Apriori-sort 算法研究

汪维清,罗先文,胡继宽
WANG Wei-qing,LUO Xian-wen,HU Ji-kuan

西南大学 信息管理系,重庆 402460
Department of Information Management,Southwest University,Chongqing 402460,China
E-mail:wwqlhy@163.com

WANG Wei-qing,LUO Xian-wen,HU Ji-kuan.Apriori-sort algorithm.Computer Engineering and Applications,2008,44(36):156-159.

Abstract: Although Apriori algorithm uses cut-technology when it generate item sets of candidates,it has to scan the entire database while scanning the transaction database each time.The scanning speed is very slow for its large amount of data.The Apriori-sort algorithm is improved from the Apriori algorithm.Its basic idea is to transform a transaction database into a transaction degree database,and sort the transaction-degree database.When the Apriori-sort algorithm searches the frequent item sets,it only scans the transactions in the database *Dd* that appeases $d(Ck)\leqq d(Ti)$.It scans the database with an effective cut.Therefore the Apriori-sort algorithm's computing time is very fast.The results of simulation experiments for the Apriori-sort algorithm and the Apriori algorithm show the new algorithm's efficiency.

Key words: sort;Apriori;association rule;market basket analysis;Knowledge Discovery in Database(KDD)

摘 要: Apriori 算法虽然在候选集的产生时利用了剪支技术,但每次扫描数据库时都必须扫描整个数据库,因此扫描的数据量大,速度较慢。Apriori-sort 算法是在 Apriori 算法基础上的改进,基本思想是把事务数据库变为以度表示的事务度数据库,并对事务度数据库进行排序。Apriori-sort 算法查找频繁项集时,只扫描数据库 *Dd* 中满足 $d(Ck)\leqq d(Ti)$的事务。对扫描数据库进行了有效剪支,因此 Apriori-sort 算法的计算效率高。并用仿真数据对 Apriori-sort 算法和 Apriori 算法进行了仿真对比实验,实验结果证明了新算法的高效性。

关键词:排序;Aprior;关联规则;购物篮分析;数据库知识发现
DOI:10.3778/j.issn.1002-8331.2008.36.044 **文章编号:**1002-8331(2008)36-0156-04 **文献标识码:**A **中图分类号:**TP301

1 引言

关联规则挖掘是 KDD 的重要研究领域之一。Apriori[1]算法是一种有影响的挖掘关联规则的算法。Apriori 使用一种称为逐层搜索的迭代方法,并且把频繁项集的反单调性用于压缩搜索空间,从而达到快速搜索频繁项集的目的。DHP[2]算法采用 Hash 技术和数据库中的事务的缩减技术来提高 Apriori 算法的效率。TreeProjection[3]算法通过逐渐建立字典顺序的项集来产生频繁项集。关联规则挖掘 AprioriTid 优化算法在 AprioriTid 算法[4]基础上进行下列优化:(1)事务压缩和项目压缩;(2)候选项目集关键字识别。改进购物篮分析的关联规则挖掘算法[5]则在数据处理时引入兴趣度加权。关联规则挖掘的矩阵算法[6]是通过构造有效的数据矩阵并且通过对数据矩阵的有效裁剪,达到查找频繁项集的目的,极大地减少了高次频繁项集的查找时间。等价关系的关联规则挖掘算法[7]是基于等价关系和等价类来生成候选频繁项目集,它可以减少系统的开销。文献[8]提出的 Apriori 算法是最有影响的挖掘布尔型频繁项目集的算法。

Apriori 算法虽然在候选集的产生时利用了剪支技术,但每次扫描数据库时都必须扫描整个数据库,因此扫描的数据量大,速度较慢,针对 Apriori 算法的这一缺点提出了一种新的 Apriori-sort 算法。

2 Apriori-sort 算法原理

2.1 相关定义

定义 1 事务。商品的一次交易就是一个事务,事务可用一个布尔向量表示,即令事务 $I=[I1,I2,\cdots,In]$是 n 个不同项目的集合(Item set)。

