Available online at www.sciencedirect.com
SCIENCE DIRECT®
ELSEVIER
Information and Software Technology 48 (2006) 756–764
INFORMATION AND SOFTWARE TECHNOLOGY
www.elsevier.com/locate/infsof

# Incremental mining of generator representation using border sets

Lijun Xu *, Kanglin Xie

*Department of Computer Science and Engineering, Shanghai JiaoTong University, 282#, No.1954 HuaShan Road, Shanghai 200030, China*

Received 13 March 2005; received in revised form 23 September 2005; accepted 9 October 2005
Available online 7 December 2005

**Abstract**

Incremental frequent itemset mining refers to the maintenance and utilization of the knowledge discovered in the previous mining operations for later frequent itemset mining. This paper describes an incremental algorithm for maintaining the generator representation in dynamic datasets. The generator representation is a kind of lossless, concise representation of the set of frequent itemsets. It may be orders of magnitude smaller than the set of frequent itemsets. Furthermore, the algorithm utilizes a novel optimization based on generator borders for the first time in the literature. Generator borders are the borderline between frequent generators and other itemsets. New frequent generators can be generated through monitoring them. Extensive Experiments show that this algorithm is more efficient than previous solutions.
© 2006 Elsevier B.V. All rights reserved.

*Keywords:* Frequent itemset; Frequent generator; Border set; Incremental mining

## 1. Introduction

There has been an increasing focus on data mining, which is defined as the application of data analysis and discovery algorithms to large datasets with the goal of discovering predictive or decision-making models. The task of data mining is an interactive and iterative process in nature. Thus interactivity often plays a key role to facilitating effective data understanding and knowledge discovery. In such an environment, response time is crucial because lengthy time delay between responses of consecutive user requests can disturb the flow of human perception and formation of insight. However, the task of guaranteeing a quick response time is more difficult in dynamic datasets. Constant changes can invalidate existing patterns or introduce new knowledge. Simply re-executing algorithms from scratch may result in an explosion in the computational and I/O resources required. The problem of mining dynamic datasets has received some attention and incremental model maintenance algorithms for several data mining tasks have been proposed. In this paper we present such an approach for a key data mining field: frequent itemset mining [2,3].

Frequent itemset mining is an important subject in many data mining applications, such as the discovery of association rules, correlations, sequential rules and episodes. In brief, frequent itemset mining is described as follows: Given a dataset including a large number of transactions, find all frequent itemsets, where a frequent itemset is one that occurs in at least a user-defined percentage of the dataset. A large number of algorithms have been proposed for frequent itemset mining. But most algorithms assume that all transactions are available prior to the execution of the algorithm. However, in most cases this assumption does not hold. Many datasets are updated with blocks of data at regular time intervals. For example, data warehouses are always updated at large intervals and data streams are updated at small intervals. Recognizing the importance of incremental frequent itemset mining, many researchers have proposed their solutions and efficient algorithms.

In this paper, we present an efficient algorithm, called GBorder2, to maintain the generator representation in dynamic datasets. The generator representation is a kind of lossless, concise representation of the set of frequent itemsets. The usage of the generator representation can significantly reduce the times of data scans and the number of candidates in that the generator representation can be orders of magnitude smaller than the set of frequent itemsets. Moreover, to the best of our knowledge, the algorithm introduces a novel optimization utilizing the generator borders and the generator representation. The generator borders are the borderline between frequent generators and other itemsets. This optimization

* Corresponding author. Tel.: 86 21 62934265.
*E-mail addresses:* lijunxu@sjtu.edu.cn (L. Xu), xie-kl@cs.sjtu.edu.cn (K. Xie).

0950-5849/$ - see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.infsof.2005.10.001

provides significantly computational or I/O savings as new frequent generators can be generated through monitoring generator borders.

The remaining of the paper is organized as follows. In Section 2, we define related concepts used in this paper. In Section 3, we briefly discuss related work of frequent itemset mining. Then we give the detailed description of the GBorder2 algorithm in Section 4. Section 5 reports the experimental results. We summarize this paper in Section 6.

## 2. Problem definition

Let us first present some standard terminology for discovering the generator representation.

**Definition 1**. Let $I=\{i_1, i_2,\ldots,i_m\}$, be a set of $m$ items. A subset of $I$ is denoted as an itemset. An itemset with $k$ items is called a $k$-itemset.

