MINING FUZZY ASSOCIATION RULES IN LARGE DATABASES WITH QUANTITATIVE ATTRIBUTES

BY

KUOK, CHAN MAN

SUPERVISED BY :

PROF. ADA FU & PROF. WONG, MAN HON

A THESIS

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF PHILOSOPHY

DIVISION OF COMPUTER SCIENCE & ENGINEERING

THE CHINESE UNIVERSITY OF HONG KONG

AUGUST 1997

Mining Fuzzy Association Rules in Large Databases with Quantitative Attributes

submitted by

Kuok, Chan Man

for the degree of Master of Philosophy

at the Chinese University of Hong Kong

Abstract

Databases have been used to store huge amount of raw data for a long time. Recently, people are interested in the information hidden in the databases. Different methods have been proposed to discover different kinds of knowledge in the databases. The discovered knowledge can be divided into several categories, e.g. *characterization*, *clustering*, *classification*, *association* and *sequences*. In this thesis, we concentrate on the discovery of association rules. Many algorithms have been proposed to find binary association rules in databases containing only binary attributes.

In real life, however, database will contain not only binary attributes, but also categorical and numerical attributes. There are several mining algorithms [29, 10, 11, 21] trying to find association rules in databases with quantitative attributes. In [29], an attribute domain partitioning algorithm has been proposed to find quantitative association rules. However, there are problems for both discrete and overlapped intervals.

Therefore, we introduce the fuzzy association rule (FAR) to handle quantitative attributes such that we can avoid the problems of [29] as well as give better semantics to the discovered rules. We have introduced new interest measure for itemsets and rules so that we can incorporate the fuzzy set concepts in association rule mining. We use *significance* factor to generate large itemsets and use *certainty* factor to test the usefulness of the discovered fuzzy association rules. The fuzzy association rule is of the form, "If X is A then Y is B", where "X is A" is called antecedent and "Y is B" is called consequent of the rule. X and Y are sets containing attributes of the database. A and B are sets of fuzzy sets which describe X and Y respectively. Using fuzzy set concept, the discovered rules are more understandable to human because of the linguistic terms associated with the fuzzy sets. Moreover, fuzzy sets handle numerical values better than existing methods because fuzzy sets soften the effect caused by partitioning the attribute domain. From experimental results, we can see that fuzzy set concepts really solve the problems caused by sharp boundary. Moreover, one of our algorithms has compatible performance with respect to the interval partitioning method.

Acknowledgments

To my parents and two elder brothers.

My deepest gratitude goes to my advisors, Prof. Ada Fu and Prof. Man-hon Wong for their illuminating and stimulating advice as well as their encouragements.

Last but not least, my friends around me in the office here in Room 1013 and 1026, Ho Sin Hang Engineering Building have made the research life more enjoyable. Tai-man Wong, Lun-hsing Tung, Kwok-wai Law, Sze-kin Lam, Shing-kwong Cheung, Yen-hui Hung, Yuk-yee Leung, Yin-bun Wong, Chi-chiu Cheung, Kingly Cheung and Chiu-fai Fong give their supports in different ways.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

In the past, useful data are written on papers and stored in safe places. When people need to access these data, they will take them out and analyze by human eyes. However, with the rapid growth of businesses and industries, more and more data are generated from growing organizations. It is probably impossible to handle the vast amount of cumulated data by human. Database is one way to solve part of the problem, storing the huge amount of data. In recent years, computing resources become widely available and the price become reasonably affordable. These lead to the increase in the automation of business activities. More and more companies and organizations switch to store their data in databases such as the use of credit or the medical diagnosis. The growth of number and size of databases becomes faster than ever before.

At the very beginning, the role of databases is a container. Users can store their data as well as performing some simple operation, e.g. sorting the records, indexing, issuing simple queries. However, the results of the queries are not surprising to the users. Recently, this concept has been changed. People not only put data into databases, but also want to get some useful and interesting information from databases since the most tedious tasks are performed by computer. As a result, *data mining* has been introduced.

1.1 Data Mining

Data mining or *knowledge discovery* is the tasks of finding *implicit*, *previously unknown* and *potentially useful* information hidden in databases [25]. Given a database, we want to find different kinds of regularities. The mining process and discovered knowledge should possess some special characteristics. The discovered knowledge should be understandable by human so that they can use the knowledge directly. Moreover, the discovered knowledge should be accurate with respect to the database. Otherwise, people may get wrong information from the database. Furthermore, we should guarantee the interestingness of the discovered knowledge in order to fulfill users' expectation. Finally, the mining process should be efficient. We can summarize the characteristics of data mining as follows.

- Understandability

  The representation of the discovered knowledge should be understandable by human. Therefore, if we can incorporate high level language into the discovered knowledge, the expressive power is better.

- Accuracy

  We have to introduce some measuring criteria for the discovered knowledge with respect to the database. Otherwise, the discovered knowledge may be incomplete or misleading.

- Interestingness

  In the mining process, we have to take the users' preference into account. The discovered knowledge is interesting if it satisfies the user-specified thresholds. Otherwise, we may spend time on discovering useless or trivial information.

- Efficiency

We have to guarantee the mining process is efficient. The performance of mining process should not degrade exponentially as the database size increases.

There are several kinds of knowledge hidden in databases and we can classify the discovered knowledge into three categories, e.g. *classification*, *association* and *sequences*. *Classification* tries to classify the given data set into different classes in order to reduce the database size. In [25], two of the tasks are *summarization* and *discrimination*. *Summarization* searches for common characteristic features of one class. However, *discrimination* tries to distinguish different classes. *Sequences* are patterns in time series data. In sequence data search, algorithms have been proposed to find exact or similar patterns. For example, we can find patterns in weather and stock market data. *Association* represents the knowledge in the form of a rule, i.e. $X \rightarrow Y$. Example can be found in retail data.

1.2 Association Rule Mining

Many algorithms have been proposed in finding classification. However, mining association rules is still a new research area in data mining.

During the past years, *boolean association rule* mining [3, 4, 23, 9, 26] has received considerable attention. Boolean association rule mining tries to find consumer behavior in retail data which are consumer transactions stored in databases of large retail companies. The domain of retail data is binary which indicates whether a customer has bought an item or not. Therefore, the discovered rule can tell, e.g. people who buy butter and milk will also buy bread. Such rules can be used in customizing marketing program, advertisement and sales promotion. However, binary association rule mining limits the application area of the mining process since only binary attributes will be considered.

In practice, database will contain not only binary, but also categorical and numerical values. We cannot directly apply the algorithms of binary association rule to deal with attributes other than binary ones. In [29], mining *quantitative association rule* has been proposed. The algorithm first partitions the attribute domains into small intervals and combines adjacent intervals into larger one such that the combined intervals will have enough supports. Replacing the original attribute by its attribute-interval pairs, the quantitative problem has been transformed to binary one. Although quantitative association rule mining can solve the problem introduced by numerical attributes, there are still other problems. We observe that one problem of partitioning is the sharp boundary between intervals. The algorithm either ignore or over emphasize the elements near the boundary of an interval in the mining process. However, the interval method is not intuitive with respect to human mind. For example, the interval method may classify a person is *young* if age is less than 40 and *old* if age is greater than 40. The first problem is how we can classify a person whose age is 40. The second problem is how we can distinguish the degree of membership. For example, using the interval method, age of 45 and age of 90 will both be classified into *old*. By intuition, however, we know that ago of 90 is older than age of 45. They should not contribute the same to the interval. Therefore, we look for other ways to handle the quantitative attributes.

In this thesis, we propose algorithm for mining *fuzzy association rule* of the form, "If X is A then Y is B". "X is A" is called the antecedent of the rule and "Y is B" is called the consequent of the rule. X, Y are sets of attributes of database and A, B are sets containing fuzzy sets which characterize X and Y respectively. We use the fuzzy set concept to deal with the numerical attribute of the database so that we can also solve the problem of infinite values of numerical attribute. Moreover, we have also solved the problem of sharp boundary introduced by partitioning of attribute domain. It is because the fuzzy sets provide a smooth transition between member and non-member of a set so that no sharp boundary exists. Moreover, the fuzzy association rule is more understandable because the semantics of the rule can be reflected clearly by the linguistic terms associated with the fuzzy sets.

The thesis is organized as follows. In chapter 2, we will give some literature survey on different existing association rule mining algorithms. Both binary and quantitative algorithms will be covered. The problem definition will be presented in chapter 3. We will define the fuzzy association rule and the interest measure used in the mining process. After that, in chapter 4, the mining steps for fuzzy association rules and the implementation details will be covered. In chapter 5, we will compare our algorithms with the one in [29]. Moreover, we will have extensive experiments under different user specified thresholds. Chapter 6 is a brief discussion and we will give the conclusion in chapter 7.

Chapter 2

Background

In this chapter, we will first define some general terms for association rule mining. These terms are commonly used in existing association rule mining algorithms. After that, we will describe several existing algorithms in finding binary and quantitative association rules.

2.1 Framework of Association Rule Mining

In [3], association rule mining has been divided into two subproblems. The first one is to find all frequent/large itemsets. The second subproblem is to use the discovered large itemsets to generate interesting rules. Therefore, we will first give the definition of large itemsets and then the definition of association rules.

2.1.1 Large Itemsets

Let $T = \{t_1, t_2, ..., t_n\}$ be the database containing n user *transactions.* Each transaction is represented by t_k which shows the *items* bought by a customer in one purchase. All the possible items which can be appeared in a transaction is represented by $I = \{i_1, i_2, ..., i_m\}$. Any subset of items in I can be contained in a transaction.

For simplicity, given a set of items, we call it an *itemset* instead. For a set of k items, we will call it a *k-itemset.*

For an itemset X in I ($\subset I$), a transaction t_k is said to *support* X, if t_k contains all items in X. The *support* of X, $support(X)$, is defined to be the fraction of all transactions in T that support X. We define the *large itemset* as follows. Given a *minimum support* value, $minsup$, if we have $support(X) \geq minsup$, the itemset X is called *large*. The $minsup$ is a user specified threshold which constrains ourselves to consider only those itemsets that appear often enough in T to be interesting. Those that do not have minimum support are called *small itemsets* and we will not use them to generate association rules.

According to the definition of large itemsets and the support of itemset, we can find some interesting properties of large itemsets which are useful in association rules mining. The properties are as follows.

- Subsets of Itemset

  Given two itemsets X and Y, if X is a subset of Y, then we can conclude that the support of X is not less than the support of Y, i.e. $support(X) \geq support(Y)$. The reason is that all transactions in T that support Y need to support X, too.

- Supersets of Small Itemset

  Given a small itemset X in T, i.e. $support(X) < minsup$, every superset Y of X will also be small itemsets because $support(Y) \leq support(X) < minsup$ according to the above subset property.

- Subsets of Large Itemset

Given a large itemset X in T, i.e. $support(X) \geq minsup$, every subset Y of X is also large itemsets because $support(Y) \geq support(X) \geq minsup$ according to the subset property of itemset. However, it is not necessarily true for the supersets of a large itemset.

2.1.2 Association Rules

An association rule is of the form, $X \rightarrow Y$, where X and Y are disjoint itemsets, i.e. $X, Y \subset I$ and $X \cap Y = \emptyset$. X is called the antecedent and Y is called the consequent. The semantics of the rule is an implication, given those transactions that support X, a fraction of these transactions will also support Y. The fraction of the transactions is called the *confidence* of the rule and is denoted by $confidence(X \rightarrow Y)$.

The confidence of a rule reflects the number of transactions which support X will also support Y and can be calculated as follows.

$$confidence(X \rightarrow Y) = \frac{support(X \cup Y)}{support(X)}$$

Both support and confidence is important to a rule. The support of a rule, $X \rightarrow Y$, is denoted by $support(X \cup Y)$ which shows the fraction of transactions containing both X and Y. The support of a rule can reflects the importance of the rule. On the other hand, the confidence shows the strength of the rule. We say a rule holds if it has support and confidence greater than or equal to the user specified thresholds, *minsup* and *minconf* respectively. The *minconf* is the minimum confidence level that a rule has to attained. Furthermore, according to the property of large itemset, the antecedent and consequent of the rule should also be large itemsets such that the rule will hold.

We use some examples to illustrate that a rule with only enough support or only sufficient confidence will not hold. If there is only one transaction in

T supporting a rule, we will have 100% confidence level. However, we do not have enough support for this rule since only one transaction contains the rule. Therefore, the rule is not likely to appear often in the database T. On the other hand, a rule with insufficient confidence may not be true. Suppose there are 10,000 transactions that support the itemset X and Y respectively and only 100 transactions that support both X and Y. Using the formula of confidence, the confidence of the rule is only 1% which is probably too low for a rule to be true. From these examples, it is clear that a rule will hold only with enough support as well as sufficient confidence.

The following properties of association rule are also important in association rule mining. Without using these properties, we may discover some duplicated rules, or even worse some incorrect rules.

- Triviality

  In the definition of association rules, we have mentioned that the antecedent and consequent should be disjoint itemsets, i.e. $X \cap Y = \emptyset$. We now show why this restriction is necessary to the association rules. Given X with enough support, $X \rightarrow X$ will hold with 100%. This kind of rules, however, is not interesting because it is trivial to us. It gives us no surprising information. Moreover, given $X \rightarrow X \cup Y$, we know that it is also not interesting because it is same as $X \rightarrow Y$. Therefore, we let X and Y be disjoint itemsets such that there is no trivial or duplicated rule.

- Composition

  The composition of association rules with either same consequent or antecedent is not possible. Given $X \rightarrow Z$ and $Y \rightarrow Z$ with enough support and sufficient confidence, however, we cannot imply $X \cup Y \rightarrow Z$ holds. It is because we only have the supports of the itemsets X, Y, Z, $X \cup Z$ and $Y \cup Z$. With only these values, we cannot know what is the exact value of the support of $X \cup Y \cup Z$ according to the property of large itemset. Therefore, we have to count the support of $X \cup Y \cup Z$ in order to determine whether $X \cup Y \rightarrow Z$ holds.

In a similar sense, the composition of rules with the same antecedent is shown to be impossible, i.e. given $X \rightarrow Y$ and $X \rightarrow Z$, we also cannot imply $X \rightarrow Y \cup Z$ holds.

- Decomposition

  The decomposition of association rules is also impossible or useless. We first describe the impossibility of decomposition of the antecedent. If $X \cup Y \rightarrow Z$ holds, $X \rightarrow Z$ and $Y \rightarrow Z$ may not hold. For example, if Z appears in a transaction only if both X and Y are present, i.e. $support(X \cup Y) = support(Z)$. Then, if the support of X and Y is sufficiently greater than $X \cup Y$. $X \rightarrow Z$ and $Y \rightarrow Z$ do not have enough confidence since support of X and Y is also sufficiently greater than support of Z.

  Nevertheless, given $X \rightarrow Y \cup Z$ with the required *minsup* and *minconf*, then $X \rightarrow Y$ and $X \rightarrow Z$ hold in T. It is because both $support(X \cup Y)$ and $support(X \cup Z)$ are greater than $support(X \cup Y \cup Z)$. Therefore, the resulting rules have support and confidence greater than the user specified thresholds. However, the rules with more items in both antecedent and consequent are more desirable. Therefore, the decomposition of association rules is not very useful in this sense.

- Transitivity

  We will show that the transitivity of rules does not hold in association rules. Even we know that $X \rightarrow Y$ and $Y \rightarrow Z$ hold in T, we cannot infer $X \rightarrow Z$ holds. Assume all transactions containing Z will also contain Y and all transactions containing Y will also contain X. Let $confidence(X \rightarrow Y) = confidence(Y \rightarrow Z) = minconf$. The following equations show how we can get the confidence of $X \rightarrow Z$.

$$\begin{aligned}
confidence(Y \rightarrow Z) &= minconf \\
\frac{support(Y \cup Z)}{support(Y)} &= minconf \\
\frac{support(Z)}{support(Y)} &= minconf \\
support(Y) &= \frac{support(Z)}{minconf} \\
confidence(X \rightarrow Y) &= minconf \\
\frac{support(X \cup Y)}{support(X)} &= minconf \\
\frac{support(Y)}{support(X)} &= minconf \\
\frac{support(Z)}{support(X)} &= minconf^2 \\
\frac{support(X \cup Z)}{support(X)} &= minconf^2 \\
confidence(X \rightarrow Z) &= minconf^2
\end{aligned}$$

In the above equations, we know that the confidence of $X \rightarrow Z$ is $minconf^2$. Since $minconf < 1$, $X \rightarrow Z$ does not have enough confidence. Therefore, there is no transitivity in association rules.

