2. Data Mining and Multi-database Mining

2.1 Introduction

This chapter provides an introduction to data mining, reviews existing research into multi-database mining, and describes some required concepts.

The pressure to enhance corporate profitability has caused companies to spend more time identifying diverse opportunities in areas such as sales and investment. To this end, huge amounts of data are collected in company databases for decision-support purposes. Also, amalgamations, new partnerships, and takeovers have resulted in the formation of particularly large companies, or organizations, that utilize increasingly larger multi-database systems. Government enterprises and academic research are also generating, and making use of, growing amounts of data.

The examples below should be sufficient to put the current situation into perspective.

– NASA's Earth Observing System (EOS), for orbiting satellites and other space-borne instruments, sends one terabyte of data to receiving stations each day.
– By the year 2000 a typical Fortune 500 company was projected to possess more than 400 trillion characters in their electronic databases, requiring 400 terabytes of mass storage.

With the increasing use of databases the need to be able to digest large volumes of data is now critical. Therefore, data mining techniques are being widely researched as new innovations become imperative.

As we have stated, this book presents new techniques for multi-database mining. By way of a preliminary discussion, this chapter briefly introduces data mining techniques, existing research into multi-database mining, and basic concepts.

We begin by summarizing the process of knowledge discovery. Then, in Section 2.3, we introduce some of the basic concepts, the knowledge of which is required for understanding this book. In Section 2.4, we outline past research into mono-database mining and, in Section 2.5, past research into multi-database mining. Finally we summarize the chapter.

2.2 Knowledge Discovery in Databases

Knowledge discovery in databases (KDD) (also referred to as data mining), is the extraction of hidden predictive information from large databases. It is a powerful new technology with great potential to help companies, for example, to focus on the most important information in their data warehouses. KDD tools predict future trends and behavior, allowing businesses to make proactive, knowledge-driven decisions. The automated prospective analyses offered by KDD move beyond the analysis of past events provided by retrospective tools typical of decision-support systems. KDD tools can answer business questions that were traditionally too time consuming to resolve. They scour databases for hidden patterns, finding predictive information that experts might miss because it lies outside their expectations.

Most companies already collect and refine massive quantities of data. KDD techniques can be implemented rapidly on existing software and hardware platforms to enhance the value of existing information resources, and can be integrated with new products and systems as they are brought online. When implemented on high performance client/server or parallel processing computers, KDD tools can analyze massive databases to deliver answers to questions such as: Which clients are most likely to respond to my next promotional mailing, and why?

A widely accepted definition of KDD is given by Fayyad et al. in which KDD is defined as the nontrivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data (Fayyad-Piatetsky-Smyth 1996). The definition regards KDD as a complicated process comprising a number of steps. Data mining is one step in the process.

2.2.1 Processing Steps of KDD

In general, the process of knowledge discovery in databases consists of an iterative sequence of the following steps (Han-Huang-Cercone-Fu 1996, Han 1999, Liu-Motoda 1998, Wu 1995, Zhang 1989).

Defining the problem. The goals of the knowledge discovery project must be identified and must be verified as actionable. For example, if the goals are met, a business can then put newly discovered knowledge to use. The data to be used must also be identified.

Data preprocessing. This includes data collection, data cleaning, data integration, data selection, and data transformation.

- Data collection obtains necessary data from various internal and external sources; resolves representation and encoding differences; and joins data from various tables to create a homogeneous source.
- Data cleaning checks and resolves data conflicts, outliers (unusual or exception values), noisy, erroneous, missing data, and ambiguity; and uses conversions and combinations to generate new data fields, such

as ratios or rolled-up summaries. These steps require considerable effort, often as much as 70 percent, or more, of the total data mining effort.

- Data integration integrates multiple, heterogeneous data-sources into a single source.
- Data selection is where data relevant to the analysis task is retrieved from the database. In other words, it selects a dataset, or focuses on a subset of variables or data samples, on which discovery is to be performed.
- Data transformation is where data are transformed or consolidated into forms appropriate for mining, by performing summary, or aggregation, operations.

Data mining is an essential process where intelligent methods are applied in order to extract data patterns. It searches for patterns of interest in a particular representational form, or a set of such representations, including classification rules or trees, regression, clustering, sequence modeling, dependency, and so forth. The user can significantly aid the data mining method by correctly performing the preceding steps.

Post data mining includes pattern evaluation, deployment of the model, maintenance, and the presentation of knowledge.

- Pattern evaluation identifies the truly interesting patterns representing knowledge, based on certain *interesting measures*, tests the model for accuracy on an independent dataset — one that has not been used to create the model; assesses the sensitivity of a model, and pilot tests the model for usability. For example, if a model is used to predict customer response, then a prediction can be made and a test mailing done to a subset in order to check how closely the responses match predictions.
- Deployment of the model. A predictive model is used to predict results for new cases. The prediction is then used to improve organizational behavior. Deployment may require building computerized systems that capture the appropriate data and generate a prediction in real time so that a decision maker can apply the prediction. For example, a model can determine whether a credit card transaction is likely to be fraudulent.
- Maintenance. Whatever is being modeled, things are likely to change over time. The economy changes, competitors introduce new products, or the news media comes up with a new hot topic. Any of these forces can alter customer behavior. So the model that was correct yesterday and today might no longer be appropriate tomorrow. Maintaining models requires constant revalidation of the model using new data to assess whether it is still appropriate.

- The presentation of knowledge is where visualization and knowledge representation techniques are used to present mined knowledge to users.

The knowledge discovery process is iterative. For example, while cleaning and preparing data you might discover that data from a certain source are unusable, or that you require data from a previously unidentified source to be merged with other data. Often, the first time through, the data mining step will reveal that additional data cleaning is required.

2.2.2 Data Pre-processing

The ability to analyze and understand massive datasets lags far behind the ability to gather and store the data. Therefore, knowledge discovery and data mining are rapidly becoming an important field of research. No matter how powerful computers are now, or will be in the future, KDD researchers and practitioners must consider ways of efficiently managing the ever-growing data generated by the extensive use of computers and ease of data collection. Many different approaches have been introduced to address the data explosion issue. These include algorithm scale-up and data reduction (by data pre-processing). As a result, data pre-processing can be more time consuming, and can present more challenges than can data mining (Fayyad-Simoudis 1997).

Data collection is a very important step in knowledge discovery within databases. This step obtains necessary data from various internal and external data-sources that are relevant to a mining application. This can take advantage of the remarkable possibilities of access to information and knowledge that the Internet provides. Web technologies, such as HTTP and HTML, have dramatically changed enterprise information management. The vast amount of information available on the World Wide Web has great potential to improve the quality of decision-making (Lesser 1998, 2000). A corporation can benefit from intranets and the Internet to gather, manage, distribute, and share data, inside and outside the corporation.

Real-world data often have to be transformed in many ways for use in different situations. Also, the data can have discrepancies in structure and content that must be cleaned. In addition, many visualization tools, or tools for analysis, require the data to be in particular formats. Traditionally, such transformation has been done through ad hoc scripts, or through cookie-cutter transformation tools that require much laborious and errorprone programming. Moreover, the transformation process is typically decoupled from the analysis process. On large datasets, such transformation and analysis is quite timeconsuming. Users often need to perform many iterations of analysis and transformation, and have to endure many long, frustrating delays.

Data integration joins all relevant internal and external data to create a single homogeneous dataset. When internal and external data are joined into a single dataset for mining tasks all of the data play an equal role in data mining. However, because some collected data may be untrustworthy (even fraudulent), useful patterns can be disguised. If external data are not pre-processed before they are applied, patterns identified from the data can result in a high-risk application. For example, a stock investor might need to collect information from outside data-sources to make an investment decision. If the investor gathers fraudulent information, and the information is directly applied to investment decisions, he or she might lose money. Hence, it is very important to collect quality data.

Data often contain noise and erroneous components, and can have missing values. There is also the possibility that redundant or irrelevant variables have been recorded, while important features have been overlooked. Data cleaning includes provision for correcting inaccuracies, removing anomalies, eliminating duplicate records, filling holes in the data, and checking entries for consistency. Cleaning is required to make the necessary transformation of the original into a format suitable for use by data mining tools.

Another important requirement with the KDD process is feature selection (Liu-Motoda 1998, Wu 2000). KDD is a complicated task and often depends on the proper selection of features. Feature selection is the process whereby features are chosen that are necessary and sufficient to represent the data. There are several issues that influence feature selection. These include: masking variables, the number of variables employed in the analysis, and relevancy of variables.

Masking variables hides, or disguises, patterns in data. Numerous studies have shown that inclusion of irrelevant variables can hide real clustering of the data, so only those variables which help discriminate the clustering should be included in the analysis.

The number of variables used in data mining is also an important consideration. There is generally a tendency to use more and more variables. However, increased dimensionality has an adverse effect because, for a fixed number of data patterns, increased dimensionality makes the multi-dimensional data space sparse.

However, failing to include relevant variables also causes failure in identifying the clusters. A practical difficulty in mining some industrial data is knowing whether all important variables have been included in the data records.

2.2.3 Data Mining

Data mining has been popularly treated as a synonym of knowledge discovery in databases, although some researchers even view data mining as the kernel step of knowledge discovery. Data mining derives its name from similarities between searching for valuable business information in a large database and mining a mountain for a vein of valuable ore. Both processes require either sifting through an immense amount of material, or intelligently probing it to find exactly where the value resides.

> Strictly speaking, *data mining is the process of discovering interesting knowledge, such as patterns, associations, changes, anomalies and significant structures, from large amounts of data stored in databases, data warehouses, or other information repositories.* This valuable information can be in the form of patterns, associations, changes, anomalies, and significant structures (Fayyad-Piatetsky-Smyth 1996). In other words, data mining attempts to extract potentially useful knowledge from data.

