation, and the comparison of different classification models. Using this foundation, Chapter 5 describes a number of other important classification techniques: rule-based systems, nearest-neighbor classifiers, Bayesian classifiers, artificial neural networks, sup port vector machines, and ensemble classifiers, which are collections of classi
fiers. The multiclass and imbalanced class problems are also discussed. These topics can be covered independently.

Association analysis is explored in Chapters 6 and 7. Chapter 6 describes the basics of association analysis: frequent itemsets, association rules, and some of the algorithms used to generate them. Specific types of frequent itemsets—maximal, closed, and hyperclique—that are important for data mining are also discussed, and the chapter concludes with a discussion of evaluation measures for association analysis. Chapter 7 considers a variety of more advanced topics, including how association analysis can be applied to categorical and continuous data or to data that has a concept hierarchy. (A concept hierarchy is a hierarchical categorization of objects, e.g., store items, clothing, shoes, sneakers.) This chapter also describes how association analysis can be extended to find sequential patterns (patterns involving order), patterns in graphs, and negative relationships (if one item is present, then the other is not).

Cluster analysis is discussed in Chapters 8 and 9. Chapter 8 first describes the different types of clusters and then presents three specific clustering techniques: K-means, agglomerative hierarchical clustering, and DBSCAN. This is followed by a discussion of techniques for validating the results of a clustering algorithm. Additional clustering concepts and techniques are explored in Chapter 9, including fuzzy and probabilistic clustering, Self-Organizing Maps (SOM), graph-based clustering, and density-based clustering. There is also a discussion of scalability issues and factors to consider when selecting a clustering algorithm.

The last chapter, Chapter 10, is on anomaly detection. After some basic definitions, several different types of anomaly detection are considered: statistical, distance-based, density-based, and clustering-based. Appendices A through E give a brief review of important topics that are used in portions of the book: linear algebra, dimensionality reduction, statistics, regression, and optimization.

The subject of data mining, while relatively young compared to statistics or machine learning, is already too large to cover in a single book. Selected references to topics that are only briefly covered, such as data quality, are provided in the bibliographic notes of the appropriate chapter. References to topics not covered in this book, such as data mining for streams and privacy preserving data mining, are provided in the bibliographic notes of this chapter.

1.6 Bibliographic Notes

The topic of data mining has inspired many textbooks. Introductory textbooks include those by Dunham [10], Han and Kamber [21], Hand et al. [23], and Roiger and Geatz [36]. Data mining books with a stronger emphasis on business applications include the works by Berry and Linoff [2], Pyle [34], and Parr Rud [33]. Books with an emphasis on statistical learning include those by Cherkassky and Mulier [6], and Hastie et al. [24]. Some books with an emphasis on machine learning or pattern recognition are those by Duda et al. [9], Kantardzic [25], Mitchell [31], Webb [41], and Witten and Frank [42]. There are also some more specialized books: Chakrabarti [4] (web mining), Fayyad et al. [13] (collection of early articles on data mining), Fayyad et al. [11] (visualization), Grossman et al. [18] (science and engineering), Kargupta and Chan [26] (distributed data mining), Wang et al. [40] (bioinformatics), and Zaki and Ho [44] (parallel data mining).

There are several conferences related to data mining. Some of the main conferences dedicated to this field include the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), the IEEE International Conference on Data Mining (ICDM), the SIAM International Conference on Data Mining (SDM), the European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD), and the Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD). Data min ing papers can also be found in other major conferences such as the ACM SIGMOD/PODS conference, the International Conference on Very Large Data Bases (VLDB), the Conference on Information and Knowledge Management (CIKM), the International Conference on Data Engineering (ICDE), the In ternational Conference on Machine Learning (ICML), and the National Con ference on Artificial Intelligence (AAAI).

Journal publications on data mining include *IEEE Transactions on Knowledge and Data Engineering*, *Data Mining and Knowledge Discovery*, *Knowledge and Information Systems*, *Intelligent Data Analysis*, *Information Systems*, and the *Journal of Intelligent Information Systems*.

There have been a number of general articles on data mining that define the field or its relationship to other fields, particularly statistics. Fayyad et al. [12] describe data mining and how it fits into the total knowledge discovery process. Chen et al. [5] give a database perspective on data mining. Ramakrishnan and Grama [35] provide a general discussion of data mining and present several viewpoints. Hand [22] describes how data mining differs from statistics, as does Friedman [14]. Lambert [29] explores the use of statistics for large data sets and provides some comments on the respective roles of data mining and statistics.

Glymour et al. [16] consider the lessons that statistics may have for data mining. Smyth et al. [38] describe how the evolution of data mining is being driven by new types of data and applications, such as those involving streams, graphs, and text. Emerging applications in data mining are considered by Han et al. [20] and Smyth [37] describes some research challenges in data mining. A discussion of how developments in data mining research can be turned into practical tools is given by Wu et al. [43]. Data mining standards are the subject of a paper by Grossman et al. [17]. Bradley [3] discusses how data mining algorithms can be scaled to large data sets.

With the emergence of new data mining applications have come new chal lenges that need to be addressed. For instance, concerns about privacy breaches as a result of data mining have escalated in recent years, particularly in ap plication domains such as Web commerce and health care. As a result, there is growing interest in developing data mining algorithms that maintain user privacy. Developing techniques for mining encrypted or randomized data is known as privacy-preserving data mining. Some general references in this area include papers by Agrawal and Srikant [1], Clifton et al. [7] and Kargupta et al. [27]. Vassilios et al. [39] provide a survey.

Recent years have witnessed a growing number of applications that rapidly generate continuous streams of data. Examples of stream data include network traffic, multimedia streams, and stock prices. Several issues must be considered when mining data streams, such as the limited amount of memory available, the need for online analysis, and the change of the data over time. Data mining for stream data has become an important area in data mining. Some selected publications are Domingos and Hulten [8] (classification), Giannella et al. [15] (association analysis), Guha et al. [19] (clustering), Kifer et al. [28] (change detection), Papadimitriou et al. [32] (time series), and Law et al. [30] (dimensionality reduction).

Bibliography

[1] R. Agrawal and R. Srikant. Privacy-preserving data mining. In *Proc. of 2000 ACM SIGMOD Intl. Conf. on Management of Data*, pages 439–450, Dallas, Texas, 2000. ACM Press.

[2] M. J. A. Berry and G. Linoff. *Data Mining Techniques: For Marketing, Sales, and Customer Relationship Management*. Wiley Computer Publishing, 2nd edition, 2004. [3] P. S. Bradley, J. Gehrke, R. Ramakrishnan, and R. Srikant. Scaling mining algorithms to large databases. *Communications of the ACM*, 45(8):38–43, 2002.

[4] S. Chakrabarti. *Mining the Web: Discovering Knowledge from Hypertext Data*. Morgan Kaufmann, San Francisco, CA, 2003.

[5] M.-S. Chen, J. Han, and P. S. Yu. Data Mining: An Overview from a Database Perspective. *IEEE Transactions on Knowledge abd Data Engineering*, 8(6):866–883, 1996.

[6] V. Cherkassky and F. Mulier. *Learning from Data: Concepts, Theory, and Methods*. Wiley Interscience, 1998.

[7] C. Clifton, M. Kantarcioglu, and J. Vaidya. Defining privacy for data mining. In *National Science Foundation Workshop on Next Generation Data Mining*, pages 126– 133, Baltimore, MD, November 2002.

[8] P. Domingos and G. Hulten. Mining high-speed data streams. In *Proc. of the 6th Intl. Conf. on Knowledge Discovery and Data Mining*, pages 71–80, Boston, Massachusetts, 2000. ACM Press.

[9] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. John Wiley & Sons, Inc., New York, 2nd edition, 2001.

[10] M. H. Dunham. *Data Mining: Introductory and Advanced Topics*. Prentice Hall, 2002.

[11] U. M. Fayyad, G. G. Grinstein, and A. Wierse, editors. *Information Visualization in Data Mining and Knowledge Discovery*. Morgan Kaufmann Publishers, San Francisco, CA, September 2001.

[12] U. M. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. From Data Mining to Knowledge Discovery: An Overview. In *Advances in Knowledge Discovery and Data Mining*, pages 1–34. AAAI Press, 1996.

[13] U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors. *Advances in Knowledge Discovery and Data Mining.* AAAI/MIT Press, 1996.

[14] J. H. Friedman. Data Mining and Statistics: What's the Connection? Unpublished. www-stat.stanford.edu/~jhf/ftp/dm-stat.ps, 1997.

[15] C. Giannella, J. Han, J. Pei, X. Yan, and P. S. Yu. Mining Frequent Patterns in Data Streams at Multiple Time Granularities. In H. Kargupta, A. Joshi, K. Sivakumar, and Y. Yesha, editors, *Next Generation Data Mining*, pages 191–212. AAAI/MIT, 2003.

[16] C. Glymour, D. Madigan, D. Pregibon, and P. Smyth. Statistical Themes and Lessons for Data Mining. *Data Mining and Knowledge Discovery*, 1(1):11–28, 1997. [17] R. L. Grossman, M. F. Hornick, and G. Meyer. Data mining standards initiatives. *Communications of the ACM*, 45(8):59–61, 2002.

[18] R. L. Grossman, C. Kamath, P. Kegelmeyer, V. Kumar, and R. Namburu, editors. *Data Mining for Scientific and Engineering Applications*. Kluwer Academic Publishers, 2001. [19] S. Guha, A. Meyerson, N. Mishra, R. Motwani, and L. O'Callaghan. Clustering Data Streams: Theory and Practice. *IEEE Transactions on Knowledge and Data Engineering*, 15(3):515–528, May/June 2003.

[20] J. Han, R. B. Altman, V. Kumar, H. Mannila, and D. Pregibon. Emerging scientific applications in data mining. *Communications of the ACM*, 45(8):54–58, 2002. [21] J. Han and M. Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers, San Francisco, 2001.

[22] D. J. Hand. Data Mining: Statistics and More? *The American Statistician*, 52(2): 112–118, 1998.

[23] D. J. Hand, H. Mannila, and P. Smyth. *Principles of Data Mining*. MIT Press, 2001. [24] T. Hastie, R. Tibshirani, and J. H. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, Prediction*. Springer, New York, 2001.

[25] M. Kantardzic. *Data Mining: Concepts, Models, Methods, and Algorithms*. Wiley-IEEE Press, Piscataway, NJ, 2003.

[26] H. Kargupta and P. K. Chan, editors. *Advances in Distributed and Parallel Knowledge Discovery*. AAAI Press, September 2002.

[27] H. Kargupta, S. Datta, Q. Wang, and K. Sivakumar. On the Privacy Preserving Prop erties of Random Data Perturbation Techniques. In *Proc. of the 2003 IEEE Intl. Conf. on Data Mining*, pages 99–106, Melbourne, Florida, December 2003. IEEE Computer

Society.

[28] D. Kifer, S. Ben-David, and J. Gehrke. Detecting Change in Data Streams. In *Proc. of the 30th VLDB Conf.*, pages 180–191, Toronto, Canada, 2004. Morgan Kaufmann. [29] D. Lambert. What Use is Statistics for Massive Data? In *ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, pages 54–62, 2000. [30] M. H. C. Law, N. Zhang, and A. K. Jain. Nonlinear Manifold Learning for Data Streams. In *Proc. of the SIAM Intl. Conf. on Data Mining*, Lake Buena Vista, Florida, April 2004. SIAM.

[31] T. Mitchell. *Machine Learning*. McGraw-Hill, Boston, MA, 1997. [32] S. Papadimitriou, A. Brockwell, and C. Faloutsos. Adaptive, unsupervised stream min ing. *VLDB Journal*, 13(3):222–239, 2004.

[33] O. Parr Rud. *Data Mining Cookbook: Modeling Data for Marketing, Risk and Customer Relationship Management*. John Wiley & Sons, New York, NY, 2001. [34] D. Pyle. *Business Modeling and Data Mining*. Morgan Kaufmann, San Francisco, CA, 2003.

