第23卷第4期
2002年1月

小型微型计算机系统
MINI-MICRO SYSTEM

Vol. 23 No. 4
Apr. 2002

文章编号:1000-1220(2002)04-0450-03

基于关联图的关联规则挖掘算法研究

蔡之华　吕维先　颜雪松

(中国地质大学 信息工程学院,湖北 武汉 430074)

摘　要:在挖掘关联规则的过程中,一个关键的步骤是产生频繁项目集.本文给出一种基于关联图的关联规则挖掘算法,并将它与性能比较好的关联规则挖掘算法DHP进行了比较,结果表明,本文的算法优于DHP算法.

关键词:数据挖掘;关联规则;关联图

中图分类号:TP311.13　　**文献标识码**:A

1 引　言

近年来,随着计算机技术的不断发展,人类积累的数据量正在以指数速度增长,如何从这些数据中发现有用的或有趣的模式是人们关注的课题.例如,保险公司对从数据库中发现可能是欺诈行为的模式很感兴趣,超市经理对商品在不同季节或一天的不同时间的销售量有何变化规律很感兴趣.现今的状况是数据太多而知识不够.面对浩渺无际的数据,如何从中去粗取精,去伪存真,具有很强的实用价值.数据挖掘正是在这种背景下而出现的.它是数据库研究的一个很有发展前景的新领域.

关联规则是KDD(*Knowledge Discovery in Database*)研究中的一个重要课题,典型的例子是对商品销售数据中的规律发现.例如,90%的顾客在购买商品A和B的同时也购买了商品C,这样的消费模式对顾客的分类、商品的采购、商场中商品的摆放都具有指导意义.

关联规则的形式定义为:

设$I=\{i_1,i_2,\ldots,i_m\}$是文字集合,称为项目集,D是事务集,每个事务T是一个项目子集($T\subseteq I$),它有唯一一个标识符TID.设X是I中一些项目的集合,若$X\subseteq T$,就说T包含X.关联规则是形如$X\rightarrow Y$的逻辑蕴含式,其中$X\subset I$,$Y\subset I$,且$X\cap Y=\Phi$.如果D中$c\%$的事务包含X且同时包含Y,就说$X\rightarrow Y$的置信度为c.如果D中$s\%$的事务包含$X\cup Y$,就说规则$X\rightarrow Y$的支撑度为s.给定一个事务集D,关联规则的挖掘即是产生所有其支撑度和置信度大于用户事先给定的最小支撑和最小置信的规则.由此,我们得知挖掘关联规则分为两步,第一步,找出所有的支撑大于最小支撑的项目集,一个项目集的支撑即为包含项目集的事务项目,具有最小支撑的项目集称为频繁项目集.第二步,用频繁项目集产生合适的规则.

目前有很多学者提出了发现频繁项目集的算法[1,2,3,4],这些算法产生频繁k-项目集时,扫描数据库的每个事务用以统计这些候选K-项目集的支撑,并按照事务确定的最小支撑在第K次迭代时找出所有频繁k-项目集.然而,由于数据库的规模通常是非常大的,所以在每次迭代时产生候选项目集以统计其支撑是非常耗时的.因此,改进频繁项目的发现过程是问题的关键.

本文给出一个有效的频繁项目集产生的算法,它不需要产生候选集,且只需扫描数据库一次,所以此算法是非常有效的.此算法构造一个关联图以展现项目集之间的关系,然后再遍历关联图以产生频繁集.

2 相关工作

最早开展关联规则挖掘研究的是美国IBM Almaden Research Center的Rabesh Agrawal领导的研究组,他们给出了一个称为AIS的算法[2],AIS在每次迭代时扫描数据库以产生候选项目集且统计其支撑,在读入一个事务后,可以确定上一次迭代中发现的频繁项目集是否包含在这个事务中.在扫描数据期间,通过扩展这些频繁集与事务集中另外的项目,以产生新的候选项目集,并且收集支撑信息以评估哪些候选项是真正频繁的.然而,人们发现用AIS时会产生很多的候选项,因此,AIS是非常低效的.