定义 2 事务数据库 D。事务 $Ti(i=1,2,3,\cdots,n)$的集合,在事务数据库 D 中,事务 T 可表示为$\{Ti,<I1,I2,I3,\cdots,Im>\}$,其中 Ti 为事务标识,每一个事务有一个在数据库 D 中全局唯一标识;$I1,I2,\cdots,Ii,\cdots,Im\in I,1\leqslant i\leqslant m\leqslant n$,即每个事务 T 是项目的集合,使得 $T\subseteq I$。设 A 是一个项目集,事务 T 包含 A 当且仅当 $A\subseteq T$。

基金项目:西南大学荣昌校区科研项目(No.2007S10)。
作者简介:汪维清(1969-),男,高级工程师,主要研究领域为地理信息系统、图论、人工智能;罗先文,男,副教授,主要研究领域为数据挖掘、网格计算;胡继宽,男,讲师,主要研究领域为数据挖掘、数据仓库。
收稿日期:2008-07-31 **修回日期:**2008-10-20

表 1 为一简化的事务数据库 D。其中 $T001$ 为事务数据库的首记录标号,$T001$ 相对应的交易包括了 $I1$、$I2$、$I5$,记录中的 item 按标号的顺序排列。

定义 3 关联规则。一个形如 $A=>B$ 的逻辑蕴涵式,其中 $A\subset I, B\subset I$ 且 $A\cap B=\Phi$。若数据库 D 中有支持度 S 的事务包含 $A\cup B$,则关联规则 $A=>B$。

定义 4 支持度 $S(Support)$。$Support(A=>B)=P(A\cup B)=S$,其中 S 是 D 中事务包含 $A\cup B$(即 A 和 B 二者)的百分比,是概率。

定义 5 置信度 $C(Confidence)$。$Confidence(A=>B)=P(A|B)=Support(A=>B)/Support(A)=C$,即在 D 中,C 是在事务中包含 A 同时也包含 B 的百分比,$P(B|A)$ 是条件概率。

定义 6 最大项集 P。在事务数据库 D 中,事务 $Ti(i=1,2,3,\cdots,n)$ 项集中的最大集合 P,即 $P=\{Pm|m=1,2,3,\cdots$,并且 $Pm\in Ti$,其中 $i=1,2,3,\cdots,n\}$。

定义 7 事务的度。设事务数据库 D 的最大项集为 $P=[I1,I2,I3,\cdots,In]$,令 P 中项 $Ij(j=1,2,\cdots,n)$ 对应的度为十进制数 2^{j-1},对任意事务 $Tk=[Ik1,Ik2,Ik3,\cdots,Ikm]$,令 Tk 中项 $It(t=k1,k2,\cdots,km)$ 在 P 中对应项的位置为 u,则事务 Tk 的度为 $d(Tk)=\sum 2^{u-1}$。

如表 1 数据库 D 的最大项集为 $P=[I1,I2,I5]$,事务 $T001$ 的度为 $d(T001)=2^{1-1}+2^{2-1}+2^{5-1}=1+2+16=19$。

2.2 相关定理

定理 1 事务数据库 D 中的任意事务 $Tk(k=1,2,\cdots,n)$ 对应的度为 $d(Tk)$,对任意项集 Ts,且 Ts 的度为 $d(Ts)$,事务数据库 D 包含项集 Ts 的任意事务 Tk 必满足 $d(Tk)\geqslant d(Ts)$。

证明 由定义 7,最大项集 $P=[I1,I2,I3,\cdots,In]$ 中的项对应的度分别为:

$$2^0,2^1,2^2,\cdots,2^n \tag{1}$$

则有 $\sum_{i=0}^{k-1}2^i<2^k$ (2)

由式(1)和式(2)可知:对任何事务(或项集)Tk,只要 $d(Tk)<2^u$,则

$$Iu\not\subset Tk \tag{3}$$

设任意项集 $Ts=[Is1,Is2,Is3,\cdots Ism]$ 最大项为 Ism,Ism 在 P 中对应的项为 It,由式(3)可知:若 Ism 在最大项集 P 中的位置小于 Ik 的位置,即 $Ik\not\subset Ts$,则有

$$d(Ts)=\sum 2^s<\sum 2^k=d(Tk)$$

所以,当 $T(s)\subset T(k)$ 时,满足 $d(Ts)\leqslant d(Tk)$。

证毕。

例 1 如表 1 数据库 D 的最大项集为 $P=[I1,I2,I3,I4,I5]$,事务 $T001$ 的度为 $d(T005)=4$。对于所有包含 $T005$ 的记录 $T008$ 和 $T009$ 满足 $d(T008)=23>4=d(T005)$,$d(T009)=7>4=d(T005)$。

