An Extensive Survey on Apriori-like Algorithms

[1]K.J.Paulraj Ananth, [2]T.Arun Nehru

[1, 2] Assistant Professor, PG Department of Computer Applications,
[1, 2] Sacred Heart College (Autonomous), Tirupattur, Vellore Dt, Tamilnadu, India.

Abstract

Data Mining is a process, where intelligent methods are applied to extract data patterns. Association rule mining is a technique in data mining used to find the objective measures based on the support and confidence. Nowadays, there are many efficient algorithms coping with computationally expensive task of association rule mining. Apriori is one of the algorithms that are employed in association rule mining. In this paper, an extensive survey on Apriori-like algorithms is done. The working process, efficiency of each apriori-like algorithm is also clearly explained in this paper.

Keywords–Data Mining, Association Rule Mining, Apriori.

1. Introduction

Data Mining is the process of extracting non-trivial, implicit, previously unknown and potentially useful information from large information repositories such as relational database, data warehouses, etc [1]. In simple, data mining refers to extracting or mining knowledge from large amounts of data. Data mining techniques are employed in various kinds of research fields. Association rule mining is one of the data mining techniques used to find out the association from a large set of data items, usually from large databases.

The idea of association rules were first introduced in (Agrawal, Imielinski, Swami, 1993). Association rule mining has a wide range of applicability such Market basket analysis, Medical diagnosis/ research, Website navigation analysis, Homeland security and so on. Association rules are used to identify relationships among a set of items in database [9]. Subsequent researches have made Association rule mining as one of the most interesting ideas by the introduction of the very-well known algorithm Apriori. The Apriori algorithm has not only influenced the association rule mining community, but it also affected other data mining functionalities.

Since the introduction of Apriori algorithm in 1994 (Agrawal, Srikant), number of improvements has been carried out on the original algorithm in order to raise its efficiency. This paper is organized as follows. In Section 2, the concepts related to association rule mining are introduced. In Section 3, different kinds of Apriori-like algorithms are illustrated.

2. Association Rule Mining

Association rule mining is a mining technique that searches for interesting relationships among items in a given data set [1]. The research on the association rule mining originated from **market basket analysis.** Market managers like to find out customers' buying habit through studying the itemsets which frequently appear together in the shopping basket. Consequently, managers can use the discovered result to plan his/her marketing or advertising strategies, store layout design, and so on. For example, if computer and printer are likely to be bought together, then a sale on printer may encourage the sale of printers as well as computers.

A formal definition of association rule is: Let J= {i1, i2…im} be a set of items. Let D be the set of database transactions where each transaction T is a set of items such that $T \subseteq J$. Each transaction is associated with an identifier, called TID. Let A be a set of items. A transaction T is said to contain A if and only if $A \subseteq T$. An association rule is an implication of the form $A \Rightarrow B$, where $A \subset J$, $B \subset J$, and $A \cap B = \emptyset$. The rule $A \Rightarrow B$ holds in the transaction set D with **support** s, where s is the percentage of transactions in D that contain $A \cup B$ (i.e., both A and B). This is taken to be the probability, P $(A \cup B)$. The rule $A \Rightarrow B$ has **confidence** c in the transaction set D if c is the percentage of transactions in D containing A that also contain B. This is taken to be the conditional probability, P (B\A). Mathematically **support** and **confidence** measures are implied as,

Support $(A \Rightarrow B) = P(A \cup B)$.

Confidence $(A \Rightarrow B) = P(B/A)$.

P(B/A) = Support Count (AUB) / Support Count (A).

A rule is said to be frequent if its support is greater than the minimum support threshold, and it is said to be strong if its confidence is more than the minimum confidence threshold.

3. Algorithms

3.1. Apriori Algorithm

The Apriori algorithm is one of the most important algorithms for association rule mining. Apriori is a seminal algorithm for finding frequent itemsets using candidate generation [2]. Most of the other algorithms are based on it or extensions of it. It is a main-memory based algorithm. Main memory imposes a limitation on the size of the dataset that can be mined.

