Knowl Inf Syst (2008) 17:99–120
DOI 10.1007/s10115-007-0113-3

REGULAR PAPER

Protecting business intelligence and customer privacy while outsourcing data mining tasks

Ling Qiu · Yingjiu Li · Xintao Wu

Received: 7 February 2007 / Revised: 4 July 2007 / Accepted: 20 September 2007 /
Published online: 16 November 2007

Abstract Nowadays data mining plays an important role in decision making. Since many organizations do not possess the in-house expertise of data mining, it is beneficial to outsource data mining tasks to external service providers. However, most organizations hesitate to do so due to the concern of loss of business intelligence and customer privacy. In this paper, we present a Bloom filter based solution to enable organizations to outsource their tasks of mining association rules, at the same time, protect their business intelligence and customer privacy. Our approach can achieve high precision in data mining by trading-off the storage requirement.

1 Introduction

1.1 Background and motivation

The pervasive impact of business computing has made information technology (IT) an indispensable part of daily operations and the key to success for organizations. Many organizations

This research was supported by the USA National Science Foundation Grants CCR-0310974 and IIS-0546027.

L. Qiu (✉)
School of Mathematics, Physics and Information Technology, James Cook University, Townsville, QLD 4811, Australia
e-mail: ling.qiu@jcu.edu.au

Y. Li
School of Information Systems, Singapore Management University, Singapore 178902, Singapore
e-mail: yjli@smu.edu.sg

X. Wu
Department of Software and Information Systems, University of North Carolina at Charlotte, Charlotte, NC 28223, USA
e-mail: xwu@uncc.edu

have accumulated large amount of data from various channels in today's digitalized age. It is important to make these data available for decision making. Data mining, as one of the IT services needed by organizations, provides such a technique for the exploration and analysis of these raw data so as to reveal hidden information and knowledge; it has also been realized as an important way for discovering knowledge from the data and converting "data rich" to "knowledge rich" so as to assist strategic decision making. The benefits of using data mining for decision making have been demonstrated in various industries and governmental sectors [7], e.g., banking, insurance, direct-mail marketing, telecommunications, retails, and health care [32]. Among all of the available data mining methods, the discovery of associations between business events or transactions is one of the most commonly used techniques. Association rule mining has been an important application in decision support and marketing strategy [25] for an organization.

Let us consider a typical application scenario as follows. In an organization (e.g., an enterprise or a governmental sector), there are several divisions including an IT division which provides IT services for the whole organization. A functional division may have to delegate or outsource its data mining tasks to the IT division due to the lack of IT expertise and powerful computing infrastructure which are usually centrally managed by the IT division.

This scenario can be extended to a more general circumstance in which all divisions are individually independent organizations (or companies). This is because in today's fast-paced business environment, it is impossible for any single organization to understand, develop, and implement every IT needed. By outsourcing, an organization can obtain specific human resources (e.g., skilled programming personnel) and technological resources (e.g., more powerful computing infrastructure) for its needs of IT services (e.g., data analysis) with lower costs [12]. It can also be extended to online scenarios, e.g., a distributed computing environment comprising of a center server and some edge servers.

The practice of outsourcing data mining tasks involves extensive collaboration (e.g., exchange or share of data) across different organizations. Either the raw data or the revealed information after analysis contains the business intelligence (BI) and customer privacy of an organization. There is a security concern of potential risk of exposing private information in outsourcing activities [28]. Without proper security policy and technology, these privacies could be very vulnerable to security breaches. Therefore, to protect BI and customer privacy, it is urgent and critical to provide solutions from the perspectives of both legality or regulation and technology [27]. In this paper, we focus on the technology-based solutions. When outsourcing mining tasks,[1] we should protect the following three elements which may expose BI and customer privacy: (1) the source data which contain all transactions and items; (2) the mining requests which are itemsets of interests; and (3) the mining results which are frequent itemsets and association rules.

There are various methods proposed to preserve privacy in data mining; but they cannot protect all three elements simultaneously. This is because with these methods, when a first party[2] outsources its mining tasks to a third party,[3] it has to provide the source database (which might be someway encrypted) together with some additional information (e.g., plain text of mining requests) without which the mining tasks may not be carried out. Given this situation, the existing methods are unable to efficiently prevent the exposure of private infor-

[1] Without further specification, we always refer to association rule mining tasks.

[2] This is the party that outsouces its data mining tasks. It may be a functional division of an organization or the center server in a distributed environment with client–server architecture.

[3] This is the party that is authorized by the first party to undertake the outsourced data mining tasks. It may be the IT division of an organization or an edge server in a distributed environment with client–server architecture.

mation to the third party, or unable to prevent the third party from (either intentionally or unintentionally) deciphering (and possibly spreading) further information from the mining results (which would be sent back to the first party) with the additional information.

1.2 Overview of our paper

1.2.1 Our solutions

Given the situation discussed in the previous paragraphs, to protect BI and customer privacy when outsourcing data mining tasks, we should control the direct access to the original data. In other words, the third party should not be allowed to access the original transactional data while performing mining tasks, and should not be able to interpret the mining results.

In this paper, we present a Bloom filter based approach which provides an algorithm for privacy preserving association rule mining with computation efficiency and predictable (controllable) analysis precision. The Bloom filter [10] is a stream (or a vector) of binary bits. It is a computationally efficient and irreversible coding scheme that can represent a set of objects while preserving privacy of the objects. With our approach, firstly the source data are converted to Bloom filter representation and handed over to a third party together with mining algorithms. Then the first party sends its mining requests to the third party. Mining requests are actually candidates of frequent itemsets which are also represented by Bloom filters. Lastly, the third party runs the mining algorithms with source data and mining requests, and comes out the mining results which are frequent itemsets or association rules represented by Bloom filters. In the above mining process, what the first party exposes to the third party does not violate privacy [22]; that is, the third party would not be able to distill down private information from Bloom filters. Therefore, all three elements aforementioned are fully protected by Bloom filters.

The goal of protecting BI and customer privacy during outsoucing can be achieved by Bloom filter because it satisfies simultaneously the following three conditions. First, transactions containing different numbers of items are mapped to Bloom filters with the same length. This prevents an adversary from deciphering the compositions of transactions by analyzing the lengths of transactions. Second, Bloom filters support membership queries. This allows an authorized third party to carry out data mining tasks with *only* Bloom filters (i.e., Bloom filters of either transactions or candidates of frequent itemsets). Third, without knowing all possible individual items in the transactions, it is difficult to identify what items are included in the Bloom filter of a transaction by counting the numbers of 1's and 0's. This is because the probability of a bit in a Bloom filter being 1 or 0 is 0.5 given that the parameters of the Bloom filter are optimally chosen (see Sect. 3 for details).

1.2.2 Main contributions

A condensed (5-page) version of our study has been published in [34]. In this version, we present in details the mathematical analysis model for problem formulation and error rate estimation, the analysis of the mining algorithm, and the results and analysis of experiments.

In brief, our main contributions include the following:

- We propose an approach allowing to outsource data mining tasks while protecting BI and customer privacy. It prevents third parties from interpreting customer privacy from the source data and from deciphering BI from the mining results.
- We present solid theoretical analysis for our approach. Our analysis shows that a tradeoff can be made between storage requirement and mining precision.

- We test the proposed approach rigorously on both real and synthetic datasets. Our experimental results show that our approach is both effective and flexible for practical usage.

1.2.3 Organization of the paper

The remaining sections are organized as follows. We review the related work in Sect. 2. We revisit the basics of Bloom filters and formulate our data mining problem in Sect. 3. In Sect. 4, we present theoretical analysis for the formulated problems and estimate the errors in data mining. We also present a method of using multiple groups of Bloom filters to further reduce data mining errors. Following that in Sect. 5, we develop a multi-phased algorithm and conduct a set of empirical simulations to verify our approach. Lastly we conclude our paper in Sect. 6 with a discussion of contribution, limitation, and future research directions.

2 Literature review

Association rule mining has been an active research area since its introduction [2]. Many algorithms have been proposed to improve the performance of mining association rules or frequent itemsets. An interesting direction is the development of techniques that incorporate privacy concerns.

