# A Meta-Mining Approach to Discovering Regularities, Differences, and Changes in Databases

by

Wai-Ho Au

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY
in
Department of Computing
The Hong Kong Polytechnic University

June 2005

CERTIFICATE OF ORIGINALITY

I hereby declare that this thesis is my own work and that, to the best of my knowledge and belief, it reproduces no material previously published or written, nor material that has been accepted for the award of any other degree or diploma, except where due acknowledgement has been made in the text.

_____ (Signed)

Wai-Ho Au (Name of Student)

# Abstract

We propose to mine a set of rules from a collection of rule sets, each rule being discovered in a data set using a data mining algorithm. These *meta-rules*, rules about rules, represent the kind of knowledge that few existing data mining algorithms have been developed to mine for. In this study, we define problems in discovering the underlying regularities, differences, and changes hidden in rule sets and propose a new approach, *meta-mining*, which mines previous data mining results to discover these underlying regularities, differences, and changes.

The purpose of meta-mining for regularities and for differences in rule sets is to discover association relationships. Meta-mining for regularities seeks to discover association relationships supported by a sufficiently large number of rules contained in just a few records in many data sets. Meta-mining for differences seeks to discover association relationships supported by a sufficiently small number of rules contained in many records in a small number of data sets. It would not be possible to distinguish between these two kinds of association relationships if the data sets were concatenated into a single data set. The associations that a large number of data sets have in common can be discovered in the form of rules. Their rule sets will contain a correspondingly large number of rules that support the associations. As these rules govern regular characteristics in the data sets, we refer to the rules for these rules as *regular meta-rules*. In contrast, the rules for some associations will be found in just a few data sets and their rule sets will contain a correspondingly smaller number of rules that support the associations. As these associations contribute to distinguishing or differentiating the data sets which contain them, we refer to the rules for these rules as *differential meta-rules*.

Meta-mining can also be used to reveal changes in rule sets and this information can be used to discover *change meta-rules*, regularities governing how rules change over time. Change meta-rules can be used to predict how the rules will change in the future, freeing users from dependence on the historical data, allowing better planning, and making it possible to obviate or delay undesirable change.

A meta-mining approach to the discovery of regular, differential, and change meta-rules should be able to 1) automatically generate fuzzy sets from data; 2) use linguistic variables and linguistic terms to represent regularities, differences, and changes; 3) exploit the scalability of parallel computer systems; 4) group and select a subset of attributes; and 5)

enable the mining of association relationships involving attributes that were not originally contained in the data.

To generate fuzzy sets directly from data, we present a new fuzzy partitioning method to maximize the class-attribute interdependence, thereby improving the classification results. This method uses an information-theoretic measure to evaluate the interdependence between the class and an attribute.

So that association relationships can be represented using easily-understood linguistic variables and terms, we propose new algorithms for mining fuzzy rules and meta-rules. These utilize an objective measure to discover interesting associations among attributes without the need for a user to supply any thresholds. We also extend these new algorithms to exploit the scalability of parallel systems so as to handle very large data sets and rule sets. The parallel algorithms produce the same results as their serial counterparts in a fraction of the time.

We also define the problem of attribute clustering and introduce a methodology for solving it. Our proposed method groups interdependent attributes into clusters by optimizing a criterion function derived from an information measure that reflects the interdependence between attributes. The partitioning of a relational table into attribute subgroups allows a small number of attributes within or across the groups to be selected for analysis. Clustering attributes reduces the search dimension of a mining algorithm.

To allow the discovery of association relationships involving attributes that are not originally contained in the data, we introduce the concept of using transformation functions and propose a formal approach to this problem. This approach can also handle the union of relational and transactional data stored in a relational database.

In this study, we also tested our proposed techniques with extensive experiments on many synthetic and real-world data sets. The results show that they are very effective in mining not just rules from data sets, but also meta-rules from rule sets.

# List of Publications Arising from the Thesis

- **W.-H. Au** and K. C. C. Chan, "Classification with Degree of Membership: A Fuzzy Approach," in *Proc. of the 1st IEEE Int'l Conf. on Data Mining*, San Jose, CA, 2001, pp. 35–42.
- **W.-H. Au** and K. C. C. Chan, "An Evolutionary Approach for Discovering Changing Patterns in Historical Data," in B. V. Dasarathy (Ed.), *Data Mining and Knowledge Discovery: Theory, Tools, and Technology IV*, Proc. of SPIE Vol. 4730, 2002, pp. 398–409.
- **W.-H. Au** and K. C. C. Chan, "Fuzzy Data Mining for Discovering Changes in Association Rules over Time," in *Proc. of the 11th IEEE Int'l Conf. on Fuzzy Systems*, Honolulu, HI, 2002, pp. 890–895.
- **W.-H. Au** and K. C. C. Chan, "Mining Fuzzy Association Rules in a Bank-Account Database," *IEEE Trans. on Fuzzy Systems*, vol. 11, no. 2, pp. 238–248, 2003.
- **W.-H. Au** and K. C. C. Chan, "Mining Fuzzy Rules for Time Series Classification," in *Proc. of the 13th IEEE Int'l Conf. on Fuzzy Systems*, Budapest, Hungary, 2004, pp. 239–244.
- **W.-H. Au** and K. C. C. Chan, "Mining Changes in Association Rules: A Fuzzy Approach," *Fuzzy Sets and Systems*, vol. 149, no. 1, pp. 87–104, 2005.
- **W.-H. Au**, K. C. C. Chan, and A. K. C. Wong, "A Fuzzy Approach to Partitioning Continuous Attributes for Classification," to appear in *IEEE Trans. on Knowledge and Data Engineering*.
- **W.-H. Au**, K. C. C. Chan, A. K. C. Wong, and Y. Wang, "Attribute Clustering for Grouping, Selection, and Classification of Gene Expression Data," *IEEE/ACM Trans. on Computational Biology and Bioinformatics*, vol. 2, no. 2, pp. 83–101, 2005.
- **W.-H. Au**, K. C. C. Chan, and X. Yao, "A Novel Evolutionary Data Mining Algorithm with Applications to Churn Prediction," *IEEE Trans. on Evolutionary Computation*, vol. 7, no. 6, pp. 532–545, 2003.
- K. C. C. Chan and **W.-H. Au**, "Mining Fuzzy Association Rules in a Database Containing Relational and Transactional Data," in A. Kandel, M. Last, and H. Bunke (Eds.), *Data Mining and Computational Intelligence*, New York, NY: Physica-Verlag, 2001, pp. 95–114.
- K. C. C. Chan, **W.-H. Au**, and B. Choi, "Mining Fuzzy Rules in a Donor Database for Direct Marketing by a Charitable Organization," in *Proc. of the 1st IEEE Int'l Conf. on Cognitive Informatics*, Calgary, Alberta, Canada, 2002, pp. 239–246.

# Acknowledgements

First of all, I thank my advisor, Prof. Keith C. C. Chan, for the continuous support and the constructive advice. The experience obtained by working with him in the past few years is definitely invaluable.

Next, I present my sincere thank-you to Prof. Andrew K. C. Wong of the Department of Systems Design Engineering, University of Waterloo, Waterloo, Ontario, Canada for the discussion of many interesting research ideas. His enthusiasm in research has inspired me to become an outstanding researcher.

I thank Prof. Xin Yao of the School of Computer Science, The University of Birmingham, Edgbaston, Birmingham, U.K. for working on the paper on genetic algorithms and churn prediction, and Dr. Yang Wang of the Pattern Discovery Software Systems, Ltd., Waterloo, Ontario, Canada for working on the paper on attribute clustering.

I also thank my examiners for their valuable comments. They are Prof. Qin Lu of the Department of Computing, The Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong, Prof. Witold Pedrycz of the Department of Electrical and Computer Engineering, University of Alberta, Edmonton, Alberta, Canada, and Prof. Xindong Wu of the Department of Computer Science, University of Vermont, Burlington, Vermont, USA.

Lastly, but certainly not the least, I thank my wife for encouraging me to finish the doctoral study. I also thank my parents. Without my wife's and my parents' encouragement, I would not be able to make the difficult decision to relocate to Seattle, Washington, USA.

# Table of Contents

# List of Figures

# List of Tables

# Chapter 1
# Introduction

*Data mining* is concerned with the nontrivial extraction of implicit, previously unknown, and potentially useful information from data [Frawley, Piatetsky-Shapiro, and Matheus 1991]. It involves the search for patterns of interest in a particular representational form or in a set of such representations (e.g., decision trees, association rules) [Fayyad, Piatetsky-Shapiro, and Smyth 1996].

Data mining is also an important step in what is called *knowledge discovery in databases* (KDD) [Fayyad, Piatetsky-Shapiro, and Smyth 1996] and, indeed, many researchers use the term data mining to mean KDD (e.g., [Agrawal *et al.* 1996; Han *et al.* 1996; Imielinski, Virmani, and Abdulghani 1996; Silberschatz, Stonebraker, and Ullman 1996]). In this thesis, we use data mining as a synonym for KDD.

To quote from [Matheus, Chan, and Piatetsky-Shapiro 1993], "the grand challenge of data mining is to collectively handle the problems imposed by the nature of real-world databases, which tend to be dynamic, incomplete, redundant, noisy, sparse, and very large." Many interesting studies of data mining have been carried out, drawing upon methods, algorithms, and techniques from fields as diverse as machine learning, pattern recognition, database systems, statistics, artificial intelligence, knowledge acquisition, and data visualization (see, e.g., [Fayyad *et al.* 1996; Piatetsky-Shapiro and Frawley 1991]).

Data mining techniques can be classified according to the kind of knowledge they mine for. The mining of *association rules* aims at discovering interesting relationships or associations among different attribute values [Agrawal, Imielinski, and Swami 1993b; Agrawal and Shafer 1996; Agrawal and Srikant 1994; Cheung *et al.* 1996a; Han and Fu 1995; Houtsma and Swami 1995; Mannila, Toivonen, and Verkamo 1994; Park, Chen, and Yu 1995a, 1995b; Savasere, Omiecinski, and Navathe 1995; Srikant and Agrawal 1995, 1996]. A *Boolean association rule* involves binary attributes; a *generalized association rule* involves attributes that are hierarchically related; a *quantitative association rule* involves attributes that can take on quantitative or categorical values. An example of an association rule is "90% of transactions that contain bread also contain butter; 3% of all transactions contain both of these items." The 90% is referred to as the *confidence* and the 3%, the *support*, of the rule. The discovered association rules can be used later for human examination and machine inference, e.g., classification [Liu, Hsu, and Ma 1998].

*Classification* is another important topic in data mining research [Agrawal *et al.* 1992; Agrawal, Imielinski, and Swami 1993a; Lu, Setiono, and Liu 1995; Mehta, Agrawal, and Rissanen 1996; Shafer, Agrawal, and Mehta 1996]. Classification involves finding a classification model or a *classifier* which can classify data records into different predefined classes. This requires a set of records to be used in training which are classified by reference to an attribute which allows records in the training set to be classified by domain experts. If a data mining technique is a good one, it should be possible to construct a classifier that can for classify records using other attribute values not originally in the training set. The classification problem has been studied extensively in the area of *supervised learning* by machine learning and pattern recognition researchers and various techniques have been proposed to solve it [Michie, Spiegelhalter, and Taylor 1994].

*Clustering* is the process of grouping a set of records into *clusters* [Bradley, Fayyad, and Reina 1998; Cheeseman and Stutz 1996; Ganti *et al.* 1999b; Zhang, Ramakrishnan, and Livny 1996]. Unlike in classification, the class label of each record is not known. Data clusters can be discovered from a set of records based on their attribute values by maximizing the intraclass and minimizing the interclass similarities. Common features of data records in the same cluster can then be identified and used to derive a set of rules which serves as a description of that cluster [Jain, Murty, and Flynn 1999].

Regardless of whether a data mining algorithm is developed for association rule mining, classification, or clustering, its application to a data set typically results in a set of production (if-then) rules [Agrawal *et al.* 1992; Agrawal, Imielinski, and Swami 1993a, 1993b; Agrawal and Shafer 1996; Agrawal and Srikant 1994; Bradley, Fayyad, and Reina 1998; Cheeseman and Stutz 1996; Cheung *et al.* 1996a; Ganti *et al.* 1999b; Han and Fu 1995; Houtsma and Swami 1995; Lu, Setiono, and Liu 1995; Mannila, Toivonen, and Verkamo 1994; Mehta, Agrawal, and Rissanen 1996; Park, Chen, and Yu 1995a, 1995b; Savasere, Omiecinski, and Navathe 1995; Shafer, Agrawal, and Mehta 1996; Srikant and Agrawal 1995, 1996; Zhang, Ramakrishnan, and Livny 1996]. It is for this reason that we focus on the mining tasks in rule sets.

## 1.1 The Problem

This thesis contributes to the problem definitions of mining the underlying regularities, differences, and changes hidden in rule sets and the introduction of a new approach to dealing with the problems.

Given a collection of rule sets discovered by existing data mining techniques (e.g.,

[Agrawal *et al.* 1992; Agrawal, Imielinski, and Swami 1993a, 1993b; Agrawal and Shafer 1996; Agrawal and Srikant 1994; Cheung *et al.* 1996a; Han and Fu 1995; Houtsma and Swami 1995; Lu, Setiono, and Liu 1995; Mannila, Toivonen, and Verkamo 1994; Mehta, Agrawal, and Rissanen 1996; Park, Chen, and Yu 1995a, 1995b; Savasere, Omiecinski, and Navathe 1995; Shafer, Agrawal, and Mehta 1996; Srikant and Agrawal 1995, 1996]), we propose a *meta-mining* approach to discovering a set of rules in the rule sets. These rules are called *meta-rules* because they are rules about rules.

The meta-mining approach, which is composed of a collection of techniques, enables the discovery of patterns that existing data mining techniques have not been developed to mine for. These patterns are regularities, differences, and changes in the underlying patterns hidden in databases. Sections 1.1.1–1.1.3 describes these features more fully and explains the importance of mining them. Section 1.1.4 presents the proposed meta-mining techniques and, again, explains their importance.

### *1.1.1 Mining Regularities in Rule Sets*

Meta-mining is able to discover the underlying regularities hidden in rule sets. Let us take as an example an interstate or international company. It consists of a number of offices at different geographical locations and each office (or group of offices) maintains its own database [Bright, Hurson, and Pakzad 1992]. In general, local decisions are made at the branches of the international company, whereas global decisions are made at the head office and the branches contribute to these decisions in various ways. To facilitate effective decision making in such an environment, many international companies need to mine multiple data sets throughout their branches [Zhang, Wu, and Zhang 2003; Zhang, Zhang, and Wu 2004]. To do so, one can extract relevant data from multiple data sets to amass a single data set and apply existing data mining techniques (e.g., [Agrawal *et al.* 1992; Agrawal, Imielinski, and Swami 1993a, 1993b; Agrawal and Shafer 1996; Agrawal and Srikant 1994; Bradley, Fayyad, and Reina 1998; Cheeseman and Stutz 1996; Cheung *et al.* 1996a; Ganti *et al.* 1999b; Han and Fu 1995; Houtsma and Swami 1995; Lu, Setiono, and Liu 1995; Mannila, Toivonen, and Verkamo 1994; Mehta, Agrawal, and Rissanen 1996; Park, Chen, and Yu 1995a, 1995b; Savasere, Omiecinski, and Navathe 1995; Shafer, Agrawal, and Mehta 1996; Srikant and Agrawal 1995, 1996; Zhang, Ramakrishnan, and Livny 1996]) to the single data set [Liu, Lu, and Yao 1998; Ribeiro, Kaufman, and Kerschberg 1995; Wrobel 1997; Yao and Liu 1997; Zhong, Yao, and Ohsuga 1999].

However, this approach is unable to distinguish the relationships supported by a number of tuples in many data sets from those supported by many tuples in only a few data

sets. For example, a data mining algorithm may discover a rule stating that "if a customer is married and middle-aged, then he/she gets a home mortgage." This rule may be supported by many tuples in the data sets in only one or two branches. The decisions made by the head office based on this rule may therefore be good for these one or two branches; but they may not be beneficial or may even be harmful to the company as a whole.

To discover the regularities in common in the branches' data sets, we proposed to use a meta-mining approach. Given the rule sets discovered in the data sets, it mines a set of meta-rules from them. These meta-rules represent the regularities hidden in the rule sets, which in turn reflect the regularities embedded in the data sets. Based on the meta-rules discovered, the head office can better make global decisions that are beneficial to the whole company.

Realistically, the meta-mining of regularities in rule sets is not limited to use in international companies. Any public or private organization that maintains a collection of data sets or a data set with implicit groupings in terms of geographical locations, time periods, etc. can benefit from meta-mining. For example, meta-mining techniques can be applied to the rule sets discovered from the data sets collected in different outlets operated by a supermarket chain, different shops operated by an apparel retailer, or different post offices or public libraries operated by a government.

Example 1.1 shows how meta-rules can represent the underlying regularities hidden in rule sets and how an organization can use the discovered relationships to better make decisions.

**Example 1.1** Let us consider a supermarket chain, which operates five outlets, $S_1$, …, $S_5$, at different geographical locations. Let us suppose that rule sets $R_1$, …, $R_5$ contain the association rules[1] discovered in the transaction data sets collected in outlets $S_1$, …, $S_5$, respectively. The rule sets are given in the following:

$$R_1: \{i_1, i_2\} \Rightarrow \{i_3\}$$
$$\{i_4\} \Rightarrow \{i_1\}$$

$$R_2: \{i_1, i_2\} \Rightarrow \{i_3\}$$
$$\{i_2, i_3, i_5\} \Rightarrow \{i_4\}$$

[1] An association rule is a production (if-then) rule associated with support and confidence as its interestingness measures.

$$\{i_2, i_3\} \Rightarrow \{i_4\}$$

$$R_3\text{: } \{i_2, i_3, i_5\} \Rightarrow \{i_4\}$$

$$R_4\text{: } \{i_1, i_2\} \Rightarrow \{i_3\}$$
$$\{i_2, i_3, i_5\} \Rightarrow \{i_4\}$$

$$R_5\text{: } \{i_1, i_2\} \Rightarrow \{i_3\},$$

where $i_1$, …, $i_5$ are items.

Rule $\{i_1, i_2\} \Rightarrow \{i_3\}$ is found in four out of the five rule sets. This rule states that "if a customer purchases items $i_1$ and $i_2$, then he/she also purchases $i_3$." It holds in all the outlets except $S_3$. A meta-rule discovered in the rule sets would be:

$$\{i_1, i_2\} \Rightarrow \{i_3\}.$$

This meta-rule states that "in general, if a customer purchases $i_1$ and $i_2$, then he/she also purchases $i_3$." The difference between the rule and the meta-rule is that the former represents a relationship that holds in only an outlet and provides no information about whether it holds in any other outlets, whereas the latter represents a relationship that holds in the outlets in general. Based on this meta-rule, the supermarket chain may like to bundle $i_1$ and $i_2$ together in its outlets to increase the sales of $i_3$. Although this decision would not affect the revenue of outlet $S_3$, it may significantly increase the revenue of the supermarket chain as a whole.

Another meta-rule discovered in the rule sets would be:

$$\{i_2, i_3\} \Rightarrow \{i_4\}.$$

It states that "in general, if a customer purchases $i_2$ and $i_3$, then he/she also purchases $i_4$." This meta-rule is supported by the following rules in $R_2$, $R_3$, and $R_4$:

$$R_2\text{: } \{i_2, i_3, i_5\} \Rightarrow \{i_4\}$$
$$\{i_2, i_3\} \Rightarrow \{i_4\}$$

$$R_3\text{: } \{i_2, i_3, i_5\} \Rightarrow \{i_4\}$$

$$R_4\text{: } \{i_2, i_3, i_5\} \Rightarrow \{i_4\}.$$

Although rule $\{i_2, i_3\} \Rightarrow \{i_4\}$ is not found in any of the five rule sets, such a relationship can be revealed by mining them for meta-rules. □

### 1.1.2 Mining Differences in Rule Sets

Discovered meta-rules can also represent the differences in rules sets. A meta-rule is differential if it is supported by only a few rule sets, representing a relationship that holds in those few rule sets but not in the others. It therefore distinguishes these rule sets from the others. In other words, the meta-rule represents one of the distinctive characteristics of these rule sets and in turn reflects the distinctive characteristics of the corresponding data sets.

For example, let us consider an apparel retailer operating a number of shops at different geographical locations. To maintain its brand, the retailer has each shop supply a basic range of apparel. The differential meta-rules are useful for the retailer as it allows the retailer to identify the differences in the apparel sold in its shops while each shop, in addition to providing the basic clothing range, caters to the preferences of its own customers.

Example 1.2 shows how meta-rules can represent the differences in rule sets and how an organization can make use of the discovered relationships.

**Example 1.2** Let us consider the supermarket chain given in Example 1.1. The following differential meta-rule would be mined from the rule sets:

$$\{i_4\} \Rightarrow \{i_1\}.$$

It is supported by $R_1$ only and represents a relationship that "in an exceptional manner, if a customer purchases $i_4$, then he/she also purchases $i_1$." This buying habit differentiates between $S_1$ and all the other outlets.

Based on this meta-rule, the supermarket chain may like to stop selling $i_4$ except in outlet $S_1$, selling some other item in its place. The rationale for this decision would be that the sales of $i_4$ affect the sales of $i_1$ in $S_1$ but in no other outlet. This decision would not reduce the revenues of $S_1$, and would improve the revenues of all the other outlets, assuming that the newly-offered items are more profitable than $i_4$. The revenue of the whole

supermarket chain may therefore be improved. □

### 1.1.3 Mining Changes in Rule Sets

The ability to detect and adapt to changes is critical to the success of many individuals and business organizations as it allows decision makers to take the changes into consideration and even take advantage of the changes when they make decisions. Knowing how circumstances will change enables a business organization to not only provide new products and services to satisfy the changing needs of its customers, but also to design corrective actions to prevent or delay undesirable changes.

Existing data mining techniques (e.g., [Agrawal *et al.* 1992; Agrawal, Imielinski, and Swami 1993a, 1993b; Agrawal and Shafer 1996; Agrawal and Srikant 1994; Bradley, Fayyad, and Reina 1998; Cheeseman and Stutz 1996; Cheung *et al.* 1996a; Ganti *et al.* 1999b; Han and Fu 1995; Houtsma and Swami 1995; Lu, Setiono, and Liu 1995; Mannila, Toivonen, and Verkamo 1994; Mehta, Agrawal, and Rissanen 1996; Park, Chen, and Yu 1995a, 1995b; Savasere, Omiecinski, and Navathe 1995; Shafer, Agrawal, and Mehta 1996; Srikant and Agrawal 1995, 1996; Zhang, Ramakrishnan, and Livny 1996]) aim at producing accurate models of the real world in an efficient manner. They are very useful for human users to better understand the problem domains and for prediction. However, regardless of how accurately a model predicts, it can only predict based on historical data. An approach to this data that does not take into account the information about change that is hidden in its patterns is not optimal, especially when the discovered models are used for classification.

In this thesis, we also study the problem of mining changes in the context of production rules. Given a rule associated with a sequence of interestingness measures (e.g., the Dempster-Shafer measure [Dempster 1967; Shafer 1976], support and confidence [Agrawal, Imielinski, and Swami 1993b], *conviction* [Brin *et al.* 1997], the chi-squared measure [Brin, Motwani, Silverstein 1997], the *J*-measure [Smyth and Goodman 1992], the *adjusted residual* and *weight of evidence* [Chan and Wong 1990, 1991], etc.) in different time periods, we propose to mine a set of meta-rules to represent the regularities governing how a rule changes over time. The change in the rule, in turn, reflects the change in the underlying characteristics hidden in the data. Human users can use the discovered meta-rules to examine the rule and to predict how the rule will change.

Example 1.3 illustrates the problem of mining changes in rule sets, showing how meta-rules can represent the changes in the discovered rules.

**Example 1.3** Let us consider the association rules concerned with items $i_1$, $i_2$, $i_3$, and $i_4$

discovered in three consecutive time periods, $t_1$, $t_2$, and $t_3$. Assume that the association rule discovered in time period $t_1$ is:

$$r: \quad \{i_1, i_2, i_3\} \Rightarrow \{i_4\}$$

whose support and confidence in $t_1$ are $support_1(r)$ = 37.8% and $confidence_1(r)$ = 95.0%, respectively. This association rule states that "if a customer purchases $i_1$, $i_2$, and $i_3$, then he/she also purchases $i_4$." A support of 37.8% for this rule means that 37.8% of records in the database being mined show that items $i_1$, $i_2$, $i_3$, and $i_4$ are purchased together, whereas a confidence of 95.0% means that 95.0% of the customers who purchased items $i_1$, $i_2$, and $i_3$ also bought $i_4$.

In time period $t_2$, the association rule becomes:

$$r': \quad \{i_1, i_2, i_3\} \Rightarrow \{i_4\}$$

whose support and confidence in $t_2$ are $support_2(r)$ = 34.9% and $confidence_2(r)$ = 94.8%, respectively.

Then in time period $t_3$, the association rule becomes:

$$r'': \quad \{i_1, i_2, i_3\} \Rightarrow \{i_4\}$$

whose support and confidence in $t_3$ are $support_3(r)$ = 28.4% and $confidence_3(r)$ = 94.5%, respectively.

The support of the association rule decreases in the period from $t_1$ to $t_2$ and in the period from $t_2$ to $t_3$. A meta-rule of support mined from these rules would be:

*Change in support in this period = Fairly decrease*
$\Rightarrow$ *Change in support in next period = Highly decrease*.

This meta-rule of support states that "if the change in support in this period fairly decreases, then the change in support in next period will decrease significantly." The support of the association rule in $t_j$ can then be predicted given the support of this rule in $t_{j-1}$ and that in $t_{j-2}$.

On the other hand, the confidence of the association rule is more or less the same in the

period from $t_1$ to $t_2$ and in the period from $t_2$ to $t_3$. A meta-rule of confidence discovered in these rules would be:

*Change in confidence in this period = More or less the same*
$\Rightarrow$ *Change in confidence in next period = More or less the same*.

It states that "if the change in confidence in this period is more or less the same, then the change in confidence in next period will be more or less the same." The confidence of the association rule in $t_j$ can then be predicted given the confidence of this rule in $t_{j-1}$ and that in $t_{j-2}$. □

## 1.2 An Overview of the Proposed Approach

To mine meta-rules from rule sets effectively, a meta-mining approach should be able to 1) generate fuzzy sets from data automatically; 2) use linguistic variables and linguistic terms to represent the discovered regularities, differences, and changes; 3) exploit the scalability of parallel computer systems to mine meta-rules efficiently; 4) group and select a subset of attributes for meta-mining; and 5) enable the mining of meta-rules involving attributes that are not originally contained in the database. This study proposes a meta-mining approach composed of a collection of techniques that satisfy these requirements. These techniques are applicable to both the mining of meta-rules from rule sets and the mining of rules from data sets.

### *1.2.1 Fuzzy Partitioning*

Many of the existing data mining algorithms (e.g., ID3 [Quinlan 1986], AQ15 [Michalski *et al.* 1986], ITRule [Smyth and Goodman 1992], CN2 [Clark and Niblett 1989], and CBA [Liu, Hsu, and Ma 1998]) can be applied only to discrete-valued data. To deal with continuous or mixed continuous and discrete valued data, the domain of each continuous attribute is typically discretized into a finite number of intervals [Ching, Wong, and Chan 1995; Chiu, Wong, and Cheung 1991; Dougherty, Kohavi, and Sahami 1995; Fayyad and Irani 1993; Kerber 1992; Kurgan and Cios 2001; Liu and Setiono 1997; Liu, Wong, and Wang 2004; Wong and Chiu 1987]. The discrete-valued and the discretized data can then be handled in a uniform fashion and rules can be mined from them. Instead of using a discretization algorithm to preprocess continuous data, some data mining algorithms use built-in discretization mechanisms. For example, when a continuous attribute is encountered in the data mining process, C4.5 [Quinlan 1993], CART [Breiman *et al.* 1984], and the association rule mining algorithm proposed in [Srikant and Agrawal 1996] discretize

it into two or more intervals so that their criterion functions are optimized. Although they do not require continuous attributes to be discretized in advance, they discretize the attributes when they are mining rules.

However, if too many data lie on the boundaries of the intervals due to the ambiguous or fuzzy nature of the attribute values near the boundary regions, discretization could result in very different discoveries in the data that could be both misleading and meaningless. Data mining algorithms therefore could not discover accurate models in the discretized data. To better handle continuous data, the use of fuzzy sets for data mining has recently been proposed in the literature [Mitra, Pal, and Mitra 2002]. This allows continuous data lying on the interval boundaries to partially belong to multiple intervals. Its resilience to noise and affinity with human knowledge representation make the use of fuzzy sets a key component of many data mining systems (e.g., [Au and Chan 1998, 1999, 2001, 2003; Chan and Au 1997b, 2001; Chan, Au, and Choi 2002; Delgado *et al.* 2003; Hirota and Pedrycz 1999; Hüllermeier 2001; Ishibuchi, Yamamoto, and Nakashima 2001; Janikow 1998; Kacprzyk and Zadrozny 2001; Lee and Kim 1997; Maimon, Kandel, and Last 1999; Yager 1991]). These systems typically require fuzzy sets to be predefined as input and they perform data mining based on these fuzzy sets.

A fuzzy set is defined by a *membership function*, which maps objects in a domain of concern to their membership values in the fuzzy set. It is associated with a *linguistic term*, which allows human users both to easily express their knowledge and to comprehend the expressed knowledge [Pedrycz and Gomide 1998; Yen and Langari 1999]. Since membership functions can profoundly affect the performance of fuzzy models, the determination of membership functions or *fuzzy partitioning* is an important problem in fuzzy data mining. A membership function can be either determined by human experts or generated directly from data. A weakness of having human experts provide input is that in most situations it is difficult for them to express or formalize their knowledge and experience [Buchanan *et al.* 1983; Johnson-Laird 1989]. It is for this reason that in this study we propose a new method for constructing fuzzy partitions directly from data.

### *1.2.2 Meta-Rule Mining Algorithms*

Based on the fuzzy sets generated, we propose to use linguistic variables and linguistic terms to represent the underlying regularities, differences, and changes hidden in the rule sets. The use of fuzzy set based techniques not only better handles the noise embedded in the data, but because of the affinity of fuzzy sets with human knowledge representation also enables human users to better comprehend the discovered meta-rules [Au and Chan 1998,

1999, 2001, 2003; Chan and Au 1997b, 2001; Chan, Au, and Choi 2002; Delgado *et al.* 2003; Hirota and Pedrycz 1999; Hüllermeier 2001; Ishibuchi, Yamamoto, and Nakashima 2001; Janikow 1998; Kacprzyk and Zadrozny 2001; Lee and Kim 1997; Maimon, Kandel, and Last 1999; Yager 1991]. In this study, we propose two new algorithms for handling fuzzy data and for mining meta-rules. These algorithms use linguistic variables and linguistic terms to represent the discovered regularities, differences, and changes.

### 1.2.3 Parallel Meta-Rule Mining Algorithms

Data mining techniques may generate a surplus of patterns and, as a result, very large rule sets [Frawley, Piatetsky-Shapiro, and Matheus 1991; Klemettinen *et al.* 1994; Matheus, Piatetsky-Shapiro, and McNeill 1996; Piatetsky-Shapiro 1991; Silberschatz and Tuzhilin 1996]. To efficiently mine meta-rules from very large rule sets, we propose to exploit the scalability of parallel computer systems. We enhance the proposed meta-mining algorithms into distributed ones to take advantage of the scalability of parallel systems.

### 1.2.4 Attribute Clustering

Given a relational table, a conventional clustering algorithm groups tuples, each of which is characterized by a set of attributes, into clusters based on similarity [Jain, Murty, and Flynn 1999]. Intuitively, tuples in a cluster are more similar to each other than those belonging to different clusters. It has been shown that clustering is very useful in many data mining applications (e.g., [Fayyad *et al.* 1996; Piatetsky-Shapiro and Frawley 1991]).

When applied to data sets such as gene expression data that are “wide” and “shallow,” conventional clustering algorithms often encounter the problem that data sets usually contain a huge number of attributes (genes) and a small number of tuples (gene expression profiles). This often compromises the performance of conventional clustering algorithms.

Euclidean distance and Pearson’s correlation coefficient are widely used as the distance measure for clustering [Jiang, Tang, and Zhang 2004]. However, when Euclidean distance is applied to the measurement of the similarity between genes, it does not effectively reflect functional similarities such as positive and negative correlations, interdependency or closeness in values. In fact, Euclidean distance accounts only for the last. In other words, the primary interest of the overall shapes of genes [Jiang, Tang, and Zhang 2004] is not well accounted for. Pearson’s correlation coefficient has been proposed for dealing with this but an empirical study [Heyer, Kruglyak, and Yooseph 1999] has shown that Pearson’s correlation coefficient is not robust to outliers and may assign a high similarity score to a pair of dissimilar genes.

Having so many attributes (genes) relative to so few tuples (samples) is also likely to result in the discovery of irrelevant patterns (i.e., gene combinations which correlate with a target variable purely by chance) [Piatetsky-Shapiro, Khabaza, and Ramaswamy 2003]. A useful technique for dealing with this is to select a small number of the most promising genes and use them solely to build models [Piatetsky-Shapiro, Khabaza, and Ramaswamy 2003]. To select genes, the *t*-value is widely used [Piatetsky-Shapiro, Khabaza, and Ramaswamy 2003]. It is important to note that the *t*-value can only be used when the samples are pre-classified. Without class information, it cannot be used for gene selection.

In this study, we present a methodology for grouping attributes that are interdependent or correlated. We refer to such a process as *attribute clustering*. Attribute clustering is based on the observation that attributes in a cluster are more correlated with each other than are attributes in different clusters. Attribute clustering allows the reduction of the search dimension of a data mining or meta-mining algorithm, facilitating the search for interesting relationships or the construction of models in a tightly correlated subset of attributes and obviating the need to search the entire attribute space. After attributes are clustered, one can select a smaller number for further analysis.

### 1.2.5 Data Transformation

*Data transformation* is an essential step in KDD [Fayyad, Piatetsky-Shapiro, and Smyth 1996]. If performed effectively, it is able to reduce the effective number of variables under consideration or to find invariant representations of the data [Fayyad, Piatetsky-Shapiro, and Smyth 1996]. However, existing data mining techniques (e.g., [Agrawal *et al.* 1992; Agrawal, Imielinski, and Swami 1993a, 1993b; Agrawal and Shafer 1996; Agrawal and Srikant 1994; Bradley, Fayyad, and Reina 1998; Cheeseman and Stutz 1996; Cheung *et al.* 1996a; Ganti *et al.* 1999b; Han and Fu 1995; Houtsma and Swami 1995; Lu, Setiono, and Liu 1995; Mannila, Toivonen, and Verkamo 1994; Mehta, Agrawal, and Rissanen 1996; Park, Chen, and Yu 1995a, 1995b; Savasere, Omiecinski, and Navathe 1995; Shafer, Agrawal, and Mehta 1996; Srikant and Agrawal 1995, 1996; Zhang, Ramakrishnan, and Livny 1996]) do not provide any explicit methodology for data transformation.

Without using data transformation, it is not possible to find some useful and important features that represent the data. It is also impossible to discover rules or meta-rules involving attributes not originally contained in the database. For example, neither the rule "if a subscriber's average monthly payment is less than fifty dollars and he/she makes a phone call during Christmas, then the phone call is over an hour in duration" nor the meta-rule "in the past few years, if a subscriber's average monthly payment is more than two

hundred dollars and he/she makes a phone call on Thanksgiving Day, then the phone call is less than half an hour in duration" can be discovered because the database does not contain explicitly the attribute values of "Christmas," "Thanksgiving Day," and "average monthly payment." These attributes are functions of the "date of call" and "monthly payment" and are not stored in the original data. To mine rules and meta-rules of this kind, one must calculate the charge of each phone call based on the start time, the end time, and the charge per minute for that call period.

Without data transformation, useful and important features may not be utilized in data mining and meta-mining tasks. As a result, interesting and meaningful rules (meta-rules) may not be discovered even with the most effective data mining (meta-mining) algorithms. In this study, to enable the mining of interesting and meaningful rules and meta-rules, we propose a data transformation method. This method also enables data mining in the union of relational and transaction data that existing techniques are not developed for [Au and Chan 2003; Chan and Au 2001].

## 1.3 Organization of the Thesis

The rest of this thesis is organized as follows. In Chapter 2, we survey related work. In Chapter 3, we present the problem definitions of mining meta-rules of regularities, differences, and changes in rule sets. We also give an overview of our proposed meta-mining approach. This approach is comprised of a collection of techniques, including a fuzzy partitioning algorithm, serial and parallel algorithms for mining meta-rules, a data transformation technique, and an attribute clustering method.

In Chapter 4, we propose a new approach to data transformation in databases. In addition to enabling the discovery of rules involving attributes that are not originally contained in the data, it also enables data mining in the union of relational and transaction data. The proposed approach involves the use of *transformation functions* to transform the original data. The application of transformation functions to the original data results in a set of *transformed data*. Instead of mining the original data, we mine rules from the transformed data. From the rule sets discovered in the transformed data, we can mine meta-rules involving attributes not contained in the original data.

In Chapter 5, we introduce a new fuzzy partitioning method to determine the membership functions of fuzzy sets directly from data. In other words, this method forms a fuzzy partition of the input space automatically. The proposed method uses an information-theoretic measure, which evaluates the interdependence between the class and an attribute,

as the objective function for fuzzy partitioning. It employs fractional programming (iterative dynamic programming) to find the global optimum of the measure. Fuzzy partitioning enables our proposed meta-rule mining algorithms and other fuzzy data mining techniques to build fuzzy models or discover fuzzy rules based on the generated fuzzy sets instead of relying on user-specified ones. To evaluate the effectiveness of the fuzzy partitioning method, several real-world data sets were used in our experiments. The experimental results show that this method is very effective when compared to other well-known discretization and fuzzy partitioning approaches.

Chapter 6 defines the problem of attribute clustering and introduces a methodology for solving it. Our proposed method groups interdependent attributes into clusters by optimizing a criterion function derived from an information measure that reflects the interdependence between attributes. By applying our algorithm to a data set, meaningful clusters of attributes are discovered. The grouping of attributes based on attribute interdependence within group helps to capture different aspects of association relationships in each group. Significant attributes selected from each group then contain useful information for classification and identification. To evaluate the performance of the proposed approach, we applied it to two well-known gene expression data sets and compared our results with those obtained by other methods. Our experiments show that the proposed method is able to find the meaningful clusters of genes. By selecting a subset of genes which have high multiple-interdependence with others within clusters, significant classification information can be obtained. Thus a small pool of selected genes can be used to build classifiers with very high classification rates. From the pool, gene expressions of different categories can be identified.

In Chapter 7, we propose two new algorithms for mining meta-rules in rule sets. One mines rules and meta-rules based on heuristics, whereas the other mines them using a genetic algorithm. Both algorithms employ an objective interestingness measure to distinguish interesting association relationships from uninteresting ones. They also utilize linguistic variables and linguistic terms to represent the discovered relationships. To evaluate their performance, we applied them to several real-world data sets. The experimental results of the data mining tasks show that they can build very accurate models.

We then enhance these algorithms into distributed ones to exploit the scalability of parallel systems in Chapter 8. The parallel algorithms divide a data set into several horizontal partitions and assign them to different sites in a distributed system. Each site scans its database partition to obtain the number of tuples characterized by different attribute values and then exchanges the local counts with all the other sites to find the global counts.

Based on the global counts, the interestingness measures are computed and the sites are able to uncover interesting association relationships. The parallel algorithms were implemented in an experimental test bed. Their scalability was tested using a popular benchmarking data set and the results show that they have very good size-up, speedup, and scale-up performance.

In Chapter 9, we apply our proposed meta-mining approach to several synthetic and real-world data sets for experimentation. The results show that useful and meaningful regularities, differences, and changes can be discovered.

Finally, we conclude this study with a summary in Chapter 10.

# Chapter 2
# Related Work

In this chapter, we survey the related work in the literature. We first provide the state of the art of existing data mining techniques in Section 2.1. We then give the work related to meta-mining and the mining of regularities, differences, and changes in the subsequent sections. We also discuss the pros and cons of different approaches in this same chapter.

## 2.1 Data Mining

### *2.1.1 Association Rule Mining*

An example of an association rule is "90% of transactions that contain bread also contain butter; 3% of all transactions contain both of these items." The 90% is referred to as the *confidence* and the 3%, the *support*, of the rule. More formally, an association rule is defined as follows [Agrawal, Imielinski, and Swami 1993b].

Let $I = \{i_1, \ldots, i_m\}$ be a set of binary attributes called *items* and $T$ be a set of *transactions*. Each transaction $t \in T$ is represented as a binary vector with $t[k] = 1$ if $t$ contains item $i_k$ and $t[k] = 0$, otherwise, for $k = 1, \ldots, m$. A set of items is known as an *itemset*. The support of an itemset, $X \subset I$, is defined as the percentage of tuples containing $X$. The itemset is *frequent* if its support is greater than or equal to the user-specified *minimum support*. An association rule is defined as an implication of the form $X \Rightarrow Y$ where $X \subset I$, $Y \subset I$, and $X \cap Y = \varnothing$. The rule $X \Rightarrow Y$ holds in $T$ with support defined as the percentage of tuples containing $X$ and $Y$ and confidence defined as the percentage of tuples containing $Y$ given that they also contain $X$. An association rule is interesting if its support and confidence are greater than or equal to the user-supplied minimum support and *minimum confidence*, respectively. Since they are defined over binary data, association rules of such type are often referred to as *Boolean association rules*.

Algorithms for mining Boolean association rules first find all frequent itemsets in a database and then generate association rules from these frequent itemsets. Since the former step consumes most of the computational resources, current research focuses mainly on the speeding up of the process of discovering frequent itemsets (e.g., [Agrawal, Imielinski, and Swami 1993b; Agrawal and Srikant 1994; Houtsma and Swami 1995; Mannila, Toivonen, and Verkamo 1994; Park, Chen, and Yu 1995a; Savasere, Omiecinski, and Navathe 1995]).

Apriori [Agrawal and Srikant 1994] is a well-known algorithm for mining Boolean association rules. At each iteration, it generates a set of *candidate itemsets* from the frequent itemsets found at the previous iteration. It then scans all the transactions to obtain the *support counts* of the candidate itemsets. Subsequently, Apriori finds all the frequent itemsets for that iteration and proceeds to the next iteration. To improve the computational efficiency of the algorithm, different techniques have been proposed. For example, DHP [Park, Chen, and Yu 1995a] extends Apriori by using a hashing technique to prune away some candidate itemsets at the second iteration.

Instead of scanning through a large database multiple times, another algorithm known as Partition [Savasere, Omiecinski, and Navathe 1995] accomplishes the mining of frequent itemsets in only two scans of the database. Partition starts the data mining process by dividing the database into a number of non-overlapping partitions. In the first database scan, each partition is scanned to find all frequent itemsets in that partition. The frequent itemsets are then merged to generate all candidate itemsets. In the second database scan, it counts the actual support of these itemsets and identifies the frequent itemsets.

Unlike these techniques, a method called FP-growth [Han, Pei, and Yin 2000] has been proposed to mine frequent itemsets without candidate generation. It first compresses the database into a FP-tree, but retains the itemset association information at the same time. It then divides the FP-tree into a set of *conditional databases*, each of which is associated with one frequent item, and it mines each such database separately. The FP-growth method transforms the problem of finding long frequent itemsets to looking for shorter ones recursively and then concatenating the suffix [Han, Pei, and Yin 2000]. It has been shown in [Han, Pei, and Yin 2000] that this method is about an order of magnitude faster than Apriori. Although both FP-growth and Partition accomplish the mining of frequent itemsets in a small number of database scans (one in FP-growth and at most two in Partition), FP-growth does not generate any candidate itemsets in the data mining process.

Techniques for mining Boolean association rules have recently been extended to take *is-a* hierarchies (i.e., taxonomies) into consideration. An example of a three-level is-a hierarchy is "professor *is-a* faculty member *is-a* staff." Association rules involving is-a hierarchies are known as *multiple-level association rules* [Han and Fu 1995] or *generalized association rules* [Srikant and Agrawal 1995]. In this thesis, we use the term generalized association rules to refer to both of their work. The mining of these rules involves mining a database of transactions consisting of sets of items, each of which is defined at some level in a hierarchy. In other words, the antecedent and the consequent of a generalized association rule can be some set of items and/or their ancestors in the corresponding hierarchies. Like

Boolean association rules, generalized association rules are also defined over binary data. They are therefore rather restrictive in their applications in many different areas. It is for this reason that a lot of recent efforts have been put into the mining of *quantitative association rules* [Srikant and Agrawal 1996].

Quantitative association rules are defined over quantitative (continuous) and categorical (discrete) attributes [Srikant and Agrawal 1996]. The statement "70% of tertiary educated people between age 25 and 30 are unmarried" is one such example. To handle quantitative attributes, the domains of these attributes are discretized into intervals. The discretization can be performed as a part of the algorithms (e.g., [Srikant and Agrawal 1996]) or as a preprocessing step before data mining (e.g., [Liu, Hsu, and Ma 1998]). Both categorical and quantitative attributes can be handled in a uniform fashion as a set of <attribute, integer value> pairs by mapping the values of categorical attributes to a set of consecutive integers and by mapping the discretized intervals of quantitative attributes to consecutive integers, which preserve the order of the intervals [Srikant and Agrawal 1996]. Instead of having just one field for each attribute, there is a need to use as many fields as the number of different attribute values. For example, the value of a Boolean field corresponding to <$attribute_1$, $value_1$> would be "1" if $attribute_1$ has $value_1$ in the original record and "0," otherwise [Srikant and Agrawal 1996]. After the mappings, the algorithms for mining Boolean association rules (e.g., [Agrawal, Imielinski, and Swami 1993b; Agrawal and Srikant 1994; Houtsma and Swami 1995; Mannila, Toivonen, and Verkamo 1994; Park, Chen, and Yu 1995a; Savasere, Omiecinski, and Navathe 1995]) can be applied to the encoded data.

Recently, the problem of mining association rules has further been extended in [Lu, Han, and Feng 1998] for the mining of *n-dimensional inter-transaction association rules*. An *n*-dimensional inter-transaction association rule is concerned with the association among items from different transaction records, each of which is characterized by *n dimensional attributes* (e.g., time, location, etc.). Two algorithms, E-Apriori and EH-Apriori, which are extensions of Apriori, have been proposed in [Lu, Han, and Feng 1998] to deal with the huge search space.

For association rule mining algorithms such as those described in [Agrawal, Imielinski, and Swami 1993b; Agrawal and Srikant 1994; Han and Fu 1995; Lu, Han, and Feng 1998; Houtsma and Swami 1995; Mannila, Toivonen, and Verkamo 1994; Park, Chen, and Yu 1995a; Savasere, Omiecinski, and Navathe 1995; Srikant and Agrawal 1996] to determine if a Boolean, generalized, quantitative, or *n*-dimensional inter-transaction association rule is interesting, its support and confidence have to be greater than or equal to the user-supplied

thresholds (i.e., minimum support and minimum confidence). A weakness of such approach is that many users do not have any idea what the thresholds should be. If they are set too high, a user may miss some useful rules; but if they are set too low, the user may be overwhelmed by many irrelevant ones [Han and Kamber 2001; Hand, Mannila, and Smyth 2001].

To ease the burden of having a user determine minimum support, an automatic mechanism is employed in WEKA [Witten and Frank 2005]. It sets the upper bound and the lower bound for minimum support to 1.0 and 0.1, respectively. Apriori in WEKA starts with the upper bound and incrementally decreases minimum support in a pre-defined step, which is 0.05 by default. It stops when a user-specified number of rules are generated or the lower bound is reached.

#### 2.1.1.1 Parallel Algorithms for Mining Association Rules

Serial algorithms for mining association rules (e.g., [Agrawal, Imielinski, and Swami 1993b; Agrawal and Srikant 1994; Houtsma and Swami 1995; Mannila, Toivonen, and Verkamo 1994; Park, Chen, and Yu 1995a; Savasere, Omiecinski, and Navathe 1995]) have been extended to take advantage of the scalability of parallel systems to handle very large databases.

For mining association rules, three algorithms, namely, *Count Distribution*, *Data Distribution*, and *Candidate Distribution*, which adopt Apriori in a distributed-memory architecture, have been proposed in [Agrawal and Shafer 1996]. These algorithms divide a database into several horizontal partitions and assign them to different processors. In the case of Count Distribution, every processor runs Apriori over its database partition with a modification that it exchanges the *local support counts* of candidate itemsets in its database partition with all the other processors to find the *global support counts* in the whole database and then identifies frequent itemsets based on the global support counts at each iteration.

Data Distribution partitions candidate itemsets and assigns them to different processors in a round-robin fashion. At each iteration, every processor broadcasts its database partition to all the other processors to find the global support counts of its candidate itemsets. Candidate Distribution starts the data mining process by employing Count Distribution or Data Distribution. At certain iteration, it divides the candidate itemsets into several disjoint subsets and assigns different subsets to different processors. At the same time, the database is repartitioned in such a way that each processor can find the (global) support counts of its candidate itemsets in its database partition independent of other processors. To achieve this,

parts of the database may have to be replicated on several processors. Each processor can then generate candidate itemsets and count the supports of these candidate itemsets independently at subsequent iterations.

The experimental results presented in [Agrawal and Shafer 1996] show that the performance of Count Distribution is superior to Data Distribution and Candidate Distribution. The broadcasting of database partitions involves high communication overhead. Furthermore, having each processor to scan the entire database at each iteration makes Data Distribution perform relatively poorly when compared to Count Distribution. Candidate Distribution also performs less satisfactorily than Count Distribution because of the overhead of repartitioning the database and replicating parts of the database, which may be large, on several processors.

In addition to these three algorithms, a number of parallel algorithms based on Apriori have also been described in the literature (e.g., [Cheung *et al.* 1996a; Han, Karypis, and Kumar 1997; Park, Chen, and Yu 1995; Shintani and Kitsuregawa 1996]). They use different optimization techniques to improve the performance. For example, *Intelligent Data Distribution* [Han, Karypis, and Kumar 1997] improves the performance of Data Distribution by employing a *ring-based all-to-all broadcast* to exchange database tuples, switching to Count Distribution when the total number of candidate itemsets falls below a threshold, and dividing candidate itemsets using a prefix-based partitioning. *Hybrid Distribution* [Han, Karypis, and Kumar 1997] further improves the performance of Intelligent Data Distribution by combining it with Count Distribution. It splits a system of multiple processors into several equal-sized groups, where each group is considered as a *hypothetical processor*. Hybrid Distribution applies Count Distribution to the database among the hypothetical processors, whereas it uses Intelligent Data Distribution among the processors within each group. At each iteration, Hybrid Distribution also dynamically adjusts the number of hypothetical processors.

Furthermore, FDM [Cheung *et al.* 1996a] extends Count Distribution by adopting a new approach to reduce the number of candidate itemsets for counting. Since every *globally frequent* itemset must be *locally frequent* at some site, each site only considers the candidate itemsets generated from the globally and locally frequent itemsets at that site. It then scans through its database partition to find the local support counts of these candidate itemsets. Three optimization methods have been presented in [Cheung *et al.* 1996a]. The *local pruning* method has each site to remove any itemset that is not locally frequent; the *global pruning* method is to find the upper bounds of the supports of itemsets and remove those itemsets whose upper bounds are less than the minimum support; and the *count*

*polling* method lets each *polling site* request the local support counts of the itemsets assigned to it from all the other sites, calculate the global support counts, and broadcast the globally and locally frequent itemsets with their support counts to all the other sites.

Similar to the serial association rule mining algorithms, these parallel algorithms identify interesting association rules based on the user-specified thresholds (i.e., minimum support and minimum confidence). They may not find some interesting rules if the thresholds are set too high, whereas they may find irrelevant ones if the thresholds are set too low [Han and Kamber 2001; Hand, Mannila, and Smyth 2001].

### 2.1.2 Classification

The classification problem typically involves finding a classification model or a *classifier* to classify a set of records into different predefined classes. To do so, the *class* attribute – the attribute in a database in which records should be classified according to – is first identified by domain experts. A set of records, called the *training set*, is then used to construct a classifier. Using the classifier, a record that is not originally in the training set can be classified based on its attribute values. The classification problem has been studied extensively by researchers in the machine learning community and various techniques have been proposed to solve it [Michie, Spiegelhalter, and Taylor 1994]. Among the many solution techniques, the decision-tree based approaches are the most popularly adopted [Agrawal *et al.* 1992; Agrawal, Imielinski, and Swami 1993a; Mehta, Agrawal, and Rissanen 1996; Shafer, Agrawal, and Mehta 1996].

Most of the decision-tree based algorithms (e.g., IC [Agrawal *et al.* 1992], CDP [Agrawal, Imielinski, and Swami 1993a], CART [Breiman *et al.* 1984], SLIQ [Mehta, Agrawal, and Rissanen 1996], C4.5 [Quinlan 1993], Serial SPRINT [Shafer, Agrawal, and Mehta 1996], etc.) are composed of two phases: the tree-building phase and the tree-pruning phase. In the tree-building phase, a decision tree is constructed by recursively partitioning the training set. This process continues until all or the majority of the records in each partition belong to a single class. At the end of this process, a decision tree is constructed. Each non-leaf node in the resulting decision tree carries out a test on an attribute so as to determine how the training set should be partitioned. Since the decision tree may contain branches that are created due to noises in the data set, these branches have to be deleted. The tree-pruning phase therefore consists of, for example, selecting and removing the subtree with the least estimated error rate. Tree pruning has been shown to increase the classification accuracy of a decision tree on one hand and reduce the complexity of the tree on the other.

Decision-tree based approaches (e.g., IC, CDP, CART, ID3 [Quinlan 1986], and C4.5) originally require the entire database to fit in the real memory of a computer and hence they cannot handle large databases. Some recent efforts have been put into improving the scalability of decision-tree based algorithms by handling disk-resident data that are too large to fit in memory (e.g., SLIQ and Serial SPRINT).

SLIQ creates a set of *attribute lists* and a *class list*. An attribute list, in which an entry consists of an attribute value and a record identifier, is created for each attribute. The attribute lists for continuous attributes are sorted by attribute values when they are created. In the class list, each entry contains a class label, a record identifier, and a pointer to a node in the decision tree that indicates to which node the corresponding training record currently belongs. When a decision-tree node is split to create new children and a training record is assigned to one of the children, the reassignment is done simply by changing the pointer field of the corresponding entry in the class list. Only a portion of an attribute list is required to fit in real memory when a node is being split. However, the class list has to fit in real memory all the time or else the performance will be degraded severely because the class list is randomly assessed and frequently updated. Since the size of the class list grows in direct proportion to the size of the training set, this limits the size of the database that SLIQ can handle.

Serial SPRINT overcomes this problem by using different data structure. It maintains an attribute list for each attribute, in which an entry consists of an attribute value, a class label, and a record identifier. The attribute lists for continuous attributes are sorted by attribute values when they are created. The initial lists are associated with the root of the decision tree at first. Nodes are then split to create new children. The attribute lists belonging to each node are therefore partitioned and the partitioned attribute lists are associated with the children. The order of the entries in a list is preserved when it is partitioned so that the lists for continuous attributes are sorted once only. In order to split the attribute lists according to the splitting decision, Serial SPRINT creates a hash table that keeps a mapping between a record identifier and the node with which the record is associated based on the splitting decision. The elimination of the use of the class list makes Serial SPRINT can handle very large databases.

In the process of constructing decision trees, all arcs labeled by the values of selected attributes have to be expanded. This may introduce irrelevant variables and make resulting paths longer than what are actually needed. Furthermore, the construction of decision trees usually involves binarizing continuous attributes into two intervals so that the records with some attribute values greater than some threshold belong to one branch, whereas those

records with the attribute values less than or equal to that threshold belong to the other branch (e.g., [Agrawal, Imielinski, and Swami 1993a; Breiman *et al.* 1984; Mehta, Agrawal, and Rissanen 1996; Quinlan 1993; Shafer, Agrawal, and Mehta 1996]). Such binarization on continuous attributes may result in multiple tests on the same attribute and hence a substantial increase in the complexity of the resulting decision trees. The decision-tree based algorithms are also sensitive to the small differences in training data. For example, a very different decision tree can be constructed when some records appear more than once in the training set.

In addition to the abovementioned problems, when decision-tree based algorithms are extended to determine the probabilities associated with such classifications (see, e.g., [Quinlan 1987b]), it is possible that some leaves in a decision tree have similar class probabilities.

#### 2.1.2.1 Parallel Algorithms for Classification

To build decision trees in very large databases, some recent efforts have been put into exploiting the scalability of parallel systems (e.g., [Joshi, Karypis, and Kumar 1998; Shafer, Agrawal, and Mehta 1996; Srivastava *et al.* 1998; Zaki, Ho, and Agrawal 1999]).

Parallel SPRINT [Shafer, Agrawal, and Mehta 1996] extends Serial SPRINT by distributing the attribute lists evenly among all the processors and finding the split point for a node in the decision tree in parallel. To split the attribute lists according to the splitting decision, the hash table is required on all the processors to keep a mapping between a record identifier and the node with which it is associated based on the splitting decision. In order to construct this hash table, each processor requires $O(N)$ memory to store the hash table and $O(N)$ communication overhead for *all-to-all broadcast*, where $N$ is the number of records in the data [Kumar *et al.* 1994]. This makes Parallel SPRINT to be unscalable with respect to runtime and memory requirements [Joshi, Karypis, and Kumar 1998]. To overcome this shortage, ScalParC [Joshi, Karypis, and Kumar 1998] employs a distributed hash table, which is split among all the processors, and uses an efficient personalized communication to update the hash table.

Parallel classification algorithms such as Parallel SPRINT and ScalParC are originally developed for the distributed-memory architecture. Recently, techniques proposed in [Zaki, Ho, and Agrawal 1999] extend Parallel SPRINT to work on the shared-memory architecture.

Furthermore, two basic parallel formulations for the construction of decision trees (i.e., the *synchronous* and *partitioned tree construction* approaches) have been proposed in

[Srivastava *et al.* 1998]. In the synchronous tree construction approach, all processors construct a decision tree synchronously by sending and receiving class distribution information of local data. It incurs a high communication overhead as the number of nodes in the decision tree gets larger and larger in the tree-building phase. On the other hand, in the partitioned tree construction approach, different processors work on different parts of the decision tree. It incurs the cost of load balancing when each node is split. To combine the strengths and eliminate the weaknesses of these approaches, a hybrid scheme is given in [Srivastava *et al.* 1998] that keeps continuing with the synchronous tree construction approach as long as the communication cost incurred is not too high. Once this cost becomes too high, the hybrid scheme switches to the partitioned tree construction approach. The experimental results in [Srivastava *et al.* 1998] show that the hybrid scheme outperforms the partitioned tree construction approach, which in turn outperforms the synchronous tree construction approach.

These parallel algorithms focus mainly on the speedup of the construction of decision trees. They suffer from the same problems experienced by their serial counterparts, that is, the potentially high complexity of resulting decision trees because of the binarization of continuous attributes and the high sensitivity to the small differences in data sets.

### 2.1.3 Discretization and Fuzzy Partitioning

Regardless of a data mining algorithm is developed for association rule mining or classification, it typically requires the domains of continuous attributes to be discretized into a finite number of intervals [Breiman *et al.* 1984; Clark and Niblett 1989; Liu, Hsu, and Ma 1998; Michalski *et al.* 1986; Quinlan 1986, 1993; Smyth and Goodman 1992; Srikant and Agrawal 1996].

Discretization techniques can be classified into two categories: *unsupervised* and *supervised*. Unsupervised methods simply apply a prescribed scheme to discretize the continuous values without making use of the attribute-class information, whereas supervised methods take into consideration the attribute-class information.

The representatives of unsupervised discretization methods are equal-width and equal-frequency [Chiu, Wong, and Cheung 1991]. The equal-width discretization merely divides the range of observed values for a continuous attribute into $k$ equal-sized intervals, where $k$ is a user-specified parameter. Given $m$ records, the equal-frequency discretization divides the range of values for a continuous attribute into $k$ intervals, where each interval contains $m / k$ attribute values. A typical problem of unsupervised methods is that it is difficult to determine how many intervals are the best for a given attribute.

Theoretically, directed by class information, supervised discretization methods can automatically determine the best number of intervals for each given continuous attribute for classification. Examples of supervised methods are maximum entropy [Wong and Chiu 1987], CADD [Ching, Wong, and Chan 1995], information entropy maximization [Fayyad and Irani 1993], Paterson-Niblett [Paterson and Niblett 1982] (which is built into C4.5), ChiMerge [Kerber 1992], Chi2 [Liu and Setiono 1997], and CAIM [Kurgan and Cios 2001]. These supervised methods usually rely on heuristics to attain the local optimum of their objective functions that measure the class and attribute dependence. For example, CADD discretizes data by heuristically maximizing the interdependence between the class and the continuous attribute [Ching, Wong, and Chan 1995]. CAIM differs from CADD by using a different objective function to capture the dependency relationship between the class and the continuous attribute while keeping the number of intervals as minimal as possible [Kurgan and Cios 2001]. The use of heuristics makes supervised methods cannot always find the global optimum of the objective functions.

An alternative to discretization is fuzzy partitioning. Fuzzy partitioning techniques generate fuzzy sets to represent the domains of continuous attributes. They can be classified into three categories: 1) *grid partitioning*, 2) *tree partitioning*, and 3) *scatter partitioning* [Yen and Langari 1999]. The grid partitioning forms a partition by dividing the input space into several fuzzy slices, each of which is specified by a membership function for each feature dimension. The tree partitioning constructs a partition by applying a series of *guillotine cuts* such that each is a cut that is made across the subspace to be partitioned and each of the regions so produced can be subject to further independent guillotine cutting. The scatter partitioning finds a subset of the input space that characterizes the fuzzy regions of possible occurrence of records in the data set instead of covering the whole input space [Yen and Langari 1999]. Of the different fuzzy partitioning methods, the grid partitioning is the most commonly used in practice, particularly in system control applications [Yen and Langari 1999].

A grid partition can be *uniform*, if formed by uniformly symmetric membership functions, or *non-uniform*, if formed by non-uniformly spaced asymmetric membership functions. Although a uniform grid partition is easier to construct, a non-uniform partition is more flexible in adapting to the specific nonlinear characteristics of the function being approximated [Yen and Langari 1999]. In addition to having human experts to form a grid partition, learning techniques can also be used to construct the partition. Typical learning methods used for such purpose include *fuzzy clustering* (e.g., [Bezdek 1981; Fajfer and Janikow 2000; Janikow and Fajfer 1999; Liao, Celmins, and Hammell II 2003]), *neural*

*networks* (e.g., [Jang 1993; Kohonen 2001]), and *genetic algorithms* (e.g. [Arslan and Kaya 2001; Karr 1991]).

Fuzzy clustering algorithms aim at finding soft partitions of data sets based on certain criteria. A datum in a data set can partially belong to multiple soft partitions (clusters). It is important to note that a soft partition is not necessarily a fuzzy partition because the input space can be larger than the data set. However, most fuzzy clustering algorithms, including the *fuzzy c-means* (FCM) algorithm [Bezdek 1981], generate a soft partition that also forms a fuzzy partition [Pedrycz and Gomide 1998; Yen and Langari 1999]. Fuzzy clustering algorithms can therefore be used for fuzzy partitioning.

To form a fuzzy partition, we can construct a neural network that takes variables of a fuzzy model as inputs and generates the degrees with which the input data belong to a predetermined number of fuzzy regions. If supervised training algorithms for neural networks (e.g., *backpropagation* [Rumelhart, Hinton, and Williams 1986; Werbos 1974]) are used, one needs to have a set of training data, which can be obtained either by asking domain experts to assign membership degrees to a sample set of input data or by clustering a sample set of input data using a clustering algorithm (e.g., the FCM algorithm) [Yen and Langari 1999]. Consequently, the fuzzy partitions so produced should be more or less the same as those obtained by fuzzy clustering. This limits the merits of using neural networks to find fuzzy partitions. Instead of using supervised training algorithms for neural networks, one can employ unsupervised training algorithms (e.g., Kohonen's s*elf-organizing maps* (SOM) [Kohonen 2001]) for fuzzy partitioning. A SOM is composed of an input layer of units, a one- or two-dimensional output grid of processing units, and a set of connections linking the input units to the output units. To cluster the input data, one simply feeds each record into a SOM, while each output unit of the SOM competes with all of the others to "win" the record, and the training algorithm updates the weights of the connections to the winning unit along with those nearby units to better match the record. Unlike supervised algorithms, it does not require the assignment of any membership degree to the input record in the training. After training, the SOM usually ends up with a few units that summarize many observations (*strong* units) and several units that do not really correspond to any of the observations (*weak* units). The strong units represent the prototypes of the clusters formed. These clusters form the fuzzy partition.

Since fuzzy partitioning can be formulated as an optimization problem of finding the parameters of fuzzy sets composing a fuzzy partition that optimizes the resultant fuzzy model based on certain evaluation criteria, genetic algorithms (GAs) can be applied to construct fuzzy partitions. Given an attribute, let us assume that a predetermined number of

fuzzy sets are used to characterize it. To form a fuzzy partition, the parameters of the fuzzy sets are encoded in a fixed-length chromosome. As a result, each chromosome represents a fuzzy partition and it is evaluated by an appropriate fitness function (e.g., classification accuracy if the resultant model is for classification or the difference between the actual and the expected output if the resultant fuzzy model is for control applications, etc.). The GA starts from generating a population of chromosomes in a random manner. It then evaluates the fitness of the chromosomes by the fitness function. Based on their fitness, chromosomes are selected and a new population of chromosomes is generated by *crossover* and *mutation*. These steps repeat until some termination criteria are satisfied. The interested readers are referred to [Goldberg 1989] for the details.

The abovementioned fuzzy partitioning techniques are unsupervised because they do not take into account the interdependence between the class and the attribute.

Recently, the fuzzy interpretation of discretized intervals has been proposed in [Wu 1999]. It represents one of the first attempts, if not the first, on discretizing attributes with fuzzy border. Given a user-specified spread parameter, the linear, polynomial, and arctan membership functions are proposed to fuzzify the borders of an interval. When a value is covered by more than one fuzzy interval, the match degree is given by either the maximum of the membership degrees of the value in all the intervals or the *fuzzy plus* of all the membership degrees. The three membership functions do not show any significant difference in the experiments so that the polynomial function is chosen as the default [Wu 1999]. The experimental results in [Wu 1999] show that HCV [Wu 1995] exhibits better performance when it is equipped with fuzzy interpretation as compared to not equipping with fuzzy interpretation.

### 2.1.4 Fuzzy Sets in Data Mining

Regardless of how the values of continuous attributes are discretized, the intervals may not be concise and meaningful enough for human users to easily obtain non-trivial knowledge from the discovered relationships. To better handle continuous data, the use of fuzzy sets for data mining has recently been proposed in the literature [Mitra, Pal, and Mitra 2002]. The resilience to noises and the affinity with the human knowledge representation make fuzzy sets to be used in many data mining systems (e.g., [Au and Chan 1998, 1999, 2001, 2003; Chan and Au 1997b, 2001; Chan, Au, and Choi 2002; Delgado *et al.* 2003; Hirota and Pedrycz 1999; Hüllermeier 2001; Ishibuchi, Yamamoto, and Nakashima 2001; Janikow 1998; Kacprzyk and Zadrozny 2001; Lee and Kim 1997; Maimon, Kandel, and Last 1999; Yager 1991]).

*Linguistic summaries* introduced in [Yager 1991] express knowledge using a linguistic representation that is natural for human users to comprehend. An example of a linguistic summary is the statement "about half of the people in the database are middle-aged." However, no algorithm was proposed for generating linguistic summaries in [Yager 1991]. Recently, the use of an algorithm for mining association rules for the purpose of linguistic summaries has been studied in [Kacprzyk and Zadrozny 2001]. This technique extends AprioriTid [Agrawal and Srikant 1994], a well-known algorithm for mining association rules, to handle linguistic terms (fuzzy values). An attribute is replaced by a set of artificial attributes (items) so that a tuple supports a specific item to a certain degree, which is in the range from 0 to 1. Given two user-specified thresholds, $threshold_1$ and $threshold_2$, an item or an itemset (i.e., a combination of items) is considered interesting if its *fuzzy support* is greater than $threshold_1$ and it is also less than $threshold_2$. Although this technique is very useful, many users may not be able to set the thresholds appropriately.

In addition to linguistic summaries, an interactive process for the discovery of top-down summaries, which utilizes *fuzzy is-a hierarchies* as domain knowledge, has been described in [Lee and Kim 1997]. This technique aims at discovering a set of *generalized tuples*, such as <technical writer, documentation>. In contrast to association rules, which involve the implications between different attributes, linguistic summaries and generalized tuples only provide the summarization on different attributes. The idea of implication has not been taken into consideration and hence these techniques are not developed for the task of rule discovery.

Furthermore, the applicability of fuzzy modeling techniques to data mining has been discussed in [Hirota and Pedrycz 1999]. Given a relational table, *X*, and a context variable, *A*, the *context-sensitive fuzzy clustering* method reveals the structure in *X* in the context of *A*. Since this method can only manipulate continuous attributes, the values of any discrete attributes are first encoded into numeric values. The context-sensitive fuzzy clustering method is then applied to the encoded data to induce clusters in the context of *A*. Although the encoding technique allows this method to deal with discrete attributes, the distances between the encoded numeric values, which do not possess any meaning in the original discrete attributes, are used to induce the clusters. Therefore, the associations that are concerned with these attributes, which are discovered by the context-sensitive fuzzy clustering method, may be misleading.

### 2.1.5 Data Mining Based on Genetic Algorithms

Other than the use of decision-tree based algorithms, techniques based on genetic algorithms

(GAs) have also been proposed for predictive modeling. There are currently two different GA-based approaches for rule discovery: the Michigan approach and the Pittsburgh approach. The Michigan approach, exemplified by Holland's classifier system [Holland 1986], represents a rule set by the entire population, whereas the Pittsburgh approach, exemplified by Smith's LS-1 system [Smith 1983], represents a rule set by an individual chromosome. Although the Michigan approach is able to deal with multi-class problems, one of the major difficulties in using it is the problem in credit assignment, which gives the activated classifiers a reward if the classification they produced is correct and gives them a punishment, otherwise. Specifically, it is extremely hard to come up with a good credit assignment scheme that works.

The algorithms based on the Pittsburgh approach (e.g., [DeJong, Spears, and Gordon 1993; Janikow 1993; Smith 1983]) represent an entire rule set as a chromosome, maintain a population of candidate rule sets, and use selection and genetic operators to produce new generation of chromosomes and, hence, new rule sets. Each chromosome competes with one another in terms of classification accuracy on the application domain. Individuals are selected for reproduction using *roulette wheel selection* and a whole new population is generated based on *crossover* and *mutation*. The selected chromosomes produce offspring using an extended version of the standard *two-point crossover* operator such that the crossover points can occur either both on rule boundaries or within rules [DeJong, Spears, and Gordon 1993; Smith 1983]. That is, if one parent is being cut on a rule boundary, then the other parent must be cut on a rule boundary as well; similarly, if one parent is being cut at a point, say, 5 bits to the right of a rule boundary, then the other parent must be cut in a similar spot [DeJong, Spears, and Gordon 1993; Smith 1983]. The mutation operator is identical to the classical one, which performs bit-level mutations. The fitness of each individual rule set is computed by testing the rule set on the current set of training examples [DeJong, Spears, and Gordon 1993; Smith 1983].

The Pittsburgh approach is originally designed for single-class learning problems and hence only the antecedent of a rule is encoded into an allele of a chromosome [DeJong, Spears, and Gordon 1993; Janikow 1993; Smith 1983]. An instance that matches one or more rules is classified as a positive example of the concept (class) and an instance that fails to match any rule is classified as a negative example [DeJong, Spears, and Gordon 1993; Janikow 1993; Smith 1983]. To tackle multi-class problems, they could be extended by introducing multiple populations so that a specific population is dedicated to learn each concept. It is possible that an instance is matched by more than one rule of different concepts on one hand and it is also possible that an instance is matched by none of any rule

of any concept on the other. Unfortunately, this problem has not been addressed in many of the systems based on the Pittsburgh approach (e.g., [DeJong, Spears, and Gordon 1993; Janikow 1993; Smith 1983]).

Recently, the use of GAs for rule discovery in the application of data mining has been studied in [Choenni 2000; Fidelis, Lopes, and Freitas 2000; Freitas 2002; Kwedlo and Kretowski 1998]. These algorithms are based on the Michigan approach in such a way that each rule is encoded in a chromosome and the rule set is represented by the entire population. Unlike classifier systems (e.g., [Greene and Smith 1994; Holland 1986; McAulay and Oh 1994]), they 1) have modified the individual encoding method to use non-binary representation; 2) do not encode the consequents of rules into the individuals; 3) use extended version of crossover and mutation operators suitable to their representations; 4) do not allow rules to be invoked as a result of the invocation of other rules; and 5) define fitness functions in terms of some measures of classification performance (e.g., *cover* [Choenni 2000], *sensitivity* and *specificity* [Fidelis, Lopes, and Freitas 2000], etc.).

It is important to note that these algorithms [Choenni 2000; Fidelis, Lopes, and Freitas 2000; Freitas 2002] are developed to discover rules for a single class only. When they are used to deal with multi-class problems, the GAs are run once for each class. Specifically, they would search rules predicting the first class in the first run; they would search rules predicting the second class in the second run and so on. Similar to the Pittsburgh approach, it is possible that an instance is matched by more than one rule predicting different classes on one hand and it is also possible that an instance is matched by none of any rule predicting any class on the other. This problem has not been addressed by these algorithms.

Although GA-based rule discovery approaches can produce accurate predictive models, they cannot determine the likelihood associated with their predictions. This prevents these techniques from being applicable to the task of predicting churn, which requires the ranking of subscribers according to their likelihood to churn (see, e.g., [Au, Chan, and Yao 2003; Mozer *et al.* 2000]).

#### 2.1.5.1 Parallel Genetic Algorithms

There are two main types of parallel GAs: single-population (e.g., [Abramson and Abela 1992; Bethke 1976; Fogarty and Huang 1991; Hauser and Manner 1993]) and multiple-population (e.g., [Grefenstette 1981; Grosso 1985; Tanese 1987]). A single-population parallel GA uses a single population of chromosomes and can be implemented on shared-memory and distributed-memory computers.

On a shared-memory computer, the population is stored in the shared memory and each processor evaluates the fitness of the chromosomes assigned to it and writes the fitness values back. One of the processors (the master processor) is responsible for applying the genetic operators (i.e., selection, crossover, and mutation) to produce the next generation.

On a distributed-memory computer, the population is stored in one processor (the master processor) and this processor sends the chromosomes to the other processors (the slave processors) for evaluation, collects the fitness values from the slave processors, and executes the genetic operators to produce the next generation. Communication occurs only when the slave processors receive their subsets of chromosomes for evaluation and when they return the fitness values.

A multiple-population parallel GA, which uses multiple populations of chromosomes, is a simple extension of the serial GA. It consists of multiple serial GAs, runs each of them on a processor of a parallel computer, and exchanges some individuals at certain predetermined times. The exchange (migration) of individuals from one population to another population is controlled by several parameters: the topology that defines the connection of the multiple populations, the number of individuals that are exchanged (the migration rate), and the frequency of migrations [Cantu-Paz 1998].