Given databases of sufficient size and quality, data mining technology can generate new business opportunities as follows.

1. **Prediction of trends and behaviors**. Data mining seeks predictive information in large databases. Questions that traditionally required extensive hands-on analysis can now be quickly answered directly from the data. A typical example of a predictive problem is targeted marketing. Data mining uses data on past promotional mailings to identify the targets most likely to maximize return on investment in future mailings. Other predictive problems include forecasting bankruptcy and other forms of default, and identifying segments of a population likely to respond similarly to given events.
2. **Discovery of previously unknown patterns**. Data mining tools sweep through databases and, in one step, identify previously hidden patterns. An example of pattern discovery is the analysis of retail sales data to identify seemingly unrelated products that are often purchased together. Other pattern discovery problems include detecting fraudulent credit card transactions and identifying anomalous data that could represent data entry keying errors.

Data mining techniques can yield the benefits of automation on existing software and hardware platforms, and can be implemented on new systems as existing platforms are upgraded and new products developed. When data mining tools are implemented on high-performance parallel processing systems, they are able to analyze massive databases in minutes. Faster processing means that users can automatically experiment with more models to understand complex data. This makes it practical for users to analyze huge quantities of data. Larger databases, in turn, yield improved predictions. With the rapid advance in data capture, transmission, and storage, large-system users will increasingly need to implement new and innovative ways to mine the after-market value of their vast stores of detailed data, employing MPP (massive parallel processing) systems to create new sources of business advantage.

The most commonly used techniques in data mining are as follows.

Decision trees: Tree-shaped structures that represent sets of decisions. These decisions generate rules for the classification of a dataset. Specific decision tree methods include Classification and Regression Trees (CART) and Chi-Square Automatic Interaction Detection (CHAID).

Nearest neighbor method: A technique that classifies each record in a dataset based on a combination of the classes of k record(s) most similar to it in an historical dataset, sometimes known as the k-nearest neighbor technique.

Rule induction: The extraction of useful if-then rules from data, based on statistical significance.

Genetic algorithms: Optimization techniques that use processes such as genetic combination, mutation, and natural selection in a design based on the concepts of evolution.

Artificial neural networks: Nonlinear predictive models that learn through training, and resemble biological neural networks in structure.

Many of these technologies have been in use for more than a decade with specialized analysis tools that work with relatively small volumes of data. These capabilities are now evolving to integrate directly with industry-standard data warehouses and OLAP platforms.

One of the important topics in data mining is association rule mining. Since its introduction in 1993 (Agrawal-Imielinski-Swami 1993) the method of association rule mining has received a great deal of attention. It has been mainly developed to identify the relationships among itemsets that have high frequency and strong correlation. Association rules enable us to detect the items that often occur together in an application. This book illustrates proposed techniques for mining multi-databases, using association rules.

2.2.4 Post Data Mining

Post data mining is used to analyze, cluster, and maintain the patterns discovered from databases. Pattern analysis and clustering are helpful in improving efficiency when using patterns. Pattern maintenance is necessary to identify the changes of patterns in real-world applications.

However, even when patterns have been identified from a database, it does not mean that the mining process can be terminated. We must analyze and cluster the mined patterns so as to detail a way to use the patterns. On the other hand, the number of patterns discovered may be so large that browsing the pattern set and finding interesting patterns can be somewhat difficult for users. It can be hard to identify which of the patterns are really useful to applications. One of the tasks required in post data mining is to improve efficiency when using the patterns.

In many applications, the databases are dynamic, that is, transactions are continuously being added. There is much work that focuses on mining frequent itemsets in market basket datasets (e.g., (Agrawal-Imielinski-Swami

1993, Brin-Motwani-Silverstein 1997)). However, many items such as suits, toys and some foods, represent smart model in market basket data. For example, jeans and white shirt may have often been purchased at one time from a department store, and black trousers and blue T-shirt often purchased at another time. The department store may have made different decisions when buying according to such different purchasing models. This means that certain goods are very often purchased at one time according to market basket data, and they are solely purchased at another time. These items are called *fashionable goods*. Although most fashionable items may not be large itemsets in a market basket dataset, such itemsets are useful when making decisions on buying. Consequently, mining fashionable patterns is an important issue when mining market basket data. Indeed, since new data may represent a changing trend of customer buying patterns, we should intuitively have more confidence in the new data than in the old. Thus, novelty of data should be highlighted in mining models. However, although mining customer buying patterns based on the support-confidence framework can reflect the frequency of itemsets, it cannot catch the stylishness of data. Another task of post data mining is to maintain patterns and identify trend patterns.

2.2.5 Applications of KDD

KDD is potentially valuable in virtually any industrial and business sector where database and information technology are used. A wide range of companies has deployed successful applications of data mining. While early adopters of this technology have tended to be in information-intensive industries such as financial services and direct-mail marketing, the technology is applicable to any company looking to leverage a large data warehouse to better manage their customer relations. Two critical factors for success with data mining are: a large, well-integrated data warehouse and a well-defined understanding of the business process within which data mining is to be applied (such as customer prospecting, retention, campaign management, and so on).

Some successful application areas include the followings.

1. A pharmaceutical company can analyze its recent sales force activities and its results to improve its targeting of high-value physicians and to determine which marketing activities will have the greatest impact in the next few months. The data need to include competitor market activity as well as information about the local health-care systems. The results can be distributed to the sales force via a wide-area network that enables the representatives to review the recommendations from the perspective of the key attributes in the decision process. The ongoing dynamic analysis of the data warehouse allows best practices from throughout the organization to be applied in specific sales situations.
2. A credit card company can leverage its vast warehouse of customer transaction data to identify customers most likely to be interested in a new

credit product. Using a small test mailing, the attributes of customers with an affinity to the product can be identified. Recent projects have indicated more than a twenty-fold decrease in costs for targeted mailing campaigns over conventional approaches.

3. A diversified transportation company, with a large direct sales force, can apply data mining to identify the best prospects for its services. Using data mining to analyze its own customer experience, this company can build a unique segmentation identifying the attributes of high-value prospects. Applying this segmentation to a general business database, such as those provided by Dun and Bradstreet, can yield a prioritized list of prospects by region.
4. A large consumer package goods company can apply data mining to improve its sales process to retailers. Data from consumer panels, shipments, and competitor activity can be applied to understand the reasons for brand and store switching. Through this analysis, the manufacturer can select promotional strategies that best reach their target customer segments.

Each of these examples has a clear common ground. They leverage the knowledge about customers implicit in a data warehouse to reduce costs and improve the value of customer relations. These organizations can now focus their efforts on the most important (profitable) customers and prospects, and design targeted marketing strategies to best reach them.

One of the most popular and successful applications of database systems is in the area of marketing, where a great deal of information about customer behavior is collected. Marketers are interested in finding customer preferences so as to target them in future campaigns (Berry, 1994, Fayyad-Simoudis 1997).

There are various applications reported in recent data mining conferences and journals and in various journal special issues on data mining. The following are some applications reported in (Fayyad-Simoudis 1997, Piatetsky-Matheus 1992).

- The SKICAT (Sky Image Cataloging and Analysis Tool) system concerns an automation of reduction and analysis of the large astronomical dataset known as the Palomar Observatory Digital Sky Survey (POSS-II) (Fayyad-Piatetsky-Smyth 1996). The database is huge: three terabytes of images containing on the order of two billion sky objects. This research was initiated by George Djorgovski from the California Institute of Technology who realized that new techniques were required in order to analyze such huge amounts of data. He teamed up with Jet Propulsion Laboratory's Usama Fayyad and others. The result was SKICAT.
- Health-KEFIR (Key Findings Reporter) is a knowledge discovery system used in health care as an early warning system (Fayyad-Piatetsky-Smyth 1996). The system concentrates on ranking deviations according to mea-

sures of how interesting these events are to the user. It focuses on discovering and explaining key findings in large and dynamic databases. The system performs an automatic drill-down through data along multiple dimensions to determine the most interesting deviations of specific quantitative measures relative to their previous and expected values. The deviation technique is a powerful tool used in KEFIR to identify interesting patterns from the data. The deviations are then ranked, using some measure of interestingness, such as looking at the actions that can be taken in response to the relevant deviations. They might even generate recommendations for corresponding actions. KEFIR uses Netscape to present its findings in a hypertext report, using natural language and business graphics.

- TASA (Telecommunication Network Alarm Sequence Analyzer) was developed for predicting faults in a communication network (Fayyad-Piatetsky-Smyth 1996). A typical network generates hundreds of alarms per day. The TASA system generates rules like if a certain combination of alarms occurs within (...) time, then an alarm of another type will occur within (...) time. The time periods for the "if" part of the rules are selected by the user, who can rank or group the rules once they are generated by TASA.
- R-MINI system uses both deviation detection and classification techniques to extract useful information from noisy domains (Fayyad-Piatetsky-Smyth 1996). It uses logic to generate a minimal size rule set that is both complete and consistent.
- Knowledge Discovery Workbench (KDW) (Piatetsky-Matheus 1992) is a collection of methods used for interactive analysis of large business databases. It includes many different methods for clustering, classification, deviation detection, summarization, dependency analysis, and so forth. It is the user, however, who needs to guide the system in searches. Thus, if the user is knowledgeable in both the domain and the tools used, the KDW system can be domain independent and versatile.
- Experiment result analysis summarizes experiment results and predictive models.
- Clementine is a commercial software package for data mining (Integrated Solutions, Ltd.) (Fayyad-Piatetsky-Smyth 1996). Basically, it is a classifier system based on neural networks and inductive machine learning. It has been applied for the prediction of viewing audiences for the BBC, selection of retail outlets, anticipating toxic health hazards, modeling skin corrosivity, and so on.

2.3 Association Rule Mining

This section recalls some concepts required for association rule mining in this book.

