ISSN (Online) : 2319 – 8753
ISSN (Print) : 2347 - 6710

International Journal of Innovative Research in Science, Engineering and Technology

An ISO 3297: 2007 Certified Organization Volume 3, Special Issue 4, April 2014

Two days National Conference – VISHWATECH 2014

On 21st & 22nd February, Organized by

Department of CIVIL, CE, ETC, MECHNICAL, MECHNICAL SAND, IT Engg. Of Vishwabharati Academy's College of engineering, Ahmednagar, Maharastra, India

The Apriori algorithm: Data Mining Approaches Is To Find Frequent Item Sets From A Transaction Dataset

Abhang Swati Ashok, JoreSandeep S.

Assistant Professor, Dept. of IT, College of Engineering, Kopargaon, Maharashtra, India

Abstract: Aprioriis an algorithm for learning association rules. Apriori is designed to operate on databases containing transactions. As is common in association rule mining, given a set of item sets, the algorithm attempts to find subsets which are common to at least a minimum number candidate C of the item sets. Apriori uses a "bottom up" approach, where frequent subsets are extended one item at a time, and groups of candidates are tested against the data. The algorithm terminates when no further successful extensions are found. The purpose of the Apriori Algorithm is to find associations between different sets of data. It is sometimes referred to as "Market Basket Analysis". Each set of data has a number of items and is called a transaction. The output of Apriori is sets of rules that tell us how often items are contained in sets of data.

Keywords:: itemsets, transaction IDs, patterns Apriori, candidate key

I. INTRODUCTION

One of the most popular data mining approaches is to find frequent itemsets from a transaction dataset and derive association rules. Finding frequent itemsets (itemsets with frequency larger than or equal to a user specified minimum support) is not trivial because of its combinatorial explosion. Once frequent itemsets are obtained, it is straightforward to generate association rules with confidence larger than or equal to a user specified minimum confidence[1].

II. DESCRIPTION OF THE ALGORITHM

Apriori is a seminal algorithm for finding frequent itemsets using candidate generation. It is characterized as a level-wise complete search algorithm using anti-monotonicity ofitemsets, "if an itemset is not frequent, any of its superset is never frequent"[2]. By convention,Apriori assumes that items within a transaction or itemset are sorted in lexicographic order.Let the set of frequent itemsets of size k be Fkand their candidates be C_k. Apriori first scans the database and searches for frequent itemsets of size 1 by accumulating the count for eachitem and collecting those that satisfy the minimum support requirement. It then iterates onthe 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 itemsetsof 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}.[1]

The Apriori algorithm is shown in fig 1 as follows[4]

Function apriori-gen in line 3 generates $C_{k+1}$from F_k in the following two step process:

1. Join step: Generate R_{K+1}, the initial candidates of frequent itemsets of size k + 1 by taking the union of the two frequent itemsets of size k, Pk and Q_k that have the first k−1elements in common.

$R_{K+1} = P_k \cup Q_k$= {iteml, . . . ,i
temk−1, itemk,itemk_ }
P_k= {iteml,i tem2, . . . , i temk−1, itemk}
O_k= {iteml,i tem2, . . . , i temk−1, itemk_

}
where, iteml<i tem2 <· · · <itemk<itemk_ .

```
$f_i$=(frequent itemsets of cardinality 1);
    for(k=1;$F_k \neq \Phi$ ;k++) do begin
        $C_{k+1}$= apriori-gen($F_k$);//new candidate
            for all transactions t $\in$ Database   do begin
                $C_t'$=subset($C_{k+1}$,t)// candidate contained in t
                For all candidate c $\in C_t'$ do
                c.count++;
                end
                Fk+1={C $\in C_{k+1}$| c.count$\geq$ minimum support}
                end
    end
    Answer $U_k F_k$;
```

Fig 1.Apriori Algorithm

2. Prune step: Check if all the itemsets of size k in Rk+1 are frequent and generate$_{Ck+1}$ by removing those that do not pass this requirement from Rk+1.

This is because any subset of size k of Ck+1 that is not frequent cannot be a subset of a frequent itemset of size k + 1.

Function subset in line 5 finds all the candidates of the frequent itemsets included in transaction t. Apriori, then, calculates frequency only for those candidates generated this way by scanning the database.

It is evident that Apriori scans the database at most kmax+1 times when the maximum size of frequent itemsets is set at kmax.

