Volume 1, No. 2, September 2010
Journal of Global Research in Computer Science
ISSN-2229-371X
RESEARCH PAPER
Available Online at www.jgrcs.info

Mining association rules for clustered domains by separating disjoint sub-domains in Large Databases

Kanhaiya Lal[1] and N. C. Mahanti[2]

[1]Department of Computer Science, BIT, Patna Campus, Patna, India
[2]Department of Applied Mathematics, BIT, Mesra, Ranchi, India

Abstract: Association rule mining algorithms focus on the discovery of valid rules by testing all the items or elements in the domain, rather testing some known elements, which makes the process inefficient as it generates a very large number of candidates. Also, most algorithms take multiple passes over the database and this results a very high I/O cost. As the database is disk resident and can't be managed completely in main memory, multiple passes over the database reduces the performance considerably for any known association rule mining algorithm . The proposed solution to this problem is to separate disjoint sub-domains, which are co-related.
In this paper we concentrate over
1. The separation of sub-domains which are composed of co-related elements in the domain.
2. Database summarization.
The items of a large domain correlate with each other forming small sub groups i.e. the domain is clustered in small groups [3]. This property appears in many real world cases, e.g. Bioinformatics, e-commerce etc [3]. The element of a sub- group can be processed for discovery of association rules as in this case the size of candidate set is comparably small to the exponential size of candidates of Apriori in pure form. Most algorithms take multiple passes over the entire database which results in inefficiency and high IO overheads. The proposed algorithm maintains a list of transactions that are related with the component in processing and hence only those transactions are processed instead of the entire database.

Keywords: Pattern Clustering; Localized Pattern; Domain Partitioning; Database Partitioning;

INTRODUCTION

The mining of association rules involves the discovery of significant and valid correlations among items that belong to a particular domain [1]. The development of association rule mining algorithms has attracted a remarkable attention during the last years [2], which are concerned on the efficiency and scalability.
Association rule mining, as originally proposed in [1] with its Apriori algorithm , has developed into an active research area. Many additional algorithms have been proposed for association rule mining [5], [6], [7] and [8]. Also, the concept of association rule has been extended in many different ways, such as **generalized association** rules, association rules with item constraints, sequence rules [9] etc. Apart from the earlier analysis of market basket data, these algorithms have been widely used in many other practical applications such as customer profiling, analysis of products and so on. We have used these algorithms in innovative applications such as warranty claims analysis and inventory analysis. Several commercial data mining tools now offer variants of the association rule mining algorithms.
End users of association rule mining tools encounter several well-known problems in practice. First, the algorithms do not always return the results in a reasonable time. Typically, this happens because the algorithms generate an exponential number of candidate frequent itemsets. Although several different strategies have been proposed to tackle efficiency issues, they are not always successful. Also, in many cases, the algorithms generate an extremely large number of association rules, often in thousands or even millions. Further, the association rules are sometimes very large. It is nearly impossible for the end users to comprehend or validate such large number of complex association rules, thereby limiting the usefulness of the data mining results. Several strategies have been proposed to reduce the number of association rules, such as generating only "interesting" rules, generating only "non-redundant" rules, or generating only those rules satisfying certain other criteria such as coverage, leverage, lift or strength. While these are promising strategies, none of them seem to sufficiently "compress" or reduce the generated association rules, for easy comprehension by end users.
And the second important problem is processing exponential number of candidates. This can be solved using the feature of correlated patterns. A database can be viewed as a group of small groups (clusters or small worlds). This is supported by the fact that the elements of the clustered domains are related closely to some elements defining a close proximity. In several cases it can be expected that items will correlate with each other (i.e., create patterns) in a way that small worlds [20] are formed.