It is important to note that the performance of multiple-population parallel GAs that communicate every generation using a fully connected topology and the maximal migration rate closely resembles the performance of single-population parallel GAs [Cantu-Paz and Goldberg 1999]. Regardless of whether a parallel GA is single- or multiple-population, it has been shown in [Cantu-Paz and Goldberg 1999] that the optimal number of processors that minimizes the execution time is directly proportional to the square root of the population size and the fitness evaluation time. This theoretically confirms the claim that parallel GAs can reduce the execution time by using multiple processors.

Although parallel GAs have been used in many practical applications, they have not been applied to the data mining process. Our work represents the first attempt of using parallel GAs for data mining, in particular, the mining of fuzzy association rules.

### *2.1.6 Mining Rules in Time Series Data*

Many data mining methods have been proposed for time series classification in the literature (e.g., [Andre-Jonsson and Badal 1997; Bozkaya, Yazdani, and Ozsoyoglu 1997; Huang and Yu 1999; Indyk, Koudas, and Muthukrishnan 2000; Kalpakis, Gada, and Puttagunta 2001; Keogh and Smyth 1997; Park, Kim, and Chu 2001; Pratt and Fink 2002; Struzik and Siebes

1999; Wang and Wang 2000]). They typically focus on the introduction of new similarity measures as a subroutine to an existing classification algorithm (e.g., the 1-Nearest Neighbor algorithm). Although these methods may classify unseen time series accurately, they are not developed to explicitly reveal the underlying patterns hidden in the time series data. It has been shown in [Keogh and Kasetty 2003] that these methods perform poorly when compared to Euclidean distance in the experiments with some well-known time series data sets. Their dissatisfactory performance is perhaps due to the noisy and fuzzy nature of time series data.

The mining of *sequential patterns* [Agrawal and Srikant 1995], *frequent episodes* [Mannila, Toivonen, and Verkamo 1995], and *partial periodic patterns* [Han, Dong, and Yin 1999] have also been proposed in the literature. They are concerned with discovering event sequences (i.e., groups of events ordered by time). For example, an event can be the purchase of an item in market basket data or the increase of a stock price in financial data. Sequential patterns and frequent episodes represent frequent event sequences, whereas partial periodic patterns are event sequences that reoccur for a period or a set of periods. Furthermore, an algorithm for *clustering* time series has been presented in [Gavrilov *et al.* 2000]. The problem of *event detection* [Guralnik and Srivastava 1999] is concerned with finding time points at which the parameters in a data model or even the model itself are changed. It is important to note that these techniques are not developed for mining rules (i.e., the if-then relationships between events) in time series data.

An approach for discovering rules in time series data has been proposed in [Das *et al.* 1998]. It first forms a set of subsequences by sliding a window through a time series and clusters the subsequences by using a suitable measure of time series similarity (e.g., Euclidean distance). The center of each cluster is then encoded to a sequence of *primitive shapes*. From these sequences of primitive shapes, it mines a set of rules in the form of "if *A*, then *B* within time *T*" where *A* and *B* are sequences of primitive shapes. Each rule is associated with two parameters: *frequency* and *confidence*. This approach only discovers those rules whose frequencies and confidences are greater than or equal to the user-specified *minimum frequency* and *minimum confidence*, respectively. The discovered rules are then ranked by the *J-measure* [Smyth and Goodman 1992]. However, it can be difficult for the users to decide what the thresholds should be and the inappropriate setting can result in the neglect of some useful rules or the discovery of many irrelevant rules [Han and Kamber 2001; Hand, Mannila, and Smyth 2001].

Furthermore, E-Apriori and EH-Apriori can also be used to mine *n*-dimensional inter-transaction association rules from time series data [Lu, Han, and Feng 1998]. Similar to

other association rule mining algorithms, they require human users to supply thresholds. However, many users have no idea what these thresholds should be.

Recently, an information-theoretic fuzzy approach has been proposed in [Last, Klein, and Kandel 2001] for knowledge discovery in time series data. This approach first cleans and preprocesses the time series data based on signal processing techniques. It then constructs an information-theoretic connectionist network to identify the most useful features of the preprocessed data. A set of rules can be extracted from the connectionist network. The set of discovered rules is further reduced by 1) fuzzifying the rules; 2) reducing the set of fuzzified rules by conflict resolution; and 3) merging rules from the reduced set. This approach is to fuzzify crisp rules discovered in crisp data instead of handling fuzzy data and discovering fuzzy rules.

### 2.1.7 Attribute Clustering and Data Mining in Gene Expression Data

A gene expression data set from a microarray can be represented by an *expression table*, $T = \{w_{ij} \mid i = 1, \ldots, p, j = 1, \ldots, n\}$, where $w_{ij} \in \Re$ is the measured expression level of gene $g_i$ in sample $s_j$ [Domany 2003]. Each row in the expression table corresponds to one particular gene and each column to a sample. Such a data set is typically composed of a large number of genes but a small number of samples. For example, the *colon-cancer* data set [Alon *et al.* 1999] consists of 62 samples and 2,000 genes and the *leukemia* data set [Golub *et al.* 1999] contains 72 samples and 7,129 genes. The number of samples is likely to remain small for many areas of investigation, especially for human data, due to the difficulty of collecting and processing microarray samples [Piatetsky-Shapiro, Khabaza, and Ramaswamy 2003].

Classification and clustering are two major tasks in gene expression data analysis. Classification is concerned with assigning memberships to samples based on expression patterns, whereas clustering aims at finding new biological classes and refining existing ones [Piatetsky-Shapiro, Khabaza, and Ramaswamy 2003]. To cluster and/or recognize patterns in gene expression data sets, dimension problems are encountered. Typically, gene expression data sets consist of a large number of genes (attributes) but a small number of samples (tuples). Many data mining algorithms (e.g., classification [Agrawal *et al.* 1992; Chan and Wong 1990, 1991; Janikow 1998; Maimon, Kandel, and Last 1999; Quinlan 1993; Smyth and Goodman 1992], association rule mining [Agrawal, Imielinski, and Swami 1993; Agrawal and Srikant 1994; Delgado *et al.* 2003; Liu, Hsu, and Ma 1998; Park, Chen, and Yu 1995; Savasere, Omiecinski, and Navathe 1995], *pattern discovery* [Wong and Wang 1997, 2003], *linguistic summaries* [Kacprzyk and Zadrozny 2001; Yager 1991], and *context-*

*sensitive fuzzy clustering* [Hirota and Pedrycz 1999]) are developed and/or optimized to be scalable with respect to the number of tuples, so as not to handle a large number of attributes.

The distinctive characteristic of gene expression data allows clustering both genes and samples [Domany 2003; Jiang, Tang, and Zhang 2004]. With conventional clustering methods, the genes are considered as the tuples and the samples as the attributes. Thus it allows genes with similar expression patterns (i.e., *co-expressed genes*) to be identified [Jiang, Tang, and Zhang 2004]. On the other hand, to cluster samples, the samples are considered as the tuples and the genes as the attributes. The clustering analysis of samples is to find new biological classes or to refine existing ones [Piatetsky-Shapiro, Khabaza, and Ramaswamy 2003]. By this token, conventional clustering algorithms are able to group both samples and genes from the data.

To apply existing clustering algorithms to genes, various algorithms have been used. Well-known examples are: *k*-means algorithms [De Smet *et al.* 2002; Heyer, Kruglyak, and Yooseph 1999; Ralf-Herwig *et al.* 1999], Kohonen's *self-organizing maps* (SOM) [Tamayo *et al.* 1999], and various hierarchical clustering algorithms [Alon *et al.* 1999; Eisen *et al.* 1998].

As for distance measures, Euclidean distance and Pearson's correlation coefficient are widely used for clustering genes [Jiang, Tang, and Zhang 2004]. Given two genes $A_i$ and $A_j$, $i, j \in \{1, \ldots, p\}$, $i \neq j$, the Euclidean distance between $A_i$ and $A_j$ is given by:

$$d_E(A_i, A_j) = \sqrt{\sum_{k=1}^{n} (w_{ik} - w_{jk})^2}\ , \tag{2.1}$$

where $w \in \Re$ is the measured expression level.

$d_E$ measures the difference in the individual magnitudes of each gene. The genes regarded as similar by Euclidean distance may be very dissimilar in terms of their shapes or vice versa. For example, let us consider the two genes, which have an identical shape but only differ from each other by a large scaling factor. Their Euclidean distance is large although they have an identical shape. However, for gene expression data, the overall shapes of genes are of the primary interest [Jiang, Tang, and Zhang 2004]. It is for this reason that Euclidean distance may not be able to yield a good proximity measurement of genes.

The Pearson's correlation coefficient between genes $A_i$ and $A_j$ is defined as:

$$d_C(A_i, A_j) = \frac{\sum_{k=1}^{n}(w_{ik} - \overline{w}_i)(w_{jk} - \overline{w}_j)}{\sqrt{\sum_{k=1}^{n}(w_{ik} - \overline{w}_i)^2}\sqrt{\sum_{k=1}^{n}(w_{jk} - \overline{w}_j)^2}}, \quad (2.2)$$

where $\overline{w}_i$ and $\overline{w}_j$ are the means of $w_{ik}$ and $w_{jk}$, $k = 1, \ldots, n$, respectively. It considers each gene as a random variable with $n$ observations and measures the similarity between the two genes by calculating the linear relationship between the distributions of the two corresponding random variables. An empirical study [Heyer, Kruglyak, and Yooseph 1999] has shown that Pearson's correlation coefficient is not robust to outliers and it may assign high similarity score to a pair of dissimilar genes.

Recently, biclustering algorithms (e.g., [Cheng and Church 2000; Madeira and Oliveira 2004]) have been proposed to cluster both genes and samples simultaneously. Biclustering algorithms aim at identifying subsets of genes and subsets of samples by performing simultaneous clustering of both rows and columns of a gene expression table instead of clustering columns and rows (genes and samples) separately [Madeira and Oliveira 2004]. Specifically, these algorithms group a subset of genes and a subset of samples into a *bicluster* such that the genes and samples exhibit similar behavior. A popular measure of the coherence of genes and samples in a bicluster is the *mean squared residue* [Cheng and Church 2000]. Let $I \subseteq \{1, \ldots, p\}$ and $J \subseteq \{1, \ldots, n\}$. The mean squared residue of a bicluster, $T_{IJ} = \{w_{ij} \mid i \in I, j \in J\}$, is defined in [Cheng and Church 2000] as:

$$d_R(T_{IJ}) = \frac{1}{|I||J|}\sum_{i \in I, j \in J}(w_{ij} - w_{iJ} - w_{Ij} + w_{IJ})^2, \quad (2.3)$$

where $w_{iJ}$ is the mean of $w_{ij}$, $j \in J$, $w_{Ij}$ is the mean of $w_{ij}$, $i \in I$, and $w_{IJ}$ is the mean of $w_{ij}$, $i \in I, j \in J$. A bicluster is formed if its mean squared residue is less than or equal to a user-specified threshold.

Gene selection is another important step to further narrowing down the attribute number prior to data mining. A good number of algorithms have been developed for this purpose (e.g., [Mukherjee *et al.* 2003; Pan 2002]). To select genes, the *t*-value is widely used in the literature [Piatetsky-Shapiro, Khabaza, and Ramaswamy 2003]. Assuming that there are two classes of samples in a gene expression data set, the *t*-value $t(A_i)$ for gene $A_i$ is given by:

$$t(A_i) = \frac{\mu_1 - \mu_2}{\sqrt{\sigma_1^2 / n_1 + \sigma_2^2 / n_2}}, \tag{2.4}$$

where $\mu_r$ and $\sigma_r$ are the mean and the standard deviation of the expression levels of gene $A_i$ for class $r$, respectively, and $n_r$ is the number of samples in class $r$ for $r$ = 1, 2. The top genes ranked by the *t*-value can then be selected for data mining. When there are multiple classes of samples, the *t*-value is typically computed for one class versus all the other classes.

A weakness of using the *t*-value to select genes is the redundancy among the selected genes [Ding and Peng 2003; Xing, Jordan, and Karp 2001; Yu and Liu 2004]. To solve this problem, methods that can handle both the *gene-class relevance* and the *gene-gene redundancy* have been proposed (e.g., [Ding and Peng 2003; Xing, Jordan, and Karp 2001; Yu and Liu 2004]). These methods typically use some metric to measure the gene-class relevance (e.g., mutual information, the *F*-test value [Ding and Peng 2003], information gain, *symmetrical uncertainty* [Yu and Liu 2004], etc.) and employ the same or a different metric to measure the gene-gene redundancy (e.g., mutual information, the $L_1$ distance [Ding and Peng 2003], Pearson's correlation coefficient, etc.). To find a subset of relevant but non-redundant genes, they usually use a methodology called *redundant cover* to eliminate redundant genes with respect to a subset of genes selected according to the metric for measuring the gene-class relevance and the gene-gene redundancy (see, e.g., [Xing, Jordan, and Karp 2001; Yu and Liu 2004]). Another approach to doing so combines the metric for measuring the gene-class relevance and that for measuring the gene-gene redundancy into a single criterion function and then selects genes so that the criterion function is optimized (see, e.g., [Ding and Peng 2003]).

It is important to note that both the *t*-value and the methods that handle the gene-class relevance and the gene-gene redundancy can only be used to select genes when the samples are pre-classified.

### 2.1.8 Handling Both Transaction and Relational Data

To deal with both transaction and relational data, there is a related, but not directly applicable, work presented in [Dhar and Tuzhilin 1993]. In [Dhar and Tuzhilin 1993], a database containing a customer table and a transaction table is described. Some techniques to discover rules from this database have been proposed. However, the problem of how to handle both of transaction and relational data has not been discussed. It is not clear how this approach can be used to handle the union of transaction and relational data in general.

### 2.1.9 Data Transformation

Existing data mining techniques do not provide any explicit methodology for data transformation. Related, but not directly applicable, work includes *attribute-oriented induction* [Cai, Cercone, and Han 1991; Han, Cai, and Cercone 1992, 1993; Han and Fu 1996] and an *abstract-driven* approach [Dhar and Tuzhilin 1993]. As a means of rule discovery, both of them do not provide any formalism for the problem of data transformation.

Attribute-oriented induction makes use of *concept hierarchies* defined by knowledge engineers or domain experts. Each of these concept hierarchies defines a sequence of mappings from a set of concepts to their higher-level correspondences according to a general-to-specific ordering with the most general concept defined by a reserved word "*any*" and the most specific concepts corresponding to the specific data in the database [Han and Fu 1996]. An attribute is at the desirable level if it contains no more distinct values than its *attribute threshold* – a small integer that can be specified by users or set to the default value. The *minimum desirable level* of an attribute is the level in the concept hierarchy such that the attribute would have more distinct values than its threshold when it was specialized to one level lower. Attribute-oriented induction transforms an *initial relation*, which contains the data relevant to the task the user has on hand, into *generalized relations* using *generalization*, *attribute removal*, *concept tree ascension*, and *vote propagation* [Han, Cai, and Cercone 1993]. In a generalized relation, some or all of its attribute values are higher-level concepts, that is, non-leaf nodes in the concept hierarchies. A generalized relation becomes a *prime relation* if all of its attributes are at the minimum desirable level. This generalization process continues recursively until a prime relation is obtained. A set of rules can then be discovered from the prime relation using *rule transformation* [Han, Cai, and Cercone 1993].

Another related, but not directly applicable, work is the *abstract-driven* approach, which is based on a *vocabulary*, a set of *classification hierarchies*, and a set of *abstraction functions* [Dhar and Tuzhilin 1993]. The vocabulary consists of a set of *user-defined predicates*, which are defined as disjunctions of conjunctive clauses, where each atomic formula is either a database relation, or another previously introduced user-defined predicate, or a condition involving attributes from database relations [Dhar and Tuzhilin 1993]. These user-defined predicates are grouped into classification hierarchies such that a partial order is imposed on all the predicates in the vocabulary based on the logical implication [Dhar and Tuzhilin 1993]. As a result, a predicate at higher level of a hierarchy logically implies those predicates at lower level of the same hierarchy. Furthermore, an abstraction function of an

attribute maps the domain values of the attribute into some other domain; for example, the abstraction function *year* maps a date into a year by "extracting" the year from the date [Dhar and Tuzhilin 1993]. These abstraction functions can also be grouped into *abstraction hierarchies* by their composition [Dhar and Tuzhilin 1993]. Based on the vocabulary, the classification hierarchies, and the abstraction functions, a new relation, called *abstract*, whose attributes come from the union of the above three components is generated. A set of interesting patterns can then be extracted from the abstract.

Each abstraction function used in [Dhar and Tuzhilin 1993] can only deal with one attribute. This prohibits the generation of new attributes that are composed of more than one primitive attribute. For instance, the attribute "commission," which is calculated by the multiplication of the primitive attributes "transaction amount" and "commission rate," cannot be produced. The applicability of such abstraction functions is therefore quite restrictive. Although the abstract generated by the abstract-driven approach can be considered as a set of transformed data, this technique is developed for the task of pattern discovery but not for the task of data transformation.

Furthermore, since both of attribute-oriented induction [Cai, Cercone, and Han 1991; Han, Cai, and Cercone 1992, 1993; Han and Fu 1996] and the abstract-driven approach [Dhar and Tuzhilin 1993] are not developed for the task of data transformation, they do not provide any explicit methodology for defining the necessary components (e.g., concept hierarchies, vocabularies, classification hierarchies, abstraction functions, etc.) for performing data transformation. However, it is difficult for human users to express or formalize their knowledge and experience in most situations [Buchanan *et al.* 1983; Johnson-Laird 1989]. It is especially difficult when there is no explicit methodology to do so. These methods are therefore inadequate for casual users to perform data transformation due to the lack of explicit methodology for specifying the process of data transformation.

## 2.2 Meta-Mining

*Meta-mining* is concerned with mining previously discovered patterns, which are typically represented in the form of production (if-then) rules [Au and Chan 2002a, 2002b, 2005; Roddick and Spiliopoulou 2002; Roddick and Spiliopoulou 2000; Kurgan and Cios 2004]. It can be used to discover many useful patterns that existing data mining techniques (e.g., [Agrawal *et al.* 1992; Agrawal, Imielinski, and Swami 1993a, 1993b; Agrawal and Shafer 1996; Agrawal and Srikant 1994; Bradley, Fayyad, and Reina 1998; Cheeseman and Stutz 1996; Cheung *et al.* 1996a; Ganti *et al.* 1999b; Han and Fu 1995; Houtsma and Swami 1995; Lu, Setiono, and Liu 1995; Mannila, Toivonen, and Verkamo 1994; Mehta, Agrawal, and

Rissanen 1996; Park, Chen, and Yu 1995a, 1995b; Savasere, Omiecinski, and Navathe 1995; Shafer, Agrawal, and Mehta 1996; Srikant and Agrawal 1995, 1996; Zhang, Ramakrishnan, and Livny 1996]) are not developed to mine for. These patterns are represented in the form of production rules and they are called *meta-rules* because they are rules about rules. The discovered meta-rules are arguably closer to the forms of knowledge that might be considered interesting [Roddick and Spiliopoulou 2002]. For example, the meta-rule "High Income is becoming more associated with Mercedes Benz Ownership" is arguably more interesting than the rule "High Income is associated with Mercedes Benz Ownership."

Although meta-mining is an important problem, it has received little attention in the literature. To our best knowledge, in addition to our previous work [Au and Chan 2002a, 2002b, 2005], this problem has only been studied in [Roddick and Spiliopoulou 2000; Kurgan and Cios 2004].

A framework for analyzing data mining results, called *higher order mining*, has been proposed in [Roddick and Spiliopoulou 2000]. In this framework, a *first order rule* is a rule discovered in a data set, whereas a *second order rule* is a sequence of first order rules discovered in different data sets. Given a second order rule, the interestingness measures (e.g., the Dempster-Shafer measure [Dempster 1967; Shafer 1976], support and confidence [Agrawal, Imielinski, and Swami 1993b], *conviction* [Brin *et al.* 1997], the chi-squared measure [Brin, Motwani, Silverstein 1997], the *J*-measure [Smyth and Goodman 1992], the *adjusted residual* and *weight of evidence* [Chan and Wong 1990, 1991], etc.) of its first order rules can be considered as a time series. One can then apply time series analysis (e.g., ARIMA [Box, Jenkins, and Reinsel 1994]) to analyze the time series. Some of the first order rules of a second order rule may not hold in the corresponding data sets because their interestingness measures may fall below the user-specified thresholds, for example. The time series may therefore contain missing values. However, time series analysis is not developed to deal with missing values. Furthermore, the discovered patterns are embedded in the parameters of the statistical model constructed and hence they are unnatural for human users to comprehend.

This framework has also been used in a meta-mining system proposed in [Kurgan and Cios 2004] to generate data models from already generated data models. The system 1) divides a data set into a number of subsets; 2) generates a set of rule from each data subset using a supervised learning algorithm; and 3) mines a set of (meta-) rules from the rule sets using the same algorithm. The discovered meta-rules can then be used for classification. The experimental results reported in [Kurgan and Cios 2004] show that the performance of the meta-rules discovered from the already discovered rule sets is a little inferior to that of

the rules discovered from the data sets in terms of classification rate.

[Roddick and Spiliopoulou 2000] is concerned with revealing changes in rule sets, whereas [Kurgan and Cios 2004] aims at discovering regularities in rule sets. None of them is developed to uncover all of the regularities, differences, and changes.

A related, but not directly applicable, work is *meta-learning* [Prodromidis, Chan, and Stolfo 2000]. Given a collection of data sets or data subsets, it runs a supervised learning algorithm or different learning algorithms on each of them. It then combines the predictions of the learned classifiers to produce a *meta-classifier* by recursively learning *arbiter* and *combiner* models in a bottom-up tree manner [Prodromidis, Chan, and Stolfo 2000]. An arbiter plays the role as a judge whose own prediction is used if the participating classifiers cannot reach a consensus decision. A combiner can further be classified as *class-combiner*, *class-attribute-combiner*, and *binary-class-combiner*. In a class-combiner, the meta-level training instances consist of the correct classification and the predictions; in a class-attribute-combiner, the instances are formed as in a class-combiner with the addition of the attribute vectors; and a binary-class-combiner, the instances are composed in a manner similar to that in a class-combiner except that each prediction has *l* binary predictions where *l* is the number of classes [Prodromidis, Chan, and Stolfo 2000]. An example of the patterns revealed by meta-learning is "given a record, if classifier 1 classifies it into class A and classifier 2 classifies it into class B, then it is classified into class A." Meta-learning indeed is not developed to reveal the underlying patterns hidden in the classifiers.

### 2.2.1 Mining Regularities in Multiple Data Sets

For an interstate or international company, which comprises a number of offices at different geographical locations and has each office (or group of offices) to maintain its own database, to better make decisions, it needs to mine multiple databases throughout their offices [Zhang, Wu, and Zhang 2003]. However, existing data mining techniques (e.g., [Agrawal *et al.* 1992; Agrawal, Imielinski, and Swami 1993a, 1993b; Agrawal and Shafer 1996; Agrawal and Srikant 1994, 1995; Bradley, Fayyad, and Reina 1998; Cheeseman and Stutz 1996; Cheung *et al.* 1996a; Ganti *et al.* 1999b; Han, Dong, and Yin 1999; Han and Fu 1995; Houtsma and Swami 1995; Lu, Setiono, and Liu 1995; Mannila, Toivonen, and Verkamo 1994, 1995; Mehta, Agrawal, and Rissanen 1996; Park, Chen, and Yu 1995a, 1995b; Savasere, Omiecinski, and Navathe 1995; Shafer, Agrawal, and Mehta 1996; Srikant and Agrawal 1995, 1996; Zhang, Ramakrishnan, and Livny 1996]) are developed to handle a single database and they are not directly applicable to mining multiple databases.

Recently, several techniques for data mining in multiple databases, including [Liu, Lu,

and Yao 1998; Ribeiro, Kaufman, and Kerschberg 1995; Wrobel 1997; Yao and Liu 1997; Zhong, Yao, and Ohsuga 1999], have been proposed in the literature. These multi-database mining techniques typically involve 1) selecting relevant data from multiple databases; 2) extracting the selected data to amass a single database; and 3) applying existing data mining techniques, such as association rule mining (e.g., [Agrawal, Imielinski, and Swami 1993b; Agrawal and Shafer 1996; Agrawal and Srikant 1994; Cheung *et al.* 1996a; Han and Fu 1995; Houtsma and Swami 1995; Mannila, Toivonen, and Verkamo 1994; Park, Chen, and Yu 1995a, 1995b; Savasere, Omiecinski, and Navathe 1995; Srikant and Agrawal 1995, 1996]), classification (e.g., [Agrawal *et al.* 1992; Agrawal, Imielinski, and Swami 1993a; Lu, Setiono, and Liu 1995; Mehta, Agrawal, and Rissanen 1996; Shafer, Agrawal, and Mehta 1996]), and clustering (e.g., [Bradley, Fayyad, and Reina 1998; Cheeseman and Stutz 1996; Ganti *et al.* 1999b; Zhang, Ramakrishnan, and Livny 1996]), to the single database.

They can therefore discover only the same kind of patterns as conventional (single-) database mining techniques. They are unable to discover some patterns such as "in general, if a customer is married and middle-aged, then he/she gets a home mortgage." They also cannot discover such patterns as "in an exceptional manner, if a customer is single and tertiary educated, then he/she has more than one car." The former represents a regular pattern supported by many branches of an international company, whereas the latter represents a differential pattern supported by only a few branches.

Recently, the mining of *high-vote patterns* in multiple databases has been proposed in [Zhang, Zhang, and Wu 2004]. Given the $m$ databases, $D_1$, …, $D_m$ in the $m$ branches of a company, a conventional (single-) database mining algorithm is first applied to $D_i$ to discover a set of patterns, $R_i$, $i = 1, \ldots, m$. Let $R = \{r_j \mid r_j \in R_1 \cup \ldots \cup R_m\}$ and $n = |R|$. The *average voting rate*, *AVR*, is given by:

$$AVR = \frac{1}{n}\sum_{j=1}^{n} voting(r_j), \tag{2.5}$$

where $voting(r_j)$ is the *voting rate* of $r_j$ and is calculated by:

$$voting(r_j) = \frac{|\{R_i \mid r_j \in R_i, i = 1, ..., m\}|}{m}. \tag{2.6}$$

The interestingness of $r_j$, $interest(r_j)$, is then defined in [Zhang, Zhang, Wu 2004] as:

$$interest(r_j) = \frac{voting(r_j) - AVR}{1 - AVR}. \quad (2.7)$$

A pattern is high-voting if its voting rate is greater than the average voting rate and its interestingness is greater than or equal to a user-specified threshold [Zhang, Zhang, and Wu 2004]. A weakness of this approach is that many users do not have any idea what the threshold should be. Some useful patterns may be missed if it is set too high, whereas many irrelevant patterns may be found if it is set too low.

Instead of concatenating multiple data sources to amass a single data set, a set of association rules can be synthesized from the association rules discovered in the data sources [Wu and Zhang 2003]. The supports of these association rules are estimated in terms of the supports of the underlying association rules and the popularities of the data sources. The experimental results in [Wu and Zhang 2003] show that the synthesized rules are a good approximate of the rules discovered in the concatenated data set. Although this synthesizing technique starts from multiple data sources, it is not developed to discover the regularities in the rule sets.

### 2.2.2 Mining Differences in Multiple Data Sets

In [Ganti *et al.* 1999a], a framework has been proposed to measure the difference between two data sets by building two models (one from each data set) and measuring the amount of work required to transform one model to the other. It results in a real number to reflect to which degree the two data sets differ from each other. However, it is not developed to explicitly reveal what the differences are.

Recently, the mining of *exceptional patterns* in multiple databases in the context of association rules has been proposed in [Zhang, Zhang, and Wu 2004]. Given the $m$ databases, $D_1, \ldots, D_m$ in the $m$ branches of a company, an association rule mining algorithm is first applied to $D_i$ to discover a set of patterns (i.e., association rules), $R_i$, $i = 1, \ldots, m$. Let $R = \{r_j \mid r_j \in R_1 \cup \ldots \cup R_m\}$. The interestingness of $r_j$, *exceptional interest*($r_j$), is defined in [Zhang, Zhang, and Wu 2004] as:

$$exceptional\ interest(r_j) = \frac{voting(r_j) - AVR}{-AVR}, \quad (2.8)$$

where $voting(r_j)$ is the voting rate of $r_j$ given by Equation (2.6) and $AVR$ is the average voting rate calculated by Equation (2.5). In addition to this measure, another interestingness

measure of $r_j$ with respect to $D_i$ is also defined in [Zhang, Zhang, and Wu 2004] as:

$$exceptional\ interest_i(r_j) = \frac{support_i(r_j) - minsupport_i}{minsupport_i}, \quad (2.9)$$

where $support_i(r_j)$ is the support of $r_j$ in $D_i$ and $minsupport_i$ is the user-specified minimum support for mining patterns in $D_i$. A pattern $r_j$ is exceptional if 1) its voting rate is greater than the average voting rate and *exceptional interest*($r_j$) is greater than or equal to a user-specified threshold; and 2) *exceptional interest*$_i$($r_j$) is greater than or equal to another user-specified threshold for all $i \in \{i \mid r_j \in R_i\}$. Similar to the mining of high-vote patterns, a weakness of this approach is that many users have no idea what the thresholds should be. If they are set too high, some useful patterns may be missed; but if they are set too low, many irrelevant patterns may be found.

### 2.2.3 Mining Changes in Multiple Data Sets

To deal with the data collected in different time periods, the maintenance of discovered association rules (e.g., FUP [Cheung *et al.* 1996b]) and *active data mining* [Agrawal and Psaila 1995] have been proposed in the literature. Incremental updating techniques (e.g., FUP) can be used to update the discovered association rules if there are additions, deletions, or modifications of any tuples in a database after a set of association rules has been discovered. Active data mining is concerned with representing and querying the shape of the history of parameters for the discovered association rules. Although these techniques can be used to track the variations in supports and confidences of association rules, both of them are not developed to discover and predict rule changes.

Although the mining of rule changes over time is an important problem, it has received little attention. To our best knowledge, in addition to our previous work [Au and Chan 2002a, 2002b, 2005], this problem has only been studied in [Liu *et al.* 2000], [Liu, Hsu, and Ma 2001], and [Roddick and Spiliopoulou 2000]. [Liu *et al.* 2000] is concerned with finding whether a decision tree built in a time period is applicable in other time periods. Given two data sets collected in two different time periods, this method builds a decision tree based on one of the data sets and then builds another based on the other data set such that the latter tree uses the same attribute and chooses the same cut point for the attribute as the former at each step of partitioning. This method can be used to identify three categories of changes in the context of decision tree building: *partition change*, *error rate change*, and *coverage change* [Liu *et al.* 2000]. Compared to [Liu *et al.* 2000], instead of building a decision tree in the next time instance to ensure that it resembles the first, our goal is to

discover the changes in rules discovered in different time periods.

Following the idea presented in [Liu *et al.* 2000], a method has been proposed in [Liu, Hsu, and Ma 2001] to find whether a set of association rules discovered in a time period is applicable in other time periods. To do so, it employs chi-square test to determine whether there are any changes in the supports and confidences of the association rules discovered in different time periods. Unlike this method, our goal is to mine (meta-) rules to represent the changes and to predict any changes in the future.

If the underlying data sets are collected in different time periods, the higher order mining framework proposed in [Roddick and Spiliopoulou 2000] can be used to find the changes in the discovered rules. Given a second order rule, the interestingness measures (e.g., the Dempster-Shafer measure [Dempster 1967; Shafer 1976], support and confidence [Agrawal, Imielinski, and Swami 1993b], conviction [Brin *et al.* 1997], the chi-squared measure [Brin, Motwani, Silverstein 1997], the *J*-measure [Smyth and Goodman 1992], the adjusted residual and weight of evidence [Chan and Wong 1990, 1991], etc.) of its first order rules can be considered as a time series, which can be analyzed by time series analysis (e.g., ARIMA [Box, Jenkins, and Reinsel 1994]). The time series may contain missing values because some of the first order rules of a second order rule may not hold in the corresponding data sets as their interestingness measures may fall below the user-specified thresholds, for example. However, time series analysis is not developed to deal with missing values. Furthermore, the discovered patterns are embedded in the parameters of the statistical model constructed. They are therefore not natural for human users to comprehend.

# Chapter 3
# The Proposed Approach

In this chapter, we define the problems of mining regularities, differences, and changes in rule sets and propose a new meta-mining approach to solving them. The proposed approach is composed of a collection of techniques for 1) generating fuzzy sets from data automatically; 2) using linguistic variables and linguistic terms to represent the discovered regularities, differences, and changes; 3) exploiting the scalability of parallel computer systems to mine meta-rules efficiently; 4) grouping and selecting a subset of attributes for meta-mining; and 5) enabling the mining of meta-rules involving attributes that are not originally contained in the database. This chapter also describes how these techniques fit into the meta-mining approach.

## 3.1 A Formal Problem Description

Let us suppose that there is a collection of data sets, $D_j$, $j$ = 1, …, $n$. A set of rules, $R_j = \{r_{j1}, ..., r_{js_j}\}$, is mined from $D_j$, $j$ = 1, …, $n$. A rule, $r_{ju} \in R_j$, is an implication of the form $X \Rightarrow Y$, where $X$ and $Y$ are conjunctions of conditions. The antecedent and the consequent of the rule $X \Rightarrow Y$ are denoted as *antecedent*($X \Rightarrow Y$) = $X$ and *consequent*($X \Rightarrow Y$) = $Y$, respectively.

**Example 3.1** An example rule, $r$, is:

$$Sex = Male \wedge Education = Tertiary \wedge Income = High$$
$$\Rightarrow Mercedes\ Benz\ Ownership = True.$$

The antecedent and the consequent of this rule are:

$$antecedent(r) = (Sex = Male \wedge Education = Tertiary \wedge Income = High)$$

and

$$consequent(r) = (Mercedes\ Benz\ Ownership = True),$$

respectively. □

Given a rule, $X \Rightarrow Y$, let *condition*($X$) and *condition*($Y$) be the sets of all the conditions in its antecedent and consequent, respectively. The set of conditions in the rule $X \Rightarrow Y$ is then given by *condition*($X \Rightarrow Y$) = *condition*($X$) $\cup$ *condition*($Y$). Let us further suppose that $condition(R_j) = \bigcup_{u=1}^{s_j} condition(r_{ju})$.

**Example 3.2** Let us consider the rule $r$ given in Example 3.1. The set of conditions in its antecedent is:

$$condition(antecedent(r)) = \{Sex = Male, Education = Tertiary, Income = High\},$$

the set of conditions in its consequent is:

$$condition(consequent(r)) = \{Mercedes\ Benz\ Ownership = True\},$$

and the set of conditions in the rule is:

$$condition(r) = \{Sex = Male, Education = Tertiary, Income = High, Mercedes\ Benz\ Ownership = True\}.$$

□

In general, the rule $X \Rightarrow Y$ is associated with one or more interestingness measures (e.g., the Dempster-Shafer measure [Dempster 1967; Shafer 1976], *support* and *confidence* [Agrawal, Imielinski, and Swami 1993b], *conviction* [Brin *et al.* 1997], the chi-squared measure [Brin, Motwani, Silverstein 1997], the *J*-measure [Smyth and Goodman 1992], the *adjusted residual* and *weight of evidence* [Chan and Wong 1990, 1991], etc.). We denote the interestingness measure of the rule $X \Rightarrow Y$ in $D_j$ as $interestingness_j(X \Rightarrow Y)$.

**Example 3.3** In an association rule mining algorithm, the interestingness of a rule such as that in Example 3.1 is measured in terms of support and confidence. It holds in data set $D_j$ with support,

$$support_j(r) = \frac{|\sigma_{Sex=Male \wedge Education=Tertiary \wedge Income=High \wedge Mercedes\ Benz\ Ownership=True}(D_j)|}{|D_j|}, \quad (3.1)$$

and confidence,

$$confidence_j(r) = \frac{|\sigma_{Sex=Male \wedge Education=Tertiary \wedge Income=High \wedge Mercedes\ Benz\ Ownership=True}(D_j)|}{|\sigma_{Sex=Male \wedge Education=Tertiary \wedge Income=High}(D_j)|}, \quad (3.2)$$

where $\sigma$ denotes the SELECT operation in *relational algebra* and $|S|$ denotes the cardinality of set $S$. □

### 3.1.1 Mining Regularities and Differences

From $R_1, \ldots, R_n$, we aim at mining a set of meta-rules to reveal the underlying regularities hidden in the rule sets and the differences between different rule sets.

**Definition 3.1** A *meta-rule* mined from rule sets $R_1, \ldots, R_n$ is an implication of the form:

$$X \Rightarrow Y,$$

where $X$ and $Y$ are conjunctions of conditions such that $condition(X) \subseteq \bigcup_{j=1}^{n} condition(R_j)$, $condition(Y) \subseteq \bigcup_{j=1}^{n} condition(R_j)$, and $condition(X) \cap condition(Y) = \varnothing$. □

Rather than being supported by data records, a meta-rule is supported by the rules in the rule sets. We say that a rule supports a meta-rule if the set of conditions in the meta-rule is a subset of that in the rule.

**Definition 3.2** A meta-rule, $X \Rightarrow Y$, mined from rule sets $R_1, \ldots, R_n$, is supported by a set of rules:

$$\mathcal{R}(X \Rightarrow Y) = \{r \mid r \in R_1 \cup \ldots \cup R_n, condition(X) \cup condition(Y) \subseteq condition(r)\}.$$

□

A meta-rule represents an association relationship in common in the rule sets if many rules support it. In other words, it represents an underlying regularity hidden in the rule sets.

**Definition 3.3** A meta-rule, $X \Rightarrow Y$, mined from rule sets $R_1, \ldots, R_n$, represents a regularity embedded in them if $|\mathcal{R}(X \Rightarrow Y)|$ is *sufficiently large*. We refer to this meta-rule as a *regular meta-rule*. □

On the other hand, a meta-rule represents a distinctive association relationship in the rule sets if only a few rules support it. In other words, it represents a difference between the rule sets.

**Definition 3.4** A meta-rule, $X \Rightarrow Y$, mined from rule sets $R_1, \ldots, R_n$, represents a difference between them if $|\mathcal{R}(X \Rightarrow Y)|$ is *sufficiently small*. We refer to this meta-rule as a *differential meta-rule*. □

To reveal regularities and differences in rule sets, we mine regular and differential meta-rules from the rule sets, respectively.

**Example 3.4** Let us consider rule sets $R_1, \ldots, R_5$, each of which contains a set of association rules. They are given in the following:

$$R_1: \{i_1, i_2\} \Rightarrow \{i_3\}$$
$$\{i_4\} \Rightarrow \{i_1\}$$

$$R_2: \{i_1, i_2\} \Rightarrow \{i_3\}$$
$$\{i_2, i_3, i_5\} \Rightarrow \{i_4\}$$
$$\{i_2, i_3\} \Rightarrow \{i_4\}$$

$$R_3: \{i_2, i_3, i_5\} \Rightarrow \{i_4\}$$

$$R_4: \{i_1, i_2\} \Rightarrow \{i_3\}$$
$$\{i_2, i_3, i_5\} \Rightarrow \{i_4\}$$

$$R_5: \{i_1, i_2\} \Rightarrow \{i_3\},$$

where $i_1, \ldots, i_5$ are items.

The meta-rule $\{i_2, i_3\} \Rightarrow \{i_4\}$ is supported by the following rules:

$$R_2: \{i_2, i_3, i_5\} \Rightarrow \{i_4\}$$
$$\{i_2, i_3\} \Rightarrow \{i_4\}$$

$$R_3: \{i_2, i_3, i_5\} \Rightarrow \{i_4\}$$

$$R_4\text{: } \{i_2, i_3, i_5\} \Rightarrow \{i_4\},$$

whereas the meta-rule $\{i_4\} \Rightarrow \{i_1\}$ is supported by the following rule:

$$R_1\text{: } \{i_4\} \Rightarrow \{i_1\}.$$

The former and the latter meta-rules are supported by 44.4% (= 4 / 9) and 11.1% (= 1 / 9) of all the rules, respectively.

A straightforward approach to determining whether a meta-rule is supported by a sufficiently large or small number of rules is to have a user supply thresholds. For example, if the threshold for determining regular meta-rules is set to 40%, the former meta-rule is found to be regular; and if the threshold for determining differential meta-rules is set to 15%, the latter meta-rule is found to be differential.

A weakness of this approach is that it is difficult to determine what the thresholds should be. An effective algorithm should use an objective measure to mine regular and differential meta-rules, instead of having a user supply thresholds. □

### 3.1.2 Mining Changes

We are also concerned with mining a set of meta-rules to reveal how the rules in the rule sets change over time.

Now, let us further suppose that $D_j$ is collected in time periods $t_j$, $j = 1, \ldots, n$, where $t_1, \ldots, t_n$ are consecutive and $t_j$ happens before $t_k$ if $j < k$. Let us consider rules $r_{ju} \in R_j$ and $r_{kv} \in R_k$, $j, k \in \{1, \ldots, n\}$, $j < k$. These represent the same association relationship if, and only if, *antecedent*($r_{ju}$) = *antecedent*($r_{kv}$) and *consequent*($r_{ju}$) = *consequent*($r_{kv}$).

**Definition 3.5** Given a rule, $r_{ju} \in R_j$, if there exists another rule, $r_{kv} \in R_k$, $j < k$, such that *antecedent*($r_{ju}$) = *antecedent*($r_{kv}$) and *consequent*($r_{ju}$) = *consequent*($r_{kv}$), $r_{ju}$ is *equivalent* to $r_{kv}$, denoted as $r_{ju} \equiv r_{kv}$, because they represent the same association relationship. □

It is important to note that although $r_{ju} \equiv r_{kv}$, its interestingness measure in $t_j$ may be different from that in $t_k$ because the rule may change as will be discussed in Definitions 3.6–3.8.

**Definition 3.6** Given two rules, $r_{ju} \in R_j$ and $r_{kv} \in R_k$, $j < k$, such that $r_{ju} \equiv r_{kv}$, $r_{ju}$ *changes* during the period from $t_j$ to $t_k$ if $interestingness_j(r_{ju}) \neq interestingness_k(r_{ju})$. We say that $r_{ju}$ is a *changed rule* in $t_k$. □

It is possible that rule $r_{ju}$ is found in $R_j$ but not in $R_k$ because it is interesting in $t_j$ but it becomes uninteresting in $t_k$, $j < k$.

**Definition 3.7** Given $R_j$ and $R_k$, $j < k$, if $r_{ju} \in R_j$ and there does not exist $r_{kv} \in R_k$ such that $r_{ju} \equiv r_{kv}$, we say that $r_{ju}$ is *perished* in $t_k$ and $r_{ju}$ is a *perished rule* in $t_k$. In this case, the interestingness measure of $r_{ju}$ in $t_k$ is missing, denoted as $interestingness_k(r_{ju}) = ?$. □

On the other hand, it is also possible that $r_{kv}$ is not found in $R_j$ but is found in $R_k$ because it is uninteresting in $t_j$ but it becomes interesting in $t_k$, $j < k$.

**Definition 3.8** Given $R_j$ and $R_k$, $j < k$, if $r_{kv} \in R_k$ and there does not exist any $r_{ju} \in R_j$ such that $r_{ju} \equiv r_{kv}$, we say that $r_{kv}$ is *added* in $t_k$ and $r_{kv}$ is an *added rule* in $t_k$. In this case, the interestingness measure of $r_{kv}$ in $t_j$ is missing, denoted as $interestingness_j(r_{kv}) = ?$. □

An added rule or a perished rule is a special case of a changed rule. It is special in that an added rule's interestingness measure changes from below a threshold to above it, whereas a perished rule's interestingness measure changes in the reverse direction, from above the threshold to below it. The threshold can be specified by a user or determined by an objective means. Revealing how a rule changed in the past allows one to predict whether it will be added or perished or to what degree it will change in the future.

For each rule in $R_1 \cup \ldots \cup R_n$, we are interested in mining a set of meta-rules to represent the regularities governing how the rule changes during the period from $t_1$ to $t_n$. We refer to these meta-rules as *change meta-rules* because they represent how the rule changes over time.

**Definition 3.9** For $r \in R_1 \cup \ldots \cup R_n$, a *change meta-rule* is an implication of the form:

$$L^r_{j_1} = l^r_{p_1} \wedge \ldots \wedge L^r_{j_h} = l^r_{p_h} \Rightarrow L^r_{j_q} = l^r_{p_q},$$

where $L^r_{j_k}$ is an attribute representing $\Delta interestingness_{j_k}(r) = interestingness_{j_k+1}(r) - interestingness_{j_k}(r)$ (i.e., the difference in

the interestingness measure of $r$ during the period from $t_{j_k}$ to $t_{j_k+1}$) and $l^r_{p_k}$ is an attribute value in $dom(L^r_{j_k})$, which denotes the domain of $L^r_{j_k}$, for $k$ = 1, …, $h$, $q$ and $j_1 < \ldots < j_h < j_q$. □

The mining of change meta-rules allows the examination of the regularities governing how a rule changes during a period $t_1$ to $t_n$. The discovered meta-rules can also be used to predict how the rule will change in $t_{n+1}$. The ability to predict how rules will change allows accurate results to be achieved when the discovered rules in the past are used for classification in the future.

**Example 3.5** Let us consider the association rules of items $i_1$, $i_2$, $i_3$, and $i_4$ discovered in three consecutive time periods, $t_1$, $t_2$, and $t_3$. Assume that the association rule discovered in time period $t_1$ is:

$$r: \quad \{i_1, i_2, i_3\} \Rightarrow \{i_4\}$$

whose support and confidence in $t_1$ are $support_1(r)$ = 37.8% and $confidence_1(r)$ = 95.0%, respectively.

In time period $t_2$, the association rule becomes:

$$r': \quad \{i_1, i_2, i_3\} \Rightarrow \{i_4\}$$

whose support and confidence in $t_2$ are $support_2(r)$ = 34.9% and $confidence_2(r)$ = 94.8%, respectively.

Then in time period $t_3$, the association rule becomes:

$$r'': \quad \{i_1, i_2, i_3\} \Rightarrow \{i_4\}$$

whose support and confidence in $t_3$ are $support_3(r)$ = 28.4% and $confidence_3(r)$ = 94.5%, respectively.

The support of the association rule decreases in the period from $t_1$ to $t_2$ and in the period from $t_2$ to $t_3$. A change meta-rule of support mined from these rules would be:

*Change in support in this period = Fairly decrease*

$\Rightarrow$ *Change in support in next period = Highly decrease*.

This meta-rule of support states that "if the change in support in this period moderately decreases, then the change in support in next period will decrease significantly." The support of the association rule in $t_j$ can then be predicted given the support of this rule in $t_{j-1}$ and that in $t_{j-2}$.

On the other hand, the confidence of the association rule is more or less the same in the period from $t_1$ to $t_2$ and in the period from $t_2$ to $t_3$. A change meta-rule of confidence discovered in these rules would be:

*Change in confidence in this period = More or less the same*

$\Rightarrow$ *Change in confidence in next period = More or less the same*.

This states that "if the change in confidence in this period is more or less the same, then the change in confidence in next period will be more or less the same." The confidence of the association rule in $t_j$ can then be predicted given the confidence of this rule in $t_{j-1}$ and that in $t_{j-2}$. □

## 3.2 The Solution

Given a collection of data sets or data subsets, we propose to use a meta-mining approach to the discovery of regularities, differences, and changes in these rules. It comprises a collection of techniques for: 1) data transformation, 2) fuzzy partitioning, 3) attribute clustering, and 4) rule mining. They enable the mining of rules from data sets and meta-rules from rule sets. Fig. 1 shows the proposed meta-mining approach and how these techniques fit in it.

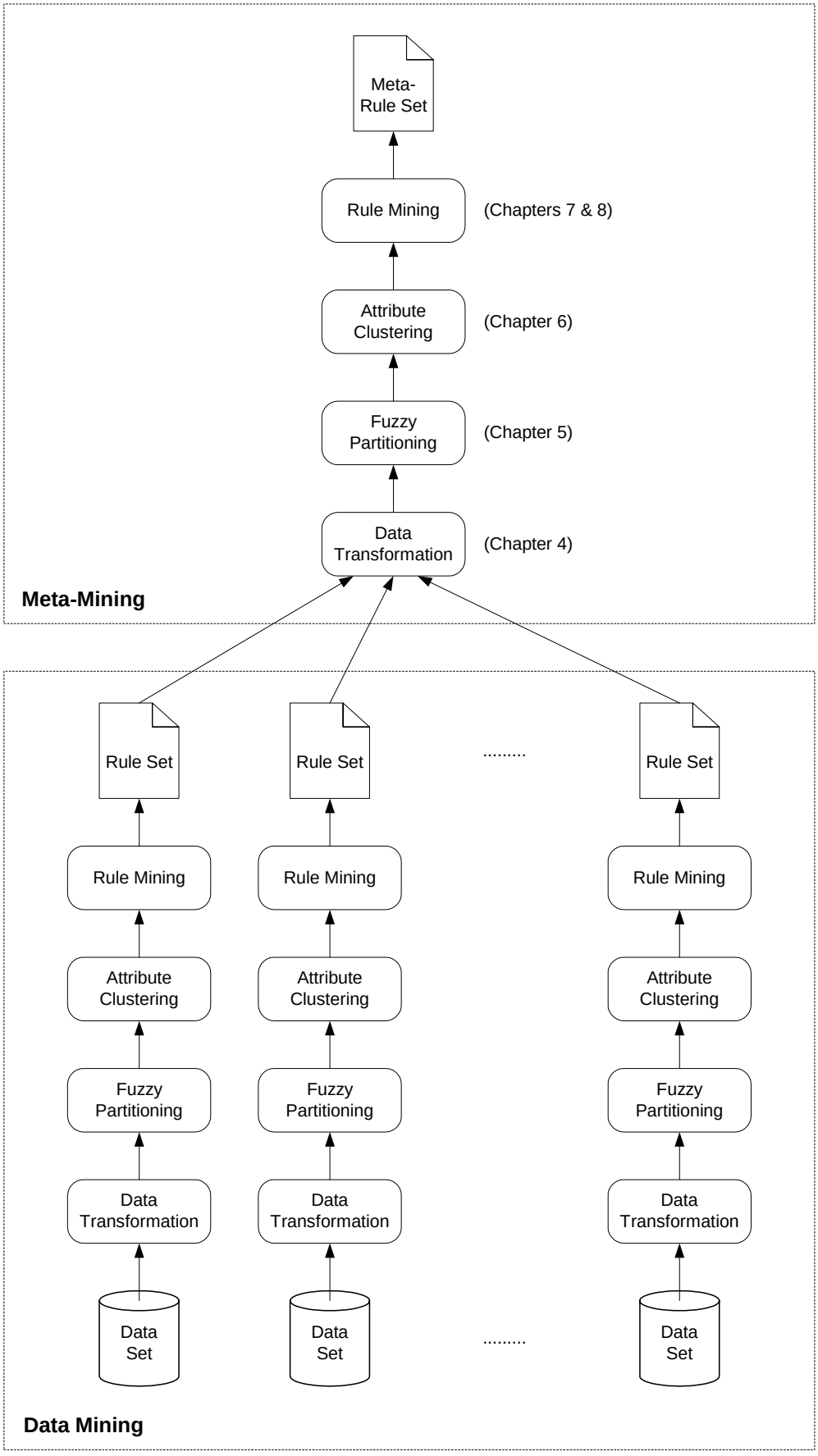



Fig. 1. The proposed meta-mining approach.

### 3.2.1 Data Transformation

Some existing algorithms can be used to discover association rules in transactional data, whereas other algorithms can be used to mine association rules in relational data. None of them has been explicitly developed to handle both transactional and relational data. We introduce a novel approach to handling both transactional and relational data at the same time. This type of data allows data mining algorithms to discover multi-dimensional association rules. To effectively uncover the hidden association relationships, the proposed approach combines the relational and transactional data by performing data transformations. It has been applied to a data mining task involving a large database that was provided by an international bank with offices in Hong Kong. The database contains the demographic data of over 320,000 customers and their banking transactions collected over a six-month period. By mining the database, the bank would like to be able to discover interesting patterns that would reveal different characteristics about different customers so that it could better serve and retain them. With our approach, fuzzy association rules obtained by our mining algorithms proposed in Chapter 7 are judged by the experts from the bank to be very useful. In particular, they discovered that they had identified some interesting characteristics about the customers who had once used the bank's loan services but then decided later to cease using them. The bank translated what they discovered into actionable items by offering some incentives to retain their existing customers.

To enable effective mining of fuzzy rules in the data mining phase, the proposed meta-mining approach applies the data transformation method to each data set. This method makes use of *transformation functions* to transform the original data into a set of *transformed data*. Instead of mining the original data, we mine rules from the transformed data. This enables not only the discovery of rules involving attributes not originally contained in the data, but also data mining in the combination of relational and transactional data.

In the meta-mining phase, the data transformation method can also be applied to the rule sets if the introduction of new attributes is desirable. This method transforms the attributes involved in the conditions of the rules in the rule sets using transformation functions to produce a set of transformed data. For clarity, we here refer to the transformed data as the *transformed meta-data*.

### 3.2.2 Fuzzy Partitioning

To deal with continuous or mixed continuous valued and discrete valued data, the domains of continuous attributes are typically discretized into a finite number of intervals. However,

if too many data points lie on the boundaries of the intervals due to the ambiguity or fuzziness of the attribute values near the boundary regions, this could sensitively affect the usefulness of the discovered patterns, especially when they are used for classification. To better handle continuous data, we propose to use fuzzy sets to represent interval events in the domains of continuous attributes, allowing continuous data lying on the interval boundaries to partially belong to multiple intervals. The effect of noise to fuzzy set based techniques has been evaluated experimentally in [Gayme, Menon, and Ball 2003; Pacini and Kosko 1992; Postlethwaite 1991; Sun and Wang 2005; Xie *et al.* 1994]. These empirical studies demonstrated the resilience to noise of fuzzy sets. Realistically, the resilience to noise of fuzzy sets and their affinity with human knowledge representation make them very useful in many data mining applications.

Since the membership functions of fuzzy sets can profoundly affect the performance of the models or rules discovered, the determination of membership functions or fuzzy partitioning is crucial. In this chapter, we present a new method to determine the membership functions of fuzzy sets directly from data to maximize the class-attribute interdependence and thence improve the classification results. In other words, it forms a fuzzy partition of the input space automatically, using an information-theoretic measure to evaluate the interdependence between the class membership and an attribute as the objective function for fuzzy partitioning. To find the global optimum of the measure, it employs fractional programming (iterative dynamic programming). Fuzzy partitioning then enables fuzzy data mining techniques to build fuzzy models or discover fuzzy rules based on the generated fuzzy sets instead of relying on the user-specified ones. To evaluate the effectiveness of the proposed method, several real-world data sets are used in our experiments. The experimental results show that this method is very effective in classification when compared to other well-known discretization and fuzzy partitioning approaches.

After data transformation, the meta-mining approach applies the fuzzy partitioning technique to the transformed data in the data mining phase. It also applies the fuzzy partitioning technique to the transformed meta-data in the meta-mining phase to generate fuzzy sets automatically for the new attributes introduced by data transformation. Because those attributes have been fuzzy partitioned in the data mining phase, it is not necessary to fuzzy partition them again in this step.

### *3.2.3 Attribute Clustering*

We propose an attribute clustering method, which is able to group genes based on their

interdependence so as to mine meaningful patterns from the gene expression data. This method can be used for gene grouping, selection, and classification. The partitioning of a relational table into attribute subgroups allows a small number of attributes within or across the groups to be selected for analysis. The clustering of attributes reduces the search dimension of a data mining algorithm. The reduction of the search dimension is especially important to data mining in gene expression data because such data typically consist of a huge number of genes (attributes) and a small number of gene expression profiles (tuples). Most data mining algorithms are typically developed and optimized to scale to the number of tuples rather than to the number of attributes. The situation becomes even worse when the number of attributes overwhelms the number of tuples, in which case, the likelihood of reporting patterns that are actually irrelevant due to chance becomes rather high. Gene grouping and selection are thus important preprocessing steps when many data mining algorithms are applied to gene expression data. This work defines the problem of attribute clustering and introduces a methodology for solving it. Our proposed method groups interdependent attributes into clusters by optimizing a criterion function derived from an information measure that reflects the interdependence between attributes. By applying our algorithm to gene expression data, meaningful clusters of genes are discovered. The grouping of genes based on attribute interdependence within group helps to capture different aspects of gene association patterns in each group. Significant genes selected from each group then contain useful information for gene expression classification and identification. To evaluate the performance of the proposed approach, we applied it to two well-known gene expression datasets and compared our results with those obtained by other methods. Our experiments show that the proposed method is able to find the meaningful clusters of genes. By selecting a subset of genes which have high multiple-interdependence with others within clusters, significant classification information can be obtained. Thus a small pool of selected genes can be used to build classifiers with very high classification rate. From the pool, gene expressions of different categories can be identified.

After fuzzy partitioning, our meta-mining approach employs the attribute clustering method to group interdependent attributes into clusters. It also applies the attribute clustering method to group interdependent attributes in the transformed and fuzzy partitioned meta-data into clusters. Significant attributes selected from each group contain useful information for classification and identification.

### *3.2.4 Fuzzy Rule Mining*

Existing data mining algorithms (e.g., decision-tree based approaches and association rule mining algorithms) typically require the domains of continuous attributes to be discretized

into a finite number of intervals. These intervals may not be concise and meaningful enough for humans to easily obtain nontrivial knowledge from the discovered rules. Instead of using intervals, we propose two new algorithms, called FARM and EFARM, for mining fuzzy rules, which employ linguistic variables and linguistic terms to represent the revealed regularities. The linguistic representation is especially useful when the discovered rules are presented to human users for examination. The use of fuzzy set based techniques not only makes the proposed algorithms resilient against noise such as inaccuracies in physical measures of real-world entities and missing values in databases, but also enables the prediction of attribute values to be associated with degrees of membership. Our algorithms are therefore able to deal with those cases where an object can belong to more than one class. For example, a person can suffer from a cold and fever at the same time. To distinguish interesting association relationships from uninteresting ones, both of our proposed algorithms employ an objective interestingness measure, which reflects the difference in the observed and the expected degree to which an object is characterized by different linguistic terms. Being based on the objective measure, they do not require any user-supplied thresholds, which are usually difficult to determine. To evaluate the performance of our algorithms, we tested them using several real-life data sets for data mining. The experimental results show that they are very effective at the tasks. When compared to popular data mining algorithms, they are better able to uncover useful rules hidden in databases. Furthermore, we also applied our proposed algorithms to synthetic data sets for meta-mining. The results show that they can reveal the embedded regularities, differences, and changes effectively.

### 3.2.5 Parallelization of Fuzzy Rule Mining

We also extend the FARM and EFARM algorithms, which are developed to mine fuzzy rules in data sets and meta-rules in rule sets in the last chapter, to exploit the scalability of parallel computer systems. The parallel versions of FARM and EFARM are called Parallel-FARM and Parallel-EFARM, respectively. Given a very large data set, Parallel-FARM divides it into several horizontal partitions and assigns them to different sites in a distributed system. Each site scans its database partition to obtain the number of tuples characterized by different linguistic terms and then exchanges the local counts with all the other sites to find the global counts. Based on the global counts, the interestingness measures are computed and the sites are able to uncover interesting associations. By repeating this process of counting, exchange of counts, and calculation of interestingness measures, Parallel-FARM is able to discover all interesting associations in a data set. On the other hand, Parallel-EFARM employs a parallel genetic algorithm for mining rules. It encodes a complete set of rules in one single chromosome and each allele encodes one rule which is

represented by some non-binary symbolic values. It stores a single population of chromosomes in a master processor. In each generation of Parallel-EFARM, the master processor performs selection, crossover, and mutation. It then distributes all the chromosomes among the processors in a distributed system. Each processor evaluates the fitness of the chromosomes assigned to it and sends the fitness of these chromosomes back to the master processor. Parallel-EFARM then proceeds to the next generation. Both Parallel-FARM and Parallel-EFARM were implemented in an experimental test bed. Their scalability was tested using a popular benchmarking data set. The results show that Parallel-FARM and Parallel-EFARM have very good sizeup, speedup, and scaleup performance.

The meta-mining approach applies one of the serial and parallel mining algorithms to mine rules (meta-rules) on the selected attributes from the transformed and fuzzy partitioned data (meta-data). The discovered rules and meta-rules represent the revealed association relationships using linguistic variables and linguistic terms. The discovered meta-rules are also able to represent regularities, differences, and changes in the rule sets.

As a remark, in order to mine meaningful meta-rules, the underlying data sets are ought to be "comparable." For example, one data set contains the close price of a stock listed in the Hong Kong Stock Exchange; another data set contains the close price of a stock listed in the New York Stock Exchange; and so on. This perspective is consistent with [Pedryz 2002]. In [Pedryz 2002], how fuzzy sets discovered in different data sets affect each other have been taken into consideration and the *collaborative fuzzy clustering* technique has been proposed to adjust these fuzzy sets.

# Chapter 4
# Data Transformation

Given a database system consisting of transactional data (e.g., records of purchase, electronic fund transfer, phone calls, etc.) and relational data (e.g., customer information, inventory records, etc.), our goal is to discover a set of interesting rules that describes the relationship between the patterns underlying them. The mining of *association rules* (e.g., [Agrawal, Imielinski, and Swami 1993b; Agrawal and Shafer 1996; Agrawal and Srikant 1994; Cheung *et al.* 1996a; Han and Fu 1995; Houtsma and Swami 1995; Mannila, Toivonen, and Verkamo 1994; Park, Chen, and Yu 1995a, 1995b; Savasere, Omiecinski, and Navathe 1995; Srikant and Agrawal 1995, 1996]) is developed to reveal the patterns or associations hidden in the data. Some of existing algorithms (e.g., [Agrawal, Imielinski, and Swami 1993b; Agrawal and Shafer 1996; Agrawal and Srikant 1994; Cheung *et al.* 1996a; Han and Fu 1995; Houtsma and Swami 1995; Mannila, Toivonen, and Verkamo 1994; Park, Chen, and Yu 1995a, 1995b; Savasere, Omiecinski, and Navathe 1995; Srikant and Agrawal 1995]) can be used to discover association rules in transactional data with no reference to relational data. On the other hand, other algorithms (e.g., [Srikant and Agrawal 1996]) can be used to mine association rules in relational data without any reference to transactional data. None of them is explicitly developed for handling both transactional and relational data. It is not clear how they, which deal with either transactional or relational data, can be applied to discover association rules relating the union of both transactional and relational data.

However, many database systems contain not only transactional data but also relational data concerning information such as customer background and inventory records, etc. For example, the rule "70% of the phone calls made by lawyers are to Canton, China; 8% of all transaction records exhibit such characteristics" cannot be discovered by existing techniques. This rule relates the phone call patterns to the career of customers. To discover rules of such kind, both of transactional and relational data have to be taken into consideration. Since both transactional and relational data are usually collected in many database systems, it is important that this problem to be dealt with effectively.

To deal with the problem that is created by the fact that there is more than one database relation, the concept of a *universal relation* needs to be used. A universal relation is an imaginary relation that can be used to represent the data that is constructed by logically joining all of the separate tables of a relational database [Ullman 1988]. The use of a

universal relation, therefore, makes it possible for the existing data mining systems (e.g., [Matheus, Chan, and Piatetsky-Shapiro 1993]) to deal with both transactional and relational data. Unfortunately, the construction of universal relations will very likely lead to the introduction of redundant information, which will mislead the rule discovery process of many data mining algorithms.

Existing data mining algorithms (e.g., [Agrawal, Imielinski, and Swami 1993b; Agrawal and Shafer 1996; Agrawal and Srikant 1994; Cheung *et al.* 1996a; Han and Fu 1995; Houtsma and Swami 1995; Mannila, Toivonen, and Verkamo 1994; Park, Chen, and Yu 1995a, 1995b; Savasere, Omiecinski, and Navathe 1995; Srikant and Agrawal 1995, 1996]) can be made more powerful if they can overcome such a problem. They can also be further improved if they can discover rules that involve attributes that are not originally contained in a database. The ability to do so is essential to the mining of interesting patterns in many different application areas. For example, rules regarding consumers' buying habits at Christmas cannot be discovered if a new attribute of "holiday" has not been considered.

Taking into consideration the need to address these issues, our proposed approach is equipped with some transformation functions that can be used to deal with both transactional and relational data and the different types of attributes in the databases of a database system so as to construct new relations.

The rest of this chapter is organized as follows. In Section 4.1, we introduce a formalism to handle the union of relational and transactional data. In Section 4.2, we describe the bank-account database that was provided by the bank in a consultancy project of the Department of Computing, The Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong. Prof. Keith C. C. Chan was the principal investigator of the consultancy project. In Section 4.3, we discuss the fuzzy association rules that are discovered by our rule mining algorithms proposed in Chapter 7 in the bank-account database.

## 4.1 Transformation Functions

To discover interesting associations in a relational database, some variables of interest can be extracted directly from the database, whereas some of them are not contained in the original data and they are produced by the transformation functions. To handle the union of both relational and transactional data, we define a set of *transformation functions* to operate on multiple relations. The application of these transformation functions to the database results in a set of *transformed data*. To manage the data mining process effectively, the

transformed data is stored in a relation in the relational database. We refer to this relation as the *transformed relation*. We define the problem formalism in the following.

Let $A_{i1}, ..., A_{iK_i}$, for $i$ = 1, …, $I$, be the attributes of the real-world entities represented by the relational tables, $R_i$, $i$ = 1, …, $I$, respectively. Let the domain of $A_{ik}$, $k$ = 1, …, $K_i$, be represented by $dom(A_{ik}) = \{a_{ik}^{(1)}, ..., a_{ik}^{(m_{ik})}\}$, $i$ = 1, 2, …, $I$, $k$ = 1, 2, …, $K_i$. In other words, $R_i \subseteq dom(A_{i1}) \times ... \times dom(A_{iK_i})$. For any $R_i$, we use $\mathcal{A}_{R_i}$ to denote the set of attributes of $R_i$, that is, $\mathcal{A}_{R_i} = \{A_{i1}, ..., A_{iK_i}\}$. The primary key of $R_i$, which is composed of one or more attributes and is associated with each tuple in a relation, is represented by $\mathcal{K}_i \subseteq \{A_{i1}, ..., A_{iK_i}\}$.

For a database system, a set of transaction records can be denoted by $T_j$, $j$ = 1, …, $J$, where each $T_j$ is characterized by a set of attributes, which are denoted by $A_{j1}, ..., A_{jL_j}$, and has a unique transaction identifier $TID_j$. In another words, $T_j \subseteq TID_j \times dom(A_{j1}) \times ... \times dom(A_{jL_j})$.

The definition of the transaction records, which is used here, follows the idea presented in [Srikant and Agrawal 1996]. It is a generalization of the definition of the transactions used in many of the existing algorithms for mining association rules (e.g., [Agrawal, Imielinski, and Swami 1993b; Agrawal and Shafer 1996; Agrawal and Srikant 1994; Cheung *et al.* 1996a; Han and Fu 1995; Houtsma and Swami 1995; Mannila, Toivonen, and Verkamo 1994; Park, Chen, and Yu 1995a, 1995b; Savasere, Omiecinski, and Navathe 1995; Srikant and Agrawal 1995]). In these algorithms, a transaction, $t$, is typically defined as <*TID*, $\mathcal{I}$'>, where *TID* is the transaction identifier of $t$, $\mathcal{I}' \subseteq \mathcal{I}$, and $\mathcal{I} = \{item_1, …, item_n\}$ is a set of items. To store transactions of this kind in a relational database, one can define a relation, $T$(*TID*, $A_1$, …, $A_n$), where *TID* is a transaction identifier. For any $t \in T$, $t[A_k] = 1$ if $t$ contains $item_k$; otherwise, $t[A_k] = 0$, for $k$ = 1, …, $n$. This is a special case of the definition of the transaction records used in this chapter. In addition to handling items, our definition can also handle categorical (discrete-valued) and quantitative (continuous-valued) attributes. This allows richer semantics to be captured in the transaction records as compared to the definition that is only concerned with items (e.g., [Agrawal, Imielinski, and Swami 1993b; Agrawal and Shafer 1996; Agrawal and Srikant 1994; Cheung *et al.* 1996a; Han and Fu 1995; Houtsma and Swami 1995; Mannila, Toivonen, and Verkamo 1994; Park, Chen, and Yu 1995a, 1995b; Savasere, Omiecinski, and Navathe 1995; Srikant and Agrawal 1995]).

In a database system, there are some one-to-many relationships between the records in

$R_i$, $i$ = 1, …, $I$, and those in $T_j$, $j$ = 1, 2, …, $J$. For example, the bank-account database contains a set of relational tables (i.e., CUSTOMER and ACCOUNT) that contain background information about each customer and a transactional table (i.e., TRANSACTION) that contains the details of each transaction made by a customer. The relational data are related to the transactional data by some one-to-many relationships in such a way that we can find $\mathcal{K}_i$, which is the primary key of $R_i$, in $\{A_{j1}, ..., A_{jL_j}\}$, which can be used as a foreign key to provide a reference to the corresponding tuple in $R_i$, $i$ = 1, …, $I$.

Given $R_i$ and $T_j$, to deal with both relational and transactional data and to consider additional attributes that are not originally in the database, we propose the concept of using transformation functions that are defined on the original attributes in $R_i$ and $T_j$. Let $f_1$, …, $f_p$ be a set of transformation functions, where:

$$f_p : A_{p1} \times ... \times A_{pr_p} \rightarrow A'_p, \; p = 1, \ldots, P,$$

where $r_p \geq 1$,

and $$A_{pu} \in \left( \bigcup_{i=1}^{I} \mathcal{A}_{R_i} \right) \bigcup \left( \bigcup_{j=1}^{J} \mathcal{A}_{T_j} \right), \; u = 1, \ldots, r_p.$$

We can construct a new relation $R'$ that contains both the original attributes in $R_i$ and $T_j$ and the transformed attributes that are obtained by applying appropriate transformation functions. Let $R'$ be composed of attributes, $A'_1, ..., A'_n$, that is, $R' \subseteq dom(A'_1) \times ... \times dom(A'_n)$, where $A'_u$, $u$ = 1, …, $n$, can be any attribute in $R_i$, $i$ = 1, …, $I$, or $T_j$, $j$ = 1, …, $J$, or any transformed attribute. In other words, $A'_u \in \left( \bigcup_{i=1}^{I} \mathcal{A}_{R_i} \right) \bigcup \left( \bigcup_{j=1}^{J} \mathcal{A}_{T_j} \right) \bigcup \left( \bigcup_{p=1}^{P} f_p(A_{p1}, ..., A_{pr_p}) \right)$. Instead of performing data mining on the original $R_i$ and $T_j$, we perform data mining on $R'$.

Given a database, different kinds of transformation functions can be performed. They include *logical*, *arithmetic*, *substring*, and *discretization* functions. Depending on the type of attribute, one or more of these functions can be applied to the attribute. We provide the definition of each type of transformation functions in the following sections.

### 4.1.1 The Logical Functions

The logical functions are composed of a combination of logical operators, such as NOT, AND, OR, etc. A logical function can take one or more attributes as arguments. Let $f_1$, …, $f_n$ be a set of functions so that:

$$f_j(a_1, a_2, ..., a_r) = (a_1 = c_{j1}) \oplus ... \otimes (a_r = c_{jr}),\ j = 1, \ldots, n,$$

where $a_i \in dom(A_i)$, $c_{ji} \in dom(A_i)$, $A_i \in \left(\bigcup_{i=1}^{I} \mathcal{A}_{R_i}\right) \bigcup \left(\bigcup_{j=1}^{J} \mathcal{A}_{T_j}\right)$, $i = 1, \ldots, r$,

and $\oplus, \ldots, \otimes \in \{\text{AND, OR, NOT, XOR, NAND, NOR}\}$.

A generic way of utilizing these functions is to construct a logical function, *f*, defined in terms of $f_1, \ldots, f_n$, as follows:

$$f(a_1, ..., a_r) = \begin{cases} 1 & \text{if } f_1(a_1, ..., a_r) = \text{true} \\ 2 & \text{else if } f_2(a_1, ..., a_r) = \text{true} \\ . & \\ . & \\ . & \\ n & \text{else if } f_n(a_1, ..., a_r) = \text{true} \end{cases},$$

where $a_i \in dom(A_i)$, $A_i \in \left(\bigcup_{i=1}^{I} \mathcal{A}_{R_i}\right) \bigcup \left(\bigcup_{j=1}^{J} \mathcal{A}_{T_j}\right)$, $i = 1, \ldots, r$.

In the case where none of $f_1, \ldots, f_n$ are evaluated as being true, the logical function, *f*, produces an unknown value as its output. Furthermore, if the value of any attribute, $A_i$, $i = 1, \ldots, r$, of a tuple is unknown, the logical function, *f*, also produces an unknown value as its output.

### 4.1.2 The Arithmetic Functions

The arithmetic functions can involve addition, subtraction, multiplication, and division. An arithmetic function takes a set of attributes as its argument and produces an attribute that has a type of real or integer. Let $f_1, \ldots, f_r$ be operations in relational algebra, each of which produces an integer or a real number. The arithmetic function *f* is defined as follows:

$$f(a_1, ..., a_r) = f_1(a_1) \oplus ... \otimes f_r(a_r),$$

where $a_i \in dom(A_i)$, $A_i \in \left(\bigcup_{i=1}^{I} \mathcal{A}_{R_i}\right) \bigcup \left(\bigcup_{j=1}^{J} \mathcal{A}_{T_j}\right)$, $i = 1, \ldots, r$,

and $\oplus, \ldots, \otimes \in \{+, -, \times, \div\}$.

In the case where the value of any attribute, $A_i$, $i = 1, \ldots, r$, of a tuple is unknown, the arithmetic function, *f*, produces an unknown value as its output.

### 4.1.3 The Substring Functions

The substring functions extract a specific portion of a given attribute. Let the given attribute, *A*, be a string of *s* characters. For any $a \in dom(A)$, we use $a[i]$ to denote the *i*-th character of *a*. The substring function, *f*, is defined as follows:

$$f(a) = a[l]a[l+1]...a[u],$$

$$\text{where} \quad a \in dom(A),\ A \in \left(\bigcup_{i=1}^{I} \mathcal{A}_{R_i}\right) \bigcup \left(\bigcup_{j=1}^{J} \mathcal{A}_{T_j}\right),$$

$$\text{and} \quad 1 \le l \le u \le s.$$

In the case where the value of an attribute, *A*, of a tuple is unknown, the substring function, *f*, produces an unknown value as its output.

### 4.1.4 The Discretization Functions

The discretization functions discretize the domain of any numeric attribute into a finite number of intervals. Let *f* be the discretization function that creates *r* intervals. We use $u_i$ to denote the upper limit of the *i*-th interval, for $i = 1, \ldots, r - 1$. Then, *f* is defined as follows:

$$f(a) = \begin{cases} 1 & \text{if } a \le u_1 \\ 2 & \text{if } u_1 < a \le u_2 \\ . & \\ . & \\ . & \\ r-1 & \text{if } u_{r-2} < a \le u_{r-1} \\ r & \text{if } a > u_{r-1} \end{cases},$$

$$\text{where} \quad a \in dom(A),\ A \in \left(\bigcup_{i=1}^{I} \mathcal{A}_{R_i}\right) \bigcup \left(\bigcup_{j=1}^{J} \mathcal{A}_{T_j}\right).$$

In the case where the value of an attribute, *A*, of a tuple is unknown, the discretization function, *f*, produces an unknown value as its output.

The boundaries of the intervals can be specified by users or determined automatically by using various algorithms (e.g., [Ching, Wong, and Chan 1995; Liu, Wong, and Wang 2004]). One of the commonly used algorithms involves discretizing the attribute into equal

intervals. Another popular algorithm involves discretizing the attribute into intervals in such a way that the number of tuples in each interval is the same. As a result, each tuple has an equal probability of lying in any interval.

