Data Mining: Concepts and Techniques

2nd Edition

Solution Manual

Jiawei Han and **Micheline Kamber**

The University of Illinois at Urbana-Champaign

Contents

Chapter 1

Introduction

1.11 Exercises

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

(a) Is it another hype?

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

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

(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

■

1.2. Present an example where data mining is crucial to the success of a business. What *data mining functions* does this business need? Can they be performed 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.

■

1.3. 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 the 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 concept hierarchies and metadata (e.g., describing 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 interesting patterns.
- **A graphical user interface** that provides the user with an interactive approach to the data mining system.

■

1.4. 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. Additional differences are detailed in Section 3.1.1 Differences between Operational Databases Systems and Data Warehouses.
- Similarities between a data warehouse and a database: Both are repositories of information, storing huge amounts of persistent data.

■

1.5. 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.

■

1.6. Define each of the following *data mining functionalities*: characterization, discrimination, association and correlation analysis, classification, prediction, clustering, and evolution 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

$$major(X, \text{“}computing\ science\text{”}\text{”}) \Rightarrow owns(X, \text{“}personal\ computer\text{”}) \quad [support = 12\%, confidence = 98\%]$$

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.

- **Classification** differs from **prediction** in that the former constructs a set of models (or functions) that describe and distinguish data classes or concepts, whereas the latter builds a model to predict some missing or unavailable, and often numerical, data values. Their similarity is that they are both tools for prediction: Classification is used for predicting the class label of data objects and prediction is typically used for predicting missing numerical data values.
- **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.
- **Data evolution analysis** describes and models regularities or trends for objects whose behavior changes over time. Although this may include characterization, discrimination, association, classification, or clustering of time-related data, distinct features of such an analysis include time-series data analysis, sequence or periodicity pattern matching, and similarity-based data analysis.

■

1.7. What is the difference between discrimination and classification? Between characterization and clustering? Between classification and prediction? 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 **prediction** in that the former is the process of finding a set of models (or functions) that describe and distinguish data class or concepts 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: Classification is used for predicting the class label of data objects and prediction is typically used for predicting missing numerical data values.

■

1.8. 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.

■

1.9. 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. Concept hierarchies and user beliefs regarding relationships in the data are forms of background knowledge.
- **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.

■

1.10. 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.

■

1.11. *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. Taking fraudulence detection as an example, propose two methods that can be used to detect outliers and discuss which one is more reliable.

Answer:

- Using clustering techniques: After clustering, the different clusters represent the different kinds of data (transactions). The outliers are those data points that do not fall into any cluster. Among the various kinds of clustering methods, density-based clustering may be the most effective. Clustering is detailed in Chapter 8.
- Using prediction (or regression) techniques: Constructed a probability (regression) model based on all of the data. If the predicted value for a data point differs greatly from the given value, then the given value may be consider an outlier.

Outlier detection based on clustering techniques may be more reliable. Because clustering is unsupervised, we do not need to make any assumptions regarding the data distribution (e.g., density-based methods). In contrast, regression (prediction) methods require us to make some assumptions of the data distribution, which may be inaccurate due to insufficient data.

■

1.12. 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:

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

i. Sequences of sensor images of a geographical region along time.

ii. The climate images from satellites.

iii. Data that describe the evolution of natural phenomena, such as forest coverage, forest fire, and so on.

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

The knowledge that can be mined from spatiotemporal data streams really depends on the application. However, one unique type of knowledge about stream data is the patterns of spatial change with respect to the time. For example, the changing of the traffic status of several highway junctions in a city, from the early morning to rush hours and back to off-peak hours, can show clearly where the traffic comes from and goes to and hence, would help the traffic officer plan effective alternative lanes in order to reduce the traffic load. As another example, a sudden appearance of a point in the spectrum space image may indicate that a new planet is being formed. The changing of humidity, temperature, and pressure in climate data may reveal patterns of how a new typhoon is created.

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

One major challenge is how to deal with the continuing large-scale data. Since the data keep flowing in and each snapshot of data is usually huge (e.g., the spectrum image of space), it is [OLD: almost][NEW:

often] impossible to store all of the data. Some aggregation or compression techniques may have to be applied, and old raw data may have to be dropped. Mining under such aggregated (or lossy) data is challenging. In addition, some patterns may occur with respect to a long time period, but it may not be possible to keep the data for such a long duration. Thus, these patterns may not be uncovered. The spatial data sensed may not be so accurate, so the algorithms must have high tolerance with respect to noise.

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

Take mining space images as the application. We seek to observe whether any new planet is being created or any old planet is disappearing. This is a change detection problem. Since the image frames keep coming, that is, $f_1, \ldots, f_t, f_{t+1}, \ldots$, we can simplify the overall problem to that of detecting whether any planet appears or disappears between two consecutive image frames, f_t and f_{t+1}. The algorithm can be sketched as follows. For each incoming frame, f_{t+1}, compare it with the previous frame, f_t.

i. Match the planets in f_{t+1} with f_t.

ii. Detect whether there are any "unmatched" planets (where a planet in one of the two frames does not occur in the other).

iii. If yes, report a planet appearance (if an unmatched planet appears in the new frame) or a planet disappearance (if an unmatched planet appears in the old frame).

In fact, matching between two frames may not be easy because the earth is rotating and thus, the sensed data may have slight variations. Some advanced techniques from image processing may be applied.

The overall skeleton of the algorithm is simple. Each new incoming image frame is only compared with the previous one, satisfying the time and resource constraint. The reported change would be useful since it is [OLD: almost impossible][NEW: infeasible] for astronomers to dig into every frame to detect whether a new planet has appeared or an old one has disappeared.

■

1.13. 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 nontrivial. Therefore, the most popular architecture is currently semitight coupling as it provides a compromise between loose and tight coupling.

■

1.14. 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.

■

1.15. 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.

■

1.16. 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:

Students must research their answers for this question. Major data mining challenges for two applications, that of data streams and bioinformatics, are addressed here.

Data Stream

Data stream analysis presents multiple challenges. First, data streams are continuously flowing in and out as well as changing dynamically. The data analysis system that will successfully take care of this type of data needs to be in real time and able to adapt to changing patterns that might emerge. Another major challenge is that the size of stream data can be huge or even infinite. Because of this size, only a single or small number of scans are typically allowed. For further details on mining data stream, please consult Chapter 8.

Bioinformatics

The field of bioinformatics encompasses many other subfields like genomics, proteomics, molecular biology, and chemi-informatics. Each of these individual subfields has many research challenges. Some of the major challenges of data mining in the field of bioinformatics are outlined as follows. (Due to limitations of space, some of the terminology used here may not be explained.)

- **Data explosion**: Biological data are growing at an exponential rate. It has been estimated that genomic and proteomic data are doubling every 12 months. Most of these data are scattered around in unstructured and nonstandard forms in various different databases throughout the research community. Many of the biological experiments do not yield exact results and are prone to errors because it is very difficult to model exact biological conditions and processes. For example, the structure of a protein is not rigid and is dependent on its environment. Hence, the structures determined by nuclear magnetic resonance (NMR) or crystallography experiments may not represent the exact structure of the protein. Since these experiments are performed in parallel by many institutions and scientists, they may each yield slightly different structures. The consolidation and validation of these conflicting data is a difficult challenge. Some research labs have come up with public domain repositories of data, such as the Protein Data Bank (PDB). These have become very popular in the past few years. However, due to concerns of Intellectual Property, a great deal of useful biological information is buried in proprietary databases within large pharmaceutical companies.
- **Text mining from research publications/repositories**: Most of the data generated in the biological research community is from experiments. Most of the results are published, but they are seldom recorded in databases with the experiment details (who, when, how, etc.). Hence, a great deal of useful information is buried in published and unpublished literature. This has given rise to the need for the development of text mining systems. For example, many experimental results regarding protein interactions have been published. Mining this information may provide crucial insight into biological pathways and help predict potential interactions. The extraction and development of domain-specific ontologies is also another related research challenge.
- **Mining large databases of compounds/molecules**: The major steps in a drug discovery phase include target identification, target validation, lead discovery, and lead optimization. The most time-consuming step is the lead discovery phase. In this step, large databases of compounds are needed to be mined to identify potential lead candidates that will suitably interact with the potential target. Currently, due to the lack of effective data mining systems, this step involves many trial-and-error iterations of wet lab or protein assay experiments. These experiments are highly time-consuming and costly. Hence, one of the current challenges in bioinformatics includes the development of intelligent and computational data mining systems that can eliminate false positives and generate more true positives before the wet lab experimentation stage. This task is particularly challenging, because it involves the development of a mining/screening system that can identify compounds that can dock[1] well with the target compound. The docking problem is an especially tricky problem, because it is governed by many physical interactions at the molecular level. Some progress has been made regarding pair-wise docking, where large time-consuming Molecular Dynamics (MD) simulation-based optimization

[1]Docking is a term used for computational schemes that attempt to find the best "matching" between two molecules: a receptor and a ligand.

methods can predict docking to a good degree of success. The main problem is the large solution space generated by the complex interactions at the molecular level. The molecular docking problem remains a fairly unsolved problem. The major research challenges in the mining of these interactions include the development of fast and accurate algorithms for screening and ranking these compounds/molecules based on their ability to interact with a given compound/molecule. Other related research areas include protein classification systems based on structure and function.

- **Pattern analysis and classification of microarray data**: A great deal of progress has been made in the past decade regarding the development of algorithms for the analysis of genomic data. Statistical and other methods are available. A large research community in data mining is focusing on adopting these pattern analysis and classification methods for mining microarray and gene expression data.

■

Chapter 2

Data Preprocessing

2.8 Exercises

2.1. *Data quality* can be assessed in terms of accuracy, completeness, and consistency. Propose two other dimensions of data quality.

Answer:

Other dimensions that can be used to assess the quality of data include *timeliness, believability, value added*, *interpretability* and *accessability*, described as follows:

- **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.

■

2.2. 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
5-15	450
15-20	300
20-50	1500
50-80	700
80-110	44

Compute an *approximate median* value for the data.

Answer:

Using Equation (2.3), we have $L_1 = 20$, $N = 3194$, $(\sum freq)_l = 950$, $freq_{median} = 1500$, $width = 30$, $median = 32.94$ years.

■

2.3. 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. 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.

■

2.4. 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: $\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i = 809/27 = 30$ (Equation 2.1). 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. (Omitted here. Please refer to Figure 2.3 of the textbook.)

(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 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.

■

2.5. In many applications, new data sets are incrementally added to the existing large data sets. Thus an important consideration for computing descriptive data summary 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:

- Count: The current count can be stored as a value, and when x number of new values are added, we can easily update *count* with *count* + x. This is a distributive measure and is easily updated for incremental additions.
- Standard deviation: If we store the sum of the squared existing values and the count of the existing values, we can easily generate the new standard deviation using the formula provided in the book. We simply need to calculate the squared sum of the new numbers, add that to the existing squared sum, update the count of the numbers, and plug that into the calculation to obtain the new standard deviation. All of this is done without looking at the whole data set and is thus easy to compute.

- Median: To accurately calculate the median, we have to look at every value in the dataset. When we add a new value or values, we have to sort the new set and then find the median based on that new sorted set. This is much harder and thus makes the incremental addition of new values difficult.

■

2.6. 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 the attribute mean for quantitative (numeric) values or attribute mode for categorical (nominal) values:** For example, suppose that the average income of *AllElectronics* customers is \$28,000. Use this value to replace any missing values for *income*.

(e) **Using the attribute mean for quantitative (numeric) values or attribute mode for categorical (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.

(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 the data set, we can construct a decision tree to predict the missing values for *income*.

■

2.7. Using the data for *age* given in Exercise 2.4, answer the following.

(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 equal-frequency bins of size 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

(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, equal-width 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. Classification techniques can be used to implement concept hierarchies that can smooth the data by rolling-up lower level concepts to higher-level concepts.

■

2.8. 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. 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.

■

2.9. 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.

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

(e) Calculate the *correlation coefficient* (Person's product moment coefficient). Are these two variables positively or negatively correlated?

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 2.1.

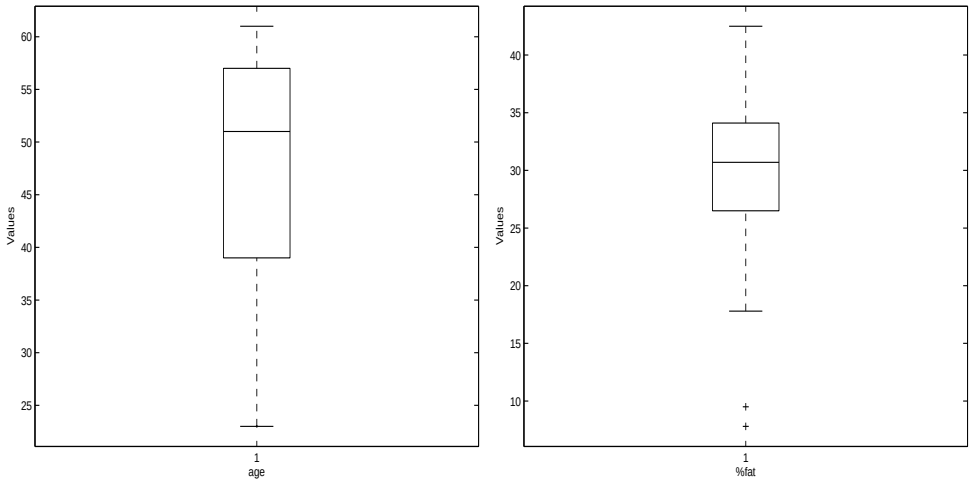

Figure 2.1: A boxplot of the variables *age* and *%fat* in Exercise 2.9.

(c) Draw a *scatter plot* and a *q-q plot* based on these two variables.
See Figure 2.2.

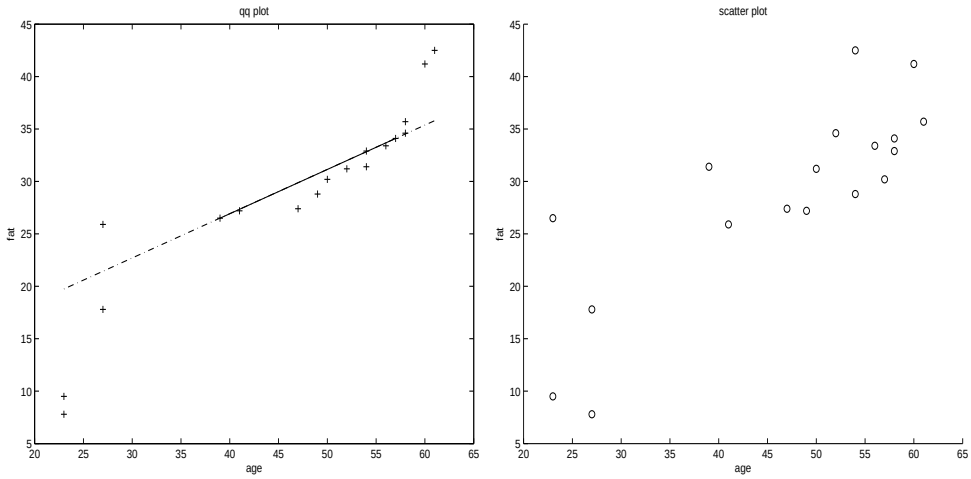

Figure 2.2: A *q-q plot* and a *scatter plot* of the variables *age* and *%fat* in Exercise 2.9.

(d) 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

(e) 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.

■

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

(a) min-max normalization

(b) z-score normalization

(c) normalization by decimal scaling

Answer:

(a) min-max normalization
The rage is [new_min, new_max]

(b) z-score normalization
The range is $[(old\_min - mean)/stddev, (old\_max - mean)/stddev]$. In general the range for all possible data sets is $(-\infty, +\infty)$.

(c) normalization by decimal scaling
The range is $(-1.0, 1.0)$.

■

2.11. Use the two 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

Answer:

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

original data	200	300	400	600	1000
[0,1] normalized	0	0.125	0.25	0.5	1

(b) z-score normalization

original data	200	300	400	600	1000
z-score	-1.06	-0.7	-0.35	0.35	1.78

■

2.12. Using the data for *age* given in Exercise 2.4, 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 $[0.0, 1.0]$.
For readability, let A be the attribute *age*. Using Equation (2.11) 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 Equation (2.12) 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 Equation (2.13) 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 because 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 of the current minimum and maximum values without encountering an "out of bounds error". As such values may be present in future data, this method is less appropriate. [NEW Both min-max and Z-score normalization change the original data quite a bit.] Z-score normalization transforms values into measures that represent their distance from the mean, in terms of standard deviations. This type of transformation may not be as intuitive to the user in comparison with decimal scaling.

■

2.13. Use a flow chart 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:

(a) Stepwise forward selection
See Figure 2.3.

(b) Stepwise backward elimination
See Figure 2.4.

(c) A combination of forward selection and backward elimination
See Figure 2.5.

■

2.14. 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 partitioning

(b) equal-width partitioning

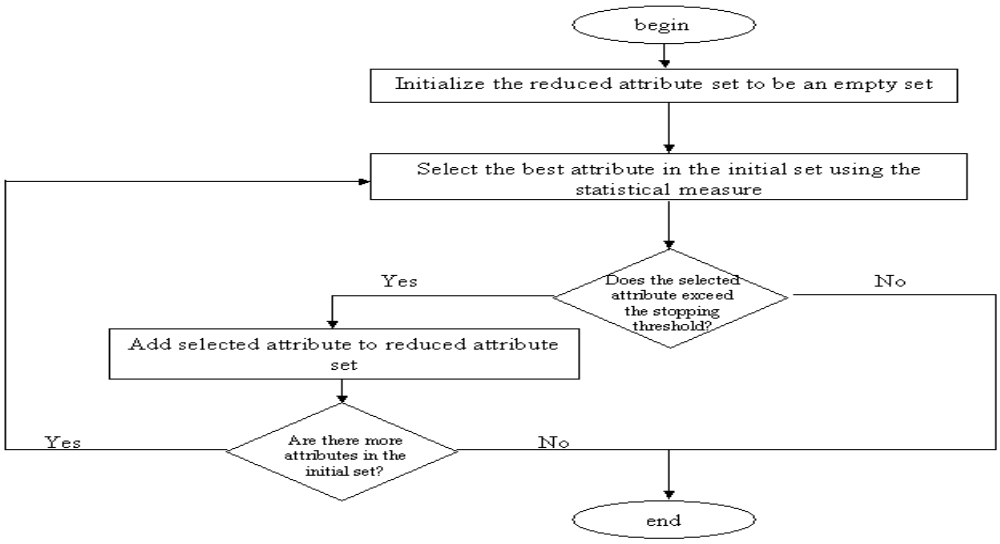

Figure 2.3: Stepwise forward selection.

(c) clustering

Answer:

(a) equal-frequency partitioning

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

(b) equal-width partitioning
The width of each interval is $(215 - 5)/3 = 70$.

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

(c) clustering
We will use a simple clustering technique: partition the data along the 2 biggest gaps in the data.

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

■

2.15. Using the data for *age* given in Exercise 2.4,

(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:

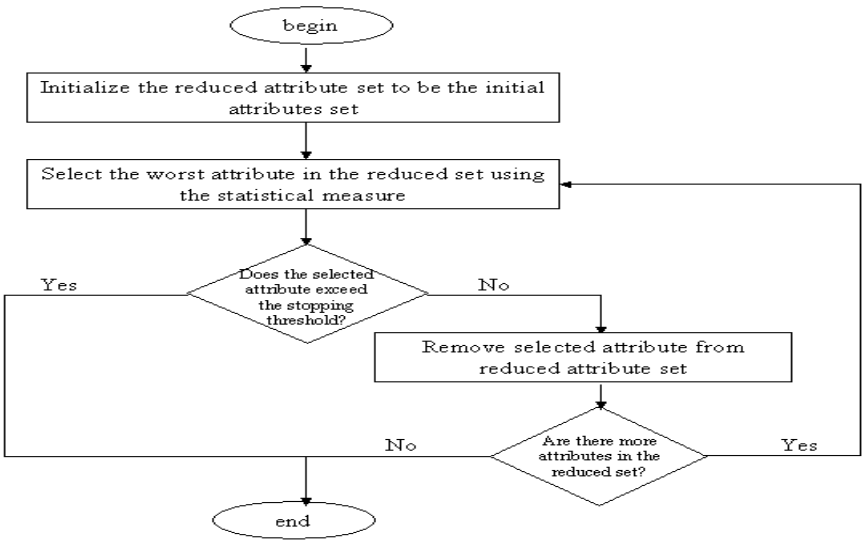

Figure 2.4: Stepwise backward elimination.

(a) Plot an equiwidth histogram of width 10.
See Figure 2.6.

(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 2.7.

■

2.16. [Contributed by Chen Chen] 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.

Given are some data sets sampled from different distributions, e.g., uniform, Gaussian, exponential, and gamma. (The former two distributions are symmetric, whereas the latter two are skewed). For example, if using Equation (2.3) to do approximation as proposed in the chapter, the most straightforward approach is to partition all of the data into k equal-length intervals.

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

where L_1 is the lower boundary of the median interval, N is the number of values in the entire data set, $(\sum freq)_l$ 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; however, the time used in the whole procedure will also increase. The product of error made and time used are good optimality measures. From this point, we can perform many tests for each type of distributions (so that the result won't be dominated

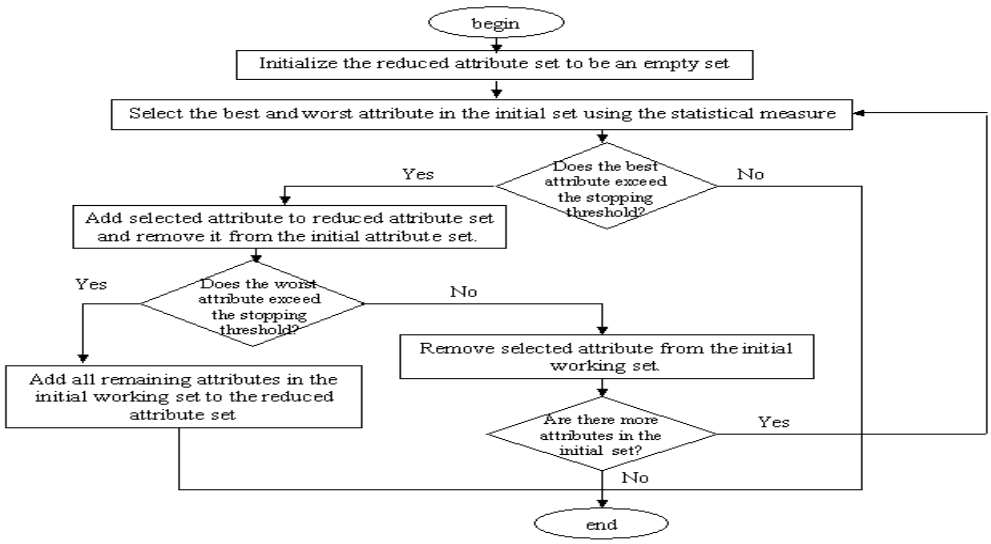

Figure 2.5: A combination of forward selection and backward elimination.

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 other approaches for median approximation. The student may suggest a few, analyze the best trade-off point, and compare the results from the different approaches. A possible such approach is as follows: Hierarchically divide the whole data set into intervals: first, divide it into k regions and find the region in which the median resides; second, divide this particular region into k subregions, find the subregion in which the median resides; .... This iterates until the width of the subregion reaches a predefined threshold, and then the median approximation formula as above stated is applied. In this way, we can confine the median to a smaller area without globally partitioning all of data into shorter intervals, which would be expensive. (The cost is proportional to the number of intervals.)

■

2.17. [Contributed by Deng Cai] It is important to define or select similarity measures in data analysis. However, there is no commonly accepted subjective similarity measure. Using different similarity measures may deduce different results. Nonetheless, some apparently 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 (1) Euclidean distance (Equation 7.5), and (2) cosine similarity (Equation 7.16).

(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:

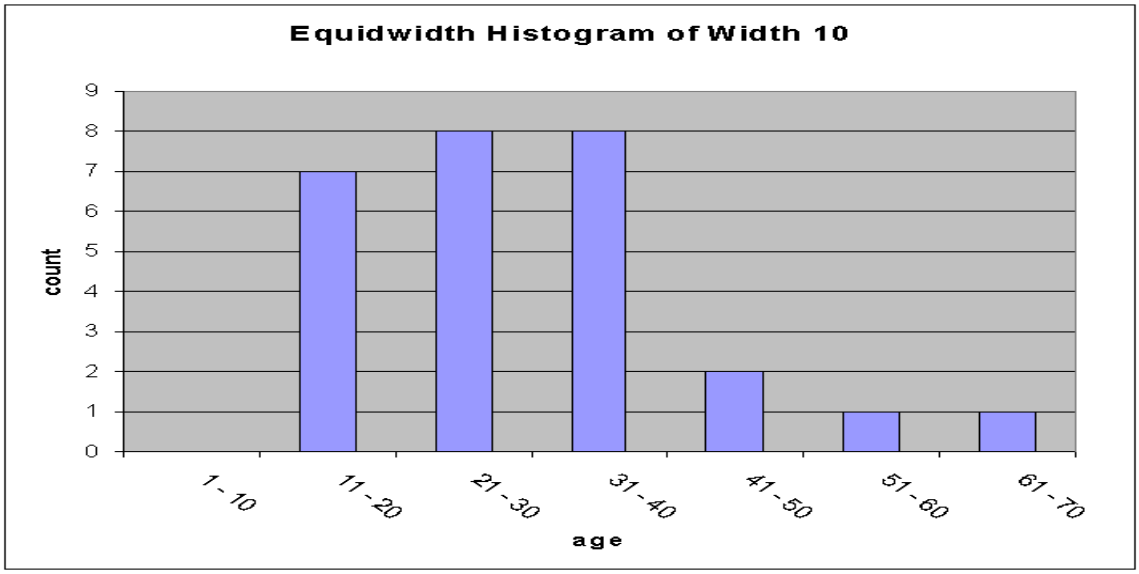

Figure 2.6: An equiwidth histogram of width 10 for *age*.

