Department of Computing

Techniques For Improving Clustering And Association Rules Mining From Very Large Transactional Databases

Yanrong Li

This thesis is presented for the Degree of
Master of Philosophy
of
Curtin University of Technology

August 2009

Declaration

This thesis contains no material which has been accepted for the award of any other degree or diploma in any university.

To the best of my knowledge and belief, this thesis contains no material previously published by another person except where due acknowledgement has been made.

Signature: …….

Date: ……

To my son

Contents

List of Figures

List of Tables

Abstract

Clustering and association rules mining are two core data mining tasks that have been actively studied by data mining community for nearly two decades. Though many clustering and association rules mining algorithms have been developed, no algorithm is better than others on all aspects, such as accuracy, efficiency, scalability, adaptability and memory usage. While more efficient and effective algorithms need to be developed for handling the large-scale and complex stored datasets, emerging applications where data takes the form of streams pose new challenges for the data mining community. The existing techniques and algorithms for static stored databases cannot be applied to the data streams directly. They need to be extended or modified, or new methods need to be developed to process the data streams.

In this thesis, algorithms have been developed for improving efficiency and accuracy of clustering and association rules mining on very large, high dimensional, high cardinality, sparse transactional databases and data streams.

A new similarity measure suitable for clustering transactional data is defined and an incremental clustering algorithm, INCLUS, is proposed using this similarity measure. The algorithm only scans the database once and produces clusters based on the user's expectations of similarities between transactions in a cluster, which is controlled by the user input parameters, a similarity threshold and a support threshold. Intensive testing has been performed to evaluate the effectiveness, efficiency, scalability and order insensitiveness of the algorithm.

To extend INCLUS for transactional data streams, an equal-width time window model and an elastic time window model are proposed that allow mining of clustering changes in evolving data streams. The minimal width of the window is determined by the minimum clustering granularity for a

particular application. Two algorithms, CluStream_EQ and CluStream_EL, based on the equal-width window model and the elastic window model respectively, are developed by incorporating these models into INCLUS. Each algorithm consists of an online micro-clustering component and an offline macro-clustering component. The online component writes summary statistics of a data stream to the disk, and the offline components uses those summaries and other user input to discover changes in a data stream. The effectiveness and scalability of the algorithms are evaluated by experiments.

This thesis also looks into sampling techniques that can improve efficiency of mining association rules in a very large transactional database. The sample size is derived based on the binomial distribution and central limit theorem. The sample size used is smaller than that based on Chernoff Bounds, but still provides the same approximation guarantees. The accuracy of the proposed sampling approach is theoretically analyzed and its effectiveness is experimentally evaluated on both dense and sparse datasets.

Applications of stratified sampling for association rules mining is also explored in this thesis. The database is first partitioned into strata based on the length of transactions, and simple random sampling is then performed on each stratum. The total sample size is determined by a formula derived in this thesis and the sample size for each stratum is proportionate to the size of the stratum. The accuracy of transaction size based stratified sampling is experimentally compared with that of random sampling.

The thesis concludes with a summary of significant contributions and some pointers for further work.

Preface

This thesis contains seven chapters with references appended at the end of it. The first chapter provides the background for the research while chapter 2 reviews related literature. Chapters 3 to 6 contain details of the research conducted for this thesis and chapter 7 concludes the thesis with a summary of contributions and some directions for future research.

Chapter 3 is based on the paper we published in the proceedings of the IEEE International Conference on Granular Computing (Atlanta, USA, 2006). Chapter 4 is the extension of work published in the proceedings of the 19th Australian Joint Conference on Artificial Intelligence (Tasmania, Australia, 2006). Chapter 5 contains results of the research published in the proceedings of the 17th Australian Joint Conference on Artificial Intelligence (Cairns, Australia, 2004) and contents of chapter 6 was published in the proceeding of the Second IFIP Conference on Artificial Intelligence Applications and Innovations (Beijing, China, 2005)

My Supervisor, Dr. Raj P. Gopalan is the co-author of these papers.

This research was initially conducted towards a PhD degree. Due to the changes in my personal circumstances, I had to take leave of absence from the PhD program for most of the time since September 2006. As I am unable to resume the PhD research soon, this thesis is being presented for the degree of Masters of Philosophy.

Acknowledgement

First, I would like to thank my supervisor, Dr. Raj P. Gopalan, for his guidance, for his patience and understanding, for the time and effort he put in to lead me through the whole journey of the research. For this, I will be always grateful.

I am grateful to my associate supervisor Dr. Narasimaha Achuthan for all his time and the valuable discussions.

I have received generous help from many people in the Department of Computing and the graduate studies office at the Faculty of Science and Engineering, to name a few, Dr. Michael Robey, Ms Mary Simpson, Ms Mary Mulligan, Ms Tamara Leonard and Ms Reisha Thomas. I would like to thank them for being supportive and helpful over the years.

I would like to thank my colleagues Amit Rudra and Yudho Sucahyo for sharing many ideas and the office space with me. I would also like to thank my peers Sharon Meng, Perry Peng, Monica Ou and Amal Ghanim for their friendships.

This thesis could not have come into existence without the previous work done by other researchers cited within the body of this thesis. I am grateful for the inspiration I could draw from them.

Chapter 1

Introduction

1.1 Background

The capacity of digital data storage worldwide has been doubling every nine months for at least a decade (Porter, 1998), and our ability to capture and store data has far outpaced our ability to process and utilize them (Fayyad and Uthurusamy, 2002). This growing challenge has produced a phenomenon called the data tombs, where data are deposited and in all likelihood will never be accessed again. Nevertheless, the deposited data are a potentially valuable resource. With appropriate data analysis tools, new knowledge can be discovered from the existing databases. Data mining is one of the most general approaches for such a purpose (Fayyad and Uthurusamy, 2002)

Data mining is the process of extracting valid, useful and previously unknown information from large databases, such as patterns, statistical models of data and relationships among parts of data, that can be used to make crucial business decisions or to guide scientific activities (Fayyad *et al.*, 1996). What kind of knowledge is embedded in the collected data and how to discover them effectively and efficiently are among the questions faced by researchers in the data mining community. Cluster analysis and association rules mining are two core data mining approaches to answer such questions (Han and Kamber, 2000).

Clustering is used to partition a dataset into a set of clusters such that similar objects are in the same cluster while dissimilar objects are in different clusters. Clustering has many applications in marketing, land use, insurance, city planning, etc. (Han and Kamber, 2000). For example, cluster analysis can help marketers discover distinct groups in their customer bases, and then use this knowledge to develop targeted marketing programs. An insurance company can use clustering to indentify groups of insurance policy holders with high claim costs.

Association rules mining, on the other hand, discovers relationships among items in a transactional database. It is to find the probability that one set of items will appear in a transaction whenever another set of items appear in the same transaction. Association rules mining can be applied in the areas of cross-marketing, catalog design, sale campaign analysis, web log (click stream) analysis, webpage linkage design, etc.

Cluster analysis can be performed on any type of data, including numerical, categorical or spatial data while association rule mining is defined only for transactional databases. The algorithms for clustering and association rules mining have been intensively studied for the stored databases during the last two decades (Jain *et al.*, 1999; Berkhin, 2002; Han and Kamber, 2006), but none of the algorithms is better than the others on all aspects, such as accuracy, efficiency, scalability, adaptability and memory usage. While more efficient and effective algorithms need to be developed for handling the large-scale and complex stored datasets, emerging applications where data arrive in streams pose new challenges for the data mining community. The existing techniques and algorithms for static stored databases cannot be applied to the data streams directly. They need to be extended or new methods need to be developed to process the data streams.

Clustering algorithms can be categorized as structure imposing algorithms and structure seeking algorithms. Cluster imposing algorithms produce k clusters based on user's input of a k value regardless of the underlying structures in the data. K-Means (MacQueen, 1967) is a representative of cluster imposing algorithms. Since in most cases the user does not know a

priori the cluster characteristics of the data, the value of k input by the user for a structure imposing algorithm may result in distinct clusters being merged when the real number of clusters is greater than the given value of k or big clusters being split into small chunks when k is greater than the real number of clusters. Moreover, for a given k, using different induction principles can produce the same number of clusters with different cluster features.

Structure seeking algorithms partition the database based on the user's expectations of the similarity of objects. LargeItem (Wang *et al.*, 1999) and CLOPE (Yang *et al.*, 2002) are two examples of such algorithms. The number of clusters output by a cluster seeking algorithm is determined by the user input values for the parameters that implicitly or explicitly define the degree of similarity among objects in a cluster, e.g. compulsion number r in CLOPE. Definitions of similarity or distance measures are often data and application dependant. Many similarity measures have been proposed in the literature for different types of data, such as Jacard coefficient for transactional data, and Euclidean distance for numerical data.

For a given dataset, different similarity or distance measures with different induction principles can result in different number of clusters with different cluster features. That is why we have a large number of clustering algorithms and yet we are still searching for more efficient and effective algorithms (Estivill-Castro, 2002). Unlike clustering, the set of association rules is fixed in a given dataset for given support and confidence thresholds. Therefore the focus in association rules mining is on improving efficiency. The input/output overhead in scanning the database plays an important role in the performance of association rule mining algorithms. Many data structures and corresponding algorithms have been developed to reduce the number of database scans from as many as the size of the longest frequent itemsets in (Agrawal and Srikant, 1994; Mannila *et al.*, 1994) to as small as two in (Han *et al.*, 2000) and one in (Cheung and Zaïane, 2003). When dealing with very large transactional databases or data streams, sampling techniques are applied as a tradeoff of accuracy for efficiency. While the

main purpose of sampling a static large disk resident database is to reduce the amount of data to be processed, sampling seems to be the only choice for processing a data stream where data flow faster than how quickly it can be processed (Babcock *et al.*, 2002).

Although many algorithms have been developed for clustering and association rules mining, more research is needed to develop effective and efficient algorithms for very large high dimensional sparse transactional databases and data streams.

1.2 Aims of This Thesis

This thesis aims to develop efficient and effective algorithms that are suitable for clustering very large high dimensional sparse transactional databases and data streams. The algorithms will be structure seeking rather than structure imposing. It will take into account users' expectations of the closeness among transactions when performing cluster analysis.

This thesis will also investigate sampling techniques that can improve the efficiency of mining association rules in a very large database. The sampling techniques will not only require small sample sizes but also provide approximation guarantees.

1.3 Overview of the Thesis

The reminder of this thesis is organized as follows.

Chapter 2 reviews previous research in clustering and association rules mining that are closely related to this research. These include the definitions of relevant terms, existing similarity measures for transactional data clustering, time window models proposed in the literature for data streams mining, and algorithms for clustering and association rules mining.

Chapter 3 defines a new measure for the similarity between transactions based on the items present in them. A new incremental structure seeking clustering algorithm INCLUS is then proposed incorporating the newly

defined similarity measure for clustering very large transactional databases. The algorithm is extensively tested and compared with two existing structure seeking clustering algorithms LargeItem and CLOPE.

Chapter 4 defines two time window models: *equal width time window model* and *elastic time window model* for mining evolving data streams. Two new algorithms named CluTranStream_EQ and CluTranStream_EL are proposed for clustering transactional data streams by incorporating these window models into INCLUS. Each algorithm consists of an online micro-clustering component and an offline macro-clustering component. The effectiveness and scalability of the online components are evaluated by experiments.

Chapter 5 presents a random sampling approach for association rules mining from very large databases. The sample size is determined based on binomial distribution and the central limit theorem. It has smaller sample size than that based on Chernoff Bounds, but still provides the same approximation guarantees. The accuracy of the proposed sampling approach is theoretically analyzed and its effectiveness is experimentally evaluated on both dense and sparse datasets. Methods for reducing false positives and false negatives in frequent itemsets are also discussed.

Chapter 6 explores the application of stratified sampling to association rules mining. The database is first partitioned into strata based on the length of transactions, and simple random sampling is then performed on each stratum. The effectiveness of the proposed sampling technique is evaluated and compared with simple random sampling.

Chapter 7 concludes the thesis with some directions for future research.

Some of the results of this thesis were published in (Li and Gopalan, 2004; 2005; 2006a; 2006b).

Chapter 2

Literature Review

Clustering and association rules mining have been active research areas in data mining, for which many algorithms have been developed (Jain *et al.*, 1999; Berkhin, 2002; Kantardzic, 2002; Gaber *et al.*, 2005; Han and Kamber, 2006; Hruschka *et al.*, 2009). This chapter reviews related research in these two fields that are directly relevant to the work being reported in this thesis.

2.1 Basic Concepts for Clustering and Association Rules Mining

As mentioned in Chapter 1, the goal of this thesis is to develop effective and efficient algorithms for clustering and association rules mining from very large transactional databases and data streams. Before the relevant existing algorithms being reviewed, the basic concepts and definitions associated with these problems are described below.

2.1.1 Transactional Database

Let $I = \{I_1, I_2, \dots, I_d\}$ be a set of d distinct items. A transaction T is a non-empty subset of I (i.e. $T \subseteq I$). A transactional database TDB is a collection of n transactions $\{T_1, T_2, \dots, T_n\}$, where d is the dimension of the TDB and n is the cardinality of the TDB. Each item is an attribute of the TDB. Transactional data is a special case of categorical data, where each attribute takes a value from a binary domain indicating either the presence or absence of an item in the transaction.

A super market basket database is a typical real life transactional database. Table 2.1 shows an example transactional database. There are six distinct items {A, B, C, D, E, F} and five transactions {A, B, D}, {A, C,D}, {A,D, E}, { B, E, F} and { B, C, D, E, F} in the database. In other words, the dimension of the database is 6 and the cardinality of the database is 5.

Table 2.1 A sample Transactional Database

Transaction ID	Items
10	A, B, D
20	A, C, D
30	A, D, E
40	B, E, F
50	B, C, D, E, F

2.1.2 Data Streams and Models for Data Stream Mining

A data stream is an ordered sequence of data records that can be read only once or a small number of times (Guha *et al.*, 2000a). Retail chain transactions, web logs and web page click streams, credit card transaction flows, and real time stock exchange data are some examples of real life data streams. The characteristics of data streams include the following (Han and Kamber, 2006):

- Huge volumes of continuous data, possibly infinite;
- Fast changing and requiring fast real time response;
- Random access is expensive to perform;
- Most data streams being at a low level and multi-dimensional, need multi-level, multidimensional processing.

There are four prominent models (Fig 2.1) for data stream mining which are described here in the context of clustering:

- The landmark model (Guha *et al.*, 2003) assumes that the clusters are to be computed over the entire data stream. The set of data points to be clustered includes all the data points from beginning of a data stream to the current time. Data stream clustering problem is simply viewed as a variant of one-pass clustering algorithms. It is suitable for a data stream where the mechanism of the data generation does not change over time.

- The sliding window model (O'Callaghan *et al.*, 2002; Babcock *et al.*, 2003) assumes that only the most recent data in the stream are of interest. The set of data points to be clustered is chosen by a sliding window of the most recent data. Clustering is performed from the beginning of the stream but only keeps clustering results for the set of data points within the sliding window.

- In the tilted time window model (Giannella *et al.*, 2003), at any moment, the stream is partitioned into windows of different granularities with respect to the time of their arrival. The most recent data has the finest granularity while the most remote has the coarsest.

- In the pyramidal time window model (Aggarwal *et al.*, 2003), the data stream is partitioned into windows based on various granularities, but only a certain number of windows is kept at any given time for each granularity. Both the tilted-window model and the pyramid window model can be employed for approximation of changes in the data stream. Sampling can also be incorporated into these models (O'Callaghan *et al.*, 2002). Existing algorithms for static datasets such as K-Means (or K-Median) have been incorporated into these models with or without modifications for data stream clustering (Ong *et al.*, 2004; MacQueen, 1967; O'Callaghan *et al.*, 2002; Aggarwal *et al.*, 2003; Guha *et al.*, 2003).

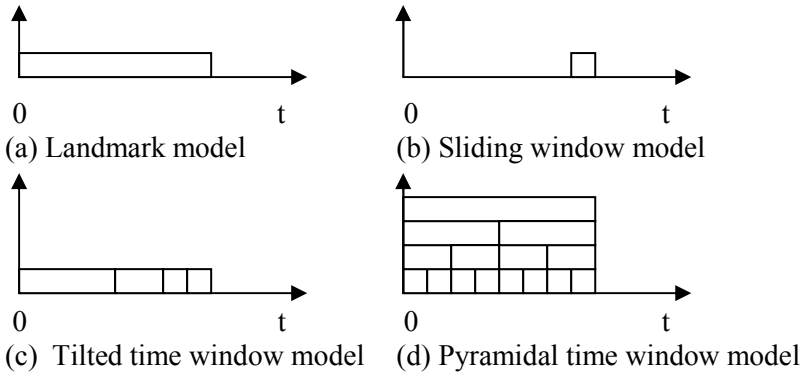


Figure 2.1 Models for Data Stream Processing

In this thesis, an equal-width window model and an elastic window model are proposed for clustering transactional data streams.

2.1.3 **Clustering**

A cluster is a collection of data objects. Clustering is to partition a dataset into clusters so that similar objects are in the same cluster while dissimilar objects are in different clusters based on predefined similarity/distance measures.

Given a transactional database $\{T_1, T_2, \dots, T_n\}$ over d items $\{I_1, I_2, \dots, I_d\}$, a clustering C is a partition $\{C_1, C_2, \dots, C_k\}$ of $\{T_1, T_2, \dots, T_n\}$ such that similar transactions are in the same cluster and dissimilar transactions are in different clusters, $C = C_1 \cup C_2 \cup \dots \cup C_k$, and $C_i \cap C_j = \emptyset, \forall i \neq j$. For example, the sample database listed in Table 2.1 may be partitioned into two clusters $\{C_1$={ {A, B, D}, {A, C, D}, {A, D, E} }, C_2= { { B, E, F}, { B, C, D, E, F} } according to the number of common items in the transactions.

In order to partition a database into clusters so that similar transactions are in the same clusters and the dissimilar transactions are in different clusters, a measure to determine the degree of similarity between a pair of transactions is needed. Currently, there are several similarity measures being used in clustering transactional databases as described in detail below.

A transaction T in a TDB over a set of d distinct items can be represented by a $d-$dimensional vector $T < i_1, i_2, \dots, i_d >$. When an item $I_k | k = 1,2, \dots, d$ appears in T, $i_k = 1$; otherwise $i_k = 0$.

Example 2.1 Suppose I ={A, B, C, D, E, F, G, H, J, K}, and TDB over I consists of T_1={A, B, C, D}, T_2= {C, D}, T_3={C, D, E, F, G, H}, T_4={E, F}, then the TDB can be represented by a set of vectors:

T_1<1, 1, 1, 1, 0, 0, 0, 0, 0, 0>

T_2<0, 0, 1, 1, 0, 0, 0, 0, 0, 0>

T_3<0, 0, 1, 1, 1, 1, 1, 1, 0, 0>

T_4<0, 0, 0, 0, 1, 1, 0, 0, 0, 0>

A conventional method for obtaining a distance measure between two transactions T_i and T_j is to use the 2x2 contingency table illustrated in Table 2.2. In the table, X_{11} represents the total number of attributes present in both transactions ("positive" matches) and X_{00} represents the total number of attributes absent from both transactions ("negative" matches). The total number of attributes present in T_i but not in T_j is denoted by X_{10} while the total number of attributes not present in T_i but in T_j is denoted by X_{01}. $X_{11} + X_{10} + X_{01} + X_{00} = d$, where d is the total number of distinct items in the database, i.e. the dimension of the database.

Table 2.2 2x2 Contengency Table

		T_j	
		1	0
T_i	1	X_{11}	X_{10}
	0	X_{01}	X_{00}

Following similarity measures have been proposed in the literature employing the quantities of X_{00}, X_{01}, X_{10} and X_{11} (Everitt, 1993; Huang, 1997a; Kantardzic, 2002).

- Simple Matching Coefficient (SMC)

  SMC is the ratio of the total number of positive and negative matches $X_{11} + X_{00}$ to the total number of attributes d. It can be expressed as follows:

$$S_{smc}(T_i, T_j) = (X_{11} + X_{00}) / d$$
$$= (|T_i \cap T_j| + |\overline{T_i} \cap \overline{T_j}|) / d \quad (2.1)$$

- Rao's Coefficient

  Rao's Coefficient is the ratio of the total number of positive matches X_{11} to the total number of attributes d which can be expressed as:

$$S_{rc}(T_i, T_j) = X_{11} / d$$
$$= |T_i \cap T_j| / d \quad (2.2)$$

- Jaccard Coefficient

  Jaccard coefficient is the ratio of the total number of positive matches to the total number of distinct attributes present in both transactions:

$$S_{jc}(T_i, T_j) = X_{11} / (X_{11} + X_{10} + X_{01})$$
$$= X_{11} / (d - X_{00})$$
$$= |T_i \cap T_j| / |T_i \cup T_j| \quad (2.3)$$

It can be easily proved that

$$S_{smc}(T_i, T_j) \geq S_{jc}(T_i, T_j) \geq S_{rc}(T_i, T_j) \quad (2.4)$$

Table 2.3 shows the similarity between transactions in ***Example 2.1*** based on these similarity measures.

