GCARM: A Combined Approach to Data Mining

Seema Desai[1], Lata Ragha[2] and Vimla Jethani[3]
[1]*Department of Information Technology, SIES Graduate School of Technology, Nerul, Navi Mumbai, India*
[2&3]*Department of Computer Science and Engineering, Ramrao Adik Institute of Technology, Nerul Navi Mumbai, India*
[1]desaiseema83@gmail.com, [2]lata.ragha@gmail.com

***Abstract* - Mining association rules is an essential task for knowledge discovery. From a large amount of data, potentially useful information may be discovered. Association rules are used to discover the relationships of items or attributes among huge data. These rules can be effective in uncovering unknown relationships, providing results that can be the basis of forecast and decision. The effective management of business is significantly dependent on the quality of its decision making. Past transaction data can be analyzed to discover customer behaviors such that the quality of business decision can be improved. The approach of mining association rules focuses on discovering large itemsets, which are groups of items that appear together in an adequate number of transactions. The proposed method focuses on a combined approach to generate association rules from a large database of customer transactions. This approach scans the database once to construct an association graph and clustering tables and then traverses the graph to generate all large itemsets. The proposed algorithm will outperforms other algorithms which need to make multiple passes over the database.**

***Keywords* - Association graph, association rule, clustering tables**

I. Introduction

Data mining has high applicability in the retail industry. The effective management of business is extensively dependent on the quality of its decision making. Therefore, it is important to improve the quality of business decisions by analyzing past transaction data to discover customer purchasing behaviours. In order to support this analysis, a sufficient amount of transactions needs to be collected and stored in a database. Each transaction in the database consists of the items purchased in the transaction besides other information like transaction date and time, customer name, quantity, price, and other information. All what was taken in consideration is the set of items bought together in a transaction. Because the amount of these transactions' data can be very large, an efficient algorithm needs to be designed for discovering useful information from these huge transactional datasets. Mining frequent itemset and association rules is a popular and well researched method for discovering interesting relations between variables in large databases. Association rules, first introduced in 1993, are used to identify relationships among a set of items in a database. These relationships are not based on inherent properties of the data themselves, but rather based on co- occurrence of the data items. Association rules are used to discover the relationships, and potential associations, of items or attributes among huge data. These rules can be effective in uncovering unknown relationships, providing results that can be the basis of forecast and decision. They have proven to be very useful tools for an enterprise as it strives to improve its competitiveness and prosperity. Association rule mining (ARM) in relational database management systems generally transforms the database into (TID, item) format, where TID stands for a unique transaction identifier and item stands for different items purchased by the customers. There will be multiple entries for a given transaction ID, because one transaction ID indicates purchase of one particular customer and a customer can purchase as many items as he/ she want. Any association rule will hold if its support and confidence are equal to or greater than the user specified minimum support (S) and confidence (C). The final step involves generating strong rules having a minimum confidence from the frequent itemsets. It also includes generating and testing the confidence of all rules.

II. Background

The literature review is done to get an insight of the association rule mining algorithms in data mining. It is necessary to identify various methodologies that could possibly used to identify the relationships among itemsets of various transactions in database. The objective of this portion of literature review is to identify the existing association rule mining techniques..

A. Basic Concepts & Basic Association Rules Algorithms

Let I=I1, I2, … , Im be a set of m distinct attributes, T be transaction that contains a set of items such that $T \subseteq I$, D be a database with different transaction records Ts. An association rule is an implication in the form of $X \subseteq Y$, where $X, Y \subset I$ are sets of items called itemsets, and $X \cap Y = \varnothing$. X is called antecedent while Y is called consequent, the rule means X implies Y. There are two important basic measures for association rules, support(s) and confidence(c). Since the database is large and users concern about only those frequently purchased items, usually thresholds of support and confidence are predefined by users to drop those rules that are not so interesting or useful. The two thresholds are called minimal support and minimal confidence respectively. Support(s) of an association rule is defined as the percentage/fraction of records that contain $X \cup Y$ to the total number of records in the database. Suppose the support of an item is 0.1%, it means only 0.1 percent of the transaction contain purchasing of this item. Confidence of an association rule is defined as the percentage/fraction of the number of transactions that contain $X \cup Y$ to the total number of records that contain X.

