Experiences of Using a Quantitative Approach for Mining Association Rules

Liang Dong and Christos Tjortjis

Department of Computation, UMIST, PO Box 88, Manchester, M60 1QD, UK
christos@co.umist.ac.uk

Abstract. In recent years interest has grown in "mining" large databases to extract novel and interesting information. *Knowledge Discovery in Databases* (KDD) has been recognised as an emerging research area. Association rules discovery is an important KDD technique for better data understanding. This paper proposes an enhancement with a memory efficient data structure of a quantitative approach to mine association rules from data. The best features of the three algorithms (the *Quantitative Approach*, *DHP*, and *Apriori*) were combined to constitute our proposed approach. The obtained results accurately reflected knowledge hidden in the datasets under examination. Scale-up experiments indicated that the proposed algorithm scales linearly as the size of the dataset increases.

1 Introduction

Progress in data acquisition and storage facilitated the explosive growth in the amount of data collected by businesses. The impetus to effectively harness the increased volumes of data now available has lead to the need for new data analysis techniques to build data characterisations and extract useful patterns and models. Consequently, the research field of KDD, also known as data mining, has arisen with *mining association rules* becoming one of the most prominent techniques within the context of extracting relationships among items hidden within datasets.

A promising algorithm for mining association rules in terms of accuracy and performance is the *Quantitative Approach* [9]. It generates the set of frequent itemsets by first partitioning the values of quantitative attributes and then using an interesting measure to prune any uninteresting candidate itemset.

In this paper, we present a combined approach which consists of various parts from existing algorithms, such as the *Quantitative Approach*, the hash-based technique from *DHP* (Direct Hashing and Pruning) algorithm [5], and the methodology for generating association rules from *Apriori* algorithm [2]. We present experimental results showing that the proposed approach precisely reflects the information hidden in the datasets. As the size of the dataset increases, the proposed approach scales-up linearly in terms processing time and memory usage. In section 2, some algorithms used within the framework of association rules are investigated. Encouraging results obtained by comparing the Quantitative Approach to other association rule algorithms are outlined in Section 3. The experimental results are analysed in Section 4. Conclusions and future work are discussed in Section 5.

J. Liu et al. (Eds.): IDEAL 2003, LNCS 2690, pp. 693–700, 2003.

2 Related Work

This section aims at investigating six prominent algorithms for mining association rules: the *AIS* [1], *SETM* [2], *Apriori* [2], *AprioriTid* [2], *Quantitative Approach* [7], and *Boolean* Algorithm [10]. Advantages and disadvantages for each of them are discussed in terms of speed, accuracy, and suitability.

- Speed

Finding all frequent itemsets greatly affects the speed of algorithms, because it requires one or more database scans, which result in a time overhead for I/O connections to a database stored in a secondary storage device. Such overhead is much greater than that because of computation in the CPU [2].

In both *AIS* and *SETM* algorithms, candidate itemsets are generated "on-the-fly" during the first pass as data are being read. This results in unnecessarily generating and counting a lot of invalid candidate itemsets thus wasting a lot of time. In general, *Apriori*, *AprioriTid* and *Quantitative Approach* consider the itemsets found to be frequent in the previous pass, with no need to access the database again. Consequently, fewer candidate itemsets are generated. Taking the above statements into consideration, the *Apriori*, *AprioriTid*, and *Quantitative Approach* are superior to *AIS* and *SETM* for all problem sizes.

Although *Apriori* counts too many small sets in the second pass, this wastage decreases dramatically from the third pass onwards [2]. However, each iteration in *Apriori* requires a pass of scanning the database, incurring a severe performance penalty. The *Quantitative Approach* is similar to the *Apriori* algorithm, in this respect.

In the *AprioriTid* algorithm a pass over the database is replaced by a pass over the set of candidate itemsets associated with *TID*s[1] after the first pass. Hence, *AprioriTid* is more effective in later passes when the size of this encoding can become much smaller than that of the database. When the sets of this encoding fit in memory, *AprioriTid* is superior to *Apriori*; otherwise, *Apriori* beats *AprioriTid*.