(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 (1) Euclidean distance (Equation 7.5), and (2) cosine similarity (Equation 7.16).

The Euclidean distance of two n-dimensional vectors, $\boldsymbol{x}$ and $\boldsymbol{y}$, is defined as: $\sqrt[2]{\sum_{i=1}^{n}(x_i - y_i)^2}$. The cosine similarity of $\boldsymbol{x}$ and $\boldsymbol{y}$ is defined as: $\frac{\boldsymbol{x}^t \cdot \boldsymbol{y}}{||\boldsymbol{x}||||\boldsymbol{y}||}$, where $\boldsymbol{x^t}$ is a transposition of vector $\boldsymbol{x}$, $||\boldsymbol{x}||$ is the Euclidean norm of vector $\boldsymbol{x}$,[1] and $||\boldsymbol{y}||$ is the Euclidean norm of vector $\boldsymbol{y}$. Using these definitions we obtain the distance from each point to the query point.

	$\boldsymbol{x_1}$	$\boldsymbol{x_2}$	$\boldsymbol{x_3}$	$\boldsymbol{x_4}$	$\boldsymbol{x_5}$
Euclidean distance	0.14	0.67	0.28	0.22	0.61
Cosine similarity	0.9999	0.9957	0.9999	0.9990	0.9653

Based on the Euclidean distance, the ranked order is $\boldsymbol{x_1, x_4, x_3, x_5, x_2}$. Based on the cosine similarity, the order is $\boldsymbol{x_1, x_3, x_4, x_2, x_5}$.

(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.

After normalizing the data we have:

$\boldsymbol{x}$	$\boldsymbol{x_1}$	$\boldsymbol{x_2}$	$\boldsymbol{x_3}$	$\boldsymbol{x_4}$	$\boldsymbol{x_5}$
0.6585	0.6616	0.7250	0.6644	0.6247	0.8321
0.7526	0.7498	0.6887	0.7474	0.7809	0.5547

The new Euclidean distance is:

	$\boldsymbol{x_1}$	$\boldsymbol{x_2}$	$\boldsymbol{x_3}$	$\boldsymbol{x_4}$	$\boldsymbol{x_5}$
Euclidean distance	0.0041	0.0922	0.0078	0.0441	0.2632

[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.

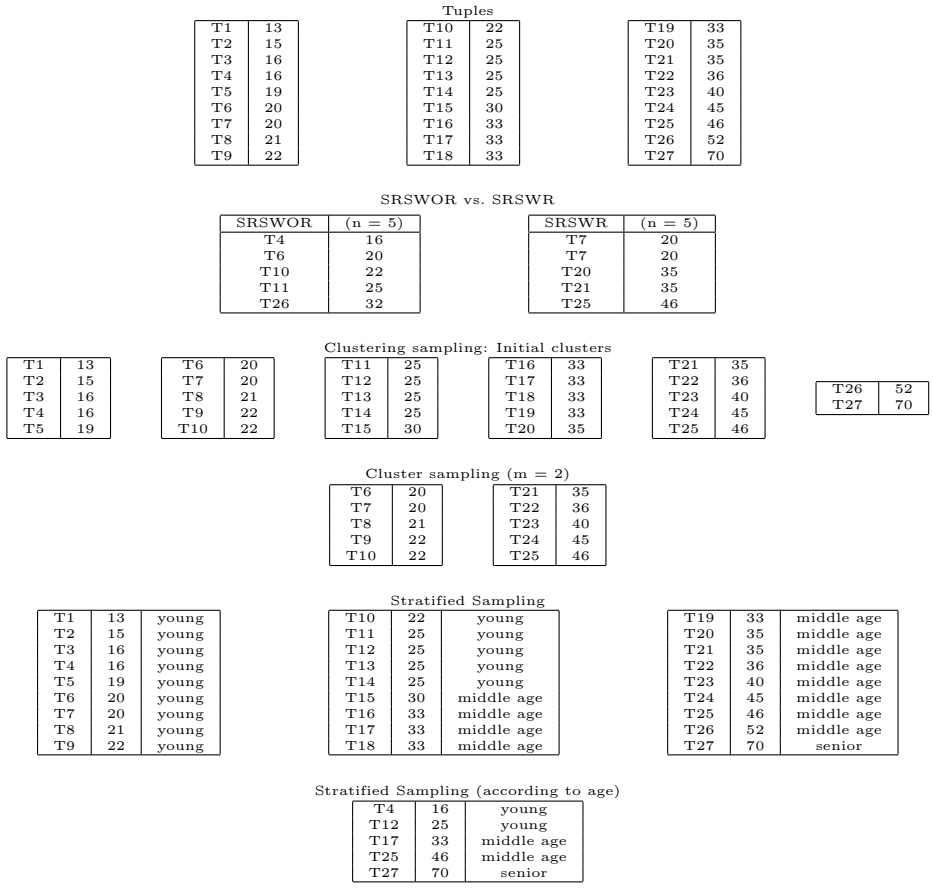

Figure 2.7: Examples of sampling: SRSWOR, SRSWR, cluster sampling, stratified sampling.

Based on the Euclidean distance of the normalized points, the order is $\boldsymbol{x_1, x_3, x_4, x_2, x_5}$, which is the same as the cosine similarity order.

■

2.18. 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.

(a) Briefly describe how ChiMerge works.

(b) Take the IRIS data set, obtained from *http://www.ics.uci.edu/∼mlearn/MLRepository.html* (UC-Irvine Machine Learning Data Repository), 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) Briefly describe how ChiMerge works.

The basic algorithm of chiMerge is:

begin

sort values in ascending order

```
        assign a separate interval to each distinct value
        while stopping criteria not met
        begin
            compute χ² of every pair of adjacent intervals
            merge the two intervals with smallest χ² value
        end
    end
```

(b) Take the IRIS data set, obtained from *http://www.ics.uci.edu/∼mlearn/MLRepository.html* (UC-Irvine Machine Learning Data Repository), 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.

The final intervals are:

Sepal length: [4.3 - 4.8],[4.9 - 4.9], [5.0 - 5.4], [5.5 - 5.7], [5.8 - 7.0], [7.1 - 7.9].

Sepal width: [2.0 - 2.2], [2.3 - 2.4], [2.5 - 2.8], [2.9 - 2.9], [3.0 - 3.3], [3.4 - 4.4].

Petal length: [1.0 - 1.9], [3.0 - 4.4], [4.5 - 4.7], [4.8 - 4.9], [5.0 - 5.1], [5.2 - 6.9].

Petal width: [0.1 - 0.6], [1.0 - 1.3], [1.4 - 1.6], [1.7 - 1.7], [1.8 - 1.8], [1.9 - 2.5].

The split points are:

Sepal length: 4.3, 4.9, 5.0, 5.5, 5.8, 7.1

Sepal width: 2.0, 2.3, 2.5, 2.9, 3.0, 3.4

Petal length: 1.0, 3.0, 4.5, 4.8, 5.0, 5.2

Petal width: 0.1, 1.0, 1.4, 1.7, 1.8, 1.9

■

2.19. 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

(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
```

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 *equal-width* 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 *equal-frequency* partitioning rule

Pseudocode for the automatic generation of a concept hierarchy for numeric data based on the equal-frequency 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. 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.

■

2.20. Robust data loading poses a challenge in database systems because the input data are often dirty. In many cases, an input record may have several missing values and some records could be *contaminated* (i.e., 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:

```
begin
    for each record r
    begin
        check r for missing values
            If possible, fill in missing values according to domain knowledge
            (e.g. mean, mode, most likely value, etc.).
        check r for out of range values
            If possible, correct out of range values according to domain knowledge
            (e.g. min or max value for the attribute).
        check r for erroneous data types
            If possible, correct data type using domain knowledge
        If r could not be corrected, mark it as bad and output it to a log,
        otherwise load r into the database.
    end
end
```

The domain knowledge can be a combination of manual and automatic work. We can, for example, use the data in the database to construct a decision tree to induce missing values for a given attribute, and at the same time have human-entered rules on how to correct wrong data types.

■

Chapter 3

Data Warehouse and OLAP Technology: An Overview

3.7 Exercises

3.1. 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. For the data collected in multiple heterogeneous databases to be used in decision-making processes, any semantic heterogeneity problems among multiple databases must be analyzed and solved so that the data can be integrated and summarized. 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 most of the queries needed could be done off-line.

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

■

3.2. 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) Enterprise warehouse, data mart, 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. Each radial line represents one dimension and each point (called a "footprint") along the line represents a level of the dimension. Each step away from the center represents the stepping down of a concept hierarchy of the dimension. 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) Enterprise warehouse, data mart, virtual warehouse

An enterprise warehouse collects all of the information about subjects spanning the entire organization, whereas a data mart contains a subset of corporate-wide data that is of value to a specific group of users. An enterprise warehouse provides corporate-wide data integration, usually from one or more operational systems or external information providers, and is cross-functional in scope, whereas the data mart is confined to specific selected subjects (such as customer, item, and sales for a marketing data mart). An enterprise warehouse typically contains detailed data as well as summarized data, whereas the data in a data mart tend to be summarized. An enterprise data warehouse may be implemented on traditional mainframes, computer superservers, or parallel architecture platforms, while data marts are usually implemented on low-cost departmental servers that are UNIX/LINUX- or Windows-based. The implementation cycle of an enterprise warehouse may take months or years, whereas that of a data mart is more likely to be measured in weeks. A virtual warehouse is a set of views over operational databases. For efficient query processing, only some of the possible summary views may be materialized. A virtual warehouse is easy to build but requires excess capacity on operational database servers.

■

3.3. 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 constellations 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 3.1.

(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 2004?

The operations to be performed are:

- Roll-up on *time* from *day* to *year*.

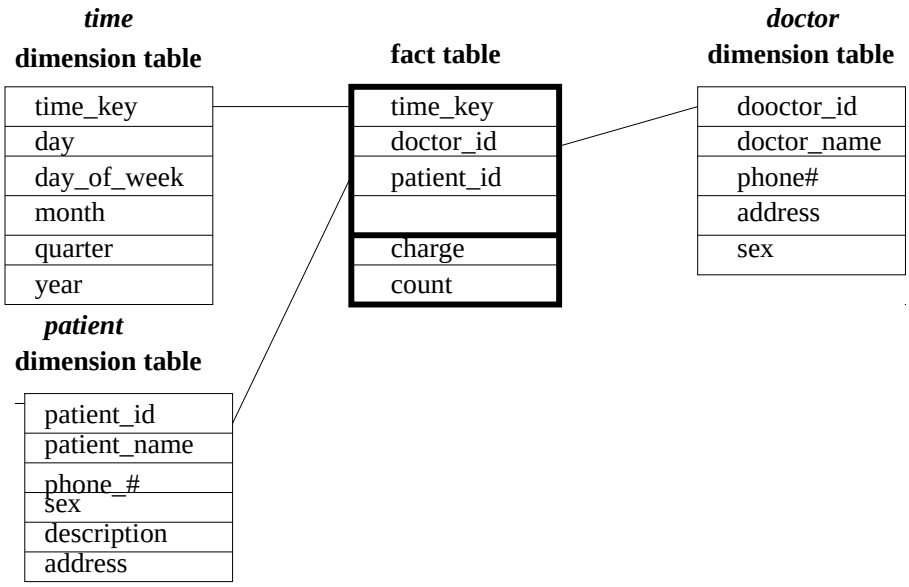

Figure 3.1: A star schema for data warehouse of Exercise 3.3.

- Slice for *time=2004*.
- 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=2004
group by doctor
```

■

3.4. 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 3.2.

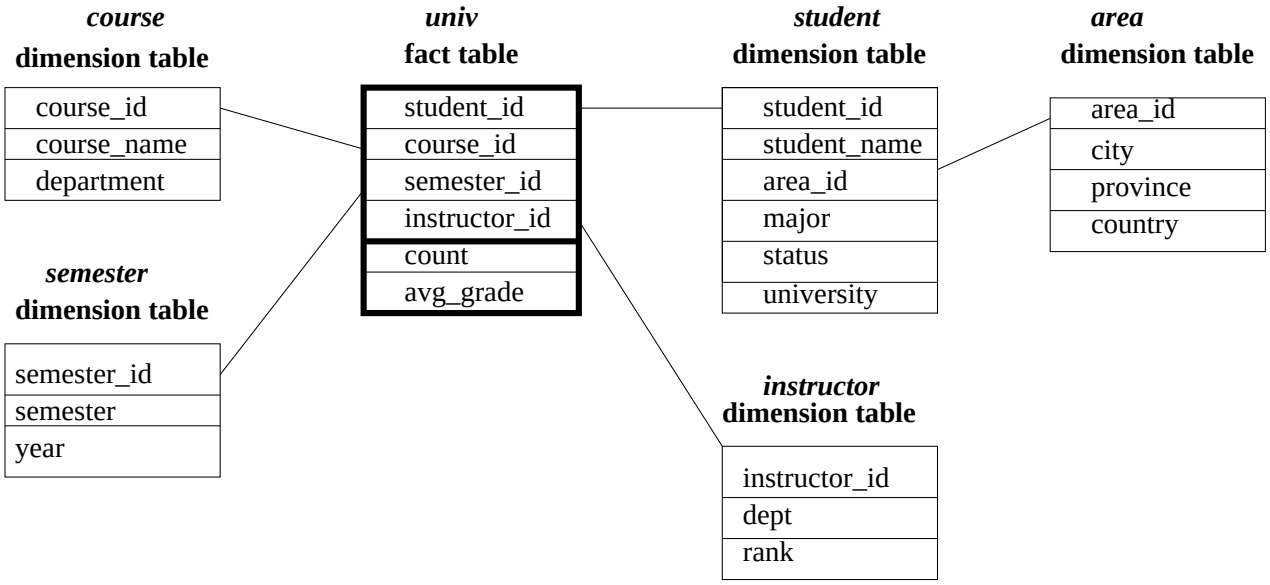

Figure 3.2: A snowflake schema for data warehouse of Exercise 3.4.

(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 student from *student_id* to *university*.
- Dice on course, student with *department="CS"* and *university = "Big University"*.
- Drill-down on *student* from *university* to *student_name*.

(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.

■

3.5. 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 2004?

(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 3.3.

(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 2004?

The specific OLAP operations to be performed are:

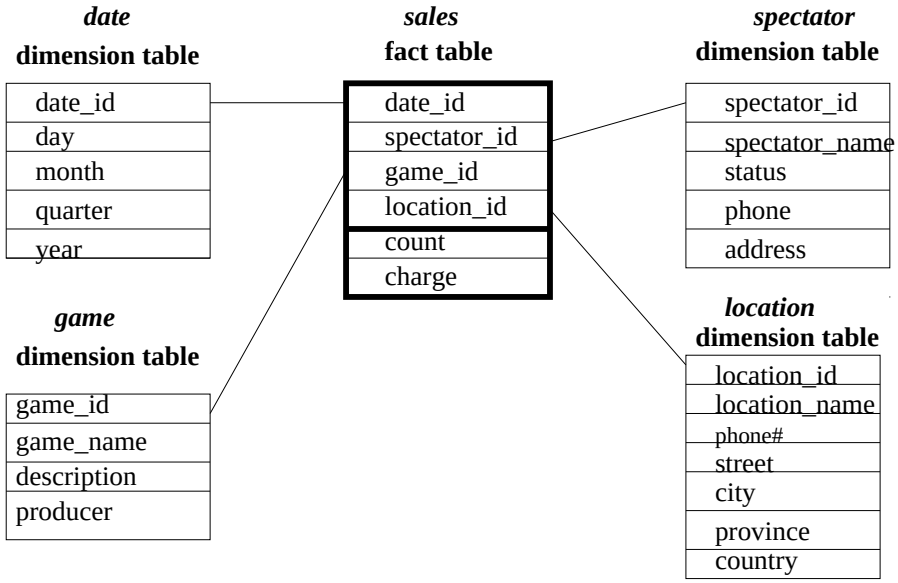

Figure 3.3: A star schema for data warehouse of Exercise 3.5.

- 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=2004*.

(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, then 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 used to represent the bitmap index could be very long. For example, a 10-year collection of data could result in 3650 date records, meaning that every tuple in the fact table would require 3650 bits (or approximately 456 bytes) to hold the bitmap index.

■

3.6. [*Contributed by Tao Cheng*] 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 of the star schema is its simplicity, which will enable efficiency, but it requires more space. For the 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, the star schema is better simply because efficiency typically has higher priority over space as long as the space requirement is not too huge. In industry, sometimes the data from a snowflake schema may be denormalized

into a star schema to speed up processing [1]. Another option is to use a snowflake schema to maintain dimensions, and then present users with the same data collapsed into a star [2].

References for the answer to this question include:

[1] Oracle Tip: Understand the difference between star and snowflake schemas in OLAP. *http://builder.com.com/-5100-6388-5221690.html*.

[2] Star vs. Snowflake Schemas. *http://blogs.msdn.com/bi_systems/articles/164525.aspx*.

■

3.7. 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 on-line analytical processing, and derive general weather patterns in multidimensional space.

Answer:

Since the weather bureau has about 1,000 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 3.4.

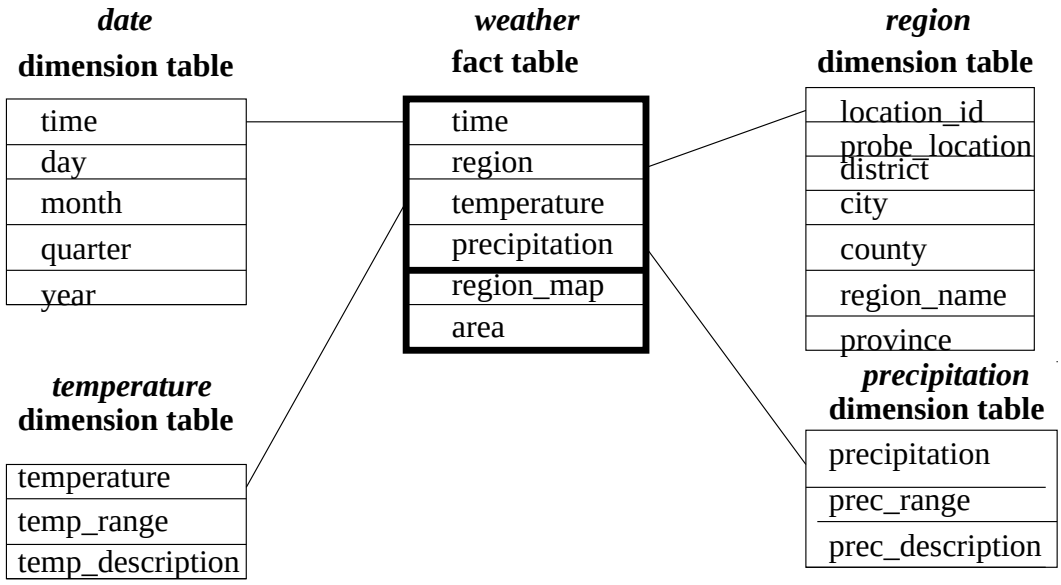

Figure 3.4: A star schema for a *weather* spatial data warehouse of Exercise 3.7.

To construct this spatial data warehouse, we may need to integrate spatial data from heterogeneous sources and systems. Fast and flexible on-line analytical processing in spatial data warehouses 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 on-line computation.

■

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

Answer: Present an example illustrating such a huge and sparse data cube.

An example of a huge and sparse data cube is 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.

■

3.9. 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 N 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 N x_i's.
The *variance* function 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 of 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 obtain 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.

■

3.10. 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. Compute $average = sum/count$.

For median, keep a small number, p, of centered values (e.g., $p = 10$) and 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 easily be calculated from the above set.

■

3.11. 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. 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.

■

3.12. 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 efficiently support this preference?

(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 efficiently support this preference?
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.

■

3.13. 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.
This is the maximum number of distinct tuples that you can form with p distinct values per dimensions.

(b) What is the *minimum number of cells* possible in the base cuboid?
p.
You need at least p tuples to contain p distinct values per dimension. In this case no tuple shares any value on any dimension.

(c) What is the *maximum number of cells* possible (including both base cells and aggregate cells) in the data cube, C?
$(p+1)^n$.
The argument is similar to that of part (a), but now we have $p+1$ because in addition to the p distinct values of each dimension we can also choose $*$.

(d) What is the *minimum number of cells* possible in the data cube, C?
$(2^n - 1) \times p + 1$.
The minimum number of cells is when each cuboid contains only p cells, except for the apex, which contains a single cell.

■

3.14. 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 involves using queries to find and report useful information 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 on-line 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.

On-line 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.

■

Chapter 4

Data Cube Computation and Data Generalization

4.5 Exercises

4.1. 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 5×2^7 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) $(*, *, *, d_4, \ldots, d_9, d_{10}) : 3$.

■

4.2. 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.

BUC: top-down (see above note), recursive partition and conquer, shared sorting

Star-Cubing: top-down and bottom-up integration using star-tree, enables simultaneous computation while allows Apriori pruning.

(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.

■

4.3. [*Contributed by Chen Chen*] Suppose a data cube 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 c tuples, then all aggregate cells will also satisfy the condition.

1. Minimum case: The **distinct** condition no longer holds here, since c 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 $\lfloor \frac{k}{c} \rfloor$ 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, c, 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.

■

4.4. 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 4.1.

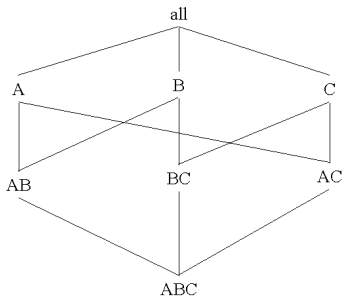

Figure 4.1: A complete lattice for the cube of Exercise 4.4.

(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
- 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 $\times$ 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 4.2.

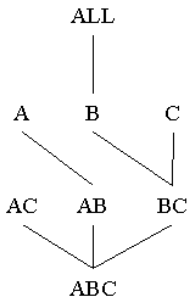

Figure 4.2: The order of computation in Exercise 4.4 that requires the least amount of space.

The total amount of main memory space required for computing the 2-D planes is: Total space = (100 $\times$ 1,000) + (1,000,000 $\times$ 10) + (100 $\times$ 10,000) = 20,100,000 cells = 80,400,000 bytes.

■

4.5. Often, the aggregate measure 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.

■

4.6. 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 by $*$'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: $\{(a_1, a_2, a_3, \ldots, a_{100}) : 10, (a_1, a_2, b_3, \ldots, b_{100}) : 10, (a_1, a_2, *, \ldots, *) : 20\}$.

■

4.7. 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 [LPH02].

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.

■

4.8. 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 4.3.

■

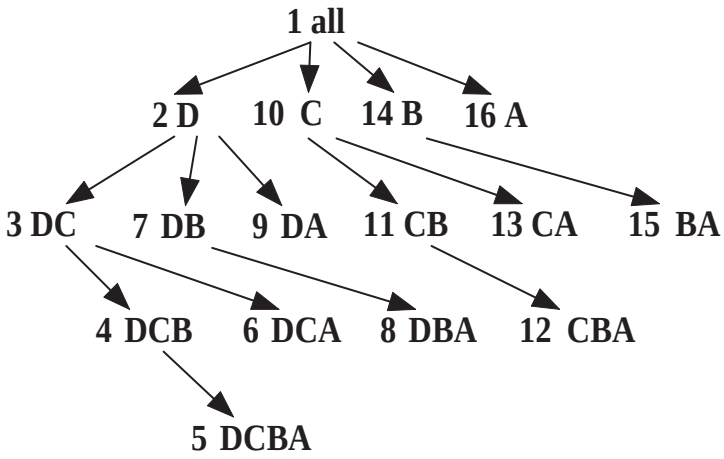

Figure 4.3: BUC processing order for Exercise 4.7.

4.9. Discuss how you might extend the *Star-Cubing* algorithm to compute iceberg cubes where the iceberg condition tests for 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. This is analogous to the optimization presented in Section 4.1.6.

■

4.10. A flight data warehouse for a travel agent consists of six dimensions: *traveller, 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* [*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 traveller* who flies American Airlines (*AA*) from L.A. in the year 2004?

(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 traveller* who flies American Airlines (*AA*) from L.A. in the year 2004?

The OLAP operations are:

i. roll-up on *traveller* 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 "ANY" (∗)
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 *traveller* 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.

■

4.11. (**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 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.

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 $(a_{1,1}, a_{2,1}, ..., a_{99,1}, *)$. Let the support threshold be 10. Then, all 299 descendent cells of this cell satisfy the threshold. In this case, iceberg cubing cannot benefit from the pruning effect.

(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.

■

4.12. Consider the following *multifeature cube* query: Grouping by all subsets of {*item, region, month*}, find the minimum shelf life in 2004 for each group, and the fraction of the total sales due to tuples whose price is less than \$100, and whose shelf life is between 1.25 and 1.5 of the minimum shelf life.