[35] N. Ramakrishnan and A. Grama. Data Mining: From Serendipity to Science—Guest Editors' Introduction. *IEEE Computer*, 32(8):34–37, 1999.

[36] R. Roiger and M. Geatz. *Data Mining: A Tutorial Based Primer*. Addison-Wesley, 2002.

[37] P. Smyth. Breaking out of the Black-Box: Research Challenges in Data Mining. In *Proc. of the 2001 ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, 2001.

[38] P. Smyth, D. Pregibon, and C. Faloutsos. Data-driven evolution of data mining algo rithms. *Communications of the ACM*, 45(8):33–37, 2002.

[39] V. S. Verykios, E. Bertino, I. N. Fovino, L. P. Provenza, Y. Saygin, and Y. Theodoridis. State-of-the-art in privacy preserving data mining. *SIGMOD Record*, 33(1):50–57, 2004. [40] J. T. L. Wang, M. J. Zaki, H. Toivonen, and D. E. Shasha, editors. *Data Mining in Bioinformatics*. Springer, September 2004.

[41] A. R. Webb. *Statistical Pattern Recognition*. John Wiley & Sons, 2nd edition, 2002. [42] I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Tech niques with Java Implementations*. Morgan Kaufmann, 1999.

[43] X. Wu, P. S. Yu, and G. Piatetsky-Shapiro. Data Mining: How Research Meets Practical Development? *Knowledge and Information Systems*, 5(2):248–261, 2003. [44] M. J. Zaki and C.-T. Ho, editors. *Large-Scale Parallel Data Mining*. Springer, September 2002.

1.7 Exercises

1. Discuss whether or not each of the following activities is a data mining task.
   (a) Dividing the customers of a company according to their gender. (b) Dividing the customers of a company according to their profitability. (c) Computing the total sales of a company.
   (d) Sorting a student database based on student identification numbers. (e) Predicting the outcomes of tossing a (fair) pair of dice.
   (f) Predicting the future stock price of a company using historical records.

(g) Monitoring the heart rate of a patient for abnormalities.

(h) Monitoring seismic waves for earthquake activities.

(i) Extracting the frequencies of a sound wave.

2. Suppose that you are employed as a data mining consultant for an Internet search engine company. Describe how data mining can help the company by giving specific examples of how techniques, such as clustering, classification, association rule mining, and anomaly detection can be applied.

3. For each of the following data sets, explain whether or not data privacy is an important issue.

(a) Census data collected from 1900–1950.

(b) IP addresses and visit times of Web users who visit your

Website. (c) Images from Earth-orbiting satellites.

(d) Names and addresses of people from the telephone book.

(e) Names and email addresses collected from the Web.

2

Data

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

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

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

Preprocessing Steps to Make the Data More Suitable for Data Min ing Often, the raw data must be processed in order to make it suitable for analysis. While one objective may be to improve data quality, other goals focus on modifying the data so that it better fits a specified data mining technique or tool. For example, a continuous attribute, e.g., length, may need to be transformed into an attribute with discrete categories, e.g., *short*, *medium*, or *long*, in order to apply a particular technique. As another example, the number of attributes in a data set is often reduced because many techniques are more effective when the data has a relatively small number of attributes.

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

Example 2.1 (An Illustration of Data-Related Issues). To further il lustrate the importance of these issues, consider the following hypothetical sit uation. You receive an email from a medical researcher concerning a project that you are eager to work on.

> Hi,
>
> I’ve attached the data file that I mentioned in my previous email. Each line contains the information for a single patient and consists of five fields. We want to predict the last field using the other fields. I don’t have time to provide any more information about the data since I’m going out of town for a couple of days,

but hopefully that won't slow you down too much. And if you don't mind, could we meet when I get back to discuss your preliminary results? I might invite a few other members of my team.

Thanks and see you in a couple of days.

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

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

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

Statistician: So, you got the data for all the patients?
Data Miner: Yes. I haven't had much time for analysis, but I do have a few interesting results.
Statistician: Amazing. There were so many data issues with this set of patients that I couldn't do much.
Data Miner: Oh? I didn't hear about any possible problems.
Statistician: Well, first there is field 5, the variable we want to predict. It's common knowledge among people who analyze this type of data that results are better if you work with the log of the values, but I didn't discover this until later. Was it mentioned to you?
Data Miner: No.
Statistician: But surely you heard about what happened to field 4? It's supposed to be measured on a scale from 1 to 10, with 0 indicating a missing value, but because of a data entry error, all 10's were changed into 0's. Unfortunately, since some of the patients have missing values for this field, it's

impossible to say whether a 0 in this field is a real 0 or a 10. Quite a few of the records have that problem.

Data Miner: Interesting. Were there any other problems?

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

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

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

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

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

Data Miner: Nonetheless, my results speak for themselves.

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

Although this scenario represents an extreme situation, it emphasizes the importance of "knowing your data." To that end, this chapter will address each of the four issues mentioned above, outlining some of the basic challenges and standard approaches.

2.1 Types of Data

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

Example 2.2 (Student Information). Often, a data set is a file, in which the objects are records (or rows) in the file and each field (or column) corre sponds to an attribute. For example, Table 2.1 shows a data set that consists of student information. Each row corresponds to a student and each column is an attribute that describes some aspect of a student, such as grade point average (GPA) or identification number (ID).

Table 2.1. A sample data set containing student information.

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

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

2.1.1 Attributes and Measurement

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

What Is an attribute?

We start with a more detailed definition of an attribute.

Definition 2.1. An attribute is a property or characteristic of an object that may vary, either from one object to another or from one time to another.

For example, eye color varies from person to person, while the temperature of an object varies over time. Note that eye color is a symbolic attribute with a small number of possible values *{brown, black, blue, green, hazel, etc.}*, while temperature is a numerical attribute with a potentially unlimited number of values.

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

Definition 2.2. A measurement scale is a rule (function) that associates a numerical or symbolic value with an attribute of an object.

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

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

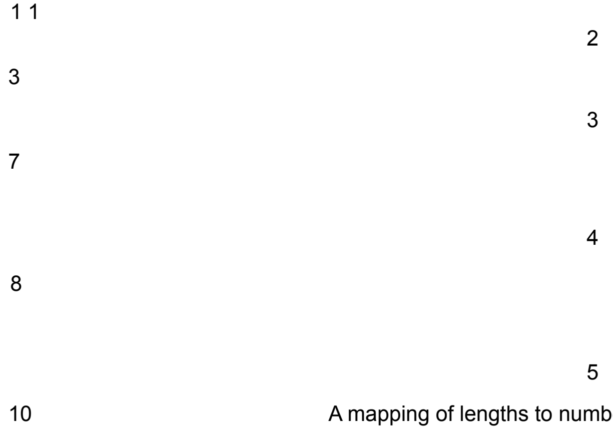
The Type of an Attribute

It should be apparent from the previous discussion that the properties of an attribute need not be the same as the properties of the values used to mea sure it. In other words, the values used to represent an attribute may have properties that are not properties of the attribute itself, and vice versa. This is illustrated with two examples.

Example 2.3 (Employee Age and ID Number). Two attributes that might be associated with an employee are *ID* and *age* (in years). Both of these attributes can be represented as integers. However, while it is reasonable to talk about the average age of an employee, it makes no sense to talk about the average employee ID. Indeed, the only aspect of employees that we want to capture with the ID attribute is that they are distinct. Consequently, the only valid operation for employee IDs is to test whether they are equal. There is no hint of this limitation, however, when integers are used to represent the employee ID attribute. For the age attribute, the properties of the integers used to represent age are very much the properties of the attribute. Even so, the correspondence is not complete since, for example, ages have a maximum, while integers do not.

Example 2.4 (Length of Line Segments). Consider Figure 2.1, which shows some objects—line segments—and how the length attribute of these objects can be mapped to numbers in two different ways. Each successive line segment, going from the top to the bottom, is formed by appending the topmost line segment to itself. Thus, the second line segment from the top is formed by appending the topmost line segment to itself twice, the third line segment from the top is formed by appending the topmost line segment to itself three times, and so forth. In a very real (physical) sense, all the line segments are multiples of the first. This fact is captured by the measurements on the right-hand side of the figure, but not by those on the left hand-side. More specifically, the measurement scale on the left-hand side captures only the ordering of the length attribute, while the scale on the right-hand side captures both the ordering and additivity properties. Thus, an attribute can be measured in a way that does not capture all the properties of the attribute.

The type of an attribute should tell us what properties of the attribute are reflected in the values used to measure it. Knowing the type of an attribute is important because it tells us which properties of the measured values are consistent with the underlying properties of the attribute, and therefore, it allows us to avoid foolish actions, such as computing the average employee ID. Note that it is common to refer to the type of an attribute as the type of a measurement scale.



A mapping of lengths to numbers that captures only the order properties of length.

A mapping of lengths to numbers that captures both the order and additivity properties of length.

Figure 2.1. The measurement of the length of line segments on two different scales of measurement.

The Different Types of Attributes

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

1. Distinctness = and =

2. Order <, ≤, >, and ≥

3. Addition + and −

4. Multiplication ∗ and /

Given these properties, we can define four types of attributes: nominal, ordinal, interval, and ratio. Table 2.2 gives the definitions of these types, along with information about the statistical operations that are valid for each type. Each attribute type possesses all of the properties and operations of the attribute types above it. Consequently, any property or operation that is valid for nominal, ordinal, and interval attributes is also valid for ratio attributes. In other words, the definition of the attribute types is cumulative. However,

Table 2.2. Different attribute types.

ttribute

ype Description Examples Operations

l a c i r o g e t a

c i r e m u N)e v i t

a t i l a u Q(
)e
v i t a t i t n a u Q(

Nominal The
values of a
nominal attribute
are just different
names; i.e.,
nominal values
provide only
enough

information to
distinguish
one object from
another.
(=, =)
Ordinal The
values of an
ordinal attribute
provide enough
information to
order
objects.
(<, >)
Interval For
interval attributes,
the differences
between values
are meaningful,
i.e., a unit
of measurement
exists.
(+, −)
Ratio For ratio
variables, both

differences and
ratios are
meaningful.
(*, /)
zip codes,
employee ID
numbers, eye
color, gender

hardness of
minerals, *{good, better, best}*,
grades,
street numbers

calendar dates,
temperature in
Celsius or
Fahrenheit

temperature in
Kelvin, monetary
quantities,
counts, age,
mass, length,
electrical current
mode, entropy,
contingency
correlation,
χ^2 test

median,
percentiles,
rank correlation,
run tests,
sign tests
mean,
standard
deviatioPearson's
correlation,

t and *F* tests geometric mean, harmonic mean, percent variation

this does not mean that the operations appropriate for one attribute type are appropriate for the attribute types above it.

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

The types of attributes can also be described in terms of transformations that do not change the meaning of an attribute. Indeed, S. Smith Stevens, the psychologist who originally defined the types of attributes shown in Table 2.2, defined them in terms of these permissible transformations. For example,

Table 2.3. Transformations that define attribute levels.

	ttribute ype	Transformation	Comment
Categorical (Qualitative)	Nominal	Any one-to-one mapping, e.g., a permutation of values	If all employee ID numbers are reassigned, it will not make any difference.
	Ordinal	An order-preserving change of values, i.e., *new value* = *f*(*old value*), where *f* is a monotonic function.	An attribute encompassing the notion of good, better, best can be represented equally well by the values *{*1*,* 2*,* 3*}* or by *{*0.5*,* 1*,* 10*}*.
Numeric (Quantitative)	Interval	*new value* = *a* ∗ *old value* + *b*, *a* and *b* constants.	The Fahrenheit and Celsius temperature scales differ in the location of their zero value and the size of a degree (unit).
	Ratio	*new value* = *a* ∗ *old value*	Length

can be measured in

meters or feet.

the meaning of a length attribute is unchanged if it is measured in meters instead of feet.