Let $I = \{i_1, i_2, ..., i_m\}$ be a set of literals, or *items*. For example, goods such as milk, sugar, and bread for purchase in a store are items; and $A_i = v$ is an item, where v is a domain value of attribute A_i in a relation $R(A_1, ..., A_n)$.

X is an *itemset* if it is a subset of I. For example, a set of items for purchase from a store is an itemset; and a set of $A_i = v$ is an itemset for the relation $R(PID, A_1, A_2, ..., A_n)$, where PID is a key.

$D = \{t_i, t_{i+1}, ..., t_n\}$ is a set of transactions, called the *transaction database*, where each transaction t has a *tid* and a t-itemset: $t = (tid, t\text{-itemset})$. For example, the shopping cart of a customer going through checkout is a transaction; and a tuple $(v_1, ..., v_n)$ of the relation $R(A_1, ..., A_n)$ is a transaction.

A transaction t contains an itemset X if, and only if, (iff) for all items, $i \in X$, i is in t-itemset. For example, a shopping cart contains all items in X when going through checkout; and for each $A_i = v_i$ in X, v_i occurs at position i in the tuple $(v_1, ..., v_n)$.

An itemset X in a transaction database D has a *support*, denoted as $supp(X)$ (we also use $p(X)$ to stand for $supp(X)$), that is, the ratio of transactions in D contains X. Or

$$supp(X) = |X(t)|/|D|,$$

where $X(t) = \{t \text{ in } D | t \text{ contains } X\}$.

An itemset X in a transaction database D is called a large (frequent) itemset if its support is equal to, or greater than, a threshold of minimal support ($minsupp$), which is given by users or experts.

An *association rule* is an implication $X \rightarrow Y$, where itemsets X and Y do not intersect.

Each association rule has two quality measurements: *support* and *confidence*, defined as

– the support of a rule $X \rightarrow Y$ is the support of $X \cup Y$, where $X \cup Y$ means both X and Y occur at the same time;
– the confidence of a rule $X \rightarrow Y$ is $conf(X \rightarrow Y)$ as the ratio $|(X \cup Y)(t)|/|X(t)|$ or $supp(X \cup Y)/supp(X)$.

That is, support = frequencies of occurring patterns; confidence = strength of implication.

The *support-confidence framework* (Agrawal-Imielinski-Swami 1993): Let I be the set of items in database D, $X, Y \subseteq I$ be itemsets, $X \cap Y = \emptyset$, $p(X) \neq 0$, and $p(Y) \neq 0$. Minimal support ($minsupp$) and minimal confidence ($minconf$) are given by users or experts. Then $X \rightarrow Y$ is a valid rule if

(1) $supp(X \cup Y) \geq minsupp$,
(2) $conf(X \rightarrow Y) = \frac{supp(X \cup Y)}{supp(X)} \geq minconf$,

where $conf(X \to Y)$ stands for the confidence of the rule $X \to Y$.

Mining association rules can be broken down into the subproblems:

(1) generating all itemsets that have support greater than, or equal to, the user-specified minimum support; that is, generating all large itemsets;

(2) generating all the rules that have minimum confidence in the following naive way: for every large itemset X, and any $B \subset X$, let $A = X - B$. If the confidence of a rule $A \to B$ is greater than, or equal to, the minimum confidence (or $supp(X)/supp(A) \geq minconf$), then it can be extracted as a valid rule.

To demonstrate the use of the support-confidence framework, we detail the process of mining association rules by an example as follows.

Let the item universe be $I = \{A, B, C, D, E\}$; and a transaction database be $TID = \{100, 200, 300, 400\}$. The data in the transactions are listed in Table 2.1.

Table 2.1 *Sample transaction database*

TID	*Items*				
100	*A*		*C*	*D*	
200		*B*	*C*		*E*
300	*A*	*B*	*C*		*E*
400		*B*			*E*

In Table 2.1, 100, 200, 300, 400 are the unique identifiers of the four transactions; and A = sugar, B = bread, C = coffee, D = milk, and E = cake.

Each row in Table 2.1 can be taken as a transaction. We can discover association rules from these transactions using the support-confidence framework. Let

$minsupp = 50\%$ (to be frequent, an itemset must be in at least two transactions);
$minconf = 60\%$ (to be a high-confidence (valid) rule, at least 60% of the time the antecedent is found in the transaction, the consequent must also be found there).

By the support-confidence framework, we present the two-step association rule mining as follows.

(1) The first step is to count the frequencies of k-itemsets.

For Table 2.1, item $\{A\}$ occurs in two transactions $TID = 100$ and $TID = 300$, its frequency is 2, and its support ($supp(A)$) is 50%, which is equal to $minsupp = 50\%$; item $\{B\}$ occurs in three transactions $TID = 200$, $TID = 300$, and $TID = 400$, its frequency is 3, and its support $supp(B)$ is

75%, which is greater than *minsupp*; item $\{C\}$ occurs in three transactions $TID = 100$, $TID = 200$ and $TID = 300$, its frequency is 3, and its support *supp*(C) is 75%, which is greater than *minsupp*; item $\{D\}$ occurs in one transaction $TID = 100$, its frequency is 1, and its support *supp*(D) is 25%, which is less than *minsupp*; item $\{E\}$ occurs in three transactions $TID = 200$, $TID = 300$, and $TID = 400$, its frequency is 3, and its support *supp*(E) is 75%, which is greater than *minsupp*. They are summarized in Table 2.2.

Table 2.2 *1-itemsets in the database*

Itemsets	*Frequency*	$>$ *minsupp*
$\{A\}$	*2*	*y*
$\{B\}$	*3*	*y*
$\{C\}$	*3*	*y*
$\{D\}$	*1*	*n*
$\{E\}$	*3*	*y*

We now consider 2-itemsets. For Table 2.1, itemset $\{A, B\}$ occurs in one transaction $TID = 300$, its frequency is 1, and its support *supp*$(A \cup B)$ is 25%, which is less than *minsupp* $= 50\%$, where $A \cup B$ is used to stand for $\{A, B\}$ in formulas in this book; itemset $\{A, C\}$ occurs in two transactions $TID = 100$ and $TID = 300$, its frequency is 2, and its support *supp*$(A \cup C)$ is 50%, which is equal to *minsupp* $= 50\%$; itemset $\{A, D\}$ occurs in one transaction $TID = 100$, its frequency is 1, and its support *supp*$(A \cup D)$ is 25%, which is less than *minsupp* $= 50\%$; itemset $\{A, E\}$ occurs in one transaction $TID = 300$, its frequency is 1, and its support *supp*$(A \cup E)$ is 25%, which is less than *minsupp* $= 50\%$; itemset $\{B, C\}$ occurs in two transactions $TID = 200$ and $TID = 300$, its frequency is 2, and its support *supp*$(B \cup C)$ is 50%, which is equal to *minsupp*; and so on. This is summarized in Table 2.3.

Table 2.3 *2-itemsets in the database*

Itemsets	*Frequency*	$>$ *minsupp*
$\{A, B\}$	*1*	*n*
$\{A, C\}$	*2*	*y*
$\{A, D\}$	*1*	*n*
$\{A, E\}$	*1*	*n*
$\{B, C\}$	*2*	*y*
$\{B, E\}$	*3*	*y*
$\{C, D\}$	*1*	*n*
$\{C, E\}$	*2*	*y*

Also, we can obtain 3-itemsets and 4-itemsets as listed in Table 2.4 and Table 2.5.

Table 2.4 *3-itemsets in the database*

Itemsets	*Frequency*	*> minsupp*
$\{A, B, C\}$	*1*	n
$\{A, B, E\}$	*1*	n
$\{A, C, D\}$	*1*	n
$\{A, C, E\}$	*1*	n
$\{B, C, E\}$	*2*	y

Table 2.5 *4-itemsets in the database*

Itemsets	*Frequency*	*> minsupp*
$\{A, B, C, E\}$	*1*	n

Here, 5-itemsets in the database is empty. According to the above definitions, $\{A\}$, $\{B\}$, $\{C\}$, $\{E\}$, $\{A, C\}$, $\{B, C\}$, $\{B, E\}$, $\{C, E\}$, and $\{B, C, E\}$ in the dataset are frequent itemsets.

(2) The second step is to generate all association rules from the frequent itemsets.

Because there is no frequent itemset in Table 2.5, 4-itemsets do not contribute any valid association rule. In Table 2.4, there is one frequent itemset, $\{B, C, E\}$, with $supp(B \cup C \cup E) = 50\% = minsupp$. For frequent itemset $\{B, C, E\}$, because $supp(B \cup C \cup E)/supp(B \cup C) = 2/2 = 100\%$ is greater than $minconf = 60\%$, $B \cup C \to E$ can be extracted as a valid rule; because $supp(B \cup C \cup E)/supp(B \cup E) = 2/3 = 66.7\%$ is greater than $minconf$, $B \cup E \to C$ can be extracted as a valid rule; because $supp(B \cup C \cup E)/supp(C \cup E) = 2/2 = 100\%$ is greater than $minconf$, $C \cup E \to B$ can be extracted as a valid rule; and because $supp(B \cup C \cup E)/supp(B) = 2/3 = 66.7\%$ is greater than $minconf$, $B \to C \cup E$ can be extracted as a valid rule; and so on. The generated association rules from $\{B, C, E\}$ are in Tables 2.6 and 2.7.