The Apriori achieves good performance by reducing the size of candidate sets (Fig. 1).However, in situations with very many frequent itemsets, large itemsets, or very low minimum support; it still suffers from the cost of generating a huge number of candidate sets and scanning the database repeatedly to check a large set of candidate itemsets. In fact, it is necessary to generate 2100 candidate itemsets to obtain frequent itemsets of size 100.[1]

III. EXAMPLE[3][7]

Assume that a large supermarket tracks sales data by stock-keeping unit (SKU) for each item: each item, such as "butter" or "bread", is identified by a numerical SKU. The supermarket has a database of transactions where each transaction is a set of SKUs that were bought together.

Let the database of transactions consist of following itemsets shown in fig 2 a):

We will use Apriori to determine the frequent item sets of this database. To do so, we will say that an item set is frequent if it appears in at least 3 transactions of the database: the value 3 is thesupport threshold.

The first step of Apriori is to count up the number of occurrences, called the support, of each member item separately, by scanning the database a first time. We obtain the following result as fig 2 b).

Itemsets
{1,2,3,4}
{1,2,4}
{1,2}
{2,3,4}
{2,3}
{3,4}
{2,4}

a) Database

Item	Support
{1}	3
{2}	6
{3}	4
{4}	5

b) output of step 1

Item	Support
{1,2}	3
{1,3}	1
{1,4}	2
{2,3}	3
{2,4}	4
{3,4}	3

c) output of step 2

Item	Support
{2,3,4}	2

d) output of step 3 : final output

Fig 2. . database and output at different steps

All the itemsets of size 1 have a support of at least 3, so they are all frequent.
The next step is to generate a list of all pairs of the frequent items, shown in fig 2 c)

The pairs {1,2}, {2,3}, {2,4}, and {3,4} all meet or exceed the minimum support of 3, so they are frequent. The pairs {1,3} and {1,4} are not. Now, because {1,3} and {1,4} are not frequent, any larger set which contains {1,3} or {1,4} cannot be frequent. In this way, we can prune sets: we will now look for frequent triples in the database, but we can already exclude all the triples that contain one of these two pairs, shown in fig 2 d).

In the example, there are no frequent triplets -- {2,3,4} is below the minimal threshold, and the other triplets were excluded because they were super sets of pairs that were already below the threshold.
We have thus determined the frequent sets of items in the database, and illustrated how some items were not counted because one of their subsets was already known to be below the threshold.

IV. THE IMPACT OF THE ALGORITHM[8]

Many of the patterns finding algorithms such as decision tree, classification rules and clustering techniques that are frequently used in data mining have been developed in machine learning research community. Frequent pattern and association rule mining is one of the few exceptions to this tradition. The introduction of this technique boosted data mining research and its impact is tremendous. The algorithm is quite simple and easy to implement. Experimenting with Apriori-like algorithm is the first thing that data miners try to do.

V. CURRENT AND FURTHER RESEARCH[9]

Since Apriori algorithm was first introduced and as experience was accumulated, there have been many attempts to devise more efficient algorithms of frequent itemset mining. Many of them share the same idea with Apriori in that they generate candidates. These include hash-based technique, partitioning, sampling and using vertical data format. Hash-based technique can reduce the size of candidate itemsets.

Each itemset is hashed into a corresponding bucket by using an appropriate hash function. Since a bucket can contain different itemsets, if its count is less than a minimum support, these itemsets in the bucket can be removed from the candidate sets. A partitioning can be used to divide the entire mining problem into n smaller problems. The dataset is divided into n non-overlapping partitions such that each partition fits into main memory and each partition is mined separately. Since any itemset that is potentially frequent with respect to the entire dataset must occur as a frequentitemset in at least one of the partitions, all the frequent itemsets found thisway are candidates,which can be checked by accessing the entire dataset only once.

Sampling is simply to minea random sampled small subset of the entire data. Since there is no guarantee that we can find all the frequent itemsets, normal practice is to use a lower support threshold. Trade off has tobe made between accuracy and efficiency.

Apriori uses a horizontal data format, i.e. frequentitemsets are associated with each transaction. Using vertical data format is to use a differentformat in which transaction IDs (TIDs) are associated with each itemset. With this format,mining can be performed by taking the intersection of TIDs. The supportcount is simply thelength of the TID set for the itemset. There is no need to scan the database because TID setcarries the complete information required for computing support.