The *Boolean* algorithm produces frequent itemsets without constructing candidate itemsets. In contrast, this construction of candidate itemsets is required by the "Apriori family" algorithms (for example, *AIS*, *SETM*, *Apriori*, *Quantitative Approach*, and *AprioriTid*). As a result, the *Boolean* algorithm should outperform the "Apriori family" algorithms for all problem sizes as reported on the literature [10]. However, when we applied this algorithm to mine quantitative association rules in a large dataset, it was found to be unsuitable for solving such problems (see Section 3).

- Accuracy

The *AIS*, *SETM*, *Apriori*, *AprioriTid*, and *Boolean* algorithms can only deal with Boolean Association Rules problems, while the *Quantitative Approach* can also deal with Quantitative problems. Since a transactional database often has richer attribute types like quantitative and categorical attributes, taking into account only Boolean attributes should result in a heavy loss of valuable information. Thus the *Quantitative Approach* is expected to be more accurate than the other algorithms discussed here.

- Suitability

Different algorithms suit different domains. The *AIS* and *SETM* are only effective in small databases [2]. As descendants of the *AIS* and *SETM*, the *Apriori* and *AprioriTid*

[1] It stands for Transaction Identifier, which is unique for each transaction in a dataset.

not only perform well in small databases, but also more efficiently in medium size databases [2]. The *Quantitative Approach* [7] is tailored to large databases.
Another important factor seldom mentioned in the literature is the memory usage, when different algorithms are applied to the same dataset [2, 7]. We investigate the scaling-up property of our approach on memory usage and discuss experimental results in Section 4.3.

3 Approach

We considered the following requirements to select a suitable algorithm.

- The algorithm should generate as many *interesting*[2] association rules as possible. All frequent itemsets must be identified based on the *minimum support threshold* specified by the end-users.
- The algorithm must have the ability to deal with quantitative and categorical values in addition to Boolean ones.
- The algorithm must perform well in medium or large databases.

Based on these requirements and the characteristics of the algorithms discussed in section 2, we decided to keep the advantage of *Quantitative Approach* in dealing with multiple data types and enhance this by using the *Boolean* algorithm to create the frequent itemsets.
However, a major shift on the initial selection and design of this approach arose. After investigation of the storage pattern involved in the *Boolean* algorithm, it was concluded that memory resources are wasted when the algorithm stores the quantitative and categorical values in the form of a truth table.
For example, suppose we have a table with three attributes that have a, b, and c possible values, respectively. As the *Boolean* algorithm stores the table content in the form of truth tables, $a \times b \times c$ truth tables are needed for all possible combinations of values for each attribute. This wastes memory resources especially as the datasets size grows. High performance achieved by the *Boolean* algorithm is offset by this deficiency. Consequently, the *Boolean* algorithm was abandoned in order to achieve better memory utilisation at the expense of processing speed.
Our final approach, therefore, can be decomposed into three parts (the first two parts are similar with the respective phase of the *Quantitative Approach*): a) Pre-processing the input dataset (such as the partition operation), b) creating frequent itemsets (the hash-based technique proposed by the *DHP* algorithm was introduced into this phase to reduce the number of the candidate k-itemsets examined), and c) generating association rules (this phase can be found in the *Apriori* algorithm). This approach was the most appropriate to satisfy all the requirements stated above. Nevertheless, the *Boolean* and *Quantitative Approaches* can be treated as a complementary alternative, in cases when the number of different attribute values is quite small.

[2] An association rule is *interesting* if it is unexpected and/or actionable [6].

3.1 Approach Decomposition

Let $I = \{i_1, i_2, \ldots, i_m\}$ be a set of attributes. Let P express the set of positive integers. Let I_v express the set $I \times P$. Let I_R express the set $\{<x,l,u> \in I \times P \times P \mid l \leq u$, if x is quantitative; $l=u$, if x is categorical$\}$ [7].
As a result, a triple $<x,l,u> \in I_R$ represents an item which refers to either a quantitative attribute x with a value in the range $[l,u]$ or a categorical attribute x with a value l.