Table 2.6 *For frequent 3-itemsets, start with 1-item consequences*

RuleNo	*Rule*	*Confidence*	*support*	$> minconf$
Rule1	$B \cup C \to E$	*100%*	*50%*	y
Rule2	$B \cup E \to C$	*66.7%*	*50%*	y
Rule3	$C \cup E \to B$	*100%*	*50%*	y

Table 2.7 *Form all 2-item consequences from high-conf 1-item consequences*

RuleNo	*Rule*	*Confidence*	*support*	$> minconf$
Rule4	$B \to C \cup E$	*66.7%*	*50%*	y
Rule5	$C \to B \cup E$	*66.7%*	*50%*	y
Rule6	$E \to B \cup C$	*66.7%*	*50%*	y

Also, we can generate all association rules from frequent 2-itemsets as shown in Table 2.3. They are illustrated in Tables 2.8 through 2.11.

Table 2.8 *For 2-itemsets, start with 1-item consequences for* $\{A, C\}$

RuleNo	*Rule*	*Confidence*	*support*	$> minconf$
Rule7	$A \rightarrow C$	*100%*	*50%*	y
Rule8	$C \rightarrow A$	*66.7%*	*50%*	y

Table 2.9 *For 2-itemsets, start with 1-item consequences for* $\{B, C\}$

RuleNo	*Rule*	*Confidence*	*support*	$> minconf$
Rule9	$B \rightarrow C$	*66.7%*	*50%*	y
Rule10	$C \rightarrow B$	*66.7%*	*50%*	y

Table 2.10 *For 2-itemsets, start with 1-item consequences for* $\{B, E\}$

RuleNo	*Rule*	*Confidence*	*support*	$> minconf$
Rule11	$B \rightarrow E$	*100%*	*75%*	y
Rule12	$E \rightarrow B$	*100%*	*75%*	y

Table 2.11 *For 2-itemsets, start with 1-item consequences for* $\{C, E\}$

RuleNo	*Rule*	*Confidence*	*support*	$> minconf$
Rule13	$C \rightarrow E$	*66.7%*	*50%*	y
Rule14	$E \rightarrow C$	*66.7%*	*50%*	y

According to the above definitions, the 14 association rules listed in the above data set can be extracted as valid rules.

2.4 Research into Mining Mono-databases

Association rule mining is one of the most prevailing research topics in mono-database mining. We now briefly introduce some well-known research into mining association rules.

1. The support-confidence framework measures the uncertainty of an association rule with two factors: support and confidence.

However, this measure is inadequate for modeling some uncertainties of association rules. For instance, the measurement does not provide a test to capture the correlation of two itemsets, and some of the association rules mined are not of interest. In order to improve this framework, some measures on the support and confidence of association rules, such as the chi-squared test model (Brin-Motwani-Silverstein 1997) and the collective-strength-based measure (Aggarawal-Yu 1998), have recently been proposed. These different measurements on support and confidence lead to different models for mining association rules. Hence, the measuring of uncertainty of association rules has recently become one of the crucial problems in mining association rules.

In fact, measurement of the uncertainty of an event is a well-known topic. Mathematical probability theory and statistics offer many mature techniques for measuring uncertainty. Thus, there are a great many measuring models that can be applied to estimate the uncertain factors (*supp* and $conf$) of an association rule. Below, we briefly recall some of the familiar methods for measuring association rules, which are relevant to the work in this book.

2. Piatetsky-Shapiro (Piatetsky 1991) argued that a rule $X \to Y$ is not interesting if

$$support(X \to Y) \approx support(X) \times support(Y)$$

where $support(X \to Y) = support(X \cup Y)$.

According to probability interpretation, $support(X \cup Y) = p(X \cup Y)$ and $confidence(X \to Y) = p(Y|X) = p(X \cup Y)/p(X)$. Then Piatetsky-Shapiro's argument can be denoted as

$$p(X \cup Y) \approx p(X)p(Y).$$

This means that $X \to Y$ cannot be extracted as a rule if $p(X \cup Y) \approx p(X)p(Y)$. In fact, in probability theory, $p(X \cup Y) \approx p(X)p(Y)$ denotes X is approximately independent of Y.

3. A statistical definition of (Brin-Motwani-Silverstein 1997) for the dependence of the sets X and Y is

$$Interest(X, Y) = \frac{p(X \cup Y)}{p(X)p(Y)},$$

with the obvious extension to more than two sets. This formula, which we refer to as the *interest* of Y, given X, is one of the main measurements of uncertainty for association rules. Certainly, the further the value is from 1, the more the dependence. Or, for $1 > mininterest > 0$, if $|\frac{p(X \cup Y)}{p(X)p(Y)} - 1| \geq mininterest$, then $X \to Y$ is a rule of interest.

By Piatetsky-Shapiro's argument, we can divide $Interest(X, Y)$ into several cases as follows:

(1) if $p(X \cup Y)/(p(X)p(Y)) = 1$, or $p(X \cup Y) = p(X)p(Y)$, then Y and X are independent;
(2) if $p(X \cup Y)/(p(X)p(Y)) > 1$, or $p(X \cup Y) > p(X)p(Y)$, then Y is positively dependent on X;
(3) if $p(X \cup Y)/(p(X)p(Y)) < 1$, or $p(X \cup Y) < p(X)p(Y)$, then Y is negatively dependent on X, or $\neg Y$ is positively dependent on X.

In this way, we can define another form of interpretation of rules of interest as follows. For $1 > mininterest > 0$, (a) if

$$\frac{p(X \cup Y)}{p(X)p(Y)} - 1 \geq mininterest$$

then $X \Rightarrow Y$ is a rule of interest; and (b) if

$$-(\frac{p(X \cup Y)}{p(X)p(Y)} - 1) \geq mininterest$$

then $X \rightarrow \neg Y$ is a rule of interest, where $\neg Y$ is the logical "not" of Y, or Y is not contained for transactions in a database.

This leads to two new definitions of association rules of interest as follows.

Definition 2.1 (*Piatetsky-Shapiro's argument*) *Let I be the set of items in database TD, $X, Y \subseteq I$ be itemsets, $X \cap Y = \emptyset$, $p(X) \neq 0$, and $p(Y) \neq 0$. $minsupp, minconf$, and $mininterest > 0$ are given by users or experts. Then, $X \rightarrow Y$ can be extracted as a valid rule of interest if*

(1) $p(X \cup Y) \geq minsupp$,
(2) $p(Y|X) \geq minconf$, *and*
(3) $|p(X \cup Y) - p(X)p(Y)| \geq mininterest$.

Definition 2.2 (*Brin, Motwani, and Silverstein's argument*) *Let I be the set of items in database D, $X, Y \subseteq I$ be itemsets, $X \cap Y = \emptyset$, $p(X) \neq 0$, and $p(Y) \neq 0$. The thresholds minimum support ($minsupp$), minimum confidence ($minconf$), and minimum interest ($mininterest > 0$) are given by users or experts. Then, $X \rightarrow Y$ can be extracted as a valid rule of interest if*

(1) $p(X \cup Y) \geq minsupp$,
(2) $p(Y|X) \geq minconf$, *and*
(3) $|\frac{p(X \cup Y)}{p(X)p(Y)} - 1| \geq mininterest$.

Here, condition (3) ensures that $X \rightarrow Y$ is a rule of interest.

According to the above framework, we can take

(1) $X \cap Y = \emptyset$;
(2) $p(X \cup Y) \geq minsupp$;
(3) $p(Y|X) \geq minconf$ (e.g., $conf(X \rightarrow Y) \geq minconf$); and
(4) $|\frac{p(X \cup Y)}{p(X)p(Y)} - 1| \geq mininterest$

as the *conditions* under which association rule $X \rightarrow Y$ can be extracted as a valid rule of interest in this book, where the thresholds *minimum support* ($minsupp$), *minimum confidence* ($minconf$), and *minimum interest* ($mininterest > 0$) are given by users or experts.

4. The J-measure is as

$$J(X;Y) = p(Y)[p(X|Y)\log(\frac{p(X|Y)}{p(X)}) + (1-p(X|Y))\log(\frac{1-p(X|Y)}{1-p(X)})]$$

for a rule $X \to Y$, where the term inside the square brackets is the relative (or cross) entropy. (Relative entropy is the similarity, or goodness of fit, of two probability distributions.)

The J-measure is the average information content of a probabilistic classification rule (Smyth-Goodman 1991). It is used to find the best rules relating to discrete attributes. A probabilistic classification rule is a logical implication $X \to Y$ with some probability p, where the left- and right-hand sides correspond to a single attribute. The right-hand side is restricted to simple, single-valued, assignment expressions, while the left-hand side may be a conjunction of these simple expressions.

High values for $J(X;Y)$ are desirable, but are not necessarily associated with the best rule. For example, rare conditions may be associated with the highest values for $J(X;Y)$ (i.e., where a particular Y is highly unlikely), but the resulting rule is insufficiently general to provide any new information. Consequently, analysis may be required in which the accuracy of a rule is traded for some level of generality or goodness-of-fit.

5. The distance metric between two rules R_1 and R_2 is defined in (Dong-Li 1998) as

$$D(R_1, R_2) = \delta_1 \times |(X_1 \cup Y_1)\Theta(X_2 \cup Y_2)| + \delta_2 \times |X_1 \Theta X_2| + \delta_3 \times |Y_1 \Theta Y_2|,$$

where $R_1 = X_1 \to Y_1$, $R_2 = X_2 \to Y_2$, δ_1, δ_2, and δ_3 are parameters to weight the relative importance of all three terms, and Θ is an operator denoting the symmetric difference between X and Y (i.e., $(X-Y)\cup(Y-X)$).