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

Example 2.5 (Temperature Scales). Temperature provides a good illustration of some of the concepts that have been described. First, temperature can be either an interval or a ratio attribute, depending on its measurement scale. When measured on the Kelvin scale, a temperature of 2° is, in a physically meaningful way, twice that of a temperature of 1°. This is not true when temperature is measured on either the Celsius or Fahrenheit scales, because, physically, a temperature of 1° Fahrenheit (Celsius) is not much different than a temperature of 2° Fahrenheit (Celsius). The problem is that the zero points of the Fahrenheit and Celsius scales are, in a physical sense, arbitrary, and therefore, the ratio of two Celsius or Fahrenheit temperatures is not physi cally meaningful.

Describing Attributes by the Number of Values

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

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

Continuous A continuous attribute is one whose values are real numbers. Ex amples include attributes such as temperature, height, or weight.

Continuous attributes are typically represented as floating-point variables. Practically, real values can only be measured and represented with limited precision.

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

Asymmetric Attributes

For asymmetric attributes, only presence—a non-zero attribute value—is regarded as important. Consider a data set where each object is a student and each attribute records whether or not a student took a particular course at a university. For a specific student, an attribute has a value of 1 if the student took the course associated with that attribute and a value of 0 otherwise. Because students take only a small fraction of all available courses, most of the values in such a data set would be 0. Therefore, it is more meaningful and more efficient to focus on the non-zero values. To illustrate, if students are compared on the basis of the courses they don't take, then most students would seem very similar, at least if the number of courses is large. Binary attributes where only non-zero values are important are called asymmetric
binary attributes. This type of attribute is particularly important for association analysis, which is discussed in Chapter 6. It is also possible to have discrete or continuous asymmetric features. For instance, if the number of credits associated with each course is recorded, then the resulting data set will consist of asymmetric discrete or continuous attributes.

2.1.2 Types of Data Sets

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

General Characteristics of Data Sets

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

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

Sparsity For some data sets, such as those with asymmetric features, most attributes of an object have values of 0; in many cases, fewer than 1% of the entries are non-zero. In practical terms, sparsity is an advantage because usually only the non-zero values need to be stored and manipulated. This results in significant savings with respect to computation time and storage. Furthermore, some data mining algorithms work well only for sparse data.

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

Record Data

Much data mining work assumes that the data set is a collection of records (data objects), each of which consists of a fixed set of data fields (attributes). See Figure 2.2(a). For the most basic form of record data, there is no explicit relationship among records or data fields, and every record (object) has the same set of attributes. Record data is usually stored either in flat files or in relational databases. Relational databases are certainly more than a collection of records, but data mining often does not use any of the additional information available in a relational database. Rather, the database serves as a convenient place to find records. Different types of record data are described below and are illustrated in Figure 2.2.

Transaction or Market Basket Data Transaction data is a special type of record data, where each record (transaction) involves a set of items. Con sider a grocery store. The set of products purchased by a customer during one shopping trip constitutes a transaction, while the individual products that were purchased are the items. This type of data is called market basket data because the items in each record are the products in a person's "mar ket basket." Transaction data is a collection of sets of items, but it can be viewed as a set of records whose fields are asymmetric attributes. Most often, the attributes are binary, indicating whether or not an item was purchased, but more generally, the attributes can be discrete or continuous, such as the number of items purchased or the amount spent on those items. Figure 2.2(b) shows a sample transaction data set. Each row represents the purchases of a particular customer at a particular time.

The Data Matrix If the data objects in a collection of data all have the same fixed set of numeric attributes, then the data objects can be thought of as points (vectors) in a multidimensional space, where each dimension represents a distinct attribute describing the object. A set of such data objects can be interpreted as an *m* by *n* matrix, where there are *m* rows, one for each object,

No

Single

1 125K

Yes

9 No Married **No**

10 Married Single **Yes No** 2 3 4 5 Beer, Soda,

2

No Single 100K 70K **No**

3

No 120K 95K Bread, Soda, Diaper, Milk

Married **Yes No**

4 Milk

Yes No 60K 220K

Divorced **Yes**

5

No 85K 75K 90K 1 Beer, Bread,

Married

6 Beer, Bread

Yes No

Divorced **No**

7

No Single **No**

8

Diaper, Milk   Soda, Diaper,Milk

data.

(a) Record data.

(b) Transaction

10.23   5.27   27   Document 1 3 0 5 0 2 6 0 2 0 2

15.22   1.2

12.65 13.54   6.25 7.23 8.43 18.45   1.1 1.2 0.9   Document 3   0702100300   0100122030

14.27   16.22 17.34   22 23 25

Document 2

(d) Document-term matrix.

(c) Data matrix.

Figure 2.2. Different variations of record data.

and *n* columns, one for each attribute. (A representation that has data objects as columns and attributes as rows is also fine.) This matrix is called a data matrix or a pattern matrix. A data matrix is a variation of record data, but because it consists of numeric attributes, standard matrix operation can be applied to transform and manipulate the data. Therefore, the data matrix is the standard data format for most statistical data. Figure 2.2(c) shows a sample data matrix.

The Sparse Data Matrix A sparse data matrix is a special case of a data matrix in which the attributes are of the same type and are asymmetric; i.e., only non-zero values are important. Transaction data is an example of a sparse data matrix that has only 0–1 entries. Another common example is document data. In particular, if the order of the terms (words) in a document is ignored,

then a document can be represented as a term vector, where each term is a component (attribute) of the vector and the value of each component is the number of times the corresponding term occurs in the document. This representation of a collection of documents is often called a document-term matrix. Figure 2.2(d) shows a sample document-term matrix. The documents are the rows of this matrix, while the terms are the columns. In practice, only the non-zero entries of sparse data matrices are stored.

Graph-Based Data

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

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

Data with Objects That Are Graphs If objects have structure, that is, the objects contain subobjects that have relationships, then such objects are frequently represented as graphs. For example, the structure of chemical compounds can be represented by a graph, where the nodes are atoms and the links between nodes are chemical bonds. Figure 2.3(b) shows a ball-and-stick diagram of the chemical compound benzene, which contains atoms of carbon (black) and hydrogen (gray). A graph representation makes it possible to determine which substructures occur frequently in a set of compounds and to ascertain whether the presence of any of these substructures is associated with the presence or absence of certain chemical properties, such as melting point or heat of formation. Substructure mining, which is a branch of data mining that analyzes such data, is considered in Section 7.5.

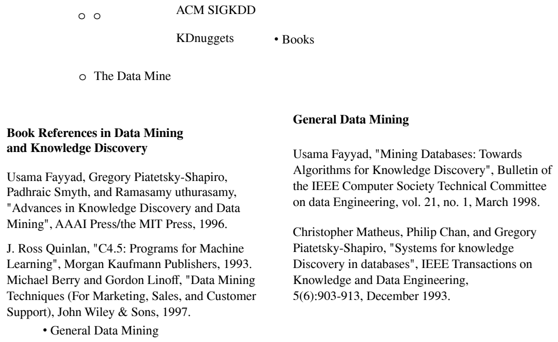
Useful Links:

· Bibliography **Knowledge** **Discovery and**

(Gets updated frequently, so visit often!)

• Other Useful Web sites

Data Mining Bibliography



○ ○ ACM SIGKDD

KDnuggets • Books

○ The Data Mine

Book References in Data Mining and Knowledge Discovery

Usama Fayyad, Gregory Piatetsky-Shapiro, Padhraic Smyth, and Ramasamy uthurasamy, "Advances in Knowledge Discovery and Data Mining", AAAI Press/the MIT Press, 1996.

J. Ross Quinlan, "C4.5: Programs for Machine Learning", Morgan Kaufmann Publishers, 1993. Michael Berry and Gordon Linoff, "Data Mining Techniques (For Marketing, Sales, and Customer Support), John Wiley & Sons, 1997.

• General Data Mining

General Data Mining

Usama Fayyad, "Mining Databases: Towards Algorithms for Knowledge Discovery", Bulletin of the IEEE Computer Society Technical Committee on data Engineering, vol. 21, no. 1, March 1998.

Christopher Matheus, Philip Chan, and Gregory Piatetsky-Shapiro, "Systems for knowledge Discovery in databases", IEEE Transactions on Knowledge and Data Engineering, 5(6):903-913, December 1993.

(a) Linked Web pages. (b) Benzene molecule. **Figure 2.3.** Different variations of graph data.

Ordered Data

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

Sequential Data Sequential data, also referred to as temporal data, can be thought of as an extension of record data, where each record has a time associated with it. Consider a retail transaction data set that also stores the time at which the transaction took place. This time information makes it possible to find patterns such as “candy sales peak before Halloween.” A time can also be associated with each attribute. For example, each record could be the purchase history of a customer, with a listing of items purchased at different times. Using this information, it is possible to find patterns such as “people who buy DVD players tend to buy DVDs in the period immediately following the purchase.”

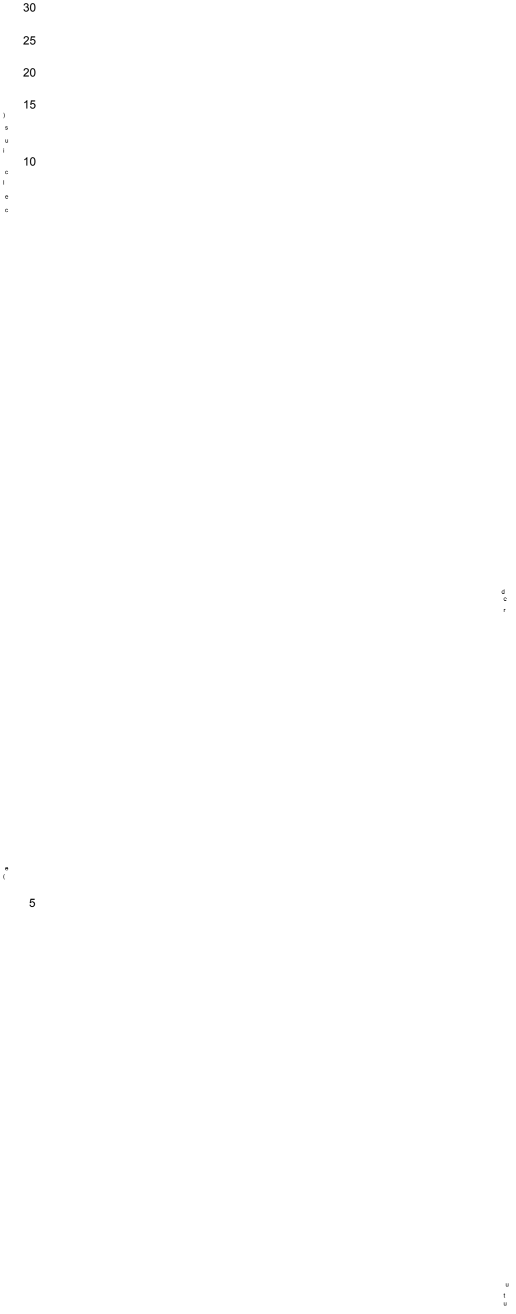
Figure 2.4(a) shows an example of sequential transaction data. There are five different times—*t1*, *t2*, *t3*, *t4*, and *t5* ; three different customers—C1,

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

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

(a) Sequential transaction data.

Minneapolis Average Monthly Temperature (1982–1993)



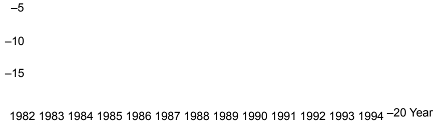
i
t

a
r

t
a

L
e
p
m
e
T

0

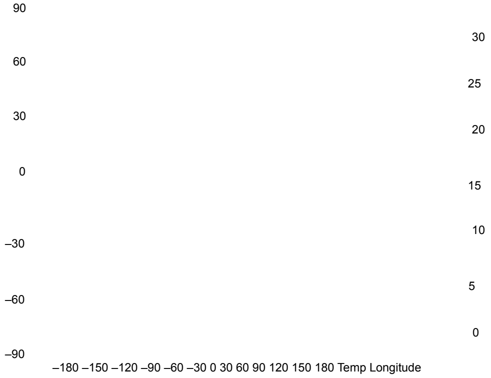


(c) Temperature time series.

