Mining Frequent Itemsets Using Apriori Algorithm

Jogi.Suresh[1], T.Ramanjaneyulu [2]
[1] Dept. of CSE, KL UNIVERSITY, Vijayawada, A.P., 535003, India
[2] Dept. of IT, JNTUK UCEV, Vizianagaram, A.P., 535003, India

***Abstract*-Mining required data from voluminous Data has been recognized as one of the most challenging problems in data mining approach. In many real world scenarios, the data is not extracted from single data source but from distributed and heterogeneous data sources. The discovered knowledge is expected comprehensive so that it can better fit in business environment Enterprise data mining applications involve dealing with complex data such as data from multiple heterogeneous data sources, extracting data in single step from such data sources such data sources is time and space consuming. So effective approaches are needed to decrease the time as well as space. Here we use Apriori Algorithm for discovering informative patterns in complex data sets.**

***Keywords*-heterogeneous data, complex data, frequent patterns, informative patterns.**

I. Introduction

In data mining, association rule learning is a popular and well researched method for discovering interesting relations between variables in large datasets. Piatetsky-Shapiro describes analyzing and presenting strong rules discovered in databases using different measures of interestingness patterns. Based on the concept of the strong rules, Agrawal introduced an association rules for discovering regularities between products in large scale transaction data recorded by point-of-sale (POS) systems in all supermarkets. For example the rule {onion, potatoes}=>{burger} found in the sales data of a supermarket would indicate that if a customer buys onions and potatoes together, he or she is likely to buy burger. Such information can be used as the basis for decisions about marketing database activities such as promotional pricing or product placements.

In addition to the above example from the market basket analysis association rules are Employed today in many application areas including Web usage mining, intrusion detection and bioinformatics etc... In computer science and data mining approach, Apriori is a classic algorithm for learning association rules. Apriori is designed to operates on different databases, it contains different transactions. For example, collections of items bought by different customers, or details of a website frequentation. And other algorithms are designed for finding an association rules in data having no transactions or having no timestamps.

This paper is organized in the fallowing way: In section 2 we present Apriori algorithm, In section 3 we present the General Process of Apriori Algorithm, In sections 4 we present Sample usage of Apriori algorithm, In section 5 we present conclusions of the research.

II. Apriori algorithm

The Apriori is the best-known algorithm to mine association rules. It uses a breadth-first search technique to counting the support of itemsets and uses a candidate generation function which exploits the downward closure property of support. An apriori uses a bottom up strategy, where frequent subsets are extended one item at a time it is known as candidate key generation, and groups of candidates are tested against the data.

Apriori algorithm uses breadth-first search and a tree structure to count candidate item sets an efficiently. It generates the candidate item sets of length k from item sets of length $k-1$. Then it prunes the candidates which have an infrequent sub pattern. According to the downward closure property, the candidate set contains all frequent k-length item sets. After that, it scans again the transaction database to determine frequent item sets among the candidates.

Apriori, while historically significant and it suffers from a number of inefficiencies or trade-offs, compare with other algorithms. Candidate key generation generates large numbers of subsets. The algorithm is attempts to load up the candidate set with as many as possible before each scan. Bottom-up subset exploration (essentially a breadth-first traversal of the subset lattice) finds any maximal subset S only after all 2^lsl− 1 of its proper subsets.

Useful Concept

To illustrate the concepts, we use a small example from the supermarket domain. The set of items is *I* = {milk, bread, butter, beer} and a small database containing the items (1 codes presence and 0 absence of an item in a transaction) is shown in the below table. An example rule for the supermarket could be {milk, bread} => {butter} meaning that if milk and bread is bought, customers also buy butter. This example is small. In practical applications a rule needs a support of several hundred transactions before it can be considered statistically significant and datasets often contain thousands or millions of transactions.

For select interesting rules from the set of all possible rules, we constraints on various measures of significance and interests. The best-known constraints are minimum thresholds on support and confidence.

Support

The support supp(*X*) is an itemset *X* is defined as the proportion of transactions in the data set which contain the itemset.

$$supp(X)=\frac{no.of\ transactions\ which\ contain\ the\ itemset\ X}{total\ no.of\ transactions}$$

In the example database, the itemset {milk, bread, butter} has a support of 4 /15 = 0.26 since it occurs in 26% of all transactions. To be even more explicit we can point out that 4 is the number of transactions from the database which contain the itemset {milk, bread, butter} while 15 represents the total number of transactions.

