Constraint Graph Based Streaming Data Reduction for Updating Association Rules

Jason J. Jung[1] and Geun-Sik Jo[1]

Intelligent E-Commerce Systems Laboratory,
School of Computer Science and Engineering, Inha University,
253 Yonghyun-dong, Incheon, Korea 402-751
j2jung@intelligent.pe.kr, gsjo@inha.ac.kr

Abstract. Association rule mining from large dataset, such as Apriori, is time-consuming task. In this paper, we examine the issue of maintaining association rules from streaming dataset. In particular, we have been focusing on the exploitation of two kinds of constraints generated by users and adaptation. Based on our scheme, we provide the reduction of a given dataset during mining process and the gathering information while updating.

1 Introduction

Since association rule mining algorithm was introduced in [1], there have been many studies focusing on how to find frequent patterns from a given itemset such as market basket analysis. Traditionally, *Apriori* algorithm [2] and FP-Growth [3] have been the most well-known methods. These algorithms, however, have considered only static datasets. It means that the streaming dataset like online transactional logs is difficult to be driven by generic *Apriori*-like algorithms. In fact, many applications on the web have focused on mining sequential patterns from data streams. For example, on-line newspaper article recommendation, web proxy server for prefetching content, and user preference extraction for supporting adaptive web browsing can be told as the domains relevant to analyzing data streams from many clients.

Several studies thereby have been proposed for maintaining the set of mined association rules. FUP (Fast UPdate) is an incremental updating technique based on *Apriori* and DHP (Direct Hashing and Pruning) [5]. After a set of new transactions are piled up, FUP finds out new large itemsets from a new dataset and compared them with old ones based on heuristics, in order to determine which operation should be executed like removing losers, generating candidate sets, and finding winners. Furthermore, FUP$_2$ was more generalized algorithm of FUP, as handling other maintenanace problems [6]. In [4], DELI (Difference Estimation for Large Itemsets) was proposed as a way of estimating the difference between the association rules in a database before and after they are updated. DELI is used as an indicator for whether the FUP$_2$ should be applied to the database to accurately find out new association rules. However, these algorithms are highly time consuming, because they are basically composed of the repetition of the same tasks such as scanning dataset, counting itemsets, and measuring their supports in order to generate the candidate set finding out the large itemsets iteratively.

In this paper, we have been focusing on the problem whose search space is fixed with respect to the size, but dynamic with respect to time. We applied constraint satisfaction techniques to improve the performance of the association rule mining from this kind of problems. We thereby propose how to organize constraint graphs and how to reduce the search space of a given dataset based on these constraints.

2 Problem Description

A set of literals, called *items* is represented as X. Let D be a set of transactions, where each transaction T_i is the i-th itemset $\{x_1, x_2, \ldots, x_\alpha, \ldots, x_n\}_i$. In gerneral scheme of data mining, we say that a transaction T_i supports not only an item $x_{(\alpha,i)}$ but also the other items related to this item. These relationships are explicitly predefined as hierarchical taxonomies. In this paper, we assume that the new dataset *db* is added to old dataset DB. During merging these two datasets, because the space storing dataset is limited, some itemsets from DB and *db* should be removed by constraint conflictions.

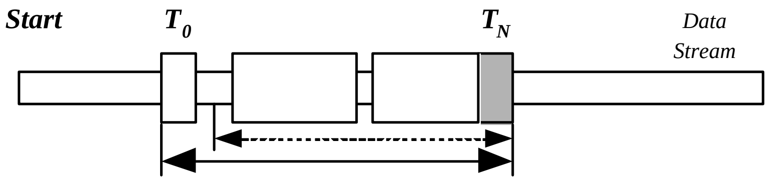

Fig. 1. Streaming Dataset and Sliding Window

As shown in Fig. 1, the merged dataset therefore can be fragmented. The size of sliding window (T_0 to T_N) is larger than that of all fragmented itemsets. Therefore, we note the problem statements concentrated for updating association rules from streaming dataset in this paper, as follows.

- **Dataset Filtering.** Conceptually this task should prune the search space of a given data mining problem and also satisfy the space constraints by transforming into an equivalent one operating on a smaller dataset.
- **Gathering Information While Updating.** Generally this task learns experience in a search, in order to avoid the same mistake in the future. While updating frequent itemsets from data stream, the existing constraints may be more tight or loose. Especially, new constraints can be generated.

Users generally define the minimum support as the threshold value for mining frequent patterns. Minimum support can be regarded as an explicit constraints for dataset filtering. Meanwhile, implicitly recognized constraints should be also stored, in order to increase the performance of mining process. Contrary to user-defined constraints, constraints discovered while updating can discriminate whether a given dataset should be counted or not.