GGTTCCGCCTTCAGCCCCGCGC
C
CGCAGGGCCCGCCCCGCGCCGT
C
GAGAAGGGCCCGCCTGGCGGGC
G
GGGGAGGCGGGGCCGCCCGAG
C
CCAACCGAGTCCGACCAGGTGC
C
CCCTCTGCTCGGCCTAGACCTG
A
GCTCATTAGGCGGCAGCGGACA
G

GCCAAGTAGAACACGCGAAGCG
C
TGGGCTGCCTGCTGCGACCAGG
G

(b) Genomic sequence data.



(d) Spatial temperature data.

Figure 2.4. Different variations of ordered data.

C2, and C3; and five different items—A, B, C, D, and E. In the top table, each row corresponds to the items purchased at a particular time by each customer. For instance, at time *t3*, customer C2 purchased items A and D. In the bottom table, the same information is displayed, but each row corresponds to a particular customer. Each row contains information on each transaction involving the customer, where a transaction is considered to be a set of items and the time at which those items were purchased. For example, customer C3 bought items A and C at time *t2*.

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

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

Spatial Data Some objects have spatial attributes, such as positions or ar eas, as well as other types of attributes. An example of spatial data is weather data (precipitation, temperature, pressure) that is collected for a variety of geographical locations. An important aspect of spatial data is spatial auto correlation; i.e., objects that are physically close tend to be similar in other ways as well. Thus, two points on the Earth that are close to each other usually have similar values for temperature and rainfall.

Important examples of spatial data are the science and engineering data sets that are the result of measurements or model output taken at regularly or irregularly distributed points on a two- or three-dimensional grid or mesh. For instance, Earth science data sets record the temperature or pressure mea sured at points (grid cells) on latitude–longitude spherical grids of various resolutions, e.g., 1° by 1°. (See Figure 2.4(d).) As another example, in the simulation of the flow of a gas, the speed and direction of flow can be recorded for each grid point in the simulation.

Handling Non-Record Data

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

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

2.2 Data Quality

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

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

2.2.1 Measurement and Data Collection Issues

It is unrealistic to expect that data will be perfect. There may be problems due to human error, limitations of measuring devices, or flaws in the data collection process. Values or even entire data objects may be missing. In other cases, there may be spurious or duplicate objects; i.e., multiple data objects that all correspond to a single "real" object. For example, there might be two different records for a person who has recently lived at two different addresses. Even if all the data is present and "looks fine," there may be inconsistencies—a person has a height of 2 meters, but weighs only 2 kilograms.

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

Measurement and Data Collection Errors

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

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

Noise and Artifacts

Noise is the random component of a measurement error. It may involve the distortion of a value or the addition of spurious objects. Figure 2.5 shows a time series before and after it has been disrupted by random noise. If a bit

(a) Time series. (b) Time series with noise. **Figure 2.5.** Noise in a time series context.

(a) Three groups of points. (b) With noise points (+) added. **Figure 2.6.** Noise in a spatial context.

more noise were added to the time series, its shape would be lost. Figure 2.6 shows a set of data points before and after some noise points (indicated by '+'s) have been added. Notice that some of the noise points are intermixed with the non-noise points.

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

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

Precision, Bias, and Accuracy

In statistics and experimental science, the quality of the measurement process and the resulting data are measured by precision and bias. We provide the standard definitions, followed by a brief discussion. For the following definitions, we assume that we make repeated measurements of the same underlying quantity and use this set of values to calculate a mean (average) value that serves as our estimate of the true value.

Definition 2.3 (Precision). The closeness of repeated measurements (of the same quantity) to one another.

Definition 2.4 (Bias). A systematic variation of measurements from the quantity being measured.

Precision is often measured by the standard deviation of a set of values, while bias is measured by taking the difference between the mean of the set of values and the known value of the quantity being measured. Bias can only be determined for objects whose measured quantity is known by means external to the current situation. Suppose that we have a standard laboratory weight with a mass of 1g and want to assess the precision and bias of our new laboratory scale. We weigh the mass five times, and obtain the following five values: $\{1.015, 0.990, 1.013, 1.001, 0.986\}$. The mean of these values is 1.001, and hence, the bias is 0.001. The precision, as measured by the standard deviation, is 0.013.

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

Definition 2.5 (Accuracy). The closeness of measurements to the true value of the quantity being measured.

Accuracy depends on precision and bias, but since it is a general concept, there is no specific formula for accuracy in terms of these two quantities. One important aspect of accuracy is the use of significant digits. The goal is to use only as many digits to represent the result of a measurement or calculation as are justified by the precision of the data. For example, if the length of an object is measured with a meter stick whose smallest markings are millimeters, then we should only record the length of data to the nearest mil limeter. The precision of such a measurement would be ± 0.5mm. We do not review the details of working with significant digits, as most readers will have encountered them in previous courses, and they are covered in considerable depth in science, engineering, and statistics textbooks.

Issues such as significant digits, precision, bias, and accuracy are sometimes overlooked, but they are important for data mining as well as statistics and science. Many times, data sets do not come with information on the precision of the data, and furthermore, the programs used for analysis return results without any such information. Nonetheless, without some understanding of the accuracy of the data and the results, an analyst runs the risk of committing serious data analysis blunders.

Outliers

Outliers are either (1) data objects that, in some sense, have characteristics that are different from most of the other data objects in the data set, or (2) values of an attribute that are unusual with respect to the typical values for that attribute. Alternatively, we can speak of anomalous objects or values. There is considerable leeway in the definition of an outlier, and many different definitions have been proposed by the statistics and data mining communities. Furthermore, it is important to distinguish between the notions of noise and outliers. Outliers can be legitimate data objects or values. Thus, unlike noise, outliers may sometimes be of interest. In fraud and network intrusion detection, for example, the goal is to find unusual objects or events from among a large number of normal ones. Chapter 10 discusses anomaly detection in more detail.

Missing Values

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

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

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

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

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

Inconsistent Values

Data can contain inconsistent values. Consider an address field, where both a zip code and city are listed, but the specified zip code area is not contained in that city. It may be that the individual entering this information transposed two digits, or perhaps a digit was misread when the information was scanned
from a handwritten form. Regardless of the cause of the inconsistent values, it is important to detect and, if possible, correct such problems. Some types of inconsistences are easy to detect. For instance, a person’s height should not be negative. In other cases, it can be necessary to consult an external source of information. For example, when an insurance company processes claims for reimbursement, it checks the names and addresses on the reimbursement forms against a database of its customers.

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

The correction of an inconsistency requires additional or redundant information.

Example 2.6 (Inconsistent Sea Surface Temperature). This example illustrates an inconsistency in actual time series data that measures the sea surface temperature (SST) at various points on the ocean. SST data was origi nally collected using ocean-based measurements from ships or buoys, but more recently, satellites have been used to gather the data. To create a long-term data set, both sources of data must be used. However, because the data comes from different sources, the two parts of the data are subtly different. This discrepancy is visually displayed in Figure 2.7, which shows the correlation of SST values between pairs of years. If a pair of years has a positive correlation, then the location corresponding to the pair of years is colored white; otherwise it is colored black. (Seasonal variations were removed from the data since, oth erwise, all the years would be highly correlated.) There is a distinct change in behavior where the data has been put together in 1983. Years within each of the two groups, 1958–1982 and 1983–1999, tend to have a positive correlation with one another, but a negative correlation with years in the other group. This does not mean that this data should not be used, only that the analyst should consider the potential impact of such discrepancies on the data mining analysis.

Duplicate Data

A data set may include data objects that are duplicates, or almost duplicates, of one another. Many people receive duplicate mailings because they appear in a database multiple times under slightly different names. To detect and eliminate such duplicates, two main issues must be addressed. First, if there are two objects that actually represent a single object, then the values of corresponding attributes may differ, and these inconsistent values must be

90

60 65 70 75 80 85 90 95 Year

95

Figure 2.7. Correlation of SST data between pairs of years. White areas indicate positive correlation. Black areas indicate negative correlation.

resolved. Second, care needs to be taken to avoid accidentally combining data objects that are similar, but not duplicates, such as two distinct people with identical names. The term deduplication is often used to refer to the process of dealing with these issues.

In some cases, two or more objects are identical with respect to the attributes measured by the database, but they still represent different objects. Here, the duplicates are legitimate, but may still cause problems for some algorithms if the possibility of identical objects is not specifically accounted for in their design. An example of this is given in Exercise 13 on page 91.

2.2.2 Issues Related to Applications

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

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

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

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

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

2.3 Data Preprocessing

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

- Aggregation
- Sampling
- Dimensionality reduction
- Feature subset selection

- Feature creation
- Discretization and binarization
- Variable transformation

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

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

2.3.1 Aggregation

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

An obvious issue is how an aggregate transaction is created; i.e., how the values of each attribute are combined across all the records corresponding to a particular location to create the aggregate transaction that represents the sales of a single store or date. Quantitative attributes, such as price, are typically aggregated by taking a sum or an average. A qualitative attribute, such as item, can either be omitted or summarized as the set of all the items that were sold at that location.

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

Table 2.4. Data set containing information about customer purchases.

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

number of values for a particular attribute; e.g., reducing the possible values for date from 365 days to 12 months. This type of aggregation is commonly used in Online Analytical Processing (OLAP), which is discussed further in Chapter 3.

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

Example 2.7 (Australian Precipitation). This example is based on pre cipitation in Australia from the period 1982 to 1993. Figure 2.8(a) shows a histogram for the standard deviation of average monthly precipitation for 3,030 0.5° by 0.5° grid cells in Australia, while Figure 2.8(b) shows a histogram for the standard deviation of the average yearly precipitation for the same lo cations. The average yearly precipitation has less variability than the average monthly precipitation. All precipitation measurements (and their standard deviations) are in centimeters.

c a

o c

L o
100

d L

n d

a n

L a
80

f L

o f

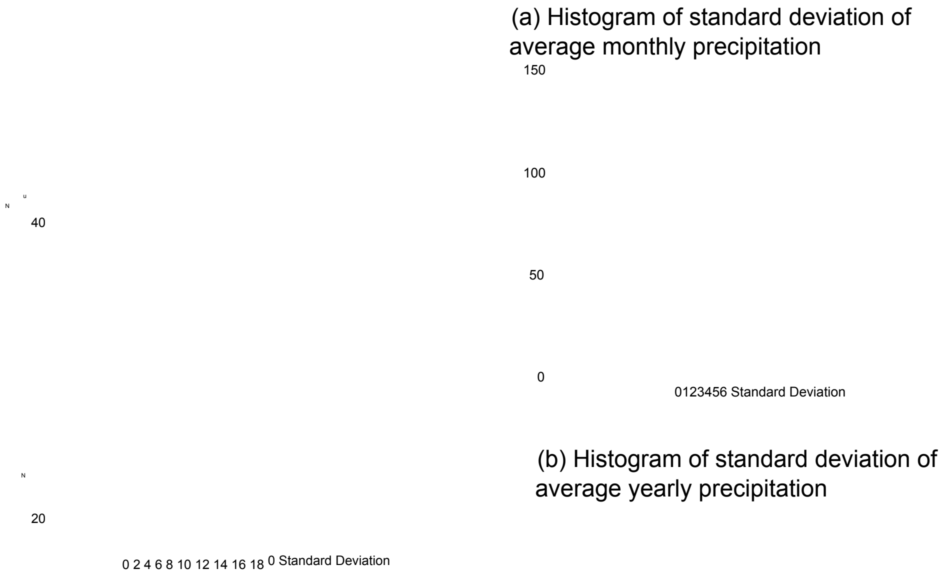
r o

e r

b e
60

m b

u m



(a) Histogram of standard deviation of average monthly precipitation

(b) Histogram of standard deviation of average yearly precipitation

Figure 2.8. Histograms of standard deviation for monthly and yearly precipitation in Australia for the period 1982 to 1993.

