一种优化的 Apriori 算法

An Optimized Apriori Algorithm Based on the Business Address Index Table

文　蓉　（湖南大学软件学院 长沙 410082）
（湖南财经高等专科学校 长沙 410205）
李仁发　（湖南大学计算机与通信学院 长沙 410082）

摘　要： 深入研究 Apriori 算法，针对 Apriori 算法的性能瓶颈，以 Apriori 算法的运行事实为前提，给出了约简事务数据库中事务记录的理论，提出了一种利用事务地址索引表来有效约简事务数据库中事务记录的 Apriori 优化算法，以提高 Apriori 算法的执行效率。

关键词： 关联规则 Apriori 算法 事务地址索引表 约简事务

关联规则挖掘是数据挖掘领域研究的一个重要分支。关联规则挖掘一般应用在事物数据库 D 中，用一连串的“如果——则”的逻辑规则来描述一个事物中某些属性同时出现的规律和模式，从而发现大量数据中项集之间有趣的关联或相关联系。它最典型的应用是在销售事务数据库中发现商品销售中顾客的购买模式，因而在购物篮分析等商务决策中得到了广泛应用。由于事务数据库通常是相当庞大的，因此需要高效的算法来挖掘关联规则。

```
input: 事务数据库D ;  最小支持度阈值min_sup ;
output: D中的频繁项集L。
L1 = find_frequent_1-itemsets(D);
for ( k = 2; Lk-1 ≠Φ; k++ ) {
  Ck = apriori_gen (Lk-1, min_sup) ;
  for each transaction t ∈D {
    Ct = subset ( Ck , t ) ;
    for each candidate c ∈Ct
      c.count++ ;
  }
  Lk = {c ∈Ck|c.count≥min_sup}
}
return L = ∪kLk ;
```

图 1　Apriori 算法

1　Apriori 算法[1]

1.1　Apriori 算法的描述

Agrawal 等人在 1993 年提出的 Apriori 算法[1]是一种最有影响的挖掘关联规则频繁项集的算法，能通过频繁项集产生强的关联规则，在寻找频繁项集时必须满足 Aprior 性质——频繁项集的所有非空子集都必须也是频繁的。它利用频繁项集性质的先验知识，使用逐层搜索的迭代方法：K－项集用于搜索（K＋1）－项集。首先，找出频繁 1－项集的集合。该集合记作 L_1，它用于找频繁 2－项集的集合 L_2，而 L2 用于找 L_3，如此下去，直到不能找到频繁 K－项集。找每个 LK 需要扫描数据库一次。Apriori 算法主要是在遍历的基础上进行关联规则的挖掘。其具体算法如图 1 所示描述如下：令 K－属性序列集为具有 K 个属性的集合，L_K 为频繁 k－属性序列集，而 C_K 为候选 K－属性序列集。

算法中 apriori_gen（）函数产生候选，做两个动作：连接和剪枝。在连接部分，Lk－1 与 Lk－1 连接产生可能的候选。剪枝部分使用 Apriori 性质删除具有非频繁子集的候选。Subset（）函数用来找出事务中是候选的所有子集，并对每个这样的候选累加计数计算支持度。最后，所有满足最小支持度的候选集合形成频繁项集 L，然后由频繁项集产生关联规则。

使用 Apriori 算法进行关联规则挖掘时主要分为两个步骤:第一步,是从数据库或数据仓库中寻找所有的频繁项集;第二步,是由频繁项集产生关联规则。这两步中,第二步较容易,但挖掘关联规则的总体性能由第一步决定,目前大部分研究集中在第一个问题上。

```
input 事务数据库 D
output 事务数据链表 newdlist
D1=D;
maxtid=0;
while not empty(D1)
{ creat data;  //创建空 data 链表
  for each transaction de D1     //事务数据库中的任一事务记录
     data=d      //把事务记录中的 itemset 内容放入 data 链表中
  creat newdlist;  //创建空 newdlist 链表
  newdlist.itemset=&data; //把 data 的头结点地址赋给 newdlist 的 itemset 域
  newdlist.itemcount=count(data.item)
   D1=D1-d;
   maxtid++;}
 sort(newdlist.itemsetcount);
```

