一个改进的关联规则的频繁项目集数据挖掘算法

吴振光
(贺州学院初等教育系　广西贺州 542800)

摘　要　在关联规则中的 Apriori 算法，具有天生的缺陷，运行效果很不理想。为了克服 Apriori 算法的缺点，本文提出了一个改进的算法：在产生频繁项目集组合时，只需扫描数据库一次，这样就可以有效率地降低 I/O 的存取时间，更快速地找出符合使用者需求的关联规则。仿真实验表明，该算法是有效的。

关键词　数据挖掘，关联规则，频繁项目集

An Improved Algorithm of Data Mining Based on Association Rule of Frequent Itemstes

WU Zhen-Guang
(Department of Elementary Education, Hezhou University, Guangxi Hezhou 542800)

Abstract　The Apriori algorithm is one of the most frequently used algorithms. Its inherent defects lead to inefficient performance. In this paper, an improved method is proposed: only needs to scan whole database once during extracting the frequent itemsets from large database. Therefore, this algorithm can efficiently reduce the I/O time, and rapidly extract association rules which can meet requirements of users. Finally, the experiment results show: the proposed method is effective.

Keywords　Data mining, Association rule, Frequent itemsets

1　引言

数据挖掘的主要目的，便是要从拥有海量数据的数据库中，正确有效地提取出有潜在价值的信息及知识。近年来，数据挖掘技术一直在持续发展，目前已有许多成熟的方法与技术，如关联规则分析、聚类规则、分类规则等。其中关联规则主要是从庞大的交易记录数据库中，寻找出商品项目之间的关联性。在关联规则中最常被使用的方法为 Apriori 算法，本文针对 Apriori 算法的固有缺点，提出了对算法的优化，并用实验进行了验证。

2　Apriori 算法

关联规则的主要目的是从庞大的交易记录数据库中，寻找出商品项目之间的关联性。关联规则分析(Association rule analysis)又称为购物篮分析(Market Basket Analysis)，主要是探讨在大型数据库中，商品项目与其属性之间共同发生的关联，来分析消费者的购买行为及购买商品的消费模式。关联规则分析有两个参考依据，也就是支持度(Support)与信赖度(Confidence)，主要是用来衡量所找出的规则是否有意义。

关联规则分析的最终目的为“找出哪些商品项目会一起被购买”，例如“80%购买牛奶的人，也会同时购买面包”。决策者可针对所得到的关联规则，来移动商品架或针对商品进行促销的活动，所以利用这个技术除了可以分析顾客的消费行为、找出商品项目间彼此的关联性，同时也可改善商品货架的摆设方式。此举不但有利于提升商品的竞争力，同时也可以提升商品的销售利润。

关联规则中常用的 Apriori 算法由 Agrawal[1] 等人于 1994 提出，此算法是研究关联规则中最具代表性的方法。基于 Apriori 算法的关联规则推导过程包含两个步骤：

(1) 反复地产生候选项目集和扫瞄整个数据库，找出所有的频繁项目集(使用 Apriori 算法)，这部份就是 Apriori 算法的主体。

(2) 利用步骤(1)所产生的频繁项目集，推导出有意义的关联规则。

由此可知，Apriori 算法是利用简单渐进的组合方式，但其有两个效率上的瓶颈：

(1) 会产生大量的候选项目集

第二候选项目集是由第一频繁项目集两两合并所产生的，所以若第一频繁项目集中有 k 个项目，则共会产生 $(k-1)+(k-2)+\cdots+1$ 个第二候选项目集。此时，假设第一频繁项目集有 1000 个项目，则约会产生 50 万个第二候选项目集。

(2) 需要多次扫瞄数据库

因为会有大量的候选项目集产生，而且每一个候选项目集都必须扫瞄整个数据库，才能求取其支持度，所以会造成整体执行效率不佳。虽然之后有许多改进的方法被提出，如 AprioriTid[1]、AprioriHybrid[1]、DHP[2]、Partition[3]、DIC[4]、Cloum-Wise Apriori[5]、Multiple-Leve[6,7] 等等，但这些方法都是依据 Apriori 算法的架构做改进或延伸，其算法的效率，并未得到很大的改善[8]。