Dong and Li's interestingness is used to evaluate the importance of an association rule by considering its unexpectedness in terms of other association rules in its neighborhood. The *neighborhood* of an association rule consists of all association rules within a given distance. An r-neighborhood of a rule is given by the set

$$N(R_0, r) = \{R | D(R, R_0) \le r, R \text{ is a potential rule}\}.$$

The set is used to define the interestingness of a rule. Two types of interestingness are unexpected confidence and isolation. *Unexpected confidence interestingness* is defined as

$$UCI = \begin{cases} 1, & \text{if } ||c(R_0) - ac(R_0, r)| - sc(R_0, r)|, \\ 0, & \text{otherwise,} \end{cases}$$

where $c(R_0)$ is the confidence of R_0, $ac(R_0, r)$ are the average confidence and standard deviation of the rules in the set $M \cap N(R_0, r) - \{R_0\}$ (M being the set of rules satisfying the minimum support and confidence), and t_1 is a threshold. *Isolated interestingness* is defined as

$$II = \begin{cases} 1, & \text{if } ||N(R_0, r)| - |M \cap N(R_0, r)| > t_2, \\ 0, & \text{otherwise,} \end{cases}$$

where $|N(R_0, r)|$ is the number of potential rules in an r-neighborhood, $|M \cap N(R_0, r)|$ is the number of rules generated from the neighborhood, and t_2 is a threshold.

6. The ratio $\frac{p(Y|X) - p(Y)}{1 - p(Y)}$ of the conditional probability and the priori probability to describe the increased degree of $p(Y|X)$ relative to $p(Y)$, referred to as the $CPIR$ model, is defined in (Wu-Zhang-Zhang 2002) as

$$CPIR(Y|X) = \frac{supp(X \cup Y) - supp(X)supp(Y)}{supp(X)(1 - supp(Y))}.$$

   $CPIR$ is taken as a metric for the confidence measure $conf$ of the rule $X \Rightarrow Y$ in the following discussion. Here, $supp(X \cup Y) \geq supp(X)supp(Y)$ and $supp(X)(1 - supp(Y)) \neq 0$, where $supp(Y|X) = p(Y|X)$ in the model is replaced with $supp(X \cup Y)/supp(X)$ for the convenience of mining association rules.

While positive association rules that enable us to detect the *companionate* correlations among items are useful in decision-making, it is also desirable in applications to capture the *mutually exclusive* correlations among items. These mutually exclusive correlations are referred to as *negative association rules*, and are hidden in infrequent itemsets. The development of negative association rule mining will mean that companies will gain more business opportunities through using infrequent itemsets of interest than will those that only take into account frequent itemsets.

The interestingness in (Wu-Zhang-Zhang 2002) is used for identifying both positive and negative association rules in databases. This method extends traditional associations to include association rules of the forms $A \Rightarrow \neg B$, $\neg A \Rightarrow B$, and $\neg A \Rightarrow \neg B$, which indicate negative associations between itemsets.

With the increasing use and development of data mining techniques and tools, much work has recently focused on finding alternative patterns, including unexpected patterns (Padmanabhan 1998, 2000), exceptional patterns (Hussain 2000, Hwang 1999, Liu 1999, Suzuki 1996, 1997), and strong negative associations (Savasere 1998).

Unexpected patterns and exceptional patterns are referred to as *exceptions of rules*, also known as *surprising patterns*. An exception, which is defined as a deviational pattern to a well-known fact, exhibits unexpectedness.

For example, while $bird(x) \Rightarrow flies(x)$ is a well-known fact, mining exceptional rules is used to find patterns such as $bird(x), penguin(x) \Rightarrow \neg flies(x)$. This means unexpected patterns and exceptional patterns are default rules as well as not being negative rules.

A strong negative association is referred to as a *negative relation* between two itemsets. This negative relation really implies a negative rule between the two itemsets. However, strong negative association only reveals the existence in a hidden representation. For example, $X \not\Rightarrow Y$ is a strong negative association. Obviously, this rule cannot be used in an automated reasoning system.

Unlike existing mining techniques, the interestingness in (Wu-Zhang-Zhang 2002) extends traditional associations to include association rules of forms $A \Rightarrow \neg B$, $\neg A \Rightarrow B$, and $\neg A \Rightarrow \neg B$, which indicate negative associations between itemsets. We call rules of the form $A \Rightarrow B$ *positive rules*, and rules of the other forms *negative rules*. This work differs from existing work in association analysis in two aspects. Infrequent itemsets in databases are of interest to us for mining negative association rules. The following constraints have been defined for identifying interesting negative rules of the form $A \Rightarrow \neg B$.

(1) $A \cap B = \emptyset$;
(2) $supp(A) \geq minsupp$, $supp(B) \geq minsupp$, and $supp(A \cup \neg B) \geq minsupp$;
(3) $supp(A \cup \neg B) - supp(A)supp(\neg B) \geq mininterest$;
(4) $supp(A \cup \neg B)/supp(A) \geq minconf$.

Also, to design an efficient model for mining both positive and negative association rules, the $CPIR$ model was designed to estimate the confidence of association rules. This uses the increasing degree of the rule's conditional probability relative to its priori probability.

7. An anytime framework for mining very large databases shared by multi-users has been designed by (Zhang-Zhang 2002b). It generates approximate results that can be accessed at any time while the system is autonomously mining a database.

Mining *approximate frequent itemsets* from the sample of a large database can reduce computation costs significantly. For example, we can select a sample from a large database for estimating the support of candidates using Chernoff bounds (Srikant-Agrawal 1997, Toivonen 1996). This technique is effective for *mono-user applications*, where mono-user applications are those that can work well under a unique precision on frequent itemsets. However, *multi-user applications* require different precisions.

In real-world applications, a database is developed to be shared by multi-users. Therefore, data mining must be developed to serve multi-user applications. For a very large database, multi-users might demand different precisions on results for different applications. For example, a short-term stock investor might demand approximate frequent itemsets quickly from a shared stock database for high-profits as time may mean money. A long-term stock investor is more likely to wait for more accurate results.

Using traditional instance-selection (sampling) techniques, one must resample a database multiple times, and mine the selected instance sets for different precisions when the database is very large. For example, consider a very large database TD, shared by five users. For the time/performance tradeoff, the five users require 0.85, 0.90, 0.92, 0.95, and 0.98 precisions when estimating frequent itemsets. Existing instance selection based approaches are efficient for meeting the requirements of a user when identifying approximate frequent itemsets by sampling (Liu 2001-Motoda, Srikant-Agrawal 1997, Toivonen 1996, Zhang-Zhang 2001c). However, for the five different precisions, we need to select five instance sets, and mine them.

Zhang and Zhang's anytime mining framework can support inquiries from multiple users at any time. In this way, users can make tradeoff decisions when they choose, depending upon the required accuracy of results. Our approach is different from traditional mining techniques because it aims at attacking the multi-user application problem.

As there are a great many large databases shared by multi-users, the mining of large databases for serving multi-user applications is a new and pressing topic in data mining research. Because of the essential differences between mining tasks for multi- and mono-user applications, research into the multi-user application problem will have an impact on both industry and academia.

8. Han et al. have proposed a novel frequent pattern mining model, based on the frequent pattern tree (FP-tree) (Han-Pei-Yin 2000).

An FP-tree is a tree structure as defined below.

It consists of one root labeled *null*, a set of item prefix subtrees, which are the children of the root, and a frequent-item header table.

Each node in the item prefix subtree consists of three fields: item-name, count and node-link, where item-name registers which item the particular node represents, count registers the number of transactions represented by the portion of the path reaching the node, and node-link links to the next node in the FP-tree which carries the same item-name; or it is null if there are none.

Each entry in the frequent-item header table consists of two fields, item-name and head of node-link, which points to the first node in the FP-tree carrying the item-name.

The process of the FP-tree-based model is as follows.

First, an FP-tree is constructed, which is an extended prefix-tree structure storing crucial quantitative information about frequent patterns. Only frequent length-1 items will have nodes in the tree, and the tree nodes are arranged in such a way that more frequently occurring nodes will have a better chance of sharing nodes than less frequently occurring ones.

Second, an FP-tree-based pattern fragment growth mining method is developed, which starts from a frequent length-1 pattern (as an initial suffix pattern), and examines only its conditional pattern base (a subdatabase which consists of the set of frequent items co-occurring with the suffix pattern). It then constructs its (conditional) FP-tree, and performs mining recursively with such a tree. The pattern growth is achieved via concatenation of the suffix pattern with the new patterns generated from a conditional FP-tree. Since a frequent itemset in any transaction is always encoded in the corresponding path of the frequent pattern trees, pattern growth ensures the completeness of the result. In this context, the method is not an *Apriori*-like restricted generation-and-test, but a restricted test only. The major operations of mining are count accumulation and prefix path count adjustment, which are usually much less costly than the candidate generation and pattern-matching operations performed in most *Apriori*-like algorithms.

Third, the search technique employed in mining is a partitioning-based, divide-and-conquer method, rather than the *Apriori*-like bottom-up generation of frequent itemset combinations. This dramatically reduces the size of the conditional pattern base generated at the subsequent level of search, as well as the size of its corresponding conditional FP-tree. Moreover, it transforms the problem of finding long frequent patterns into looking for shorter ones, and then concatenating the suffix. It employs the least frequent items as the suffix, which offers good selectivity. All these techniques contribute to a substantial reduction in search costs.

9. Zhang and Zhang have proposed a method for identifying causality between variables X and Y, represented in the form $X \rightarrow Y$ with conditional probability matrix $M_{Y|X}$ (Zhang-Zhang 2002e).

While there has been much work done on discovering item-based association rules and quantitative association rules, some work has focused on mining causal rules in databases (Cooper 1997, Heckerman 1995, Silverstein 1998). Mining models for causality, such as the LCD algorithm (Cooper 1997) and the CU-path algorithm (Silverstein 1998), which are used for constraint-based causal discovery, have been proposed for mining causal relationships in market basket data. In fact, the CU-path algorithm is an improved model of the LCD algorithm, which applies the chi-squared formula to test the dependence, independence, and conditional independence between variables so as to discover the possible causal relationships between those variables.