定理 2 对事务数据库中的任意两事务 $Ti(i=1,2,3,\cdots,n)$ 和 $Tj(j=1,2,3,\cdots,n)$,如果 $d(Ti)\&d(Tj)=d(Ti)$,则 $Ti\subset Tj$;如果 $d(Ti)\&d(Tj)=d(Tj)$,则 $Tj\subset Ti$。

证明 由定义 7,最大项集 $P=[I1,I2,I3,\cdots,In]$ 中的项对应的度分别为:

$$2^0,2^1,2^2,\cdots,2^n \tag{4}$$

设 Ti 和 Tj 分别为:$Ti=[Ii1,Ii2,Ii3,\cdots,Iim]$,$Tj=[Ij1,Ij2,Ij3,\cdots,Ijk]$。$Ti$ 和 Tj 项在 P 中的位置分别为:$i1,i2,\cdots,im$ 和 $j1,j2,\cdots,jk$。

用布尔型来表示 Ti 为:$0,\cdots,1,\cdots,1,\cdots,1,\cdots,1$ (5)

项在 P 中对应的位置:$i1\cdots i2\cdots..i\cdots.im$

用布尔型来表示 Tj 为:$0,\cdots,1,\cdots,1,\cdots,1,\cdots,1$ (6)

项在 P 中对应的位置:$j1\cdots j2\cdots j3\cdots jk$

当 $Ti\subset Tj$ 时,由二进制"与"定义和式(5)、(6)得:

$Ti\&Tj=0,\cdots,1,\cdots,1,\cdots,1,\cdots,1=Ti$ (7)

项在 P 中对应的位置:$i1\cdots i2\cdots..i\cdots.im$

根据度的定义 7,式(7)可表示为:$d(Ti)\&d(Tj)=d(Ti)$

所以:如果 $d(Ti)\&d(Tj)=d(Ti)$,则 $Ti\subset Tj$。

同理可证:如果 $d(Ti)\&d(Tj)=d(Tj)$,则 $Tj\subset Ti$。

证毕。

例 2 如表 1 数据库 D 的最大项集为 $P=[I1,I2,I3,I4,I5]$,事务 $T001$ 和 $T007$ 的度分别为 $d(T001)=19$、$d(T007)=3$。因为 $d(T001)\&d(T007)=19\&3=3=d(T007)$,所以 $T007\subset T001$。

表 1　事务数据库

TID	List of item_ID's	TID	List of item_ID's
$T001$	$I1,I2,I5$	$T006$	$I2,I3$
$T002$	$I2,I4$	$T007$	$I1,I2$
$T003$	$I2,I3$	$T008$	$I1,I2,I3,I5$
$T004$	$I1,I2,I4$	$T009$	$I1,I2,I3$
$T005$	$I1,I3$		

2.3 算法的工作原理

Apriori 算法虽然在候选集的产生时利用了剪支,但每次扫描数据库时都必须扫描整个数据库,因此扫描的数据量大,速度较慢。Apriori-sort 算法是在 Apriori 算法基础上的改进,基本思想是基于对事务度数据库进行排序,每次数据库的扫描只扫描数据库 Dd 中满足 $d(Ck)\leqslant d(Ti)$ 的事务,对扫描数据库进行了有效剪支,因此扫描的数据量小,速度较快。对于有 n 个数据记录的事务数据库 D,其任意事务 $Tk=[Ik1,Ik2,Ik3,\cdots,Ikm]$ $(k=1,2,\cdots,n)$。则 Apriori-sort 算法核心分三步处理:

求最大项集:扫描整个事务数据库 D,找出事务 $Ti(i=1,2,3,\cdots,n)$ 项集中的最大集合 P,即 $P=\{Pm|m=1,2,3,\cdots$,并且 $Pm\in Ti$,其中 $i=1,2,3,\cdots,n\}$。

求数据库的度 d:根据最大集合 P,再次扫描数据库 D,求出事务 $Ti(i=1,2,3,\cdots,n)$ 的度 $d(Ti)$,并按 $d(Ti)$ 的大小升序重组事务度数据库 Dd。

