Mining Quantitative Association Rules under Inequality Constraints

Charles Lo and Vincent Ng

Department of Computing, Hong Kong Polytechnic University,
Hong Kong, China
cschamlo@comp.polyu.edu.hk, cstyng@comp.polyu.edu.hk

Abstract

In the past several years, there has been much active work in developing algorithms for mining association rules. However, in many real-life situations, not all association rules are of interest to the users. An user may want to find association rules which satisfy a given inequality constraint for a set of quantitative items. In other words, users are more interested in the subsets of those associations. In this paper, we present how to integrate the inequality constraints into the mining process and reduce the number of database scanning. The algorithm we present generates the large itemsets by building the expression tree and prunes away the undesired one by checking the acceptance range. In our work, we consider constraints of arithmetic inequalities which are composed of common operators such as $(+, -, *, /)$. Preliminary experimental results of the algorithm in comparison with the classical Apriori algorithm are also reported[1].

1 Overview of our work

The problem of discovering association rules was first introduced in [1]. It is to find the possible associations between items when a user specifies a minimum support (s) and a minimum confidence (c). For example, one possible association rule can be "there are 10% of customers who made long distance calls to England last month and 80% of those customers also called USA." In this example, the support is 10% and the confidence is 80%. However, the items in the above rule are binary attributes and does not posses any quantitative values. It merely reveals the existence of different items in transactions but not the possible quantitative relationships between them. Different extensions on the binary association rules have been appeared in the past several years [2, 3, 4, 5, 6, 8].
In Section 2, we introduce the important research work of mining association rules and a clear problem is stated in Section 3. Later in Section 4, we will discuss our new algorithm. In Section 5, preliminary experimental results are presented, which is followed by a conclusion of our work.

[1]The work of the authors were supported in part by the Hong Kong RGC Grant: Q120.

2 Related Work

Database mining has recently attracted tremendous amount of attention in database research because of its applicability in many areas. Many interesting and efficient association mining algorithms have been proposed. One of the most important work was written in 1993.

2.1 Mining Association Rules between Sets of Items in Large Database

Although this paper was written in 1993, it provides the fundamental idea for mining association rules [1]. It proposed the famous Apriori algorithm which can be divided into two steps.

- Find all itemsets whose supports are greater than the minimum support s and these itemsets are called *large itemsets*.
- Generate the association rules. If {ABC} and {AB} are both in the large itemsets, we compute its confidence which is (the support of {ABC})/(the support of {AB}). If the ratio is higher than a preset threshold c, the rule is established.

Apriori gives the fundamental idea of how to find out all the large itemsets effectively among thousands of transactions. However, the algorithm is only dealing with binary attributes. In practice, we usually deal with transactions containing items with positive values instead of 0 or 1. Also, Apriori only focuses on the existence of items without performing any manipulation on the input.

Later on, many different algorithms [5, 6, 7, 9] were introduced to enrich the mining methodology. In general, they improve the mining process by two ways: (1) finding the more useful association rules or (2) speeding up the mining process of association rules. In this paper, we present the

QMIC algorithm which also enhances the mining process in the similar ways.

3 Problems Statement

In practice, not all association rules are of interest to the users. In addition, databases contain quantitative attributes instead of binary attributes. A user may want to find association rules which satisfy a given inequality constraint for a set of quantitative items. We consider the constraints of arithmetic inequalities which are composed of $(+, -, *, /)$.

Let $V = I_1, I_2, \ldots, I_M$ be a set of quantitative items, and T be the transactions of a database D. For each transaction t, t[k] > 0 means that t contains item I_k with the value t[k], and t[k]=0 means that I_k does not exist in t. In our work, we are interested in finding all quantitative association rules that satisfy the user specified constraint S which consist of an *arithmetic expression*(E), an *inequality operator*(∇) and a *constant*(C). Therefore, a given constraint can be specified as $E \nabla C$, where

- E is any arithmetic expression composing of a number of unknown quantitative items and the operators $(+, -, \times, /)$ with or without brackets,
- $\nabla \epsilon \{<, >, =, \leq, \geq\}$,
- C is a scalar value.

Some examples of the inequality constraints are given below.

1. $A + B + C < 100$ - Find any three items in the database where their sum of values is smaller than 100.
2. $A - 2 * B = 0$ - Find any pair of items where one's value is the double of the other.