However, these models are only suitable for mining causal rules among simple variables, such as "*states* $\rightarrow$ *united*" for words in the *clari.world* news hierarchy (Silverstein 1998). They are inadequate for discovering causal rules among multi-value variables in large databases, and for representing them. In fact, mining causality among multi-value variables in many applications, such as decision-making, diagnosis, and planning, is useful for solving application problems.

Another distinct difference with Zhang and Zhang's model is that it offers a method for optimizing conditional probability matrices for causal rules, which merges unnecessary information for the extracted causal rules. Obviously, this model can be used to optimize the knowledge in intelligent systems.

10. Webb has presented an alternative approach to a direct search for association rules for some applications (Webb 2000). This method applies the OPUS search to prune the search space on the basis of interrelationships between itemsets.

In Webb's model, association rule mining is tackled as a search process that starts with general rules (rules with one condition on the left-hand side (LHS)) and searches through successive specializations (rules formed by adding additional conditions to the LHS). Such a search is unordered. That is, the order in which successive specializations are added to an LHS is not significant, and $A \wedge B \wedge C \rightarrow X$ is the same as $C \wedge B \wedge A \rightarrow X$. An important component of an efficient search in this context is minimizing the number of association rules that need to be considered. A key technique used to eliminate potential association rules from consideration is "optimistic pruning".

Optimistic pruning operates by forming an optimistic evaluation of the highest rule value that might occur in a region of the search space. An optimistic evaluation is one that cannot be lower than the actual maximum value. If the optimistic value for the region is lower than the lowest value that is of interest, then that region can be pruned. If a search seeks the top m association rules, then it can maintain a list of the top m rules encountered at that point during the search. If an optimistic evaluation is lower than the lowest value of a rule in the top m, then the corresponding region of the search space may be pruned. Other pruning rules could perhaps identify regions that could be pruned if they contained only rules that failed to meet pre-specified constraints such as:

- minimum support (the frequency in the data of the right-hand side (RHS) or of the RHS and LHS in combination);
- minimum *lift*; or
- being one of the top m association rules on some specified criterion.

Here, $lift$ is a frequently utilized measure of association rule utility. The lift of an association rule $= \frac{|LHS \wedge RHS|}{|LHS|} / \frac{|RHS|}{n}$, where $|X|$ is the number of cases with conditions X, and n is the total number of cases in the dataset.

The term *credible rule* is used to denote association rules for which, at some given point in a search, it is possible that the rule will be of interest, using whatever criterion of interest applies for the given search.

If we restrict association rules to having a single condition on the RHS, these search strategies are plausible:

(1) *for each potential* RHS*, the condition explores the space of possible* LHS *conditions*; or
(2) *for each potential* LHS *combination of conditions, the space of possible* RHS *conditions is explored.*

The former strategy leads to the most straightforward implementation, as it involves a simple iteration through a straightforward search for each potential RHS condition. However, this implies accessing the count of the number of cases covered by the LHS many times, once for each RHS condition for which an LHS is considered. At the very least, this entails computational overheads for caching information. At worst, it requires a pass through the data each time the value is to be utilized. While a pass through the data has lower overheads when the data are stored in memory rather than on disk, it is still a time-consuming operation that must be avoided if computation is to be efficient.

The algorithm, which applies the OPUS search algorithm to obtain an efficient search for association rules, is designed as a recursive procedure with these arguments:

(1) **CurrentLHS**: the set of conditions in the LHS of the rule currently being considered;
(2) **AvailableLHS**: the set of conditions that may be added to the LHS of rules to be explored below this point; and
(3) **AvailableRHS**: the set of conditions that may appear on the RHS of a rule in the search space at this point and below.

This algorithm is computationally efficient for an association rule analysis during which the number of rules to be found can be constrained and all data can be maintained in memory.

A number of efforts on research into efficient algorithms for mining association rules (Agrawal-Srikant 1994, Park-Chen-Yu 1997), measures of itemsets (Aggarawal-Yu 1998), parallel data mining for association rules (Han-Karypis-Kumar 1997), FP-tree-based model (Han-Pei-Yin 2000), and OPUS-based algorithm (Webb 2000), have been reported.

There has also been much work on mining special databases. For example, spatial data mining is the discovery of novel and interesting relationships and characteristics that may exist implicitly in spatial databases (Cai et al. 1991, Ester et al. 1997, Han 1997, Ng 1994); temporal database mining (Chen 1998), image data mining for multi-dimensional remotely sensed images (Cromp 1993); probabilistic database mining (Zhang-Zhang 2004); mining time-series databases (Tsumoto 1999); text mining (Feldman et al. 1999); and Web mining for the discovery and application of usage patterns from Web data (Srivastava 2000).

2.5 Research into Mining Multi-databases

As we have seen, knowledge discovery in databases aims at the discovery of useful information from large collections of data. The discovered knowledge can consist of rules describing properties of the data, frequently occurring patterns, clustering of objects in the database, and so on. These can be used to support various intelligence activities, such as decision-making, planning, and problem-solving.

Recently, it has been recognized in the KDD community that multi-database mining is an important research topic (Zhong-Yao-Ohsuga 1999). So far, most of the KDD methods that have been developed are on the single universal relation level. Although, theoretically, any multi-relational database can be transformed into a single universal relation, in fact this can lead to many complications such as universal relations of unmanageable size, infiltration of uninteresting attributes, the loss of useful relation names, unnecessary join operations, and inconveniences inherent in distributed processing. In particular, certain concepts, regularities, causal relationships, and rules cannot be discovered if we just search a single database, since some basic knowledge hides in multiple databases.

Multi-database mining involves many related topics, including interestingness checking, relevance, database reverse engineering, granular computing, and distributed data mining. For example, Liu et al. have proposed an interesting method for relevance measurement, and an efficient implementation for identifying relevant databases, as the first step for multi-database mining (Liu-Lu-Yao 1998, Yao-Liu 1997). Zhong et al. have proposed a way of mining peculiarity rules from multiple statistical and transaction databases (Zhong-Yao-Ohsuga 1999). Ribeiro et al. have described a method for extending the INLEN system for multi-database mining by the incorporation of primary and foreign keys, as well as the development and processing of knowledge segments (Ribeiro-Kaufman-Kerschberg 1995). Wrobel has extended the concept of foreign keys into foreign links, because multi-database mining is also concerned with getting to non-key attributes (Wrobel 1997). Aronis et al. have introduced a system called WoRLD that uses spreading activation to enable inductive learning from multiple tables in multiple databases spread across the network (Aronis et al. 1997).

In this section, we briefly recall some related work, including the multi-database mining techniques in (Liu-Lu-Yao 1998, Yao-Liu 1997, Zhong-Yao-Ohsuga 1999).

2.5.1 Parallel Data Mining

Due to the size of large databases, and the amount of intensive computation involved in association analysis, parallel and distributed data mining has been a crucial mechanism for large-scale data mining applications. Existing research in this area has focused on the study of the degree of parallelism, synchronization, data locality issues, and optimization techniques necessary for global association computation.

For example, Cheung et al. have proposed some strategies to leverage the skew of association patterns in a distributed database environment, and have offered some optimizations to efficiently generate global frequent sets (Cheung-Ng-Fu-Fu 1996). Their main idea is to use local pruning to support count exchange to achieve efficient association analysis.

As discussed in Chapter 1, there are some limitations in applying parallel and distributed data mining techniques when searching for patterns from multi-databases. However, parallel and distributed data mining can be combined with our synthesizing model by weighting, for very large database mining applications, as we demonstrate in Chapter 6. If each of the databases is still large, we can apply an association rule mining technique upon the paralleling (MARP) algorithm to discover the local associations from each data-source,. We can then analyze local patterns.

2.5.2 Distributed Data Mining

Distributed data mining (DDM) deals with different possibilities of data distribution. A well-known method is hierarchical meta-learning, which has a similar goal of efficiently processing large amounts of data (Chan 1996, Prodromidis-Stolfo 1998, Prodromidis-Chan-Stolfo 2000). Meta-learning starts with a distributed database, or a set of data subsets of an original database, concurrently running a learning algorithm (or different learning algorithms) on each of the subsets. It combines the predictions from classifiers learned from these subsets by recursively learning "combiner" and "arbiter" models in a bottom-up tree manner. The focus of meta-learning is to combine the predictions of learned models from the partitioned data subsets in a parallel and distributed environment.

In addition, Kargupta et al. have built a collective mining technique for distributed data (Kargupta-HS 1997, Kargupta-HSPW 2000, Kargupta-HSJ 2001); and Grossman et al. have established a system, known as Papyrus, for distributed data mining (Grossman-BRMT 2000, Turinsky-Grossman 2001).

However, unlike the mining strategy in this book, meta-learning, collective mining, and Papyrus do not produce a global learning model from classifiers from different data subsets.

Meta-learning Strategy for Mining Multi-databases. Meta-learning is a technique that seeks to compute higher-level classifiers (or classification models), referred to as meta-classifiers, that integrate in a certain basic fashion multiple classifiers, which compute separately over different databases (Prodromidis-Stolfo 1998, Prodromidis-Chan-Stolfo 2000). Meta-learning starts with a distributed database, or a set of data subsets of an original database, concurrently running a learning algorithm (or different learning algorithms) on each of the subsets. It combines the predictions from classifiers learned from these subsets by recursively learning combiner and arbiter models in a bottom-up tree manner. The focus of meta-learning is to combine the predictions of learned models from the partitioned data subsets in a parallel and distributed environment.

Given a set of training examples, that is, $\{(x_1, y_1), ..., (x_n, y_n)\}$, for some unknown function, $y = f(x)$, with each x_i interpreted as a set of attribute (feature) vectors x_i of the form $\{x_{i1}, x_{i2}, ..., x_{ik}\}$, and with each y_i representing the class label associated with each vector ($y_i \in \{y_1, y_2, ..., y_m\}$), the task is to compute a classifier or model $\hat{f}$ that approximates f, and correctly labels any feature vector drawn from the same source as the training set. It is common to refer to the body of knowledge that classifies data with the label y as the concept of class y.