Confidence is a measure of strength of the association rules, suppose the confidence of the association rule X⇒Y is 80%, it means that 80% of the transactions that contain X also contain Y together. In general, a set of items (such as the antecedent or the consequent of a rule) is called an itemset. The number of items in an itemset is called the length of an itemset. Itemsets of some length k are referred to as k-itemsets.

Generally, an association rules mining algorithm contains the following steps:

- The set of candidate k-itemsets is generated by 1-extensions of the large (k -1)-itemsets generated in the previous iteration.
- Supports for the candidate k-itemsets are generated by a pass over the database.
- Itemsets that do not have the minimum support are discarded and the remaining itemsets are called large k-itemsets.

This process is repeated until no more large itemsets are found.

1) Apriori Algorithm

Apriori is a classic algorithm of association rule mining. Apriori is designed to operate on large database containing transactions (for example, collections of items bought by customers, or details of a website frequentation). As is common in association rule mining, given a set of itemsets (for instance, sets of retail transactions, each listing individual items purchased), the algorithm attempts to find subsets which are common to at least a minimum number C of the itemsets. Apriori uses a "bottom up" approach, where frequent subsets are extended one item at a time (a step known as candidate generation), and groups of candidates are tested against the data. The algorithm terminates when no further successful extensions are found.

Apriori is more efficient during the candidate generation process [3]. Apriori uses pruning techniques to avoid measuring certain itemsets, while guaranteeing completeness. These are the itemsets that the algorithm can prove will not turn out to be large. However there are two bottlenecks of the Apriori algorithm. One is the complex candidate generation process that uses most of the time, space and memory. Another bottleneck is the multiple scan of the database. Based on Apriori algorithm, many new algorithms were designed with some modifications or improvements.

2) FP-Tree

Han *et. al.*[4] proposed a novel frequent pattern tree (FP-tree) structure, which is an extended prefix-tree structure for storing compressed, crucial information about frequent patterns, and developed an efficient FP-tree based mining method. It is another milestone in the development of association rule mining, which breaks the main bottlenecks of the Apriori. The frequent itemsets are generated with only two passes over the database and without any candidate generation process. FP-tree is an extended prefix-tree structure storing crucial, quantitative information about frequent patterns. Only frequent length-1 items will have nodes in the tree, and the tree nodes are arranged in such a way that more frequently occurring nodes will have better chances of sharing nodes than less frequently occurring nodes will have better chances of sharing nodes than less frequently occurring ones. FP-Tree scales much better than Apriori because as the support threshold goes down, the number as well as the length of frequent itemsets increase dramatically. The candidate sets that Apriori must handle become extremely large, and the pattern matching with a lot of candidates by searching through the transactions becomes very expensive. The frequent patterns generation process includes two sub processes: constructing the FT-Tree, and generating frequent patterns from the FP-Tree. The mining result is the same with Apriori series algorithms. To sum up, the efficiency of FP-Tree algorithm account for three reasons. First the FP-Tree is a compressed representation of the original database because only those frequent items are used to construct the tree, other irrelevant information are pruned. Secondly this algorithm only scans the database twice. Thirdly, FP-Tree uses a divide and conquer method that considerably reduced the size of the subsequent conditional FP-Tree.

Every algorithm has his limitations, for FP-Tree it is difficult to be used in an interactive mining system. During the interactive mining process, users may change he threshold of support according to the rules. However for FP-Tree the changing of support may lead to repetition of the whole mining process. Another limitation is that FP-Tree is that it is not suitable for incremental mining. Since as time goes on databases keep changing, new datasets may be inserted into the database, those insertions may also lead to a repetition of the whole process if we employ FP-Tree algorithm.