3 Organizing Constraint Graph

We have noted two kinds of constraints, which are user-defined and inductively gernated constraints as follows.

- **User-defined constraint.** This should be, in advance, explicitly configurated. The taxonomy of items and minimum support of an item can be classified into this category.
- **Inductively generated constraint.** This is implicit constraints generated during on-line mining tasks. For this constraint, we need to establish several functions.

It means that constraint graph, firstly, should be configured through the user's requests represented as predefined operators. Then, this graph can keep adjusting itself to on-line streaming dataset.

Basically, we assume that three kinds of simple functions are applicable to define an explicit constraint in this paper. These are existence, equality, and comparison. In particular, the taxonomy constraint is explicitly established as hierarchical tree structure. Therefore, as mentioned in [11], we need to devide the existence function into the functions *ancestors* and *descendants*, in order to more specifically indicate the relationship between items of this tree.

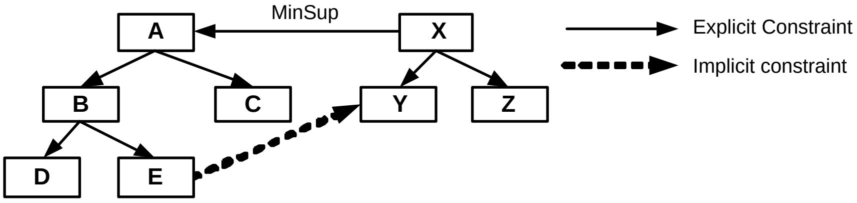

Fig. 2. Two kinds of constraints

For example, in Fig. 2, parents and child nodes of node `B` can be simply described as:

$$ancestors(B) = \{A\} \tag{1}$$

$$descendants(B) = \{D, E\} \tag{2}$$

in which inclusion constraints are implying. Also, minimum support, as another explicit constraint, can be represented as the equality and comparison functions. After each item is put different restrictions (e.g., support or confidence), they can be compared with each other, as follows:

$$Sup(A) \geq \theta_{Sup} \tag{3}$$

$$Sup(A) \geq Sup(X) \times 2 \tag{4}$$

where θ_{Sup} is the user-specified minimum support. More importantly, a dotted arrow means the implicit constraint generated while adding new dateset. In order to represent this kind of constraints,

4 Maintenance of Association Rules

As sliding windows is shifted (in Fig. 1) by newly inserted data, consistency checking should be applied to test their satisfiability.

4.1 Consistency Checking by Constraint Graphs

We have described how to organize constraint graph by users. Basically, in order to reduce search space of a given problem, consistency checking can be conducted. It finds out the redundant parts that we should not scan any more. We are focusing on node-consistency (*NC*) and arc-consistency (*AC*). NC checking is based on unary constraints involved with a particular item x_i. Algorithm *NC* presents the pseudo code for node-consistency achievement:

```
Algorithm NC
Input:
  Time Window, $TW = [T_0, \ldots, T_N]$;
  Old Dataset, $DB$; New Dataset, $db$;
  Set of Assoication Rules Discovered from DB, $R_{DB}$;
  Constraint Graph, $CG$;
Procedure:
  begin
    $i \leftarrow N$;
    while $i \geq (N - |db|)$ and $T_i \in TW$ do
    begin
      for each $x_j \in T_i$ do
        $Update(x_j)$;
        if (not $Satisfies(x_j, CG_1(x_j))$) then
          $Prune(x_j)$
      $i \leftarrow i - 1$;
    end
    Prune(<list of conflicted items>, DB);
  end.
```

In this code, the function $Update(x_j)$ represents the aggregation operations related to input node, such as counting. The function $Satisfies(x_j, CG_1(x_j))$ evaluates input node x_j with unary constraints involved in the corresponding node. More importantly, the function $Prune$ removes the transactions conflicted with from old dataset DB. For example, let the minimum support of an item x_i be $\theta_{Sup}(x_i)$. During checking NC of new dataset db, transactions including x_i can be pruned, as shown in the following equation:

$$count(x_i, db) \geq \theta_{Sup}(x_i) \times (|DB| + |db|) - Sup(x_i, DB) \times |DB| \tag{5}$$

where *count* is the function for counting the itemset including an item in a given dataset.