4 The QMIC Approach

Frequency constraint, which considers the support count of an itemset is a common criterion widely used in today's association mining algorithm. In our interest, mining association rules in a database of quantitative items, we like to consider the value constraint of the itemsets as the second criterion in the pruning process. Our strategy is to divide the pruning process into two stages. Stage 1 defines the acceptance range (AR) for the candidate itemsets in each iteration. Whenever the value

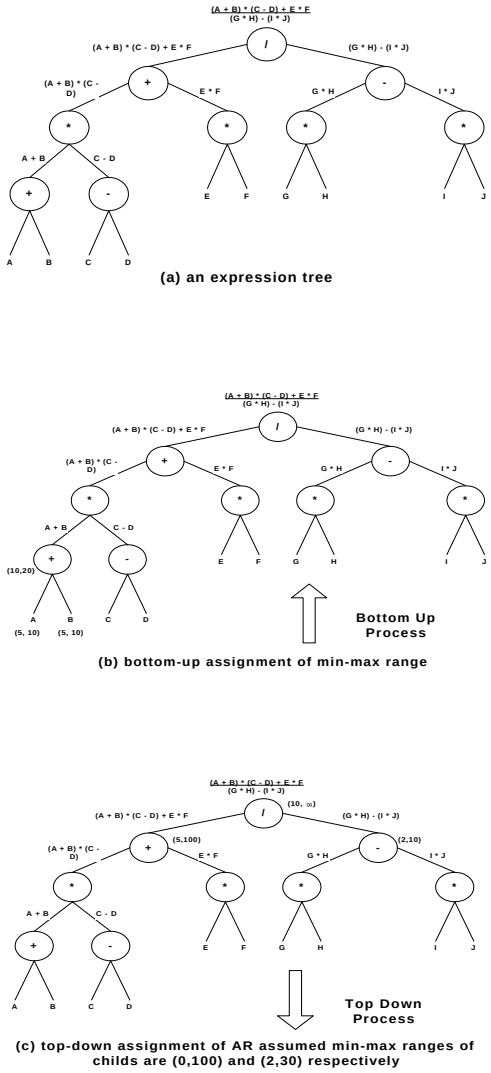


Figure 1: Building an IET.

range (VR) [2] of a candidate itemset is overlapped with any AR, it can be passed to stage 2. Otherwise, it will be pruned away without the support counting. Stage 2 is simply the original support counting which makes use of the frequency constraint. The QMIC algorithm implements stage 1 by constructing the IET.

4.1 Inequality Expression Tree (IET)

4.1.1 Expression Tree There are three steps in constructing an IET. The first step is to convert an expression E of the user specified constraint into the expression tree structure. There are two interesting properties about an expression tree built in QMIC.

- A leaf node represents any possible item in the database.
- Each internal node represents a sub-expression rooted at itself. The *itemset size* of the node represents the number of items that this sub-expression has.

For example, an internal node rooted with the expression $x + y - z$ has node size equal to three. A more detail example is shown in Figure 1a which shows the expression tree with size equals to 10.

An arithmetic expression can be represented by different expression trees. In constructing the expression tree, we have adopted a simple heuristic. The expression is separated into terms and the $(+, -)$ operators. Each term is either composed of $(\times, /)$ operators and/or brackets. Amongst the $(+, -)$ operators in the expression, we choose the center one as the root, and recursively selected the two middle ones in the left and right hand side. The same heuristic is applied to each term until the tree is completely done. The details of how the heuristic work will be given in our complete paper.

4.1.2 Minimum and Maximum Range The second step is to assign each node a *min-max range*, (Ei_{min}, Ei_{max}) which represents the minimum and maximum values of the associated expression of the node. The calculation of the min-max ranges is a bottom-up process. That is, it starts at the leaf level and processes up until the root. The min-max ranges of the leaf nodes which represent the single items are all the same and it can be simply obtained by scanning the database once. Based on the operators in the intermediate nodes, all the min-max ranges of the intermediate nodes can be calculated. In Figure 2, we show the four general formulae to calculate the ranges under the bottom-up process.

