Prototype and Modeling of Association Rule Algorithm Based on Bitmap and Granular Computing

Yogendra Yati[1] and R.C. Jain[2]

ABSTRACT

We present an association rule algorithm based on granular computing (Bit-Compu-AssocRule) that doesn't follow the generation-and-test strategy of Apriori algorithm It adopts the divide and conquers strategy, thus avoids the time consuming table scan to find and prune the item sets. It is the fast bit operations based on its corresponding granular for all the operations of finding large item sets from the datasets. We also develop a prototype and model of association rules based on bitmap and granular computing. Our research indicates that bitmap and granular computing can greatly improve the performance of association rule algorithm.

Keywords : Data Mining, Association rules, Bitmap, granule.

1. INTRODUCTION

An association rule algorithms have been developed in the last decades[2][3][4][5] [6] ,Which can be classified into two categories: (a) candidate generation/test approach such as Apriori [5] (b) pattern growth approach [2]. The challenging issues of association rule algorithms are multiple scan of transaction databases and huge number of candidates. We present a novel association rule algorithm Bit -Compu- AssocRule based on bit map and granular computing approach. Traditional Apriori algorithms require full table scan and multiple passes of the item-sets in order to finding association rules from large database. Bit- Compu-AssocRule avoids these time-consuming operations and relies on the fast bit operations of its granular to find the large Bit bit item sets. Bitmap represents each distinct value as array of bits where a 1 or 0 in each relative position in the array represents True or False for that value for the corresponding relative record within the database relational table. This approach is sometimes referred as inverted-list. But the real benefit of the bitmap index is the processing speed. With bitmap techniques, we can greatly improve the performance of the association rule algorithms.

Recently, many attempts have been given to applying bitmap techniques in the association rule algorithms in [1].The use of bitmaps improves the performance to find association rules. The bit representation of bitmaps offers efficient storage while the intersection of bitmaps offers fast computation in finding association rules. The AND,SHIFT, and COUNT operations among bitmaps are extremely fast. Unlike the traditional Apriori algorithm which generates k-candidate by combining two (k-l) large item sets. Our Bit-Compu-AssocRule algorithm generates k- candidate by interesting the bitmap of l attribute value with bitmaps of other (k-l) attribute values.

Here, we describe the algorithm in details.

Bit-Compu–apriori takes the same or litter longer time than the various Apriori algorithms in constructing the 1-itemsets because of the extra cost of building the bitmaps for the 1-itemsets. But after the 1-itsemtset is done, Bit-

[1]Department of Computer Science & Application, S.V. College, Bairagarh, BHOPAL(M.P.)

[2] Director, Samrat Ashok Technological Institute, VIDISHA(M.P.) E-mail: yogendrayati@rediffmail.com

AssocRule is significant faster than the Apriori algorithms in constructing large frequent item sets because it only uses the fast bit is significant faster than the Apriori algorithms in constructing large frequent item sets.

Granule mining is a new initiative that attempts to improve the quality of discovered knowledge in multidimensional databases. The basic idea of granule mining came from decision tables presented by [9], where the attributes are divided by users into two groups: condition attributes and decision attributes.

The remainder of the paper is organized as follows:

we discuss bitmap techniques and granular computing in Section 2. We present the granular-based association rule algorithm .In section 3, we focuses on Bit-Compu–apriori and the comparison results of Bit-Compu–apriori with various Apriori algorithms in Section 4. In Section 5 ,we develop prototype and model of association rules based on bitmap and granular computing. In Section 6, we conclude the paper with some discussions.

2. Bitmap Techniques and Granular Computing

The bitmap technique was proposed in the 1960 [7] and has been used by a variety of products since then. Bitmap technique delivers far superior query performance on unselective data than traditional B-Tree indexing techniques. Bitmap represents each distinct value as array of bits where 1 or 0 in each relative position in the array represents True or False for that value for the corresponding relative record within the database relational table. This approach is sometimes referred as an inverted-list. But the real benefit of the bitmap index is the processing speed. Combining the process of performing a logical operation (AND, OR or NOT) on a serious of bitmaps is very efficient, particularly compared with performing similar processes on lists of tuple -Ids.