One type of these techniques is perturbation based, which perturbs the data to a certain degree before data mining so that the real values of sensitive data are obscured while non-sensitive statistics on the collection of data are preserved. In an early work [5] a perturbation-based approach was proposed for decision tree learning. This approach was further studied in [1,19,23,24]. Some recent work [6,8,9,11,14,15,29–31,36,37,39] investigates the tradeoff between leakage of private information and accuracy of mining results. In [8,11], the authors considered the problem of limiting disclosure of sensitive rules, aiming at selectively hiding some frequent itemsets from large databases with as little impact on other, non-sensitive frequent itemsets as possible. The idea is to modify a given database so that the support of a given set of sensitive rules decreases below a predetermined threshold. Similarly, in [37] a method is presented for selectively replacing individual values with unknowns from a database to prevent the discovery of a set of rules, while minimizing the side effects on non-sensitive rules. In [9,31], the authors studied the impact of hiding strategies on an original data set by quantifying how much information is preserved after sanitizing the data set. In [6,15,36], researchers also studied the problem of mining association rules from transactions in which the data has been randomized to preserve the privacy of individual transactions. One problem of perturbation-based approach is that it may introduce some false association rules. Another drawback of this approach is that it cannot always fully preserve data privacy while achieving high mining precision [23,24].

The second type is distributed privacy preserving data mining [13,21,26,33,38] based on secure multi-party computation [40]. Though this approach can preserve privacy, it works only in distributed environment (with several parties to collaborate in mining process) and needs sophisticated protocols (secure multi-party computation based), which makes it infeasible for our scenario.

Both types of techniques are designed to protect privacy by masquerading the source data, and cannot protect privacy distillable from the mining requests and results accessible to data miners.

Related, but not directly relevant to our work, is the research in outsourced databases [3,16–18,20] and most recently in data confidentiality [35]. Hacigumus et al. [16–18,20]

explored a new paradigm for data management in which a third party service provides host database as a service. They proposed several encryption techniques to process as many queries as possible at the service providers' site without having to decrypt the data. Agrawal et al. [3] presented an order-preserving encryption scheme for numeric data that allows comparison operations to be directly applied on encrypted data. However, encryption is time consuming and it may require auxiliary indices. It is only designed for certain type of queries but may not be suitable for complex tasks such as association rule mining. Most recently Raś et al. [35] presented a generalized strategy to reduce a disclosure risk of confidential data by hiding some attributes of the source data. Hiding (or replacing) some attributes may break the integrity of source data and thus the mining results may not be meaningful if it is applied to our application scenario.

3 Problem formulation

Our research question is how to outsource the association rule data ming tasks, at the same time, protect BI and customer privacy. We propose a Bloom filter based approach with which an authorized third party (e.g., an edge server or an IT division as mentioned earlier) will perform the association rule mining based on the dataset transformed by Bloom Filters and report the frequent itemsets with controllable error boundary.

In this section, we first briefly review the basics of Bloom filter, and then present the mathematic formalization for our problem.

3.1 Bloom filter revisited

A Bloom filter is a simple, space-efficient, randomized data structure for representing a set of objects so as to support membership queries.

Definition 3.1 Given an n-element set $S = \{s_1, \ldots, s_n\}$ and k hash functions $h_1, \ldots, h_k$ of range m, the Bloom filter of S, denoted as $B(S)$, is a binary vector of length m that is constructed by the following steps: (i) every bit is initially set to 0; (ii) every element $s \in S$ is hashed into the bit vector through the k hash functions, and the corresponding bits $h_i(s)$ are set to 1.[4] A Bloom filter function, denoted as $B(\cdot)$, is a mapping from a set (not necessarily n-element set) to its Bloom filter.

For membership queries, i.e., whether an item $x \in S$, we hash x to the Bloom filter of S (through those hash functions) and check whether all $h_i(x)$ are 1's. If not, then clearly x is not a member of S. If yes, we say x is in S although this could be wrong with some probability.

Definition 3.2 For an element s and a set S, define $s \in_B S$ if s hashes to all 1's in the Bloom filter of S, and $s \notin_B S$ otherwise. The false positive rate of the Bloom filter of S is defined as the probability of $s \in_B S$ while $s \notin S$, or $\Pr(s \in_B S \mid s \notin S)$.

Assuming that all hash functions are perfectly random, we have the following

Lemma 3.3 *Given an n-element set $S = \{s_1, \ldots, s_n\}$ and its Bloom filter $B(S)$ of length m constructed from k hash functions, the probability for a specific bit in $B(S)$ being 0 is*

$$p_0 = \left(1 - \frac{1}{m}\right)^{kn} \approx \mathrm{e}^{-kn/m}$$

[4] A location can be set to 1 multiple times, but only the first change has an effect.

and the probability for a specific bit being 1 *is*

$$p_1 = 1 - p_0 \approx 1 - \mathrm{e}^{-kn/m}$$

Then the false positive rate of $B(S)$ *is*

$$f = p_1^k \approx \left(1 - \mathrm{e}^{-kn/m}\right)^k \tag{1}$$

Given n *and* m, f *is minimized when* $p_0 = p_1 = 0.5$ *and* $k = \frac{m}{n} \ln 2$, *in which case* $f = 1/2^k = (0.6185)^{m/n}$.

What's missing from traditional Bloom filter? Traditional Bloom filters cannot solve the privacy issue because anybody knowing the hash function can derive the original itemsets based on Bloom Filters.

To preserve the privacy of Bloom filters, we propose *keyed Bloom filter* by augmenting the hash functions h_i with a secret key K. To represent set S, an element $s \in S$ is inserted into Bloom filter B by setting the corresponding bits $h_i(s \circ K)$ in B to 1, where $\circ$ represents concatenation. Also, to query whether an item $x \in S$, we check whether all $h_i(x \circ K)$ bits are set to 1. Without knowing the secret key, one is unable to derive the original set by examining a Bloom filter. Without further mention we always assume that Bloom filters in our paper are constructed with secret keys.

3.2 Our problem

Fundamentally we want to provide a solution for privacy preserving frequent itemsets mining. The frequent itemset mining has been a common task in many data mining projects for the past decade. From frequent itemsets, one can easily derive all association rules. The mining of frequent itemsets of association rules has a wide range of applications in many areas, from the analysis of customer preferences to DNA patterns.

For market basket data, we define each transaction, such as a list of items purchased, as a subset of all possible items.

Definition 3.4 Let $\mathcal{I} = \{I_1, \ldots, I_d\}$ be a set of d boolean variables called items. Let database D be a set of transactions $T_1, T_2, \ldots, T_N$ where each transaction T_i is a set of items such that $T_i \subseteq \mathcal{I}$. The support of an itemset S over $\mathcal{I}$, denoted $support(S)$, is defined as the number of the transactions that contain S. The frequency of an itemset S, denoted $\mathrm{freq}(S)$, is defined as $support(S)/N$.

Problem 1 ***Traditional problem: frequent itemsets mining.*** *Mathematically, given a transaction database* $\mathcal{D}$ *over* $\mathcal{I}$ *and a threshold* $\tau \in [0, 1]$, *traditional research focuses on finding all frequent itemsets* $FS \in 2^{\mathcal{I}}$ *such that* $\mathrm{freq}(FS) \geq \tau$.

Our idea is to transform transaction database to a collection of Bloom filters to preserve the privacy in frequent itemset mining. Each transaction $T_i \in \mathcal{T}$ is transformed to Bloom filter $B(T_i)$ of size m using k hashed functions. To preserve the privacy of items I_i, we assume that the mining process is done on the (keyed) Bloom filters $B(I_i)$ of the items rather than on the items themselves.

Problem 2 ***Our research problem: privacy preserving frequent itemsets mining.*** *Given* (i) *a collection of Bloom filters* $\{B(T_1), \ldots, B(T_N)\}$ *for transaction database* $\mathcal{D}$ *over* $\mathcal{I}$, (ii) *a set of Bloom filters* $\{B(I_1), \ldots, B(I_d)\}$ *for items in* $\mathcal{I}$, *and* (iii) *a threshold* $\tau \in [0, 1]$, *find all Bloom filters* $B(FS)$ *of itemsets* $FS \in 2^{\mathcal{I}}$ *such that* $\mathrm{freq}(FS) \geq \tau$.