(a) Draw the multifeature cube graph for the query.

(b) Express the query in extended SQL.

(c) Is this a *distributive* multifeature cube? Why or why not?

Answer:

(a) Draw the multifeature cube graph for the query.

$$R0 \rightarrow R1(>= 1.25 * min(shelf) \ and \ <= 1.5 * min(shelf))$$

(b) Express the query in extended SQL.

select	item, region, month, MIN(shelf), SUM(R1)
from	Purchases
where	year = 2004
cube by	item, region, month: R1
such that	R1.Shelf >= 1.25*MIN(Shelf) and (R1.Shelf <= 1.5*MIN(Shelf) and R1.Price < 100

(c) Is this a *distributive* multifeature cube? Why or why not?
No, this is not a distributive multifeature cube because of the $\leq$ conditions in the such that clause.

■

4.13. For *class characterization*, what are the major differences between a data cube-based implementation and a relational implementation such as attribute-oriented induction? Discuss which method is most efficient and under what conditions this is so.

Answer:

For class characterization, the major differences between a data cube-based implementation and a relational based implementation such as attribute-oriented induction include the following:

- **Process control:** Under a data cube-based approach, the process is user-controlled at every step. This includes the selection of the relevant dimensions to be used as well as the application of OLAP operations such as roll-up, roll-down, slicing and dicing. A relational approach does not require user interaction at every step, however, as attribute relevance and ranking is performed automatically.
- **Supported data types and measures:** The relational approach supports complex data types and measures, which restrictions in current OLAP technology do not allow. Thus, OLAP implementations are limited to a more simplified model for data analysis.
- **Precomputation:** An OLAP-based implementation allows for the precomputation of measures at different levels of aggregation (materialization of subdata cubes), which is not supported under a relational approach.

Based upon these differences, it is clear that a relational approach is more efficient when there are complex data types and measures being used, as well as when there are a very large number of attributes to be considered. This is due to the advantage that automation provides over the efforts that would be required by a user to perform the same tasks. However, when the data set being mined consists of regular data types and measures that are well supported by OLAP technology, then the OLAP-based implementation provides an advantage in efficiency. This results from the time saved by using precomputed measures, as well as the flexibility in investigating mining results provided by OLAP functions. ■

4.14. Suppose that the following table is derived by *attribute-oriented induction*.

class	*birth_place*	count
Programmer	USA	180
	others	120
DBA	USA	20
	others	80

(a) Transform the table into a crosstab showing the associated t-weights and d-weights.

(b) Map the class *Programmer* into a (bidirectional) *quantitative descriptive rule*, for example,

$$\forall \boldsymbol{X},\ Programmer(\boldsymbol{X}) \Leftrightarrow (birth\_place(\boldsymbol{X}) = \text{“}USA\text{”} \wedge \ldots)[t : x\%, d : y\%] \ldots \Theta(\ldots)[t : w\%, d : z\%].$$

Answer:

(a) Transform the table into a crosstab showing the associated t-weights and d-weights.
See Table 4.1.

location \ item	Canada			Other			*both_locations*		
	count	t-weight	d-weight	count	t-weight	d-weight	count	t-weight	d-weight
Programmer	180	60%	90%	120	40%	60%	300	100%	75%
DBA	20	20%	10%	80	80%	40%	100	100%	25%
both_classes	200	50%	100%	200	50%	100%	400	100%	100%

Table 4.1: A crosstab for *birth_place* of Programmers and DBAs.

(b) Map the class *Programmer* into a (bidirectional) *quantitative descriptive rule*, for example,

$$\forall \boldsymbol{X},\ Programmer(\boldsymbol{X}) \Leftrightarrow (birth\_place(\boldsymbol{X}) = \text{“}USA\text{”} \wedge \ldots)[t : x\%, d : y\%] \ldots \Theta(\ldots)[t : w\%, d : z\%].$$

$$\begin{aligned} &\forall \boldsymbol{X},\ class(\boldsymbol{X}) = \text{“}Programmer\text{”} \ \Leftrightarrow \\ &\quad (birth\_place(\boldsymbol{X}) = \text{“}Canada\text{”})\ [t : 60\%, d : 90\%]\ \Theta\ (birth\_place(\boldsymbol{X}) = \text{“}Other\text{”}) \\ &\quad [t : 40\%, d : 60\%]. \end{aligned}$$

■

4.15. Discuss why *relevance analysis* is beneficial and how it can be performed and integrated into the characterization process. Compare the result of two induction methods: (1) with relevance analysis and (2) without relevance analysis.

Answer:

In characterization, specifying the set of relevant attributes may be difficult for the user if he/she is not familiar with the underlying distribution of values for an attribute or the possible inter-attribute dependencies. For example, he/she may specify attributes for which there is no good generalization operator or that are irrelevant to the characterization process. The benefit of doing characterization with relevance analysis

Algorithm: Incremental class comparison. Data-cube-based incremental algorithm for mining class comparisons with dimension relevance analysis.

Input:

- *DB*, a relational database (or other data source such as a flat file);
- a data mining query;
- a list of attributes;
- a set of concept hierarchies or generalization operators on the attributes;
- attribute generalization thresholds.

Output: P, a *Prime_generalized_relation* used to build the data cube.

Method: The method is outlined as follows. To build the initial data cube for mining:

(a) Data Collection: The incremental part of the data is identified to produce a target class and contrasting class(es) from the set of task relevant data to generate the initial working relations.

(b) Incremental dimension relevance analysis: This is performed on the initial working relation for the target class in order to determine which attributes should be retained (attribute relevance). An attribute will have to be added that indicates the class of the data entry.

(c) Incremental synchronous generalization: The desired level of generalization is determined to form prime target class and prime contrasting class cuboid(s). This generalization will be synchronous between all of the classes, as the contrasting class relation(s) will be generalized to the same level.

To process revisions to the relevant data set and thus make the algorithm incremental, perform the following. (Note: we do not want to completely rebuild the data cube for all relevant data as this would be time consuming and inefficient. Instead, only the changes to the relevant data will be processed and added to the prime relation as held in the data cube.)

(a) Generalize the changed data to the same level of abstraction, with the same attributes as held in the data cube, and add the attribute required to indicate the class of each tuple.

(b) Calculate count and other required aggregate values as done for the initial relations, and then merge the statistical information from this changed data with those results as currently held in the data cube.

Figure 4.4: A data-cube-based algorithm for incremental class comparison.

versus without is that the former can uncover patterns that may be hidden by irrelevant attributes present in the latter.

■

4.16. Given a generalized relation, R, derived from a database, DB, suppose that a set, ΔDB, of tuples needs to be deleted from DB. Outline an *incremental* updating procedure for applying the necessary deletions to R.

Answer:

An incremental updating procedure for applying the necessary deletions to R to reflect the changes in ΔDB is as follows:

- Generalize the tuples to be deleted (ΔDB) to the same level of abstraction as those in the generalized relation, R.
- Include in the tuples to be deleted, the attribute required to indicate the class of each tuple as it relates to the generalized relation, R.
- Calculate count and other required aggregate values for the tuples in ΔDB, as reflected by the values held in R.
- Update the affected tuples in R to reflect the difference in count between their current value and the value in the relation to be deleted (ΔDB) and recalculate aggregate values using the difference in measures between R and ΔDB.

■

4.17. Outline a data cube-based *incremental* algorithm for mining class comparisons.

Answer:

A data-cube-based algorithm for incremental class comparison is given in Figure 4.4.

■

Chapter 5

Mining Frequent Patterns, Associations, and Correlations

5.7 Exercises

5.1. The Apriori algorithm uses *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. 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, $d_1, d_2, d_3, \ldots, d_n$. Thus, $D = d_1 d_2 d_3 \ldots d_n$.

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. Thus, $A = a_1 + a_2 + a_3 + \ldots + a_n$.

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.

■

5.2. Section 5.2.2 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 in Section 5.2.2. (*Hint:* Consider incorporating the properties of Exercise 5.1(b) and 5.1(c) into your design.)

Answer:

An algorithm for generating strong rules from frequent itemsets is given in Figure 5.1. It is more efficient than the method proposed in Section 5.2.2 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 because their respective confidences will never be greater than the confidence of x (see Exercise 5.1(b) and 5.1(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 5.2.2, on the other hand, is a brute-force method that generates all of 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 5.2.2 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.

■

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

Algorithm: Rule_Generator. Given a set of frequent itemsets, output all of its strong rules.

Input:

- l, set of frequent itemsets;
- *min_conf*, the minimum confidence threshold.

Output: Strong rules of itemsets in l.

Method: The method is outlined as follows.

```
1)  for each frequent itemset, l
2)      rule_generator_helper(l, l, min_conf);

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

Figure 5.1: An algorithm for generating strong rules from frequent itemsets.

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.

Apriori:

C1 =

m	3
o	3
n	2
k	5
e	4
y	3
d	1
a	1
u	1
c	2
i	1

L1 =

m	3
o	3
k	5
e	4
y	3

C2 =

mo	1
mk	3
me	2
my	2
ok	3
oe	3
oy	2
ke	4
ky	3
ey	2

L2 =

mk	3
ok	3
oe	3
ke	4
ky	3

C3=

oke	3
key	2

L3=

oke	3

FP-growth: See Figure 5.2 for the FP-tree.

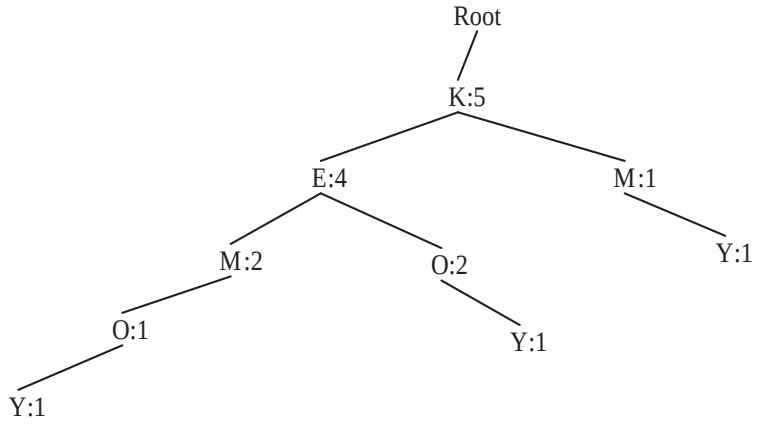

Figure 5.2: FP-tree for Exercise 5.3.

item	conditional pattern base	conditional tree	frequent pattern
y	{ {k,e,m,o:1}, {k,e,o:1}, {k,m:1} }	k:3	{k,y:3}
o	{ {k,e,m:1}, {k,e:2}	k:3,e:3	{k,o:3}, {e,o:3}, {k,e,o:3}
m	{ {k,e:2}, {k:1} }	k:3	{k,m: 3}
e	{ {k:4} }	k:4	{ k,e:4 }

Efficiency comparison: Apriori has to do multiple scans of the database while FP-growth builds the FP-Tree with a single scan. Candidate generation in Apriori is expensive (owing to the self-join), while FP-growth does not generate any candidates.

(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.):

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

k,o $\rightarrow$ e [0.6,1]
e,o $\rightarrow$ k [0.6,1]

■

5.4. (**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:

This is to be evaluated on an individual basis as there is no standard answer.

■

5.5. 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 {(*Wonder-Bread, Dairyland-Milk, Tasty-Pie*), (*Wonder-Bread, Sunset-Milk, Dairyland-Cheese*)}.

■

5.6. Suppose that a large store has a transaction 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 (without considering multilevel associations). 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.

■

5.7. Suppose that frequent itemsets are saved for a large transaction 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.

■

5.8. [*Contributed by Tao Cheng*] 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 transaction 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:

Consider an item and its occurrence *count* as a combined item in a transaction. For example, we can consider (*jug*, 3) as one item. In the first scan for finding single frequent item i, register i's highest *count* if it reaches the frequency threshold. For instance, (*jug*, 3) may be a frequent item. For (i, max_count), try to find k-itemsets for *count* from 1 to max_count. This can be done either by Apriori or FP-growth. In FP-growth, one can create a node for each $(i, count)$ combination, however, for efficient implementation, such nodes can be combined into one using combined counters (i.e., by storing more information).

■

5.9. (**Implementation project**) Implement three *closed frequent itemset mining* methods (1) A-Close [PBTL99] (based on an extension of Apriori [AS94]), (2) CLOSET+ [WHP03] (based on an extension of FP-growth [HPY00]), and (3) CHARM [ZH02] (based on an extension of ECLAT [Zak00]). Compare their performance with various kinds of large data sets. Write a report to answer the following questions:

(a) Why is mining the set of closed frequent itemsets often more desirable than mining the complete set of frequent itemsets (based on your experiments on the same data set as Exercise 5.4)?

(b) Analyze in which situations (such as data size, data distribution, minimal support threshold setting, and pattern density) and why that one algorithm performs better than the others.

Answer:

(a) Why is mining the set of closed frequent itemsets often more desirable than mining the complete set of frequent itemsets (based on your experiments on the same data set as Exercise 5.4)?

Mining closed frequent itemsets leads to a much more compact answer set but has the same expressive power as mining the complete set of frequent itemsets. Moreover, it leads to more efficient mining algorithms if one explores optimization methods as discussed in the above research papers.

(b) Analyze in which situations (such as data size, data distribution, minimal support threshold setting, and pattern density) and why that one algorithm performs better than the others.

Please check IlliMine (*http://illimine.cs.uiuc.edu*) for some implementation code. The discussion of these algorithms are in the three papers listed above. Since different implementations may lead to different performance results, we will not provide a "standard answer" here.

■

5.10. Suppose that a data relation describing students at *Big University* has been generalized to the generalized relation R in Table 5.1.

Table 5.1: 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.*

Students can easily sketch the corresponding 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%]

■

5.11. Propose and outline a **level-shared mining** approach to mining multilevel association rules in which each item is encoded by its level position, and 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 5.3 for illustration.

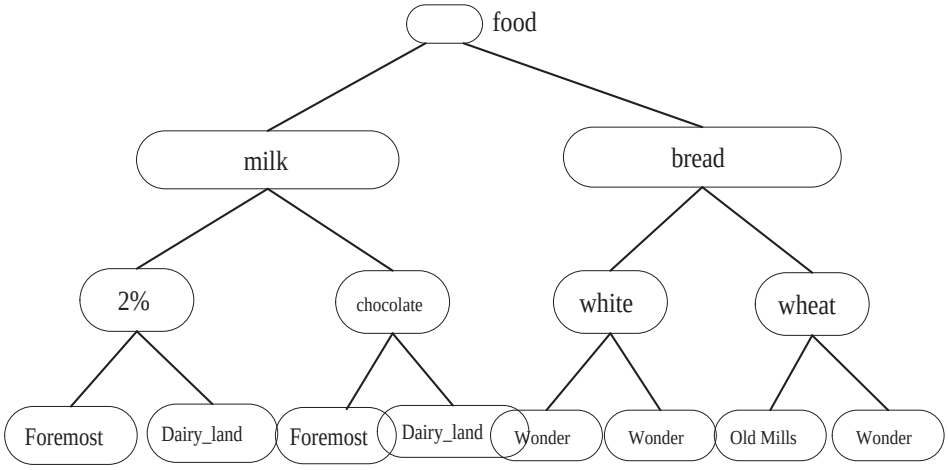

Figure 5.3: A taxonomy of data items.

- **Step 1:** Scan the original transactional 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'.

  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 versus 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.

■

5.12. (**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, and compare the performance of your new implementation with the one that does not incorporate such optimization.

(a) The previously proposed frequent pattern mining with FP-tree generates conditional pattern bases using a bottom-up projection technique (i.e., project on the prefix path of an item p). However, one can develop a **top-down projection** technique (i.e., project on 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 and compare your performance with the bottom-up projection method.

(b) Nodes and pointers are used uniformly in FP-trees 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 an **array- and pointer-based hybrid implementation**, where a node may store multiple items when it contains no splitting point to multiple subbranches. 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. One 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:

There is no standard answer for an implementation project. However, several papers discuss such implementations in depth and can serve as good references.

(a) For a **top-down projection** technique, one can refer to the CLOSET+ paper: Wang, Han, and Pei, *CLOSET+: Searching for the best strategies for mining frequent closed itemsets*, Proc. 2003 ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining (KDD'03), Washington, DC, pp. 236-245, Aug. 2003 [WHP03].

(b) For an **array- and pointer-based hybrid implementation**, one can refer to the paper by Grahne and Zhu entitled *Efficiently using prefix-trees in mining frequent itemsets*, Proc. ICDM'03 Int. Workshop on Frequent Itemset Mining Implementations (FIMI'03), Melbourne, FL, Nov. 2003 [GZ03].

(c) The **push right** technique can be found in the paper by Liu, Lu, Lou, and Yu entitled *On computing, storing and querying frequent patterns*, Proc. 2003 ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining (KDD'03), Washington, DC, pp. 607-612, Aug. 2003 [LLLY03].

■

5.13. 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.

■

5.14. 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	2,000	500	2,500
$\overline{hamburgers}$	1,000	1,500	2,500
Σ_{col}	3,000	2,000	5,000

(a) Suppose that the association rule "*hot dogs* ⇒ *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?

Answer:

(a) Suppose that the association rule "*hotdogs* ⇒ *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.

■

5.15. 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 one-dimensional mutation (i.e, *siblings*) operations. Such an analysis is called **cube gradient analysis**. Suppose the cube measure 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:

This answer is based on the "All-Significant-Pairs" algorithm presented by Dong, Han, Lam, et al. in the paper *Mining multi-dimensional constrained gradients in data cubes*, Proc. 2001 Int. Conf. on Very Large Data Bases (VLDB'01), Rome, Italy pp. 321-330, Sept. 2001 [DHL+01].

For a more efficient algorithm, see the "LiveSet-Driven" algorithm in the same paper.

Assume that we are given a set of probe cells P, a significant constraint C_{sig} (e.g. *count* > 100), and a gradient constraint C_{grad} (e.g. *count*(probe cell) / *count*(gradient cell) > 1.2).

(a) Compute an iceberg cube with iceberg condition C_{sig}.

(b) For each probe cell c_p retrieve the gradient cells c_g that are ancestors of c_p, and if C_{grad} is satisfied output (c_p, c_g). If the gradient constraint is antimonotonic, use this property to prune the search (i.e., if the measure m of a cell c is no greater than t, none of c's descendants can have the measure greater than t).

(c) In a similar manner, check for the descendants of c_p. Use the antimonotonic property to prune.

(d) Check the siblings of c_p and output the pairs that satisfy the gradient constraint.

■

5.16. Association rule mining often generates a large number of rules. Discuss effective methods that can be used to reduce the number of rules generated while still preserving most of the interesting rules.

Answer:

There are several approaches to reduce the number of rules. Here we list a few:

- Mine only closed frequent patterns to reduce the number of redundant rules.
- Use multilevel rule mining and generate lower-level rules only when they are nonredundant given the high-level rules. For example, we may find rules at the product category level first. If we find that *milk* $\rightarrow$ *cheese* [*support*=0.1, *conf*=0.9] and at the lower level we get *milk* 2% $\rightarrow$ *provolone* [*support*=0.01, *conf*=0.92], this may be redundant, i.e., this would be the expected support and confidence given the high-level rule.
- Use domain knowledge to define templates for the rules to be mined and define minimum support, confidence, and correlation measures.

■

5.17. Sequential patterns can be mined in methods similar to the mining of association rules. Design an efficient algorithm to mine **multilevel sequential patterns** from a transaction database. An example of such a pattern is the following: "*A customer who buys a PC will buy Microsoft software within three months*", on which one may drill down to find a more refined version of the pattern, such as "*A customer who buys a Pentium PC will buy Microsoft Office within three months*".

Answer:

The following is taken from *Mining Sequential Patterns* by Agrawal and Srikant from *Proc. 1995 Int. Conf. Data Engineering (ICDE'95)*, Taipei, Taiwan, pp. 3-14, March 1995 [AS95].

Mining single-level sequential patterns:

- **Step 1:** Sort the transaction database D into terms of *customer_id* as the primary key and *transaction_time* as the secondary key, which will implicitly convert D into a database of customer sequences (each customer's ordered list of transactions).
- **Step 2:** Using Apriori, find the set of all frequent itemsets. Let us call this set $L = \{L_1, L_2, \ldots, L_n\}$. From this set, find the set of all frequent 1-sequences where each sequence of the set is defined as $\{\langle l \rangle | l L\}$. Support counts for finding frequent sequential patterns is slightly different than for finding frequent itemsets. The latter defines support as the occurrence frequency of that itemset in the database, while the former defines it as the fraction of customers that purchased the itemset. Therefore, even though a customer may buy an itemset multiple times, only one is added to the support count.
- **Step 3:** The database D is then transformed into a database DT, which consists of transformed customer sequences, one for each customer. To do so, each transaction in D is replaced by the set of all frequent itemsets contained in that transaction. Using the frequent itemsets found in Step 2, replace each transaction in D accordingly. If a transaction does not contain any frequent itemsets, then it will not appear in the transformed customer sequence. If an implicit customer sequence (that customer's ordered list of transactions) does not contain any frequent itemsets, then that customer will not appear in the transformed database.

- **Step 4:** Perform the Apriori algorithm on DT with the following minor modifications: Replace any instances of the word "itemset" with "sequence". In particular, the first step of Apriori is replaced with "*L1 = find_frequent_1_sequences(D)*," which is basically what was stated in Step 2.
- **Step 5:** From the set of all frequent sequences, S, found in Step 4, determine the maximal sequences. A maximal sequence is one that is not contained in any other sequence. Let n denote the length of the longest sequence. Here is the algorithm that filters out the maximal sequences.