RELATED WORKS

Since its introduction [1], the problem of association-rule mining has been studied thoroughly[10]. Earlier approaches are characterized as Apriori-like, because they constitute improvements over the Apriori algorithm [11]. They explore the search space in a BFS manner, resulting in an enormous cost when patterns are long and/or when the domain size is large. Techniques that reduce the number of candidates [12] may improve the situation in the latter case, but are unable to handle a very large domain. Algorithms for mining generalized association rules are also Apriori-like. They do not operate on individual items but they focus on categories at different levels, by assuming that the items are categorized hierarchically. Therefore, a large domain

can be replaced by a much smaller number of categories. Nevertheless, this approach requires the existence of a predetermined hierarchy and the knowledge of the items that belong in each category. In contrast, we focus on a kind of grouping that is determined by the in-between correlations of items, which is not predefined and not hierarchical

A different paradigm consists of algorithms that operate in a DFS manner. This category includes algorithms like Tree Projection [13], Eclat [14], and FP-growth, whereas extensions have been developed for mining maximal patterns. Eclat uses a vertical representation of the database, called covers, which allows for efficient support counting through intersections of item's covers. For large domains, this may lead to a significant overhead, since a large number of intersections have to be computed. FP-growth uses a prefix tree, called FP-tree, as a condense representation of the database and performs mining with a recursive procedure over it. Both Eclat and FP-growth require that their representations have to fit entirely in main memory. For large domains, the aforementioned requirement may not always hold. Nevertheless, as will be shown, for very large domains this technique results in high execution times, since projection is performed for each item.

All the previous methods are based on column enumeration (also denoted as feature enumeration). For mining closed patterns over databases with much more columns than rows, e.g., gene-expression data, row enumeration has been proposed. CARPENTER [15] is such an algorithm, which finds closed patterns by testing combination of rows. CARPENTER works on the basis that its entire database representation is held in main memory, which holds for relatively small data sets like microarray data. As the number of rows increases, pure row enumeration may become inefficient.

Other algorithms that are influenced by the characteristics of gene-expression data include FARMER algorithm , which mines a particular type of closed patterns from microarray data, and BSC- and FIS-trees [16], which mine regular association rules from gene-expression data. Although [16] shows an improvement over FP-growth, it is limiting because it runs in main memory and only considers data sets with few hundred transactions.

Finally, the concept of finding localized patterns is presented in [17]. This work inspired us to consider that patterns may not always exist globally. Instead, in several applications the data set may contain clusters that provide their own (local) patterns. Nevertheless, [17]is based on the different direction of clustering the transactions, whereas we are interested in clustering the domain. For this reason, [17] does not pay attention to the problems resulting from a large domain.

Algorithm

The proposed method of discovering association rules progress phase by phase. The proposed method can be informally stated as::

Phase 1

1. The database is partitioned in small sub databases according to the available computational memory, which can be processed entirely once at a time. This approach is adopted from [4].
2. This partitioned database is scanned to obtain localized patterns[3].
3. All obtained localized patterns from the different partitions are merged to obtain disjoint patterns.
4. These disjoint patterns give the groups of items which can be used as the base itemset for candidate generation.

Phase 2

1. These localized patterns are processed one at a time.
2. A special number **pattern_num** is added to the transactions in the database, which represent the localized pattern they belong to.
   At the time of processing of some localized pattern only those transactions will be searched/ processed whose **pattern_num**
   Matches with that of the localized pattern under processing.

The Algorithm can be represented as :

Algorithm: mine_database
Input: D, where D = {T1, T2, T3, T4,……. Tn}
Output: X → Y,
Pi = partition_database (D), where i= 1 to m
LPj = find_patterns (P1,P2,…..,Pm) where j = 1 to k
for LP_1 to LP_j
S_i= group transactions with same pattern_num that of the localized pattern
generate_rule (LP_i)
end

The database is partitioned as the entire database is disk resident and very large which can't be handled by the present hardware i.e. entire database can't be kept in the memory at a time. Thus, the divide and conquer approach is applied and the database is partitioned in small partitions which can be accommodated in the memory in entirety at one time. The partitioning approach is same as presented in [4].