Normally without knowing the secret key, the third party, which may not be fully trusted, will be unable to interpret sensitive and private information either from the contents of databases or from mining results. However, we still need to handle some extreme cases for the protection of BI and customer privacy. Case 1, some transactions may contain only one item. The number of 1's in these Bloom filters are no more than but very close to k. Therefore, the outsourced database may divulge partial individual items. To prevent such divulgence, one of the solutions is to insert several virtual items as white noise to those transactions in which item numbers are smaller than a threshold. Case 2, candidates of frequent 1-itemsets are exactly individual items which may divulge some sensitive information. To prevent this, we extend the usage of secret key by inserting several virtual items, denoted by $k_1, k_2, \ldots, k_i$, into all transactions before outsoucing. At the same time, each candidate of frequent 1-itemsets is inserted with a virtual item randomly chosen from k_1 to k_i and is sent to the third party together with mining requests (see Sect. 5.1 for detailed mining process). Thus the edge servers cannot easily identify candidates of frequent 1- and 2-itemsets because both types of candidates look alike. This method can be applied to conceal candidates of frequent 2-, 3-, …, and k-itemsets. Moreover, this operation can be done before or after the mixing of white noise discussed in case 1.

4 Analysis

In this section, we analyze the possible error rates introduced by mining the frequent itemsets based on Bloom filters instead of the original dataset. For any given itemset, the frequency learnt from Bloom filters may be larger than its real frequency learnt from original transactions due to the false positive of Bloom filters. We make this clear in the following analysis. By default, we assume that for any itemset, there is a Bloom filter function $B(\cdot)$ which produces binary vector of length m through k hash functions.

4.1 Preliminaries

The false positive of a Bloom filter was defined for checking an element from a Bloom filter. Now we extend the concept of false positive to checking an itemset from a Bloom filter.

Definition 4.1 Given an itemset S and a transaction T_i, define $S \subseteq_B T_i$ if for all items $s \in S$, $s \in_B T_i$, and define $S \not\subseteq_B T_i$ otherwise. The false positive rate for checking S from the Bloom filter of T_i, denoted as f_i, is defined as the probability of $S \subseteq_B T_i$ while $S \not\subseteq T_i$, or $\Pr(S \subseteq_B T_i \mid S \not\subseteq T_i)$.

Due to the false positive of checking an itemset from a Bloom filter, the support or frequency learnt from a collection of Bloom filters is different from that learnt from original transactions. Regarding such support and frequency, we have the following

Definition 4.2 Given a collection of N Bloom filters $\{B(T_1), \ldots, B(T_N)\}$ for transaction database $\mathcal{D}$ over $\mathcal{I}$, the support of an itemset $S \in 2^{\mathcal{I}}$ that is learnt from the collection of filters, denoted as $Bsupport(S)$, is defined as the number of filters $B(T_i)$ that satisfy $S \subseteq_B T_i$. The frequency of S that is learnt from the collection of filters, denoted as $B\text{freq}(S)$, is defined as $Bsupport(S)/N$.

Lemma 4.3 *In the setting of Definition 4.2, the following statements hold*: (i) $S \subseteq_B T_i$ *iff* $B(S) \wedge B(T_i) = B(S)$, *where* $\wedge$ *is bitwise AND.* (ii) *If* $S \subseteq T_i$, *then* $S \subseteq_B T_i$. (iii)

If $S \not\subseteq T_i$, *then* $S \subseteq_B T_i$ *with probability* f_i, *and* $S \not\subseteq_B T_i$ *with probability* $1 - f_i$. (iv) $B\text{freq}(S) \geq \text{freq}(S)$.

Theorem 4.4 *Given an itemset* S *and a transaction* T_i, *the false positive rate of checking* S *from the Bloom filter of* T_i *is*

$$f_i = \left(1 - \mathrm{e}^{-kn_i/m}\right)^{||B(S-T_i)||}$$

where $n_i = |T_i|$ *is the length of transaction* T_i *in terms of the number of items, and* $||\cdot||$ *indicates the number of* 1*'s in a binary vector.*

Proof From Eq. (1), one can derive that the false positive rate for checking any single item $s \in S - T_i$ is p_1^k, where $p_1 = \left(1 - \mathrm{e}^{-kn_i/m}\right)$ is the probability that a specific bit is 1 in $B(T_i)$ and k is the number of bits to which the item is hashed. From Lemma 4.3, we know that $S \cap T_i \subseteq_B T_i$. Since the items in $S - T_i$ are hashed to $||B(S - T_i)||$ bits all together, the false positive rate f_i for checking S from $B(T_i)$ is $p_1^{||B(S-T_i)||} = \left(1 - \mathrm{e}^{-kn_i/m}\right)^{||B(S-T_i)||}$. □

Corollary 4.5 *Given an itemset* S *and a transaction* T_i, *the false positive rate* f_i *of checking* S *from the Bloom filter of* T_i *is bounded:*

$$\left(1 - \mathrm{e}^{-kn_i/m}\right)^{||B(S)||} \leq f_i \leq \left(1 - \mathrm{e}^{-kn_i/m}\right)^k$$

Proof According to the definition of false positive, we have $S \not\subseteq T_i$ and thus $1 \leq ||B(S - T_i)|| \leq ||B(S)||$. Combining this with Theorem 4.4, we have $\left(1 - \mathrm{e}^{-kn_i/m}\right)^{||B(S)||} \leq f_i \leq \left(1 - \mathrm{e}^{-kn_i/m}\right)^k$. □

4.2 False positive and false negative

In our data mining problem, an outsourced server has access to the Bloom filters $\{B(T_i)\}$ of all transactions. However, it has no access to the original data. Therefore, given a Bloom filter $B(S)$, the server cannot compute the frequency $\text{freq}(S)$ directly. The approaches to solving the traditional frequent itemset mining problem cannot be applied directly to solving our data mining problem.

According to Lemma 4.3 (i), the frequency $B\text{freq}(S)$ can be derived from those Bloom filters. Our solution is to find all Bloom filters $B(S)$ such that $B\text{freq}(S) \geq \tau'$ where τ' is a revised threshold. Note that $\text{freq}(S) \geq \tau$ is required in our data mining problem; thus, we must ensure that $\{B(S) : B\text{freq}(S) \geq \tau'\} \approxeq \{B(S) : \text{freq}(S) \geq \tau\}$. Because the two sets are not necessarily the same, we need to define the false positive rate and false negative rate for checking an itemset from all Bloom filters.

Definition 4.6 Given an itemset S and N Bloom filters $B(T_i)$, $i = 1, \ldots, N$, the false positive rate for checking S from all Bloom filters using revised threshold $\tau' \geq \tau$, denoted as $f_{\tau'}^+$, is defined as the probability of $B\text{freq}(S) \geq \tau'$ while $\text{freq}(S) < \tau$, or $\Pr\left(B\text{freq}(S) \geq \tau' \mid \text{freq}(S) < \tau\right)$. The false negative rate for checking S from all Bloom filters using τ', denoted as $f_{\tau'}^-$, is defined as the probability of $B\text{freq}(S) < \tau'$ while $\text{freq}(S) \geq \tau$, or $\Pr\left(B\text{freq}(S) < \tau' \mid \text{freq}(S) \geq \tau\right)$.

If the revised threshold τ' is the same as the threshold τ, then the false negative rate will be zero due to the fact $B\text{freq}(S) \geq \text{freq}(S)$ (Lemma 4.3 (iv)); however, the false positive rate may be greater than zero in this case. In general, one may use $\tau' \geq \tau$ so as to balance

false negative rate and false positive rate. The higher the τ', the higher the false negative rate, and the lower the false positive rate. If the false negative rate is of major concerns, one may choose $\tau' = \tau$ to zero out false negative rate. Note that choosing $\tau' < \tau$ is meaningless as it only increases the false positive rate without decreasing the false negative rate which is zero.

To formalize our analysis, we define a random variable for checking an itemset against a Bloom filter. Then we re-write in terms of the defined variables the frequency of an itemset learnt from Bloom filters and the false positive/negative rates for checking an itemset from all Bloom filters.

Definition 4.7 For an itemset S and a transaction T_i such that $S \not\subseteq T_i$, define a random 0–1 variable e_i such that $e_i = 1$ if $S \subseteq_B T_i$ and $e_i = 0$ if $S \not\subseteq_B T_i$.