2.3.2 Sampling

Sampling is a commonly used approach for selecting a subset of the data objects to be analyzed. In statistics, it has long been used for both the preliminary investigation of the data and the final data analysis. Sampling can also be very useful in data mining. However, the motivations for sampling in statistics and data mining are often different. Statisticians use sampling because obtaining the entire set of data of interest is too expensive or time consuming, while data miners sample because it is too expensive or time con suming to process all the data. In some cases, using a sampling algorithm can reduce the data size to the point where a better, but more expensive algorithm can be used.

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

Sampling Approaches

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

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

Example 2.8 (Sampling and Loss of Information). Once a sampling technique has been selected, it is still necessary to choose the sample size. Larger sample sizes increase the probability that a sample will be representa tive, but they also eliminate much of the advantage of sampling. Conversely, with smaller sample sizes, patterns may be missed or erroneous patterns can be detected. Figure 2.9(a) shows a data set that contains 8000 two-dimensional points, while Figures 2.9(b) and 2.9(c) show samples from this data set of size 2000 and 500, respectively. Although most of the structure of this data set is present in the sample of 2000 points, much of the structure is missing in the sample of 500 points.

(a) 8000 points (b) 2000 points (c) 500 points **Figure 2.9.** Example of the

loss of structure with sampling.

Example 2.9 (Determining the Proper Sample Size). To illustrate that determining the proper sample size requires a methodical approach, consider the following task.

> Given a set of data that consists of a small number of almost equal sized groups, find at least one representative point for each of the groups. Assume that the objects in each group are highly similar to each other, but not very similar to objects in different groups. Also assume that there are a relatively small number of groups, e.g., 10. Figure 2.10(a) shows an idealized set of clusters (groups) from which these points might be drawn.

This problem can be efficiently solved using sampling. One approach is to take a small sample of data points, compute the pairwise similarities between points, and then form groups of points that are highly similar. The desired set of representative points is then obtained by taking one point from each of these groups. To follow this approach, however, we need to determine a sample size that would guarantee, with a high probability, the desired outcome; that is, that at least one point will be obtained from each cluster. Figure 2.10(b) shows the probability of getting one object from each of the 10 groups as the sample size runs from 10 to 60. Interestingly, with a sample size of 20, there is little chance (20%) of getting a sample that includes all 10 clusters. Even with a sample size of 30, there is still a moderate chance (almost 40%) of getting a sample that doesn't contain objects from all 10 clusters. This issue is further explored in the context of clustering by Exercise 4 on page 559.

y
t
i
l
i
b
a
b
o
r
P

(a) Ten groups of points.

1

0.8

0.6

0 10 20 30 40 50 60 70 0 Sample Size

0.4

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

0.2

Figure 2.10. Finding representative points from 10 groups.

Progressive Sampling

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

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

2.3.3 Dimensionality Reduction

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

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

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

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

The Curse of Dimensionality

The curse of dimensionality refers to the phenomenon that many types of data analysis become significantly harder as the dimensionality of the data increases. Specifically, as dimensionality increases, the data becomes increasingly sparse in the space that it occupies. For classification, this can mean that there are not enough data objects to allow the creation of a model that reliably assigns a class to all possible objects. For clustering, the definitions of density and the distance between points, which are critical for clustering, become less meaningful. (This is discussed further in Sections 9.1.2, 9.4.5, and 9.4.7.) As a result, many clustering and classification algorithms (and other data analysis algorithms) have trouble with high-dimensional data—reduced classification accuracy and poor quality clusters.

Linear Algebra Techniques for Dimensionality Reduction

Some of the most common approaches for dimensionality reduction, particularly for continuous data, use techniques from linear algebra to project the data from a high-dimensional space into a lower-dimensional space. Principal Components Analysis (PCA) is a linear algebra technique for continuous attributes that finds new attributes (principal components) that (1) are linear combinations of the original attributes, (2) are orthogonal (perpendicular) to each other, and (3) capture the maximum amount of variation in the data. For example, the first two principal components capture as much of the variation in the data as is possible with two orthogonal attributes that are linear combi nations of the original attributes. Singular Value Decomposition (SVD) is a linear algebra technique that is related to PCA and is also commonly used for dimensionality reduction. For additional details, see Appendices A and B.

2.3.4 Feature Subset Selection

Another way to reduce the dimensionality is to use only a subset of the fea tures. While it might seem that such an approach would lose information, this is not the case if redundant and irrelevant features are present. Redundant features duplicate much or all of the information contained in one or more other attributes. For example, the purchase price of a product and the amount of sales tax paid contain much of the same information. Irrelevant features contain almost no useful information for the data mining task at hand. For instance, students' ID numbers are irrelevant to the task of predicting stu dents' grade point averages. Redundant and irrelevant features can reduce classification accuracy and the quality of the clusters that are found.

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

Embedded approaches Feature selection occurs naturally as part of the data mining algorithm. Specifically, during the operation of the data mining algorithm, the algorithm itself decides which attributes to use and which to ignore. Algorithms for building decision tree classifiers, which are discussed in Chapter 4, often operate in this manner.

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

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

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

An Architecture for Feature Subset Selection

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

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

An integral part of the search is an evaluation step to judge how the current subset of features compares to others that have been considered. This requires an evaluation measure that attempts to determine the goodness of a subset of attributes with respect to a particular data mining task, such as classification

Procedure

Selected
Attributes

Attributes

Validation

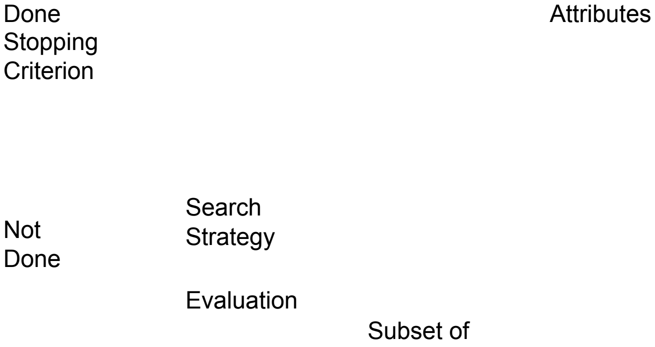


Figure 2.11. Flowchart of a feature subset selection process.

or clustering. For the filter approach, such measures attempt to predict how well the actual data mining algorithm will perform on a given set of attributes. For the wrapper approach, where evaluation consists of actually running the target data mining application, the subset evaluation function is simply the criterion normally used to measure the result of the data mining.

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

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

Feature Weighting

Feature weighting is an alternative to keeping or eliminating features. More important features are assigned a higher weight, while less important features are given a lower weight. These weights are sometimes assigned based on do main knowledge about the relative importance of features. Alternatively, they may be determined automatically. For example, some classification schemes, such as support vector machines (Chapter 5), produce classification models in which each feature is given a weight. Features with larger weights play a more important role in the model. The normalization of objects that takes place when computing the cosine similarity (Section 2.4.5) can also be regarded as a type of feature weighting.

2.3.5 Feature Creation

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

Feature Extraction

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

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

1

0.5 1

5

0.5 0

0 0.2 0.4 0.6 0.8 1 Time (seconds)

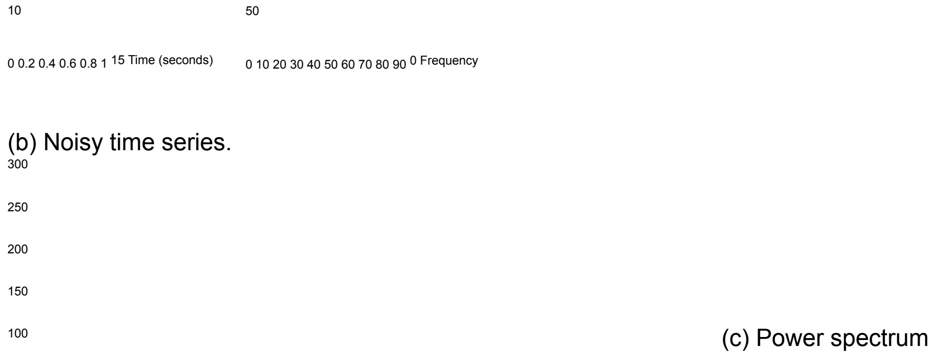
(a) Two time series.

0

15

5

10



(b) Noisy time series.

(c) Power spectrum

Figure 2.12. Application of the Fourier transform to identify the underlying frequencies in time series data.

Mapping the Data to a New Space

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

Example 2.10 (Fourier Analysis). The time series presented in Figure 2.12(b) is the sum of three other time series, two of which are shown in Figure 2.12(a) and have frequencies of 7 and 17 cycles per second, respectively. The third time series is random noise. Figure 2.12(c) shows the power spectrum that can be computed after applying a Fourier transform to the original time series. (Informally, the power spectrum is proportional to the square of each frequency attribute.) In spite of the noise, there are two peaks that correspond to the periods of the two original, non-noisy time series. Again, the main point is that better features can reveal important aspects of the data.

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

Feature Construction

Sometimes the features in the original data sets have the necessary information, but it is not in a form suitable for the data mining algorithm. In this situation, one or more new features constructed out of the original features can be more useful than the original features.

Example 2.11 (Density). To illustrate this, consider a data set consisting of information about historical artifacts, which, along with other information, contains the volume and mass of each artifact. For simplicity, assume that these artifacts are made of a small number of materials (wood, clay, bronze, gold) and that we want to classify the artifacts with respect to the material of which they are made. In this case, a density feature constructed from the mass and volume features, i.e., *density* = *mass/volume*, would most directly yield an accurate classification. Although there have been some attempts to automatically perform feature construction by exploring simple mathematical combinations of existing attributes, the most common approach is to construct features using domain expertise.

2.3.6 Discretization and Binarization

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

As with feature selection, the best discretization and binarization approach is the one that "produces the best result for the data mining algorithm that will be used to analyze the data." It is typically not practical to apply such a criterion directly. Consequently, discretization or binarization is performed in

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

Categorical Value	Integer Value	x_1	x_2	x_3
awful	0	0	0	0

poor	1	0	0	1
OK	2	0	1	0
good	3	0	1	1
great	4	1	0	0

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

Categorical Value	Integer Value	x_1	x_2	x_3	x_4	x_5
awful	0	1	0	0	0	0
poor	1	0	1	0	0	0
OK	2	0	0	1	0	0
good	3	0	0	0	1	0
great	4	0	0	0	0	1

a way that satisfies a criterion that is thought to have a relationship to good performance for the data mining task being considered.

Binarization

A simple technique to binarize a categorical attribute is the following: If there are m categorical values, then uniquely assign each original value to an integer in the interval $[0, m - 1]$. If the attribute is ordinal, then order must be maintained by the assignment. (Note that even if the attribute is originally represented using integers, this process is necessary if the integers are not in the interval $[0, m-1]$.) Next, convert each of these m integers to a binary number. Since $n = \log_2(m)$ binary digits are required to represent these integers, represent these binary numbers using n binary attributes. To illustrate, a categorical variable with 5 values *{awful*, *poor*, *OK*, *good*, *great}* would require three binary variables x_1, x_2, and x_3. The conversion is shown in Table 2.5.

Such a transformation can cause complications, such as creating unintended relationships among the transformed attributes. For example, in Table 2.5, attributes x_2 and x_3 are correlated because information about the *good* value is encoded using both attributes. Furthermore, association analysis re quires asymmetric binary attributes, where only the presence of the attribute (value = 1) is important. For association problems, it is therefore necessary to introduce one binary attribute for each categorical value, as in Table 2.6. If the number of resulting attributes is too large, then the techniques described below can be used to reduce the number of categorical values before binarization. Likewise, for association problems, it may be necessary to replace a single binary attribute with two asymmetric binary attributes. Consider a binary attribute that records a person's gender, male or female. For traditional as sociation rule algorithms, this information needs to be transformed into two asymmetric binary attributes, one that is a 1 only when the person is male and one that is a 1 only when the person is female. (For asymmetric binary attributes, the information representation is somewhat inefficient in that two bits of storage are required to represent each bit of information.)