Confidence

The *confidence* of a rule is defined as:

$$Conf(X \rightarrow Y) = \frac{supp(X \cup Y)}{supp(X)}$$

For the rule {milk, bread} => {butter} we have the following confidence:

Supp ({milk, bread, butter}) / supp ({milk, bread}) = 0.26 / 0.4 = 0.65

This means that for 65% of the transactions containing milk and bread the rule is correct.

Confidence can be used for an estimate of the probability $P(Y \mid X)$, the probability of finding the RHS of the rule in transactions under the condition that these transactions also contain the LHS.

Lift

The *lift* of a rule is defined as:

$$Lift(X \rightarrow Y) = \frac{supp(X \cup Y)}{supp(Y) * supp(X)}$$

The rule {milk, bread}=>{butter} has the following lift: supp({milk,bread,butter}) / supp({butter}) x supp({milk, bread})= 0.26/0.46 x 0.4= 1.4.

III. The General Process of Apriori Algorithm

Association rule generation is usually divided into two separate steps:

1. First, minimum support is applied to find all *frequent itemsets* in a database.

2. Second, these frequent itemsets and the minimum confidence constraint are used to form rules.

While the second step is straight forward, finding all frequent itemsets in a database is difficult since it involves searching all possible itemsets. The set of possible itemsets is the power set over *I* and has size 2n – 1, excluding the empty set which is not a valid item sets. Although the size of the powerset grows exponentially in the number of items *n* in *I*, efficient search is possible by using the downward-closure property of support (also called *anti-monotonicity*) which guarantees that for a frequent itemset, all its subsets are also a frequent and thus for an infrequent itemsets, all its supersets also must be infrequent.

Working of apriori algorithm

1) Find all frequent itemsets:
   - Get frequent items:
   - Items whose occurrence in database is greater than or equal to the min_support threshold.
   - Get frequent itemsets:
   - Generate candidates from frequent items.
   - Prune the results to find the frequent itemsets.
2) Generate strong association rules from frequent itemsets
   - Rules which satisfy the min_support and min_confidence threshold.

Apriori Algorithm Pseudo code

The following is process of finding associations using apriori algorithm.

A. Let's define:

Ck as a candidate itemset of size k.

Lk as a frequent itemset of size k.

B. Main steps of iteration are:

a. Find frequent set Lk-1.

b. Join step: Ck is generated by joining Lk-1 with itself (Cartesian product Lk-1 x Lk-1).

c. Prune step (apriori property): Any (k − 1) size itemset that is not frequent cannot be a subset of a frequent k size itemset, it should be removed.

d. Frequent set Lk has been achieved.

The Apriori algorithm is:

L1= {frequent items};

For (k= 2; Lk-1! =∅; k++) **do begin**

Ck= candidates generated from Lk-1 (that is Cartesian product Lk-1 x Lk-1 and eliminating any k-1 size itemset that is not frequent);

For each transaction t in database **do**

Incrementing the count of all candidates in Ck that are contained in t

Lk = candidates in Ck with min_sup

end

return Èk Lk;

As is common in association rule mining, given a set of *itemsets* (for instance, sets of retail transactions, each listing individual items is purchased), the algorithm attempts to finding subsets which are common to at least a minimum number C of the itemsets. Apriori uses a bottom up strategy, where frequent subsets are extended one item at a time it is known as *candidate key generation*, and groups of candidate keys are tested against the data. The algorithm terminate when no further successful extensions are found.

Methods to Improve Apriori's Efficiency

1. *Hash-based itemset counting:* A *k*-itemset whose corresponding hashing bucket count is below the threshold cannot be frequent.
2. *Transaction reduction:* A transaction that does not contain any frequent k-itemsetis useless in subsequent scans.
3. *Partitioning:* Any itemset that is potentially frequent in DB must be frequent in at least one of the partitions of DB.
4. *Sampling:* mining on a subset of given data, lower support threshold + a method to determine the completeness.
5. *Dynamic itemset counting:* add new candidate itemsets only when all of their subsets are estimated to be frequent.