The defined variable indicates whether $S \subseteq_B T_i$; that is, $e_i = 1$ with probability f_i and $e_i = 0$ with probability $1 - f_i$ (see Lemma 4.3). In other words, e_i represents a Bernoulli trial with probabilities f_i of success and $1 - f_i$ of failure. Without loss of generality, we can assume that $S \not\subseteq T_i$ for the first $N \cdot (1 - \text{freq}(S))$ transactions T_i. Then we have

$$B\text{freq}(S) = \text{freq}(S) + \frac{1}{N} \cdot \sum_{i=1}^{N(1-\text{freq}(S))} e_i \tag{2}$$

Lemma 4.8 *Let s_e be the sum of $N \cdot (1 - \text{freq}(S))$ random 0-1 variables e_i defined for an itemset S. The false positive and false negative rates for checking S from all Bloom filters using a revised threshold $\tau' \geq \tau$ are*

$$f^+_{\tau'} = \Pr\left(s_e \geq N\left(\tau' - \text{freq}(S)\right) \mid \text{freq}(S) < \tau\right)$$
$$f^-_{\tau'} = \Pr\left(s_e < N\left(\tau' - \text{freq}(S)\right) \mid \tau \leq \text{freq}(S) < \tau'\right)$$

4.3 Estimate of false positive and false negative

We estimate the false positive and false negative rates in the case of $n_i \cong n$, where n_i is the size of each transaction, and $n = \frac{1}{N}\sum_{i=1}^{N} n_i$ is the average size of N transactions. Note that if $n_i \ncong n$, the transaction data can be clustered into multiple groups such that in each group, the size of each transaction is equal or close to the average size of the transactions in the group. The data mining task can be easily extended to each group. For simplicity, we assume that $n_i = n$ in our analysis; we leave the multi-group case in Sect. 4.4.

Let $k = \frac{m}{n} \ln 2$ be the optimal number of hash functions that are used for generating the Bloom filter of length m for each transaction, which consists of n items. According to Lemma 3.3 and Corollary 4.5, the false positive rate f_i for checking an itemset S from a Bloom filter has a lower bound and an upper bound:

$$2^{-||B(S)||} \leq f_i \leq 2^{-k} \tag{3}$$

First consider a special case where $f_i = \bar{f}$ for all $i = 1, \ldots, N$. In this case s_e is the sum of $N \cdot (1 - \text{freq}(S))$ independent Bernoulli trials with probabilities $\bar{f}$ for success and $1 - \bar{f}$ for failure. Let $b(\alpha, \beta, \bar{f}) = \frac{\beta!}{\alpha!(\beta-\alpha)!}\bar{f}^{\alpha}(1-\bar{f})^{\beta-\alpha}$ be the probability that α Bernoulli trials with probabilities $\bar{f}$ for success and $(1 - \bar{f})$ for failure result in β successes and $\beta - \alpha$ failures ($\alpha \leq \beta$). Let $\mathcal{C}(\alpha, \beta, \bar{f}) = \sum_{i=\alpha}^{\beta} b(i, \beta, \bar{f})$ be the cumulative binomial probability of having at least α successes in β trials.[5] Then the false positive and false negative rates for

[5] Function $\mathcal{C}$ is a standard function provided in many commercial software packages such as Matlab.

checking an itemset S from all Bloom filters are

$$f^+_{\tau'}(\bar{f}) = \mathcal{C}\left(N \cdot \left(\tau' - \text{freq}(S)\right),\ N \cdot (1 - \text{freq}(S)),\ \bar{f}\right), \quad \text{where } 0 < \text{freq}(S) < \tau \tag{4}$$

$$f^-_{\tau'}(\bar{f}) = 1 - \mathcal{C}\left(N \cdot \left(\tau' - \text{freq}(S)\right),\ N \cdot (1 - \text{freq}(S)),\ \bar{f}\right), \quad \text{where } \tau \le \text{freq}(S) < \tau' \tag{5}$$

Since the cumulative binomial probability $\mathcal{C}(\alpha, \beta, \bar{f})$ is monotonic increasing with $\bar{f}$, from formulae (3) to (5) it is easy to know the lower bounds and upper bounds for the false positive rate and false negative rate in general case:

$$f^+_{\tau'}\left(2^{-||B(S)||}\right) \le f^+_{\tau'} \le f^+_{\tau'}\left(2^{-k}\right), \quad \text{where } 0 < \text{freq}(S) < \tau$$

$$f^-_{\tau'}\left(2^{-k}\right) \le f^-_{\tau'} \le f^-_{\tau'}\left(2^{-||B(S)||}\right), \quad \text{where } \tau \le \text{freq}(S) < \tau'$$

In a special case where $\tau' = \tau$, we have $f^-_{\tau} = 0$ and

$$\mathcal{C}\left(A,\ B,\ 2^{-||B(S)||}\right) \le f^+_{\tau} \le \mathcal{C}\left(A,\ B,\ 2^{-k}\right) \tag{6}$$

where $A = N \cdot (\tau - \text{freq}(S))$ and $B = N \cdot (1 - \text{freq}(S))$.

Equation (6) indicates that the greater the number k of hash functions, the smaller the false positive and false negative rates. In other words, the longer the Bloom filters, the smaller the false positive and false negative rates. To further understand this, we compare $B\text{freq}(S)$ with $\text{freq}(S)$ in the average case. Let $E[\cdot]$ denote the mean of a random variable. From Eq. (2), we have

$$E\left[B\text{freq}(S)\right] = \text{freq}(S) + \frac{1}{N} \cdot E[s_e] = \text{freq}(S) + \frac{1}{N} \cdot \sum_{i=1}^{N(1-\text{freq}(S))} f_i$$

Recall the bounds for f_i (see Eq. 3). In the false positive case (where $\text{freq}(S) \le \tau$), we have

$$N \cdot (1 - \tau) \cdot 2^{-||B(S)||} \le E[s_e] \le N \cdot 2^{-k}$$

Similarly, in the false negative case (where $\tau \le \text{freq}(S) < \tau'$), we have

$$N \cdot (1 - \tau') \cdot 2^{-||B(S)||} \le E[s_e] \le N \cdot (1 - \tau) \cdot 2^{-k}$$

The above three equations imply that the average value of $B\text{freq}(S)$ is greater than $\text{freq}(S)$, but the difference is bounded. Note that $||B(S)|| \ge k$. We have

$$E\left[B\text{freq}(S)\right] - \text{freq}(S) \le (1 - \tau) \cdot 2^{-k} < 2^{-k} \tag{7}$$

The longer the Bloom filters (i.e., the greater the k), the smaller the difference between $B\text{freq}(S)$ and $\text{freq}(S)$. If the length of Bloom filters increases linearly, then the difference of frequencies decreases exponentially. For example, if $k \ge 20$, then the difference of frequencies will be less than 10^{-6}. This means that in the average case, the frequency of an itemset detected from Bloom filters will be greater than that detected from original data by at most 10^{-6}. Note that the above analysis is conducted for each itemset. The overall false positive and false negative rates depend on the distribution of itemsets and their frequencies. Empirical study will be conducted in Sect. 5.

4.4 Multiple groups of Bloom filters

The estimate of false positive and false negative rates is based on the assumption that the size of each transaction is equal or close to the average size of all transactions. This might not be true in many real data sets. A simple solution is to cluster the transactions into multiple groups such that in each group, the size of each transaction is equal or close to the average size of the transactions in the group. Across all groups, the same number k of hash functions are used for generating Bloom filters. In each group, $k = \frac{m}{n} \ln 2$ is optimized for the length m of Bloom filters and the average size n of the transactions in the group. This means that for different groups, the length of Bloom filters are different depending on the average size of transactions in the group. Roughly speaking, long Bloom filters will be used for long transactions, while short Bloom filters for short transactions.

Using different Bloom filters for different groups of transactions will not only save storage requirement but also increase the precision of data mining. Note that the bounds presented in the previous section are based on k. Since the same k is optimized across multiple groups, the bounds can be used to estimate the false positive and false negative rates in all groups.

In the case of multiple groups, the transactions are represented by the Bloom filters of different lengths. This requires that each candidate itemset be represented by Bloom filters of different lengths too. This is because in our data mining problem, the Bloom filter of each candidate itemset needs to be checked against the Bloom filters of the same length. To avoid transmitting multiple Bloom filters for each candidate itemset between client and server, only the longest Bloom filter of each candidate itemset is sent to the server at one time. In data mining process, the outsourced server needs to transform it into different lengths so as to check it against different groups of Bloom filters. We provide a simple solution called δ-folding to transform the Bloom filters.

Definition 4.9 Given an m-bit Bloom filter B, δ-folding of B, denoted as B_δ, is defined as a δm-bit vector generated from B: for $0 < i \leq \delta m$, $bit(B_\delta, i) = \text{bit}(B, i) \bigvee_{j=i \bmod \delta m}^{0<j\leq m} bit(B, j)$, where $bit(B, i)$ denotes the i^{th} bit of B, $\vee$ the bitwise OR, and $0 < \delta \leq 1$.