Some of the common representations used for the generated classifiers are decision trees, rules, version spaces, neural networks, distance functions, and probability distributions. In general, these representations are associated with different types of algorithms that extract different types of information from the database. They also provide alternative capabilities besides the common ability to classify unseen exemplars drawn from a certain domain. For example, decision trees are declarative, and thus more comprehensible to humans than weights computed within a neural network architecture. However, both are able to compute concept y, and classify unseen records (examples).

Meta-learning is loosely defined as learning from learned knowledge. In this case, we concentrate on learning from the output of concept learning systems. This is achieved by learning from the predictions of these classifiers on a common validation dataset. Thus, we are interested in the output of the classifiers, not the internal structure and strategies of the learning algorithms themselves. Moreover, in some of the schemes defined, the data presented to the learning algorithms may also be available to the meta-learner. The different stages in a simplified meta-learning scenario are listed below.

1. The classifiers (base classifiers) are trained from the initial (base-level) training sets.
2. Predictions are generated by the learned classifiers on a separate validation set.
3. A meta-level training set is composed from the validation set, and the predictions generated by the classifiers on the validation set.
4. The final classifier (meta-classifier) is trained from the meta-level training set.

In meta-learning, a learning algorithm is used to learn how to integrate the learned classifiers. That is, rather than having a predetermined and fixed integration rule, the integration rule is learned based on the behavior of the trained classifiers.

For example, let x be an instance whose classification we seek, and $C_1(x), C_2(x), ..., C_k(x)$ be the predicted classifications of x from the k base classifiers, C_i, $i = 1, 2, ..., k$. Then, $class(x)$ and $attrvec(x)$ denote the correct classification and attribute vector of example x, respectively.

In the combiner strategy, the predictions of the learned base classifiers on the validation set form the basis of the meta-learner's training set. A composition rule, which varies in different schemes, determines the content of training examples for the meta-learner. From these examples, the meta-learner generates a meta-classifier, that we call a combiner. In classifying an instance, the base classifiers first generate their predictions. Based on the same composition rule, a new instance is generated from the predictions, which is then classified by the combiner. The aim of this strategy is to "coalesce" the predictions from the base classifiers by learning the relationship, or correlation, between these predictions and the correct prediction. A combiner computes a prediction that may be entirely different from any proposed by a base classifier, whereas an arbiter chooses one of the predictions from the base classifiers and the arbiter itself.

Several schemes for the composition rule are evaluated. First, the predictions $C_1(x), C_2(x)$, ..., $C_k(x)$, for each example x in the validation set of examples, E, are generated by the k base classifiers. These predicted classifications are used to form a new set of "meta-level training instances" T, which is used as input to a learning algorithm that computes a combiner. The manner in which T is computed varies, as defined below.

class-combiner: The meta-level training instances consist of the correct classification and predictions; that is, $T = \{(class(x), C_1(x), C_2(x), ..., C_k(x)) \mid x \in E\}$. This "stacking" scheme was also proposed by Wolpert (Wolpert 1992).

class-attribute-combiner: The meta-level training instances are formed as in a class-combiner, with the addition of the attribute vectors; that is, $T = \{(class(x), C_1(x), C_2(x), ..., C_k(x), attrvec(x)) \mid x \in E\}$.

binary-class-combiner: The meta-level training instances are composed in a manner similar to that in the class-combiner scheme, except that each prediction $C_i(x)$ has l binary predictions, $C_{i1}(x)$, ..., $C_{il}(x)$, where l is the number of classes. Each prediction $C_{ij}(x)$ is produced from a binary classifier, which is trained on examples that are labeled with classes j and $\neg j$. In other words, we are using more specialized base classifiers and attempting to learn the correlation between the binary predictions and the correct prediction. For concreteness, $T = \{(class(x), C_{11}(x), ..., C_{1l}(x), C_{21}(x), ..., C_{2l}(x), ..., C_{k1}(x), ..., C_{kl}(x)) \mid x \in E\}$.

These three schemes for the composition rule are defined in the context of forming a training set for the combiner. These composition rules are also used in a similar manner during classification after a combiner has been computed. Given an instance where classification is sought, we first compute the classifications predicted by each of the base classifiers. The composition rule is then applied to generate a single meta-level test instance, which is then classified by the combiner to produce the final predicted class of the original test datum.

Meta-learning improves efficiency by executing in parallel the base-learning processes (each implemented as a distinct serial program) on (possibly disjoint) subsets of the training data set (a data reduction technique). This approach has the advantage, first, of using the same serial code without the time-consuming process of parallelizing it and, second, of learning from small subsets of data that fit in the main memory.

Meta-learning improves predictive performance by combining different learning systems, each having a different inductive bias (e.g., representation, search heuristics, search space). By combining separately learned concepts, meta-learning is expected to derive a higher-level model that explains a large database more accurately than any of the individual learners. Furthermore, meta-learning constitutes a scalable machine-learning method, since it can be generalized to hierarchical, multi-level meta-learning.

Meta-learning is particularly suitable for distributed data mining applications, such as fraud detection in financial information systems. Financial institutions today typically develop custom fraud detection systems targeted to their own asset bases. Recently though, banks have come to search for unified and global approaches that would also involve the periodic sharing with each other of information about attacks.

The key difficulties in this approach are: financial companies avoid sharing their data for a number of competitive and legal reasons; the databases that companies maintain on transaction behavior are huge, and growing rapidly; real-time analysis is highly desirable to update models when new events are detected; and easy distribution of models in a networked environment is essential to maintain up-to-date detection capability. Meta-learning is a general strategy that provides the means of learning how to combine and integrate a number of classifiers, or models, learned separately at different financial institutions. The designed system JAM allows financial institutions to share their models of fraudulent transactions that each computes separately, while not disclosing their own proprietary data.

Determining the optimal set of classifiers for meta-learning is a combinatorial problem. Hence, the objective of pruning is to utilize heuristic methods to search for partially grown meta-classifiers (meta-classifiers with pruned subtrees) that are more efficient and scalable, and at the same time, achieve comparable or better predictive performance results than fully grown (unpruned) meta-classifiers. Two stages of pruning meta-classifiers are: the a priori pruning, or pre-training pruning, and the a posteriori pruning, or post-training pruning, stages. Both levels are essential, and complementary to each other, with respect to the improvement of accuracy and efficiency of the system.

A priori pruning, or pre-training pruning, refers to the filtering of the classifiers before they are combined. Instead of combining classifiers in a brute force manner, with pre-training pruning we introduce a preliminary stage for analyzing the available classifiers and qualifying them for inclusion in a combined meta-classifier. Only those classifiers that appear, according to one or more pre-defined metrics, to be most "promising" participate in the final meta-classifier. Here, we adopt a "black-box" approach, which evaluates the set of classifiers based only on their input and output behavior, not their internal structure. Conversely, a posteriori pruning, or post-training pruning, denotes the evaluation and pruning of constituent base classifiers after a complete meta-classifier has been constructed.

Parallel Data Mining Agent (PADMA). PADMA is an agent-based architecture for parallel/distributed data mining (Kargupta-HS 1997). The PADMA system attempts to develop a flexible system that will exploit data mining agents from a special application. Although PADMA is not specialized for any particular kind of data mining domain, the current implementation uses agents specializing in unstructured text document classification. The main structural components of PADMA are:

(1) data mining agents,
(2) facilitator for coordinating the agents, and
(3) Web- based user interface.

Data mining agents are responsible for accessing data and extracting higher level useful information from the data. A data mining agent specializes in performing some activity in the domain of interest. In the current implementation, data mining agents specialize in text classification. Agents work in parallel and share their information through the facilitator. The facilitator module coordinates the agents, presents information to the user interface, and provides feedback to the agents from the user.

The PADMA has a graphical, Web-based, interface for presenting information extracted by the agents for the user. The facilitator accepts queries from the user interface in standard SQL (Structured Query Language) format; and the queries are broadcast to the agents. Agents come up with the extracted information relevant to the query. The facilitator collects the information and presents it to the user.

The PADMA model has demonstrated that agent-based data mining tools are suitable for exploiting the benefits of parallel computing. The PADMA model presents some distinct characteristics as follows:

(1) parallel query processing and data accessing,
(2) parallel data analysis, and
(3) interactive data/cluster visualization.

Collective Data Mining (CDM). Collective data mining offers a framework for distributed data modeling and knowledge discovery (Kargupta-HSPW 2000). It draws its motivations from theories of communication and blends them with our existing understanding of statistics and machine learning. This merger has evolved into an interdisciplinary framework for designing and implementing efficient algorithms that generate models from heterogeneous and distributed data with guaranteed global correctness of the model.

The CDM model makes use of an appropriate set of *orthonormal* basis functions, and computes the basis coefficients to generate a global model of the data. Basis functions are chosen to be orthonormal, since the orthonormality property can be exploited for generating correct, unambiguous local basis coefficients. Computing the basis coefficients requires computation of the basis functions of the different domain features. The CDM model distributes the task of approximate computation of the basis coefficient among different sites, using the decomposition outlined below.

1. It generates the coefficients that can be computed, using only the locally available feature data.
2. It computes the coefficients, corresponding to the basis functions that require features from different sites, using the locally generated basis coefficients and a small dataset collected from different sites.

The main steps of the CDM are:

1. generate approximate orthonormal basis coefficients at each local site;
2. move an appropriately chosen sample of the datasets from each site to a single site, and generate the approximate basis coefficients corresponding to nonlinear cross terms; and
3. combine the local models, transforming the model into the user-described canonical representation, and then output the model.