3) Cluster Based Association Rule Mining

Tsay and Chiang proposed an efficient cluster based association rule mining method (CBAR)[7] for discovering the large itemsets. The CBAR method create cluster tables by scanning the database once, and then clustering the transaction records to the k-th cluster table, where the length of a record is k. Moreover, the large itemsets are generated by contrasts with the partial cluster tables. This method not only prunes considerable amounts of data reducing the time needed to perform data scans and requiring less contrast, but also ensures the correctness of the mined results.

4) Cluster Based Association Rule (CBAR)

The performance is dramatically decreased in the process of many association rule algorithms. This is due to the fact that a database is repeatedly scanned to contrast each candidate itemset with the whole database level by level in the process

of mining association rules. If an alternative method can decrease the number of database scans, and reduce the number of contrasts, efficiency will be improved. Thus we propose an efficient CBAR method for discovering the large itemsets, and the main characteristics are the following. CBAR only requires a single scan of the transaction database, followed by contrasts with the partial cluster tables. This not only prunes considerable amounts of data reducing the time needed to perform data scans and requiring less contrast, but also ensures the correctness of the mined results.

III. Proposed Method

This chapter sets out the methodology adopted for the project study. Through the study of literature survey has identified different association rule mining techniques still there is a need to find the time efficient method to mine strong association rules from large database. The proposed method to find relationship among itemsets scans the database once to construct an association graph and then traverses the graph to generate all large itemsets. In the clustering step, each transaction is clustered to the k-th cluster, where the length of a transaction is *k*. Meanwhile, a clustering table is built to count the occurrences of each item in each cluster, subsequently the clustering- table will be used in the candidate generation.

The proposed algorithm (GCARM) has two main advantages: one is the reduction of the database scans and the other is the elimination of candidate k-itemsets of order 3 and above. Figure 3.1 presents an overview of GCARM algorithm steps where t_j is j^{th} transaction and M is total no of transactions.

A. Algorithm

The algorithm scans the database once to build a graph of items and a clustering-table. This scan is enough to find frequent 1-itemsets and frequent 2-itemsets. There is no need to generate candidate 2-itemsets and hence no need to scan the database to discover frequent 2-itemsets. After that, GCARM works iteratively starting from frequent 2-itemsets (F2) in the sense that frequent itemsets that are discovered in one iteration will be used as the basis to generate candidate itemsets in the next iteration. As will be described later, the candidate generation step is similar to that in the Apriori algorithm but here we employ clustering technique to eliminate some infrequent candidate itemsets.

The Build-Graph algorithm build a complete undirected graph G = (V, E) using all transaction in the database. Initially, the graph G is the subgraph of the first transaction. For each transaction t in the database, the algorithm build a complete undirected subgraph Gt = (Vt, Et), where Vt is the set of all items in t and Et is the set of all edges between every 2- subset itemsets in t. A counter is associated with each vertex or edge that stores the occurrences of that vertex or edge and is initialized to 1. After building the subgraph Gt, a new version of graph G is created by merging G and Gt. If there are any similar vertices and edges between Gt and G, their counters are summed up.

```
GCARM(int minsup)
G ← ∅
C1 = {set of all items}
for all transactions t ∈ D do
  Build_Graph(G, t);
Create_Cluster(t, length(t));
  Parse_Graph(G);
for (k = 3; Ck ≠ ∅; k++) do
 For every itemset P ∈ Ck do
   Wp = Calculate_weight(P,k)
    If Wp > minsup then
     Add P to Fk
   If Fk ≠ ∅ then
     Ck+1 ← Fk U Fk
// Union Fk with Fk to generate ck+1,     Ck+2,….
Build-Graph(graph G, transaction t)
for each item i ∈ t do
V[G] = V[G] U {i}
  Count[i] = 1;
for each 2-subset itemset e ∈ t       do
 E[G]← E[G] ∪ {e}
  Count[e] = 1;
if (there are similar ertices and edges)
   Merge(vertices and edges);
```