Table 2.3 Similarity Between Transactions in *Example 2.1*

Pair of transactions	S_{smc}	S_{rc}	S_{jc}
T_1, T_2	80%	20%	50%
T_1, T_3	40%	20%	25%
T_1, T_4	40%	0%	0%
T_2, T_3	60%	20%	33.3%
T_2, T_4	60%	0%	0%
T_3, T_4	60%	20%	33.3%

To consider the suitability of these similarity measures for clustering typical real life high dimensional sparse transactional datasets, Table 2.4 lists the characteristics of two transactional databases that are widely used in the validation of data mining algorithms. BMSPOS (Zheng *et al.*, 2001) contains point-of-sale data from an electronics retailer and Retail (Brijs *et al.*, 1999) consists of the retail market basket data from an anonymous Belgian retail store. 99% of transactions in BMSPOS have less than 33 items. The minimum SMC similarity between a pair of those transactions is $(1657 - 2 * 32)/1657 = 96.13\%$ (when two transactions are of size 32 and are totally different) and the maximum value for Rao's similarity is $32/1657 = 1.93\%$ (when two transactions are of size 32 and are exactly the same). In the Retail dataset, 99% has transaction size less than 45 items, the minimum SMC similarity between a pair of transactions is $(16470 - 2 * 44)/16470 = 99.47\%$ (when two transactions are of size 44 and are totally different), and the maximum Rao's similarity is $44/16470 = 0.27\%$ (when two transactions are of size 44 and are exactly the same). Table 2.5 shows the maximum and minimum similarity values for 99% of records in each dataset according to different similarity measures.

Table 2.4 Two High Dimensional Sparse Datasets

Dataset	N	D	I_{avg}	F_{avg}	L_{max}	$L_{99\%}$
BMSPO	51559	1657	7.5	2032	164	<33
Retail	88162	16470	10.3	55	76	<45

Note: N - total number of transactions; d - dimensions of a dataset; I_{avg} - average numbers of items in a transaction; L_{max} - size of the longest transaction; F_{avg} - average occurrence of each item. $L_{99\%}$ - length of 99% of records in the dataset

Table 2.5 Range of Similarity Values According to Different Measures

Dataset	BMSPOS		Retail	
	Min	Max	Min	Max
S_{smc}	96.13%	100%	99.47%	100%
S_{rc}	0	1.93%	0	0.27%
S_{jc}	0	100%	0	100%

It is obvious that the discrimination power of simple matching coefficient and Rao's coefficient is very poor in dealing with such high dimensional sparse datasets. As a result, all the transactions will be in the same cluster according to SMC while every transaction will form a singleton cluster according to Rao's coefficient. Moreover, Rao's coefficient will assign the same similarity values for two pairs of transactions with the same number of common items but different number of non-common items. For example, for transactions in ***Example 2.1***, $S_{rc}(T_1, T_2) = S_{rc}(T_2, T_3) = 20\%$, although T_2 is more similar to T_1 than to T_3 with respect to the items present in the transactions.

As pointed out in (Everitt, 1993), no hard and fast rule can be given regarding the inclusion or otherwise of negative matches, since it is data and application dependent. For applications where the presence of items is of interest, such as customer segmentation, Jaccard coefficient is more suitable as it only compares items appearing in the pair of transactions. However, Jaccard coefficient still underestimates the similarity between two

transactions when one transaction is not a subset of the other. For example, for the pair of transactions T_1 and T_3 in ***Example 2.1***, $S_{jc}(T_3, T_1) = 2/8 = 1/4$ although $1/2$ of the items in T_1 also appear in T_3 and $1/3$ of items in T_3 also appear in T_1 . In a real life scenario, it is more meaningful to say that $1/2$ of the items in T_1 appear in T_3 and $1/3$ of items in T_3 appear in T_1 when two transactions are compared. Therefore, in this thesis, a new similarity measure that is more suitable for transactional data will be defined.

2.1.4 **Association Rules Mining**

Association rules mining was first introduced by R. Agrawal et al. in 1993 (Agrawal *et al.*, 1993). The relevant concepts and terms relating to it are described below.

Let $I = \{I_1, I_2, \dots, I_d\}$ be a set of d distinct items and TDB be a transactional database over I, $TDB = \{T_1, T_2, \dots, T_n\}$. A set of items is called an itemset, and an itemset with k items is called a k-itemset. The support p of an itemset X in TDB, is the proportion of the database that contains X, and $p = y/n$ where y is the number of occurrences of X in TDB. An itemset is called a frequent itemset if its support $p \geq p_t$ where p_t is the support threshold specified by the user. Otherwise, the itemset is not frequent.

An association rule is an expression of the form $X \rightarrow Y$, where non-empty itemsets $X \subseteq I$, $Y \subseteq I$ and $X \cap Y = \emptyset$. The support of the rule is the proportion of transactions that contains both X and Y, i.e the probability that both X and Y occur in a transaction. The confidence of the rule is the proportion of transactions that contain both X and Y to those that contain X i.e., the conditional probability that a transaction contains the itemset Y given that it contains the itemset X.

$$Support(X \rightarrow Y) = P(X \cup Y) \tag{2.5}$$

$$Confidence(X \rightarrow Y) = P(Y|X) = \frac{P(X \cup Y)}{P(X)} \tag{2.6}$$

An association rule with $Confidence > c_t$, where c_t is *confidence threshold* specified by the user, is considered as a valid association rule.

It is noted that $Confidence(X \rightarrow Y) \geq Support(X \rightarrow Y)$ holds. When support threshold p_t and confidence threshold c_t are chosen for association rules mining, c_t should be greater than or equal to p_t.

Association rules mining consists of two steps. All frequent itemsets, also called the *complete frequent itemsets* (CFI) are discovered in the first step and the rules based on the CFI are derived in the second step.

For the sample database listed in Table 2.1, let p_t= 50% and c_t = 50%, then frequent itemsets are {A}, {B}, {D}, {E} and {A,D}, and the valid association rules are

$A \rightarrow D$ with support = 60% and confidence = 100%;

$D \rightarrow A$ with support = 60% and confidence = 75% **.**

Frequent items are also called “large items” by some authors (Wang *et al.*, 1999).

For a d dimensional transactional database, the maximum number of frequent itemsets L_d and the maximum number of association rules R_d are as follows:

$$L_d = 2^d - 1 \tag{2.7}$$

$$R_d = 3^d - 2^{d+1} + 1 \tag{2.8}$$

Table 2.6 and Fig.2.2 show the maximum number of itemsets and rules for a database with d dimensions.

Table 2.6 Maximum Number of Rules and Itemsets Given *d* Items

d	3	5	7	9	10	20	100
R_d	12	180	1932	18660	57002	3.48×10^9	5.15×10^{47}
L_d	7	31	127	511	1023	1.05×10^6	1.26×10^{30}

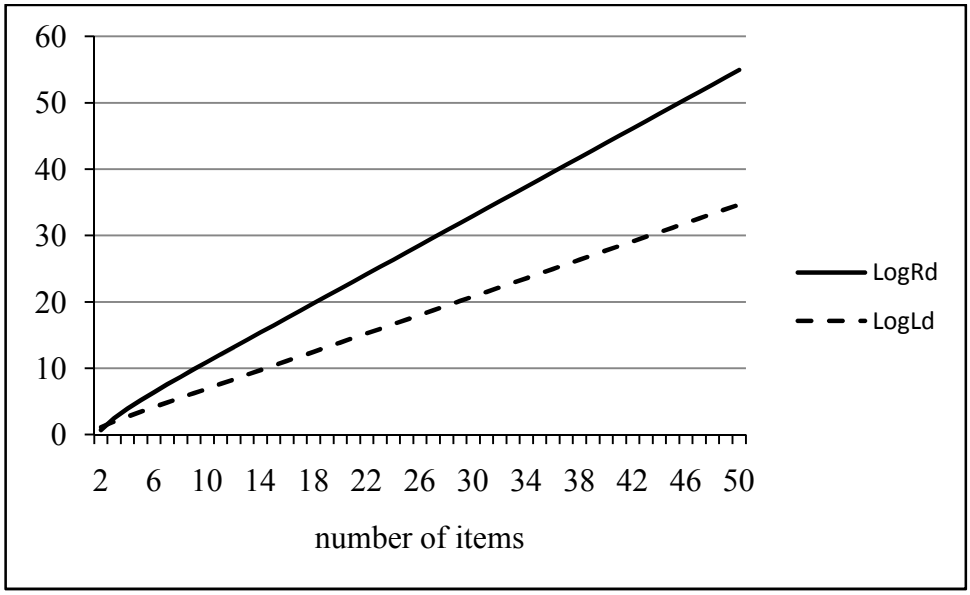


Figure 2.2 Maximum Number of Rules and Itemsets Given d Items

2.2 Algorithms and Frameworks for Clustering Transactional Data

In this section, transactional data clustering algorithms K-Mode, LargeItem and CLOPE are reviewed. CluStream, a framework for data stream clustering, is also reviewed.

2.2.1 K-Mode, Extension of K-Means for Transactional Data

K-Means (MacQueen, 1967) is a well known clustering algorithm which is suitable for clustering numeric datasets. It partitions a dataset into clusters by minimizing the within-cluster sum of squared distance between individual data points and their mean

$$\sum_{i=1}^{k} \sum_{x_j \in S_i} (x_j - \mu_i)^2 \tag{2.9}$$

where k is the number of clusters, x_j is a point in cluster S_i and μ_i is the mean of all the points in cluster S_i. The algorithm proceeds as follows:

1. Partition the data objects at random into k nonempty subsets.
2. Calculate the centroid (i.e.mean point) for each cluster of the current partition.
3. Assign each object to the cluster with the nearest centroid.
4. Repeat the last two steps until no object has changed clusters during a whole dataset scan.

The input for the K-Means clustering algorithm is a dataset, the desired number of clusters k. It will produce k clusters regardless of the underlying data structures. It is a structure imposing rather than a structure seeking algorithm.

K-Mode (Huang, 1997b) extends the K-Means paradigm to categorical domains. It replaces the mean of a cluster with mode, and uses a frequency based method to update modes in the clustering process to minimize the clustering cost function.

Let X, Y be two categorical objects described by m categorical attributes. The dissimilarity is defined as:

$$d(X,Y) = \sum_{j=1}^{m} \delta(x_j, y_j) \tag{2.10}$$

where

$$\delta(x_j, y_j) = \begin{cases} 0 & (x_j = y_j) \\ 1 & (x_j \neq y_j) \end{cases} \tag{2.11}$$

$d(X,Y)$ gives equal importance to each category of an attribute. When the frequencies of categories is taken into account, the dissimilarity between X and Y is defined as

$$d_{x^2}(X,Y) = \sum_{j=1}^{m} \frac{(n_{x_j} + n_{y_j})}{n_{x_j} n_{y_j}} \delta(x_j, y_j) \tag{2.12}$$

Where n_{x_j}, n_{y_j} are the numbers of objects in the dataset that have categories x_j and y_j for attribute j.

Let X be a set of categorical objects described by categorical attributes $A_1, A_2, \ldots, A_m$. A mode of X is a vector $Q =< q_1, q_2, \ldots, q_m >$ that minimizes

$$D(Q, X) = \sum_{i=1}^{n} d(X_i, Q) \tag{2.13}$$

where $X = \{X_1, X_2, \ldots, X_n\}$ and the distance function d can be either defined as Eq.(2.10) or Eq.(2.12). The mode $Q =< q_1, q_2, \ldots, q_m >$ of X is obtained by calculating the most frequent category in each attribute. In other words, q_1 is the most frequent category in attribute A_1, q_2 is the most frequent category in attribute A_2, and so on.

Since K-Mode is an extension of K-Means, it inherits the weakness of K-Means, i.e., it produces k clusters regardless the underlying cluster structure in the database. As for K-Means, the centroid (modes) of clusters are d dimensional vectors, where d is the dimension of the database. For a high dimensional database, the description of the clusters in terms of centroid will be very hard for users to comprehend.

2.2.2 LargeItem

Wang et al. proposed a large item based algorithm for clustering transactional data (Wang *et al.*, 1999). For a user-specified minimum support p_t, an item is large in cluster C_i if its occurrences are at least $p_t * |C_i|$; otherwise the item is small in C_i. Let $Large_i$ denote the set of large items and $Small_i$ denote the set of small items in C_i. Consider a clustering $C = \{C_1, C_2, \ldots, C_k\}$, the cost of C has two components: the intra-cluster cost $Intra(C)$ and inter-cluster cost $Inter(C)$.

The intra-cluster cost is charged for intra-cluster dissimilarity, measured by the total number of small items:

$$Intra(C) = |\bigcup_{i=1}^{k} Small_i| \tag{2.14}$$

This component will restrain the creation of loosely bound clusters that have too many small items.

The inter-cluster cost is charged for inter-cluster similarity. Since large items contribute to similarity in a cluster, each cluster should have as little overlapping of large items as possible. The overlapping of large items is defined by

$$Inter(C) = \sum_{i=1}^{k} |Large_i| - |\cup_{i=1}^{k} Large_i| \quad (2.15)$$

$Inter(C)$ measures the duplication of large items in different clusters. This component will restrain the creation of similar clusters.

The criterion function of a clustering C is defined as

$$Cost(C) = w * Intra(C) + Inter(C) \quad (2.16)$$

A weight $w > 1$ gives more emphasis to the intra-cluster similarity and a weight $w < 1$ gives more emphasis to the inter-cluster dissimilarity. By default $w = 1$.

Given a collection of transactions and a minimum support, transaction clustering is to find a clustering C such that $Cost(C)$ is minimum.

The algorithm LargeItem has two phases: the allocation phase and the refinement phase. In the allocation phase, each transaction is read in sequence and assigned to an existing cluster or a new cluster, whichever minimizes the $Cost(C)$. In the refinement phase, each transaction is read in sequence again, a transaction is moved to a new cluster or stays in the same cluster to minimize the cost. If no transaction is moved in one pass of transactions, the refinement phase terminates; otherwise, a new pass begins.

LargeItem is a structure seeking clustering algorithm. It produces a number of clusters based on the user's expectation of intra-cluster similarity. However, the input parameter w, which is used to control the weight of inter-cluster similarity and intra-cluster similarity in the cost function, does not have clear semantic meaning to the users and the range of this parameter is not defined. Therefore it is very difficult for users to choose a proper value for

it. The algorithm needs to scan the database at least twice, and therefore it is not so efficient for very large databases and is not suitable for data streams.

2.2.3 **CLOPE**

CLOPE (Yang *et al.*, 2002) is another structure seeking algorithm specifically designed for transactional data.

Given a cluster C, let $D(C)$ be the set of distinct items in C, $Occ(i, C)$ be the occurrence of item i in cluster C. The histogram of a cluster C is drawn with items as the X-axis and their occurrences as Y-axis, in decreasing order of occurrences.

The size $S(C)$ and width $W(C)$ of a cluster C are defined as

$$S(C) = \sum_{i \in D(C)} Occ(i, C) = \sum_{t_i \in C} |t_i| \tag{2.17}$$

where t_i is the i^{th} transaction in C.

$$W(C) = |D(C)| \tag{2.18}$$

The height of a cluster is defined as

$$H(C) = S(C)/W(C) \tag{2.19}$$

A larger height means a greater overlap among the items in a cluster, and thus more similarity among the transactions in the cluster.

Figure 2.3 shows the histogram for a cluster consisting of transactions $\{acd\}$, $\{de\}$ and $\{def\}$.

The criterion function for clustering is

$$Profit_r(C) = \frac{\sum_{i=1}^{k} \frac{S(C_i)}{W(C_i)^r} \times |C_i|}{\sum_{i=1}^{k} |C_i|} \tag{2.20}$$

In Eq. (2.20), r is a positive real number called repulsion, used to control the

level of intra-cluster similarity. When r is large, transactions within the same cluster must share a large portion of common items.

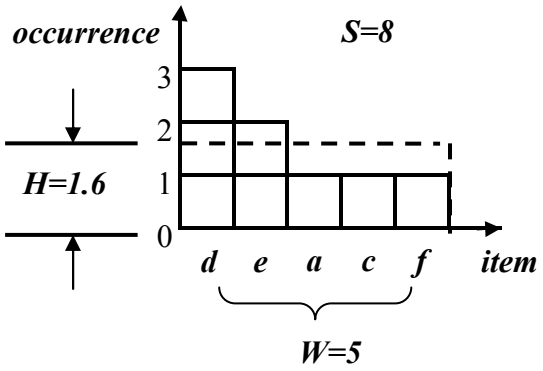


Figure 2.3 Histogram of Cluster {acd, de, def}

The object of clustering is to find a set of clusters that can maximize the profit.

The algorithm for CLOPE is very similar to that for LargeItem, except that the criterion function is different. It contains two phases: the initialization phase and the iteration phase. During the initialization phase, the database is scanned in sequence. Each transaction is allocated to an existing cluster or a new cluster so that profit can be maximized. In the iteration phase, the database is scanned repeatedly. A transaction is either moved from one cluster to another cluster or stays in the same cluster to maximize the profit. When no transaction is moved during a database scan, the iteration stops.

Like LargeItem, CLOPE is a structure seeking clustering algorithm. It produces clusters based on the user's expectations of intra-cluster similarity which is controlled by repulsion number r. However, r is not bounded and has no clear semantic meaning to the user. Therefore it is very difficult for users to choose a proper value for it.

2.3 CluStream: A Framework for Clustering Evolving Data Streams

The challenge of designing algorithms for data stream mining is three fold: (1) the algorithm is subject to sequential one-pass constraint over the data; (2) it must work under limited resources with respect to the unlimited data stream; (3) it should be able to reveal changes in the data stream over time.

Aggarwal et al. propose a framework, CluStream, for clustering evolving data streams(Aggarwal *et al.*, 2003). The clustering process is divided into an online micro-clustering component and an offline macro-clustering component. The online component periodically stores detailed summary statistics onto disk and the offline component uses the summary statistics in conjunction with other user input to answer time sensitive queries.

The separation of the data stream clustering approach into online and offline components raises these important questions:

What kind of summary information should be stored to provide sufficient temporal and spatial information for the offline clustering process w.r.t a particular application?

At what moments in time should the summary information be stored away on disk so that time sensitive queries can be answered with a desired level of approximation?

How can the periodic summary statistics be used to provide clustering and evolution insights over a user specified time horizon?

In order to address these issues, CluStream utilizes two concepts: micro-clusters and a pyramidal time frame.

Assume that the data stream consists of a set of multi-dimensional records $\overline{X_1}$ …$\overline{X_k}$…arriving at time stamps $T_1 \ldots T_k \ldots$. Each $\overline{X_\iota}$ is a multi-dimensional record containing d dimensions which is denoted by $\overline{X_\iota} = (x_i^1 \ldots x_i^d)$.

A **micro-cluster** for a set of d dimensional points $X_{i1} \ldots X_{in}$ with timestamps $T_{i1} \ldots T_{in}$ is defined as the $(2.d + 3)$ tuple ($\overline{CF2^x}$, $\overline{CF1^x}$, $\overline{CF2^t}$, $\overline{CF1^t}$, n), wherein $\overline{CF2^x}$ and $\overline{CF1^x}$ each correspond to a vector of d entries. The definition of each of these entries is as follows:

For each dimension, the sum of the squares of the data values is maintained in $\overline{CF2^x}$. Thus $\overline{CF2^x}$ contains d values. The p-th entry of $\overline{CF2^x}$ is equal to $\sum_{j=1}^{n} (x_{ij}^p)^2$.

For each dimension, the sum of the data values is maintained in $\overline{CF1^x}$. Thus $\overline{CF1^x}$ contains d values. The p-th entry of $\overline{CF1^x}$ is equal to $\sum_{j=1}^{n} x_{ij}^p$.

The sum of the squares of the time stamps $T_{i1} \ldots T_{in}$ is maintained in $\overline{CF2^t}$.

The sum of the time stamps $T_{i1} \ldots T_{in}$ is maintainded in $\overline{CF1^t}$.

The number of data points is maintained in n.

It is noted that the above definition of micro-clusters is a temporal extension of the cluster feature vector in (Zhang *et al.*, 1996).

The micro-clusters are also stored at particular moments in the stream which are referred to as *snapshots*. In the pyramidal time frame, the snapshots are stored at different levels of granularity depending on the recency. Snapshots are classified into different orders which can vary from 1 to $\log T$, where T is the clock time elapsed since the beginning of the stream. The snapshots of different orders are maintained as follows:

Snapshots of the i-th order occur at time intervals of α^i, where α is an integer, $\alpha \geq 1$. Specifically, each snapshot of the i-th order is taken at a moment in time when the clock value from the beginning of the stream is exactly divisible by α^i

At any given moment in time, only the last $\alpha + 1$ snapshots of order i are stored.