One of the most popular data mining approaches is, finding frequent itemsets from a transaction dataset and deriving association rules. Once frequent itemsets are obtained, it is easy to generate association rules with confidence larger than or equal to a user specified minimum confidence.

It is characterized as a level-wise complete search algorithm using anti-monotonicity of itemsets. An Apriori property is used to reduce the search space, "All non empty subsets of a frequent item set must also be frequent". By convention, Apriori assumes that items within a transaction or itemset are sorted in lexicographic order.

The algorithm executes in two steps i.e. frequent itemsets generation and association rule generation. The frequent itemsets generation is again a two step process:

- **Candidate itemsets (Ck) generation** i.e. all possible combination of items those are potential candidates for frequent itemsets.
- **Frequent itemsets (Fk) generation** - support for all candidate itemsets are generated and itemsets having support greater than the user-specified minimum support are qualified as the frequent itemsets.

Let, the set of frequent itemsets of size *k* be $F_{k,}$ and its candidate set be C_k. Apriori first scans the database and searches for frequent itemsets of size 1 by accumulating the count for each item and collecting those that satisfy the minimum support requirement. It then iterates on the following three steps and extracts all the frequent itemsets.

1) Generate C_{k+1}, candidates of frequent itemsets of size *k* +1, from the frequent itemsets of size *k*.
2) Scan the database and calculate the support of each candidate of frequent itemsets.
3) Add those itemsets that satisfies the minimum support requirement to F_{k+1}.

The working of the Apriori algorithm consists of two major steps:

1) **Join step:** Ck (candidate set) is generated by joining Lk-1 with itself (cartesian product Lk-1 x Lk-1).
2) **Prune step (A-Priori property):** Any (k − 1) size itemset that is not frequent can't be a subset of a frequent k size itemset, and hence it should be removed.

It is evident that Apriori scans the database at most k+1 times when the maximum size of frequent itemsets is set at k. The Apriori achieves good performance by reducing the size of candidate sets.

3.2. Apriori-C Algorithm

This section presents the APRIORI-C algorithm [3], which adapts the APRIORI algorithm for classification purposes. The advantages of APRIORI-C over its predecessors are lower memory consumption, decreased time complexity, and improved understandability of results. In APRIORI-C, the association rule mining algorithm "Apriori" is adapted for classification purposes by implementing the following steps:

1) Discretize continuous attributes.
2) Binarize all (discrete) attributes.
3) Perform data pre-processing through feature subset selection.
4) Run the optimized APRIORI algorithm by considering only the rules, whose right-hand side consist of a single item, representing the target class value.
5) Post-process the set of induced rules by rule ordering and best rule subset selection.
6) Use these rules to classify unclassified examples.

APRIORI-C algorithm includes the following optimizations to do classification effectively:

Classification rule generation:

Rules with a single target item at the right hand side can be created during the search. To do so, the algorithm needs to save only the supported itemsets of sizes k and k+1. This results in decreased memory consumption (improved by factor 10). However, this does not improve the algorithm's time complexity.

Prune irrelevant rules:

Classification rule generation can be suppressed if one of the existing generalizations of the rule has support and confidence above the given minSup and minConf thresholds. To prevent rule generation, the algorithm simply excludes the corresponding itemset from the set of supported itemsets of size k+1. Time and space complexity reduction are considerable (improved by factor 10 or more).

Prune irrelevant items:

If an item cannot be found in any of the itemsets containing the target item, then it is impossible to create a rule containing this item. Hence, APRIORI-C prunes the search by discarding all itemsets containing this item.