Discretization of Continuous Attributes

Discretization is typically applied to attributes that are used in classification or association analysis. In general, the best discretization depends on the algo rithm being used, as well as the other attributes being considered. Typically, however, the discretization of an attribute is considered in isolation.

Transformation of a continuous attribute to a categorical attribute involves two subtasks: deciding how many categories to have and determining how to map the values of the continuous attribute to these categories. In the first step, after the values of the continuous attribute are sorted, they are then divided into n intervals by specifying $n - 1$ split points. In the second, rather trivial step, all the values in one interval are mapped to the same categorical value. Therefore, the problem of discretization is one of deciding how many split points to choose and where to place them. The result can be represented either as a set of intervals $\{(x_0, x_1],(x_1, x_2],...,(x_{n-1}, x_n)\}$, where x_0 and x_n may be $+\infty$ or $-\infty$, respectively, or equivalently, as a series of inequalities $x_0 < x \leq x_1$, ..., $x_{n-1} <x<x_n$.

Unsupervised Discretization A basic distinction between discretization methods for classification is whether class information is used (supervised) or not (unsupervised). If class information is not used, then relatively simple approaches are common. For instance, the equal width approach divides the range of the attribute into a user-specified number of intervals each having the same width. Such an approach can be badly affected by outliers, and for that reason, an equal frequency (equal depth) approach, which tries to put the same number of objects into each interval, is often preferred. As another example of unsupervised discretization, a clustering method, such as K-means (see Chapter 8), can also be used. Finally, visually inspecting the data can sometimes be an effective approach.

Example 2.12 (Discretization Techniques). This example demonstrates how these approaches work on an actual data set. Figure 2.13(a) shows data points belonging to four different groups, along with two outliers—the large dots on either end. The techniques of the previous paragraph were applied to discretize the *x* values of these data points into four categorical values. (Points in the data set have a random *y* component to make it easy to see how many points are in each group.) Visually inspecting the data works quite well, but is not automatic, and thus, we focus on the other three approaches. The split points produced by the techniques equal width, equal frequency, and K-means are shown in Figures 2.13(b), 2.13(c), and 2.13(d), respectively. The split points are represented as dashed lines. If we measure the performance of a discretization technique by the extent to which different objects in different groups are assigned the same categorical value, then K-means performs best, followed by equal frequency, and finally, equal width.

Supervised Discretization The discretization methods described above are usually better than no discretization, but keeping the end purpose in mind and using additional information (class labels) often produces better results. This should not be surprising, since an interval constructed with no knowledge of class labels often contains a mixture of class labels. A conceptually simple approach is to place the splits in a way that maximizes the purity of the intervals. In practice, however, such an approach requires potentially arbitrary decisions about the purity of an interval and the minimum size of an interval. To overcome such concerns, some statistically based approaches start with each attribute value as a separate interval and create larger intervals by merging adjacent intervals that are similar according to a statistical test. Entropy based approaches are one of the most promising approaches to discretization, and a simple approach based on entropy will be presented.

First, it is necessary to define entropy. Let *k* be the number of different class labels, m_i be the number of values in the i^{th} interval of a partition, and m_{ij} be the number of values of class *j* in interval *i*. Then the entropy e_i of the i^{th} interval is given by the equation

$$p_{ij} \log_2 p_{ij}$$

$$e_i = \;{}^{k}\;{}_{i=1}\;,$$

where $p_{ij} = m_{ij}/m_i$ is the probability (fraction of values) of class *j* in the i^{th} interval. The total entropy, *e*, of the partition is the weighted average of the individual interval entropies, i.e., 0 5 10 15 20 (a) Original data. 0 5 10 15 20 (b) Equal width discretization.

0 5 10 15 20 (c) Equal frequency 0 5 10 15 20 (d) K-means discretization.

discretization.

Figure 2.13. Different discretization techniques.

i=1

w_ie_i,

n

e =

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

A simple approach for partitioning a continuous attribute starts by bisect ing the initial values so that the resulting two intervals give minimum entropy. This technique only needs to consider each value as a possible split point, be cause it is assumed that intervals contain ordered sets of values. The splitting process is then repeated with another interval, typically choosing the interval with the worst (highest) entropy, until a user-specified number of intervals is reached, or a stopping criterion is satisfied.

Example 2.13 (Discretization of Two Attributes). This method was used to independently discretize both the x and y attributes of the two dimensional data shown in Figure 2.14. In the first discretization, shown in Figure 2.14(a), the *x* and *y* attributes were both split into three intervals. (The dashed lines indicate the split points.) In the second discretization, shown in Figure 2.14(b), the *x* and *y* attributes were both split into five intervals.

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

Categorical Attributes with Too Many Values

Categorical attributes can sometimes have too many values. If the categorical attribute is an ordinal attribute, then techniques similar to those for con tinuous attributes can be used to reduce the number of categories. If the categorical attribute is nominal, however, then other approaches are needed. Consider a university that has a large number of departments. Consequently, a *department name* attribute might have dozens of different values. In this situation, we could use our knowledge of the relationships among different departments to combine departments into larger groups, such as *engineering*, *social sciences*, or *biological sciences*. If domain knowledge does not serve as a useful guide or such an approach results in poor classification performance, then it is necessary to use a more empirical approach, such as grouping values

1

3

y y

5

5 2

4

1

4

0

3 2

0
0 1 2 3 4 5 x

(a) Three intervals

(b) Five intervals

0 1 2 3 4 5 x

Figure 2.14. Discretizing *x* and *y* attributes for four groups (classes) of points.

together only if such a grouping results in improved classification accuracy or achieves some other data mining objective.

2.3.7 Variable Transformation

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

Simple Functions

For this type of variable transformation, a simple mathematical function is applied to each value individually. If x is a variable, then examples of such transformations include x^k, log x, e^x, $\sqrt{x}$, $1/x$, sin x, or $|x|$. In statistics, variable transformations, especially *sqrt*, *log*, and $1/x$, are often used to transform data that does not have a Gaussian (normal) distribution into data that does. While this can be important, other reasons often take precedence in data mining. Suppose the variable of interest is the number of data bytes in a session, and the number of bytes ranges from 1 to 1 billion. This is a huge range, and it may be advantageous to compress it by using a $\log_{10}$ transformation. In this case, sessions that transferred 10^8 and 10^9 bytes would be more similar to each other than sessions that transferred 10 and 1000 bytes ($9 - 8=1$ versus $3 - 1 = 2$). For some applications, such as network intrusion detection, this may be what is desired, since the first two sessions most likely represent transfers of large files, while the latter two sessions could be two quite distinct types of sessions.

Variable transformations should be applied with caution since they change the nature of the data. While this is what is desired, there can be problems if the nature of the transformation is not fully appreciated. For instance, the transformation $1/x$ reduces the magnitude of values that are 1 or larger, but increases the magnitude of values between 0 and 1. To illustrate, the values $\{1, 2, 3\}$ go to $\{1, \frac{1}{2}, \frac{1}{3}\}$, but the values $\{1, \frac{1}{2}, \frac{1}{3}\}$ go to $\{1, 2, 3\}$. Thus, for all sets of values, the transformation $1/x$ reverses the order. To help clarify the effect of a transformation, it is important to ask questions such as the following: Does the order need to be maintained? Does the transformation apply to all values, especially negative values and 0? What is the effect of the transformation on the values between 0 and 1? Exercise 17 on page 92 explores other aspects of variable transformation.

Normalization or Standardization

Another common type of variable transformation is the standardization or normalization of a variable. (In the data mining community the terms are often used interchangeably. In statistics, however, the term normalization can be confused with the transformations used for making a variable normal, i.e., Gaussian.) The goal of standardization or normalization is to make an en tire set of values have a particular property. A traditional example is that of "standardizing a variable" in statistics. If x is the mean (average) of the attribute values and s_x is their standard deviation, then the transformation $x \;\; = (x - x)/s_x$ creates a new variable that has a mean of 0 and a standard deviation of 1. If different variables are to be combined in some way, then such a transformation is often necessary to avoid having a variable with large values dominate the results of the calculation. To illustrate, consider compar ing people based on two variables: age and income. For any two people, the difference in income will likely be much higher in absolute terms (hundreds or thousands of dollars) than the difference in age (less than 150). If the differ ences in the range of values of age and income are not taken into account, then the comparison between people will be dominated by differences in income. In particular, if the similarity or dissimilarity of two people is calculated using the similarity or dissimilarity measures defined later in this chapter, then in many cases, such as that of Euclidean distance, the income values will dominate the calculation.

The mean and standard deviation are strongly affected by outliers, so the above transformation is often modified. First, the mean is replaced by the median, i.e., the middle value. Second, the standard deviation is replaced by the absolute standard deviation. Specifically, if x is a variable, then the absolute standard deviation of x is given by $\sigma_A = \; {}^{m}_{i=1} |x_i - \mu|$, where x_i is the i^{th} value of the variable, m is the number of objects, and μ is either the mean or median. Other approaches for computing estimates of the location (center) and spread of a set of values in the presence of outliers are described in Sections 3.2.3 and 3.2.4, respectively. These measures can also be used to define a standardization transformation.

2.4 Measures of Similarity and Dissimilarity

Similarity and dissimilarity are important because they are used by a number of data mining techniques, such as clustering, nearest neighbor classification, and anomaly detection. In many cases, the initial data set is not needed once these similarities or dissimilarities have been computed. Such approaches can be viewed as transforming the data to a similarity (dissimilarity) space and then performing the analysis.

We begin with a discussion of the basics: high-level definitions of similarity and dissimilarity, and a discussion of how they are related. For convenience, the term proximity is used to refer to either similarity or dissimilarity. Since the proximity between two objects is a function of the proximity between the corresponding attributes of the two objects, we first describe how to measure the proximity between objects having only one simple attribute, and then consider proximity measures for objects with multiple attributes. This in cludes measures such as correlation and Euclidean distance, which are useful for dense data such as time series or two-dimensional points, as well as the Jaccard and cosine similarity measures, which are useful for sparse data like documents. Next, we consider several important issues concerning proximity measures. The section concludes with a brief discussion of how to select the right proximity measure.

2.4.1 Basics

Definitions

Informally, the similarity between two objects is a numerical measure of the degree to which the two objects are alike. Consequently, similarities are *higher* for pairs of objects that are more alike. Similarities are usually non-negative and are often between 0 (no similarity) and 1 (complete similarity).

The dissimilarity between two objects is a numerical measure of the de gree to which the two objects are different. Dissimilarities are *lower* for more similar pairs of objects. Frequently, the term distance is used as a synonym for dissimilarity, although, as we shall see, distance is often used to refer to a special class of dissimilarities. Dissimilarities sometimes fall in the interval [0*,* 1], but it is also common for them to range from 0 to ∞.

Transformations

Transformations are often applied to convert a similarity to a dissimilarity, or vice versa, or to transform a proximity measure to fall within a particular range, such as [0,1]. For instance, we may have similarities that range from 1 to 10, but the particular algorithm or software package that we want to use may be designed to only work with dissimilarities, or it may only work with similarities in the interval [0,1]. We discuss these issues here because we will employ such transformations later in our discussion of proximity. In addi tion, these issues are relatively independent of the details of specific proximity measures.