For a data stream, the maximum order of any snapshot stored at T time units since the beginning of the stream mining process is $\log_\alpha T$.

For a data stream, the maximum number of snapshots maintained at T time units since the beginning of the stream mining process is $(\alpha + 1)\log_\alpha T$. For any user specified time window of h, at least one stored snapshot can be found within $(2.h)$ units of the current time.

It can be seen that CluStream suits numeric data as it stores the sum of the squares of the data values and the sum of data values in the snapshots. In this thesis, the ideas of CluStream frame work are borrowed, i.e, dividing the clustering process to an online micro-clustering component and an offline macro-clustering component, but using different summary statistics that are suitable for clustering transactional data streams.

2.4 Sampling Techniques and Algorithms for Association Rules Mining

While many sound algorithms have been developed to provide accurate association rules in a database, such as Apriori (Agrawal and Srikant, 1994), FP-Growth (Han *et al.*, 2000), CT-Mine (Gopalan and Sucahyo, 2003) and CATS-Tree (Cheung and Zaïane, 2003), sampling techniques are actively studied as a tradeoff of accuracy with efficiency when dealing with very large databases or data streams. This section will review sampling techniques for association rules mining, including two-phase sampling, progressive sampling and random sampling based on the Chernoff Bounds.

2.4.1 Two-Phase Sampling

FAST (Chen *et al.*, 2002) is a two-phase sampling algorithm for mining association rules in large databases. In Phase I, a large initial sample of transactions S is collected and used to quickly estimate the support of each individual item in the database. In Phase II, these estimated supports are used to either trim outliers or select representatives from the initial sample, resulting in a small final sample S_0 that can more accurately reflect the

support of items in the entire database as explained below. Any standard association rules mining algorithm can then be used to discover association rules in the final sample.

Since the supports of 1-itemsets in the original database are unknown, they are estimated by the corresponding supports in the original larger sample S. The discrepancy is measured by the distance function

$$Dist(S_0, S) = \frac{|L(S) - L(S_0)| - |L(S_0) - L(S)|}{|L(S_0)| + |L(S)|} \quad (2.21)$$

where $L(S)$ and $L(S_0)$ denote the set of of frequent 1-itemsets in S and S_0 , respectively.

Two different approaches are presented for obtaining the final sample in Phase II: “trimming” and “growing”. The trimming procedure starts with the initial sample S and continuously removes “outliers” until a specified stopping criterion is met. An outlier is a transaction whose removal from the sample maximally reduces the discrepancy between the supports of the 1-itemsets in the sample and the corresponding supports in the original database. In contrast, the growing procedure selects representative transactions from the initial sample and adds it to an initially empty dataset S_0. In either approach, by forcing the supports of a 1-itemsets in the sample to approximate those in the original database, the Phase II procedure helps ensure that the support of every 1-itemset in the sample is close to that in the database.

FAST-trim and FAST-grow are algorithms resulting from the trimming and growing approaches, respectively.

Given the minimum support and confidence thresholds, the FAST-trim algorithm proceeds as follows:

1. Obtain a simple random sample S from the database.
2. Compute support for each 1-itemset in S.
3. Using the support computed in step 2, obtain a reduced sample S_0 from S by trimming away outlier transactions.

4. Run standard association rules mining algorithm against S_0 for the given minimum support and confidence thresholds.

FAST-grow algorithm has an input parameter $k \in \{1, 2, \ldots, |S|)$ and n, where n is the final sample size. Like FAST-trim, it proceeds in stages. Initially, S_0 is empty. At each stage representative transactions are added to S_0. In order to identify representative transactions, the transactions in $S - S_0$ are divided into disjoint groups, with each group has $\min(|S - S_0|, k)$ transactions. For each group, a transaction T that minimizes $\mathrm{Dist}(S_0 \cup \{T\}, S)$ over all transactions in the group is added to S_0. The algorithm proceeds until $|S_0| = n$.

It can be seen that choosing the right sample size for the initial sample is critical for the success of the two-phase sampling. A wrong initial sample size will result in the failure of the sampling since the final sample is a subset of the original sample. Since there is no theory to back up the selection of the sample size, the process for choosing the initial sample size becomes arbitrary.

2.4.2 **Progressive Sampling**

Progressive sampling (Provost *et al.*, 1999) starts with a small sample and uses progressively larger ones until the model accuracy no longer improves. The learning curve in Fig. 2.4 depicts the relationship between sample size and model accuracy. The horizontal axis represents n, the number of objects in a given training sets, that can vary from zero to N, the total number of available instances. Most learning curves typically have steeply sloping portion early in the curve, and a plateau late in the curve. When the curve reaches its final plateau, it is said to have converged. The training set size at which convergence occurs is denoted as n_{min}, where n_{min} is the size of the smallest sufficient sample size for an induction algorithm.

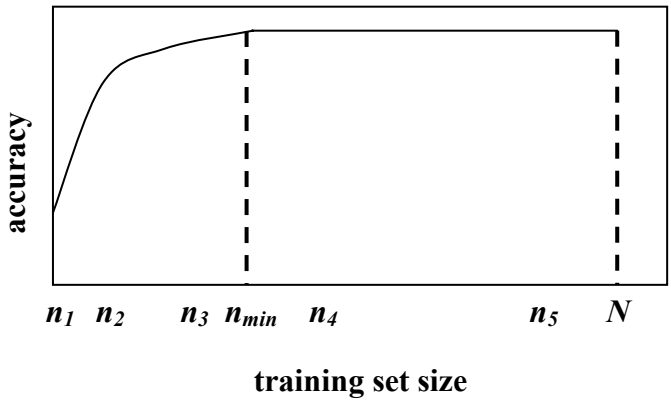


Figure 2.4 Learning Curves and Progressive Samples

A central component of progressive sampling is a sampling schedule $S = \{n_0, n_1, n_2, \ldots, n_k\}$ where each n_i is an integer that specifies the size of a sample to be provided to an induction algorithm. For $i < j$, $n_i < n_j$. If the dataset contains N instances in total, $n_i \leq$ N for all i.

Figure 2.5 is a generic algorithm that defines the family of progressive sampling methods.

```
/* generic progressive sampling algorithm */
compute schedule S = {n_0, n_1, n_2, ... , n_k} of sample sizes
n ← n_0
M ← model induced from n instances
while not converged do
        recompute S if necessary
        n ← next element of S larger than n
        M ← model induced from n instances
endwhile
return M
```

Figure 2.5 Generic Progressive Sampling Algorithm

It is obvious that sampling schedule plays a very important role in progressive sampling as it will determine the number of samples processed before a final sample is selected.

John and Langely (John and Langley, 1996) defines an arithmetic sampling using the schedule

$$S_a = n_0 + (i.n_\delta) = \{n_0,\ n_0 + n_\delta,\ n_0 + 2n_\delta, \dots,\ n_0 + kn_\delta\} \qquad (2.22)$$

For example, when $n_0 = 100$, and $n_\delta = 100$, the schedule will be $\{100, 200, 300, \dots\}$.

Provost et al (Provost *et al.*, 1999) propose Geometric sampling using the schedule:

$$S_g = a^i.n_0 = \{n_0,\ a.n_0,\ a^2.n_0,\ \dots,\ a^k.n_0\} \qquad (2.23)$$

for some constant n_0 and a. An example schedule is $\{100, 200, 400, 800 \dots\}$ when $n_0 = 100$ and $a = 2$.

Parthasarathy (Parthasarathy, 2002) adopts the progressive sampling approach for association rules mining. In the context of association rules mining, the model accuracy is defined as

$$Sim(d_1, d_2) = \frac{\sum_{x \in A \cap B} \max\{0, 1 - \alpha\left|sup_{d_1}(x) - sup_{d_2}(x)\right|\}}{\|A \cup B\|} \qquad (2.24)$$

where d_1 and d_2 are two database samples, A and B are respectively the set of frequent itemsets for d_1 and d_2, x is an element in frequent itemsets A and B . values for Sim are bounded and lie in [0,1] .

As mentioned above, how to choose a sampling schedule remains a question for progressive sampling for association rules. How big the initial sample should be, and how the next sample size is calculated are two questions for which there are no theoretically sound answers. The selection of the initial sample size and the incremental method are very much subjective and experimental in nature. Since the frequent itemsets calculations are expensive, progressive sampling that involves several

samples may defeat the purpose of sampling as frequent itemsets computations have to be performed on each sample.

2.4.3 Sampling Based on Chernoff Bounds

Toivonen presents a sampling technique for association rules in (Toivonen, 1996) based on Chernoff Bounds. The idea is to pick a random sample and use the sample to find all association rules that probably hold in the whole database, and then verify the results with the rest of the database.

The sample size n is determined based on Chernoff Bounds (Alan and Spencer, 1992) for a given error bound e and the maximum probability δ for an error exceeding the error bound:

$$n \geq \frac{1}{2e^2} \ln \frac{1}{\delta} \tag{2.25}$$

In order to uncover all frequent itemsets in the sample, the support threshold is lowed to

$$p_l < p_t - \sqrt{\frac{1}{2n} \ln \frac{1}{\delta}} \tag{2.26}$$

The main steps of the algorithm are as follows.

1. Draw a random sample S of size n from the database;
2. Calculate frequent itemsets in S using the lowered support threshold;
3. Scan the database. If an itemset is frequent in the sample according to lowered support threshold, and is also frequent in the database based on the support threshold, then output the itemset.

The algorithm needs to scan the original database to verify frequent itemsets discovered in the sample. It is not an algorithm purely working on samples to get estimated results. The sample size based on the Chernoff Bounds is also very conservative (Zaki *et al.*, 1997). In this thesis, sample sizes are derived

based on the binomial distribution and the central limit theorem, which is much smaller than that based on Chernoff Bounds, yet still provides the same approximation guarantees.

2.5 Summary

In this chapter, the definitions of terms relating to transactional database, data streams, association rules and clustering are provided. The previous research in clustering and association rules mining that are closely related to this research are reviewed, including the existing similarity measures for transactional data, the time window models for data stream mining, algorithms for transactional database clustering, and sampling techniques for association rules mining.

Chapter 3

Clustering Transactional Data

Transactional data are often characterized by high cardinality, high dimensionality and high sparsity. Traditional centroid-based iterative structure imposing clustering algorithms are not efficient in dealing with very large databases as they need to scan the databases more than once. High dimensionality of the transactional database also makes it hard or even impossible to comprehend the description of a cluster by a centroid-based algorithm because the centroid is expressed by a d dimensional vector, where d is the dimension of the database.

This thesis aims to develop more effective, efficient and scalable algorithms for transactional data clustering, and to provide more meaningful descriptions of clusters. The algorithms will seek the natural cluster structures in the database rather than impose a cluster structure on the data.

Fewer algorithms for transactional database clustering have been proposed in the literature compared with that for numeric data. In this chapter, an incremental structure seeking clustering algorithm is proposed for clustering very large high dimensional sparse transactional databases. The extensive testing results show that the algorithm is effective, efficient, scalable and order insensitive. The descriptions of clusters are expressed in terms of locally hot items which are easy to comprehend for end users. It seeks clusters based on the user's expectations of cluster features. The number of clusters produced varies as the user's expectations change.

This chapter is organized as follows. Section 3.1 states the problem to be solved and Section 3.2 defines a new similarity measure for transactional

data. Section 3.3 presents the principles and models for the problem and Section 3.4 proposes an incremental clustering algorithm INCLUS based on the new similarity measure and induction principles defined in the previous sections. Section 3.5 analyzes the complexity of INCLUS while Section 3.6 empirically evaluates INCLUS in terms of effectiveness, order dependency and scalability. The advantages of newly defined similarity measure are also tested in Section 3.6. Section 3.7 summarizes the chapter.

3.1 Problem Definition

The problem in this study is as follows: Given a very large high dimensional sparse transactional database TDB, find a partition $P = \{C_1, C_2, \dots, C_k\}$ of TDB where $(C_i \cap C_j = \emptyset, \forall i \neq j)$ and $\cup_{i=1}^{k} C_i = TDB$ such that similar transactions are in the same cluster, dissimilar transactions are in different clusters, as well as provide a meaningful description of the clusters.

3.2 A New Similarity Measure

As discussed in Chapter 2, Rao's Coefficient and Simple Matching Coefficient take into account negative matches(i.e. items missing in both transactions) when comparing transactions. For a high dimensional sparse database, as pointed out in Chapter 2, these measures will lose their discrimination power. Jaccard Coefficient does not take into account negative matches, but it still underestimates the similarity between two transactions when one transaction is not a subset of the other. For example, for the pair of transactions T_1 and T_3 in ***Example 2.1***, $S_{jc}(T_1, T_3) = 1/4$, although $1/2$ of the items in T_1 also appear in T_3 and $1/3$ of items appearing in T_3 also appears in T_1. In a real life scenario, it is more meaningful to say that $1/2$ of the items in T_1 appear in T_3 and $1/3$ of items in T_3 appear in T_1 when the two transactions are compared. Based on this perception, a new similarity measure for transactional databases is defined as given below.

Definition 3.1 (*Similarity*) Let T_i and T_j be two transactions in a *TDB*. The similarity between T_i and T_j is defined as

$$S(T_i,\ T_j) = \frac{|T_i \cap T_j|}{\max(|T_i|,\ |T_j|)} \tag{3.1}$$

It can also be expressed as :

$$S(T_i,\ T_j) = \max(\frac{|T_i \cap T_j|}{|T_i|},\ \frac{|T_i \cap T_j|}{|T_j|}\ ) \tag{3.2}$$

S has the following properties:

$$0 \leq S(T_i, T_j) < 1, \forall T_i \neq T_j$$

$$S(T_i, T_j) = 1, \forall T_i = T_j$$

$$S(T_i, T_j) = S(T_j, T_i)$$

This definition ignores the negative matches and is more suitable for an application where only the items present in the dataset are of interest, such as finding customers with similar purchasing patterns in a supermarket basket data. S indicates that at least S proportion of items in one transaction is present in the other. For example, if one customer bought {bread, milk, pen, eraser} and the other bought {bread, milk, lettuce, carrot} from a super market, S will be 50%. This figure is the same as what we will infer in our daily life, i.e. 50% of the items bought by the two customers are the same. On the other hand, if one customer bought {bread, milk, pen, eraser} and the other bought {bread, milk, lettuce, carrot, apple}, S will be 40%, i.e. at least 40% items in one basket are the same as in the other basket.

The relationship between this newly defined similarity measure and Jacard Coefficient is as follows: if one transaction is a subset of the other, then $S = S_{jc}$, otherwise $S > S_{jc}$.

3.3 Cluster Representatives and Induction Principle

3.3.1 Cluster Representatives

As stated in Section 3.1, clustering a transactional dataset is to group similar transactions together. Since similar transactions must share some common items, those items are said to be hot in a cluster, i.e. they appear more frequently in the cluster than the other items. Hence it is meaningful to use hot items as cluster representative to describe clusters.

Definition 3.2 (*Hot items*) Let C be a cluster, I be a set of distinct items in C, and $\gamma_t \in [0, 1]$. Then the hot items in cluster C are defined as

$$hot(C) = \{i \in I | freq(i, C)/|C| \geq \gamma_t\} \tag{3.3}$$

where $freq(i, C)$ is frequency of item i in C, i.e., the total number of occurrences of item i in C, and γ_t is the *support threshold* above which an item is considered hot.

Hot items are those items that appear in at least γ_t percent of transactions in a cluster.

For the sample database listed in ***Example 2.1***, if $\gamma_t = 50\%$, then $\mathrm{C, D, E, F}$ are hot items, if $\gamma_t = 60\%$, then $\mathrm{E, F}$ are not hot anymore while $\mathrm{C, D}$ are still hot items.

The hot items of a cluster will be used as its cluster representative, also denoted as rep. It can be easily seen that rep may or may not be a transaction in a cluster. The cluster features will be described by $(rep, |C|)$ with frequencies for each item in rep attached.

$T_1 \cap T_2 \neq \emptyset$ and $T_1 \cap T_3 \neq \emptyset$ does not mean $T_2 \cap T_3 \neq \emptyset$. For example, in ***Example 2.1***, $T_2 \cap T_3 = \{\mathrm{C, D}\} \neq \emptyset$, $T_3 \cap T_4 = \{\mathrm{E, F}\} \neq \emptyset$ but $T_2 \cap T_4 = \emptyset$. Therefore the pure pairwise similarity approach is no longer suitable for transactional data (Wang *et al.*, 1999). The choice of cluster representatives in this thesis overcomes the shortcomings of the pure pairwise similarity

approach to some extent. It ensures that if a transaction is assigned to a cluster, it will have some items in common with at least $\gamma_t * |C_i|$ transactions in that cluster. In other words, when $T_1 \cap T_2 \neq \emptyset$ and $T_1 \cap T_3 \neq \emptyset$ hold, $T_2 \cap T_3 \neq \emptyset$ is very likely to hold.

3.3.2 Induction Principle

As given in (Liu, 1968), for a dataset with n transactions, the number of distinct ways of partitioning n transactions into k non-empty clusters is given by

$$N(n,k) = \frac{1}{k!}\sum_{i=0}^{k}(-1)^{k-i}\binom{k}{i}i^n \tag{3.4}$$

Therefore the total number of different ways to partition a dataset is:

$$N = \sum_{k=1}^{n}\frac{1}{k!}\sum_{i=0}^{k}(-1)^{k-i}\binom{k}{i}i^n \tag{3.5}$$

The goal of clustering is to choose the best partitioning with respect to a given clustering criterion. Even with today's computers, the complete enumeration of every possible partitioning is simply not possible for a large value of n (Liu, 1968). Consequently, optimization approaches are adopted for clustering analysis with induction principles applied to resolve optimization problems. For example, the induction principle of K-Means algorithm is to "*pick the model (set of k representatives) that minimizes the total squared error"* (Estivill-Castro*,* 2002*)*. The mathematical formulation for this clustering criterion is

$$minimize\ L_2(C) = \sum_{i=1}^{n} Euclid^2\ (\overrightarrow{x_i}, rep[\overrightarrow{x_i}, C]) \tag{3.6}$$

where $Euclid(\vec{x},\vec{y}) = \sum_{j=1}^{d}[|x_j - y_j|^{1/2}]^{1/2}$ is the Euclidean metric; $C = \{\overrightarrow{c_1}, \overrightarrow{c_2}, \dots, \overrightarrow{c_k}\}$ is the set of k centres; and for $i = 1,2, \dots, n,$ the point $rep[\overrightarrow{x_i}, C]$ is the closest point in C to $\overrightarrow{x_i}$. Equation (3.6) expresses the search for a set C of

k representatives, where the partition into clusters is defined by assigning each $\overrightarrow{x_i}$ to its representative $rep[\overrightarrow{x_i}, C]$.

In this thesis, the model (set of k representatives) that maximizes the total intra-cluster similarity and minimizes the inter-cluster similarity will be picked based on the similarity measure proposed in **Definition 3.1**. The mathematical formulation for this criterion is

$$maximize\ S(P) = \sum_{k=1}^{K} \sum_{i=1}^{n_k} S(T_{ik}, rep(C_k)\ ) \tag{3.7}$$

where $S(P)$ is the total similarity of a partitioning, k is the number of clusters in the transactional database, n_k is the number of transactions in cluster C_k and T_{ik} is the i-th transaction in C_k. $rep(C_k)$ is the representative of C_k, i.e, the set of hot items in C_k, it may or may not be a transaction in the transactional database.

3.4 An Incremental Clustering Algorithm

The incremental clustering approach is popular in dealing with very large datasets where the cost of multiple scans of a disk resident dataset is too expensive and the entire dataset cannot be stored in the main memory because of its size (Kantardzic, 2002). In this Section, an incremental clustering algorithm (INCLUS) is proposed for very large transactional database based on the new similarity measure and induction principle described in the previous sections. The sketch of the algorithm is given in Figure 3.1.

The algorithm has two input parameters, γ_t and S_t . γ_t controls how frequent an item should appear in a cluster to be the cluster representative while S_t is the similarity threshold which controls closeness of transactions in a cluster. S_t reflects the users' expectations on how close the transactions in a cluster should be. For example, $S_t = 50\%$ indicates that at least 50% of items in two transactions should be the same to be considered as similar.

Algorithm 3.1: INCLUS

Input: dataset TDB, γ_t , S_t

Ouput: cluster features of each cluster

1. create a cluster with the first transaction
2. **while** not end of the file **do**
3. read the next transaction T
4. allocate T to an existing cluster or a new cluster to maximize $S(P)$
5. update rep for the most recently modified cluster
6. **endwhile**
6. output cluster features

Figure 3.1 High Level Description of INCLUS

The algorithm creates a cluster for the first transaction (step 1). For the remaining transactions (step 2), each transaction T will be compared with representatives of existing clusters and assigned to an existing or new cluster to maximize $S(P)$ (step 4). If $S\big(T, rep(C)\big) < S_t$ for all existing clusters, then a new cluster is created for T. If C is the only cluster where $S\big(T, rep(C)\big) \geq S_t$ and is maximum, then T will be assigned to cluster C; otherwise T will be added to the one with the least number of transactions in order to balance the sizes of clusters. Once a transaction is assigned to a new cluster or an existing cluster, the representative of the cluster will be updated(step 5).