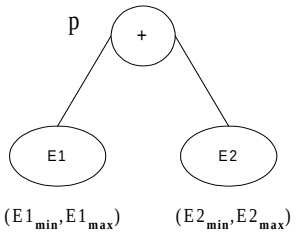
All min-max ranges of the leaf nodes, size equals to 1, are the same at the beginning. The example of an expression tree with min-max ranges is shown in Figure 1b.

4.1.3 Acceptance Range The third step is the assignments of the acceptance range (AR), (E_{low}, E_{up}) to the nodes. The AR of a node is the valid range of values that an itemset can be accepted after evaluating with the sub-expression at the node. Unlike the min-max range, the AR assignment is a top-down process which starts from the root this time. The AR of the root can be found easily since it represents the whole left-hand-side expression of the input constraint. In our continuing example, it is $(10, \infty)$. Once the root's AR is defined, other AR's can be calculated. Because of space limitation, we omit the detail of the calculation procedure but show the four general formulae for the $(\geq)$ condition in Figure 3. The other four formulae for the $(\leq)$ is skipped here.

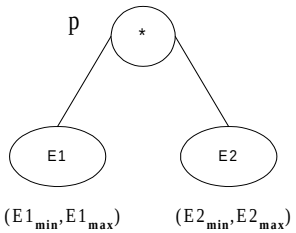
After the calculation of the ARs, the initial IET for the expression in Figure 1a is completed as in Figure 1c. It has a number of properties.

1. If the value range of a candidate itemset overlaps with any AR's of the leaf node, it is passed to stage 2 for support counting.
2. If the value range of a candidate itemset does not overlap any AR's, it is pruned away.
3. During mining, each leaf node is associated with a set of large itemsets whose size is the same length as of the sub-expression.
4. Both min-max range and AR's intervals will be adjusted after each iteration (will be discussed in Section 4.2).
5. The structure, especially the depth information, of the IET determines the Generating Mining Sequence, GMS, which will be discussed next.

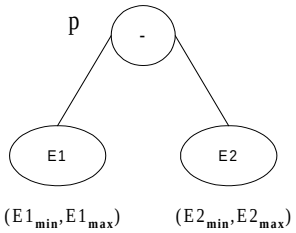
[2] The VR represents the possible values after evaluating the itemset with the sub-expression associated with the node.



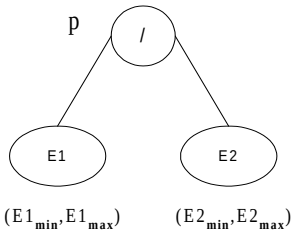
The min-max range of the parent node p is $(E1_{min} + E2_{min}, E1_{max} + E2_{max})$ respectively.



The min-max range of the parent node p is $(E1_{min} * E2_{min}, E1_{max} * E2_{max})$ respectively.

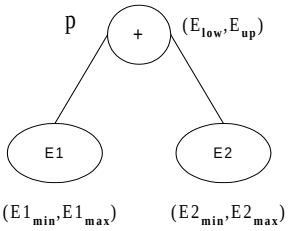


The min-max range of the parent node p is $(E1_{min} - E2_{max}, E1_{max} - E2_{min})$ respectively.

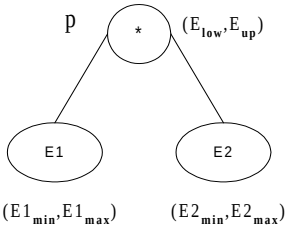


The min-max range of the parent node p is $(E1_{min} / E2_{max}, E1_{max} / E2_{min})$ respectively.

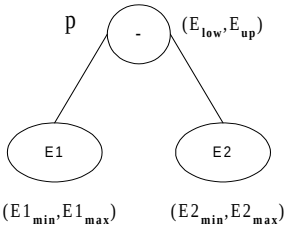
Figure 2: The four general formulae in the bottom-up process



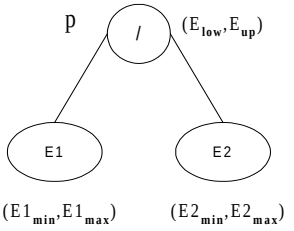
The AR range of E1 and E2 are $(E_{low} - E2_{max},E1_{max})$ and $(E_{low} - E1_{max},E2_{max})$ respectively.



The AR range of E1 and E2 are $(E_{low} / E2_{max},E1_{max})$ and $(E_{low} / E1_{max},E2_{max})$ respectively.