3.3. Apriori-SP Algorithm

In this section, Apriori-SP algorithm is illustrated briefly. Apriori-SP algorithm [4] is the improved version of Apriori algorithm. In this algorithm, the basic Apriori algorithm is improved with Sampling and Partitioning techniques. The "SP" in the name of this algorithm came after the Sampling and Partitioning techniques only. The methodology of Apriori-SP algorithm is explained as follows,

1) Pick a random sample 'S', from the transaction database (D)
2) Partition the random sample (S) into subsets (or partitions), so that each partition size is appropriate to the computer memory.
3) Read one partition into the memory.
4) Collect all the candidate 1-itemsets in the partition.
5) Count the number of occurrences of each 1-itemsets to create the candidate set called C_1. {C_1: candidate 1-items}.
6) Delete the infrequent candidate 1-items (i.e., itemsets that does not meet local-minimum-support requirement) to generate a frequent or large 1-itemset called L_1. {Local-minimum-support = (User threshold/ Number of partitions)}
7) Join L_1 with itself to generate candidate 2-itemsets C_2. C_2 consists of all combinations of 2-itemsets and the algorithm checks for the Apriori property that is, "all nonempty subsets of frequent itemsets must also be frequent" which is the foundation for pruning the infrequent itemsets.
8) Recursively perform steps 4 to 6 with the addition of an itemset every pass, until some L_k becomes empty which means that the join operation (step 6) is not possible.
9) Combine all frequent L_n-itemsets into one list called local frequent itemsets (LLp) for the given partition. <where n=1, 2… K>.
10) Recursively perform steps 2 to 8 for each partition.
11) Combine all LLp to generate global candidate itemsets (LG).
12) Scan the whole transactional database to count the number of occurrences of each candidate K-itemset in LG.
13) Delete infrequent candidate K-items (itemsets did not meet global minimum-support threshold) to generate global frequent itemset in the database (all partitions).

The Apriori-SP algorithms produces better results than Apriori due to the reduction of the number of scans (actually two scans) of the database.

3.4. Apriori_HEAVY Algorithm

This section presents the Apriori_heavy algorithm [5]. Suppose A is the set of all frequent 1-temsets in a given transaction database D and a collection H = {h_1, h_2… h_k} of *k* heavy itemsets in D. Let B be the set of all frequent items in A, which do not occur in any heavy itemset in S. Apart from the association rules consisting of items only in B, there may be additional association rules involving (i) relationships between items in different heavy itemsets in H; and (ii) relationships between items in B and items in one or more heavy itemsets in H.

Apriori_heavy is an association rule mining algorithm, which uses H and B as given inputs and finds the set of all other "missing" association rules. The algorithm also finds more heavy itemsets, not necessarily disjoint from the given ones and adds them to H. Thus the generated collection of heavy itemsets H and the generated association rules complete the mining process.

The algorithm Apriori_heavy is nearly the same as the original Apriori algorithm, except for the following. The initial candidate itemsets are of size 2, obtained by taking pair-wise Cartesian product of the heavy itemsets in H among themselves and with the set B of "non-heavy" frequent items. After finding the frequent *k* itemsets, the algorithm checks (using subroutine is_heavy) if any of them are heavy itemsets; if so, then it removes that set from L_k and adds it to H, taking care to remove all proper subsets of the newly added heavy itemset from H. Since the set L_k may become empty in this process, the terminating condition stated differently (stop when no new frequent itemsets are found).

The apriori_heavy algorithm usually shows a substantial improvement over the performance of the Apriori algorithm, due to its use of heavy itemsets.

3.5. Apriori-INVERSE Algorithm

This section presents the Apriori-Inverse algorithm. Like Apriori, this algorithm is based on a level-wise search. On the first pass through the database, an inverted index is built using the unique items as keys and the transaction IDs as data. At this point, the support of each unique item (the 1-itemsets) in the database is available as the length of each data chain. To generate *k*-itemsets under maxsup, the $(k-1)$-itemsets are extended in precisely the same manner as Apriori to generate candidate k-itemsets. That is, a $(k-1)$-itemset *i*1 is turned into a *k*-itemset by finding another $(k-1)$-itemset *i*2 that has a matching prefix of size $(k-2)$, and attaching the last item of *i*2 to *i*1. For example, the 3- itemsets {1, 3, 4} and {1, 3, 6} can be extended to form the 4-itemset {1, 3, 4, 6}, but {1, 3, 4} and {1, 2, 5} will not produce a 4-itemset due to their prefixes not matching right up until the last item.