## 4.2 A Case Study on the Bank-Account Database

The bank-account database was provided by a bank in Hong Kong. The bank does not want to be identified in our work because customer attrition rates are confidential. The bank-account database is stored in an Oracle database, which is one of the most popular *relational database management systems* [Date 2000]. It is composed of 3 relations, namely, CUSTOMER, ACCOUNT, and TRANSACTION. Of these relations, CUSTOMER and ACCOUNT contain relational data, whereas TRANSACTION contains transactional data. Specifically, the bank maintains a tuple in CUSTOMER for each customer (e.g., sex, age, marital status, etc.), a tuple in ACCOUNT for each account owned by a customer (e.g., account type, loan amount limit, etc.), and a tuple in TRANSACTION for each transaction made by a customer on one of his/her accounts (e.g., cash deposit, cash withdrawal, etc.). A customer can have one or more accounts and an account can have one or more transactions. Accordingly, a tuple in CUSTOMER is associated with one or more tuples in ACCOUNT, and a tuple in ACCOUNT is associated with one or more tuples in TRANSACTION.

Fig. 2 shows the schema of the bank-account database. Since each relation in the bank-account database contains many attributes, we only show a subset of these attributes in Fig. 2.

CUSTOMER (CUST_ID, SEX, AGE, MARITAL_STATUS, …)
ACCOUNT (ACCT_ID, CUST_ID, OVERDRAFT_LIMIT, BALANCE, …)
TRANSACTION (TID, ACCT_ID, DATE, AMOUNT, …)

Fig. 2. Schema of the bank-account database.

It is important to note that a relation in a relational database may contain relational data or transactional data. The entity that a relation represents is what makes it either relational or transactional. In a relation that contains transactional data, each tuple (transaction record) represents a business transaction. Specifically, a transaction record represents a debit or credit transaction in the bank-account database. A transaction record, therefore, has to store the account involved in the transaction, the date of the transaction, the amount of the transaction, etc.

In the bank-account database, CUSTOMER contains data for 320,000 customers. Each customer had opened one or more bank accounts for the purpose of using loan services, such

as a mortgage loan, a tax payment loan, etc. In this data, 99.5% of all customers were from Hong Kong and the remaining 0.5% of customers were from other countries (e.g., Singapore, Taiwan, France, the United States, etc.). The total loan balance of all customers in the bank-account database was H.K. $11.8 billion in November 1999.

The bank-account database was extracted from the time interval of September 1999 through to November 1999. The task was to reveal the interesting association relationships in the data so as to better serve and retain customers. These relationships are represented in the form of fuzzy association rules. Table 1 gives a summary of the bank-account database.

Table 1. Summary of the bank-account database.

| Relation | No. of Attributes | No. of Tuples |
|---|---|---|
| CUSTOMER | 48 | 320,000 |
| ACCOUNT | 42 | 558,431 |
| TRANSACTION | 37 | 1,746,996 |

### 4.2.1 The Transformation Functions Defined

In this section, we describe how we can construct a transformed relation, R (T_ACCT_TYPE, T_AMOUNT, T_NATIONALITY, …), using the transformation functions. To obtain the transformed relation, we (including a domain expert from the bank) defined 102 transformation functions in total. From the 102 transformation functions, in this section, we present three of them as an illustration.

Consider the attribute ACCOUNT[ACCT_ID]. The first digit of this attribute denotes the type of account. Let us suppose that it is a personal account if this digit is 1 and that it is a corporate account if this digit is 2. There exists a transformation function, $f_1$, defined as:

$$f_1(s) = first\_digit\_of(s),$$

where *first_digit_of*(*s*) returns the first digit of string *s*. The *transformed attribute* T_ACCT_TYPE was produced by applying $f_1$(ACCOUNT[ACCT_ID]) to every tuple in ACCOUNT, which is an example of the substring functions that are defined in Section 4.1.3.

To compute the average amount in the customers' accounts, we make use of another transformation function, $f_2$, which is defined as follows:

$$f_2(id) = \frac{\sum_{t \in \sigma_{\text{CUST\_ID}=id}(\text{ACCOUNT})} t[\text{AMOUNT}]}{|\sigma_{\text{CUST\_ID}=id}(\text{ACCOUNT})|},$$

where $\sigma$ denotes the SELECT operation from *relational algebra* and $|S|$ denotes the cardinality of set $S$. The function, $f_2$, is an example of the arithmetic functions that are defined in Section 4.1.2. The transformed attribute, T_AMOUNT, was produced by applying the function $f_2$(CUSTOMER[CUST_ID]) to every tuple in CUSTOMER.

The nationality of the customers can be grouped into different geographical regions for the purpose of discovering more meaningful rules. Such a grouping is performed by a transformation function, $f_3$, which is defined as:

$$f_3(n) = \begin{cases} \text{Asian} & \text{if } n = \text{Chinese or Japanese or ... or Korean} \\ \text{European} & \text{else if } n = \text{UK or French or ... or German} \\ \text{North American} & \text{else if } n = \text{US or Canadian} \end{cases}.$$

This function, $f_3$, is an example of the logical functions that are defined in Section 4.1.1. The transformed attribute, T_NATIONALITY, was produced by applying the function $f_3$(CUSTOMER[NATIONALITY]) to every tuple in CUSTOMER.

By applying the transformation functions to the bank-account database, we obtained the required transformed relation. There are 102 attributes in the transformed relation. Among the 102 transformed attributes, 6 are categorical (discrete-valued) and 96 are quantitative (continuous-valued). Instead of performing data mining on the original data, we discovered interesting associations from the transformed data.

### 4.2.2 Fuzzy Association Rules Discovered

Instead of applying our fuzzy association rule mining algorithms proposed in Chapter 7 to the three original relations in the bank-account database, we performed data mining on the transformed relation. The results obtained by FARM and EFARM are more or less the same. We report only the results obtained by FARM in this section for clarity.

In consultation with the banking officials, we defined appropriate linguistic terms for some attributes in the transformed relation, whereas we applied our fuzzy partitioning technique proposed in Chapter 5 to generate linguistic terms for other attributes automatically. As an example, two linguistic terms, *Small* and *Large*, were defined for the attribute called *Loan Balance*. The definitions of these linguistic terms are given in Fig. 3.

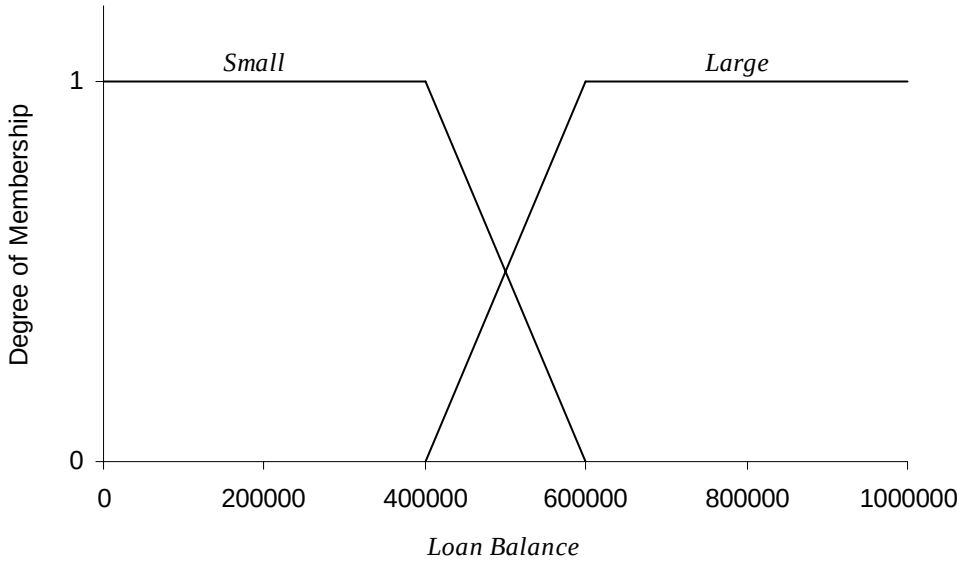



Fig. 3. The definitions of the linguistic terms for the attribute called *Loan Balance*.

As another illustration, let us consider the attribute called *Customer Age*. Four linguistic terms *Young*, *Youth*, *Middle Aged*, and *Elderly* were defined for *Customer Age* (Fig. 4).

Using the linguistic terms that were defined by the domain expert, we applied our rule mining algorithm to the transformed relation. From the discovered fuzzy association rules, we selected 200 rules randomly and presented them to the banking officials whom we consulted on the definition of the linguistic terms. The rules were evaluated according to how useful and how unexpected they were, as judged by the domain expert. The domain expert classified the rules into three categories: *very useful*, *useful*, and *less useful*. The result of the classification of these rules is summarized in Table 2.

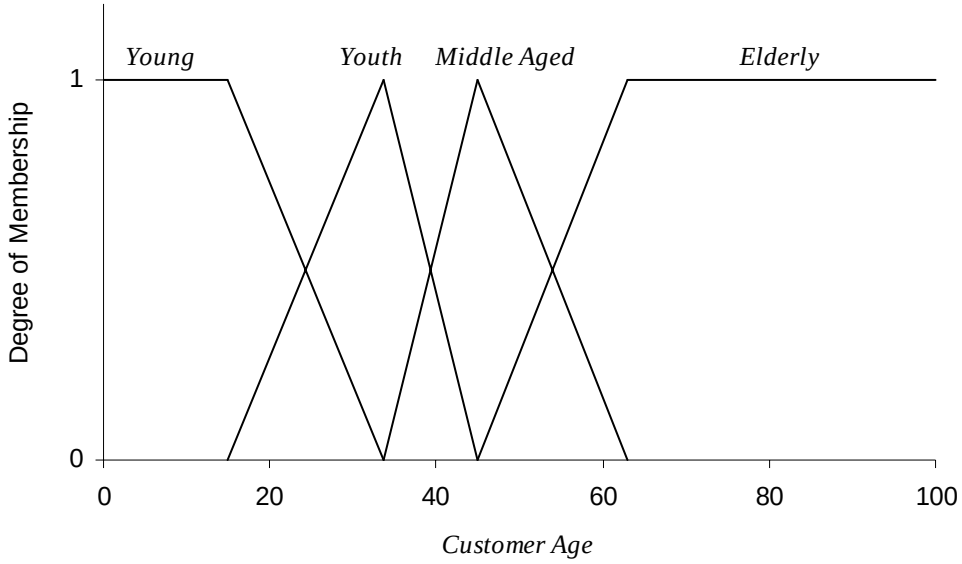



Fig. 4. The definitions of linguistic terms for the attribute called *Customer Age*.

Table 2. Classification of the fuzzy association rules discovered in the bank-account database.

| | **No. of Rules** | **Percentages** |
|---|---|---|
| **Very useful** | 51 | 25.5% |
| **Useful** | 132 | 66.0% |
| **Less useful** | 17 | 8.5% |

Among the 200 rules, the domain expert found 91.5% of them to be either useful or very useful. We expect that the evaluation of the remaining rules will follow a similar distribution because the 200 evaluated rules were selected randomly. This evaluation is quite high for an automated data mining tool. The reasons for this are likely to be that our interestingness measure can effectively reveal the interesting associations that are hidden in the data and that the fuzzy association rules, which employ linguistic terms to represent the underlying relationships, are more natural for human users to understand.

In the rest of this section, we show some of the discovered fuzzy association rules, which were identified as very useful by the domain expert. The following rule, regarding the affect that the annual income of a customer and the number of accounts that he/she holds has on the length of the customer relationship, was found being very useful.

$$\textit{Annual Income} = \textit{Very Large} \wedge \textit{No. of Accounts} = \textit{Very Small}$$
$$\Rightarrow \textit{Relationship Length} = \textit{Very Short}\ [w = 0.71],$$

where *Relationship Length* is produced by an arithmetic function, $f_{\textit{Relationship Length}}$, which is defined as follows:

$$f_{\textit{Relationship Length}}(id) = \mathsf{SYSDATE} - \pi_{\mathsf{MEMBER\_SINCE}}(\sigma_{\mathsf{CUST\_ID}=id}(\mathsf{CUSTOMER})),$$

where $\pi$ is the PROJECT operation in relational algebra, and SYSDATE returns the current date in Oracle.

This rule states that a customer who has a very large annual income and who holds a very small number of accounts will have a very short relationship with the bank. The length of the relationship that the bank has with a customer is important because the bank has a greater opportunity to cross-sell its products and services to a customer if he/she stays with the bank for a longer time. The domain expert found this rule being useful because it identifies the characteristics of customers who are more likely to have a short-tem relationship with the bank. By providing incentives to these customers, the bank can lengthen the relationships with them and increase its cross-selling opportunities (and hence we hope also improve its profitability). It is important to note that this rule involves only the attributes in the relational data.

The following fuzzy association rule, regarding the factors affecting the transaction costs, was also found to be very useful.

$$\textit{Sales Cost (Direct)} = \textit{Large} \wedge \textit{Sales Cost (Branch)} = \textit{Very Large}$$
$$\Rightarrow \textit{ATM Transaction Cost} = \textit{Very Large}$$
$$\wedge \textit{Branch Transaction Cost} = \textit{Very Large}\ [w = 5.38].$$

This rule describes the costs of ATM transactions and branches as being very large if the cost of direct sales is large and the cost of branch sales is also very large. The rule identifies the factors that affect the costs of ATM transactions and branches. Based on this rule, the domain expert suggested that the bank could provide better control of the costs of direct and branch sales so that the costs of ATM transactions and branches could be reduced. It is also important to note that this rule involves only the attributes in the transactional data.

Let us consider the fuzzy association rules that involve attributes that are in both the

relational and transactional data.

$$\textit{Customer Sex} = \textit{Female} \Rightarrow \textit{Loan Balance} = \textit{Small}\ [w = 1.23]$$
$$\textit{Customer Sex} = \textit{Male} \Rightarrow \textit{Loan Balance} = \textit{Large}\ [w = 0.67],$$

where *Loan Balance* is produced by an arithmetic function, $f_{Loan\ Balance}$, which is defined as follows:

$$f_{Loan\ Balance}(id) = \frac{\sum_{t \in \sigma_{\text{CUST\_ID}=id}(\text{ACCOUNT})} t[\text{LOAN\_BALANCE}]}{|\sigma_{\text{CUST\_ID}=id}(\text{ACCOUNT})|}.$$

The former rule states that female customers are more likely to use small loans, whereas the latter rule describes male customers as being more likely to use large loans. It is important to note that these rules are concerned with how the demographics of a customer affect his/her transactions. Specifically, they describe the association relationships between a customer's gender, which is contained in the relational data, and his/her total loan balances, which are contained in the transactional data. These rules cannot be discovered unless both relational and transactional data are considered together.

In addition to these rules, let us also consider the following fuzzy association rule:

$$\textit{Customer Sex} = \textit{Female} \wedge \textit{Marital Status} = \textit{Widowed}$$
$$\Rightarrow \textit{Loan Balance} = \textit{Large}\ [w = 3.62].$$

This rule states that female customers who are widowed are more likely to use large loans. As discussed above, a female customer is expected to make use of only small loans. However, the fact that these women are widowed, means that they tend to use large loans. Similar to the rules discussed above, this rule associates the demographics (i.e., gender and marital status) of a customer with his/her transactions (i.e., loan balances). This rule can only be revealed if relational and transactional data are considered together.

#### 4.2.1.1 Customer Retention

On the basis of the fuzzy association rules concerning the loan balance, the domain expert revealed that customers who use small loans could easily settle the loans as compared to those with larger loans. Because of this, customers who use small loans are more likely to stop using the loan services and cease to be a customer. Based on the rules concerning a

small loan balance, the bank is able to identify the characteristics of customers that may cease being customers. The bank can retain more of its customers in the future by offering incentives to the customers that have the same characteristics. In this way, our approach can be used for customer retention or to help reduce the customer attrition rate.

Let us consider the fuzzy association rules concerning the affect of the gender of a customer on his/her loan balance. Specifically, they state that female customers are more likely to use small loans, whereas male customers tend to use large loans. Based on these rules, the domain expert also revealed that female customers usually have a significant amount of savings and it is probably because of this reason that they tend to use small loans. This characteristic means that female customers tend to find it easier to settle loans, and hence they are more likely to cease using the loan services as compared to male customers. The attrition of customers is therefore related to gender. This finding is very useful to the domain expert because customers who are likely to cease using the loan services could be identified using these rules. To reduce the attrition rate, the domain expert suggested that incentives, such as lower interest rates, could be offered to female customers.

Let us also consider the fuzzy association rule that states that female customers who are widowed are more likely to use large loans. From other rules, we have revealed that female customers are more likely to cease using the loan services. However, the fact that these women are widowed, means that they tend to continue using the loan services. The domain expert found this rule being especially useful because it identifies a new niche market for promoting the bank's loan services.

# Chapter 5
# Partitioning Continuous Attributes

Many of existing data mining algorithms (e.g., ID3 [Quinlan 1986], AQ15 [Michalski *et al.* 1986], ITRule [Smyth and Goodman 1992], CN2 [Clark and Niblett 1989], and CBA [Liu, Hsu, and Ma 1998]) can only be applied to discrete-valued data. In order to deal with continuous or mixed continuous and discrete valued data, the domain of each continuous attribute is typically discretized into a finite number of intervals [Ching, Wong, and Chan 1995; Chiu, Wong, and Cheung 1991; Dougherty, Kohavi, and Sahami 1995; Fayyad and Irani 1993; Kerber 1992; Kurgan and Cios 2001; Liu and Setiono 1997; Liu, Wong, and Wang 2004; Wong and Chiu 1987]. The discrete-valued and the discretized data can then be handled in a uniform fashion and rules or models can be mined from them. Some data mining algorithms use built-in discretization mechanisms instead of using a discretization algorithm to preprocess continuous data. For example, when a continuous attribute is encountered in the data mining process, C4.5 [Quinlan 1993], CART [Breiman *et al.* 1984], and the association rule mining algorithm proposed in [Srikant and Agrawal 1996] discretize its domain into two or more intervals so that their criterion functions are optimized. Although they do not require continuous attributes to be discretized in advance, they discretize the attributes while mining rules.

Discretization enables many data mining algorithms to handle real-world data sets that consist of not only discrete, but also continuous data. However, if too many data lie on the boundaries of the intervals due to the ambiguous or fuzzy nature of the attribute values near the boundary regions, this could drastically affect the discovered rules or models that could be misleading and meaningless. As a consequence, data mining algorithms could not discover accurate models or rules from the discretized data.

To better handle continuous data, the use of fuzzy sets for data mining has recently been proposed in the literature [Mitra, Pal, and Mitra 2002]. It allows continuous data that lie on the interval boundaries to partially belong to multiple intervals. The resilience to noises and the affinity with the human knowledge representation of fuzzy sets make them a key component in many data mining systems (e.g., [Au and Chan 1998, 1999, 2001, 2003; Chan and Au 1997b, 2001; Chan, Au, and Choi 2002; Delgado *et al.* 2003; Hirota and Pedrycz 1999; Hüllermeier 2001; Ishibuchi, Yamamoto, and Nakashima 2001; Janikow 1998; Kacprzyk and Zadrozny 2001; Lee and Kim 1997; Maimon, Kandel, and Last 1999; Yager 1991]). By and large, these systems require fuzzy sets to be predefined as input upon

which they perform data mining.

A fuzzy set is defined by a *membership function* which maps objects in a domain of concern to their membership values in the fuzzy set. It is associated with a *linguistic term*, which allows human users to easily express their knowledge on one hand and comprehend the expressed knowledge on the other (see, e.g., [Pedrycz and Gomide 1998; Yen and Langari 1999]). Since membership functions can profoundly affect the performance of fuzzy models, the determination of membership functions or *fuzzy partitioning* is an important problem in fuzzy data mining. A membership function can be either determined by human experts or generated directly from data. A weakness of having human experts to provide inputs is that it is usually difficult for them to express or formalize their knowledge and experience in most situations [Buchanan *et al.* 1983; Johnson-Laird 1989]. It is for this reason that significant efforts have been put into generating membership functions from data recently (e.g., [Arslan and Kaya 2001; Fajfer and Janikow 2000; Jang 1993; Janikow and Fajfer 1999; Karr 1991; Lee and Takagi 1993; Liao, Celmins, and Hammell II 2003]).

In this chapter, we propose a new method, called Information-Theoretic Fuzzy Partitioning (ITFP), to construct fuzzy partitions directly from data. It uses an information-theoretic measure, which evaluates the interdependence between the class membership and an attribute, as the objective function for fuzzy partitioning. ITFP employs fractional programming (iterative dynamic programming) to find the global optimum of the measure. An advantage of ITFP is that it can determine the number of fuzzy sets automatically.

To evaluate the performance of ITFP, we applied it to several data sets when fuzzy partitions are constructed. We then fed the fuzzified data sets into a well-known decision-tree based algorithm, C4.5 [Quinlan 1993]. For C4.5 to handle the fuzzified data, it is extended in such a way that it uses the fuzzy membership values of continuous data in the calculation of the *gain ratio*. We compared the classification accuracies of ITFP with those obtained by other discretization and fuzzy partitioning methods. The experimental results show that ITFP is more effective.

The rest of this chapter is organized as follows. In the next section, we present our approach to fuzzy partitioning. We then show how C4.5 is extended to deal with fuzzy data in Section 5.2. To evaluate the performance of ITFP, it is applied to several real-life data sets. In order to facilitate a comparison, we also applied other well-known discretization and fuzzy partitioning algorithms to these data sets. The experimental results are given in Section 5.3.

## 5.1 An Fuzzy Partitioning Algorithm

### 5.1.1 An Class-Attribute Interdependence Measure

The fuzzy membership functions most commonly used in practice are the *triangular* membership functions and the *trapezoid* membership functions [Yen and Langari 1999]. Since the former can be considered a special case of the latter, we consider only the trapezoid membership functions in this chapter.

Let $A_1$, …, $A_K$ be the attributes of the real-world entities represented by a relational table, $D$. If $A_k$, $k \in \{1, \ldots, K\}$, is discrete, let its domain be represented by $dom(A_k) = \{a_{k1}, ..., a_{kR_k}\}$. Otherwise, if $A_k$, $k \in \{1, \ldots, K\}$, is continuous, let its domain be represented by $dom(A_k) = [l_k, u_k]$, where $l_k, u_k \in \Re$. For classification, one of the discrete attributes is chosen as the class label and let us denote it as $A_c$, $c \in \{1, \ldots, K\}$.

For each continuous attribute, $A_k$, $k \in \{1, \ldots, K\} - \{c\}$, its domain is represented by fuzzy sets, $S_{k1}$, …, $S_{kJ_k}$, such that:

$$\mu_{S_{kj}}(x) = \begin{cases} 1 & x < b_{j3} \\ (b_{j4} - x)/(b_{j4} - b_{j3}) & b_{j3} \le x < b_{j4} \\ 0 & x \ge b_{j4} \end{cases} \quad \text{for } j = 1, \tag{5.1}$$

$$\mu_{S_{kj}}(x) = \begin{cases} 0 & x < b_{j1} \\ (x - b_{j1})/(b_{j2} - b_{j1}) & b_{j1} \le x < b_{j2} \\ 1 & b_{j2} \le x < b_{j3} \\ (b_{j4} - x)/(b_{j4} - b_{j3}) & b_{j3} \le x < b_{j4} \\ 0 & x \ge b_{j4} \end{cases} \quad \text{for } j = 2, \ldots, J_k - 1, \tag{5.2}$$

and

$$\mu_{S_{kj}}(x) = \begin{cases} 0 & x < b_{j1} \\ (x - b_{j1})/(b_{j2} - b_{j1}) & b_{j1} \le x < b_{j2} \\ 1 & x \ge b_{j2} \end{cases} \quad \text{for } j = J_k, \tag{5.3}$$

where $\mu_{S_{kj}}$ is the membership function of fuzzy set $S_{kj}$, $j = 1, \ldots, J_k$.

The *fuzzy partition* of $dom(A_k)$ is composed of $S_{k1}, \ldots, S_{kJ_k}$ that satisfies the following condition [Ruspini 1969]:

$$\sum_{j=1}^{J_k} \mu_{S_{kj}}(x) = 1, \ \forall\ x \in dom(A_k). \tag{5.4}$$

It has been shown that this condition is a desirable property for the stability of fuzzy logic controllers (see, e.g., [Pedrycz and Gomide 1998; Yen and Langari 1999]). In order to satisfy the condition, we set $b_{j1} = b_{(j-1)3}$, $b_{j2} = b_{(j-1)4}$, $b_{j3} = b_{(j+1)1}$, and $b_{j4} = b_{(j+1)2}$, for $j = 2, \ldots, J_k - 1$, when we generate the fuzzy sets.

Let $\varphi = (S_{k1}, ..., S_{kJ_k})$ denote the fuzzy partition and let $L_{k\varphi}$ denote the *linguistic variable*, which represents the partitioned attribute. For clarity, we denote the expression "$L_{k\varphi}$ is $S_{kj}$" as $L_{k\varphi} = S_{kj}$ in this chapter. We say that a record, $d$, is with $L_{k\varphi} = S_{kj}$ to a degree of $\mu_{S_{kj}}(d[A_k])$. The joint probability of a record in $D$, which belongs to class $a_{cr}$, $c \in \{1, \ldots, K\}$, $r \in \{1, \ldots, R_c\}$, with $L_{k\varphi} = S_{kj}$, $k \in \{1, \ldots, K\} - \{c\}$, $j \in \{1, \ldots, J_k\}$, is given by:

$$p_{S_{kj}a_{cr}} = \frac{\sum_{d \in \sigma_{A_c = a_{cr}}(D)} \mu_{S_{kj}}(d[A_k])}{\sum_{d \in D} \mu_{S_{kj}}(d[A_k])}, \tag{5.5}$$

where $\sigma$ denotes the SELECT operation from *relational algebra*.

The estimated marginal probability of $A_c = a_{cr}$ and that of $L_{k\varphi} = S_{kj}$ are calculated by:

$$p_{a_{cr}} = \sum_{j=1}^{J_k} p_{S_{kj}a_{cr}} \tag{5.6}$$

and

$$p_{S_{kj}} = \sum_{r=1}^{R_c} p_{S_{kj}a_{cr}}, \tag{5.7}$$

respectively.

**Definition 5.1** The *interdependence redundancy measure* between the class attribute, $A_c$, and fuzzy set $S_{kj}$ is defined as:

$$R(A_c; S_{kj}) = \frac{I(A_c; S_{kj})}{H(A_c, S_{kj})}, \tag{5.8}$$

where $I(A_c; S_{kj})$ is the *mutual information* between $A_c$ and $S_{kj}$, which is given by:

$$I(A_c; S_{kj}) = \sum_{r=1}^{R_c} p_{S_{kj}a_{cr}} \log \frac{p_{S_{kj}a_{cr}}}{p_{a_{cr}} p_{S_{kj}}} \tag{5.9}$$

and $H(A_c, S_{kj})$ is the *joint entropy* of $A_c$ and $S_{kj}$ and is calculated by:

$$H(A_c, S_{kj}) = -\sum_{r=1}^{R_c} p_{S_{kj}a_{cr}} \log p_{S_{kj}a_{cr}} \ . \tag{5.10}$$

□

In addition to the interdependence redundancy measure between the class attribute and a fuzzy set, we can also define the measure between the class attribute and a linguistic variable [Wang and Wong 1979; Wong and Liu 1975].

**Definition 5.2** The interdependence redundancy measure between the class attribute, $A_c$, and linguistic variable $L_{k\varphi}$ is defined as:

$$R(A_c; L_{k\varphi}) = \frac{I(A_c; L_{k\varphi})}{H(A_c, L_{k\varphi})}, \tag{5.11}$$

where $I(A_c; L_{k\varphi})$ is the mutual information between $A_c$ and $L_{k\varphi}$, which is given by:

$$I(A_c; L_{k\varphi}) = \sum_{j=1}^{J_k} I(A_c; S_{kj}) \tag{5.12}$$

and $H(A_c, L_{k\varphi})$ is the joint entropy of $A_c$ and $L_{k\varphi}$ and is calculated by:

$$H(A_c, L_{k\varphi}) = \sum_{j=1}^{J_k} H(A_c, S_{kj}) . \quad (5.13)$$

$I(A_c; L_{k\varphi})$ measures the average reduction in uncertainty about $A_c$ that results from learning the value of $L_{k\varphi}$ [MacKay 2003]. To maximize the use of attribute $A_k$ after partitioning for classification, we should maximize the dependence of $A_c$ on $L_{k\varphi}$ during the fuzzy partitioning process. $I(A_c; L_{k\varphi})$ initially appears to be a good candidate for such a partitioning criterion. However, a weakness of using $I(A_c; L_{k\varphi})$ as the partitioning criterion is that its value increases with the number of fuzzy sets. In fact, $I(A_c; L_{k\varphi})$ is at maximum before any partitioning and it decreases as the number of fuzzy sets is reduced. It is for this reason that we need to normalize $I(A_c; L_{k\varphi})$ by $H(A_c, L_{k\varphi})$, which yields the interdependence redundancy measure, $R(A_c; L_{k\varphi})$.

$R(A_c; L_{k\varphi})$ reflects the degree of deviation from interdependence between $A_c$ and $L_{k\varphi}$. If $R(A_c; L_{k\varphi}) = 1$, $A_c$ and $L_{k\varphi}$ are strictly dependent. If $R(A_c; L_{k\varphi}) = 0$, they are statistically independent. If $0 < R(A_c; L_{k\varphi}) < 1$, then $A_c$ and $L_{k\varphi}$ are partially dependent. The definition of the interdependence redundancy measure shows that it is independent of the composition of $A_c$ and $L_{k\varphi}$. This implies that the number of attribute values can be reduced without destroying the interdependence relationship between $A_c$ and $L_{k\varphi}$. As a result, partitioning can be considered as a process to remove the redundancy introduced by too many possible attribute values. At the same time, the fuzzy partitioning process should minimize the loss of correlation between the class attribute and any other attribute. The properties of the interdependence redundancy measure clearly render an ideal candidate as a class-dependent partitioning criterion [Ching, Wong, and Chan 1995; Liu, Wong, and Wang 2004], which is used in our partitioning method as the optimization criterion.

The partitioning problem can therefore be solved by finding the fuzzy partition of the domain of $A_k$ such that the interdependence redundancy measure after partitioning is maximized. Let $\psi$ represent the set of all possible finite fuzzy partitions. Given a class-attribute pair, we need to find $\varphi_{max} \in \psi$ such that:

$$\forall \varphi \in \psi, R(A_c; L_{k\varphi_{\max}}) \geq R(A_c; L_{k\varphi}) . \quad (5.14)$$

### 5.1.2 Fuzzy Partitioning of Continuous Data

To find $\varphi_{max}$, we propose to use *fractional programming* [Sniedovich 1992], which is a

branch of nonlinear optimization involving ratio functions. Specifically, given a set, $Z$, and real-valued functions on $Z$, $r$, $v$, and $w$, such that $r(z) = \frac{v(z)}{w(z)}$, where $w(z) > 0$ for all $z \in Z$, fractional programming can be used to find $c \in Z$ such that:

$$c = \max_{z \in Z}(r(z)). \tag{5.15}$$

Let $Z^*$ be the set of solutions to the problem of finding $c$. By assuming that $Z^*$ is not empty, it can be solved as a parametric problem of finding $a(\lambda)$ such that:

$$a(\lambda) = \max_{z \in Z}(v(z) - \lambda w(z)), \tag{5.16}$$

where $\lambda \in \Re$. Let $Z^*(\lambda)$ be the set of optimal solutions given $\lambda$ and let us assume that it has at least one solution.

It is proved in [Sniedovich 1992] that $z \in Z^*$ if, and only if, $r \in Z^*(r(z))$ and hence $a(\lambda) = 0$ if, and only if, $\lambda = c$. The Dinkelbach's algorithm shown in Fig. 5 can be used to solve $a(\lambda) = 0$.

$k = 1$;
select some $z \in Z$;
$z^{(k)} = z$;
$\lambda^{(k)} = r(z^{(k)})$;
loop
    find $a(\lambda^{(k)}) = \max_{z \in Z}(v(z) - \lambda^{(k)} w(z))$;
    select some $z \in Z^*(\lambda^{(k)})$;
    $z^{(k+1)} = z$;
    $\lambda^{(k+1)} = r(z^{(k+1)})$;
    $k = k + 1$;
until ($a(\lambda^{(k)}) = 0$)
$z' = z$;
$\lambda' = r(z')$;

Fig. 5. The Dinkelbach's algorithm.

The Dinkelbach's algorithm obtains the optimal solution, $z'$, and it is guaranteed to terminate in finite steps if $w(z) > 0$ for all $z \in Z$ and if $Z$ is finite [Sniedovich 1992].

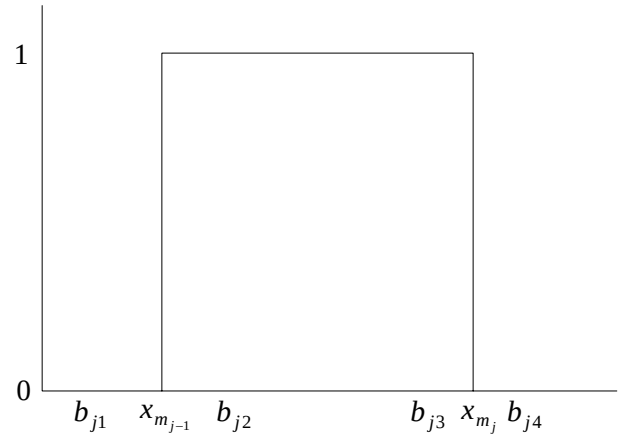
Based on the Dinkelbach's algorithm, we propose to use a new fuzzy partitioning algorithm, called Information-Theoretic Fuzzy Partitioning (ITFP), to partition the domains of continuous attributes. It has two important components: one is an iterative process that

uses the first component to drive towards the final global optimum solution and the other attempts to attain the maximum value of the objective function by applying dynamic programming for optimizing the class-attribute interdependence. This algorithm is presented in Fig. 6. For every continuous attribute, $A_k$, $k \in \{1, \ldots, K\} - \{c\}$, the algorithm obtains the optimal fuzzy partition $\varphi_{max}$.

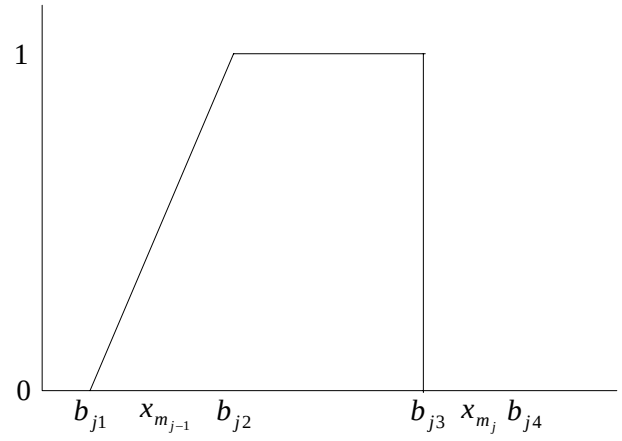
for each continuous attribute $A_k$, $k \in \{1, \ldots, K\} - \{c\}$
begin
  select some $\varphi \in \psi$;
  $u = R(A_c; L_{k\varphi}) = \dfrac{I(A_c; L_{k\varphi})}{H(A_c, L_{k\varphi})}$;
  loop
    find $\varphi'$ such that $I(A_c; L_{k\varphi'}) - uH(A_c, L_{k\varphi'})$ is maximized using
      a dynamic programming algorithm (see below);
    $u' = R(A_c; L_{k\varphi'}) = \dfrac{I(A_c; L_{k\varphi'})}{H(A_c, L_{k\varphi'})}$;
    $u = u'$;
  until ($u = u'$)
  $\varphi_k = \varphi'$;
end

Fig. 6. The ITFP algorithm.

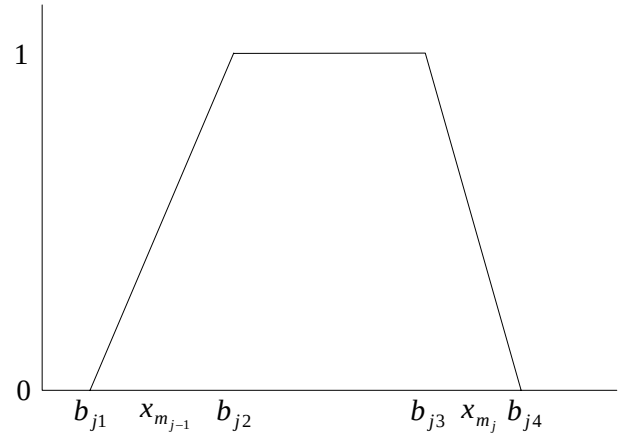
Now, we describe the dynamic programming algorithm to obtain the fuzzy partition $\varphi_{max}$, such that $I(A_c; L_{k\varphi_{max}}) - uH(A_c, L_{k\varphi_{max}})$ is maximized for any given $u \geq 0$. Let $\pi_{A_k}(D) = \{x_1, ..., x_{M_k}\}$ such that $x_1 \leq ... \leq x_{M_k}$, where $\pi$ denotes the PROJECT operation from relational algebra. Let the domain of $A_k$ be partitioned into fuzzy partition $\varphi = (S_{k1}, ..., S_{kJ_k})$ such that $b_{j1} \leq x_{m_{j-1}} \leq b_{j2}$ and $b_{j3} \leq x_{m_j} \leq b_{j4}$, $j = 1, \ldots, J_k$, $m_j \in \{1, \ldots, M_k\}$, and $m_1 < ... < m_{J_k}$. $b_{j1}$ and $b_{j2}$ are already set when $S_{k(j-1)}$ is determined to satisfy the condition of fuzzy partition (Equation (5.4)). To generate $S_{kj}$, we need to find $b_{j3}$ and $b_{j4}$ only. For the purpose of determining $b_{j3}$ and $b_{j4}$, let us suppose that $x_{m_{j-1}}$, $x_{m_j}$, and $x_{m_{j+1}}$ are given here. The way to determine $x_{m_j}$, $j = 1, \ldots, J_k$, will be presented later in this section.




(a) $X_j$.




(b) $Y$.




(c) $S_{kj}$.

Fig. 7. Fuzzy sets $X_j$, $Y$, and $S_{kj}$.

Given $x_{m_{j-1}}$ and $x_{m_j}$, we can form a special fuzzy set, $X_j$, to represent the interval, $[x_{m_{j-1}}, x_{m_j}]$, whose membership function is defined as:

$$\mu_{X_j}(x) = \begin{cases} 0 & x < x_{m_{j-1}} \\ 1 & x_{m_{j-1}} \le x \le x_{m_j} \\ 0 & x > x_{m_j} \end{cases}, \tag{5.17}$$

for $j$ = 1, …, $J_k$. $X_j$ is given in Fig. 7(a). In fact, $\tau = (X_1, ..., X_{J_k})$ is also a fuzzy partition of $dom(A_k)$.

We can then calculate the interdependence redundancy measure $R(A_c; X_j)$ between $A_c$ and $X_j$ by Equation (5.8). We are going to determine $S_{kj}$ so that $R(A_c; S_{kj}) = R(A_c; X_j)$. This ensures that the fuzzy partitioning process does not introduce any change in the interdependence relationship in terms of the interdependence redundancy measure. Following the idea presented in [Wu 1999], we have the user to supply a parameter, which controls how a fuzzy set spreads out into its adjacent fuzzy sets. In our approach, the user-specified parameter, $\varepsilon$, is interpreted as the percentage of the interdependence redundancy measure between $A_c$ and $S_{kj}$ to be spread out into $S_{k(j+1)}$. Initially, let us set $b_{j3} = \frac{1}{2}(x_{m_j} - b_{j2})$ and define a temporary fuzzy set, $Y$, whose membership function is defined as:

$$\mu_Y(x) = \begin{cases} 0 & x < b_{j1} \\ (x - b_{j1})/(b_{j2} - b_{j1}) & b_{j1} \le x < b_{j2} \\ 1 & b_{j2} \le x \le b_{j3} \\ 0 & x > b_{j3} \end{cases}. \tag{5.18}$$

We then employ the binary search to find the value of $b_{j3}$ in the range between $b_{j2}$ and $x_{m_j}$ such that $R(A_c; Y) = (1 - \varepsilon)\, R(A_c; X_j)$. Fig. 7(b) shows fuzzy set $Y$.

After the value of $b_{j3}$ is found, let us set $b_{j4} = \frac{1}{2}(x_{m_{j+1}} - b_{j3})$. Again, we use the binary search to find the value of $b_{j4}$ in the range between $b_{j3}$ and $x_{m_{j+1}}$ such that $R(A_c; S_{kj}) = R(A_c; X_j)$. $S_{kj}$ is then determined. $S_{kj}$ is shown graphically in Fig. 7(c). It is important to note that this method makes $p_{S_{kj}a_{cr}} = p_{X_j a_{cr}}$ and hence $R(A_c; L_{k\varphi}) = R(A_c; L_{k\tau})$. Consequently, $R(A_c; L_{k\varphi})$ is maximized when $R(A_c; L_{k\tau})$ is maximized.

Now, we present how to determine $x_{m_j}$, $j$ = 1, …, $J_k$. Let $F_\tau = I(A_c; X_{k\tau}) - uH(A_c, X_{k\tau})$ for fuzzy partition $\tau = (X_1, ..., X_{J_k})$. We have:

$$F_\tau = \sum_{r=1}^{R_c} \sum_{j=1}^{J_k} (p_{X_j a_{cr}} \log \frac{p_{X_j a_{cr}}}{p_{a_{cr}} p_{X_j}} + up_{X_j a_{cr}} \log p_{X_j a_{cr}}) . \quad (5.19)$$

$F_\tau$ can therefore be considered as the sum of $J_k$ terms, each of which corresponds to a fuzzy set, $X_j$, and is given by:

$$\sum_{r=1}^{R_c} (p_{X_j a_{cr}} \log \frac{p_{X_j a_{cr}}}{p_{a_{cr}} p_{X_j}} + up_{X_j a_{cr}} \log p_{X_j a_{cr}}) , \quad (5.20)$$

for $j \in \{1, \ldots, J_k\}$. The value of $F_\tau$ is fixed if the lower and upper bounds of the corresponding region are given. Let us consider the case that the $s$-th region boundary (i.e., $x_{m_s}$) is set to $x_i$, $i \in \{1, \ldots, M_k\}$, that is, $m_s = i$. The partitioning of the first $i$ attribute values has nothing to do with that of the last $M_k - i$ attribute values in terms of $F_\tau$. This observation allows us to use a dynamic programming algorithm to optimize $F_\tau$.

Let $g_{is}$ be the sum of the first $s$ terms of $F_\tau$ given that $x_i$ is the $s$-th region boundary, i.e.,

$$g_{is} = \sum_{j=1}^{s} \sum_{r=1}^{R_c} (p_{X_j a_{cr}} \log \frac{p_{X_j a_{cr}}}{p_{a_{cr}} p_{X_j}} + up_{X_j a_{cr}} \log p_{X_j a_{cr}}) . \quad (5.21)$$

Let $T_{is}$ be the set of all possible partition schemes with the first $i$ continuous values being partitioned into $s$ regions, we write $T_{is} = \{(x_{m_1}, ..., x_{m_s}, ..., x_{m_{J_k}}) \mid m_s = i\}$. Let $f_{is}$ denote the optimal value of $g_{is}$ among all possible partition schemes in $T_{is}$, i.e.,

$$f_{is} = \max_{(x_{m_1}, ..., x_{m_{J_k}}) \in T_{is}} (g_{is}) . \quad (5.22)$$

$f_{is}$ drives the partitioning of the first $i$ attribute values given that they are partitioned into $s$ regions. Since $A_k$ has $M_k$ values in $D$, it can be partitioned into at most $M_k$ regions. Therefore, we have:

$$\max_\tau (I(A_c; L_{k\tau}) - uH(A_c, L_{k\tau})) = \max_{1 \le J_k \le M_k} (f_{M_k J_k}) . \quad (5.23)$$

To apply the dynamic programming algorithm to calculate $f_{is}$, we need to figure out a recursive equation to represent $f_{is}$. Let us assume that there are $t$ attribute values in the $s$-th region (i.e., $m_s - m_{s-1} = t$). We obtain the following recursive equation:

$$f_{is} = \max_{(i_{k1},\ldots,i_{kJ_k})\in T_{is}} (f_{(i-t)(s-1)}) + \sum_{r=1}^{R_c} (p_{X_s a_{cr}} \log \frac{p_{X_s a_{cr}}}{p_{a_{cr}} p_{X_s}} + up_{X_s a_{cr}} \log p_{X_s a_{cr}}). \quad (5.24)$$

We also have the following initial conditions:

$$f_{i1} = \sum_{r=1}^{R_c} (p_{X_1 a_{cr}} \log \frac{p_{X_1 a_{cr}}}{p_{a_{cr}} p_{X_1}} + up_{X_1 a_{cr}} \log p_{X_1 a_{cr}}), \forall\ 1 \le i \le M_k, \quad (5.25)$$

where $m_0 = 1$ and $m_1 = i$, and

$$f_{1s} = \begin{cases} \sum_{r=1}^{R_c} (p_{X_1 a_{cr}} \log \frac{p_{X_1 a_{cr}}}{p_{a_{cr}} p_{X_1}} + up_{X_1 a_{cr}} \log p_{X_1 a_{cr}}) & s = 1 \\ -\infty & otherwise \end{cases}. \quad (5.26)$$

Based on the above equations, we can formulate the following dynamic programming algorithm:

1. Create a table with size $U \times U$, where $U$ is the number of unique values of attribute $A_k$. The element in the $i$-th row and the $s$-th column gives the value of $f_{is}$.
2. Initialize the elements in the first row and the first column of the table according to Equation (5.26) and Equation (5.25), respectively.
3. Calculate all the elements in this table according to the recursive equation of $f_{is}$ (Equation (5.24)).
4. Find the maximum value in the last row. Let us assume that the maximum value is in the $s^*$-th column. Then the optimal partition consists of $s^*$ regions (i.e., $s^*$ fuzzy sets). We then trace back to obtain the optimal region boundaries.

To incorporate semantics into the fuzzy sets to be discovered, one can define the interval boundaries himself/herself and make use of our proposed method to fuzzify the boundaries. In this way, the semantics can also be taken into consideration. Furthermore, since humans can typically handle only $7 \pm 2$ concepts at the same time, one may also like to supply such an upper bound of the number of fuzzy sets to be discovered.

Finally, let us consider the complexity of ITFP. The complexity of the dynamic programming component is $O(n^2)$, where $n$ is the number of values contained in the data. The binary search for fuzzifying a boundary can complete in $O(n \log n)$. Hence the

complexity of our proposed method is $O(n^2)$. This kind of task is able to be completed in a reasonable amount of time by any modern off-the-shelf single-processor machine.

## 5.2 An Example Application in Fuzzy Decision Tree Construction

In this section, we describe how C4.5, which is a well-known decision-tree based classification approach, can be extended to handle fuzzy data. In the tree-building phase, when C4.5 encounters a discrete attribute, it does what it does as usual without any change. When it encounters a continuous attribute, which has already been fuzzy partitioned, we extend it in the following.

Let $A_k$ be the attribute under consideration. $A_k$ is, in turn, represented by a linguistic variable $L_{k\varphi}$. Given a set of records, $D$, that belongs to some class, $a_{cr}$, the average amount of information, $info(D)$, needed to identify the class of a record in $D$ is given by:

$$info(D) = -\sum_{r=1}^{R_c} p_{a_{cr}} \log p_{a_{cr}} \ , \tag{5.27}$$

where $p_{a_{cr}}$ is the estimated marginal probability of $A_c = a_{cr}$ calculated by Equation (5.6). In fact, $info(D)$ is equivalent to the joint entropy of $A_c$ and $L_{k\varphi}$, $H(A_c, L_{k\varphi})$, given by Equation (5.13).

Let us further suppose that $D$ is divided into $D_1, \ldots, D_{J_k}$ in accordance with $L_{k\varphi}$ so that all the records in $D_j$ are with $L_{k\varphi} = S_{kj}$, $j = 1, \ldots, J_k$. The expected information requirement, $info_{L_{k\varphi}}(D)$, can then be computed as the weighted sum over the subsets:

$$info_{L_{k\varphi}}(D) = \sum_{j=1}^{J_k} p_{S_{kj}} info(D_j) , \tag{5.28}$$

where $p_{S_{kj}}$ is the estimated marginal probability of $L_{k\varphi} = S_{kj}$ calculated by Equation (5.7).

The information that is gained, $gain(L_{k\varphi})$, by dividing $D$ in accordance with $L_{k\varphi}$, is then given by:

$$gain(L_{k\varphi}) = info(D) - info_{L_{k\varphi}}(D) . \tag{5.29}$$

It is equivalent to the mutual information between $A_c$ and $L_{k\varphi}$, $I(A_c; L_{k\varphi})$, calculated by Equation (5.12).

Now, let us consider the potential information generated by dividing $D$ into $J_k$ subsets, *split info*($L_{k\varphi}$). It is calculated by:

$$split\ info(L_{k\varphi}) = -\sum_{j=1}^{J_k} p_{S_{kj}} \log p_{S_{kj}} \ . \tag{5.30}$$

The *gain ratio* is then defined as:

$$gain\ ratio(L_{k\varphi}) = \frac{gain(L_{k\varphi})}{split\ info(L_{k\varphi})} \ . \tag{5.31}$$

Intuitively, it expresses the proportion of the information generated by the division that appears useful for classification [Quinlan 1993].

The pruning mechanism used in C4.5 can also be extended to handle fuzzy data in a similar manner. We omit the discussion here for simplicity.

## 5.3 Evaluating Its Effectiveness

In order to evaluate the performance of ITFP, we applied it to several real-world data sets, which are the public data sets used in the StatLog project [Michie, Spiegelhalter, and Taylor 1994]. Of all the public data sets used in the StatLog project, the *dna* and the *letter* data sets contain discrete valued data only. We therefore did not use these data sets in our experiments. A summary of the datasets used in our experiments is given in Table 3. The interested readers are referred to [Michie, Spiegelhalter, and Taylor 1994] for the details.

Table 3. A summary of the data sets used in our experiments.

| Data Set | No. of Attributes | No. of Continuous Attributes | No. of Classes | No. of Records | Largest Class |
|---|---|---|---|---|---|
| *australian* | 14 | 6 | 2 | 690 | 55.5% |
| *diabetes* | 8 | 8 | 2 | 768 | 65.1% |
| *german* | 24 | 7 | 2 | 1,000 | 70.0% |
| *heart* | 13 | 7 | 2 | 270 | 55.6% |
| *satimage* | 36 | 36 | 6 | 6,435 | 23.8% |
| *segment* | 19 | 19 | 7 | 2,310 | 14.3% |
| *shuttle* | 9 | 9 | 7 | 58,000 | 78.6% |
| *vehicle* | 18 | 18 | 4 | 846 | 25.8% |

First we applied our ITFP to each of the eight data sets to produce a set of fuzzified data. Each fuzzified data set was divided into two subsets, one for training and the other for testing. We fed the training set to the modified version of C4.5, which is extended to deal with fuzzy data (see Section 5.4), to build a fuzzy decision tree. The resultant decision tree was then used to classify the test records. The classification accuracy of the decision tree was recorded. This step was repeated ten times and the average classification accuracy was calculated. The experimental results are given in Table 4 ("Cont." denotes running C4.5 on the original data (it has a built-in to discretize continuous values into discrete values), "Equal Freq." refers to the equal-frequency discretization, "Entropy" refers to the information entropy maximization discretization, "HCV" refers to the fuzzy interpretation of discretized intervals, "S" stands for supervised, and "U" denotes unsupervised).

In our experiments, we set the fuzziness parameter of FCM to 2 because the study in [Pal and Bezdek 1995] suggests that the best choice is probably in the interval between 1.5 and 2.5, whose mean and midpoint (i.e., 2) is usually the preferred choice for many uses of FCM. We set the dimension of SOM's output nodes to $100 \times 100$ so that SOM is able to determine the number of clusters automatically by not assigning any input vector to some of the output nodes.

Table 4. Performance of C4.5 averaged over 10 trials.

| **Dataset** | **Classification (Standard Deviation)** | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | **Discretization** | | | | **Fuzzy Partitioning** | | | | | | |
| | **Cont.** | **Equal Width** | **Equal Freq.** | **Entropy** | **Top Down** | **Bottom Up** | **FCM** | **SOM** | **SGA** | **HCV** | **ITFP** |
| | **S** | **U** | **U** | **S** | **U** | **U** | **U** | **U** | **U** | **S** | **S** |
| *australian* | 86.2% (4.2%) | 82.5% (4.5%) | 86.8% (2.9%) | 81.6% (4.1%) | 85.1% (4.4%) | 82.2% (5.5%) | 81.9% (2.8%) | 85.1% (4.4%) | 74.6% (12.7%) | 84.8% (2.7%) | **87.1% (2.8%)** |
| *diabetes* | 72.6% (7.7%) | 66.2% (4.5%) | 68.2% (3.7%) | 75.1% (5.1%) | 63.9% (4.7%) | 68.8% (4.0%) | 74.9% (5.1%) | 71.0% (5.9%) | 68.6% (4.7%) | 65.7% (4.3%) | **76.5% (4.0%)** |
| *german* | 71.9% (4.1%) | 67.7% (5.1%) | 68.4% (4.5%) | 70.9% (5.0%) | 67.1% (3.3%) | 67.1% (3.3%) | 67.7% (4.3%) | 68.1% (5.5%) | 60.8% (15.6%) | 67.8% (3.9%) | **75.2% (3.8%)** |
| *heart* | 77.8% (8.7%) | 75.2% (5.3%) | 78.9% (6.8%) | 78.5% (4.6%) | 73.7% (10.1%) | 71.1% (7.4%) | **79.6% (7.7%)** | 75.2% (9.2%) | 77.4% (4.8%) | 74.8% (5.7%) | **79.6% (4.7%)** |
| *satimage* | 85.8% (1.1%) | 82.2% (0.5%) | 83.1% (2.0%) | 81.8% (1.7%) | 21.4% (2.5%) | 21.8% (1.9%) | 84.7% (0.5%) | 82.4% (0.7%) | 46.2% (4.0%) | 22.7% (2.4%) | **86.8% (1.6%)** |
| *segment* | 96.5% (1.6%) | 91.7% (1.0%) | 94.2% (1.0%) | 95.3% (1.1%) | 11.2% (1.1%) | 85.1% (2.3%) | 94.5% (2.1%) | **96.8% (1.5%)** | 36.2% (1.9%) | 1.9% (0.8%) | 95.5% (0.8%) |
| *shuttle* | **99.9% (0.0%)** | 89.6% (0.4%) | 98.7% (0.1%) | **99.9% (0.0%)** | 78.7% (0.3%) | 78.4% (0.4%) | 91.6% (0.9%) | 99.7% (0.0%) | 11.9% (0.0%) | 77.3% (0.1%) | 98.2% (0.2%) |
| *vehicle* | 71.2% (5.6%) | 63.9% (3.5%) | 66.4% (6.0%) | 69.3% (6.1%) | 23.2% (2.1%) | 61.9% (4.9%) | **72.7% (5.6%)** | 66.0% (3.7%) | 38.4% (4.8%) | 22.7% (3.5%) | **72.7% (2.9%)** |
| **Average** | 82.7% | 77.4% | 80.6% | 81.5% | 53.0% | 67.0% | 81.0% | 80.5% | 51.8% | 52.2% | **84.0%** |

As shown in Table 4, ITFP obtains the best results on six out of the eight data sets. On the remaining data set *shuttle*, discretization algorithms achieve the highest accuracy, and on *segment*, the fuzzy partitioning technique SOM does the best. We also find that supervised methods yield the best results on seven of all the eight data sets. An unsupervised method produces the best result only on the remaining data set *segment*. In fact, the best three algorithms in terms of the average classification accuracy are all supervised ones. By this token, supervised methods perform better than unsupervised ones.

Our ITFP, which is a supervised approach to fuzzy partitioning, outperforms all the other discretization and fuzzy partitioning methods in average. It is the best on six of all the eight data sets. The average performance of the built-in discretization mechanism of C4.5 is second to ITFP only, but it yields the best accuracy on only one of the eight data sets.

The equal-frequency discretization, information entropy maximization, FCM, and SOM achieve more or less the same average classification accuracy. The performance of the equal-width discretization is a little inferior to these four methods.

Although the top-down and the bottom-up fuzzy partitioning are developed for a fuzzy decision-tree based classification approach, the experimental results show that they do not perform well. On the contrary, they are among the most disappointing ones in our experiments. Of the remaining algorithm, SGA's performance is similar to the top-down fuzzy partitioning.

To statistically test whether ITFP outperforms the built-in of C4.5, we use the *sign test* (see, e.g., [Walpole and Myers 1993]). The null hypothesis is that the classification accuracy obtained by ITFP and the built-in are the same, whereas the alternative hypothesis is that the classification accuracy obtained by ITFP is higher than that obtained by the built-in. ITFP. Of the eight datasets, ITFP performs better than the built-in on six datasets. The *p*-value is then equal to $\sum_{x=0}^{6} b(x;8,0.5) = 0.9648$. Since it is greater than 0.95, the null hypothesis can certainly be rejected at the 0.05 level of significance. We therefore conclude that ITFP outperforms the built-in of C4.5.

# Chapter 6
# Attribute Clustering

Clustering is an important topic in data mining research. Given a relational table, a conventional clustering algorithm groups tuples, each of which is characterized by a set of attributes, into clusters based on similarity [Jain, Murty, and Flynn 1999]. Intuitively, tuples in a cluster are more similar to each other than those belonging to different clusters. It has been shown that clustering is very useful in many data mining applications (e.g., [Fayyad *et al.* 1996; Piatetsky-Shapiro and Frawley 1991]).

When applied to gene expression data analysis, conventional clustering algorithms often encounter the problem related to the nature of gene expression data which is normally "wide" and "shallow." In another words, data sets usually contain a huge number of genes (attributes) and a small number of gene expression profiles (tuples). This characteristic of gene expression data often compromises the performance of conventional clustering algorithms. In this chapter, we present a methodology to group attributes that are interdependent or correlated with each other. We refer to such a process as *attribute clustering*. In this sense, attributes in a cluster are more correlated with each other, whereas attributes in different clusters are less correlated. Attribute clustering is able to reduce the search dimension of a data mining algorithm to effectuate the search of interesting relationships or for construction of models in a tightly correlated subset of attributes rather than in the entire attribute space. After attributes are clustered, one can select a smaller number for further analysis.

A gene expression data set from a microarray can be represented by an *expression table*, $T = \{w_{ij} \mid i = 1, \ldots, p, j = 1, \ldots, n\}$, where $w_{ij} \in \Re$ is the measured expression level of gene $g_i$ in sample $s_j$ [Domany 2003]. Each row in the expression table corresponds to one particular gene and each column to a sample. Such a data set is typically composed of a large number of genes but a small number of samples. For example, the *colon-cancer* data set [Alon *et al.* 1999] consists of 62 samples and 2,000 genes and the *leukemia* data set [Golub *et al.* 1999] contains 72 samples and 7,129 genes. The number of samples is likely to remain small for many areas of investigation, especially for human data, due to the difficulty of collecting and processing microarray samples [Piatetsky-Shapiro, Khabaza, and Ramaswamy 2003].

The distinctive characteristic of gene expression data allows clustering both genes and samples [Domany 2003; Jiang, Tang, and Zhang 2004]. With conventional gene clustering

methods, the genes are considered as the tuples and the samples as the attributes. Thus it allows genes with similar expression patterns (i.e., *co-expressed genes*) to be identified [Jiang, Tang, and Zhang 2004]. On the other hand, to cluster samples, the samples are considered as the tuples and the genes as the attributes. The clustering analysis of samples is to find new biological classes or to refine existing ones [Piatetsky-Shapiro, Khabaza, and Ramaswamy 2003]. By this token, conventional clustering algorithms are able to group both samples and genes from the data. In general, Euclidean distance and Pearson's correlation coefficient are widely used as the distance measure for clustering [Jiang, Tang, and Zhang 2004]. However, when Euclidean distance is applied to measure the similarity between genes, it is not effective to reflect functional similarity such as positive and negative correlation, interdependency as well as closeness in values. In fact, Euclidean distance accounts only for the last. In another words, the primary interest of the overall shapes of genes [Jiang, Tang, and Zhang 2004] is not well accounted for. Hence, Pearson's correlation coefficient is proposed by some researchers. An empirical study [Heyer, Kruglyak, and Yooseph 1999] has also shown that Pearson's correlation coefficient is not robust to outliers and it may assign high similarity score to a pair of dissimilar genes. Hence, a new method to cluster attributes in a relation is presented in this work which takes into consideration the abovementioned issues. It is known as *k*-modes Attribute Clustering Algorithm, referred to as ACA. ACA employs an information measure to evaluate the interdependence between attributes. It is used to direct the grouping of attributes into clusters. By applying ACA to gene expression data, clusters of genes based on their mutual correlation can be discovered. We can then select a small number of the top-ranked genes in each cluster for further analysis.

Furthermore, having so many genes relative to so few samples is likely to result in the discovery of irrelevant patterns (i.e., gene combinations which correlate with a target variable purely by chance) [Piatetsky-Shapiro, Khabaza, and Ramaswamy 2003]. A useful technique to deal with it is to select a small number of the most promising genes and use them solely to build models [Piatetsky-Shapiro, Khabaza, and Ramaswamy 2003]. To select genes, the *t*-value is widely used [Piatetsky-Shapiro, Khabaza, and Ramaswamy 2003]. It is important to note that the *t*-value can only be used when the samples are pre-classified. If no class information is provided, it cannot be used for gene selection. In this chapter, we introduce a *multiple interdependence measure* [Chiu and Wong 2004; Wong, Liu, and Wang 1976] for selection of genes with the highest correlation with the rest of attributes within a cluster.

To demonstrate ACA's usefulness for mining and analyzing gene expression data and

to evaluate its performance, two gene expression data sets, *colon-cancer* and *leukemia*, are used. We first applied ACA to each of them, selecting the most promising genes; then fed the selected genes into several well-known classification algorithms and compared their classification accuracies with those yielded by other gene selection methods. These classification algorithms, including a decision-tree based algorithm, neural networks, the nearest neighbor approach, and the naïve Bayes method, are used in this chapter because they have been employed in classification of gene expression data in the literature [Ben-Dor *et al.* 2000; Dudoit, Fridlyand, and Speed 2002; Friedman, Nachman, and Pe'er 2000; Keller *et al.* 2000; Khan *et al.* 2001; Lu and Han 2003; Zhang *et al.* 2001]. The experimental results demonstrate that ACA is more effective.

Each tuple in a relation $R$ is characterized by a set of attributes, $A_1$, …, $A_p$. If $A_i$, $i \in \{1, \ldots, p\}$, takes on discrete values, let its domain be represented by $dom(A_i) = \{a_{i1}, ..., a_{im_i}\}$. Otherwise, if $A_i$, $i \in \{1, \ldots, p\}$, is continuous, let its domain be represented by $dom(A_i) = [l_i, u_i]$, where $l_i, u_i \in \Re$. Let us suppose that $R$ consists of $n$ tuples, $t_1$, …, $t_n$. Each tuple, $t_u$, $u \in \{1, \ldots, n\}$, is represented by a vector of $p$ attribute values: $t_u = (x_{u1}, \ldots, x_{up})$, where $x_{ui} \in dom(A_i)$, $i = 1, \ldots, p$.

**Definition 6.1** *Attribute clustering* is a process which finds $c$ disjoint clusters, $C_1$, …, $C_c$, of correlated attributes by assigning each attribute in $\{A_1, \ldots, A_p\}$ to one of these clusters. Formally, we define attribute clustering as a process that $\forall A_i$, $i \in \{1, \ldots, p\}$, $A_i$ is assigned to a $C_r$, $r \in \{1, \ldots, c\}$, where $C_r \cap C_s = \varnothing$ for all $s \in \{1, \ldots, c\} - \{r\}$. □

To find meaningful clusters, attribute clustering is conducted so that attributes within a cluster should have high correlation with or high interdependence to each other, whereas attributes in different clusters are less correlated or more independent. Most of the conventional clustering methods use some distance metric to measure the dissimilarity or distance between two objects. In this chapter, we introduce the new interdependence information measure which we believe are more meaningful if interdependent patterns are the most significant characteristics of a cluster reflecting the inter-relationship among attributes.

## 6.1 An Attribute Interdependence Measure

For each continuous attribute in relation $R$, its domain is typically discretized into a finite number of intervals for data mining. In this chapter, we use our fuzzy partitioning technique ITFP introduced in the last chapter to partition the continuous data. It uses the normalized

mutual information measure that reflects interdependence between the class label and the attribute to be partitioned as the objective function, and fractional programming (iterative dynamic programming) to find a global optimal solution.

Let us suppose that the domain of $A_i$, $i \in \{1, \ldots, p\}$, is fuzzy partitioned by ITFP into $m_i$ fuzzy sets. After fuzzy partitioning, the domains of all the attributes in $R$ can be represented by $dom(A_i) = \{v_{i1}, ..., v_{im_i}\}$, $i = 1, \ldots, p$, where $v_{ik} = a_{ik}$, $k = 1, \ldots, m_i$, if $A_i$ is discrete and $v_{ik} = l_{ik}$, which is a linguistic term, if $A_i$ is a fuzzy partitioned continuous attribute.

Let $\sigma$ denote the SELECT operation from relational algebra and $|S|$ denote the cardinality of set $S$. The probability of a record in $R$ having $A_i = v_{ik}$, $i \in \{1, \ldots, p\}$, $k \in \{1, \ldots, m_i\}$, is then given by:

$$\Pr(A_i = v_{ik}) = \frac{|\sigma_{A_i = v_{ik}}(R)|}{|\sigma_{A_i \neq \text{NULL}}(R)|} \tag{6.1}$$

and the joint probability of a record in $R$ having $A_i = v_{ik}$ and $A_j = v_{jl}$, $i, j \in \{1, \ldots, p\}$, $i \neq j$, $k \in \{1, \ldots, m_i\}$, $l \in \{1, \ldots, m_j\}$, is calculated by:

$$\Pr(A_i = v_{ik} \wedge A_j = v_{jl}) = \frac{|\sigma_{A_i = v_{ik} \wedge A_j = v_{jl}}(R)|}{|\sigma_{A_i \neq \text{NULL} \wedge A_j \neq \text{NULL}}(R)|}. \tag{6.2}$$

**Definition 6.2** The *interdependence redundancy measure* [Wong and Liu 1975] between two attributes, $A_i$ and $A_j$, $i, j \in \{1, \ldots, p\}$, $i \neq j$, is defined as:

$$R(A_i : A_j) = \frac{I(A_i : A_j)}{H(A_i, A_j)}, \tag{6.3}$$

where $I(A_i : A_j)$ is the *mutual information* between $A_i$ and $A_j$, which is given by:

$$I(A_i : A_j) = \sum_{k=1}^{m_i} \sum_{l=1}^{m_j} \Pr(A_i = v_{ik} \wedge A_j = v_{jl}) \log \frac{\Pr(A_i = v_{ik} \wedge A_j = v_{jl})}{\Pr(A_i = v_{ik}) \Pr(A_j = v_{jl})} \tag{6.4}$$

and $H(A_i, A_j)$ is the *joint entropy* of $A_i$ and $A_j$ and is calculated by:

$$H(A_i, A_j) = -\sum_{k=1}^{m_i}\sum_{l=1}^{m_j} \Pr(A_i = v_{ik} \wedge A_j = v_{jl}) \log \Pr(A_i = v_{ik} \wedge A_j = v_{jl}) . \quad (6.5)$$

$I(A_i : A_j)$ measures the average reduction in uncertainty about $A_i$ that results from learning the value of $A_j$ [MacKay 2003]. If $I(A_i : A_j) > I(A_i : A_h)$, $h \in \{1, \ldots, p\}$, $h \neq i \neq j$, the dependence of $A_i$ on $A_j$ is greater than the dependence of $A_i$ on $A_h$. $I(A_i : A_j)$ initially appears to be a good candidate for measuring the interdependence between $A_i$ and $A_j$. However, a weakness of using $I(A_i : A_j)$ is that its value increases with the number of possible attribute values (i.e., $m_i$ and $m_j$). It is for this reason that we need to normalize $I(A_i : A_j)$ by $H(A_i, A_j)$, which yields the interdependence redundancy measure, $R(A_i : A_j)$.

More accurately stated, $R(A_i : A_j)$ reflects the degree of deviation from independence between $A_i$ and $A_j$. If $R(A_i : A_j) = 1$, $A_i$ and $A_j$ are strictly dependent. If $R(A_i : A_j) = 0$, they are statistically independent. If $0 < R(A_i : A_j) < 1$, then $A_i$ and $A_j$ are partially dependent. The definition of the interdependence redundancy measure shows that it is independent of the composition of $A_i$ and $A_j$. This implies that the number of attribute values does not affect the interdependence relationship between $A_i$ and $A_j$. The properties of the interdependence redundancy measure clearly render an ideal candidate to measure the dependence between different attributes.

If two attributes are dependent on each other, they are more correlated with each other when compared to two independent attributes. The interdependence redundancy measure is therefore able to evaluate the interdependence or correlation of attributes. If $R(A_i : A_j) > R(A_i : A_h)$, $h \in \{1, \ldots, p\}$, $h \neq i \neq j$, the dependence between $A_i$ and $A_j$ is greater than that between $A_i$ and $A_h$. In attribute clustering, we use $R(A_i : A_j)$ to measure the interdependence between attributes $A_i$ and $A_j$.

In order to investigate the interdependency of an attribute with all the other within a group, we introduce the concept of *significant multiple interdependency*.

**Definition 6.3** The *multiple interdependence redundancy measure* [Chiu and Wong 2004; Wong, Liu, and Wang 1976] of an attribute $A_i$ within an attribute group or cluster, $C = \{A_j \mid j = 1, \ldots, p\}$, is defined as:

$$MR(A_i) = \sum_{j=1}^{p} R(A_i : A_j) , \quad (6.6)$$

where $R(A_i : A_j)$ is the interdependence redundancy measure between $A_i$ and $A_j$. □

Based on the concept of $MR(A_i)$, we introduce the concept of the "mode" which is an attribute with the highest multiple interdependence redundancy in an attribute group.

**Definition 6.4** The *mode* of an attribute group, $C = \{ A_j \mid j = 1, \ldots, p\}$, denoted by $\eta(C)$ is an attribute, say $A_i$, in that group such that:

$$MR(A_i) \geq MR(A_j) \text{ for all } j \in \{1, \ldots, p\}.$$

□

## 6.2 An Attribute Clustering Algorithm

To group attributes $A_1, \ldots, A_p$ into clusters, we build our information-theoretic attribute clustering algorithm by converting the popular *k*-means algorithm into what we call the *k*-modes algorithm by replacing: 1) the concept of the term "mean," which represents the center of a cluster of entities, by the concept of mode which is the attribute with the highest multiple interdependence within an attribute group and 2) the distance measure used in *k*-means by the interdependence redundancy measure between attributes. We can then formulate the *k*-modes algorithm in the following.

1. ***Initialization.*** Let us assume that the number of clusters, *k*, where *k* is an integer greater than or equal to 2, is given. Of the *p* attributes, we randomly select *k* attributes, each of which represents a candidate for a mode $\eta_r$, $r \in \{1, \ldots, k\}$. Formally, we have $\eta_r = A_i$, $r \in \{1, \ldots, k\}$, $i \in \{1, \ldots, p\}$, to be the mode of $C_r$ and $\eta_r \neq \eta_s$ for all $s \in \{1, \ldots, k\} - \{r\}$.
2. ***Assignment of each attribute to one of the clusters.*** For each attribute, $A_i$, $i \in \{1, \ldots, p\}$, and each cluster mode, $\eta_r$, $r \in \{1, \ldots, k\}$, we calculate the interdependence redundancy measure between $A_i$ and $\eta_r$ , $R(A_i : \eta_r)$. We assign $A_i$ to $C_r$ if $R(A_i : \eta_r) \geq R(A_i : \eta_s)$ for all $s \in \{1, \ldots, k\} - \{r\}$.
3. ***Computation of mode for each attribute cluster.*** For each cluster, $C_r$, $r \in \{1, \ldots, k\}$, we set $\eta_r = A_i$ if $MR(A_i) \geq MR(A_j)$ for all $A_i, A_j \in C_r$, $i \neq j$.
4. ***Termination.*** Steps 2 and 3 are repeated until the $\eta_r$ for the clusters does not change. Alternatively, ACA also terminates when the pre-specified number of iterations is reached.

It is important to note that the number of clusters, $k$, is fed to ACA as an input parameter. To find the best choice for $k$, we use the sum of the multiple significant interdependence redundancy measure, $\sum_{r=1}^{k}\sum_{A_i \in C_r} R(A_i : \eta_r)$, to evaluate the overall performance of each clustering. With this measure, we can run ACA for all $k \in \{2, \ldots, p\}$ and select the value $k$ that maximizes the sum of the multiple significant interdependence redundancy measure over all the clusters as the number of clusters. That is,

$$k = \arg\max_{k \in \{2, \ldots, p\}} \sum_{r=1}^{k}\sum_{A_i \in C_r} R(A_i : \eta_r) . \tag{6.7}$$

To investigate the complexity of ACA algorithm, we consider a gene expression table, which is composed of $n$ samples such that each sample is characterized by $p$ gene expression levels. The $k$-modes algorithm requires $O(np)$ operations to assign each gene to a cluster (Step 2). It then performs $O(np^2)$ operations to compute the mode for each cluster (Step 3). Let $t$ be the number of iterations, the computational complexity of the $k$-modes algorithm is given by:

$$\begin{aligned} O(\text{ACA}) &= O(k(np + np^2)t) \\ &= O(knp^2t) \end{aligned} . \tag{6.8}$$

This kind of task is able to be completed in a reasonable amount of time by any modern off-the-shelf single-processor machine. Furthermore, the $k$-modes algorithm can easily be parallelized to run on clusters of processors because the calculation of the interdependence redundancy measure is an independent task.

# 6.3 Performance Evaluation

## *6.3.1 A Synthetic Data Set*

To evaluate the clusters of attributes formed by ACA, we first applied it to a synthetic data set. Each tuple in the synthetic data set is composed of 20 continuous attributes and is pre-classified into one of the 3 classes: $C_1$, $C_2$, and $C_3$. Let us denote the attributes as $A_1$, …, $A_{20}$. In the designed experiment, attribute values of $A_1$ and $A_2$ alone can determine the class membership of a tuple (Fig. 8). As shown in Fig. 8, data points lying on the rectangles, the circle, and the triangle belong to $C_1$, $C_2$, and $C_3$, respectively. Values of the other attributes (i.e., $A_3$, …, $A_{20}$) in the tuple are randomly generated in the following manner:

- $A_3$–$A_6$: uniformly distributed from 0 to 0.5 if the value of $A_1$ < 0.5; uniformly distributed from 0.5 to 1, otherwise.
- $A_7$–$A_{11}$: uniformly distributed from 0 to 0.5 if the value of $A_1$ ≥ 0.5; uniformly distributed from 0.5 to 1, otherwise.
- $A_{12}$–$A_{15}$: uniformly distributed from 0 to 0.5 if the value of $A_2$ < 0.5; uniformly distributed from 0.5 to 1, otherwise.
- $A_{16}$–$A_{20}$: uniformly distributed from 0 to 0.5 if the value of $A_2$ ≥ 0.5; uniformly distributed from 0.5 to 1, otherwise.