图 2　建立事务数据库算法

```
creat array addressindex[maxitemset]; //创建数组类型的地址索引表,maxitemset 是
                                      最大项目种类数
address=&(newdlist.1)//给索引表赋初值,使得 addrss 数组元素的所有值为
                      newdlist 中第一个结点的地址
k=1
     for (i=1; i<maxtid;i++)
      {if (k=newdlist.i.itemcount)
          continue;
        else
          { k=newdlist.i.itemcount;
            address[k]=&(newdlist.i);}
      }
     flag=0;
     for (i=1; i<maxtid;i++)
     { if address[i]=& (newdlist.1)
          {  if flag=1
               address[i]=address[i+1];}
       else flag=1 }
```

图 3　创建地址索引表算法

1.2 Apriori 算法的性能瓶颈问题

Apriori 算法作为经典的关联规则挖掘算法,在数据挖掘中具有里程碑的作用,但 Apriori 算法本身的执行效率并不十分理想,特别是对于大型数据库进行操作时,此问题更加突出,并且系统的 I/O 开销也很大,存在两大性能瓶颈。

首先,它可能产生大量的候选项目集,并呈现组合式的增长速度。造成这种情况的主要原因是在每一步产生候选项目集时循环产生的组合过多,没有排除不应该参与组合的元素。

其次,在每次计算子项集的支持度时,都需从上至下依次遍历事务数据库 D 中的各个事务记录,进行一遍全部的扫描比较,通过这种模式匹配检查一个很大的候选集合,它就需要重复地扫描数据库 D,这种扫描会大大增加系统的 I/O 开销。

2 基于事务地址索引表来约简事务的 Apriori 优化算法

针对上述问题,为提高 Apriori 算法的性能,现针对第二个瓶颈问题,使用一个有效约简事务数据库中事务的策略对算法进行优化。

目前通过约简事务数据库的中事务的策略对 Apriori 算法进行优化的研究已经取得了一些成果。如文[2]、文[3]、文[4]中提出的改进方法都是基于约简事务数据库中事务的可行、有效策略。通过这些策略虽能有效地减少事务数据库中一定的事务记录,但还存在一些问题:第一方面,在算法执行过程中存在裁减事务记录不及时的缺陷;第二方面,对事务数据库中事务集的减小操作,一般都是采用直接删除或是标上删除标记的方法,而实际上,在算法中不管采用以上的

哪种方法对事务数据库中事务集进行减少操作时都要通过依次遍历事务数据库中的事务记录的方法来完成，这样既不能实现查找的快速定位，又增加了扫描数据库的次数，使得算法的复杂度，时空开销也就增大。

Apriori 算法的中大型事务数据库中的数据量是巨大的，要进一步提高 Apriori 算法的执行效率，最大程度地优化 Apriori 算法，除了针对 Apriori 算法本身的瓶颈问题利用一些策略进行改进优化之外，对事物数据库的处理也是个不可忽视的问题。

```
input:事务数据 newdlist; 最小支持度阈值min_sup。
output:newdlist中的频繁项集L。
    C1={find candidate 1-itemsets(newdlist)}.//候选1-项集的集合
      for(k=1;Ck≠∅;k++){
      Ck =aproiri_gen(Lk-1 ,min_sup);
       scan newdlist start from address[k];//对候选项集进行记数时，扫描事务
                                  数据库直接从newdlist itemcount=k的事务记录开始
       for each transactions d∈ newdlist{
           Cd =subset(Ck,d);;      //计算包含在d 的候选项集集合
           for each candidates c∈Cd do
               c.count++}
       Lk ={ c∈Ck |c.count≥min_snp};}
     return L=UkLk;
```

图 4　约简事务算法

2.1 Apriori 算法的优化策略

2.1.1 针对算法中对 Ck 进行支持度计数步骤，提出约简事务策略

多情况下，Apriori 性质已经大幅度压缩了候选项集，也因此提高了效率。然而，通过对 Apriori 算法的实际运行的分析，可以看出，在对每一个 C_k 进行支持度计数时，均对数据库扫描一次，而这时有些事务记录已对频繁项集的生成不产生任何作用。减少数据库 D 内不起作用的事务记录对于算法来说很有必要。