A histogram is kept for each cluster in the main memory. When a new transaction is added or deleted, the histogram is updated. The hot items are those items in the histogram whose frequency is greater than or equal to $\gamma_t * |C|$ and can be easily computed from the histogram.

3.5 Complexity of INCLUS

As a histogram is kept for each cluster in the main memory, and so the space usage is $O(k * d)$ in the worst case, where k is the number of clusters and d is the dimension of a TDB. The space requirement is very small since only the histogram is kept for a cluster. Since the algorithm is non-iterative, the processing time is $O(n)$, where n is the total number of transactions in a TDB.

3.6 Evaluation of INCLUS

In this section, the effectiveness of INCLUS is evaluated and compared with LargeItem (Wang *et al.*, 1999) and CLOPE (Yang *et al.*, 2002). CLOPE is provided by its authors and LargeItem is implemented by the author of this thesis based on the algorithm described in (Wang *et al.*, 1999). LargeItem and CLOPE are chosen for comparison because both of them are designed for transactional databases and based on the same philosophy as INCLUS, i.e. to seek clusters according to the user's expectation on the closeness of transactions in a cluster rather than to force the algorithm to find a certain number of clusters as for K-Means and its variants.

The experiments are performed on the labeled congressional Vote and Mushroom datasets (Blake and Merz, 1998) to evaluate the effectiveness and order-dependence properties of the proposed algorithm. These labeled datasets are widely used in the literature, such as (Wang *et al.*, 1999; Guha *et al.*, 2000b; Yang *et al.*, 2002; Wang and Karypis, 2004), for evaluation purposes.

Vote dataset is the record of 1984 United States Congressional Votes. It has 435 records, 168 for Republicans and 267 for Democrats. Each record contains the voter's affiliation (Republican or Democrat) and the answers of 'y'(yes) or 'n'(no) to 16 issues. In other words, Vote is a labeled categorical dataset with 16 categories and each category has two values 'y' or 'n'. The 249th record is deleted before clustering since all its values are missing.

Mushroom is a categorical dataset with 22 categories and 116 values in total. It contains 8,124 records with class label ‘e’ (for edible) or ‘p’(for poisonous) for each record. 4,208 edible mushrooms and 3,916 poisonous mushrooms are recorded in the dataset.

Vote and Mushroom are converted to transactional datasets using the method mentioned in (Han and Kamber, 2000): treating each value of a category as an attribute of a transaction. Therefore, Vote is converted to a transactional dataset with 32 attributes while Mushroom is converted to a transactional dataset with 116 attributes. All the missing values are ignored. The class labels of these datasets are not used in clustering but used for evaluating the effectiveness of clustering algorithms.

The real life unlabelled transactional dataset BMSPOS is used to test the scalability of the proposed algorithm.

Meanings of symbols used in the tables and figures in this section are:

$|C|_{min}$, $|C|_{max}$ - the cardinalities of the smallest and the biggest clusters, respectively.

γ_t - support threshold,

S_t - similarity threshold,

k - number of clusters.

3.6.1 **Effectiveness of INCLUS**

Effectiveness of the proposed algorithm is evaluated in terms of *impurity* (defined below).

Transactions in each cluster may belong to different classes if the cluster is not pure. The dominant class of a cluster is the class of the majority transactions. For instance, if a cluster contains 100 types of mushrooms, where 95% of them are edible while the rest are poison ones, then “edible” is the dominant class of the cluster. The number of transactions of a dominant

class in cluster C_i is denoted as M_i. *Purity* is defined as $\sum_{i=1}^{k} M_i$ in (Yang *et al.*, 2002) to evaluate the quality of clustering. In this thesis, *impurity* e is defined as follows to measure the quality of clustering:

$$e = 1 - \sum_{i=1}^{k} \frac{M_i}{n} \tag{3.8}$$

where n is the total number of transactions in a dataset, $n > k \geq 2$ and $e \in [0, 1)$. In the context of supervised classification, e is the proportion of transactions being misclassified.

The impurity cannot be defined when $k = 1$ because when all the transactions are in the same cluster, it is impossible and inappropriate to determine if a given transaction is misclassified or not. In the best case, all the transactions in a cluster belong to the same class and $e = 0$.

In this section, the effectiveness of INCLUS will be tested and compared with CLOPE and LargeItem.

For a fair comparison, the same methodology is used as in (Yang *et al.*, 2002), i.e. different values for input parameters are tried for each algorithm so that the same or very similar number of clusters are obtained by all the algorithms. For the Vote dataset, when $\gamma_t = 60\%$ and $S_t = 30\%$ for INCLUS, $\gamma_t = 60\%$ and $w = 1$ for LargeItem and $r = 1.5$ for CLOPE, all these algorithms obtain two big clusters which contain more than 99.8% of the total number of records. Table 3.1 illustrates the clustering results by these algorithms. The impurities are 12.9%, 20.3% and 16.8% for INCLUS, LargeItem and CLOPE, respectively. It shows that INCLUS produces better quality clusters for Vote than other algorithms.

It is noticed that the 108th and the 184th voting records are very different from the others: they only vote one 'y' for all the 16 issues while about 90% of members vote 'y' for 6 to 10 issues. The two distinguished (abnormal) votes are picked up by INCLUS and are assigned to two separate singleton clusters. CLOPE and LargeItem did not pick them up (The singleton cluster produced by CLOPE consists the 341th record).

Table 3.1 Testing Results for Vote

	Cluster ID	Democrat	Republican	*e*(%)
INCLUS	1	49	159	12.9
	2	217	7	
	3	0	1	
	4	1	0	
LargeItem	1	87	166	20.3
	2	180	1	
CLOPE	1	71	165	16.8
	2	195	2	
	3	1	0	

The cluster features gives another view of the quality of clustering. Table 3.2 illustrates the frequent items in two big clusters produced by INCLUS in Table 3.1. It shows the clustering qualities from another angle. The number following Y (or N) denotes the number of votes with 'y' (or 'n') for a particular issue. For example, the second line of the table tells that 216 out of 224 members in cluster 1 vote 'y' to "aid-to-Nicaraguan-contras" while 172 out of 208 members in cluster 2 vote 'n' for the issue. It can be seen that the majority of members in the two clusters have opposite points of view on 11 issues, such as handicapped-infants, physician-fee-freeze, etc. Thus the two clusters are well separated. Frequent items in two big clusters obtained by CLOPE and LargeItem are the same as that by INCLUS, but the number of transactions in each cluster and the frequency of each item are different.

As for Vote, different parameters were tried for the three algorithms to get as similar a number of clusters as possible for the Mushroom dataset for comparison purposes. Table 3.3 presents the results for a set of parameters using which a similar number of clusters are produced by the three algorithms. Table 3.4 is the results for another set of parameters. The number of clusters, the cardinality of the biggest and the smallest clusters, and the impurity using each of the three algorithms are shown in those tables.

It can be seen from the above results that INCLUS is effective in clustering transactional data though it takes only one pass over the dataset while CLOPE and LargeItem iteratively scan the dataset many times.

Table 3.2 Cluster Features for Vote by INCLUS

Cluster ID	C_1	C_2
Cardinality	224	208
aid-to-nicaraguan-contras	Y(216)	N(172)
physician-fee-freeze	N(212)	Y(169)
adoption-of-the-budget-Resolution	Y(208)	N(158)
el-salvador-aid	N(202)	Y(197)
anti-satellite-test-ban	Y(200)	N(163)
education-spending	N(190)	Y(152)
x-missile	Y(184)	N(181)
superfund-right-to-sue	N(177)	Y(174)
crime	N(162)	Y(194)
export-administration-act-south-africa	Y(157)	
duty-free-exports	Y(154)	N(176)
handicapped-infants	Y(143)	N(161)
religious-groups-in-schools	N(139)	Y(194)
synfuels-corporation-cutback		N(156)

Table 3.3 Testing Results for Mushroom with One Set of Parameters

Features	INCLUS $(\gamma_t = 60\%, S_t = 45\%)$	LargeItem $(\gamma_t = 60\%, w = 4)$	CLOPE $(r = 1.2)$
k	11	10	10
$\|C\|_{\min}$	8	53	24
$\|C\|_{\max}$	1828	3359	2563
$e(\%)$	4.0	12.2	9.0

Table 3.4 Testing Results for Mushroom with other Set of Parameters

Features	INCLUS $(\gamma_t = 60\%, S_t = 30\%)$	LargeItem $(\gamma_t = 60\%, w = 1)$	CLOPE $(r = 1.5)$
k	25	25	27
$\|C\|_{\min}$	8	8	1
$\|C\|_{\max}$	1558	1728	1726
$e(\%)$	0.7	4.8	0.4

The test results listed in Tables 3.3 and 3.4 also show that INCLUS is structure seeking rather than structure imposing. It produces a certain number of clusters based on the users' expectations of closeness of transactions in a cluster which is controlled by the corresponding input parameters.

3.6.2 **Order-dependence Property of INCLUS**

Order-dependence is an important property of incremental clustering algorithms. An algorithm is order independent if it generates the same partitioning for any order of transactions in a dataset. Most of the incremental algorithms are order-dependent (Jain *et al.*, 1999). Here, the order-dependence property of INCLUS is to be tested and compared with that of LargeItem and CLOPE by rearranging the order of records in the tested datasets. Vote-Random is the data file obtained by randomly shuffling the records in Vote. Vote-Sorted is the sorted file using the Unix sort facility. In Vote-Sorted, the first 267 records are for Democrats while the rest are for Republicans. Vote_Sorted should be very powerful data order rearrangement for order-dependence testing. Tables 3.5 and 3.6 illustrate the test results on Vote_Random and Vote_Sorted, respectively. The input parameters for each algorithm are the same as those for Vote. By comparing Tables 3.1, 3.5 and 3.6, it can be seen that INCLUS is not sensitive to the order of transactions. Tests on several other datasets obtained by shuffling Vote led to the same conclusions.

Order-dependence test was also performed on the Mushroom dataset. The figures for the original Mushroom dataset are used as the base for comparison. Transactions in Mushroom are shuffled to get 4 datasets, named as Rd_1 to Rd_4. Another dataset Sorted is obtained using Unix sort facilities. Thus the first block of transactions in Sorted is for edible mushrooms while the rest is for poisonous ones. Using the same input parameters as that in Table 3.3, INCLUS got the results as shown in Figures 3.2 and 3.3.

While the order of transactions changes, the biggest changes of k for INCLUS, LargeItem and CLOPE, are 4, 10 and 40, respectively. Similarly, the biggest change in impurity is 4.48% for INCLUS while that for LargeItem and CLOPE are 15% and 33.6%, respectively. The biggest changes of $|C|_{min}$ are 41, 49 and 40 and the biggest changes of $|C|_{max}$ are 158, 2732 and 2256, respectively.

Table 3.5 Testing Results for Vote_Random

	Cluster ID	Democrat	Republican	*e*(%)
INCLUS	1	49	159	
	2	217	7	12.9
	3	0	1	
	4	1	0	
LargeItem	1	60	157	16.1
	2	207	10	
CLOPE	1	264	167	
	2	1	0	38.5
	3	1	0	
	4	1	0	

Table 3.6 Testing Results for Vote_Sorted

	Cluster ID	Democrat	Republican	*e*(%)
INCLUS	1	206	8	
	2	0	150	
	3	60	8	3.7
	4	1	0	
	5	0	1	
LargeItem	1	57	160	14.8
	2	210	7	
CLOPE	1	63	77	
	2	48	55	
	3	70	1	30.6
	4	85	20	
	5	1	10	

To sum up, as the order of transactions are changed, the quality of clustering by INCLUS is better sustained than the other algorithms with respect to the number of clusters, the size of clusters and impurity.

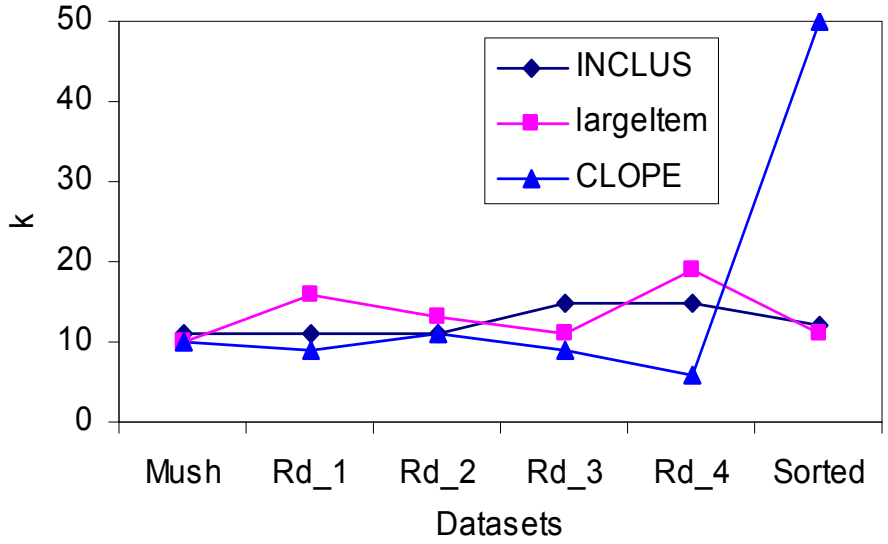


Figure 3.2 Changes of Number of Clusters while the Order of Transactions Changes

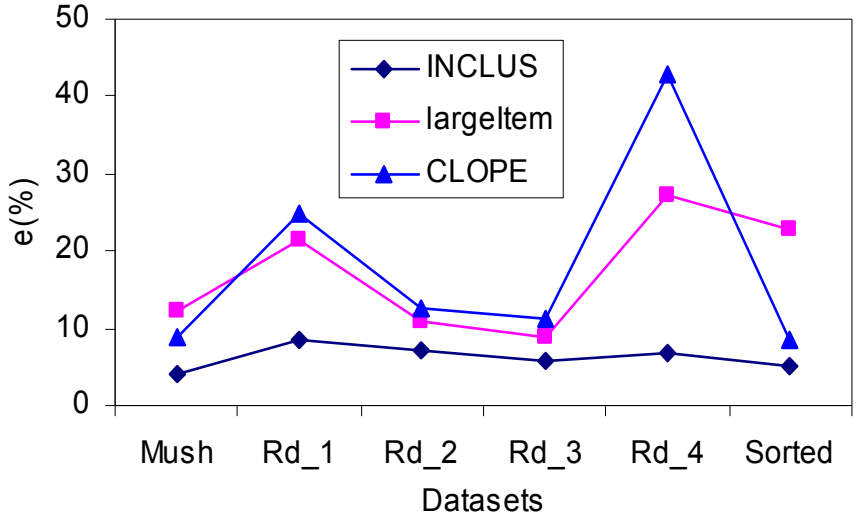


Figure 3.3 Changes of Errors while the Order of Transactions Changes

3.6.3 **Scalability of INCLUS**

BMSPOS is a high dimensional sparse real life dataset. INCLUS obtained 1906 clusters when $\gamma_t = 30\%$ and $S_t = 20\%$. The biggest cluster has 2728 transactions while the smallest one has only 1 transaction. Cluster features of the five largest clusters are shown in Table 3.7. It can be seen that common items are shared among clusters. It is consistent with the real life scenario: some items are very popular regardless of which group a customer belongs to. For example, in a grocery store, milk and bread are such items. The results also show that INCLUS can discover clusters with overlapping items.

Table 3.7 Clustering Results for BMSPOS

Cluster ID	$\|C_i\|$	Hot items
1	2728	2, 4 ,9, 10, 11, 24, 31, 34, 35, 36, 37, 38, 43, 54, 63, 64, 66, 71, 82, 93
2	2722	2, 4, 9, 10, 11, 31, 36, 37, 43, 54, 110, 128, 173, 174, 230
3	2378	2, 4, 9, 10, 11, 24, 31, 34, 35, 36, 37, 43, 54, 63, 64, 66, 82, 93, 97, 99, 158
4	2368	2, 4, 9, 10, 11, 12, 13, 24, 31, 34, 35, 36, 37, 43, 48, 54, 62, 63, 64, 66, 82, 93, 97, 158
5	2354	2, 4, 9, 10, 11, 12, 17, 24, 31, 34, 35, 37, 43, 62, 75, 82

To test the scalability property of the proposed algorithm, 9 random samples of 10%, 20%, … 90% of transactions in the original BMSPOS dataset were used and the testing results are shown in Figure 3.4. It can be seen that the data processing time is linear to the size of the dataset as analyzed in Section 3.5, i.e. INCLUS is scalable.

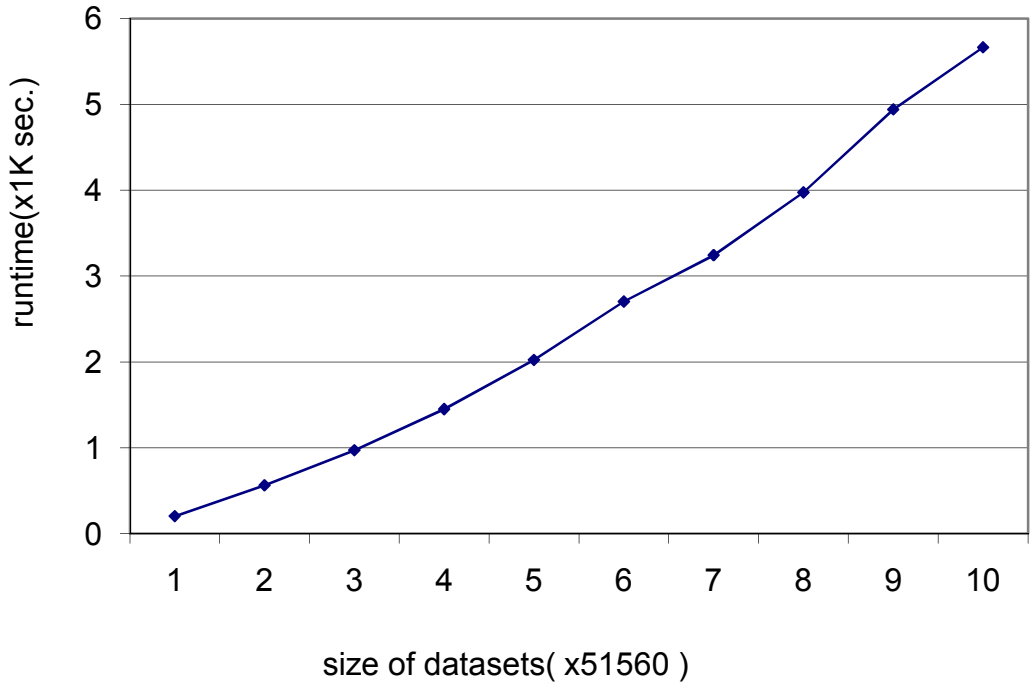


Figure 3.4 Scalability Testing Result

3.6.4 Advantage of Using the New similarity Measure

Since the new similarity measure is designed particularly for high dimensional sparse transactional data, Mushroom is chosen as a test dataset because its dimensionality and sparsity are higher than Vote.

The advantage of the newly defined similarity measure is tested on Mushroom using INCLUS equipped with different kinds of similarity measures. Testing results for $\gamma_t = 60\%$ and $S_t = 45\%$ are shown in Table 3.8. It can be seen that all the transactions are assigned to a singleton cluster by using Rao's Coefficient S_{rc} (i.e. all transactions are in the same cluster). On the other hand, each transaction is assigned to a different singleton cluster by using Simple Matching Coefficient SMC (i.e. every transaction is a cluster). It confirms the previous analysis in Chapter 2. It produces 11 clusters with $e = 4.0\%$ when S, the new defined similarity measure in this thesis, is applied. When Jacard Coefficient S_{jc} is used, it produces 16 clusters with $e = 3.7\%$. A good clustering should produce as

few clusters as possible and has as low an impurity as possible. Hence it can be concluded that S has superior discriminating power than S_{jc}.

Table 3.8 Comparison of Similarity Measures

Features	S	S_{jc}	S_{rc}	S_{smc}
k	11	16	8124	1
$\|C\|_{max}$	1828	1729	1	8124
e	4.0%	3.7%	0	not defined

3.7 Summary

In this chapter, a new similarity measure and a new notion of cluster representative were proposed for high dimensional sparse transactional datasets. An incremental algorithm was then presented based on these definitions for clustering very large transactional datasets. The algorithm is structure seeking rather than structure imposing. It produces a certain number of clusters based on the user's expectations on the closeness of transactions in a cluster. To get good clustering results, the users do not need to know the structure of the data in terms of the number of clusters existing in the data. An intensive empirical study shows that the new algorithm INCLUS is not only effective, efficient and scalable, but also insensitive to the order of transactions, which is crucial for an incremental algorithm. Since it is a one-pass algorithm, it can be extended for clustering data streams.