2.2 Association Rule Algorithms For Binary Attributes

As mentioned above, association rule mining algorithms consist of generation of large itemsets and construction of rules from the discovered large itemsets. However, the rule construction is a straightforward procedure when we have found the necessary large itemsets. Therefore, the discovery of large itemsets is the most interesting part in association rule mining and different existing algorithms find large itemsets differently in order to give better performance. The algorithms use one of the following data representations to store the database, i.e. item-lists, candidate-lists and TID-lists. We will describe how these representations influence the performance of individual algorithms.

2.2.1 AIS

In [3], the problem of association rule mining was first introduced. Moreover, the algorithm is called AIS in [4]. In AIS, there are several passes and the entire database has to be read one transaction after the other in each pass. A candidate itemset is generated by extending *frontier sets* which are large itemsets in the previous passes. Initially, the frontier set is an empty set in the first pass. A candidate itemset is called a *k-extension* of a frontier set F if the candidate itemset is created by adding k items to F. As mentioned above, we assume all the items are sorted. Therefore, we can avoid duplicate candidate itemsets by adding items which are larger than the largest item in F. Moreover, we will not extend F with all items. On the contrary, we will only consider items appeared in the transactions. Given a transaction $\{1, 2, 3, 5, 6, 7, 8\}$ and a frontier set $F = \{2, 3, 5\}$, 1-extension of F should be $\{2, 3, 5, 6\}$, $\{2, 3, 5, 7\}$ and $\{2, 3, 5, 8\}$. We can see that 1 does not appear in the 1-extension of F because it is smaller than the largest item in F. The 2-extension of F can be generated in similar way, i.e. $\{2, 3, 5, 6, 7\}$, $\{2, 3, 5, 6, 8\}$ and $\{2, 3, 5, 7, 8\}$. Initially, F is an empty set. Therefore, the candidate 1-itemsets are those items appeared in all transactions.

Each candidate itemset is associated with a counter. When it is first created, the counter will be set to 1. After that, the counter will be incremented when subsequent transactions contain that candidate itemset. Using the counter, we can calculate the support of the candidate itemset after a complete pass through all transactions. If the candidate itemset has support greater than the minimum support threshold, it will become the new frontier set in next pass. After the new frontier sets have been generated, a new pass begins unless there is no large itemset in the previous pass.

However, this algorithm will generate a large number of candidate itemsets which are expected to be large and turn out to be small. Therefore, sophisticated pruning techniques are necessary to decide whether an extension should be included in the candidate itemsets. The algorithm uses the relative frequencies of items of candidate itemset to calculate its expected support such that a candidate itemset expected to be small will not be extended. The pruning function optimization is based on the total transaction price. If the price of extending an itemset is too high, it cannot possibly be large.

2.2.2 SETM

In SETM [16], it stores the database, candidate and large itemsets in the form, ⟨TID,itemset⟩. SETM will modify the entire database in each pass to perform candidate itemset generation, counting support and find out the large itemsets. Therefore, the database will contain candidate itemsets or large itemsets. Table 2.1 is an example which illustrates the generation of candidate and large itemsets. The minimum support has been set to 2 records.

The procedures of generating candidate and large itemsets are as follows. Suppose we have the (k-1)-itemsets, we first have to sort the large (k-1)-itemsets by the TIDs such that the creation of the candidate k-itemsets of a specific TID can be done by joining the large (k-1)-itemset pairs, which must have (k-2) items in common, of that TID. After obtaining the candidate k-itemsets, we sort the records by itemsets such that we can delete the small k-itemsets by counting the number of TIDs of the k-itemsets. Therefore, we have our large k-itemsets ready for the generation of candidate (k+1)-itemsets. All size of candidate and large

Database

TID	Items
1	2,3,5
2	1,2,3,5
3	2,5

L_1

TID	Items
1	2
1	3
1	5
2	2
2	3
2	5
3	2
3	5

C_2

TID	Items
1	2,3
1	2,5
1	3,5
2	2,3
2	2,5
2	3,5
3	2,5

L_2

TID	Items
1	2,3
1	2,5
1	3,5
2	2,3
2	2,5
2	3,5
3	2,5

C_3

TID	Items
1	2,3,5
2	2,3,5

L_3

TID	Items
1	2,3,5
2	2,3,5

Table 2.1: Example of SETM.

itemsets are generated in similar fashion.

The advantage of SETM is that it generates fewer candidate itemsets than AIS. For example, if AIS use L_2 as frontier sets, it will generate the candidate itemsets {1, 2, 3}, {1, 2, 5} and {1, 3, 5} according to the transaction with TID 2. However, these itemsets are absolutely small since only one record contains item 1. SETM will never generate those candidate itemsets since the item 1 has already been dropped in L_1.

2.2.3 Apriori, AprioriTid and AprioriHybrid

Apriori, AprioriTid and AprioriHybrid are introduced in [4]. They outperform the previous algorithms since they use a new candidate itemset generation method called *Apriori-gen* function. Apriori uses *item-list* to store the database and AprioriTid uses *candidate-list* to discard useless data in the database. We will briefly describe the candidate generation function followed by the description of the three algorithms and their data representation.

Apriori-gen

The *Apriori-gen* function uses the information of previous passes and the property of subsets of large itemsets to generate the candidate itemsets of current pass. If we want to generate a k-candidate itemset, we will use the large (k-1)-itemsets as input and search for pairs of itemsets that have their smallest (k-2) items in common. The candidate itemset is the union of two large itemsets. All its (k-1)-subsets should be large. Otherwise, the itemset will not be considered as candidate itemset. For example, given two large itemsets {1,2} and {1,3}, we can generate a candidate itemset {1,2,3}. We have to check whether its subset {2,3} is large before putting it into the set of candidate itemsets.

We only use Apriori-gen to generate candidate itemsets and the counting step will not be performed in that function. The advantage of Apriori-gen is that it generates fewer candidate itemsets and the candidate itemsets will not be created repeatedly for every transaction.

Apriori

Apriori uses Apriori-gen to generate all size of candidate itemsets. In each pass, Apriori calls Apriori-gen to generate all candidate itemsets of a given size, e.g. candidate k-itemsets in k^{th} pass. Then, the support of all these candidate k-itemsets will be obtained in the counting phase. In each counting phase, the entire database will be scanned.

Apriori stores the data in item-list representation which is of the form, $\langle$TID,list of items$\rangle$. The transactions are stored as a sequence of sorted item-lists in this representation. Since Apriori has to read the entire database in each pass, if the useless items and transactions can be deleted, the performance will be better. However, the item-list representation prevents Apriori to do this optimization. We have to scan the database once more in order to get the knowledge of which items and transactions to discard. Nevertheless, the item-list representation has its own advantage. The database size will not grow through out the algorithm.

AprioriTid

To solve the shortcoming of Apriori, AprioriTid was proposed to prune the useless items and transactions during the process of the algorithm. AprioriTid also uses Apriori-gen to produce all candidate itemsets. However, the database is stored using candidate-list representation. Each transaction is associated with a list of candidate itemsets which are supported by it. Table 2.2 is an example of candidate-list representation. AprioriTid will still use the initial database to generate candidate and large 1-itemsets. After the first pass, initial database will be transformed similar to table 2.2. In each pass, the generation of candidate and large itemsets will refer to the new database generated in the previous pass.

Database

TID	Items
1	2,3,5
2	1,2,3,5
3	2,5

C_1

TID	Candidate Itemsets
1	{2}, {3}, {5}
2	{1}, {2}, {3}, {5}
3	{2}, {5}

C_2

TID	Candidate Itemsets
1	{2,3}, {2,5}, {3,5}
2	{2,3}, {2,5}, {3,5}
3	{2,5}

C_3

TID	Candidate Itemsets
1	{2,3,5}
2	{2,3,5}

Table 2.2: Candidate-list representation.

The disadvantage of candidate-list is that the size of intermediate results will be unexpected large, although it is much smaller than SETM. These intermediate results may be too large to fit into main memory such that the data may be needed to swap to disk. However, this situation will usually happen in second pass. After the second pass, the intermediate results are expected to be rather small. Moreover, the advantage of candidate-list is that the useless items and transactions are pruned automatically. AprioriTid outperforms Apriori after second pass.

AprioriHybrid

Comparing Apriori and AprioriTid, Apriori outperforms AprioriTid in early passes since Apriori avoids swapping data to disk. In later passes, however, AprioriTid gives better performance because it prunes out the useless items and transactions from the database. AprioriHybrid is the combination of Apriori and AprioriTid. For the initial passes, Apriori is employed and is switched to AprioriTid when the new database is expected to fit in main memory. AprioriHybrid performs better than Apriori as long as AprioriTid can be used long enough to compensate the overhead of switching from Apriori to AprioriTid.

2.2.4 PARTITION

In the previous algorithms, the database needs to be scanned several times because the number of possible itemsets to be tested is exponentially large if it must be done in one pass. Moreover, we do not know the number of passes in advance. In [26], an algorithm called PARTITION has been proposed. This algorithm only scans the database two times. In the first scan, it generates all candidate itemsets and count the support of these itemsets in the second scan. Therefore, the number of passes has been limited. Moreover, the algorithm uses the *TID-list* to store the database such that the database size will be pruned in later passes.

The algorithm will first divide the database into equally sized horizontal partitions. For each partition, the algorithm will find the corresponding local candidate and large itemsets. The partition size is chosen such a way that the entire partition can reside in main memory. Therefore, no data has to be swapped to disk.

Since the local large itemsets are only large in their own partitions, we have counted the database once again to obtain the global support of these local large itemsets in order to determine which of them are globally large. In this counting phase, we group all local large itemsets in a so called *global candidate itemsets* which is the set of all local large itemsets. All the itemsets in global candidate itemsets have to be counted, except for the ones that were large in every partition because we already have their counts. After the counting phase, we will have our *global large itemsets* by pruning away those itemsets that do not have enough supports. The total IO requirement is only two scans of the database.

The PARTITION algorithm reduces the database size by using the TID-list to store the candidate and large itemsets. TID-list stores the itemsets and the TIDs of the supporting transactions. The representation is of the form, ⟨Itemset,list of TIDs⟩. Table 2.3 is an example of TID-list and the generation of candidate itemsets. The minimum support is 3 records in this example.

Database

TID	Items
1	1,2,3
2	3,4,5,6
3	2,4,6
4	1,3,5
5	3,5,6
6	3,5,6

C_1

Candidate 1-itemsets	TIDs
{1}	1,4
{2}	1,3
{3}	1,2,4,5,6
{4}	2,3
{5}	2,4,5,6
{6}	2,3,5,6

C_2

Candidate 2-itemsets	TIDs
{3,5}	2,4,5,6
{3,6}	2,5,6
{5,6}	2,5,6

C_3

Candidate 3-itemsets	TIDs
{3,5,6}	2,5,6

Table 2.3: TID-list representation.

In table 2.3, the initial database and the intermediate results of AprioriTID are presented. The TID-list for a candidate k-itemset can be obtained by intersecting the TID-lists of its (k-1)-subsets. For example, the TID-list of the itemset {2,3} is the result of intersecting the TID-lists of itemsets {2} and {3}. For efficient computation, the intersection can be done with a merge-join operation.

The intermediate results caused by TID-list are still unpredictable. However, it is less severe than the candidate-list representation since the candidate generation and counting are both done in the merge-join phase. Although TID-list still has the size problem, it removes the useless data in later passes. Both the small itemsets and the useless transactions can be dropped automatically in the merge-join operation.

2.3 Association Rule Algorithms For Numeric Attributes

The previous algorithms are used to deal with database containing binary attributes only. In this section, we talk about the algorithms for mining association rules in databases with numeric attributes.

2.3.1 Quantitative Association Rules

[29] introduces the concept of quantitative association rules such that association rule mining can be performed on databases with numeric attributes. The algorithm is a mapping from quantitative association rules problem into the boolean association rules problem. The mapping is as follows. If the categorical or quantitative attributes have only a few values, the most intuitive and simplest way to handle this kind of attributes is replacing the original attribute by its domain values. Let x be a quantitative attribute in the database and D_x be the domain of x. The value of x is represented by d_x^i such that $D_x = \{d_x^1, d_x^2, ..., d_x^n\}$. In this simple method, we will replace x by its *attribute-value* pairs, i.e. $\langle x, d_x^1\rangle, \langle x, d_x^2\rangle, ..., \langle x, d_x^n\rangle$. Table 2.4 is a sample database with two attributes.

In table 2.5, we can see that $Children$ is replaced by its *attribute-value* pair, i.e. $\langle Children, 0\rangle$, $\langle Children, 1\rangle$, $\langle Children, 2\rangle$, $\langle Children, 3\rangle$. Each of these

Retired	*Children*
Yes	2
No	3
No	0
No	1
Yes	2

Table 2.4: A Sample Database For Small Attribute Domain.

attribute-value pairs will be considered as binary attributes in the database. In table 2.4 and table 2.5, both databases have the same number of records. However, the number of attributes in table 2.5 has been increased from 2 to 5. The increment of number of attribute can be calculated by a simple formula. If we have k quantitative attributes, we can use the following equation to calculate the number of attributes increased:

$$\sum_{i=1}^{k}(|D_i| - 1)$$

In the above equation, $|D_i|$ represents the cardinality of the domain of the i^{th} quantitative attribute. We can see that number of attribute increased as the size of domain grows. The result of the above formula will increase to infinity if one of $|D_i|$ represents the domain of a numerical attribute. Therefore, this method can only apply to finite and small attribute domains. We need other methods to solve the problem caused by infinite domain.

Retired	$\langle Children, 0\rangle$	$\langle Children, 1\rangle$	$\langle Children, 2\rangle$	$\langle Children, 3\rangle$
1	0	0	1	0
0	0	0	0	1
0	1	0	0	0
0	0	1	0	0
1	0	0	1	0

Table 2.5: Resulting Database After Discrete Value Replacement.

For a large attribute domain, an obvious approach is to first partition the attribute domain into intervals, e.g. $d_x^i \leq interval_x^i < d_x^{i+1}$, and then map each *attribute-interval* pair to a binary attribute, i.e. $\langle x, interval_x^1 \rangle$, $\langle x, interval_x^2 \rangle$, ..., $\langle x, interval_x^n \rangle$. Table 2.6 and table 2.7 illustrate how attribute partitioning works.

Graduate	*Salary*
Yes	12500
No	10000
Yes	12000
Yes	11000
No	10000

Table 2.6: A Sample Database For Large Attribute Domain.

We can see that the domain of *Salary* in table 2.6 has been partition into three intervals, i.e. $\langle Salary, 10k\text{-}11k \rangle$, $\langle Salary, 11k\text{-}12k \rangle$, $\langle Salary, 12k\text{-}13k \rangle$. This time, the number of columns in the new database can be calculated by $\sum_{i=1}^{k}$ (*Number of intervals for* i^{th} *attribute*). Therefore, the problem of infinite number of attribute has been solved since we can limit the number of intervals for each attribute.

Graduate	$\langle Salary, 10k\text{-}11k \rangle$	$\langle Salary, 11k\text{-}12k \rangle$	$\langle Salary, 12k\text{-}13k \rangle$
1	0	0	1
0	1	0	0
1	0	0	1
1	0	1	0
0	1	0	0

Table 2.7: Resulting Database After Interval Replacement.

The algorithm will first use the *partial completeness level*, which gives a handle on the amount of information lost by partitioning, to calculate the number of partitions for each quantitative attribute.

