УДК 336.74:004.738.5

APRIORI FAST ALGORITHMS FOR MINING ASSOCIATION RULES

L. N.GIVEN
Master student, Department of Mathematical and Information Support of Economic Systems, Yanka Kupala State University of Grodno

Yanka Kupala State University of Grodno, Republic of Belarus
E-mail: g_lufungula@yahoo.com

Abstract. In this article I solve association rules between items in a large database of sales transactions also two algorithms for solving this problem that are fundamentally different algorithm.

Key words*:* apriori algorithm, association rule.

INTRODUCTION: Database mining is motivated by the decision support problem faced by largest retail organizations. Progress in bar-code technology has made it possible for retail organizations to collect and store massive amounts of sales data, referred to as the basket data. A record in such data typically consists of the transaction date and the items bought in the transaction. Successful organizations view such databases as important pieces of the marketing infrastructure. [1]

They are interested in instituting information-driven marketing processes, managed by database technology, that enable marketers to develop and implement customized marketing programs and strategies [2]. The problem of mining association rules over basket data was introduced in.

An example of such a rule might be that 98% of customers that purchase tires and auto accessories also get automotive services done. Finding all such rules is valuable for cross-marketing and attached mailing applications. Other applications include catalog design, add-on sales, store layout, and customer segmentation based on buying patterns. The databases involved in these applications are very large. It is imperative, therefore, to have fast algorithms for this task. [2]

The following is a formal statement of the problem:

Let $I = \{i_1; i_2; ...; i_m\}$ be a set of literals, called items. Let D be a set of transactions, where each transaction T is a set of items such that T. associated with each transaction is a unique identifier, called its TID. We say that a transaction T contains X, a set of some items in I, if X[T]. An association rule is an implication of the form $X \rightarrow Y$, where X[I], Y[I], and The rule $X \rightarrow Y$ holds in the transaction set D with confidence c if c% of transactions in D that contain X also contain Y. The rule $X \rightarrow Y$ has support s in the transaction set D if s% of transactions in D contain X [Y]. Our rules are somewhat more general than in that we allow a consequent to have more than one item.

Given a set of transactions D, the problem of mining association rules is to generate all association rules that have support and confidence greater than the user-specified minimum support (called minsup) and minimum confidence (called minsup) respectively. Our discussion is neutral with respect to the representation of D. For example, D could be a data le, a relational table, or the result of a relational expression.

An algorithm for finding all association rules, henceforth referred to as the AIS algorithm, was presented in [1]. Another algorithm for this task, called the SETM algorithm, has been proposed in [4].

In this paper, we present two new algorithms, Apriori and Apriori Tid, that is fundamentally from these algorithms. We present experimental results, using both synthetic and real-life data, showing that the proposed algorithms always outperform the earlier algorithms. The performance gap is shown to increase with problem size, and ranges from a factor of three for small problems to more than an order of magnitude for large problems. We then discuss how the best features of Apriori and Apriori Tid can be combined into a hybrid algorithm, called Apriori Hybrid. Experiments show that the Apriori Hybrid has excellent scale-up properties, opening up the feasibility of mining association rules over very large databases. [3]

Functional dependencies are rules requiring strict satisfaction. Consequently, having determined a dependency X! A, Any other dependency of the form X + Y! A redundant and do not generate it. The association rules we consider are probabilistic in nature. The presence of a rule X! A does not necessarily mean that X + Y! A also holds because the latter may not have minimum support. Similarly, the presences of rules X! Y and Y! Z does not necessarily mean that X! Z holds because the latter may not have minimum confidence. [5]

ALGORITHM APRIORI.

Apriori algorithm. The pass of the algorithm simply counts item occurrences to determine the large itemsets. A subsequent pass, say pass k, consists of two phases. First, the large item sets L_{k-1} found in the (k-1) the pass are used to generate the candidate item sets C_k , using the apriori-general function described.

```
1)  L_1 = {large 1-itemsets};
2)  for (k = 2; L_{k-1} =0; k++) do begin
 3)     C_k = Apriori-gen(L_{k-1}); // New candidates
4)      for all transactions t and E begin
5)      C_t = subset (C_k, t); // Candidate
6)         for all candidate's
7)         c.count ++;
8) end
9) L_k = { C_{K-1} -> minsup}
10)  end
11) Answer = U_k L_{-k};
```

Apriori Candidate Generation

The apriori-gen function takes as argument $\mathbf{L_{k-1}}$, the set of all large ($_{k-1}$)-item sets. It returns a superset of the set of all large k-item sets. The function works as follows. 1 First, in the join step, we join $\mathbf{L_{k-1}}$ with $\mathbf{L_{k-1}}$**:**

```
Insert in to C_k
Select p. item1, p. item2, ..., p. itemk_1, q. itemk_1
From L_{k-1} p, L_{k-1} q
Where p. item1 = q. item1, ..., p. itemk2 = q. itemk2, p. itemk1 < q. itemk_1;
```