算法Apriori和Apriori Tid[3]通过只需使用前一次迭代产生的候选项来产生候选项目集,而不考虑数据库中的事务,这个结果会产生较小的候选项目集,然而,对每个候选项来说,需要统计它在所有事务中的出现.在Apriori算法中,每次迭代需要对数据库一次扫描,在Apriori Tid算法中,数据库只须扫描一次,代之的是每次迭代产生出现在每个事务中的事务标识和候选k-项目集,它被用来统计下一次迭代过程中k+1项目集候选项的支撑.在初始阶段,Apriori比Apriori Tid有效,因为有太多的k-项目候选项需要在这个过程的早期阶段被处理.[4]中还给出了这两种算法的混合算法.且一般情况下,它的性能较好.

Park等在[1]中指出改进频繁项目集发现的性能是初始候选集的产生,特别是2-项目的产生及每次迭代时以产生频繁项目集不得不扫描大量的数据.他们利用Hash方法来减少2-项目集的产生,而且使用剪枝技术来修剪事务数据库.所谓剪枝技术即为:对于一个事务,如果它不出现在第k次迭

收稿日期:2000-11-11　基金项目:湖北省自然科学基金项目(编号2001ABB006)　作者简介:蔡之华,副教授,在职博士生,主要从事并行计算、数据挖掘方面的研究.吕维先,教授,主要从事计算机应用研究.颜雪松,硕士研究生,主要从事数据挖掘方面的研究.

代的候选 k-项目集,则一个事务的一个项目可被剪枝.然而,为了减少 2-项目集候选者的数目,构造 hash 表的开销较大,而且为了剪裁数据库,要扫描数据库中的每个事务以确定项目集在事务中包括那些候选项.[1]中给出了具有较好性能的频繁项目产生的算法 DHP,所以 DHP 被用来作为基本算法,以与本文中的算法进行比较.

3　基于关联图的关联规则挖掘算法

下面我们给出有效的频繁项目集产生的有效算法 DLG,DLG 共有三步.第一步是产生频繁 1-项目集阶段:它产生频繁 1-项目集并记录相关信息;第二步是关联图的构造,用以展示频繁项目之间的关联;最后一步是频繁 k-项目集(k>2)产生阶段,它基于关联图的构造来产生 k 频繁项目集.

3.1　关联图构造

执行 DLG 算法之前,给每个项目赋一个整数.第一步,DLG 扫描一遍数据以统计其支撑,且为每个项目建立一个比特向量,每个比特向量的长度是数据库中事务的数目.如果一个项目出现在第 i 个事务中,则与这个项目相关联的比特向量的第 i 个比特置 1,否则,比特向量的第 i 个比特置为 0.与项目 i 相关联的比特向量表示为 BV_i,BV_i 中"1"的数目等于支撑项目 i 的事务数,即项目 i 的支撑.

Tid	Itemset
100	A,C,D
200	B,C,E
300	A,B,C,E
400	B,E

图 1　事务数据库

例如,考虑图 1 中的数据库,每个记录是(TID Itemset)对,这里 Tid 是相关事务的标识符,而 Itemset 记录交易中的项目,对每个项目按字典顺序排列,并赋以一个顺序值,如 A 赋值 1,B 赋值 2,……设最小支撑是 2,在频繁 1-项目集产生阶段,则从图中可得到频繁项目为 A,B,C,E,且 BV_1,BV_2,BV_3 和 BV_5 分别是(1010),(0111),(1110),(0111).

性质:项目集$\{i_1,i_2,\ldots,i_k\}$的支撑是 $BV_{i1} \wedge BV_{i2} \ldots \wedge BV_{ik}$ 中的"1"的数目,这里符号"∧"是逻辑"与"运算.