For any $X \subseteq I_R$, attributes (X) expresses the set $\{x \mid <x,l,u> \in X\}$. $\dot{X}$ is defined as a generalisation of X, if attributes (X)=attributes $(\dot{X})$ and $\forall x \in$ attributes(X) $[<x,l,u> \in X \wedge <x,l',u'> \in \dot{X} \Rightarrow l' \leq l \leq u \leq u']$.
As mentioned above, this approach can be decomposed into three major phases: data pre-processing, creating frequent itemsets, and generating association rules.

3.1.1 Data Pre-processing

This can be divided into two tasks.

1) Decide the number of partitions for each quantitative attribute.
   Assuming equi-depth partitioning [7], we get:

$$\text{Number of Intervals} = \frac{2 \times n}{\min \sup \times (K-1)} \tag{1}$$

   where K refers to partial completeness level which gives a handle on the amount of information lost by partitioning, *minsup* is the user supplied minimum support, and n means the number of quantitative attributes participating in the partition activity.
   When K increases, there are fewer intervals (candidates for frequent itemsets) for the quantitative attributes. As a result, the number of frequent itemsets decreases. The number of rules based on those frequent itemsets decreases accordingly.
   For categorical attributes, map the values to a set of consecutive integers. For quantitative attributes partitioned into intervals, the ranges are mapped to consecutive integers; otherwise the values of quantitative attributes are simply mapped to consecutive integers. In both cases, the order of ranges or values is preserved. The set of mapping rules, which are used throughout the rest of the algorithm, is then established.
2) Calculate the support for each value or range of the attributes.
   In addition, adjacent values of quantitative attributes are combined as long as their support is less than the user-specified *maximum support*. All corresponding mapping rules are updated immediately to reflect the current state.

3.1.2 Creating Frequent Itemsets

This phase is similar to that of the *Apriori* algorithm except that the *interesting measure* is adopted at the end of the first database scan to prune all candidate 1-itemsets whose support is greater than $\frac{1}{R}$ in the partitioned quantitative attributes,

where R refers to the interesting measure supplied by the user [7]. The frequent 1-itemsets are then identified as long as their support values are greater than, or equal to, the user-specified minimum support threshold and they pass through the examination of the interesting measure.
The subsequent *join-prune* procedure is composed of two steps: *Join* and *Subset Prune*. First, candidate itemsets are generated on the basis of the frequent itemsets found during the previous database scan. Let C_k represent the set of candidate k-itemsets and L_k the set of frequent k-itemsets. C_k is produced by joining any two different $L_{(k-1)}$ as long as their first lexicographically ordered (k-2) items are identical. Then, all candidate itemsets in C_k that have at least one (k-1)-subset not in L_{k-1} are removed. Next, a database scan calculates the support for each candidate itemset in C_k. Any candidate itemsets with support below the minimum support value are pruned, yielding the set of frequent itemsets L_k. Such set is stored into a *hash tree* in order to be used during the generation process of $C_{(k+1)}$. This procedure terminates when L_k becomes empty.

3.1.3 Generation of Association Rules

In this last phase, all strong association rules are created from frequent itemsets. The general idea can be expressed as follows:
First, generate all non-empty subsets for each frequent itemset *l*. Then, for every non-empty subset of *l*, the rule "$s \Rightarrow (l\text{-}s)$" holds if the ratio of

$$\frac{\sup\ port\ \_\ count(l)}{\sup\ port\ \_\ count(s)} \geq \frac{\min\_conf}{K} \tag{2}$$

in which *min_conf* is the minimum confidence supplied by the user. The confidence for those rules can be easily obtained by accessing the hash tree established in earlier stages.

4 Experimental Results

To evaluate the effectiveness of our approach, we performed several experiments on a personal computer with CPU clock rate 800 MHz, 128MB of main memory, and running Microsoft Windows 2000. The performance of our approach was assessed in terms of parameter testing, scale-up testing, and memory usage testing. All the testing datasets used here were obtained from the UCI repository [9].