AC checking is based on binary constraints involved with a pair of item x_i and x_j. An AC achievement algorithm is shown below:

```
Algorithm $AC$
Input:
  Time Window, $TW = [T_0, \ldots, T_N]$;
  Old Dataset, $DB$; New Dataset, $db$;
  Set of Assoication Rules Discovered from $DB$, $R_{DB}$;
  Constraint Graph, $CG$;
Procedure:
  begin
    $k \leftarrow N$;
    while $k \geq (N - |db|)$ and $T_k \in TW$ do
    begin
      for each $x_i \in T_k$ do
        $NC(x_i)$;
        if (not $Satisfies(x_i, CG_2(x_i))$) then
          $Prune(x_i)$
      $k \leftarrow k - 1$;
    end
    $Prune(<list\ of\ conflicted\ items>, DB)$;
  end.
```

After NC of a certain item, we can retrieve binary constraints by function CG_2. If a user establishes a constraint

"For an item x, $\sum_{x_i \in descendents(x)} [Sup(x_i)] \geq Sup(y)$",

transactions including x_i can be pruned, as shown in the following equation:

$$\begin{aligned} &count(descendents(x), db) \\ &\geq (count(y, DB) - count(descendents(x), DB))\frac{|db|}{|DB|} + count(y + \delta, db) \end{aligned} \tag{6}$$

where δ is the variable for estimated value with respect to the ratio of new datasets $(\frac{|db|}{|DB|})$.

4.2 Gather-Information-While-Updating

While scaning datasets for finding frequent large itemsets, constraints can be adaptive to new datasets. As a matter of fact, due to the difficulties of the description of constraints, users have to be supported, as constraint information is notified. To do this, we need to define some problem-depended functions for retrieving new information from transaction data. During shopping, as an example, a group of customers under the similar circumstance (e.g., preferences and economical condition) have almost the same behavioral patterns such as the number of items, the total price of all items, and the quality of items in a basket.

5 Conclusions and Future Work

In this paper, we have considered the problem of analyzing the streaming data for efficiently updating association rules. We have proposed consistency checking scheme based on user-defined constraints, as filtering redundant part of data. Moreover, gathering information while updating have been proposed to adaptively control the tightness of constraints of given problems.

As a future work, we need the additional research for applying not only NC and AC, but also path consistency (PC) checking.

References

1. Agrawal, R., Imielinski, T., Swami, A.: Mining association rules between sets of items in large databases. In Proc. of the ACM SIGMOD Conference on Management of Data (1993) 207–216
2. Agrawal, R., Srikant, R.: Fast Algorithms for Mining Association Rules. In Proc. of the 20th VLDB Conference (1994)
3. Han, J., Pei, J.: Mining Frequent Patterns by Pattern-Growth: Methodology and Implications. ACM SIGKDD Explorations (2000) 31–36
4. Lee, S.D., Cheung, D.W.: Maintenance of Discovered Association Rules: When to Update? In Proc. of ACM SIGMOD Workshop on Data Mining and Knowledge Discovery (DMKD) (1997)
5. Cheung, D.W., Han, J., Ng, V.T., Wong, C.Y.: Maintenance of Discovered Rules in Large Databases: An Incremental Updating Technique. In Proc. of Int. Conf. on Data Engineering (1996) 106–114
6. Cheung, D.W., Lee, S.D., Kao, B.: A General Incremental Technique for Maintaining Discovered Association Rules. In Proc. of Int. Conf. on Database Systems for Advanced Applications (DASFAA) (1997) 185–194
7. Zheng, Q., Xu, K., Ma, S.: When to Update the Sequential Patterns of Stream Data? In: Whang, K.-Y., Jeon, J., Shim, K., Srivastava, J. (eds.): Advances in Knowledge Discovery and Data Mining. Lecture Notes in Artificial Intelligence, Vol. 2637. Springer-Verlag (2003) 545–550
8. Hidber, C.: Online Association Rule Mining. In Proc. of the ACM SIGMOD Conference on Management of Data (1999) 145–156
9. Pudi, V., Haritsa, J.: How Good are Association-rule Mining Algorithm? In Proc. of the 18th Int. Conf. on Data Engineering (2002)

10. Wojciechowski, M., Zakrzewicz, M.: Dataset Filtering Techniques in Constraint-Based Frequent Pattern Mining In: Hand, D.J., Adams, N.M., Bolton, R.J. (eds.): Pattern Detection and Discovery. Lecture Notes in Computer Science, Vol. 2447 Springer-Verlag (2002) 77–91
11. Srikant, R., Vu, Q., Agrawal, R.: Mining Association Rules with Item Constraints In Proc. of the 3rd Int. Conf. on Knowledge Discovery and Data Mining (1997) 67–73