3　算法的改进

本文的目的主要在于改善产生频繁项目集时所需花费的时间。我们利用拆分交易记录的方式，将每一笔交易记录拆分到最小项目为止，进而得到单笔交易记录所有的项目集组合，所以当整个数据库读取完毕时，所有的交易记录也就随之

吴振光　讲师，主要研究方向：代数论、数据挖掘、信息处理等。

拆分完成，同时可以得到所有的项目集组合。而后，使用者便可动态的任意输入最小门坎值（即最小支持度与最小信赖度），来产生所需的频繁项目集以及关联规则。

本文所提出的改进算法，只须扫瞄数据库一次，除了降低I/O的存取时间外，在执行效率上也非常平稳，并不会随着最小支持度的变动，出现执行效率上的差异。算法的运行方式相当简单快速，其主要规则为：在读取到每一笔交易记录时，将该笔交易记录可能产生的子集合一一拆分完成（例如：ABC此笔交易记录，可拆分出 AB、AC、BC、A、B、C 等项目集组合），拆分后的单笔记录的所有项目集组合，都会储存在暂存的二叉树（Binary Tree）中。单笔记录拆分完成后，再统一存入一个记录项目集次数的最终二叉树中进行计数。而且在存入时，若此项目集已存在于最终二叉树中，则将其项目集之值加1；若不存在则将此项目集加入，并给予初值1。所以当数据库中的交易记录已读取完毕时，表示所有的交易记录也已被拆分完成。此时，可得到所有的项目集组合以及项目集的支持度。接着，只须等待使用者输入最小支持度以及最小信赖度。之后，先产生频繁项目集，再产生符合条件的关联规则。而在使用者输入最小门坎值的这个部份，使用者可动态地任意更新最小门坎值的数值，而这个过程并不须要重新再扫描整个数据库，进行重拆分的动作。

3.1 改进的算法的流程图

算法共分三个步骤：

（1）读取到交易记录后，随及进行一连串“拆分”的动作，被拆分的项目集组合都会暂存在二叉树中，而拆分的主要目的是要将交易记录拆分到最小项目（$k=1$）为止。如此，当交易记录拆分完成后，即可得到此笔交易记录的所有项目集组合。最后，将暂存于二叉树中单笔记录的所有项目集组合，储存至最终二叉树中进行计数，若此项目集已存在最终二叉树中，只须将项目集之值加1；若未存在则将该项目集加入最终二叉树中，并给予初值1。重复上述的动作直到没有任何交易记录为止。

（2）根据使用者设定的最小支持度，从最终二叉树中取出符合条件的项目集，即称之频繁项目集。

（3）利用步骤二产生的频繁项目集以及使用者所设定的最小信赖度，推导出符合使用者所需的关联规则。

3.2 改进的算法的说明

算法主程序的伪代码如下：

```
Input: Transaction DB
Output: All_Itemsets
1 sacnDB;
2 Final_Binary_Tree FBT;
3 Temp_Binary_Tree TBT;
4 n=0 , m=0;
5 forall transcation t contains DB do begin
6         analysisElements(transcation t);
7         forall item i contains TBT do
8         if (i contains FBT)
9              i. count++;
10         else
11              add i to FBT;
12              i. count=1;
13 end
14 clear TBT , n=0 , m=0;
15 end
```