Granular computing was first proposed by [1] and has become a very important tool in data mining since then. A granule is a clump of objects drown together by similarity, proximity or functionality. The equivalent relations are the granules of the relation. Each unique attribute value in the relational table is a granule, and each granule is a list of tuples that have the same attribute value. Only the tuple name, or the reference to the tuple, is stored in the granules.

The granular concept and bitmap index are very closely related to each other. Other than a list, the granular can be represented in bitmap. Each tuple in the relation has one unique offset position in the bitmap. The bits are set to 1 for those tuples having the attribute value of the granule, and the bit are set to 0 for those tuples not having the attribute values of the granules. This is the bitmap representation of the granules of the lists.

3. Apriori Algorithm

The most influential algorithm Apriori developed by Rakesh et al. [6][5][10] generates the k-candidate by combining two (k-1)-itemset that have the first k-2 attribute values in the two (k-1)-itemset the same the last pair does not. A new K-candidate becomes a k large item set if every (k-1)-subset of the k-candidate is a large item set otherwise it is removed. This algorithm need to do table scan of the whole data set and examine the item sets multiple times, the process is very time consuming.

The algorithm Apriori is as following :

L1={large 1-itemset};

For (k=2; L_{k-1} "" φ;k + +)

{

Ck=apriori (L_k);

For all transactions t ∈ D

{

Ct=subset(Ck,t);

For all candidates $c \in C_t$

c.count++;

$L_k = \{ c \ C_k \mid c.count >= minsup\}$

Return ($_kL_k$)

}

The aprior-gen function takes as an argument L_{k-1}, the set of all large (k-1)- itemsets. It returns a superset of the set of all large k-itemsets.

4 Bit-Compu–apriori Algorithm

4.1 The Generation of Combinations

Creating combinations, or patterns, is a computational process of forming sets of attribute values. The number of combinations for one selected attribute is the number of distinct values of the selected attribute. Each candidate contains one attribute value from that selected attributes is the Cartesian product, $p_1 \times p_2$ where p_1 and p_2 are the number of distinct values the first and second selected attribute values respectively. Each combination contains two attribute values: one attribute value form the first and one attribute value form the second. For the general case, the number of k-candidates is $p_1 \times p_2 \times ... \times p_k$ .Each k candidate contains one attribute value for each selected attribute. The more general case is to create k-candidate among the n attributes in the relation. Since no columns are selected beforehand, k unique attributes are chosen from the n attributes, all combinations of attributes values form the domains of those attributes are formed.

There are *C(n,k)* possible ways to choose k attributes form n attributes, and each possible way has its own set of k-candidates among the k attributes. In a word, the number of k-candidate is the total sum of the each subtotal combination for each possible way to select k attributes.

The general equation for the total number of k- candidate on n attributes in the relation is the following:

$k = \Sigma_{g=1} n \text{ “ } (k \text{ “ } 1)(p_g (\Sigma_{h=g+1} n \text{ “ } (k \text{ “ } 2)$

$p_h (\Sigma_{i=h+1} n - (k- 3) p_i (...\Sigma_{z=y+1} np_z))))$

The equation deals with various numbers of distinct values on the columns in the relation. For the special case, suppose all attributes in the relation have the same number of distinct values, F, the equation simplifies to $C(n,k)F^k$ .Generally, the number of distinct values of each attribute in the relation are not the same. Nonetheless, taking the average number of distinct values of all the attributes may be useful to estimate the number of combinations of length

For a relation with many attributes or attribute values, the number of combinations can be very large. Each combination requires a count of the number of tuples in the relation that the combination contains. The combinations with the count greater than the minimal support are association rules. The next subsection demonstrates methods to reduce the number of combinations. In doing so, the number of comparisons between the combinations and the tuples in the relation are saved.

The number of combinations is an important factor to consider in the process. Each combination has the bitmaps in the combinations intersected and the result counted. If a combination does not have the potential of becoming a large itemset, the combination should not be undergoing this process. So, only potential combinations are generated in the process.