In the clustering step, each transaction is clustered to the *k*-th cluster, where the length of a transaction is *k*. Cluster the transaction records by length and store each transaction record into the cluster table.

```
Create_Cluster(transaction t, int n)
// Add entry in the cluster table(n)
  for each item i ∈ t do
   Cluster-table[i][n]=1;
```

The function *Parse_Graph* searches the graph to find frequent 1- itemsets and Candidate 2-itemsets. The function *Parse_Graph* traverses each vertex and edge in the graph, if the counter of a vertex is greater than or equal the minimum support then the corresponding item is inserted into the set of frequent 1-itemsets (F_1) and all the edges in the graph represents association between 2 items which gives candidate itemset (C2)

```
Parse_Graph(graph G)
 for each vertex v ∈ V[G]              do
  If (count[v] ≥ minsup) then
   F1 ← F1 ∪ {v};
  for each edge e ∈ E[G]do
   C2 ← C2 ∪ {e};
```

Now algorithm parses each candidate itmesets and calcaulates weight (W_p) of each itemset P. Association between two items considering W_p can also be represented in a matrix format as shown in figure 3.5. If this weight W_p is more than minimum

support then we add this itemset (P) to frequent itemset (Fk). Then we perform Union operation on Fk with Fk to generate candidate itemset Ck+1 and so on. The algorithm repeated until Ck results empty.

```
Calculate_weight(P,k)
Count(P)= Get count of occurrence of
 P from Cluster_table
// For k=2, this is count of edge from graph
Fot each subset item S P do
  Psum = 0
// Get count of individual items and add it
Psum = Count[occurance(S1)
 OR occurance(S2)… OR occurance(P)
  Wp = Count(P) / Psum
  Return Wp
```

Once the frequent itemsets have been found, generating interesting association rules is a straightforward step. Interesting association rules are the ones that satisfy the minimum support and the minimum confidence. Figure shows the algorithm for association rules generation

```
Generate-Rule(float minconf)
for all frequent itemset f do
for all nonempty subset s of f do
If (support(s)/support(f) > minconf) then
 Add "s → (f-s)" to the set of rules
```

B. An Example

We provide an example to further explain the application of proposed algorithm. There are 20 transactions in the database. An example transaction database is shown in Table 1.

Part 1 of the algorithm scans the database once to build a graph of items and a clustering-table. This scan is enough to find frequent 1-itemsets and Candidate 2-itemsets

TABLE I TRANSACTION DATABASE

TID	ITEMS	TID	ITEMS
T1	A,B,C	T11	A,B,C
T2	B	T12	C,E
T3	A,E	T13	B,C,D,E
T4	A,C,D,E	T14	C,D
T5	A,C	T15	B,C,D
T6	A,C,E	T16	A,D,E
T7	C	T17	B,C
T8	B,C,E	T18	E
T9	A,B,C,D	T19	A,C,D
T10	D	T20	A,B,C,D

Figure 3.2 shows sub graphs for three transactions. After all transactions have been read, the graph is built. Vertices' counters hold the supports of the corresponding items. Edge's counters on the other hand hold the supports of the corresponding 2-subset itemsets.

For every transaction the subgraph will be drawn as shown,

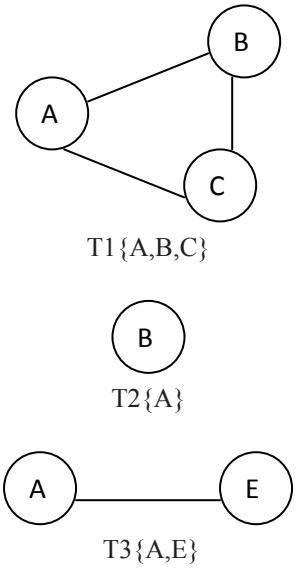

Fig. 3.2 Subgraph

Similarly for all transactions subgraphs will be drawn and all subgraphs will be merged to get the final graph G as shown in Figure 3.3.