Next, in the prune step, we delete all item sets c D C_k such that some ($_{k-1}$)-subset of c is not in L_{k-1}:

```
For all item sets C_d C_k do
for all (k_1)-subsets s of c do
if (Lk_{-1}) then
Delete c from C_k;
```

Example Let L_3 be **{{123}, {124}, {134}, {135}, {f234}}.** After the join step, C_4 will be **{{1234}, {1345}}.** The prune step will delete the item set **{1345}** because the item set **{1 4 5}** is not in L_3. We will then be left with only **{1234}** in $C_{4.}$

Contrast this candidate generation with the one used in the AIS and SETM algorithms. In pass k of these algorithms, a database transaction t is read and it is determined which of the large item sets in L_{k1} are present in t. Each of these large item sets l is then extended with all those large items that are present in t and occur later in the lexicographic ordering than any of the items in l. continuing with the previous example, consider a transaction {12345}. In the fourth pass, AIS and SETM will generate two candidates, {1234} and {1235}, by extending the large item set {123}. Similarly, an additional three candidate item sets will be generated by extending the other large item sets in L_3, leading to a total of 5 candidates for consideration in the fourth pass. Apriori, on the other hand, generates and counts only one item set, {1345}, because it concludes a priori that the other combinations cannot possibly have minimum support. [7]

• Pseudo-code:

```
Ck: Candidate item set of size k
Lk: frequent item set of size k
L1 = {frequent items};
for (k = 1; L k! = ∅; k++) do begin
Ck+1 = candidates generated from Lk;
for each transaction t in database do increment the count of all candidates in Ck+1 that are contained in t
Lk+1 = candidates in Ck+1 with min support end return Lk
```

CONCLUSIONS: I presented two new algorithms, Apriori and Apriori Tid, for discovering all significant association rules between items in a large database of transactions and compared these algorithms to the previously known algorithms

I did not consider the quantities of the items bought in a transaction, which are useful for some applications.

References

[1]. Tarek M. Anwar, Howard W. Beck, and Shamkant B. Navathe. Knowledge mining by imprecise querying: A classication-based approach. In IEEE 8th Int'l Conf. on Data Engineering, Phoenix, Arizona, February 1992.

[2]. Rakesh Agrawal, Christos Faloutsos, and Arun Swami. Ecient similarity search in sequence databases. In Proc. of the Fourth International Conference on Foundations of Data Organization and Algorithms, Chicago, October 1993. Also in Lecture Notes in Computer Science 730, Springer Verlag, 1993, 69-84.

[3]. Rakesh Agrawal, Sakti Ghosh, Tomasz Imielinski, Bala Iyer, and Arun Swami. An interval classier for database mining applications. In Proc. of the VLDB Conference, pages 560-573, Vancouver, British Columbia, Canada, August 1992.

[4]. D. H. Fisher. Knowledge acquisition via incremental conceptual clustering. Machine Learning, 2(2), 1987.

[5]. J. Han, Y. Cai, and N. Cercone. Knowledge discovery in databases: An attribute oriented approach. In Proc. of the VLDB Conference, pages 547-559, Vancouver, British Columbia, Canada, 1992.

[6]. M. Holsheimer and A. Siebes. Data mining: The search for knowledge in databases. Technical Report CS-R9406, CWI, Netherlands, 1994.

[7]. M. Hosma and A. Swami. Set-oriented mining of association rules. Research Report RJ 9567, IBM Almaden Research Center, San Jose, California, October 1993.

[8]. R. Krishnamurthy and T. Imielinski. Practitioner problems in need of database research: Research directions in knowledge discovery. SIGMOD RECORD, 20(3):76-78, September 1991.

[9]. P. Langley, H. Simon, G. Bradshaw, and J. Zytkow. Scientific Discovery: Computational Explorations of the Creative Process. MIT Press, 1987.

[10].H. Mannila and K.-J. Raiha. Dependency inference. In Proc. of the VLDB Conference, pages 155-158, Brighton, England, 1987.

[11].H. Mannila, H. Toivonen, and A. I. Verkamo. Efficient algorithms for discovering association rules. In KDD-94: AAAI Workshop on Knowledge Discovery in Databases, July 1994.

[12].S. Muggleton and C. Feng. Efficient induction of logic programs. In S. Muggleton, editor, Inductive Logic Programming. Academic Press, 1992.