After partitioning the attribute domain, we have to find the support for each partition of the quantitative attribute. For each quantitative attribute, adjacent partitions are combined as long as their support is less than the user specified *max support* which prevents the resulting intervals becoming too large. These form the set of all large 1-itemsets, then we can find all large itemsets using the boolean algorithms.

We can use the large itemsets to generate all the possible rules which is the same as that for binary association rules. However, this algorithm may produce redundant rules since there are some intervals with similar information will be generated in merging adjacent intervals. Therefore, we use a *greater than expected value* interest measure to prune the uninteresting rules in those possible rules.

2.3.2 Optimized Association Rules

In [11], algorithm have also been proposed to find association rules for numeric attributes. The rules are called *optimized association rules* and are of the form $(A \in [v_1, v_2]) \Rightarrow C$, where $[v_1, v_2]$ is the range that attribute A falls in and C is a condition containing boolean attribute. There are two kinds of optimized association rules. One is the *optimized support rule* which maximizes the support of $A \in [v_1, v_2]$ and the other is *optimized confidence rule* which maximizes the confidence of the rule.

For attribute A, the algorithm first partition its domain into a sequence of disjoint intervals called *buckets*, i.e. B_1, B_2, ..., B_M and $B_i = [x_i, y_i]$ such that $x_i \leq y_i < x_{i+1}$. If $x_i = y_i$, we call B_i a finest bucket. The range of an optimized association rule is the combination of consecutive buckets, i.e. B_p, B_{p+1},..., B_q. The number of tuples that values of A fall into B_i is denoted by u_i and the number of tuples that values of A fall into B_i and the corresponding tuples meet C is denoted by v_i. Therefore, the support of $A \in [x_p, y_q]$, *support*(p, q), is $(\sum_{i=p}^{q} u_i)/N$

and the confidence of $(A \in [x_p, y_q]) \Rightarrow C$, *conf*$(p, q)$, is $(\sum_{i=p}^{q} v_i)/(\sum_{i=s}^{q} u_i)$.

For computing u_i and v_i, the algorithm has to locate the values into buckets record by record. One way is to sort the database over the attribute A and divide the sorted data into finest buckets. However, the time needed to sort a huge database is long. Therefore, [11] uses an approximation algorithm to make *equi-depth buckets*. The algorithm first takes an S-sized random sample from database, then sorts the sample. Therefore, the time for sorting is reduced a lot. After that, the algorithm scans the sorted sample and locate the values into buckets. Finally, it scans the original database to assign values into buckets. Moreover, the process of assigning values into buckets can be parallel. Thus, the time needed for constructing buckets is further reduced

After talking about the construction of buckets, we talk about how the algorithm generates optimized confidence rules. It first creates a sequence of points $Q_k = (\sum_{i=1}^{k} u_i, \sum_{i=1}^{k} v_i)$ for k = 1,..., M and let Q_0 = (0,0). We can see that the slope of the line Q_mQ_n is exactly *conf*$(m+1, n)$ and the difference of x-coordinate of Q_m and Q_n is *support*$(m+1, n)$. The algorithm uses the convex hull concept to find the line Q_mQ_n, where $m + 1 \leq n$ with largest slope such that the confidence of rule is maximized.

The generation of optimized support rules is simpler. We only have to ensure that the confidence of the rules are not less than the threshold and try to maximize the support. Given a value of $p \leq q$ and *conf*$(p, q) \geq$ *threshold*, the algorithm tries to maximize $\sum_{i=p}^{q} u_i$.

In optimized association rule, there is only a single numeric attribute. In [11], the algorithm does not mention how to handle rules with multiple numeric attributes. It only mentions the extension of the optimized association rules with multiple boolean attributes.

Chapter 3

Problem Definition

Mining fuzzy association rules is the discovery of association rules using fuzzy set concepts such that the sharp boundary and problem with quantitative attributes can be solved. In this chapter, we will define the various terms used in mining fuzzy association rules. We will first describe the method used in solving the problem with quantitative attributes and then the definition and semantics of fuzzy association rule will be given. After that, we will discuss the interest measures in determining the large itemsets and interesting rules.

3.1 Handling Quantitative Attributes

In this section, we will first talk about the advantages and disadvantages of different methods in handling quantitative attributes. As we mentioned above, binary association rule algorithms cannot be directly applied to the quantitative attributes. Therefore, [29] first introduces the partitioning method which divides the attribute domain into intervals. However, there are some limitations in both discrete and overlapped intervals. We will describe the problems caused by attribute partitioning and how the fuzzy set concept can solve those problems.

3.1.1 Discrete intervals

Figure 3.1 shows one of the methods to partition the attribute domain. This method divides the attribute domain into equal discrete intervals such that each attribute value belongs to exactly one interval. Moreover, an element will contribute a vote of value one to the interval where it is located. After summing these votes, we can use this value to estimate the importance of an interval. However, there are limitations when we use this partition method. Since we partition the attribute domain into non-overlapped intervals, the sharp boundaries between intervals will prevent us from estimating the importance of an interval correctly. We will not consider the elements located near the outer boundaries of an interval when we calculate the support of that interval. Hence, we may exclude some possible interesting intervals, which have supports close to the user specified threshold, by ignoring some elements near the sharp boundaries.

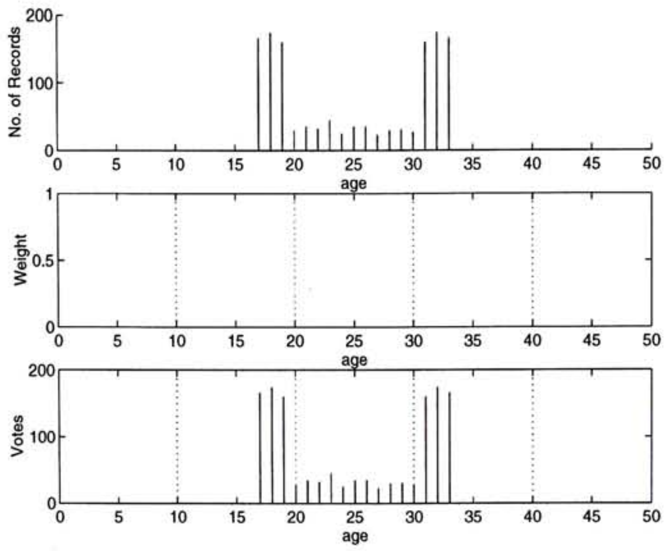


Figure 3.1: Partitioning Attribute Values Into Discrete Intervals.

The effect of sharp boundary is shown in figure 3.1. The first graph in figure 3.1 is the data distribution of *age*. It shows how many records contain specific attribute value. The middle one shows how the attribute domain of *age* has been partitioned. The vertical axis represents that each attribute value will contribute a vote of one to the interval that it belongs when a record contains that value. We use the last graph to illustrate the problem of discrete intervals. We assume that 50% of records are evenly distributed in three intervals, i.e. 10 to 20, 20 to 30 and 30 to 40, and the user specified support is 20%. Then, none of the three intervals will be considered as interesting. However, for the interval where *age* starts from 20 to 30, it is obvious that it should be interesting if we consider the values near the boundaries of both sides. We cannot avoid this kind of problems if we use discrete partitioning method to deal with quantitative attributes.

3.1.2 Overlapped intervals

Another attribute partitioning [29] is to divide the attribute domain into overlapped regions. Figure 3.2 shows the result of dividing the attribute domain into overlapped intervals. The data distribution in the first graph is the same as the one in figure 3.1. In the second graph, we can see that the size of the interval is larger than that of discrete intervals with overlapped boundaries.

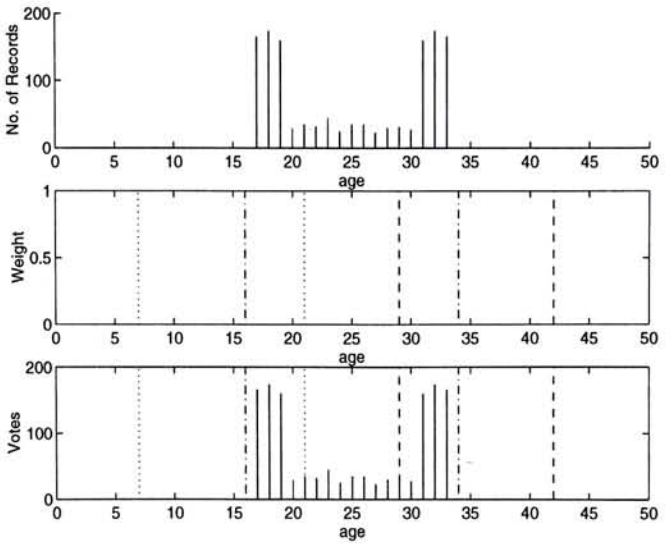


Figure 3.2: Partitioning Attribute Values Into Overlapped Intervals.

As a result, the elements may locate in more than one interval and contribute their votes to all these intervals. Comparing with discrete intervals, we may have more interesting intervals in the overlapped case. However, it is not reasonable for an element near the boundary to contribute the same as those located in the middle of interval. This may overemphasize the importance of an interval. When interpreting a rule, intervals will correspond to some higher level concepts, e.g. **age** : *young, old.* In such concepts, a clear boundary is not intuitive, e.g. we cannot say age of 35 is strictly young or old. We may consider the lesser the value is, the stronger is the value considered young. Therefore, we need other methods to solve both of the above problems.

3.1.3 Fuzzy sets

The above attribute partitioning methods are subject to the effect of sharp boundary, i.e. an element is either in an interval but not both, or an element contributes to more than one interval with equal weight. The problem is caused by the characteristic of classical set theory. In classical set theory, the value of membership of an element can only be either 0 or 1. It means that an element is located either in a set or outside a set but not both. We can, however, have set inclusion property that differs from the classical one when we use fuzzy sets. For each fuzzy set, a membership grade/value is associated with the attribute value. In fuzzy set concept, an element can belong to more than one set associated with its membership value. The membership value, which falls between [0,1], represents the degree of set inclusion. Larger values denote higher degrees of set membership. This value is assigned by the membership function associated with each fuzzy set. Basically, we can view fuzzy set as a mapping from the attribute domain to [0,1]. For example, let x be the attribute and D_x be the domain of x. For a given value in x, we can use the membership function m_{f_x} to map the value to a membership grade. The mapping is as follows:

$$m_{f_x}(x) : D_x \rightarrow [0, 1]$$

Instead of the abrupt change, a fuzzy set provides a smooth transition between the boundaries and the effect is shown in figure 3.3. The second graph of figure 3.3 shows a traditional fuzzy set of which curve of the membership function is a bell shape curve. However, in practice, we will only use several points to represent a fuzzy set. In the third graph, we can see that the values located outside the interval, 20 to 30, have been considered. Therefore, the sharp boundary problem has been tackled. Moreover, the contribution of a value has been restricted by the membership function of the fuzzy set as illustrated in figure 3.3. Hence, the boundary elements will not be overemphasized when we use fuzzy set to deal with quantitative attributes.

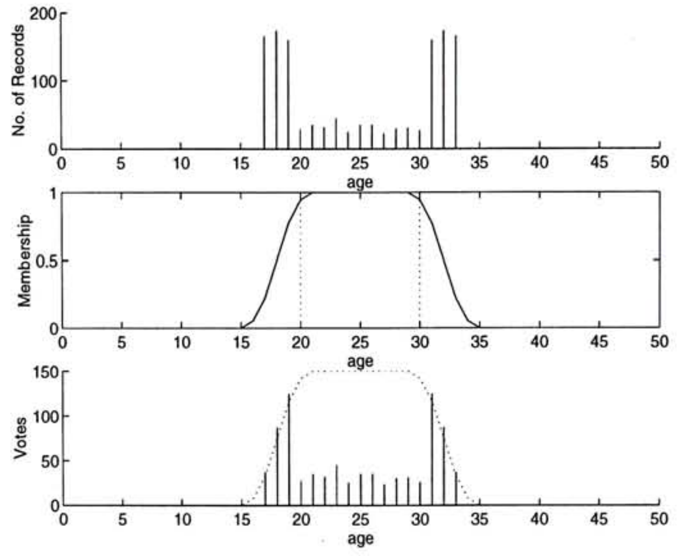


Figure 3.3: Fuzzy Set Soften The Effect Of Sharp Boundary.

Using fuzzy set, we have two ways to find fuzzy association rules.

1. The first one is similar to the one in [29]. We can transform the fuzzy association rule mining problem into a binary association rule mining problem. We replace the original attribute by their *attribute-fuzzy set* pairs which are binary attributes. The values are computed according to the membership functions and the user specified membership threshold.

2. The second method is to store the membership grades in the newly gener-

ated table and use the new database to find the fuzzy association rules.

Given the attribute *Age* has the fuzzy set *mid-age* associated with it, the transformation of database is illustrate in table 3.1.

Age
19
10
28
31
34

$\langle Age, mid\text{-}age \rangle$
0.9
0
1
0.9
0.1

$\langle Age, mid\text{-}age \rangle$
1
0
1
1
0

Table 3.1: Database Transformation.

The original values of the attribute *Age* are shown in the first table of table 3.1. The second table is the membership grades calculated by the membership function of the fuzzy set *mid-age*. The third table is generated from the values in the second table by setting the user specified membership threshold to 0.2. In the first method, if the membership value is not less than the user specified threshold, we put 1 in the corresponding position. Otherwise, we will put 0 in the database. After obtaining the binary database, we can apply existing binary association rule mining algorithms with little modification. Using the first method, it reflects the number of records supporting the attribute-fuzzy set pair. In the second method, we will directly use the membership grades as the database. Therefore, we have to use new interest measures for the determination of interesting itemsets and rules. Nevertheless, this method reflects the number of records as well as the degree of support which satisfy the attribute-fuzzy set pair. In this thesis, we will use the second method to transform the original database to database with membership grades.

3.2 Fuzzy association rule

Let $T = \{t_1, t_2, ..., t_n\}$ be the database containing n records and t_i represents the corresponding i^{th} tuple in the database. In this context, we will use the terms record and tuple interchangeably. Moreover, we use $I = \{i_1, i_2, ..., i_m\}$ to represent all attributes that appear in the database T and i_j to represent the j^{th} attribute of T. Here, the terms attribute and item will be used interchangeably, too. Since I contains a set of items, we call I an *itemset* which is the same as previous papers [3] in data mining. Table 3.2 shows a sample database with three attributes which contains five records.

Retired	*Children*	*Salary*
Yes	2	0
No	3	15000
No	0	10000
No	1	20000
Yes	2	0

Table 3.2: A Sample Database With Three Attributes.

In table 3.2, $T = \{t_1, t_2, t_3, t_4, t_5\}$ and $I = \{$*Retired*, *Children*, *Salary*$\}$. We can retrieve the value of attribute i_k in the j^{th} record of the database T simply by $t_j[i_k]$. For example, if we want to know the value of *Salary* of the fourth record, we can use $t_4[Salary]$ and get the value 20000.

Besides, each attribute i_k will associate with one or more fuzzy sets. We use $F_{i_k} = \{f^1_{i_k}, f^2_{i_k}, ..., f^l_{i_k}\}$, where $l \geq 1$, to represent the set of fuzzy sets associated with i_k and $f^j_{i_k}$ represent the j^{th} fuzzy set of i_k. For example, if the attribute *Salary* in T has three fuzzy sets: *high*, *medium* and *low*, we will have F_{Salary} $= \{$*high*, *medium*, *low*$\}$. Here, we do not assume any order to exist in those fuzzy sets. Furthermore, we assume that the fuzzy sets and the corresponding membership functions are provided by domain experts.