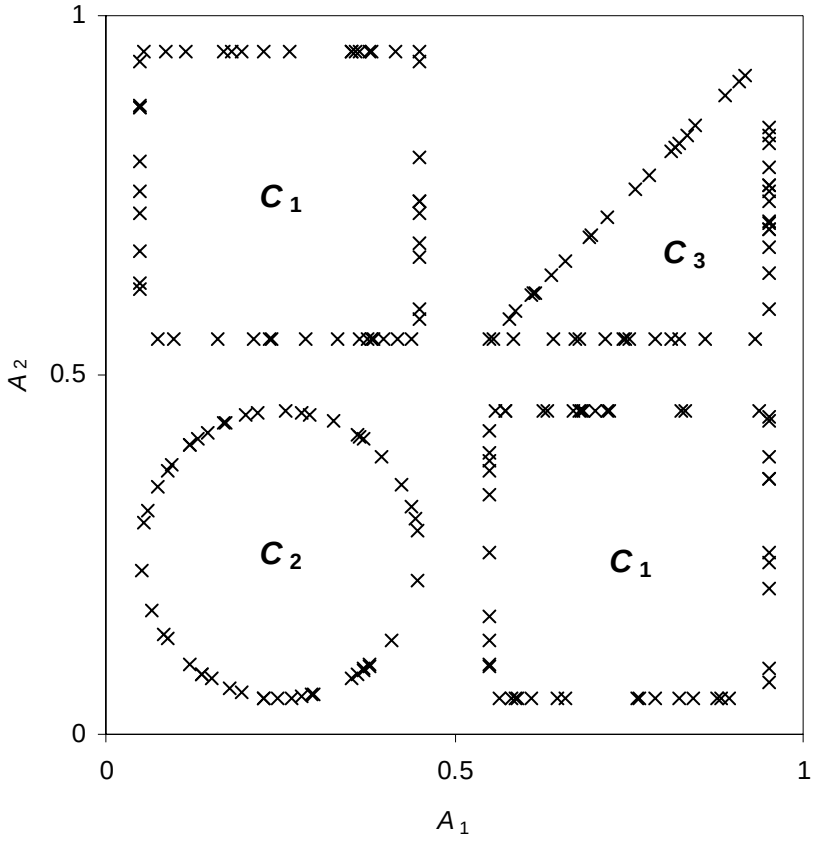



Fig. 8. Attribute values of $A_1$ and $A_2$ in the tuples in the synthetic data set.

It is obvious that $A_3$, …, $A_{11}$ are correlated with $A_1$, whereas $A_{12}$, …, $A_{20}$ are correlated with $A_2$. For an attribute clustering algorithm to be effective, it should be able to reveal such correlations. In our experiments, we generated 200 tuples in the synthetic data set and added noises to the data set by replacing the attribute values of $A_3$, …, $A_{20}$ in 25% of the tuples with a random real number between 0 and 1.

We first used our fuzzy partitioning technique ITFP proposed in the last chapter to fuzzy partition the domain of each attribute. As expected, it partitions the domain of each attribute into 2 fuzzy intervals: $\int_0^{x_1} \frac{1}{x} + \int_{x_1}^{x_2} \frac{(x - x_2)/(x_1 - x_2)}{x}$ and $\int_{x_1}^{x_2} \frac{(x - x_1)/(x_2 - x_1)}{x} + \int_{x_2}^{1} \frac{1}{x}$, where $x_1 \leq x_2$ and $x_1 \approx x_2 \approx 0.5$. We then applied ACA to the fuzzy partitioned data to find clusters of attributes. Fig. 9 shows the sum of the interdependence redundancy measure over all the clusters versus the number of clusters found in the synthetic data set. As shown in Fig. 9, it finds that the optimal number of clusters is 2. ACA identifies 2 clusters of attributes: $\{A_1, A_3, \ldots, A_{11}\}$ and $\{A_2, A_{12}, \ldots, A_{20}\}$. $A_1$ is the mode of the former cluster, whereas $A_2$ is the mode of the latter. It shows that ACA is able to reveal the correlations between the attributes hidden in the synthetic data set.

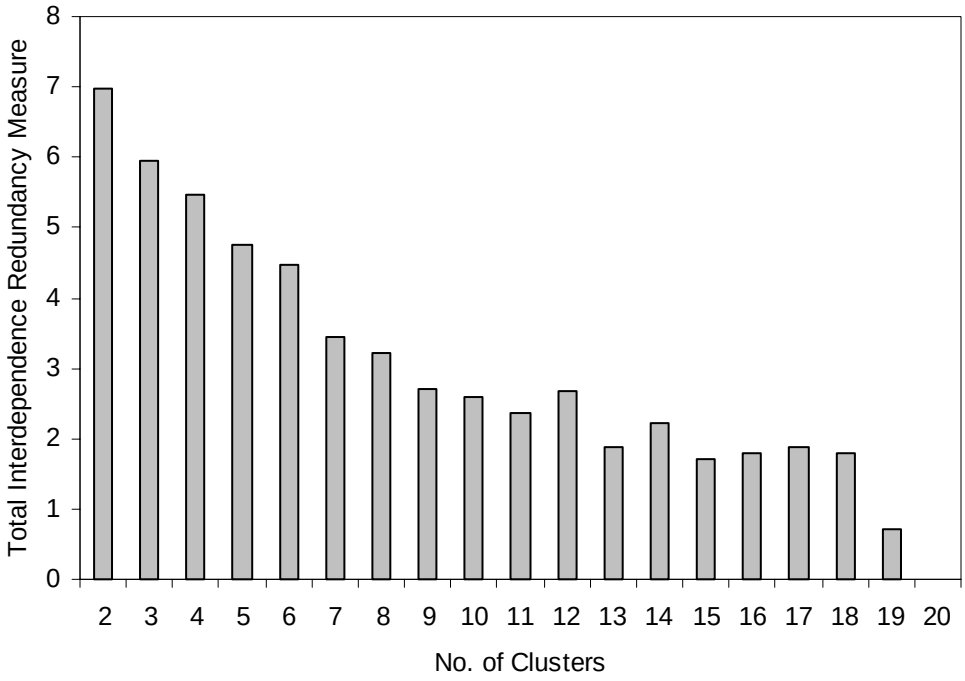



Fig. 9. The total interdependence redundancy measure over all the clusters found in the synthetic data set.

To evaluate the stability of the cluster configuration, we set the number of clusters to be 2 and ran ACA 190 times with different settings of initial modes. We ran 190 trials because there are ${}_{20}C_2$ (= 190) possible settings of initial modes for grouping the 20 attributes into 2 clusters in the synthetic data set. We examined the clusters of attributes formed in each trial. We found that ACA groups the attributes into the same cluster configuration in all the 190 trials. This shows that the cluster configuration formed by ACA

is optimal and stable over all the possible settings of initial modes in the synthetic data set.

For the purpose of comparison, we applied the *k*-means algorithm [McQueen 1967], Kohonen's SOM [Kohonen 2001], and the biclustering algorithm [Cheng and Church 2000] to the synthetic data set. When *k* is set to be 2, the *k*-means algorithm groups $\{A_1, A_3, \ldots, A_6, A_{17}, \ldots, A_{20}\}$ into a cluster and $\{A_2, A_7, \ldots, A_{16}\}$ into another cluster, whereas the biclustering algorithm groups $\{A_1, A_3, A_8, A_9, A_{10}, A_{13}, A_{14}, A_{16}, A_{17}, A_{20}\}$ into a cluster and $\{A_2, A_4, \ldots, A_7, A_{11}, A_{12}, A_{15}, A_{18}, A_{19}\}$ into another cluster. SOM produces 7 clusters: $\{A_{12}, A_{16}\}$, $\{A_7, A_8, A_{10}, A_{11}\}$, $\{A_9\}$, $\{A_2, A_{13}, A_{14}, A_{15}\}$, $\{A_1, A_3, A_5\}$, $\{A_{17}, \ldots, A_{20}\}$, and $\{A_4, A_6\}$. It is clear that the cluster configurations obtained by the *k*-means algorithm, SOM, and the biclustering algorithm are not able to represent the correlations between attributes hidden in the data.

After clusters of attributes were obtained, we selected the top attribute in each cluster for classification. The selected attributes were fed to C5.0 (a commercial version of C4.5 [Quinlan 1993], which is a popular decision tree based classification algorithm) for building classification models. We used C5.0 in this experiment because the classification models it builds are represented in the form of decision trees, which can be further examined.

For ACA, attributes $A_1$ and $A_2$ are selected and fed to C5.0. C5.0 builds a decision tree consisting of 5 leaf nodes and 4 non-leaf nodes that classifies all the tuples in the synthetic data set correctly. For *k*-means, $A_2$ and $A_6$ are selected and fed to C5.0. The decision tree built is composed of 6 leaf nodes and 5 non-leaf nodes. It misclassifies 23 tuples, which belong to $C_3$ but are classified as $C_2$. Biclusering algorithm selects $A_{12}$ and $A_{14}$. The decision tree built upon this result consists of 5 leaf nodes and 4 non-leaf nodes. It misclassifies 72 tuples, including 1 tuple belonging to $C_1$, 48 tuples belonging to $C_2$, and 23 tuples belonging to $C_3$. For SOM, $A_2$, $A_4$, $A_5$, $A_8$, $A_9$, $A_{12}$, and $A_{19}$ are selected. The decision tree built consists of 9 leaf nodes and 8 non-leaf nodes. Although the decision tree is rather complicated when compared to those constructed using the genes selected by ACA, the *k*-means algorithm, and the biclustering algorithm, it correctly classifies all of the tuples in the synthetic data set.

The experimental results on the synthetic data set show that ACA is a very promising and robust technique 1) to group attributes into clusters; 2) to select a subset of attributes from the clusters formed; and 3) to allow classification algorithms to build accurate classification models.

### *6.3.2 Gene Expression Data Sets*

To evaluate the performance of ACA, we applied it to two well-known gene expression data

sets: the *colon-cancer* data set [Alon *et al.* 1999] and the *leukemia* data set [Golub *et al.* 1999]. They are the same data sets used in [Li and Wong 2002a, 2002b] for gene selection.

#### 6.3.2.1 The Methodology for Evaluation

The difficulty of evaluation of the attribute clustering results is that we know too little about how genes actually associate among themselves. Although the rationale behind ACA is to group attributes by optimizing the intra-group attribute interdependence, we still have to justify the meaningfulness of such assumption backed by certain ground truth. Hence to have an objective and meaningful evaluation of ACA and others, we have to use what we know about the data to devise an evaluation scheme.

What we know about the two test data sets we used is that each of them could be classified into classes. The *colon-cancer* data set consists of 62 samples and 2,000 genes, which is represented by a 2,000 × 62 expression table. The samples are composed of tumor biopsies collected from tumors and normal biopsies collected from healthy part of the colons of the same patient. Each sample has been pre-classified into one of the two classes: *normal* and *cancer*. The *leukemia* data set consists of 72 samples and 7,129 genes, which is represented by a 7,129 × 72 expression table. The samples are taken from 63 bone marrow samples and 9 peripheral blood samples. They are either of type AML of leukemia or of type ALL as the two classes. Taking the pre-classified knowledge as ground truth we could devise an evaluation scheme as follows.

Since the task objective of the proposed methodology is clustering, we would like to ask how meaningful the clusters obtained are and what more useful information they contain. In view of this, we should first examine the cluster configuration and infer by observation, which one reveals more information about the data and gene groupings obtained. Next, we would like to get significant and insightful information from each cluster by selecting a subset of most representative genes and examining their patterns. Finally, we could use this extracted information for classification to see how the results obtained are backed by the ground truth. Our proposed scheme for evaluation and comparison can be outlined as follows.

1. The study of the cluster configuration obtained by different methods.
2. The study of representative patterns in each cluster found by them.
3. The result of gene classification based on the pool of top significant genes selected from each of the clusters.

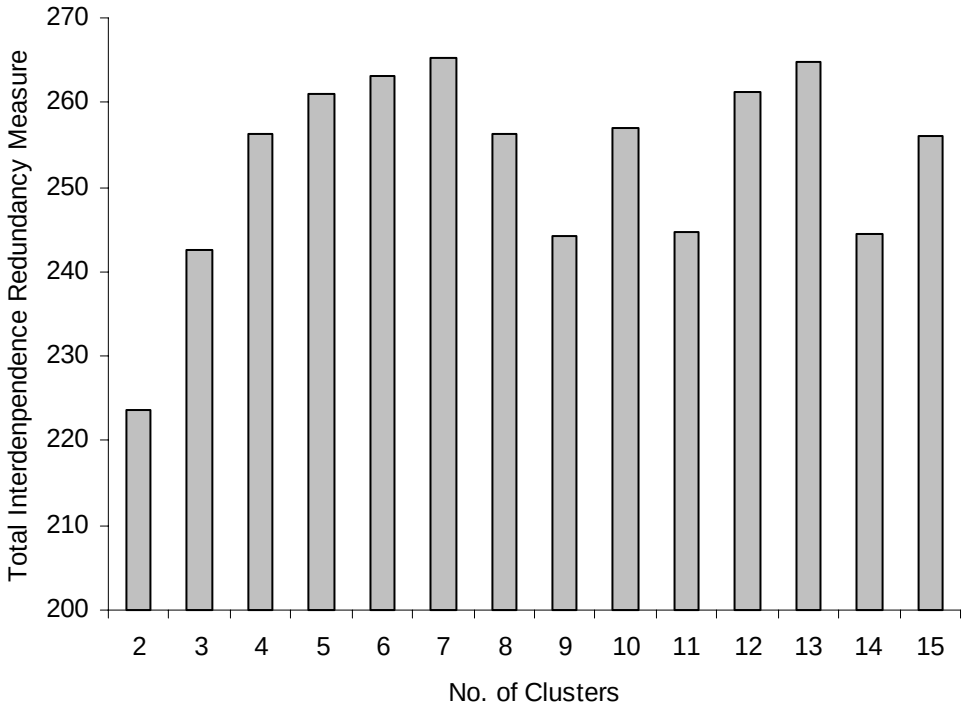
#### 6.3.2.2 The Cluster Configurations

In this study we would like to find out:

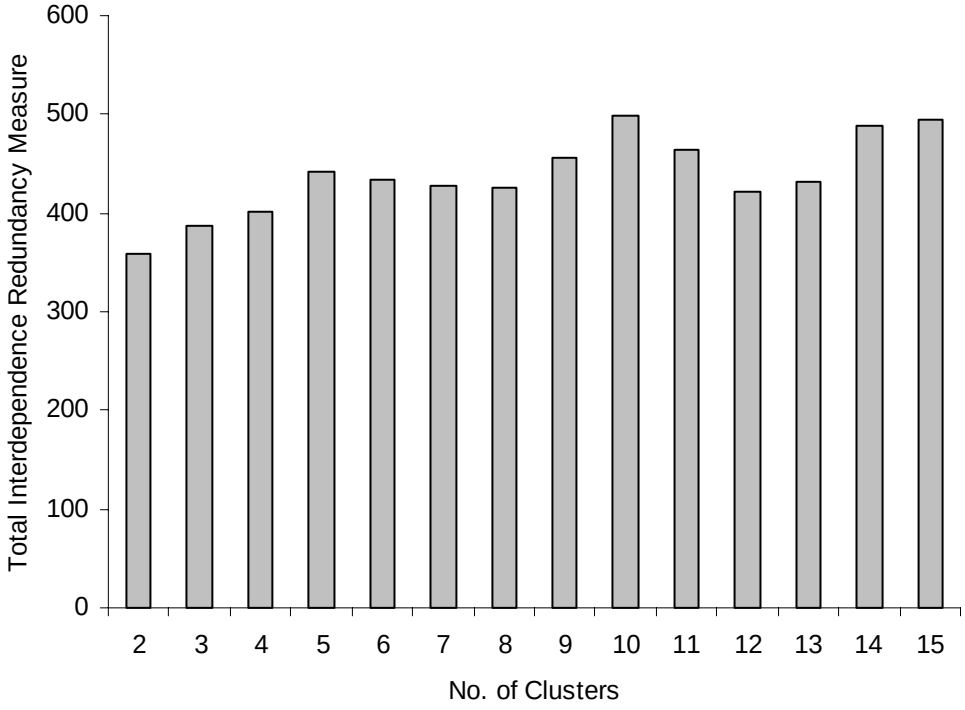
1. how optimal is the cluster configurations;
2. how do the clustering configuration patterns look like, viz. how evenly or lopsided are the cluster configurations; and
3. does each cluster contain distinctive patterns, and how discriminative they are between classes.

We first used our ITFP to fuzzy partition the domains of the genes (attributes) in the *colon-cancer* and *leukemia* data sets into 2 fuzzy intervals since there are only two classes in each case. This method was used because it can minimize the information lost in the fuzzy partitioning. We then applied ACA to the discretized data to find clusters of genes. Fig. 29 shows the sum of the interdependence redundancy measure over all the clusters versus the number of clusters formed from the *colon-cancer* and *leukemia* data sets.

In ACA, the cluster configuration is formed based on the maximization of intra-group attribute interdependence. As shown in Fig. 10, it reports that the optimal numbers of clusters for the *colon-cancer* and *leukemia* data sets are 7 and 10, respectively. The number of clusters found is optimal with respect to the intra-group attribute interdependence. This has been supported by various experiments on synthetic data including the one presented above. To investigate the representative patterns in each cluster, the top 5 genes, ranked according to the magnitude of their multiple interdependence redundancy in each cluster are selected and listed in Tables 5 and 6. We will study their patterns in next section.




(a) The *colon-cancer* data set.




(b) The *leukemia* data set.

Fig. 10. The total interdependence redundancy measure over all the clusters found in the gene expression data sets.

Table 5. The top 5 genes in each of the 7 clusters found in the *colon-cancer* data set.

| Cluster | Rank | Accession Number | Name |
|---|---|---|---|
| 1 | 1 | H05814 | PUTATIVE ATP-DEPENDENT RNA HELICASE C06E1.10 IN CHROMOSOME III (Caenorhabditis elegans) |
| 1 | 2 | X02874 | Human mRNA for (2'-5') oligo A synthetase E (1,6 kb RNA) |
| 1 | 3 | U33429 | human K+ channel beta 2 subunit mRNA, complete cds |
| 1 | 4 | H22579 | INTEGRIN ALPHA-6 PRECURSOR (Homo sapiens) |
| 1 | 5 | H25940 | PUTATIVE SERINE/THREONINE-PROTEIN KINASE PSK-H1 (Homo sapiens) |
| 2 | 1 | T73092 | EUKARYOTIC INITIATION FACTOR 4A-I (Homo sapiens) |
| 2 | 2 | R26146 | NUCLEAR FACTOR NF-KAPPA-B P105 SUBUNIT (HUMAN) |
| 2 | 3 | T90851 | ADP-RIBOSYLATION FACTOR-LIKE PROTEIN 4 (Rattus norvegicus) |
| 2 | 4 | R93337 | HOMEOTIC GENE REGULATOR (Drosophila melanogaster) |
| 2 | 5 | T69446 | EUKARYOTIC INITIATION FACTOR 4A-I (HUMAN) |
| 3 | 1 | M26383 | Human monocyte-derived neutrophil-activating protein (MONAP) mRNA, complete cds |
| 3 | 2 | U34252 | Human r-aminobutyraldehyde dehydrogenase mRNA, complete cds |
| 3 | 3 | T59162 | SELENIUM-BINDING PROTEIN (Mus musculus) |
| 3 | 4 | M27749 | IMMUNOGLOBULIN-RELATED 14.1 PROTEIN PRECURSOR (HUMAN) |
| 3 | 5 | T54341 | P25886 60S RIBOSOMAL PROTEIN L29 |
| 4 | 1 | T51849 | TYROSINE-PROTEIN KINASE RECEPTOR ELK PRECURSOR (Rattus norvegicus) |
| 4 | 2 | D13243 | Human pyruvate kinase-L gene, exon 12 |
| 4 | 3 | X52008 | H.sapiens alpha-2 strychnine binding subunit of inhibitory glycine receptor mRNA |
| 4 | 4 | R48936 | GLYCOPROTEIN VP7 (Chicken rotavirus a) |
| 4 | 5 | X14968 | Human testis mRNA for the RII-alpha subunit of cAMP dependent protein kinase |
| 5 | 1 | T90036 | CLASS I HISTOCOMPATIBILITY ANTIGEN, E-1 ALPHA CHAIN PRECURSOR (Pongo pygmaeus) |
| 5 | 2 | R81170 | TRANSLATIONALLY CONTROLLED TUMOR PROTEIN (Homo sapiens) |
| 5 | 3 | X67235 | H.sapiens mRNA for proline rich homeobox (Prh) protein |
| 5 | 4 | L20469 | Human truncated dopamine D3 receptor mRNA, complete cds |
| 5 | 5 | T63133 | THYMOSIN BETA-10 (HUMAN) |
| 6 | 1 | T92451 | TROPOMYOSIN, FIBROBLAST AND EPITHELIAL MUSCLE-TYPE (HUMAN) |
| 6 | 2 | H11460 | GOLIATH PROTEIN (Drosophila melanogaster) |
| 6 | 3 | H23975 | IG ALPHA-1 CHAIN C REGION (Gorilla gorilla gorilla) |
| 6 | 4 | R70030 | IG MU CHAIN C REGION (HUMAN) |
| 6 | 5 | D10522 | Human mRNA for 80K-L protein, complete cds. (HUMAN);contains element TAR1 repetitive element |
| 7 | 1 | H71627 | VITELLOGENIN A2 PRECURSOR (Xenopus laevis) |
| 7 | 2 | X74795 | H.sapiens P1-Cdc46 mRNA |
| 7 | 3 | T55840 | TUMOR-ASSOCIATED ANTIGEN L6 (Homo sapiens) |
| 7 | 4 | D17400 | Human mRNA for 6-pyruvoyl-tetrahydropterin synthase, complete cds |
| 7 | 5 | R71585 | EBNA-2 NUCLEAR PROTEIN (Epstein-barr virus) |

Table 6. The top 5 genes in each of the 10 clusters found in the *leukemia* data set.

| Cluster | Rank | Accession Number | Name |
|---|---|---|---|
| 1 | 1 | D21261_at | SM22-ALPHA HOMOLOG |
| 1 | 2 | X14362_at | CR1 Complement component (3b/4b) receptor 1, including Knops blood group system |
| 1 | 3 | HG3514-HT3708_at | Tropomyosin Tm30nm, Cytoskeletal |
| 1 | 4 | U91903_at | Frezzled (fre) mRNA |
| 1 | 5 | U44975_at | DNA-binding protein CPBP (CPBP) mRNA, partial cds |
| 2 | 1 | D25248_at | Randomly sequenced mRNA |
| 2 | 2 | X06290_at | APOLIPOPROTEIN(A) PRECURSOR |
| 2 | 3 | M21305_at | GB DEF = Alpha satellite and satellite 3 junction DNA sequence |
| 2 | 4 | HG3437-HT3628_s_at | Myelin Proteolipid Protein, Alt. Splice 2 |
| 2 | 5 | J03027_at | HLA-G MHC class I protein HLA-G |
| 3 | 1 | D26018_at | KIAA0039 gene, partial cds |
| 3 | 2 | X82018_at | ZID protein |
| 3 | 3 | U19107_rna1_at | ZNF127 (ZNF127) gene |
| 3 | 4 | U46746_s_at | Dystrobrevin-alpha mRNA |
| 3 | 5 | L39009_at | GB DEF = Class IV alcohol dehydrogenase 7 (ADH7) gene, 5' flanking region |
| 4 | 1 | M27891_at | CST3 Cystatin C (amyloid angiopathy and cerebral hemorrhage) |
| 4 | 2 | D26308_at | NADPH-flavin reductase |
| 4 | 3 | U10473_s_at | GB DEF = Clone p4betaGT/3 beta-1,4-galactosyltransferase mRNA, partial cds |
| 4 | 4 | Z35227_at | TTF mRNA for small G protein |
| 4 | 5 | Z32684_at | XK mRNA for membrane transport protein |
| 5 | 1 | D28124_at | Unknown product |
| 5 | 2 | U72648_s_at | GB DEF = Alpha2-C4-adrenergic receptor gene |
| 5 | 3 | HG4417-HT4687_f_at | Homeotic Protein Hpx-2 |
| 5 | 4 | HG2239-HT2324_r_at | Potassium Channel Protein (Gb:Z11585) |
| 5 | 5 | S59049_at | RGS1 Regulator of G-protein signaling 1 |
| 6 | 1 | D28416_at | GB DEF = Esterase D, 5'UTR (sequence from the 5'cap to the start codon) |
| 6 | 2 | D10656_at | CRK V-crk avian sarcoma virus CT10 oncogene homolog |
| 6 | 3 | M63483_at | MATRIN 3 |
| 6 | 4 | U13680_at | LDHC Lactate dehydrogenase C |
| 6 | 5 | M64571_at | MAP4 Microtubule-associated protein 4 |
| 7 | 1 | D29642_at | HYPOTHETICAL MYELOID CELL LINE PROTEIN 3 |
| 7 | 2 | U69108_at | TNF receptor associated factor 5 mRNA, partial cds |
| 7 | 3 | L07738_at | DIHYDROPRYRIDINE-SENSITIVE L-TYPE, SKELETAL MUSCLE CALCIUM CHANNEL GAMMA SUBUNIT |
| 7 | 4 | X83107_at | Bmx mRNA for cytoplasmic tyrosine kinase |
| 7 | 5 | U69140_s_at | RPS26 Ribosomal protein S26 |
| 8 | 1 | D30036_at | PHOSPHATIDYLINOSITOL |
| 8 | 2 | X58723_at | GB DEF = MDR1 (multidrug resistance) gene for P-glycoprotein |
| 8 | 3 | X67683_at | GB DEF = Keratin 4 |
| 8 | 4 | L00635_at | FNTB Farnesyltransferase, CAAX box, beta |
| 8 | 5 | J03890_rna1_at | SP-C1 gene (pulmonary surfactant protein SP-C) extracted from Human pulmonary surfactant protein C (SP-C) and pulmonary surfactant protein C1 (SP-C1) genes |
| 9 | 1 | D31764_at | KIAA0064 gene |
| 9 | 2 | S82471_s_at | GB DEF = SSX3=Kruppel-associated box containing SSX gene [human, testis, mRNA Partial, 675 nt] |
| 9 | 3 | D87434_at | KIAA0247 gene |
| 9 | 4 | U09877_at | Helicase-like protein (HLP) mRNA |
| 9 | 5 | L27624_s_at | TISSUE FACTOR PATHWAY INHIBITOR 2 PRECURSOR |
| 10 | 1 | D31891_at | KIAA0067 gene |
| 10 | 2 | Z22534_at | SERINE/THREONINE-PROTEIN KINASE RECEPTOR R1 PRECURSOR |
| 10 | 3 | HG4312-HT4582_s_at | Transcription Factor Iiia |
| 10 | 4 | U09477_at | Clone 53BP1 p53-binding protein mRNA, partial cds |
| 10 | 5 | U50315_at | EZH1 Enhancer of zeste (Drosophila) homolog 1 |

To facilitate the comparison of the cluster configurations, we applied the *k*-means algorithm [McQueen 1967], Kohonen's SOM [Kohonen 2001], and the biclustering algorithm [Cheng and Church 2000] to the original *colon-cancer* and *leukemia* data sets and compared the cluster results with that obtained by ACA on the respective sets of fuzzy partitioned data. Here, we shall discuss the issues of optimality of cluster configuration with regards to the number of clusters obtained.

By virtue of the theoretical basis and the design of the algorithm, given a specific setting of initial modes (cluster centers), ACA is able to determine the *k* that renders a clustering configuration that maximizes the intra-cluster interdependence of genes over various *k*. The cluster configuration selected is therefore an optimal one with respect to the setting of initial modes. It is important to note that the cluster configuration may not be optimal with different settings of initial modes. However, the experimental results on the synthetic data set presented in Section 6.3.1 show that the cluster configuration formed by ACA is optimal and stable over all the possible settings of initial modes. Although ACA does not guarantee to form an optimal cluster configuration because initial modes are chosen randomly, the experimental results show that it is able to produce a suboptimal, if not globally optimal, and stable configuration.

In forming clusters, both the *k*-means algorithm and the biclustering algorithm do not have a measure of the total dissimilarity over all the clusters. They cannot find the cluster number to justify the optimality of the cluster configuration. To deal with this problem, the *k*-means algorithm and the biclustering algorithm require a user to supply the number of clusters in advance.

SOM aims at optimizing the distances between the input vectors and the reference vectors. In other words, the reference vectors are moved towards the denser areas of the input vector space. To determine the number of clusters, SOM does so by not assigning any input vector to some output nodes in the neural network. This process is implicit in the training process of SOM and it does not explicitly optimize any measure of the total dissimilarity or distance measure over all the clusters. The number of clusters resulted is, by and large, conditioned by the convergence of the weights of the network links, which is, in a certain sense, a little ad hoc.

We next proceed to compare the representative patterns selected from each of the cluster. Since only ACA provides a clearly defined way to determine the number of clusters,

we apply each of the above methods to produce 7 and 10 clusters in the *colon-cancer* and *leukemia* data sets respectively for comparison purpose. We also apply the *t*-value and the methods that handle both the gene-class relevance and the gene-gene redundancy (i.e., the MRMR algorithm [Ding and Peng 2003] and the RBF algorithm [Yu and Liu 2004]) to rank the genes in the two data sets for the purpose of comparison. For the MRMR algorithm, we used the *F*-test correlation quotient as the criterion function because the experimental results in [Ding and Peng 2003] show that it yields better classification results than the other criterion functions for continuous features.

In each of the two data sets, the clusters found by ACA consist of more or less the same number of genes. However, the *k*-means algorithm groups 1,592 of the 2,000 genes (i.e., 79.6% of all the genes) into one cluster for the *colon-cancer* data set and groups 6,514 of the 7,129 genes (i.e., 91.4% of all the genes) into one cluster for the *leukemia* data set. The cluster distribution produced by SOM is less lopsided. It groups 708 of the 2,000 genes (i.e., 35.4% of all the genes) into one cluster for the *colon-cancer* data set, whereas the clusters it finds in the *leukemia* data set contain more or less the same number of genes. Similar to ACA, the biclustering algorithm also forms clusters containing more or less the same number of genes. Comparing the cluster size distribution, those produced by ACA and the biclustering algorithm are less lopsided. Of the other two, *k*-means produces the most lopsided distribution for both data sets.

In the rest of this section, we examine the gene ranking obtained by different approaches. Since the clusters found by ACA are less lopsided and the genes selected are informative (whose effectiveness is reflected by the classification experiments presented in Section 7.4.4), the cluster configuration obtained by it and the top genes selected would provide a reasonable basis for performance comparison. Therefore, they will be used as the benchmark in the comparison process.

In the *colon-cancer* data set, of the top 35 genes ranked by the *t*-value, 22 are in Cluster 2 and none is in Cluster 1 found by ACA. Furthermore, none of the 35 genes is ranked in the top 5 in any of the clusters found by ACA. On the other hand, in the *leukemia* data set, none of the top 50 genes ranked by the *t*-value is in Clusters 5 and 8 found by ACA. 7 of these genes are ranked in the first 100th in one of the 10 clusters found by ACA. Specifically, genes M27891_at and D21261_at are also selected by ACA. However, many of the genes selected by the *t*-value are ranked very low in the clusters found by ACA. For example, gene J05032 in the *colon-cancer* data set, which is ranked the fourth by the *t*-value, is ranked the 174th in Cluster 2 found by ACA, whereas gene J03589_at in the *leukemia* data set, which is ranked the 46th by the *t*-value, is ranked the 796th in Cluster 3 found by

ACA.

For the MRMR algorithm, 3 of the 35 genes selected in the *colon-cancer* data set (i.e., genes R93337, U33429, and R81170) and 1 of the 50 genes selected in the *leukemia* data set (i.e., gene M63483_at) are also selected by ACA. In spite of these genes, many of the genes selected by the MRMR algorithm in both the *colon-cancer* and *leukemia* data sets are ranked very low in the clusters found by ACA. For example, gene R16255 in the *colon-cancer* data set, which is ranked the first by the MRMR algorithm, is ranked the 218th in Cluster 3 found by ACA, whereas gene M19483_at in the *leukemia* data set, which is ranked the fourth by the MRMR algorithm, is ranked the 944th in Cluster 10 found by ACA.

The RBF algorithm selects 3 genes only in both of the *colon-cancer* and *leukemia* data sets. None of these genes is ranked in the first 100th in any cluster found by ACA. For example, gene J05032 in the *colon-cancer* data set, which is ranked the third by the RBF algorithm, is ranked the 175th in Cluster 2 found by ACA, whereas gene X61373_at in the *leukemia* data set, which is ranked the first by the RBF algorithm, is ranked the 170th in Cluster 7 found by ACA.

For the *k*-means algorithm, none of the 35 genes selected in the *colon-cancer* data set is in Cluster 7 found by ACA. One of these genes (i.e., gene U34252) is also selected by ACA. In both of the *colon-cancer* and *leukemia* data sets, many of the genes selected by the *k*-means algorithm are ranked very low in the clusters found by ACA. For example, gene T51496 in the *colon-cancer* data set, which is ranked the third in Cluster 4 found by the *k*-means algorithm, is ranked the 313th in Cluster 4 found by ACA, whereas gene X62691_at in the *leukemia* data set, which is ranked the first in Cluster 9 found by the *k*-means algorithm, is ranked the 607th in Cluster 3 found by ACA.

For SOM, in the *colon-cancer* data set, none of the 35 genes selected is in Cluster 6 found by ACA and none of them is ranked in the top 5 in any of the clusters found by ACA. In the *leukemia* data set, one of the 50 genes selected by SOM (i.e., gene D21261_at) is also selected by ACA. Nevertheless, many of these genes selected by SOM are ranked very low in the clusters found by ACA. For example, gene T47424 in the *colon-cancer* data set, which is ranked the second in Cluster 3 found by SOM, is ranked the 314th in Cluster 2 found by ACA, whereas gene D14710_at in the *leukemia* data set, which is ranked the second in Cluster 2 found by SOM, is ranked the 466th in Cluster 3 found by ACA.

For the biclustering algorithm, none of the 35 and 50 genes selected in the *colon-cancer* and *leukemia* data sets, respectively, is ranked in the top 5 in any of the clusters

found by ACA. Many of these genes selected by the biclustering algorithm are ranked very low in the clusters found by ACA. For example, gene L07032 in the *colon-cancer* data set, which is ranked the third in Cluster 1 found by the biclustering algorithm, is ranked the 269th in Cluster 3 by ACA, whereas gene S79862_s_at in the *leukemia* data set, which is ranked the first in Cluster 1 found by the biclustering algorithm, is ranked the 382th in Cluster 4 found by ACA.

The ranking of the genes selected by the *t*-value, the *k*-means algorithm, SOM, the biclustering algorithm, the MRMR algorithm, and the RBF algorithm with respect to that selected by ACA in the *colon-cancer* and *leukemia* data sets is summarized in Tables 7 and 8, respectively. The first row in Table 7 gives the number of the 35 genes selected by the *t*-value, the *k*-means algorithm, SOM, the biclustering algorithm, the MRMR algorithm, and the RBF algorithm that are ranked in the top 5 in any of the 7 clusters found by ACA; the second row in Table 7 gives the number of the 35 genes selected by the *t*-value, the *k*-means algorithm, SOM, the biclustering algorithm, the MRMR algorithm, and the RBF algorithm that are ranked from the 6th to the 15th in any of the 7 clusters found by ACA; and so on for the other rows. The details of Table 8 can be interpreted in a similar fashion.

The comparison of the ranking of genes by the other six methods with the benchmark ranking by ACA is important. Since top ranking genes selected by ACA yield excellent classification results, the cross comparison of genes selected by other six methods would shed light on which genes would have high or low classificatory value and why. This will be discussed in Sections 6.3.2.3 and 6.3.2.4.

Table 7. The ranking of the 35 genes selected by different approaches in the *colon-cancer* data set.

| **Rank in the clusters found by ACA** | ***t*-value** | ***k*-means** | **SOM** | **Biclustering** | **MRMR** | **RBF** |
|---|---|---|---|---|---|---|
| 1–5 | 0 | 1 | 0 | 0 | 3 | 0 |
| 6–15 | 4 | 1 | 3 | 3 | 2 | 0 |
| 16–50 | 2 | 6 | 6 | 5 | 1 | 0 |
| 51–100 | 14 | 5 | 2 | 5 | 6 | 0 |
| 101–200 | 7 | 5 | 9 | 3 | 17 | 2 |
| 201–350 | 8 | 17 | 15 | 19 | 21 | 1 |

Table 8. The ranking of the 50 genes selected by different approaches in the *leukemia* data set.

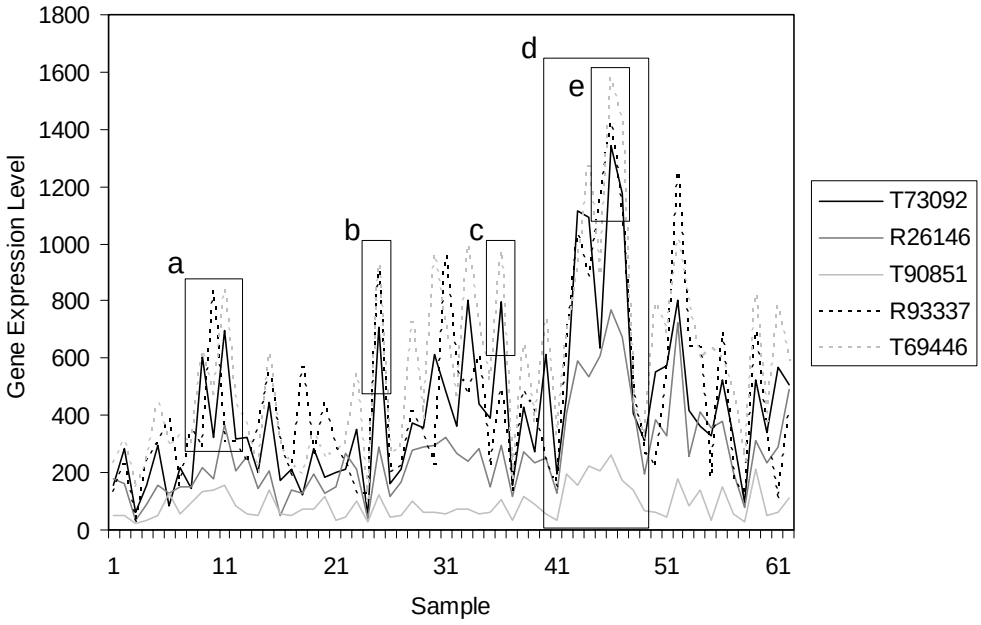
| Rank in the clusters found by ACA | *t*-value | *k*-means | SOM | Biclustering | MRMR | RBF |
|---|---|---|---|---|---|---|
| 1–10 | 2 | 1 | 1 | 1 | 1 | 0 |
| 11–50 | 2 | 2 | 1 | 2 | 2 | 0 |
| 51–100 | 3 | 3 | 3 | 1 | 4 | 0 |
| 101–200 | 7 | 8 | 7 | 6 | 4 | 1 |
| 201–300 | 4 | 5 | 10 | 10 | 6 | 1 |
| 301–500 | 13 | 21 | 12 | 11 | 15 | 1 |
| 501–700 | 8 | 9 | 9 | 8 | 6 | 0 |
| 701–1000 | 11 | 1 | 7 | 11 | 12 | 0 |

#### 6.3.2.3 The Patterns of Genes in the Clusters

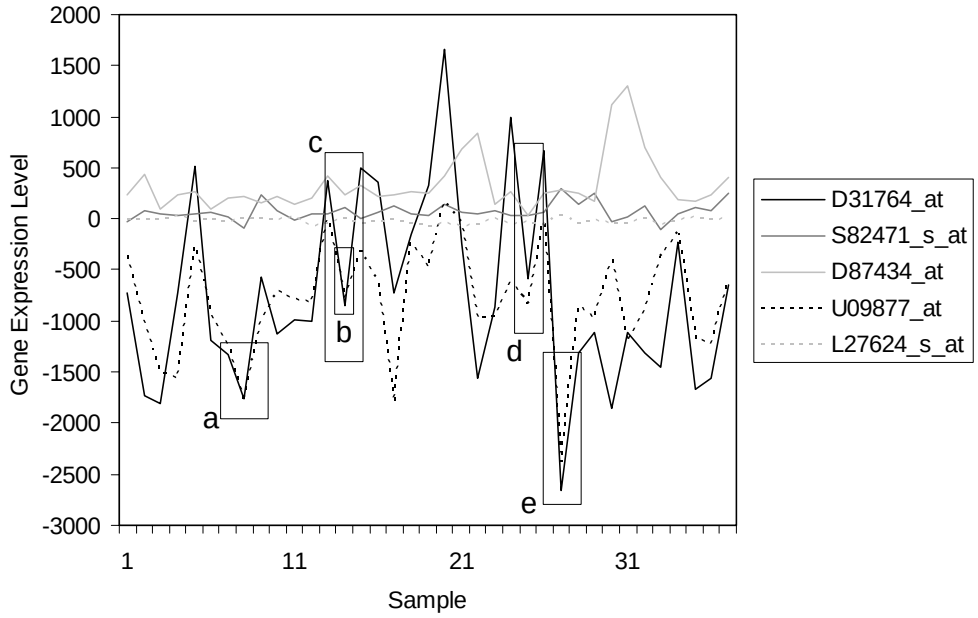
In this section, we will select the most significant genes in each cluster based on certain criterion functions and examine their patterns, respectively. More specifically, we would like to find out:

1. how coherent are the most representative genes and
2. do they reflect coherence, interdependence, similarity or both and what are the implications of such relationship in these patterns.

To address the above issues, we selected some of the results obtained from ACA, *k*-means, SOM, and biclustering for discussions. Fig. 11 shows the most representative genes in Cluster 2 found by ACA in the *colon-cancer* data set and those in Cluster 9 found by ACA in the *leukemia* data set. The gene segments highlighted in boxes b, c, and e in Fig. 11(a) and boxes a, b, and e in Fig. 11(b) are similar to each other. Gene segments that are interdependent with each other are grouped together and shown in box d in Fig. 11(a). Note that the two plots at the lower part of the box are more or less correlated or interdependent with the curve near the top of the box although they are not similar to them because of the huge distance magnitude from them. Gene segments highlighted in box a in Fig. 11(a) and boxes c and d in Fig. 11(b) are also interdependent even though some of the segment pairs are negatively correlated. It illustrates that the interdependence redundancy measure can clusters genes using both similarity and interdependence measures. This may contribute to the high attribute association results of ACA as reported in Section 6.3.2.4.




(a) Cluster 2 in the *colon-cancer* data set.




(b) Cluster 9 in the *leukemia* data set.

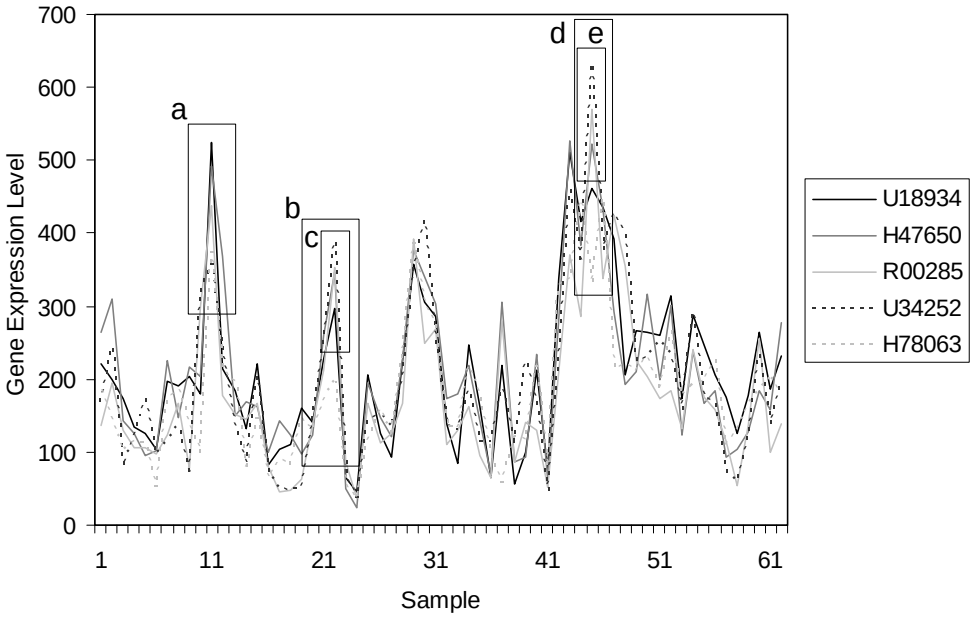
Fig. 11. The most representative genes found by ACA.

Fig. 12 shows the most representative genes in Cluster 2 found by the *k*-means algorithm in the *colon-cancer* data set and those in Cluster 1 found by the *k*-means algorithm in the *leukemia* data set. The gene segments highlighted in boxes a, c, and e in Fig. 12(a) and box c in Fig. 12(b) are similar to each other. In Fig. 12(a), those highlighted in boxes b and d are dissimilar in such a way that gene H78063's shape is distanced from

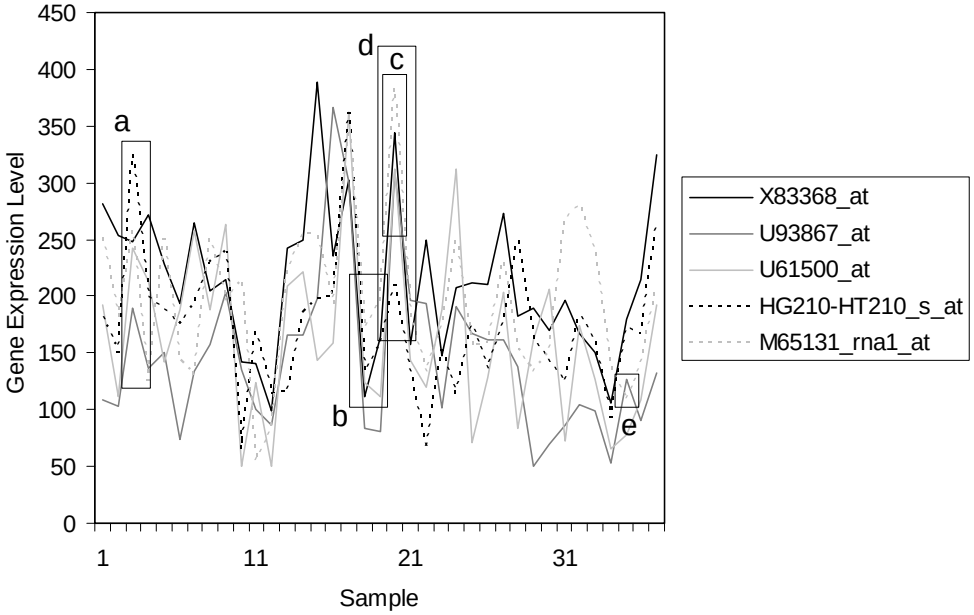
the others. However, these gene segments are still interdependent although they are not similar. The similar phenomena are observed in the gene segments highlighted in boxes a, b, and d in Fig. 12(b). Those highlighted in box e are interdependent but negatively correlated. Hence, clustering algorithms based on similarity are unable to group genes which are interdependent. It is perhaps for this reason that the genes selected by ACA contain more classificatory information. The high classification rate of ACA as reported later in Section 6.3.2.4 may attribute to gene interdependence as it is conceivable that interdependence is a key factor that makes up classes.

Figs. 13 and 14 show the most representative genes found by SOM and the biclustering algorithm in the *colon-cancer* and *leukemia* data sets, respectively. The gene segments highlighted in boxes in Figs. 30 and 31 are less coherent. This indicates that the genes in a cluster found by SOM and the biclustering algorithm are less coherent, i.e., they are by and large not that much similar nor interdependent. It is perhaps for this reason that the genes selected by SOM and the biclustering algorithm are not very useful for classification (see Section 6.3.2.4).

On the whole, the patterns in the top representative genes selected by ACA are most coherent in the sense of interdependence that embodies similarity as well as positive and negative correlation. The patterns selected in association with the *k*-means results are coherent only in the similarity sense. There are positively correlated, negatively correlated and/or interdependent segments which are not accounted for by the distance measure employed. The plots from SOM and the biclustering algorithm show that they are less coherent. While they are able to account for positive correlation and negative correlation separately, they are not able to account for both especially when they occur along the gene segments in the comparison. The interdependence measure accounts for all.

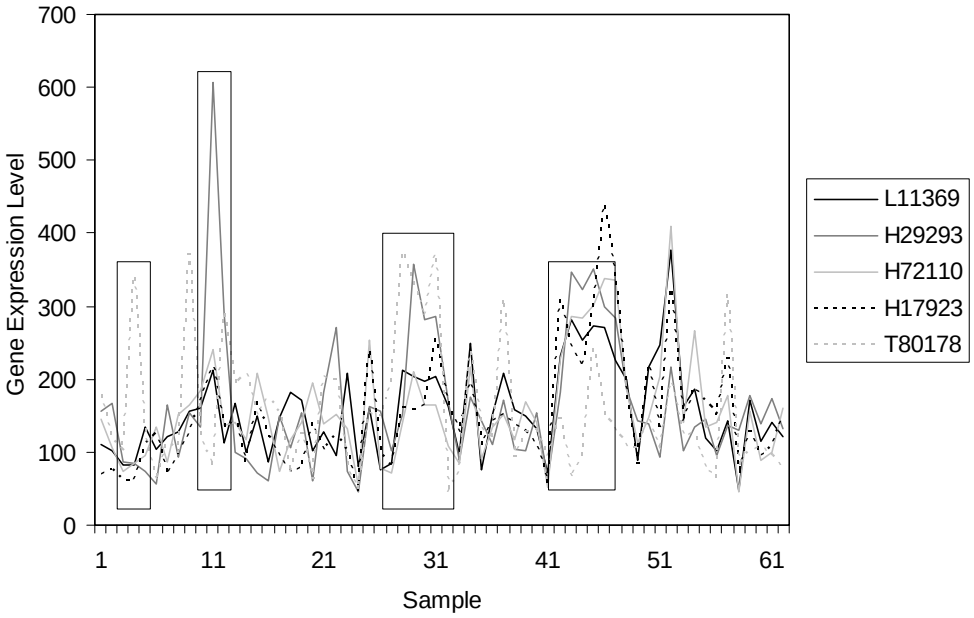



(a) Cluster 2 in the *colon-cancer* data set.

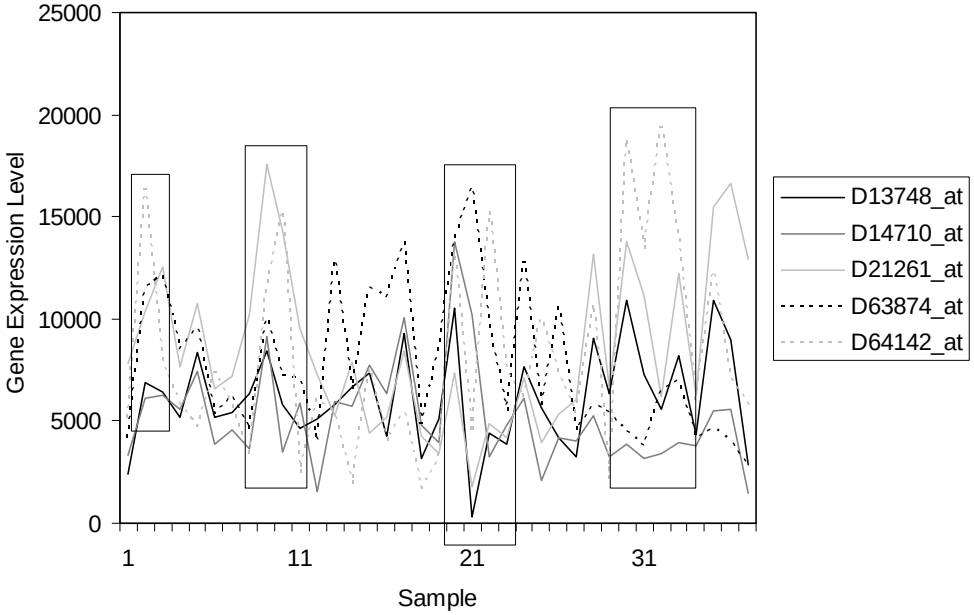



(b) Cluster 1 in the *leukemia* data set.

Fig. 12. The most representative genes found by the *k*-means algorithm.

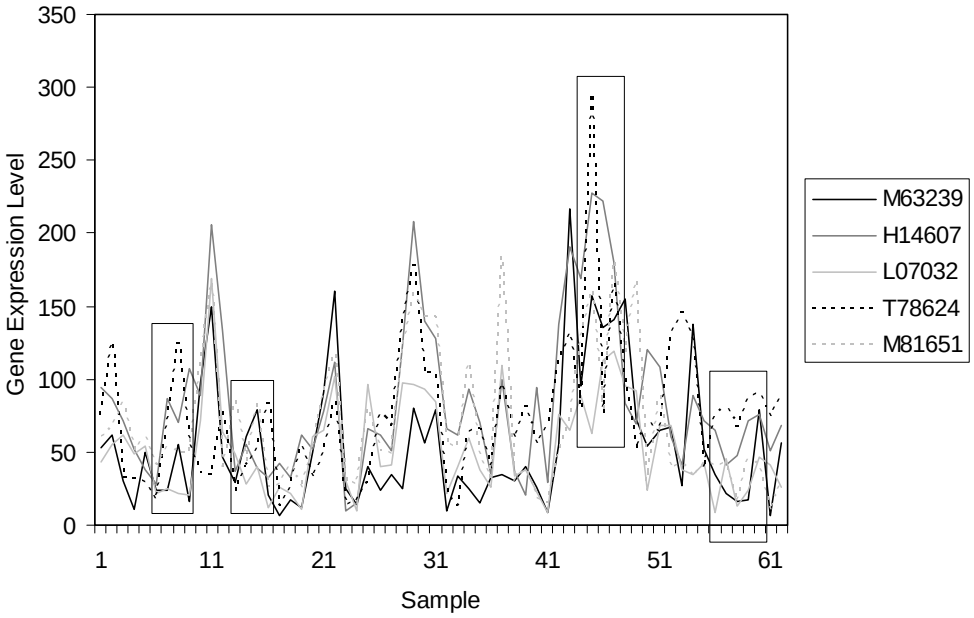



(a) Cluster 1 in the *colon-cancer* data set.

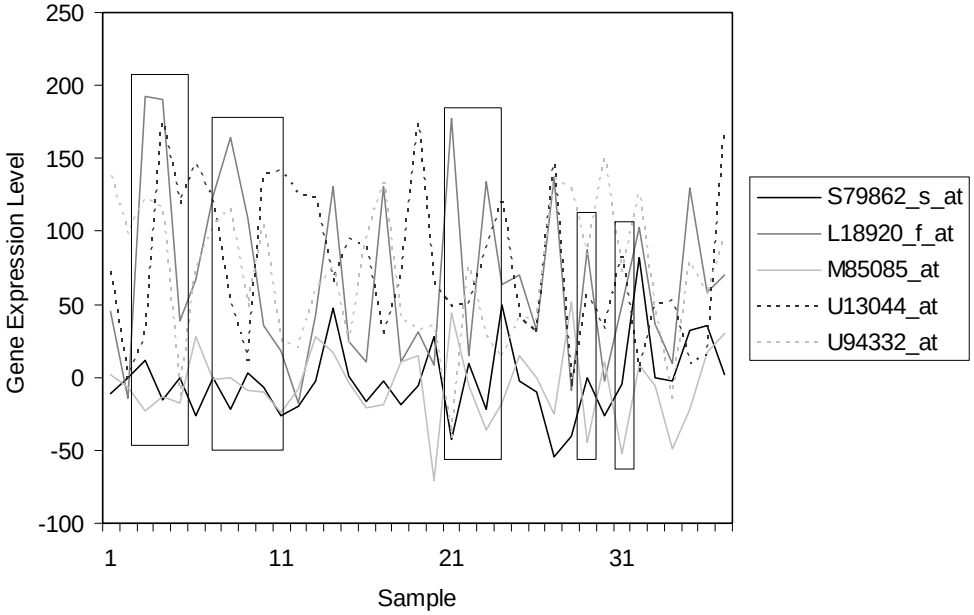



(b) Cluster 2 in the *leukemia* data set.

Fig. 13. The most representative genes found by SOM.




(a) Cluster 1 in the *colon-cancer* data set.




(b) Cluster 1 in the *leukemia* data set.

Fig. 14. The most representative genes found by the biclustering algorithm.

### 6.3.2.4 Gene Expression Classification

Since the ground truth of class labels for these two gene expression datasets is known, we use this information to devise experiments for assessing the performance of various methods. The evaluation scheme is depicted in Fig. 15. First, to show how much classificatory information could get from the data, we use both the clustering and the attribute selection results obtained by the listed methods for evaluation. That is, we obtain a set of clusters from the genes. We then select a subset of top genes from each cluster to make up a gene pool. We then run classification experiments on the selected gene pool to see whether or not the results are backed by the ground truth and which method performs the best.

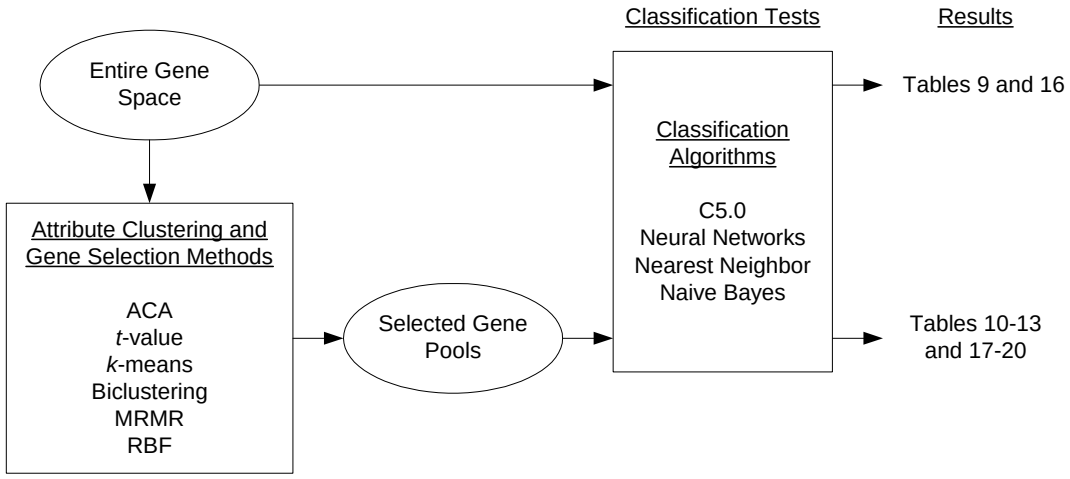



Fig. 15. The scheme for evaluating the classificatory effectiveness of gene pools.

The results obtained by applying the listed classifiers on the data taken from the entire gene space are given in Tables 9 and 16 while the results obtained by the same set of classifiers on the gene pools selected by different attribute clustering and gene selection methods are documented on Tables 10–13 and Tables 17–20.

The argument on the appropriateness of such evaluation scheme is as follows. If the selected genes are informative, an inductive learning algorithm should be able to build an accurate classifier on top of them. Based on this idea, we selected the top $k$ genes from each of the clusters so that a total of $7 \times k$ and $10 \times k$ genes are selected for $k = 1, \ldots, 5$ in the *colon-cancer* and *leukemia* data sets, respectively. We then used C5.0, nonlinear neural networks with a single hidden layer and weight decay [Bishop 1995], the nearest neighbor method, and the naïve Bayes method to build classifiers on top of the selected genes. These classification algorithms are used in this work because they have been employed in classification of gene expression data in the literature [Ben-Dor *et al.* 2000; Dudoit, Fridlyand, and Speed 2002; Friedman, Nachman, and Pe'er 2000; Keller *et al.* 2000; Khan

*et al.* 2001; Lu and Han 2003; Zhang *et al.* 2001].

#### *6.3.2.4.1 The Colon-Cancer Data Set*

In the classification performance evaluation process, we employed the *leave-one-out cross-validation* (LOOCV), which is a widely used process for gene expression data classification [Simon 2003]. With LOOCV, we selected the first sample as the test set and the remaining 61 samples as the training set. Repeating through the first sample to the 62nd sample, we got the classification accuracy (i.e., the percentage of the samples, which are predicted correctly).

As the benchmark, we first trained C5.0, neural networks, the nearest neighbor method, and the naïve Bayes method with all 2,000 genes without gene selection. The classification accuracy by LOOCV is given in Table 9. To evaluate the attribute clustering and gene selection performance of ACA, the selected gene pools were fed to the same group of classification algorithms. For comparison purpose, we repeated the gene selection process using the *t*-value, the *k*-means algorithm, SOM, the biclustering algorithm, the MRMR algorithm, and the RBF algorithm. The classification results of the classifiers built on different gene pools are provided in Tables 10–13.

Table 9. The performance of different classification algorithms in the *colon-cancer* data set.

| Classification Algorithm | Classification Accuracy |
|---|---|
| C5.0 | 82.3% |
| Neural Networks | 83.9% |
| Nearest Neighbor | 79.0% |
| Naïve Bayes | 35.5% |

Table 10. The performance of C5.0 on the top genes selected by different techniques in the *colon-cancer* data set.

| No. of Genes Selected | Classification Accuracy | | | | | |
|---|---|---|---|---|---|---|
| | **ACA** | ***t*-value** | ***k*-means** | **SOM** | **Biclustering** | **MRMR** |
| 7 | **88.7%** | 83.9% | 64.5% | 64.5% | 67.7% | 80.6% |
| 14 | **91.9%** | 77.4% | 77.4% | 59.7% | 58.1% | 75.8% |
| 21 | **91.9%** | 82.3% | 75.8% | 58.1% | 69.4% | 83.9% |
| 28 | **91.9%** | 85.5% | 74.2% | 48.4% | 77.4% | 83.9% |
| 35 | **91.9%** | 74.2% | 71.0% | 43.5% | 75.8% | 83.9% |

* The RBF algorithm selects 3 genes only and achieves a classification accuracy of 82.3%.

Table 11. The performance of neural networks on the top genes selected by different techniques in the *colon-cancer* data set.

| No. of Genes Selected | Classification Accuracy | | | | | |
|---|---|---|---|---|---|---|
| | **ACA** | ***t*-value** | ***k*-means** | **SOM** | **Biclustering** | **MRMR** |
| 7 | **90.3%** | 80.6% | 71.0% | 64.5% | 75.8% | 87.1% |
| 14 | **90.3%** | 87.1% | 83.9% | 75.8% | 72.6% | **90.3%** |
| 21 | **90.3%** | 83.9% | 77.4% | 75.8% | 82.3% | 87.1% |
| 28 | **90.3%** | 80.6% | 85.5% | 67.7% | 72.6% | **90.3%** |
| 35 | **90.3%** | 80.6% | 87.1% | 67.7% | 79.0% | **90.3%** |

* The RBF algorithm selects 3 genes only and achieves a classification accuracy of 90.3%.

Table 12. The performance of the nearest neighbor method on the top genes selected by different techniques in the *colon-cancer* data set.

| No. of Genes Selected | Classification Accuracy | | | | | |
|---|---|---|---|---|---|---|
| | **ACA** | ***t*-value** | ***k*-means** | **SOM** | **Biclustering** | **MRMR** |
| 7 | **83.9%** | 80.6% | 58.1% | 50.0% | 69.4% | 64.5% |
| 14 | **82.3%** | 80.6% | 69.4% | 59.7% | 62.9% | 56.5% |
| 21 | **82.3%** | 80.6% | 64.5% | 59.7% | 53.2% | 61.3% |
| 28 | **82.3%** | 79.0% | 61.3% | 58.1% | 64.5% | 67.7% |
| 35 | **80.6%** | 75.8% | 62.9% | 54.8% | 53.2% | 72.6% |

* The RBF algorithm selects 3 genes only and achieves a classification accuracy of 67.7%.

Table 13. The performance of the naïve Bayes method on the top genes selected by different techniques in the *colon-cancer* data set.

| No. of Genes Selected | Classification Accuracy | | | | | |
|---|---|---|---|---|---|---|
| | **ACA** | ***t*-value** | ***k*-means** | **SOM** | **Biclustering** | **MRMR** |
| 7 | 64.5% | 56.5% | 62.9% | 64.5% | **67.7%** | 64.5% |
| 14 | **67.7%** | 53.2% | 62.9% | 29.0% | **67.7%** | 64.5% |
| 21 | **67.7%** | 45.2% | 62.9% | 29.0% | 48.4% | 38.7% |
| 28 | **67.7%** | 35.5% | 56.5% | 29.0% | 48.4% | 43.5% |
| 35 | **67.7%** | 38.7% | 56.5% | 29.0% | 48.4% | 43.5% |

* The RBF algorithm selects 3 genes only and achieves a classification accuracy of 64.5%.

The experimental results in Tables 10–13 show that ACA is, by and large, superior to the other six attribute clustering and gene selection methods by selecting a better small set of discriminative genes in the *colon-cancer* data set than the others as reflected by the classification results. It is surprised to observe that the classification results obtained using the gene pools selected by ACA and *t*-value are even better than those using all the genes. And, as shown by the results, ACA outperforms *t*-value in all cases. Although the MRMR and RBF algorithms can find good discriminative genes for C5.0, neural networks, and the naïve Bayes method, they are unable to do so for the nearest neighbor method. As shown in the results, ACA outperforms the MRMR and RBF algorithms in all cases except neural networks, in which the three approaches yield comparable classification rate. The *k*-means

algorithm, SOM, and the biclustering algorithm fail to find the good discriminative genes as shown in the results. This result shows that it is able to build a more accurate classifier if a subset of more informative genes based on multiple interdependence is selected by ACA before feeding them into the classifier for training.

It is interesting to note that the performance of C5.0 is able to achieve a 91.9% when using the 14 genes selected by ACA and maintain at the same accuracy even when more genes are selected by ACA (see Table 10). This implies that the good diagnostic information exists in a small set of genes which can be effectively selected by ACA and a small set of genes can be used to build classifiers for diagnostic purpose. This has a significant implication to clinical, pharmaceutical, and bioengineering applications. Similarly, the same phenomenon is observed in the 90.3% rate when 7 genes selected by ACA is fed into neural networks classifier and its performance remains at that level even when more genes selected by ACA are fed in (see Table 11). This suggests that using only the top 1 or 2 genes in each cluster found by ACA are already good enough for training C5.0 and neural networks.

On the other hand, the poor classification performance using the set selected by the *k*-means algorithm, SOM, and the biclustering algorithm (see Tables 9–13) may be explained by our observation that their selected top genes are ranked very low by ACA (see Table 7). In another words, they are less interdependent with other genes in the group. To further this argument, we also observe that the genes selected by *t*-value are ranked relatively high by ACA in comparison to the other three (see Table 7).

Since the *k*-means algorithm and the biclustering algorithm do not provide a criterion function to show which *k* would give the most optimal configuration, we will evaluate it by varying *k* to see which *k* will produce the best result. As shown in Tables 10–13, the *k*-means algorithm yields the best result when the top 5 genes in each cluster are selected and fed to neural networks (87.1% as shown in Table 11), whereas the biclustering algorithm achieves the best result when the top 3 genes in each cluster are selected and fed to neural networks (82.3% as shown in Table 11). In order to use their best performance results for comparison, we select the top 5 genes from each cluster for the *k*-means algorithm and the top 3 genes for the biclustering algorithm. The classification performance by neural networks on the top genes selected by the *k*-means algorithm and the biclustering algorithm with different number of clusters is given in Tables 14 and 15, respectively. The experimental results show that the performance of using 7 clusters (where 7 is the cluster number determined by ACA) is close to the best result (87.1% for the *k*-means algorithm and 82.3% for the biclustering algorithm as shown in Table 11). With the same

configuration ACA achieves at a 90.3% rate. It is interesting to observe that the number of clusters determined by ACA, if used as a candidate of *k*, both of the *k*-means algorithm and the biclustering algorithm yields the second best result.

Table 14. The performance of neural networks on the top genes selected by the *k*-means algorithm in the *colon-cancer* data set.

| No. of Clusters Found | Classification Accuracy |
|---|---|
| 2 | 64.5% |
| 4 | 80.6% |
| 6 | 80.6% |
| 8 | 88.7% |
| 10 | 83.9% |
| 15 | 88.7% |
| 20 | 88.7% |

Table 15. The performance of neural networks on the top genes selected by the biclustering algorithm in the *colon-cancer* data set.

| No. of Clusters Found | Classification Accuracy |
|---|---|
| 2 | 74.2% |
| 4 | 64.5% |
| 6 | 80.6% |
| 8 | 79.0% |
| 10 | 83.9% |
| 15 | 83.9% |
| 20 | 64.5% |

SOM is able to determine the number of clusters automatically. It determines that there are 35 clusters. As shown in Tables 10–13, SOM produces the best result when the top 2 and 3 genes in each cluster are selected and fed to neural networks (75.8% as shown in Table 11). We therefore evaluated the performance of neural networks using the top 2 and 3 genes in each of the 35 clusters found by SOM and found that the classification accuracy is 87.1% and 88.7%, respectively. It is important to note that ACA obtains a classification accuracy of 90.3% using 7 genes only (see Table 11).

#### *6.3.2.4.2 The Leukemia Data Set*

We next report the performance of ACA based on the classification results on the *leukemia* data set. The data set taken from the website is already divided into a training set, which consists of 38 samples, and a test set, which consists of 34 samples, by the donor of the data set. Like what we did for the *colon-cancer* data, we used C5.0, neural networks, the nearest neighbor method, and the naïve Bayes method to build classifiers using the selected genes as the training set. The classifiers thus built were tested on the samples in the test set.

Again, as the benchmark, we first trained C5.0, neural networks, the nearest neighbor method, and the naïve Bayes method with all 7,129 genes. The classification results are given in Table 16. To evaluate the attribute clustering and gene selection performance of ACA, its selected gene pools were fed to the same group of classification algorithms. For the classification comparison purpose, we fed into the same group of classifiers the gene selected by the *t*-value, the *k*-means algorithm, SOM, the biclustering algorithm, the MRMR algorithm, and the RBF algorithm using the similar process. The classification results of the classifiers built on respective gene pools are provided in Tables 17–20.

Table 16. The performance of different classification algorithms in the *leukemia* data set.

| **Classification Algorithm** | **Classification Accuracy** |
|---|---|
| C5.0 | 91.2% |
| Neural Networks | 91.2% |
| Nearest Neighbor | 82.4% |
| Naïve Bayes | 41.2% |

Table 17. The performance of C5.0 on the top genes selected by different techniques in the *leukemia* data set.

| **No. of Genes Selected** | **Classification Accuracy** | | | | | |
|---|---|---|---|---|---|---|
| | **ACA** | ***t*-value** | ***k*-means** | **SOM** | **Biclustering** | **MRMR** |
| 10 | **94.1%** | **94.1%** | 47.1% | 55.9% | 71.1% | 91.2% |
| 20 | **94.1%** | **94.1%** | 55.9% | 55.9% | 60.5% | 91.2% |
| 30 | **94.1%** | **94.1%** | 55.9% | 64.7% | 65.8% | 91.2% |
| 40 | **94.1%** | **94.1%** | 55.9% | 61.8% | 57.9% | 91.2% |
| 50 | **94.1%** | **94.1%** | 55.9% | 61.8% | 60.5% | 91.2% |

* The RBF algorithm selects 3 genes only and achieves a classification accuracy of 85.3%.

Table 18. The performance of neural networks on the top genes selected by different techniques in the *leukemia* data set.

| **No. of Genes Selected** | **Classification Accuracy** | | | | | |
|---|---|---|---|---|---|---|
| | **ACA** | ***t*-value** | ***k*-means** | **SOM** | **Biclustering** | **MRMR** |
| 10 | **97.1%** | 82.4% | 70.6% | 61.8% | 52.9% | **97.1%** |
| 20 | **97.1%** | 82.4% | 64.7% | 61.8% | 58.8% | 94.1% |
| 30 | **97.1%** | 82.4% | 64.7% | 52.9% | 47.1% | 94.1% |
| 40 | **94.1%** | 88.2% | 61.8% | 73.5% | 58.8% | **94.1%** |
| 50 | **97.1%** | 82.4% | 58.8% | 58.8% | 52.9% | **97.1%** |

* The RBF algorithm selects 3 genes only and achieves a classification accuracy of 94.1%.

Table 19. The performance of the nearest neighbor method on the top genes selected by different techniques in the *leukemia* data set.

| No. of Genes Selected | Classification Accuracy | | | | | |
|---|---|---|---|---|---|---|
| | **ACA** | ***t*-value** | ***k*-means** | **SOM** | **Biclustering** | **MRMR** |
| 10 | **91.2%** | 82.4% | 50.0% | 50.0% | 52.9% | 61.8% |
| 20 | **91.2%** | 88.2% | 44.1% | 61.8% | 52.9% | 70.6% |
| 30 | **91.2%** | 88.2% | 44.1% | 67.6% | 58.8% | 67.6% |
| 40 | **91.2%** | 88.2% | 47.1% | 70.6% | 58.8% | 70.6% |
| 50 | **91.2%** | 82.4% | 47.1% | 67.6% | 52.9% | 70.6% |

* The RBF algorithm selects 3 genes only and achieves a classification accuracy of 47.1%.

Table 20. The performance of the naïve Bayes method on the top genes selected by different techniques in the *leukemia* data set.

| No. of Genes Selected | Classification Accuracy | | | | | |
|---|---|---|---|---|---|---|
| | **ACA** | ***t*-value** | ***k*-means** | **SOM** | **Biclustering** | **MRMR** |
| 10 | **82.4%** | 55.9% | 58.8% | 58.8% | 58.8% | 67.6% |
| 20 | **61.8%** | 47.1% | 58.8% | 58.8% | 58.8% | 55.9% |
| 30 | **61.8%** | 38.2% | 58.8% | 58.8% | 58.8% | 50.0% |
| 40 | **61.8%** | 29.4% | 52.9% | 58.8% | 58.8% | 47.1% |
| 50 | **61.8%** | 20.6% | 52.9% | 58.8% | 58.8% | 47.1% |

* The RBF algorithm selects 3 genes only and achieves a classification accuracy of 58.8%.

The experimental results in Tables 17–20 show that ACA is, by and large, superior to the other six attribute clustering and gene selection methods as it selects a better small set of discriminative genes from the *leukemia* data set than the others. As in the *colon-cancer* cases, the classification results obtained using the gene pools selected by ACA are also better than those using all the *leukemia* genes. In all cases, ACA outperforms *t*-value. However, although the *t*-value can also find the good discriminative genes for C5.0 and the nearest neighbor method, yet it fails to find good discriminative genes for the training of neural networks and the naïve Bayes method. The MRMR and RBF algorithms find good discriminative genes for C5.0, neural networks, and the naïve Bayes method but are unable to do so for the nearest neighbor method. ACA outperforms the MRMR and RBF algorithms in all cases except neural networks, in which the three approaches produce comparable classification accuracy. The *k*-means algorithm, SOM, and the biclustering algorithm cannot find the good discriminative genes as shown in the results. Similar to the result found in the *colon-cancer* data set, this result shows that it is able to build a more accurate classifier if a subset of more informative genes based on multiple interdependence selected by ACA are fed into the classifier for training.

It is interesting to note that the performance of C5.0 is able to achieve a 94.1% rate when using the 7 genes selected by ACA and maintain at the same accuracy level even more

genes selected by ACA are used (see Table 17). This again supports that using only the top genes in each cluster found by ACA are good enough for training C5.0.