Example 1 Let B be a 10-bit Bloom filter. $B_{0.7}$ is defined as a 7-bit vector $B_{0.7}$: $bit(B_{0.7}, i) = bit(B, i) \vee bit(B, i + 7)$ for $i = 1, 2, 3$, and $bit(B_{0.7}, i) = bit(B, i)$ for $i = 4, 5, 6, 7$.

From the definition of Bloom filter, it is clear that vector B_δ is a Bloom filter generated with the same k hash functions which are used for generating Bloom filter B.

Given a Bloom filter $B(S)$ of longest length m of candidate itemset S, the frequency $B\text{freq}(S)$ is computed by checking $B(S)$ against the Bloom filters of transactions in all groups. For a particular group in which the Bloom filters have length $m' \leq m$ (note that m is the longest length for all groups), the server applies $\frac{m'}{m}$-folding to $B(S)$ such that the transformed Bloom filter has the length m'.

5 Experiments

To evaluate the performance of our Bloom filter based method for mining frequent itemsets, we conduct experiments based on both synthetic data and real data. A framework of our method is shown in Algorithm 1.

5.1 Algorithm

Algorithm 1 can be divided into three phases: *counting phase* (lines 3–5), *pruning phase* (lines 6–8), and *candidates generating phase* (lines 9–10) in each round ℓ, where ℓ indicates the size of each candidate itemset dealt with. In the counting phase, each candidate filter is checked against all transaction filters and the candidate's count is updated. In the pruning phase, any Bloom filter is eliminated from the candidate set if its count (i.e., Bsupport) is less than a revised threshold $N \cdot \tau'$. Finally, in the candidates generating phase, new candidate Bloom filters are generated from the Bloom filters discovered in the current round. The new candidates will be used for data mining in the next round.

Algorithm 1 Mining frequent itemsets from Bloom filters

1: $C_1 = \{B(I_1), \ldots, B(I_d)\}$ // $B(I_i)$ is the Bloom filter of item I_i
2: **for** ($\ell = 1$; $C_\ell \neq \varnothing$; ℓ++) **do**
3: **for** each $B(S) \in C_\ell$ and each transaction filter $B(T_i)$ **do**
4: **if** $S \subseteq_B T_i$ **then** Bsupport(S)++ // $S \subseteq_B T_i$ iff $B(S) \wedge B(T_i) = B(S)$
5: **end for**
6: **for** each $B(S) \in C_\ell$ **do**
7: **if** $Bsupport(S) < N \cdot \tau'$ **then** delete $B(S)$ from C_ℓ // τ' is the revised threshold in data mining
8: **end for**
9: $F_\ell = C_\ell$ // F_ℓ is the collection of Bloom filters of all "frequent" itemsets with length ℓ
10: $C_{\ell+1} = can\_gen(F_\ell)$ // generate filters of candidate itemsets for the next round
11: **end for**
12: Answer = $\bigcup_\ell F_\ell$ // all filters of frequent itemsets

5.1.1 Efficient counting

To improve the efficiency in the counting phase, we organize the Bloom filters of the transactions of each group in a tree hierarchy and use every q bits to partition them at different levels, where q is a parameter. For example, at the root level, the partition leads to 2^q child nodes; the Bloom filters in each node share the same first q bits. A node splits if it contains more than c Bloom filters, where c is another parameter. At the end of partition, each leaf node contains limited number of Bloom filters, while each non-leaf node (except the root) is associated with a q-bit segment with which the node shares.

Because of the randomness of keyed hash functions, the distribution of Bloom filters is uniform,[6] which implies that the tree is well balanced. Therefore, an L-level tree can be used to index up to $c \cdot 2^{qL}$ Bloom filters. Given $q = 5$ and $c = 20$, for example, a 4-level tree can be used to index 20M Bloom filters.

Heuristic 5.1 *Let s be the q-bit segment associated with a non-leaf node and $B(S)$ be a Bloom filter of candidate itemset S. If any bit in s is* 1 *while the corresponding bit in $B(S)$ is* 0*, then no Bloom filter in the subtree rooted at the non-leaf node needs to be checked in the counting phase.*

In the counting phase, we traverse the tree to compare each candidate filter with the transaction filters stored in the leaf nodes and update the count of the candidate filter appropriately. According to the above heuristic, we may skip some subtrees in the counting process.

[6] Using the optimal number of hash functions, each bit in a Bloom filter has the equal probability of being 1 and 0.

An alternative way to do this is to organize candidate filters in a tree structure and update their counts appropriately while traversing the tree for each transaction filter. In multiple group case, this solution requires that the tree of candidate filters be built differently for each group, while in the above method a static tree of transaction filters can be used for any candidate filters.

5.1.2 Candidates generating

To support interactive data mining and discover all frequent itemsets as required in Problem 2, a multiple step interaction can be conducted between client and server in candidates generating phase. The client provides the server with a set of candidate filters C_ℓ; after the server sends back the mining result F_ℓ, the client generates another set $C_{\ell+1}$ of candidate filters and sends it to the server for data mining. This process can be done repetitively in each round. As mentioned in Sect. 3.2, with C_1 (i.e., Bloom filters of individual items) an edge server may decipher partial sensitive data (e.g., the compositions of transactions). Therefore, for the concern of protecting BI and preserving customer privacy, it is advisory not to outsource frequent 1-itemset mining tasks, or to use alternative choice of concealing C_1 discussed in Sect. 3.2.

The candidate generation $C_{\ell+1} = \text{can\_gen}(F_\ell)$ in this case is conducted at client side for privacy reasons. The Bloom filters in F_ℓ are transformed back to itemsets with the help of secret key. From this collection of itemsets, the client generates a new set of candidate itemsets using the well-known method $apriori\_gen$ as proposed in [4]. The basic idea of $apriori\_gen$ is that a candidate itemset of length $\ell + 1$ is generated only if all its subsets of length ℓ appear in the collection of itemsets. The client may also edit the set of candidates according to application requirements and constraints. Finally, the client transforms the candidate itemsets to Bloom filters and sends them back (in $C_{\ell+1}$) to the server. All of our experiments presented in the next section are based on this scenario.

Another choice to perform candidate generation is at server side. However, the server has no secret key to perform the hash functions, so it cannot transform back and forth between Bloom filters and itemsets. A possible solution is to use $C_{\ell+1} = \{B(S_1) \vee B(S_2) : B(S_1), B(S_2) \in F_\ell\}$ as candidate set for the next round. It is easy to verify that $B(S_1) \vee B(S_2)$ is the Bloom filter of itemset $S_1 \cup S_2$; therefore, this solution generates all Bloom filters of the itemsets that are unions of any two frequent itemsets (clearly, no frequent itemset is missed in this process). The disadvantage of this solution is that the server cannot exploit Apriori property using $apriori\_gen$ at itemset level.

5.2 Experiment settings

Rigorous experiments are conducted on both real data and synthetic data so as to evaluate the performance of our Bloom filter method in terms of mining precision, storage requirement, and computation time. All the experiments are run on a Compaq desktop computer with Pentium-4 CPU clock rate of 3.00 GHz, 3.25 GB of RAM, and 150 GB harddisk, running Microsoft Windows XP Professional Version 2002 with SP2.

5.2.1 Real data

The real data sets we adopt for this experiment are BMS-POS, BMS-WebView-1, and BMS-WebView-2 available at http://www.ecn.purdue.edu/KDDCUP. The dataset BMS-POS contains several years worth of point-of-sale data from a large electronics retailer, whereas the

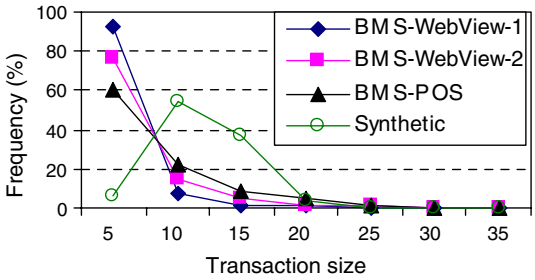

Fig. 1 Distribution of transaction sizes

datasets BMS-WebView-1 and BMS-WebView-2 contain several months worth of click-stream data from two e-commerce websites.

5.2.2 Synthetic data

We generate synthetic data using the transaction generator designed in IBM Quest project [4]. A synthetic dataset contains 100–750K transactions; each transaction is generated from a set of 1,000 frequent itemsets. The size of each frequent itemset is picked from Poisson distribution with mean 4. There are totally 1,000 distinct items. The size of each transaction is picked from Poisson distribution with mean 10. We use *four* sets of synthetic data, namely Syn1, Syn2, …, and Syn4.