```
for (k = n; k > 1; k − −) do
    for each k-sequence sk do
        delete from S all subsequences of sk
```

- **Step 6:** To go from mining single-level sequential patterns to mining multilevel sequential patterns is trivial. We can employ the same techniques used to mine multilevel association rules. The general idea is to find frequent sequences at the top level. For those that are not frequent, we avoid checking their corresponding descendant sequences at the next level since it would be impossible for any of them to be frequent. For the sequences that are frequent, we find their corresponding frequent descendant sequences at the next level. Proceed in this manner until we reach the bottom level or until we cannot find any new frequent sequences, whichever comes first. This method assumes uniform minimum support for all levels. If we are using reduced minimum supports at lower levels, then we can modify this method to allow subfrequent (relatively frequent) sequences to be "passed down" to lower levels.

■

5.18. 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

Answer:

(a) $v \in S$: not antimonotonic, but monotonic, succinct.

Answer:

- *Antimonotone 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 antimonotonic 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'. If we assume that $v \in S'$, 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 monotonic constraint.
- *Succinct Proof*:
  To show that it is succinct we need to show that there is a precise formula to generate all of the sets that satisfy the constraint. In this case, the formula is $S = S_1 \cup S_2$ where $S_1 = \{v\}$.

(b) $S \subseteq V$: antimonotonic, not monotonic, succinct.

Answer:

- *Antimonotone Proof*: Let $S \subseteq V$. There must exist an element $e \in S$ such that $e \in V$. Let S' be a superset of S. Then that same element e is contained in every superset S' of S. Thus, $S' \subseteq V$. Therefore, $S \subseteq V$ is an antimonotonic constraint.
- *Monotone Counter-example*: Let $S = \{1, 2, 3\}$ and $V = \{1, 2, 3, 4, 5\}$ so that $S \subseteq V$ is true. However, if we add more elements to S, such as adding 8, then $S \subseteq V$ is false. Thus, $S \subseteq V$ is not a monotonic constraint.

- *Succinct Proof*:
  We can enumerate the sets that satisfy the constraint, which are the powerset of V, so the formula is $S \in \mathcal{P}(V)$.

(c) $min(S) \leq v$: not antimonotonic, monotonic, succinct.

Answer:

- *Antimonotone Counter-example*:
  Let $S = \{v_1\}$ and $v_1 > v$ where S violates the constraint. If we add s_2 to S, and $s_2 < v < s_1$ then $S' = \{s_1, s_2\}$ satisfies the constraint.
- *Monotone Proof:* Let $min(S) = v$. There must exist an element $e \in S$ such that $e \in v$. Let S' be a superset of S, then that same element e is contained in every superset S' of S. Thus, $min(S') = e = v$. Therefore, $min(S) \leq v$ is a monotone constraint.
- *Succinct Proof*:
  Let $A = \{v' such\ that\ v' < v\}$. We can get a formula for the sets that satisfy the constraint: $S = S_1 \cup S_2$, $S_1 \subseteq A$, $S_1 \neq \phi$.

(d) $range(S) \leq v$: antimonotonic, not monotonic, not succinct.

Answer:

- *Antimonotone Proof*:
  Assume S violates the constraint, so $max(S) - min(S) > v$. Now assume that we add a new element to v' to V. There are three cases to consider: (1) $v' < min(S)$ and clearly, $range(S) > v$; (2) $v' > max(S)$ and clearly, $range(S) > v$; (3) $min(S) \leq v' \leq max(S)$, where the range is unchanged so we still have $range(S) > v$.
- *Monotone Counter-example*:
  Assume it is monotonic. Let $S = \{v_1, ..., v_n\}$ satisfy the constraint, so $max(S) - min(S) \leq v$. Let $t = v - range(S)$. Add an element v' (such that $v' < min(S) - t$) to S. The new set violates the constraint, which contradicts our assumption of monotonicity.
- *Succinct Counter-example*: $range(S) \leq v$ is not a succinct constraint because we cannot explicitly and precisely generate all of 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.

■

5.19. 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.*

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 monotonic 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 $S2$ 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.

■

5.20. The price of each item in a store is nonnegative. For each of the following cases, identify the kinds of constraint they represent and briefly discuss how to mine such association rules *efficiently*.

(a) Containing at least one Nintendo game

(b) Containing items the sum of whose prices is less than \$150

(c) Containing one free item and other items the sum of whose 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 Nintendo game
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 Nintendo games need to be derived from the global FP-tree and mined recursively.

(b) Containing items the sum of whose prices is less than \$150
The constraint is antimonotonic. This constraint can be mined efficiently using Apriori as follows. Only candidates the sum of whose prices is less than \$150 need to be checked.

(c) Containing one free item and other items the sum of whose prices is at least \$200
The constraint is monotonic. (Or, subconstraints "*containing one free item*" and "*the sum of whose prices is less than \$150*" are succinct and monotonic, respectively.) This constraint can be mined efficiently using FP-growth as follows.
- Put all frequent free items at the end of the list of frequent items L.
- 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 items the sum of whose prices 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.

(d) Where the average price of all the items is between \$100 and \$150
The constraint is nonconvertible. (Or, the subconstraints "*the average price is at least \$100*" and "*the average price is at most \$500*" are convertible.) This constraint can be mined efficiently using FP-growth as follows.
- All 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.

■

Chapter 6

Classification and Prediction

6.17 Exercises

6.1. 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 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.

■

6.2. 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.

■

6.3. 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.

■

6.4. 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 $O(n \times |D| \times log(|D|))$.

■

6.5. 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.

■

6.6. 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.

■

6.7. RainForest is an interesting 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.

■

6.8. 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.)

■

6.9. 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 6.7. 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.

■

6.10. What is *association-based classification*? Why is association-based classification able to achieve higher classification accuracy than a classical decision-tree method? Explain how association-based classification can be used for text document classification.

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. Association-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, and uses very high confidence rules that combine multiple attributes.

For text document classification, we can model each document as a transaction containing items that correspond to terms. (We can preprocess the data to do stemming and remove stop words.) We also add the document class to the transaction. We then find frequent patterns and output rules of the form $term_1$, $term_2$, ..., $term_k \rightarrow class_i$ [sup=0.1, $conf$=0.9]. When a new document arrives for classification, we can apply the rule with highest support and confidence that matches the document, or apply a combination of rules as in CMAR.

■

6.11. 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?

(d) Design a multilayer feed-forward neural network for the given data. Label the nodes in the input and output layers.

(e) 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) 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 with the values *"systems"*, *"junior"*, and *"26...30"* for the attributes *department, status*, and *age*, respectively, what would a naïve Bayesian classification of the *salary* for the tuple be?

$P(X|senior) = 0$; $P(X|junior) = 0.018$. Thus, a naïve Bayesian classification predicts "*junior*".

(d) Design a multilayer feed-forward neural network for the given data. Label the nodes in the input and output layers.

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.

(e) 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.

No standard answer. Every feasible solution is correct.

■

6.12. 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.

■

6.13. Write an algorithm for *k-nearest neighbor classification* given k and n, the number of attributes describing each tuple.

Answer:

Figure 6.1 presents an algorithm for k-nearest neighbor classification.

■

6.14. The following table shows the midterm and final exam grades obtained for students in a database course.

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 6.1: k-nearest neighbor algorithm.

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.

■

6.15. 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 $y' = w_0' + w_1' x'$.

■

6.16. 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. This will help improve the accuracy of the next classifier, C_{t+1}. Using this technique, each classifier should have greater accuracy than its predecessor. 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.

■

6.17. Show that accuracy is a function of *sensitivity* and *specificity*, that is, prove Equation (6.58).

Answer:

$$\begin{aligned}
\text{accuracy} &= \frac{t_pos + t_neg}{(pos + neg)} \\
&= \frac{t_pos}{pos + neg} + \frac{t_neg}{(pos + neg)} \\
&= \frac{t_pos}{(pos + neg)} \times \frac{pos}{pos} + \frac{t_neg}{(pos + neg)} \times \frac{neg}{neg} \\
&= sensitivity \frac{pos}{(t_pos + t_neg)} + specificity \frac{neg}{(t_pos + t_neg)}.
\end{aligned}$$

■

6.18. 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:

$$t = \frac{\bar{d}}{s_d/\sqrt{n}}$$

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.

■

6.19. 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:

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.

■

Chapter 7

Cluster Analysis

7.13 Exercises

7.1. Briefly outline how to compute the *dissimilarity* between objects described by the following types of variables:

(a) Numerical (interval-scaled) variables

(b) Asymmetric binary variables

(c) Categorical variables

(d) Ratio-scaled variables

(e) Nonmetric vector objects

Answer:

(a) Numerical (interval-scaled) variables

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

$$d(i,j) = \sqrt{(x_{i1} - x_{j1})^2 + (x_{i2} - x_{j2})^2 + \cdots + (x_{in} - x_{jn})^2}. \tag{7.1}$$

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

$$d(i,j) = |x_{i1} - x_{j1}| + |x_{i2} - x_{j2}| + \cdots + |x_{in} - x_{jn}|. \tag{7.2}$$

(b) Asymmetric binary variables

If all binary variables have the same weight, we have the contingency Table 7.1.

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

Table 7.1: 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,

$$d(i,j) = \frac{r+s}{q+r+s}. \tag{7.3}$$

(c) Categorical variables

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 [OLD: the simple matching approach][NEW: on the ratio of mismatches]:

$$d(i,j) = \frac{p-m}{p}, \tag{7.4}$$

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.

(d) Ratio-scaled variables

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.

(e) Nonmetric vector objects

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:

$$s(\boldsymbol{x}, \boldsymbol{y}) = \frac{\boldsymbol{x}^t \cdot \boldsymbol{y}}{||\boldsymbol{x}||||\boldsymbol{y}||} \tag{7.5}$$

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 $\boldsymbol{y}$.

■

7.2. 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*.

(b) Compute the z-score for the first four measurements.

Answer:

(a) Compute the mean absolute deviation of *age*.

[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.

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.
According to the z-score computation formula,

$$z_{if} = \frac{x_{if} - m_{if}}{s_f}. \tag{7.6}$$

We have

$$\begin{aligned} z_{1f} &= \frac{18 - 33}{8.8} = -1.70 \\ z_{2f} &= \frac{22 - 33}{8.8} = -1.25 \\ z_{3f} &= \frac{25 - 33}{8.8} = -0.91 \\ z_{4f} &= \frac{42 - 33}{8.8} = 1.02 \end{aligned}$$

■

7.3. 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.

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

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

Answer:

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

$$\begin{aligned} d(i, j) &= \sqrt{(x_{i1} - x_{j1})^2 + (x_{i2} - x_{j2})^2 + \cdots + (x_{in} - x_{jn})^2} \\ &= \sqrt{|22 - 20|^2 + |1 - 0|^2 + |42 - 36|^2 + |10 - 8|^2} = 6.71 \end{aligned}$$

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

$$\begin{aligned} d(i, j) &= |x_{i1} - x_{j1}| + |x_{i2} - x_{j2}| + \cdots + |x_{in} - x_{jn}| \\ &= |22 - 20| + |1 - 0| + |42 - 36| + |10 - 8| = 11 \end{aligned}$$

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

$$\begin{aligned} d(i, j) &= (|x_{i1} - x_{j1}|^q + |x_{i2} - x_{j2}|^q + \cdots + |x_{in} - x_{jn}|^p)^{1/p} \\ &= (|22 - 20|^3 + |1 - 0|^3 + |42 - 36|^3 + |10 - 8|^3)^{1/3} = 6.15 \end{aligned}$$

■

7.4. Section 7.2.3 gave a method wherein a categorical variable having M states can be encoded by M *asymmetric binary variables*. Propose a more efficient encoding scheme and state why it is more efficient.

Answer:

Key points: Use $M-1$ variables. The M^{th} variable is instead represented by having all of the other variables set at 0.

■

7.5. Briefly describe the following approaches to clustering: *partitioning* methods, *hierarchical* methods, *density-based* methods, *grid-based* methods, *model-based* methods, methods for *high-dimensional data*, and *constraint-based* methods. Give examples in each case.

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.

- **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". The k-means algorithm is a commonly used partitioning method.
- **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 that are 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 of the objects in the same cluster. In each successive iteration, a cluster is split up into smaller clusters, until eventually each object forms its own cluster or until a termination condition holds. AGNES and DIANA are examples of hierarchical clustering. BIRCH integrates hierarchical clustering with iterative (distance-based) relocation.
- **Density-based methods**: These methods are based on the notion of density. 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 and OPTICS are typical examples of density-based clustering.
- **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 grid-based clustering.
- **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. A model-based algorithm may locate 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. It takes "noise" or outliers into account, therein contributing to the robustness of the approach. COBWEB and self-organizing feature maps are examples of model-based clustering.
- **Methods for high-dimensional data**: High-dimensional data can typically have many irrelevant dimensions. As the dimensionality increases, the data usually become increasingly sparse because the data points are likely located in different dimensional subspaces. The distance measurement between pairs of points become meaningless and the average density of points anywhere in the data is likely to

be low. Distance- and density-based clustering methods are therefore ineffective for clustering high-dimensional data. Alternative approaches have been proposed, such as *subspace clustering methods*, which search for clusters in subspaces (or subsets of dimensions) of the data, rather than over the entire data space. CLIQUE and PROCLUS are examples of subspace clustering methods. *Frequent pattern-based clustering* is another clustering methodology, which extracts distinct frequent patterns among subsets of dimensions that occur frequently. pCluster is an example of frequent pattern-based clustering that groups objects based on their pattern similarity.

- **Constraint-based methods**: These perform clustering by incorporating user-specified or application-oriented constraints. A constraint can express a user's expectation or describe "properties" of the desired clustering results, and provides an effective means for communicating with the clustering process. Constraint-based methods are used in spatial clustering for *clustering with obstacle objects* (e.g., considering obstacles such as rivers and highways when planning the placement of automated banking machines) and *user-constrained cluster analysis* (e.g, considering specific constraints regarding customer groups when determining the best location for a new service station, such as '*"must serve at least 100 high-value customers"*). In addition, semi-supervised clustering employs, for example, pairwise constraints (such as pairs of instances labeled as belonging to the same or different clusters) in order to improve the quality of the resulting clustering.

■

7.6. Suppose that the data mining task is to cluster the following eight points (with (x, y) representing location) into three clusters.

$$A_1(2, 10), A_2(2, 5), A_3(8, 4), B_1(5, 8), B_2(7, 5), B_3(6, 4), C_1(1, 2), C_2(4, 9).$$

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 and

(b) The final three clusters

Answer:

(a) 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 are: (1) $\{A_1, C_2, B_1\}$, (2) $\{A_3, B_2, B_3\}$, (3) $\{C_1, A_2\}$.

■

7.7. Both *k-means* and *k-medoids* algorithms can perform effective clustering. Illustrate the strength and weakness of *k-means* in comparison with the *k-medoids* algorithm. Also, illustrate the strength and weakness of these schemes in comparison with a hierarchical clustering scheme (such as AGNES).

Answer:

(a) Illustrate the strength and weakness of *k-means* in comparison with the *k-medoids* algorithm.

The k-medoids algorithm is more robust than k-means in the presence of noise and outliers, because a medoid is less influenced by outliers or other extreme values than a mean. However, its processing is more costly than the k-means method.

(b) Illustrate the strength and weakness of these schemes in comparison with a hierarchical clustering scheme (such as AGNES).

Both *k-means* and *k-medoids* perform partitioning-based clustering. An advantage of such partitioning approaches is that they can undo previous clustering steps (by iterative relocation), unlike hierarchical methods, which cannot make adjustments once a split or merge has been executed. This weakness of hierarchical methods can cause the quality of their resulting clustering to suffer. Partitioning-based

methods work well for finding spherical-shaped clusters. The quality of the resulting clustering is good, in general, for small to medium-sized databases. Their need to know the number of clusters in advance can be considered a weaknesses. Hierarchical clustering methods can determine the number of clusters automatically. However, they have difficulty scaling because each decision to merge or split may require the examination and evaluation of a good number of objects or clusters. Hierarchical methods, however, can be integrated with other clustering approaches, for improved clustering, such as in BIRCH, ROCK, and Chameleon.

■

7.8. 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 7.1. 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 $C2$. ($MinPts = 3$.)

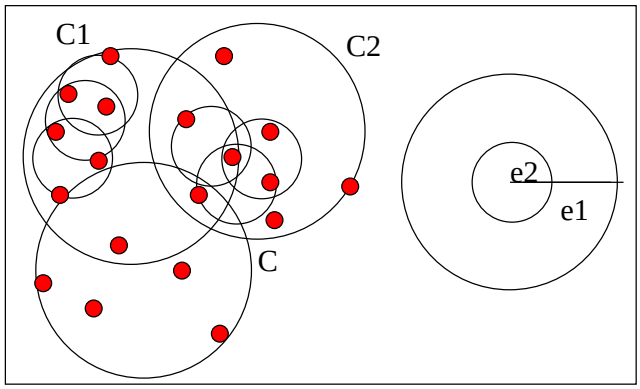

Figure 7.1: A diagram illustrating concepts of density-based clusters.

■

7.9. 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:

(a) Why is it that *BIRCH* encounters difficulties in finding clusters of arbitrary shape but *OPTICS* does not?

As a distance measure, BIRCH uses Euclidean distance and inter-cluster proximity, which leads to spherical shaped clusters.

OPTICS uses a density-based (connectivity) measure, which grows clusters based on the connected points within a defined radius, and thus can find arbitrary-shaped clusters.

(b) Can you propose some modifications to BIRCH to help it find clusters of arbitrary shape?

We could modify BIRCH so to use a density-based and connectivity-based distance measure to cluster low-level B+-trees and build the levels of the CF-tree. This would lead to connectivity-based (arbitrary-shaped) clusters. There could be several other ways to modify BIRCH, as well.

■

7.10. Present conditions under which *density-based* clustering is more suitable than *partitioning-based* clustering and hierarchical clustering. Give some sample data sets to support your argument.

Answer:

Partitioning-based clustering and hierarchical clustering methods cluster objects based on the distance between objects. Such methods can find only spherical-shaped clusters and encounter difficulty at discovering clusters of arbitrary shapes. Partitioning methods also have difficulty finding clusters of different diameters and densities. Hierarchical methods are expensive and cannot undo previous merge and split steps.

Density-based clustering methods (such as DBSCAN) are based on the notion that each cluster has a typical density of points that is considerably higher than outside of the cluster. These methods regard clusters as dense regions of objects in the data space that are separated by regions of low density (representing noise). Density-based clustering can discover clusters with arbitrary shape, unlike traditional clustering methods, and can handle noise in the data. Figure 7.2 shows three sample databases of objects. Traditional distance-based clustering methods will have difficulty finding the clusters in these data sets, yet they can easily be handled by DBSCAN.

Figure 7.2: Sample databases of objects.

■

7.11. 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. In addition, provide reasoning on why the integration of two methods may sometimes lead to improved clustering quality and efficiency.

Answer:

One example is BIRCH, a multi-phase clustering method that integrates hierarchical clustering (as a preprocessing step) with other clustering techniques. It begins by partitioning objects hierarchically using tree structures, and then applies other clustering algorithms (such as iterative partitioning) to refine the clusters. BIRCH incrementally constructs a clustering feature tree (CF tree), which is a height-balanced tree that stores the clustering features (summarized statistics) for a hierarchical clustering. BIRCH consists of two phases. Phase 1 (microclustering) 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 (macroclustering) applies another type of clustering algorithm (such as iterative relocation) to cluster the leaf nodes of the CF-tree, which removes sparse clusters as outliers and groups dense clusters into larger ones.

Integrating different types of clustering methods may help overcome some of the weaknesses of the individual approaches. For example, by integrating hierarchical clustering with other clustering techniques, BIRCH overcomes two major limitations of hierarchical clustering—scalability and the inability to undo what was done in the previous step.

■

7.12. 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*

(b) An application that takes clustering as a *preprocessing tool* for data preparation for other data mining tasks

Answer:

(a) An application that takes clustering as a major *data mining function*

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

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 of nonspatial components in the next step if such a generation may form interesting groups of objects.

■

7.13. *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 clusters in a large data cube because a data cube can be interpreted as a multiresolution grid data structure. We first need to preprocess and discretize existing data (such as ordinal and numerical data) 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 them. 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.

■

7.14. *Subspace clustering* is a methodology for finding interesting clusters in high-dimensional space. This methodology can be applied to cluster any kind of data. Outline an efficient algorithm that may extend density connectivity-based clustering for finding clusters of arbitrary shapes in projected dimensions in a high-dimensional data set.

Answer:

One possible solution is to extend CLIQUE as follows:

(a) Partition each dimension into grids.

(b) Let $k = 1$, repeat the following:

i. Find the dense cells in dimension k. A cell is dense if its number of points (inside the slot) is $\geq \sigma$.

ii. Based on Apriori, construct candidate sets for the $(k+1)$-dimensions.

(c) Until (no dense cells) or (candidate set is empty).

(d) After finding dense cells in a k-D plane, find clusters as the most-connected dense units and describe them using simplified (rectangular) regions.

■

7.15. [*Contributed by Alex Kotov*] Describe each of the following clustering algorithms in terms of the following criteria: (i) shapes of clusters that can be determined; (ii) input parameters that must be specified; and (iii) limitations.

(a) k-means

(b) k-medoids

(c) CLARA

(d) BIRCH

(e) ROCK

(f) CHAMELEON

(g) DBSCAN

Answer:

(a) k-means
(i) Spherical-shaped clusters; (ii) The number of clusters; (iii) Sensitive to noise and outliers. Works well on small data sets only.

(b) k-medoids
(i) Spherical-shaped clusters; (ii) The number of clusters; (iii) Small data sets (not scalable).

(c) CLARA
(i) Spherical-shaped clusters; (ii) The number of clusters; (iii) Sensitive to the selection of initial samples.

(d) BIRCH
(i) Spherical-shaped clusters; (ii) N d-dimensional data points; (iii) Because a CF tree can hold only a limited number of entries due to its size, a CF tree does not always correspond to what a user may consider a natural cluster.

(e) ROCK
(i) Arbitrary shape; (ii) N d-dimensional categorical data points; (iii) Designed for categorical data, emphasizes interconnectivity, ignores closeness between clusters.

(f) CHAMELEON
(i) Arbitrary shape; (ii) N d-dimensional categorical points; (iii) Quadratic time in the worst case.

(g) DBSCAN
(i) Arbitrary shape; (ii) Maximum possible distance for a point to be considered density-reachable and minimum number of points in a cluster; (iii) Quadratic time in the worst case.

■

7.16. [*Contributed by Tao Cheng*] 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 perform clustering. Due to the probabilistic nature of such models, EM clustering can easily be extended to handle data with both numerical attributes and categorical attributes. Under the assumption that each attribute is independent of one other, we can model numerical attributes by using density functions, such as Poisson, Gaussian, etc.; categorical attributes can be modeled by associating discrete distributions 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 can easily use EM clustering to cluster data with both numerical attributes and categorical attributes.

Reference: P. Bradley, U. Fayyad, C. Reina, Scaling Clustering Algorithms to Large Databases, in *Proc. 1998 Int. Conf. Knowledge Discovery and Data Mining (KDD'98)*, pages 9-15, Aug. 1998 [BFR98].

■

7.17. Human eyes are fast and effective at judging the quality of clustering methods for two-dimensional data. Can you design a *data visualization* method that may help humans visualize data clusters and judge the clustering quality for three-dimensional 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.

■

7.18. 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):

- **Microcluster.** Objects are clustered locally into microclusters. In a microcluster, no obstacle can appear between any two other objects in the microcluster.
- **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. Suppose 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.

■

7.19. 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:

Research by Tung, Han, Lakshmanan, and Ng [THLN01] and Tung, Hou, and Han [THH01] illustrate different kinds of constraints and how to handle them for constraint-based clustering. Section 7.10 of the book presents a general overview of this topic. Students may further summarize these ideas for concise answers.

■

7.20. 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:

A basic approach to preserving privacy is to let users provide a modified value for sensitive attributes. Here we summarize two such methods from R. Agrawal and R. Srikant, Privacy-preserving data mining, in *Proc. of the 2000 ACM SIGMOD International Conference on Management of Data.* pages 439-450, May 2000 [AS00].

- **The value-class membership method**: In this method, the values for an attribute are partitioned into a set of disjoint, mutually-exclusive classes. Consider the special case of discretization in which values for an attribute are discretized into intervals. All intervals need not be of equal width. For example, salary may be discretized into 10K intervals for lower values and 50K intervals for higher values. Instead of a true attribute value, the user provides the interval in which the value lies. Discretization is the method used most often for hiding individual values.
- **The value distortion method**: Return a value $x + r$ instead of x, where r is random value drawn from some distribution. Two random distributions include:
  - Uniform: The random variable has a uniform distribution, between $[-\alpha, +\alpha]$. The mean of the random variable is 0.
  - Gaussian: The random variable has a normal distribution, with mean 0 and standard deviation.

■

7.21. Why is outlier mining important? Briefly describe the different approaches behind *statistical-based outlier detection, distanced-based outlier detection, density-based local 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 $DB(p, dmin)$-outlier is an object $\boldsymbol{o}$ in a data set D such that at least a fraction p of the objects in D lies at a distance greater than $dmin$ from $\boldsymbol{o}$. 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.

Deviation-based outlier detection does not use statistical tests or distance-based measures to identify exceptional objects. 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 approach include a sequential exception technique and an OLAP data cube technique.

■

7.22. *Local outlier factor* (LOF) is an interesting notion for the discovery of local outliers in an environment where data objects are distributed rather unevenly. However, its performance should be further improved in order to efficiently discover local outliers. Can you propose an efficient method for effective discovery of local outliers in large data sets?

Answer:

Local outlier factor (LOF) is an interesting notion for the discovery of local outliers in an environment where data objects are distributed rather unevenly, however, the computation of LOF value for every data object is costly, requiring the computation of a large number of k-nearest neighbors. Since only a small portion of the object can be outliers, an alternative way is to compute only the top-n local outliers. This will save a great deal of computational effort. An efficient microclustering-based method for this is described in W. Jin, K. H. Tung and J. Han, *Mining Top-n Local Outliers in Large Databases*", in *Proc. 2001 ACM SIGKDD Int. Conf. Knowledge Discovery in Databases (KDD'01)*, San Fransisco, CA, pages 293-298, Aug. 2001 [JTH01]. Interested readers may refer to this paper for details. Moreover, the method was further developed for improved definition of local outliers, leading to improved quality on local outlier analysis. This refinement is detailed in W. Jin, A. K. H. Tung, J. Han, and W. Wang, *Ranking Outliers Using Symmetric Neighborhood Relationship*, in *Proc. 2006 Pacific-Asia Conf. Knowledge Discovery and Data Mining (PAKDD'06)*, Singapore, April 2006 [JTHW06].

■

Chapter 8

Mining Stream, Time-Series, and Sequence Data

8.6 Exercises

8.1. A *stream data cube* should be relatively stable in size with respect to infinite data streams. Moreover, it should be incrementally updateable with respect to infinite data streams. Show that the stream cube proposed in Section 8.1.2 satisfies these two requirements.

Answer:

A stream data cube takes a set of potentially infinite data streams as input. If the size of the base cuboid grows indefinitely with the size of data streams, the size of the stream data cube will grow indefinitely. It is impossible to realize such a stream data cube. Fortunately, with the tilted time frame model, the distant time is compressed substantially and the very distant data beyond the specified time frame are faded out (i.e., removed). Thus, the bounded time frames transform infinite data streams into a finite, compressed representation. If the data in the other dimensions of the base cuboid (i.e., the minimal interest layer in the design) are relatively stable with time, then the entire base cuboid (with the time dimensions included) should be relatively stable in size. Similarly, any optional cuboid along the popular path and the observation cuboid (i.e., the observation layer in the design) should also be relatively stable in size. Hence the whole stream cube, should be relatively stable in size with respect to infinite data streams.

To incrementally update a stream data cube, one must start from the incremental portion of the base cuboid and use an efficient algorithm to compute it. The time to compute such an incremental portion of the cube should be proportional (desirably, linear) to the size of the incremental portion of the base cuboid of the cube.

■

8.2. In stream data analysis, we are often interested in only the nontrivial or exceptionally large cube cells. These can be formulated as *iceberg conditions*. Thus, it may seem that the iceberg cube [BR99] is a likely model for stream cube architecture. Unfortunately, this is not the case because iceberg cubes cannot accommodate the incremental updates required due to the constant arrival of new data. Explain why.

Answer:

Given the incremental and gradual arrival of new stream data as well as the incremental fading of the obsolete data from the time scope of a data cube, we need to be able to perform *incremental updates* on such a stream data cube. It is unrealistic to constantly recompute the data cube from scratch upon incremental updates due to the tremendous cost of recomputing the cube on the fly. Unfortunately, such an incremental model does not fit the iceberg cube computation model due to the following observation.

Let a cell "$\langle d_i, \ldots, d_k \rangle : m_{ik}$" represent a $(k - i + 1)$-dimensional cell with $d_i, \ldots, d_k$ as its corresponding dimension values and m_{ik} as its measure value. If m_{ik} does not satisfy the iceberg condition, the cell is dropped from the iceberg cube. However, at a later time slot, t', a new measure, m'_{ik}, related to t' may be added to the cell $\langle d_i, \ldots, d_k \rangle$. Because m_{ik} has been dropped at a previous instance of time due to its inability to satisfy the iceberg condition, the new measure for this cell cannot be calculated correctly without such information. Thus one cannot use the iceberg architecture to model a stream data cube without recomputing the measure from the base cuboid upon each update.

■

8.3. An important task in stream data analysis is to *detect outliers* in a multidimensional environment. An example is the detection of unusual power surges, where the dimensions include *time* (i.e., comparing with the normal duration), *region* (i.e., comparing with surrounding regions), *sector* (i.e., university, residence, government), and so on. Outline an efficient stream OLAP method that can detect outliers in data streams. Provide reasons as to why your design can ensure such quality.

Answer:

A popular approach in stream data analysis is to think of stream data as a stream data cube. The advantage of this approach is the ability to view data at different granularities in a multidimensional environment. Let us assume that the user has specified two critical layers: the minimal interest layer and the observation layer. We will outline an efficient stream OLAP method to detect outliers in data streams.

The method employs the "discovery-driven" approach presented in Sarawagi, Agrawal, and Megiddo [SAM98], where the search for outliers is guided by potential "exception cells" that are precomputed. A cell is considered exceptional if its value differs significantly from the expected value based on a statistical model. The model is described in detail in the forementioned paper and hence will be explained very briefly here. (See also Section 4.2.1. for a summary.) Basically, there are three logical phases in computing exceptions in the stream data cube. The first phase computes the aggregated values efficiently. These values are used to fit the model in the second phase and then summarized top-down in the third phase. These phases are efficiently computed using the UpDown method presented in the paper and model equation rewriting.

The exception model is based on several criteria, which provide reasons on why this design can ensure quality. First, the model should consider patterns and variations across all dimensions. Second, exceptions can appear at all aggregated levels (including the observation layer), and the detailed level (from the minimal interest layer). Third, analysts/users should be able to interpret why a cell is marked as an exception. Lastly, the procedure to compute these exception cells needs to be efficient. Since these four desiderata are all satisfied by the aforementioned model, it is adequate to ensure quality.

■

8.4. *Frequent itemset mining in data streams* is a challenging task. It is too costly to keep the frequency count for every itemset. However, because a currently infrequent itemset may become frequent, and a currently frequent one may become infrequent in the future, it is important to keep as much frequency count information as possible. Given a fixed amount of memory, can you work out a good mechanism that may maintain high quality approximation of itemset counting?

Answer:

The number of frequent itemsets grows exponentially compared to the number of frequent items. As in *Lossy Counting* we process as many buckets that fit into memory at a time. If we want the approximation with error bound ϵ, transactions are divided by buckets with bucket size $\lceil 1/\epsilon \rceil$. We maintain a frequency list (*count* and *maximum error* for each itemset), and we update it as follows: 1) if an itemset already exists in the list, we update its count, and 2) if the new count plus maximum error is less than the current bucket number b, we delete the itemset from the list. If an itemset has frequency greater than number of buckets in the memory β and it is not in the list, we add it to the list with maximum error equal to $b - \beta$. This method works efficiently when β is large (greater than 30) since all itemsets with frequency less than β are not recorded.

The frequency list may grow infinitely as the stream flows. If a new itemset has frequency greater than the number of buckets in the memory and does not appear in the frequency list, it should be inserted as a new

entry. However, if there is no memory available, we can remove from the frequency list the itemsets for which the itemset count plus maximum error is low. If, after this, there is still no memory available for the insertion, a disk-based *Lossy Counting* should be used.

■

8.5. For the above approximate frequent itemset counting problem, it is interesting to incorporate the notion of *tilted time frame*. That is, we can put less weight on the more remote itemsets when counting frequent itemsets. Design an efficient method that may obtain high-quality approximation of itemset frequency in data streams in this case.

Answer:

We divide time up into logarithmic time frames (or windows) and save an FP-tree (frequent-pattern tree) for each time frame. Except for the first two (which have a size of 1 batch), the size of each frame is double that of the last. So when a user queries from the last h batches, we can always query within $h/2$ of the batches.

To update and weigh these in an efficient manner, we move the data down the frame every time a new batch arrives. Because of the tilted time frame, we only need to combine, not to split, the FP-tree. The amortized number of combinations per batch is 1 for each frame that is replaced. Half the time a batch will overflow into a temporary buffer, but the other half of the time the buffer will already be full, and the two will combine to create a new FP-tree for the next frame. Combining only requires linear time with respect to the number of items in the FP-tree, therefore, this approach is scalable.

We can multiply the combined FP-tree at the combining step by a constant less than 1 to introduce exponential decay of the weights of the frequency counts. Furthermore, we can eliminate low counts at each step, since their weights are decreasing. Another approach is to just multiply by the constant at query time (n number of times for the n^{th} frame, where n goes up as time goes back). However, although this approach allows users to define the "forgetfulness" of the model, it requires multiplication at query time, and worse, requires much more space since low frequency counts in the past cannot be discounted and eliminated. Hence, this alternative approach is not recommended.

■

8.6. A classification model may change dynamically along with the changes of training data streams. This is known as *concept drift*. Explain why decision tree induction may not be a suitable method for such dynamically changing data sets. Is naïve Bayesian a better method on such data sets? Comparing with the naïve Bayesian approach, is lazy evaluation (such as the k-nearest neighbor approach) even better? Explain your reasoning.

Answer:

Suppose concept drift is handled as follows: A classifier is built for the current time window and will be constantly updated as time goes by. Specifically, suppose we have a classifier for time window starting at t, and we need to update the classifier for the next time window starting at $t + 1$. Let the size of the time window (number of data points) be n, so that the two time windows (or data sets) share $n - 1$ data points. Let's compare the suitability of decision tree induction, naïve Bayesian classification, and lazy evaluation with respect to data from each time window (i.e., for the classification of dynamically changing data sets).

- *Decision tree* as the classifier: Even though the two data sets have all but one data point in common, we need to learn each decision tree from scratch because the splitting attribute for a decision tree can be changed by a single data point.
- *Naïve Bayesian* as the classifier: Naïve Bayesian classification uses only the counts to compute the parameters needed, thus it is easy to update the classifier by simply changing the counts. To delete an expired data point and add the incoming one, we only need to reduce the count corresponding to the expired data point and increase the count corresponding to the new data point. All the other counts and hence parameters will not be affected.

- *Lazy evaluation* (such as k-nearest neighbor classification): Classifiers that use lazy evaluation store all of the training data points in pattern space and wait until presented with a test data point before performing generalization. Hence, there is no need to update the classifier, except to drop the old data point and add the new one.

In summary, in terms of efficiency in stream data classification: lazy evaluation > naïve Bayesian > decision tree.

■

8.7. The concept of microclustering has been popular for on-line maintenance of clustering information for data streams. By exploring the power of microclustering, design an effective *density-based* clustering method for clustering evolving data streams.

Answer:

Basically, we can adopt the mining framework proposed in CluStream [AHWY03], but where the clustering method is replaced by a density-based approach. CluStream is an algorithm for the clustering of evolving data streams based on user-specified, on-line clustering queries. It divides the clustering process into on-line and off-line components. The on-line component computes and stores summary statistics about the data stream using microclusters, and performs incremental on-line computation and maintenance of the microclusters. The off-line component does macroclustering and answers various user queries using the stored summary statistics, which are based on the tilted time frame model.

The main techniques include:

(a) **Tilted time frame:**
We can use a tilted time frame for the time dimension to keep the data snapshots. That is, we take a snapshot of the data at every time point but only keep a few of them such that the granularity of data snapshots stored are finer for recent time points and coarser for historical time points.

(b) **On-line microclustering:**
We can do microclustering on the stored data snapshots and only keep a fixed number of microclusters. A microcluster is represented by a *clustering feature (CF)*, which summarizes statistical information regarding data locality. Clustering features, developed in BIRCH (Section 7.5.2), are additive. For example, if merging two disjoint microclusters, we simply add their respective CF's to obtain the CF of the newly formed cluster. For clustering data streams, we can extend the concept of the clustering feature to include the temporal domain. Basically, BIRCH 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 tree* (Clustering Feature tree), which is a height-balanced tree that stores the clustering features for a hierarchical clustering. It first 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. A density-based clustering algorithm can then be applied to cluster the leaf nodes of the CF tree. The hierarchical clustering technique is therefore used as a preprocessing step. On-line incremental updates of the microclusters can be easily maintained because the microclusters are additive. When new data come in, we either merge the data with an existing microcluster, split the cluster under some criteria, or construct a new cluster and drop an old one.

(c) **Off-line macroclustering:**
Query-based macroclustering can be approached based on a user-specified time-horizon h and the number of macroclusters K. Given a time period h, we can find two data snapshots based on the tilted time frame and use the difference between the two snapshots to approximate the incoming data in this time period. Again, we can use a density-based approach (e.g., DBSCAN, Section 7.6.1) to cluster these microclusters. Because the macroclustering here is based on centers of microclusters instead of original data points, this process is very efficient.

■

8.8. Suppose that a power station stores data regarding power consumption levels by time and by region, in addition to power usage information per customer in each region. Discuss how to solve the following problems in such a *time-series database*.

(a) Find similar power consumption curve fragments for a given region on Fridays.

Answer:

We use similarity search methods for this problem. The power consumption curve for a given region for each Friday is first broken down into a set of fragments using a user-specified method such as a sliding window. Each fragment is indeed a sequence. Finding similar power consumption curve fragments among curves on different Fridays is then reduced to similarity search of a fragment in one Friday curve against any fragment generated from any other Friday curve. Similarity search between two fragments is performed as follows: First, gaps are removed. The resulting subsequences are normalized (with respect to offset translation and amplitude scaling) to solve the baseline and scale problem (we replace each point with $(x_i - \mu)/\sigma$). At the end, two subsequences are considered similar if one lies within an envelope of ϵ width around the other, ignoring outliers (where ϵ is a small number, specified by a user or expert). This method finds the similar curve fragments for a given region on Fridays.

■

(b) Every time a power consumption curve rises sharply, what may happen within the next 20 minutes?

Answer:

This problem involves time-series forecasting, which is used to make long-term or short-term predictions of future values. First, we use similarity search methods to find the curves that rise sharply. We can then retrieve the subsequences of the next 20 minutes of these curves. We decompose our data into the four major components for modeling time series: T (trend), C (cyclic), S (seasonal), and I (irregular). We want to make a cyclic prediction. To do this, we first normalize the C component. We can then compute the averages of different curves and use these averages to make predictions about the future. Alternatively, instead of computing averages, we can find similar curves within these 20 minute subsequences, from which we find similarity classes. Each class has a probability assigned to it (e.g., based on the number of curves that lie in one class). We can then claim, with some probability, that the new curve will be in a certain similarity class, and so on.

■

(c) How can we find the most influential features that distinguish a stable power consumption region from an unstable one?

Answer:

In order to find the most influential features, we need to perform classification over regions. We divide regions into two classes: stable and unstable. To find the most influential features that determine the class of a particular region, we apply a classification algorithm, such as decision tree induction. For example, the influential features may be those with the highest information gain.

■

8.9. Regression is commonly used in trend analysis for *time-series data sets*. An item in a time-series database is usually associated with properties in multidimensional space. For example, an electric power consumer may be associated with consumer location, category, and time of usage (weekdays vs. weekends). In such a multidimensional space, it is often necessary to perform *regression analysis in an OLAP manner* (i.e., drilling and rolling along any dimension combinations that a user desires). Design an efficient mechanism so that regression analysis can be performed efficiently in multidimensional space.

Answer:

Multidimensional regression analysis has been studied by Chen et al. [CDH+02]. The general idea is that instead of computing all of the regression cuboids from scratch, one can compute them level by level by computing a few essential primitive or regression parameters. Regression lines or curves can be derived from such parameters easily. Chen has shown that such computation is very efficient, costs little memory space, and is suited for data cube and data stream computation. For details, please refer to [CDH+02].

■

8.10. Suppose that a restaurant chain would like to mine customers' consumption behavior relating to major sport events, such as *"Every time there is a major sport event on TV, the sales of Kentucky Fried Chicken will go up 20% one hour before the match".*

(a) For this problem, there are multiple sequences (each corresponding to one restaurant in the chain). However, each sequence is long and contains multiple occurrences of a (sequential) pattern. Thus this problem is different from the setting of sequential pattern mining problem discussed in this chapter. Analyze what the differences are in the two problem definitions and how such differences may influence the development of mining algorithms.

Answer:

In the original problem definition, any sequence S that contains a pattern P will contribute to the support of P by one, regardless of how many times P occurs in S. In the new setting, clearly we would like to incorporate the actual count information of P in the sequences. Intuitively, a pattern that has a larger number of total counts would be more interesting than a pattern that occurs in the same number of sequences but with a smaller number of counts. There are different ways of formulating the new problem. Two of them are as follows. Let P be a sequential pattern:

i. P is frequent if (1) the number of sequences that include P as a subsequence is greater than or equal to the minimum support, *min_sup*; and (2) the total number of occurrences of P in all sequences is greater than or equal to *min_total_counts*. We need to satisfy both (1) and (2) instead of only (2) because if (1) is not a constraint, then we may find patterns that occur many times in a single sequence, but that do not occur in all of the other ones. It would thus be unclear whether these patterns are frequent across the database.

ii. P is frequent if the number of sequences where P occurs as a subsequence for no less than *min_count* times is greater than or equal to *min_sup*. In other words, only the sequences where P occurs for at least a certain number of times will contribute to the support of P. Note that when *min_count* is equal to 1, this reduces to the original problem definition.

■

(b) Develop a method for finding such patterns efficiently.

Answer:

For the first problem formulation, a possible method consists of two steps as follows. At the first step, we use any sequential pattern mining algorithm, such as PrefixSpan to mine the candidate patterns. These patterns will satisfy the first constraint. In the second step, we select the patterns that also meet the second criterion by collecting the counts of each candidate pattern. A prefix-tree can be used to hold all of the candidate patterns and only a scan of one restaurant sequence is needed to collect the counts in that restaurant sequence.

For the second problem formulation, a possible method may consist of two steps as follows. In the first step, we find within each restaurant sequence any candidate pattern that occurs no less than *min_count* times, and build a prefix-tree to hold all of the candidate patterns. In the second step, we merge all of the prefix-trees to collect the number (i.e., inter-sequence count) of sequences in which each candidate pattern occurs, and output the pattern if the inter-sequence count count is greater than or equal to *min_sup*.

■

8.11. (**Implementation project**) The sequential pattern mining algorithm introduced by Srikant and Agrawal [SA96] finds sequential patterns among a set of sequences. Although there have been interesting follow-up studies, such as the development of the algorithms SPADE (Zaki [Zak01]), PrefixSpan (Pei, Han, Mortazavi-Asl, et al. [PHMA$^+$01]), and CloSpan (Yan, Han, and Afshar [YHA03]), the basic definition of "sequential pattern" has not changed. However, suppose we would like to find frequently occurring subsequences (i.e., *sequential patterns*) *within one given sequence*, where, say, gaps are not allowed. (That is, we do not consider AG to be a subsequence of the sequence ATG). For example, the string ATGCTCGAGCT contains a substring GCT with a support of 2. Derive an efficient algorithm that finds the complete set of subsequences satisfying a minimum support threshold. Explain how your algorithm works using a small example, and show some performance results for your implementation.

Algorithm: Find the complete set of subsequences satisfying a minimum support threshold.

Input:

- S: a sequence
- *min_sup*: the minimum support threshold

Output: The complete set of sequential patterns.

Method: Call SubSeqs(ϕ, 0, S).

Subroutine: SubSeqs(α, l, $S|_\alpha$).
The parameters are:
1) α is a sequential pattern
2) l is the length of α
3) $S|_\alpha$ consists of the α-projected sequences if $\alpha \neq \phi$, otherwise, it is the original sequence S.
Method:
1. Scan $S|_\alpha$ once, find each frequent item b
   such that b can be appended to α to form a sequential pattern.
2. For each frequent item b, append it to α to form a sequential pattern α', and output α'.
3. For each α', construct α'-projected sequences $S|_{\alpha'}$, and call SubSeqs(α', 1, $S|_{\alpha'}$).

Figure 8.1: Find frequently occurring subsequences within a given sequence.

AACGGTCAACGAT						
A	C	G	T	AA	AC	...
2,3,9,10,13	4,8,11	5,6,12	7,14	3,10	4,11	...

Table 8.1: An input sequence (AACGGTCAACGAT) and its pseudo-projected subsequences.

Answer:

An algorithm that follows the basic idea of PrefixSpan can be used for this problem, as illustrated in Figure 8.1. Based on this algorithm, we also need to conduct some optimization strategies to improve the performance. First, the idea of pseudo-projection from PrefixSpan as described in the text can be directly applied in this problem. That is, each α-projected sequence is basically a subsequence with α as the prefix. Using pseudo-projection, we don't need to copy every α-prefixed subsequence when doing the depth-first-search. Instead, we can just register the starting position of the projected suffix for each α-prefixed subsequence. This saves both space and time by avoiding the generation of numerous physical projected subsequences. Table 8.1 shows an example of a sequence and some of its pseudo-projected subsequences. When we grow the patterns, we only need to increase the registered position by one since gaps are not allowed. During the depth-first-search, any sequence pattern with a support of less than *min_sup* can be pruned. This saves the time of searching in the subtree of this pattern.

The amount of resulting frequent subsequences may be huge. In this case, mining closed sequential patterns may be more desirable and efficient.

In this problem, we can adopt the method of checking for closed subsequences during the mining process as proposed in BIDE [WH04]. Basically, before mining the α-projected sequences $S|_\alpha$, we can simply compute the indices of the positions that are exactly one position before the beginning of the α-prefixed subsequences, and check whether the set of these indices is a subset of a frequent single-item-projected subsequence. That is, we check whether all of the α-prefixed subsequences follow the same item in the input sequence. For example, in Table 8.1, before growing the AC-projected subsequences, we get the set of its pseudo-projected indices, $\{4, 11\}$. The indices of the positions that are exactly one position before AC can be computed as $\{4-2, 11-2\} = \{2, 9\}$, which is a subset of the indices of A-projected subsequences $\{2, 3, 9, 10, 13\}$. Therefore, we can infer that every AC-prefixed subsequence with length l must be a subsequence of some

AAC-prefixed subsequence with length $l+1$. Thus, AC-prefixed subsequences cannot satisfy the closedness constraints and must be pruned immediately to save time and space. The main computation involved here is to check whether the items by indices $\{2, 9\}$ are the same. This small overhead of closed-sequence checking requires much less time than searching in the subtrees and will likely be faster than complete sequential pattern mining.

■

8.12. Suppose frequent subsequences have been mined from a sequence database, with a given (relative) minimum support, *min_sup*. The database can be updated in two cases: (i) adding new sequences (e.g., new customers buying items), and (ii) appending new subsequences to some existing sequences (e.g., existing customers buying new items). For *each case*, work out an efficient *incremental mining* method that derives the complete subsequences satisfying *min_sup*, without mining the whole sequence database from scratch.

Answer:

Let DB be the original sequence database. For case (i), let DB' be a database containing all of the new sequences to be inserted into DB. Let S be a new sequence from DB' to be inserted into DB. If S is frequent in the final updated database $DB + DB'$, then S must be frequent in either DB or DB'. Based on this observation, we can apply a sequential mining algorithm (e.g., PrefixSpan) to DB and DB', respectively. Because DB has already been mined, only DB' should be checked. The set of frequent patterns in DB and the set in DB' will serve as candidate patterns for the final frequent patterns. Thus, we only need to scan to get the candidates' frequency count to determine which patterns are truly frequent.

In case (ii), a pattern S that is infrequent in both DB and DB' may become frequent in $DB+DB'$, because infrequent sequential patterns in DB can become frequent by appending new items. To deal with this, we can maintain another set, C, of subfrequent pattern candidates. The sequences in C are below the *min_sup* threshold, but not by too much, which means that they are likely to become frequent in the future. Note that the prefix of a frequent sequential pattern must also be frequent. Thus, C can be extended from the PrefixSpan algorithm. Each time a new sequence is added to DB', we can check for each pattern S in C to see if it becomes frequent. If so, S will be moved from C to the frequent set. In addition, if any pattern in the frequent set becomes subfrequent but not infrequent, we move it into candidate set, C.

A serious study of this problem has been done by Cheng, Yan, and Han [CYH04]. Please refer to that paper for the detailed algorithm.

■

8.13. Closed sequential patterns can be viewed as a lossless compression of a large set of sequential patterns. However, the set of closed sequential patterns may still be too large for effective analysis. There should be some mechanism for *lossy compression* that may further reduce the set of sequential patterns derived from a sequence database.

Answer:

Note: This problem is a good research topic. We have not seen an existing research paper on this but some of my students may take this as a potential research project.

■

(a) Provide a good definition of lossy compression of sequential patterns, and reason why such a definition may lead to effective compression with minimal information loss (i.e., high compression quality).

Answer:

Xin et al. [XHYC05] has recently performed an interesting study on compressing frequent patterns. The essential definition is based on both pattern representation and patter support. That is, two patterns are similar if they have similar representation and similar support. This notion can be extended to define compressed sequential patterns.

■

(b) Develop an efficient method for such pattern compression.

Answer:

This algorithm is a research topic, whose standard answer will be worked out after our further research.

■

(c) Develop an efficient method that may mine such compressed patterns directly from a sequence database.

Answer:

This algorithm is a research topic, whose standard answer will be worked out after our further research.

■

8.14. As discussed in Section 8.3.4, mining partial periodic patterns requires a user to specify the length of the period. This may burden the user and reduces the effectiveness of mining. Propose a method that will *automatically mine the minimal period of a pattern* without requiring a predefined period. Moreover, extend the method to find *approximate periodicity* where the period need not be precise (i.e., it can fluctuate within a specified small range).

Answer:

We first examine the case where the period is exact, albeit unknown. We consider only categorical data. The input of the problem is a sequence S, defined on some alphabet Σ. Each symbol in Σ represents an event. Our goal is to find all events that show periodicity in S.

Suppose some minimum support, *min_sup*, is defined as the minimum number of consecutive occurrences of an event (spaced by the same period) required in order for the event to be called periodic. The first step of the algorithm extracts the positions of each event. Thus, for each event, we have a sequence of monotonically increasing integers that represents where this event occurs in S.

The next step finds the hidden period for each event. Let the event be E. Note that in addition to the periodic occurrences of E, there may be other occurrences of E. For example, suppose $\{2, 8, 14, 15, 20, 26, 29, 32\}$ is the sequence for E. The actual period for E is 6. The values 15 and 29 represent noise in the sequence. Thus, there are 5 periods in the sequence. An approach to finding the minimal period of E is to find the *longest arithmetic progression* in the sequence for E. That is, given an array $A[1 \ldots n]$ of integers, we wish to find the largest sequence of indices $\langle i_0, i_1, \ldots, i_{k-1} \rangle$ such that $A[i_j] - A[i_{j-1}] = A[i_1] - A[i_0]$ for all i. This will lead to the period for E and the number of times E occurs in the periodic pattern. Dynamic programming algorithms have been developed for the longest arithmetic progression problem (e.g., see *compgeom.cs.uiuc.edu/~jeffe/pubs/arith.html*).

To find events with approximate periodicity, we adopt the same framework described above but add a parameter, ϵ. If T is the real period, then two consecutive occurrences of E separated by $T + \epsilon$ or $T - \epsilon$ would also be acceptable. It is not hard to see that the longest arithmetic progression algorithm can be modified to accommodate this situation as well.