As in the *colon-cancer* cases, the poor classification performance using the set selected by the *k*-means algorithm, SOM, and the biclustering algorithm (see Tables 16–20) may follow the same argument as in the last section (see Table 8).

Process similar to the *colon-cancer* cases are used to evaluate the performance of the *k*-means algorithm and the biclustering algorithm except the numbers may be different (Tables 17–20). The *k*-means algorithm obtained the best result when the top gene in each cluster is selected and fed to neural networks (70.6% as shown in Table 18), whereas the biclustering algorithm produced the best result when the top gene in each cluster is selected and fed to C5.0 (71.1% as shown in Table 17). Based on their best performance scenarios, the experimental results of using their optimal configuration of both 10 clusters (where 10 happens to be the cluster number determined by ACA as well) yields one of the best results (70.6% for the *k*-means algorithm as shown in Table 18, whereas 71.1% for the biclustering algorithm as shown Table 20). The performance by neural networks on the top genes selected by the *k*-means algorithm and that by C5.0 on the top genes selected by the biclustering algorithm with different number of clusters are given in Tables 21 and 22, respectively. With the same configuration, ACA obtains a classification accuracy of 97.1% (see Table 18) and 94.1% (see Table 17), respectively, far superior to their performance. It is interesting to observe that the number of clusters determined by ACA (10 in this case), if used as a candidate of *k*, both the *k*-means algorithm and the biclustering algorithm yield the best result.

Table 21. The performance of neural networks on the top genes selected by the *k*-means algorithm in the *leukemia* data set.

| No. of Clusters Found | Classification Accuracy |
|---|---|
| 2 | 58.8% |
| 4 | 61.8% |
| 6 | 58.8% |
| 8 | 58.8% |
| 10 | 70.6% |
| 15 | 70.6% |
| 20 | 67.6% |

Table 22. The performance of C5.0 on the top genes selected by the biclustering algorithm in the *leukemia* data set.

| No. of Clusters Found | Classification Accuracy |
|---|---|
| 2 | 58.8% |
| 4 | 58.8% |
| 6 | 55.9% |
| 8 | 58.8% |
| 10 | 71.1% |
| 15 | 41.2% |
| 20 | 44.1% |

Kohonen's SOM determines that there are 54 clusters, far too many for practical reasons. As shown in Tables 17–20, SOM produces the best result when the top 4 genes in each cluster are selected and fed to neural networks (73.5% as shown in Table 18). The classification accuracy of neural networks using the top 4 genes in each of the 54 clusters is 73.5%. It is important to note that ACA obtains a classification accuracy of 97.1% using 10 genes only (see Table 18).

#### 6.3.2.5 Can a Specific Gene(s) Governed a Disease Be Found by ACA?

To answer the question on what more lights could the multiple interdependence results could shed on the nature and the usefulness of the information obtained by ACA, the following experiment is conducted.

We first examined the decision tree built on top of the genes selected by ACA in the *leukemia* data set. We found that the decision tree built by C5.0 uses only gene M27891_at, which is the first gene in Cluster 4 found by ACA (see Table 4), to classify any samples. This gene is also ranked as the second by the *t*-value. The decision tree achieves a classification accuracy of 94.1%. Next, we examined the decision tree built using all the 7,129 genes in the *leukemia* data set. We found that the decision tree built in this way does not use gene M27891_at. It surprises us to notice that the decision tree built on top of all the genes obtains a classification accuracy of 91.2%, which is lower than what it does if using the top gene M27891_at selected by ACA.

Although we cannot comment on the biological impacts of gene M27891_at to leukemia at this moment, the experimental results show that this gene is very useful in the classification of leukemia and the usefulness of this gene cannot be identified if gene selection has not been done properly. As researchers are devoting immense effort to identify genes that govern various diseases, the method we propose may provide a new way

of not only reducing the search dimensionality of gene expressions in analysis, but also singling out potential candidates for the classification and identification of diseases.

# Chapter 7
# Mining Fuzzy Rules in Data Sets and Rule Sets

The problem of mining *association rules* is introduced in [Agrawal, Imielinski, and Swami 1993b] to reveal interesting patterns in the data. The mining of association rules is originally defined for transaction data. This is later extended to also handle relational data containing categorical (discrete-valued) and quantitative (continuous-valued) data [Srikant and Agrawal 1996]. In its most general form, an association rule is defined for the attributes of a database relation, $T$. It is an implication of the form $X \Rightarrow Y$, where $X$ and $Y$ are conjunctions of certain conditions. A condition is either $A_i = a_i$, where $a_i$ is a value in the domain of the attribute $A_i$ if $A_i$ is discrete, or $a_i \in [l_i, u_i]$, where $l_i$ and $u_i$ are bounding values in the domain of the attribute $A_i$ if $A_i$ is continuous. The association rule $X \Rightarrow Y$ holds in $T$ with a certain *support*, which is defined as the percentage of tuples that have the characteristics satisfying $X$ and $Y$, and a certain *confidence*, which is defined as the percentage of tuples that have the characteristics satisfying $Y$ given that they also satisfy $X$. An association relationship is usually considered interesting if its support and confidence values are greater than or equal to some user-specified minimum [Agrawal, Imielinski, and Swami 1993b; Agrawal and Srikant 1994, 1996; Cheung *et al.* 1996a; Han and Fu 1995; Park, Chen, and Yu 1995a, 1995b; Savasere, Omiecinski, and Navathe 1995; Srikant and Agrawal 1995, 1996].

An example of an association rule is:

$$\begin{aligned} &\textit{Marital Status} = \textit{Single} \wedge \textit{Age} \in [35, 45] \wedge \textit{Account Balance} \in [1\,000, 2\,500] \\ &\quad \Rightarrow \textit{Loan Balance} = [10\,000, 15\,000], \end{aligned}$$

which describes a person who is single, aged between 35 and 45, and with an account balance that is between $1,000 and $2,500, as someone who is likely to use a loan that is between $10,000 and $15,000. An association rule defined over market basket data has a special form. The antecedent and the consequent are conjunctions involving Boolean attributes that take on the value of 1. An example of an association rule that is defined over market basket data is:

$$\textit{Pizza} = 1 \wedge \textit{Chicken Wings} = 1 \Rightarrow \textit{Coke} = 1 \wedge \textit{Salad} = 1.$$

This rule states that a customer who buys pizza and chicken wings also buys coke and salad.

Although the existing algorithms for mining association rules (e.g., [Liu, Hsu, and Ma 1998; Srikant and Agrawal 1996]) can be used to identify interesting association relationships in continuous or mixed continuous and discrete valued data, they require the domains of continuous attributes to be discretized into intervals. These intervals are often hard to define. If too much data lies on the boundaries of the intervals, this could result in very different discoveries in the data that could be both misleading and meaningless. In addition to the need for discretization, there is a requirement for users to provide the thresholds for minimum support and confidence, and this also makes the existing techniques (e.g., [Agrawal, Imielinski, and Swami 1993b; Agrawal and Srikant 1994, 1996; Cheung *et al.* 1996a; Han and Fu 1995; Park, Chen, and Yu 1995a, 1995b; Savasere, Omiecinski, and Navathe 1995; Srikant and Agrawal 1995, 1996]) to be difficult to use. If the thresholds are set too high, a user may miss some useful rules; but if the thresholds are set too low, the user may be overwhelmed by too many irrelevant rules [Han and Kamber 2001; Hand, Mannila, and Smyth 2001].

To better represent the underlying association relationships hidden in the data, we develop two new fuzzy algorithms for data mining. They employ *linguistic variables* and *linguistic terms* to represent the revealed regularities and exceptions. This linguistic representation is especially useful when the discovered rules are presented to human experts for examination because of its affinity with the human knowledge representation. Since our interpretation of linguistic terms is based on fuzzy set theory, the rules that are expressed in these terms are referred to hereinafter as *fuzzy association rules* [Au and Chan 1998, 1999, 2001, 2003, 2004; Chan and Au 1997b, 2001, 2002].

An example of a fuzzy association rule is given as follows:

$$\textit{Marital Status} = \textit{Single} \wedge \textit{Age} = \textit{Middle} \wedge \textit{Account Balance} = \textit{Small}$$
$$\Rightarrow \textit{Loan Balance} = \textit{Moderate},$$

where *Single* is a crisp value,

*Middle* is a linguistic term that is represented by the fuzzy set $\int_{30}^{40} \frac{\frac{1}{10}(x-30)}{x} + \int_{40}^{50} \frac{\frac{1}{10}(50-x)}{x}$, and

*Moderate* is a linguistic term that is represented by the fuzzy set

$$\int_{10000}^{20000} \frac{\frac{1}{10000}(x-10000)}{x} + \int_{20000}^{30000} \frac{\frac{1}{10000}(30000-x)}{x}.$$

This rule states that a middle-aged person who is single and has a small balance in his/her bank account is likely to use a loan for a moderate amount. When this rule is compared to the association rule involving discrete intervals, the fuzzy association rule is easier for human users to comprehend. In addition to the linguistic representation, the use of fuzzy set based techniques hides the boundaries of the adjacent intervals of the continuous attributes. This makes our proposed algorithms to be resilient to noises in the data, such as inaccuracies in the physical measurements of real-life entities. Furthermore, the fact that 0.5 is the fuzziest degree of membership of an element in a fuzzy set provides a new means for them to deal with missing values in databases. Using defuzzification techniques, our algorithms allow continuous values to be inferred when fuzzy association rules are applied to as yet unseen records.

To avoid the need for user-specified thresholds, both of the two proposed algorithms utilize an objective interestingness measure, which is defined in terms of a fuzzy support and confidence measure [Au and Chan 1998, 1999, 2001, 2002a, 2002b, 2003, 2004; Au, Chan, and Yao 2003; Chan and Au 1997a, 1997b, 2001; Chan, Au, and Choi 2002], that reflects the actual and the expected degree to which a tuple is characterized by different linguistic terms. Unlike other data mining algorithms (e.g., [Agrawal, Imielinski, and Swami 1993b; Agrawal and Srikant 1994, 1996; Cheung *et al.* 1996a; Han and Fu 1995; Park, Chen, and Yu 1995a, 1995b; Savasere, Omiecinski, and Navathe 1995; Srikant and Agrawal 1995, 1996]), the use of this interestingness measure has the advantage that it does not require any user-specified thresholds.

Using the discovered rules, our proposed algorithms can be used to classify records with unknown class membership. In particular, they are able to predict churn, which is concerned with the loss of subscribers who switch from one carrier to another. To reduce churn rate, a carrier in Malaysia gives us a database of 100,000 subscribers. For such an application, the goal is not only to predict whether or not a subscriber would switch from one carrier to another, it is also important that the likelihood of the subscriber's doing so be predicted. Otherwise, it can be difficult for the carrier to take advantage of the discovery because the carrier does not have enough resources to contact all or a large fraction of the subscribers. Although logit regression and neural networks can determine a probability for a prediction with its likelihood, they do not explicitly express the uncovered patterns in a symbolic, easily understandable form. It is for this reason that the carrier does not consider these approaches as the best for their task concerned as they could not verify and interpret

the uncovered churning patterns.

Unlike existing techniques, our algorithms are able to mine rules representing the churning patterns and to predict whether a subscriber is likely to churn in the near future. The experimental results show that they are able to discover the regularities hidden in the database and to predict the probability that a subscriber churns under different churn rates. In addition, since some attributes in the subscriber database contains significant amount of missing values, the ability of the proposed algorithms to handle missing values effectively is important to their success in churn prediction.

In addition to mining rules from data sets, our proposed algorithms can also mine meta-rules from rule sets. Specifically, they are able to discover 1) regular meta-rules to represent association relationships in common in the rule sets; 2) differential meta-rules to represent distinguishing associations in only a few rule sets; and 3) change meta-rules to represent the regularities governing how rules change over time.

The rest of this chapter is organized as follows. In Section 7.1, we present what fuzzy association rules are and how to use an objective measure to find the interesting associations that are hidden in databases. We then propose two algorithms for mining fuzzy association rules. One is based on a heuristic and the other employs an evolutionary approach. The details of the former algorithm and those of the latter are given in Sections 7.2 and 7.3, respectively. To evaluate the performance of our algorithms, we applied them to several real-life data sets for data mining. The experimental results are provided in Section 7.4. The details of the subscriber database provided by the carrier in Malaysia and the experimental results using this database to test if our proposed algorithms are effective for churn prediction are also given in this same section. Furthermore, we also applied our algorithms to synthetic data sets for meta-mining. The details and the results of the experiments are described in Section 7.5.

## 7.1 Fuzzy Association Rules

In the following subsections, we present 1) the definition of linguistic variables and linguistic terms; 2) how to identify interesting association relationships between linguistic terms; 3) the formation of fuzzy rules to represent the interesting associations and how to represent the uncertainty associated with the rules; and 4) how to predict previously unknown values using the discovered fuzzy rules.

### 7.1.1 Linguistic Variables and Linguistic Terms

Given a database relation, $D$, each tuple, $t$, in $D$ consists of a set of attributes, $\mathcal{A} = \{A_1, \ldots, A_n\}$, where $A_1, \ldots, A_n$ can be continuous or discrete. For any tuple, $t \in D$, $t[A_i]$ denotes the value $a_i$ in $t$ for attribute $A_i \in \mathcal{A}$. Let $\mathcal{L} = \{L_1, \ldots, L_n\}$ be a set of linguistic variables such that $L_i \in \mathcal{L}$ represents $A_i \in \mathcal{A}$.

For any continuous attribute, $A_i \in \mathcal{A}$, let $dom(A_i) = [l_i, u_i] \subseteq \Re$ denote the domain of the attribute. $A_i$ is represented by a linguistic variable, $L_i$, whose value is a linguistic term in $T(L_i) = \{l_{ij} \mid j = 1, \ldots, s_i\}$, where $l_{ij}$ is a linguistic term characterized by a fuzzy set, $F_{ij}$, that is defined on $dom(A_i)$ and whose membership function is $\mu_{F_{ij}}$ so that:

$$\mu_{F_{ij}} : dom(A_i) \rightarrow [0,1].$$

The fuzzy sets $F_{ij}$, $j = 1, \ldots, s_i$, are then represented by:

$$F_{ij} = \begin{cases} \sum_{dom(A_i)} \dfrac{\mu_{F_{ij}}(a_i)}{a_i} & \text{if } A_i \text{ is discrete} \\ \int_{dom(A_i)} \dfrac{\mu_{F_{ij}}(a_i)}{a_i} & \text{if } A_i \text{ is continuous} \end{cases}, \tag{7.1}$$

where $a_i \in dom(A_i)$. The degree of compatibility of $a_i \in dom(A_i)$ with linguistic term $l_{ij}$ is given by $\mu_{F_{ij}}(a_i)$.

For any discrete attribute, $A_i \in \mathcal{A}$, let $dom(A_i) = \{a_{i1}, ..., a_{im_i}\}$ denote the domain of $A_i$. $A_i$ is represented by linguistic variable $L_i$ whose value is a linguistic term in $T(L_i) = \{l_{ij} \mid j = 1, \ldots, m_i\}$, where $l_{ij}$ is a linguistic term characterized by a fuzzy set, $F_{ij}$, so that:

$$F_{ij} = \sum_{dom(A_i)} \frac{\mu_{F_{ij}}(a_i)}{a_i}, \tag{7.2}$$

where $a_i \in dom(A_i)$. The degree of compatibility of $a_i \in dom(A_i)$ with linguistic term $l_{ij}$ is given by $\mu_{F_{ij}}(a_i)$.

In addition to handling discrete and continuous attributes in a uniform fashion, the use

of linguistic terms to represent discrete attributes also allows the fuzzy nature of some real-world entities to be easily captured. For example, it may be difficult to distinguish the color orange from the color red in some situations. It is for this reason that an object, which is orange in color, can be perceived as red in color to certain extent. Such kind of fuzziness in attribute *Color* can be represented by linguistic terms *Red* and *Orange*. Based on these linguistic terms, the color of an object can be compatible with the term *Red* to a degree of 0.7 and with the term *Orange* to a degree of 0.3.

Interested readers are referred to [Mendel 1995] and [Yen 1999] for the details of the linguistic variables, linguistic terms, fuzzy sets, and membership functions.

Using the above technique, the original attributes, $\mathcal{A}$, are represented by a set of linguistic variables, $\mathcal{L} = \{L_i \mid i = 1, \ldots, n\}$. These linguistic variables are associated with a set of linguistic terms, $\mathcal{l} = \{l_{ij} \mid i = 1, \ldots, n, j = 1, \ldots, s_i\}$. These linguistic terms are, in turn, characterized by a set of fuzzy sets, $\mathcal{F} = \{F_{ij} \mid i = 1, \ldots, n, j = 1, \ldots, s_i\}$. Given a tuple, $t \in D$, and a linguistic term, $l_{ij} \in \mathcal{l}$, which is characterized by a fuzzy set, $F_{ij} \in \mathcal{F}$, the degree of membership of the values in $t$ with respect to $F_{ij}$ is given by $\mu_{F_{ij}}(t[A_i])$. The degree to which $t$ is characterized by $l_{ij}$, $\lambda_{l_{ij}}(t)$, is defined as follows:

$$\lambda_{l_{ij}}(t) = \mu_{F_{ij}}(t[A_i]). \tag{7.3}$$

If $\lambda_{l_{ij}}(t) = 1$, $t$ is completely characterized by the linguistic term $l_{ij}$. If $\lambda_{l_{ij}}(t) = 0$, $t$ is undoubtedly not characterized by the linguistic term $l_{ij}$. If $0 < \lambda_{l_{ij}}(t) < 1$, $t$ is partially characterized by the linguistic term $l_{ij}$. In the case where $t[A_i]$ is unknown, $\lambda_{l_{ij}}(t) = 0.5$, which indicates that there is no information available concerning whether $t$ is or is not characterized by the linguistic term $l_{ij}$.

It is important to note that $t$ can also be characterized by more than one linguistic term. Let $\varphi$ be a subset of integers so that $\varphi = \{i_1, \ldots, i_h\}$, where $\varphi \subseteq \{1, \ldots, n\}$ and $|\varphi| = h \geq 1$. We also suppose that $\mathcal{A}_\varphi$ is a subset of $\mathcal{A}$ so that $\mathcal{A}_\varphi = \{A_i \mid i \in \varphi\}$. Given any $\mathcal{A}_\varphi$, it is associated with a set of linguistic terms, $T(L_\varphi) = \{l_{\varphi j} \mid j = 1, \ldots, s_\varphi = \prod_{i \in \varphi} s_i\}$, where $l_{\varphi j}$ is represented by a fuzzy set, $F_{\varphi j}$, so that $F_{\varphi j} = F_{i_1 j_1} \cap \ldots \cap F_{i_h j_h}$, $i_k \in \varphi$, $j_k \in s_{i_k}$. The degree to which $t$ is characterized by the term $l_{\varphi j}$, $\lambda_{l_{\varphi j}}(t)$, is defined as follows:

$$\lambda_{l_{\varphi j}}(t) = \min(\mu_{L_{i_1 j_1}}(t[A_{i_1}]), ..., \mu_{L_{i_h j_h}}(t[A_{i_h}])) . \quad (7.4)$$

Based on the linguistic variables and linguistic terms, we can apply our proposed algorithms to discover the fuzzy association rules, which are represented in a manner that is natural for human users to understand.

#### 7.1.1.1 An Illustrative Example

In this section, we illustrate how a relation in a relational database can be transformed to a fuzzy relation based on linguistic variables and linguistic terms. Let us consider a sample relation shown in Fig. 16 ("U" stands for unmarried and "M" stands for married).

| ***Age*** | ***Marital Status*** | ***Salary*** |
|---|---|---|
| 23 | U | 40,000 |
| 29 | M | 43,000 |
| 33 | M | 55,000 |
| 35 | U | 64,000 |
| 55 | M | 62,000 |

Fig. 16. A sample relation.

Let us further suppose that the *Marital Status* attribute, which is a discrete attribute, is represented by two linguistic terms defined as:

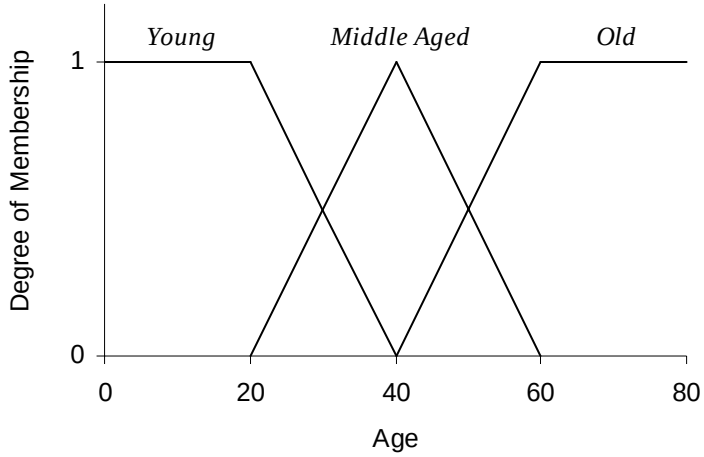
$$Unmarried = \frac{1}{\mathrm{U}}$$

and

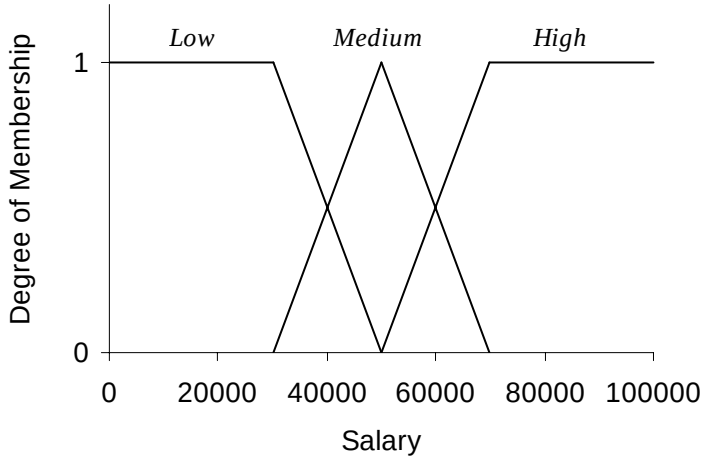
$$Married = \frac{1}{\mathrm{M}} .$$

For the remaining two continuous attributes, *Age* and *Salary*, they are represented by the linguistic terms given in Fig. 17.

Based on these linguistic terms, the sample relation is transformed to a fuzzy relation shown in Fig. 18. Instead of mining interesting rules from the original relation, we perform data mining in the resulting fuzzy relation.




(a) The *Age* attribute.




(b) The *Salary* attribute.

Fig. 17. The definitions of linguistic terms.

| ***Age*** | ***Marital Status*** | ***Salary*** |
|---|---|---|
| {(*Young*, 0.85), (*Middle Aged*, 0.15)} | {(*Unmarried*, 1)} | {(*Low*, 0.5), (*Medium*, 0.5)} |
| {(*Young*, 0.55), (*Middle Aged*, 0.45)} | {(*Married*, 1)} | {(*Low*, 0.35), (*Medium*, 0.65)} |
| {(*Young*, 0.35), (*Middle Aged*, 0.65) | {(*Married*, 1)} | {(*Medium*, 0.75), (*High*, 0.25)} |
| {(*Young*, 0.25), (*Middle Aged*, 0.75)} | {(*Unmarried*, 1)} | {(*Medium*, 0.3), (*High*, 0.7)} |
| {(*Middle Aged*, 0.25), (*Old*, 0.75)} | {(*Married*, 1)} | {(*Medium*, 0.4), (*High*, 0.6)} |

Fig. 18. The resulting fuzzy relation.

### 7.1.2 Identification of Interesting Associations between Linguistic Terms

The *fuzzy support* of a linguistic term, $l_{\varphi k}$, is represented by $fsup(l_{\varphi k})$, and it is defined as follows:

$$fsup(l_{\varphi k}) = \frac{\sum_{t \in D} \lambda_{l_{\varphi k}}(t)}{\sum_{t \in D} \sum_{j=1}^{s_\varphi} \lambda_{l_{\varphi j}}(t)}. \tag{7.5}$$

The fuzzy support of the linguistic term $l_{\varphi k}$, $fsup(l_{\varphi k})$, can be considered as being the probability that a tuple is characterized by $l_{\varphi k}$.

In the rest of this chapter, the association between a linguistic term, $l_{\varphi k}$, and another linguistic term, $l_{pq}$, is expressed as $l_{\varphi k} \rightarrow l_{pq}$. The fuzzy support of the association $l_{\varphi k} \rightarrow l_{pq}$, $fsup(l_{\varphi k} \rightarrow l_{pq})$, is given by:

$$fsup(l_{\varphi k} \rightarrow l_{pq}) = \frac{\sum_{t \in D} \min(\lambda_{l_{\varphi k}}(t), \lambda_{l_{pq}}(t))}{\sum_{t \in D} \sum_{j=1}^{s_\varphi} \sum_{u=1}^{s_p} \min(\lambda_{l_{\varphi j}}(t), \lambda_{l_{pu}}(t))}. \tag{7.6}$$

In fact, other *t*-norms (e.g., the multiplication operation) can also be used in the calculation of the fuzzy support. We use the minimum operation here because it is one of the most popular *t*-norms used in the literature (see, e.g., [Yen and Langari 1999]).

The *fuzzy confidence* of the association $l_{\varphi k} \rightarrow l_{pq}$ is represented by $fconf(l_{\varphi k} \rightarrow l_{pq})$ and this is calculated by:

$$fconf(l_{\varphi k} \rightarrow l_{pq}) = \frac{fsup(l_{\varphi k} \rightarrow l_{pq})}{fsup(l_{\varphi k})}. \tag{7.7}$$

Intuitively, the fuzzy support of $l_{\varphi k} \rightarrow l_{pq}$, $fsup(l_{\varphi k} \rightarrow l_{pq})$, can be considered as being the probability that a tuple is characterized by $l_{\varphi k}$ and $l_{pq}$, whereas the fuzzy confidence of $l_{\varphi k} \rightarrow l_{pq}$, $fconf(l_{\varphi k} \rightarrow l_{pq})$, can be considered as being the probability that a tuple is characterized by $l_{pq}$ given that it is also characterized by $l_{\varphi k}$.

To decide whether an association, $l_{\varphi k} \rightarrow l_{pq}$, is interesting, we determine whether the difference between $fconf(l_{\varphi k} \rightarrow l_{pq})$ and $fsup(l_{pq})$ is significant. The significance of the difference can be objectively evaluated using the *adjusted residual* [Chan and Wong 1990, 1991]. This is defined in terms of fuzzy confidence and support measures [Au and Chan 1998, 1999, 2001, 2002a, 2002b, 2003, 2004; Au, Chan, and Yao 2003; Chan and Au 1997a, 1997b, 2001; Chan, Au, and Choi 2002] that reflect the differences in the actual and the expected degree to which a tuple is characterized by different linguistic terms. The adjusted residual, $d(l_{\varphi k} \rightarrow l_{pq})$, is defined as [Chan and Wong 1990, 1991]:

$$d(l_{\varphi k} \rightarrow l_{pq}) = \frac{z(l_{\varphi k} \rightarrow l_{pq})}{\sqrt{\gamma(l_{\varphi k} \rightarrow l_{pq})}}, \tag{7.8}$$

where $z(l_{\varphi k} \rightarrow l_{pq})$ is the *standardized residual* and is defined as [Chan and Wong 1990, 1991]:

$$z(l_{\varphi k} \rightarrow l_{pq}) = \frac{fsup(l_{\varphi k} \rightarrow l_{pq}) - e(l_{\varphi k} \rightarrow l_{pq})}{\sqrt{e(l_{\varphi k} \rightarrow l_{pq})}}, \tag{7.9}$$

$e(l_{\varphi k} \rightarrow l_{pq})$ is the expected degree to which a tuple is characterized by $l_{\varphi k}$ and $l_{pq}$ and is calculated by:

$$e(l_{\varphi k} \rightarrow l_{pq}) = fsup(l_{\varphi k}) \times fsup(l_{pq}) \times \sum_{t \in D} \sum_{j=1}^{s_\varphi} \sum_{u=1}^{s_p} \min(\lambda_{l_{\varphi j}}(t), \lambda_{l_{pu}}(t)), \tag{7.10}$$

and $\gamma(l_{\varphi k} \rightarrow l_{pq})$ is the *maximum likelihood estimate* [Chan and Wong 1990, 1991] of the variance of $z(l_{\varphi k} \rightarrow l_{pq})$ and is given by:

$$\gamma(l_{\varphi k} \rightarrow l_{pq}) = (1 - fsup(l_{\varphi k}))(1 - fsup(l_{pq})). \tag{7.11}$$

The measure defined by (7.8) can be considered as being an objective interestingness measure because it does not depend on a user's subjective input. Since $d(l_{\varphi k} \rightarrow l_{pq})$ has a normal distribution [Agresti 1990], if $d(l_{\varphi k} \rightarrow l_{pq}) > 1.96$, then the presence of $l_{\varphi k}$ implies the presence of $l_{pq}$. In other words, whenever $l_{\varphi k}$ is found in a tuple, the probability that $l_{pq}$ is also found in the same tuple is expected to be significantly higher than when $l_{\varphi k}$ is not found.

### 7.1.3 Formation of Fuzzy Association Rules

In the context of rule mining, the number of conditions in the antecedent of a rule is often referred to as its *order* [Smyth and Goodman 1992; Wong and Wang 1997, 2003]. A first-order fuzzy association rule can be defined as a rule involving one linguistic term in its antecedent. A second-order fuzzy association rule can be defined as a rule involving two linguistic terms in its antecedent. A third-order fuzzy association rule can be defined as a rule involving three linguistic terms in its antecedent, and so on for other higher orders.

Given that $l_{\varphi k} \rightarrow l_{pq}$ is interesting, we can form the following fuzzy association rule:

$$l_{\varphi k} \Rightarrow l_{pq}\ [w(l_{\varphi k} \Rightarrow l_{pq})],$$

where $w(l_{\varphi k} \Rightarrow l_{pq})$ is the *weight of evidence* measure, which is a confidence measure that represents the uncertainty associated with $l_{\varphi k} \Rightarrow l_{pq}$. This measure is defined as follows [Chan and Wong 1990, 1991].

Since the relationship between $l_{\varphi k}$ and $l_{pq}$ is interesting, there is some evidence for a record to be characterized by $l_{pq}$ given it has $l_{\varphi k}$. The weight of evidence measure is defined in terms of an information-theoretic measure known as *mutual information*. Mutual information measures the change of uncertainty about the presence of $l_{pq}$ in a tuple given that it has $l_{\varphi k}$. It is defined as:

$$I(l_{pq} : l_{\varphi k}) = \log \frac{fconf(l_{\varphi k} \rightarrow l_{pq})}{fsup(l_{pq})}. \tag{7.12}$$

Based on mutual information, the weight of evidence measure is defined as [Chan and Wong 1990, 1991]:

$$\begin{aligned} w(l_{\varphi k} \Rightarrow l_{pq}) &= I(l_{pq} : l_{\varphi k}) - I(\bigcup_{j \neq q}(l_{pj} : l_{\varphi k})) \\ &= \log \frac{fsup(l_{\varphi k} \rightarrow l_{pq}) \big/ fsup(l_{pq})}{fsup(\bigcup_{j \neq q} l_{\varphi k} \rightarrow l_{pj}) \big/ fsup(\bigcup_{j \neq q} l_{pj})}. \end{aligned} \tag{7.13}$$

$w(l_{\varphi k} \Rightarrow l_{pq})$ can be interpreted intuitively as a measure of the difference in the gain in information when a tuple that is characterized by $l_{\varphi k}$ is also characterized by $l_{pq}$ as opposed to being characterized by other linguistic terms.

Since $l_{\varphi k}$ is defined by a set of linguistic terms, $l_{i_1 j_1}, ..., l_{i_h j_h} \in \ell$, we have a high-order fuzzy association rule:

$$L_{i_1} = l_{i_1 j_1} \wedge ... \wedge L_{i_h} = l_{i_h j_h} \Rightarrow L_p = l_{pq}\ [w(l_{\varphi k} \Rightarrow l_{pq})],$$

where $i_1, \ldots, i_h \in \varphi$.

In the case that a class label is given, our algorithms can be modified in such a way that they discover only those rules whose consequents are concerned with only the class label. The generation of the rules that are not useful for classification can therefore be avoided.

### 7.1.4 Predicting Previously Unknown Values Using Fuzzy Association Rules

Using the discovered fuzzy association rules, we are able to predict the values of some of the characteristics of previously unseen records. The results can be continuous or discrete, depending on the nature of the attributes whose values are to be predicted. Unlike other classification techniques, which classify records into distinct classes, ours allows continuous values to be inferred from fuzzy association rules.

Given a tuple, $t \in dom(A_1) \times \ldots \times dom(A_p) \times \ldots \times dom(A_n)$, let $t$ be characterized by $n$ attribute values, $\alpha_1, \ldots, \alpha_p, \ldots, \alpha_n$, where $\alpha_p$ is the value to be predicted. Let $l_p$ be a linguistic term with a domain of $T(L_p)$. The value of $\alpha_p$ is determined according to $l_p$. To predict the correct value of $\alpha_p$, we search the discovered rules. If some attribute value, say $\alpha_j$, $j \neq p$, of $t$ is characterized by the linguistic term in the antecedent of a rule that implies $l_{pq}$, then it can be considered providing some confidence that the value of $l_p$ should be assigned to $l_{pq}$. By repeating this procedure, that is, by matching each attribute value of $t$ against the rules, we can determine the value of $l_p$ by computing the total confidence measure.

Each of the attributes of $t$ may or may not provide a contribution to the total confidence measure, and those that do may support the assignment of different values. Therefore, the different contributions to the total confidence measure are measured quantitatively and then combined for comparison in order to find the most suitable value of $l_p$. For any combination of the attribute values, $\alpha_\varphi$, $p \notin \varphi$, of $t$, it is characterized by a linguistic term, $l_{\varphi k}$, to a degree of compatibility, $\lambda_{l_{\varphi k}}(t)$, for each $k \in \{1, \ldots, s_\varphi\}$. Given the rules that imply the assignment of $l_{pq}$, $l_{\varphi k} \Rightarrow l_{pq}\ [w(l_{\varphi k} \Rightarrow l_{pq})]$, for all $k \in \zeta \subseteq \{1, \ldots, s_\varphi\}$, the confidence provided by $\alpha_\varphi$ for such an assignment is given by:

$$w_{l_{pq}\alpha_{\varphi}} = \sum_{k \in \zeta} w(l_{\varphi k} \Rightarrow l_{pq}) \times \lambda_{l_{\varphi k}}(t) . \tag{7.14}$$

Suppose that, among the $n - 1$ attribute values excluding $\alpha_p$, only some combinations of them, $\alpha_{[1]}$, …, $\alpha_{[j]}$, …, $\alpha_{[\beta]}$, where $\alpha_{[j]} = \{\alpha_i \mid i \in \{1, \ldots, n\} - \{p\}\}$, are found matching one or more rules. Then, the total confidence measure for assigning the value of $l_p$ to $l_{pq}$ is given by:

$$w_q = \sum_{j=1}^{\beta} w_{l_{pq}\alpha_{[j]}} . \tag{7.15}$$

In the case that $A_p$ is discrete, $l_p$ is assigned to $l_{pc}$ if:

$$w_c > w_g, g = 1, \ldots, s'_p \text{ and } g \neq c,$$

where $s'_p$ ($\leq s_p$) denotes the number of linguistic terms that are implied by the rules, and $\alpha_p$ is, therefore, assigned to $a_{pc} \in dom(A_p)$.

If $A_p$ is continuous, a new method is used to assign an appropriate value to $\alpha_p$. Given the linguistic terms, $l_{p1}, ..., l_{ps_p}$, and their total confidence measures, $w_{p1}, ..., w_{ps_p}$, let $\mu'_{F_{pu}}(a_p)$ be the weighted degree of membership of $a_p \in dom(A_p)$ to the fuzzy set $F_{pu}$, $u \in \{1, \ldots, s_p\}$. The value of $\mu'_{F_{pu}}(a_p)$ is given by:

$$\mu'_{F_{pu}}(a_p) = w_{pu} \cdot \mu_{F_{pu}}(a_p) , \tag{7.16}$$

where $a_p \in dom(A_p)$ and $u = 1, \ldots, s_p$. The predicted value, $\alpha$, is then defined as:

$$\alpha = \frac{\int_{dom(A_p)} \mu'_{F_{p1} \cup \ldots \cup F_{ps_p}}(a_p) \cdot a_p \, da_p}{\int_{dom(A_p)} \mu'_{F_{p1} \cup \ldots \cup F_{ps_p}}(a_p) \, da_p} , \tag{7.17}$$

where $\mu'_{X \cup Y}(a) = \max(\mu'_X(a), \mu'_Y(a))$ for any fuzzy sets $X$ and $Y$. This prediction, $\alpha$, provides an appropriate value for $\alpha_p$.

## 7.2 The FARM Algorithm

In this section, we propose a new algorithm for mining fuzzy association rule based on a heuristic. It is known as FARM (Fuzzy Association Rule Mining) in the rest of this thesis.

To discover the high-order fuzzy association rules, FARM makes use of a heuristic in which the association between $l_{\varphi' k}$, where $\varphi' = \varphi_1 \cup \varphi_2$, and $l_{pq}$ is considered being more likely to be interesting if the association between $l_{\varphi_1 k}$ and $l_{pq}$ and the association between $l_{\varphi_2 k}$ and $l_{pq}$ are interesting. Based on such a heuristic, FARM evaluates the interestingness of only the associations between different combinations of conditions in lower-order association rules. This approach can effectively prevent an exhaustive search for the interesting associations involving all combinations of the linguistic terms.

FARM starts the data mining process by finding a set of first-order fuzzy association rules using the objective interestingness measure introduced in Section 7.1.2. After these rules are discovered, they are stored in rule set $R_1$. The rules in $R_1$ are then used to generate second-order rules, which are, in turn, stored in $R_2$. The rules in $R_2$ are then used to generate third-order rules, which are stored in $R_3$, and so on for fourth and higher orders. FARM iterates until no higher-order association rule is found. The details of the algorithm are given in Fig. 19.

$R_1 \leftarrow \{l_{ik} \Rightarrow l_{pq}\ [w(l_{ik} \Rightarrow l_{pq})] \mid i \neq p$ and $d(l_{ik} \rightarrow l_{pq}) > 1.96\}$;
$h \leftarrow 2$;
**while** $R_{h-1} \neq \varnothing$ **do**
**begin**
  $C \leftarrow$ {each linguistic term in the antecedent of $r \mid r \in R_{h-1}$};
  **forall** $l_{\varphi k}$ comprising $h$ linguistic terms in $C$ **do**
  **begin**
    **forall** $l_{pq}$, $q = 1, \ldots, s_p$, **do**
    **begin**
      **if** $d(l_{\varphi k} \rightarrow l_{pq}) > 1.96$ **then**
        $R_h \leftarrow R_h \cup \{l_{\varphi k} \Rightarrow l_{pq}\ [w(l_{\varphi k} \Rightarrow l_{pq})]\}$;
    **end**
  **end**
  $h \leftarrow h + 1$;
**end**
$Rules = \bigcup_h R_h$ ;

Fig. 19. The FARM algorithm.

FARM employs the objective interestingness measure described in Section 7.2.2 to determine whether the association relationship $l_{\varphi k} \rightarrow l_{pq}$ is interesting. If $l_{\varphi k} \rightarrow l_{pq}$ is

identified as being interesting, then it generates a rule, $l_{\phi k} \Rightarrow l_{pq}$, whose uncertainty is represented by the confidence measure that is defined in Section 7.2.3. All generated rules are stored in a rule set, which is used later for inference or for human users to examine.

## 7.3 The EFARM Algorithm

FARM will perform slowly when there are enormous attributes. To perform search more effectively in a huge rule set space, we propose to use an evolutionary algorithm, called EFARM (Evolutionary FARM), in this section. Although EFARM performs slower than FARM when there are a moderate number of attributes, our experimental results on several data sets show that the former algorithm achieves more accurate classification results than the latter (Section 7.5.1).

EFARM discovers rules by an iterative process. It begins with the generation of a set of first-order rules using the objective interestingness measure given in Section 7.2.2. Based on these rules, it then discovers a set of second-order rules in the next iteration and based on the second-order rules, it discovers third-order rules, etc. In other words, if we refer to the initial set of first-order rules as $R_1$, the rules in $R_1$ are then used to generate a set of second-order rules, $R_2$. $R_2$ is then used to generate a set of third-order rules, $R_3$, and so on for fourth and higher order rules. In general, at the $(h - 1)$-th iteration, EFARM begins an evolutionary learning process by generating an initial population of individuals (each represents a set of $h$-th order rules) by randomly combining the rules in $R_{h-1}$ to form a set of rules of order $h$. Once started, the iterative learning process goes on uninterruptedly until no more interesting rules in the current population can be identified. The EFARM algorithm is given in Fig. 20.

The *decode* function in Fig. 20 is to extract all the interesting rules encoded in a chromosome and store them in $R_h$. If an allele in the chromosome is found interesting based on the objective measure defined in Section 7.1.2, the *decode* function will extract the rules it encodes. The rule set returned by the *decode* function therefore contains interesting rules only. When none of the rules encoded in the individual is found interesting, the *decode* function will return a null set and hence $R_h$ will become a null set.

$R_1 \leftarrow \{l_{ik} \Rightarrow l_{pq}\ [w(l_{ik} \Rightarrow l_{pq})] \mid i \neq p$ and $d(l_{ik} \rightarrow l_{pq}) > 1.96\}$;
$h \leftarrow 2$;
**while** $R_{h-1} \neq \varnothing$ **do**
**begin**
    $t \leftarrow 0$;
    *population*[$t$] $\leftarrow$ *initialize*($R_{h-1}$);
    *fitness*(*population*[$t$]);
    **while not** *terminate*(*population*[$t$]) **do**
    **begin**
        $t \leftarrow t + 1$;
        *population*[$t$] $\leftarrow$ *reproduce*(*population*[$t - 1$]);
        *fitness*(*population*[$t$]);
    **end**
    $R_h \leftarrow$ *decode*(the fittest individual in *population*[$t$]);
    $h \leftarrow h + 1$;
**end**
$Rules = \bigcup_h R_h$ ;

Fig. 20. The EFARM algorithm.

### 7.3.1 Encoding Rules in the Chromosomes

For the evolutionary process, EFARM encodes a complete set of rules in a single chromosome in such a way that each gene encodes a single rule. Specifically, given the following $h$-th order rule, for example:

$$L_1 = l_{1k_1} \wedge ... \wedge L_h = l_{hk_h} \Rightarrow L_p = l_{pq}\ [w(l_{\varphi k} \Rightarrow l_{pq})],$$

where $w(l_{\varphi k} \Rightarrow l_{pq})$, given by Equation (7.13), is an uncertainty measure associated with it, this rule is encoded in EFARM by the allele given in Fig. 21.


Fig. 21. An allele representing an $h$-th order rule.

It should be noted that the consequent and the uncertainty measure are not encoded. This is because the consequent is not, and in fact, should not be determined by chance. In EFARM, both the consequent and the uncertainty measure are determined when the fitness of a chromosome is computed. Given this representation scheme, the number of genes in the chromosome is, therefore, the same as the number of rules in the rule set.

### 7.3.2 Generating First-Order Rules

EFARM begins the evolutionary process by the generation of a set of first-order rules. When compared to randomly generated initial population, it has been shown that heuristically-generated initial populations can improve convergence speed and find better solutions [Hill 1999; Ishibuchi and Nakashima 1999; Julstrom 1994; Yang and Nygard 1993]. Based on these findings, EFARM first discovers a set of first-order rules and places it in the initial population. Furthermore, the initial first-order rules are generated very rapidly. The time it takes to generate the initial population that contains the first-order rules is negligible when compared to the time it takes for the best set of rules to be evolved.

By using the interestingness measure given by Equation (7.8) and the weight of evidence measure given by Equation (7.13), a set of interesting first-order rules can be discovered. Once these rules are discovered, EFARM will begin an iterative process of initialization of population, evaluation of fitness of individuals, selection, reproduction, and termination, etc., so as to discover higher order rules.

### 7.3.3 Initialization of Populations

Since a good initial population may improve the speed of the evolutionary process and make it easier for an optimal solution to be found, EFARM does not generate its initial populations completely randomly. Instead, it makes use of a heuristic in which the association between $l_{ij} \wedge l_{ks}$ and $l_{pq}$ is more likely to be interesting if the association between $l_{ij}$ and $l_{pq}$ and the association between $l_{ks}$ and $l_{pq}$ are interesting. Based on this heuristic, EFARM generates different sets of $h$-th order rules by randomly combining the $(h-1)$-th order rules discovered in the previous iteration. The details of the initialization process are given in the *initialize* function in Fig. 22.

The *initialize* function takes as argument, $R_{h-1}$. The $chrom_i.allele_j$ in Fig. 22 denotes the $j$-th allele of the $i$-th chromosome. The $rand_h(C)$ function returns an $h$-th order allele constructed by randomly combining $h$ elements in $C$. For our experiments, *popsize* was set to 30 and the number of alleles in each chromosome was set to $nalleles = |R_{l-1}|$, where $|R_{l-1}|$ denotes the number of rules in $R_{h-1}$. We set $nalleles = |R_{h-1}|$ because each allele represents the antecedent of a rule and the chromosome is used to encode $R_{h-1}$.

```
population initialize(R_{h-1})
begin
    C ← {all conjuncts in the antecedent of all r ∈ R_{h-1}};
    i ← 1;
    while i ≤ popsize do
    begin
        j ← 1;
        while j ≤ nalleles do
        begin
            chrom_i.allel_j ← rand_h(C);
            j ← j + 1;
        end
        i ← i + 1;
    end
    return ⋃_i chrom_i ;
end
```

Fig. 22. The *initialize* function.

### 7.3.4 The Genetic Operators

The genetic operators used by EFARM are implemented in the *reproduce* function shown in Fig. 23. The *select*(*population*[$t$ – 1]) function uses the *roulette wheel selection* scheme [Fogel 1995; Goldberg 1989; Michalewicz 1996] to select two different chromosomes, $chrom_1$ and $chrom_2$, with respect to their fitness values from the current population, i.e., *population*[$t$ – 1]. These two chromosomes are then passed as arguments to the *crossover* function.

```
population reproduce(population[t – 1])
begin
    chrom_1 ← select(population[t – 1]);
    chrom_2 ← select(population[t – 1]);
    nchrom_1, nchrom_2 ← crossover(chrom_1, chrom_2);
    mutation(nchrom_1);
    mutation(nchrom_2);
    population ← steady-state(population[t – 1], nchrom_1, nchrom_2);
    return population;
end
```

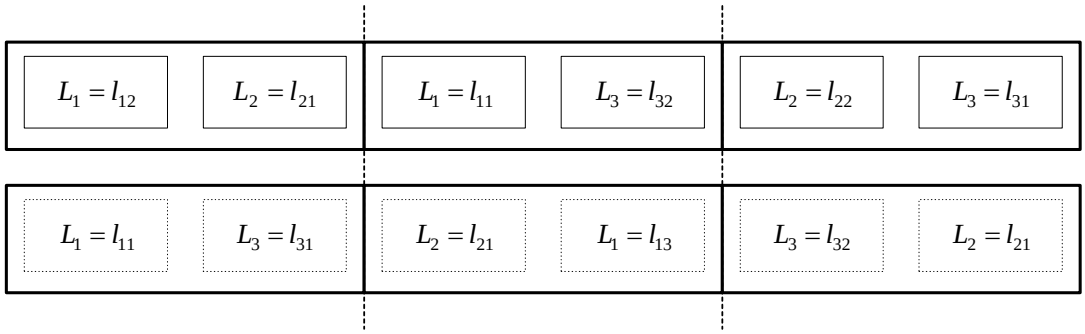
Fig. 23. The *reproduce* function.

The *crossover*($chrom_1$, $chrom_2$) function uses the *two-point crossover* operator because it allows the combination of schemata, which is not possible with the classical, *one-point crossover* [Michalewicz 1996]. EFARM uses two different strategies in choosing the crossover points, namely, *crossover*-1 and *crossover*-2. The *crossover*-1 operator allows the crossover points to occur between two rules only, whereas the *crossover*-2 operator allows the crossover points to occur within one rule only. An example of the *crossover*-1 operator

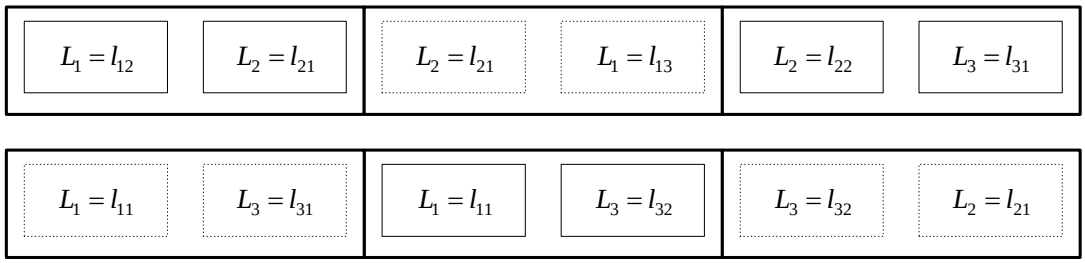
and that of the *crossover*-2 operator are graphically depicted in Fig. 24 and Fig. 25, respectively.

In EFARM, the crossover probability for the *crossover*-1 operator and that for the *crossover*-2 operator are denoted as $p_1$ and $p_2$, respectively. For our experimentation, four different setups are used and they are summarized in Table 23.

The first three setups, EFARM-1, EFARM-2, and EFARM-3, use constant values of $p_1$ and $p_2$, whereas the last setup, EFARM-4, uses adaptive values of $p_1$ and $p_2$. In EFARM-4, $p_1$ is increased by 0.05 and $p_2$ is decreased by 0.05 whenever the termination criteria specified in Section 7.4.6 are satisfied. The evolutionary process ends when $p_1$ and $p_2$ reach 0.75 and 0.25, respectively, and the termination criteria are satisfied. The performance of EFARM under different setups will further be discussed in Section 7.6.2.

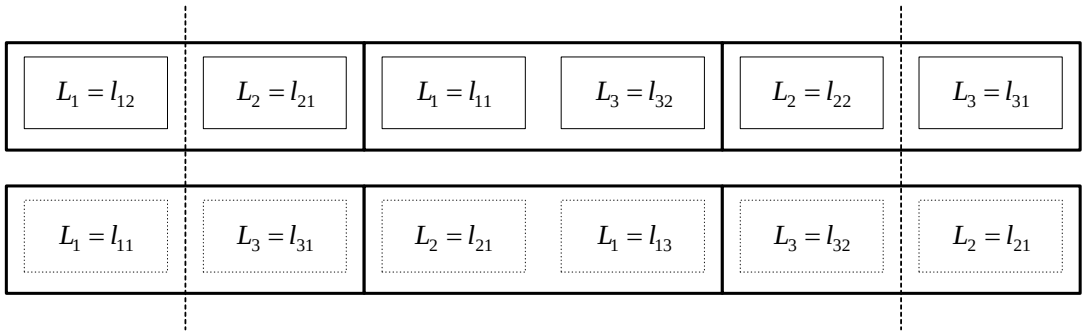



(a) Before crossover.

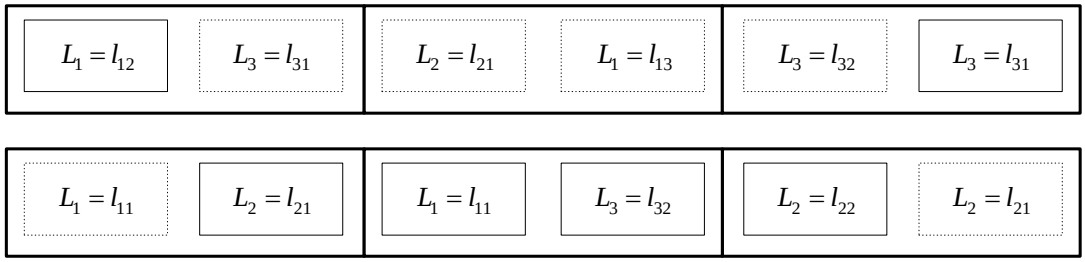



(b) After crossover.

Fig. 24. An example of the *crossover*-1 operator (the thick borders indicate the rule boundaries).




(a) Before crossover.




(b) After crossover.

Fig. 25. An example of the *crossover*-2 operator (the thick borders indicate the rule boundaries).

Table 23. Different setups of crossover probabilities $p_1$ and $p_2$.

| | **Beginning of evolution** | | **End of evolution** | |
|---|---|---|---|---|
| | $p_1$ | $p_2$ | $p_1$ | $p_2$ |
| **EFARM-1** | 0.5 | 0.5 | 0.5 | 0.5 |
| **EFARM-2** | 0.75 | 0.25 | 0.75 | 0.25 |
| **EFARM-3** | 0.25 | 0.75 | 0.25 | 0.75 |
| **EFARM-4** | 0.25 | 0.75 | 0.75 | 0.25 |

The *mutation*(*nchrom*$_1$) function, which is different from the traditional mutation operator [Fogel 1995; Goldberg 1989; Michalewicz 1996], takes a chromosome as argument. Its details are given in Fig. 26. The *random* function returns a real number between 0 and 1 and *pmutation* contains the mutation rate and is a constant. The *random*(1, *h*) function returns an integer between 1 and *h*. The *nchrom.allele*$_j$*.rule*$_k$ denotes the *k*-th rule in the *j*-th allele of chromosome *nchrom*. The *hill-climb*(*C*) function replaces the *k*-th rule with each element in *C* and evaluates the chromosome's fitness value. It returns the one producing the greatest fitness. Instead of replacing a rule with an element in *C* randomly, the use of the *hill-climb* function allows EFARM to search for improvements even when premature convergence occurs [Fogel 1995].

The *steady-state*(*population*[*t* – 1], *nchrom*$_1$, *nchrom*$_2$) function in *reproduce* produces

a new population, *population*[*t*], by removing the two least-fit chromosomes in *population*[*t* – 1] and replacing them with $nchrom_1$ and $nchrom_2$ while keeping the rest of the other chromosomes intact.

```
mutation(nchrom)
begin
    C ← {all conjuncts in the antecedent of all r ∈ R_{h-1}};
    j ← 1;
    while j ≤ nalleles do
    begin
        if random < pmutation then
        begin
            k = random(1, h);
            nchrom.allele_j.rule_k ← hill-climb(C);
        end
        j ← j + 1;
    end
end
```

Fig. 26. The *mutation* function.

### 7.3.5 Selection and the Fitness Function

To determine the fitness of a chromosome that encodes a set of *h*-th order rules, EFARM uses a performance measure defined in terms of the probability that the value of an attribute of a tuple can be correctly predicted based on the rules in $R = R_1 \cup \cdots \cup R_{h-1} \cup$ {rules encoded in the chromosome being evaluated}. The use of this fitness measure is to allow EFARM to maximize the number of records that it can correctly predict. How exactly such fitness value can be determined is given in the following.

An attribute, say, $A_i$ of a tuple *o* characterized by $A_1 = v_1, \ldots, A_i = v_i, \ldots, A_n = v_n$ is randomly selected and the value $v_i$ deleted from *o*. The rules contained in *R* are then used to see if the value of $A_i$ can be correctly predicted based on $v_1, \ldots, v_{i-1}, v_{i+1}, \ldots, v_n$. Assume that a rule which predicts $A_i = a_{ip} \in dom(A_i)$ is matched, this rule can be considered providing some evidence for or against $A_i$ to have the value $a_{ip}$ and the strength of the evidence is given by the weight of evidence associated with it. By matching $v_1, \ldots, v_{i-1}, v_{i+1}, \ldots, v_n$ against the rules in *R*, the value that $A_i$ should take on can be determined based on a total weight of evidence measure which we describe in Section 7.2.4.

### 7.3.6 Criteria for Termination

The *terminate*(*population*[*t*]) function in Fig. 20 implements the following termination criteria: 1) terminate when the best and the worst performing chromosome in *population*[*t*] differs by less than 0.1% because in this case, the whole population becomes very similar

and it is not likely to achieve any improvement in the future generations; 2) terminate when the total number of generations specified by the user is reached; and 3) terminate when no more interesting rules in the current population can be identified because it is unlikely to find any interesting $h$-th order rules if no $(h-1)$-th order rule is found interesting.

## 7.4 Applications in Mining Meta-Rules in Rule Sets

Following the definitions of regular, differential, and change meta-rules given in Chapter 3, we present how our proposed algorithms can be used to mine such meta-rules from rule sets in this section. Specifically, given a collection of data sets, $D_1, \ldots, D_m$, FARM and EFARM are used to discover a set of rules, $R_j$, $j \in \{1, \ldots, m\}$, from each data set. Our task here is to mine regular, differential, and change meta-rules from $R_1, \ldots, R_m$.

### *7.4.1 Mining Regularities and Differences*

Given $R_1, \ldots, R_m$, a condition, $(L_i = l_{ik}) \in condition(R_1) \cup \ldots \cup condition(R_m)$, is supported by a set of rules:

$$\mathcal{R}(l_{ik}) = \{r \mid r \in R_1 \cup \ldots \cup R_m, (L_{ik} = l_{ik}) \in condition(r)\},$$

where $condition(r)$ denotes the set of conditions in $r$ and $condition(R_j) = \bigcup_{r \in R_j} condition(r)$ (defined in Chapter 3).

The support of linguistic term $l_{ik}$ is then given by:

$$fsup(l_{ik}) = \frac{|\mathcal{R}(l_{ik})|}{|R_1 \cup \ldots \cup R_m|}. \tag{7.18}$$

Similarly, an association, $l_{ik} \rightarrow l_{pq}$, where $(L_i = l_{ik})$, $(L_p = l_{pq}) \in condition(R_1) \cup \ldots \cup condition(R_m)$, is supported by a set of rules:

$$\mathcal{R}(l_{ik} \rightarrow l_{pq}) = \{r \mid r \in R_1 \cup \ldots \cup R_m, (L_{ik} = l_{ik}), (L_p = l_{pq}) \in condition(r)\}.$$

The support and the confidence of the association $l_{ik} \rightarrow l_{pq}$ are then given by:

$$fsup(l_{ik} \rightarrow l_{pq}) = \frac{|\mathcal{R}(l_{ik} \rightarrow l_{pq})|}{|R_1 \cup \ldots \cup R_m|} \tag{7.19}$$

and

$$fconf(l_{ik} \rightarrow l_{pq}) = \frac{fsup(l_{ik} \rightarrow l_{pq})}{fsup(l_{ik})}, \tag{7.20}$$

respectively.

Intuitively, $fsup(l_{ik})$ and $fsup(l_{pq})$ can be considered as being the probability that a rule has the condition $L_i = l_{ik}$ and $L_p = l_{pq}$, respectively. Similarly, $fsup(l_{ik} \rightarrow l_{pq})$ can be considered as being the probability that a rule has both $L_i = l_{ik}$ and $L_p = l_{pq}$. If $L_i = l_{ik}$ and $L_p = l_{pq}$ are independent of each other, then $fsup(l_{ik} \rightarrow l_{pq}) = fsup(l_{ik}) \times fsup(l_{pq})$. Hence $fsup(l_{ik}) \times fsup(l_{pq}) \times |R_1 \cup \ldots \cup R_m|$ yields the expected value of $|\mathcal{R}(l_{ik} \rightarrow l_{pq})|$ $(= fsup(l_{ik} \rightarrow l_{pq}) \times |R_1 \cup \ldots \cup R_m|)$. If $|\mathcal{R}(l_{ik} \rightarrow l_{pq})|$ is *significantly larger* than its expected value, it is sufficiently large. The regular meta-rule $l_{ik} \Rightarrow l_{pq}$ can therefore be formed (Definition 3.3). On the other hand, if $|\mathcal{R}(l_{ik} \rightarrow l_{pq})|$ is *significantly smaller* than its expected value, it is sufficiently small. Consequently, the differential meta-rule $l_{ik} \Rightarrow l_{pq}$ can be formed (Definition 3.4).

The difference between $fsup(l_{ik} \rightarrow l_{pq})$ and $fsup(l_{ik}) \times fsup(l_{pq})$ and hence the difference between $|\mathcal{R}(l_{ik} \rightarrow l_{pq})|$ and its expected value can be objectively evaluated in terms of the adjusted residual, $d(l_{ik} \rightarrow l_{pq})$, given by Equation (7.8). Since the adjusted residual has a normal distribution [Agresti 1990], we can conclude that $fsup(l_{ik} \rightarrow l_{pq})$ is significantly larger than $fsup(l_{ik}) \times fsup(l_{pq})$ if $d(l_{ik} \rightarrow l_{pq}) > 1.96$ (the $95^{th}$ percentile of the normal distribution). In other words, $|\mathcal{R}(l_{ik} \rightarrow l_{pq})|$ is significantly larger than its expected value and it is therefore sufficiently large. On the other hand, if $d(l_{ik} \rightarrow l_{pq}) < -1.96$, we can conclude that $fsup(l_{ik} \rightarrow l_{pq})$ is significantly smaller than $fsup(l_{ik}) \times fsup(l_{pq})$. In other words, $|\mathcal{R}(l_{ik} \rightarrow l_{pq})|$ is significantly smaller than its expected value and it is therefore sufficiently small.

It is important to note that we need to take care of not only the criterion $d(l_{ik} \rightarrow l_{pq}) > 1.96$, but also $d(l_{ik} \rightarrow l_{pq}) < -1.96$ for meta-mining. The former is to test for the regularities in common in the rule sets (i.e., regular meta-rules), whereas the latter is to test for the distinguishing relationships in only a few rule sets (i.e., differential meta-rules).

By replacing Equations (7.5)–(7.7) with Equations (7.18)–(7.20), the adjusted residual can be calculated by Equation (7.8). It can be used as a measure to identify whether the support of an association hidden in the rule sets is sufficiently large or sufficiently small in

order to identify regular or differential meta-rules, respectively. The uncertainty associated with the regular or differential meta-rules can then be evaluated by the weight of evidence given by Equation (7.13).

### 7.4.2 Mining Changes

For each rule, $r \in R_1 \cup \ldots \cup R_m$, we have a sequence of adjusted residuals, $\mathcal{S}^r = (d_1(r), \ldots, d_m(r))$, where $d_j(r)$ is the adjusted residual of $r$ in $R_j$ for $j = 1, \ldots, m$, and a sequence of weights of evidence, $C^r = (w_1(r), \ldots, w_m(r))$, where $w_j(r)$ is the weight of evidence of $r$ in $R_j$ for $j = 1, \ldots, m$. $\mathcal{S}^r$ and $C^r$ are then converted to sequences $\Delta\mathcal{S}^r = (\Delta d_1(r), \ldots, \Delta d_{m-1}(r))$ and $\Delta C^r = (\Delta w_1(r), \ldots, \Delta w_{m-1}(r))$, where $\Delta d_j(r) = d_{j+1}(r) - d_j(r)$ and $\Delta w_j(r) = w_{j+1}(r) - w_j(r)$, $j = 1, \ldots, m$, respectively. By sliding a window of width $g$ across $\Delta\mathcal{S}^r$, it is divided into a set of subsequences, $\Delta S_1^r, ..., \Delta S_{m-g}^r$, where $\Delta S_j^r = (\Delta d_j(r), ..., \Delta d_{j+g-1}(r))$. Similarly, $\Delta C^r$ is also divided into a set of subsequences, $\Delta C_1^r, ..., \Delta C_{m-g}^r$, where $\Delta C_j^r = (\Delta w_j(r), ..., \Delta w_{j+g-1}(r))$. We can then mine a set of change meta-rules of adjusted residual in subsequences $\Delta S_1^r, ..., \Delta S_{m-g}^r$ and a set of change meta-rules of weight of evidence in subsequences $\Delta C_1^r, ..., \Delta C_{m-g}^r$.

**Example 7.1** Let us consider a fuzzy association rule, $r$. Let us suppose that its adjusted residuals in certain 6 consecutive periods are given by the sequence $\mathcal{S}^r$ = (3.78, 3.49, 2.84, 2.93, 2.89, 2.97) and its weights of evidence in these 6 periods are given by the sequence $C^r$ = (5.09, 4.89, 4.59, 2.97, 3.08, 5.18). We have $\Delta\mathcal{S}^r$ = (–0.29, –0.65, 0.09, –0.04, 0.08) and $\Delta C^r$ = (–0.2, –0.3, –1.62, 0.11, 2.1). By sliding a window of width $g$ = 3 across $\Delta\mathcal{S}^r$, we obtain a set of subsequences, $\Delta S_1^r$ = (–0.29, –0.65, 0.09), $\Delta S_2^r$ = (–0.65, 0.09, –0.04), and $\Delta S_3^r$ = (0.09, –0.04, 0.08). Similarly, by sliding the window across $\Delta C^r$, we obtain a set of subsequences, $\Delta C_1^r$ = (–0.2, –0.3, –1.62), $\Delta C_2^r$ = (–0.3, –1.62, 0.11), and $\Delta C_3^r$ = (–1.62, 0.11, 2.1). □

For simplicity, we only discuss how to mine change meta-rules of adjusted residual in subsequences $\Delta S_1^r, ..., \Delta S_{m-g}^r$ in the rest of this section. It is straightforward to extend the description to mine change meta-rules of weight of evidence in subsequences $\Delta C_1^r, ..., \Delta C_{m-g}^r$.

### 7.4.2.1 Linguistic Variables and Linguistic Terms

Given subsequences $\Delta S_1^r, ..., \Delta S_{m-g}^r$, where $\Delta S_j^r = (\Delta d_j(r), ..., \Delta d_{j+g-1}(r))$, $j$ = 1, …, $m - g$, we define a set of linguistic variables, $\mathcal{L}^r = \{L_1^r, ..., L_g^r\}$, such that $L_i^r$ represents $\Delta d_{j+i-1}(r)$ in $\Delta S_j^r$ for $i$ = 1, …, $g$. The value of $L_i^r$ is a linguistic term in $T(L_i^r) = \{l_{i1}^r, ..., l_{is_i}^r\}$, where $l_{ik}^r$, $k \in \{1, …, s_i\}$, is a linguistic term defined by a fuzzy set, $F_{ik}^r$, that is defined on $\Re$, which is the domain of $\Delta d_{j+i-1}(r)$, and whose membership function is $\mu_{F_{ik}^r}$ so that:

$$\mu_{F_{ik}^r} : \Re \rightarrow [0,1].$$

The degree of compatibility of $x \in \Re$ with $L_i^r = s_{ik}^r$ is given by $\mu_{F_{ik}^r}(x)$. Since it may not be trivial for one to define the fuzzy sets, we propose to use a fuzzy partitioning technique to generate the membership functions of the fuzzy sets in Chapter 5.

Given $\Delta S_j^r$ and a linguistic term, $l_{ik}^r \in T(L_i^r)$, which is characterized by a fuzzy set, $F_{ik}^r$, the degree of membership of the values in $\Delta S_j^r$ with respect to $F_{ik}^r$ is given by $\mu_{F_{ik}^r}(\Delta d_{j+i-1}(r))$. The degree to which $\Delta S_j^r$ is characterized by $L_i^r = l_{ik}^r$, $\lambda_{l_{ik}^r}(\Delta S_j^r)$, is defined as:

$$\lambda_{l_{ik}^r}(\Delta S_j^r) = \mu_{F_{ik}^r}(\Delta d_{j+i-1}(r)). \tag{7.21}$$

If $\lambda_{l_{ik}^r}(\Delta S_j^r) = 1$, $\Delta S_j^r$ is completely characterized by $L_i^r = l_{ik}^r$. If $\lambda_{l_{ik}^r}(\Delta S_j^r) = 0$, $\Delta S_j^r$ is undoubtedly not characterized by $L_i^r = l_{ik}^r$. If $0 < \lambda_{l_{ik}^r}(\Delta S_j^r) < 1$, $\Delta S_j^r$ is partially characterized by $L_i^r = l_{ik}^r$. In the case that $\Delta d_{j+i-1}(r)$ is missing because $d_{j+i-1}(r)$ = ? and/or $d_{j+k}(r)$ = ?, $\lambda_{l_{ik}^r}(\Delta S_j^r) = 0.5$, which indicates that there is no information available concerning whether $\Delta S_j^r$ is or is not characterized by $L_i^r = l_{ik}^r$.

Each subsequence $\Delta S_j^r \in \{\Delta S_1^r, ..., \Delta S_{m-g}^r\}$ is represented by a set of ordered triples, $o_j^r$ = { $(L_1^r, l_{11}^r, \lambda_{l_{11}^r}(\Delta S_j^r))$, …, $(L_1^r, l_{1s_1}^r, \lambda_{l_{1s_1}^r}(\Delta S_j^r))$, …, $(L_g^r, l_{g1}^r, \lambda_{l_{g1}^r}(\Delta S_j^r))$, …, $(L_g^r, l_{gs_g}^r, \lambda_{l_{gs_g}^r}(\Delta S_j^r))$ }.

**Example 7.2** Let us consider the rule, *r*, described in Example 7.1. We have a set of subsequences of adjusted residual, $\Delta S_1^r$ = (–0.29, –0.65, 0.09), $\Delta S_2^r$ = (–0.65, 0.09, –0.04), and $\Delta S_3^r$ = (0.09, –0.04, 0.08). Each subsequence is then represented by three linguistic variables, $S_1^r$ (which represents "*Change in adjusted residual in 1 period ago*"), $S_2^r$ (which represents "*Change in adjusted residual in this period*"), and $S_3^r$ (which represents "*Change in adjusted residual in next period*"). The value of each linguistic variable can take from 5 linguistic terms whose membership functions are defined in the following:

$$\mu_{Highly\ decrease}(x) = \begin{cases} 1 & \text{if } x \le -0.1 \\ \frac{-1}{0.05}(x + 0.05) & \text{if } -0.1 \le x \le -0.05 \\ 0 & \text{otherwise} \end{cases}$$

$$\mu_{Fairly\ decrease}(x) = \begin{cases} \frac{1}{0.05}(x + 0.1) & \text{if } -0.1 \le x \le -0.05 \\ \frac{-1}{0.05}(x) & \text{if } -0.05 \le x \le 0 \\ 0 & \text{otherwise} \end{cases}$$

$$\mu_{More\ or\ less\ the\ same}(x) = \begin{cases} \frac{1}{0.05}(x + 0.05) & \text{if } -0.05 \le x \le 0 \\ \frac{-1}{0.05}(x - 0.05) & \text{if } 0 \le x \le 0.05 \\ 0 & \text{otherwise} \end{cases}$$

$$\mu_{Fairly\ increase}(x) = \begin{cases} \frac{1}{0.05}(x) & \text{if } 0 \le x \le 0.05 \\ \frac{-1}{0.05}(x - 0.1) & \text{if } 0.05 \le x \le 0.1 \\ 0 & \text{otherwise} \end{cases}$$

$$\mu_{Highly\ increase}(x) = \begin{cases} \frac{1}{0.05}(x - 0.05) & \text{if } 0.05 \le x \le 0.1 \\ 1 & \text{if } x \ge 0.1 \\ 0 & \text{otherwise} \end{cases}$$

.

$\Delta S_1^r$ is then represented by a set of ordered triples, $o_1^r$, where

$o_1^r$ = {($L_1^r$, *Highly decrease*, 0), ($L_1^r$, *Fairly decrease*, 0.58), ($L_1^r$, *More or less the same*, 0.42), ($L_1^r$, *Fairly increase*, 0), ($L_1^r$, *Highly increase*, 0), ($L_2^r$, *Highly decrease*, 0.3), ($L_2^r$, *Fairly decrease*, 0.7), ($L_2^r$, *More or less the same*, 0), ($L_2^r$, *Fairly increase*, 0), ($L_2^r$, *Highly increase*, 0), ($L_3^r$, *Highly decrease*, 0), ($L_3^r$, *Fairly decrease*, 0), ($L_3^r$, *More or less the same*, 0.82), ($L_3^r$, *Fairly increase*, 0.18), ($L_3^r$, *Highly increase*, 0)}.

Similarly, $\Delta S_2^r$ and $\Delta S_3^r$ are represented by $o_2^r$ and $o_3^r$, respectively. □

The fuzzy support of the linguistic term $l_{ik}^r$, $fsup(l_{ik}^r)$, is given by:

$$fsup(l_{ik}^r) = \frac{\sum_{j=1}^{m-g} \lambda_{l_{ik}^r}(\Delta S_j^r)}{\sum_{j=1}^{m-g}\sum_{k=1}^{s_i} \lambda_{l_{ik}^r}(\Delta S_j^r)}, \tag{7.22}$$

and the fuzzy support of the association $l_{ik}^r \rightarrow l_{pq}^r$, $fsup(l_{ik}^r \rightarrow l_{pq}^r)$, is calculated by:

$$fsup(l_{ik}^r \rightarrow l_{pq}^r) = \frac{\sum_{j=1}^{m-g} \min(\lambda_{l_{ik}^r}(\Delta S_j^r), \lambda_{l_{pq}^r}(\Delta S_j^r))}{\sum_{j=1}^{m-g}\sum_{k=1}^{s_i}\sum_{q=1}^{s_p} \min(\lambda_{l_{ik}^r}(\Delta S_j^r), \lambda_{l_{pq}^r}(\Delta S_j^r))}. \tag{7.23}$$

The *fuzzy confidence* of the association $l_{ik}^r \rightarrow l_{pq}^r$, $fconf(l_{ik}^r \rightarrow l_{pq}^r)$, is then given by:

$$fconf(l_{ik}^r \rightarrow l_{pq}^r) = \frac{fsup(l_{ik}^r \rightarrow l_{pq}^r)}{fsup(l_{ik}^r)}. \tag{7.24}$$

By replacing Equations (7.5)–(7.7) with Equations (7.22)–(7.24), we use 1) the adjusted residual given by Equation (7.8) as an objective interestingness measure to identify interesting change meta-rules and 2) the weight of evidence given by Equation (7.13) to evaluate the uncertainty associated with the interesting change meta-rules.

## 7.5 Comparing the FARM and EFARM Algorithms

In this section, we evaluate the performance of our proposed algorithms, FARM and EFARM, for data mining. We first applied them to several real-world data sets to test their classification performance in general (Section 7.5.1). These data sets are obtained from the UCI Machine Learning Repository [Blake and Merz 1998]. We next applied them to the subscriber database provided by a carrier in Malaysia to test their performance on assigning likelihood to their classification (Section 7.5.2).

### *7.5.1 Different Data Sets*

For each trial in each experiment, each of the data sets used was divided into two data sets with records in each of them randomly selected. The mining of rules was performed on one

of the data sets (i.e., the training data set). The other data set was reserved for testing (i.e., the testing data set). For each of these testing data sets, the values of one of the attributes were deleted. We refer to this attribute as the class attribute in the rest of this section. The rules discovered by mining the training data set were used to predict the class attribute values in the testing data set. The predicted values were then compared against the original values to see if they are the same. If it is the case, the accuracy count was incremented correspondingly. Based on this accuracy count, the percentage accuracy for each of FARM, EFARM, C4.5 [Quinlan 1993] (a well-known decision-tree classifier), CBA [Liu, Hsu, and Ma 1998] (an association rule mining algorithm), SCS [Goldberg 1989] (a Michigan-style classifier system), and GABL [DeJong, Spears, and Gordon 1993] (a Pittsburgh-style concept learner), was computed. The accuracy, averaged over a total of ten trials for each experiment, was recorded and compared and they are given in Table 24.

Since GABL is originally developed to solve "single-class (or concept)" problems, multiple populations have to be used in our experiments so that each of them can be dedicated to the learning of relationship between a single value in a multiple-valued attribute and other attribute values in a database. In our experiments, when a test record is matched by none of any rule of any class, we assign the record to the most common or the majority class in the training data set; on the other hand, when a test record is matched by more than one rule of different classes, we assign the record to the majority class that matches the record.