As mentioned, it is not practical to generate all the candidates from all the elements (items in the market basket data) present in the database. It will result in the exponential number of candidates, and testing support and confidence for each of them is not practical. Hence it is necessary to reduce the number of items, that are above the minimum support. This is done by identifying the localized patterns which are related to each other and each such component is such a group which can be processed for all valid rules present within them. The algorithm can be stated as:

Algorithm: find_patterns(P1,P2,…..,Pm) where j = 1 to k
Sup=minimum_support
Begin
Initialize the graph structure
Foreach partition P_i
foreach transaction in P_i

```
        foreach pair of items(i, j) in T
if i,j belong to the same component do nothing
else if i and j belong to different components C and C'
find support of i and j
else if support of i and j is greater than sup
merge_components C and C' as C''
return C''
end
```

A component C refers to a disjoint set hence, the meging can be done using disjoint set data structures. Each different component C returned by the algorithm is a separate domain containing items which are non overlapping. All items present in one component form a localized pattern. Thus, the entire domain size is reduced and divided into small groups which can be used as base 1-itemset and can be mined for valid rules.

Lemma 1: The total space requirement for the algorithm find_pattern algorithm is given as $O(gs^2)$, where g is the pace required by one node and s is the number of items in the domain[Appendix].

The next stage of this algorithm is focused on summarizing the database for efficient processing. Apriori needs frequent passes over the database to generate the frequent itemset which is not possible with disk resident database as it will result in a very high IO cost. This can be reduced if the transactions which are related to one component are present when processing the component. i.e. transactions are grouped separately for each separate component/ localized pattern. The next stage of mining association rules can be merged in this phase of algorithm. Algorithm can be represented as :

Algorithm: summarize_database(component C)

```
Begin
For all partitions Pi
Begin loop
For all transactions Ti
If Ti contains any item i which is in C
Add pattern num to Ti
End loop
Forall items i in C
Generate_rules(C, Si)
```

For generation of valid rules we can use any efficient frequent itemset mining algorithm based on Apriori.[1].

Lemma 2: Maximum number of pattern_num associated with each transaction can't be greater than the total number of transactions in a partition [Appendix].

CONCLUSIONS

The proposed method employees the divide and conquer approach in successive stages and as a result a very large database is treated as a optimal chunk of data which can be handled efficiently. Also, the number of possible candidates is educed to a remarkable extent by partitioning the domain into localized patterns, which is because of the presence of the clusters.

The proposed algorithm generates frequent itemsets for each localized pattern or the clustered sub domain one by one. This method can be employed to obtain valid rules for the desired localized pattern, which is useful when the end user needs the rule to be generated for only some elements of the domain and not for the entire elements. This solves the problem of generation of large number of rules which may not seem much useful to the end user.

The independent processing of localized patterns adds an additional field pattern_num to the transactions which contain the elements of the pattern. The transactions without this number are neglected when the frequent itemset mining algorithm processes the transactions for calculating support and confidence. This saves the computational effort at the cost of one additional field 'pattern_num'.

The proposed approach is highly parallelizable, as each clustered sub domain can be processed individually and independently.

APPENDIX

Lemma 1: The total space requirement for the algorithm find_pattern algorithm is given as $O(gs^2)$, where g is the pace required by one node and s is the number of items in the domain.

Proof

Step 1: For a given set $A = \{x_1, x_2, \ldots\ldots\ldots, x_n\}$ the total number of pairs generated is n(n-1)/2. Formally, this can be represented as:

$$|A| = n(n-1)/2 = 0.5 * (n^2 - n);$$

Which can be expressed as $O(n^2)$. As the total number of the elements in one partition (ie. the domain size in the partition) is s, the space requirement is $O(s^2)$.

Step 2: Assuming that the support of all the pairs generated is greater than the minimum support threshold. Then the total number of nodes needed to represent the graph is equal to the size of domain. In our case the domain size is s. Hence, the space required to represent this graph structure in memory is $\mathbf{s^2 * g}$ **,** where g is the space needed to represent one node. This can be represented as $\mathbf{O}(gs^2)$.