Given a database T with attributes I and those fuzzy sets associated with attributes in I, we want to find out some interesting and potentially useful regularities in a guided way. Our proposed fuzzy association rule has the following form:

$$\text{If } X \text{ is } A \text{ then } Y \text{ is } B.$$

In the above rule, $X = \{x_1, x_2, ..., x_p\}$ and $Y = \{y_1, y_2, ..., y_q\}$ are itemsets. X and Y are subsets of I and they are disjoint. That means that they share no common attributes. $A = \{f_{x_1}, f_{x_2}, ..., f_{x_p}\}$ and $B = \{f_{y_1}, f_{y_2}, ..., f_{y_q}\}$ contain the fuzzy sets associated with the corresponding attributes in X and Y. For example, an attribute x_k in X will have a fuzzy set f_{x_k} in A such that $f_{x_k} \in F_{x_k}$ is satisfied.

The first part of the rule 'X is A' is called the antecedent and 'Y is B' is called the consequent of the rule. The semantics of the rule is when 'X is A' is satisfied, we can imply that 'Y is B' is also satisfied. Here, *satisfied* means there are sufficient amount of records which contribute their votes to the *attribute-fuzzy set* pairs and the sum of these votes is greater than a user specified threshold.

For a rule to be interesting, it should have enough *significance* as well as sufficient *certainty.* We use a *significance* factor to test the satisfiability of an itemset and use a certainty factor to determine the strength of a rule. In the following sections, we will describe both significance and certainty factors in detail.

3.3 Significance factor

In generating fuzzy association rule, we have first to find out all *large k-itemsets* which are itemsets with *significance* not less than the value specified by the user.

Given a *large itemset* L, we will take all possible subsets, X, of the *large itemset* L as antecedent and take the remaining $Y = L - X$ as the consequent. Moreover, we also need to measure the $significance$ of the antecedent and consequent. Therefore, calculating the significance of itemsets is very important in finding fuzzy association rules. The $significance$ factor is calculated by first summing all votes of each records with respect to the specified itemset, then dividing it by the total number of records in database T. Each record contributes a vote which falls between 0 and 1 to a series of itemsets. Therefore, *significance* factor reflects not only the number of records supporting the itemset, but also shows their degree of support. Before we talk about the calculation of the significance factor, we first introduce some symbols and functions in table 3.3.

T	Database with quantitative attributes
x_j	Attribute in database
a_j	Fuzzy set corresponds to x_j
t_i	i^{th} record in T
$t_i[x_j]$	Value of x_j in i^{th} record
$m_{a_j}(t_i[x_j])$	Membership function of a fuzzy set a_j
opr_x	Any function that is used in fuzzy set, e.g. max, min, multiplication($\prod$)
$total(T)$	Number of records in T

Table 3.3: Table of symbol.

Let $X = \{x_1, x_2, ..., x_l\}$ and $A = \{a_1, a_2, ..., a_l\}$ be ordered sets and a_i corresponds to x_i. We use the following formula to calculate the significance factor of an itemset X with respect to fuzzy sets A associated with the attributes x_j in X, i.e. $S_{\langle X,A\rangle}$.

$$\begin{aligned} Significance &= \frac{\text{Sum of votes satisfying } \langle X, A\rangle}{\text{Number of records in database } T} \\ S_{\langle X,A\rangle} &= \frac{\sum_{t_i \in T} opr_{x_j \in X}\{\alpha_{a_j}(t_i[x_j])\}}{total(T)} \\ &= \frac{\sum_{t_i \in T} \prod_{x_j \in X}\{\alpha_{a_j}(t_i[x_j])\}}{total(T)} \end{aligned}$$

$$\text{where} \quad \alpha_{a_j}(t_i[x_j]) = \begin{cases} m_{a_j}(t_i[x_j]) & \text{if } m_{a_j} \geq \text{threshold}, \\ 0 & \text{otherwise.} \end{cases}$$

In the above equation, $\langle X, A\rangle$ represents the *itemset-fuzzy set* pair, where X is the itemset containing attributes x_j and A is the set of fuzzy set containing fuzzy sets a_j associated with x_j. A record satisfies $\langle X, A\rangle$ means that the vote of the record is greater than the one specified by user. The vote of a record is calculated by the membership grade of each x_j in that record and is described as follows. We use $t_i[x_j]$ to obtain the value of x_j in the i^{th} records, then transform this value into membership grade by $m_{a_j}(t_i[x_j])$ which is the membership function of fuzzy set a_j. After obtaining all membership grades of each x_j in a record, we use $opr_{x_j \in X}\{\alpha_{a_j}(t_i[x_j])\}$ to calculate the vote of record t_i, where *opr* can be any function that is used in fuzzy set. For example, we can use *minimum*, *maximum* or *multiplication*. In the above equation, we use multiplication($\prod$) as *opr*, i.e. $opr_{x_j \in X}\{\alpha_{a_j}(t_i[x_j])\}$ is replaced by $\prod_{x_j \in X}\{\alpha_{a_j}(t_i[x_j])\}$. After summing up all these votes from each record t_i, we can divide this value by $total(T)$ to get the desired *significance* factor of $\langle X, A\rangle$.

In this thesis, we use multiplication to calculate the votes of each record because it magnifies the effect of low membership grade in the calculation. Moreover, the multiplication operator gives a simple and meaningful relationship among attributes in an itemset. Table 3.4 illustrates why we use multiplication operator instead of others.

			Max	Min	Mul
0.9	0.2	0	0.9	0	0
0.9	0.9	0.2	0.9	0.2	0.162
0.3	0.3	0.2	0.3	0.2	0.018

Table 3.4: The Effect Of Different Operators.

In table 3.4, we can see that both *Max* and *Min* operators ignore the membership values of other attributes except the one with maximum or minimum values. In the first and the second records, *Max* finds the same significance values even though there is a zero in the first record. In the second and the third record, *Min* also finds the same significance although the membership values of the two records are extremely different. On the contrary, the *Mul* operator does not take the value of a single attribute as the vote of a record. Instead, it uses the values of all attributes of an itemset to generate the vote of a record. Therefore, *Mul* gives the correct vote of a record and this resulting vote reflects the membership grades of all attributes of an itemset.

$\langle Salary, high\rangle$	$\langle Balance, low\rangle$
0.9	0.2
0.2	0.7
0.5	0.4
0.3	0.7
0.6	0.3

Table 3.5: Database Containing Membership Values.

Given a partial database in table 3.5, examples are provided to illustrate the computation of *significance* factor. We want to find the $significance$ factor of an *itemset-fuzzy set* pair $\langle X, A\rangle$, where $X = \{Salary, Balance\}$ and $A = \{high, low\}$. In table 3.5, the attribute values have been already transformed into membership grades. Therefore, the significance of $\langle X, A\rangle$ is as follows.

$$\begin{aligned} S_{\langle X,A\rangle} &= \frac{(0.9 \times 0.2) + (0.2 \times 0.7) + (0.5 \times 0.4) + (0.3 \times 0.7) + (0.6 \times 0.3)}{5} \\ &= 0.182 \\ \text{where} \quad & X = \{Salary, Balance\}, A = \{high, low\} \end{aligned}$$

In the above calculation, we have assumed the user membership threshold value is zero. However, we may not want to consider the low membership grade. The following example shows how we can prevent these membership grades from contributing to a record. We set the threshold of *membership* grade to 0.4 and the *significance* will be as follows.

$$\begin{aligned} S_{\langle X,A\rangle} &= \frac{(0.9\times 0)+(0\times 0.7)+(0.5\times 0.4)+(0\times 0.7)+(0.6\times 0)}{5} \\ &= 0.04 \\ \text{where} \quad & X=\{Salary, Balance\}, A=\{high, low\} \end{aligned}$$

We can see that the membership grades which are less than the user specified threshold are eliminated from the calculation of significance factor such that the resulting value is different. Using the above equation, we can obtain all itemsets with sufficiently high $significance$. These itemsets are called *large itemsets* and we will use these *large itemsets* to generate the fuzzy association rules. In the following section, we will discuss the way we determine a rule as interesting.

3.4 Certainty factor

As we mentioned above, not all itemsets will be considered as interesting. We have to use *significance* factor as a measure to screen out those itemsets that are not important to us. For fuzzy association rules, the situation is similar. We use the discovered *large itemsets* to generate the rules and we have to use some measure to drop those uninteresting rules. The criteria for considering importance of a rule is called *certainty* factor. If a rule has enough *significance* and sufficient *certainty*, this rule will be considered as interesting. In this section, we will describe two ways to calculate the *certainty* factor of a potential fuzzy association rule.

3.4.1 Using significance

When we obtain a *large itemset* $\langle Z, C\rangle$, where $Z = \{z_1, z_2, ..., z_l\}$ and $C = \{c_1, c_2, ..., c_l\}$ are ordered sets and c_j corresponds to z_j, we want to generate fuzzy association rules of the form, 'If X is A then Y is B.', where X is a subset of Z and $Y = Z - X$, A is a subset of C and $B = C - A$. If $X = \{x_1, x_2, ..., x_k\}$, then $A = \{a_1, a_2, ..., a_k\}$. Having the *large itemset*, we know the *significance* factor of this large itemset as well as its subsets according to the property of subsets of large itemsets. Therefore, we can use the following formula to calculate the *certainty* factor of fuzzy association rules.

$$\begin{aligned}
Certainty &= \frac{\text{Significance of } \langle Z, C\rangle}{\text{Significance of } \langle X, A\rangle} \\
C_{\langle\langle X,A\rangle,\langle Y,B\rangle\rangle} &= \frac{\sum_{t_i \in T} opr_{z_k \in Z}\{\alpha_{c_k}(t_i[z_k])\}}{\sum_{t_i \in T} opr_{x_j \in X}\{\alpha_{a_j}(t_i[x_j])\}} \\
&= \frac{\sum_{t_i \in T} \prod_{z_k \in Z}\{\alpha_{c_k}(t_i[z_k])\}}{\sum_{t_i \in T} \prod_{x_j \in X}\{\alpha_{a_j}(t_i[x_j])\}} \\
\text{where} \quad & Z = X \cup Y, C = A \cup B \\
& \alpha_{c_k}(t_i[z_k]) = \begin{cases} m_{c_k}(t_i[z_k]) & \text{if } m_{c_k} \geq \text{threshold}, \\ 0 & \text{otherwise.} \end{cases}
\end{aligned}$$

Since the *significance* factor of an itemset is the measure of degree of support given by records, we use *significance* to help us estimate the interestingness of the generated fuzzy association rules. In the above equation, we divide the *significance* of $\langle Z, C\rangle$ by *significance* of $\langle X, A\rangle$. The *certainty* factor reflects the fraction of votes supporting $\langle X, A\rangle$ that will also support $\langle Z, C\rangle$. Again, we replace the *opr* operator by the *multiplication* operator such that the result can reflect the contribution of each attribute in the records. We will use the information in table 3.5 to illustrate the calculation of *certainty* factor. We let $Z = \{$*Salary*, *Balance*$\}$ and $C = \{$*high*,*low*$\}$. Given this *large itemset*, we want to calculate the *certainty* of the rule, 'If *Salary* is *high* then *Balance* is *low*.'. We assume that the values of user specified membership and significance threshold are zero.

$$\begin{aligned} C_{\langle\langle X,A\rangle,\langle Y,B\rangle\rangle} &= \frac{(0.9\times 0.2)+(0.2\times 0.7)+(0.5\times 0.4)+(0.3\times 0.7)+(0.6\times 0.3)}{0.9+0.2+0.5+0.3+0.6} \\ &= 0.364 \\ \text{where} \quad & X = Salary, A = high, \\ & Y = Balance, B = low. \end{aligned}$$

We may wonder why the rule has *certainty* factor more than 30% even though there are so many low membership grades. It is because we have consider these low membership values in the computation of the *certainty* factor. We use another example to illustrate the computation of certainty factor. Suppose the user specified membership threshold is 0.4, the calculation of the *certainty* factor is as follows.

$$\begin{aligned} C_{\langle\langle X,A\rangle,\langle Y,B\rangle\rangle} &= \frac{(0.9\times 0)+(0\times 0.7)+(0.5\times 0.4)+(0\times 0.7)+(0.6\times 0)}{0.9+0+0.5+0+0.6} \\ &= 0.1 \\ \text{where} \quad & X = Salary, A = high, \\ & Y = Balance, B = low. \end{aligned}$$

After eliminating the low membership values, the certainty value is 10% which seems more reasonable with respect to the given database. Therefore, users can change the membership threshold to obtain more rules but less precision, or fewer rules but higher precision.

3.4.2 Using correlation

Another way to calculate the *certainty* factor of a rule is to compute a kind of correlation between $\langle X, A\rangle$ and $\langle Y, B\rangle$. However, the calculation of the correlation is somewhat different from the one commonly used in statistics. The treatment of $\langle X, A\rangle$ and $\langle Y, B\rangle$ are not symmetrical and we call the correlation of $\langle X, A\rangle$

over $\langle Y, B\rangle$, *XYCorrelation.* When we calculate the XYCorrelation, we have to take $\langle X, A\rangle$ into account. For both *covariance* and *variance*, we have to compute the expectation in order to get the *certainty.* The calculation of expectation of the antecedent is similar to statistics except that we have taken the user specified membership grade into account. The vote of record will be considered as zero if the membership grade of $\langle X, A\rangle$ in that record is less than a user specified threshold. However, in calculating the expectation of the consequent, we will also consider the vote of consequent only if the vote of the antecedent is not less than the user specified threshold. We use the following equation to calculate the correlation of the antecedent $\langle X, A\rangle$ and the consequent $\langle Y, B\rangle$ as the *certainty* factor of the rule, i.e. $C_{\langle\langle X,A\rangle,\langle Y,B\rangle\rangle}$.

$$\begin{aligned}
Certainty &= \text{XYCorrelation of } \langle X, A\rangle \text{ and } \langle Y, B\rangle \\
C_{\langle\langle X,A\rangle,\langle Y,B\rangle\rangle} &= \frac{Covariance(\langle X, A\rangle, \langle Y, B\rangle)}{\sqrt{Variance(\langle X, A\rangle) \times Variance(\langle Y, B\rangle)}} \\
&= \frac{E[\langle X \cup Y, A \cup B\rangle] - E[\langle X, A\rangle] \times E'[\langle Y, B\rangle]}{\sqrt{(E[\langle X, A\rangle^2] - E[\langle X, A\rangle]^2) \times (E'[\langle Y, B\rangle^2] - E'[\langle Y, B\rangle]^2)}} \\
\text{where} & \\
E[\langle X, A\rangle] &= \frac{\sum_{t_i \in T} \prod_{x_j \in X} \{\alpha_{a_j}(t_i[x_j])\}}{total(T)} \\
\alpha_{a_j}(t_i[x_j]) &= \begin{cases} m_{a_j}(t_i[x_j]) & \text{if } m_{a_j} \geq \text{threshold}, \\ 0 & \text{otherwise.} \end{cases} \\
E'[\langle Y, B\rangle] &= \frac{\sum_{t_i \in T} \beta[t_i]}{total(T)} \\
\beta[t_i] &= \begin{cases} \prod_{y_k \in Y} \{\alpha_{b_k}(t_i[y_k])\} & \text{if } \prod_{x_j \in X} \{\alpha_{a_j}(t_i[x_j])\} \geq \text{threshold}, \\ 0 & \text{otherwise.} \end{cases}
\end{aligned}$$

We modify the calculation of expectation because of the meaning of correlation in statistics. In statistics, we say X and Y are correlated means that X implies Y as well as Y implies X. However, fuzzy association rules do not necessarily have this property. In data mining, a rule $X \rightarrow Y$ usually means X implies

Y and we do not assume Y also implies X because of the data distribution of X and Y. Therefore, we change the calculation of expectation such that we can accommodate the meaning of fuzzy association rules. In the above equations, we can see that the calculation of $E[\langle X, A\rangle]$ is similar to an ordinary expectation except it has taken the membership threshold into account. Moreover, we use $E'[\langle Y, B\rangle]$ to calculate the expectation of the consequent of a rule. If the product of membership grades of the antecedent of a record is less than the threshold, we will not consider the membership grades of the consequent.