Chapter 4

Clustering Transactional Data Streams

The challenge of designing algorithms for data stream mining is three fold: (1) the algorithm is subject to sequential one-pass constraint over the data; (2) it must work under limited resources with respect to the unlimited data stream; (3) it should be able to reveal changes in the data stream over time.

For a finite statically stored dataset, the clustering problem is defined as follows: Given a set of data points, partition them into groups so that similar objects are in the same group according to a predefined similarity measure or objective function. In data stream settings, the set of data points to be studied is application dependent. It can be the whole data stream or a part of it depending on the purpose of clustering. As mentioned in Chapter 2, four prominent models have been proposed to filter the data points to be studied in data stream environments : landmark model (Guha *et al.*, 2003), sliding window model (O'Callaghan *et al.*, 2002; Babcock *et al.*, 2003) ; tilted time window model (Giannella *et al.*, 2003) and pyramidal time window model (Aggarwal *et al.*, 2003) .

In this chapter, the problem of clustering evolving transactional data streams is studied. Firstly, an *equal-width time window model* is proposed where the width of the window is the minimum granularity of interest for a particular application. Clustering snapshots need to be stored only for the minimum granularity from which the clustering for coarser granularities can be computed. Clustering can be obtained for the same or a higher level and the changes in clustering at different granularities can be evaluated. Secondly,

an *elastic window model* is proposed where the size of windows is adaptively resized based on the changes in clustering. In doing so, large amount of computing resources (memory and disk space) is saved in most cases and yet sufficient summary information is maintained to answer time sensitive queries at different time granularities.

Algorithms specific to transactional data stream clustering is designed. It incorporates INCLUS (Li and Gopalan, 2006a), an algorithm suitable for high dimensional sparse transactional data, into the equal-width time window model and elastic time window model so that changes over the data stream can be computed within the limited resources. The empirical results show that the algorithms are efficient and scalable.

The rest of the chapter is organized as follows. The framework for clustering transactional data streams is described in Section 4.1 and the corresponding algorithms are presented in Section 4.2. Section 4.3 describes the experimental results and Section 4.4 provides a summary of the chapter.

4.1 The Framework for Clustering Transactional Data Stream

A transactional data stream D consists of transactions $T_1, T_2, T_3, \ldots$ over a set I of d distinct items (attributes) arriving at time $t_1, t_2, t_3, \ldots$. Clustering transactional data is to partition the transactions into groups so that similar transactions are in the same cluster and dissimilar transactions are in different clusters.

In the data stream settings, people are more interested in the changes in the data stream. Mining changes in data streams is one of the core issues in data stream mining (Dong *et al.*, 2003). Aggarwal et al. (Aggarwal *et al.*, 2003) propose a framework for clustering evolving data streams. It splits the clustering process into an online micro-clustering component which is subject to a one-pass constraint and an offline macro-clustering component which is not constrained. In this chapter, a clustering algorithm for transactional data streams is developed based on the same framework.

As pointed out in (Aggarwal *et al.*, 2003), the separation of the clustering process into online and offline components raises the following questions:

1. What kind of summary information is to be stored?
2. When should the summary information be stored away on disk?
3. How can the summary information be used to reveal the changes in the data stream?

It is noted that the answer for the first question depends on the data and the induction principles for clustering. For example, CluStream which deals with d-dimensional numeric data using K-Means, keeps the summary information as the sum of squared data values and sum of data values for each dimension, sum of squares of the time stamps and sum of time stamps for data points in the cluster, and the number of transactions in a cluster. The summary information kept by CluStream is the temporal extension of cluster feature vectors (Zhang *et al.*, 1996) which is appropriate for numeric data streams. In this chapter, the summary information to be stored will be the temporal extension of cluster features defined in INCLUS (Li and Gopalan, 2006a), where the cluster features are described by the histogram of the cluster, start and finish time at which the cluster is computed, and the number of transactions. Cluster representatives are implicitly recorded in the histogram. The temporal extension of cluster features is called ***cluster snapshot*** which is defined below.

Definition 4.1. (**Cluster snapshot**). A cluster snapshot for a set of transactional data points in a time window w is $C(H,\ t_s,\ t_f,\ N)$, where H is the histogram of the cluster C, t_s and t_f are the start and finish times of the window, and N is the total number of transactions in the cluster.

The cluster representative (i.e. hot items) is implicitly recorded in the histogram of a cluster. In the histogram, items with $frequency > \gamma_t * |C|$ make up the cluster representative.

Definition 4.2. (**Clustering snapshot**). The clustering snapshot is the set of cluster snapshots for a time window.

Based on these two definitions, it can be easily seen that the clustering snapshot has following properties.

Additive property 4.1. Let $C_1(H_1, t_{s1}, t_{f1}, N_1)$ and $C_2(H_2, t_{s2}, t_{f2}, N_2)$ be two clusters in different clustering snapshots. If $t_{f1} = t_{s2}$, then the cluster features of $C = C_1 \cup C_2$ is $C(H_1 + H_2, t_{s1}, t_{f2}, N_1 + N_2)$.

Additive property 4.2. Let $C_1(H_1, t_{s1}, t_{f1}, N_1)$ and $C_2(H_2, t_{s2}, t_{f2}, N_2)$ be two clusters in the same clustering snapshot, i.e. $t_{s1} = t_{s2} = t_s$ and $t_{f1} = t_{f2} = t_f$, then the cluster features of $C = C_1 \cup C_2$ is $C(H_1 + H_2, t_s, t_f, N_1 + N_2)$.

Property 4.1 can be applied when merging clusters in two consecutive time windows while property 4.2 can be used to merge two similar clusters in the same time window.

The time interval at which summary information is to be stored onto the disk is also application dependent. For the supermarket basket data, keeping clustering snapshot at week level might be enough as promotions are often run on a weekly bases. For air traffic control, finding cluster changes in the air probably need to be based on seconds.

Definition 4.3. (**Clustering granularity**). Clustering granularity is the time period upon which clustering is performed.

For example, if clustering is performed every hour on the data points that arrived within the hour, the clustering granularity is an hour.

Definition 4.4. (Minimum clustering granularity). For a given application, the minimum granularity is the time period upon which the summary information is to be maintained to enable the time sensitive queries at the same or coarser granularities.

The minimum clustering granularity should be determined based on the nature of an application. For example, in order to analyze stock price changes within a week, a month or a year, daily price should be recorded, i.e. the minimum granularity should be a day.

To answer the second question, CluStream stores clustering snapshots based on the pyramidal time frame (Aggarwal *et al.*, 2003). In doing so, the disk space requirement is reduced by trading off accuracy. Snapshots are classified into different orders from 1 to $Log(T)$, where T is the time elapsed since the beginning of the stream. Each snapshot of the i-th order is taken at a moment in time when the time elapsed is a^i and only the last $a + 1$ snapshots are stored for each order.

An *equal-width time window model* (Figure 4.1a) is proposed in this thesis where the width of each window is equal to the minimum clustering granularity. The additive properties of cluster snapshots ensure that clustering for coarser time granularity can be obtained from the results of the minimum granularity.

The additive properties of cluster snapshots also indicate that it is not necessary to store snapshots for every window of the finest granularity; consecutive windows with same clustering features can be merged to save disk space, thus making the window size stretchable. So it is called the *elastic window model* (Figure 4.1b). In the worst case, the elastic window model will have the same number of clustering snapshots as for the equal-widrh window model when clustering features for any pair of consecutive windows are different. In the best case, only one clustering snapshot is stored when clustering features do not change over time.

Since the clustering snapshots are recorded on disk, it is possible to analyze the changes of clusters during the course of the data stream. For example, two clustering snapshots can be compared to evaluate changes in the number of clusters, the relative size of clusters and the cluster representatives.

(a) equal-width window model (b) elastic window model

Figure 4.1 New Models for Data Stream Processing

4.2 Algorithms for Clustering Transactional Data Stream

In this section, algorithms for mining transactional data streams will be presented. Each algorithm consists of an online micro-clustering module and an offline macro-clustering module. The online micro-clustering module gets clusters for each window and store them on the disk, the offline component discover changes over the data stream based on the results of online module. Two versions of the online clustering module are proposed by incorporating INCLUS with the equal-width window model and the elastic window model, respectively. The latter uses less memory and saves a lot of disk space, and yet provides good approximations.

Algorithm **CluTranStream_EQ** is based on the equal-width window model. Clustering snapshots will be written to disk at the end of each window. Algorithm **CluTranStream_EL** is based on the elastic window model. Except for the first window, the clustering snapshot will be stored to disk when changes occur. The online components of these algorithms are shown in Figure 4.2 and 4.3 respectively while the common offline component for both algorithms is shown in Figure 4.4.

Algorithm 1: CluTranStream_EQ Online Component

Input: *minimum support, minimum similarity, width of window*

Output: clustering snapshots

1. create a cluster with the first transaction
2. **while** not the end of the first window **do**
3. read the next transaction *T*;
4. allocate *T* to an existing cluster or a new cluster to maximize *S(P);*
5. update the cluster representatives of the modified cluster;
6. **endwhile**
7. write clustering snapshots to disk;

Figure 4.2 Online Component for CluTranStream_EQ

Algorithm 2: Online Component of CluTranStream_EL

Input: *minimum support, minimum similarity, width of window*

Output: clustering snapshots

1. create a cluster with the first transaction;
2. **while** not the end of the first window **do**
3. read the next transaction *T*;
4. allocate *T* to an existing cluster or a new cluster to maximize *S(P);*
5. update the cluster representatives of the modified cluster;
6. **endwhile**
7. write clustering snapshots to disk;

/* for the rest of the data stream */

8. read next transaction T;
9. **if** T cannot be assigned to an existing cluster **then**
10. write clustering snapshots to disk;
    create a new cluster for T;
11. **else**
    allocate *T* to an existing cluster;
12. repeat 8-12;

Figure 4.3 Online Component for CluTranStream_EL

Algorithm 3: Offline Macro-Clustering Component

Input: micro-clusters, period1, period2.

Ouput: clustering features for period1 and period2.

1. **for** each query period *i*
2. get all the clustering snapshots for the period;
3. compute clustering for period *i* according to property 1 and user input support and similarity thresholds;
4. **endfor**

Figure 4.4 Offline Macro-Clustering Component

4.3 Evaluation of the Algorithms

In this section, the proposed algorithms are evaluated in terms of accuracy, performance and scalability. The tests were performed on the online component only.

4.3.1 Test Datasets

Mushroom (Blake and Merz, 1998) is a categorical dataset with 22 categories and 116 values in total. It contains 8,124 records with class label 'e' (for edible) or 'p'(for poisonous) for each record. 4,208 edible mushrooms and 3,916 poisonous mushrooms are recorded in the dataset. Mushroom is converted to transactional data using the method mentioned in (Han and Kamber, 2000). All the missing values were ignored. The class labels of these datasets were not used in clustering but were used for evaluating the effectiveness of the clustering algorithms.

BMSPOS (Zheng *et al.*, 2001) is a real life high dimensional sparse transactional dataset which contains point-of-sale data from an electronics retailer. It has 515596 transactions, 1657 distinct items with an average 7.5

items per transaction. This dataset is used to test the scalability of the algorithms.

To test the scalability of the algorithm, some synthetic datasets are generated using the IBM synthetic data generator. Some datasets have the same number of transactions (10,000) but different number of attributes ranging from 500 to 4000, and some datasets with same number of attributes (1000) but different number of transactions in the range of 100K to 500K.

4.3.2 Testing Results

First the accuracy and performance of CluTranStream_EQ are tested using the Mushroom dataset by treating it as a data stream, i.e. each record in the dataset is read in the sequence as it appears and read only once. Assume t is the total time it takes for the 8120 records past the reader at a constant rate and the minimum granularity is $t/10$, then the whole stream(8120 records) can be divided into 10 windows, each having 812 transactions. The input parameters for INCLUS are ($\gamma_t = 100\%$, $S_t = 60\%$), where γ_t and S_t are support and similarity threshold, respectively. $\gamma_t = 100\%$ will ensure that some items are shared by all the transactions in the cluster. Impurity (Li and Gopalan, 2006a) is used as the measure of accuracy. The result is shown in Table 4.1. It can be seen that the clustering features are changing over time. In the first window, there are 22 clusters while in the fourth window there are only 6 clusters. The speed of processing the dataset is about 2000 records per second.

Table 4.1 Testing Results for Mushroom

	W_1	W_2	W_3	W_4	W_5	W_6	W_7	W_8	W_9	W_{10}
k	22	20	16	6	7	14	15	19	11	15
e(%)	7.9	0.7	0.7	0	0.1	0.9	2.6	0.7	0	0
t(s)	0.58	0.59	0.44	0	0.13	0.41	0.41	0.48	0.29	0.6

Note: W_i - ith window , k-number of clusters, e-impurity, t-runtime

It can be seen from Table 4.1 that the average error is 1.36%, lower than that for any combination of micro-clusters and window sizes reported for SCLOPE(Ong *et al.*, 2004).

In order to compare the disk space usage by equal-width window model and elastic window model, a new database was obtained by appending Mushroom dataset to itself 7 times to model a data stream where underlying data generation mechanism does not change. The input parameters were $\gamma_t = 60\%$, $S_t = 45\%$, and window size is 8124, the size of Mushroom dataset. As expected, only two clustering snapshots were stored to disk for the elastic window model. The disk space was largely reduced by using the elastic window model.

To do the scalability test, the whole dataset was treated as the content of one window. Figure 4.5 illustrates the scalability test with respect to the number of transactions and the number of attributes using synthetic data. Figure. 4.5a shows results of scalability testing with respect to the number of transactions. Five synthetic datasets are used for the test. All the datasets have 1000 attributes, but with 100K, 200K, 300K, 400K and 500K transactions, respectively. It can be seen that the algorithm scales up as the number of transactions increase. Figure 4.5b shows results of scalability testing with respect to the number of attributes in the dataset. Eight datasets with 10K transactions are used for the test. The number of attributes in those datasets is 500, 1000, 1500, 2000, 2500, 3000, 3500 and 4000. It can be seen that the algorithm scales up as the number of attributes increases. Figure 4.6 shows the scalability test on the real dataset BMSPOS given $(\gamma_t = 10\%, S_t = 10\%)$. Processing speed for this dataset is more than 1000 records per second. It shows again that the algorithm is scalable with respect to the number of transactions in datasets.

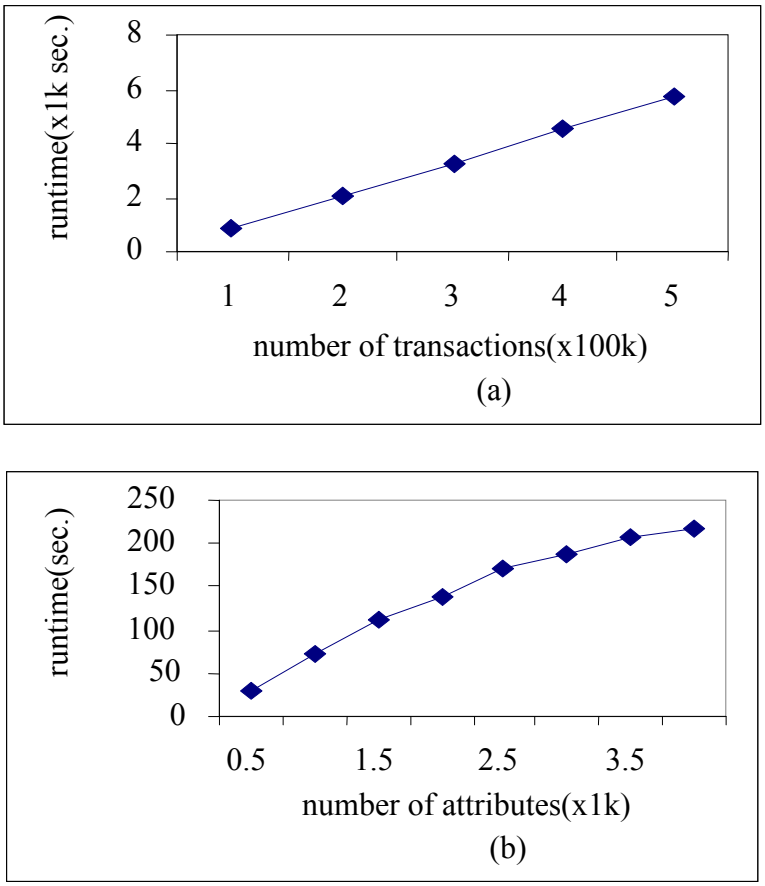


Figure 4.5 Scalability Test Using Synthetic Datasets

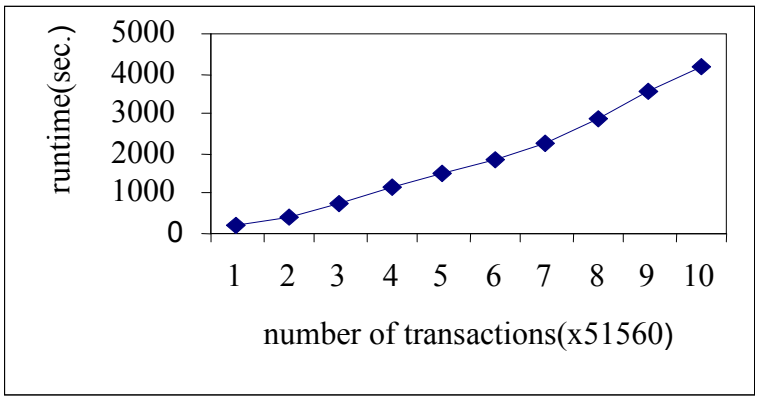


Figure 4.6 Scalability Test on BMSPOS

4.4 Summary

In this chapter, an equal-width time window model and an elastic time window model are proposed for the cluster analysis of data streams. The incremental transactional data clustering algorithm INCLUS are incorporated into these models in order to detect clustering changes at different granularities.

The width of a window in the equal-width time window model is set to the minimum granularity of interest for a particular application. In doing so, clustering can be obtained for the same or a coarser granularity, and hence the changes in clustering at different granularities can be evaluated. The original size of an elastic window is set to the minimum granularity, and then resized, if applicable, based on the changes in clustering. A large amount of computing resources (memory and disk space) is saved in most cases and yet sufficient summary information is maintained to answer time sensitive queries at different time granularities. The empirical results show that the algorithms are efficient and scalable.

Chapter 5

Sampling Large Databases for Association Rules

Although the main purpose of sampling a static large disk resident database is to reduce the amount of data to be mined, sampling seems to be the only choice for processing a data stream where data flows at a rate faster than it can be processed (Babcock *et al.*, 2002). Motivated by sampling data streams for mining association rules, this thesis investigates effective sampling methods that not only require small sample sizes but also provide approximation guarantees.

In this thesis, the datasets are randomly sampled by replacement and the sufficient sample size is derived using binomial distribution and the central limit theorem (CLT). The accuracy of the new sampling approach is theoretically analyzed and its effectiveness is evaluated on both dense and sparse datasets. Methods for reducing false positives and false negatives of frequent itemsets are also discussed.

The rest of this chapter is organized as follows: A theoretical analysis of random sampling for association rules is presented in Section 5.1 and the experimental evaluation shown in Section 5.2. Section 5.3 discusses the methods for reducing errors and Section 5.4 provides a summary of this chapter.

5.1 Sampling Techniques for Association Rules Mining

In the context of sampling large databases for association rules, a transaction database TDB of size N is the population to be studied, and a

sample is a subset of TDB that consists of n transactions selected from the population of size N. A sampling method is proposed for mining association rules in this section, along with the derivation of the sufficient sample size using binomial distribution and central limit theorem. An analysis of the accuracy of itemset supports computed from a random sample is also given.

5.1.1 Random Sampling of a Database with Replacement

There are two kinds of random sampling methods, random sampling with replacement and random sampling without replacement (Thomson, 1992). Random sampling without replacement obtains a sample of size n by selecting n units from the population and at each step every unit in the population not already selected has an equal chance of being selected. Sampling with replacement obtains n units independently and at each step every unit in the population has an equal chance of inclusion in the sample.

A process is called a Bernoulli process if it meets the following criteria (Walpole *et al.*, 1998):

The experiment consists of n repeated trials;

Each trial results in an outcome that may be classified as a success or a failure;

The probability of success, denoted by p , remains constant from trial to trial;

The repeated trials are independent.

In this thesis, a transactional database is sampled by sampling with replacement so that the process of selecting n transactions out of N transactions has all the properties of a Bernoulli process:

There are n trials;

There are only two complementary outcomes for each trial: an itemset appears or does not appear in a trial. The probability p of an itemset appearing remains the same from trial to trial;

Let q denote the probability that an itemset does not appear in a trial, then $q = 1 - p$;

Each trial is independent. The probability of a transaction being selected in a trial is independent of which transactions have been selected in the previous trials.

