Technical Report No. CSD-99-1089

An Efficient Method for Mining Association Rules with Item Constraints

Shin-Mu Vincent Tseng
Computer Science Division, EECS Department
University of California, Berkeley
Berkeley, CA94720
Email: tsengsm@cs.berkeley.edu

Abstract

Most existing studies on association rules discovery focused on finding the association rules between all items in a large database that satisfy user-specified minimum confidence and support. In practice, users are often interested in finding association rules involving only some specified items. Meanwhile, based on the search results in former queries, users tend to change the minimal confidence and support requirements to obtain suitable number of rules. Under these constraints, the existing mining algorithms can not perform efficiently due to high and repeated disk access overhead. In this research, we present a novel mining algorithm that can efficiently discover the association rules between the user-specified items or categories via feature extraction approach. At most one scan of the database is needed for each query; hence, the disk access overhead can be reduced substantially and the query be responded quickly.

1. Introductions

Data mining is the process of extracting previously unknown and useful information from a large database. It has been extensively applied to a wide variety of applications like sales analysis, healthcare, manufacturing, etc. A number of studies have been made on efficient data mining methods and the relevant applications.

Among the data mining problems, association rules discovery might be the most studied ones. This problem was introduced in [1]. Given a set of transactions, where each transaction is a set of items, an association rule is an expression $X \Rightarrow Y$, where X and Y are sets of item. An example of an association rule is: "25% of transactions that contain beer also contain diapers; 5% of all transactions contain both items". Here, 25% is called the *confidence* of the rule, and 5% the *support* of the rule [1, 2]. Most existing work on this problem focused on finding the association rules among all items in a large database that satisfy user-specified minimum confidence and support [1-4, 6-7]. A number of studies were made to propose efficient methods for mining association rules by reducing either the CPU computation time or the disk access overhead [3-4, 7]. Some studies considered the usage of sampling techniques for reducing the processing overhead [11, 12]. Srikant and Agrawal [9] incorporated into considerations the quantities of the items purchased in discovering association rules. Some approaches were proposed for mining generalized association rules [8, 10]. Sarawagi et al. [10] considered the integration of association rules mining with SQL in standard databases.

In practice, users are often interested in finding association rules involving only some specified items rather than all items in a query. Meanwhile, based on the search results in former queries, users might change the minimum confidence and support requirements to obtain suitable number of rules. In this scenario, the users tend to make several consecutive queries with

expected quick response time and different interested items, minimum confidence and support for potential rules, rather than wait a long time to get a lot of association rules for all itemsets in only partial of which the users are really interested.

Under these item constraints, the existing mining algorithms have the following drawbacks:

Firstly, they can not perform efficiently in terms of responding the user's query quickly though they can perform well in finding the association rules among all itemsets. The main reason is that the existing mining algorithms are mostly designed in forms of several passes so that the whole database needs to be read from disks several times for each user's query under the constraint that the whole database is too large to be stored in memory. This is very inefficient in considering the big overhead of reading the large database even though only partial items are interested in fact. The worse part is the existing mining algorithms will repeat reading the whole database several passes for a subsequent query even it involves the same specified items as the previous query but changes only the minimum confidence and support.

Secondly, no guiding information is provided for users to choose suitable settings for the constraints (like support and confidence) such that an appropriate number of association rules are discovered. Consequently, the users have to use a try-and-error approach to get suitable number of rules. This is very time-consuming and inefficient.

Srikant and Agrawal [8] introduced the problem of mining generalized association rules as follows: Given a set of transactions, where each transaction consists of a set of items, and a taxonomy on the items, we find associations between items at any level of the taxonomy. For example, given a taxonomy that says that jackets *is-a* outerwear *is-a* clothes, we may infer a rule that "50% of people who buy outerwear tend to buy shoes. 5% of all transactions contain both

these items". This reflects the fact that the taxonomies over the items exist in many real-life applications.

In this paper, we also investigate another problem related to mining generalized association rules. We call the problem as "mining categorized association rules" and the scenario in this problem is as follows:

1. Although the taxonomy may be used to model the hierarchical relationships of items in an application very well, people are often interested in association rules under a simpler taxonomy, i.e., the one-level category an item belongs to.
2. Users are often interested in finding association rules involving only some specified categories rather than all categories in a query. Hence, constraints on categories will be specified.
3. Users may make continuous queries, in which new requirements for parameters like support are set based on the previous query results. This is because the number of discovered rules may not be suitable. Therefore, quick response to users' queries is needed.

