International Journal on Recent and Innovation Trends in Computing and Communication ISSN: 2321-8169
Volume: 4 Issue: 1 339 - 341

Improved Association Rule Mining with Positive and Negative Integration

Prof. Punam Khandar
Assistant Professor,
Dept. of Computer Application
RCOEM, Nagpur
punam.khandar@gmail.com

Prof. Mohini Upasani
Assistant Professor,
Dept. of Computer Application
RCOEM, Nagpur
upasanims@gmail.com

Prof. Pranali Dandekar
Assistant Professor,
Dept. of Computer Application
RCOEM, Nagpur
dandekarpr@rknec.edu

Abstract:-Association rule mining is used to find association relationships among large data sets. Mining frequent patterns is an important aspect in association rule mining. Finding frequent itemsets in databases is crucial in data mining for purpose of extracting association rules. Many algorithms were developed to find those frequent itemsets. Mining frequent patterns in transaction databases, time-series databases, and many other kinds of databases has been studied popularly in data mining research. Most of the previous studies adopt an *Apriori*-like candidate set generation-and-test approach. However, candidate set generation is still costly, especially when there exist a large number of patterns and/or long patterns. The FP-growth algorithm is currently one of the fastest approaches to frequent item set mining.

In this paper, we proposed to compress the frequent pattern set mined from a transaction database to a compact set. The compact set is useful in application where the longest pattern is usually used.

Keywords: *Association rule mining, Apriori, FP-growth*

*****

1. Introduction

Data mining has recently attracted considerable attention from database practitioners and researchers because it has been applied to many fields such as market strategy, financial forecasts and decision support. An association rule is defined as the relation between the itemsets, since its introduction in 1993 [1] the process of finding the association rules has received a great deal of attention. Today the extracting of association rules is still one of the main popular pattern discovery techniques in knowledge discovery and data mining (KDD). Mining using Association rules discover appealing links or relationship among the data items sets from huge amount of data [4].

For this, association uses various techniques like *Apriori* and *FP* rules. The *Apriori* heuristic achieves good performance gained by (possibly significantly) reducing the size of candidate sets. However, in situations with a large number of frequent patterns, long patterns, or quite low minimum support thresholds. Can one develop a method that may avoid candidate generation-and-test and utilize some novel data structures to reduce the cost in frequent-pattern mining? One of the currently fastest and most popular algorithms for frequent item set mining is the FP-growth algorithm [8]. It is based on a prefix tree representation of the given database of transactions (called an FP-tree), which can save considerable amounts of memory for storing the transactions.

The rest of the paper is organized as: section 2 introduces to the classical mining approaches. Section 3 proposes improved association rule mining with positive and negative integration.

2. Two Classical Mining Algorithms

2.1 Apriori Algorithm

Agarwal proposed an algorithm called Apriori to the problem of mining association rules first. Apriori algorithm is a bottom – up , breadth first approach. The frequent item sets are extended one item at a time. It's main idea is to generate k-th candidate itemsets from the (k-1) –th frequent itemsets and to find the k-th frequent item sets from the k-th candidate item sets. The algorithm terminates when frequent item sets cannot be extended anymore. But it has to generate a large amount of candidate item sets and scans the data as many times as the length of the longest frequent item sets.

2.2 FP-tree introduction

Han, et al, proposed the FP-tree data structure that can store the *complete set of frequent patterns* using only *two scans* over the *DB*. The biggest contribution to speed up the frequent pattern mining task is reduction of the number of scans over the DB down to only 2, since the speed of reading data in the secondary storage is slow. FP-tree is a tree structure as defined below:

1. It consists of one root labeled as "null", a set of item prefix sub-trees as the children of the root, and a frequent-item header table.

2. Each node in the item prefix sub-tree consists of 4 fields: *item-name*, (support) *count*, *parent-link*, and *node-link*, where item-name registers which item this node represents, count registers the number of transactions represented by the portion of the path reaching this node, and node-link links to the next node in the FP-tree carrying the same item-name, or null if there is none, the parent-link links to the parent node1.