The AR range of E1 and E2 are $(E_{low} - E2_{min},E1_{max})$ and $(E2_{min}, E1_{max} - E_{low})$ respectively.



The AR range of E1 and E2 are $(E_{low} / E2_{min},E1_{max})$ and $(E2_{min}, E1_{max} / E_{low})$ respectively.

Figure 3: The AR calculations for the $\geq$ condition.

The first two properties are guaranteed by the general formulae. The rest will be useful in defining the GMS of the mining process.

4.2 Generating Mining Sequence

GMS is responsible to determine the sizes of the iterations during the mining process. In the Apriori algorithm, the generation sequence is simply as $1, 2, 3, \ldots$. However, this is not necessary in our case since we can skip the generation of the itemset with certain size. That is, we can shorten the generating sequence of mining. In QMIC, the depth of the IET determines the GMS. This is because QMIC generates the large itemsets by traversing from the leaf nodes of IET to its root. For the example in Figure 1c, the sequence from GMS is $\{1, 2, 4, 6, 10\}$ and it is only half of the length of the one in Apriori $\{1, 2, 3, \ldots, 10\}$.

4.2.1 AR Pruning In the iteration k of Apriori, the candidate itemset C_k is generated from the large itemset L_{k-1}. For those itemsets in C_k which have sufficient support counts, they will be in L_k. Then the iteration is completed and move to iteration $k+1$. In the QMIC algorithm, we start from the leaf nodes of IET and use the sequence from GMS to guide the candidate itemset generation steps. At each iteration, once the candidate itemsets with C_k are generated, their itemsets' VRs are verified with AR's in each leaf node. If the VRs of the itemsets in C_k does not overlap with any ARs, it should be pruned away because it fails to satisfy the inequality constraint. By finding the support counts of the rest, we can determine L_k. Unlike Apriori, the next iteration of QMIC is not necessary L_{k+1}. It will be L_{k+l} where l is the size of the siblings of the current node. In other words, if both large itemsets associated with child nodes are available, the next iteration is to generate the large itemsets to be associated with their parent. That is why the depth of the IET defines the GMS.

4.2.2 The QMIC Algorithm In this section, we show the complete QMIC algorithm below. Note that the notations used are shown in Figure 5. The details of candidate itemset generation and its support counting is omitted because their functions are similar to those in Apriori.

1. Given the database D and the completed IET, scan the database for L_1 and VR_1.
2. Itemset generation of C_2 from L_1
3. i:=2
4. Repeat {
   - delete the LN_i from the IET
   - for x = 1 to sizeof(C_i)
     for y= 1 to sizeof(LN_i)
     - if $((VR_{ix} \cap AR_{iy}) \neq \phi)$ append C_{ix} into C'_i
   - support counting of the C'_i to obtain the large itemsets L_i
   - /*adjust the AR of the parent node of LN_i*/
     - for any node associated with the L_i
       * update the min-max range of their parent node
       * update AR of parent since their min-max range has been updated
   - /*update the IET and its parameter*/
     - for any pair of leaf nodes (LN_i, LN_j) which are siblings
       * i= min (sum of the node size of each pair)
   - itemset generation of C_i from the large itemsets which are represented by LN_i and LN_j

   } while depth of the IET > 1

Figure 4: The QMIC Algorithm.

5 Experiments

To evaluate the performance of QMIC, we compare it to a modified Apriori algorithm which has

Notation	Meaning
LN_m	leaf nodes with node size equal to m
LN_{mn}	specify nth leaf node in LN_m
AR_m	acceptance range set of the leaf nodes LN_m
AR_{mn}	acceptance range of the specified leaf node LN_{mn}
VR_i	value range set of all the itemsets with size i
VR_{ij}	value range of a specify jth itemset in VR_i
L_i	large itemsets with size i
C_i	candidate itemsets with size i
C_{ij}	jth candidate itemset in C_i
C'_i	candidate itemsets which satisfy the user constraint
N_k	tree node which represent the large itemsets

Figure 5: Notations for the QMIC Algorithm.

been enhanced to handle the constraint mining. The experiment was conducted under the constraint size of 5, 10 and 15. And the result of the size 10 is shown in following figures. We notice that it is not sufficient to determine the efficiency of the algorithms by only counting the number of steps in generating the candidate itemsets.