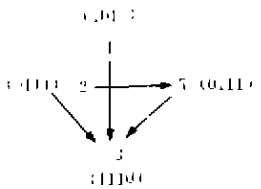

图 2　对应于图 1 的关联图

在经过第一步后,不必再扫描数据库.在关联构图造阶段,DLG 构造一个关联图以表示项目间的关联.对于关联图来说,如果 $BV_i \wedge BV_j (i<j)$ 中"1"的数目不小于最小支撑,则构造一条从项目 i 到项目 j 的有向边,而且项目集{i,j}是一个频繁 2-项目集.上述例子的关联图见图 2,且频繁 2-项目集是{1,3},{2,3},{2,5}和{3,5}.

3.2　频繁项目集的产生

基于关联图我们可以产生频繁 k(k>2)项目集.实现关联图的数据结构是链表.在关联图构造阶段,可以产生频繁 2-项目集 L_2,在频繁项目集产生阶段,DLG 算法产生频繁 k(k>2)项目集 L_k.对每一个 L_k(k>2)中的频繁 k-项目集,k-项目集的最后项用来扩展项目集为 k+1 项目集.设 $(i_1,i_2,\ldots,i_k)$是一个频繁 k-项目集,如果有一个从项目 i_1 到项目 u 的有向边,那么项目集$\{i_1,i_2,\ldots,i_k\}$被扩展为 k+1 项目集.如果 $BV_{i1} \wedge BV_{i2} \wedge \ldots \wedge BV_{ik} \wedge BV_u$ 中"1"的数目不小于最小支撑,那么$\{i_1,i_2,\ldots,i_k\}$是频繁 k+1 项目集.如果再不能继续产生频率 k-项目集,则 DLG 算法终止.

```
/* 基于关联图的关联规则挖掘算法 */
/* 频繁 1-项目集产生阶段 */
for all items i do
    set all bits of BV_i to 0;
    for (j=1;j<=N;j++) do begin /* N 是数据库
                                   中的事务数目 */
      for all items i in the jth transaction do begin
          i.count++;
          set the jth bit of BV_i to 1;
      end
    end
    L_1=C
    for all items i in database D do begin
      If i.count>=minsup then /* minsup 是最小支
                                 撑阈值 */
      L_1=L_1∪{i};
    end
 /* 关联图构造阶段 */
 if L_1≠C then begin
 for all frequent 1-itemsets l∈L_1 do
 allocate a node for Item [l] and
 Item [l].link=NULL;
 L_2=C
for every two frequent items i,j(i<j) do begin
 if (the number of 1's in BV_i∧BV_j)>=minsup
  Then begin
        /* 在关联图中产生一条从 i 到 j 的有向边 */
        from i to j in the association graph
         allocate a node p;
         p.link=Item [la].link;
         p.Item=lb;
         Item [la].link=p;
      L_2=L_2∪{(i,j)};
      /* 产生频繁 2-项目集 */
       end
      end
```

```
end
/* 频繁项目集产生阶段 */
    k=2;
    while L_k≠∅ do begin
L_k = ∅
for all itemsets(i_1,i_2,...,i_k)∈L_k do begin
    pointer=Item[i_1].link;
    while pointer≠NULL do begin
    u=pointer.Item;
    if number of 1's in BV_{i_1} ∧ ... BV_{i_k} ∧ BV_u >=min
                                                sup then
    L_{k+1} = L_{k+1} ∪ (i_1,...,i_k,u);
    pointer = pointer.link;
    end
  end
  k=k+1
  end
```

例如，考虑图 1 中的数值，在第二阶段，可产生频繁 2-项目集 $L_2=\{\{1,3\},\{2,3\},\{2,5\},\{3,5\}\}$，对于频繁 2-项目集 $\{2,3\}$ 来说，有一个从项目集{2,3}中最后项目 3 到项目 5 的有向边，因此，2-项目集{2,3}扩展为 3-项目集{2,3,5}，$BV_2 \wedge BV_3 \wedge BV_5$ 中的"1"的数目（即(0110)为 2），既然它的比特向量中"1"的数目不少于最小支撑，所以 3-项目集{2,3,5}是一个频繁 3-项目集。因为不能继续生成频繁 4-项目集，所以 DLG 算法终止。