In our experiments, the crossover rate in EFARM was set to 0.6, the mutation rate was set to 0.0001, and the population size was set to 30. Since the performances of EFARM under different setups (Table 1) are more or less the same, we only report the experimental results of EFARM under the setup where both the crossover probability for the *crossover*-1 and that for the *crossover*-2 operator are set to 0.5 (i.e., EFARM-1) in this section. The performance of EFARM for churn prediction under different setups will be discussed in the next section.

For GABL, the mutation probability was set to 0.001, the crossover probability was set to 0.6, and the population size was set to 100 [DeJong, Spears, and Gordon 1993]. For SCS, the population size was set to 1,000, the bid coefficient was set to 0.1, the bid spread was set to 0.075, the bidding tax was set to 0.01, the existence tax was set to 0, the generality probability was set to 0.5, the bid specificity base was set to 1, the bid specificity multiplier was set to 0, the ebid specificity base was set to 1, the ebid specificity multiplier was set to 0, the reinforcement award was set to 1, the proportion to select per generation was set to 0.2, the number to select was set to 1, the mutation probability was set to 0.02, the crossover

probability was set to 1, the crowding factor was set to 3, and the crowding sub-population was set to 3 [Goldberg 1989].

All the experiments reported in this section and Section 4.5 were performed using a personal computer with Intel Pentium III 1 GHz processor as CPU, 256 MB of main memory, and running Red Hat Linux 7.1. In the following, we describe the data sets used in our experiments and present the results analyzing the performance of the different approaches.

#### 7.5.1.1 The Zoo Data Set

Each record in the *zoo* data set [Forsyth 1990] is characterized by 18 attributes. Since the unique name of each animal is irrelevant, it is ignored. All the 17 remaining attributes are discrete. The class attribute is concerned with the type of the animals are classified into. The value of the class attribute can be one of: mammal, bird, reptile, fish, amphibian, insect, and coelenterate.

#### 7.5.1.2 The DNA Data Set

Each record in the *DNA* data set [Noordewier, Towell, and Shavlik 1991] consists of a sequence of DNA, an instance name, and the class attribute. Since the unique name of each instance is irrelevant, it is ignored. A sequence of DNA contains 60 fields, each of which can be filled by one of: A, G, T, C, D (i.e., A or G or T), N (i.e., A or G or C or T), S (i.e., C or G), and R (i.e., A or G). The class attribute is concerned with the splice junctions that are points on a DNA sequence at which "superfluous" DNA is removed during the process of protein creation. It indicates the boundaries between extrons (the parts of the DNA sequence retained after splicing) and introns (the parts of the DNA sequence that are spliced out) and can be one of EI (extron-intron boundary), IE (intron-extron boundary), and N (neither extron-intron nor intron-extron boundary).

#### 7.5.1.3 The Credit Card Data Set

The *credit card* data set [Quinlan 1987a] contains data about credit card applications. It consists of 15 attributes of which the class attribute is concerned with whether or not an application is successful. The meaning of these attributes is not known as the names of the attributes and their values are changed by the donor of the database to meaningless symbols to protect the confidentiality of the data. Out of the 15 attributes, 6 are continuous and 9 are discrete. The 6 continuous attributes were fuzzy partitioned using the fuzzy partitioning technique introduced in Chapter 6.

#### 7.5.1.4 The Diabetes Data Set

Each record in the *diabetes* data set [Smith *et al.* 1988] is characterized by 9 attributes. The value of the class attribute can be either "1" (tested positive for diabetes) or "2" (tested negative for diabetes). The other attributes are continuous and they were fuzzy partitioned using the fuzzy partitioning technique proposed in Chapter 6.

#### 7.5.1.5 The Satellite Image Data Set

Each record in the *satellite image* data set corresponds to a $3 \times 3$ square neighborhood of pixels completely contained within an area. Each record contains the pixel values in the four spectral bands of each of the 9 pixels in the $3 \times 3$ neighborhood and the class attribute is the class of the central pixel that is one of: red soil, cotton crop, grey soil, damp grey soil, soil with vegetation stubble, and very damp grey soil. All the 36 (= 4 spectral bands $\times$ 9 pixels in neighborhood) attributes other than the class attribute is continuous and in the range between 0 and 255. For our experiments, these continuous attributes were fuzzy partitioned using the fuzzy partitioning technique described in Chapter 6.

#### 7.5.1.6 The Social Data Set

The *social* data set [Kohavi 1996] contains data collected by the U.S. Census Bureau. The records in the database are characterized by 15 attributes. Of these attributes, 6 of them are continuous. These continuous attributes were fuzzy partitioned using the fuzzy partitioning technique described in Chapter 6. The remaining 9 attributes are all discrete. The class attribute is concerned with whether the annual salary of a person exceeds \$50K or not.

#### 7.5.1.7 The PBX Data Set

A private branch exchange (PBX) system is a multiple-line business telephone system that resides on a company's premises. One of the significant features of a PBX system is its ability to record call activity such as keeping records of all calls and callers. In one of our experiments, we used the data from the database of a PBX system used in a telecommunications company in Indonesia. The *PBX* data set contains data about the usage of the PBX system in the company. Each record in the *PBX* data set is characterized by 13 attributes. Except for two attributes that are discrete, all the remaining attributes are continuous. The continuous attributes were fuzzy partitioned using the technique described in Chapter 6. There are many missing values in this data set. In particular, 98.4% of records have missing values in one or more attributes. The class attribute is concerned with the identification of the calling party.

### 7.5.1.8 Summary

In summary, both FARM and EFARM perform better than the other four approaches in all the seven data sets. EFARM achieves an average accuray of 91.7%, whereas FARM obtains an average rate of 88.1%. This shows that EFARM outperforms FARM in terms of classification rate on the data sets used in our experiments.

Table 24. Percentage accuracy of the six different approaches.

| Data Set | No. of Records | Class Attribute | Percentage Accuracy (Standard Deviation) | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | **FARM** | **EFARM** | **C4.5** | **CBA** | **SCS** | **GABL** |
| *zoo* | 101 | *Type* | 96.4% (4.7%) | **100.0% (0.0%)** | 90.9% (8.4%) | 92.2% (9.1%) | 27.3% (10.3%) | 28.2% (2.9%) |
| *DNA* | 3,190 | *Splice* | 93.6% (1.0%) | **95.4% (0.6%)** | 92.9% (0.8%) | 51.4% (48.9%) | 23.2% (15.1%) | 53.7% (0.0%) |
| *credit card* | 690 | *Success* | 88.8% (1.8%) | **95.6% (4.0%)** | 82.6% (4.3%) | 85.1% (5.7%) | 58.9% (9.2%) | 58.9% (0.0%) |
| *diabetes* | 768 | *Test-result* | 76.0% (2.9%) | **79.8% (1.7%)** | 73.8% (2.6%) | 69.3% (4.3%) | 61.3% (7.3%) | 61.3% (0.0%) |
| *satellite image* | 6,435 | *Soil* | 83.6% (1.0%) | **85.5% (0.8%)** | 85.2% (0.5%) | 72.1% (8.3%) | 19.9% (5.6%) | 23.1% (0.0%) |
| *social* | 48,843 | *Salary* | **86.1% (0.4%)** | 85.7% (0.2%) | 85.4% (0.3%) | 82.6% (0.3%) | 23.6% (17.2%) | 75.8% (0.0%) |
| *PBX* | 3,009 | *Calling-party-identification* | 93.7% (0.9%) | **99.9% (0.2%)** | 94.6% (5.2%) | 90.1% (5.6%) | 59.6% (26.4%) | 94.2% (0.2%) |
| | | **Average** | 88.1% | **91.7%** | 86.5% | 77.5% | 39.1% | 56.5% |

## *7.5.2 The Subscriber Database*

Since competition in the telecommunications industry is very fierce, many carriers consider reducing churn as an important business venture to maintain profitability. Churn costs carriers a large amount of money annually in North America and Europe [Lockwood 1997]. A small reduction in annual churn rate can result in a substantial increase in the valuation and the shareholder value of a carrier [Lockwood 1997]. Consequently, analyzing and controlling churn is critical for carriers to improve their revenues.

To reduce churn rate, a carrier in Malaysia gave us a database of 100,000 subscribers in a consultancy project of the Department of Computing, The Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong. Prof. Keith C. C. Chan was the principal investigator of the consultancy project. Among these subscribers, some of them have already switched to another carrier. The task assigned to us is to mine the database to uncover patterns that relate the demographics and behaviors of subscribers with churning so that further loss of subscribers can be prevented as much as possible. Efforts are then made to retain subscribers that are identified to have a high probability of switching to other

carriers.

Since the customer services center of the carrier has a fixed number of staff available to contact only a small fraction of all subscribers, it is important for it to distinguish subscribers with high probability of churning from those with low probability so that, given the limited resources, the high probability churners can be contacted first.

The subscriber database was extracted randomly from the time interval of August through October 1999. The task is to discover interesting relationships concerning with the demographics and the behaviors of the subscribers who had churned in the period between August and September 1999. By representing these relationships in the form of rules, they would then be used to predict whether a subscriber would churn in October 1999. According to the definition of the carrier, a subscriber churns when all services held by him/her are closed.

The subscriber database provided by the carrier is stored in an Oracle database. It contains three relations which are listed in Table 25. It is important to note that some attributes in some relations contain significant amount of missing values, for example, 62.4% of values in attribute LOCATION in relation DEMOGRAPHICS are missing. The handling of missing values is an important problem to be tackled for mining interesting rules in this database.

Table 25. Relations in the subscriber database.

| Relation | Description |
|---|---|
| CDR | Call detail records (each tuple, which is characterized by date, time, duration, location, etc., represents a phone call). |
| BILLING | Billing records (each tuple, which is characterized by fee, additional charges for roaming and other value-added services, etc., represents a monthly bill). |
| DEMOGRAPHICS | Demographic records (each tuple, which is characterized by service plan, handset type, etc., represents an application for services made by a subscriber). |

We, together with a domain expert from the carrier, identified 251 variables associated with each subscriber that might affect his/her churn. Some of these variables were extracted directly from the database, whereas some of them required *data transformation*, which is one of the key steps in the *knowledge discovery process* [Fayyad, Piatetsky-Shapiro, and Smyth 1996], on the original data. One of the ways to perform data transformation is the use of *transformation functions* [Au and Chan 2003; Chan and Au 2001]. Instead of

discovering rules in the original data, we applied FARM and EFARM to the *transformed data*. Table 26 lists some of the variables in the transformed data.

To manage the data mining process effectively, the transformed data are stored in a relation in the Oracle database. We refer to this relation as the *transformed relation* in the rest of this chapter. Each attribute in the transformed relation corresponds to an identified variable. The details of the use of transformation functions are given in Chapter 4.

Table 26. Some of the identified variables in the transformed data.

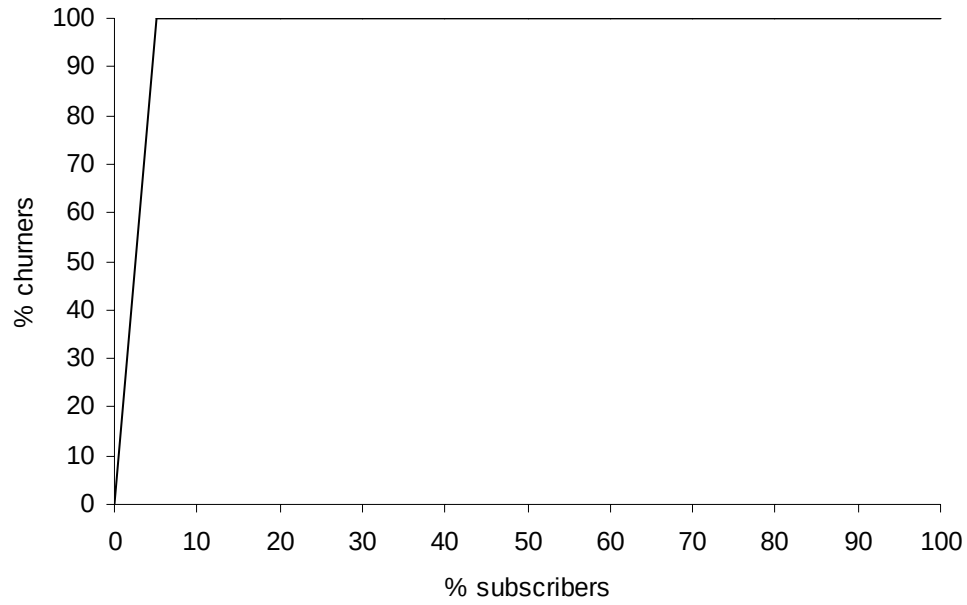
| Variable | Description |
|---|---|
| *Location* | Subscriber location. |
| *Type* | Customer type (e.g., government versus corporate). |
| *Payment Method* | Payment method (e.g., cash versus credit card). |
| *Plan* | Service plan. |
| *Charge* | Monthly charge. |
| *Usage* | Monthly usage. |
| *Calls* | Number of calls made. |
| *Abnormal Calls* | Number of abnormally terminated calls. |

Instead of mining the subscriber database, we used FARM and EFARM to mine the transformed relation. The transformed relation was divided into two partitions: the data concerning with whether subscribers had churned or had not churned in the time interval from August to September 1999 and the data concerning with whether subscribers would churn or would not churn in October 1999. The former was used as the training data set for FARM and EFARM to discover rules and the latter was used as the testing data set for them to make the "churn" and "no churn" predictions based on the discovered rules.

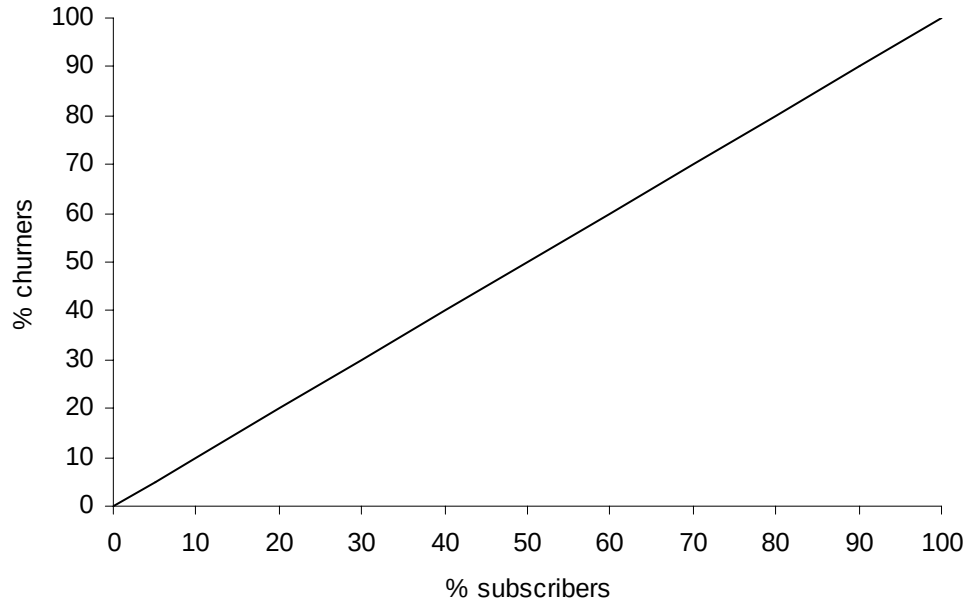
We applied FARM and EFARM to the training data set to discover rules and predict the "churn" or "no churn" of the subscribers in the testing data set. In the telecommunications industry, the "churn" and "no churn" prediction is usually expressed as a *lift curve*. The lift curve plots the fraction of all churners having churn probability above the threshold against the fraction of all subscribers having churn probability above the threshold. It indicates the fraction of all churners could be caught if a certain fraction of all subscribers were contacted. Since the customer services center of a carrier has a fixed number of staff that is able to contact only a fixed fraction of all subscribers, the lift curve, which can estimate the fraction of churners can be caught given the limited resources, is very useful in the telecommunications industry.

The lift curve representing perfect discrimination of churners from non-churners and

that representing no discrimination of churners from non-churners under a churn rate of 5% are shown in Fig. 27(a) and Fig. 27(b), respectively. We refer to the former and the latter as the *perfect churn predictor* and the *random churn predictor*, respectively.

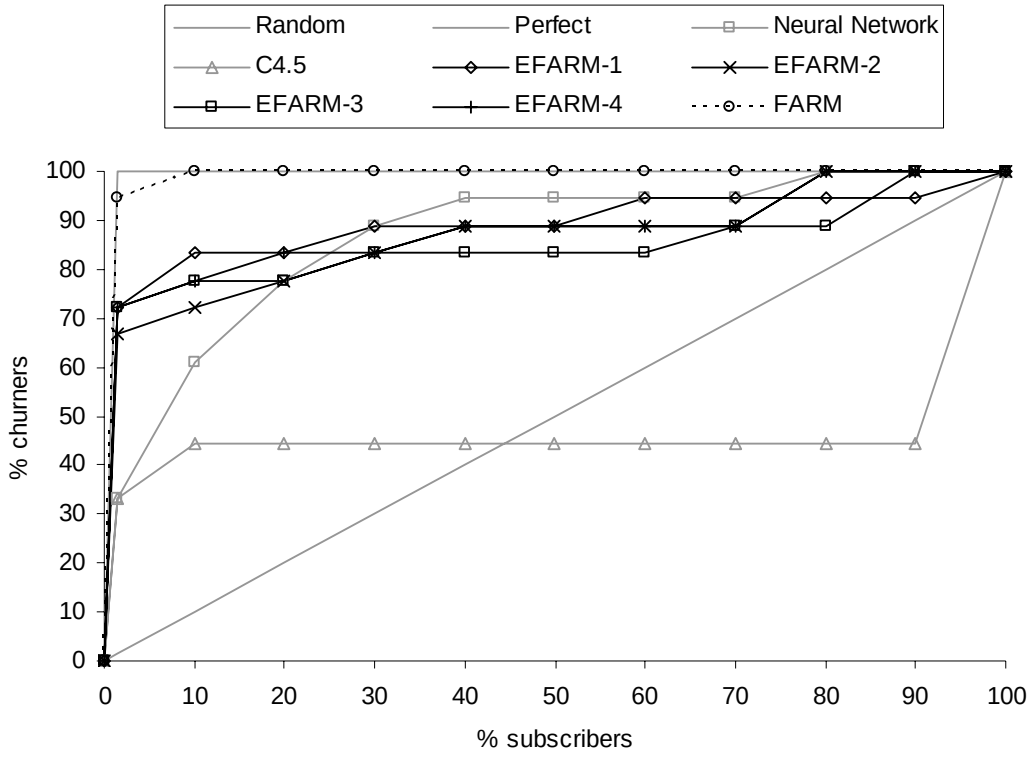



(a) Lift curve representing perfect discrimination of churners from non-churners.

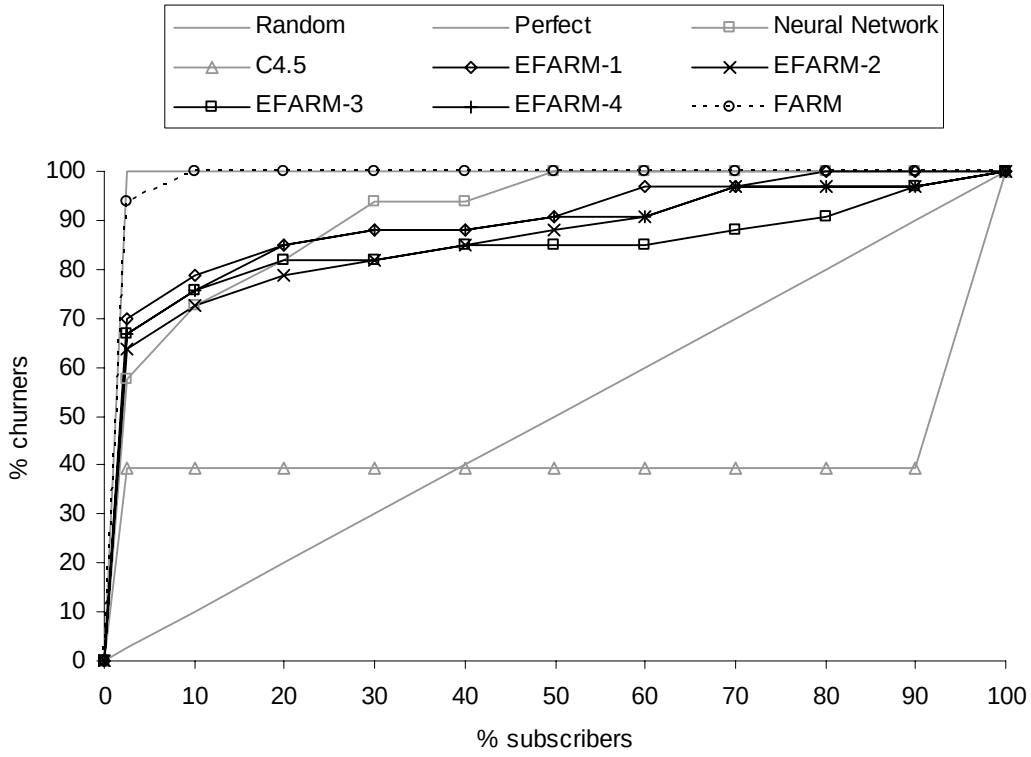



(b) Lift curve representing no discrimination of churners from non-churners.

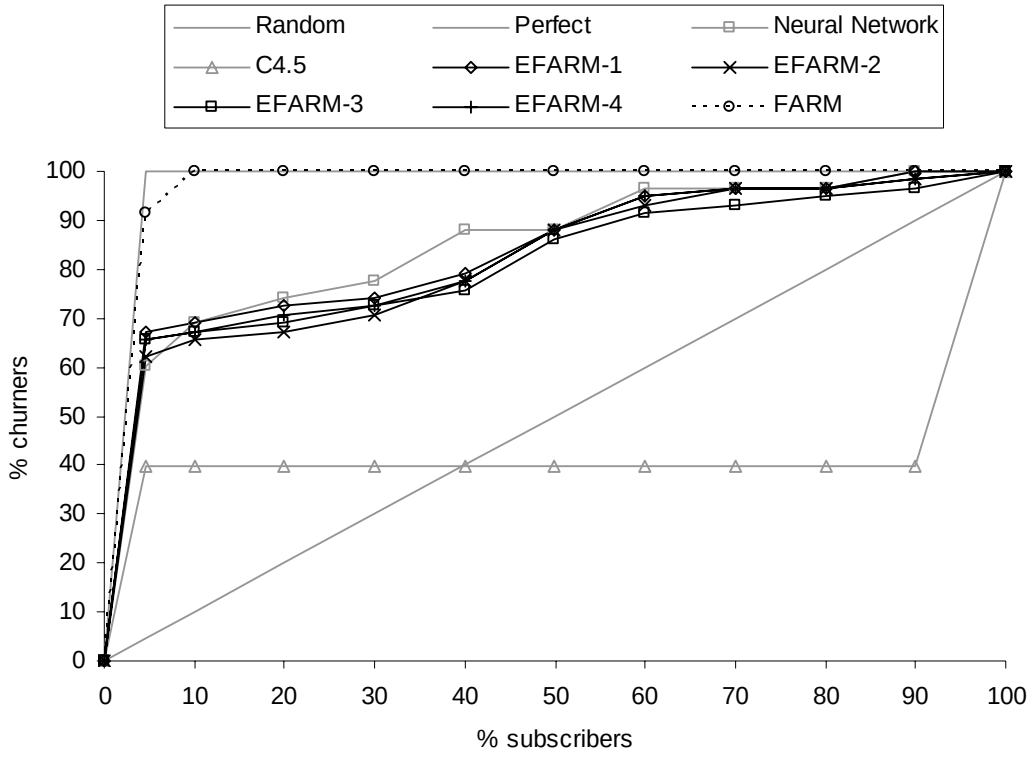
Fig. 27. Reference lift curves.



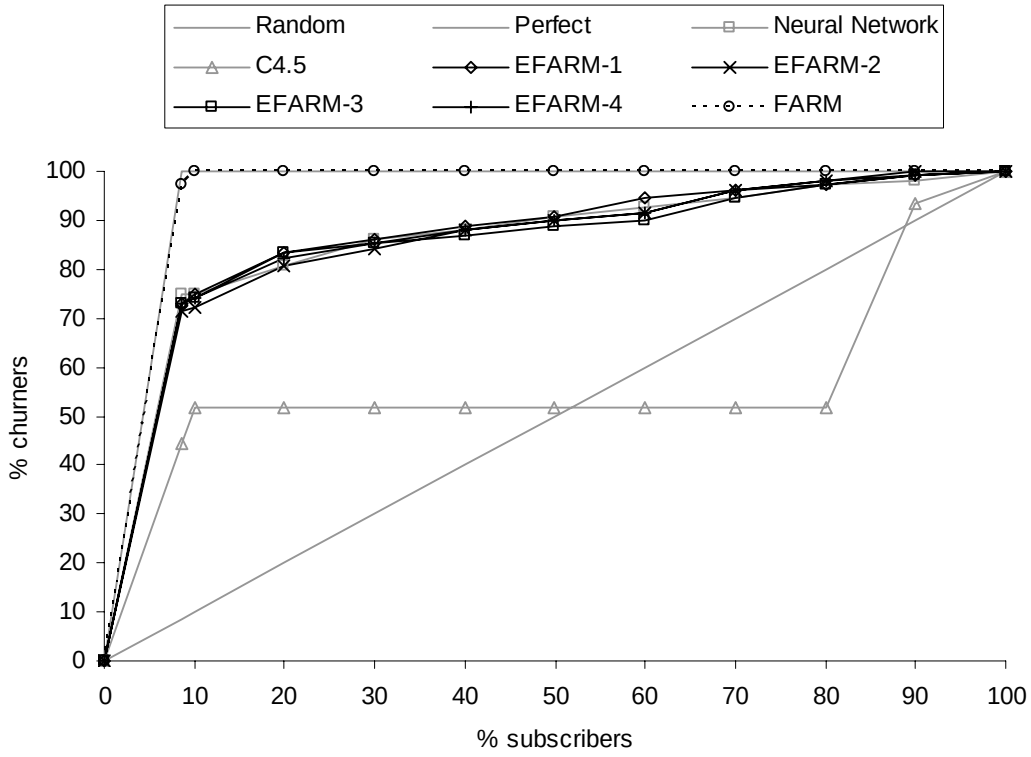

(a) Monthly churn rate = 1%.



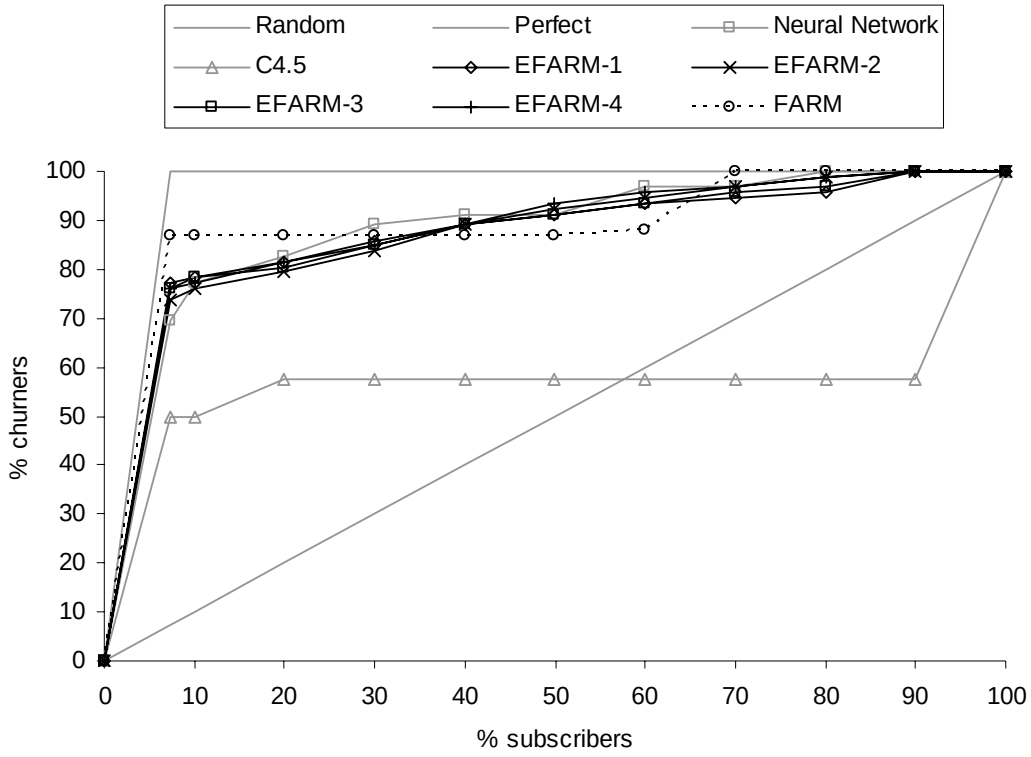

(b) Monthly churn rate = 2%.



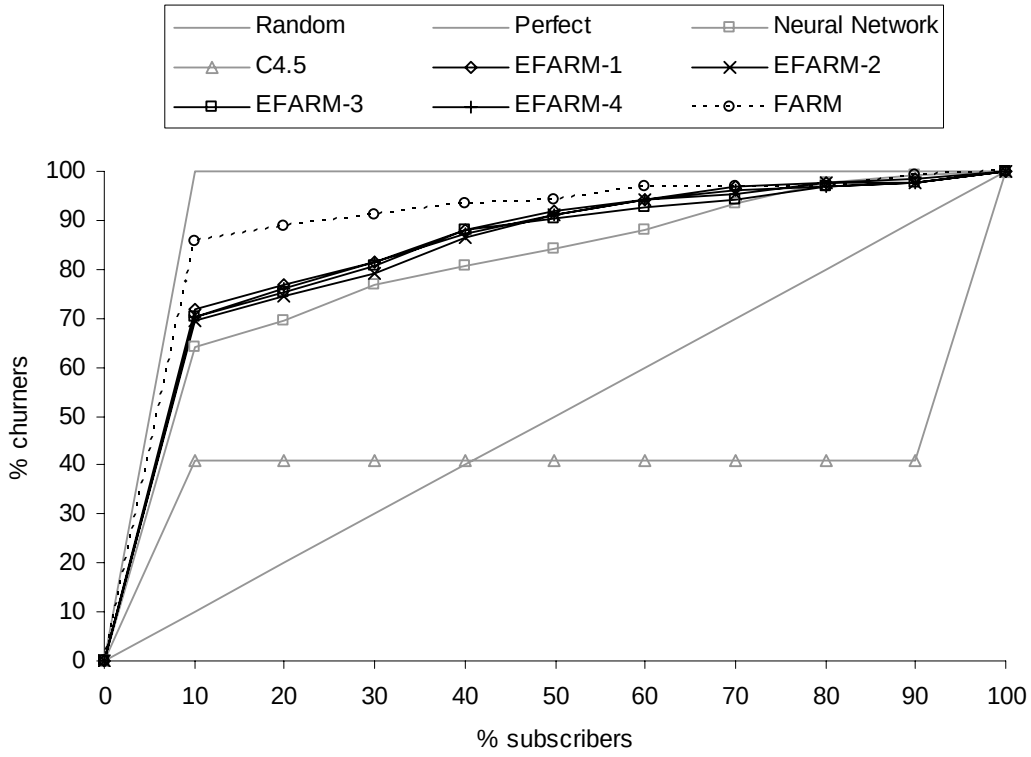

(c) Monthly churn rate = 4%.




(d) Monthly churn rate = 6%.




(e) Monthly churn rate = 8%.




(f) Monthly churn rate = 10%.

Fig. 28. Lift curves for FARM, EFARM, C4.5, and neural network under different monthly churn rates averaged over ten runs.

In order to evaluate the performance of FARM and EFARM using the lift curve, we ranked the tuples in the testing data set according to the total weight of evidence. Given the prediction and the total weight of evidence produced by FARM and EFARM over the testing data set, the tuples predicted to churn were sorted in the descending order of the total weight of evidence, whereas those tuples predicted not to churn were sorted in the ascending order of the total weight of evidence. The tuples predicted to churn came before the tuples predicted not to churn. Using the above method, we had an ordering of the tuples in the testing data set such that the ones with a higher probability to churn came before the ones with a lower probability.

Since the churn rates of different carriers are different and the churn rate of a specific carrier varies from time to time, we created several data sets with different monthly churn rates by randomly deleting tuples in the training and the testing data sets until appropriate fractions of churners and non-churners were obtained. We can then plot the performance of FARM and EFARM in the form of lift curves under different monthly churn rates (Fig. 28). The performance of EFARM under different setups (Table 23) is also given in Fig. 28.

For the purpose of comparison, we also applied C4.5 and nonlinear neural networks to these data sets. C4.5 was used because it performs better than CBA, SCS, and GABL in the experimental results given in the last section, whereas neural networks were used because they are the best churn predictor reported in an empirical study [Mozer *et al.* 2000]. The neural networks used in our experiments are multilayer perceptrons with a single hidden layer which contains 20 nodes and they were trained by the backpropagation algorithm with the learning rate was set to 0.3 and the momentum term was set to 0.7. The lift curves for C4.5 and neural networks are also shown in Fig. 28.

As shown in Fig. 28, the performances of EFARM are more or less the same under different setups of the crossover probability for the *crossover*-1 and the *crossover*-2 operator. This is a nice feature because it is usually difficult for human users to determine the appropriate values of an algorithm's parameters for it may perform well under a specific setup in a certain environment and may perform poorly under the same setup in another environment.

Regardless of the values of $p_1$ and $p_2$, the performance of EFARM is always better than that of the random churn predictor when different fraction of subscribers were contacted under different monthly churn rates. When compared to C4.5, EFARM identifies more churners than C4.5 under different monthly churn rates. It is important to note that neural networks also identify more churners than C4.5, which is consistent with the study in

[Mozer *et al.* 2000]. When compared to neural networks, EFARM identifies more churners than neural networks do when a small fraction (≤ 10%) of subscribers were contacted under different monthly churn rates. When the fraction of subscribers contacted is relatively large (> 10%), the performance of EFARM is better than that of neural networks under a monthly churn rate of ≤ 4%, whereas its performance is comparable to neural networks' under a monthly churn rate of 6% and 8%. It is interesting to note that EFARM outperforms neural networks when ≤ 80% of subscribers were contacted under a monthly churn rate of 10%.

Among all the approaches tested in our experiments, FARM is, by and large, the best churn predictor. With it, a very large fraction (> 85%) of churners could be caught when *x*% of subscribers were contacted under a monthly churn rate of *x*%. From this point of view, it performs very well as the perfect churn predictor outperforms it by only less than 15% under different monthly churn rates.

To better compare the performance of FARM, EFARM, C4.5, and neural networks, let us consider the *lift factor*, which is defined as the ratio of the fraction of churners identified and the fraction of subscribers contacted. For example, if *y*% of churners are identified when *z*% of subscribers are contacted, the lift factor is *y* / *z*. Owing to the limited number of staff in the carrier's customer services center, it can only contact 5% of all subscribers. It is important to note that the lift factor for the random churn predictor is 1, whereas the lift factor for the perfect churn predictor is 20 (= 100% / 5%) under a monthly churn rate of ≤ 5% and it is (5% / monthly churn rate) / 5 under a monthly churn rate of > 5%. The lift factors for FARM, EFARM, C4.5, and neural networks when 5% of subscribers were contacted under different monthly churn rates are shown in Fig. 29.

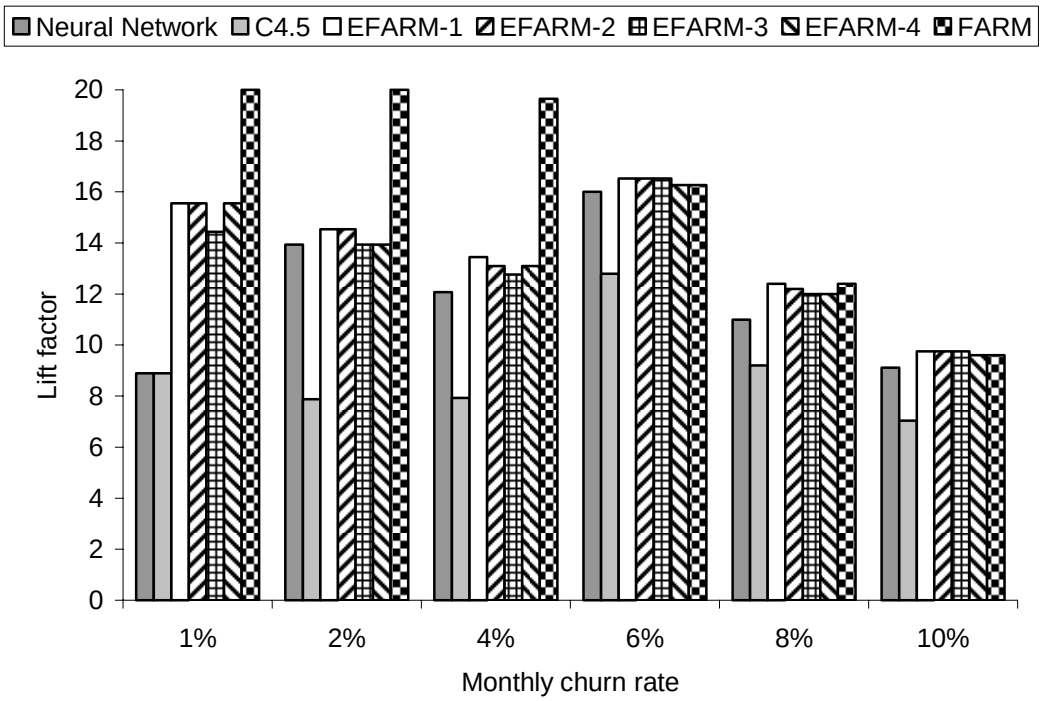



Fig. 29. Lift factors for FARM, EFARM, C4.5, and neural network under different monthly churn rates averaged over ten runs.

Again, regardless of the values of $p_1$ and $p_2$, EFARM obtains higher lift factors than neural networks, which in turn obtain higher lift factors than C4.5, when 5% of subscribers were contacted under different monthly churn rates. The experimental results show that EFARM is able to make accurate churn prediction under different churn rates. FARM achieves a very high lift factor (> 19) under a monthly churn rate of ≤ 4%. Its performance is very close to the perfect churn predictor under such a monthly churn rate. Under a monthly churn rate of ≥ 6%, it obtains a lift factor comparable to EFARM.

Furthermore, the relationships discovered by neural networks are encoded in the weights of the connections. It is difficult, if not impossible, to decode the discovered relationships and present them to the domain expert in an interpretable form. Unlike neural networks, FARM and EFARM are able to present the discovered relationships in the form of rules, which are easy for the domain expert to comprehend. Although the relationships discovered by C4.5 can also be represented in the form of rules, the experimental results show that FARM and EFARM outperform C4.5.

To evaluate their computation efficiency, Table 27 shows the execution times for FARM, EFARM, C4.5, and neural networks under different monthly churn rates. At a specific monthly churn rate, the execution times for EFARM-1, EFARM-2, EFARM-3, and

EFARM-4 are more or less the same because they differ from each other by using different values of $p_1$ and $p_2$ only. Since $p_1 + p_2 = 1$ in different setups, their time complexities should be more or less the same. When the monthly churn rate increases, the execution time for EFARM increases because more and more relationships are found interesting and hence the number of alleles in a chromosome increases.

Table 27. Execution time of FARM, EFARM, C4.5, and neural network under different monthly churn rates averaged over ten runs.

| **Monthly Churn Rate** | **Execution Time (sec.)** | | | | | | |
|---|---|---|---|---|---|---|---|
| | **Neural Network** | **C4.5** | **EFARM-1** | **EFARM-2** | **EFARM-3** | **EFARM-4** | **FARM** |
| 1% | 54,852 | 1,306 | 21,852 | 18,135 | 19,056 | 17,744 | **836** |
| 2% | 55,117 | 1,765 | 24,234 | 21,499 | 26,534 | 21,886 | **1,366** |
| 4% | 55,691 | **2,071** | 33,028 | 28,054 | 28,103 | 28,807 | 3,332 |
| 6% | 58,022 | **1,646** | 31,806 | 31,075 | 30,115 | 28,884 | 4,908 |
| 8% | 55,447 | **1,280** | 34,333 | 35,366 | 34,186 | 34,211 | 13,598 |
| 10% | 55,568 | **1,046** | 38,903 | 38,169 | 39,128 | 42,981 | 18,018 |

The experimental results show that EFARM accomplishes the data mining task faster than neural networks. They also show that EFARM runs longer than FARM under different monthly churn rates because the former requires a number of iterations for generating rules of each order, whereas the latter generates rules of each order in only one scan of the data set. Similar to EFARM, the running time of FARM increases as the monthly churn rate increases since many and many association relationships are found interesting as the monthly churn rate increases.

Of the four approaches, C4.5 requires, by and large, the least execution time to complete since it uses less number of iterations than neural networks and EFARM. When compared to FARM, it performs less efficiently than FARM under a small ($\leq$ 2%) monthly churn rate but it performs more efficiently than FARM under a relatively high ($\geq$ 4%) monthly churn rate. However, C4.5 is unable to produce churn prediction as accurate as neural networks, EFARM, and FARM (Fig. 28 and Fig. 29).

As demonstrated in the experimental results in the last section, EFARM outperforms the other techniques in the data sets used in the experiments. In the experimental results on churn prediction given in this section, EFARM also outperforms neural networks and C4.5. Although it is relatively computationally expensive, the accurate classification and prediction results can justify its usefulness. As compared to FARM, it can accomplish the mining tasks more efficiently when there are a large number of attributes.

In the rest of this section, we present the rules discovered by both FARM and EFARM and found interesting and useful by the domain expert from the carrier in Malaysia. The domain expert found the following rule very useful:

$$Type = Personal \wedge Bonus = No \Rightarrow Churn = True\ [w = 1.75].$$

This rule states that a subscriber churns if he/she subscribes the service plan personally and he/she is not admitted to any bonus scheme with a weight of evidence of 1.75. According to this rule, the domain expert suggested that the carrier could admit those subscribers who subscribe the service plan personally and have not already admitted to any bonus scheme to a bonus scheme so as to retain them.

Another rule the domain expert found interesting is listed in the following:

$$Sex = Male \wedge Tenure \in [378, 419] \Rightarrow Churn = True\ [w = 0.78].$$

The above rule states that a male subscriber who has used the service plan for a period between 378 and 419 days churns with a weight of evidence of 0.78. Although the domain expert cannot explain why this rule is applicable to male subscribers only, he found this rule meaningful because a new subscriber is usually entitled a rebate after using the service plan for a period of one year and one can still keep the money even though he churns after receiving the rebate. In order to retain these subscribers, the domain expert suggested that the carrier could offer them incentives or rebates after using the service plan for another year when they have used the service plan for a period of one year.

In addition to the above rules, the following rule was discovered:

$$District = Kuala\ Lumpur \wedge Payment\ Method = Cash \wedge Age = Middle\text{-}Aged \Rightarrow Churn = True\ [w = 1.20].$$

This rule states that a subscriber churns if he/she lives in Kuala Lumpur, is middle-aged, and pays bills using cash with a weight of evidence of 1.20. Although the domain expert could hardly explain why this rule applies to those subscribers in this age group living in Kuala Lumpur only, he found it meaningful because it is easier for a subscriber to churn if he/she pays bills using cash when compared to one who pays bills using auto pay. The domain expert found this rule useful because it identifies a niche for the carrier to retain its subscribers.

Furthermore, the domain expert also found the following rule interesting:

$$Sex = Male \wedge District = Penang \wedge Subscription\ Channel = Dealer \wedge Dealer\ Group = A$$
$$\Rightarrow Churn = True\ [w = 1.84].$$

This rule states that a male subscriber who lives in Penang and subscribes the service through a dealer, which is under Dealer Group A[2], churns with a weight of evidence of 1.84. The domain expert suggested that the churn of the subscribers might be due to the poor customer services provided by the dealers, which are under Dealer Group A, in Penang. He recommended the carrier to investigate into the service level of these dealers so as to introduce corrective actions.

[2] In order to maintain the anonymity of the carrier, we cannot disclose the name of the dealer group and we simply call it Dealer Group A in this work.

# Chapter 8
# Parallelization of Fuzzy Rule Mining Algorithms

To discover *interesting* associations in databases, many algorithms (e.g., [Agrawal, Imielinski, and Swami 1993b; Agrawal and Srikant 1994, 1996; Cheung *et al.* 1996a; Han and Fu 1995; Park, Chen, and Yu 1995a, 1995b; Savasere, Omiecinski, and Navathe 1995; Srikant and Agrawal 1995, 1996]) employ the support-confidence framework. Based on it, an association relationship is considered interesting if it satisfies the *minimum support* and the *minimum confidence* threshold defined by the users. While these algorithms can be effective in different tasks, it should be noted that what the thresholds should be set are often difficult to decide. In fact, it has been shown that association relationships discovered by algorithms employing the support-confidence framework can be quite misleading [Han and Kamber 2001; Hand, Mannila, and Smyth 2001]. For a data mining algorithm to be more effective, an objective interestingness measure that does not require a lot of effort of trial-and-error on the users' part is needed. In the last chapter, we propose one such measure. Based on the concept of statistical residual analysis, it is defined in terms of a fuzzy support and confidence measure [Au and Chan 1998, 1999, 2001, 2002a, 2002b, 2003, 2004; Au, Chan, and Yao 2003; Chan and Au 1997a, 1997b, 2001; Chan, Au, and Choi 2002] that reflects the observed and the expected degree to which a tuple is characterized by different linguistic terms.

In addition to the advantage associated with the proposed measure being objective, it also has the advantage that it can be computed in a distributed environment. In this chapter, we show how this can be made possible in two parallel algorithms: Parallel-FARM and Parallel-EFARM, which are the parallel versions of FARM (Section 7.2) and EFARM (Section 7.3), respectively. With Parallel-FARM and Parallel-EFARM, the discovery of association relationships in extremely large databases can be accomplished effectively and efficiently.

Parallel-FARM performs its tasks by first dividing a very large data set into several horizontal partitions and assigning them to different sites in a distributed system. Each site next scans its database partition to obtain the local counts of tuples that are characterized by different linguistic terms. The local counts obtained from all the other sites are then obtained to find the global counts. Based on the global counts, interesting associations can be identified using the proposed objective interestingness measure. To discover high-order associations, Parallel-FARM repeats the counting, exchanges of counts, and calculation of

the interestingness iteratively until no more interesting associations can be found.

On the other hand, Parallel-EFARM employs a parallel genetic algorithm (parallel GA) to find interesting associations so as to avoid exhaustive search in the rule space. It encodes a complete set of rules in one single chromosome and each allele encodes one rule, which is represented by some non-binary symbolic values. It stores a single population of chromosomes in a master processor. In each generation, the master processor performs selection, crossover, and mutation. It then distributes all the chromosomes among the processors in the distributed system. Each processor first determines the interestingness of each allele in each chromosome and next evaluates the fitness of the chromosomes assigned to it. After the evaluation, the processor sends the fitness of these chromosomes back to the master processor. Parallel-EFARM then proceeds to the next generation.

Both of Parallel-FARM and Parallel-EFARM were implemented in a distributed system and evaluated for effectiveness and scalability with a benchmarking data set. The experimental results show that they have very good size-up, speedup, and scale-up performance.

The rest of this chapter is organized as follows. The details of Parallel-FARM and Parallel-EFARM are given in Sections 8.1 and 8.2, respectively. To evaluate their performance, we have applied them to a popular benchmarking data set. The results are discussed in Section 8.3.

## 8.1 The Parallel-FARM Algorithm

In this section, we describe how Parallel-FARM extends FARM to discover fuzzy association rules in a distributed environment. Similar to FARM, Parallel-FARM is developed to mine high-order fuzzy association rules. A first-order fuzzy association rule can be defined as a rule involving one linguistic term in its antecedent. A second-order fuzzy association rule can be defined as a rule involving two linguistic terms in its antecedent. A third-order fuzzy association rule can be defined as a rule involving three linguistic terms in its antecedent, and so on for other higher orders.

Parallel-FARM employs the same objective interestingness measure as FARM to distinguish interesting associations from uninteresting ones. In order to handle the large combinations of linguistic terms, it also uses the same heuristic that the association between a linguistic term, $l_{\varphi' k}$, where $\varphi' = \varphi_1 \cup \varphi_2$, and another linguistic term, $l_{pq}$, is considered being more likely to be interesting if the association between $l_{\varphi_1 k}$ and $l_{pq}$ and the association

between $l_{\varphi_2 k}$ and $l_{pq}$ are interesting. Based on such a heuristic, Parallel-FARM evaluates the interestingness of only the associations between different combinations of conditions in lower-order association rules. The details of Parallel-FARM are given in the following.

Given a database relation, $D$, each tuple, $t$, in $D$ consists of a set of attributes, $\mathcal{A} = \{A_1, \ldots, A_n\}$, where $A_1, \ldots, A_n$ can be continuous or discrete. Let $\mathcal{L} = \{L_1, \ldots, L_n\}$ be a set of linguistic variables such that $L_i \in \mathcal{L}$ represents $A_i \in \mathcal{A}$ and the value of $L_i$ is a linguistic term in $T(L_i) = \{l_{ij} \mid j = 1, \ldots, s_i\}$. Let us further suppose that $\mathcal{l} = \{l_{ij} \mid i = 1, \ldots, n, j = 1, \ldots, s_i\}$.

In a distributed system comprising $m$ sites, $S_1, \ldots, S_m$, the database relation $D$ with $N$ tuples is horizontally partitioned over the $m$ sites into $D_1, \ldots, D_m$. Let the number of tuples in database partition $D_j$ be $N_j$, $j = 1, \ldots, m$.

The *fuzzy support count* of linguistic term $l_{\varphi k}$, where $\varphi \subseteq \{1, \ldots, n\}$ and $|\varphi| = h \geq 1$, in $D$ is given by:

$$count(l_{\varphi k}) = \sum_{t \in D} \lambda_{l_{\varphi k}}(t), \tag{8.1}$$

where $\lambda_{l_{\varphi k}}(t)$ is the degree to which $t$ is characterized by $l_{\varphi k}$ defined by Equation (7.4). Similarly, the fuzzy support count of $l_{\varphi k}$ in $D_j$ is calculated by:

$$count_j(l_{\varphi k}) = \sum_{t \in D_j} \lambda_{l_{\varphi k}}(t). \tag{8.2}$$

It is obvious to note that:

$$count(l_{\varphi k}) = \sum_{j=1}^{m} count_j(l_{\varphi k}). \tag{8.3}$$

We refer to $count(l_{\varphi k})$ as the *global fuzzy support count* of $l_{\varphi k}$ and $count_j(l_{\varphi k})$ as the *local fuzzy support count* of $l_{\varphi k}$ at site $S_j$.

Let us consider an $h$-th order association, $l_{\varphi k} \rightarrow l_{pq}$. The fuzzy support count of $l_{\varphi k} \rightarrow l_{pq}$ in $D$ is given by:

$$count(l_{\varphi k} \rightarrow l_{pq}) = \sum_{t \in D} \min(\lambda_{l_{\varphi k}}(t), \lambda_{l_{pq}}(t)), \tag{8.4}$$

whereas the fuzzy support count of $l_{\varphi k} \rightarrow l_{pq}$ in $D_j$ is calculated by:

$$count_j(l_{\varphi k} \rightarrow l_{pq}) = \sum_{t \in D_j} \min(\lambda_{l_{\varphi k}}(t), \lambda_{l_{pq}}(t)). \tag{8.5}$$

Again, it is obvious to note that:

$$count(l_{\varphi k} \rightarrow l_{pq}) = \sum_{j=1}^{m} count_j(l_{\varphi k} \rightarrow l_{pq}). \tag{8.6}$$

We refer to $count(l_{\varphi k} \rightarrow l_{pq})$ as the global fuzzy support count of $l_{\varphi k} \rightarrow l_{pq}$ and $count_j(l_{\varphi k} \rightarrow l_{pq})$ as the local fuzzy support count of $l_{\varphi k} \rightarrow l_{pq}$ at site $S_j$.

The fuzzy support of the linguistic term $l_{\varphi k}$ and that of the association $l_{\varphi k} \rightarrow l_{pq}$ are given by:

$$fsup(l_{\varphi k}) = \frac{count(l_{\varphi k})}{\sum_{j=1}^{s_\varphi} count(l_{\varphi j})} \tag{8.7}$$

and

$$fsup(l_{\varphi k} \rightarrow l_{pq}) = \frac{count(l_{\varphi k} \rightarrow l_{pq})}{\sum_{j=1}^{s_\varphi} \sum_{u=1}^{s_u} count(l_{\varphi j} \rightarrow l_{pq})}, \tag{8.8}$$

respectively. Based on Eqautions (8.7) and (8.8), we can calculate $d(l_{\varphi k} \rightarrow l_{pq})$ defined by Equation (7.8) to evaluate whether the association $l_{\varphi k} \rightarrow l_{pq}$ is or is not interesting.

To mine fuzzy association rules, each site in the distributed system runs Parallel-FARM. Each site is required to scan its database partition in each pass. For the $h$-th pass, each site $S_j$ generates the candidate $h$-th order rules from the $(h - 1)$-th order rules. Site $S_j$ then scans its database partition $D_j$ to obtain the local fuzzy support counts of all the candidate $h$-th order rules. After that, site $S_j$ exchanges the local fuzzy support counts to all

the other sites to find the global fuzzy support counts. Subsequently, each site $S_j$ evaluates the interestingness of the candidate $h$-th order rules to obtain the interesting ones (i.e., the $h$-th order rules). Site $S_j$ then generates the candidate ($h$ + 1)-th order rules from the $h$-th order rules and this process repeats. The algorithm terminates when neither $h$-th order rule nor candidate ($h$ + 1)-th order rule is found. Fig. 30 gives this algorithm.

**if** $h = 1$ **then {**
  **forall** $l_{ik}$, $l_{pq} \in \ell$, $i \neq p$ **do**
  **begin**
    scan $D_j$ to find $count_j(l_{ik})$, $count_j(l_{pq})$, and $count_j(l_{ik} \rightarrow l_{pq})$;
  **end**
  exchange $count_j(l_{ik})$, $count_j(l_{pq})$, and $count_j(l_{ik} \rightarrow l_{pq})$ with all the other sites to calculate $count(l_{ik})$, $count(l_{pq})$, and $count(l_{ik} \rightarrow l_{pq})$;
  $R_1 = \{l_{ik} \Rightarrow l_{pq} [w(l_{ik} \Rightarrow l_{pq})] \mid i \neq p$ and $d(l_{ik} \rightarrow l_{pq}) > 1.96\}$;
**} else {**
  $C \leftarrow$ {each linguistic term in the antecedent of $r \mid r \in R_{h-1}$}
  **forall** $l_{\varphi k}$ comprising $h$ linguistic terms in $C$ **do**
  **begin**
    **forall** $l_{pq}$, $q = 1, \ldots, s_p$, **do**
    **begin**
      scan $D_j$ to find $count_j(l_{\varphi k})$, $count_j(l_{pq})$, and $count_j(l_{\varphi k} \rightarrow l_{pq})$;
    **end**
  **end**
  exchange $count_j(l_{\varphi k})$, $count_j(l_{pq})$, and $count_j(l_{\varphi k} \rightarrow l_{pq})$ with all the other sites to calculate $count(l_{\varphi k})$, $count(l_{pq})$, and $count(l_{\varphi k} \rightarrow l_{pq})$;
  $R_h = \{l_{\varphi k} \Rightarrow l_{pq} [w(l_{\varphi k} \Rightarrow l_{pq})] \mid d(l_{\varphi k} \rightarrow l_{pq}) > 1.96\}$;
**}**

Fig. 30. The Parallel-FARM algorithm.

Since each site in the distributed system exchanges its local fuzzy counts with all the other sites to calculate the global fuzzy counts, the ($h$ – 1)-th order rules and hence the candidate $h$-th order rules, which are generated from the ($h$ – 1)-th order rules, found at different sites are identical for all $h$. After the termination of Parallel-FARM, each site therefore discovers an identical set of fuzzy association rules.

## 8.2 The Parallel-EFARM Algorithm

Parallel-EFARM is able to mine fuzzy association rules in large databases without any need for user-specified thresholds or mapping of quantitative into binary attributes. It is developed to run on a distributed system for fast execution. In the distributed system, one of the processors is chosen as the master and the other processors are selected as the slaves. The master processor is responsible for the initialization of population, the selection of chromosomes, and the recombination of chromosomes using the genetic operators, whereas each of the slave processors is responsible for the evaluation of fitness of chromosomes

assigned to it.

It is important to note that the master processor has to stop and wait to receive the fitness values for all the population before it can proceed to the next generation. In order to fully utilize the computation power of the distributed system, Parallel-EFARM has the master processor together with the slave processors to evaluate the fitness of chromosomes. Given that there are $m$ processors in the distributed system, Parallel-EFARM divides the population into $m$ subsets of chromosomes and assigns each subset to a processor for fitness evaluation.

Parallel-EFARM discovers fuzzy association rules by an iterative process. It begins with the generation of a set of first-order fuzzy association rules using the objective interestingness measure introduced in Section 7.1.2. Based on these rules, it then discovers a set of second-order fuzzy association rules in the next iteration and based on the second-order fuzzy association rules, it discovers third-order fuzzy association rules, etc. In other words, if we refer to the initial set of first-order fuzzy association rules as $R_1$, the rules in $R_1$ are then used to generate a set of second-order fuzzy association rules, $R_2$. $R_2$ is then used to generate a set of third-order fuzzy association rules, $R_3$, and so on for fourth and higher order fuzzy association rules.

In general, at the $(h - 1)$-th iteration, Parallel-EFARM begins a parallel GA by generating an initial population of chromosomes (each represents a set of $h$-th order fuzzy association rules) by randomly combining the rules in $R_{h-1}$ to form a set of $h$-th order fuzzy association rules. Once started, the parallel GA goes on uninterruptedly until no more interesting fuzzy association rule in the current population can be identified. Parallel-EFARM is given in Fig. 31.

It should be noted that the processors in the distributed system are labeled by 1,…, $m$, where $m$ is the number of processors. The details of the *evaluate* function are given in Fig. 32. In the *evaluate* function, the master processor uses the *divide* function to partition *population*[$t$] into $m$ subsets of chromosomes, which are, in turn, stored in *subset*[1], …, *subset*[$m$]. The master processor then sends *subset*[$i$] to processor $i$ for fitness evaluation for $i = 1, \ldots, m$. Upon receiving the subset of chromosomes, a processor evaluates their fitness and sends the fitness values back to the master processor. On the receipt of the fitness values from all the processors, the master processor updates the fitness values of the chromosomes in *population*[$t$].

All the remaining components of Parallel-EFARM are the same as EFARM given in

Section 7.3.

**MASTER PROCESSOR:**

$R_1 \leftarrow \{ l_{ik} \Rightarrow l_{pq} [w(l_{ik} \Rightarrow l_{pq})] \mid i \neq p$ and $d(l_{ik} \rightarrow l_{pq}) > 1.96\}$;
$h \leftarrow 2$;
**while** $R_{h-1} \neq \varnothing$ **do**
**begin**
    $t \leftarrow 0$;
    *population*[$t$] $\leftarrow$ *initialize*($R_{h-1}$);
    *evaluate*(*population*[$t$]);  /* see Fig. 32 */
    **while not** *terminate*(*population*[$t$]) **do**
    **begin**
        $t \leftarrow t + 1$;
        *population*[$t$] $\leftarrow$ *reproduce*(*population*[$t - 1$]);
        *evaluate*(*population*[$t$]); /* see Fig. 32 */
    **end**
    $R_h \leftarrow$ *decode*(the fittest individual in *population*[$t$]);
    $h \leftarrow h + 1$;
**end**
$Rules \leftarrow \bigcup_h R_h$ ;

**SLAVE PROCESSOR:**

receive *subset*[$i$] from the master processor;
send *fitness*(*subset*[$i$]) to the master processor;

Fig. 31. The Parallel-EFARM algorithm.

*evaluate*(*population*[$t$])
**begin**
    *subset*[1], …, *subset*[$m$] $\leftarrow$ *divide*(*population*[$t$]);
    **for** $i = 1$ **to** $m$ **do**
    **begin**
        send *subset*[$i$] to processor $i$;
        receive *fitness*[$i$] from processor $i$;
    **end**
    *update_fitness*(*population*[$t$], *subset*[1], …, *subset*[$m$]);
**end**

Fig. 32. The *evaluate* function.

## 8.3 Scalability Evaluation

We implemented Parallel-FARM and Parallel-EFARM in a distributed system using PVM (Parallel Virtual Machine) [Geist *et al.* 1994]. To perform our experiments, a 100 Mb LAN was used to connect ten Sun Ultra 5 workstations, each of which has 64 MB of main memory running Solaris 2.5.1. Each workstation has a local drive and its database partition is loaded on its local drive before each experiment started. The databases used in our

experiments on scalability are synthetic data generated using the tool provided by [IBM 1996]. Each tuple in the databases is characterized by 9 attributes. Of the 9 attributes, 3 are discrete and 6 are continuous.

In order to evaluate the performance of Parallel-FARM, we also implemented *Count Distribution* in our test bed. For each database, we discretized the domain of continuous attributes into several intervals and mapped the values of discrete attributes and the intervals of discretized continuous attributes into integers. We then applied *Count Distribution* to the transformed data. Since *Count Distribution* finds frequent itemsets based on support constraint, we applied it to the databases using various minimum supports so as to evaluate how its performance is affected by the setting of minimum support.

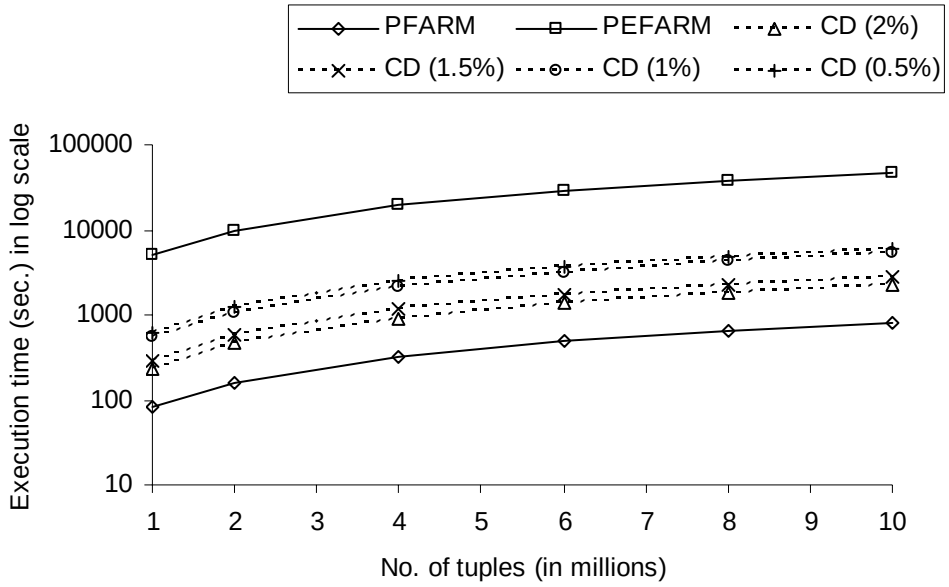
We ran a number of experiments to evaluate the sizeup, speedup, and scaleup performance of Parallel-FARM, Parallel-EFARM, and *Count Distribution*. In the rest of this section, we refer to *Count Distribution* as CD.

### 8.3.1 Sizeup

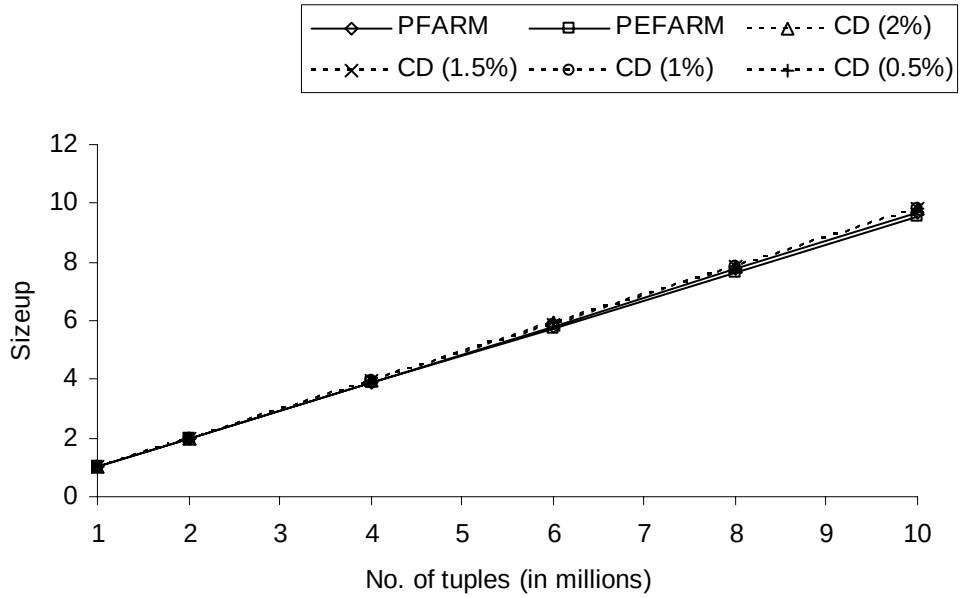
In our first experiment, we fixed the number of sites in the distributed system to 10. To evaluate the performance of Parallel-FARM, Parallel-EFARM, and CD with respect to different database sizes, we increased the number of tuples from 1 million to 10 million in our experiment. Fig. 33 shows the performance of Parallel-FARM, Parallel-EFARM, and CD as the database size increases. In addition to the absolute execution times, we also plot sizeup, which is the execution time normalized with respect to the execution time for 1 million tuples, in Fig. 33 ("PFARM" denotes Parallel-FARM, "PEFARM" denotes Parallel-EFARM, and "CD (*x*%)" denotes running CD with minimum support = *x*%).

As shown in Fig. 33, both Parallel-FARM and Parallel-EFARM scale almost linearly in this experiment. When the database size increases, more I/O and CPU processing are required to scan the database for obtaining the local counts and to compute the interestingness measure for identifying interesting association relationships. The amount of execution time spent in communication is more or less the same regardless of the database size because the number of association relationships is independent of the database size and only their local counts are exchanged between different sites in the distributed system. This characteristic of the algorithm results in the reduction of the percentage of the overall execution time spent in communication. Since the I/O and CPU processing in Parallel-FARM and Parallel-EFARM scales linearly with the database size, they show sublinear performance.

This experiment also shows that the performance of Parallel-FARM is superior to CD with respect to different database sizes. Specifically, Parallel-FARM is 2.8 times faster than CD with minimum support = 2% and 7.6 times faster than CD with minimum support = 0.5%.




(a) Exection time.




(b) Sizeup.

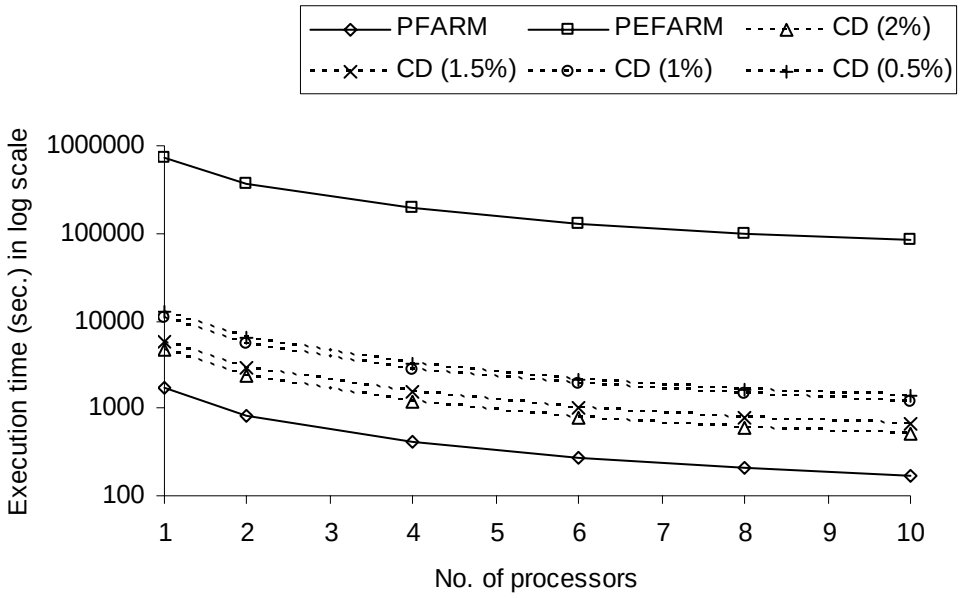
Fig. 33. Sizeup performance.

### 8.3.2 Speedup

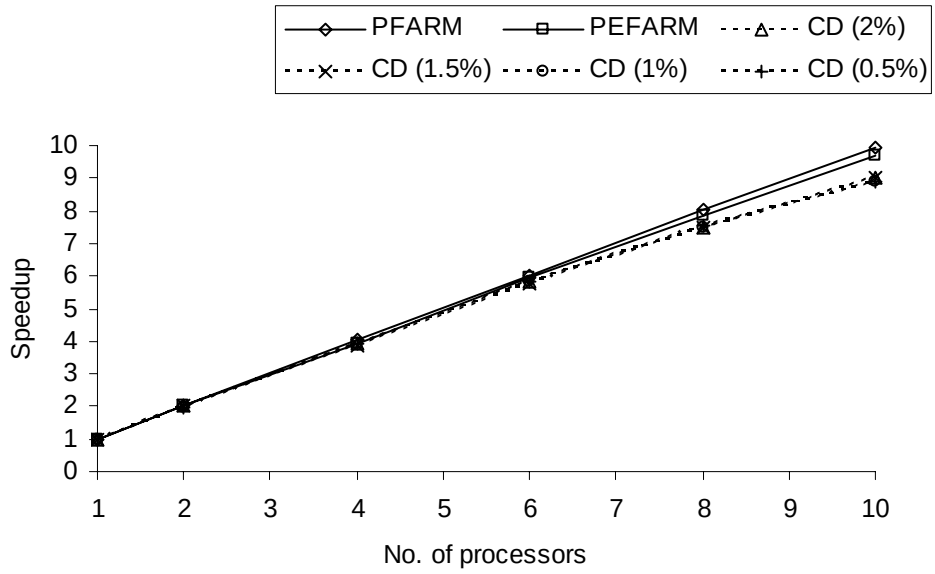
In our second experiment, we fixed the database size to 2 million tuples. To evaluate the performance of Parallel-FARM and Parallel-EFARM with respect to different number of sites in the distributed system, we increased the number of sites from 1 to 10 in our experiment. Fig. 34 shows their performance as the number of sites increases. In addition to the absolute execution times, we also plot speedup, which is the execution time normalized with respect to the execution time for a single site, in Fig. 34.

As shown in Fig. 34, Parallel-FARM and Parallel-EFARM exhibit very good speedup performance in this experiment. In particular, when there are *m* sites in the distributed system, they can shorten the execution time to about 1 / *m* of the execution time for a single site. It is important to note however that given the same amount of data, the speedup performance will deteriorate as the number of sites in the distributed system increases. The deterioration is due to the communication time becoming a significant percentage of the overall execution time when compared to the relatively small processing time for the small amount of data to process on each site.

This experiment also shows that Parallel-FARM performs better than CD with respect to different number of sites in the distributed system. In particular, when there are 2 sites in the distributed system, Parallel-FARM is 2.7 times faster than CD with minimum support = 2% and 7.4 times faster than CD with minimum support = 0.5%. When there are 10 sites in the distributed system, Parallel-FARM is 3 times faster than CD with minimum support = 2% and 8.3 times faster than CD with minimum support = 0.5%.




(a) Execution time.




(b) Speedup.

Fig. 34. Speedup performance.

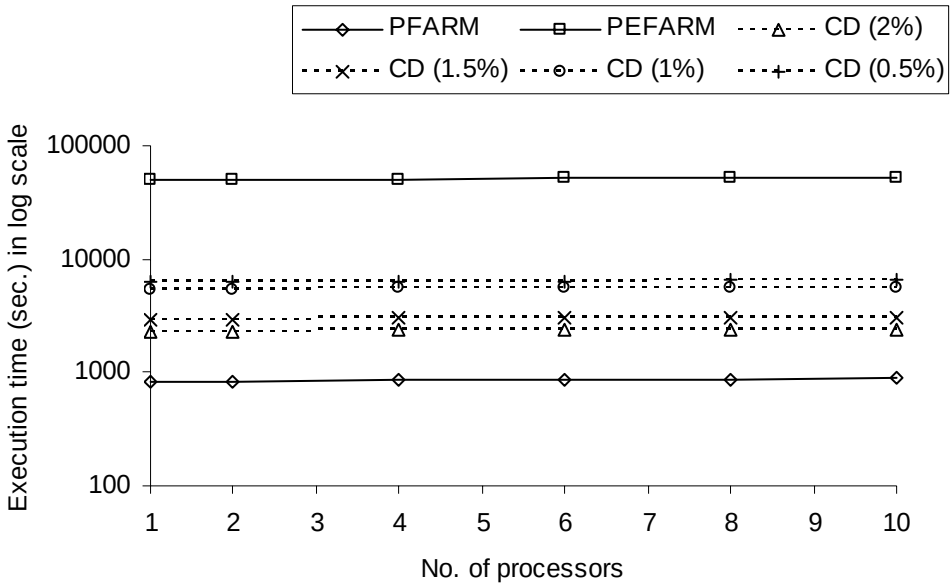
### *8.3.3 Scaleup*

In this experiment, we fixed the size of the database partition at a site to 1 million tuples. We increased the number of sites in the distributed system from 1 to 10. Fig. 35 shows the performance of Parallel-FARM and Parallel-EFARM as the number of sites increases. In addition to the absolute execution time, we also plot scaleup, which is the execution time

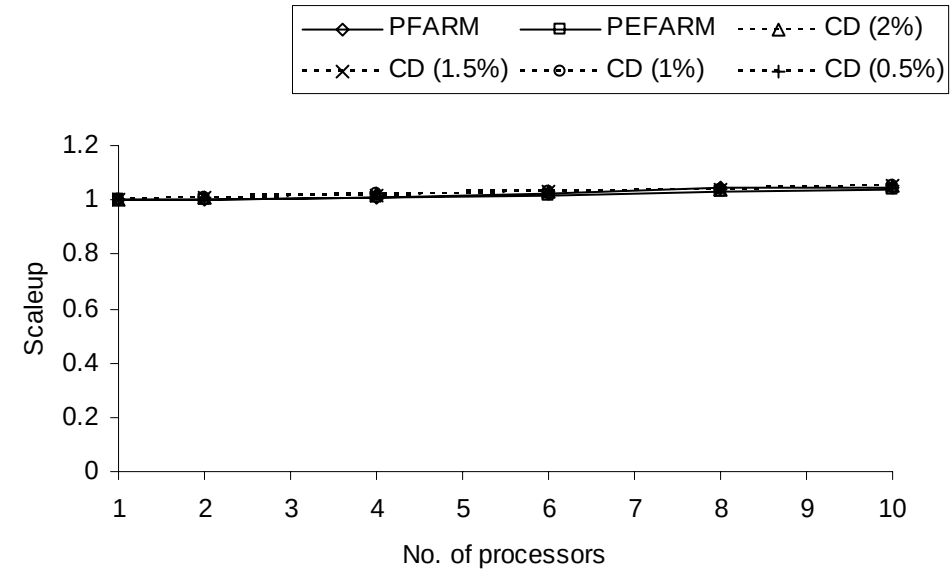
normalized with respect to the execution time for a single site, in Fig. 35.