With the scenario as described above, the existing mining algorithms may not perform well in terms of responding the user's query quickly. The main reason is that the existing mining algorithms are mostly designed in forms of several passes so that the whole database needs to be read from disks several times for each user's query under the constraint that the whole database is too large to fit in memory. This is very inefficient in considering the big overhead of reading the large database even though only partial items are interested in fact. The worse part is the existing mining algorithms will repeat reading the whole database several passes for a

subsequent query even it involves the same specified items as the previous query but changes only the minimum confidence and support.

A new approach is proposed in this paper for mining categorized association rules efficiently. The main features of the method are as follows: For each interested item or category, a compressed feature vector and feature record are built to represent the occurrence patterns of the items belonged to this category. The feature vector and feature record are built only once while reading the database first time. Then, the associations between the interested items or categories are constructed by using the feature record information and performing simple logical operations on the feature vectors without reading the large database again. Hence, all the disk access overhead, calculation time for mapping the items to belonged categories, and the calculation time for finding the associations between the categories can be reduced substantially.

Besides, we also describe a methodology to provide users with useful information regarding the database to be mined, like the estimated number of potential rules under various settings of the constraints. Thus, users can make suitable constraint settings more easily and less mining processes are needed.

The rest of the paper is organized as follows. We state the problem formally in Section 2. In Section 3, the proposed method is described in details. A conclusion and the future work is given in Section 4.

2. Problem Statement

In the following, we give a formal statement of the problem [1]: Let $I = \{i_1, i_2, \ldots, i_m\}$ be a set of items. Let D be a set of transactions, where each transaction T is a set of items such *that* $T \subseteq I$.

Associated with each transaction is a unique identifier, called its *TID*. An association rule is an implication of the form $X \Rightarrow Y$, where $X \subset I$, $Y \subset I$, and $X \cap Y = \varnothing$. The rule $X \Rightarrow Y$ has support s if s% of transactions in D contain $X \cup Y$, and it has confidence c if c% of transactions in D that contain X also contain Y. Let $IS \subset I$ be the set of items the users are interested. Given the constraints of *IS*, s and c, the problem is to find all association rules involving items contained in *IS* with support and confidence larger than s and c, respectively. Itemsets with satisfied support are called large itemsets. We call s and c as *minsup* and *minconf*, respectively.

For the categorized association rules mining, the following information is complemented. Let $C = \{ C_1, C_2, \ldots, C_m \}$ be a set of categories and $M = \{(i_k, C_k) | \ i_k \in I$ and $C_k \in C\}$ be a set of mapping which specifies which category an item belongs to. Here, we assume that an item belongs to only one category. Besides, let $IC \subset C$ be the set of categories the users are interested in the query the user wants to submit. The set of *IC*, s and c constitutes the mining constraints. Given the mining constraints, the problem is to find all categorized association rules involving items contained in *IC* with support and confidence larger than s and c, respectively. Category sets with satisfied support are called large category sets. We call s and c as *minsup* and *minconf*, respectively.

To meet the goals mentioned in Section 1, we propose an intelligent and efficient data miner for discovering association rules as shown in Figure 1. The data miner consists of two components: a mining algorithm and the pre-miner, which exploits some sampling technique to analyze the database speedily.

In the system model, initially an user submits a query with some mining constraints. After the association rules are detected, the user might change some of the mining constraints like *IC* for submitting subsequent query since the number of discovered rules may not be suitable. That

is, the user may make continuous queries based on previous mining results. The function of Pre-Miner is to make speedy analysis of the database and then provide users with useful analyzed information, which can help users choose suitable settings for the constraints like support and confidence. The main techniques for the pre-miner are still under designing, but they will basically be based on some sampling methods since the purpose of pre-miner is to get estimated information of the database instead of the precise one.