The number of times X that an itemset appears in n Bernoulli trials (i.e. the number of times X an itemset appears in a sample), is a binomial random variable and the probability distribution of this discrete random variable follows the binomial distribution (Mendenhall and Sincich, 1992).

If we denote outcomes of the ith trial as $X_i (i = 1,\ 2, \dots, n)$ where $X_i = 1$ if an itemset appears, and $X_i = 0$ if an itemset does not appear, then the number of appearances of an itemset in the sample is

$$X = \sum_{i=1}^{n} X_i \tag{5.1}$$

Therefore $P_s = X/n$, the support of an itemset in the sample, is the sample mean. P_s is an unbiased estimator of p (Mendenhall and Sincich, 1992).

5.1.2 **Determining the Sufficient Sample Size**

According to the central limit theorem (CLT), when the sample size is large, P_s is approximately normally distributed with mean $\mu = p$ and variance $\sigma^2 = pq/n$ (Mendenhall and Sincich, 1992). The normal distribution of P_s can be transformed to standard normal distribution of a standard random variable

$$Z = \frac{P_s - \mu}{\sigma} = \frac{P_s - p}{\sqrt{pq/n}} \tag{5.2}$$

Therefore we can assert that the probability that Z lies in $[-Z_{\alpha/2}, Z_{\alpha/2}]$ is $1-\alpha$:

$$\Pr\left(-Z_{\alpha/2} < Z < Z_{\alpha/2}\right) = 1-\alpha \tag{5.3}$$

where $Z_{\alpha/2}$ is the Z value above which the area under the standard normal curve is $\alpha/2$. $1-\alpha$ is called confidence coefficient in (Mendenhall and Sincich, 1992) and we call it "confidence level" since it represents the degree of confidence that Z lies in $[-Z_{\alpha/2}, Z_{\alpha/2}]$. We can derive the following equation from Equations (5.2) and (5.3):

$$\Pr\left(P_s - Z_{\alpha/2}\sqrt{pq/n} < p < P_s + Z_{\alpha/2}\sqrt{pq/n}\right) = 1-\alpha \tag{5.4}$$

Because the normal curve is symmetric, Eq.(5.4) can be decomposed into

$$\Pr\left(p > P_s + Z_{\alpha/2}\sqrt{pq/n}\right) = \alpha/2 \tag{5.5}$$

and

$$\Pr\left(p < P_s - Z_{\alpha/2}\sqrt{pq/n}\right) = \alpha/2 \tag{5.6}$$

Let's denote the differences between the estimated support of an itemset in a sample RD and its support in the original database TDB as $\Delta p = |P_s - p|$, then Eq. (5.4) can be rewritten as

$$\Pr\left(\Delta p < Z_{\alpha/2}\sqrt{pq/n}\right) = 1-\alpha \tag{5.7}$$

Given an error bound e and the confidence level $1-\alpha$, we must choose sample size n such that

$$\Delta p < Z_{\alpha/2}\sqrt{pq/n} \leq e \tag{5.8}$$

Thus we have

$$n \geq \frac{Z_{\alpha/2}^2 pq}{e^2} \tag{5.9}$$

For an itemset with support p, Eq. (5.9) will give the sufficient sample size that can estimate p with $1-\alpha$ confidence that an error will not exceed e. Since pq has the maximum value of $1/4$ when $p = q = 1/2$, if we choose

$$n \geq \frac{Z_{\alpha/2}^2}{e^2} \quad , \tag{5.10}$$

then we will be at least $1-\alpha$ confident that Δp will not exceed e.

For a given error bound and confidence level, the sample size calculated using Eq. (5.10) which is based on central limit theorem(CLT), is much smaller than that based on Chernoff Bounds (Mannila *et al.*, 1994; Toivonen, 1996). Table 5.1 provides some comparisons.

Table 5.1 Sufficient Sample Size

e	α	Chernoff Bounds	CLT
0.01	0.01	26492	16513
0.005	0.01	105966	66049
0.01	0.05	18445	9604
0.005	0.05	73778	38416

5.1.3 Accuracy of Sampling

Theorem 5.1. Given an itemset X whose support is p in database D, a confidence level $1-\alpha$, and a random sample RD of size

$$n \geq \frac{Z_{\alpha/2}^2}{4e^2} \ ,$$

the probability that the difference in support Δp between the sample and the database exceeds e is at most α.

Proof.

$$\Pr(\Delta p > e) = \Pr\left(\Delta p > Z_{\alpha/2}\sqrt{1/4n}\right)$$

$$\leq \Pr\left(\Delta p > Z_{\alpha/2}\sqrt{pq/n}\right)$$

$$\leq 1 - \Pr\left(\Delta p < Z_{\alpha/2}\sqrt{pq/n}\right)$$

$$\leq \alpha \quad \text{(apply Eq. (5.7))}$$

5.2 Effectiveness of Sampling

The effectiveness of the proposed sampling method is experimentally studied on both dense and sparse datasets. The datasets used in the experiments, the measurement of errors, and the experimental results are described below.

5.2.1 Datasets Studied

The experiments are performed on both dense and sparse datasets. The datasets used include: (1) a synthetic sparse dataset, T10I4D100K , generated by the synthetic data generator provided by the QUEST project(Agrawal and Srikant, 1994) to simulate market basket data; (2) a sparse real dataset BMSPOS; (3) a dense dataset Connect-4 which is gathered from connect-4 game state information and are available from the UCI Machine Learning Repository (Blake and Merz, 1998). These datasets are benchmarked at FIMI (Frequent Itemsets Mining Implementations Repository). Table 5.2 summarizes their characteristics, where N is the number of transactions in the dataset, T is the average transaction length and $|R|$ is the number of distinct items in the dataset.

Table 5.2 Database Summaries

Dataset Name	N	|R|	T
T10I4D100K	100000	870	10
BMSPOS	515597	1657	7.5
Connect-4	67557	129	43

5.2.2 **Measurement of Errors**

Errors in the estimation of itemset support and the errors in the estimation of the complete frequent itemsets (CFI) will be checked in this section.

The errors in itemset support estimation are evaluated as follows. s samples of size n are taken from database TDB, and for each item in the TDB, the number of times x that $\Delta p > e$ in s samples are counted, and the experimental probability of f that $\Delta p > e$ is calculated as $f = x/s$.

The CFI in the original database and the sample are denoted as FI_o and FI_s, respectively. If an itemset exists in FI_o but not in FI_s, then this itemset is called a false negative. If an itemset exists in FI_s but not FI_o, then the itemset is called a false positive. The collection of all the false positives is denoted by F_p and the collection of all the false negatives is denoted by F_n. The errors are measured by

$$f_p = \frac{|F_p|}{|FI_s|} \tag{5.11}$$

which represents the proportion of the false frequent itemsets in a sample, and

$$f_n = \frac{|F_n|}{|FI_s|} \tag{5.12}$$

which represents the proportion of the frequent itemsets that are missing in a sample.

A set of frequent itemsets FI can be partitioned into ml subsets according to the size of each itemset.

$$FI = \bigcup_{l=1}^{ml} FI_l \tag{5.13}$$

where FI_l is a set of itemsets with size of l and ml is the size of the longest itemset. The errors in CFI estimation and the errors in each partition of CFI as well will be checked in the next section.

5.2.3 **Experimental Results**

Given a transactional database TDB with N transactions, random sampling with replacement for association rules proceeds as follows:

1. Calculate the sample size n for a given error bound and confidence level using Eq. (5.10).

2. Generate a set of n random integers $Rand$ where

$Rand = \{I_1, I_2, \dots, I_j, \dots I_n\}$, and $I_j \in \{1, 2, \dots, n)$. Duplicates are allowed in order to simulate random sampling with replacement.

3. For each I_j in $Rand$, retrieve I_jth transaction in the TDB and add it to the sample.

4. Apply any standard association rules mining algorithm to the sample.

According to **Theorem 5.1**, for a given confidence level $1 - \alpha = 0.95$ and a random sample RD of size 9604, the probability that Δp exceed $e = 1\%$ is at most 5%. Tests were performed on the datasets listed in Table 5.2 to check if the claim holds for the proposed sampling approach.

100 samples of size 9604 from each database were obtained through random sampling with replacement. The support of each item in each sample was computed and compared with the support of the item in the original database. For each item, the number of times(samples) x that $\Delta p > 1\%$ is counted in the 100 samples. The probability of f for $\Delta p > 1\%$ obtained from the test is $f = x/100$. Table 5.3 lists the experimental probability of f that $\Delta p > 1\%$ for items in each database. For T10I4D100K, none of the items in the dataset has $\Delta p > 1\%$ in any of the 100 samples. In other words, the probability for $\Delta p > 1\%$ for each item is 0.

For BMSPOS, 1654 items in each sample has $\Delta p < 1\%$ while 2 items have $\Delta p > 1\%$ in one sample and 1 item has $\Delta p > 1\%$ in 2 samples. In other

words, the probability that $\Delta p > e(1\%)$ for an item in this database is no more than 2%.

There is only one item in Connect-4 with 7% probability that $\Delta p > 1\%$ while all other items have at most 5%. The results confirm the theorem and empirically prove that the proposed sampling approach can provide the expected approximation guarantees.

Table 5.3 Frequency Distribution of *f* in Each Dataset

No of times $\Delta p > 1\%$		0	1	2	3	4	5	6	7
$f(\%)$		0	1	2	3	4	5	6	7
No of items	Connect-4(129 items)	106	8	9	1	0	4	0	1
	BMSPOS(1657 items)	1654	2	1					
	T10I4D100K(870 items)	870							

Next, the errors in frequent itemsets estimation were checked. Since different samples may result in different error rates, the average outcomes of 50 samples are taken to evaluate the errors. Error bound $e = 0.01$ and confidence level $1 - \alpha = 0.99$ are chosen in the experiments to evaluate the effectiveness of the proposed sampling approach. The sufficient sample size is 16513 for $e = 0.01$ and $1 - \alpha = 0.99$. The experiments for error bound $e = 0.01$ and confidence level $1 - \alpha = 0.95$, which result in a sample size of 9604 are also performed for comparison. Support thresholds were chosen in such a manner that at least frequent itemsets of size 4 can be produced. The following analyses of the experimental results were performed on the samples of size 16513 if the sample size is not explicitly stated.

Figure 5.1 shows the errors (f_p and f_n) for different support thresholds in each dataset. In the figure, the number following f_p or f_n is the sample size. It can be seen that the errors fluctuate as the support threshold changes. For Connect-4, the errors increase as the support threshold increases while for the other datasets the errors decrease as the support threshold increases. The errors for dense datasets are small for every support threshold

computed and the changes in the error are relatively small compared with the changes in support threshold. For example, for Connect-4, f_p and f_n are 2.6% and 3.4%, respectively when support threshold is 80%; and they change to 4.6% and 4.7%, respectively when the support threshold increases to 95%. For the sparse datasets, the errors are relatively large and so are the changes in errors compared with the changes in support threshold. For instance, in BMSPOS, when the support threshold increases from 0.5% to 1%, the f_p value decreases from 9.3% to 5.4% and f_n decreases from 8.4% to 6.5%. For all the datasets and all the computed support thresholds, at least 85% of FI_o is discovered by sampling. It confirms that the proposed sampling approach is effective.

Let's take a closer look at errors in FI_s by inspecting each partition FI_l (l = 1, 2, …, ml) and the results are shown in Figure. 5.2. The errors for frequent 1-itemsets are always small for both dense and sparse datasets. It also reveals that within the overall errors in FI_s, the errors for each partition may vary dramatically and are not predictable.

The causes of errors f_p and f_n in frequent itemsets estimation not only depend on the errors in support estimations of itemsets, but also on two other factors given below.

(1) The propagation error.

If an itemset is missed in the sample, then its super sets will be missed; if an itemset is mistaken as a frequent itemset, then its super sets may be mistaken as frequent as well. This is because the association rules mining algorithms apply the apriori principle: if an itemset is frequent, then all its subsets must be frequent. For example, for a sample of Connect-4, when $p_t = 95\%$, itemset {109,121} is missed in the sample, and its super sets {109, 121, 124}, {109, 121, 127} and {109, 121, 124, 127} are consequently missed, too; Itemset {19 72, 88, 124} is mistaken as frequent itemset, its super sets {19 72, 75, 88, 124} and {19 72, 75, 88, 124, 127} are mistaken as frequent itemsets as well.

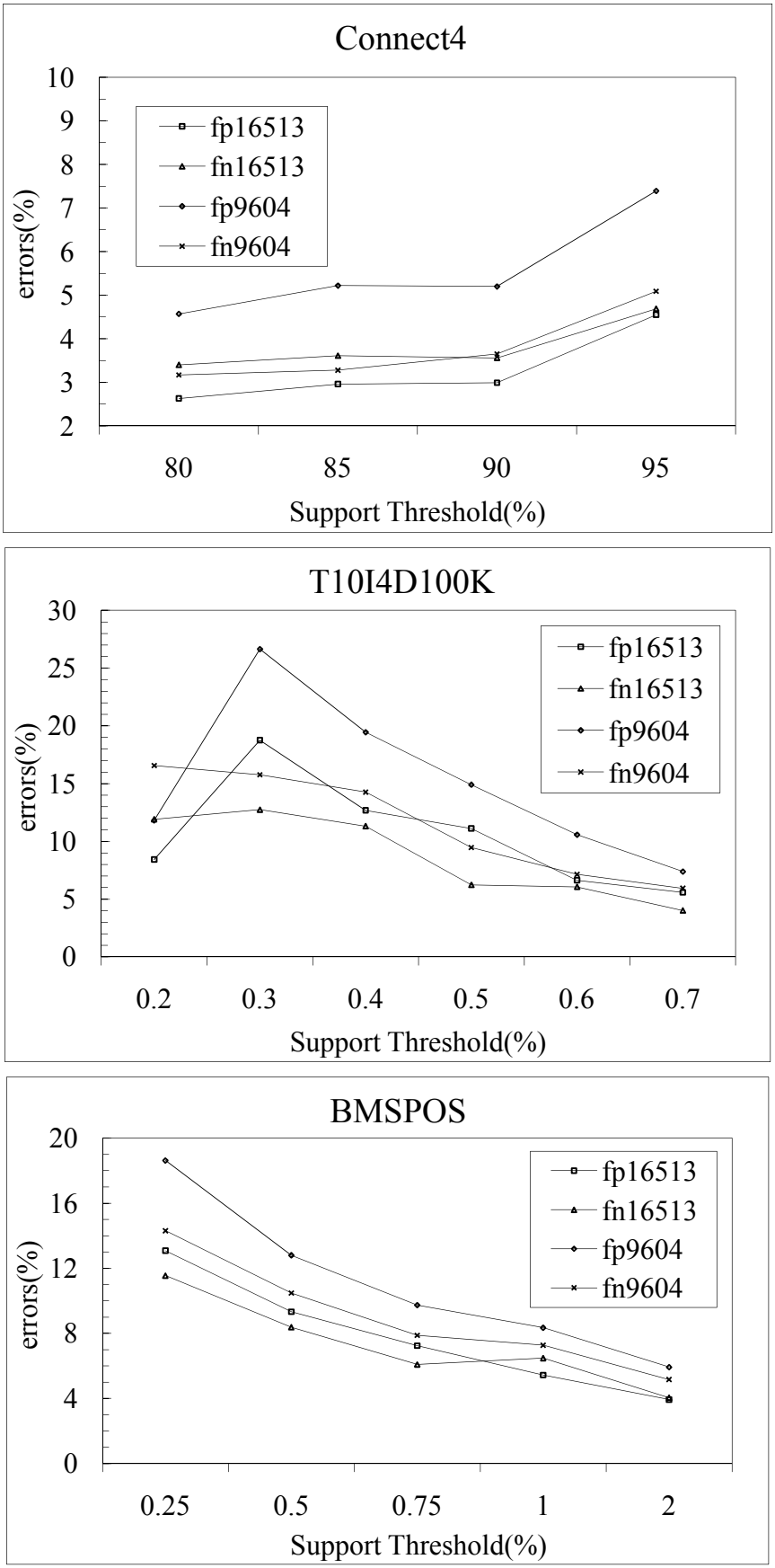
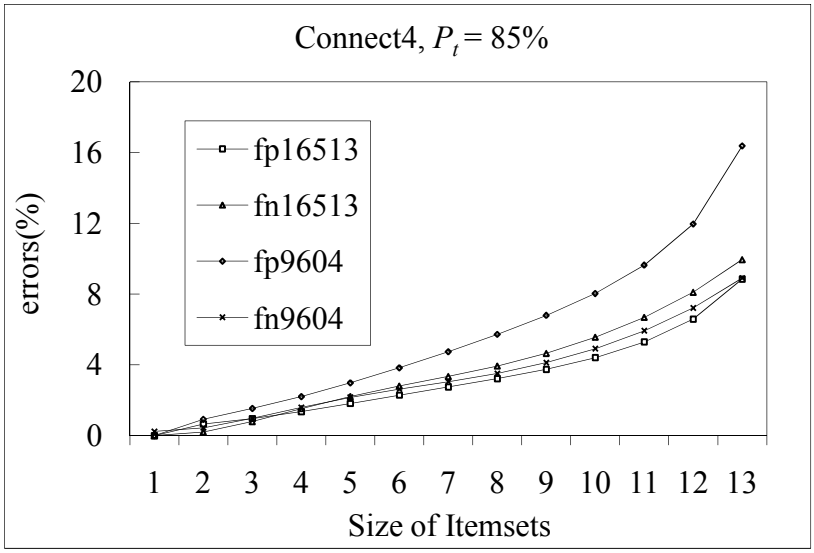
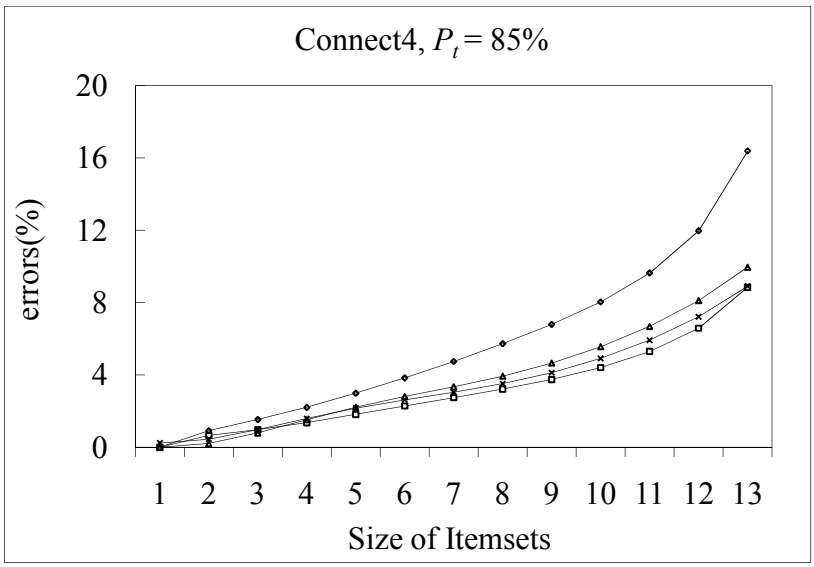


Figure 5.1 Errors for Different Support Thresholds





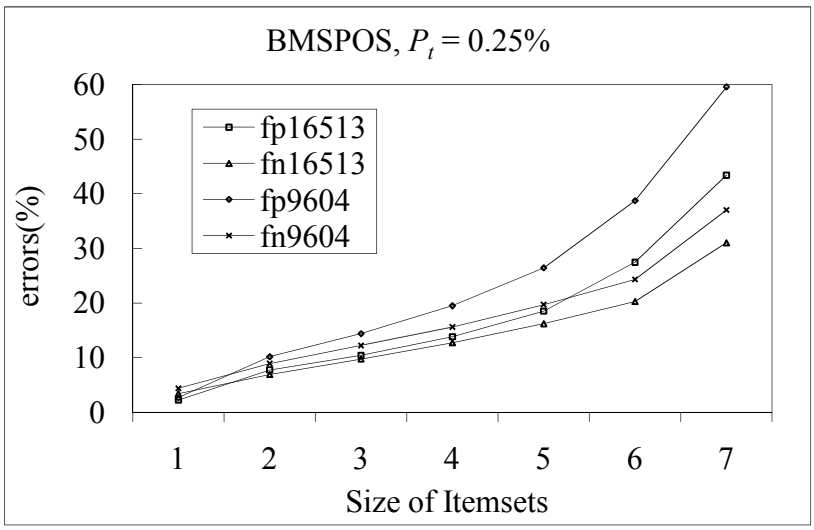


Figure 5.2 Errors in Each Partition of FIs

(2) The proportion of frequent itemsets whose support is close to p_t.