（行1）扫瞄整个数据库，取得所有的交易记录。（行2～4）设定初值，$n=0$、$m=0$、TBT 为暂存的二叉树，初值为 null，FBT 为存终值的二叉树，初值为 null。（行5～15）为算法主程序，其被执行的次数等于资料中交易记录的笔数。（行6）将读取到的交易记录传入拆分程序，进行拆分动作。（行7～13）将 TBT 里的项目集组合，存到 FBT 中计数，若此项目集已存在 FBT 中，则将该项目集之值加1；不存在则将此项目集加入，并给予初值1。（行14）清空初值 $n=0$、$m=0$ 以及 TBT 设为 null，以供下笔交易记录拆分用。拆分部分的程序如下：

```
Input: transaction t
Output: t_all_Itemsets
16 if (t contains TBT) then
17 getNextElement();
18 else
19 add t to TBT;
20 n++;
21 getNextElement();
22 endif
23 if (m <> n) then
24 tm= TBT. getitem[m];
25 m++;
26 if (tm[elements] == 1) then
27     getNextElement();
28 else
29       for(int i=1 ; i<=tm[elements] ; i++)
30            analysisElements(tm[elements]-1);
31       end
32 endif
```

（行16～35）拆分交易记录；（行16～22）判断记录（项目集）是否包含在 TBT 中，已存在则提取下一项欲拆分的项目集。如不存在则新增至 TBT 中，同样的再提取下一项欲拆分的项目集；（行23～35）分解项目集，当 m≠n 表示还有欲拆分的项目集。读取到欲拆分的项目集后，若项目集长度大于1，则继续向下拆分此项目集。若项目集长度为1，就不必再拆分，直接提取下一个欲拆分的项目集。该步骤一直执行到没有任何可拆分的项目集为止。

该算法的时间复杂度为 $O(n*\lceil m_1+m_2\rceil)$，其中 n 表示为数据库的总笔数；m_1 为单笔交易记录拆分时可能产生的项目集组合数；m_2 则为储存所有项目集组合所需的次数。

4 仿真实验结果分析

本文实验数据库是由 IBM generator 产生。Apriori 算法和改进的算法除了使用相同数据库做测试外，其数据结构存放的方式都是二叉树。如此，才能使得两个算法可以在公平的情况下，进行效率的比对。产生测试数据库参数如表1所示。

表1 定义的参数

D	数据库的总交易量（笔数）
L	数据库中单笔交易记录最大的项目数
I	产生的频繁项目集平均最大的项目数
N	交易数据库中项目的总数

本文用来实验的数据库有6个，数据库笔数（D）介于10K至100K，包含的项目个数（N）介于300到500之间，交易记录长度（L）最长均为10个项目，平均最大频繁项目集的项目数（I）为4，数据库如表2所示。

表2 实验数据库内容

数据库名	L	N	I	D
L10N500I4D100K	10	500	4	100k
L10N400I4D100K	10	400	4	100k
L10N300I4D100K	10	300	4	100k

测试 D100K 的实验数据库，数据库中交易记录最长为10，平均频繁项目集为4，总项目个数为 N300、N400、N500，数据库总笔数为100K，以改进的算法及 Apriori 算法作比较，