4.1 Parameter Testing

In order to assess the results of this approach in the case of some critical parameters varying, a public dataset, called *Abalone*, was used. It has eleven attributes: eight quantitative and three categorical. There are 5,000 records in this dataset with no missing values for any of the attributes.

Figure 1 shows the results obtained after performing twelve tests, in each of which the minimum support was set to 10%, maximum support to 20%, and minimum confidence to 30%. As expected in Section 3.1.1, the number of rules decreases as the partial completeness level increases.

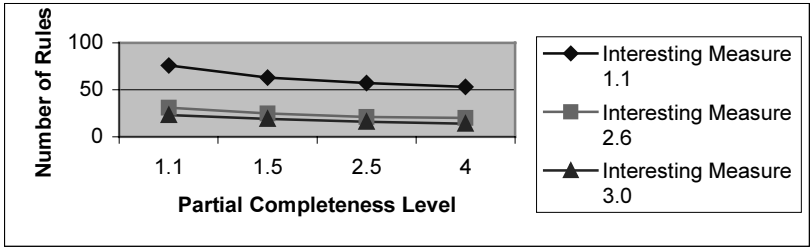

Fig. 1. Parameter Testing

4.2 Scale-up Testing

In order to evaluate the scale-up property of this approach, fifteen tests were performed using datasets with different sizes but the same structure as *Abalone* for comparison. In each of these tests, the partial completeness level was set to 1.1, interesting measure to 1.5, maximum support to 3%, and minimum confidence to 20%. Figure 2 shows the results obtained.

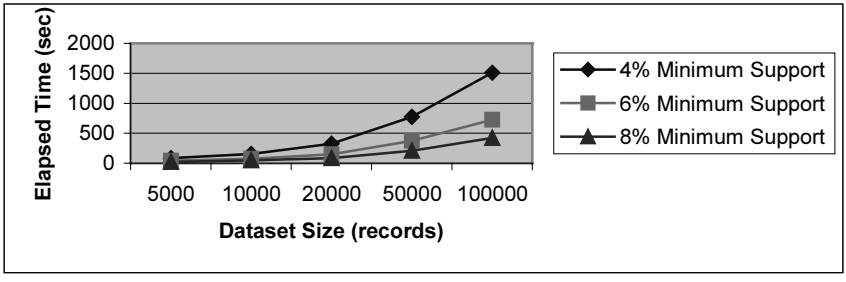

Fig. 2. Scale-up Testing

Increased minimum support prunes a number of frequent itemsets. Consequently, fewer rules are generated and the time spent on each phase decreases accordingly. Figure 2 confirms this by showing that the running time for this approach decreases as the minimum support increases. Figure 2 also shows that the whole processing time scales linearly as the size of the dataset increases from 5,000 records to 100,000 records by appending a number of different records to the same dataset.

4.3 Memory Usage Testing

Twelve tests used datasets with various sizes but the same structure as *Abalone*. That was achieved but reproducing randomly parts of this dataset. In each test, partial completeness was set to 1.1, maximum support to 8%, and interesting measure to 1.5. As shown in Table 1, similar results were achieved by varying the minimum support and confidence. Our approach uses memory when loading the dataset, saving the set of mapping rules, saving the frequent itemsets, and saving the association rules. As the dataset increases in size, the number of mapping rules, frequent itemsets, and association rules does not necessarily increase. When the size of the dataset becomes large enough, it will dominate the memory usage. Table 1 confirms this by showing that the total amount of memory usage does not double when the size of the dataset increases from 5.000 to 10.000 neither from 10.000 to 20.000 records.