Frequently, proximity measures, especially similarities, are defined or trans formed to have values in the interval [0,1]. Informally, the motivation for this is to use a scale in which a proximity value indicates the fraction of similarity (or dissimilarity) between two objects. Such a transformation is often relatively straightforward. For example, if the similarities between objects range from 1 (not at all similar) to 10 (completely similar), we can make them fall within the range [0*,* 1] by using the transformation *s* = (*s*−1)/9, where *s* and *s* are the original and new similarity values, respectively. In the more general case, the transformation of similarities to the interval [0*,* 1] is given by the expression *s* = (*s*−*min s*)/(*max s*−*min s*), where *max s* and *min s* are the maximum and minimum similarity values, respectively. Likewise, dissimilarity measures with a finite range can be mapped to the interval [0,1] by using the formula *d* = (*d* − *min d*)/(*max d* − *min d*).

There can be various complications in mapping proximity measures to the interval [0*,* 1], however. If, for example, the proximity measure originally takes
values in the interval [0,∞], then a non-linear transformation is needed and values will not have the same relationship to one another on the new scale. Consider the transformation *d* = *d*/(1 + *d*) for a dissimilarity measure that ranges from 0 to ∞. The dissimilarities 0, 0.5, 2, 10, 100, and 1000 will be transformed into the new dissimilarities 0, 0.33, 0.67, 0.90, 0.99, and 0.999, respectively. Larger values on the original dissimilarity scale are compressed into the range of values near 1, but whether or not this is desirable depends on the application. Another complication is that the meaning of the proximity measure may be changed. For example, correlation, which is discussed later, is a measure of similarity that takes values in the interval [-1,1]. Mapping these values to the interval [0,1] by taking the absolute value loses information about the sign, which can be important in some applications. See Exercise 22 on page 94.

Transforming similarities to dissimilarities and vice versa is also relatively straightforward, although we again face the issues of preserving meaning and changing a linear scale into a non-linear scale. If the similarity (or dissimilarity) falls in the interval [0,1], then the dissimilarity can be defined as *d* = 1−*s* (*s* = 1 − *d*). Another simple approach is to define similarity as the negative of the dissimilarity (or vice versa). To illustrate, the dissimilarities 0, 1, 10, and 100 can be transformed into the similarities 0, −1, −10, and −100, respectively.

The similarities resulting from the negation transformation are not restricted to the range [0*,* 1], but if that is desired, then transformations such as $s = {}^{1}$

$_{d+1}$, $s = e^{-d}$, or $s = 1 - {}^{d-min\ d}$

$s = {}^{1}$ transformation

$_{max\ d - min\ d}$ can be used. For the

$_{d+1}$, the dissimilarities 0, 1, 10, 100 are transformed into 1, 0.5, 0.09, 0.01, respectively. For $s = e^{-d}$, they become 1.00, 0.37, 0.00, 0.00, respectively, while for $s = 1 - {}^{d - min\ d}$

$_{max\ d - min\ d}$ they become 1.00, 0.99, 0.00, 0.00, respectively. In this discussion, we have focused on converting dissimilarities to similarities. Conversion in the opposite direction is considered in Exercise 23 on page 94.

In general, any monotonic decreasing function can be used to convert dis similarities to similarities, or vice versa. Of course, other factors also must be considered when transforming similarities to dissimilarities, or vice versa, or when transforming the values of a proximity measure to a new scale. We have mentioned issues related to preserving meaning, distortion of scale, and requirements of data analysis tools, but this list is certainly not exhaustive.

2.4.2 Similarity and Dissimilarity between Simple Attributes

The proximity of objects with a number of attributes is typically defined by combining the proximities of individual attributes, and thus, we first discuss proximity between objects having a single attribute. Consider objects described by one nominal attribute. What would it mean for two such objects to be similar? Since nominal attributes only convey information about the distinctness of objects, all we can say is that two objects either have the same value or they do not. Hence, in this case similarity is traditionally defined as 1 if attribute values match, and as 0 otherwise. A dissimilarity would be defined in the opposite way: 0 if the attribute values match, and 1 if they do not.

For objects with a single ordinal attribute, the situation is more complicated because information about order should be taken into account. Consider an attribute that measures the quality of a product, e.g., a candy bar, on the scale *{poor*, *fair*, *OK*, *good*, *wonderful}*. It would seem reasonable that a prod uct, P1, which is rated *wonderful*, would be closer to a product P2, which is rated *good*, than it would be to a product P3, which is rated *OK*. To make this observation quantitative, the values of the ordinal attribute are often mapped to successive integers, beginning at 0 or 1, e.g., *{poor*=0, *fair*=1, *OK*=2, good=3, *wonderful*=4*}*. Then, $d(\text{P1},\text{P2}) = 3 - 2 = 1$ or, if we want the dis similarity to fall between 0 and 1, $d(\text{P1},\text{P2}) = {}^{3-2}$

$_4$ = 0.25. A similarity for ordinal attributes can then be defined as $s = 1 - d$.

This definition of similarity (dissimilarity) for an ordinal attribute should make the reader a bit uneasy since this assumes equal intervals, and this is not so. Otherwise, we would have an interval or ratio attribute. Is the difference between the values *fair* and *good* really the same as that between the values *OK* and *wonderful*? Probably not, but in practice, our options are limited, and in the absence of more information, this is the standard approach for defining proximity between ordinal attributes.

For interval or ratio attributes, the natural measure of dissimilarity between two objects is the absolute difference of their values. For example, we might compare our current weight and our weight a year ago by saying “I am ten pounds heavier.” In cases such as these, the dissimilarities typically range from 0 to ∞, rather than from 0 to 1. The similarity of interval or ratio attributes is typically expressed by transforming a similarity into a dissimilarity, as previously described.

Table 2.7 summarizes this discussion. In this table, *x* and *y* are two objects that have one attribute of the indicated type. Also, *d*(*x, y*) and *s*(*x, y*) are the dissimilarity and similarity between *x* and *y*, respectively. Other approaches are possible; these are the most common ones.

The following two sections consider more complicated measures of proximity between objects that involve multiple attributes: (1) dissimilarities between data objects and (2) similarities between data objects. This division allows us to more naturally display the underlying motivations for employing various proximity measures. We emphasize, however, that similarities can

Table 2.7. Similarity and dissimilarity for simple attributes

Attribute Type	Dissimilarity	Similarity
Nominal	$d =$ 0 if $x = y$ 1 if $x = y$	$s =$ 1 if $x = y$ 0 if $x = y$
Ordinal	$d = \lvert x - y\rvert/(n - 1)$ (values mapped to integers 0 to $n-1$, where n is the number of values)	$s = 1 - d$
Interval or Ratio	$d = \lvert x - y\rvert$	$s = -d$, $s = \frac{1}{1+d}$, $s = e^{-d}$, $s = 1 - \frac{d-\min d}{\max d-\min d}$

be transformed into dissimilarities and vice versa using the approaches described earlier.

2.4.3 Dissimilarities between Data Objects

In this section, we discuss various kinds of dissimilarities. We begin with a discussion of distances, which are dissimilarities with certain properties, and then provide examples of more general kinds of dissimilarities.

Distances

We first present some examples, and then offer a more formal description of distances in terms of the properties common to all distances. The Euclidean distance, *d*, between two points, x and y, in one-, two-, three-, or higher dimensional space, is given by the following familiar formula:

$${}^{n}\ {}_{k=1}$$

$$d(\mathrm{x}, \mathrm{y}) = \qquad (x_k - y_k)^2, \ (2.1)$$

where *n* is the number of dimensions and x_k and y_k are, respectively, the k^{th} attributes (components) of *x* and *y*. We illustrate this formula with Figure 2.15 and Tables 2.8 and 2.9, which show a set of points, the *x* and *y* coordinates of these points, and the distance matrix containing the pairwise distances of these points.

The Euclidean distance measure given in Equation 2.1 is generalized by the Minkowski distance metric shown in Equation 2.2,

$${}^{n}\ {}_{k=1} \qquad |x_k - y_k|^r$$

$${}_{1/r} \qquad , \ (2.2)$$

$$d(\mathrm{x}, \mathrm{y}) =$$

where *r* is a parameter. The following are the three most common examples of Minkowski distances.

- *r* = 1. City block (Manhattan, taxicab, L_1 norm) distance. A common example is the Hamming distance, which is the number of bits that are different between two objects that have only binary attributes, i.e., between two binary vectors.

- *r* = 2. Euclidean distance (L_2 norm).

- $r = \infty$. Supremum (L_{max} or L_∞ norm) distance. This is the maximum difference between any attribute of the objects. More formally, the L_∞ distance is defined by Equation 2.3

$$d(x, y) = \lim_{r\to\infty} \; {}^{n}{}_{k=1} \; |x_k - y_k|^r \; {}^{1/r} . \quad (2.3)$$

The r parameter should not be confused with the number of dimensions (at tributes) n. The Euclidean, Manhattan, and supremum distances are defined for all values of n: 1, 2, 3,..., and specify different ways of combining the differences in each dimension (attribute) into an overall distance.

Tables 2.10 and 2.11, respectively, give the proximity matrices for the L_1 and L_∞ distances using data from Table 2.8. Notice that all these distance matrices are symmetric; i.e., the ij^{th} entry is the same as the ji^{th} entry. In Table 2.9, for instance, the fourth row of the first column and the fourth column of the first row both contain the value 5.1.

Distances, such as the Euclidean distance, have some well-known proper ties. If d(x, y) is the distance between two points, x and y, then the following properties hold.

1. Positivity

   (a) d(x, x) ≥ 0 for all x and y,

   (b) d(x, y) = 0 only if x = y.

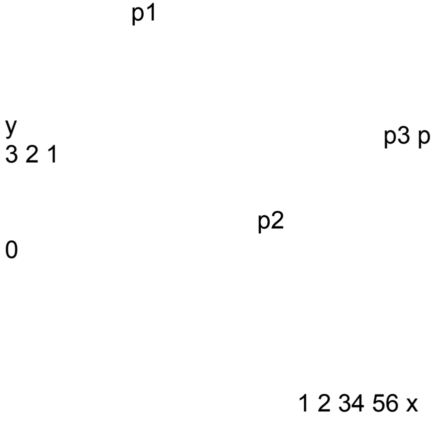


Figure 2.15. Four two-dimensional points.

Table 2.8. *x* and *y* coordinates of four points.

point	*x* coordinate	*y* coordinate
p1	0	2
p2	2	0
p3	3	1
p4	5	1

Table 2.9. Euclidean distance matrix for Table 2.8.

	p1	p2	p3	p4
p1	0.0	2.8	3.2	5.1
p2	2.8	0.0	1.4	3.2
p3	3.2	1.4	0.0	2.0
p4	5.1	3.2	2.0	0.0

Table 2.10. L_1 distance matrix for Table 2.8.

L_1	p1	p2	p3	p4
p1	0.0	4.0	4.0	6.0
p2	4.0	0.0	2.0	4.0
p3	4.0	2.0	0.0	2.0
p4	6.0	4.0	2.0	0.0

Table 2.11. L_∞ distance matrix for Table 2.8.

L_∞	p1	p2	p3	p4
p1	0.0	2.0	3.0	5.0
p2	2.0	0.0	1.0	3.0
p3	3.0	1.0	0.0	2.0
p4	5.0	3.0	2.0	0.0

2. Symmetry

d(x*,* y) = *d*(y*,* x) for all x and y.

3. Triangle Inequality

d(x*,* z) ≤ *d*(x*,* y) + *d*(y*,* z) for all points x, y, and z.

Measures that satisfy all three properties are known as metrics. Some people only use the term distance for dissimilarity measures that satisfy these properties, but that practice is often violated. The three properties described here are useful, as well as mathematically pleasing. Also, if the triangle in equality holds, then this property can be used to increase the efficiency of tech niques (including clustering) that depend on distances possessing this property. (See Exercise 25.) Nonetheless, many dissimilarities do not satisfy one or more of the metric properties. We give two examples of such measures.