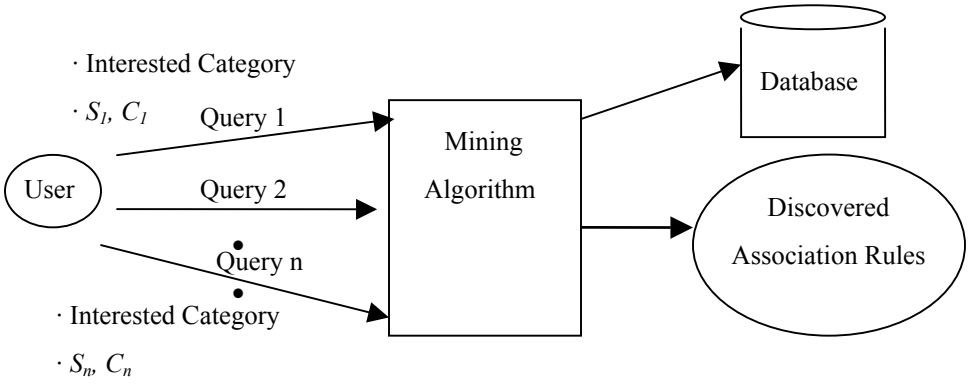

Figure 1. The system architecture

Based on the analyzed information provided by pre-miner, the users could make several queries with different constraints which include the interested items, the support (S_i for *ith* query) and the confidence (C_i for *ith* query). According to the constraints specified by the user in each query, the mining algorithm reports the discovered associated rules after processing the database.

The problem of discovering the association rules for all items can be decomposed into two sub-problems [1, 2]:

1. Find all sets of items (or itemsets) that have transaction support above minimum support.
2. Use the large itemsets to generate the desired rules.

It was pointed out that sub-problem 2 is quite trivial as compared to sub-problem 1. Hence, most work focus on solving sub-problem 1, i.e., to find the large itemsets. In our problem, the focus shifts from itemsets to categories. Similarly, we will aim to find large category sets, which can be used to generate the association rules between categories easily.

3. Proposed Methods

In this section, we firstly describe some previous work on the defined problem. Then we describe in details the proposed mining algorithm and the pre-miner.

3.1 Previous Work

The problem of discovering the association rules can be decomposed into two sub-problems [1, 2]:

1. Find all sets of items (itemsets) that have transaction support above minimum support.
2. Use the large itemsets to generate the desired rules. A general way to do this is examining all large itemsets, say ABCD and AB, and determine if the rule AB $\Rightarrow$ CD holds by computing the ratio *conf* = support(*ABCD*)/support(*AB*). If *conf* $\geq$ *minconf*, then the rule holds.

Clearly the sub-problem 2 is quite straightforward once sub-problem 1 is resolved. Hence, most researches are focused on solving sub-problem 1. This research is also aimed at sub-problem 1, too.

A number of studies have been done on association rules mining [1-9]. Most of them follow the representative approach by Agrawal et al. [2], namely Apriori. The principle of Apriori is as follows. Let L_k denote the large itemset with k items (also called large k-itemset). The first pass of the algorithm scans the database and counts item occurrences to determine the large 1-itemsets. A subsequent pass, say pass k, consists of two phases. First, the large itemsets L_{k-1} found in the ($k-1$)th pass are used to generate the candidate itemset C_k. Then the database is scanned and the support of candidates in C_k is counted. This repeats until the generated large itemsets become empty set. Using the notation defined in Section 2, the Apriori algorithm is as shown in Figure 2.

Scan the database once to get large 1-itemsets L_1;
For (k=2; $L_{k-1} \neq 0$; k++) do begin
    C_k = apriori-gen(L_{k-1});
forall transactions $t \in D$ do begin
C_t = subset(C_k, t);
Forall candidates $c \in C_t$ do
increase the count of c by 1;
end
$L_k = \{ c \in C_k \mid \text{c.count} \geq minsup\}$
end
Answer = union of all L_k ;

Figure 2. The Apriori algorithm.

Agrawal et al. [2] showed that Apriori algorithm outperforms other mining algorithms due to the stable performance under various number of data items. However, it incurs the problem of high disk access overhead as stated in Section 1. In particular, this problem is more obvious when only partial data items instead of all items are interested.

3.2 The Proposed Mining Algorithm

The idea of the proposed mining method is to extract the features of the interested items and store them in compressed vectors. Once the compressed vectors of the interested categories are built, all computations for the concurrent occurrences of the interested categories are made by Boolean AND operation by using the feature vectors without accessing the database again. Hence, the high disk overhead incurred in other mining algorithms due to multiple parses of the database can be reduced substantially. The algorithm of the proposed method is as shown in Figure 3.

Central to the proposed mining method are two databases, namely *Support DB* and *Feature DB*. *Feature DB* records the feature vectors for the interested data items, while *Support DB* records the calculated support for each data item. When a user submits a query for mining a database with the constrained support s, confidence c and interested items IT, the proposed method firstly examines if the *Support DB* exists and builds it if it is absent. Building *Support DB* requires one pass of the whole database (recalled that the support for a data item is the number of total occurrences in ratio of the total number of transactions). During the scanning process, the *Feature DB* for the interested items is built, too (the details of building the feature vector for each interested item will be described later). Once *Support DB* is built and stored in

the disk, it takes very short time in further queries to determine L_1 by accessing *Support DB* directly instead of scanning the whole database. After L_1 is determined, any vectors of the interested items which do not exist in *Feature DB* will be built.