The value of the *certainty* is ranging from -1 to 1. We are only interested in the positive XYCorrelation values since we have modified the semantics of correlation in statistics. A zero means that there is no relationship between the antecedent and consequent. A positive value tells that the antecedent and consequent are positively related. A high XYCorrelation value means that we expect a high membership grade will appear in the consequent if we have a high membership in the antecedent. Therefore, if the rule 'If X is A then Y is B.' holds, the *certainty* of this rule should be at least greater than zero.

Given the database in table 3.5, we can use the above formula to compute the *certainty* factor of the rule, 'If *Salary* is *high* then *Balance* is *low.*'. The following example illustrates the computation of *certainty* factor of the rule.

$$\begin{aligned}
E[\langle Z, C\rangle] &= (0.18 + 0.14 + 0.2 + 0.21 + 0.18)/5 \\
E[\langle X, A\rangle^2] &= (0.9^2 + 0.2^2 + 0.5^2 + 0.3^2 + 0.6^2)/5 \\
E'[\langle Y, B\rangle^2] &= (0.2^2 + 0.7^2 + 0.4^2 + 0.7^2 + 0.3^2)/5 \\
E[\langle X, A\rangle] &= (0.9 + 0.2 + 0.5 + 0.3 + 0.6)/5 \\
E'[\langle Y, B\rangle] &= (0.2 + 0.7 + 0.4 + 0.7 + 0.3)/5 \\
C_{\langle\langle X,A\rangle,\langle Y,B\rangle\rangle} &= \frac{0.182 - 0.5 \times 0.46}{\sqrt{(0.31 - 0.25) \times (0.254 - 0.2116)}} \\
&= \frac{-0.048}{0.05}
\end{aligned}$$

$$
\begin{aligned}
&= -0.96 \\
\text{where} \quad & Z = X \cup Y, C = A \cup B \\
& X = Salary, A = high, \\
& Y = Balance, B = low.
\end{aligned}
$$

The result tells us that the rule, 'If *Salary* is *high* then *Balance* is *low.*', does not hold since the certainty value are negative. Referring to table 3.5, we can see that $\langle Salary, high \rangle$ has high membership grade when there is a low membership grade in $\langle Balance, low \rangle$.

In the above example, the membership threshold has been set to zero. In the following example, we let the membership threshold be 0.4 such that we can ignore the low membership values.

$$
\begin{aligned}
E[\langle Z, C \rangle] &= (0 + 0 + 0.2 + 0 + 0)/5 \\
E[\langle X, A \rangle^2] &= (0.9^2 + 0 + 0.5^2 + 0 + 0.6^2)/5 \\
E'[\langle Y, B \rangle^2] &= (0 + 0 + 0.4^2 + 0 + 0)/5 \\
E[\langle X, A \rangle] &= (0.9 + 0 + 0.5 + 0 + 0.6)/5 \\
E'[\langle Y, B \rangle] &= (0 + 0 + 0.4 + 0 + 0)/5 \\
C_{\langle \langle X,A \rangle, \langle Y,B \rangle \rangle} &= \frac{0.04 - 0.4 \times 0.08}{\sqrt{(0.284 - 0.4) \times (0.032 - 0.08)}} \\
&= \frac{0.008}{0.075} \\
&= 0.107 \\
\text{where} \quad & Z = X \cup Y, C = A \cup B \\
& X = Salary, A = high, \\
& Y = Balance, B = low.
\end{aligned}
$$

We can see that the certainty is about 10% which is similar to the result of the first method which uses significance to calculate the certainty factor. It is because the low membership values have been eliminated from the computation of the certainty factor. Therefore, we have to calibrate the membership threshold such that membership values which are too low should be dropped when we calculate the certainty factor.

3.4.3 Significance vs. Correlation

In previous paragraphs, We have described two methods to compute the *certainty* factor. The first one uses *significance* factor of the antecedent and the union of antecedent and consequent to calculate the certainty which reflects the fraction of votes which contributes to the antecedent will also contribute to the union of antecedent and consequent. The second method computes the *XYCorrelation* of the antecedent over the consequent as the certainty which reflects the membership relation of the antecedent and consequent, for example, high certainty means that high membership in antecedent will imply high membership in consequent. There is a tradeoff between accuracy and speed. Since the first method only uses the significance of itemsets which are already known when we generate the large itemsets. We need not search the database again. Therefore, the speed of the first method is comparably faster. On the contrary, the second method has to compute the XYCorrelation of the antecedent over the consequent. For computing XYCorrelation, we have to know the relationship of antecedent and consequent record by record. The significance factor does not convey this kind of information. Therefore, we have to search the database again in order to compute the XYCorrelation. However, the accuracy of the second method should be higher than the first method since the correlation reflects the relationship of antecedent and consequent in statistical way. Therefore, we can choose either method under different situations.

Chapter 4

Steps For Mining Fuzzy Association Rules

In this chapter, we will talk about the basic steps for fuzzy association rule mining and the implementation of our algorithms. As mentioned above, the basic framework for association rule mining includes finding all large k-itemsets and generating all interesting rules from these large itemsets. For fuzzy association rule mining, we first transform the original database into a database which contains only membership values. This step consists of replacing original attributes to their *attribute-fuzzy set* pairs and converting the data value into membership grades. After this transformation, we can use a candidate generation function, which is similar to *Apriori-gen*, to find all candidate k-itemsets. After that, we have to calculate the *significance* factor of the discovered candidate itemsets and keep those candidate itemsets with enough significance as large itemsets. The rest of mining process is to generate all possible rules from all discovered large itemsets. We will consider a rule as interesting if it has sufficient *certainty* factor. The steps for fuzzy association rule mining can be summarized as follows.

1. Transforming the original database

2. Generating candidate k-itemsets from the large (k-1)-itemsets
3. Generating large k-itemsets from candidate k-itemsets
4. Generating interesting rules from all large k-itemsets

In the following sections, we will use pseudo-codes to illustrate the implementation of the generation of *candidate itemsets*, *large itemsets* and *fuzzy association rules.*

4.1 Candidate itemsets generation

Before we can generate fuzzy association rules, we have to first find out the potential itemsets which will be used in generating the interesting rules. We call these potential itemsets, *candidate itemsets.* We will generate candidate itemset according to the *Apriori-gen* function in mining binary association rules. We have modified the candidate generation function such that it is suitable for fuzzy association rule mining. The function generates *candidate k-itemsets* C_k using the *large (k-1)-itemsets* L_{k-1}. However, in mining binary association rules, the candidate 1-itemsets will be all the attributes appeared in the database. In mining fuzzy association rule, we can use the property of fuzzy set concept to reduce the number of candidate 1-itemsets such that the number of itemsets generated in later stages is reduced. We will first talk about the fuzzy set order before we continue the description of the generation of candidate itemsets.

In fuzzy sets, we can use their membership functions to identify their orders. Figure 4.1 shows two fuzzy sets, f_1 and f_2. Given an attribute *age* and its domain D_{age}, let f_1 and f_2 be the fuzzy sets of *age*. We said $f_1 > f_2$ if the following holds.

$$\forall d_i \in D_{age}, m_{f_1}(d_i) \geq m_{f_2}(d_i)$$

In figure 4.1, we can see that the curve of f_2 is contained in f_1. As we mentioned above, we can use this ordering property of fuzzy sets to reduce the number of candidate 1-itemsets generated in the mining process.

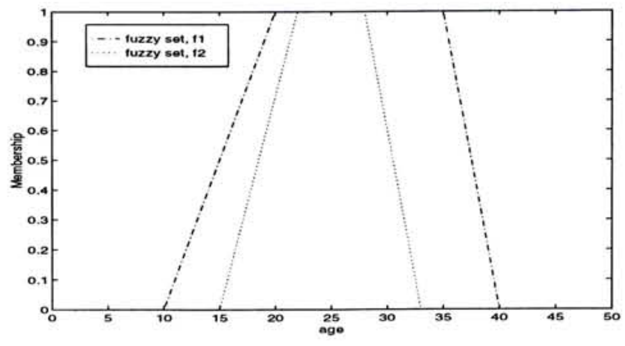


Figure 4.1: Fuzzy Set Order.

4.1.1 Candidate 1-Itemsets

As we mentioned above, we have first to sort the fuzzy sets of each attribute. For example, we try to sort the fuzzy sets of the attribute *age*. There are four fuzzy sets for the attribute *age* as illustrated in figure 4.2, i.e. $F_{age} = \{f_1, f_2, f_3, f_4\}$. Our aim is to divide the set of fuzzy sets F_{age} into subsets such that $F_{age} = \{F^1_{age}, F^2_{age}, ..., F^m_{age}\}$. Every fuzzy set in F^i_{age} will be in descending order. In this example, $F_{age} = \{F^1_{age}, F^2_{age}, F^3_{age}\}$, where $F^1_{age} = \{f_1\}$, $F^2_{age} = \{f_2, f_3\}$ and $F^3_{age} = \{f_4\}$. We have to ensure the relationship $f_2 > f_3$ in F^2_{age} such that our algorithm works.

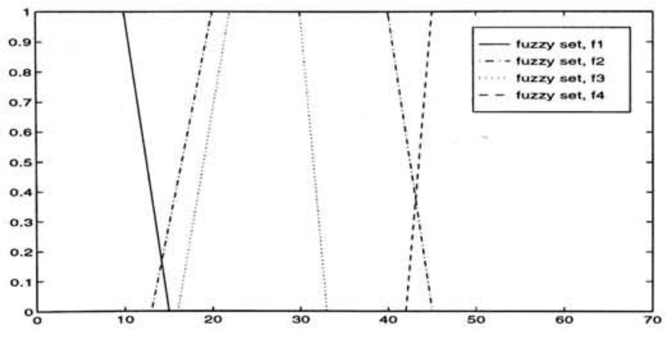


Figure 4.2: Fuzzy Sets For Attribute *Age*.

Given the above settings, we will talk about how we can reduce the number of candidate 1-itemsets generated. Since we have four fuzzy sets for attribute *age*, we replace *age* by four *attribute-fuzzy set* pairs. The intuitive method to generate candidate 1-itemsets is to consider these *attribute-fuzzy set* pairs as our candidate 1-itemsets and calculate their $significance$ factors. However, in our algorithm, we have first sorted the fuzzy sets of each attribute in descending order. The following pseudo-code illustrates how the algorithm works.

```
let $C_1$ = $\emptyset$
for each attribute $x_i$ in $X$
    for each subset $F^j_{x_i}$ of $F_{x_i}$
        for each fuzzy set $f_k$ in $F^j_{x_i}$
            if the attribute-fuzzy set pair $\langle x_i, f_k \rangle$ is not significant
               consider other subsets of $F_{x_i}$
            else
               add $\langle x_i, f_k \rangle$ into $C_1$
```

In the above pseudo-code, we can see that not every *attribute-fuzzy set* pair will be considered as candidate itemset because of the fuzzy set ordering. We will not consider the subset of fuzzy sets $F^j_{x_i}$ when the fuzzy set in $F^j_{x_i}$ is not significant such that the remaining fuzzy sets in $F^j_{x_j}$ will not be added into candidate itemsets. For example, if f_2 in F^2_{age} is not significant, we need not calculate the $significance$ of f_3 since f_2 is greater than f_3. It is because:

$$\begin{aligned}
\forall d_i \in D_{age}, m_{f_2}(d_i) &\geq m_{f_3}(d_i) \\
\sum_{t_i \in T} m_{f_2}(t_i[age]) &\geq \sum_{t_i \in T} m_{f_3}(t_i[age]) \\
S_{\langle age, f_2 \rangle} &\geq S_{\langle age, f_3 \rangle}
\end{aligned}$$

There are two effects in using this algorithm. First, the number of accesses to the database is reduced because there are fewer number of *attribute-fuzzy set* pairs considered as candidate 1-itemsets. Moreover, we will reduce the number of uninteresting candidate and large itemsets in later stages since we have reduced the number of candidate 1-itemsets in the very beginning of the mining process. The generation of candidate 1-itemsets is in fact the generation of large 1-itemsets since the resulting attribute-fuzzy set pair will have enough significance.

4.1.2 Candidate k-Itemsets (k $>$ 1)

After talking about the generation of candidate 1-itemsets, we will use the pseudo-code to describe the generation of candidate k-itemsets, where k is the number of items in an itemset and k is greater than one. Following is the pseudo-code for generating candidate k-itemsets.

```
for k = 2 to number of attributes
    let $C_k$ = $\emptyset$
    for each large itemset $l_i$ in $L_{k-1}$
        for each large itemset $l_j$ in $L_{k-1}$
            if the first k-2 items of $l_i$ and $l_j$ are the same
                    and the k-1$^{th}$ item of $l_i$ is less than $l_j$
               $c$ = $l_i \cup$ k-1$^{th}$ item of $l_j$
               if all (k-1) subsets of $c$ in $L_{k-1}$
                  add $c$ into $C_k$
```

We can see that the candidate k-itemset is generated from the large (k-1)-itemsets, i.e. $L_{k-1} \times L_{k-1}$. However, we will not consider all itemsets in $L_{k-1} \times L_{k-1}$ as candidate itemsets. Otherwise, we will have many duplicated candidate itemsets. For every two large (k-1)-itemsets l_i and l_j in L_{k-1}, if the first k-2 items of these large itemsets are the same and the last items of l_i is less than l_j, we will consider the union of l_i and l_j as the newly generated candidate k-itemset. In fact, this is how the candidate generation function *Apriori-gen* works. However, we have to modify this function such that the attribute-fuzzy set pairs from the same attribute will not join together. Since we have associated numbers with both attributes and fuzzy sets, we can identify the order of *attribute-fuzzy set* pair easily. After talking about the generation of candidate itemsets, we can proceed to the implementation for discovery of large itemsets.

4.2 Large itemsets generation

When we have generated the candidate k-itemsets, we can use these itemsets to generate the desired large k-itemsets. All itemsets in candidate itemsets have potential to be large itemsets. Therefore, we have to verify whether they are really large. Otherwise, we may generate many false large itemsets. We calculate the *significance* of the candidate itemsets and use it as the interest measure to eliminate candidate itemsets of which significance is lower than user specified significance threshold. We prevent the generation of uninteresting rules by eliminating those candidate itemsets with low *significance*. The large itemsets are generated as follows.

```
for k = 2 to number of attributes
    let $L_k$ = $\emptyset$
    for each candidate itemset $c$ in $C_k$
        calculate the significance of $c$
        if the significance is not less than threshold
            add $c$ into $L_k$
```

When we calculate the significance factor of candidate itemsets, we have also taken the user specified membership threshold into account such that we will treat the membership grade which is lower than the threshold as zero. Therefore, elements that are too far away from the sharp boundary will not contribute their votes to the itemsets.

4.3 Fuzzy association rules generation

After obtaining all the *large k-itemsets*, we can generate rules using these large itemsets. For each large itemset L, we use the subset X of the large itemset L as the antecedent and $Y = L - X$ as the consequent to generate all possible rules. The discovered rules are of the form, 'If X is A then Y is B.'. The following pseudo-code shows the generation of interesting rules.

```
for k = 2 to number of attributes
    let $R$ = $\emptyset$
    for each large itemset $l$ in $L_k$
        for each subset $X$ of $l$
            $Y = l - X$
            $r$ = $X \rightarrow Y$
            calculate the certainty of $r$
            if the certainty is not less than threshold
               add $r$ into $R$
```

When we generate the rule, we have to measure its strength such that we can decide whether we will include the rule in our result. The interest measure of rule is called the *certainty* factor. We can use either method mentioned in chapter 3 to calculate the certainty factor of each rule, i.e. using *significance* or *correlation*. The first method will lead to shorter execution time and the second one will give more accurate relationship between the antecedent and consequent of the rule.

Chapter 5

Experimental Results

After describing the definitions and steps of fuzzy association rule mining, we would like to verify the correctness of our algorithms. We will first compare our algorithms with the interval method to examine the accuracy and performance. Moreover, we will present several experiments which examine the accuracy and performance of our algorithms under different parameters, e.g. different significance threshold, membership threshold and certainty threshold. We want to see how our algorithms will be affected by these parameters. In the following sections, we will describe the experiment settings and explain the results of different methods.