Example 2.14 (Non-metric Dissimilarities: Set Differences). This ex ample is based on the notion of the difference of two sets, as defined in set theory. Given two sets *A* and *B*, *A* − *B* is the set of elements of A that are not in *B*. For example, if *A* = *{*1*,* 2*,* 3*,* 4*}* and *B* = *{*2*,* 3*,* 4*}*, then *A* − *B* = *{*1*}* and *B* − *A* = ∅, the empty set. We can define the distance *d* between two sets *A* and *B* as *d*(*A, B*) = *size*(*A* − *B*), where *size* is a function returning the number of elements in a set. This distance measure, which is an integer value greater than or equal to 0, does not satisfy the second part of the pos itivity property, the symmetry property, or the triangle inequality. However, these properties can be made to hold if the dissimilarity measure is modified as follows: *d*(*A, B*) = *size*(*A* − *B*) + *size*(*B* − *A*). See Exercise 21 on page 94.

Example 2.15 (Non-metric Dissimilarities: Time). This example gives a more everyday example of a dissimilarity measure that is not a metric, but that is still useful. Define a measure of the distance between times of the day as follows:

$$d(t_1, t_2) = \begin{matrix} t_2 - t_1 \text{ if } t_1 \leq t_2 \; 24 + (t_2 - \\ t_1) \text{ if } t_1 \geq t_2 \end{matrix} . \quad (2.4)$$

To illustrate, *d*(1PM, 2PM) = 1 hour, while *d*(2PM, 1PM) = 23 hours. Such a definition would make sense, for example, when answering the question: “If an event occurs at 1PM every day, and it is now 2PM, how long do I have to wait for that event to occur again?”

2.4.4 Similarities between Data Objects

For similarities, the triangle inequality (or the analogous property) typically does not hold, but symmetry and positivity typically do. To be explicit, if *s*(x*,* y) is the similarity between points x and y, then the typical properties of similarities are the following:

1. *s*(x*,* y) = 1 only if x = y. (0 ≤ *s* ≤ 1)

2. *s*(x*,* y) = *s*(y*,* x) for all x and y. (Symmetry)

There is no general analog of the triangle inequality for similarity measures. It is sometimes possible, however, to show that a similarity measure can easily be converted to a metric distance. The cosine and Jaccard similarity measures, which are discussed shortly, are two examples. Also, for specific sim ilarity measures, it is possible to derive mathematical bounds on the similarity between two objects that are similar in spirit to the triangle inequality.

Example 2.16 (A Non-symmetric Similarity Measure). Consider an experiment in which people are asked to classify a small set of characters as they flash on a screen. The confusion matrix for this experiment records how often each character is classified as itself, and how often each is classified as another character. For instance, suppose that “0” appeared 200 times and was classified as a “0” 160 times, but as an “o” 40 times. Likewise, suppose that ‘o’ appeared 200 times and was classified as an “o”

170 times, but as “0” only 30 times. If we take these counts as a measure of the similarity between two characters, then we have a similarity measure, but one that is not symmetric. In such situations, the similarity measure is often made symmetric by setting $s(x, y) = s(y, x) = (s(x, y)+s(y, x))/2$, where s indicates the new similarity measure.

2.4.5 Examples of Proximity Measures

This section provides specific examples of some similarity and dissimilarity measures.

Similarity Measures for Binary Data

Similarity measures between objects that contain only binary attributes are called similarity coefficients, and typically have values between 0 and 1. A value of 1 indicates that the two objects are completely similar, while a value of 0 indicates that the objects are not at all similar. There are many rationales for why one coefficient is better than another in specific instances.

Let x and y be two objects that consist of n binary attributes. The com parison of two such objects, i.e., two binary vectors, leads to the following four quantities (frequencies):

f_{00} = the number of attributes where x is 0 and y is 0
f_{01} = the number of attributes where x is 0 and y is 1
f_{10} = the number of attributes where x is 1 and y is 0
f_{11} = the number of attributes where x is 1 and y is 1

Simple Matching Coefficient One commonly used similarity coefficient is the simple matching coefficient (*SMC*), which is defined as

number of attributes = $f_{11} + f_{00}$

SMC = number of matching attribute values $f_{01} + f_{10} + f_{11} + f_{00}$. (2.5)

This measure counts both presences and absences equally. Consequently, the *SMC* could be used to find students who had answered questions similarly on a test that consisted only of true/false questions.

Jaccard Coefficient Suppose that x and y are data objects that represent two rows (two transactions) of a transaction matrix (see Section 2.1.2). If each asymmetric binary attribute corresponds to an item in a store, then a 1 indicates that the item was purchased, while a 0 indicates that the product was not purchased. Since the number of products not purchased by any customer far outnumbers the number of products that were purchased, a similarity measure such as *SMC* would say that all transactions are very similar. As a result, the Jaccard coefficient is frequently used to handle objects consisting of asymmet ric binary attributes. The Jaccard coefficient, which is often symbolized by *J*, is given by the following equation:

number of attributes not involved in 00 matches = f_{11}

$J =$ number of matching presences

$f_{01} + f_{10} + f_{11}$. (2.6)

Example 2.17 (The SMC and Jaccard Similarity Coefficients). To illustrate the difference between these two similarity measures, we calculate *SMC* and *J* for the following two binary vectors.

x = (1, 0, 0, 0, 0, 0, 0, 0, 0, 0)
y = (0, 0, 0, 0, 0, 0, 1, 0, 0, 1)

$f_{01} = 2$ the number of attributes where x was 0 and y was 1
$f_{10} = 1$ the number of attributes where x was 1 and y was 0
$f_{00} = 7$ the number of attributes where x was 0 and y was 0
$f_{11} = 0$ the number of attributes where x was 1 and y was 1

$SMC = f_{11}+f_{00}$

$f_{01}+f_{10}+f_{11}+f_{00} = 0+7$

$2+1+0+7 = 0.7$

$J = f_{11}$

$f_{01}+f_{10}+f_{11} = 0$

$2+1+0 = 0$

Cosine Similarity

Documents are often represented as vectors, where each attribute represents the frequency with which a particular term (word) occurs in the document. It is more complicated than this, of course, since certain common words are ignored and various processing techniques are used to account for different forms of the same word, differing document lengths, and different word frequencies. Even though documents have thousands or tens of thousands of attributes (terms), each document is sparse since it has relatively few non-zero attributes. (The normalizations used for documents do not create a non-zero entry where there was a zero entry; i.e., they preserve sparsity.) Thus, as with transaction data, similarity should not depend on the number of shared 0 values since any two documents are likely to "not contain" many of the same words, and therefore, if 0–0 matches are counted, most documents will be highly similar to most other documents. Therefore, a similarity measure for documents needs to ignores 0–0 matches like the Jaccard measure, but also must be able to handle non-binary vectors. The cosine similarity, defined next, is one of the most common measure of document similarity. If x and y are two document vectors, then

$$\cos(x, y) = x \cdot y$$

x

y

, (2.7)

where · indicates the vector dot product, $x \cdot y = {}^{n}_{k=1} x_k y_k$, and

x

is the

length of vector x,

x

=

$${}^{n}_{k=1} x^2{}_k = \sqrt{x \cdot x}.$$

Example 2.18 (Cosine Similarity of Two Document Vectors). This example calculates the cosine similarity for the following two data objects, which might represent document vectors:

$x = (3, 2, 0, 5, 0, 0, 0, 2, 0, 0)$

$y = (1, 0, 0, 0, 0, 0, 0, 1, 0, 2)$

$x \cdot y = 3 * 1+2 * 0+0 * 0+5 * 0+0 * 0+0 * 0+0 * 0+2 * 1+0 * 0+0 * 2=5$

x

$= \sqrt{3 * 3+2 * 2+0 * 0+5 * 5+0 * 0+0 * 0+0 * 0+2 * 2+0 * 0+0 * 0}=6.48$

y

$= \sqrt{1 * 1+0 * 0+0 * 0+0 * 0+0 * 0+0 * 0+0 * 0+1 * 1+0 * 0+2 * 2}=2.24$ $\cos(x, y) = 0.31$

As indicated by Figure 2.16, cosine similarity really is a measure of the (cosine of the) angle between x and y. Thus, if the cosine similarity is 1, the angle between x and y is 0°, and x and y are the same except for magnitude (length). If the cosine similarity is 0, then the angle between x and y is 90°, and they do not share any terms (words).

x y

θ

Figure 2.16. Geometric illustration of the cosine measure.

Equation 2.7 can be written as Equation 2.8.

x

$\cos(x, y) =$

x

y

.

y

$= x \cdot y$, (2.8)

where x = x/

x

and y = y/

y

. Dividing x and y by their lengths normal izes them to have a length of 1. This means that cosine similarity does not take the *magnitude* of the two data objects into account when computing similarity. (Euclidean distance might be a better choice when magnitude is important.) For vectors with a length of 1, the cosine measure can be calculated by taking a simple dot product. Consequently, when many cosine similarities between objects are being computed, normalizing the objects to have unit length can reduce the time required.

Extended Jaccard Coefficient (Tanimoto Coefficient)

The extended Jaccard coefficient can be used for document data and that re duces to the Jaccard coefficient in the case of binary attributes. The extended Jaccard coefficient is also known as the Tanimoto coefficient. (However, there is another coefficient that is also known as the Tanimoto coefficient.) This co efficient, which we shall represent as *EJ*, is defined by the following equation:

$EJ(x, y) = x \cdot y$

x

^{2} +

y

$^2 - x \cdot y$. (2.9)

Correlation

The correlation between two data objects that have binary or continuous vari ables is a measure of the linear relationship between the attributes of the objects. (The calculation of correlation between attributes, which is more common, can be defined similarly.) More precisely, Pearson's correlation coefficient between two data objects, x and y, is defined by the following equation:

standard deviation(x) ∗ standard deviation(y) = s_{xy}

corr(x, y) = covariance(x, y)

$s_x s_y$, (2.10)

where we are using the following standard statistical notation and definitions:

covariance(x, y) = s_{xy} ^{n} $_{k=1}$ $(x_k - x)(y_k - y)$ (2.11)

= 1 $n - 1$

standard 1 $(x_k - x)^2$ $(y_k -$

$n - 1$

standard deviation(y) $_{n}$ $y)^2$

deviation(x) = s_y = $_{k=1}$ ^{n}

$_{k=1}$

= s_x =

1

$n - 1$

of x y_k is the

1 ^{n} $_{k=1}$

x = n

1 ^{n} $_{k=1}$ mean of y

y = n

x_k is the mean

Example 2.19 (Perfect Correlation). Correlation is always in the range −1 to 1. A correlation of 1 (−1) means that x and y have a perfect positive (negative) linear relationship; that is, $x_k = ay_k + b$, where a and b are con stants. The following two sets of values for x and y indicate cases where the correlation is −1 and +1, respectively. In the first case, the means of x and y were chosen to be 0, for simplicity.

x = (−3, 6, 0, 3, −6)
y =(1, −2, 0, −1, 2)

x = (3, 6, 0, 3, 6)
y = (1, 2, 0, 1, 2)

–1.00 –0.90 –0.80 –0.70 –0.60 0.50 –0.40

–0.30 –0.20 –0.10 0.00 0.10 0.20 0.30

0.40 0.50 0.60 0.70 0.80 0.90 1.00

Figure 2.17. Scatter plots illustrating correlations from −1 to 1.

Example 2.20 (Non-linear Relationships). If the correlation is 0, then there is no linear relationship between the attributes of the two data objects. However, non-linear relationships may still exist. In the following example, x_k = y^2_k, but their correlation is 0.

x = (−3, −2, −1, 0, 1, 2, 3)
y =(9, 4, 1, 0, 1, 4, 9)

Example 2.21 (Visualizing Correlation). It is also easy to judge the cor relation between two data objects x and y by plotting pairs of corresponding attribute values. Figure 2.17 shows a number of these plots when x and y have 30 attributes and the values of these attributes are randomly generated (with a normal distribution) so that the correlation of x and y ranges from −1 to 1. Each circle in a plot represents one of the 30 attributes; its *x* coordinate is the value of one of the attributes for x, while its *y* coordinate is the value of the same attribute for y.

If we transform x and y by subtracting off their means and then normaliz ing them so that their lengths are 1, then their correlation can be calculated by