5.2.3 Distributions of real data and synthetic data

The distributions of transaction sizes for both real data and synthetic data are shown in Fig. 1. According to [41], the exponential-like distributions of transaction sizes are typical in many real data sets used for mining association rules, while the synthetic data generated by IBM generator with Poisson distribution have been most widely used for testing the scalability of association rule mining algorithms. We use the real data to test the mining precision, storage requirement, computational time, and the tradeoffs among them. We use the synthetic data for scalability test. We show that our method works well for both types of data.

5.2.4 Mining precision

For mining precision, we compare the result of our method with the solution to the traditional problem (see Problem 1). We use the standard association rule mining algorithm Apriori [4] to discover exactly all frequent itemsets for the traditional problem. Given a revised threshold τ' in our method, the mining precision is measured in terms of overall false positive rate $F^+_{\tau'}$ and overall false negative rate $F^-_{\tau'}$. Let F be the set of Bloom filters of those frequent itemsets discovered by Apriori, and F' be the set of Bloom filters discovered by our algorithm. Then

- Overall false positive rate: $F^+_{\tau'} = \frac{|F'-F|}{|F'|}$.
- Overall false negative rate: $F^-_{\tau'} = \frac{|F-F'|}{|F'|}$.

Note that the overall false positive and false negative rates are not exactly the false positive and false negative rates discussed in Sect. 4.3. Given a data set and a data mining threshold, the false positive and false negative rates studied in Sect. 4.3 are the theoretical probabilities particular to each itemset, while the overall false positive and false negative rates are the experimental results for the whole data set. Statistically, one can expect a positive correlation between the two measurements.

Table 1 Experiment parameters and their default values

Parameter	Meaning	default value
k	Number of hash functions used for generating Bloom filters	30
τ	Frequency threshold for traditional data mining problem	1%
α	Coefficient for revised threshold $\tau' = \tau + \alpha \cdot 2^{-k}$	0
g	Number of groups	4

5.2.5 Storage requirement

The storage requirement is measured in terms of the length of Bloom filters and the number of transactions that an outsourced server needs to store. If there are multiple groups of transactions represented by Bloom filters of different lengths, then the storage requirement is the sum of the product of Bloom filter length and number of transactions in each group. Let there be g groups. For each group i, let the length of Bloom filters be m_i and the number of transactions N_i. Then

$$\text{Storage requirement} = \sum_{i=1}^{g} m_i N_i.$$

5.2.6 Experiment parameters

Table 1 gives the parameters used in our experiments as well as their default values. In our experiments, unless otherwise indicated, only one parameter is changed at a time while others are kept at their default values.

The number k of hash functions is the optimal parameter to determine the length m_i of Bloom filters in each group i based on the average length $\overline{n}_i$ of transactions in the group; that is, $k = \frac{m_i}{\overline{n}_i} \ln 2$. The same k is used across all groups.

Recall that for any itemset, the average difference of the revised frequency and the original frequency is upper-bounded by 2^{-k} (see Eq. 7). We use the coefficient α (where $0 \leq \alpha \leq 1$) to calculate the revised frequency threshold $\tau' = \tau + \alpha \cdot 2^{-k}$. This coefficient is used to balance between the overall false positive and false negative rates.

When the transactions are divided into multiple groups, the grouping is based on the distribution of transaction size. In our experiments, the grouping is carried out by the following method. Let $\overline{n}_1$ be the average length of transactions in a dataset. The transactions are firstly divided into two groups such that Group 1 contains those transactions whose lengths are not greater than $\overline{n}_1$ and otherwise for Group 2. Likewise, Group 2 is further divided into two groups. The grouping operation is continued until the original dataset is divided into g groups.

5.3 Experimental results

5.3.1 Data mining time versus data conversion time

In the first set of experiments, we study the data mining time versus the data conversion time by changing the number k of hash functions from 25 to 40.

Figure 2 shows that the time of mining frequent itemsets is much more than the time of converting raw data to Bloom filter representation, the more transactions, the more mining

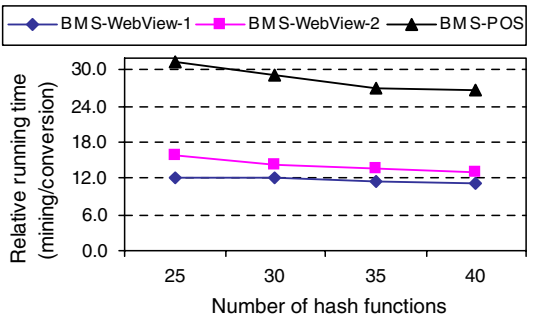

Fig. 2 Data mining time versus data conversion time

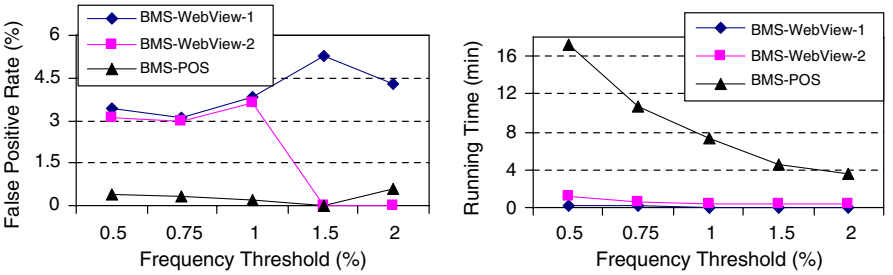

Fig. 3 Changing mining thresholds where $g = 4$, $k = 30$, and $\alpha = 0$

time. This result reveals that the mining process takes the major part of running time. It verifies the worthiness of pre-processing (i.e., data format conversion before outsourcing mining tasks).

5.3.2 Change mining thresholds

In the second set of experiments, we study the mining precision and running time by changing the frequency threshold τ from 0.5 to 2%. Other parameters are kept their default values, i.e., $g = 4$, $k = 30$, and $\alpha = 0$.

Figure 3 shows that the false positive rate is less than 6% for datasets BMS-WebView-1 and BMS-WebView-2, and less than 1% for datatset BMS-POS. For all datasets, the running time is decreasing with mining threshold. The reason is that a larger mining threshold will cause fewer frequent itemsets to be discovered in a shorter time. In addition, the experiments on dataset BMS-POS require more time than those on datasets BMS-WebView-1 and BMS-WebView-2 because dataset BMS-POS contains approximately 6–8 times more transactions than other two datasets.

The false negative rates are 0 for all datasets when $\alpha = 0$. We also run experiments for the revised frequency threshold $\tau' = \tau + \alpha \cdot 2^{-k}$ when α varies from 0.2 to 1.0 with an increment of 0.2. Since our experimental results are the same as compared with the cases for which $\alpha = 0$, the mining precision is not sensitive to the change of frequency threshold in our experiments. Therefore, in the following, we use $\tau = 1\%$ and $\alpha = 0$ and only consider the false positive in our analysis.

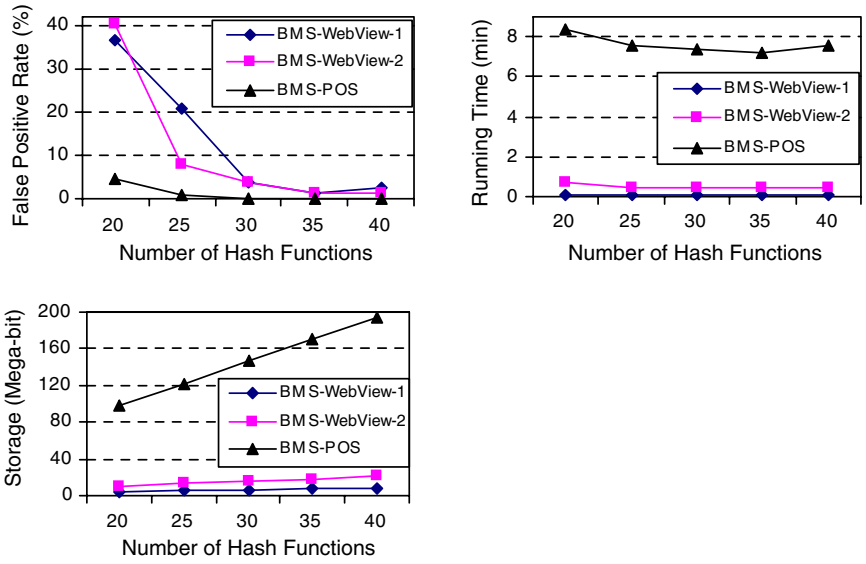

Fig. 4 Changing number of hash functions where $g = 4$, $\tau = 1\%$, and $\alpha = 0$

5.3.3 Change number of hash functions

In the third set of experiments, we study the mining precision, computation cost, and storage requirement with the change of the number k of hash functions. We change $k =$ from 20 to 40 and keep the other parameters at their default values, i.e., $g = 4$, $\tau = 1\%$, and $\alpha = 0$.