3. Each entry in the frequent-item header table consists of three fields: (1) *item-name*, (2) the (support) *count2*, and *head of node-link* which points to the first node in the FP tree carrying the *item-name*.

An example of a FP-tree is given in Figure 1, now we will study how to construct the FP-tree in this figure. With the *minsupcount* ξ=*3*, based on the *DB* listed in Table 1. This table shows a simple database of transactions of a supermarket, where the first column is the transaction identification, each row in the second column is the list of items that were bought by a customer.

The FP-tree construction is described briefly as follows:

2.3 FP-Growth Algorithm

Han, Pei et al .proposed a data structure called FP-tree (frequent pattern tree). FP-tree is a highly compact representation of all relevant frequency information in the data set . Every path of FP-tree represents a frequent item set and the nodes in the path are stored in decreasing order of the frequency of the corresponding items. A great advantage of FP-tree is that overlapping itemsets share the same prefix path. So the information of the data set is greatly compressed. It only needs to scan the data set twice and no candidate itemsets are required. An FP-tree has a header table. The nodes in the header table link to the same nodes in its FP-tree. Single items and their counts are stored in the header table by decreasing order of their counts.

Algorithm (FP-tree construction):
Input: A transaction database *DB* and a minimum support threshold ξ .
Output: FP-tree, the frequent-pattern tree of *DB*.
Method: The FP-tree is constructed as follows.

1. Scan the transaction database *DB* once. Collect *F*, the set of frequent items, and the support of each frequent item. Sort *F* in support-descending order as *FList*, the *list* of frequent items.

2. Create the root of an FP-tree, *T* , and label it as "null". For each transaction *Trans* in *DB*do the following.

Select the frequent items in *Trans* and sort them according to the order of *FList*. Let the sorted frequent-item list in *Trans* be [*p* | *P*], where *p* is the first element and *P* is the remaining list. Call *insert tree*([*p* | *P*], *T*).

The function *insert tree*([*p* | *P*], *T*) is performed as follows. If *T* has a child *N* such that *N.item-name* = *p.item-name*, then increment *N*'s count by 1; else create a new node *N*, with its count initialized to 1, its parent link linked to *T* , and its node-link linked to the nodes with the same *item-name* via the node-link structure. If *P* is nonempty, call *insert tree*(*P*, *N*) recursively.

TID	Items Bought	(Ordered) Frequent Items
100	*f, a, c, d, g, i, m, p*	*f, c, a, m, p*
200	*a, b, c, f, l, m, o*	*f, c, a, b, m*
300	*b, f, h, j, o*	*f, b*
400	*b, c, k, s, p*	*c, b, p*
500	*a, f, c, e, l, p, m, n*	*f, c, a, m, p*
600	*f, c, g, s*	*f, c*

Table 1. Transactions in DB and their frequent items

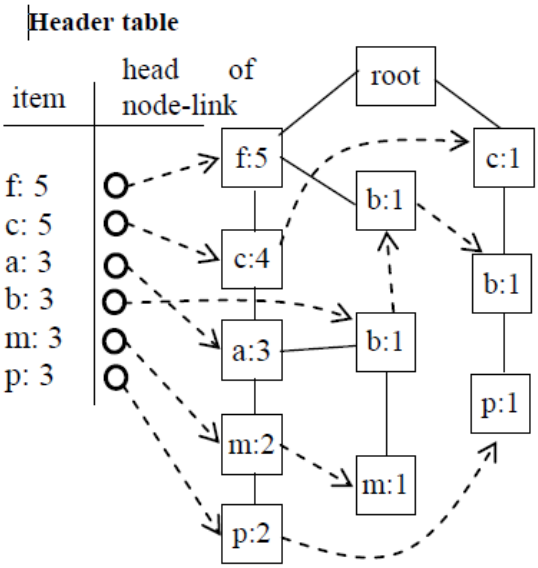

Figure 1. FPTree corresponding to the DB in Table 1

3. Improved Association Rule Mining

Existing work based on *Apriori* algorithm uses candidate sets for finding frequent pattern to generate association rules, then apply class label association rules where this work uses *FP-Tree* with *growth* for finding frequent pattern to generate association rules. *Apriori* algorithm takes more time for large data set where *FP growth* is time efficient to find frequent pattern in transaction.