4 性能比较和分析

设数据库 DB 中有 |D| 个事务，而每个事务平均有 m 个项，在第 k 阶段，产生频繁 k-项目集 Lk. 对于第一阶段，DLG 和 DHP 都需要扫描数据库 DB 的每一个事务以统计每一个项目的支撑，然而 DHP 需要额外的开销以结合每二个项目以便在每一次事务中形成 2-项目集，因此需要 $|D|\times C_m^2$ 次组合，而 DHP 利用 Hash 函数存 Hash 表中定位 2-项目集，因此 DHP 在第一阶段需要的时间比 DLG 多。

设在第 K 阶段产生 $|L_k|$ 个频繁项目集，在第二阶段，DLG 执行在比特向量上 $|L_1|\times(|L_1|-1)/2$ 次逻辑"与"运算以构造关联图和生成频繁 2-项目集。DHP 需要产生候选 2-项目集和用第一阶段生成的 Hash 表以修剪这些候选 2-项目集。另外，DHP 需要扫描数据库以统计候选 2-项目集的支撑和修剪数据库 DB，以生成一个缩减的数据库，DHP 需要做的工作比 DLG 在第二阶段执行的逻辑运算要耗时得多。

在第 k(k>2)阶段，DLG 依照关联图并执行逻辑"与"运算来扩展每一个频繁 k-1 项目集为 k-项目集。平均地，设关联图中每一个结点（项目）有 q 个输出，DLG 执行 $(k-1)\times|L_{k-1}|\times q$ 次逻辑"与"运算以发现所有频繁 K-项目集，因此，随着最小支撑减少，逻辑"与"运算由于值 L_{k-1} 和 q 的增加而增加。在第 k 阶段，DHP 从频繁 k-1 项目集 L_{k-1} 中产生候选 k-项目集 C_k，生成 C_k 后，DHP 扫描数据库 DB_k 以统计这些候选 k-项目集，并且修剪数据库 DB_k 以产生另一个缩减数据库 DB_{k+1}。因此，DHP 的执行时间依赖于产生候选项目的数量和必须扫描的数据的量。

5 结束语

本文我们研究了一种关联规则挖掘算法并将其与熟知的 DHP 算法进行了比较。当数据库的量很大时，用本文给出的算法不能将与关联图相关的信息一次装入内存，这是需要进一步研究的课题。本文的方法可以应用到文献检索和资源发现等领域。

参 考 文 献

1 J. S. park, M. S. Chen, P. S. Yu. An effiecntive Hash-based algorithm for mining association rules[C]. Processings of ACM SIGMOD, 1995 24(2):175~186

2 Maurice Houtsma Arun Swami. Set-oriented mining of association rules[C]. In int'l Conf On Data Engineering, Taibei, Taiwan, March 1995.

3 R. Agrawal, Tomasz Imielinski, Arun Swami. Mining association rules between sets of items in large databases[C]. In Proc. Washington, D. C. 0f the ACM SIGMOD Conference on Management of Data, may 1993 207~216.

4 R. Agrawal, ramakrishnan Srikant. Fast algorithms for mining association rules[C]. In Proc. 0f the 20th Int Conference on Very large databases, Santiago, Chile, Sept., 1994 487~499

Graph-Base Algorithm for Mining Association Rules

CAI Zhi-hua, LU Wei-xian, YAN Xue-song

(Department of Computer Science and Technology, China University of Geosciences, Wuhan 430074, China)

Abstract This paper presents a graph-base algorithm for mining association rule, the algorithm is compared with DHP, and show that the algorithm outperform DHP algorithm.

Key words data mining; association rules; association graph