Figure 4 shows the mining precision with the change of k. There is a globally decreasing trend of false positive rate for each real dataset. For dataset BMS-POS, the false positive rate is nearly 5% for $k = 20$, and is less than 1% for $k \geq 25$. For datasets BMS-WebView-1 and BMS-WebView-2, the false positive rates are below 5% for $k \geq 30$. It also shows that the running time changes slightly with k. The running time is around 8 min for dataset BMS-POS and within 1 min for datasets BMS-WebView-1 and BMS-WebView-2. The storage requirement is linearly increasing with k for all datasets based on the figure. The reason is that the larger the k, the longer the Bloom filters, and the more storage is required to store the Bloom filters. The experimental results show that high mining precision can be achieved by increasing the number of hash functions. Consequently, the storage requirement increases linearly due to the use of longer Bloom filters.

5.3.4 Change number of groups

In the fourth set of experiments, we study the mining precision, computation cost, and storage requirement by changing the number g of groups from $g = 2$ to 5. The other parameters are kept at their default values, i.e., $k = 30$, $\tau = 1\%$, and $\alpha = 0$.

Figure 5 shows that the false positive rate decreases with the number of groups. For datasets BMS-WebView-1 and BMS-WebView-2, the false positive rate can be as low as 5% for $g \geq 3$; whereas for dataset BMS-POS, the false positive rate is less than 2.5% for $g \geq 2$. It also shows that running time and storage requirement change little with parameter g. The

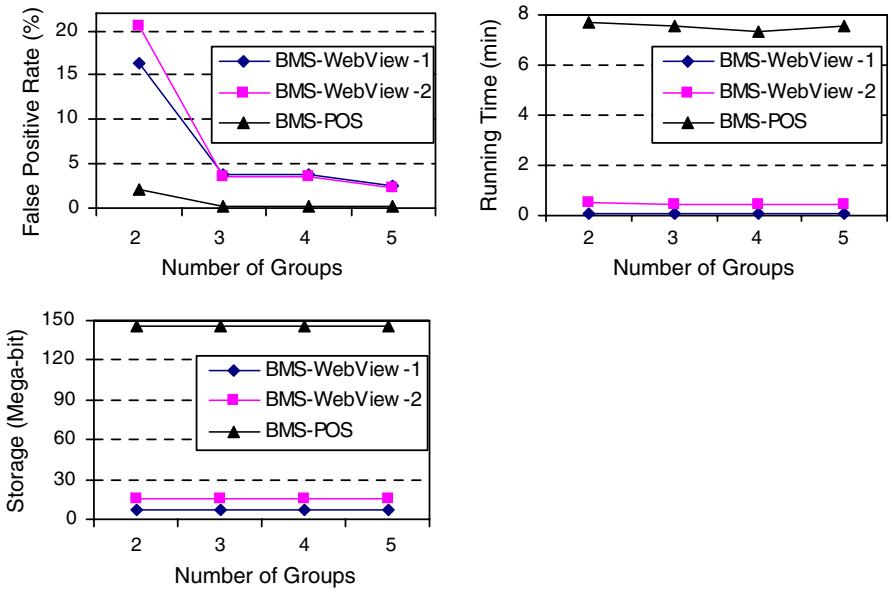

Fig. 5 Changing number of groups where $k = 30$, $\tau = 1\%$, and $\alpha = 0$

results of this set of experiments reveal that high mining precision can be achieved with larger number of groups without increasing much the storage requirement and the running time.

5.3.5 Scalability

In the last set of experiments, we use synthetic datasets Syn1, Syn2, …, and Syn4 to test the scalability of our algorithm with respect to the running time, mining precision, and storage requirement. We let $k = 20$ and keep other parameters at their default values, i.e., $g = 4$, $\tau = 1\%$, and $\alpha = 0$.

Figure 6 shows that the false positive rate for all synthetic datasets is no more than 1%, and that the running time and the storage requirement increase linearly with the number of transactions which is scaled up from 100 K in dataset Syn1 to 750 K in dataset Syn4. The scalability is also verified by experiments for $k > 20$ (e.g., $k = 25$ and 30) with *zero* false positive rate.

From the experimental results, we can conclude that the running time and storage requirement are scalable with the number of transactions while the mining precision is reasonably high.

5.4 Summary of experiments

One can draw the following conclusions from out experiments: (1) in the various cases of our experiments, zero false negative rate can be achieved. This result is not sensitive to the change of mining threshold; (2) by increasing the number of hash functions in formulating Bloom filters, the false positive rate decreases at the price of higher storage requirement; (3) by increasing the number of groups in data mining, the false positive rate decreases without increasing the running time or the storage requirement; and (4) the running time and the

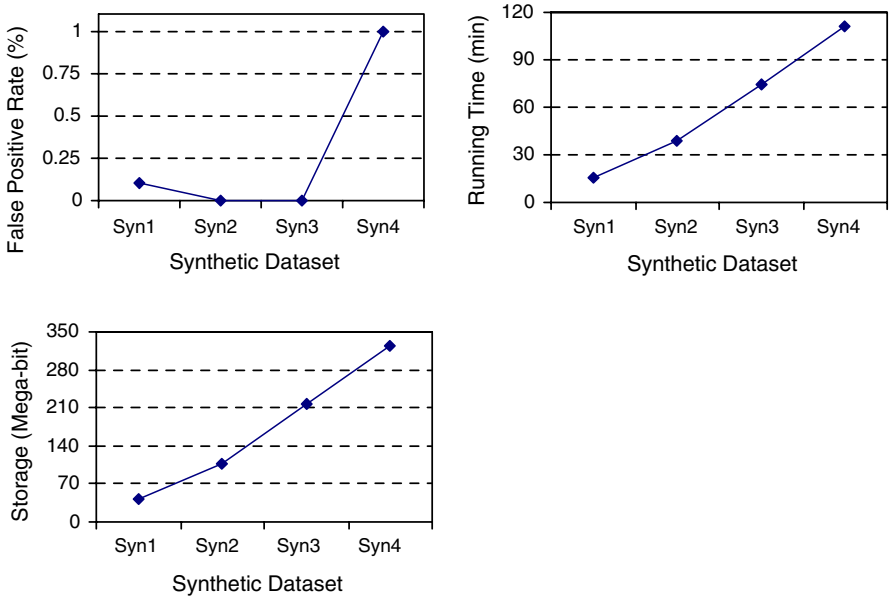

Fig. 6 Scalability results where $k = 20$, $g = 4$, $\tau = 1\%$, and $\alpha = 0$

storage requirement are scalable with the number of transactions that are processed in data mining.

6 Conclusions

The contribution of this paper is threefold. First, we proposed a new approach to outsourcing association rule mining tasks while protecting BI and customer privacy. The proposed approach is different from previous solutions in that it can protect BI and customer privacy while outsourcing mining tasks, at the same time, maintain the precision of mining results. Second, we performed theoretical analysis on the false positive and false negative rates in data mining. We also estimated the upper and lower bounds for the mining errors. Third, we investigated the tradeoffs between mining precision and storage requirement. Rigorous experiments were conducted on typical real and synthetic datasets showing that our approach can save storage significantly without sacrificing much mining precision, security level, and running time.

We are planning to investigate how to protect BI and customer privacy while outsourcing other mining tasks such as decision tree mining. It is also interesting to study privacy preserving techniques for mining specific types of frequent itemsets, such as maximum itemsets and closed itemsets.