Table 1. Memory Usage Testing (*Minsup* and *minconf* mean minimum support and confidence respectively)

Dataset Size	Total Memory Usage (MB)		
	Minsup=10% Minconf=10%	Minsup=10% Minconf=20%	Minsup=20% Minconf=20%
5.000	7.55	7.57	7.54
10.000	12.98	11.916	13.03
15.000	15.98	15.964	16.00
20.000	21.81	22.01	22.02

5 Conclusions and Further Work

We proposed an effective approach for mining association rules in large transactional databases. The main advantage of this approach over other prominent algorithms (such as *Apriori* and *AprioriTid*) is that it can deal with both Quantitative and Boolean Association Rule problems.

We also presented an alternative solution for the same problem: the enhancement of the Quantitative Approach using the part of the Boolean algorithm which creates frequent itemsets. This alternative was argued to be suitable for the cases when the number of potential different values for each attribute is small.

We conducted several experiments using datasets with different sizes. The results indicate that the performance of this approach can be regarded as satisfactory, as compared to the performance of the *Quantitative Approach*. We also concluded that the memory usage of this approach heavily depends on the size of the dataset.

A number of further improvements to the current approach are as follows:

- *Incorporation of alternative data structures*: The hash tree structure was chosen as part of our approach. However, it would be interesting to test other data structures used for mining association rules (for example, R^*-tree [3]).
- *Support for missing values*: We assumed that the input dataset has no missing values. However, this is not always the case in real life and missing values are also worth of investigation [8].

- *Improving the partitioning method*: the equi-depth partitioning method used performs well on evenly-distributed datasets. However, in the case of highly-skewed datasets, adjacent values whose behaviour would typically be similar might be split into different intervals. Consequently, for such cases we could use the *maximal appropriate abstraction* for partitioning numerical values into maximally extended intervals whose similarity is measured by the *interclass variance* between interval classes [4].
- *Incorporation of an incremental approach*: As our approach does not enable very large datasets to be loaded into the main memory as a whole, incorporation of incremental input could be beneficial.
- *Locating the breakpoint of the scale-up property*: Although the whole processing time of our approach scales linearly with the size of the dataset ranging between 5,000 and 100,000 records, it is worth finding the point when processing time starts to increase exponentially.

References

1. Agrawal, R., Imielinski, T., and Swami, A., "Mining association rules between sets of items in large databases". *Proc. ACM SIGMOD Conf. on Management of Data*, 1993.
2. Agrawal, R., and Srikant, R., "Fast Algorithm for Mining Association Rules in Large Databases", *Proc. Int'l Conf. on VLDB*, pp. 487–499, 1994.
3. Beckmann, N., Kriegel, H.-P., Schneider, R., and Seeger, B., "The R*-tree: an efficient and robust access method for points and rectangles", *Proc. of ACM SIGMOD*, pp. 322–331, 1990.
4. Narita M., Haraguchi M., and Okubo Y., "Data Abstractions for Numerical Attributes in Data Mining", *Proc. 3rd Int'l Conf. Intelligent Data Engineering Automated Learning,* 2002.
5. Park, J. S., Chen, M. S., and Yu, P. S., "An Effective Hash Based Algorithm for Mining Association Rules", *Proc. of the ACM SIGMOD*, pp. 175–186, 1995.
6. Silberschatz A. and Tuzhilin A., "On Subjective Measures of Interestingness in Knowledge Discovery", *Proc. Of the 1st Int'l Conf. on Knowledge Discovery and Data Mining*, 1995.
7. Srikant, R., and Agrawal, R., "Mining Quantitative Association Rules in Large Relational Tables", *Proc. of the ACM SIGMOD Conf. on Management of Data,* 1996.
8. Tjortjis C. and Keane J.A, "T3: an Improved Classification Algorithm for Data Mining", *Proc. 3rd Int'l Conf. Intelligent Data Engineering Automated Learning,* 2002
9. UCI ML Repository, <http://www.ics.uci.edu/~mlearn/MLRepository.html>, last accessed: 15 September 2002.
10. Wur, S.Y., and Leu, Y., "An Effective Boolean Algorithm for Mining Association Rules in Large Databases", *Proc. 6th Int'l Conf. on Database Systems for Advanced Applications,* 1998.