1) IF (*Support DB* is empty)
   Scan the database to build *Support DB* and determine L_1, and
   build *Feature DB* for interested items;
   ELSE
   Retrieve *Support DB* and determine L_1 and build *Feature DB*
   for non-existent items;
2) k=2
3) Generate candidate large itemset C_k from $L_{k-1;}$
4) IF C_k is not empty
   Obtain the count for each itemset in C_k by bitwise AND
   operations on the items' *feature vectors* and get $L_{k;}$
   k=k+1;
   go to step 3;
5) Answer = union of all L_k ;

Figure 3. Algorithm of the proposed mining method.

The process for building the feature vector V_i for a data item D_i is as follows. Basically, V_i is compressed form of a bit vector in which jth bit is set as 1 if D_i appears; otherwise, it is set as 0. However, in considering the possibility of sparse occurrences of a data item in a large database, the consecutive 0 is counted and stored as a count index if the count is larger than 8 (since we use 1 byte as a segment). By storing the appearing feature of each interested item as a compressed vector separately, the size of the database to be accessed can be reduced greatly.

Hence, the disk overhead of scanning the whole database as existed in other mining algorithms can be reduced.

Another alternative in designing the feature vector is to store only the location a data item appears. A range description can be used for consecutive appearances. This approach could be more effective than the method described in last paragraph in required size for each vector. A detailed analysis will be made for contrasting their performance in the future.

In generating and pruning candidate large itemset C_k from L_{k-1}, the proposed method uses the approach similar to Apriori. To count the number of occurrences for each candidate itemset C_k, the proposed method does not scan the whole database as Apriori does. Instead, only the vectors for the items contained in C_k are accessed for doing bitwise AND operations, and the count is obtained by counting the number of 1's in the AND operated result.

The candidates generating and counting process goes on until no candidate can be generated any more. Finally, the large data sets are the union of all L_k and the association rules can be obtained easily by using the constrained confidence c as described in Section 3.1.

For mining categorized association rules, the proposed algorithm is modified slightly as shown in Figure 4. Central to our method are two databases, namely *Category Support DB (CSDB)* and *Category Feature DB (CFDB)*. *CFDB* records the feature vectors for the interested category, while *CSDB* records the calculated support for each data item. When a user submits a query for mining a database with the constrained support *s*, confidence *c* and *interested category IC*, the proposed method firstly examines if the *CSDB* exists and builds it if it is absent.

1) IF (CS*DB* is empty)
    Scan the database to build *CSDB* and determine L_1, and
      build *CFDB* for interested category;
  ELSE
    Retrieve *CSDB* and determine L_1 and build *CFDB*
      for non-existent category;
2) k=2
3) Generate candidate large category set C_k from $L_{k-1;}$
4) IF C_k is not empty
    Obtain the count for each category in C_k by bitwise AND
      operations on the category's *feature vectors* and get $L_{k;}$
    k=k+1;
    go to step 3;
5) Answer = union of all L_k ;

Figure 4. Algorithm for finding large category set.

3.3 The Pre-miner

Almost all existing studies on association rules mining focused on improving the performance of the mining algorithms [1-9]. However, one important problem ignored is how to set suitable constraints such that appropriate number of association rules can be discovered. Arbitrary settings of the constraints may lead to extremely large number of rules beyond user's capability for analysis. Consequently, users have to adjust the constraints settings and conduct the time-consuming mining over again.

The goal of Pre-miner is to provide users with useful information regarding the database to be mined, like the estimated number of potential rules under various settings of the constraints. Thus, users can make suitable constraint settings more easily and less mining processes are needed.

The challenge lies in that the Pre-miner must be time-effective in analyzing the database while preserving reasonably accurate estimation; otherwise, users would rather do the mining directly. To meet this goal, some sampling techniques like in [10-11] are being developed to save processing time in parsing the large database. Meanwhile, visualizing approach will be deployed to clearly illustrate the analyzed results.