References

1. Agrawal D, Aggarwal CC (2001) On the design and quantification of privacy preserving data mining algorithms. In: Proceedings of the 20th ACM SIGACT-SIGMOD-SIGART symposium on principles of database systems, pp 247–255

2. Agrawal R, Imielinski T, Swami A (1993) Mining association rules between sets of items in large databases. In: Proceedings of the 1993 ACM SIGMOD international conference on management of database pp 207–216
3. Agrawal R, Kiernan J, Srikant R, Xu Y (2004) Order preserving encryption for numeric data. In: Proceedings of the ACM SIGMOD ICMD, pp 563–574
4. Agrawal R, Srikant R (1994) Faster algorithms for mining association rules in large databases. In: Proceedings of the 20th international conference on very large data bases (VLDB'94), Santiago de Chile, Chile, September 12–15, pp 487–499
5. Agrawal R, Srikant R (2000) Privacy preserving data mining. In: Proceedings of the 2000 ACM SIGMOD international conference on management of database, Texas, USA, May 16–18, pp 439–450
6. Agrawal S, Haritsa JR (2005) A framework for high-accuracy privacy-preserving mining. In: Proceedings of the 21th IEEE international conference on data engineering (ICDE 2005), Tokyo, Japan, pp 193–204
7. Apte C, Liu B, Pednault E, Smyth P (2002) Business applications of data mining. Commun ACM 45(8):49–53
8. Atallah M, Bertino E, Elmagarmid AK, Ibrahim M, Verykios VS (1999) Disclosure limitation of sensitive rules. In: Proceedings of the IEEE KDEE, pp 45–52
9. Bishop M, Bhumiratana B, Crawford R, Levitt K (2004) How to sanitize data. In: Proceedings of the 13th IEEE international workshops on enabling technologies: infrastructure for collaborative enterprises (WETICE'04), Modena, Italy, June 14–16, pp 217–222
10. Bloom B (1970) Space time tradeoffs in hash coding with allowable errors. Commun ACM 7(13):422–426
11. Dasseni E, Verykios VS, Elmagarmid AK, Bertino E (2001) Hiding association rules by using confidence and support. In: Proceedings of the 4th international information hiding workshop, pp 369–383
12. Dibbeern J, Heinzl A (2002) Outsourcing information systems in small and medium sized enterprises: a test of a multi-theoretical casaul model. In: Dibbeern J (ed) Information systems outsourcing: enduring themes, emergent patterns, and future directions. Springer, New York
13. Du W, Zhan Z (2002) Building decision tree classifier on private data. In: Proceedings of IEEE ICDM'02 workshop on privacy, security, and data mining, vol 14, pp 1–8
14. Evfimievski A, Gehrke J, Srikant R (2003) Limiting privacy breaches in privacy preserving data mining. In: Proceedings of the 22nd ACM SIGMOD-SIGACT-SIGART symposium on principles of database system, pp 211–222
15. Evfimievski A, Srikant R, Agrawal R, Gehrke J (2002) Privacy preserving mining of association rules. In: Proceedings of the 8th ACM SIGKDD KDD 2002, pp 217–228
16. Hacigumus H, Iyer B, Li C, Mehrotra S (2002) Executing SQL over encrypted data in the database-service-provider model. In: Proceedings of the ACM SIGMOD international conference on management of database, pp 216–227
17. Hacigumus H, Iyer B, Mehrotra S (2002) Providing database as a service. In: Proceedings of the international conference on data engineering, pp 29–40
18. Hacigumus H, Iyer B, Mehrotra S (2004) Efficient execution of aggregation queries over encrypted relational databases. In: Proceedings of international conference on database systems for advanced applications, pp 125–136
19. Huang Z, Du W, Chen B (2005) Deriving private information from randomized data. In: Proceedings of the ACM SIGMOD international conference on management of data, Baltimore, MA, USA, June 14–16, pp 37–48
20. Iyer B, Mehrotra S, Mykletun E, Tsudik G, Wu Y (2004) A framework for efficient storage security in RDBMS. In: Proceedings of international conference on EDBT, pp 147–164
21. Kantarcıoğlu M, Clifton C (2002) Privacy preserving distributed mining of association rules on horizontally partitioned data. In: Proceedings of the ACM SIGMOD workshop on research issues on data mining and knowledge discovery, pp 24–31
22. Kantarcıoğlu M, Jin J, Clifton C (2004) When do data mining results violate privacy? In: Proceedings of the 10th ACM SIGKDD KDD 2004, pp 599–604
23. Kargupta H, Datta S, Wang Q, Sivakumar K (2003) On the privacy preserving properties of random data perturbation techniques. In: Proceedings of the 3rd IEEE ICDM, pp 99–106
24. Kargupta H, Datta S, Wang Q, Sivakumar K (2005) Random-data perturbation techniques and privacy-preserving data mining. Knowledge Inf Syst Int J 7(4):387–414
25. Lin Q-Y, Chen Y-L, Chen J-S, Chen Y-C (2003) Mining inter-organizational retailing knowledge for an alliance formed by competitive firms. Inf Manage 40(5):431–442
26. Lindell Y, Pinkas B (2002) Privacy preserving data mining. J Cryptol 15(3):177–206
27. Lui SM, Qiu L (2007) Individual privacy and organizational privacy in business analytics. In: Proceedings of the 40th Hawaii international conference on system sciences (HICSS 2007), Hawaii, USA, January 3–6, p 216b

28. Milne G-R (2000) Privacy and ethical issues in database/interactive marketing and public policy: a research framework and overview of the special issue. J Public Policy Marketing 19:1–6
29. Oliveira S, Zaiane O (2002) Privacy preserving frequent itemset mining. In: Proceedings of the IEEE ICDM workshop on privacy, security and data mining, pp 43–54
30. Oliveira S, Zaiane O (2003) Algorithms for balancing privacy and knowledge discovery in association rule mining. In: Proceedings of the 7th international database engineering and applications symposium, pp 54–63
31. Oliveira S, Zaiane O (2003) Protecting sensitive knowledge by data sanitization. In: Proceedings of the 3rd IEEE ICDM, pp 211–218
32. Ordones C, Ezquerra N, Santana CA (2006) Constraining and summarizing association rules in medical data. Knowledge Inf Syst Int J 9(3):259–283
33. Pinkas B (2002) Cryptographic techniques for privacy preserving data mining. ACM SIGKDD Explor 4(2):12–19
34. Qiu L, Li Y, Wu X (2006) An approach to outsourcing data mining tasks while protecting business intelligence and customer privacy. In: Workshops proceedings of the 6th IEEE international conference on data mining (ICDM 2006), Hong Kong, China, December 18–22, pp 551–558
35. Raś ZW, Gürdal O, Im S, Tzacheva A (2007) Data confidentiality versus chase. In: Proceedings of the joint rough sets symposium (JRS07), Toronto, Canada, May 14–16. Springer LNAI vol 4482, pp 330–337
36. Rizvi S, Haritsa J (2002) Maintaining data privacy in association rule mining. In: Proceedings of VLDB'02, pp 682–693
37. Saygin Y, Verykios VS, Clifton C (2001) Using unknowns to prevent discovery of association rules. Sigmod Rec 30(4):45–54
38. Vaidya J, Clifton C (2004) Privacy-preserving data mining: why, how, and when. IEEE Security Privacy 2(6):19–27
39. Xu S, Zhang J, Han D, Wang J (2006) A singular value decomposition based data distortion strategy for privacy protection. Knowledge Inf Syst Int J 10(3):383–397
40. Yao AC-C (1986) How to generate and exchange secrets. In: Proceedings of the 27th IEEE symposium on foundations of computer science (FOCS'86), Xi'an, China, pp 162–167
41. Zheng Z, Kohavi R, Mason L (2001) Real world performance of association rule algorithms. In: Proceedings of the 7th ACM-SIGKDD international conference on knowledge discovery and data mining, pp 401–406

Author's biographies

Ling Qiu is a Lecturer (academic level B) of IT in the School of Mathematics, Physics and Information Technology at James Cook University, Australia. He holds a PhD degree in Computer Science & Engineering awarded by Nanyang Technological University, Singapore in 2003. He received an ME degree from Zhejiang University, China and a BE degree from Huazhong University of Science and Technology, China. His research interests include parallel and distributed computing, algorithms, database systems and data mining, information systems and security, and system simulations.

Yingjiu Li is currently an Assistant Professor in the School of Information Systems at Singapore Management University, Singapore. He received his PhD degree in Information Technology from George Mason University in 2003. His research interests include applications security, privacy protection, and data rights management. He has published over 40 technical papers in the refereed journals and conference proceedings. Yingjiu Li is a member of the ACM and the IEEE. The URL for his web page is http://www.mysmu.edu/faculty/yjli/.

Xintao Wu is an Associate Professor in the Department of Software and Information Systems at the University of North Carolina at Charlotte, the USA. He got his PhD degree in Information Technology from George Mason University in 2001. He received his BS degree in Information Science from the University of Science and Technology of China in 1994, and an ME degree in Computer Engineering from the Chinese Academy of Space Technology in 1997. His major research interests include data mining and knowledge discovery, data privacy and security, and bio-informatics.