约简事务算法的思想根据关联规则的概念和 Apriori 的性质，基于以下推导的理论进行对 Apriori 算法进行优化。具体内容如下：

定义 1 设 $I=\{i_1,i_2,i_3,\cdots,i_n\}$ 为项目集；$D=\{d_1,d_2,d_3,\cdots,d_n\}$ 为事物数据库；$d_i\subseteq I$；$|di|$ 为对应事务项目集中事物项目元素的总数。

结论 1 对于已知规模的事务数据库 D，$d_i\subseteq I$；C_k 为 k－项集，对 C_k 中任意一个子项集 C_k^i 在 D 中出现的支持频繁度的计算与 $|d_i|<k$ 的事务 d_i 无关，所以在第 k 次扫描事务数据库 D 前，可忽略该事务 d_i。

证明：因为 C_k 为 k－项集，即对 C_k 中任意一个子项集 C_k^i，其项集中的项目元素个数等于 k，在计算 C_k^i 的支持度时，扫描到任一 $|d_i|<k$ 的事务记录，由于其元素个数小于 C_k^i 中的元素个数，此事务记录存在与否都不会对支持度的计数结果产生任何效果，即 C_k^i 在 D 中出现的支持频繁度的计算与 $|d_i|<k$ 的事务 d_i 无关。

利用以上定义及结论，对算法中 C_k 进行支持度计数步骤进行优化，即在计算任一部分 C_k 支持度前，完全忽略掉事务数据库 D 中 $|d_i|<k$ 的事务 d_i，随着 k 的增大，忽略的事务也不断增大，避免了对无关事务的重复扫描，缩小了待扫描的事务数据库的大小，因而有效提高每部分候选项集 C_k 的计数速度。

2.1.2 基于事务地址索引表有序化事务数据库，提高约简事务操作的效率

Apriori 算法在计算项集的支持度时，需要访问事务数据库。事务数据库的表示方法直接影响 C_k 进行支持度计数的效率。现利用链表插入、删除、修改效率高的特性，对于已知规模的事务数据库 D 进行预处理，使得事务数据库 D 按照事务项目值有序。比较事务数据库中每个具体事务记录的事务项目值，获取事务数据库中第一条具有不同事务项目值的事务记录地址，并放入事务地址索引表中。这样就能在约简事务的算法执行过程中避免采用直接删除或是标上删除标记的操作方法，而是利用事务地址索引表中记录的相关地址，直接跳过约简的事务对象，在事务数据库有序的前提下，对事务数据库进行分区域的快速定位，避免了因删除或是标上删除标记操作而引起多次扫描事务数据库，降低计算项集的支持度时遍历节点的时间，使约简事务操作的效率在整体上更优。

（下转第 120 页）

(上接第96页)

2.2 Apriori 优化算法

(1) 预处理事务数据库

(2) 创建地址索引表

(3) 约简事务

该约简算法中 apriori_gen( ) 函数以及 Subset( ) 函数的含义及具体算法与 Apriori 算法一致。

3 结束语

Apriori 算法[1]是一种目前最具影响且经典的关联规则算法。但 Apriori 算法本身的执行效率并不十分理想,特别是对于大型数据库进行操作时,此问题更加突出。现通过对 Apriori 算法的挖掘过程进行分析,研究了事务数据库中事务的特性,提出了一个利用事务地址索引表来有效约简事务数据库中事务的 Apriori 优化算法。算法中给出了一个有效约简事务的方法,通过创建事务地址索引表缩小了对事务数据库记录的搜索范围,实现了分区域的快速定位,从而减少解空间,加快了候选集的计数过程,提高了 Apriori 算法的执行效率。当然算法还存在一定的缺陷,那就是对事务数据库仍需多次扫描,针对这个问题寻求更优的改进方法是下一步的研究重点。

参考文献

1 KAMBER M, HAN J. 数据挖掘概念与技术[M],范明、孟小峰 等译,北京:机械工业出版社,2001.

2 黄艳、王延章、苑森淼,一种高效相联规则提取算法[J],吉林大学自然科学学报,1999,4(2).

3 马盈仓,挖掘关联规则中 Apriori 算法的改进[J],计算机应用与软件,2004,21(11).

4 张瑞雪、钱真,哈尔滨工程大学,[学士论文]2006,10.