4. Conclusions

In this work, we propose an intelligent and efficient data miner for mining association rules with constrained items or categories in large databases. The proposed data miner consists of two components: a data mining algorithm named and a database analyzer called Pre-miner. The proposed mining algorithm can efficiently discover the association rules between the data items in a large database based on the dynamic constraints specified by the users. In particular, at most one scan of the whole database is needed for each query. Hence, the high repeated disk overhead incurred in other mining algorithms can be reduced significantly.

In the following, we give brief comparisons of the proposed method with Apriori-based methods. For our problem, the merits of the proposed method are;

1. Less disk access overhead. For Apriori, the database has to be read k times if we found large category sets as many as L_k.. For our method, at most one database scan is needed for each user query.
2. No repeated database access and computation occurs in our method. For two continuous queries from some user with different mining constraints, Apriori treats both queries equally and redo the mining tasks for each query. Consequently, all database access will be done repeatedly even though some of the information has been obtained in previous query. By taking this into consideration, the proposed method stores both the occurring patterns of the interested category and the mining result, such that no repeated database accesses are made. This will save considerable execution time.
3. Less computation for mapping interested items into interested categories. With Apriori-based methods, the mapping of interested items into corresponding category was calculated in each pass, incurring a big computation overhead. For our method, this computation was done only in the first pass since all further computations are done on the built feature vectors.

Therefore, the method we propose can reduce both the disk access time and computation time greatly compared to Apriori-based methods. In particular, the improvement will be especially obvious under the following conditions: i) the interested category (*IC*) involves only some portion of all categories, 2) only the value of support and confidence are changed, and the IC is kept still, 3) several queries have been made with different *IC*. In this case, most feature vectors of the categories the users may query would be built already; hence, almost no database access is needed for all subsequent queries.

We are also in designing an intelligent Pre-miner which can provide users the useful information regarding the database to be mined, like the estimated number of association rules under different settings of support and confidence. Hence, the users can make more suitable settings for the mining constraints to obtain appropriate number of rules.

References

1. R. Agrawal, T. Imielinsi and A. Swami, "Mining Association Rules between Sets of Items in Very Large Databases*," Prof. ACM SIGMOD Conf Management of Data*, pp. 207-216, 1993.
2. R. Agrawal and R. Srikant, "Fast Algorithms for Mining Association Rules in Large Databases," *Prof. 20th Int'l Conf. Very Large Data Bases*, pp. 478-499,1994.
3. S. Brin, R. Motwani, J. D. Ullman and S. Tsur, "Dynamic Itemset Counting and Implication Rules for Market Basket Data," *Proc. ACM SIGMOD*, 1997, pp. 255-264.
4. J. S. Park, M. S. Chen, and P. S. Yu, "An Effctive Hash based Algorithm for mining association rules," *Prof. ACM SIGMOD Conf Management of Data*, May, 1995.
5. M. Klementtinen, H. Mannila, P. Ronkainen, H. Toivonen and A. I. Verkamo, "Finding Interesting Rules from Large Sets of Discovered Association Rules," *Proc. CIKM*, 1994.
6. H. Mannila, H. Toivonen and A. I. Verkamo, "Efficient Algorithms for Discovering Association Rules," *AAAI Workshop on Knowledge Discovery in Databases*, pp. 181-192, 1994.
7. A. Savasere, E. Omiecinski and S. B. Navathe, "An Efficient Algorithm for Mining Association Rules in Large Databases," *Prof. 21th Int'l Conf. Very Large Data Bases*, 1995.
8. R. srikant and R. Agrawal, "Mining Generalized Association Rules," *Prof. 21th Int'l Conf. Very Large Data Bases*, pp. 407-419, 1995.
9. R. Srikant and R. Agrawal, "Mining Quantitative Association Rules in Large Relational Tables," *Prof. ACM SIGMOD Conf Management of Data*, pp. 1-12, 1996.
10. H. Toivonen, "Sampling Large Databases for Association Rules," *Prof. 22th Int'l Conf. Very Large Data Bases*, Bombay, India, 1996.
11. M. J. Zaki, S. Parthasarathy, W. Li, and M. Ogihara, "Evaluation of Sampling for Data Mining of Association Rules," Technical Report 617, CS Dept., U. Rochester, May 1996.
12. C. C. Aggarwal and P. S. Yu, "A New Framework for mining Association Rules," PODS 1998, pp. 18-24.
13. E. Cohen, M. Datar, S., et al., "Finding Interesting Association Rules without Support Pruning," ICDE, 2000, pp. 489-499.
14. K. Wang, S. Q. Zhou, S.C. Liew, "Building Hierarchical Classifiers using Class Proximity," VLDB 1999, pp. 363-374.