As shown in Fig. 35, Parallel-FARM and Parallel-EFARM have very good scaleup performance. Since the number of association relationships they find does not change when the database size increases, the I/O and CPU processing at each site remains constant. The execution time increases slightly as the database size and the number of sites increase. The small increment in execution time is due again to the increase in the communication overhead when there are more and more sites in the distributed system.

This experiment also shows that Parallel-FARM can better handle larger databases when more processors are available when compared to CD. Parallel-FARM is 2.7 times faster than CD with minimum support = 2% and 7.4 times faster than CD with minimum support = 0.5%.




(a) Execution time.




(b) Scaleup.

Fig. 35. Scaleup performance.

# Chapter 9
# Experimental Results

## 9.1 Synthetic Data Sets

To evaluate the performance of our proposed algorithms on meta-mining tasks, we carried out two experiments using synthetic data sets. In our first experiment, we embedded some association relationships in six synthetic data sets and then tested whether our algorithms were able to discover the regularities and the differences hidden in the underlying association relationships. In the second experiment, we embedded some rule changes in several synthetic rule sets. We examine whether our algorithms can reveal such rule changes.

In our experiments, our proposed algorithms, FARM and EFARM, produce more or less the same results. We therefore in this section report only the results obtained by FARM.

### *9.1.1 Mining Regularities and Differences*

In this experiment, we test the proposed algorithm for effectiveness when it is used to discover the underlying regularities and differences embedded in data sets. We generated six data sets for experimentation. Each tuple in these data sets is characterized by 3 attributes: *X*, *Y*, and *Z*. Each of these attributes can take on two values: *T* and *F*. Each data set contains 1,000 tuples. We generated the first five data sets, $D_1$, …, $D_5$, according to the following association relationships:

$$X = F \wedge Y = F \Rightarrow Z = F$$
$$X = F \wedge Y = T \Rightarrow Z = T$$
$$X = T \wedge Y = F \Rightarrow Z = T$$
$$X = T \wedge Y = T \Rightarrow Z = F.$$

The remaining data set, $D_6$, was generated according to the following association relationships:

$$X = F \wedge Y = F \Rightarrow Z = F$$

$$X = F \wedge Y = T \Rightarrow Z = F$$

$$X = T \wedge Y = F \Rightarrow Z = F$$

$$X = T \wedge Y = T \Rightarrow Z = T.$$

To further examine the performance of our algorithm in the presence of uncertainty, 5% of noise was added randomly to the data sets by randomly changing the value of $Z$ in 50 tuples (i.e., 5% of all tuples) from $F$ to $T$ or vice versa. We applied our proposed algorithm to $D_j$ to discover rules and stored the discovered rules in $R_j$, $j$ = 1, …, 6. The discovered rules together with their adjusted residuals and weights of evidence are given in Table 28.

Table 28. Rules discovered in the data sets.

| Rule Set | Rule | Adjusted Residual | Weight of Evidence |
|---|---|---|---|
| $R_1$ | $X = F \wedge Y = F \Rightarrow Z = F$ | 16.10 | 4.06 |
| | $X = F \wedge Y = T \Rightarrow Z = T$ | 16.61 | 4.33 |
| | $X = T \wedge Y = F \Rightarrow Z = T$ | 16.61 | 4.33 |
| | $X = T \wedge Y = T \Rightarrow Z = F$ | 17.13 | 5.10 |
| $R_2$ | $X = F \wedge Y = F \Rightarrow Z = F$ | 15.96 | 3.88 |
| | $X = F \wedge Y = T \Rightarrow Z = T$ | 16.03 | 3.96 |
| | $X = T \wedge Y = F \Rightarrow Z = T$ | 16.18 | 4.07 |
| | $X = T \wedge Y = T \Rightarrow Z = F$ | 16.25 | 4.08 |
| $R_3$ | $X = F \wedge Y = F \Rightarrow Z = F$ | 15.96 | 3.95 |
| | $X = F \wedge Y = T \Rightarrow Z = T$ | 15.74 | 3.71 |
| | $X = T \wedge Y = F \Rightarrow Z = T$ | 16.76 | 4.46 |
| | $X = T \wedge Y = T \Rightarrow Z = F$ | 16.54 | 4.42 |
| $R_4$ | $X = F \wedge Y = F \Rightarrow Z = F$ | 16.40 | 4.29 |
| | $X = F \wedge Y = T \Rightarrow Z = T$ | 16.03 | 3.89 |
| | $X = T \wedge Y = F \Rightarrow Z = T$ | 16.76 | 4.46 |
| | $X = T \wedge Y = T \Rightarrow Z = F$ | 16.40 | 4.29 |
| $R_5$ | $X = F \wedge Y = F \Rightarrow Z = F$ | 17.02 | 4.62 |
| | $X = F \wedge Y = T \Rightarrow Z = T$ | 16.73 | 4.71 |
| | $X = T \wedge Y = F \Rightarrow Z = T$ | 16.73 | 4.71 |
| | $X = T \wedge Y = T \Rightarrow Z = F$ | 16.43 | 4.11 |
| $R_6$ | $X = F \wedge Y = F \Rightarrow Z = F$ | 7.98 | 2.17 |
| | $X = F \wedge Y = T \Rightarrow Z = F$ | 9.95 | 3.68 |
| | $X = T \wedge Y = F \Rightarrow Z = F$ | 9.62 | 3.31 |
| | $X = T \wedge Y = T \Rightarrow Z = T$ | 27.55 | 5.40 |

As shown in Table 28, our algorithm is able to uncover all the underlying association relationships embedded in the six data sets. It was next used to mine meta-rules from the rule sets $R_1$, …, $R_6$. Table 29 shows the regular meta-rules discovered from the rule sets.

Table 29. Regular meta-rules discovered in the rule sets.

| Regular Meta-Rule | Adjusted Residual | Weight of Evidence |
|---|---|---|
| $X = F \wedge Y = F \Rightarrow Z = F$ | 2.60 | *infinity* |
| $X = F \wedge Y = T \Rightarrow Z = T$ | 2.13 | 2.56 |
| $X = T \wedge Y = F \Rightarrow Z = T$ | 2.13 | 2.56 |

The regular meta-rule "$X = F \wedge Y = F \Rightarrow Z = F$" is supported by six rules (one in each rule set), whereas the meta-rules "$X = F \wedge Y = T \Rightarrow Z = T$" and "$X = T \wedge Y = F \Rightarrow Z = T$" are supported by five rules (one in each of $R_1$, …, $R_5$). All of them represent the regularities in the rule sets, which in turn reflect the characteristics in common in the data sets.

Let us consider the meta-rule "$X = F \wedge Y = T \Rightarrow Z = T$" as an example. It is supported by five rules. Its antecedent "$X = F \wedge Y = T$" is supported by 6 rules, whereas its consequent "$Z = T$" is supported by 11 rules. Assuming that they are independent of each other, the meta-rule is expected to be supported by 2.75 (= 11 × 6 / 24) rules (given by Equation (4.10)). We next need to decide whether 5 is significantly larger than 2.75. To do so in an objective manner, we propose to use the adjusted residual analysis. The adjusted residual is 2.13 (calculated by Equation (4.8)), which is greater than 1.96 (the 95$^{th}$ percentile of the normal distribution). We therefore conclude that the meta-rule is supported by a sufficiently large number of rules and hence it represents one of the regularities in the rule sets (i.e., a regular meta-rule).

It is important to note that the meta-rule "$X = T \wedge Y = T \Rightarrow Z = F$" is also supported by five rules (one in each of $R_1$, …, $R_5$). Its antecedent "$X = T \wedge Y = T$" and its consequent "$Z = F$" are supported by 6 and 13 rules, respectively. Therefore, we expect that 3.25 (= 13 × 6 / 24) rules would support this meta-rule. To objectively decide whether 5 is significantly larger than 3.25, we make use of the adjusted residual analysis. The adjusted residual is found to be 1.66 (< 1.96). Hence we conclude that the meta-rule is not supported by a sufficiently large number of rules.

In addition to discovering regular meta-rules, our algorithms can also discover differential meta-rules for representing the distinctive relationships in only a few rule sets. Table 30 gives the differential meta-rules discovered from the rule sets.

Table 30. Differential meta-rules discovered in the rule sets.

| Differential Meta-Rule | Adjusted Residual | Weight of Evidence |
|---|---|---|
| $X = F \wedge Y = T \Rightarrow Z = F$ | –2.13 | –2.56 |
| $X = T \wedge Y = F \Rightarrow Z = F$ | –2.13 | –2.56 |

For example, the meta-rule "$X = T \wedge Y = F \Rightarrow Z = F$" is supported by only one rule in $R_6$. Its antecedent "$X = T \wedge Y = F$" and consequent "$Z = F$" are supported by 6 and 13 rules, respectively. Hence 3.25 (= 13 × 6 / 24) rules are expected to support this meta-rule. We find that 1 is significantly less than 3.25 as the adjusted residual is –2.13 (< –1.96). We conclude that the meta-rule is supported by a sufficiently small number of rules and hence it represents a distinguishing relationship (i.e., a differential meta-rule).

Let us consider the meta-rule "$X = T \wedge Y = T \Rightarrow Z = T$," which is also supported by one rule in $R_6$. Its antecedent "$X = T \wedge Y = T$" is supported by 6 rules, whereas its consequent "$Z = T$" is supported by 11 rules. We expect it would be supported by 2.75 (= 11 × 6 / 24) rules. The adjusted residual is –1.66 (> –1.96) and hence 1 is not significantly less than 2.75. We therefore conclude that the meta-rule is not supported by a sufficiently small number of rules.

### 9.1.2 Mining Changes

In our experiment, we first generated a synthetic data set using the tool provided by [IBM 1996]. The parameter setting for generating the synthetic data set is listed in Table 31. The parameters and the method for the generation of the synthetic data set are detailed in [Agrawal and Srikant 1994].

Table 31. Parameter setting for generating the synthetic data set.

| Parameter | Value |
|---|---|
| Number of transactions | 1,000 |
| Average size of transactions | 5 |
| Average size of the maximal potentially large itemsets[3] | 2 |
| Number of maximal potentially large itemsets | 20 |
| Number of items | 100 |

We used the synthetic data set as the transactions collected in time period $t_1$ (i.e., $D_1$).

[3] An itemset is *large* if its support is greater than or equal to minimum support.

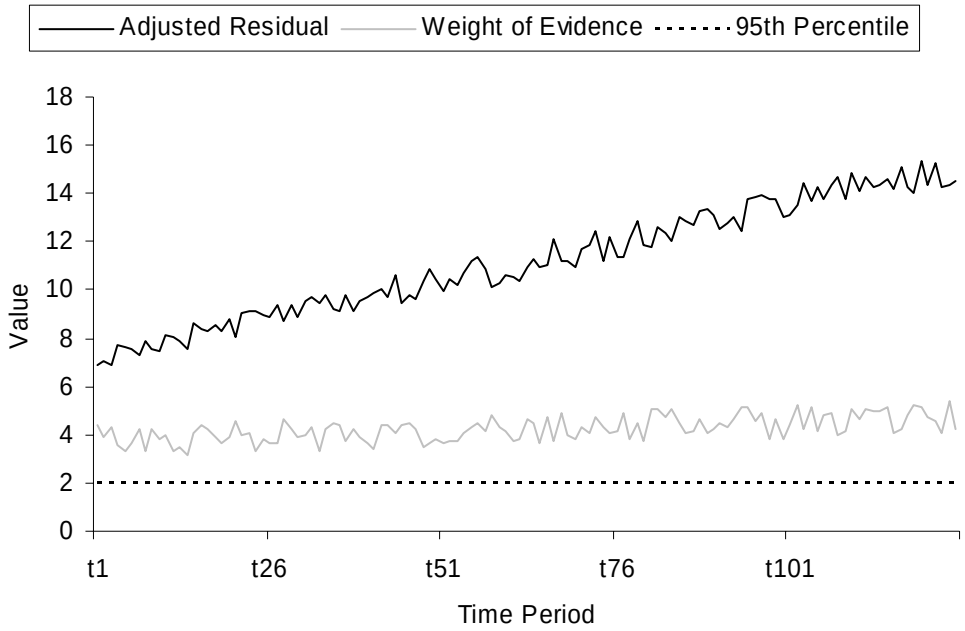
We next applied our algorithm to $D_1$ to discover rules concerned with item $i_7$. A set of 28 rules was discovered and stored in $R_1$. We selected five rules randomly for further experimentation. The selected rules are shown in Table 32.

Table 32. The rules in $R_1$ selected for further experimentation.

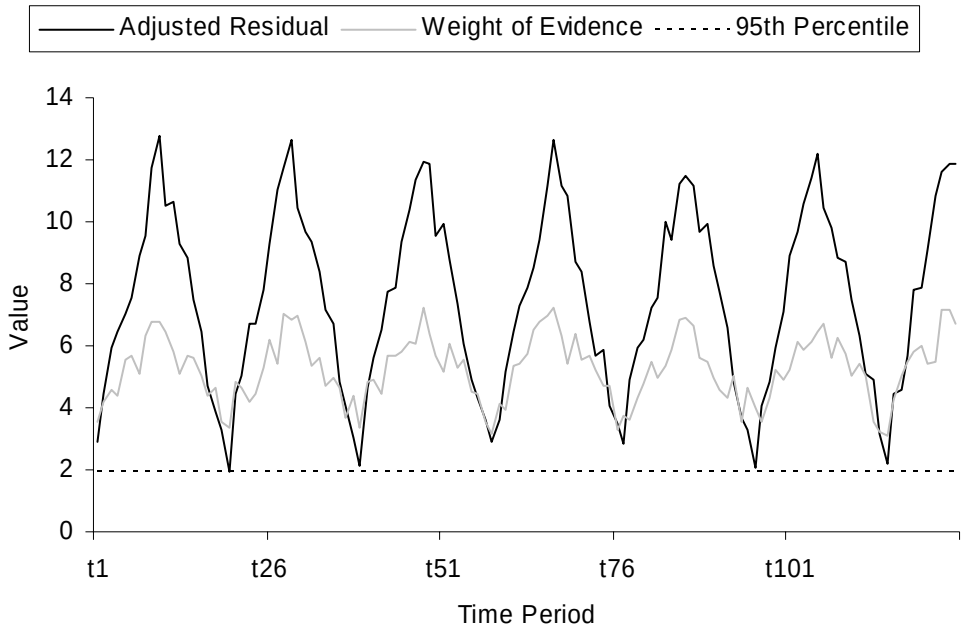
| Rule | Adjusted Residual | Weight of Evidence |
|---|---|---|
| $r_1$: $\{i_{93}, i_{94}, i_{99}\} \Rightarrow \{i_7\}$ | 6.87 | 4.40 |
| $r_2$: $\{i_{93}, i_{97}, i_{99}\} \Rightarrow \{i_7\}$ | 2.88 | 3.56 |
| $r_3$: $\{i_{93}, i_{96}, i_{99}\} \Rightarrow \{i_7\}$ | 15.87 | 6.40 |
| $r_4$: $\{i_{93}, i_{98}\} \Rightarrow \{i_7\}$ | 6.85 | 4.33 |
| $r_5$: $\{i_{93}, i_{96}\} \Rightarrow \{i_7\}$ | 7.48 | 4.21 |

We then generated another 124 data sets, $D_2$, …, $D_{125}$, in such a way that 1) $r_1$, …, $r_4$ change in the period from $t_1$ to $t_{125}$; 2) $r_5$ is perished in $t_{125}$; 3) $r_6$ is added, changes, and is perished periodically during the period from $t_1$ to $t_{125}$; 4) a new rule, $r_7$, is added in $t_{71}$; and 5) all the other rules remain the same in the period from $t_1$ to $t_{125}$. Fig. 36 shows how $r_1$, ..., $r_7$ change in the period from $t_1$ to $t_{125}$.

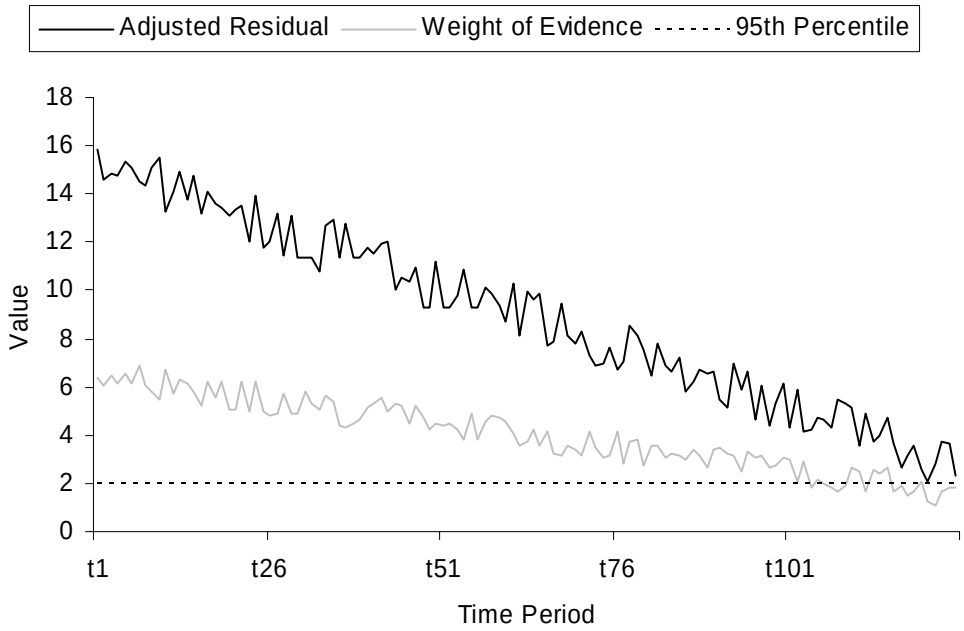
The association rules discovered in $D_{125}$ and stored in $R_{125}$ are given in Table 33.



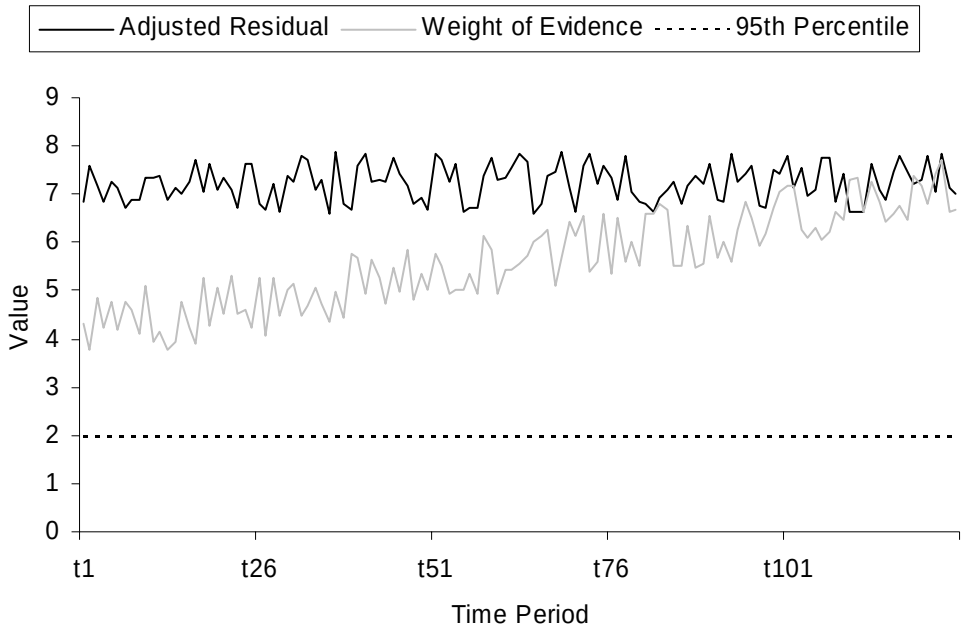

(a) $r_1$.


Adjusted Residual
Weight of Evidence
95th Percentile
14
12
10
8
6
4
2
0
Value
t1
t26
t51
t76
t101
Time Period

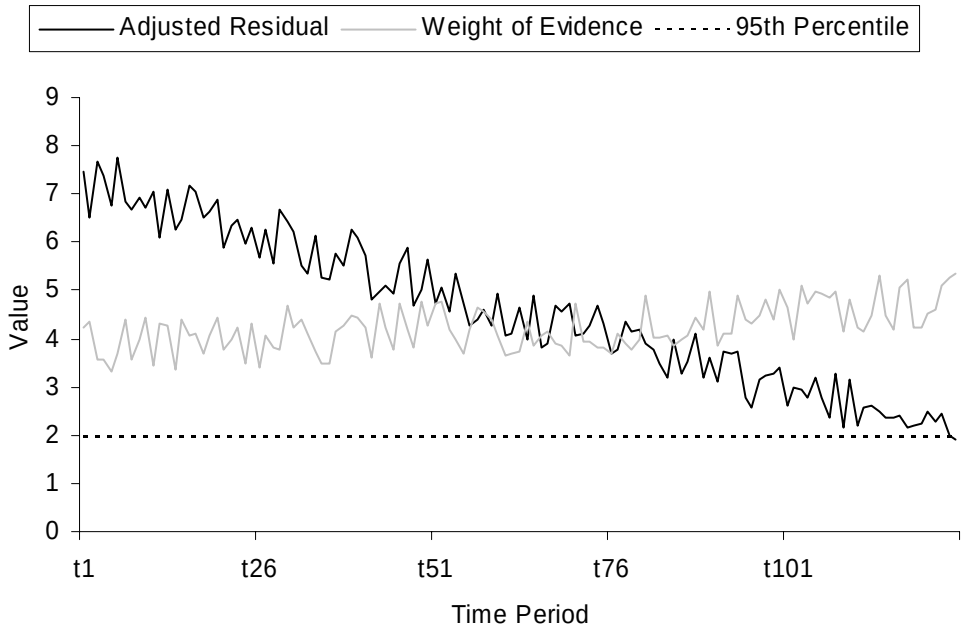
(b) $r_2$.


Adjusted Residual
Weight of Evidence
95th Percentile
18
16
14
12
10
8
6
4
2
0
Value
t1
t26
t51
t76
t101
Time Period

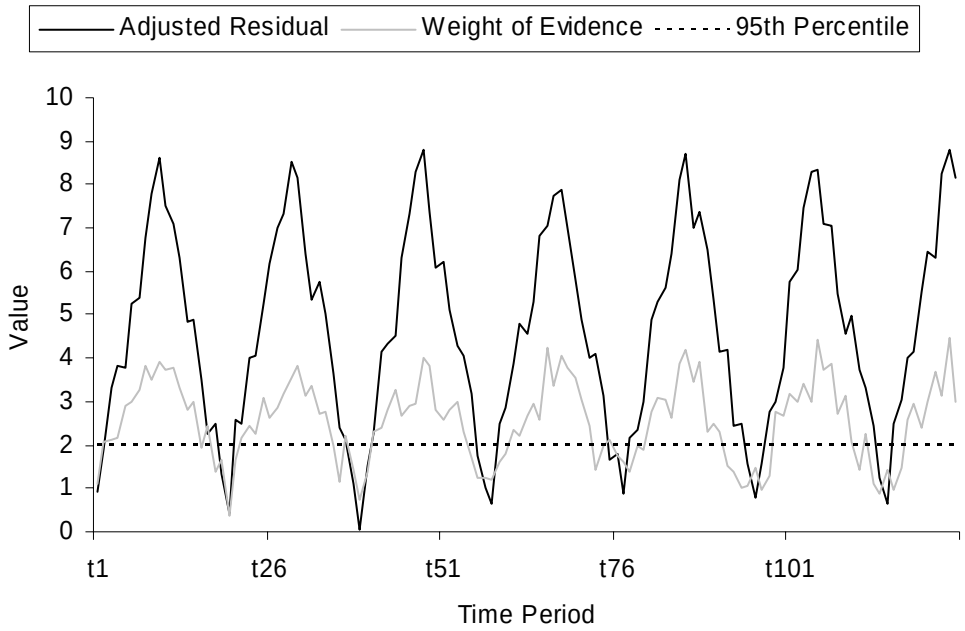
(c) $r_3$.


Adjusted Residual
Weight of Evidence
95th Percentile
Value
Time Period
t1
t26
t51
t76
t101
0
1
2
3
4
5
6
7
8
9

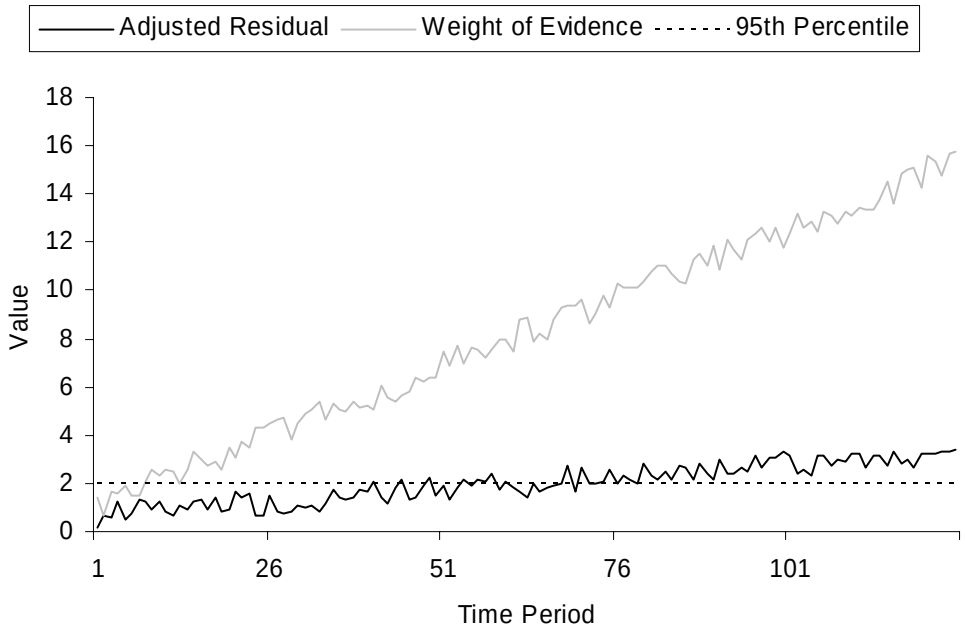
(d) $r_4$.


Adjusted Residual
Weight of Evidence
95th Percentile
Value
Time Period
t1
t26
t51
t76
t101
0
1
2
3
4
5
6
7
8
9

(e) $r_5$.




(f) $r_6$.




(g) $r_7$.

Fig. 36. The changes in $r_1$, …, $r_7$ in the period from $t_1$ to $t_{125}$.

Table 33. Rules $r_1$, …, $r_7$ in $R_{125}$.

| Rule | Adjusted Residual | Weight of Evidence |
|---|---|---|
| $r_1$: $\{i_{93}, i_{94}, i_{99}\} \Rightarrow \{i_7\}$ | 14.53 | 4.22 |
| $r_2$: $\{i_{93}, i_{97}, i_{99}\} \Rightarrow \{i_7\}$ | 11.88 | 6.71 |
| $r_3$: $\{i_{93}, i_{96}, i_{99}\} \Rightarrow \{i_7\}$ | 2.34 | 1.83 |
| $r_4$: $\{i_{93}, i_{98}\} \Rightarrow \{i_7\}$ | 7.02 | 6.69 |
| $r_6$: $\{i_{51}, i_{69}\} \Rightarrow \{i_7\}$ | 8.16 | 2.98 |
| $r_7$: $\{i_{28}, i_{63}\} \Rightarrow \{i_7\}$ | 3.38 | 15.78 |

Each rule in $R_1 \cup \ldots \cup R_{124}$ is associated with a sequence of adjusted residuals and a sequence of weights of evidence. In our experiments, we set the width of the window to 20. By sliding the window across the sequence of adjusted residual, we divided it into a set of subsequences. Similarly, we also divided the sequence of weights of evidence into another set of subsequences.

We defined 20 linguistic variables, $S_1^r, ..., S_{20}^r$, to represent each subsequence of adjusted residuals. $S_{20}^r$ represents "*Change in adjusted residual in next period,*" $S_{19}^r$ represents "*Change in adjusted residual in this period,*" and $S_k^r$, $k \in \{1, \ldots, 18\}$, represents "*Change in adjusted residual in 19 – k period(s) ago.*" The value of $S_k^r$, $k = 1, \ldots, 20$, can take from 5 linguistic terms whose membership functions are defined as follows:

$$\mu_{Highly\ decrease}(x) = \begin{cases} 1 & \text{if } x \le -0.1 \\ \frac{-1}{0.05}(x + 0.05) & \text{if } -0.1 \le x \le -0.05 \\ 0 & \text{otherwise} \end{cases}$$

$$\mu_{Fairly\ decrease}(x) = \begin{cases} \frac{1}{0.05}(x + 0.1) & \text{if } -0.1 \le x \le -0.05 \\ \frac{-1}{0.05}(x) & \text{if } -0.05 \le x \le 0 \\ 0 & \text{otherwise} \end{cases}$$

$$\mu_{More\ or\ less\ the\ same}(x) = \begin{cases} \frac{1}{0.05}(x + 0.05) & \text{if } -0.05 \le x \le 0 \\ \frac{-1}{0.05}(x - 0.05) & \text{if } 0 \le x \le 0.05 \\ 0 & \text{otherwise} \end{cases}$$

$$\mu_{Fairly\ increase}(x) = \begin{cases} \frac{1}{0.05}(x) & \text{if } 0 \le x \le 0.05 \\ \frac{-1}{0.05}(x - 0.1) & \text{if } 0.05 \le x \le 0.1 \\ 0 & \text{otherwise} \end{cases}$$

$$\mu_{Highly\ increase}(x) = \begin{cases} \frac{1}{0.05}(x - 0.05) & \text{if } 0.05 \le x \le 0.1 \\ 1 & \text{if } x \ge 0.1 \\ 0 & \text{otherwise} \end{cases}$$

.

Similarly, we defined 20 linguistic variables, $C_1^r, ..., C_{20}^r$, to represent each subsequence of weight of evidence. $C_{20}^r$ represents "*Change in weight of evidence in next period*," $C_{19}^r$ represents "*Change in weight of evidence in this period*," and $C_k^r$, $k \in \{1, \ldots, 18\}$, represents "*Change in weight of evidence in 19 – k period(s) ago*." The value of $C_k^r$, $k = 1, \ldots, 20$, can take from 5 linguistic terms whose membership functions are defined in the following:

$$\mu_{Highly\ decrease}(x) = \begin{cases} 1 & \text{if } x \le -0.1 \\ \frac{-1}{0.05}(x + 0.05) & \text{if } -0.1 \le x \le -0.05 \\ 0 & \text{otherwise} \end{cases}$$

$$\mu_{Fairly\ decrease}(x) = \begin{cases} \frac{1}{0.05}(x + 0.1) & \text{if } -0.1 \le x \le -0.05 \\ \frac{-1}{0.05}(x) & \text{if } -0.05 \le x \le 0 \\ 0 & \text{otherwise} \end{cases}$$

$$\mu_{More\ or\ less\ the\ same}(x) = \begin{cases} \frac{1}{0.05}(x + 0.05) & \text{if } -0.05 \le x \le 0 \\ \frac{-1}{0.05}(x - 0.05) & \text{if } 0 \le x \le 0.05 \\ 0 & \text{otherwise} \end{cases}$$

$$\mu_{Fairly\ increase}(x) = \begin{cases} \frac{1}{0.05}(x) & \text{if } 0 \le x \le 0.05 \\ \frac{-1}{0.05}(x - 0.1) & \text{if } 0.05 \le x \le 0.1 \\ 0 & \text{otherwise} \end{cases}$$

$$\mu_{Highly\ increase}(x) = \begin{cases} \frac{1}{0.05}(x - 0.05) & \text{if } 0.05 \le x \le 0.1 \\ 1 & \text{if } x \ge 0.1 \\ 0 & \text{otherwise} \end{cases}$$

.

Each subsequence was then converted to a set of ordered triples. After that, we applied our proposed algorithm to these ordered triples to discover meta-rules. The discovered meta-rules were then used to predict how the adjusted residuals and the weights of evidence of the rules would change in $t_{125}$. The predicted rules were stored in $\hat{R}_{125}$ (Table 34).

Table 34. Rules $r_1$, …, $r_7$ in $\hat{R}_{125}$.

| Rule | Adjusted Residual | Weight of Evidence |
|---|---|---|
| $r_1$: $\{i_{93}, i_{94}, i_{99}\} \Rightarrow \{i_7\}$ | 14.47 | 4.22 |
| $r_2$: $\{i_{93}, i_{97}, i_{99}\} \Rightarrow \{i_7\}$ | 12.33 | 6.74 |
| $r_3$: $\{i_{93}, i_{96}, i_{99}\} \Rightarrow \{i_7\}$ | 2.34 | 1.83 |
| $r_4$: $\{i_{93}, i_{98}\} \Rightarrow \{i_7\}$ | 7.07 | 6.73 |
| $r_6$: $\{i_{51}, i_{69}\} \Rightarrow \{i_7\}$ | 7.85 | 3.00 |
| $r_7$: $\{i_{28}, i_{63}\} \Rightarrow \{i_7\}$ | 3.37 | 15.70 |

It is important to note that $r_5$ is perished in $t_{125}$ and it is therefore not found in $\hat{R}_{125}$. Our algorithm is able to predict the changed rules (i.e., $r_1$, …, $r_4$), the perished rule (i.e., $r_5$), the added rules (i.e., $r_6$ and $r_7$) in $t_{125}$. The difference of the actual rules in $R_{125}$ and the predicted rules in $\hat{R}_{125}$ is shown in Fig. 37.

In the rest of this section, we present some of the meta-rules discovered by our algorithm. A meta-rule of adjusted residual for $r_1$ discovered is given in the following:

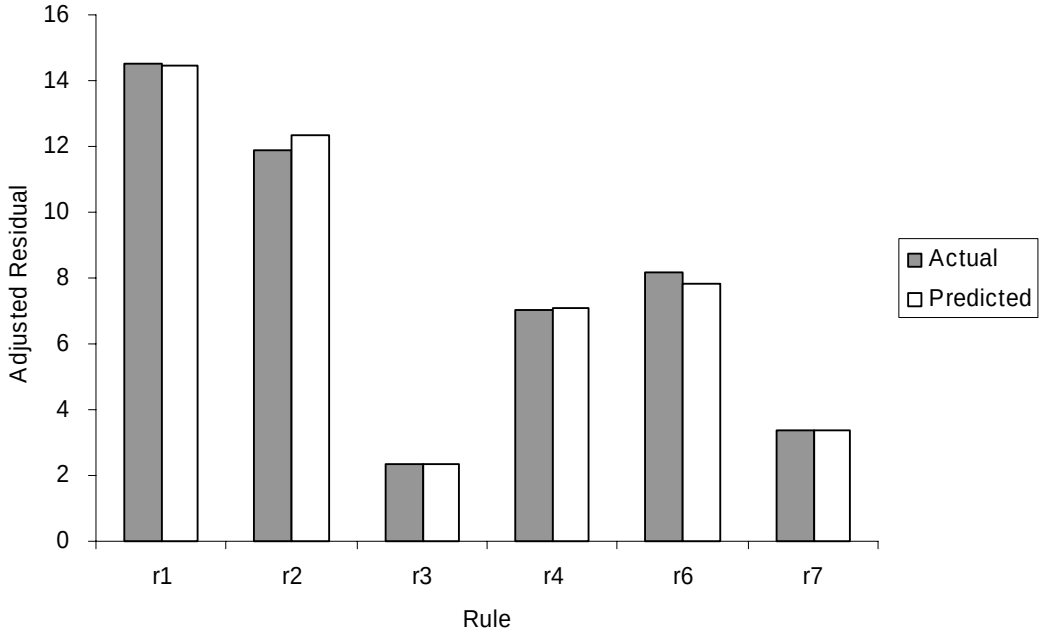
*Change in adjusted residual in this period = Fairly increase*
$\wedge$ *Change in adjusted residual in 5 periods ago = Highly increase*
$\Rightarrow$ *Change in adjusted residual in next period = Fairly decrease* [$w$ = 3.89].

This meta-rule states that "if the change in adjusted residual in this period fairly increases and the change in adjusted residual in 5 periods ago highly increases, then the change in adjusted residual in next period would fairly decrease."

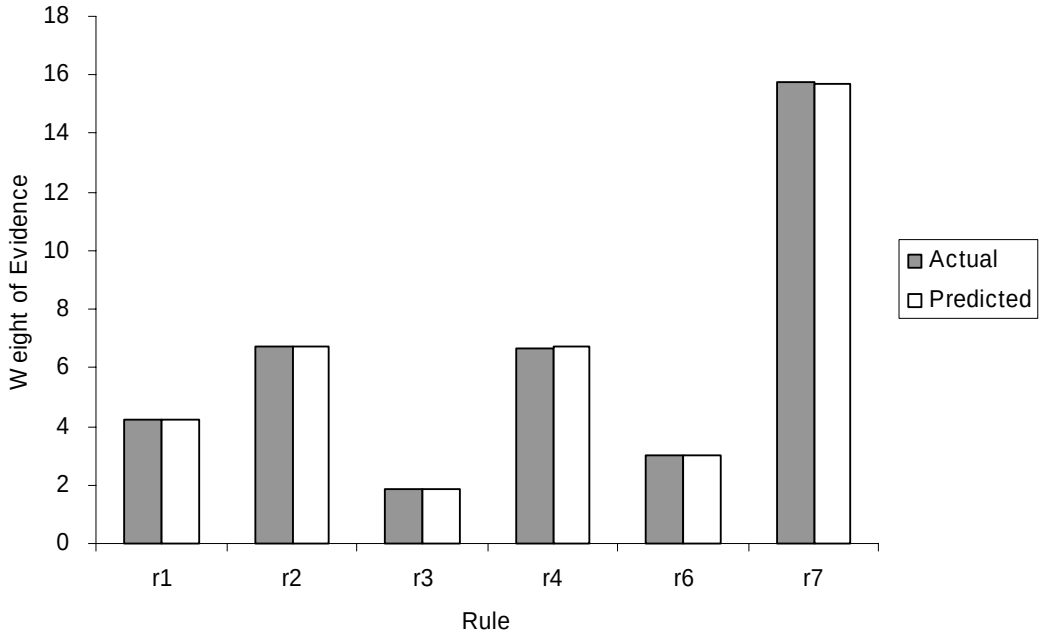
Another meta-rule of weight of evidence for $r_7$ mined from the rule sets is provided as follows:

*Change in weight of evidence in this period = More or less the same*
$\wedge$ *Change in weight of evidence in 12 periods ago = Highly increase*
$\Rightarrow$ *Change in weight of evidence in next period = Highly decrease* [$w$ = 2.15].

This meta-rule states that "if the change in weight of evidence in this period is more or less the same and the change in weight of evidence in 12 periods ago highly increases, then the change in weight of evidence in next period would highly decrease."




(a) The adjusted residual of the actual and predicted rules.




(b) The weight of evidence of the actual and predicted rules.

Fig. 37. The actual and predicted rules.

## 9.2 The Property-Valuation Database

The *property-valuation* database is extracted from the data warehouse maintained by the Hong Kong office of a worldwide property valuation company. It contains two relational tables: PROPERTY and TRANSACTION. Each tuple in the PROPERTY table represents a residential property, which is characterized by, e.g., direction, size of property, number of bed rooms, etc., in Hong Kong, whereas each tuple in the TRANSACTION table represents a buy/sell transaction, which is characterized by date of transaction, transaction amount, mortgage ratio, etc., concerned with a tuple in the PROPERTY table (i.e., a residential property). The PROPERTY and the TRANSACTION tables consist of the characteristics of 765,106 residential properties in 59 districts in Hong Kong and 909,226 transactions completed during the period between 1991 and 2001, respectively.

Fig. 38 shows the schema of the *property-valuation* database. Since each relation in the *property-valuation* database contains many attributes, we only show a subset of these attributes in Fig. 38.

PROPERTY (PROPERTY_ID, DIRECTION, SIZE, NUM_OF_BED_ROOMS, …)
TRANSACTION (TID, PROPERTY_ID, TRANS_DATE, AMOUNT, …)

Fig. 38. Schema of the *property-valuation* database.

In the *property-valuation* database, PROPERTY contains data for 765,106 residential properties in 59 districts in Hong Kong, whereas TRANSACTION consists of data for 909,226 transactions completed during the period between 1991 and 2001. Each property in the former table had been sold or bought in one or more transaction maintained in the latter. The transaction amount in the *property-valuation* database is about HK$2,450 billion in total. Table 35 gives a summary of the property-valuation database.

Table 35. Summary of the *property-valuation* database.

| Relation | No. of Attributes | No. of Tuples |
|---|---|---|
| PROPERTY | 28 | 765,106 |
| TRANSACTION | 13 | 909,226 |

### *9.2.1 The Transformation Functions Defined*

In this section, we describe how we can construct a transformed relation, R (T_BUILD_AGE, T_AMOUNT, T_NATIONALITY, …), using the transformation functions (defined in Chapter 8). To obtain the transformed relation, we, together with a domain expert from the property evaluation company, defined 14 transformation functions in total. From the 14

transformation functions, in this section, we present three of them as an illustration.

Let us consider the attributes PROPERTY[BUILD_DATE] and TRANSACTION[TRANS_DATE]. The former represents the date on which the residential property was built, whereas the latter represents the transaction date. The difference in these two attributes gives the age of the property when the transaction was made. We defined the following transformation function:

$$f_1(tid) = \pi_{\text{TRANS\_DATE} - \text{BUILD\_DATE}} (\sigma_{\text{TID} = \text{tid}} (\text{TRANSACTION} \bowtie \text{PROPERTY})),$$

where $\sigma$, $\pi$, and $\bowtie$ denote the SELECT, PROJECT, and NATURAL JOIN operations from relational algebra. This function is an example of the arithmetic functions defined in Chapter 8. The transformed attribute T_BUILD_AGE was produced by applying $f_1$(TRANSACTION[TID]) to every tuple in TRANSACTION.

The transaction amount can be partitioned into a finite number of intervals for the purpose of discovering more meaningful rules. After we consulted the domain expert, we defined another transformation function as follows:

$$f_2(amount) = \begin{cases} 0 & \text{if } amount < 1{,}000{,}000 \\ 1 & \text{if } 1{,}000{,}000 \le amount < 2{,}000{,}000 \\ 2 & \text{if } 2{,}000{,}000 \le amount < 3{,}000{,}000 \\ 3 & \text{if } 3{,}000{,}000 \le amount < 4{,}000{,}000 \\ 4 & \text{if } 4{,}000{,}000 \le amount < 5{,}000{,}000 \\ 5 & \text{if } 5{,}000{,}000 \le amount < 6{,}000{,}000 \\ 6 & \text{if } amount \ge 6{,}000{,}000 \end{cases}.$$

The transformed attribute T_AMOUNT was produced by applying $f_2$(TRANSACTION[AMOUNT]) to every tuple in TRANSACTION, which is an example of the discretization function defined in Chapter 8.

The domain expert suggested that whether there exist or do not exist any bay windows rather than the size of bay windows would be used in our analysis. We therefore made use of a transformation function defined as:

$$f_3(size) = \begin{cases} Y & \text{if } size > 0 \\ N & \text{otherwise} \end{cases}.$$

This function is an example of the logical functions defined in Chapter 8. The transformed attribute T_BAY_WINDOWS was produced by applying $f_3$(PROPERTY[SIZE_OF_BAY_WINDOWS]) to every tuple in PROPERTY.

By applying the transformation functions to the *property-valuation* database, we obtained the required transformed relation. There are 14 attributes in the transformed relation. Among the 14 transformed attributes, 8 are categorical (discrete-valued) and 6 are quantitative (continuous-valued). Instead of performing data mining on the original data, we discovered interesting associations from the transformed data.

### 9.2.2 Fuzzy Sets Resulted from Fuzzy Partitioning

After data transformation, we applied our fuzzy partitioning algorithm ITFP to the transformed data. It is used to generate fuzzy sets automatically to represent each of the 6 continuous, transformed attributes. We present some of the generated fuzzy sets in this section.

For example, ITFP generated 9 fuzzy sets for the transformed attribute T_SIZE, which represents the size of a residential property. Fig. 39 shows the generated fuzzy sets.

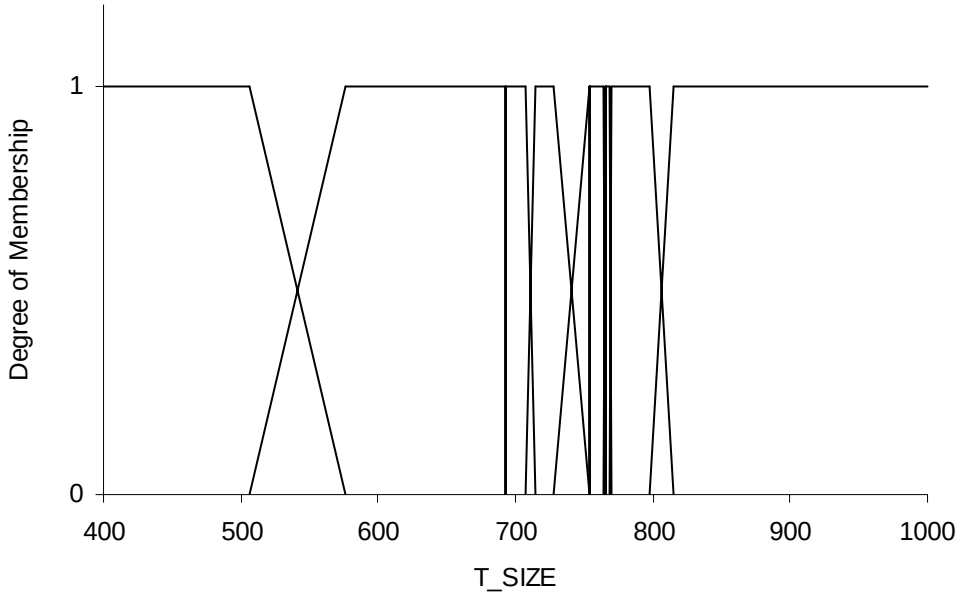



Fig. 39. Fuzzy sets for T_SIZE.

As another example, 4 fuzzy sets were generated for the transformed attribute T_FLOOR, which represents the floor of a property. They are given in Fig. 40.

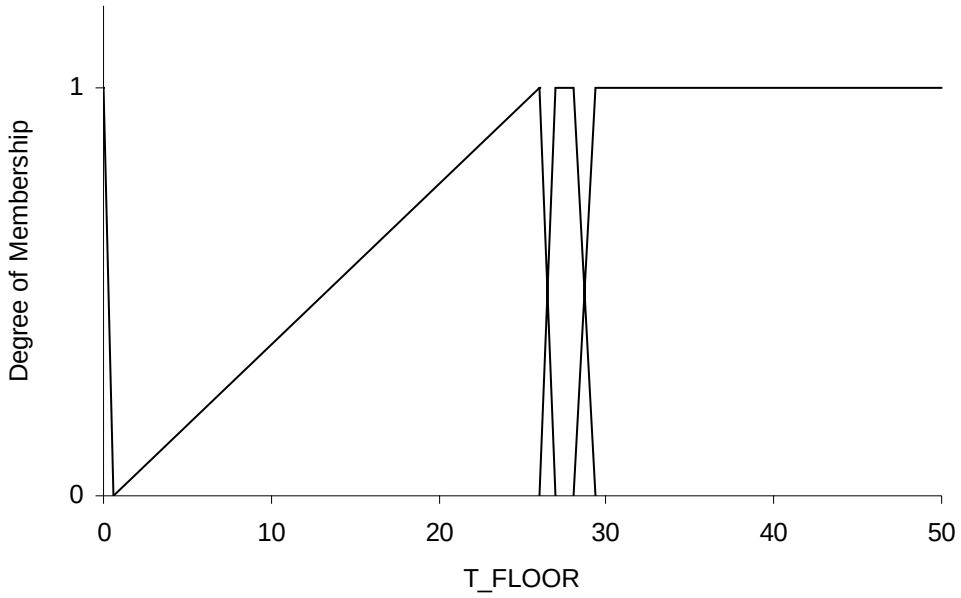



Fig. 40. Fuzzy sets for T_FLOOR.

### 9.2.3 Attribute Clustering for Grouping and Selection of Attributes

We next applied our attribute clustering algorithm, ACA, to the transformed and fuzzy partitioned data to find clusters of attributes. Fig. 41 shows the sum of the interdependence redundancy measure over all the clusters versus the number of clusters found.

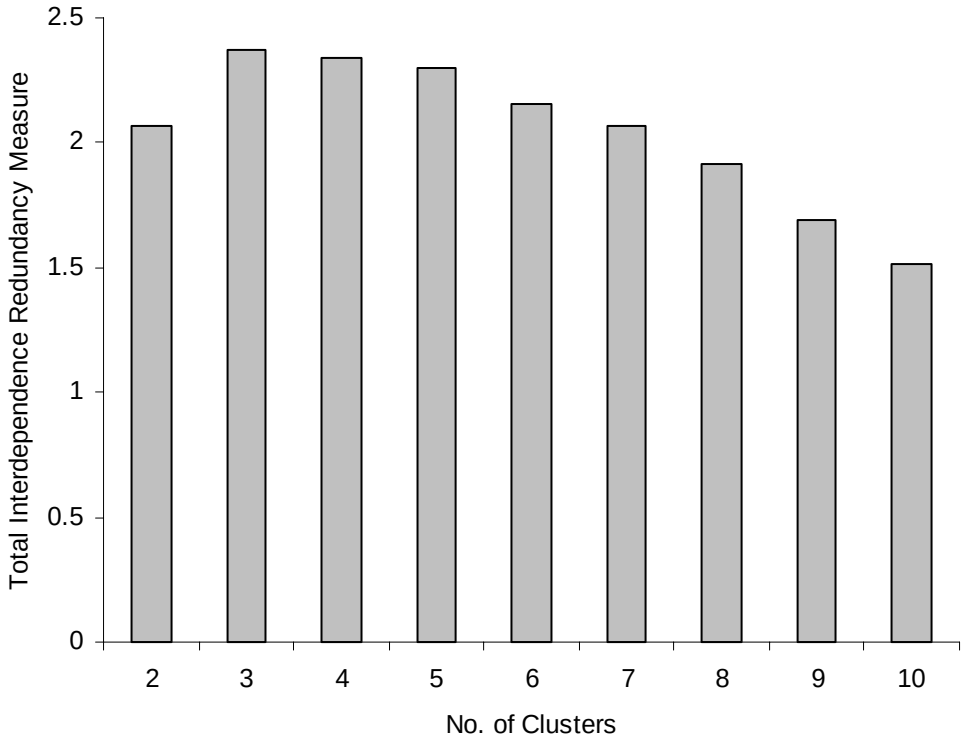



Fig. 41. The total interdependence redundancy measure over all the clusters found in the transformed relation.

As shown in Fig. 41, it finds that the optimal number of clusters is 3. ACA identifies 3 clusters of attributes, where T_CLUB_HOUSE, which represents whether there is a club house, T_NUM_OF_LIVING_ROOMS, which represents the number of living rooms, and T_BUILD_AGE, which represents the building age, are the modes. These clusters have 5, 3, and 6 attributes, respectively.

From each of the three attribute clusters, we select the top 3 attributes for data mining and meta-mining.

### *9.2.4 Mining Meta-Rules*

After the original data are transformed, the transformed data are fuzzy partitioned, and the transformed attributes are grouped and selected, we applied our fuzzy rule mining algorithms, FARM and EFARM, to discover meta-rules. Since the results they obtained are more or less the same, in this section we report only the findings of FARM.

#### 9.2.4.1 Regular and Differential Meta-Rules

For our experimentation, the domain expert identified three districts of interest: Yuen Long, Sheung Shui, and Tseung Kwan O. He was interested in association relationships

concerned with the amount of the residential properties in these districts. We first extracted the transformed data in 2001 concerned with these three districts into three data sets, one for each district. We then applied our algorithm to mine a set of rules in each data set. Next, we applied them to mine a set of regular and differential meta-rules from the rule sets.

A regular meta-rule the domain expert found being meaningful is given as follows:

$$\textit{Swimming Pool} = \textit{No} \Rightarrow \textit{Amount} \in [0, 1\,000\,000)\ [w = 7.63].$$

The meta-rule states that "in general, a residential property is worth less than 1 million dollars if there is no swimming pool." This represents an association relationship in common in the characteristics of properties in these districts. The domain expert found this meta-rule being meaningful because many people enjoy using recreational facilities (e.g., swimming pools, club houses, etc.) in their properties and hence they will not pay much money for a property if there is no swimming pool.

The domain expert also found the following regular meta-rules being meaningful:

$$\textit{Direction} = \textit{South} \Rightarrow \textit{Amount} \in [4\,000\,000, 5\,000\,000)\ [w = 4.21]$$
$$\textit{Direction} = \textit{South} \Rightarrow \textit{Amount} \in [5\,000\,000, 6\,000\,000)\ [w = 3.95].$$

They state that "in general, a residential property is worth between 4 million and 6 million dollars if it faces south." Again, it represents an association in common in the characteristics of properties in the three districts. The domain expert found them being meaningful because many Chinese prefer properties that face south and are willing to pay more for them.

Another regular meta-rule found being meaningful by the domain expert is provided in the following:

$$\textit{Estate} = \textit{No} \Rightarrow \textit{Amount} \in [0, 1\,000\,000)\ [w = 0.65].$$

This states that "in general, a residential property is worth less than 1 million dollars if it is not in any estate." This meta-rule is meaningful because the properties in an estate are usually better managed than those not in an estate. It is for this reason that many people will not pay much for properties that are not in an estate.

In addition to regular meta-rules, the domain expert also found the discovered

differential meta-rules interesting. A differential meta-rule found interesting by the domain expert is:

$$\textit{Swimming Pool} = \textit{Yes} \wedge \textit{Estate} = \textit{No} \Rightarrow \textit{Amount} \in [1\,000\,000, 2\,000\,000)\ [w = -0.18].$$

This states that "in an exceptional manner, a residential property is worth between 1 million and 2 million dollars if there is one or more swimming pool and it is not in any estate." This represents a distinguishing association relationship in the characteristics of properties in Yuen Long only. The finding of this differential meta-rule surprises us since it, together with the last regular meta-rule, indicate that properties with swimming pool(s) but not in any estate in Yuen Long are worth more than those in Sheung Shui and Tseung Kwan O. Although the domain expert had not recognized this relationship, he found it to be meaningful because many properties in Yuen Long lack recreational facilities, such as swimming pools, and people in Yuen Long are willing to pay more for a property with one than people in other districts.

Another differential meta-rule the domain expert found interesting is:

$$\textit{Size} = \int_{0}^{506} \frac{1}{x} + \int_{506}^{576.9} \frac{(506 - x)/70.9}{x} \Rightarrow \textit{Amount} \in [1\,000\,000, 2\,000\,000)\ [w = -1.12],$$

where $\int_{0}^{506} \frac{1}{x} + \int_{506}^{576.9} \frac{(506 - x)/70.9}{x}$ denotes the leftmost fuzzy set shown in Fig. 39. This states that "in an exceptional manner, a residential property is worth between 1 million and 2 million dollars if its size is small." This represents a distinctive relationship in the characteristics of properties in Sheung Shui only. Together with the following regular meta-rule:

$$\textit{Size} = \int_{0}^{506} \frac{1}{x} + \int_{506}^{576.9} \frac{(506 - x)/70.9}{x} \Rightarrow \textit{Amount} \in [0, 1\,000\,000)\ [w = 2.90],$$

which states that "in general, a property is worth less than 1 million dollars if its size is small," the domain expert found the differential meta-rule to be meaningful because it confirms that the properties in Sheung Shui are usually worth more than those in Yuen Long and Tseung Kwan O.

### 9.2.4.2 Change Meta-Rules

For our further experimentation, we selected the transformed data concerned with residential properties in Yuen Long during the period from 1991 to 2001. The average price per square foot of these properties in this period is given in Fig. 42. As shown in Fig. 42, the average price fluctuates significantly and hence the rules discovered in the historical data are unable to provide an accurate prediction of the price of properties in the future.

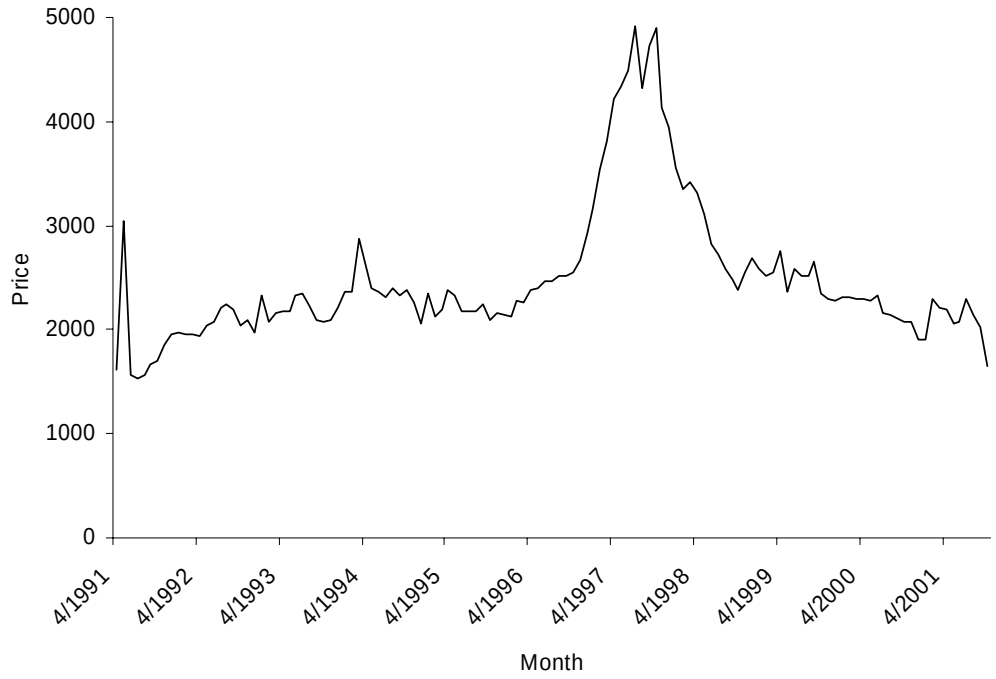



Fig. 42. The average price per square foot of residential properties in Yuen Long during the period from 1991 to 2001.

The domain expert from the company aimed at predicting the amount of a property based on other attributes. To perform this task, we first divided the database into 127 partitions, $T_1$, …, $T_{127}$, where $T_1$ contains the buy/sell transactions in April 1991, $T_2$ contains the buy/sell transactions in May 1991, and so on. We next made use of our algorithm to discover 126 sets of fuzzy association rules, $R_1$, …, $R_{126}$, from the first 126 database partitions, $T_1$, …, $T_{126}$. Finally, we applied our fuzzy rule mining algorithm to discover a set of change meta-rules, which represent the regularities about the changes in the adjusted residual and weight of evidence of each fuzzy rule in $R_1 \cup \ldots \cup R_{126}$.

Using the change meta-rules, we predicted how the adjusted residual and weight of evidence of each fuzzy rule in $R_1 \cup \ldots \cup R_{126}$ would change in October 2001. This resulted

in a set of fuzzy association rules, $R'_{127}$, such that the adjusted residual and weight of evidence of each rule in $R'_{127}$ was predicted based on the changes in the fuzzy rules discovered in the time period from April 1991 to September 2001 (i.e., the discovered change meta-rules).

In our experiments, we set the width of the sliding window to 20. The average percentage error of the adjusted residuals and weights of evidence of the rules predicted using the change meta-rules is given in Table 36.

Table 36. The average percentage error of the adjusted residuals and weights of evidence of the rules predicted using change meta-rules.

| | **Percentage Error** |
|---|---|
| **Adjusted residual** | 0.36% |
| **Weight of evidence** | 1.01% |

In addition to the above, we predicted the amount of each record in the last database partition, $T_{127}$, using $R'_{127}$. To further evaluate the performance of our approach, we used the fuzzy rules discovered in $T_1 \cup \ldots \cup T_{126}$ to predict the amount of each record in $T_{127}$. We denote these fuzzy rules as $R$. For the purpose of comparison, we also applied C4.5, a well-known decision tree classifier, to $T_1 \cup \ldots \cup T_{126}$ for training and to $T_{127}$ for testing.

The experimental results are given in Table 37 ("Fuzzy Rules + Meta-Rules" denotes the classification rate yielded based on $R'_{127}$ and "Fuzzy Rules" denotes the classification rate yielded based on $R$).

Table 37. Experimental results on prediction of property amount.

| | **Percentage Accuracy** |
|---|---|
| **C4.5** | 85.7% |
| **Fuzzy Rules** | 83.3% |
| **Fuzzy Rules + Meta-Rules** | 88.7% |

As shown in Table 37, the rule set produced using change meta-rules (i.e., $R'_{127}$) obtains better accuracy than the rule set discovered in $T_1 \cup \ldots \cup T_{126}$ when they were used to predict the amount of the records in $T_{127}$ collected in October 2001. The experimental results show that our approach is able to improve the performance of a data mining

algorithm by discovering and predicting the changes in rules. Our approach to mining fuzzy rules and change meta-rules also outperforms C4.5.

We also repeated our experimentation on the properties in Sheung Shui and Tseung Kwan O. The experimental results show that our approach to mining fuzzy rules and change meta-rules produces good classification rate. We report in this section only the results on Yuen Long.

## 9.3 The Stock-Price Database

The *stock-price* database contains the stock prices of three companies listed in the Stock Exchange of Hong Kong during the period from 2000 to 2004. The companies are in different industries. Specifically, they are Hang Seng Bank Ltd., a major bank in Hong Kong, Sun Hung Kai Properties Ltd., a major property developer in Hong Kong, and CLP Holdings Ltd., a major electricity supplier in Hong Kong. Fig. 43 gives the schema of the *stock-price* database. Each tuple in the STOCK table, the DIVIDEND table, the SHARES_ISSUED table, and the PRICE table represents a listed company, the dividend paid by a company, the shares issued by a company, the price of a company, respectively. Since each relation in the *stock-price* database consists of many attributes, only a subset of these attributes is given in Fig. 43.

STOCK (STOCK_CODE, NAME, LISTING_DATE, FISCAL_MONTH, …)
DIVIDEND (STOCK_CODE, DATE, AMOUNT_PER_SHARE, …)
SHARES_ISSUED (STOCK_CODE, DATE, NUM_OF_SHARES_ISSUED, …)
PRICE (STOCK_CODE, DATE, CLOSE_PRICE, VOLUME, …)

Fig. 43. The schema of the *stock-price* database.

Fig. 44 shows the stock prices of the three companies during the period between 2000 and 2004. As shown in Fig. 44, the stock prices of Hang Seng Bank Ltd. and Sun Hung Kai Properties Ltd. fluctuate more significantly than CLP Holdings Ltd.

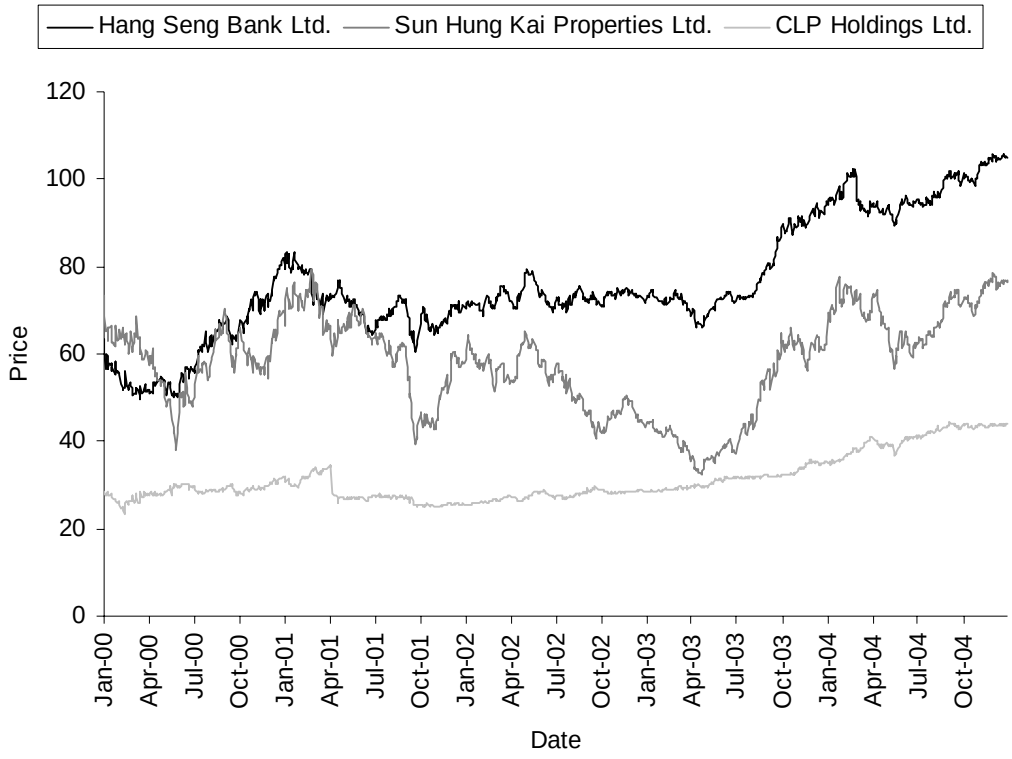



Fig. 44. The stock prices of the three companies during the period from 2000 to 2004.

### 9.3.1 The Transformation Functions Defined

In this section, we describe how to construct a transformed relation for the *stock-price* database using the transformation function introduced in Chapter 8.

Let us consider the attribute PRICE[CLOSE_PRICE]. It represents the close price of a company's stock on a specific date. Since the company may pay dividend and may issue additional shares, the close price has to be adjusted accordingly in order to reflect the actual trade price. We therefore defined the following transformation functions for the adjustment:

$$f_4(scode, date) = 1 + \frac{\sum_{t \in \sigma_{\text{STOCK\_CODE}=scode \wedge \text{DATE}=date}(\text{SHARES\_ISSUED})} t[\text{NUM\_OF\_SHARES\_ISSUED}]}{\sum_{t \in \sigma_{\text{STOCK\_CODE}=scode \wedge \text{DATE}<date}(\text{SHARES\_ISSUED})} t[\text{NUM\_OF\_SHARES\_ISSUED}]}$$

and

$$f_5(scode, date) = (1 / f_4(scode, date)) \times \pi_{\text{CLOSE\_PRICE} - \text{AMOUNT\_PER\_SHARE}}$$
$$(\sigma_{\text{STOCK\_CODE} = \text{scode} \wedge \text{DATE} = \text{date}} (\text{PRICE} \bowtie \text{DIVIDEND})),$$

where $\bowtie$ denotes the LEFT OUTER JOIN operation from relational algebra. The transformation functions $f_4$ and $f_5$ are examples of the arithmetic functions defined in Chapter 8.

Since we are interested in the stock price movements, we defined the following transformation function for computing the percentage change in price:

$$f_6(scode, date) = \frac{f_5(scode, date) - f_5(scode, date-1)}{f_5(scode, date-1)}.$$

This function is also an example of the arithmetic functions. The transformed attribute T_PERCENT_CHANGE was produced by applying $f_6$(PRICE[STOCK_CODE], PRICE[DATE]) to every tuple in PRICE.

We then constructed the transformed relation R (STOCK_CODE, DATE, T_PERCENT_CHANGE) for discovering the similarity, difference, and change in stock price movements.

We next produced a time series using the following operation:

$$transpose(\pi_{\text{T\_PERCENT\_CHANGE}}(\sigma_{\text{STOCK\_CODE} = \text{scode}}(\text{R}))),$$

where *transpose*(T) returns the transpose of relation T. The time series was then divided into a set of subsequences by sliding a window of width $w$ = 20. The subsequences are stored in the relational database for the ease of retrieval. We therefore obtained a relation, S (T_PERCENT_CHANGE$_1$, …, T_PERCENT_CHANGE$_{20}$). The attribute T_PERCENT_CHANGE$_{20}$ represents the percentage change on the next transaction date, the attribute T_PERCENT_CHANGE$_{19}$ represents the percentage change on the present transaction date, the attribute T_PERCENT_CHANGE$_{18}$ represents the percentage change on the last transaction date, and the attribute T_PERCENT_CHANGE$_i$ represents the percentage change on the (19 – $i$)-th to the last transaction date for $i$ = 1, …, 17.

### 9.3.2 Fuzzy Sets Resulted from Fuzzy Partitioning

After data transformation, we applied our fuzzy partitioning technique ITFP proposed in Chapter 6 to the transformed relation R. It was used to generate fuzzy sets to represent the transformed attribute T_PERCENT_CHANGE. Fig. 45 shows the generated fuzzy sets.

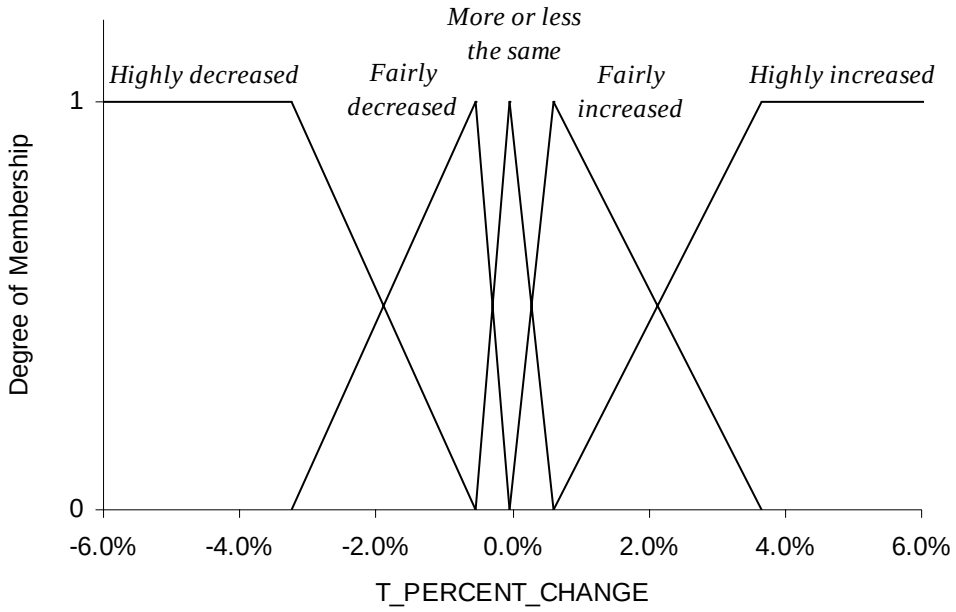



Fig. 45. Fuzzy sets for T_PERCENT_CHANGE.

The generated fuzzy sets were then used to represent the domain of each of $T\_PERCENT\_CHANGE_1$, …, $T\_PERCENT\_CHANGE_{20}$ in S.

### 9.3.4 Attribute Clustering for Grouping and Selection of Attributes

We next applied our attribute clustering algorithm ACA proposed in Chapter 7 to S. Fig. 46 shows the sum of the interdependence redundancy measure over all the clusters versus the number of clusters found.

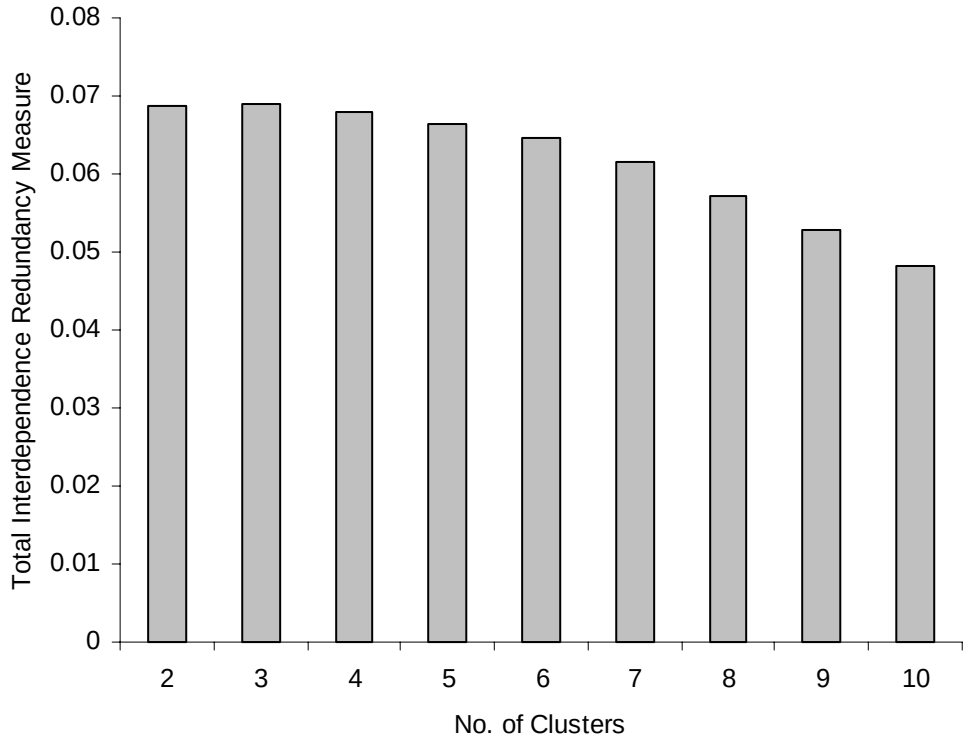



Fig. 46. Total interdependence redundancy measure over all the clusters found in the transformed relation.

As shown in Fig. 46, it finds that the optimal number of clusters is 3. ACA identifies 3 clusters of attributes, where $T\_PERCENT\_CHANGE_2$, $T\_PERCENT\_CHANGE_7$, and $T\_PERCENT\_CHANGE_{16}$ are the modes. These clusters have 8, 6, and 6 attributes, respectively. From each of the three attribute clusters, we select the top 3 attributes for data mining and meta-mining.

### *9.3.5 Mining Meta-Rules*

Similar to our experimentation on the *property-valuation* database, we next applied our fuzzy rule mining algorithms, FARM and EFARM, to the transformed data to discover meta-rules. Since the results they obtained are more or less the same, we report only the findings of FARM in this section for clarity. We first present the discovered regular and differential meta-rules in Section 9.3.5.1. We next report how change meta-rules lead to an accurate prediction of the stock price in Section 9.3.5.2.

#### 9.3.5.1 Regular and Differential Meta-Rules

For our experimentation, we aimed at finding the association relationships concerned with how the percentage change in stock price on a specific date is affected by percentage

changes in the past few days. Fig. 47 shows the value of T_PERCENT_CHANGE (i.e., the percentage change in stock price) of the three companies.