The larger the proportion of the itemsets whose support is close to the specified support threshold p_t, the more likely bigger errors will occur. According to the previous analysis, those itemsets with support $p_t - e < p < p_t + e$ are likely to be missed or mistaken as frequent itemsets. In Connect-4, among those items with support greater than $89\%, 13\%$ of them have supports within $(89\%, 91\%)$; and among those items with supports greater than 84%, only 4% of them have supports within $(84\%, 86\%)$. Consequently, when $e = 1\%, 2.27\%$ percentage of the frequent 1-itemsets in the sample are false positives for $p_t = 90\%$, while there are no false positives presented for $p_t = 85\%$. In both cases, none of the frequent 1-itemsets is missed.

The experimental results also show that both f_p and f_n for the samples of size 9604 are bigger than that for the samples of size 16513. This is a tradeoff between sample size (hence efficiency) and the confidence level.

5.3 Reducing Errors

In this section, the possibility of reducing errors in frequent itemset estimations is explored.

Theorem 5.2. Given a frequent itemset X in a TDB with $p > p_t$, a random sample RD, and a confidence level $1 - \alpha$, the probability that X is a false negative in RD is at most $\alpha/2$ when the support threshold is lowered to

$$p_{tl} = p_t - Z_{\frac{\alpha}{2}}\sqrt{\frac{1}{4n}} \tag{5.14}$$

Proof. When the support threshold is lowered to p_{tl}, the probability that an itemset X is a false negative in RD equals the probability that the estimated support P_s of X is smaller than p_{tl}.

$$\Pr(P_s < p_{tl}) = \Pr\left(P_s < p_t - Z_{\alpha/2}\sqrt{1/(4n)}\right)$$

$$\leq \Pr\left(P_s < p_t - Z_{\alpha/2}\sqrt{pq/n}\right)$$

$$\leq \alpha/2 \ \ (\text{apply Eq. } (5.5))$$

For $p > p_t \geq 50\%,\ p_t q_t \geq pq$, lowering the support threshold to

$$p_{tl} = p_t - Z_{\alpha/2}\sqrt{\frac{p_t q_t}{4n}} \tag{5.15}$$

will give the same confidence level but smaller amount by which the threshold is to be lowered. As a result, less false positives maybe present in the frequent itemsets.

Theorem 5.3. Given an itemset X with $p > p_t$ in a TDB, a random sample RD, and a confidence level $1 - \alpha$, the probability that X is a false positive in RD is at most $\alpha/2$ when the support threshold is increased to

$$p_{tu} = p_t + Z_{\frac{\alpha}{2}}\sqrt{\frac{1}{4n}} \tag{5.16}$$

Proof. When the support threshold is increased to p_{tu}, the probability that an itemset X in RD is a false positive equals the probability that the estimated support P_s of X is bigger than p_{tu}.

$$\Pr(P_s > p_{tu}) = \Pr\left(P_s > p_t + Z_{\frac{\alpha}{2}}\sqrt{\frac{1}{4n}}\right)$$

$$\leq \Pr\left(P_s > p_t + Z_{\frac{\alpha}{2}}\sqrt{\frac{pq}{n}}\right)$$

$$\leq \alpha/2 \ \ (\text{apply Eq. } (5.6))$$

For $p < p_t \leq 50\%,\ p_t q_t \geq pq$, increasing the support threshold to

$$p_{tu} = p_t + Z_{\alpha/2}\sqrt{\frac{p_t q_t}{4n}} \tag{5.17}$$

will give the same confidence level but a smaller amount of increase in threshold. In doing so, less frequent itemsets can be missed as the threshold increases.

If we do not want to miss frequent itemsets present in the original database, then we can lower the support threshold according to equations (5.14) or (5.15). On the contrary, if we do not want false frequent itemsets to appear in the mined frequent itemsets, we can increase the threshold according to equations (5.16) or (5.17). For instance, given $n = 16513$, $1 - \alpha = 0.99$ and $p_t = 2\%$, p_{tl} and p_{tu} will be 1.72% and 2.28%, respectively, according to equation (5.14) and (5.16). Experimented on a sample of BMSPOS for $p_t = 2\%$ has confirmed this. When the support threshold is lowered to 1.72%, there were no missed itemsets; when it was increased to 2.28%, only 0.42% were false frequent itemsets.

5.4 Summary

In this chapter, sampling with replacement method is proposed for the association rules mining of very large datasets. The sufficient sample size is derived based on binomial distribution and the central limit theorem. For a given confidence level and error bound, the proposed sampling approach requires smaller sample size than that based on the Chernoff Bounds but still provides the desired approximation guarantees for supports of itemsets. For applications where the false positives may be very costly, the support threshold can be increased based on Theorem 5.2 to reduce false positives. On the other hand, if all the frequent itemsets are to be fully discovered, the support threshold can be lowered according to Theorem 5.3 to reduce the number of false negatives.

Chapter 6

Stratified Sampling for Association Rules Mining

If a dataset can be partitioned into groups with distinct features for a particular data mining task, then proportionally sampling each group will give the exact result as with the whole dataset. In this chapter, the feasibility of using stratified random sampling for association rules mining is studied. A dataset is first partitioned into strata according to the size of each transaction, and then simple random sampling is applied to each stratum. The accuracy of the proposed stratified sampling method is compared with that using the simple random sampling method.

The rest of this chapter is organized as follows. Section 6.1 proposes a stratified random sampling method for association rules, and the effectiveness of the proposed stratified sampling method is experimentally studied in Section 6.2. Section 6.3 presents the conclusions and some discussions.

6.1 Transaction Size Based Stratified Random Sampling

In stratified random sampling, the population of size N is partitioned into k strata and a sample is selected by simple random sampling within each stratum (Thomson, 1992). Given a total sample size n, if the strata differ in size, proportional allocation can be used to maintain a steady sampling fraction throughout the population (Thomson, 1992). If stratum k has N_k units, the sample size allocated to it will be

$$n_k = \frac{n}{N} N_k \tag{6.1}$$

The principle of stratification is to partition the population in such a way that the units within a stratum are as similar as possible. For example, in the survey of a human population, stratification may be based on the geographic region, sex or socio-economic factors.

To the best of our knowledge, stratified sampling has not been applied in association rules mining. In this thesis, the feasibility of stratified sampling for association rules mining is explored. A dataset is partitioned according to transaction sizes. It is based on the fact that two identical transactions must have the same transaction size.

The minimum sample size n is determined using a formula given in Chapter 5 (Li and Gopalan, 2004) for a given error bound e, and a confidence level $1 - \alpha$

$$n = \frac{Z_{\alpha/2}^2}{4e^2} \tag{6.2}$$

where $Z_{\alpha/2}$ is the Z value above which the area under the standard normal curve is $\alpha/2$.

According to Equations (6.1) and (6.2), the sample size for the k-th stratum will be

$$n_k = \frac{Z_{\alpha/2}^2}{4e^2} \quad \frac{N_k}{N} \tag{6.3}$$

Definition 6.1 (*Width of stratum*). The difference between the size of the longest transaction and the shortest transaction in a stratum is called the width of the stratum, which is denoted as w.

Definition 6.2 (*Equal width partition*). The data is partitioned in such a way that each stratum has the same width.

In an equal-width partition, there will be $\lceil L_{max}/w \rceil$ strata for a dataset whose longest transaction size is L_{max}. Transactions with size s will be partitioned to k-th stratum, where $k = \lceil s/w \rceil$. For example, when $w = 2$, transactions with size $s = 1$ and $s = 2$ will go to 1st stratum, transactions with size $s = 3$ and $s = 4$ will go to 2nd stratum, and so on.

Given a minimum support, an error bound e, a confidence level $1 - \alpha$, and the desired width of a stratum w, a transaction size based stratified sampling algorithm is proposed as described in Figure 6.1.

Algorithm for Association Rules Mining by Stratified Sampling

Input: dataset *TDB*, nunmber of transactions in the database N, width of a stratum w, error bound e, confidence level$1 - \alpha$, minimum support minsup

output: frequent itemsets

initialize a sample $S = \{\}$;

calculate the sufficient sample size *n* using equation (6.2);

partition the dataset into $\lceil L_{max}/w \rceil$ strata based on the size of transactions;

calculate sample size for each stratum using equation (6.3);

sample each stratum by simple random sampling without replacement and add to *S*;

run a standard association rules mining algorithm on *S*.

Figure 6.1 Algorithm for Association Rules Mining by Stratified Sampling

6.2 Effectiveness of Stratified Sampling

6.2.1 Measurement of Accuracy and Errors

The complete frequent itemsets discovered from the original database and its sample are denoted as $L(D)$ and $L(S)$, respectively. According to the

definitions of false positive and false negative in Chapter 5, the number of false positives is $|L(S) - L(D)|$ and the number of false negatives is $|L(D) - L(S)|$. The same measure as in (Chen *et al.*, 2002) is used to obtain the accuracy of sampling:

$$\text{accuracy} = 1 - \frac{|L(D) - L(S)| + |L(S) - L(D)|}{|L(D)| + |L(S)|} \tag{6.4}$$

This measurement is sensitive to both false positives and false negatives.

The two measurements f_p and f_n defined in Chapter 5 are also used to quantify the errors of sampling, which can be expressed as

$$f_p = \frac{|L(S) - L(D)|}{|L(S)|} \tag{6.5}$$

$$f_n = \frac{|L(S) - L(D)|}{|L(D)|} \tag{6.6}$$

f_p represents the proportion of the false frequent itemsets in a sample while f_n represents the proportion of the frequent itemsets in the original dataset that is missing in a sample.

6.2.2 **Datasets Studied**

Experiments are performed on three datasets that are available at FIMI'03 (Frequent Itemsets Mining Implementations Repository). They are: (1) BMSPOS dataset, provided by Blue Martin Software, (2) Retail dataset, donated by Tom Brijs, which contains the (anonymized) retail market basket data from an anonymous Belgian retail store, and (3) Accidents dataset, donated by Karolien Geurts and contains (anonymized) traffic accident data. The density of the Accidents dataset is relatively higher than for the other two. The characteristics of these datasets are summarized in Table 6.1, where N is the number of transactions in a database, T is the average transaction length, $|R|$ is the number of distinct items in the database and L_{max} is the size of the longest transaction. The histogram for each dataset is shown in Figure 6.2. To save space, transactions with size greater than 20

in BMSPOS and transactions with size greater than 27 in Retail, which only count for less than 5% of transactions, are not shown on the histogram.

Table 6.1 Summaries of Characteristics of Datasets

Dataset Name	*N*	*|R|*	*T*	L_{max}	*Density(%)*
BMSPOS	515596	1657	7.5	164	0.45
Retail	88162	16570	13	76	0.08
Accidents	340184	468	34	51	7.26

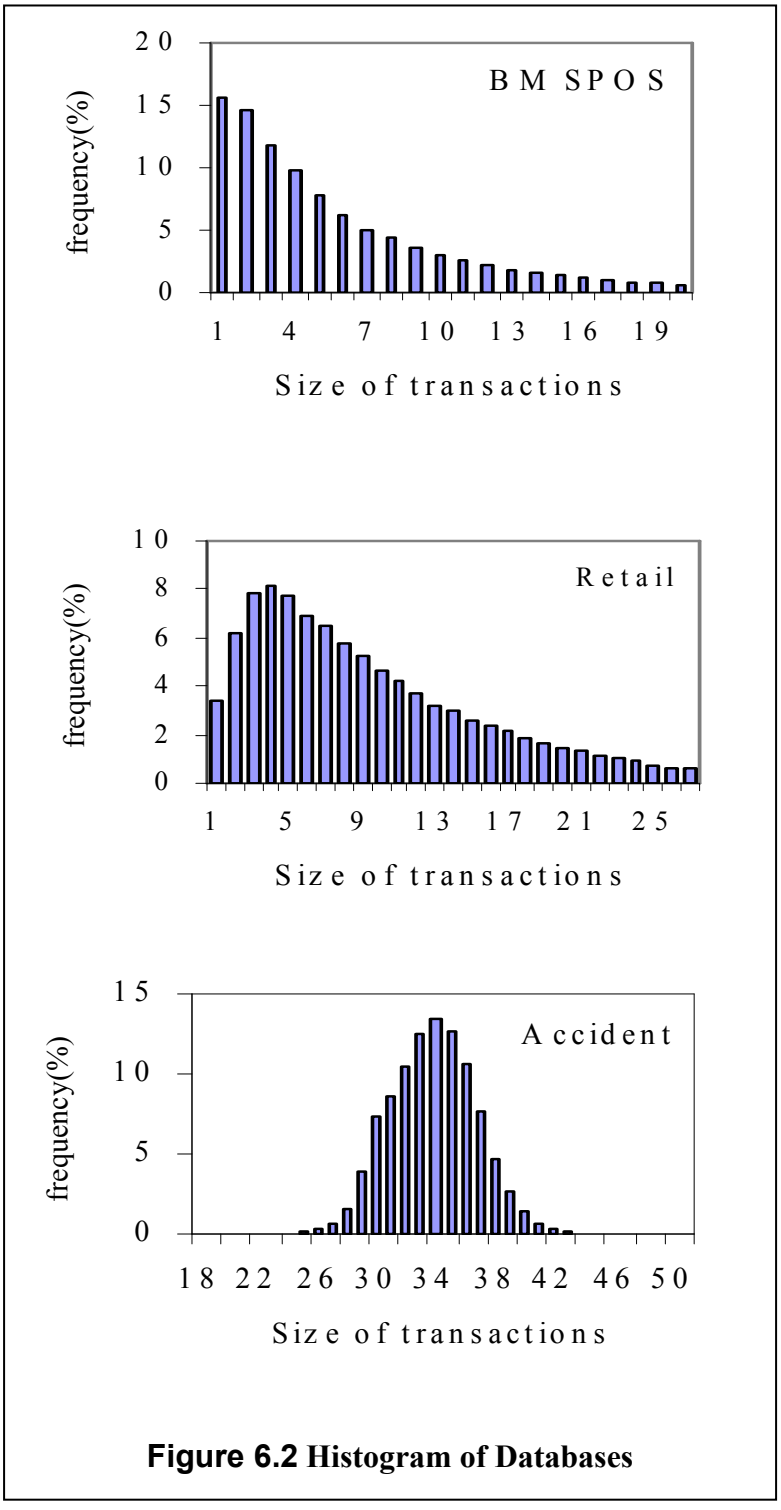


Figure 6.2 Histogram of Databases

6.2.3 **Experimental Results**

In this subsection the experimental results of the proposed stratified sampling method are described and compared with that of the simple random sampling method in (Li and Gopalan, 2004). The sample size chosen is 16513, which corresponds to an error bound of 0.01 and a confidence level of 99%.

As mentioned before, the dataset is partitioned based on the transaction sizes. Given a w value, the dataset is partitioned into $k = \lceil L_{\max} / w \rceil$ strata. The width of each stratum in the resulting strata is w except that the width of kth stratum may be less than w. Each stratum is sampled according to Eq. (6.1). The accuracy and errors of the proposed stratified sampling method were compared with that of the simple random sampling method. Figures 6.3-6.5 show some of the test results on different datasets for different minimum support levels and different widths of stratum. In each figure, SR represents the result of simple random sampling.

For BMSPOS (Fig. 6.3), the accuracy of stratified sampling method increases while w increases. When $w = 5$, the accuracy of stratified sampling is slightly higher than that of simple random sampling. It can be seen that f_p increases as w increases and f_n decreases when w increases. The f_p value of stratified sampling is lower than that of simple random sampling and the f_n value of stratified sampling is higher than that of simple random sampling for all w values.

For the Accidents dataset (Fig. 6.4), the accuracy is very high since it is a relatively dense dataset and many transactions have similar items. The accuracy does not vary too much while w changes (less than 1% for all minimum support levels). The trends of changes in f_p and f_n while w changes are the same as that for BMSPOS. In contrast to the results for BMSPOS, the f_p values of stratified sampling for Accidents is higher than that of simple random sampling while f_n of stratified sampling is lower.