In this paper, we have proposed a new dimension into the data mining technique. For this we have integrated the concept of positive and negative association rules into the frequent pattern (*FP*) method. Negative and positive rules works better than traditional association rule mining and *FP* cleverly works in large database. Our proposed algorithm has two stages:

a. Rule Generation and,
b. Classification.

In the first stage, the algorithm calculates the whole set of positive and negative class association rules such that sup(R) support and conf(R) confidence given thresholds. Furthermore, the algorithm prunes some contradictory rules and only selects a subset of high quality rules for classification.

In the second stage i.e. classification, for a given data object, the algorithm extracts a subset of rules found in the first stage matching the data object and predicts the class label of the data object by analyzing this subset of rules.

4. Conclusion and Future direction

In this paper, we proposed to compress the frequent pattern set mined from a transaction database to a compact set. The compact set is useful in application where the longest pattern is usually used. In the future direction, we will study the method to construct the compact FP-tree directly form its FP-tree.

References

[1] P. Xu, et al, 2011. *Mining Associated Factors about Emotional Disease Bases on FP-Tree Growing Algorithm*, International Journal of Engineering and Manufacturing, vol. 4, pp. 25-31.

[2] S. Kumar and K. V. Rukmani, 2010. *Implementation of Web Usage Mining Using APRIORI and FP-Growth Algorithms*, Int. J. of Advanced Networking and Applications, Vol 01, Issue: 06, pp 400-404.

[3] Borgelt, 2005. *An Implementation of the FPgrowth Algorithm*. Workshop Open Source Data Mining Software (OSDM'05, Chicago, IL), pp. 1-5, ACM Press.

[4] F. Bonchi and B. Goethals. FP-Bonsai: the Art of Growing and Pruning Small FP-trees. Proc. 8th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD'04, Sydney, Australia), 155–160. Springer-Verlag, Heidelberg, Germany 2004

[5] C. Borgelt. Efficient Implementations of Apriori and Eclat. Proc. 1st IEEE ICDM Workshop on Frequent Item Set Mining Implementations (FIMI 2003, Melbourne, FL). CEUR Workshop Proceedings 90, Aachen, Germany 2003. http://www.ceur-ws.org/Vol-90/

[6] C. Borgelt. Recursion Pruning for the Apriori Algorithm. Proc. 2nd IEEE ICDM Workshop on Frequent Item Set Mining Implementations (FIMI 2003, Brighton, United Kingdom). CEUR Workshop Proceedings 126, Aachen, Germany 2004. http://www.ceur-ws.org/Vol-126/

[7] H. Mannila, H. Toivonen, and I. Verkamo, "Discovery of Frequent Episodes in Event Sequences," Data Mining and Knowledge Discovery, vol. 1, no. 3, pp. 259-289, 1997.

[8] M. Kamber, J. Han, and J. Chiang, "Meta rule-Guided Mining of Multi-Dimensional Association Rules Using Data Cubes," Proc. ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining, pp. 207-210, Aug. 1997.

[9] D. Xin, J. Han, X. Li, and B.W. Wah, "Star-Cubing: Computing Iceberg Cubes by Top-Down and Bottom-Up Integration," Proc. Int'l Conf. Very Large Data Bases, pp. 476-487, Sept. 2003.

[10] D. Burdick, M. Calimlim, and J. Gehrke, "MAFIA: A Maximal Frequent Item set Algorithm for Transactional Databases," Proc. Int'l Conf. Data Eng., pp. 443-452, Apr. 2001.

[11] J. Pei, J. Han, and R. Mao, "CLOSET: An Efficient Algorithm for Mining Frequent Closed Item sets," Proc. ACM SIGMOD Workshop Research Issues in Data Mining and Knowledge Discovery, pp. 21-30, May 2000.