5.1 Experiment One

In this experiment, we use two attributes to illustrate how the fuzzy set concept can solve the problem of sharp boundary. We assume that the interval method only uses discrete and fixed intervals and there are three intervals/fuzzy sets for each attribute. Therefore, using the interval method, there are 9 regions in the data distribution. The data distribution is shown in figure 5.1.

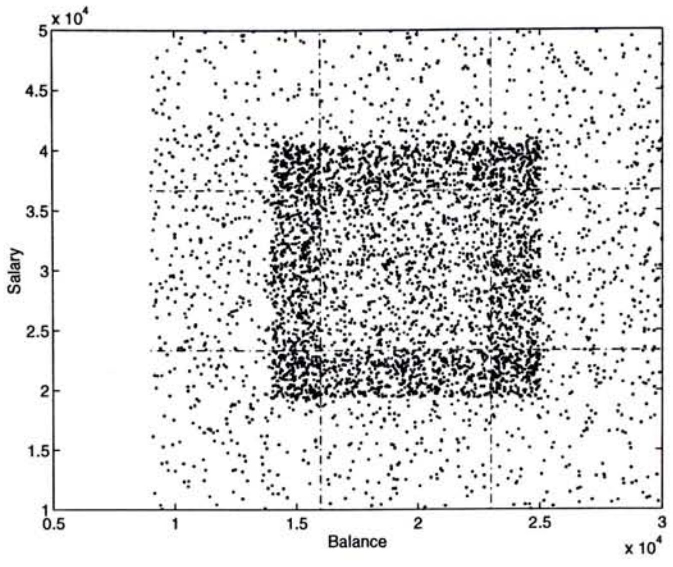


Figure 5.1: Data Distribution of Experiment One.

The horizontal-axis indicates the data distribution of *Balance* and the vertical-axis is the data distribution of *Salary*. The database is generated in such a way that no single region will be considered as important for reasonably high *significance threshold*. However, we can see that most of the data are clustered around the center region as well as its boundary. Therefore, we expect that there should be some interesting information existed in the database. For example, a rule indicates that people with medium salary will imply they have medium balance or people with medium balance will imply they have medium salary.

	Significance = 0.15			Significance = 0.2		
	Candidate	Large	Rule	Candidate	Large	Rule
Discrete	15	7	0	15	7	0
Significance	15	11	6	15	7	2
Correlation	15	11	6	15	7	2

Table 5.1: Result Of Experiment One.

Table 5.1 shows the experimental result. We can see that all methods have similar number of candidate and large itemsets. However, the discrete interval method cannot find any rule of which *confidence* is greater than 50%. It is because that method does not consider elements located outside interval boundaries. On the contrary, the other two methods have handled the boundary problem. Therefore, we can discover rules from the above database with *certainty* greater than 50%. We have set the *confidence* and *certainty threshold* to 50% such that the discovered rules are more reasonable. Moreover, we have set the user specified membership grade threshold to 0.6 such that records with low membership grade will not contribute to the discovered rules.

5.2 Experiment Two

This experiment is similar to the experiment one. We also use two attributes and there are three intervals/fuzzy sets for each attribute. However, the interval method will vary its intervals such that overlapped intervals may be produced. In this experiment, we want to show that even the overlapped intervals may cause problem in the mining process.

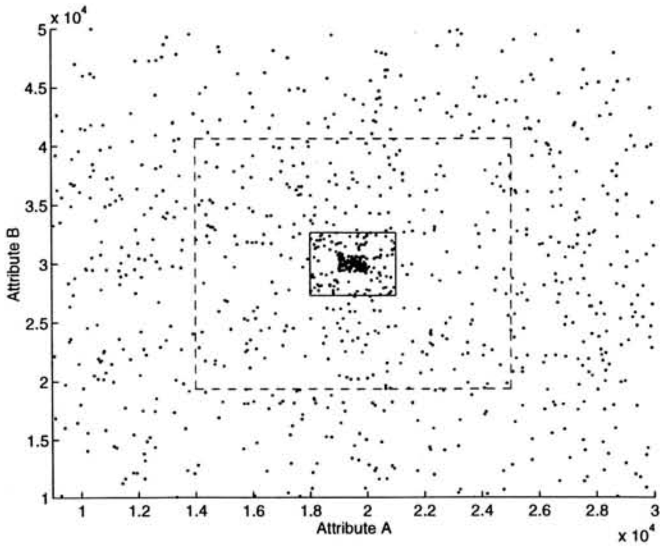


Figure 5.2: Data Distribution of Experiment Two.

In figure 5.2, the horizontal axis represents attribute A and the vertical axis represents attribute B. We generate the database such that records are clustered in the inner box. Therefore, we only expect that there are interesting information in the inner box. In this experiment, the *confidence* and *certainty* have been set to 50% to guarantee the usefulness of the discovered rules. Moreover, we have set the user specified membership threshold to 0.6. *Discrete1* uses the inner box as the interesting region and *Discrete2* uses the outer box.

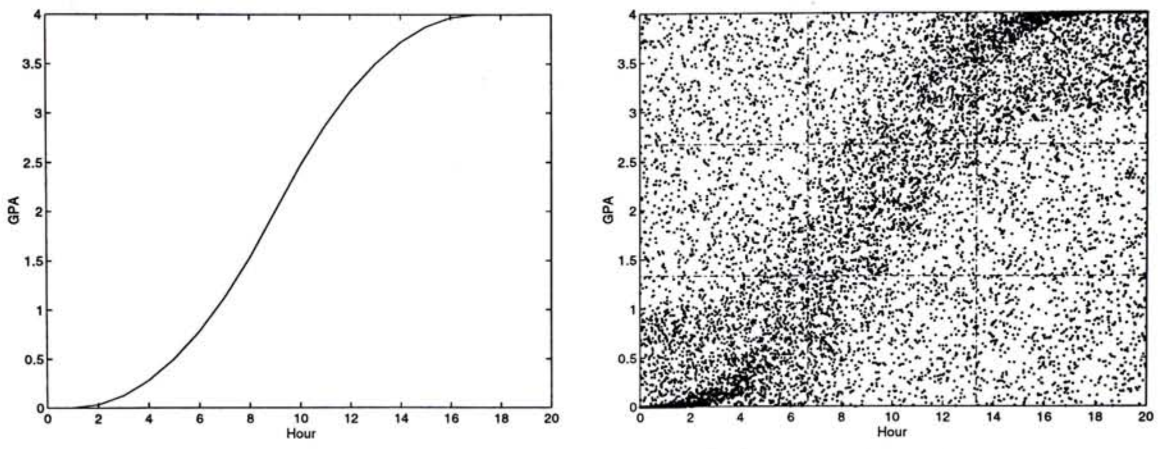
	$Significance = 0.25$		
Methods	*Candidate*	*Large*	*Rule*
Discrete1	15	6	0
Discrete2	3	3	2
Significance	7	3	2
Correlation	7	3	2

Table 5.2: Result Of Experiment Two.

We can see that *Discrete1* discovered more candidate and large itemsets, however, none of them can produce an interesting rule. To overcome the small interval problem, therefore, Discrete2 is used to find some interesting rules. However, the region is so large that the semantics of the rules becomes meaningless. On the contrary, the *significance* and *correlation* methods discovered fewer candidate and large itemsets than *Discrete1* but they can find rules from the interesting regions. Moreover, the semantics of the rules is preserved because the fuzzy sets have not overemphasized the sparse elements.

5.3 Experiment Three

We assume there is a relation between the working hour and the GPA of a student. The relation of the two attributes is shown in figure 5.3(a). The meaning of the relation is the GPA of a student will be high if he works hard. Otherwise, he will get a low GPA. The data are generated according to the relation curve in figure 5.3(a). In figure 5.3(b), we can see the data distribution of the two attributes and we can still see the shape of the curve in figure 5.3(a).



(a) The Relation Curve. (b) Data Distribution.

Figure 5.3: Database of Experiment Three.

The two attributes, *Hour* and *GPA*, have three intervals/fuzzy sets such that the plane of *Hour* and *GPA* is divided into nine regions. The relation curve passes through five of those regions. In figure 5.3(b), therefore, we can see that there are at least four areas which are heavily shaded out of the nine regions. This means there should be several rules existed in the database. For simplicity, fixed interval partition will be employed in this experiment.

The result of this experiment is shown in table 5.3. The results are quite similar to those of previous experiment. Moreover, the settings are also similar, i.e. membership threshold is 0.6 and certainty threshold is 50%. The significance threshold of this experiment has been set to 0.2 and 0.25.

In this experiment, we can see that both *significance* and *correlation* outperform the discrete interval method. They can discover more candidate and large itemsets as well as interesting rules. When significance threshold is 0.2, *significance* method can find 5 rules and *correlation* method can discover 10 rules. The discrete interval method can only find 2 rules. All the discovered rules are lo-

	Significance = 0.2			Significance = 0.25		
	Candidate	Large	Rule	Candidate	Large	Rule
Discrete	15	7	2	15	6	0
Significance	15	11	5	15	10	5
Correlation	15	11	10	15	10	8

Table 5.3: Result Of Experiment Three.

cated in the heavily shaded region which means the rules should be considered as interesting. When significance threshold increased to 0.25, our methods can still discover rules satisfying all thresholds. However, the discrete interval method find nothing but only some candidate and large itemsets.

5.4 Experiment Four

We have shown that the methods utilize fuzzy set concept can find more rules than the discrete interval method. In this section, we will give the experimental results on the performance of the three methods, i.e. discrete, significance and correlation. We will measure the performance of these methods when the database size is changing. For a fair comparison, we fix the number of rules to be discovered such that all methods will find the same number of rules.

In this experiment, there are three attributes in the database. Each attribute has three intervals/fuzzy sets. For discrete interval method, we assume the interval size will not change for simplicity. We have set the user specified parameters such that all three methods will give the same number of rules. We have run the programs with database size ranging from 5000 to 100000 records. Figure 5.4, shows the execution time of the three methods.

In figure 5.4, the execution of all methods grow linearly as the number of

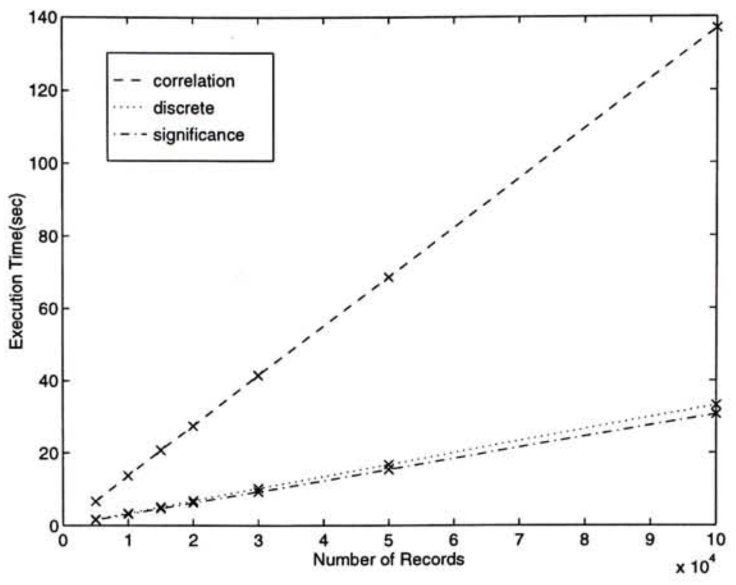


Figure 5.4: Performance Of Three Methods With Fixed Number Of Rules.

records increases. Therefore, we need not worry about the time for mining association rules will grow exponentially as the database size increased. There are three curves in figure 5.4. The first curve indicates the execution time of *correlation* method. The second one is the performance curve for discrete interval method and the third one is for *significance* method.

As we can see, the first curve tells us that the correlation method needs more time than the other two. From the other curves, moreover, we can see that the discrete interval and significance methods have similar performance. As mentioned above, the correlation method uses some augmented statistical concepts to compute the *certainty* of rule. Therefore, it can discover more rules than other methods. The performance of this method, however, turns out to be the worst because we have to scan the database again when we calculate the certainty factor of the discovered rules. The significance method, on the other hand, need not scan the database such that the significance method is faster than the correlation method. Moreover, the significance method gives comparable performance with respect to the discrete interval method and finds more rules than the discrete interval method. Therefore, the trade-off between significance and correlation methods is performance and number of rules to be discovered.

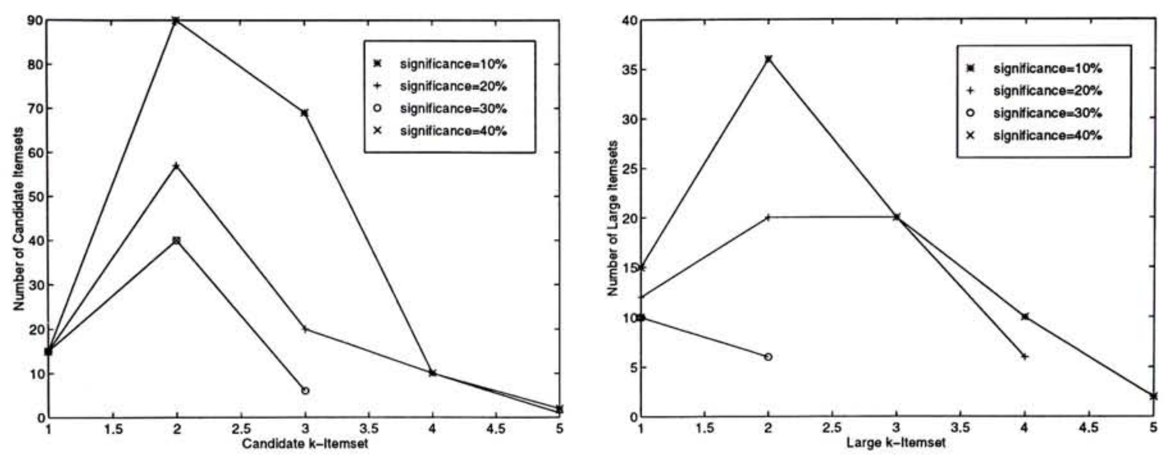
5.5 Experiment Five

We have compared our algorithms with the interval method in the previous sections. In this section, we will give extensive experiments on our proposed algorithms. We will compare the characteristics of our methods under different situations. We will vary the value of significance, membership and certainty threshold to test the generation of interesting itemsets, i.e. the number of candidate and large itemsets, discovery of interesting rules as well as the performance of the two algorithms. Hence, we can understand how the various thresholds affect the accuracy and performance of our algorithms. In this experiment, we use database with five attributes and each attribute will have three fuzzy sets associated with it.

5.5.1 Number of Itemsets

Our algorithms use the same itemset generation procedure to find all interesting candidate and large itemsets. In this experiment, we will test the itemset generation function against different significance and membership thresholds. We will not consider the certainty threshold in this experiment since it does not affect the generation of candidate and large itemsets. In this experiment, we have fixed the membership threshold to 0.6 and the certainty threshold to 50% such that the discovered itemsets and rules will be reasonably interesting and useful.

In figure 5.5, we have shown the number of candidate and large k-itemsets generated under different significance thresholds. The threshold values have been set to 10%, 20%, 30% and 40%. The curves in figure 5.5(a) represent the number of candidate itemsets generated under the above significance threshold settings and the curves in figure 5.5(b) show the number of large itemsets generated in the experiment.