IV. Sample usage of Apriori algorithm

A large supermarket tracks sales data by Stock-keeping unit (SKU) for each item, and thus it is able to know what items are typically purchased together. An apriori is a moderately efficient way to build a list of frequent purchased item pairs from this data. Let the database of all transactions consist of the sets {1,2,3,4},{1,2,3,4,5}, {2,3,4}, {2,3,5}, {1,2,4}, {1,3,4}, {2,3,4,5}, {1,3,4,5}, {3,4,5}, {1,2,3,5}.Each number corresponds to a product such as "butter" or "water". The first step of Apriori is to count up the frequencies in data set, called the supports, of each member item separately.

Item	Support
1	6
2	7
3	9
4	8
5	6

We can define a minimum support level to qualify as "frequent," which depends on the context. For this case, let min support = 4. Therefore, all are frequent. The next step is to generate a list of all 2-pairs of the frequent items. Had any of the above items not been frequent, they would not have been included as a possible member of possible 2-item pairs. In this way, Apriori *prunes* the tree of all possible data sets. In next step we again select only these items (now 2-pairs are items) which are frequent (the pairs written in bold text).

Item	Support
{1,2}	4
{1,3}	5
{1,4}	5
{1,5}	3
{2,3}	6
{2,4}	5
{2,5}	4
{3,4}	7
{3,5}	6
{4,5}	4

Here we generate list of all 3-triples of the frequent items from the dataset (by connecting frequent pair with frequent single item).

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

The algorithm will end here because the pair {2, 3, 4, 5} generated at the next step does not have the desired support.

We will now apply the same algorithm on the same set of data considering that the min support is 5. We get the following results is:

Step 1:

Item	Support
1	6
2	7
3	9
4	8
5	6

Step 2:

Item	Support
{1,2}	4
{1,3}	5
{1,4}	5
{1,5}	3
{2,3}	6
{2,4}	5
{2,5}	4
{3,4}	7
{3,5}	6
{4,5}	4

The algorithm ends here because here we don't have the 3-triples generated at Step 3 have de desired support.

V. CONCLUSION

Apriori is One of the most popular data mining approaches is to find frequent itemsets from a transaction dataset and derive association rules. Finding frequent itemsets 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. Further we use many algorithms for finding informative patterns from the complex data sources.

REFERENCES

[1] M., Suraj Kumar Sudhanshu, Ayush Kumar and Ghose M.K., "Optimized association rule mining using genetic algorithm Anandhavalli Advances in Information Mining" , ISSN: 0975–3265, Volume 1, Issue 2, 2009, pp-01-04.

[2] Daniel Hunyadi, "Performance comparison of Apriori and FP-Growth algorithms in generating association rules", Proceedings of the European Computing Conference.

[3] Jiawei Han, Jian Pei, and Yiwen Yin, "Mining Frequent Patterns without Candidate Generation" SIGMOD'2000 Paper ID: 196.

[4] Agrawal R, Srikant R (1994) Fast algorithms for mining association rules. In: Proceedings of the 20th VLDB conference, pp 487–499.

[5] J. Han, M. Kamber (2001), *Data Mining*, Morgan Kaufmann Publishers, San Francisco, CA.

[6] R. Agrawal and R. Srikant. Fast algorithms for mining association rules in large databases. In J. Bocca, M. Jarke, and C. Zaniolo, editors, *Proc. Int. Conf. on Very Large Data Bases*, pages 478–499, Santiago, Chile, 1994.

[7] Craus M., Archip A., *A Generalized Parallel Algorithm for Frequent Itemset Mining*, Proceedings of the 12th WSEAS International Conference on Computers, Heraklion, Greece, 2008, pg. 520-523.

BIOGRAPHY

Jogi.suresh received the B.Tech degree in Information Technology from JNTU affiliated College. He is currently pursuing M.Tech. In Computer Science and Engineering from K.L.University, Vijayawada.

T.Ramanjaneyulu received the B.Tech degree in Information Technology from JNTU affiliated College and also received the M.Tech degree in Computer Science and Engineering from JNTU vizianagaram.