The algorithm starts with a list L1(all the counts of the bitmaps of these 1-itemset are greater than the minimal counter number),which contains all the 1-itemset. When making k-candidates, all the elements in the list L1 are verified if they exist as elements in any (k-1)-itemset. If an element does not exist in any (k-1)- itemset, it is removed from the list. Next, new k candidates are created from the (k-1)-itemset and the list L by joining a (k-1)-

itemset with element in L that has an attribute index greater than all attribute indexes of elements in that (k-1)-itemset. Only the new k candidates that have every (k-l)-subset a large item set are kept. The Algorithm Bit-Compu-apriori association Rule is as following:

L1 = {bitmaps of large 1-itemset}

For (k=2; L_{k-1} "" φ;k++)

{

Prune the L1;

*/*Remove those elements in L1 which are not included in any itemset of L_{k-1}*/*

C_k = Bit -Compu - apriori(L_{k-1} ,L_1);

L_k = { c C_k | bitmap count of c >= minsup }

return($_k L_k$)

}

The algorithm starts with a list L1(also called 1-itemset),which contains attribute values. All the count of the bitmaps of these 1-itemset are greater than the minimal counter number.

The k-candidates consist of k attribute values ($Y_{1,i1}$, $Y_{2,i2}$,..., $Y_{k-1,ik-1}$, Y_{ij}) from k attributes. Using bitmap techniques, the candidate is a large item set if the bit count on the intersection of all the bitmaps A_1)" A_2)"... A_k (suppose A_j is the bitmap of the $Y_{k,ij}$) is equal or greater than the minimal count. The bit count is the number of 1's in the bitmap indexed from the result of the intersection of the bitmaps.

4.2 A Comparison study

The traditional Apriori algorithm, its variation and bitmap-based Bit-Compu- Apriori association rule are compared based on their key operations. For the Bit- Compu-Apriori, the number of AND operations between the bitmaps determines the cost. For the Apriori algorithm, the number of comparisons between the attribute values in the candidates and the tuples determines the cost. A ration is used to compare the two algorithms Apriori and Bit-Compu- Apriori, it is defined as follows:

X= M / N

Where, M is the number of comparison operations in Apriori; and N is the number of AND operations in Bit-Compu- Apriori.

The Apriori and Compu Apriori algorithms generate the candidate itemsets to be counted in a pass by using only the itemsets found large in the previous pass without considering the transaction in the databases. The AprioriTid algorithm has the additional property that the database is not used at all for counting the support of candidate itemsets after the first pass. AprioriHybrid uses Apriori in the initial passes and switches to AprioriTid when it expects that the set of candidate itemsets at the end of the pass will fit in the memory [6][5].

The worst case for the Apriori algorithm is m*(k*q) comparison operations, where m is the number of tuples, all the p candidates have the same length, k. q is the average number of attribute values per candidates. Each candidate compares with each tuple to determine if it is contained in the tuple. The worse case for Bit-compu-apriori association rule algorithm is (m/32)*(k-1)*q AND operations. So, the ratio is defined as follows:

X=m*(k*q)/((m/32)*(k-1)*q)=32k/(k-1)

The Bit-compu-apriori can execute 32 times faster or more than Apriori in theory.

5. Prototype and Model of Association Rules Based on Bitmap and Granule

5.1 Decision model and granules

Let T={t_1,t_2,.....,t_n} be a transaction database and each transaction is a set of items. In the multidimensional

database, Pawlak proposed the decision tables in rough set theory to represent the association rules from the hidden patterns [8][9]. A feature of decision model is related to user constraints, which divide the attributes of a database into condition attributes and decision attributes, respectively. We call the tuple (T, V^T, C, D) a *decision model* of (T, V^T) if C)" $D=\phi$ and $C \cup D \subseteq V^T$, T is a set of transactions, and V^T is the set of attributes (items). The condition attributes C represent the premisė (antecedent) of association rules, while the decision attributes D can be interpreted as the post-condition (consequent) of association rules.

In a decision model[11][12][13], there is a function for every attribute $a \in V^T$ such that a: T '! V_a, where V_a is the set of all values of a. We call V_a the domain of a. C (or D) determines a binary relation $I(C)$ (or $I(D)$) on T such that (t_1, t_2) $I(C)$ if and only if $a(t1) = a(t2)$ for every aC, where $a(t)$ denotes the value of attribute a for object tT. It is easy to prove that $I(C)$ is an equivalence relation, and

the family of all equivalence classes of $I(C)$, that is a partition determined by C, is denoted by T / C.