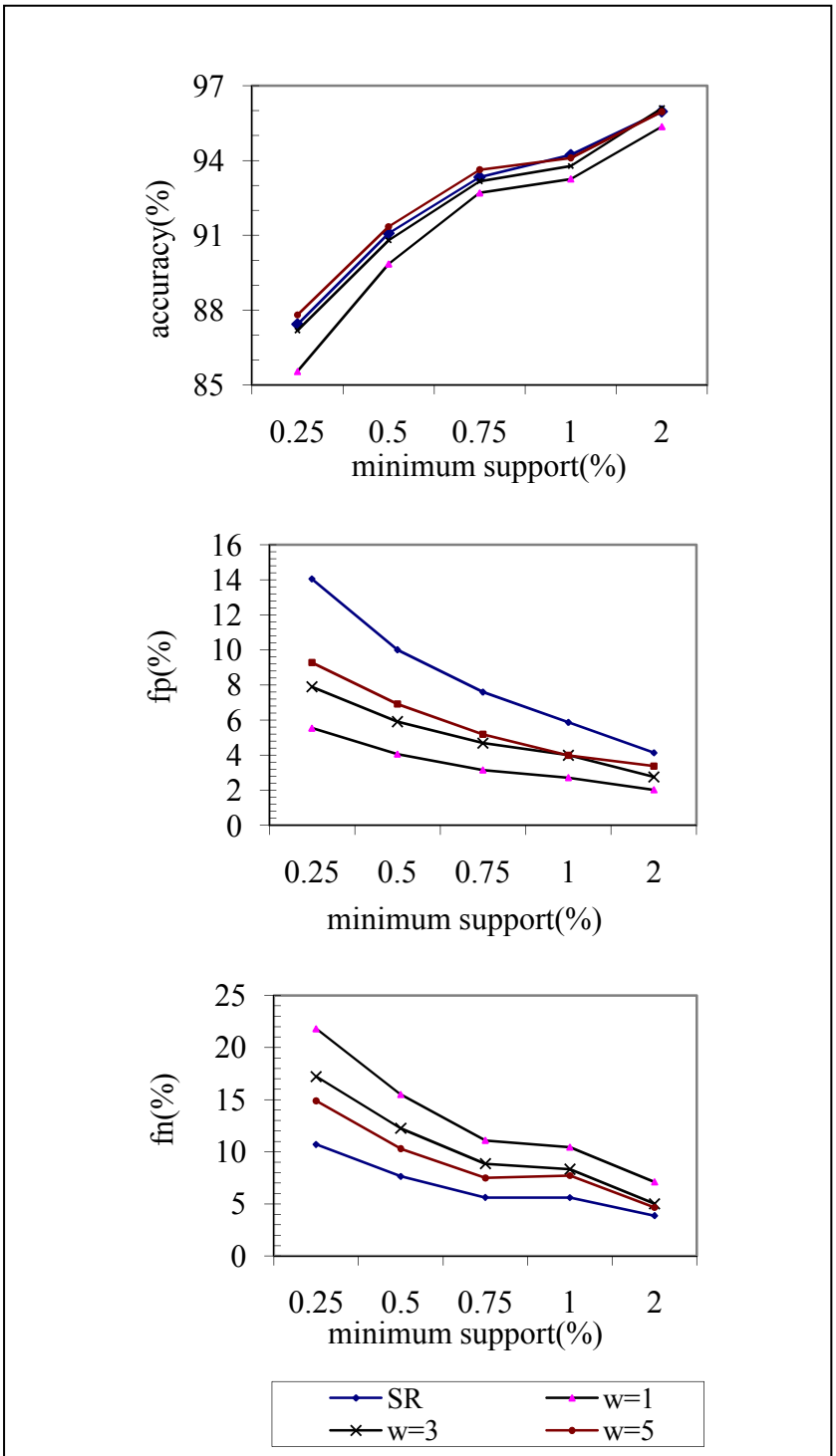


Figure 6.3 Testing Results for BMSPOS

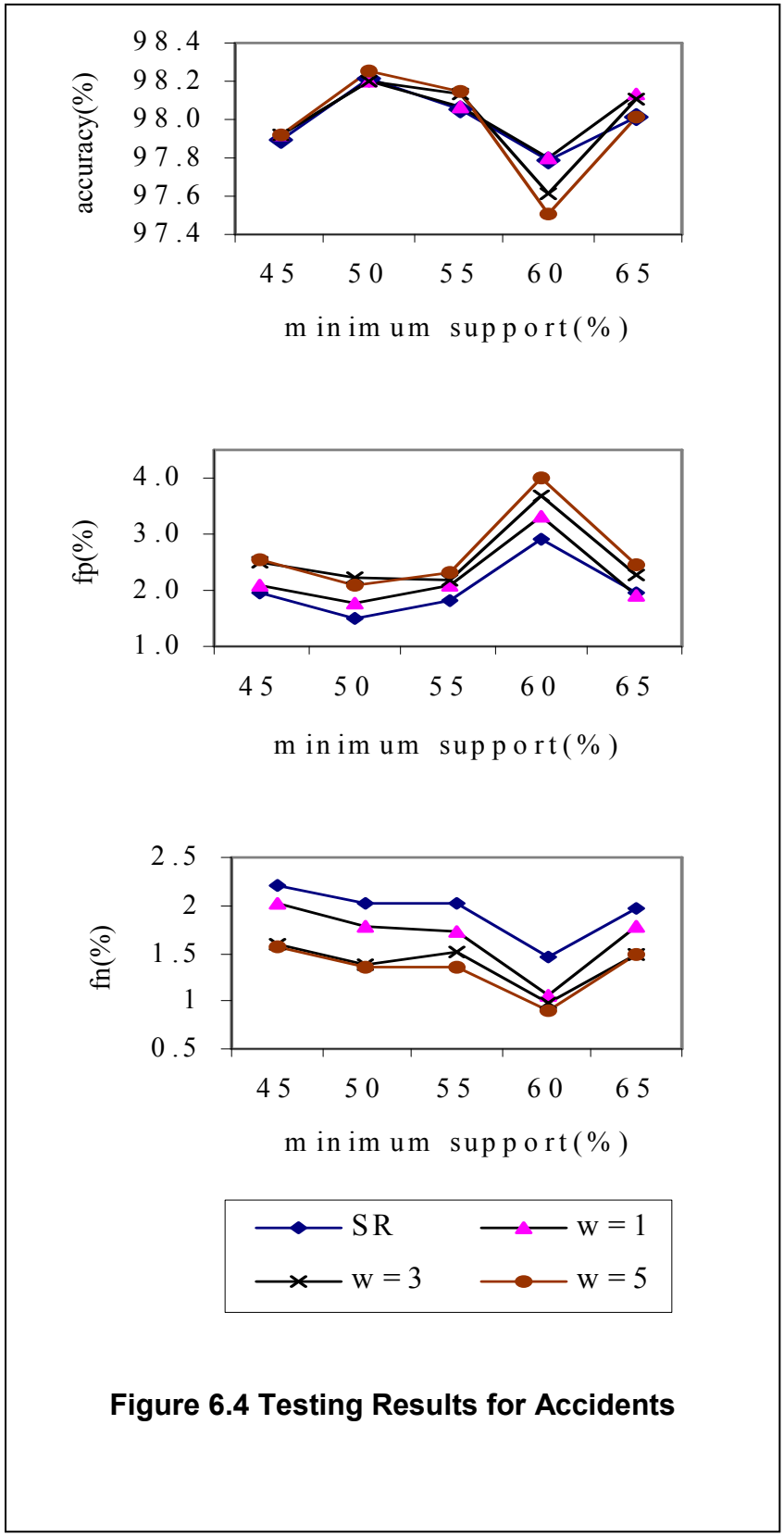


Figure 6.4 Testing Results for Accidents

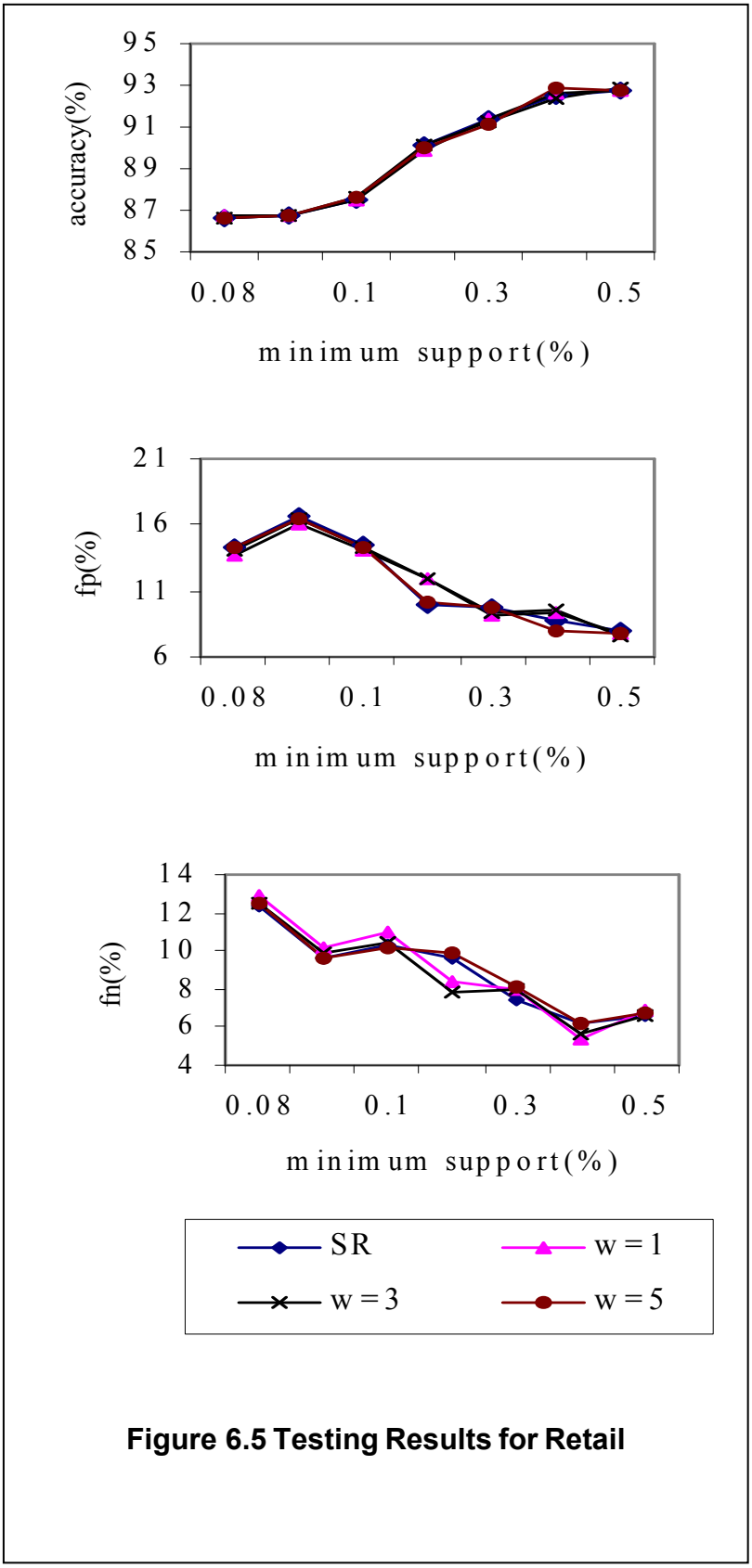


Figure 6.5 Testing Results for Retail

For Retail (Fig. 6.5), the accuracy and the errors (f_n and f_p) of stratified sampling method at different w values are almost the same as for simple random sampling. The values of f_p and f_n are not too different for most of the minimum support levels except 0.2%. It is noticed that the sampling ratio of Retail is about 6 times higher than that of BMSPOS.

Transaction size based stratified sampling method will not be any different from the simple random sampling method for a dense datasets like Connect-4 (Blake and Merz, 1998) that has all the transactions of the same size, and so belong to a single stratum.

6.3 **Summary**

Just as with other sampling methods, the proposed transaction size based stratified sampling may not suit all applications. The choice of a sampling method should depend on the characteristics of the dataset to be mined and the cost of errors in a given application. For a dataset like BMSPOS, if lower f_p is desirable, the proposed stratified sampling will be a better choice than simple random sampling. Similarly, for applications where the lower f_n is crucial, stratified sampling will perform better than the simple random sampling for a dataset like Accidents. For some datasets such as Retail, both simple random sampling and stratified sampling are suitable.

Chapter 7

Conclusion

7.1 Contributions

In this thesis, developing effective and efficient methods for clustering and association rules mining of very large transactional databases are of focus.

A new similarity measure that is more suitable for clustering of transactional data than Rao's Coefficient, Simple Matching Coefficient and Jacard Coefficient is defined. An incremental clustering algorithm INCLUS is developed using the newly defined similarity measure. INCLUS is empirically proved to be scalable and more accurate than CLOPE (Yang *et al.*, 2002) and LargeItem (Wang *et al.*, 1999).

The equal-width time window model and the elastic time window model are defined in order to evaluate changes in data streams. The width of a window in the equal-width time window model is determined by the minimum granularity with respect to an application. By doing so, it is possible to perform cluster analysis on the minimum granularity or coarser granularities. The width of an elastic time window model is initially set to the minimum granularity and subsequently resized based on the clustering changes in the data stream. Fewer clustering snapshots need to be stored on disk under the elastic window model, thereby improving efficiency and reducing disk space usage, and yet the changes at coarser granularities can still be estimated.

Data stream clustering algorithms CluStream_EQ and CluStream_EL are developed by incorporating INCLUS and the new window models under the same framework as CLuStream (Aggarwal *et al.*, 2003). The online

components for these new algorithms are empirically shown to be scalable and effective.

Sampling techniques that can improve the efficiency of mining association rules in very large databases are studied. The sample size is derived based on binomial distribution and the central limit theorem, which is smaller than that based on Chernoff Bounds (Toivonen, 1996) but still provides the same approximation guarantees. The accuracy of the proposed sampling approach is theoretically analyzed and its effectiveness is experimentally evaluated on both dense and sparse datasets. The experimental results proves that the sampling method is effective.

Applications of stratified sampling for association rules mining is also explored in this thesis. The database is first partitioned into strata based on the length of transactions and simple random sampling is then performed on each stratum. The total sample size is determined by a formula and each stratum is proportionately sampled based on its size. Experimental results show that the accuracy of transaction size based stratified sampling is very close to that of random sampling. The errors of stratified sampling can be slightly bigger or smaller than that for the random sampling for different datasets and different support thresholds. Therefore stratified sampling can be seen as an alternative option for particular datasets. In fact, when all transactions have the same number of items, stratified sampling becomes simple random sampling as all the transactions will be in the same strata.

7.2 Future Directions

The data stream clustering algorithms developed in this thesis, are based on the assumption that the processing rate of the online component is fast enough to handle the incoming data stream. A future direction of this research would be to improve the online component for handling data streams where the rate of flow is faster than the rate at which it can be processed. Sampling can be one of the techniques to be used for this purpose. Furthermore, the offline component can be tuned and implemented for discovering changes in the data streams.

One focus of this thesis, like most of the research on association rules mining, is on the critical step of frequent pattern generation. As mentioned in Chapter 2, the number of rules generated is exponential to the number of frequent items. For example, ten items can produce more than fifty thousand rules. It can be overwhelming for users seeking valuable information among such a large number of rules generated. Hence another focus for future research could be improving the efficiency of rule generation, presentation and filtering as well as the usability of the rules generated.

When sampling techniques were applied to association rules mining, it was noticed that by changing the support threshold if false positives decrease then false negatives increase, and vice versa. More investigation is needed into the relationship between these errors and methods for controlling them.

The open question for stratified sampling is how to partition the dataset so that each stratum has similar properties in relation to the association rules mining problem. It is conjectured that the accuracy of stratified sampling can be improved if the stratification scheme is based on the similarity of transactions, i.e., the number of common items between transactions. This needs further study. There is also scope for matching strata definitions with dataset characteristics to improve the accuracy and efficiency of sampling based association rules mining.

References

"Frequent Itemset Mining Dataset Repository," (http://fimi.cs.helsinki.fi/data/).

Aggarwal, C. C., Han, J., Wang, J., and Yu, P. S. (**2003**). "A Framework for Clustering Evolving Data Streams," in *Proceedings of the 29th VLDB Conference*, pp. 81-92.

Agrawal, R., Imielinski, T., and Swami, A. (**1993**). "Mining association rules between sets of items in large databases," in *Proceedings of the ACM SIGMOD Conference on Management of Data*, pp. 207-216.

Agrawal, R., and Srikant, R. (**1994**). "Fast Algorithms for Mining Association Rules," in *Proceedings of the 20th VLDB Conference*, pp. 487-499.

Alan, N., and Spencer, J. H. (**1992**). *The Probabilistic Method* (John Wiley Inc., New York).

Babcock, B., Babu, S., Datar, M., Motwani, R., and Widom, J. (**2002**). "Models and Issues in Data Stream Systems," in *Proceedings of the 21st ACM Symposium on Principles of Database Systems (PODS 2002)*, pp. 1-16.

Babcock, B., Datar, M., Motwani, R., and O'Callaghan, L. (**2003**). "Maintaining Variance and k-Medians over Data Stream Windows," in *Proceedings of the 2003 ACM Symp. on Principles of Database Systems (PODS 2003)*.

Berkhin, P. (**2002**). "Survey Of Clustering Data Mining Techniques," (Accrue Software).

Blake, C. L., and Merz, C. J. (**1998**). "UCI Repository of Machine Learning Databases," (Irvine, CA: University of California, Department of Information and Computer Science).

Brijs, T., Swinnen, G., Vanhoof, K., and Wets, G. (**1999**). "Using Association Rules for Product Assortment Decisions: A Case Study," in *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 254-260.

Chen, B., Haas, P., and Scheuermann, P. (**2002**). "A New Two Phase Sampling Based Algorithm for Discovering Association Rules," in *Proceedings of the SIGKDD '02*, pp. 462-468.

Cheung, W., and Zaïane, O. R. (**2003**). " Incremental Mining of Frequent Patterns Without Candidate Generation or Support Constraint," in

Proceedings of the Seventh International Database Engineering and Applications Symposium (IDEAS 2003), pp. 111-116.

Dong, G., Han, J., Lakshmanan, L. V. S., Pei, J., Wang, H., and Yu., P. S. (**2003**). "Online mining of changes from data streams: Research problems and preliminary results," in *In ACM SIGMOD MPDS*.

Estivill-Castro, V. (**2002**). "Why So Many Clustering Algorithm: A Position Paper," SIGKDD Explorations **4**, 65-75.

Everitt, B. S. (**1993**). *Cluster Analysis* (Halsted Press, Now York).

Fayyad, U., and Uthurusamy, R. (**2002**). "Evolving Data Into Mining Solutions For Insights," Communications Of The ACM **45**, 28-31.

Fayyad, U., Piatetsky-Shapiro, G., Smyth, P., Uthurusamy, R. (**1996**). *Advances in Knowledge Discovery and Data Mining* (AAAI Press).

Gaber, M. M., Zaslavsky, A., and Krishnaswamy, S. (**2005**). "Mining Data Streams: a Review," ACM SIGMOD Record **34**, 18-26.

Giannella, C., Han, J., Pei, J., Yan, X., and Yu, P. S. (**2003**). "Mining Frequent Patterns in Data Streams at Multiple Time Granularities," in *Next Generation Data Mining*, pp. 191-212.

Gopalan, R. P., and Sucahyo, Y. G. (**2003**). "Fast Frequent Itemset Mining using Compressed Data Representation," in *Proceedings of the IASTED International Conference on Databases and Applications (DBA '2003)*, pp. 378-373.

Guha, S., Meyerson, A., Mishra, N., Motwani, R., and O'Callaghan, L. (**2003**). "Clustering Data Streams: Theory and Practice," TKDE special issue on clustering **15**, 515-528.

Guha, S., Mishra, N., Motwani, R., and O'Callaghan, L. (**2000a**). "Clustering Data Streams," in *Proceedings of the FOCS 2000*, pp. 359-366.

Guha, S., Rastogi, R., and Shim, K. (**2000b**). "ROCK: A Robust Clustering Algorithm for Categorical Attributes," Information Systems **25**, 345-366.

Han, J., and Kamber, M. (**2000**). *Data Mining: Concepts and Techniques* (Morgan Kaufmann Publishers).

Han, J., and Kamber, M. (**2006**). *Data Mining: Concepts and Techniques* (Morgan Kaufmann Publishers).

Han, J., Pei, J., and Yin, Y. (**2000**). "Mining Frequent Patterns without Candidate Generation," in *Proceedings of the International Conference on Management of Data (SIGMOD'00)*, pp. 1-12.

Hruschka, E. R., Campello, R. J. G. B., Freitas, A. A., and de Carvalho, A. C. P. L. F. (**2009**). "A Survey of Evolutionary Algorithms for Clustering," IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews **39**, 133-155.

Huang, Z. (**1997a**). "Clustering Large Datasets with Mixed Numeric and Categorical Values," in *Proceedings of the 1st Pacific-Asia Conference on Knowledge Discovery & Data Mining*, pp. 21-34.

Huang, Z. (**1997b**). "A Fast Clustering Algorithm to Cluster Very Large Categorical Datasets in Data Mining," in *Proceedings of the Workshop on Research Issues on Data Mining and Knowledge Discovery*, pp. 21-34.

Jain, A. K., Murty, M. N., and Flynn, P. J. (**1999**). "Data Clustering: a Review," ACM Computing Surveys **31**, 264-323.

John, G. H., and Langley, P. (**1996**). "Static versus Dynamic Sampling for Data Mining," in *Proceedings of the the Second International Conference on Knowledge Discovery and Data Mining (KDD'96)*, pp. 367-370.

Kantardzic, M. (**2002**). *Data Mining : Concepts, Models, Methods and Algorithms* (Wiley, New York).

Li, Y., and Gopalan, R. P. (**2004**). "Effective Sampling for Mining Association Rules," in *Proceedings of the 17th Australian Joint Conference on Artificial Intelligence*, pp. 391-401.

Li, Y., and Gopalan, R. P. (**2005**). "Stratified Sampling for Association Rules Mining," in *Proceedings of the Second IFIP Conference on Artificial Intelligence Applications and Innovations (IPIF AIAI2005)*, pp. 79-88.

Li, Y., and Gopalan, R. P. (**2006a**). "Clustering High Dimensional Sparse Transactional Data with Constraints," in *Proceedings of the IEEE International Conference on Granular Computing(IEEE-GrC 2006)*, pp. 692 - 695.

Li, Y., and Gopalan, R. P. (**2006b**). "Clustering Transactional Data Streams," in *Proceedings of the 19th Australian Joint Conference on Artificial Intelligence*, pp. 1069-1073.

Liu, C. L. (**1968**). *Introduction to Combinatorial Mathematics* (McGraw-Hill, New York).

MacQueen, J. B. (**1967**). "Some Methods for Classification and Analysis of Multivariate Observations," in *Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability*, pp. 281-297.

Mannila, H., Toivonen, H., and Verkamo, I. (**1994**). "Efficient Algorithms for Discovering Association Rules," in *Proceedings of the AAAI Workshop on Knowledge Discovery in Databases (KDD-94)*, pp. 181-192.

Mendenhall, W., and Sincich, T. (**1992**). *Statistics for Engineering and Sciences* (Dellen Publishing Company, San Francisco).

O'Callaghan, L., Mishra, N., Meyerson, A., Guha, S., and Motwani, R. (**2002**). "Streaming-Data Algorithms For High-Quality Clustering," in *Proceedings of the IEEE International Conference on Data Engineering (ICDE02)*, p. 685.

Ong, K.-L., Li, W., Ng, W.-K., and Lim, E.-P. (**2004**). "SCLOPE: An Algorithm for Clustering Data Streams of Categorical Attributes," in *Proceedings of the 6th International Conference on Data Warehousing and Knowledge Discovery(DaWaK 2004)*, pp. 209-218.

Parthasarathy, S. (**2002**). "Efficient Progressive Sampling for Association Rules," in *Proceedings of the IEEE International Conference on Data Mining*, pp. 354-361.

Porter, J. (**1998**). "Disk Drives' Evolution," in *100th Anniversary Conference: Magnetic Recording and Information Storage* (Santa Clara University).

Provost, F., Jensen, D., and Oates, T. (**1999**). "Efficient Progressive Sampling," in *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 23-32.

Thomson, S. K. (**1992**). *Sampling* (John Wiley & Sons Inc.).

Toivonen, H. (**1996**). "Sampling Large Databases for Association Rules," in *Proceedings of the 22th International Conference on Very Large Databases (VLDB'96)*, pp. 134-145.

Walpole, R. E., Myers, R. H., and Myers, S. L. (**1998**). *Probability and Statistics for Engineers and Scientist* (Prentice hall interantional, INC, New Jersey).

Wang, J., and Karypis, G. (**2004**). "SUMMARY: Efficiently Summarizing Transactions for Clustering," in *Proceedings of the Fourth IEEE International Conference on Data Mining(ICDM 2004)*, pp. 241- 248.

Wang, K., Xu, C., and Liu, B. (**1999**). "Clustering Transactions Using Large Items," in *Proceedings of the ACM CIKM International Conference on Information and Knowledge Management*, pp. 483-490.

Yang, Y., Guan, X., and You, J. (**2002**). "CLOPE: A Fast and Effective Clustering Algorithm for Transactional Data," in *Proceedings of the KDD'02*, pp. 682-687.

Zaki, M. J., Parthasarathy, S., Li, W., and Ogihara, M. (**1997**). "Evaluation of Sampling for Data Mining of Association Rules," in *Proceedings of the 7th International Workshop on Research Issues in Data Engineering (RIDE '97) High Performance Database Management for Large-Scale Applications*, pp. 42-50.

Zhang, T., Ramakrishnan, R., and Livny, M. (**1996**). "BIRCH: An Efficient Data Clustering Method for Very Large Databases," in *Proceedings of the International Conference on Management of Data(SIGMOD96)*, pp. 103-114.

Zheng, Z., Kohavi, R., and Mason, L. (**2001**). "Real World Performance of Association Rule Algorithms," in *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 401-406.