■

8.15. There are several major differences between *biological sequential patterns* and *transactional sequential patterns*. First, in transactional sequential patterns, the gaps between two events are usually nonessential. For example, the pattern "*purchasing a digital camera two months after purchasing a PC*" does not imply that the two purchases are consecutive. However, for biological sequences, gaps play an important role in patterns. Second, patterns in a transactional sequence are usually precise. However, a biological pattern can be quite imprecise, allowing insertions, deletions, and mutations. Discuss how the mining methodologies in these two domains are influenced by such differences.

Answer:

Unlike transactional sequences, biosequences typically have a small alphabet, a long length, and patterns containing gaps of arbitrary size. Mining frequent patterns in such sequences faces a very different type of pattern explosion than mining in transactional sequences (which are common in market-basket analysis) for the following reasons.

(a) Biosequences generally have a very limited alphabet, e.g., four symbols for DNA sequences and twenty symbols for protein sequences. In contrast, transactional sequences usually have a large alphabet, and typically, only a tiny fraction of items occur in any given transaction sequence. Because of the denseness of sequential patterns in biosequences (that is, most items occur in every biosequence), pruning strategies and data structures that are used for transactional sequence mining will generally not be effective for mining biosequences. One example is the vertical data structure used in frequent pattern mining. In biosequences, the TID-list is typically very large and may incur serious overheads in time and space for the mining process.

(b) A biosequence is generally very long (> 100). The classic sequential pattern-growth strategy only extends the pattern one item at a time, and thus may require many database scans on each sequence. When the biosequence database is huge and cannot be loaded into memory as a whole, the pattern-growth strategy will likely need to scan all of the sequences at every instance of pattern growth and database projection. In this situation, the overhead of this approach is nontrivial. Therefore, techniques aimed at mining short sequences over a large alphabet are less effective for biosequences.

(c) Mutation of amino acids is a common phenomenon. For example, an amino acid is likely to mutate to another type of amino acid with little impact on the biological function of the protein. Hence, a biological pattern can be quite imprecise. The classic measure of pattern support is introduced in the context of precise sequence patterns and cannot serve the purpose of mining imprecise "noisy frequent patterns" in biosequences. To handle such patterns, we need to introduce specific probabilistic models to represent the patterns, e.g., the PWM (position-weight-matrices) representation for transcription factor binding sites [Coh04, DEKM98].

(d) For biological sequences, gaps play an important role in patterns. Some frequent pattern mining problems in biosequences require incorporating the gap and regular expression constraints. How to push these constraints deep inside the mining process should be considered when developing the mining algorithm. Exploiting the prefix antimonotonic property of some constraints would make the mining process more efficient.

■

8.16. BLAST is a typical heuristic alignment method for *pairwise sequence alignment*. It first locates high-scoring short stretches and then extends them to achieve suboptimal alignments. When the sequences to be aligned are really long, BLAST may run quite slowly. Propose and discuss some enhancements to improve the scalability of such a method.

Answer:

MEGABLAST was proposed as a method for improving the performance of BLAST for large sequences. MEGABLAST uses a greedy algorithm for sequence alignment and concatenates many queries to save time spent scanning the database. It is optimized for aligning sequences that differ slightly as a result of sequencing or other similar "errors". It is able to handle much longer DNA sequences than BLAST and can be up to ten times faster than common sequence similarity programs.

Another way to improve the BLAST method is to use hash tables for different word sizes. Recall that *word size* is roughly the minimal length of an identical match that an alignment must contain if it is to be found. The method starts with larger word sizes and tries to find a hit in a sequence database and extend it in both directions. The method finds high quality alignments. It then reduces the word size and tries to find a hit. In this case a hit would be ignored if it is part of the previous round (matching larger words). In this method, we start with word size W and we decrease W until we find a hit in our database. This decreasing can be done using a binary search. When the highest word size with a hit is found, we decrease W and try to find more matching subsequences. We ignore smaller subsequences if they are part of previously found larger subsequences.

Some fast genome alignment algorithms can be found in the following paper: Tamer Kahveci, Vebjorn Ljosa, Ambuj K. Singh: Speeding up whole-genome alignment by indexing frequency vectors. Bioinformatics 20(13): 2122-2134 (2004).

■

8.17. The Viterbi algorithm uses the equality, $argmax_\pi P(\pi|x) = argmax_\pi P(x,\pi)$ in its search for the most probable path, π^*, through a *hidden Markov model* for a given sequence of symbols, x. Prove the equality.

Answer:

We know that $P(\pi|x) = P(x,\pi)/P(x)$, so the value of π that maximizes $P(\pi|x)$ also maximizes $P(x,\pi)/P(x)$. In addition, $P(x)$, the probability of the given variable x, is constant across different values of π. Therefore, $argmax_\pi P(\pi|x) = argmax_\pi (P(x,\pi)/P(x)) = argmax_\pi P(x,\pi)$.

■

8.18. (**Implementation project**) A *dishonest casino* uses a fair die most of the time. However, it switches to a loaded die with a probability of 0.05, and switches back to the fair die with a probability 0.10. The fair die has a probability of $\frac{1}{6}$ of rolling any number. The loaded die has $P(1) = P(2) = P(3) = P(4) = P(5) = 0.10$ and $P(6) = 0.50$.

(a) Draw a hidden Markov model for the dishonest casino problem using two states, Fair (F) and Loaded (L). Show all transition and emission probabilities.

Answer:

The hidden Markov model has two states F and L. The transition probabilities are:

$$P(F|F) = 0.95, P(L|F) = 0.05, P(F|L) = 0.10, P(L|L) = 0.90$$

and the emission probabilities are:

$$P(1|F) = P(2|F) = P(3|F) = P(4|F) = P(5|F) = P(6|F) = \frac{1}{6}$$

$$P(1|L) = P(2|L) = P(3|L) = P(4|L) = P(5|L) = 0.10, P(6|L) = 0.5$$

■

(b) Suppose you pick up a die at random and roll a 6. What is the probability that the die is loaded, that is, find $P(6|D_L)$? What is the probability that it is fair, that is, find $P(6|D_F)$? What is the probability of rolling a 6 from the die you picked up? If you roll a sequence of 666, what is the probability that the die is loaded?

Answer:

We first find the probability of rolling a 6:

$$P(6) = P(6|D_L)P(D_L) + P(6|D_F)P(D_F) = \frac{1}{2}\cdot\frac{1}{2} + \frac{1}{6}\cdot\frac{1}{2} = \frac{1}{3}$$

We then find the posterior probabilities, $P(D_L|6)$ and $P(D_F|6)$, by Bayes' Theorem (Equation 6.10):

$$P(D_L|6) = \frac{P(6|D_L)P(D_L)}{P(6)} = \frac{\frac{1}{2}\cdot\frac{1}{2}}{\frac{1}{3}} = \frac{3}{4}$$

$$P(D_F|6) = 1 - P(D_L|6) = \frac{1}{4}$$

Given the sequence 666, the probability that the die is loaded can be found similarly using Bayes' Theorem:

$$P(D_L|666) = \frac{P(666|D_L)P(D_L)}{P(666|D_L)P(D_L) + P(666|D_F)P(D_F)} = \frac{(\frac{1}{2})^3\cdot\frac{1}{2}}{(\frac{1}{2})^3\cdot\frac{1}{2} + (\frac{1}{6})^3\cdot\frac{1}{2}} = \frac{27}{28}$$

■

(c) Write a program that, given a sequence of rolls (e.g., $x = 5114362366\ldots$), predicts when the fair die was used and when the loaded die was used. (Hint: This is similar to detecting CpG islands and non-CPG islands in a given long sequence.) Use the Viterbi algorithm to get the most probable path through the model. Describe your implementation in report form, showing your code and some examples.

Answer:

We apply the Viterbi algorithm to solve this problem: Let the most probable path be π^*, then $\pi^* = argmax_\pi P(\pi|x)$. The optimization is done by a dynamic programming procedure. The exact recurrence for dynamic programming is shown in Figure 8.12 of the text, and hence is not reproduced here. The result is FFFFFFFFLL for the sequence 5114362366.

Author's note: If the reader has the first printing of the text, please kindly refer to *www.cs.uiuc.edu/∼hanj/bk2/erratаindex.htm* to correct an error regarding Equation (8.15) of the Viterbi Algorithm.

■

Chapter 9

Graph Mining, Social Network Analysis, and Multirelational Data Mining

9.5 Exercises

9.1. Given two predefined sets of graphs, *contrast patterns* are substructures that are frequent in one set but infrequent in the other. Discuss how to mine contrast patterns efficiently in large graph data sets.

Answer:

The search of contrast patterns requires two parameters: the *minimum support* of a substructure in the positive set and the *maximum support* in the negative set. Borgelt and Berthold discussed the mining of contrast substructures using their MoFa algorithm [BB02]. The mining is carried out in a similar way as their frequent substructure mining algorithm, which is pattern-growth based. The pruning is done on the search of frequent substructures in the positive set while the maximum support in the negative set is used to filter out unqualified substructures.

■

9.2. Multidimensional information can be associated with the vertices and edges of each graph. Study how to develop efficient methods for mining *multidimensional graph patterns*.

Answer:

Let's first formalize the problem definition. Given a database of n graphs $DB = \{G_1, G_2, \ldots, G_n\}$, where each vertex or edge has m dimensions (values) associated with it, the goal is to mine all frequent subgraphs, where subgraphs can be in the m-dimensional space or in a lower-dimensional subspace. A graph G' is a k-dimensional subgraph of G if there exists a mapping of $G' = \{u_1, \ldots, u_{|G'|}\}$ to exactly $|G'|$ vertices $\{v_1, v_2, \ldots, v_{|G'|}\}$ in G and also at least k dimensions, so that u_i and v_i agree on all k dimensions.

It is clear that the gSpan algorithm cannot be directly applied to this problem because an m-dimensional graph will have numerous combinations of DFS code.

The key to solving the problem is to first view the m-dimensions individually. We can divide the whole database DB into m databases, $\{DB_1, \ldots, DB_m\}$, where the i^{th} DB contains the information for the i^{th} dimension. For each reduced database, gSpan can be applied to obtain all of the global frequent 1-dimensional graphs.

Based on the Apriori property, all subgraphs of a frequent k-dimensional graph must also be frequent. Therefore, the 1-dimensional frequent subgraph patterns can form a candidate set of 2-dimensional frequent

patterns by performing joins between pairs of isomorphic graphs. The whole database is then scanned to update the count information for the candidates.

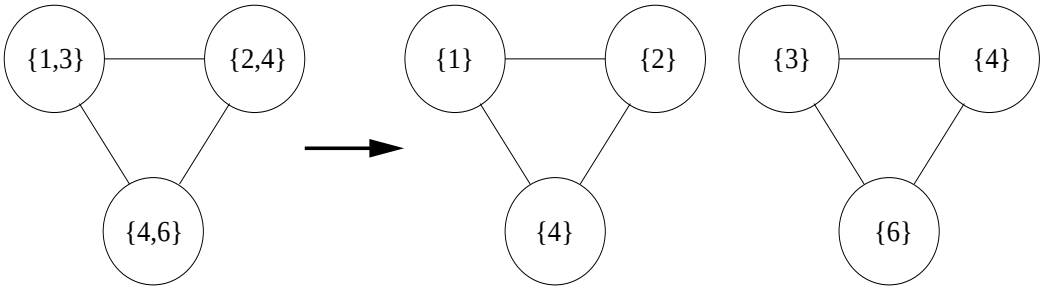

Figure 9.1: Mining multidimensional graph patterns (Exercise 9.2).

This property is straightforward as seen in Figure 9.1. If the 2-dimensional graph on the left-hand side is frequent, then the two 1-dimensional graphs on the right-hand side must also be frequent. Thus, an Apriori-like algorithm can efficiently mine multidimensional frequent graph patterns.

■

9.3. *Constraints* often play an important role in efficient graph mining. There are many potential constraints based on users' requests in graph mining. For example, one may want graph patterns containing or excluding certain vertices (or edges), with minimal or maximal size, containing certain subgraphs, with certain summation values, and so on. Based on how a constraint behaves in graph mining, give a systematic classification of constraints and work out the rules on how to maximally use such constraints in efficient graph mining.

Answer:

When mining frequent patterns, the user can specify several classes of constraints [NLHP98, PHL01]. Instead of just filtering the resulting patterns, these can be used to prune the search space while mining. We assume that the general algorithm grows closed frequent subpatterns using some type of ordering (DFS or right-most extension) to avoid repeatedly exploring part of the search space. Constraint types include the following:

(a) **Element Constraint.** Such constraints specify certain nodes or edges that either must or cannot be in the resulting patterns. If some elements must be in the pattern, we can start with those nodes/edges and grow. If certain nodes/edges cannot be in the pattern, we can exclude them from the graph during mining. This kind of constraint can be generalized to subgraphs as well.

(b) **Size Constraint.** The size constraint controls the number of nodes (or edges) of resulting patterns. When growing patterns, the algorithm will not report the patterns under a minimum threshold (monotonic), and stops growing when it reaches a maximum threshold (anti-monotonic).

(c) **Aggregate Constraint.** These are constraints on the aggregates of attributes, like weights of nodes or edges. The algorithm for this depends on the aggregate constraint. However, if we assume that the aggregate grows monotonically with the size of the pattern (like the sum of link distance), and we have a range of acceptable values, then we can treat this much like a size constraint. That is, do not report graphs where the aggregate value is below some lower threshold and stop growing when it exceeds the upper threshold. On the other hand, if the constraint is anti-monotonic, or something that needs to be checked at every node/vertex (like minimum degree), we can check to prune when we grow, as with element constraints.

Recent studies on constraint-based graph mining include [YZH05].

■

9.4. Our discussion of frequent graph pattern mining was confined to graph transactions (i.e., considering each graph in a graph database as a single "transaction" in a transactional database). In many applications, one

needs to mine frequent subgraphs in a *large single graph* (such as theWeb or large social network). Study how to develop efficient methods for mining frequent and closed graph patterns in such data sets.

Answer:

There are two ways to count subgraphs in a large single graph. The first considers two subgraphs to be different if they differ by at least by one edge. The second way requires the two graphs to be completely edge disjoint. The resulting frequency of the second method follows the downward closure property, which is discussed by most frequent pattern discovery algorithms. gSpan can be modified for this purpose as follows. Build and right-most extend the DFS tree in parallel by taking each node as the root. When an edge is added, check if the disjoint-property is violated. To optimize non-overlapping, the backward edges can be added first, and forward edges can be added more conservatively using a stack to keep track of the delayed extensions. An anchor-based technique similar to that used in the HSIGRAM-EMBED algorithm by Kuramochi and Karypis [KK04] for discovering frequent patterns in large graph can be applied further during the forward extension. Starting from the subgraph with the lowest frequency, for each new forward edge to be added (anchor edge a), check if the shortest path between a and any edge in the existing subgraph is within the diameter of the subgraph. If that is the case, this edge is naturally closer to the subgraph and will be added first. Notice that the above optimization will reorder or delay some forward extensions in order to reduce the overlapped pruning, but these extensions will not be ignored and will still be attempted.

■

9.5. What are the challenges for *classification in a large social network* in comparison with classification in a single data relation? Suppose each node in a network represents a paper, associated with certain properties, such as author, research topic, and so on, and each directed edge from node A to node B represents that paper A cites paper B. Design an effective classification scheme that may effectively build a model for highly regarded papers on a particular topic.

Answer:

Challenges: Large social networks exhibit different characteristics than single data relations and therefore introduce different challenges in classification tasks. First, large social networks are not static. As a result, features in classifiers have to be dynamically updated to handle this challenge. Second, the nodes in large social networks are linked together. These links contain semantics that classifiers need to analyze and use. Finally, nodes in large social networks are heterogeneous and noisy (they may belong to different communities for different purposes) and therefore, not all are relevant as training data for classifiers.

Classification Scheme: In traditional classification methods, which are designed for single data relations, objects are classified based on the attributes that describe them. However, when classifying in a large social network, we have to consider the link structure of the network. Link-based classification predicts the category of an object based not only on its attributes, but also on its links, and on the attributes of linked objects. Take Web page classification as an example. It predicts the category of a Web page based on word occurrence and anchor text, both of which serve as attributes. In addition, classification is based on links between pages and other attributes of the pages and links.

This gives us insight into classifying citation data. Classification in social networks include two schemes: classifying objects (nodes) and classifying links (edges). In the citation domain, classifying highly regarded papers is considered as the classification of objects. Specifically, a paper (i.e., node) has its own attributes like author, topic, keywords, abstract, and full text. In addition, it is linked to other papers by citations. To model the highly regarded papers on a particular topic, we can first select all of the papers on this topic, which forms a subgraph of the entire network. We can then apply any object ranking/classification method for general social networks (such as PageRank, HITS, etc.) on this subnetwork to obtain the highly regarded papers. Note that if training data are not available, we can rely on PageRank and HITS. If we do have training data, the classifier we build can also accommodate object attributes such as contents and authors.

■

9.6. A group of students are linked to each other in a social network via advisors, courses, research groups, and friendship relationships. Present a *clustering* method that may partition students into different groups according to their research interests.

Answer:

For this purpose we use group detection, which is a clustering task. It predicts when a set of objects belong to the same group or cluster, based on their attributes as well as their link structure. This is also known as community mining. It can be thought of as subgraph identification. Most algorithms for community mining assume that there is only one kind of relation. In this exercise, there exist multiple, heterogeneous social networks, representing various relationships.

Each kind of relation may play a distinct role in a particular task. We first need to identify which relation plays an important role. Such a relation might not exist explicitly, that is, we may need to first discover such a hidden relation before finding the community.

The algorithm for relation extraction, which models the task as an optimization problem, is as follows: Given are a set of objects and a set of relations, which can be represented by a set of graphs $G_i(V; E_i), i = 1, \ldots, n$, where n is the number of relations, V is the set of nodes (objects), and E_i is the set of edges with respect to the ith relation. Suppose a hidden relation is represented by a graph $G'(V; E')$. A user specifies her information need as a query in the form of a set of labeled objects $X = [x_1; \ldots; x_m]$ and $Y = [y_1; \ldots; y_m]$, where y_i is the label of x_i (such labeled objects indicate partial information of the hidden relation G'). The algorithm aims to find a linear combination of the weight matrices of initial graphs that can best approximate G'.

After modeling the given problem with graphs, the following was observed: (1) the research group relation plays an important role; (2) people with the same adviser have the same research interest; and (3) the friendship relation is not that important. Therefore, the distance between two objects can be defined by the distance between research groups or advisor groups. Clustering can then be performed easily with this distance metric.

■

9.7. Many diseases spread via people's physical contacts in public places such as offices, classrooms, buses, shopping centers, hotels, and restaurants. Suppose a database registers the concrete movement of many people (e.g., location, time, duration, and activity). Design a method that can be used to rank the "not visited" places during a virus-spreading season.

Answer:

We model the problem in the following manner: Assume that each place that the database keeps track of is a node. Each time a person moves from one place to another place (as the database registers this movement), we create a link between the two places (nodes). We also label the edges with the time of the movement (when the link was created). In other words, we are given a social network as input (with different places as nodes, and travel between those places as links). In this social network, each link has a weight, which is related to how recently it was created. For example, if X has moved from room A to room B one day ago, and Y has moved from A to B 10 days ago, we will have two links between A and B, where the first will have substantially more weight than the second. The motivation is that the more recent a link is, the more probable it is that it will be used in spreading the viruses and diseases. Given affected people, we can find the start seed of the affected places and mark them. To find the not-visited places during the virus-spreading season, the problem boils down to finding communities of not-visited places. That is, we need to find communities and rank them (using a clustering algorithm, for example). When ranking, the weight of the links plays an important role. We can exploit their "freshness" to rank and mine the communities. Different algorithms can be used for this purpose, such as follows:

(a) Use the algorithm for community mining from multi-relational networks as described in Section 9.2.4. Note, however, that this method does not exploit link analysis and models the problem as an optimization problem.

(b) Use a link-based hierarchical clustering method. To do so, we can define the similarity of two objects based on the nodes that link them together. This can be defined in a recursive manner. Any similarity scheme can be applied but should (1) exploit the link-based nature of the problem and (2) exploit the link weights. We can then apply a hierarchical clustering method on top of this model.

■

9.8. Design an effective method that discovers *hierarchical clusters in a social network*, such as a hierarchical network of friends.

Answer:

The essence of social networks is their link structure. The clustering of social network should use structural equivalence in measuring the similarity between clusters. Two vertices are said to be structurally equivalent if they have the same set of neighbors. Similarity measures the degree of equivalence. The similarity between vertex a and b is

$$X(a,b) = \sqrt{\sum_{c \neq a,b} (A_{ac} - A_{bc})^2},$$

where A_{ac} and A_{bc} are the elements of the adjacency matrix for vertices a and b, respectively. The adjacency matrix for a (or b) represents the connection status between a (or b) and any other vertex, c (e.g., 1 for connected, 0 for no connection). Notice that a and b can be clustered even if they are not connected to each other. Once the similarity has been calculated, a standard hierarchical clustering algorithm such as k-medoids can be applied, and the vertices and edges are added gradually in the order of decreasing similarity.

■

9.9. Social networks evolve with time. Suppose the history of a social network is kept. Design a method that may discover the *trend of evolution* of the network.

Answer:

The *Forest Fire model*, proposed by Leskovec, Kleinberg, and Faloutsos [LKF05], can be used to discover the trend of evolution of networks. Research has shown that the evolution of social networks usually involves three common phenomena: *densification power law*, *shrinking diameter*, and *heavy-tailed out-degree and in-degree distributions*. The Forest Fire model captures these characteristics of graph evolution over time. It is based on the notion that new nodes attach to the network by "burning" through existing edges in epidemic fashion. It uses two parameters: a forward burning probability, p, and a backward burning ratio, r. When a new node, v, arrives at time t, it attaches to the graph constructed so far as summarized in Section 9.2.2. (Thus, we do not repeat the details here.)

Several earlier models of network evolution were based on static graphs, identifying network characteristics from a single or small number of snapshots, with little emphasis on finding trends over time. The Forest Fire model combines the essence of several earlier models, while considering the evolution of networks over time. The success of this model explains the common characteristics of network evolution.

People may be interested in particular evolution trends, such as the change of communities, the change of density of links for specific nodes (or group of nodes), and "actively changing" nodes. These can be investigated by extending existing models that compare historical snapshots. Taking community evolution for example, we can borrow the clustering methods in data stream mining [AHWY03]. First, we store snapshots of the network in tilted time frames (Section 8.1.2). For each snapshot, microclustering is performed off-line, keeping a fixed number of "micro-communities". We can analyze the evolution of communities by comparing two snapshots. Basically, we want to determine (1) which communities no longer exist; (2) which communities are new; and (3) which communities have merged.

■

9.10. There often exist *multiple social networks* linking a group of objects. For example, a student could be in a class, a research project group, a family, a neighborhood, and so on. It is often beneficial to consider their joint effects or interactions. Design an efficient method in social network analysis that may incorporate multiple social networks in data mining.

Answer:

Typical work on social network analysis includes the discovery of groups of objects that share similar properties. This is known as community mining. Most algorithms for community mining assume that there is only one social network. In reality, there exist multiple, heterogeneous social networks. The challenge is the mining of hidden communities on such heterogeneous social networks. An example is when you want to find a group of students who have the same research interest (e.g., they are interested in data mining). A student could be in a data mining class, a research project group related to data mining, and many other social networks that reflect his interest in data mining.

There are always various kinds of relationship between the objects because an object may be in different social networks. The different relation graphs can provide us with different communities. To find a community with certain properties, we first need to identify which relation plays an important role in such a community. Such a relation might not exist explicitly, that is, we may need to first discover such a hidden relation before finding the community on such a relation network.

As in the previous exercise, we try to extract relevant relations. An algorithm for relation extraction is as follows. It models the task as an optimization problem. Given are a set of objects and a set of relations, which can be represented by a set of graphs $G_i(V; E_i), i = 1, \ldots, n$, where n is the number of relations, V is the set of nodes (objects), and E_i is the set of edges with respect to the i-th relation. The weights on the edges can be naturally defined according to the relation strength of two objects. The algorithm characterizes each relation by a graph with a weight matrix. Suppose a hidden relation is represented by a graph $G'(V; E')$, and M' denotes the weight matrix associated with G'. A user specifies her information need as a query in the form of a set of labeled objects. The algorithm aims at finding a linear combination of these weight matrices that can best approximate G' (the weight matrix associated with the labeled examples.) The obtained combination is more likely to meet the user's information need, so it leads to better performance in community mining. Therefore, given a new object, we can label it either as a member of a hidden group or not. In other words, if it is strongly connected (that is, with high weight) to the objects in the group, then it is considered to be in that group.

■

9.11. Outline an efficient method that may find strong *correlation rules* in a large, multirelational database.

Answer:

We suggest a method based on probabilistic relational models (PRM), as described by Getoor, Friedman, Koller, and Taskar [GFKT01], which are an extension of Bayesian networks to relational domains. A PRM describes a template for a probability distribution over a database. The template includes a relational component (which describes the relational schema for the domain) and a probabilistic component (which describes the probabilistic dependencies that hold in the domain). A PRM, together with a particular universe of objects, defines a probability distribution over the attributes of the objects and the relations that hold between them. The relational component describes entities in the model, attributes of each entity, and references from one entity to another. The probabilistic component describes dependencies among attributes, both within the same entity and between attributes in related entities. An edge between attributes represents a probabilistic dependence of one attribute on another.

■

9.12. It is important to take a user's advice to cluster objects across multiple relations, because many features among these relations could be relevant to the objects. A user may select a sample set of objects and claim some should be in the same cluster but some cannot. Outline an effective clustering method with such *user guidance*.

Answer:

A method similar to CrossMine (for cross-relational clustering) [YHYY04] could be adopted here. The basic idea is to search for pertinent features from the multiple relations, which may be joined through the "primary keys" or "foreign keys". A *pertinent feature* is one that satisfies user's constraints. In this exercise, the user specifies some examples that should occur in the same cluster and others that should not. If a feature is pertinent then, based on this feature, the examples that are required to be in the same cluster may have high similarity scores, whereas the examples prohibited their occurring in the same cluster may

have low similarity scores. Similar heuristics to those in [YHYY04] can be applied to confine the search for pertinent features to the most promising directions. Techniques such as tuple ID propagation can also be used to improve the efficiency of the algorithm. After finding a pertinent feature, a standard clustering algorithm such as k-medoids can be used to cluster the target tuples.

This method has been summarized in a recent paper as CrossClus algorithm by Yin, Han and Yu [YHY05].

■

9.13. As a result of the close relationships among multiple departments or enterprises, it is necessary to perform data mining across multiple but interlinked databases. In comparison with multirelational data mining, one major difficulty with mining across multiple databases is *semantic heterogeneity* across databases. For example, "William Nelson" in one database could be "Bill Nelson" or "B. Nelson" in another one. Design a data mining method that may consolidate such objects by exploring object linkages among multiple databases.