The classes in T / C (or T /D) are referred to *C-granules* (or *D-granules*). The class which contains t is called *C-granule* induced by t, and is denoted by $C(t)$.

Table 1 simulates a part of the daily transactions for product sales in a shop, which is a Multidimensional database. There are 200 transactions for 7 different products in the database. The possible attributes are department, commodity, cost, price, profit. The users choose only three attributes and let C = {department, commodity} and D = {profit}. We compress the database into a decision model, where each product is viewed as a granule and N_g is the number of transactions that belong to the granule.

Using Table 1 we can classify the condition granules (C-granules) as T / C = {{1,5}, {2,6},{3}, {4,7}} and decision granules (D-granule) as T /D = {{1,4}, {2,5}, {3,6,7}}, respectively.

Table 1. A decision model

Granule	Department	Commodity	Profit(%)	N_g
1	F & V	Accessories	Over 70	47
2	Bakery	General Merch	30-40	12
3	Bakery	Glassware	20-30	48
4	Soft Drinks	Dinners Frozen	Over 70	12
5	F & V	Accessories	30-40	21
6	Bakery	General Merch	20-30	40
7	Soft Drinks	Dinners Frozen	20-30	20

Table 2 : A Time Slice Transaction Model

Time	Product
t_1:02/12/2004	a_1,a_5,a_6,a_7
t_2:02/01/2005	a_1,a_5,a_6,a_7
t_3:02/02/2005	a_1,a_2,a_4,a_5,a_7
t_4:02/03/2005	a_1,a_2,a_3,a_5,a_6
t_5: 02/04/2005	a_1,a_2,a_4,a_5,a_6
t_6: 02/05/2005	a_1, a_2, a_3, a_5, a_6
t_7: 02/06/2005	a_1, a_4, a_5, a_7

Table 3. Granules

Granule	a_1	a_2	a_3	a_4	a_5
cg_1	1	0	0	0	1
cg_2	1	1	0	1	1
cg_3	1	1	1	0	1
cg_4	1	0	0	1	1

(a) C-granules

Granule	a_6	a_7
dg_1	1	1
dg_2	0	1
dg_3	1	0

(b) D-granules

Granule	a_1	a_2	a_3	a_4	a_5	a_6	a_7	N_g
g_1	1	0	0	0	1	1	1	2
g_2	1	1	0	1	1	0	1	1
g_3	1	1	0	1	1	1	0	1
g_4	1	1	1	0	1	1	0	0
g_5	1	0	0	1	1	0	1	2

(c) Decision Model

We also can view the transactions in Table 2, where the transactions come from the different time slices during 7 months and all products are frequent. Let $V^T = \{a_1, a_2, \ldots, a_7\}$, $T = \{t_1, t_2, \ldots, t_7\}$. According to the profit, *bananas Cavendish* (a_1), *coca cola 2LT* (a_2), *1.25 LT* (a_3), *chicken pieces* (a_4) and *potatoes brushed* (a_5) are all high profit products; *bread white* (*a*6) and *sandwich* (a_7) are both low profit products. We set up the user constraint with the profit and classify the products into two groups. Let a_1, a_2, a_3, a_4, a_5 be condition attributes and a_6, a_7 decision attributes. Table 3 (a) is the *C-granules*; Table 3 (b) is the *D-granules*; Table 3 (c) is

the decision model, where $T/C \cup D = \{g_1, g_2, g_3, g_4, g_5\}$ and N_g is the number of objects in the same granule.

In the representation of association rule mining[14][15], we view a decision model as a multidimensional database. Every granule in the decision model can be mapped into a decision rule [8]. The condition attribute can be viewed as the premise of association rules; the decision attributes can be viewed as the post-conditions. The presence or absence of items is viewed as the same position. Therefore, we can obtain 5 decision rules in Table 3 (c), and the first one can be read as the following decision rule:

$(a_1 = 1) \wedge (a_2 = 0) \wedge (a_3 = 0) \wedge (a_4 = 0) \wedge (a_5 = 1)$ '! $(a_6 = 1) \wedge (a_7 = 1)$

or in short $C(g_1)$'! $D(g_1)$ (or $C(t_1)$'! $D(t_1)$), where ^ means "*and*",

And remaining for decision Rules find out similar as above.