A transaction $T=(TID, X)$ is a tuple where '$TID$' is a transaction-id and '$X$' is an itemset. A dataset $D$ is a set of transactions, $D=\{T_1, T_2,\ldots,T_n\}$. Let $|D|$ be the number of transactions in $D$.

During each update, obsolete transactions are removed and new transactions are added. We can treat the modification of existing transactions as deletion followed by insertion. Let $d^+$ be the set of newly added transaction and $d^-$ be the set of deleted transactions. Denote the updated dataset by $N$, i.e. $N=(D-d^-)\cup d^+$.

**Definition 2**. The support value of an itemset $X$, $Sup(X)$, is the number of the transactions in the dataset that contain $X$, $Sup(X)=|\{T|T\supseteq X\wedge T\in D\}|$. An itemset is frequent if it satisfies the support threshold ($\theta$). We denote $F$ as the set of frequent itemsets, i.e., $F=\{X|Sup(X)\geqq\theta|D|\}$.

The following lists one anti-monotone constraint, which states that all subsets of a frequent itemset are frequent and supersets of an infrequent itemset are infrequent. This property is first used in the Apriori algorithm [3].

**Property 1**. $X\in F\rightarrow\forall S\subset X, S\in F$; $X\notin F\rightarrow\forall S\supset X, S\notin F$.

**Definition 3**. An itemset is a generator if none of its proper subsets has the same support value as it has. We denote $G$ as the set of generators and $FG$ as the set of frequent generators, i.e. $FG=F\cap G$.

Negative generator border, $NGB$, is defined as the set of infrequent generators whose proper subsets are frequent generators, i.e. $NGB=\{X|X\notin F\wedge(\forall S\subset X, S\in FG)\}$[1].

Positive generator border, $PGB$, is defined as the set of frequent non-generators whose proper subsets are generators, i.e. $PGB=\{X|X\notin G\wedge X\in F\wedge(\forall S\subset X, S\in FG)\}$.

**Definition 4**. The generator representation is defined as consisting of the following components:

(a) $FG$ enriched by the support value for each itemset $X\in FG$;
(b) $NGB$.

The following lists several properties of generators. Please refer to [7,11] for more details. Property 2 shows how to determine the support value of any itemset from $G$; Property 3 presents another anti-monotone constraint, which states that all subsets of a generator are generators and supersets of a non-generator are not generators either; Property 4 shows how to determine if an itemset is frequent and if so, how to compute its support value based on the generator representation without scanning the dataset.

**Property 2**. $Sup(X)=min\{Sup(S)|S\in G\wedge S\subseteq X\}$.

**Property 3**. $X\in G\rightarrow\forall S\subset X, S\in G$; $X\notin G\rightarrow\forall S\supset X, S\notin G$.

**Property 4**. Let $X\subseteq I$. If $\exists Z\in NGB$ and $Z\subseteq X$, then $X\notin F$. Otherwise, $X\in F$ and $Sup(X)=min(\{Sup(S)|S\in FG\wedge S\subseteq X\})$.

## 3. Related work

There has been a lot of research in developing efficient algorithms for frequent itemset mining. We shall introduce two related aspects in brief: concise representations of frequent itemsets and incremental frequent itemset mining.

### 3.1. Concise representations of frequent itemsets

As the number of frequent itemsets is usually huge, it is important to apply concise, preferably lossless representations of frequent itemsets. A lossless representation can allow derivation and support determination of all frequent itemsets without accessing the dataset.

A number of lossless representations of frequent itemsets have been proposed, such as closed itemsets [15], generators [7], disjunction-free itemsets [6], disjunction-free generators [11], generalized disjunction-free itemsets [12] and generalized disjunction-free generators [12]. From the application point of view, the most useful representations are frequent closed itemsets and frequent generators. An itemset is closed if none of its proper supersets has the same support value as it has. An itemset is a generator if none of its proper subsets has the same support value as it has. In most cases the total number of closed frequent itemsets or frequent generators is orders of magnitude smaller than that of frequent itemsets.

Maximal frequent itemsets [14] are a kind of lossy, concise representation of frequent itemsets. A frequent itemset is maximal if none of its proper supersets is frequent. Each frequent itemset can be derived from maximal frequent itemsets. However, Maximal frequent itemsets does not contain information of the support value of each frequent itemset unless it is a maximal frequent itemset.

[1] In [11], $NGB$ is defined as follows: $NGB=\{X|X\in G\wedge X\notin F\wedge(\forall S\subset X, S\in FG)\}$. In fact an infrequent itemset whose proper subsets are frequent generators must be a generator. So we simplify the definition of $NGB$.