These candidates are then checked against the inverted index to ensure they at least meet a *minimum absolute support* requirement (say, at least 5 instances) and are pruned if they do not (the length of the *intersection* of a data chain in the inverted index provides support for a k-itemset with k larger than 1).

The process continues until no candidate itemsets can be generated, and then association rules are formed in the usual way. It should be clear that Apriori-Inverse finds all perfectly sporadic rules, since we have simply inverted the downward-closure principle of the Apriori algorithm; rather than all subsets of rules being over minsup, all subsets are under maxsup.

Since making a candidate itemset longer cannot increase its support, all extensions are viable *except* those that fall under our minimum absolute support requirement. Those exceptions are pruned out, and are not used to extend itemsets in the next round.

Apriori-Inverse does not find any imperfectly sporadic rules, because it never considers itemsets that have support above maxsup; therefore, no subset of any itemset that it generates can have support above maxsup [6]. However, it can be extended easily to find imperfectly sporadic rules that are nearly perfect: for instance, by setting maxsup*i* to maxsup/minconf where maxsup *i* is maximum support for imperfectly sporadic rules and maxsup is maximum support for reported sporadic rules.

3.6. REVERSE Apriori Algorithm

Apriori algorithm collects candidate itemsets. A candidate itemset includes the items that have the possibility to be member of frequent itemsets and is used to discover frequent itemsets. It then discovers the frequent itemsets from the candidate itemsets.

Apriori at first finds candidate-1 itemsets. It then finds the frequent-1 itemsets by pruning candidate-1 itemsets. The pruned items are those which don't satisfy the minimum support value. Then, it generates candidate-2 itemsets from frequent-1 itemsets. Apriori algorithm generates the frequent-2 itemsets from the candidate-2 itemsets in the same process as it generated the frequent-1 itemsets.

Reverse Apriori algorithm [7] is different from the Apriori algorithm in that it generates large frequent itemsets starting from considering maximum possible number of items in the dataset. It generates this large frequent itemsets only if it satisfies the user specified minimum item support. It then gradually decreases the number of items in the itemsets until it gets largest frequent itemsets. At first, this algorithm checks for large frequent itemsets with all the combinations of the distinct values for all the items in the target dataset. If it is satisfied by minimum support value, then large frequent itemsets are revealed at first checking. If it is not satisfied, then checks for next large frequent itemsets by combinations of next large number of items and thus generate large frequent itemsets by checking the minimum support value.

Whenever Reverse Apriori algorithm finds frequent itemsets, it does not go to next searching step to check larger frequent itemsets like Apriori does. The working nature of Reverse Apriori algorithm is described in the following steps:

1) First, the total number of attributes is calculated.
2) Then, for all combinations of *k* number of attributes, every combination is checked for predefined support.
3) If it satisfies the support value, then the combination is the member of frequent itemsets. And, the itemset is said to be the large frequent itemset.
4) If the large frequent itemset is not generated, then the above said process (steps 1, 2, 3) continues until the large frequent itemset is generated.

In Apriori, large frequent itemsets were generated after fourth pass of the database, whereas in Reverse Apriori, it stops scanning as soon as it gets an itemset having items which satisfy the minimum support value.

3.7. QUANTITATIVE Apriori Algorithm

The Quantitative Apriori algorithm is an enhanced version of Apriori. This algorithm is used to minimize the number of candidate sets while generating association rules by evaluating quantitative information associated with each item that occurs in a transaction, which is usually discarded, as traditional association rules focus just on qualitative correlations. The algorithm for generating quantitative association rules starts by counting the item ranges in the database, in order to find out the frequent ones [8]. These frequent item ranges are the basis for generating higher order item ranges. The size of a transaction must be considered as the number of items that it comprises.