From the above examples, we can now interpret association rules based on granules rather than patterns. In particular, we can view the association rules based on different granularities of multidimensional databases according to what users want.

Definition A frequent pattern X is *closed* if and only if $X = Closure(X)$.

Given a *C-granule* $cg = C(t)$, its *covering set* $[cg] = \{t' \mid t' \in T, (t', t)\ I(C)\}$. Let cg be a *C-granule* and dg be a *D-granule*, we define $[cg \wedge dg] = [cg]$)" $[dg]$.For example, in Table 3 $g_1 = \{(a_1 = 1) \wedge (a_2 = 0) \wedge (a_3 = 0) \wedge (a_4 = 0) \wedge (a_5 = 1) \wedge (a_6 = 1) \wedge (a_7 = 1)\} = C(g_1) \wedge D(g_1) = cg_1 \wedge dg_1$, therefore

$[g_1] = [cg_1 \wedge dg_1] = [cg_1]$)" $[dg_1] = \{t_1, t_2, t_3, t_4, t_5, t_6, t_7\}$)" $\{t_1, t_2\} = \{t_1, t_2\}$.

$[g_2] = [cg_2 \wedge dg_2] = [cg_2]$)" $[dg_2] = \{t_3, t_5\}$)" $\{t_3, t_7\} = \{t_3\}$.

$[g_3] = [cg_2 \wedge dg_3] = [cg_2]$)" $[dg_3] = \{t_3, t_5\}$)" $\{t_4, t_5, t_6\} = \{t_5\}$.

$[g_4] = [cg_3 \wedge dg_3] = [cg_3]$)" $[dg_3] = \{t_4, t_6\}$)" $\{t_4, t_5, t_6\} = \{t_4, t_6\}$.

$[g_5] = [cg_4 \wedge dg_2] = [cg_4]$)" $[dg_2] = \{t_7\}$)" $\{t_3, t_7\} = \{t_7\}$.

Table 4 illustrates the covering sets of granules, where (a) includes the covering sets of *Cgranules*,(b) includes the covering sets of *D-granules*, and (c) includes the covering sets of *CD-granules*.

Theorem Let (T, V^T, C, D) be a *decision model*. We have

$[C(t)]$ $[CD(t)]$, for all tT. (1)

The derived decision pattern of every granule $g\ T/CD$ is a closed pattern. (2)

Table 4. Covering set of C ∪ D - granules

Granule	a_1	a_2	a_3	a_4	a_5	Covering set
cg_1	1	0	0	0	1	$\{t_1,t_2,t_3,t_4,t_5,t_6,t_7\}$
cg_2	1	1	0	1	1	$\{t_3,t_5\}$
cg_3	1	1	1	0	1	$\{t_4,t_6\}$
cg_4	1	0	0	1	1	$\{t_7\}$

(a) C-granules

Granule	a_6	a_7	Covering set
dg_1	1	1	$\{t_1,t_2\}$
dg_2	0	1	$\{t_3,t_7\}$
dg_3	1	0	$\{t_4,t_5,t_6\}$

(b) D-granules

Granule	a_1	a_2	a_3	a_4	a_5	a_6	a_7	N_g	Covering set
g_1	1	0	0	0	1	1	1	2	(t_1,t_2)
g_2	1	1	0	1	1	0	1	1	(t_3)
g_3	1	1	0	1	1	1	0	1	(t_5)
g_4	1	1	1	0	1	1	0	0	(t_4,t_6)
g_5	1	0	0	1	1	0	1	2	(t_7)

(c) Decision model

6. Conclusions

We present a Bit-compu-apriori based association rule algorithm using granular computing technique and introduce the bitmap technique to the data mining procedure and develop bitmap based algorithm to find associations. Bit-compu-apriori algorithm avoids the time-consuming table scan to find and prune the itemsets, all the operations of finding large itemsets from the datasets are the fast bit operations. This research indicates that bitmap and granular computing techniques can greatly enhance the performance for finding association rule, and bitmap techniques are very promising for the decision support query

optimization and data mining application. Bitmap technique is only one way to improve the performance data mining algorithm. Based on their published comparison results with Apriori, our Bit-AssocRule. we present granule based decision models to reduce the complexity and evaluate the effectiveness of association rule mining.