经过多次的实验产生图 1～3 的实验结果。

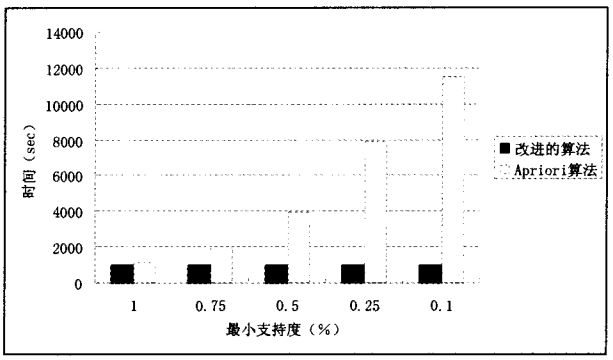

图 1　L10N500I4D100K

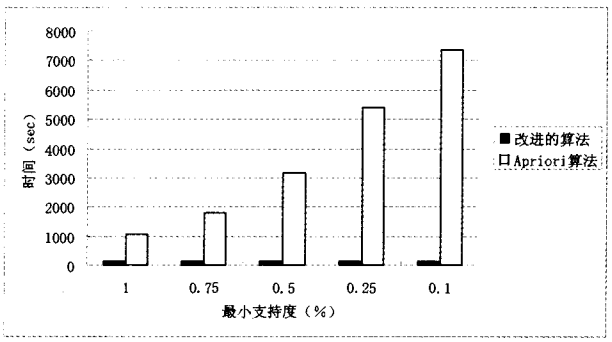

图 2　L10N400I4D100k

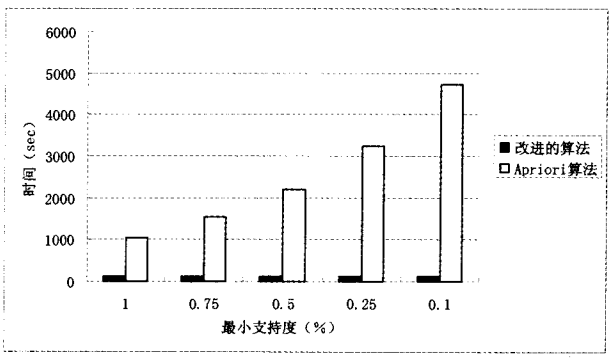

图 3　L10N300I4D100K

实验用的三个数据库 L10N500I4D100K、L10N400I4D100K、L10N300I4D100K，最长交易记录均为 10，总项目数 N500、N400、N300。由图 1～3 可知，Apriori 算法当最小支持度有所变动时，都必须重新扫瞄整个数据库。而且当最小支持度越来越低时，所产生的频繁项目集的数量随之增加，需要多次重复扫瞄数据库，而影响整体的执行效率。而本文所提出的改进算法不管是在任何最小支持度下，都只须扫瞄数据库一次，所以其执行非常平稳，而且执行效率相当优越。

结束语　一般而言，由于数据挖掘所产生的知识（如规则等）数量太过于庞大，导致使用者容易在庞大的知识库中迷失，因而无法发觉真正有用的知识。而且一般用来做数据挖掘的数据库都非常庞大，在进行数据挖掘时，需要用很长时间来扫描数据库，尤其是利用 Apriori 算法，更会使数据挖掘的效率变差。Apriori 算法最大的缺点就是每个阶段所产生的候选项目集，皆必须扫瞄数据库的原始交易资料记录来计算候选项目集的支持度，使得执行时间相当长、效率十分低下。而且当支持度改变时，也必须重新执行 Apriori 算法来产生符合条件的频繁项目集。因此，针对此缺点，本文提出了一个改进的算法，其主要是对交易记录进行拆分，跟 Apriori 算法的渐进式简单组合的架构大不相同，而且整个过程只须扫瞄数据库一次，就可获得所有可能的项目集组合，大大地降低 I/O 存取的时间。最重要的是，使用者可动态地随意意更新最小支持度，而这个过程不须再重新扫描整个数据库，所以使用者可实时、有效地得到所需的信息。

参 考 文 献

1　Agrawal R, Srikant R. Fast Algorithms for Mining Association Rules. In: Proc. of the 20th VLDB Conference Santiago, 1994

2　Park J S, Ming-Syan C, Philip S Y. An Effective Hash Based Algorithm for Mining Association Rules. In: Proc. of ACM SGMOD, 1995. 175～185

3　Savasere S, Omiecinski E, Navathe S. An Efficient Algorithm for Mining Association Rules in Large Databases. In: Proc. of 21st VLDB, 1995. 432～444

4　Brin S, Motwai R, Ullman J D, Tsur S. Dynamic Itemset Counting and Implication Rules for Market Basket Data. In: ACM SIGMOD Conference on Management of Data, 1997. 265～276

5　Dunkel B, Soparkar N. Data Organization and Access for Efficient Data Mining. ICDE, 1999

6　Han J, Fu. Yongjian. Mining Multiple-Level Association Rules in Large Database. IEEE Trans. on Knowledge and Data Engineering, 1999, 11(5): 798～805

7　张梅峰，张建伟. 基于 Apriori 的有效关联规则挖掘算法的研究. 计算机工程与应用，2003，39(19)：196～198

8　刘君强，孙晓莹，潘云鹤. 关联规则挖掘技术研究的新进展. 计算机科学，2004，31(1)：110～113