Answer:

In object reconciliation, the task is to predict whether two objects are, in fact, the same, based on their attributes and links. This task is common in information extraction, duplication elimination (deduplication), object consolidation, and citation matching, and is also known as *record linkage* or *identity uncertainty*.

An approach based on relationships, which extends work by Chen, Kalashnikov, and Mehrotra [CKM05], is proposed here. In contrast to traditional domain-independent data cleaning techniques, this approach analyzes not only object features, but also additional semantic information for the purpose of object consolidation. This semantic information is in the form of *inter-objects relationships*. For instance, suppose that "William Nelson" refers to an author of a particular publication. This publication might has other authors, which can be linked to their affiliated organizations, and so on, forming a web of entities that are inter-connected via relationships. The knowledge of relationships can be exploited alongside attribute-based similarity, resulting in improved accuracy of object consolidation.

Most of the traditional domain-independent data cleaning techniques belong to the class of *feature-based similarity* (FBS) methods. To determine if two objects/records are the same, FBS methods employ a similarity function that compares values of object/record attributes (features) for the purpose of deduplication. The values of the attributes of an object are typically derived from the object representation and the context in which it is used. However, this approach views the underlying database as an attributed relational graph (ARG), where nodes correspond to object representations and edges correspond to relationships. It first uses FBS to determine if two representations can refer to the same objects. If, based on the FBS, two representations may refer to the same object, then the relationships between those representations are analyzed to measure the connection strength between them. Graph partitioning techniques are then employed to consolidate the representations of objects based on the FBS similarity and connection strength.

■

9.14. Outline an effective method that may perform *classification* across multiple heterogeneous databases.

Answer:

The aim is to have a method that performs classification across multiple heterogeneous databases.

One approach is to use data integration to overcome the heterogeneity problem. Although the integration of heterogeneous data sources is challenging, it is not required to completely resolve semantic heterogeneity before effective classification can be performed. For example, one may identify crucial information (or attributes) whose semantic heterogeneity could seriously affect the classification accuracy, and perform in-depth processing to resolve such a heterogeneity problem first. Thus limited progress can be achieved using this approach to help effective classification.

Another approach for such classification is to use regression to predict the usefulness of inter-database links that serve as bridges for information transfer. Like classification in multirelational databases, we have a target relation whose tuples are target tuples. The goal of database classification is to build an accurate classifier for predicting the class labels of target tuples. Transferring information across databases can be difficult; it is often expensive to transfer information between two databases that may be far from each other

physically. Thus, we must be able to build accurate cross-database classifiers with as low inter-database communication cost as possible. Some links can lead to useful features, while others may be useless and only further burden the classification. The usefulness of a link is defined as the maximum information gain of any feature generated by propagating information through this link.

In order to avoid transferring information between databases we can employ *tuple ID propagation* as it is used for multirelational classification in Yin et al. [YHYY04].

■

Chapter 10

Mining Object, Spatial, Multimedia, Text, and Web Data

10.7 Exercises

10.1. An *object cube* can be constructed by generalization of an object-oriented or object-relational database into relatively structured data before performing multidimensional analysis. Because a set of complex data objects or properties can be generalized in multiple directions and thus derive multiple generalized features, such generalization may lead to a high-dimensional, but rather sparse (generalized)"object cube." Discuss how to perform effective online analytical processing in such an object cube.

Answer:

Materializing the high dimensions would be too expensive, even with an iceberg cube that prunes infrequent items. One solution is to compute shell fragments, or inverted indices on some dimensions as in Li, Han, and Gonzalez [LHG04]. A fragment for multiple dimensions in a set S can be created from smaller fragments of set A and $S - A$, where $(A \subset S)$ by joining the keys in each table and setting the new value to the intersection of the tuple IDs. In the worse case, $|fragment(S)| = |fragment(S - A)| * |fragment(A)|$.

The original inverted lists (one per dimension) can generate any possible combination of dimensions, allowing drilling and filtering on any dimension in constant time using the intermediate fragments. This also allows fast computation of count, sum, or other aggregate values across any dimension. In addition, popular dimension combinations and some 3-D combinations can be pre-materialized for quick response time.

This method works if the cardinality of the attributes is not too high but is susceptible to high cardinality of the attributes.

■

10.2. A *heterogeneous database system* consists of multiple database systems that are defined independently, but that need to exchange and transform information among themselves and answer local and global queries. Discuss how to process a descriptive mining query in such a system using a generalization-based approach.

Answer:

Database relations often contain concept hierarchies for descriptive attributes and different ranges of values for numeric attributes. As in Han, Ng, Fu, and Dao [HNFD98], suppose that we are given the relation $house(house\_id, address, construction\_date, living\_room(length, width), bed\_room(length, width)\ldots)$. The generalization can be performed as follows. The attribute *construction_date* can be transformed to *construction_year*. The length and width for each room can be abstracted using the measure of square foot. Other useful generalizations include removing uninteresting attributes, converting continuous values to discrete ones, aggregation, approximation and other techniques described in Chapter 2. The generalization is

effective in reducing the gap between heterogeneous database relations due to different schema definitions. In other words, the generalization wraps the detailed differences and brings the comparison of heterogeneous data to the same level. As generalization can have multiple layers, the wrapper can be divided into several different levels.

A query can be answered directly from the facts in relations, or intelligently by providing some generalized or comparative answers. The query may involve concepts of different layers. Relations with many attributes are likely to have a large combination of concepts levels. Materialization of all of them is not practical, therefore the query might need some forms of transformation in order to land on the right concept level and obtain the results efficiently. The right level must have a view of the attribute that is consistent with the query. For example, "greater Vancouver area" can be mapped to all of its composite regions. Each concept layer may contain summarized or statistical data to facilitate the queries requiring generalization. In practice, materialized concept layer represents the frequently used patterns. Finally, if the queries cannot be answered at the higher concepts levels, the concept generalizations can still provide a progressive query refinement, from general information browsing to specific data retrieval. Moreover, the generalization can speed up queries that require neighborhood comparisons of other associated information.

■

10.3. A *plan database* consists of a set of action sequences, such as legs of connecting flights, which can be generalized to find generalized sequence plans. Similarly, a *structure database* may consist of a set of structures, such as trees or graphs, which may also be generalized to find generalized structures. Outline a scalable method that may effectively preform such generalized structure mining.

Answer:

We first discuss how structures like trees or graphs can be generalized. A tree or graph may be *directed, labeled*, and *connected*. Possible generalizations with respect to these properties are as follows:

(a) Generalize on directed edges to undirected edges;

(b) Generalize the labeled structures to unlabeled structures so that only the topology of the tree or the graph counts;

(c) Generalize the connectivity of the tree or graph to the minimum (or maximum) degree of the tree or graph, so that only density or sparsity is considered.

There are, of course, more types of generalizations, some of which can be very computationally challenging. Our discussion is confined to the above forms of generalization.

The first two forms of generalizations (i.e., on directed edges and labels) are already handled by many existing graph mining algorithms. In fact, the directed or the labeled graphs are special cases and can be integrated into the mining processing as constraints. We discuss the method to efficiently mine the third form generalization. Since a tree is a special graph, we only need to discuss algorithms that operate on graph structures. Suppose that the graphs are labeled (i.e., we only generalize on the connectivity). Mining connected labeled subgraphs is an important problem in biological data mining.

For each graph G, we can always build a complimentary graph G', such that for every two node p, q, if p, q are connected (not connected) in G, then they are not connected (connected) in G'. As the result, the minimum degree constraint of a subgraph in G is equivalent to the maximum degree constraint of a subgraph in G'. We therefore discuss how to handle maximum degree constraints.

Assume that we have a minimum degree as well as minimum support as thresholds. We can first preprocess the graph data by: (1) removing all nodes whose degree is less than the minimum degree; and (2) removing all nodes whose frequency is less than the minimum support. These two steps are performed until no further removal can be conducted.

Take [YZH05] as an example of mining connected subgraphs by *pattern-based enumeration*. There are some application scenarios where the number of graphs is quite limited and each graph is considerably large. An alternative approach using *support set-based enumeration* can be used, which handles such cases. This method is similar to the row-based enumeration approach recently introduced in frequent pattern mining [PCT+03].

The algorithm starts from the empty data set. It adds new graph data to the working set in a depth-first manner. As soon as the number of candidate patterns supported by the current graph data set is zero, the algorithm stops including more graph data. When the current working set has only one graph, then this graph is a valid candidate pattern because the pre-processing procedure ensures that all of its nodes satisfy the minimum degree constraint. When there are two graph data in the working set, we call a recursive procedure *top-down decomposition*. The two graphs are compared and those nodes that are not shared by both graphs are removed. As the result, both graphs will be reduced and we can apply the minimum degree constraint to further remove nodes. This may cause the graphs to decompose into several smaller graphs, losing connectivity. We recursively invoke the *top-down decomposition* procedure on each pair of small graphs. Note that graphs that come from the same original graph do not need further computation.

The above procedure can be easily extended to more than two graphs. It automatically outputs closed substructures.

■

10.4. Suppose that a city transportation department would like to perform *data analysis* on *highway traffic* for the planning of highway construction based on the city traffic data collected at different hours every day.

(a) Design a spatial data warehouse that stores the highway traffic information so that people can easily see the average and peak time traffic flow by highway, by time of day, and by weekdays, and the traffic situation when a major accident occurs.

Answer:

A star schema for the highway traffic warehouse consists of four dimensions: *region* (highway), *time*, *traffic*, and *accident*, and three measures: *region_map* (spatial), *traffic_flow*, and *count*. A concept hierarchy for each dimension can be created by users or experts, or generated automatically by data clustering analysis. For example, a possible concept hierarchy for *time* is *time* $<$ *day* $<$ *month* $<$ *season*. For *region*, we may have *highway* $<$ *part_of_city* $<$ *city* $<$ *district*. A concept hierarchy for *traffic* is *traffice* $<$ *traffic_range* $<$ *traffic_description*, and for *accident*, *accident_ID* $<$ *accident_range* $<$ *accident_description*.

Of the three measures, *traffic_flow* and *count* are numerical measures, which can be computed in the same manner as for nonspatial data cubes. (We can compute the average for them.) The *region_map* measure is spatial and represents a collection of spatial pointers to the corresponding regions. Because different spatial operations result in different collections of spatial objects in a region map, it is a challenge to flexibly and dynamically compute the merges of a large number of regions. For example, two different roll-ups produce two different generalized region maps, each the result of merging a large number of small regions. With the highway traffic warehouse, people can easily see the average and peak time traffic flow by highway, by time of day, and by weekdays, and the traffic situation when a major accident occurs.

■

(b) What information can we mine from such a spatial data warehouse to help city planners?

Answer:

Different kinds of useful information for city planners can be mined from this data warehouse. For example, by aggregating on *region_map*, we can find which highways have high traffic flow. This can aid in decisions regarding where to build additional roads. We can also perform mining to learn what hours of the day the traffic is at its maximum, and which highways have the most accidents. Such information is useful for planning new highways or new regulations in a city.

■

(c) This data warehouse contains both spatial and temporal data. Propose one mining technique that can efficiently mine interesting patterns from such a spatiotemporal data warehouse.

Answer:

Spatiotemporal trend analysis is an efficient technique for mining from such data warehouses. It deals with the detection of changes and trends by time along a spatial dimension. For example, it may

find a trend in traffic change of a highway (as in time series analysis). However, by computing the trend for different locations, we can find patterns such as "The maximum traffic during weekdays is downtown, but on weekends the maximum traffic is around shopping centers." By considering both time and space, we can see the movement of high traffic in a city on specific days. With this method, we find the trend for both the time dimension and the space dimension and extract useful patterns, such as, for example, the maximum traffic moves from one location or highway to another at certain hours of the day.

■

10.5. *Spatial association mining* can be implemented in at least two days: (1) dynamic computation of spatial association relationships among different spatial objects, based on the mining query, and (2) precomputation of spatial distances between spatial objects, where the association mining is based on such precomputed results. Discuss (1) how to implement each approach efficiently and (2) which approach is preferable under what situation.

Answer:

(a) *Dynamic computation of spatial association relationships among different spatial objects, based on the mining query*: This is most useful when the association mining is user-centric and not all of the domain needs to be exploited. This is more suitable for scenarios that are more interactive and where the user has domain knowledge that can be specified in his queries, or where the user can specify objects of interest.

(b) *Precomputation of spatial distances between spatial objects, where the association mining is based on such precomputed results*: This approach is most useful when we have background knowledge of the spatial domain, so that we can systematically explore the hierarchical structure of task-relevant geographic layers and deal with numerical nonspatial properties of spatial objects.

Implementation: The first method was tackled by Koperski and Han using an AV (attribute-value) approach) [KH95]. The second method appears in Malerba and Lisi [ML01], which uses an inductive logic programming (ILP) method for spatial association rule mining. It can be considered the first-order logic counterpart of Koperski and Han's method, inspired by work on mining association rules from multiple relations. The method benefits from the available background knowledge of the spatial domain, systematically explores the hierarchical structure of task-relevant geographic layers, and deals with numerical nonspatial properties of spatial objects. It was implemented in an ILP system called SPADA (Spatial Pattern Discovery Algorithm), which operates on a deductive database that is set up by an initial step of feature extraction from a spatial database.

■

10.6. Traffic situations are often auto-correlated: the congestion at one highway intersection may trigger the congestion in nearby highway segments after a short period of time. Suppose we are given highway traffic history data in Chicago, including road construction segment, traffic speed associated with highway segment, direction, time, and so on. Moreover, we are given weather conditions by weather bureau in Chicago. Design a data mining method to find high-quality *spatiotemporal association rules* that may guide us to predict what could be the expected traffic situation at a given highway location.

Answer:

We basically model the traffic history as a transactional database. For each time point of one location, we create a "transaction", which contains the weather and traffic conditions, and the conditions of its neighboring locations after a short delay. After this transformation, the entire history is modeled as a database. We can apply standard association rule mining algorithms to find the spatiotemporal association rules, which will allow us to predict the expected traffic situation.

■

10.7. *Similarity search in multimedia* has been a major theme in developing multimedia data retrieval systems. However, many *multimedia data mining* methods are based on the analysis of isolated simple multimedia features, such as color, shape, description, keywords, and so on.

(a) Can you show that an integration of similarity-based search with data mining may bring important progress in multimedia data mining? You may take any one mining task as an example, such as multidimensional analysis, classification, association, or clustering.

Answer:

Suppose that we take clustering as the example. The most important criteria in clustering is the similarity measure between objects to be clustered. The traditional features in multimedia clustering are isolated simple characteristics such as *color*, *shape*, *description*, *keyword*, and so on. The color and shape are very coarse at catching the real similarity between multimedia data (e.g., images), and the description and keywords are sometimes difficult to interpret. This is because these text data are input by humans and naturally contain noise, typos, and inconsistencies. Some of the description may not even relate to the images.

There are several ways that similarity search may improve the clustering task. For example, we can combine the similarity distance and the basic feature distance and use the combined distance as the similarity measure in the clustering.

■

(b) Outline an implementation technique that applies a similarity-based search method to enhance the quality of clustering in multimedia data.

Answer:

Generally, similarity search is very close to clustering, because both rely on similarity measures between multimedia objects. It really depends on what kind of features are used in clustering. If the clustering method already adopts the same distance criteria as in the similarity search, then there is no difference between them. Here, we assume that clustering uses the isolated simple features and similarity search uses a distance measure for high-dimensional data. In this scenario, we can use similarity search to improve the performance of clustering and vice versa. The principal problem is how to integrate these two measure criteria to obtain a better one.

Here we consider one possible scenario where we have a description and keywords for each image data and need to find features for clustering. One option is to use each term in the description as a feature and employ a vector-space model to measure the distance between images. However, this approach does not provide a better interpretation of the features. Alternatively, we can use the similarity measure used in the search to build a neighborhood graph. We mine frequent patterns (terms or the combination of terms) from those description words. Furthermore, we check the support set of each pattern. If a pattern is supported by a set of images that form a dense subgraph in the neighborhood graph, it is potentially a good feature for clustering. Otherwise, it may not be as good and can be dropped.

There are many ways to reduce the number of patterns. For example, techniques that use compressed patterns to shrink the number of patterns can be used. This helps control the number of features as well.

■

10.8. It is challenging but important to *discover unusual events from video data* in real time or in a very short time frame. An example is the detection of an explosion near a bus stop or a car collision at a highway junction. Outline a *video data mining* method that can be used for this purpose.

Answer:

It is unrealistic to treat a video clip as a long sequence of individual still pictures and analyze each picture. This would result in too many pictures, and most adjacent images could be rather similar. In order to capture outliers from video data, it is better to first treat each video clip as a collection of actions and events in time and temporarily segment them into video shots. A *shot* is a group of frames or pictures, where the video content from one frame to adjacent ones does not change abruptly. Moreover, the most representative frame in a video shot is considered the *key frame* of the shot. Each key frame can be analyzed using the image feature extraction and analysis methods studied in the content-based image retrieval. A process can then sample the video data very quickly and analyze the static images sampled in a real-time

fashion. The process compares the features of the key frame with the current static image to see if there is a major difference. If so, it sounds an alarm that there is an unusual event happening and saves the current video clip for further analysis and processing.

■

10.9. *Precision* and *recall* are two essential quality measures of an information retrieval system.

(a) Explain why it is the usual practice to trade one measure for the other. Explain why the F-score is a good measure for this purpose.

Answer:

Precision is an evaluation measure that represents the percentage of documents returned by the information retrieval (IR) system that are actually relevant. It indicates the *quality* of the results. *Recall* represents the percentage of relevant documents that were captured by the IR system. It indicates the *coverage* of the system. Based on these measures, we can build a system that, given an IR query, can retrieve many documents with probably a large number of false positives (irrelevant ones); or we can tune the system to return only relevant documents but that misses out on others that are also relevant (that is, there may be many false negatives). The two measures naturally tune the system in opposite directions (i.e., quality versus coverage). However, we want to build a system that has reasonable quality as well as good coverage. The F-score acts as a compromise even though it ignores that one might be more important than the other. It is a good measure because it is the harmonic mean of the two competing measures, that is, $2 * recall * precision/(recall + precision)$.

■

(b) Illustrate the methods that may effectively improve the F-score in an information retrieval system.

Answer:

One method, which we consider cheating, would be to tweak the sensitivity of the system to maximize the value of F-score $= 2 * TP/(2 * TP + FP + FN)$, where T/F = true/false and P/N = positive/negative. This depends on what the system's precision-versus-accuracy curve looks like.

Another method is to incorporate user feedback for the mining results into the ranking process. This can be something as simple as using window-like weights. A third approach is to use two orthogonal methods of determining if a result is valid. This is especially useful if one method has high recall and the other has high precision, because we can apply the high-recall method first, and then apply the high-precision method. An example would be finding similar sites on the Web by first using link structure and then the content.

■

10.10. *TF-IDF* has been used as an effective measure in document classification.

(a) Given one example to show that TF-IDF may not be always a good measure in document classification.

Answer:

TF-IDF assigns large weight to rare words and such rare words may lead to ineffective classification. For example, suppose that there are two terms, t_1 and t_2, in two classes, C_1 and C_2, each having 100 training documents. Term t_1 occurs in five documents in each class (i.e., 5% of the overall corpus), but t_2 occurs in 20 documents in class C_1 only (i.e., 10% of the overall corpus). Term t_1 will have a higher TF-IDF value because it is rarer, but it is obvious that t_2 has stronger discriminative power in this case.

■

(b) Define another measure that may overcome this difficulty.

Answer:

A feature selection process can be used to remove terms in the training documents that are statistically uncorrelated with the class labels. This will reduce the set of terms to be used in classification, thus improving both efficiency and accuracy.

■

10.11. An *e-mail database* is a database that stores a large number of electronic mail (e-mail) messages. It can be viewed as a semistructured database consisting mainly of text data. Discuss the following.

(a) How can such an e-mail database be structured so as to facilitate multidimensional search, such as by sender, by receiver, by subject, and by time?

Answer:

If we consider the metadata of an e-mail, most attributes of an e-mail can be structured as data attributes in a relational database, with an added attribute for its unstructured content. The structured attributes can include: *sender*, *receiver*, *time*, *subject*, *thread_ID*, etc. We can thus divide an e-mail into a structured part and an unstructured part. We then store the structured part in a data warehouse and the (unstructured) contents in a text or XML database, or a text index. Each transaction in the structured data warehouse also stores a link (either an URL or an ID in the text database) to the contents. In this way, we can support multidimensional OLTP and OLAP in this structured data warehouse by traditional database operations and data cubing. When the relevant records are returned, we just follow the links to obtain the contents from the database/index that stores the full text.

■

(b) What can be mined from such an e-mail database?

Answer:

There are plenty of interesting things to mine from e-mail databases. Examples include the following.

- Computing behavior profiles or models of user e-mail accounts.
- Automated summarization of e-mail archives.
- Detecting e-mail security violations.
- Detecting groups of users who frequently pairwise exchange messages.
- Searching for *User Clique*, the set of accounts a particular user typically e-mails as a group.
- Mining frequent patterns in e-mail spam.
- Clustering e-mails to detect threads.

■

(c) Suppose you have roughly classified a set of your previous e-mail messages as *junk*, *unimportant*, *normal*, or *important*. Describe how a data mining system may take this as the training set to automatically classify new e-mail messages or unclassified ones.

Answer:

This can be regarded as a text classification problem or semi-supervised learning problem. We analyze both cases. In the first case, the data mining system will extract features from the labeled e-mail messages and train a classifier using the features. The features can contain not only content features (e.g., key words, part of speech tags, etc.) and textual patterns (e.g., frequent sequential patterns, phrases), but also features from other attributes, such as authors, topics, thread IDs, and timestamps. A classifier (e.g., support vector machine, decision tree, neural network) can then be trained on these features. The resulting classifier can be used to classify previously unseen e-mail messages.

In the other case, we assume that we have only a relatively small set of labeled e-mail messages. A semi-supervised learning method (Section 7.10.3) is used to label either the unlabeled or new e-mail messages. One way is to design a generative model (mixture model) and use the EM algorithm to estimate model parameters using the same constraints as for the labeled e-mail messages. The estimated model can be used to further provide labels for previously unseen e-mail messages. Another strategy is to build a connective network representation of the e-mail, and propagate the class labels from the small set of training data to the majority of unlabeled e-mail.

■

10.12. Junk e-mail is one of the most annoying things in Web-based business or personal communications. Design an effective scheme (which may consist of a set of methods) that can be used to *filter out junk e-mail* effectively and discuss how such methods should be evolved along with time.

Answer:

Spam e-mail farms attempt to aggressively flood users with useless e-mail. There are two major challenges regarding this problem. First, the source addresses of these farms are usually very volatile, making it almost impossible to tell a junk e-mail only by its source address. Second, due to the extremely large scale of the Web, an effective method must be able to filter out junk e-mail very efficiently.

From a Web-server level, e-mail spam farms often form very *dense subgraphs*: they send e-mail to a large number of Web servers and users. Thus, it is important to first discover such subgraphs on the Web. In response to the demand of high efficiency, we could use the *shingling* method from information retrieval to extract the features of Web-link graphs and then perform recursive clustering on them [BDGM95]. Shingling takes a set of contiguous terms or "shingles" of documents and compares the number of matching shingles. It is a simple yet effective feature extraction method that is based on similarity search using the Jaccard coefficient (Section 7.2.2) and *min-wise independent permutation*. The resulting recursive clustering algorithm can largely shrink the size of Web links and efficiently discover the very dense subgraphs among billions of links. Once these spam farms are discovered, proper actions should be taken to forbid them in order to control the source of junk e-mail.

The pattern growth of spam e-mail farms is more variable than the growth of individual websites, supporting the intuition that growth in the graph is likely to be focused on particular spam sources. Thus, the method itself should keep dynamically updating the very dense subgraphs and detect spam farms.