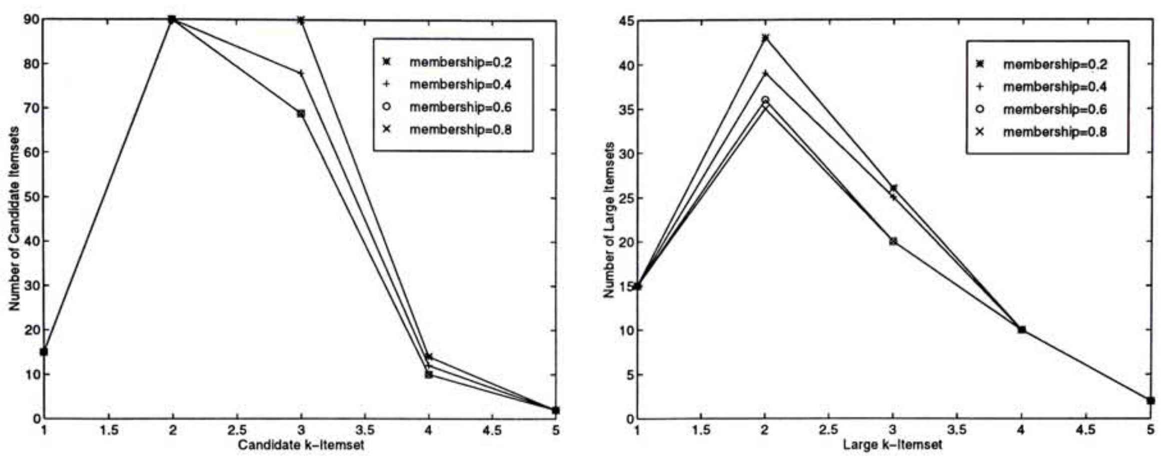


(a) Candidate Itemset (b) Large Itemset

Figure 5.5: Number of Itemsets Under Different Significance Threshold.

We can see that the number of itemsets reduce a lot when the significance threshold increases. In figure 5.5(a), the effect is obvious when the significance value changes from 10% to 20%. It is because the significance threshold is used for pruning uninteresting itemsets. Therefore, the itemsets without enough significance values will be discarded as the significance threshold increases. By varying the significance threshold, we can only increase or decrease the number of generated candidate and large itemsets. However, the significance threshold does not affect the significance value of an itemset.

After testing the effect of significance threshold, we will show the effect of membership threshold. We want to illustrate how the membership threshold affects the generation of candidate and large itemsets. In this experiment, We have fixed the significance threshold to 20% and certainty threshold to 50%. The membership threshold have been set to 0.2, 0.4, 0.6 and 0.8. The results are presented in figure 5.6.



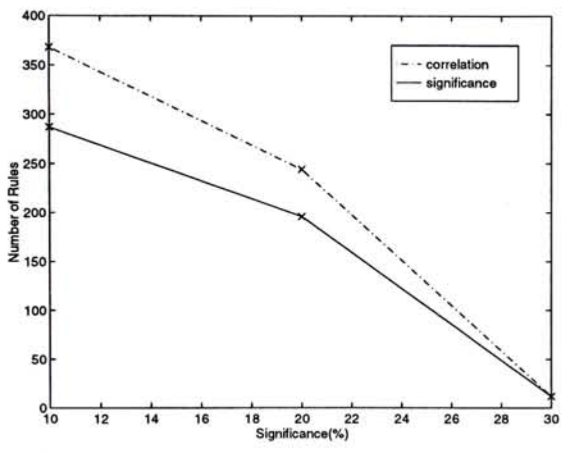
(a) Candidate Itemset (b) Large Itemset

Figure 5.6: Number of Itemsets Under Different Membership Threshold.

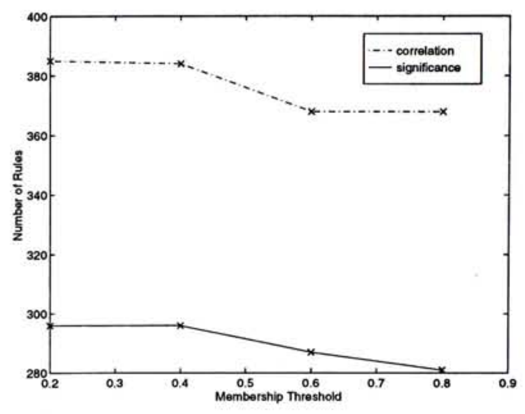
In figure 5.6, we can see the changes in number of candidate and large itemsets as we vary the membership threshold. The reason is that the significance value of an itemset will be affected by the changes of membership threshold. If the membership value of an attribute is less than the threshold, the value will be considered as zero. Therefore, if the membership threshold is high, fewer boundary elements contribute their votes to an itemset such that decreasing the significance value of the itemset. However, the boundary elements only partially contribute their votes to an itemset. Therefore, the effect of varying membership threshold is less obvious than the effect of varying significance threshold. We can use the membership threshold to control the consideration of boundary elements.

5.5.2 Number of Rules

Figure 5.7 shows number of rules generated by our algorithms with varying significance and membership thresholds. In figure 5.7(a), the effect of varying significance threshold is presented. Similar to the previous experiment, we have fixed the membership threshold to 0.6 and certainty threshold to 50%. The significance threshold has been set to 10%, 20%, 30% and 40%. The result of varying membership threshold is shown in figure 5.7(b). The significance and certainty threshold have been set to 10% and 50% respectively. The membership threshold being tested is 0.2, 0.4, 0.6 and 0.8.



(a) Varying Significance Threshold



(b) Varying Membership Threshold

Figure 5.7: Number of Rules Generated.

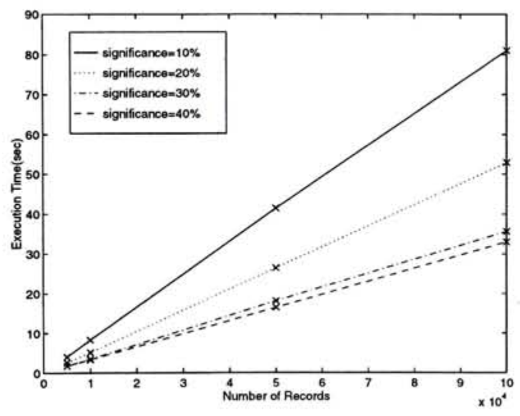
As mentioned above, significance threshold greatly reduces the number of candidate and large itemsets such that there are fewer rules generated in the mining process when the significance threshold increases. However, the effect is not so obvious when the membership threshold is changed. There are only several rules which have been dropped when the membership threshold increases. The experimental results in figure 5.7 conform to the properties of significance and membership threshold mentioned above.

5.6 Experiment Six

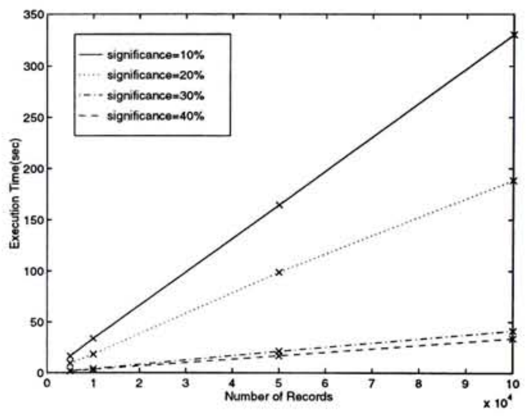
After describing the behavior of itemsets and rule generation under different significance and membership thresholds, we will discuss the performance of the methods under different significance, membership and certainty thresholds. The database settings are similar to that of experiment four.

5.6.1 Varying Significance Threshold

We will first talk about the performance of the two algorithms under varying significance threshold. Figure 5.8 shows the performance of the two algorithms. As mentioned above, the significance method outperforms the correlation method several times. This can be reflected by the figure itself. However, the performance curves of the two algorithms have similar trends, i.e. the execution time becomes faster when the significance threshold increases. It is because the candidate and large itemsets generated by both the algorithms decrease greatly as the significance threshold increases. Therefore, the time spent in counting the database will be reduced.



(a) Significance Method



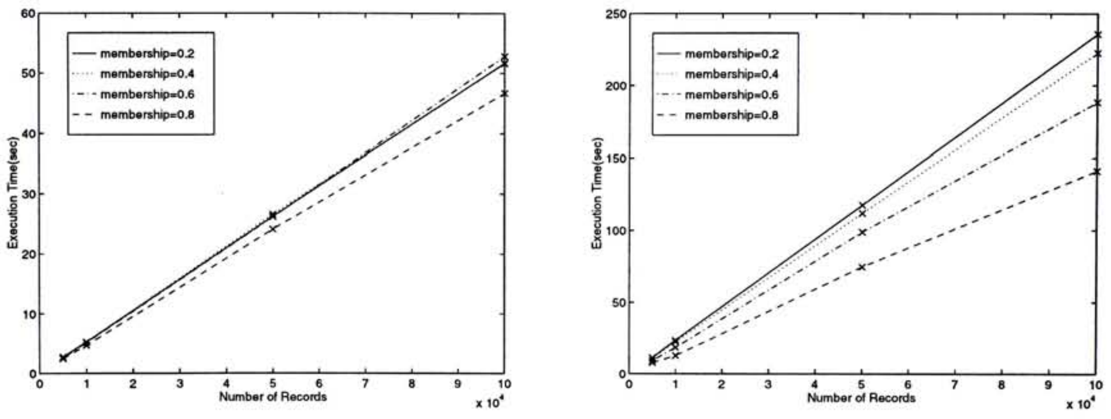
(b) Correlation Method

Figure 5.8: Performance Under Different Significance Threshold.

5.6.2 Varying Membership Threshold

Figure 5.9 shows similar trends as in figure 5.8. It is because the membership threshold also affects the number of generated candidate and large itemsets.

However, the effect is soft and not so obvious. Therefore, the performance under different membership thresholds does not change greatly. In figure 5.9, the performance curves of significance method have been clustered together because the execution time of this algorithm is dominated by the generation of candidate and large itemsets. Since the changes of candidate and large itemsets is not significant, there are less changes in the execution time. On the other hand, the performance of the correlation method is dominated by both itemset and rule generation. If there are fewer itemsets, we can be sure that there will be fewer rules generated such that the program will become faster. As a result, the changes of the performance curves of the correlation method are greater than those of significance method.

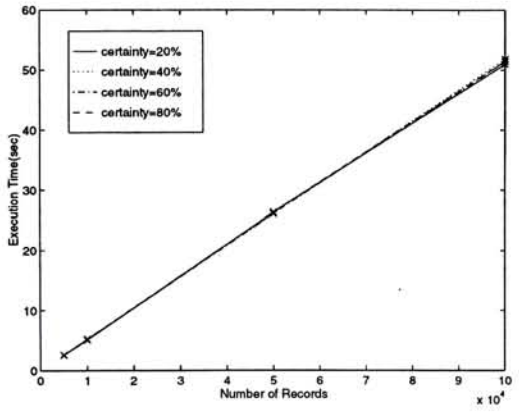


(a) Significance Method (b) Correlation Method

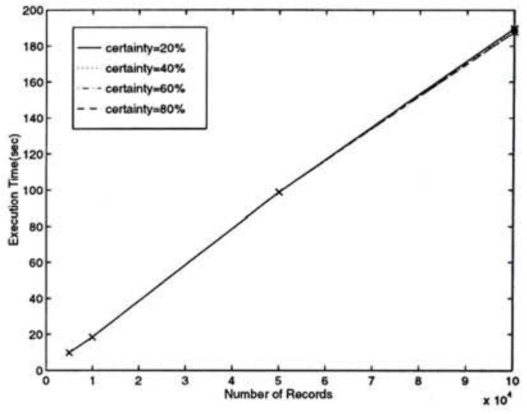
Figure 5.9: Performance Under Different Membership Threshold.

5.6.3 Varying Confidence Threshold

Finally, we will show the effect of varying certainty threshold. We have fixed the significance and membership thresholds to 20% and 0.6. Moreover, the certainty threshold is ranging from 20% to 80%. In figure 5.10, the performance curves are clustered together for both the methods. It is because the certainty threshold does not affect the generation of candidate and large itemsets. Therefore, the execution time of significance method is nearly unchanged under different certainty thresholds since the dominant factor of the algorithm is the number of generated candidate and large itemsets. However, the performance of correlation method is based on both itemsets and rule generation but it still have nearly constant execution time under different certainty threshold. It is because the potential interesting rules generated by the large itemsets are the same. We still have to calculate the certainty values of these rules under different certainty threshold. Therefore, the execution time of the correlation method is not affected by the certainty threshold.



(a) Significance Method



(b) Correlation Method

Figure 5.10: Performance Under Different Certainty Threshold.

Chapter 6

Discussions

In the previous chapters, we have concentrated on the comparison of discrete interval. In this chapter, we will compare our algorithm with the algorithm in [29] which partition the attribute domain into small discrete intervals and then merge these intervals into either discrete or overlapped intervals. We will discuss the advantages and disadvantages of our algorithm and the one in [29]. We will also describe the problem introduced by overlapped intervals.

6.1 User guidance

In [29], the algorithm first partitions the attribute domain into small intervals and then combines those intervals into larger discrete or overlapped intervals which are expected to have greater support. However, the partitioning and merging of intervals is heuristic and blind such that there are some problems using the interval methods.

Suppose the interval method can generate overlapped intervals from small intervals and we have the data distribution of attributes A and B in figure 6.1. From the data distribution graph, we can see that only one area is heavily shaded.

In figure 6.1, the interval method generates three intervals/regions which cover the same area.

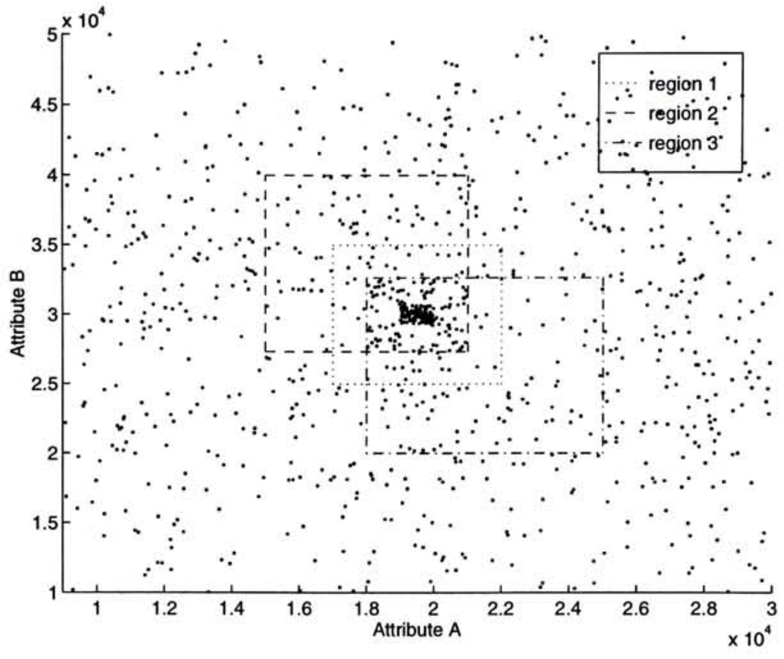


Figure 6.1: Duplicated Intervals.

However, we know that they convey no extra information. Therefore, we should discard one of them such that we will not have intervals which have duplicated information. We should eliminate the duplicated intervals as early as possible because the extra intervals will become new attributes and potential candidate and large itemsets such that influence the performance of the mining process. In [29], the elimination is done in the rule generation process and the time costed in counting the extra candidate and large itemsets still slows down the mining process. Hence, an elimination scheme should present in the mining process such that the duplicated intervals will be dropped in the early stage of the discovery process, i.e. the process of finding candidate and large itemsets.

Besides, the interval method will cause another problem. In figure 6.2, we can see that data are clustering around two intervals/regions and these two regions are separated very far. The two little boxes indicate the intervals. These two intervals have insufficient support. Therefore, the interval method will try to merge two intervals into one in order to generate an interval with enough support, i.e. the larger box.