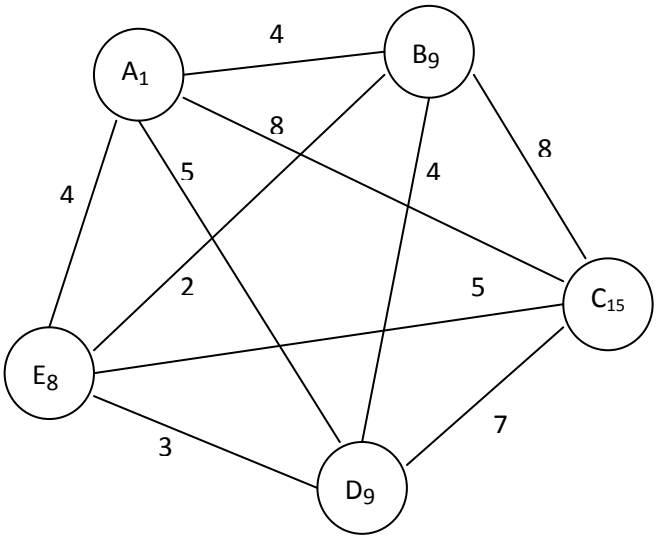

Fig. 3.3 Final graph G

For above example based on the built graph, cluster table can be formed as shown in table 2. Cluster the transaction records by length and store each transaction record into the cluster table. Using Boolean annotation to denote the appearance and disappearance of each item.

There are four cluster tables, named Cluster_Table(k),where $1 \leq k \geq 4$. For above example the cluster table is formed as shown in table 2

Table: II Cluster Table

Itm	Cluster-1				Cluster-2				
A	0	0	0	0	1	1	0	0	0
B	1	0	0	0	0	0	0	0	1
C	0	1	0	0	0	1	1	1	1
D	0	0	1	0	0	0	0	1	0
E	0	0	0	1	1	0	1	0	0

Itm	Cluster-3					Cluster-4					
A	1	1	0	1	0	1	1	1	1		1
B	1	0	1	1	1	0	0	0	1	1	1
C	1	1	1	1	1	0	1	1	1	1	1
D	0	0	0	0	1	1	1	1	1	1	1
E	0	1	1	0	0	1	0	1	0	1	0

The function Parse_Graph searches the graph to find frequent 1-itemsets and Candidate 2-itemsets. For given example following metrics is presented which gives edge count to find candidate-2 itemset.

Table: III Metrics

Items	A	B	C	D	E
A	0	4	8	5	4
B	4	0	8	4	2
C	8	8	0	7	5
D	5	4	7	0	3
E	4	2	5	3	0

Function Calculate_Weight computes weight of each itemset of Ck in order to find frequent-2 itemset. To calculate weight between two items i and j of individual itemset following method is used in the proposed algorithm.

Wp= Count of Occurrence of items (i, j) / Count of occurrence of (i) OR Count of occurrence of (j) OR Count of occurrence of (i , j)

Now to find frequent-2 itemset the appropriate support is considered and for that the association value indicated in above matrix is re-calculated to get strong frequent-2 itemset.

Table: IV Modified Metric

Items	A	B	C	D	E
A	0	26%	47%	35%	28%
B	26%	0	50%	28%	13%
C	47%	50%	0	41%	27%
D	35%	28%	41%	0	21%
E	28%	13%	27%	21%	0

Candidate-2 itemset for above example will be {AB,AC,AD, AE,BC,BD,BE,CD,CD,DE}. Considering minimum support as 30% and based on above matrix frequent-2 itemset will be resulted as {AC,AD,BC,CD}. After joining above frequent-2 itemset (F2) with (F2) we get candidate-3 itemset as below {ACD,ABC,BCD}.

In order to generate frequent-3 itemset, it is necessary to compute the number of occurrence of each candidate in the cluster table(3). Again the entire cluster table is scanned to find the number of occurrence of every item of every subset of candidate itemset. Consider the min support as 20%, for given example freguent-3 itemset (F3) is {ACD, ABC, BCD}.