- Define an item set 'm' as a set of items of size 'm'.
- Specify frequent (large) item sets by 'Fm'.
- Specify candidate item sets (possibly frequent) by 'Lm'.

A 'n' range set is a set of n- item ranges, and each m-item set has a n-range set that stores the quantitative rules of the item set. During each iteration of the algorithm, the system uses the frequent sets from the previous iteration to generate the candidate sets and check whether their support is above the threshold. The set of candidate sets found is pruned by a strategy that discards sets which contain infrequent subsets. The algorithm ends when there are no more candidates' sets to be verified.

The enhancement of Apriori is done by increasing the efficiency of candidate pruning phase by reducing the number of candidates that are generated for further verification. This algorithm uses quantitative information to estimate more accurately the overlap in terms of transactions. The major elements that should be considered in this algorithm are the number of transactions, average size of transaction, average size of the maximal large item sets, number of items, and distribution of occurrences of large item sets.

This algorithm has no need to maintain the covers of all past itemsets into main memory. In this way, this level-wise algorithm accesses a database less often than Apriori and requires less memory because of the utilization of additional upward closure properties. Though the performance of quantitative Apriori is considerably lower than Apriori, they are promising for the cases when sufficient memory for supporting full replication may not be available.

This algorithm reduces not only the number of itemsets generated but also the overall execution time of the algorithm. Any valued attribute will be treated as quantitative and will be used to derive the quantitative association rules which usually increases the rules' information content. Transaction reduction is achieved by discarding the transactions that does not contain any frequent itemset in subsequent scans which in turn reduces overall execution time.

Conclusion

Association Rule Mining is one of the well known research area in the field of data mining. It aims at extracting interesting correlations, frequent patterns, associations among sets of items in the transaction databases or other data repositories. Apriori is one of the algorithms used in association rule mining to find out the correlation between the items that present in the transaction database. In this paper, a wide survey has been carried out to find the efficiency, advantages, and drawbacks of various Apriori-like algorithms.

References

[1] Jiawei Han and Micheline Kamber, "*Data Mining: Concepts and Techniques*", Morgan Kaufmann Publishers, 2006.
[2] Agrawal R, Srikant R, "*Fast algorithms for mining association rules*", Proceedings of the 20th VLDB conference, pp.487–499, 1994.
[3] Jovanoski, V. and N. Lavrac., "*Classification rule learning with APRIORI-C*", Progress in Artificial Intelligence: Proceedings of the 10th Portuguese Conference on Artificial Intelligence, pp.44–51, Springer, 2001.
[4] F. A. El-Mouadib, Salem A. Mohammed, "*Combining Some of The Improvements of Apriori Algorithm*", Science and Its Applications, Part-II: Mathematical Sciences – Chapter-9, Garyounis University, 2006.
[5] Palshikar G., Kale M., Apte M., "*Association rules mining using heavy itemsets*", Data & Knowledge Engineering 61(1), Elsevier, pp.93-113, 2007.
[6] Yun Sing Koh and Nathan Rountree, "*Finding Sporadic Rules Using Apriori-Inverse*", Advances in Knowledge Discovery and Data Mining, Springer-Berlin/ Heidelberg, ISSN 0302-9743, pp.97-106, 2005.
[7] Kamrul Abedin Tarafer, Shah Mostafa Khaled, "*Reverse Apriori Algorithm for Frequent Pattern Mining*", Asian Journal of Information Technology 7(12), ISSN: 1682-3915, pp.524-530, Medwell Journals, 2008.
[8] S.Prakash, R.M.S.Parvathi, "*An Enhanced Scaling Apriori for Association Rule Mining Efficiency*", European Journal of Scientific Research 2(39), ISSN 1450-216X, pp.257-264, 2010.
[9] M. H.Marghny and A.A.Mitwaly., "*Fast Algorithm for Mining Association Rules*", In proc. Of the First ICGST International Conference on Artificial Intelligence and Machine Learning AIML05, pp.36-40, 2005.