The most outstanding improvement over Apriori would be a method called FP-growth (frequent pattern growth) that succeeded in eliminating candidate generation. It adoptsa divide and conquer strategy by (1) compressing the database representing frequent itemsinto a structure called FP-tree (frequent pattern tree) that retains all the essential informationand (2) dividing the compressed database into a set of conditional databases, each associatedwith one frequent itemset and mining each one separately. It scans the database only twice.Inthefirst scan, all the frequent items and their support counts(frequencies)are derived andthey are sorted in the order of descending support count in each transaction. In the secondscan, items in each transaction are merged into a prefix tree and items (nodes) that appearin common in different transactions are counted. Each node is associated with an item andits count. Nodes with the same label are linked by a pointer called node-link. Since itemsare sorted in the descending order of frequency, nodes closer to the root of the prefix treeare shared by more transactions, thus resulting in a very compact representation that storesall the necessary information.

Pattern growth algorithm works on FP-tree by choosing anitem in the order of increasing frequency and extracting frequent itemsets that contain thechosen item by recursively calling itself on the conditional FP-tree. FP-growth is an order ofmagnitude faster than the original Apriori algorithm.

There are several other dimensions regarding the extensions of frequent pattern mining.The major ones include the followings:

(1) Incorporating taxonomy in items: Use oftaxonomymakes it possible to extract frequent itemsets that are expressed by higher conceptseven when use of the base level concepts produces only infrequent itemsets.

(2) Incremental mining: In this setting, it is assumed that the database is not stationary and a new instanceof transaction keeps added. The algorithm inupdates the frequent itemsetswithoutrestarting from scratch.

(3) Using numeric valuable for item: When the item corresponds to acontinuous numeric value, current frequent itemset mining algorithm is not applicable unlessthe values are discretised. A method of subspace clustering can be used to obtain an optimalvalue interval for each item in each itemset.

(4) Using other measures than frequency,such as information gain or $\chi 2$ value: These measures are useful in finding discriminativepatterns but unfortunately do not satisfy anti-monotonicity property. However, these measureshave a nice property of being convex with respect to their arguments and it is possibleto estimate their upperbound for supersets of a pattern and thus prune unpromising patternsefficiently. AprioriSMP uses this principle.

(5)Using richer expressions than itemset:Many algorithms have been proposed for sequences, tree and graphs to enable mining frommore complex data structure.

ISSN (Online) : 2319 – 8753
ISSN (Print) : 2347 - 6710

International Journal of Innovative Research in Science, Engineering and Technology

An ISO 3297: 2007 Certified Organization Volume 3, Special Issue 4, April 2014

Two days National Conference – VISHWATECH 2014

On 21st & 22nd February, Organized by

Department of CIVIL, CE, ETC, MECHNICAL, MECHNICAL SAND, IT Engg. Of Vishwabharati Academy's College of engineering, Ahmednagar, Maharastra, India

(6) Closed itemsets: A frequent itemset is closed if it isnot included in any other frequent itemsets. Thus, once the closed itemsets are found, all thefrequentitemsets can be derived from them. LCM is the most efficient algorithm to find theclosed item sets.

VI. CONCLUSION

In this paper we have discussed the Apriori algorithm of data mining which is used to find frequent itemset from transaction dataset and derive association rules. Also we have discussed different current and future research about the Apriori algorithm.

REFERENCES

[1] Agrawal R, Srikant R (1994) Fast algorithms for mining association rules. In: Proceedings of the 20thVLDB conference, pp 487–499
[2] Ahmed S, Coenen F, Leng PH (2006) Tree-based partitioning of date for association rule mining. KnowlInfSyst 10(3):315–331
[3] http://www.ijarcsse.com/docs/papers/Volume_3/6_June2013/V3I6-0192.pdf
[4] http://en.wikibooks.org/wiki/Data_Mining_Algorithms_In_R/Frequent_Pattern_Mining/The_Apriori_Algorithm
[5] http://en.wikipedia.org/wiki/Apriori_algorithm
[6] http://www.cs.umd.edu/~samir/498/10Algorithms-08.pdf
[7] http://software.ucv.ro/~cmihaescu/ro/teaching/AIR/docs/Lab8-Apriori.pdf
[8] http://www.cs.ucsb.edu/~xyan/papers/dmkd07_frequentpattern.pdf
[9] Xindong Wu · Vipin Kumar · J. Ross Quinlan · JoydeepGhosh · Qiang Yang · Hiroshi Motoda · Geoffrey J. McLachlan · Angus Ng · Bing Liu · Philip S. Yu · Zhi-Hua Zhou · Michael Steinbach · David J. Hand · Dan Steinberg,"Top 10 algorithms in data mining",KnowlInfSyst (2008) 14:1–37
DOI 10.1007/s10115-007-0114-2