Generate the Candidate-4 itemsets (C4) by combining the items of F3 in order to generate candidate 4-itemsets that are labeled C4. The itemset of C4 is {A,B,C,D}. The minimum support (MinSup) is specified as 15%. The candidate itemset {A,B,C,D} occurs only twice in the Cluster_Table(4) therefore its support is 10%, less than the minimum support 15%. So F4=NULL, and so is C5 and the algorithm terminates.

Once the frequent itemsets have been found, generating interesting association rules is a straightforward step. Interesting association rules are the ones that satisfy the minimum support and the minimum confidence. Figure shows the algorithm for association rules generation.

Once the frequent itemsets have been found, generating interesting association rules is a straightforward step. Interesting association rules are the ones that satisfy the minimum support and the minimum confidence. Figure shows the algorithm for association rules generation.

For given example for frequent item subset {ACD} associations rules will be generated by given algorithm are as follows,

R1:A^C→D ACD/AC 4/8 50%
R2:A^D→C ACD/AD 4/5 80%
R3:C^D→A ACD/CD 4/7 57%
R4:A→C^D ACD/A 4/10 40%
R5: C→A^D ACD/C 4/15 26%
R6:D→A^C ACD/D 4/9 44%

From above results it is clear that rule R2 is strongest among all which means that whenever items A and D has been purchased maximum times item C has also been purchased. Similarly for remaining frequent itemset the association can be found.

In this way the strong association rules can be generated efficiently to discover customer behaviors such that the quality of business decision can be improved.

IV. Conclusion

From the literature review, it is clear that there are many association rule mining techniques are available and they are still evolving to get faster and accurate results. Every technique is an improvement over other but the only drawback of existing methods are they take more time to scan the huge database again and again. The only intention behind this study

is to find the efficient technique to mine strong association rule among the itemsets of large database in short span of time. This research contributes to enhancing knowledge discovery techniques used in data mining. This research involves an empirical examination of the data set which will be worked upon. The study makes some contributions to the literature on various rule mining methods and its efficiency and accuracy in the area of data mining.

References

[1] Wael Ahmad AlZoubi,Khairuddin Omar, Azuraliza Abu Bakar, An Efficient Mining of Transactional data using Graph Based Technique. IEEE, 3rd *Conference on Data Mining and Optimization*(DMO), 28-29 June 2011, Selangor, Malaysia

[2] Agrawal, R., Imielinski, T., and Swami, A. N. 1993. Mining association rules between sets of items in large databases. In Proceedings of the 1993 ACM SIGMOD *International Conference on Management of Data*, 207-216.

[3] Agrawal, R. Srikant, Fast algorithm for mining association rules in large databases, *Proceedings of 1994 International Conference on VLDB,* 1994 pp. 487–499.

[4] Jiawei Han , Jian Pei , Yiwen Yin , Runying Mao. Mining Frequent Patternswithout Candidate Generation: A Frequent-Pattern Tree Approach. Data Mining and Knowledge Discovery, 8, 53–87, 2004.

[5] Vijender Singh, Deepak Garg, Survey of Finding Frequent Patterns in Graph Mining: Algorithms and Techniques, *International Journal of Soft Computing and Engineering* (IJSCE) ISSN: 2231-2307, Volume-1, Issue-3, July 2011

[6] Deepayan Chakrabarti And Christos Faloutsos, Graph Mining: Laws, Generators, and Algorithms, *ACM Computing Surveys*, Vol. 38, March 2006, Article 2

[7] Yuh-Jiuan Tsay, Jiunn-Yann Chiang. CBAR: an efficient method for mining association rules. *Knowledge-Based Systems* 18 (2005) 99–105.

[8] Michael Hahsler Bettina Grün Kurt Hornik. Introduction to arules - A computational environment for mining association rules and frequent item sets. *Journal of Statistical Software*. October 2005, Volume 14, Issue