In addition, from the user level, we could incorporate user guidance into the automatic junk e-mail detection process, that is, where users can report unwanted junk e-mail they have received. With a number of junk e-mail examples at hand, we can do the following. A *classifier* can be built using particular text mining techniques. When an e-mail server receives a suspicious e-mail message from an unknown or unusual source, it compares the content of the received message with the e-mail in the training set and determines if their similarity is above a certain threshold. If so, the received e-mail message can be safely removed or a warning message may be sent to the user.

■

10.13. It is difficult to construct a global data warehouse for the World Wide Web due to its dynamic nature and the huge amounts of data stored in it. However, it is still interesting and useful to construct data warehouses for summarized, localized, multidimensional information on the Internet. Suppose that an Internet information service company would like to set up an Internet-based data warehouse to help tourists choose local hotels and restaurants.

(a) Can you design a Web-based tourist data warehouse that would facilitate such a service?

Answer:

The star schema can be defined as follows: (1) The fact table will contain keys to the dimension tables that include *Hotels*, *Restaurants*, *Flights*, etc. In the global mediated schema, each table that we need based on the functionalities that we require serves as a dimension table. (2) The fact table also includes measures of interest such as *units_reserved*, etc.

■

(b) Suppose each hotel and/or restaurant contains a Web page of its own. Discuss how to locate such Web pages and what methods should be used to extract information from these Web pages in order to populate your Web-based tourist data warehouse.

Answer:

In order to offer such a service, after designing our OLAP schema, we need to import the data from the local databases of different broker websites to our data warehouse. However, since we don't have access to their underlying databases, we need to extract the data that we need from their Web-based query interfaces. Let's take restaurants as an example. We need a module that uses Web search engines to locate restaurants. For example, a two step method can be used, which first performs a Web-based keyword search to locate Web pages that are related to restaurants in, say, Chicago. The second step calls a filter module to analyze the page layouts and classifies those that are Web-query interfaces, and then extracts metadata required for our data warehouse.

We then need a module for data extraction. For this we first need to learn the query interface of the website. To do so, we can run a learning algorithm to learn the grammar of the query language of the underlying data source. We then use that query interface to run different queries and extract the data from the template result pages. This can be based on any information extraction method. (The simplest ones are rule-based, while more sophisticated ones exploit machine learning techniques). Next, we reconsolidate the different schemas of the different restaurant websites to the mediated schema and populate the data into them. In addition, we need a data cleaning step to remove duplicates and resolve data level heterogeneity issues.

■

(c) Discuss how to implement a mining method that may provide additional associated information, such as "90% *of customers who stay at the Downtown Hilton dine at the Emperor Garden Restaurant at least twice,*" each time a search returns a new Web page.

Answer:

Now that we have a data warehouse of different services, such as Restaurant, Hotel, etc., we can track the customers' data as well. We can run association rule mining that would include relations and attributes that are not present in single services. For example, we may run an association mining algorithm on a dataset that covers Users, Restaurants and Hotels and find associations such as *"x% of customers who stay at hotel A dine at the Restaurant B at least twice"*. It is not possible to mine such associations from single brokers that have only restaurant Web pages or hotel reservation services.

■

10.14. Each scientific or engineering discipline has its own subject index classification standard that is often used for *classifying documents* in its discipline.

(a) Design a Web document classification method that can take such a subject index to classify a set of Web documents automatically.

Answer:

A subject index can provide good guidance in automatic classification. The first step is to build the taxonomy using the index and the keywords associated with the index. Next, the corresponding documents of the index can serve as training and test sets for automatic classification. Under this scheme, a new Web document is assigned to a category having the same taxonomy. We can use the words associated or extended by these subject indices as feature words. The support and confidence measures are used to evaluate the category of the document. The two thresholds can be set by using the training examples, or learned and adjusted gradually as documents accumulate.

■

(b) Discuss how to use Web linkage information to improve the quality of such classification.

Answer:

Hyperlinks can be employed to detect clues regarding the semantics of Web pages. Such clues can be further matched to the taxonomy of different categories. A critical issue in using linkage information is to filter out noise associated with advertisement linkage and other junk links. There are two ways to address this issue. First, block-based linkage analysis can reduce the noise. Another way explored by many researchers is to use a statistical model to filter out the junk links. The statistical measure used may be entropy-based, because links such as *weather.com* are not biased and will not increase the information gain for a particular document. Experiments also show that Markov random fields with relaxation labeling also work well in improving classification accuracy.

■

(c) Discuss how to use Web usage information to improve the quality of such classification.

Answer:

Taxonomy is expanded gradually and new buzz words emerge from time to time. Web usage information, such as user-entered keywords, can be used to expand the taxonomy, providing a knowledge base and inference for Web document classification. Web usage information can be in the form of user

interactions, from which incremental classification can be performed to enhance classification accuracy. The usage information may also be represented as a Weblog, which can be mined for frequent keywords and mouse-click patterns. Classification rules can be further generated from the knowledge mined.

■

10.15. It is interesting to *cluster* a large set of Web pages based on their similarity.

(a) Discuss what should be the similarity measure in such cluster analysis.

Answer:

The complexity of Web pages is far greater than that of any traditional text document collection. Web pages lack a unifying structure and contain far more authoring styles and content variations than traditional text-based documents. The Web also contains a rich and dynamic collection of hyperlink information. When measuring the similarity between Web pages, the content as well as the link structure should be considered. For example, the task of identifying Web communities predicts whether a collection of Web pages focuses on a particular theme or topic, based on their attributes and their link structure. Therefore, the similarity measure should combine both the content similarity (e.g., cosine similarity, KL-divergence) and link structure similarity (e.g., inlink/outlink structures, cocitations).

■

(b) Discuss how the block-level analysis may influence the clustering results and how to develop an efficient algorithm based on this philosophy.

Answer:

It is human perception that people always view a Web page as different semantic objects rather than as a single object. When a Web page is presented to the user, the spatial and visual cues can help the user subconsciously divide the page into several semantic parts. These can be used to automatically segment the Web pages. Therefore, block-level analysis of the Web page content structure may influence the clustering results.

The VIsion-based Page Segmentation (VIPS) algorithm by Cai, He, Wen, and Ma [CHWM04] is an attempt to improve clustering by using block-level analysis. VIPS aims to extract the semantic structure of a Web page based on its visual presentation. Such a semantic structure is a tree structure, where each node in the tree corresponds to a block. Each node is assigned a value (i.e., a degree of coherence) that indicates how coherent the content in the block is based on visual perception. The VIPS algorithm makes full use of page layout features. It first extracts all of the suitable blocks from the HTML DOM tree, and then finds *separators* between these blocks. Here, separators denote the horizontal or vertical lines in a Web page that visually cross with no blocks. Based on these separators, the semantic tree of the Web page is constructed. A Web page can be represented as a set of blocks (leaf nodes of the semantic tree). Compared with DOM-based methods, the segments obtained by VIPS are more semantically aggregated. Noisy information, such as navigation, advertisement, and decoration, can be easily removed because these elements are often placed in certain positions on a page. Contents with different topics are distinguished as separate blocks.

By using the VIPS algorithm, we can extract page-to-block and block-to-page relationships and then construct a page graph and a block graph. Based on this graph model, new link analysis algorithms are capable of discovering the intrinsic semantic structure of the Web. These new algorithms can improve the performance of the search for Web context.

■

(c) Since different users may like to cluster a set of Web pages differently, discuss how a user may interact with a system to influence the final clustering results, and how such a mechanism can be developed systematically.

Answer:

Consider the following interactive clustering schema of Web pages. Given a set of Web pages, a user may want to cluster them according to his or her own beliefs or intuition. We consider three typical types of interactions that a user can provide to the system:

(1) *Prior knowlege of clusters*: The user has some prior hunches or strategies for defining the clusters. For example, he or she may begin by creating clusters with documents that have already been labeled with classes.

(2) *Constraints*: The user specifies constraints regarding which pages must be assigned to the same cluster, and which pages must be kept separate.

(3) *Attributes of interest*: The user specifies interesting attributes of a Web page as features to be clustered.

These three behaviors are by no means the only possible interactions, but we use them as examples for interactive Web-page clustering.

We now study how clustering can be approached for each of the above forms of interaction. For case (1), we typically formalize the problem into clustering with prior knowledge, or a semi-supervised problem. Model-based clustering methods, such as the EM algorithm, can be used. Each time the user provides prior knowledge regarding clusters, such information is considered as a prior for the statistic models used in clustering. Case (2) can be formalized as either a constraint-based clustering problem (as in Tung, Ng, Lakshmanan, and Han [TNLH00]), or a semi-supervised learning problem (as in Basu, Bilenko, and Mooney [BBM04]). For case (3), it is easy to change the similarity measure according to the attributes that the user selects. More formally, the cross-relational clustering methods as described in Yin, Han, and Yu [YHY05] can be applied to find a solution.

■

10.16. Weblog records provide rich *Web usage information* for data mining.

(a) Mining Weblog access sequences may help prefetch certain Web pages into a Web server buffer, such as those pages that are likely to be requested in the next several clicks. Design an efficient implementation method that may help mining such access sequences.

Answer:

Raw Weblog data need to be cleaned, condensed, and transformed in order to retrieve and analyze significant and useful information. A Weblog entry includes the URL requested, the IP address from which the request originated, and a timestamp. Based on the timestamp, for each IP address we consider the set of URLs requested (sorted by time of access) as a sequence. This sequence gives us the information of what page is requested after which pages. We can use any of the sequential pattern mining techniques (e.g., GSP, SPADE, or Prefix-Span) to mine all of the frequent sequences of URLs retrieved sequentially by an individual.

Whenever an URL, *"A"*, is requested by a user, we look in our frequent pattern database and retrieve all of the frequent patterns that start with *"A"*. We then fetch the k next URLs in those frequent patterns because we know that it is likely that the user may request those pages as well (since they are frequent).

■

(b) Mining Weblog access records can help cluster users into separate groups to facilitate customized marketing. Discuss how to develop an efficient implementation method that may help user clustering.

Answer:

From the raw Weblog data, we retrieve the list of URLs that a user visits frequently (i.e., above some threshold in a year) for each user (IP address). From the URLs, we can roll up based on concept hierarchies. For example, for the URL of a specific news Web page, we generalize to "NEWS"; if the URL corresponds to stock market data, we generalize to "STOCK MARKET". We treat all of these generalized URLs as features for the clustering of users. Two users are considered similar (and therefore in same cluster) if they frequently visit the same topics on the Web. Any clustering method (e.g., k-means, etc.) can be used on these features.

We can make marketing decisions based on the resulting clusters. For example, for people who are in the same cluster because they frequently check the stock market, we can submit stock information, and so on.

■

Chapter 11

Applications and Trends in Data Mining

11.7 Exercises

11.1. 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:

Many application areas are not covered in this chapter. Thus, the student can select any that he or she believes is fresh and interesting.

For example, one can consider an Earth observation system that watches 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.

■

11.2. 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 difficulty 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 are inefficient when executing some data mining queries because 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 consider mining queries to be complex and optimizable, ignoring 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.

■

11.3. Study an existing *commercial data mining system*. Outline the major features of such a system from a multiple dimensional 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 data mining systems, such as Microsoft SQL Server 2005, IBM IntelligentMiner, Purple Insight MineSet (originally from SGI), Oracle Data Mining, SPSS Clementine, SAS Enterprise Miner, and Insightful Miner. Here is a short summary of Microsoft SQLServer 2005.

SQLServer 2005 is an multifunction 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 2005 system is its tight integration of the data mining subsystem with the relational database and data warehouse subsystems, and its natural integration with multiple Microsoft tools, such as Excel, Web publishing, etc. It provides multiple data mining functions, including association, classification, prediction, clustering, and time-series analysis. This integration leads to a promising data mining methodology that integrates multidimensional 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 2005 integrates smoothly with relational database and data warehouse systems to provide a user-friendly, interactive data mining environment with high performance.

■

11.4. [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:

This is a good research topic with no definite answer, especially considering that the length of the answer could be as long as a thesis.

■

11.5. 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 foundations 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.

■

11.6. [Research project] Building a theory for data mining is to set 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 can fit into this framework. If some functions cannot fit well in the current theoretical framework, can you propose a way to extend the framework so that it can 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.

■

11.7. 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 the current data mining methodology.

Answer:

The perception of data mining as automated and scalable methods for statistical data analysis is interesting and 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. Many clustering and classification algorithms have some roots in statistical analysis, but have been reworked in a scalable algorithmic way. However, there are data mining methods that do not have statistical counterparts. For example, many frequent pattern analysis problems, such as sequential patterns, frequent substructures,

etc., do not have obvious counterparts in statistical analysis. There are also many sophisticated statistical analysis methods that have not been studied in data mining research.

■

11.8. 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 data mining method that may help people see through the network topology to the interesting features of the social network.

Answer:

Data visualization is the presentation of data in a visual way so that the contents can be easily comprehended by human users. *Pattern or knowledge visualization* presents 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. Patterns can suffer from the pattern abundance problems as well. In this case, users should be given primitives that allow them to easily extract a portion or portions of data using constraints. This focuses the data, and the domain viewed is smaller. Another way is to provide a multiresolution capability so that at certain levels, only the data and patterns that are visible in the current resolution scale will be shown. A third approach is to select only frequent or higher-weighted nodes and labels in demonstration so that complex connections can be simplified.

■

11.9. 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 [NF00].

It is also possible to develop video data mining or other methods that integrate 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.

■

11.10. 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 specific developments are often necessary due to different data characteristics and the different pattern types to be mined. 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. Both tracks are expected to have good applications. Due to the diversity of data and mining requests, many domain-dependent data mining systems will likely be generated in the future. However, many functions of such systems may share some core principles and methods.

■

11.11. What is a collaborative 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 referred to 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 makes predictions about which music a user likes 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). A item-item-based system categorizes items based on item-item correlation, whereas a customer-item-based system performs prediction on items based on the similarity between customers.

Both methods may explore data analysis and mining techniques, such as clustering, classification, and association. However, they differ from such techniques in that they *use* them 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.

■

11.12. 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 where 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 accessing precise information in individual data records. For example, when building a decision-tree classifier, we can use training data in which the values of individual records have been perturbed (i.e., *data obscuration*. 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. The original distribution of a collection of distorted data values can be approximated using a reconstruction algorithm. 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. We can then build a classifier using the approximated values, thereby protecting the privacy of individuals. By using these reconstructed distributions, the resulting classifier will be 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.

■

11.13. 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 of providing 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 applications 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.

■

11.14. 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, can you work out a plan so that progress can be made toward a solution to such a problem? How?

Answer:

Data mining for bioinformatics, text mining, Web and multimedia data mining, and mining spatiotemporal databases are several interesting 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 notable progress can be achieved, resulting in valuable applications.

■

11.15. Based on your study, suggest a possible *new frontier* in data mining that was not mentioned in this chapter.

Answer:

One frontier in data mining is to mine compressed but valuable patterns. Consider frequent pattern mining, which can generate huge sets of patterns. Although frequent patterns have been shown to be useful for classification, not every frequent pattern is equally useful. Using too many patterns will slow down the learning process and lead to overfitting. Thus, an important research problem is how to find a much smaller but better quality set of frequents patterns that can be used for tasks like classification or clustering. Our lab is investigating the use of a feature selection method that can single out a set of frequent, discriminative, and nonredundant patterns in comparison to the use of selected single features for classification. The selected frequent patterns represent the data in a different feature space, in which any learning algorithm can be used, whereas associative classification builds a classification model using the rules. Experimental results are promising. Therefore, the mining of compressed, high quality patterns is an important frontier in data mining.

■

Bibliography

[AHWY03] C. C. Aggarwal, J. Han, J. Wang, and P. S. Yu. A framework for clustering evolving data streams. In *Proc. 2003 Int. Conf. Very Large Data Bases (VLDB'03)*, pages 81–92, Berlin, Germany, Sept. 2003.

[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.

[AS95] R. Agrawal and R. Srikant. Mining sequential patterns. In *Proc. 1995 Int. Conf. Data Engineering (ICDE'95)*, pages 3–14, Taipei, Taiwan, Mar. 1995.

[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.

[BB02] C. Borgelt and M. R. Berthold. Mining molecular fragments: Finding relevant substructures of molecules. In *Proc. 2002 Int. Conf. Data Mining (ICDM'02)*, pages 211–218, Maebashi, Japan, Dec. 2002.

[BBM04] S. Basu, M. Bilenko, and R. J. Mooney. A probabilistic framework for semi-supervised clustering. In *Proc. 2004 ACM SIGKDD Int. Conf. Knowledge Discovery in Databases (KDD'04)*, pages 59–68, Seattle, WA, Aug. 2004.

[BDGM95] S. Brin, J. Davis, and H. Garcia-Molina. Copy detection for mechanisms for digital documents. In *Proc. 1995 ACM-SIGMOD Int. Conf. Management of Data (SIGMOD'95)*, pages 398–409, May 1995.

[BFR98] P. Bradley, U. Fayyad, and C. Reina. Scaling clustering algorithms to large databases. In *Proc. 1998 Int. Conf. Knowledge Discovery and Data Mining (KDD'98)*, pages 9–15, New York, NY, Aug. 1998.

[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.

[CDH+02] Y. Chen, G. Dong, J. Han, B. W. Wah, and J. Wang. Multi-dimensional regression analysis of time-series data streams. In *Proc. 2002 Int. Conf. Very Large Data Bases (VLDB'02)*, pages 323–334, Hong Kong, China, Aug. 2002.

[CHWM04] D. Cai, X. He, J.-R. Wen, and W.-Y. Ma. Block-level link analysis. In *Proc. Int. 2004 ACM SIGIR Conf. Research and Development in Information Retrieval (SIGIR'04)*, pages 440–447, Sheffield, UK, July 2004.

[CKM05] Z. Chen, D. V. Kalashnikov, and S. Mehrotra. Exploiting relationships for object consolidation. In *ACM SIGMOD Workshop on Information Quality in Information Systems (IQIS)*, June 2005.

[Coh04] J. Cohen. Bioinformatics—an introduction for computer scientists. *ACM Computing Surveys*, 36:122–158, 2004.

[CYH04] H. Cheng, X. Yan, and J. Han. IncSpan: Incremental mining of sequential patterns in large database. In *Proc. 2004 ACM SIGKDD Int. Conf. Knowledge Discovery in Databases (KDD'04)*, pages 527–532, Seattle, WA, Aug. 2004.

[DEKM98] R. Durbin, S. Eddy, A. Krogh, and G. Mitchison. *Biological Sequence Analysis: Probability Models of Proteins and Nucleic Acids*. Cambridge University Press, 1998.

[DHL+01] G. Dong, J. Han, J. Lam, J. Pei, and K. Wang. Mining multi-dimensional constrained gradients in data cubes. In *Proc. 2001 Int. Conf. on Very Large Data Bases (VLDB'01)*, pages 321–330, Rome, Italy, Sept. 2001.

[GFKT01] L. Getoor, N. Friedman, D. Koller, and B. Taskar. Learning probabilistic models of relational structure. In *Proc. 2001 Int. Conf. Machine Learning (ICML'01)*, pages 170–177, Williamstown, MA, 2001.

[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.

[GZ03] G. Grahne and J. Zhu. Efficiently using prefix-trees in mining frequent itemsets. In *Proc. ICDM'03 Int. Workshop on Frequent Itemset Mining Implementations (FIMI'03)*, Melbourne, FL, Nov. 2003.

[HNFD98] J. Han, R. T. Ng, Y. Fu, and S. Dao. Dealing with semantic heterogeneity by generalization-based data mining techniques. In M. P. Papazoglou and G. Schlageter, editors, *Cooperative Information Systems: Current Trends Directions*, pages 207–231. Academic Press, 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.

[JTH01] W. Jin, K. H. Tung, and J. Han. Mining top-n local outliers in large databases. In *Proc. 2001 ACM SIGKDD Int. Conf. Knowledge Discovery in Databases (KDD'01)*, pages 293–298, San Fransisco, CA, Aug. 2001.

[JTHW06] W. Jin, A. K. H. Tung, J. Han, and W. Wang. Ranking outliers using symmetric neighborhood relationship. In *Proc. 2006 Pacific-Asia Conf. Knowledge Discovery and Data Mining (PAKDD'06)*, Singapore, April 2006.

[Ker92] R. Kerber. Discretization of numeric attributes. In *Proc. 1992 Nat. Conf. Artificial Intelligence (AAAI'92)*, pages 123–128, AAAI/MIT Press, 1992.

[KH95] K. Koperski and J. Han. Discovery of spatial association rules in geographic information databases. In *Proc. 1995 Int. Symp. Large Spatial Databases (SSD'95)*, pages 47–66, Portland, ME, Aug. 1995.

[KK04] M. Kuramochi and G. Karypis. Finding frequent patterns in a large sparse graph. In *Proc. 2004 SIAM Int. Conf. Data Mining (SDM'04)*, Lake Buena Vista, FL, April 2004.

[LHG04] X. Li, J. Han, and H. Gonzalez. High-dimensional OLAP: A minimal cubing approach. In *Proc. 2004 Int. Conf. Very Large Data Bases (VLDB'04)*, pages 528–539, Toronto, Canada, Aug. 2004.

[LKF05] J. Leskovec, J. Kleinberg, and C. Faloutsos. Graphs over time: Densification laws, shrinking diameters and possible explanations. In *Proc. 2005 ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD'05)*, pages 177–187, Chicago, IL, Aug. 2005.

[LLLY03] G. Liu, H. Lu, W. Lou, and J. X. Yu. On computing, storing and querying frequent patterns. In *Proc. 2003 ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining (KDD'03)*, pages 607–612, Washington, DC, Aug. 2003.

[LPH02] L. V. S. Lakshmanan, J. Pei, and J. Han. Quotient cube: How to summarize the semantics of a data cube. In *Proc. 2002 Int. Conf. on Very Large Data Bases (VLDB'02)*, pages 778–789, Hong Kong, China, Aug. 2002.

[ML01] D. Malerba and F. A. Lisi. Discovering associations between spatial objects: An ilp application. In *Proc. 2001 Int. Conf. Inductive Logic Programming (ILP'01)*, pages 156–163, Strasbourg, France, Sept. 2001.

[NF00] M. Noirhomme-Fraiture. Multimedia support for complex multidimensional data mining. In *Proc. 2000 SIGKDD Int. Workshop on Multimedia Data Mining (MDM'00)*, pages 54–59, Boston, MA, Aug. 2000.

[NLHP98] R. Ng, L. V. S. Lakshmanan, J. Han, and A. Pang. Exploratory mining and pruning optimizations of constrained associations rules. In *Proc. 1998 ACM-SIGMOD Int. Conf. Management of Data (SIGMOD'98)*, pages 13–24, Seattle, WA, June 1998.

[PBTL99] N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal. Discovering frequent closed itemsets for association rules. In *Proc. 7th Int. Conf. Database Theory (ICDT'99)*, pages 398–416, Jerusalem, Israel, Jan. 1999.

[PCT+03] F. Pan, G. Cong, A. K. H. Tung, J. Yang, and M. Zaki. CARPENTER: Finding closed patterns in long biological datasets. In *Proc. 2003 ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining (KDD'03)*, pages 637–642, Washington, DC, Aug. 2003.

[PHL01] J. Pei, J. Han, and L. V. S. Lakshmanan. Mining frequent itemsets with convertible constraints. In *Proc. 2001 Int. Conf. Data Engineering (ICDE'01)*, pages 433–332, Heidelberg, Germany, April 2001.

[PHMA+01] J. Pei, J. Han, B. Mortazavi-Asl, H. Pinto, Q. Chen, U. Dayal, and M.-C. Hsu. PrefixSpan: Mining sequential patterns efficiently by prefix-projected pattern growth. In *Proc. 2001 Int. Conf. Data Engineering (ICDE'01)*, pages 215–224, Heidelberg, Germany, April 2001.

[SA96] R. Srikant and R. Agrawal. Mining sequential patterns: Generalizations and performance improvements. In *Proc. 5th Int. Conf. Extending Database Technology (EDBT'96)*, pages 3–17, Avignon, France, Mar. 1996.

[SAM98] S. Sarawagi, R. Agrawal, and N. Megiddo. Discovery-driven exploration of OLAP data cubes. In *Proc. Int. Conf. of Extending Database Technology (EDBT'98)*, pages 168–182, Valencia, Spain, Mar. 1998.

[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.

[THLN01] A. K. H. Tung, J. Han, L. V. S. Lakshmanan, and R. T. Ng. Constraint-based clustering in large databases. In *Proc. 2001 Int. Conf. Database Theory (ICDT'01)*, pages 405–419, London, UK, Jan. 2001.

[TNLH00] A. K. H. Tung, R. Ng, L. Lakshmanan, and J. Han. Constraint-based clustering in large databases. In *CS Technical Report*, Simon Fraser University, June 2000.

[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.

[WH04] J. Wang and J. Han. BIDE: Efficient mining of frequent closed sequences. In *Proc. 2004 Int. Conf. Data Engineering (ICDE'04)*, pages 79–90, Boston, MA, Mar. 2004.

[WHP03] J. Wang, J. Han, and J. Pei. CLOSET+: Searching for the best strategies for mining frequent closed itemsets. In *Proc. 2003 ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining (KDD'03)*, pages 236–245, Washington, DC, Aug. 2003.

[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.

[XHYC05] D. Xin, J. Han, X. Yan, and H. Cheng. Mining compressed frequent-pattern sets. In *Proc. 2005 Int. Conf. Very Large Data Bases (VLDB'05)*, pages 709–720, Trondheim, Norway, Aug. 2005.

[YHA03] X. Yan, J. Han, and R. Afshar. CloSpan: Mining closed sequential patterns in large datasets. In *Proc. 2003 SIAM Int. Conf. Data Mining (SDM'03)*, pages 166–177, San Fransisco, CA, May 2003.

[YHY05] X. Yin, J. Han, and P.S. Yu. Cross-relational clustering with user's guidance. In *Proc. 2005 ACM SIGKDD Int. Conf. Knowledge Discovery in Databases (KDD'05)*, pages 344–353, Chicago, IL, Aug. 2005.

[YHYY04] X. Yin, J. Han, J. Yang, and P. S. Yu. CrossMine: Efficient classification across multiple database relations. In *Proc. 2004 Int. Conf. Data Engineering (ICDE'04)*, pages 399–410, Boston, MA, Mar. 2004.

[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.

[YZH05] X. Yan, X. J. Zhou, and J. Han. Mining closed relational graphs with connectivity constraints. In *Proc. 2005 ACM SIGKDD Int. Conf. Knowledge Discovery in Databases (KDD'05)*, pages 324–333, Chicago, IL, Aug. 2005.

[Zak00] M. J. Zaki. Scalable algorithms for association mining. *IEEE Trans. Knowledge and Data Engineering*, 12:372–390, 2000.

[Zak01] M. Zaki. SPADE: An efficient algorithm for mining frequent sequences. *Machine Learning*, 40:31–60, 2001.

[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.

[ZH02] M. J. Zaki and C. J. Hsiao. CHARM: An efficient algorithm for closed itemset mining. In *Proc. 2002 SIAM Int. Conf. Data Mining (SDM'02)*, pages 457–473, Arlington, VA, April 2002.