The development of different CDM-based, and other distributed data analysis algorithms, are listed below:

(a) collective decision rule learning using Fourier analysis;
(b) collective hierarchical clustering;
(c) collective multivariate regression using wavelets; and
(d) collective principal component analysis.

For example, given distributed heterogeneous data sites, we show how to approximate the results of the global Principal Component Analysis (PCA), namely, a certain number of dominant eigenvalues/eigenvectors with minimal data communication, as follows.

1. Centralized PCA from distributed heterogeneous datasets typically involves moving all the data to one single site and computing the eigenvalues and eigenvectors of the covariance matrix of the combined dataset.
2. Collective PCA analyzes each data partition at a site and decomposes it as the sum of the product of a certain number of score vectors and the transpose of corresponding loading vectors. The number of the

score/loading vectors is an empirical value that could be changed for different accuracy requirements. This decomposition can also be presented as the product of a score matrix (whose column vectors are score vectors) and the transpose of a loading matrix (whose column vectors are loading vectors). Then, the score matrix is sampled. That is, a certain number of rows are randomly chosen from it. The chosen rows, and the complete loading matrix, comprise the data which need to be moved, and the size of these data is far less than that of the whole data partition.

3. Once the loading matrices and sampled score matrices of all local sites are generated, they are moved to one single site, and the approximated global covariance matrix is constructed. PCA is then applied to this matrix to obtain the approximate PCA results.

2.5.3 Application-dependent Database Selection

For multi-database mining, Yao and Liu have proposed an approach to search for interesting knowledge in multiple databases, according to a user's query (Yao-Liu 1997). This process involves selecting all interesting information from many databases by retrieval. Mining only works on the selected data. Liu et al.have also proposed a mining technique that identifies relevant databases. Their work has been focused on the first step of multi-database mining, which is to identify databases that are most likely relevant to an application (Liu-Lu-Yao 1998). Thus, a relevance measure was proposed to identify relevant databases for mining tasks, with the objective of finding patterns, or regularities, in certain attributes. We briefly recall the work below. For more detail, please see (Liu-Lu-Yao 1998, Yao-Liu 1997).

A *relop* C is called a *selector*, where A is an attribute name that is not referenced by the query predicate Q, *relop* $\in \{=, <, >, \leq, \geq, \neq\}$, and C is a constant value in the domain of A. The *relevance factor* of selector s with respect to Q is

$$RF(s, Q) = Pr(s|Q)Pr(Q)\log\frac{Pr(s|Q)}{Pr(s)},$$

where $Pr(Q)$ and $Pr(s)$ are prior probabilities and are estimated by the ratios of how frequently they appear in a database; and $Pr(s|Q)$ is the posterior of the frequency ratio of s appearing, given that Q occurs. The rationale of defining relevance is as follows.

$Pr(s|Q)/Pr(s)$ shows the degree of deviation of the posterior from the prior. This ratio tells us the following different relationships between $Pr(s|Q)$ and $Pr(s)$.

Case 1 If $\frac{Pr(s|Q)}{Pr(s)}$ is close to 1 (i.e., $Pr(s|Q) \approx Pr(s)$), s is independent of Q;

Case 2 If $\frac{Pr(s|Q)}{Pr(s)}$ is close to 0 (i.e., $Pr(s|Q)$ is almost 0), s rarely occurs given Q;

Case 3 If $\frac{Pr(s|Q)}{Pr(s)}$ is less than 1, s is not frequent enough when using $Pr(Q)$ as a reference;

Case 4 If $\frac{Pr(s|Q)}{Pr(s)}$ is greater than 1, then s occurs more often given Q than without Q. Hence, s and Q are correlated.

With the above definition for the relevance factor of selectors, we can have a definition of the relevance of databases as follows:

- a selector s is relevant to Q if $RF(s, Q) > \delta$, where δ (> 0) is the given threshold;
- a table is relevant to Q if there exists at least one selector s_j $(A_i\ relop\ C)$, where s_j is relevant to Q.

With the relevance factor RF, we can determine whether a database is relevant to an application (such as for a query predicate) before applying data mining algorithms. To compute $RF(s, Q)$, we need only to count three values, $Pr(Q)$, $Pr(s)$, and $Pr(s \wedge Q)$. To determine whether a database is relevant to Q, we need to test all selectors, which can be done by scanning the table once. Let us assume that there are $m+1$ attributes, A_0, ..., A_m. Attribute A_0 is referenced by Q. For each of the other attribute, A_i $(1 \leq i \leq m)$, a table S_i is maintained to keep track of selectors and related counters. An entry of S_i is a triple of $(S_{value}, S_{counter}, SQ_{counter})$, where S_{value} is the value of the selector, $S_{counter}$ records the number of tuples for which $A_i = S_{value}$ is true, and $SQ_{counter}$ records the number of tuples for which both Q and $A_i = S_{value}$ are true. With the S_i table, we can determine the relevance of a database by scanning it once. This process has specific parts: (1) reading each record in the database; and (2) searching for the entry of selectors and updating the counters for each attribute in the record. The cost of part (1) is proportional to the number of records in the database. As for part (2), with a proper data structure, for example, using a hashing function for the selector tables, the search for selector entries can be kept constant, regardless of the number of selectors of an attribute. Therefore, the run-time for the overall calculation is $O(NM)$, where V is the number of records in the database and M is the number of attributes.

Identifying relevant databases is typically *application-dependent*. It has to be carried out multiple times to identify relevant databases for two or more real-world applications. It should be noted that, when users need to mine their multi-databases without reference to any specific application, the technique does not work well.

2.5.4 Peculiarity-oriented Multi-database Mining

Zhong et al. have proposed a way of mining peculiarity rules from multiple statistical and transaction databases (Zhong-Yao-Ohsuga 1999). A peculiarity rule is discovered from peculiar data by searching for the relevance among those data. Roughly speaking, data are peculiar if they represent a peculiar case described by a relatively small number of objects and are very different from other objects in a dataset. Although it looks like an exception rule, because it describes a relatively small number of objects, the peculiarity rule represents a well-known common-sense fact, which is a feature of the general rule. To find peculiar data, an attribute-oriented method was proposed as follows.

Let $X = \{x_1, x_2, ..., x_n\}$ be a dataset related to an attribute in a relation, where n is the number of different values in the attribute. The peculiarity of x_i can be evaluated by the *Peculiarity Factor* $PF(x_i)$,

$$PF(x_i) = \sum_{j=1}^{n} \sqrt{N(x_i, x_j)}.$$

It evaluates whether x_i occurs as a relatively small number and is very different from the other data x_j by calculating the sum of the square root of the conceptual distance between x_i and x_j. The reason why the square root is used in the peculiarity factor is that we prefer to evaluate closer distances for relatively large amounts of data so that peculiar data can be found from relatively small amounts of data.

The major merits of the method are: (1) it can handle both continuous and symbolic attributes based on a unified semantic interpretation, and (2) background knowledge represented by binary neighborhoods can be used to evaluate the peculiarity if such background knowledge is provided by a user.

If X is the dataset of a continuous attribute, and no background knowledge is available, then

$$N(x_i, x_j) = |x_i - x_j|.$$

On the other hand, if X is a data set of a symbolic attribute, and/or the background knowledge for representing the conceptual distances between x_i and x_j is provided by a user, the peculiarity factor is calculated by the conceptual distances, $N(x_i, x_j)$.

After evaluation for peculiarity, the peculiar data are elicited by using a threshold value

$$\textit{threshold} = \text{ mean of } PF(x_i) + \alpha \times \text{ variance of } PF(x_i),$$

where α can be specified by a user. That is, if $PF(x_i)$ is over the threshold value, x_i is peculiar data.

Because a peculiarity rule is discovered from the peculiar data by searching for the relevance among those data, a measurement for relevant databases is also proposed as follows.

Let $X(x)$ and $Y(y)$ be the peculiar data found in two attributes X and Y, respectively. We deal with the following two cases.

– If $X(x)$ and $Y(y)$ are found in a relation, the relevance between $X(x)$ and $Y(y)$ is evaluated in the following formula,

$$R_1 = P_1(X(x)|Y(y))P_2(Y(y)|X(x)).$$

That is, the larger the product of the probabilities of P_1 and P_2, the stronger the relevance between $X(x)$ and $Y(y)$.

– If $X(x)$ and $Y(y)$ are in two different relations, we need to use a value (or its granule) in a key (or foreign key/link) as the relevance factor, $K(k)$, to find the relevance between $X(x)$ and $Y(y)$. Thus, the relevance between $X(x)$ and $Y(y)$ is evaluated in the following formula,

$$R_2 = P_1(K(k)|X(x))P_2(K(k)|Y(y)).$$

Furthermore, the above two formulae are suitable for handling more than two lots of peculiar data, found in more than two attributes, if $X(x)$ (or $Y(y)$) is a granule of the peculiar data.

Although this work can identify new kinds of patterns in multi-databases, it still utilizes techniques already used in mono-database mining.

From the above efforts on multi-database mining, we can see that existing techniques are limited by mono-database mining techniques. Thus, as we mentioned in Chapter 1, there are still some limitations in traditional multi-database mining methods.

2.6 Summary

Due to the increasingly large number of multi-database systems, multi-database mining has become very important. Although, theoretically, any multi-relational database can be transformed into a single universal relation, in fact this can lead to many extra problems, such as universal relations of unmanageable size, infiltration of uninteresting attributes, the loss of useful relation names, unnecessary join operations, and the inconvenience for distributed processing (Zhong-Yao-Ohsuga 1999). Also, some concepts, such as regularity, causal relationships, and rules cannot be discovered if we simply search a single database, since the basic knowledge can be hidden in multiple databases. Thus, dual-level applications present more challenges than those faced by mono-database mining.

As an introduction to this book, we have described KDD techniques, existing research into multi-database mining and necessary basic concepts. From Chapter 3 on, we present techniques in multi-database mining.