In order to assess the performance of the algorithms, we have implemented the Apriori algorithm and the QMIC algorithm in C++. We ran our preliminary experiments on a Solaris workstation with 32 Mbytes of main memory. There are two sets of experiments. The first set of experiments fixed the number of items as 1000 and experimented with different database sizes. The corresponding results is shown in Figure 6. The second set experiments is also based on a fixed number of items (1000) but with a more complicated user specified constraint which involved more than 15 different items The results are shown in Figure 7. The results show us that the number of transactions has a large impact to the database. This is reasonable since we have to scan the whole database once for each size-k of the large itemset L_k. The proposed algorithm skipped the generation of many unnecessary candidate itemsets. So, the more transactions in the database, the better the algorithm can save time. As we can see from the result, the QMIC works better for larger database. In other words, it becomes even more effective when the size of the database is getting bigger.

6 Conclusion

In this paper, we present how an inequality constraint can be integrated into the mining of association rules. The QMIC algorithm is proposed. It improves the association mining algorithm in two areas: 1) reduce the size of the candidate itemsets in each iteration, 2) reduce the number of database scans. These are done by introducing the concept of IET and GSM. The preliminary

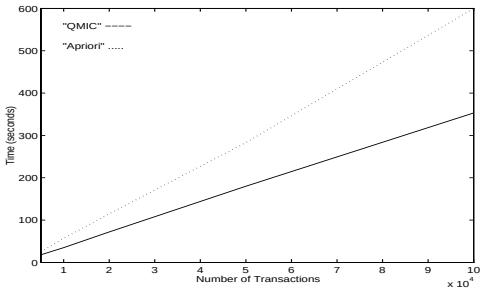


Figure 6: Discovery time for different number of transactions.

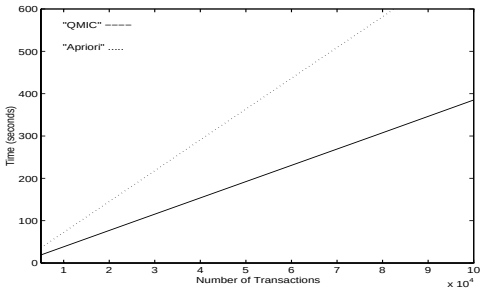


Figure 7: Discovery time for different number of transactions.

experimental results showed that the QMIC outperformed the Apriori algorithm more than 100% when there are more than 100 thousand transactions in the database.

REFERENCES

[1] R. Agrawal, R Srikant. *Fast Algorithm for Mining Association Rules in Large Databases*, In Proc. VLDB'94, Santiago, Chile, September 1994.

[2] R. Agrawal, R Srikant. *Mining Sequential Patterns,* In Proc. International Conference on Data Engineering, 1995, pp. 3-14.

[3] R. Agrawal, T Imielinski, A Swami. *Mining Association Rules between Sets of Items in Large Databases*, In Proc. ACM-SIGMOD International Conference on Management of Data, 1993, pp. 207-216.

[4] DW Cheung J Han, VT Ng, CY Wong. *Maintenance of Discovered Association Rules in Large Databases: An Incremental Updating Technique*, In Proc. International Conference on Data Engineering, New Orleans, USA, 1996, pp. 106-114.

[5] M Klemettinen, H Mannila, P Ronkainen, H Toivonen, AI Verkamo. *Finding interesting rules from large sets of discovered association rules,* Proc 3rd International Conference on Information and

Knowledge Management, pp. 401-408, Gaithersburg, Maryland, November 1994.

[6] M Houstma, A Swami. *Set-Oriented Mining of Association Rules*, Technical Report RJ 8567. IBM Almaden Research Lab., San Jose, CA, October 1993.

[7] RJ Miller and Y Yang. *Association Rules over Interval Data*, In Proc. ACM-SIGMOD 1997, pp. 452-461.

[8] R Ng, L Lakshmanan, J Han, A Pang. *Exploratory Mining and Pruning Optimizations of Constrained Association Rules*, In Proc. ACM-SIGMOD 1998.

[9] R Srikant, Q Vu and R Agrawal. *Mining Association Rules with Item Constraints*, KDD'97, pp. 67-73.

[10] R Srikant and R Agrawal. *Mining Quantitative Association Rules in Large Relational Tables*, In Proc. ACM-SIGMOD 1996.