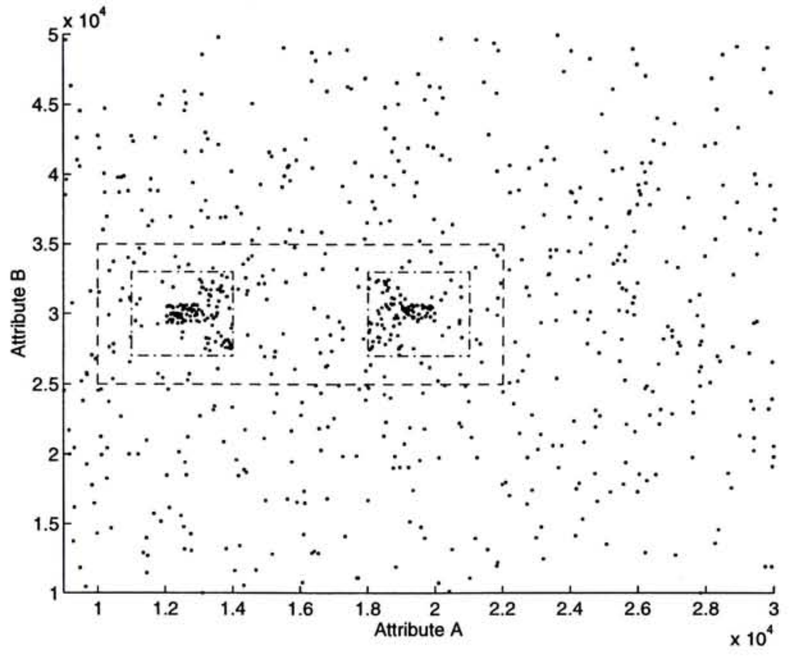


Figure 6.2: Meaningless Interval.

However, as we can see, the two intervals are separated so far such that the new interval is quite large. It covers more than a half of the range of attribute A. If we use this interval to generate rules, the semantics of the rules will be changed. Moreover, if this interval is the right one, we need not partition the attribute domain in the first place.

6.2 Rule understanding

The semantics of the discovered rule is very important in knowledge discovery. Although humans are familiar with numbers , word description is more desirable. Moreover, the expressive power of words is stronger than numbers. Therefore, if we can accomplish some linguistic terms in the discovered rules, the expressive power of the rules will surely be increased.

The discrete and overlapped intervals only use lower and upper bounds as indication. Users can hardly understand the meaning of rules with those numbers. Therefore, extra mapping may be needed such that users can have a clearer understanding of those rules. On the contrary, users can understand the fuzzy association rules without any difficulty because we have placed linguistic terms in the rules other than numbers. For each fuzzy set, there is a linguistic term associated with it and the meaning is defined by domain experts or knowledge engineers. These fuzzy sets are used to characterize the attributes in the database. Therefore, we need not perform extra mapping in our algorithm since the linguistic terms have already reflected the meaning of the fuzzy association rules sufficiently.

6.3 Number of rules

In our algorithm, the fuzzy sets of each attribute used by the mining process are defined by domain expert or knowledge engineer. The number of these fuzzy sets is expected to be small and finite. As a result, the number of rules discovered will be limited by the number of fuzzy sets defined by the experts. There are, however, several ways to increase the number of rules.

As we mentioned above, each fuzzy set is associated with a membership function which tells whether an element belongs to the fuzzy set. Since the membership function is a mapping from an attribute domain to [0,1], we can increase the number of rules simply by modifying the membership function. The simplest and intuitive function which can apply to the membership are square and square root functions. For example, if we have a fuzzy set *young* and its membership function m_{young}, we will have fuzzy sets called *quite young* as well as *very young* and their membership functions are as follows:

$$\forall d \in D_{young}, \quad m_{quite\ young}(d) = \sqrt{m_{young}(d)}$$

$$m_{very\ young}(d) = m_{young}(d)^2$$

Another way to increase the number of rules is to utilize the partition method in [29]. After partitioning the attribute domain into intervals, we can assign fuzzy sets to these intervals and hence increase the number of fuzzy association rules. Using this method, we not only can increase the number of rules, but also can handle the problem introduced by attribute partitioning method.

Chapter 7

Conclusions and Future Works

In this thesis, we have proposed a method to handle quantitative attributes. We assign each attribute with several fuzzy sets which characterize the quantitative attribute. Using the fuzzy set concept, we want to find the fuzzy association rules of the form, 'If X is A then Y is B'. The antecedent and consequent of the rule contain multiple *attribute-fuzzy set* pairs. If a rule is considered as potentially interesting, there must be sufficiently large amount of records supporting the antecedent and consequent. It means that the *significance* of both antecedent and consequent should be as high as the user specified value. However, it is not sufficient to determine the usefulness of the rules. An interesting and useful rule should also have enough *certainty* value.

We have described the procedure to find fuzzy association rules and perform several experiments. In those experiments, we have shown that our algorithm has solved the problem introduced by partitioning the attribute domain into intervals. We have utilized the fuzzy set concept such that we can handle the sharp boundary without over-emphasizing the boundary elements. We have used two methods to measure the interestingness of the fuzzy association rules. One of the method uses the *significance* of the *large itemsets* to compute the *certainty* factor. The other uses the correlation of the antecedent and consequent of the rule as the certainty factor. In the experiments, we have found that the method uses significance to calculate certainty factor will give a better performance. However, the method uses correlation as certainty factor will give more accurate results. Therefore, users can make use of the two interest measures by their interests.

In this thesis, the fuzzy association rule has the form, 'If X is A then Y is B', where $X = \{x_1, x_2, ..., x_l\}$, $Y = \{y_1, y_2, ..., y_k\}$, $A = \{a_1, a_2, ..., a_l\}$ and $B = \{b_1, b_2, ..., b_k\}$ are ordered sets. Moreover, a_i and b_j correspond to x_i and j_j respectively. Moreover, we use the multiplication operator as the fuzzy set function to calculate the significance and certainty so that the semantics of the rule is as follows.

$$\text{If } (x_1 \text{ is } a_1) \text{ and } (x_2 \text{ is } a_2) \text{ and ... and } (x_l \text{ is } a_l)$$
$$\text{then } (y_1 \text{ is } b_1) \text{ and } (y_2 \text{ is } b_2) \text{ and } (y_k \text{ is } b_k).$$

In the above rule, the antecedent and consequent are conjunction of attribute-fuzzy set pairs. However, fuzzy association rules can have other forms which give different semantics. The following are possible forms of fuzzy association rule.

- Disjunction of attribute-fuzzy set pairs

  One of the possible forms of fuzzy association rule is to replace *and* in the above rule by *or*. The resulting rule is as follows.

$$\text{If } (x_1 \text{ is } a_1) \text{ or } (x_2 \text{ is } a_2) \text{ or ... or } (x_l \text{ is } a_l)$$
$$\text{then } (y_1 \text{ is } b_1) \text{ or } (y_2 \text{ is } b_2) \text{ or } (y_k \text{ is } b_k).$$

  The attribute-fuzzy set pairs are joined by *or*. Finding this form of rules, we have to augment the calculation of *significance* and *certainty*. One of the possible ways is to use the maximum operator as *opr* operator, i.e.

$$Significance = \frac{\text{Sum of votes satisfying } \langle X, A\rangle}{\text{Number of records in database } T}$$

$$\begin{aligned} S_{\langle X,A\rangle} &= \frac{\sum_{t_i\in T} opr_{x_j\in X}\{\alpha_{a_j}(t_i[x_j])\}}{total(T)} \\ &= \frac{\sum_{t_i\in T} Max_{x_j\in X}\{\alpha_{a_j}(t_i[x_j])\}}{total(T)} \end{aligned}$$

$$\text{where} \quad \alpha_{a_j}(t_i[x_j]) = \begin{cases} m_{a_j}(t_i[x_j]) & \text{if } m_{a_j} \geq \text{threshold}, \\ 0 & \text{otherwise.} \end{cases}$$

However, we must be careful when we use the maximum operator. Since the maximum operator will find the largest membership value, it will ignore the effects of low membership values so that it may generate incorrect large itemsets and rules. We can prevent this problem by defining a threshold value. For example, we have a set of membership values {0.5, 0.6, 0.4, 0.7, 0.9} and two threshold values 0.4 and 0.6. For the first threshold value, the maximum operator will return 0.9. For the second one, we will have zero instead.

- Conjunction of fuzzy sets

  As mentioned above, each attribute is associated with a fuzzy set in a fuzzy association rule. It is, however, possible to have several fuzzy sets correspond to an attribute in a fuzzy association rule. The possible form of fuzzy association rule is as follows.

  If x_1 is ($f^1_{x_1}$ and ... and $f^p_{x_1}$) then y_1 is ($f^1_{y_1}$ and ... and $f^q_{y_1}$).

  In fuzzy set concept, the conjunction of fuzzy sets produces a new fuzzy set. Let f_x represents the conjunction of the fuzzy sets, f^i_x. The membership function of f_x is as follows.

  $$\forall d \in D_x, m_{f_x}(d) = Min(m_{f^i_x}(d))$$

  For each attribute value, the value of m_{f_x} is the lowest value of $m_{f^i_x}$. After obtaining the new fuzzy sets, we can generate large itemsets and fuzzy association rules without modifying the calculation of significance and certainty.

- Disjunction of fuzzy sets

  Similar to conjunction of fuzzy sets, an attribute can associate with disjunction of fuzzy sets. The following is the new form of fuzzy association rule.

  $$\text{If } x_1 \text{ is } (f^1_{x_1} \text{ or ... or } f^p_{x_1}) \text{ then } y_1 \text{ is } (f^1_{y_1} \text{ or ... or } f^q_{y_1}).$$

  The fuzzy set which represents the disjunction of fuzzy sets is produced by the maximum membership value of $m_{f^i_x}$. The following formula is similar to the one representing conjunction of fuzzy sets.

  $$\forall d \in D_x, m_{f_x}(d) = Max(m_{f^i_x}(d))$$

  Threshold may be needed if we do not want to consider low membership values.

Bibliography

[1] Rakesh Agrawal, Sakti Ghosh, Tomasz Imielinski, and Arun Swami. An interval classifier for database mining applications. In *19th International Conference on Very Large Databases (VLDB)*, pages 560–573, Vancouver, Canada, 1992.

[2] Rakesh Agrawal, Tomasz Imielinski, and Arun Swami. Data mining: a performance perspective. *IEEE Transactions on Knowledge and Data Engineering*, Vol. 5(No. 6):914–925, December 1993.

[3] Rakesh Agrawal, Tomasz Imielinski, and Arun Swami. Mining association rules between sets of items in large databases. In *Proceedings of the ACM SIGMOD Conference on Management of Data*, pages 207–216, Washington D.C., May 1993.

[4] Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules in large databases. In *Proceedings of the 20th International Conference on Very Large Databases (VLDB)*, pages 487–499, Santiago, Chile, Sept. 1994. Expanded version available as IBM Research Report RJ9839, June 1994.

[5] J.F. BALDWIN. Inference for information systems containing probabilistic and fuzzy uncertainties. In Lotfi Asker. Zadeh and Janusz. Kacprzyk, edi-

tors, *Fuzzy Logic for the Management of Uncertainty*, pages 353–375, New York, 1992. Wiley.

[6] Hans Bandemer and Wolfgang Näther. *Fuzzy Data Analysis.* Kluwer Academic Publishers, Dordrecht, Netherlands; Boston, 1992.

[7] D. Cheung, V.T. Ng, A. Fu, and Y. Fu. Efficient mining of association rules in distributed databases. In *IEEE Transactions on Knowledge and Data Engineering*, pages 1–23, 1996.

[8] Didier Dubois and Henri Prade. *Possibility theory : an approach to computerized processing of uncertainty.* Plenum Press, New York, 1988.

[9] Usama M. Fayyad and Ramasamy Uthurusamy. Efficient algorithms for discovering association rules. In *AAAI Workshop on Knowledge Discovery in Databases*, pages 181–192, Seattle, Washington, July 1994.

[10] Takeshi Fukuda, Yasuhiko Morimoto, Shinichi Morishita, and Takeshi Tokuyama. Data mining using two-dimensional optimized association rules: Scheme, algorithms, an visualization. In *Proceedings of ACM SIGMOD*, pages 13–23, 1996.

[11] Takeshi Fukuda, Yasuhiko Morimoto, Shinichi Morishita, and Takeshi Tokuyama. Mining optimized association rules for numeric attributes. In *Proceedings of the Fifteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 182–191, Montreal, 1996.

[12] Andreas Geyer-Schulz. *Fuzzy Rule-based Expert Systems and Genetic Machine learning.* Physica-Verlag, Heidelberg, 1995.

[13] Jiawei Han and Yongjian Fu. Discovery of multiple-level association rules from large databases. In *21st International Conference on Very Large Databases (VLDB)*, pages 420–431, Zürich, Switzerland, Sept. 1995.

[14] Marcel Holsheimer and Arno P.J.M. Siebes. Architectural support for data mining. Technical Report CS-R9429, CWI, 1994.

[15] Marcel Holsheimer and Arno P.J.M. Siebes. Data mining: the search for knowledge in databases. Technical Report CS-R9406, CWI, January 1994.

[16] Maurice A. W. Houtsma and Arun N. Swami. Set-oriented mining for association rules in relational databases. In *Proceedings of the Eleventh International Conference on Data Engineering*, pages 25–33, Taipei, 1995.

[17] Janusz KACPRZYK and Cezary IWAŃSKI. Fuzzy logic with linguistic quantifiers in inductive learning. In Lotfi Asker. Zadeh and Janusz. Kacprzyk, editors, *Fuzzy Logic for the Management of Uncertainty*, pages 465–478, New York, 1992. Wiley.

[18] *Lászl*ó T. KÓCZY and Kaoru HIROTA. A fast algorithm for fuzzy inference by compact rules. In Lotfi Asker. Zadeh and Janusz. Kacprzyk, editors, *Fuzzy Logic for the Management of Uncertainty*, pages 297–317, New York, 1992. Wiley.

[19] Abranham Kandel. *Fuzzy Expert Systems.* CRC Press, Boca Raton, Fla., 1992.

[20] George J. Klir and Tina A. Folger. *Fuzzy sets, uncertainty, and information.* Prentice Hall, Englewood Cliffs, N.J., 1988.

[21] Brian Lent, Arun Swami, and Jennifer Widom. Clustering association rules. In *Proceedings of the Thirteenth International Conference on Data Engineering*, pages 220–231, Birmingham U.K., 1997.

[22] Vilém NOVÁK. Fuzzy logic as a basis of approximate reasoning. In Lotfi Asker. Zadeh and Janusz. Kacprzyk, editors, *Fuzzy Logic for the Management of Uncertainty*, pages 247–264, New York, 1992. Wiley.

[23] Jong Soo Park, Mink-Syan Chen, and Philip S. Yu. An effective hash-based algorithm for mining associatin rules. In *Proceedings of the ACM SIGMOD Conference on Management of Data*, pages 175–186, San Jose, 1995. ACM.

[24] Zdzislaw PAWLAK. Rough sets: A new approach to vagueness. In Lotfi Asker. Zadeh and Janusz. Kacprzyk, editors, *Fuzzy Logic for the Management of Uncertainty*, pages 105–118, New York, 1992. Wiley.

[25] Gregory Piatesky-Shapiro and William J. Frawley. *Knowledge Discovery in Databases.* AAAI Press/The MIT Press, Menlo Park, California, 1991.

[26] Ashok Sarasere, Edward Omiecinsky, and Shamkant Navathe. An efficient algorithm for mining association rules in large databases. In *Proceedings of the 21st International Conference on Very Large Databases (VLDB)*, pages 432–444, Zürich, Switzerland, Sept. 1995. Also Gatech Technical Report No. GIT-CC-95-04.

[27] Ning Shan, Wojciech Ziarko, Howard J. Hamilton, and Nick Cercone. Using rough sets as tools for knowledge discovery. In *Proceedings of the First International Conference on Knowledge Discovery and Data Mining*, pages 263–268, Montreal, 1995.

[28] Andrzej Skowron and Zbigniew Suraj. Discovery of concurrent data models from experimental table: A rough set approach. In *Proceedings of the First International Conference on Knowledge Discovery and Data Mining*, pages 288–293, Montreal, 1995.

[29] Ramakrishnan Srikant and Rakesh Agrawal. Mining quantitative association rules in large relational tables. In *Proceedings of the ACM SIGMOD Conference on Management of Data*, pages 1–12, Montreal, June 1996.