From above we can see the space requirement of this complete algorithm is $\max\{O(s^2), O(gs^2)\} = O(gs^2)$ for $g >= 1$

The next stage of this algorithm is focused on summarizing the database for efficient processing. Apriori needs frequent passes over the database to generate the frequent itemset which is not possible with disk resident database as it will result in a very high IO cost. This can be reduced if the transactions which are related to one component are present when processing the component. i.e. transactions are grouped separately for each separate component/ localized pattern. The next stage of mining association rules can be merged in this phase of algorithm. Algorithm can be represented as :

```
Algorithm: summarize_database(component C)
Begin
For all partitions Pi
Begin loop
```

For all transactions T_i
If T_i contains any item i which is in C
Add pattern num to Ti
End loop
Forall items i in C
Generate_rules(C, S_i)

For generation of valid rules we can use any efficient frequent itemset mining algorithm based on Apriori.[1].

Lemma 2: Maximum number of pattern_num associated with each transaction can't be greater than the total number of transactions in a partition.

Proof
For n transactions the total number of localized pattern is always less than or equal to the number of transactions.
i.e. In the worst case no two transaction will ever be present in the same localized pattern and all transaction will form its own localized pattern.

Pattern_num <= total number of transactions , and
In the worst case:
Pattern_num = total number of transactions .

REFERENCES

1. R. Aggarwal, T. Imielinski and A. Swami, Mining association rules between sets of items in very large databases, Proceedings of the ACM SIGMOD Conference(1993), pp.207-216
2. J. Hipp, U. Guntzer and G. Nakhaeizadeh, Algorithms for association rules mining—a general survey and comparison, SIGKDD Explor. **2** (2000) (1), pp. 58–64.
3. Alexandrous Nanopoulus, Apostolos N. Padopoulos and Yannis Manolopoulous, Mining Association Rules in very large clustered domains, Science direct 16 April 2006
4. A. Savasere, E. Omiecinski and S. Navathe, An efficient algorithm for mining association rules in large databases, Proceedings of the VLDB Conference
5. R.J. Hilderman, H.J. Hamilton Knowledge discovery and interestingness measure : a survey, Tech. Report CS-99-04, Dept. of Computer Science, Univ. of Regina, 1999.
6. R.M. Karp, S. Shenker and C.H. Papadimitrious, A simple algorithm for finding frequent elements in streams and bags, ACM Trans. Database Syst. 28 (2003) (1), pp. 51–55.
7. D. Lin and Z.M. Kedem, Pincer-search: an efficient algorithm for discovering the maximum frequent set, IEEE Trans. Knowledge Data Eng. 14 (2002) (3), pp. 553–556.
8. M. J. Zaki C. Hsiao, Charm: an efficient algorithm for closed itemset mining, in: Proc. SIAM International Conference on Data Mining, 2002.
9. S. Ramaswamy, S. Mahajan, A. Silberschatz, On the discovery of interesting patterns in association rules, in: Proc. 24th Int. Conf. Very Large Data Bases (VLDB 1998), 1998, pp. 368–379.
10. B. Goethals, M. Zaki, Advances in frequent itemset mining implementations: introduction to FIMI03, in: Proceedings of the FIMI Workshop, 2003
11. R. Agrawal, H. Mannila, R. Srikant, H. Toivonen and A.I. Verkamo, Fast discovery of association rules, Advances in Knowledge Discovery and Data Mining (1996), pp. 307–328.
12. J.S. Park, M.-S. Chen and P. Yu, Using a hash-based method with transaction trimming for mining association rules, IEEE Trans. Knowl. Data Eng. **9** (1997) (5), pp. 813–825.
13. R. Agarwal, C. Aggarwal and V.V. Prasad, A tree projection algorithm for generation of frequent itemsets, J. Parallel Distrib. Comput. **61** (2001), pp. 350–371.
14. M.J. Zaki, Scalable algorithms for association mining, IEEE Trans. Knowl. Data Eng. **12** (2000) (3), pp. 372–390.
15. F. Pan, G. Cong and A.K.H. Tung, Carpenter: finding closed patterns in long biological datasets, Proceedings of the SIGKDD Conference (2003).
16. X.-R. Jiang and L. Gruenwald, Microarray gene expression data association rules mining based on BSC-tree and FIS-tree, Data Knowl. Eng. **53** (2005) (1), pp. 3–29.
17. C. Aggarwal, C. Procopiuc and P. Yu, Finding localized associations in market basket data, IEEE Trans. Knowl. Data Eng. **14** (2002) (1)