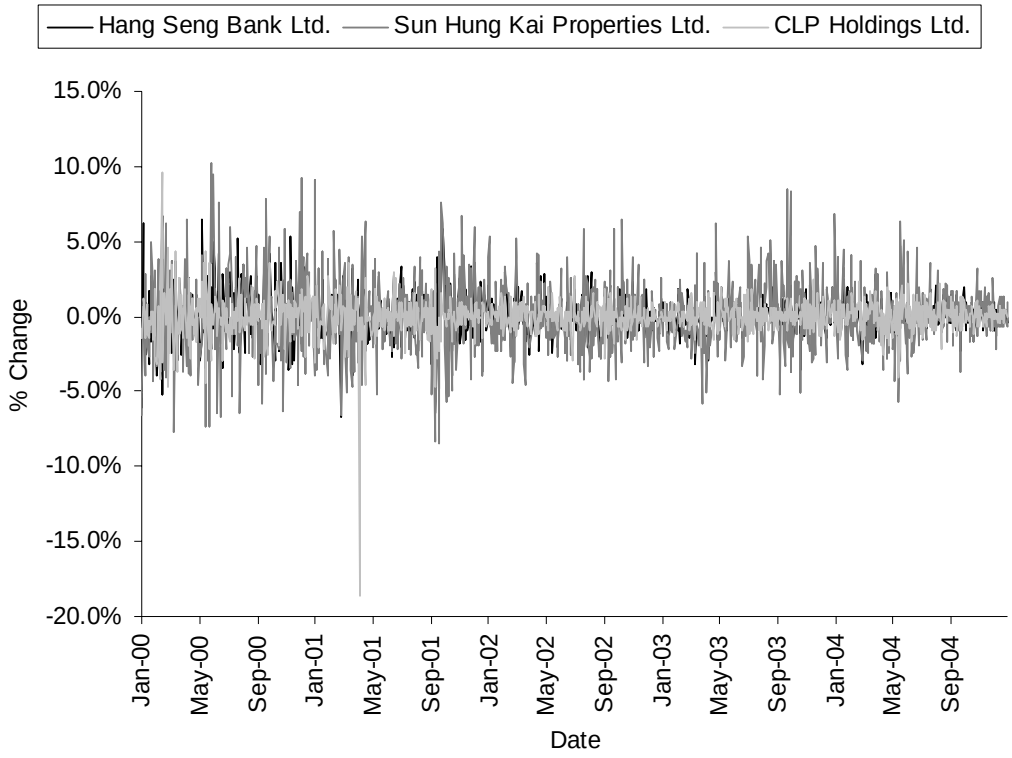



Fig. 47. Percentage change in stock prices of the three companies during the period from 2000 to 2004.

We first extracted the subsequence data concerned with the three companies into three data sets, one for each company. We then applied our algorithm to mine a set of rules in each data set. Next, we applied it to mine a set of regular and differential meta-rules from the rule set.

A regular meta-rule discovered is given as follows:

*Percentage change on the 14$^{th}$ to the last transaction date = More or less the same* $\wedge$
*Percentage change on the 12$^{th}$ to the last transaction date = Fairly increased*
$\Rightarrow$ *Percentage change on the next transaction date = More or less the same*
[*w = infinity*].

This meta-rule states that "in general, if the stock prices on the 14$^{th}$ and 12$^{th}$ to the last transaction date are more or less the same and fairly increased, respectively, then the stock price on the next transaction date will be more or less the same." This represents an

association relationship in common in the characteristics of the stock price movements of the three companies.

The following regular meta-rules are also discovered:

*Percentage change on the present transaction date = Highly increased*
$\Rightarrow$ *Percentage change on the next transaction date = Highly decreased*
[$w$ = 5.57]
*Percentage change on the present transaction date = Highly increased*
$\Rightarrow$ *Percentage change on the next transaction date = Fairly decreased*
[$w$ = 3.74].

These state that "in general, if the stock price highly increases on this transaction date, then the stock price will fairly or highly decrease on the next transaction date." Again, they represent association relationships in common in the characteristics of the stock price movements of the three companies. They suggest that it is a safe bet for the decrease in the stock price on the next transaction date if one finds that the stock price highly increases on the present transaction date.

In addition to regular meta-rules, our algorithm also discovered the following differential meta-rule:

*Percentage change on the 13$^{th}$ to the last transaction date = More or less the same*
$\Rightarrow$ *Percentage change on the next transaction date = More or less the same*
[$w$ = –4.70].

This states that "in an exceptional manner, if the stock price on the 13$^{th}$ to the last transaction date is more or less the same, then the stock price on the next transaction date will be more or less the same." This represents a distinguishing association relationship in the characteristics of the stock price movement of CLP Holdings Ltd. only.

### 9.3.5.2 Change Meta-Rules

For our further experimentation, we extracted the transformed time-series data for Hang Seng Bank Ltd. during the period from 2000 to 2004. As shown in Figs. 45 and 47, the stock price of the company fluctuates significantly and hence the rules discovered in the historical data are unable to provide an accurate prediction of the stock price in the future.

We aimed at predicting the stock price on the next transaction date given the stock

price on the previous 19 days. To perform this task, we first divided the subsequence data into 20 partitions, $T_1$, …, $T_{20}$, where $T_1$ contains the subsequences representing the stock price in the period from January 2000 to March 2000, $T_2$ contains the subsequences representing the stock price in the period from April 2000 to June 2000, and so forth. We next used our algorithm to discover 19 sets of fuzzy association rules, $R_1$, …, $R_{19}$, from the first 19 database partitions, $T_1$, …, $T_{19}$. Finally, we applied our fuzzy rule mining algorithm to discover a set of change meta-rules, which represent the regularities about the changes in the adjusted residual and weight of evidence of each fuzzy rule in $R_1 \cup \ldots \cup R_{19}$.

Using the change meta-rules, we predicted how the adjusted residual and weight of evidence of each fuzzy rule in $R_1 \cup \ldots \cup R_{19}$ would change in the period between October 2004 and December 2004. This resulted in a set of fuzzy association rules, $R'_{20}$, such that the adjusted residual and weight of evidence of each rule in $R'_{20}$ were predicted based on the change meta-rules discovered. We predicted the stock price of each subsequence in the last database partition, $T_{20}$, using $R'_{20}$.

To further evaluate the performance of our approach, we used the fuzzy rules discovered in $T_1 \cup \ldots \cup T_{19}$ to predict the stock price. We denote these discovered fuzzy rules as $R$. For the purpose of comparison, neural networks were also applied to the data because they are the most popular classifier for financial time series forecasting [Baestaens, van den Bergh, and Wood 1994; Refenes, Burgess, and Bentz 1997; Weigend, Huberman, and Rumelhart 1990]. The neural networks used in our experiments are multilayer perceptrons with a single hidden layer, which contains 20 nodes, and they were trained by the backpropagation algorithm [Rumelhart, Hinton, and Williams 1986; Werbos 1974] with the learning rate was set to 0.3 and the momentum term was set to 0.7. The neural networks used the first 4¾ years of the data for training (i.e., $T_1 \cup \ldots \cup T_{19}$) and the last ¼ year of the data for testing (i.e., $T_{20}$). The predictions produced by the neural networks were averaged over 10 runs.

Fig. 48 shows the actual and the predicted stock prices during the period between October 4, 2004 and December 31, 2004 ("Fuzzy Rules + Meta-Rules" denotes the predictions based on $R'_{20}$ and "Fuzzy Rules" denotes the predictions based on $R$).

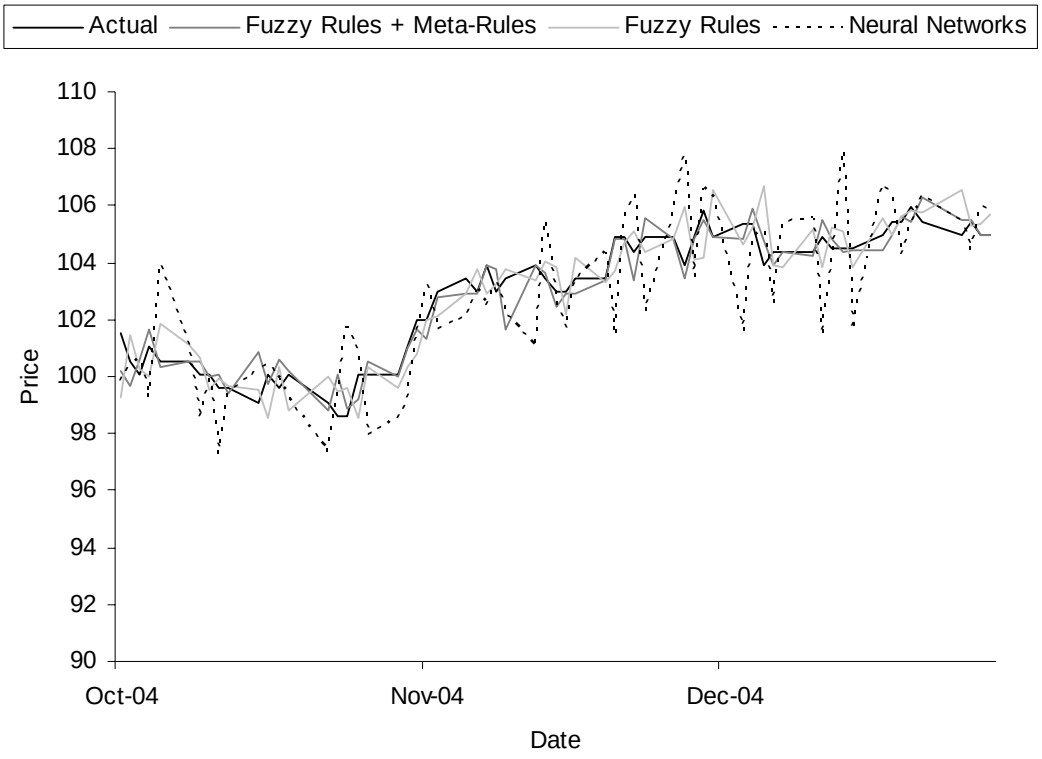



Fig. 48. Prediction of the stock price of Hang Seng Bank Ltd.

The prediction produced by $R'_{20}$, $R$, and neural networks deviates from the actual price by ±0.4%, ±0.7%, and ±1.3%, respectively, in average.

Based on the prediction of the stock price, one can make buy/sell decisions using certain trading strategies. In this section, we compare the performance of predicted fuzzy rules and neural networks using a simple trading strategy. Let us assume that we have a certain amount of capital and we are able to buy/sell the stock of Hang Seng Bank Ltd. from October 4, 2004 to December 31, 2004. In this trading strategy, if the stock price is predicted to rise by 0.5% on date $t + 1$, we buy the stock with all the capital on date $t$; if the stock price is predicted to drop by 0.5% on date $t + 1$, we sell the stock on date $t$; otherwise, we do nothing, that is, we hold the stock or the cash on hand. This strategy allows us to make a profit by buying the stock today (i.e., date $t$) if the stock price will be higher tomorrow (i.e., date $t + 1$) and to prevent a loss by selling the stock today (i.e., date $t$) if the stock price will be lower tomorrow (i.e., date $t + 1$) according to the prediction. We refer to this trading strategy as the *active trading strategy* in the rest of this section.

A weakness of the active trading strategy is that we can only make money when the market is up. When the market is down, all we can do is to prevent loss. In order to allow making money when the market is down, we modify the strategy to become more

aggressive by allowing short selling. In the modified strategy, if the stock price is predicted to rise by 0.5% on date $t + 1$, we buy the stock with all the capital on date $t$; if the stock price is predicted to drop by 0.5% on date $t + 1$, we short sell the stock on date $t$ and buy it back when there is a buy signal; otherwise, we do nothing, that is, we hold the stock or the cash on hand or keep short selling the stock. This allows us to make profit when the market is down because we will be able to buy the stock at a lower amount than the price at which we sold short. We refer to this modified strategy as the *short selling trading strategy* in the rest of this section.

The experimental results are given in Table 38. The number of buy/sell transactions made is provided in Table 39.

Table 38. Trading performance.

| | **Active Trading Strategy** | **Short Selling Trading Strategy** |
|---|---|---|
| **Neural Networks** | 3.0% | 1.4% |
| **Fuzzy Rules** | 5.5% | 6.7% |
| **Fuzzy Rules + Meta-Rules** | 11.5% | 15.3% |

Table 39. Trading signals.

| | **Active Trading Strategy** | | **Short Selling Trading Strategy** | |
|---|---|---|---|---|
| | **Buy** | **Sell** | **Buy** | **Sell** |
| **Neural Networks** | 12 | 11 | 23 | 23 |
| **Fuzzy Rules** | 8 | 7 | 16 | 15 |
| **Fuzzy Rules + Meta-Rules** | 10 | 9 | 19 | 18 |

The experimental results show that neural networks, which do not take into consideration the fact that the trends hidden in financial time series are in short-time basis, cannot produce good trading performance. Although short selling can be used to make money when the market is down, one risks greater loss if the predictions are wrong when compared to not using short selling. This is why neural networks yield even poorer trading performance when using short selling as when they do.

As shown in Table 38, our approach, which mines fuzzy rules and change meta-rules, obtains good trading performance using both the active trading strategy and the short selling trading strategy. This demonstrates the effectiveness of the mining of change meta-rules in representing rule changes in time series.

We also repeated our experimentation on the stock price data for Sun Hung Kai Properties Ltd. and CLP Holdings Ltd. The experimental results also show that our approach to mining fuzzy rules and change meta-rules obtains good prediction accuracy and trading performance although the actual figures vary. We report in this section only the results on Hang Seng Bank Ltd.

# Chapter 10
# Conclusions and Future Work

This study proposes to mine a set of rules from the rules sets discovered by a data mining algorithm. These rules are called *meta-rules* because they are rules about rules. We define the problems of discovering the underlying regularities, differences, and changes hidden in rule sets and propose a new approach to dealing with these problems. We refer to the proposed approach as a *meta-mining* approach since it mines previous mining results.

Given a collection of rule sets, each of which is discovered in a data set, the meta-mining of regularities is concerned with the discovery of association relationships that are supported by a sufficiently large number of rules in the rule sets. They are in common in different data sets (i.e., the regularities) and hence they are called *regular meta-rules*. The regular meta-rules are especially useful for an interstate or international company to better make business decisions that are beneficial to the company as a whole.

The meta-mining of differences from the rule sets aims at revealing rules that are supported by a sufficiently small number of rules. They represent the distinguishing characteristics of the few data sets. They are therefore referred to as *differential meta-rules*. The differential meta-rules are very useful for an international company to better make decisions that are beneficial to specific branches.

Based on our formalism, we can distinguish the associations supported by a number of records in many data sets from the associations supported by many records in only a few data sets. If one concatenates the data sets into amass a single data set, these two kinds of associations cannot be distinguished. With meta-mining, regular meta-rules are used to represent the former kind of associations, whereas differential meta-rules are used to represent the latter kind of associations.

In addition to discovering regularities and differences, we also propose to discover the changes in rules over time. The goal in meta-mining changes from rule sets is to uncover the regularities governing how the rules change over time (i.e., the *change meta-rules*). Change meta-rules reflect change in the underlying characteristics hidden in the data. They can be used for human examination and for predicting how the rules will change in the future. Unless one takes changes into consideration, one can only predict based on historical data and the prediction cannot lead to any change because it will no longer be

valid. Knowing the changes in advance allows a business organization not only to provide new products and services to satisfy the changing needs of its customers, but also to design corrective actions to stop or delay undesirable changes.

To discover regular, differential, and change meta-rules effectively, a meta-mining approach should be able to 1) generate fuzzy sets from data automatically; 2) use linguistic variables and linguistic terms to represent the discovered regularities, differences, and changes; 3) exploit the scalability of parallel computer systems; 4) group and select a subset of attributes; and 5) enable the mining of relationships involving attributes that are not originally contained in the data.

In this study, we propose a good number of techniques to do with the aforementioned tasks and incorporate them into our meta-mining approach. Specifically, to generate fuzzy sets directly from data, we present a new fuzzy partitioning method called ITFP to maximize the class-attribute interdependence and thence improve the classification results. It uses an information-theoretic measure effectively to evaluate the interdependence between the class and an attribute. In the comparison of discretization techniques versus fuzzy partitioning techniques, the experimental results on several real-world data sets show that the latter, if done effectively, can outperform the former. In view of unsupervised versus supervised methods, the results show that the latter perform better than the former. Our ITFP, which is a supervised and fuzzy partitioning method, indeed achieves, by and large, the best performance in our experiments. From the experimental results, the efficacy of ITFP demonstrates that fuzzy partitioning enables fuzzy data mining techniques (e.g., fuzzy decision trees [Janikow 1998], fuzzy classification [Au and Chan 2001], fuzzy association rules [Au and Chan 1998, 1999, 2003, 2004; Chan and Au 1997b, 2001; Chan, Au, and Choi 2002], fuzzy linguistic summaries [Kacprzyk and Zadrozny 2001; Yager 1991], etc.) to build fuzzy models or to discover fuzzy rules on top of the generated fuzzy sets instead of the user-specified fuzzy sets.

In order to employ linguistic variables and linguistic terms to represent the revealed association relationships so that they can be understood by human users easily because of their affinity with human knowledge representation, we propose two new algorithms, called FARM and EFARM, for mining fuzzy rules and meta-rules. FARM discovers high-order fuzzy association rules based on a heuristic, whereas EFARM mines high-order rules using an evolutionary algorithm. Both of them employ an objective interestingness measure to discover interesting association relationships among attributes without any subjective input required of the users. The discovered associations can be used later for human examination or for machine inference, e.g., classification.

We tested their performance with extensive experiments. The experimental results on several real-world data sets for data mining show that our algorithms yield accurate classification. In particular, the experimental results on the subscriber database provided by a carrier in Malaysia show that they are able to discover churn patterns and to predict churn accurately. Furthermore, they are also robust in such a way that they can discover rules hidden in the subscriber database and predict the churn of subscribers under different churn rates. Since the churn rates of different subscribers are different and the churn rate of a specific carrier varies from time to time, robustness is necessary to an effective churn predictor. The ability of our proposed algorithms to identify a large number of churners when only a small fraction of subscribers were contacted is especially important because the customer services center of the carrier has a fixed number of staff and they can contact only a small fraction of subscribers. On the other hand, the experimental results on synthetic data sets for meta-mining also show that our algorithms are effective for discovering the underlying regularities, exceptions, and changes embedded.

To handle very large data sets and rule sets, we extend the two proposed algorithms to exploit the scalability of parallel computer systems. While producing the same results, the parallel algorithms accomplish a data mining or a meta-mining task in only a fraction of the time required by their serial counterparts. The experimental results on a popular benchmarking data set show that they have very good size-up, speedup, and scale-up performance.

We also present a new method for grouping interdependent attributes into clusters by optimizing a criterion function known as interdependence redundancy. We propose a clustering algorithm known as *k*-modes Attribute Clustering Algorithm (ACA). ACA adopts the idea of *k*-means clustering algorithm in the entity space to cluster attributes in the attribute space by replacing 1) the concept of the "mean" in the former by the "mode" and 2) the distance measure used in the former to the interdependence redundancy measure between attributes. In order to have a meaningful evaluation of our methodology, we devise an experimental evaluation scheme to provide a common base of performance assessment and comparison with other methods. From the experiments on the two gene expression data sets, *colon-cancer* and *leukemia*, we find that our attribute clustering algorithm that maximizes intra-group interdependences and the attribute selection method based on multiple attribute interdependence measure works well and yields meaningful and useful results in terms of 1) finding good clustering configurations, which contain interdependence information within clusters and discriminative information for classification; 2) selecting from each cluster significant genes with high multiple interdependence with other genes

within each cluster; and 3) yielding very high classification results on both of gene expression data sets using a small pool of genes selected from the clusters found by ACA as the training set. When comparing the experimental results of ACA with those of several well-known methods, we find that, by and large, ACA outperforms the others. As shown by the surprising results in both the *colon-cancer* and the *leukemia* cases, ACA is able to select very small subsets of genes (14 out of 2,000 in the former and 10 of 7,129 in the latter) to achieve very high classification accuracy (91.9% in the former and 97.1% in the latter) much higher than when the entire set of genes are used. This reveals that the good diagnostic information existing in a small set of genes can be effectively selected by ACA for diagnostic purpose. We believe that this has a significant implication for clinical, pharmaceutical, and bioengineering applications.

To allow the discovery of association relationships involving attributes that are not contained in the original data, we propose using transformation functions and introduce a formal approach. This approach can also handle both relational and transactional data in a relational database. Depending on the type of attribute, we can apply different types of transformation functions to the attributes. The types of transformations include logical, arithmetic, substring, and discretization functions. The use of transformation functions results in a transformed relation. Instead of performing data mining on the original data, we applied our fuzzy association rule mining algorithms to the transformed data of the bank-account database provided by an international bank. Among the discovered fuzzy association rules, we selected 200 rules randomly and presented them to a domain expert from the bank. The domain expert confirmed that she could understand the fuzzy association rules without any difficulty, although it is nontrivial for her to explain the basis for some of the rules. In particular, the domain expert found that 91.5% of these randomly selected rules are useful or very useful. The reasons for this are likely to be that our interestingness measure can effectively reveal the interesting associations that are hidden in the data and that the fuzzy association rules, which employ linguistic terms to represent the underlying relationships, are more natural for human users to understand.

Finally, we applied our proposed meta-mining approach to several synthetic and real-life data sets for experimentation. The experimental results show that our approach is able to reveal the underlying regularities, differences, and changes hidden in the data.

In conclusion, our proposed meta-mining approach is very effective not only in mining rules from data sets, but also in mining meta-rules from rule sets. The discovered meta-rules effectively represent the underlying regularities, differences, and changes hidden in the rule sets, which in turn reflect the regularities, the differences, and the change of

characteristics in the data sets.

In the future, we are going to generalize our fuzzy partitioning method to handle multiple variables at the same time. Instead of fuzzy sets, it would result in fuzzy relations. The rules discovered involing such fuzzy relations would perhaps reflect a more meaningful representation of the underlying relationships as compared to those involving fuzzy sets.

# References


D. Abramson and J. Abela (1992) “A Parallel Genetic Algorithm for Solving the School Timetabling Problem,” in *Proc. of the 15th Australian Computer Science Conf.*, pp. 1–11.

R. Agrawal, S. Ghosh, T. Imielinski, B. Iyer, and A. Swami (1992) “An Interval Classifier for Database Mining Applications,” in *Proc. of the 18th Int’l Conf. on Very Large Data Bases*, Vancouver, British Columbia, Canada, pp. 560–573.

R. Agrawal, T. Imielinski, and A. Swami (1993a) “Database Mining: A Performance Perspective,” *IEEE Trans. on Knowledge and Data Engineering*, vol. 5, no. 6, pp. 914–925.

R. Agrawal, T. Imielinski, and A. Swami (1993b) “Mining Association Rules between Sets of Items in Large Databases,” in *Proc. of the ACM SIGMOD Int’l Conf. on Management of Data*, Washington D.C., pp. 207–216.

R. Agrawal, M. Mehta, J. Shafer, R. Srikant, A. Arning, and T. Bollinger (1996) “The Quest Data Mining System,” in *Proc. of the 2nd Int’l Conf. on Data Mining and Knowledge Discovery*, Portland, OR, pp. 244–249.

R. Agrawal and G. Psaila (1995) “Active Data Mining,” in *Proc. of the 1st Int’l Conf. on Knowledge Discovery and Data Mining*, Montreal, Canada.

R. Agrawal and J. C. Shafer (1996) “Parallel Mining of Association Rules,” *IEEE Trans. on Knowledge and Data Engineering*, vol. 8, no. 6, pp. 962–969.

R. Agrawal and R. Srikant (1994) “Fast Algorithms for Mining Association Rules,” in *Proc. of the 20th Int’l Conf. on Very Large Data Bases*, Santiago, Chile, pp. 487–499.

R. Agrawal and R. Srikant (1995) “Mining Sequential Patterns,” in *Proc. of the 11th IEEE Int’l Conf. on Data Engineering*, Taipei, Taiwan, pp. 3–14.

A. Agresti (1990) *Categorical Data Analysis*, New York, NY: John Wiley & Sons.

U. Alon, N. Barkai, D. A. Notterman, K. Gish, S. Ybarra, D. Mack, and A. J. Levine (1999) “Broad Patterns of Gene Expression Revealed by Clustering Analysis of Tumor and Normal Colon Tissues Probed by Oligonucleotide Arrays,” *Proc. of the National Academy of Sciences of the United States of America*, vol. 96, no. 12, pp. 6745–6750.

H. Andre-Jonsson and D. Badal (1997) “Using Signature Files for Querying Time-Series Data,” in *Proc. of the 1st European Symp. on Data Mining and Knowledge Discovery*,

Trondheim, Norway, pp. 211–220.

A. Arslan and M. Kaya (2001) “Determination of Fuzzy Logic Membership Functions Using Genetic Algorithms,” *Fuzzy Sets and Systems*, vol. 118, no. 2, pp. 297–306.

W.-H. Au and K. C. C. Chan (1998) “An Effective Algorithm for Discovering Fuzzy Rules in Relational Databases,” in *Proc. of the 7th IEEE Int’l Conf. on Fuzzy Systems*, Anchorage, AK, pp. 1314–1319.

W.-H. Au and K. C. C. Chan (1999) “FARM: A Data Mining System for Discovering Fuzzy Association Rules,” in *Proc. of the 8th IEEE Int’l Conf. on Fuzzy Systems*, Seoul, Korea, pp. 1217–1222.

W.-H. Au and K. C. C. Chan (2001) “Classification with Degree of Membership: A Fuzzy Approach,” in *Proc. of the 1st IEEE Int’l Conf. on Data Mining*, San Jose, CA, pp. 35–42.

W.-H. Au and K. C. C. Chan (2002a) “An Evolutionary Approach for Discovering Changing Patterns in Historical Data,” in B. V. Dasarathy (Ed.), *Data Mining and Knowledge Discovery: Theory, Tools, and Technology IV*, Proc. of SPIE Vol. 4730, pp. 398–409.

W.-H. Au and K. C. C. Chan (2002b) “Fuzzy Data Mining for Discovering Changes in Association Rules over Time,” in *Proc. of the 11th IEEE Int’l Conf. on Fuzzy Systems*, Honolulu, HI, pp. 890–895.

W.-H. Au and K. C. C. Chan (2003) “Mining Fuzzy Association Rules in a Bank-Account Database,” *IEEE Trans. on Fuzzy Systems*, vol. 11, no. 2, pp. 238–248.

W.-H. Au and K. C. C. Chan (2004) “Mining Fuzzy Rules for Time Series Classification,” in *Proc. of the 13th IEEE Int’l Conf. on Fuzzy Systems*, Budapest, Hungary, pp. 239–244.

W.-H. Au and K. C. C. Chan (2005) “Mining Changes in Association Rules: A Fuzzy Approach,” *Fuzzy Sets and Systems*, vol. 149, no. 1, pp. 87–104.

W.-H. Au, K. C. C. Chan, A. K. C. Wong, and Y. Wang (2005) “Attribute Clustering for Grouping, Selection, and Classification of Gene Expression Data,” *IEEE/ACM Trans. on Computational Biology and Bioinformatics*, vol. 2, no. 2, pp. 82–101.

W.-H. Au, K. C. C. Chan, and X. Yao (2003) “A Novel Evolutionary Data Mining Algorithm with Applications to Churn Prediction,” *IEEE Trans. on Evolutionary Computation*, vol. 7, no. 6, pp. 532–545.

D.-E. Baestaens, W.-M. van den Bergh, and D. Wood (1994) *Neural Networks Solutions for*

*Trading in Financial Markets*, London, U.K.: Pitman.

A. Ben-Dor, L. Bruhn, N. Friedman, I. Nachman, M. Schummer, and Z. Yakhini (2000) "Tissue Classification with Gene Expression Profiles," in *Proc. of the 4th Annual Int'l Conf. on Computational Molecular Biology*, Tokyo, Japan, pp. 54–64.

A. D. Bethke (1976), "Comparison of Genetic Algorithms and Gradient-Based Optimizers on Parallel Processors: Efficiency of Use of Processing Capacity", *Technical Report 197*, Logic of Computer Group, University of Michigan, Ann Arbor, MI.

J. C. Bezdek (1981) *Pattern Recognition with Fuzzy Objective Function Algorithms*, New York, NY: Plenum.

C. Bishop (1995) *Neural Networks for Pattern Recognition*, New York, NY: Oxford Univ. Press.

C. L. Blake and C. J. Merz (1998) *UCI Repository of Machine Learning Databases* [http://www.ics.uci.edu/~mlearn/MLRepository.html], Department of Information and Computer Science, University of California, Irvine, CA.

G. E. P. Box, G. M. Jenkins, and G. C. Reinsel (1994) *Time Series Analysis: Forecasting and Control*, 3rd Ed., Englewood Cliffs, NJ: Prentice-Hall.

T. Bozkaya, N. Yazdani, and Z. M. Ozsoyoglu (1997) "Matching and Indexing Sequences of Different Lengths," in *Proc. of the 6th Int'l Conf. on Information and Knowledge Management*, Las Vegas, NV, pp. 128–135.

P. Bradley, U. Fayyad, and C. Reina (1998) "Scaling Clustering Algorithms to Large Databases," in *Proc. of the 4th Int'l Conf. on Knowledge Discovery and Data Mining*, New York, NY, pp. 9–15.

L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone (1984) *Classification and Regression Trees*, Belmont, CA: Wadsworth.

M. Bright, A. Hurson, and S. Pakzad (1992) "A Taxonomy and Current Issues in Multidatabase Systems," *IEEE Computer*, vol. 25, no. 3, pp. 50–60.

S. Brin, R. Motwani, C. Silverstein (1997) "Beyond Market Baskets: Generalizing Association Rules to Correlations," in *Proc. of the ACM SIGMOD Int'l Conf. on Management of Data*, Tucson, AZ, pp. 265–276.

S. Brin, R. Motwani, J. D. Ullman, and S. Tsur (1997) "Dynamic Itemset Counting and Implication Rules for Market Basket Data," in *Proc. of the ACM SIGMOD Int'l Conf. on Management of Data*, Tucson, AZ, pp. 255–264.

B. G. Buchanan, D. Barstow, R. Bechtal, J. Bennett, W. Clancey, C. Kulikowski, T. Mitchell, and D. A. Waterman (1983) "Constructing an Expert System," in F. Hayes-Roth, D. A. Waterman, and D. B. Lenat (Eds.), *Building Expert Systems*, Boston, MA: Addison-Wesley, pp. 127–168.

Y. Cai, N. Cercone, and J. Han (1991) "Attribute-Oriented Induction in Relational Databases," in [Piatetsky-Shapiro and Frawley 1991], pp. 213–228.

E. Cantu-Paz (1998) "A Survey of Parallel Genetic Algorithms," *Calculateurs Paralleles, Reseaux et Systems Repartis*, vol. 10, no. 2, pp. 141–171.

E. Cantu-Paz and D. E. Goldberg (1999) "On the Scalability of Parallel Genetic Algorithms," *Evolutionary Computation*, vol. 7, no. 4, pp. 429–449.

K. C. C. Chan and W.-H. Au (1997a) "An Effective Algorithm for Mining Interesting Quantitative Association Rules," in *Proc. of the 12th ACM Symp. on Applied Computing*, San Jose, CA, pp. 88–90.

K. C. C. Chan and W.-H. Au (1997b) "Mining Fuzzy Association Rules," in *Proc. of the 6th Int'l Conf. on Information and Knowledge Management*, Las Vegas, NV, pp. 209–215.

K. C. C. Chan and W.-H. Au (2001) "Mining Fuzzy Association Rules in a Database Containing Relational and Transactional Data," in A. Kandel, M. Last, and H. Bunke (Eds.), *Data Mining and Computational Intelligence*, New York, NY: Physica-Verlag, pp. 95–114.

K. C. C. Chan, W.-H. Au, and B. Choi (2002) "Mining Fuzzy Rules in a Donor Database for Direct Marketing by a Charitable Organization," in *Proc. of the 1st IEEE Int'l Conf. on Cognitive Informatics*, Calgary, Alberta, Canada, pp. 239–246.

K. C. C. Chan and A. K. C. Wong (1990) "APACS: A System for the Automatic Analysis and Classification of Conceptual Patterns," *Computational Intelligence*, vol. 6, no. 3, pp. 119–131.

K. C. C. Chan and A. K. C. Wong (1991) "A Statistical Technique for Extracting Classificatory Knowledge from Databases," in [Piatetsky-Shapiro and Frawley 1991], pp. 107–123.

P. Cheeseman and J. Stutz (1996) "Bayesian Classification (AutoClass): Theory and Results," in [Fayyad *et al.* 1996], pp. 153–180.

Y. Cheng and G. M. Church (2000) "Biclustering of Expression Data," in *Proc. of the 8th Int'l Conf. on Intelligent Systems for Molecular Biology*, San Diego, CA, pp. 93–103.

D. W. Cheung, J. Han, V. T. Ng, A. W. Fu, and Y. Fu (1996a) "A Fast Distributed

Algorithm for Mining Association Rules," in *Proc. of the 4th Int'l Conf. on Parallel and Distributed Information Systems*, Miami Beach, FL, pp. 31–42.

D. W. Cheung, J. Han, V. T. Ng, and C. Y. Wong (1996b) "Maintenance of Discovered Association Rules in Large Databases: An Incremental Updating Technique," in *Proc. of the 12th Int'l Conf. on Data Engineering*, New Orleans, LA, pp. 106–114.

J. Y. Ching, A. K. C. Wong, and K. C. C. Chan (1995) "Class-Dependent Discretization for Inductive Learning from Continuous and Mixed-Mode Data," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 17, no. 7, pp. 641–651.

D. K. Y. Chiu and A. K. C. Wong (2004) "Multiple Pattern Associations for Interpreting Structural and Functional Characteristics of Biomolecules," *Information Sciences*, vol. 167, pp. 23–39.

D. K. Y. Chiu, A. K. C. Wong, and B. Cheung (1991) "Information Discovery through Hierarchical Maximum Entropy Discretization and Synthesis," in [Piatetsky-Shapiro and Frawley 1991], pp. 125–140.

P. Clark and T. Niblett (1989) "The CN2 Algorithm," *Machine Learning*, vol. 3, pp. 261–283.

S. Choenni (2000) "Design and Implementation of a Genetic-Based Algorithm for Data Mining," in *Proc. of the 26th Int'l Conf. on Very Large Data Bases*, Cairo, Egypt, pp. 33–42.

G. Das, K.-I. Lin, H. Mannila, G. Renganathan, and P. Smyth (1998) "Rule Discovery from Time Series," in *Proc. of the 4th Int'l Conf. on Knowledge Discovery and Data Mining*, New York, NY, pp. 16–22.

C. J. Date (2000) *An Introduction to Database Systems*, 7th Ed., Reading, MA: Addison-Wesley.

K. A. DeJong, W. M. Spears, and D. F. Gordon (1993) "Using Genetic Algorithms for Concept Learning," *Machine Learning*, vol. 13, pp. 161–188.

M. Delgado, N. Marín, D. Sánchez, and M.-A. Vila (2003) "Fuzzy Association Rules: General Model and Applications," *IEEE Trans. on Fuzzy Systems*, vol. 11, no. 2, pp. 214–225.

A. P. Dempster (1967) "Upper and Lower Probabilities Induced by a Multi-Valued Mapping," *Annals of Mathematical Statistics*, vol. 38, pp. 325–339.

F. De Smet, J. Mathys, K. Marchal, G. Thijs, B. De Moor, and Y. Moreau (2002) "Adaptive Quality-Based Clustering of Gene Expression Profiles," *Bioinformatics*, vol. 18, no. 5,

pp. 735–746.

V. Dhar and A. Tuzhilin (1993) “Abstract-Driven Pattern Discovery in Databases,” *IEEE Trans. on Knowledge and Data Engineering*, vol. 5, no. 6, pp. 926–938.

C. Ding and H. Peng (2003) “Minimum Redundancy Feature Selection from Microarray Gene Expression Data,” in *Proc. of the IEEE Computational Systems Bioinformatics Conf.*, Stanford, CA, pp. 523–528.

E. Domany (2003) “Cluster Analysis of Gene Expression Data,” *Journal of Statistical Physics*, vol. 110, pp. 1117–1139.

J. Dougherty, R. Kohavi, and M. Sahami (1995) “Supervised and Unsupervised Discretization of Continuous Features,” in *Proc. of the 12th Int’l Conf. on Machine Learning*, Tahoe City, CA, pp. 194–202.

S. Dudoit, J. Fridlyand, and T. P. Speed (2002) “Comparison of Discrimination Methods for the Classification of Tumors Using Gene Expression Data,” *Journal of the American Statistical Association*, vol. 97, no. 457, pp. 77–87.

M. B. Eisen, P. T. Spellman, P. O. Brown, and D. Botstein (1998) “Cluster Analysis and Display of Genome-Wide Expression Patterns,” *Proc. of the National Academy of Sciences of the United States of America*, vol. 95, no. 25, pp. 14863–14868.

M. Fajfer and C. Z. Janikow (2000) “Bottom-Up Fuzzy Partitioning in Fuzzy Decision Trees,” in *Proc. of the 19th Int’l Conf. of the North American Fuzzy Information Processing Society*, Atlanta, GA, pp. 326–330.

U. M. Fayyad and K. B. Irani (1993) “Multi-Interval Discretization of Continuous-Valued Attributes for Classification Learning,” in *Proc. of the 13th Int’l Joint Conf. on Artificial Intelligence*, Chambéry, France, pp. 1022–1029.

U. M. Fayyad, G. Piatetsky-Shapiro, and P. Smyth (1996) “From Data Mining to Knowledge Discovery: An Overview,” in [Fayyad *et al.* 1996], pp. 1–34.

U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy (Eds.) (1996) *Advances in Knowledge Discovery and Data Mining*, Menlo Park, CA; Cambridge, MA: AAAI/MIT Press.

M. V. Fidelis, H. S. Lopes, and A. A. Freitas (2000) “Discovering Comprehensible Classification Rules with a Genetic Algorithm,” in *Proc. of the 2000 Congress on Evolutionary Computation*, San Diego, CA, pp. 805–810.

T. C. Fogarty and R. Huang (1991) “Implementing the Genetic Algorithm on Transputer Based Parallel Processing Systems,” in *Proc. of Parallel Problem Solving from Nature*,

pp. 145–149.

D. B. Fogel (1995) *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence*, New York, NY: IEEE Press.

R. Forsyth (1990) *PC/BEAGLE User's Guide*, Pathway Research Ltd.

W. J. Frawley, G. Piatetsky-Shapiro, and C. J. Matheus (1991) "Knowledge Discovery in Databases: An Overview," in [Piatetsky-Shapiro and Frawley 1991], pp. 1–27.

A. A. Freitas (2002) "Understanding the Critical Role of Attribute Interaction in Data Mining," *Artificial Intelligence Review*, vol. 16, pp. 177–199.

N. Friedman, M. Nachman, and D. Pe'er (2000) "Using Baysian Networks to Analyze Expression Data," in *Proc. of the 4th Annual Int'l Conf. on Computational Molecular Biology*, Tokyo, Japan, pp. 127–135.

V. Ganti, J. Gehrke, R. Ramakrishnan, and W.-Y. Loh (1999a) "A Framework for Measuring Changes in Data Characteristics," in *Proc. of the 18th ACM SIGMOD-SIGACT-SIGART Symp. on Principles of Database Systems*, Philadelphia, PA, pp. 126–137.

V. Ganti, R. Ramakrishnan, J. Gehrke, A. L. Powell, and J. C. French (1999b) "Clustering Large Data Sets in Arbitrary Metric Spaces," in *Proc. of the 15th Int'l Conf. on Data Engineering*, Sydney, Australia, pp.502–511.

M. Gavrilov, D. Anguelov, P. Indyk, and R. Motwani (2000) "Mining the Stock Market: Which Measure is Best?" in *Proc. of the 6th ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining*, Boston, MA, pp. 487–496.

D. Gayme, S. Menon, and C. Ball (2003) "Fault Detection and Diagnosis in Turbine Engines Using Fuzzy Logic," in *Proc. of the 22nd Int'l Conf. on the North American Fuzzy Information Procession Society*, Chicago, IL, pp. 341–346.

A. Geist, A. Beguelin, J. Dongarra, W. Jiang, R. Manchek, and V. Sunderam (1994) *PVM: Parallel Virtual Machine. A Users' Guide and Tutorial for Networked Parallel Computing*, Cambridge, MA: MIT Press.

D. E. Goldberg (1989) *Genetic Algorithms in Search, Optimization, and Machine Learning*, Reading, MA: Addison-Wesley.

T. R. Golub, D. K. Slonim, P. Tamayo, C. Huard, M. Gaasenbeek, J. P. Mesirov, H. Coller, M. L. Loh, J. R. Downing, M. A. Caligiuri, C. D. Bloomfield, and E. S. Lander (1999) "Molecular Classification of Cancer: Class Discovery and Class Prediction by Gene Expression Monitoring," *Science*, vol. 286, pp. 531–537.

D. P. Greene and S. F. Smith (1994) “Using Coverage as a Model Building Constraint in Learning Classifier Systems,” *Evolutionary Computation*, vol. 2, no. 1, pp. 67–91.

J. J. Grefenstette (1981) “Parallel Adaptive Algorithms for Function Optimization,” *Technical Report CS-81-19*, Computer Science Department, Vanderbilt University, Nashville, TN.

P. B. Grosso (1985) “Computer Simulations of Genetic Adaptation: Parallel Subcomponent Interaction in a Multilocus Model,” *Ph.D. Thesis*, University of Michigan, Ann Arbor, MI.

V. Guralnik and J. Srivastava (1999) “Event Detection from Time Series Data,” in *Proc. of the 5th ACM SIGKDD Int’l Conf. on Knowledge Discovery and Data Mining*, San Diego, CA, pp. 33–42.

J. Han, Y. Cai, and N. Cercone (1992) “Knowledge Discovery in Databases: An Attribute-Oriented Approach,” in *Proc. of the 18th Int’l Conf. on Very Large Data Bases*, Vancouver, British Columbia, Canada, pp. 547–559.

J. Han, Y. Cai, and N. Cercone (1993) “Data-Driven Discovery of Quantitative Rules in Relational Databases,” *IEEE Trans. on Knowledge and Data Engineering*, vol. 5, no. 1, pp. 29–40.

J. Han, G. Dong, and Y. Yin (1999) “Efficient Mining of Partial Periodic Patterns in Time Series Database,” in *Proc. of the 15th IEEE Int’l Conf. on Data Engineering*, Sydney, Australia, pp. 106–115.

J. Han and Y. Fu (1995) “Discovery of Multiple-Level Association Rules from Large Databases,” in *Proc. of the 21st Int’l Conf. on Very Large Data Bases*, Zurich, Switzerland, pp. 420–431.

J. Han and Y. Fu (1996) “Exploration of the Power of Attribute-Oriented Induction in Data Mining,” in [Fayyad *et al.* 1996], pp. 399–421.

J. Han, Y. Fu, W. Wang, J. Chiang, W. Gong, K. Koperski, D. Li, Y. Lu, A. Rajan, N. Stefanovic, B. Xia, and O. R. Zaiane (1996) “DBMiner: A System for Mining Knowledge in Large Relational Databases,” in *Proc. of the 2nd Int’l Conf. on Data Mining and Knowledge Discovery*, Portland, OR, pp. 250–255.

J. Han and M. Kamber (2001) *Data Mining: Concepts and Techniques*, San Francisco, CA: Morgan Kaufmann.

E.-H. Han, G. Karypis, and V. Kumar (1997) “Scalable Parallel Data Mining for Association Rules,” in *Proc. of the ACM SIGMOD Int’l Conf. on Management of Data*,

Tucson, AZ, pp. 277–288.

J. Han, J. Pei, and Y. Yin (2000) “Mining Frequent Patterns without Candidate Generation,” in *Proc. of the ACM SIGMOD Int’l Conf. on Management of Data*, Dallas, TX, 2000, pp. 1–12.

D. Hand, H. Mannila, and P. Smyth (2001) *Principles of Data Mining*, Cambridge, MA: The MIT Press.

R. Hauser and R. Manner (1993) “Implementation of Standard Genetic Algorithm on MIMD Machines,” in Y. Davidor, H.-P. Schwefel, and R. Manner (Eds.), *Parallel Problem Solving from Nature, PPSN III*, Berlin, Germany: Springer-Verlag, pp. 504–513.

R. Herwig, A. J. Poustka, C. Müller, C. Bull, H. Lehrach, and J. O’Brien (1999) “Large-Scale Clustering of cDNA-Fingerprinting Data,” *Genome Research*, vol. 9, pp. 1093–1105.

L. J. Heyer, S. Kruglyak, and S. Yooseph (1999) “Exploring Expression Data: Identification and Analysis of Coexpressed Genes,” *Genome Research*, vol. 9, pp. 1106–1115.

R. R. Hill (1999) “A Monte Carlo Study of Genetic Algorithm Initial Population Generation Methods,” in *Proc. of the 31st Conf. on Winter Simulation – A Bridge to the Future*, Phoenix, AZ, pp. 543–547.

K. Hirota and W. Pedrycz (1999) “Fuzzy Computing for Data Mining,” *Proc. of the IEEE*, vol. 87, no. 9, pp. 1575–1600.

J. Holland (1986) “Escaping Brittleness: The Possibilities of General-Purpose Learning Algorithms Applied to Parallel Rule-Based Systems,” in R. Michalski, J. Carbonell, and T. Mitchell (Eds.), *Machine Learning: An Artificial Intelligence Approach*, San Mateo, CA: Morgan Kaufmann.

M. Houtsma and A. Swami (1995) “Set-Oriented Mining for Association Rules in Relational Databases,” in *Proc. of the 11th Int’l Conf. on Data Engineering*, Taipei, Taiwan, pp. 25–33.

Y. Huang and P. S. Yu (1999) “Adaptive Query Processing for Time-Series Data,” in *Proc. of the 5th ACM SIGKDD Int’l Conf. on Knowledge Discovery and Data Mining*, San Diego, CA, pp. 282–286.

E. Hüllermeier (2001) “Implication-Based Fuzzy Association Rules,” in *Proc. of the 5th European Conf. on Principles of Data Mining and Knowledge Discovery*, Freiburg, Germany, pp. 241–252.

IBM Quest Data Mining Project (1996) *Quest Synthetic Data Generation Code* [http://www.almaden.ibm.com/software/quest/Resources/datasets/syndata.html].

T. Imielinski, A. Virmani, and A. Abdulghani (1996) "DataMine: Application Programming Interface and Query Language for Database Mining," in *Proc. of the 2nd Int'l Conf. on Data Mining and Knowledge Discovery*, Portland, OR, pp. 256–262.

P. Indyk, N. Koudas, and S. Muthukrishnan (2000) "Identifying Representative Trends in Massive Time Series Data Sets Using Sketches," in *Proc. of the 26th Int'l Conf. on Very Large Data Bases*, Cairo, Egypt, pp. 363–372.

H. Ishibuchi and T. Nakashima (1999) "Improving the Performance of Fuzzy Classifier Systems for Pattern Classification Problems with Continuous Attributes," *IEEE Trans. on Industrial Electronics*, vol. 46, no. 6, pp. 1057–1068.

H. Ishibuchi, T. Yamamoto, and T. Nakashima (2001) "Fuzzy Data Mining: Effect of Fuzzy Discretization," in *Proc. of the 1st IEEE Int'l Conf. on Data Mining*, San Jose, CA, pp. 241–248.

A. K. Jain, M. N. Murty, and P. J. Flynn (1999) "Data Clustering: A Review," *ACM Computing Surveys*, vol. 31, no. 3, pp. 264–323.

J.-S. R. Jang (1993) "ANFIS: Adaptive-Network-Based Fuzzy Inference System," *IEEE Trans. on Systems, Man, and Cybernetics*, vol. 23, no. 3, pp. 665–685.

C. Z. Janikow (1993) "A Knowledge-Intensive Genetic Algorithm for Supervised Learning," *Machine Learning*, vol. 13, pp. 189–228.

C. Z. Janikow (1998) "Fuzzy Decision Trees: Issues and Methods," *IEEE Trans. on Systems, Man, and Cybernetics – Part B: Cybernetics*, vol. 28, no. 1, pp. 1–14.

C. Z. Janikow and M. Fajfer (1999) "Fuzzy Partitioning with FID3.1," in *Proc. of the 18th Int'l Conf. of the North American Fuzzy Information Processing Society*, New York, NY, pp. 467–471.

D. Jiang, C. Tang, and A. Zhang (2004) "Cluster Analysis for Gene Expression Data: A Survey," *IEEE Trans. on Knowledge and Data Engineering*, vol. 16, no. 11, pp. 1370–1386.

P. N. Johnson-Laird (1989) "Human Experts and Expert Systems," in L. A. Murray and J. T. E. Richardson (Eds.), *Intelligent Systems in a Human Context: Development, Implications, and Applications*, Oxford, U.K.: Oxford University Press, pp. 35–46.

M. V. Joshi, G. Karypis, and V. Kumar (1998) "ScalParC: A New Scalable and Efficient Parallel Classification Algorithm for Mining Large Datasets," in *Proc. of the 1st*

*Merged Int'l Parallel Processing Symp. and Symp. on Parallel and Distributed Processing*, Orlando, FL, pp. 573–579.

B. A. Julstrom (1994) "Seeding the Population: Improved Performance in a Genetic Algorithm for the Rectilinear Steiner Problem," in *Proc. of the ACM Symp. on Applied Computing*, Phoenix, AZ, pp. 222–226.

J. Kacprzyk and S. Zadrozny (2001) "On Linguistic Approaches in Flexible Querying and Mining of Association Rules," in H. L. Larsen, J. Kacprzyk, S. Zadrozny, T. Andreasen, and H. Christiansen (Eds.), *Flexible Query Answering Systems: Recent Advances*, Proc. of the 4th Int'l Conf. on Flexible Query Answering Systems, Heidelberg, Germany: Physica-Verlag, pp. 475–484.

K. Kalpakis, D. Gada, and V. Puttagunta (2001) "Distance Measure for Effective Clustering of ARIMA Time Series," in *Proc. of the 1st IEEE Int'l Conf. on Data Mining*, San Jose, CA, pp. 273–280.

C. L. Karr (1991) "Design of an Adaptive Fuzzy Logic Controller Using a Genetic Algorithm," in *Proc. of the 4th Int'l Conf. on Genetic Algorithms*, San Diego, CA, pp. 450–457.

A. D. Keller, M. Schummer, L. Hood, and W. L. Ruzzo (2000) "Bayesian Classification of DNA Array Expression Data," *Technical Report UW-CSE-2000-08-01*, Department of Computer Science and Engineering, University of Washington.

E. Keogh and S. Kasetty (2003) "On the Need for Time Series Data Mining Benchmarks: A Survey and Empirical Demonstration," *Data Mining and Knowledge Discovery*, vol. 7, no. 4, pp. 349–371.

E. Keogh and P. Smyth (1997) "A Probabilistic Approach to Fast Pattern Matching in Time Series Databases," in *Proc. of the 3rd Int'l Conf. on Knowledge Discovery and Data Mining*, Newport Beach, CA, pp. 24–30.

R. Kerber (1992) "ChiMerge: Discretization of Numerical Attributes," in *Proc. of the 9th National Conf. on Artificial Intelligence*, San Jose, CA, pp. 123–128.

J. Khan, J. S. Wei, M. Ringner, L. H. Saal, M. Ladanyi, F. Westermann, F. Berthold, M. Schwab, C. R. Antonescu, C. Peterson, and P. S. Meltzer (2001) "Classification and Diagnostic Prediction of Cancers Using Gene Expression Profiling and Artificial Neural Networks," *Nature Medicine*, vol. 7, no. 6, pp. 673–679.

M. Klemettinen, H. Mannila, P. Ronkainen, H. Toivonen, and A. I. Verkamo (1994) "Finding Interesting Rules from Large Sets of Discovered Association Rules," in *Proc.*

*of the 3rd Int'l Conf. on Information and Knowledge Management*, Gaithersburg, MD, pp. 401–407.

R. Kohavi (1996) "Scaling Up the Accuracy of Naive-Bayes Classifiers: A Decision Tree Hybrid," in *Proc. of the 2nd Int'l Conf. on Knowledge Discovery and Data Mining*, Portland, Oregon.

T. Kohonen (2001) *Self-Organizing Maps*, 3rd Ed., Berlin, Germany: Springer-Verlag.

V. Kumar, A. Grama, A. Gupta, and G. Karypis (1994) *Introduction to Parallel Computing: Design and Analysis of Algorithms*, Redwood City, CA: Benjamin/Cummings.

L. Kurgan and K. J. Cios (2001) "Discretization Algorithm that Uses Class-Attribute Interdependence Maximization," in *Proc. of the 2001 Int'l Conf. on Artificial Intelligence*, Las Vegas, NV, pp. 980–987.

L. A. Kurgan and K. J. Cios (2004) "Meta Mining Architecture for Supervised Learning," in *Proc. of the 7th Int'l Workshop on High Performance and Distributed Mining*, Lake Buena Vista, FL, pp. 18–26.

W. Kwedlo and M. Kretowski (1998) "Discovery of Decision Rules from Databases: An Evolutionary Approach," in *Proc. of the 2nd European Symp. on Principles of Data Mining and Knowledge Discovery*, Nantes, France, pp. 370–378.

M. Last, Y. Klein, and A. Kandel (2001) "Knowledge Discovery in Time Series Database," *IEEE Trans. on Systems, Man, and Cybernetics – Part B: Cybernetics*, vol. 31, no. 1, pp. 160–169.

D. H. Lee and M. H. Kim (1997) "Database Summarization Using Fuzzy ISA Hierarchies," *IEEE Trans. on Systems, Man, and Cybernetics – Part B: Cybernetics*, vol. 27, no. 4, pp. 671–680.

M. A. Lee and H. Takagi (1993) "Integrating Design Stages for Fuzzy Systems Using Genetic Algorithms," in *Proc. of the 2nd IEEE Int'l Conf. on Fuzzy Systems*, San Francisco, CA, pp. 612–617.

J. Li and L. Wong (2002a) "Identifying Good Diagnostic Gene Groups from Gene Expression Profiles Using the Concept of Emerging Patterns," *Bioinformatics*, vol. 18, no. 5, pp. 725–734.

J. Li and L. Wong (2002b) "Identifying Good Diagnostic Gene Groups from Gene Expression Profiles Using the Concept of Emerging Patterns (Corrigendum)," *Bioinformatics*, vol. 18, no. 10, pp. 1406–1407.

T. W. Liao, A. K. Celmins, and R. J. Hammell II (2003) "A Fuzzy C-Means Variant for the

Generation of Fuzzy Term Sets," *Fuzzy Sets and Systems*, vol. 135, no. 2, pp. 241–257.

B. Liu, W. Hsu, H.-S. Han, and Y. Xia (2000) "Mining Changes for Real-Life Applications," in *Proc. of the 2nd Int'l Conf. on Data Warehousing and Knowledge Discovery*, London Greenwich, U.K.

B. Liu, W. Hsu, and Y. Ma (1998) "Integrating Classification and Association Rule Mining," in *Proc. of the 4th Int'l Conf. on Knowledge Discovery and Data Mining*, New York, NY, pp. 80–86.

B. Liu, W. Hsu, and Y. Ma (2001) "Discovering the Set of Fundamental Rule Changes," in *Proc. of the 7th ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining*, San Francisco, CA, pp. 335–340.

H. Liu, H. Lu, and J. Yao (1998) "Identifying Relevant Databases for Multidatabase Mining," in *Proc. of the 2nd Pacific-Asia Conf. on Knowledge Discovery and Data Mining*, Melbourne, Australia, pp. 210–221.

H. Liu and R. Setiono (1997) "Feature Selection via Discretization," *IEEE Trans. on Knowledge and Data Engineering*, vol. 9, no. 4, pp. 642–645.

L. Liu, A. K. C. Wong, and Y. Wang (2004) "A Global Optimal Algorithm for Class-Dependent Discretization of Continuous Data," *Intelligent Data Analysis*, vol. 8, no. 2, pp. 151–170.

Y. Lu and J. Han (2003) "Cancer Classification Using Gene Expression Data," *Information Systems*, vol. 28, no. 4, pp. 243–268.

H. Lu, J. Han, and L. Feng (1998) "Stock Movement Prediction and N-Dimensional Inter-Transactional Association Rules," in *Proc. of the ACM SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery*, Seattle, WA, pp. 12:1–12:7.

H. Lu, R. Setiono, and H. Liu (1995) "NeuroRule: A Connectionist Approach to Data Mining," in *Proc. of the 21st Int'l Conf. on Very Large Data Bases*, Zurich, Switzerland, pp. 478–489.

L. Liu, A. K. C. Wong, and Y. Wang (2004) "A Global Optimal Algorithm for Class-Dependent Discretization of Continuous Data," *Intelligent Data Analysis*, vol. 8, no. 2, pp. 151–170.

J. Lockwood (1997) "Study Predicts 'Epidemic' Churn," Wireless Week, Aug. 25, 1997.

D. J. C. MacKay (2003) *Information Theory, Inference, and Learning Algorithms*, Cambridge, U.K.: Cambridge University Press.

S. C. Madeira and A. L. Oliveira (2004) “Biclustering Algorithms for Biological Data Analysis: A Survey,” *IEEE Trans. on Computational Biology and Bioinformatics*, vol. 1, no. 1, pp. 24–45.

O. Maimon, A. Kandel, and M. Last (1999) “Information-Theoretic Fuzzy Approach to Knowledge Discovery in Databases,” in R. Roy, T. Furuhashi, and P. K. Chawdhry (Eds.), *Advances in Soft Computing – Engineering Design and Manufacturing*, London, U.K.: Springer-Verlag, pp. 315–326.

H. Mannila, H. Toivonen, and A. I. Verkamo (1994) “Efficient Algorithms for Discovering Association Rules,” in *Proc. of the AAAI Workshop on Knowledge Discovery in Databases*, Seattle, WA, pp. 181–192.

H. Mannila, H. Toivonen, and A. I. Verkamo (1995) “Discovering Frequent Episodes in Sequences,” in *Proc. of the 1st Int’l Conf. on Knowledge Discovery and Data Mining*, Montreal, Canada, pp. 210–215.

C. J. Matheus, P. K. Chan, and G. Piatetsky-Shapiro (1993) “Systems for Knowledge Discovery in Databases,” *IEEE Trans. on Knowledge and Data Engineering*, vol. 5, no. 6, pp. 903–913.

C. J. Matheus, G. Piatetsky-Shapiro, and D. McNeill (1996) “Selecting and Reporting What is Interesting: The KEFIR Application to Healthcare Data,” in [Fayyad *et al.* 1996], pp. 495–515.

A. D. McAulay and J. C. Oh (1994) “Improving Learning of Genetic Rule-Based Classifier Systems,” *IEEE Trans. on Systems, Man, and Cybernetics*, vol. 24, no. 1, pp. 152–159.

J. B. McQueen (1967) “Some Methods for Classification and Analysis of Multivariate Observations,” in *Proc. of the 5th Berkeley Symp. on Mathematical Statistics and Probability*, Berkeley, CA, pp. 281–297.

M. Mehta, R. Agrawal, and J. Rissanen (1996) “SLIQ: A Fast Scalable Classifier for Data Mining,” in *Proc. of the 5th Int’l Conf. on Extending Database Technology*, Avignon, France, pp. 18–32.

J. M. Mendel (1995) “Fuzzy Logic Systems for Engineering: A Tutorial,” *Proc. of the IEEE*, vol. 83, no. 3, pp. 345–377.

Z. Michalewicz (1996) *Genetic Algorithms + Data Structures = Evolution Programs*, Third, Revised and Extended Ed., New York, NY: Springer-Verlag.

R. S. Michalski, I. Mozetic, J. Hong, and N. Lavrac (1986) “The AQ15 Inductive Learning System: An Overview and Experiments,” in *Proc. of the Int’l Meeting on Advances in*

*Learning*, Orsay, France.

D. Michie, D. J. Spiegelhalter, and C. C. Taylor (Eds.) (1994) *Machine Learning, Neural, and Statistical Classification*, New York, NY: Ellis Horwood.

S. Mitra, S. K. Pal, and P. Mitra (2002) "Data Mining in Soft Computing Framework: A Survey," *IEEE Trans. on Neural Networks*, vol. 13, no. 1, pp. 3–14.

M. C. Mozer, R. Wolniewicz, D. B. Grimes, E. Johnson, and H. Kaushansky (2000) "Predicting Subscriber Dissatisfaction and Improving Retention in the Wireless Telecommunications Industry," *IEEE Trans. on Neural Networks*, vol. 11, no. 3, pp. 690–696.

S. N. Mukherjee, P. Sykacek, S. J. Roberts, and S. J. Gurr (2003) "Gene Ranking Using Bootstrapped P-Values," *SIGKDD Explorations*, vol. 5, no. 2, pp. 16–22.

M. O. Noordewier, G. G. Towell, and J. W. Shavlik (1991) "Training Knowledge-Based Neural Networks to Recognize Genes in DNA Sequences," in R. P. Lippmann, J. E. Moody, and D. S. Touretzky (Eds.), *Advances in Neural Information Processing Systems*, vol. 3, San Mateo, CA: Morgan Kaufmann.

P. J. Pacini and B. Kosko (1992) "Adaptive Fuzzy Systems for Target Tracking," *Intelligent Systems Engineering*, vol. 1, no. 1, pp. 3–21.

N. R. Pal and J. C. Bezdek (1995) "On Cluster Validity for the Fuzzy c-Means Model," *IEEE Trans. on Fuzzy Systems*, vol. 3, no. 3, pp. 370–379.

W. Pan (2002) "A Comparative Review of Statistical Methods for Discovering Differentially Expressed Genes in Replicated Microarray Experiments," *Bioinformatics*, vol. 18, no. 4, pp. 546–554.

J. S. Park, M.-S. Chen, and P. S. Yu (1995a) "An Effective Hash-Based Algorithm for Mining Association Rules," *in Proc. of the ACM SIGMOD Int'l Conf. on Management of Data*, San Jose, CA, pp. 175–186.

J. S. Park, M.-S. Chen, and P. S. Yu (1995b) "Efficient Parallel Data Mining for Association Rules," in *Proc. of the 4th Int'l Conf. on Information and Knowledge Management*, Baltimore, MD, pp. 31–36.

S. Park, S. Kim, and W. W. Chu (2001) "Segment-Based Approach for Subsequence Searches in Sequence Databases," in *Proc. of the 16th ACM Symp. on Applied Computing*, Las Vegas, NV, pp. 248–252.

A. Paterson and T. B. Niblett (1982) *ACLS Manual*, Edinburgh: Intelligent Terminals Ltd.

W. Pedrycz (2002) “Collaborative Fuzzy Clustering,” *Pattern Recognition Letters*, vol. 23, pp. 1675–1686.

W. Pedrycz and F. Gomide (1998) *An Introduction to Fuzzy Sets: Analysis and Design*, Cambridge, MA: The MIT Press.

G. Piatetsky-Shapiro (1991) “Discovery, Analysis, and Presentation of Strong Rules,” in [Piatetsky-Shapiro and Frawley 1991], pp. 229–248.

G. Piatetsky-Shapiro and W. J. Frawley (Eds.) (1991) *Knowledge Discovery in Databases*, Menlo Park, CA; Cambridge, MA: AAAI/MIT Press.

G. Piatetsky-Shapiro, T. Khabaza, and S. Ramaswamy (2003) “Capturing Best Practice for Microarray Gene Expression Data Analysis,” in *Proc. of the 9th ACM SIGKDD Int’l Conf. on Knowledge Discovery and Data Mining*, Washington, DC, pp. 407–415.

B. Postlethwaite (1991) “Empirical Comparison of Methods of Fuzzy Relational Identification,” *IEE Proc. on Control Theory and Applications*, vol. 138, no. 3, pp. 199–206.

K. B. Pratt and E. Fink (2002) “Search for Patterns in Compressed Time Series,” *Int’l J. of Image and Graphics*, vol. 2, no. 1, pp. 86–106.

A. L. Prodromidis, P. K. Chan, and S. J. Stolfo (2000) “Met-Learning in Distributed Data Mining Systems: Issues and Approaches,” in H. Kargupta and P. Chan (Eds.), *Advances in Distributed and Parallel Knowledge Discovery*, Menlo Park, CA; Cambridge, MA: AAAI/MIT Press, pp. 79–112.

J. R. Quinlan (1986) “Induction of Decision Trees,” *Machine Learning*, vol. 1, pp. 81–106.

J. R. Quinlan (1987a) “Simplifying Decision Trees,” *Int’l J. of Man-Machine Studies*, vol. 27, pp. 221–234.

J. R. Quinlan (1987b) “Decision Trees as Probabilistic Classifiers,” in *Proc. of the 4th Int’l Workshop on Machine Learning*, Irvine, CA, pp. 31–37.

J. R. Quinlan (1993) *C4.5: Programs for Machine Learning*, San Mateo, CA: Morgan Kaufmann.

A.-P. N. Refenes, A. N. Burgess, and Y. Bentz (1997) “Neural Networks in Financial Engineering: A Study in Methodology,” *IEEE Trans. on Neural Networks*, vol. 8, no. 6, pp. 1222–1267.

J. Ribeiro, K. Kaufman, and L. Kerschberg (1995) “Knowledge Discovery from Multiple Databases,” in *Proc. of the 1st Int’l Conf. on Knowledge Discovery and Data Mining*,

Montreal, Canada, pp. 240–245.

J. F. Roddick and M. Spiliopoulou (2002) “A Survey of Temporal Knowledge Discovery Paradigms and Methods,” *IEEE Trans. on Knowledge and Data Engineering*, vol. 14, no. 4, pp. 750–767.

D. Rumelhart, G. Hinton, and J. Williams (1986) “Learning Internal Representations by Error Propagation,” in D. Rumelhart and J. McClelland (Eds.), *Parallel Distributed Processing*, Cambridge, MA: MIT Press, pp. 318–362.

E. H. Ruspini (1969) “A New Approach to Fuzzy Clustering,” *Information and Control*, vol. 15, pp. 22–32.

A. Savasere, E. Omiecinski, and S. Navathe (1995) “An Efficient Algorithm for Mining Association Rules in Large Databases,” in *Proc. of the 21st Int’l Conf. on Very Large Data Bases*, Zurich, Switzerland, pp. 432–444.

G. Shafer (1976) *Mathematical Theory of Evidence*, Princeton, NJ: Princeton University Press.

J. Shafer, R. Agrawal, and M. Mehta (1996) “SPRINT: A Scalable Parallel Classifier for Data Mining,” in *Proc. of the 22nd Int’l Conf. on Very Large Data Bases*, Mumbai (Bombay), India, pp. 544–555.

A. Silberschatz, M. Stonebraker, and J. Ullman (1996) “Database Research: Achievements and Opportunities into the 21st Century,” *SIGMOD Record*, vol. 25, no. 1, pp. 52–63.

A. Silberschatz and A. Tuzhilin (1996) “What Makes Patterns Interesting in Knowledge Discovery Systems,” *IEEE Trans. on Knowledge and Data Engineering*, vol. 8, no. 6, pp. 970–974.

R. Simon (2003) “Supervised Analysis When the Number of Candidate Features (p) Greatly Exceeds the Number of Cases (n),” *SIGKDD Explorations*, vol. 5, no. 2, pp. 31–36.

S. Smith (1983) “Flexible Learning of Problem Solving Heuristics through Adaptive Search,” in *Proc. of the 8th Int’l Joint Conf. on Artificial Intelligence*, Karlsruhe, Germany, pp. 422–425.

J. W. Smith, J. E. Everhart, W. C. Dickson, W. C. Knowler, and R. S. Johannes (1988) “Using the ADAP Learning Algorithm to Forecast the Onset of Diabetes Mellitus,” in *Proc. of the Symp. on Computer Applications and Medical Cares*, pp. 261–265.

P. Smyth and R. M. Goodman (1992) “An Information Theoretic Approach to Rule Induction from Databases,” *IEEE Trans. on Knowledge and Data Engineering*, vol. 4, no. 4, pp. 301–216.

M. Sniedovich (1992) *Dynamic Programming*, New York, NY: Marcel Dekker, Inc.

M. Spiliopoulou and J. F. Roddick (2000) "Higher Order Mining: Modelling and Mining the Results of Knowledge Discovery," in N. F. F. Ebecken and C. A. Brebbia (Eds.), *Data Mining II – Proc. of the 2nd Int'l Conf. on Data Mining Methods and Databases for Engineering, Finance, and Other Fields*, Southampton, U.K.: WIT Press, pp. 309–320.

R. Srikant and R. Agrawal (1995) "Mining Generalized Association Rules," in *Proc. of the 21st Int'l Conf. on Very Large Data Bases*, Zurich, Switzerland, pp. 407–419.

R. Srikant and R. Agrawal (1996) "Mining Quantitative Association Rules in Large Relational Tables," in *Proc. of the ACM SIGMOD Int'l Conf. on Management of Data*, Montreal, Canada, pp. 1–12.

A. Srivastava, E.-H. Han, V. Kumar, and V. Singh (1998) "Parallel Formulations of Decision-Tree Classification Algorithms," in *Proc. of the Int'l Conf. on Parallel Processing*, Minneapolis, MN, pp. 237–244.

Z. Struzik and A. Siebes (1999) "The Haar Wavelet Transform in the Time Series Similarity Paradigm," in *Proc. of the 3rd European Conf. on Principles of Data Mining and Knowledge Discovery*, Prague, Czech Republic, pp. 12–22.

J. Sun and X.-Z. Wang (2005) "An Initial Comparison on Noise Resisting between Crisp and Fuzzy Decision Trees," in *Proc. of the 4th Int'l Conf. on Machine Learning and Cybernetics*, Guangzhou, China, pp. 2545–2550.

P. Tamayo, D. Solni, J. Mesirov, Q. Zhu, S. Kitareewan, E. Dmitrovsky, E. S. Lander, and T. R. Golub (1999) "Interpreting Patterns of Gene Expression with Self-Organizing Maps: Methods and Application to Hematopoietic Differentiation," *Proc. of the National Academy of Sciences of the United States of America*, vol. 96, no. 6, pp. 2907–2912.

R. Tanese (1987) "Parallel Genetic Algorithm for a Hypercube," in *Proc. of the 2nd Int'l Conf. on Genetic Algorithms*, pp. 177–183.

J. D. Ullman (1988) *Principles of Database and Knowledge-Base Systems*, vol. 1, Rockville, MD: Computer Science Press.

R. E. Walpole and R. H. Myers (1993) *Probability and Statistics for Engineers and Scientists*, 5th Ed., Upper Saddle River, NJ: Prentice-Hall.

C. Wang and X. S. Wang (2000) "Supporting Content-Based Searches on Time Series via Approximation," in *Proc. of the 12th Int'l Conf. on Scientific and Statistical Database Management*, Berlin, Germany, pp. 69–81.

C. C. Wang and A. K. C. Wong (1979) "Classification of Discrete-Valued Data with

Feature Space Transformation," *IEEE Trans. on Automatic Control*, vol. AC-24, no. 3, pp. 434–437.

A. S. Weigend, B. A. Huberman, and D. E. Rumelhart (1990) "Predicting the Future: A Connectionist Approach," *Int'l J. of Neural Systems*, vol. 1, pp. 193–209.

P. J. Werbos (1974) "Beyond Regression: New Tools for Predicting and Analysis in the Behavioral Sciences," *Ph.D. Thesis*, Harvard University, Cambridge, MA.

I. H. Witten and E. Frank (2005) *Data Mining: Practical Machine Learning Tools and Techniques*, 2nd Ed., San Francisco, CA: Morgan Kaufmann.

A. K. C. Wong and D. K. Y. Chiu (1987) "Synthesizing Statistical Knowledge from Incomplete Mixed-Mode Data," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. PAMI-9, no. 6, pp. 796–805.

A. K. C. Wong and T. S. Liu (1975) "Typicality, Diversity, and Feature Patterns of an Ensemble," *IEEE Trans. on Computers*, vol. c-24, no. 2, pp. 158–181.

A. K. C. Wong, T. S. Liu, and C. C. Wang (1976) "Statistical Analysis of Residue Variability in Cytochrome C," *Journal of Molecular Biology*, vol. 102, pp. 287–295.

A. K. C. Wong and Y. Wang (1997) "High-Order Pattern Discovery from Discrete-Valued Data," *IEEE Trans. on Knowledge and Data Engineering*, vol. 9, no. 6, pp. 877–893.

A. K. C. Wong and Y. Wang (2003) "Pattern Discovery: A Data Driven Approach to Decision Support," *IEEE Trans. on Systems, Man, and Cybernetics – Part C: Applications and Reviews*, vol. 33, no. 1, pp. 114–124.

S. Wrobel (1997) "An Algorithm for Multi-Relational Discovery of Subgroups," in *Proc. of the 1st European Symp. on Principles of Data Mining and Knowledge Discovery*, Trondheim, Norway, pp. 367–375.

X. Wu (1995) *Knowledge Acquisition from Data Bases*, Norwood, NJ: Ablex.

X. Wu (1999) "Fuzzy Interpretation of Discretized Intervals," *IEEE Trans. on Fuzzy Systems*, vol. 7, no. 6, pp. 753–759.

X. Wu and S. Zhang (2003) "Synthesizing High-Frequency Rules from Different Data Sources," *IEEE Trans. on Knowledge and Data Engineering*, vol. 15, no. 2, pp. 353–367, 2003.

H. Xie, Y. C. Lee, R. L. Mahajan, and R. Su (1994) "Process Optimization Using a Fuzzy Logic Response Surface Method," *IEEE Trans. on Components, Packaging, and Manufacturing Technology – Part A*, vol. 17, no. 2, pp. 202–211.

E. P. Xing, M. I. Jordan, and R. M. Karp (2001) “Feature Selection for High-Dimensional Genomic Microarray Data,” in *Proc. of the 18th Int’l Conf. on Machine Learning*, Williamstown, MA, pp. 601–608.

R. R. Yager (1991) “On Linguistic Summaries of Data,” in [Piatetsky-Shapiro and Frawley 1991], pp. 347–363.

C.-H. Yang and K. E. Nygard (1993) “The Effects of Initial Population in Genetic Search for Time Constrained Traveling Salesman Problems,” in *Proc. of the ACM Conf. on Computer Science*, Indianapolis, IN, pp. 378–383.

J. Yao and H. Liu (1997) “Searching Multiple Databases for Interesting Complexes,” in *Proc. of the 1st Pacific-Asia Conf. on Knowledge Discovery and Data Mining*, Singapore, pp. 198–210.

J. Yen (1999) “Fuzzy Logic – A Modern Perspective,” *IEEE Trans. on Knowledge and Data Engineering*, vol. 11, no. 1, pp. 153–165.

J. Yen and R. Langari (1999) *Fuzzy Logic: Intelligence, Control, and Information*, Upper Saddle River, NJ: Prentice-Hall.

L. Yu and H. Liu (2004) “Redundancy Based Feature Selection for Microarray Data,” in *Proc. of the 10th ACM SIGKDD Int’l Conf. on Knowledge Discovery and Data Mining*, Seattle, WA, pp. 737–742.

M. J. Zaki, C.-T. Ho, and R. Agrawal (1999) “Parallel Classification for Data Mining on Shared-Memory Multiprocessors,” in *Proc. of the 15th Int’l Conf. on Data Engineering*, Sydney, Australia, pp. 198–205.

T. Zhang, R. Ramakrishnan, and M. Livny (1996) “BIRCH: An Efficient Data Clustering Method for Very Large Databases,” in *Proc. of the ACM SIGMOD Int’l Conf. on Management of Data*, Montreal, Canada, pp. 103–114.

S. Zhang, X. Wu, and C. Zhang (2003) “Multi-Database Mining,” *IEEE Computational Intelligence Bulletin*, vol. 2, no. 1, pp. 5–13.

H. Zhang, C. Y. Yu, B. Singer, and M. Xiong (2001) “Recursive Partitioning for Tumor Classification with Gene Expression Microarray Data,” *Proc. of the National Academy of Sciences of the United States of America*, vol. 98, no. 12, pp. 6730–6735.

S. Zhang, C. Zhang, and X. Wu (2004) *Knowledge Discovery in Multiple Databases*, London, U.K.: Springer-Verlag.

N. Zhong, Y. Yao, and S. Ohsuga (1999) “Peculiarity Oriented Multi-Database Mining,” in *Proc. of the 3rd European Conf. on Principles of Data Mining and Knowledge*

*Discovery*, Prague, Czech Republic, pp. 136–146.