References

[1] Lin T.Y., " Data Mining and Machine Oriented Modeling:A Granular Computing Approach ", Journal of Applied Intelligence,Oct.2000.

[2] Guizhen Yang: The complexity of mining maximal frequent itemsets and maximal frequent patterns, Proceedings of the 2004 ACM SIGKDDinternational conference on Knowledge discovery \and data mining , pages:343 353,August2004, Seattle, WA, USA.

[3] J. Hipp, U. Güntzer, and U. Grimmer. Integrating association rule mining algorithms with relational database systems. In Proceedings of the 3rd International Conference on Enterprise Information Systems (ICEIS 2001), pages 130-137, Setúbal, Portugal, July 7-10 2001.

[4] R. Bayardo and R. Agrawal. Mining the most interesting rules. In Proceedings of the 5th International Conference on Knowledge Discovery and Data Mining (KDD '99), pages 145- 154, San Diego, California, USA, August 1999.

[5] R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, and A. I. Verkamo. Fast discovery of association rules. In Advances in Knowledge Discovery and Data Mining, pages 307—328. AAAI Press, 1996.

[6] R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. In Proceedings of the ACM SIGMOD International Conference on Management of Data (ACM SIGMOD '93), pages 207-216, Washington, USA, May 1993.

[7] Bertino E., Ooi B.C., Sacks-Davis R.etc,"Indexing Techniques for Advanced Database Systems" , Kliwer Publisher.1960.

[8] Pawlak, Z. (2002). In pursuit of patterns in data reasoning from data, the rough set way, Proceedings of 3rd International Conference on Rough Sets and Current Trends in Computing, pp. 1-12, USA,2002

[9] Pawlak, Z. (2003). Flow graphs and decision algorithms, Proceedings of 9th International Conference on Rough Set, Fuzzy Sets, Data Mining and Granular Computing, pp. 1-10, Chongqing,China, 2003

[10] Agraw, R., Srikant, R. (1994). Fast Algorithms for Mining Association Rules in Large Databases, *Proceedings of the 20th International Conference on Very Large Data Bases*, pp. 487-499, 1994.

[11] Feng, L., Yu, J. X., Lu, H.&Han, J. (2002). A template model for multidimensional intertransactional association rules, *The International Journal on Very Large Data Bases*, 11(2), (2002) pp. 153 -175

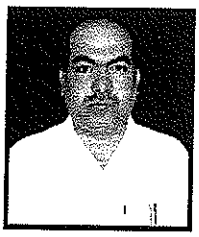
[12] Lee, A. J. T., Lin, W. & Wang, C. (2006). Mining association rules with multi- imensional constraints, *The Journal of Systems and Software*, pp. 79-92, 2006

[13] Li, Y., Yang, W. & Xu, Y. (2006). Multi-Tier Granule Mining for Representationsof Multidimensional Association Rules, *Proceedings of 6th IEEE International Conference on Data Mining*, pp. 953-958, Hong Kong, 2006.

[14] Yang, W., Li, Y., Wu, J. & Xu, Y., Granule Mining Oriented Data Warehousing Model for Representations of Multidimensional Association Rules, *International Journal of Intelligent Information and Database Systems*, Vol.2, No.1, (2008), pp. 125-145 2008.

[15] Tung, A.K.H., Lu, H., Han, J. & Feng, L. (2003). Efficient mining of intertransaction association rules, *IEEE Transactions on Knowledge and Data Engineering*, Vol.15, No.1, (2003), pp.43–56

Author's Biography



Yogendra yati is working as an assistant professor in department of computer science & application in sadhu vaswani college Bairagarh, Bhopal. He has got more than 12 years teaching experience in computer science at UG and PG level. He completed his M.Sc. Degree in computer scienc with first division from Barkatullah University, Bhopal (M.P.). He is perusing Ph.D in computer science from the same university.