扫描剪支数据库:根据候选项集 Ck 的度 $d(Ck)$,扫描数据库 Dd 中满足 $d(Ck)\leqslant d(Ti)$ 的事务。对扫描数据库进行了有效剪支。

3 Apriori-sort 算法

Apriori-sort 算法基本思想是基于对事务数据库进行排序。对于有 n 个数据记录的事务数据库 D,其任意事务 $Tk=[Ik1,Ik2,Ik3,\cdots,Ikm]$ $(k=1,2,\cdots,n)$,则 Apriori-sort 算法如下:

(1)procedure *AprioriSort*(D,n,*min_sup*) //Apriori-sort 算法,D 为事务数据库,n 为 D 中的记录数,*min_sup* 为最小支持度

(2)*sum*=*MaxItemSet*(D,p);//形成事务数据库 D 的最大项集 p,*sum* 为最大项集 p 的项目数量即$|p|$

(3)*Swap*(*pd*,*p*,*sum*);//求最大项集 *p* 的度(*d*),并把结果放在 *pd* 中,*sum* 为最大项集 *p* 的项目数量即|*p*|

(4)*SwapTi*(*db*,*pd*);//扫描事务数据库 *D*,并求 *D* 中每一个事务 *Ti*(*i*=1,2,…,*n*)的度 *d*(*Ti*),形成对应的以度表示的事务度数据库 *db*,*pd* 为最大项集中的项对应的度

(5)*sup*=*n***min_sup**100;//由最小支持度 *min_sup* 和 *D* 中的记录数 *n* 形成最小支持数

(6)*DbSort*(*dba*,*db*);//对事务度数据库 *db* 进行排序,排序后的结果放在有序事务度数据库 *dba* 中

(7)*L*1=*LargelItemsets*(*dba*,*n*,*pd*,*sum*,*sup*); //有序事务度数据库 *dba* 产生以度表示项集的频繁 1 项集 *L*1

(8)while (*Lk*−1<>Φ) do //频繁 *k*−1 项集 *Lk*−1 不为空,则计算候选集 *Ck* 和频繁 *k* 项集 *Lk*

begin

ck=*AprioriGen*(*Lk*−1,*pd*,*sum*);//产生新的以度表示项集的 *k* 项候选集 *ck*

for *all cki*∈*ck* do //对候选集 *ck* 中以度表示项集的 *cki*, 计算其支持数

begin

for all transaction *d*(*ti*)∈*dba* and *d*(*ti*)≧*cki* do //扫描度满足条件的事务,对扫描数据库进行剪支

begin

if(*d*(*ti*) & *cki*==*cki*);//如果 *d*(*ti*)和 *cki* 相“与”结果等于 *cki*,说明事务 *Ti*⊃*cki*

begin

cki.count++;//候选集 *ck* 中以度表示项集的 *cki* 的支持数加 1

end

end

if(*cki.count*≧*sup*) then //如果候选集 *ck* 中以度表示项集的 *cki* 的支持数大于最小支持数,说明 *cki* 是频繁 *k* 项集

begin

cki→*Lk*;//把该候选集加入频繁项集中

end

end

end

(9)*Answer*=*UkLk*;//输出所有频繁项集

上述 Apriori-sort 算法中的函数 *MaxItemSet*(*D*,*p*)为求事务数据库 *D* 的最大项集 *p* 的函数,它是算法的关键函数之一。

```
function MaxItemSet(D,p)
{
  p=null;sum=0;//初始化最大项集 p 为空,p 中的项目数 sum 为 0
  while(ReadString(D,str)<>null) do //扫描数据库 D 的项 str
  {
    if(str∉p)//如果 str 不是最大项集 p 中的项
        {p=p=str;sum++;}//把 str 放在 p 的最后项,且 p 中的项目数 sum 增 1
  }
  return sum;//返回 p 中的项目数 sum
}
```

Apriori-sort 算法中的函数 *Swap*(*pd*,*p*,*sum*)为求最大项集 *p* 的度(*d*), 并把结果放在以度表示的事务度数据库 *pd* 中,*Swap* 也是算法的关键函数之一。

```
function Swap(pd,p,sum)
{
  for(i=1;i<=sum;i++)//项转化为与其对应的十进制数
  {
    pd[i]=d(p[i]);//求最大项集 p 中的第 i 项的度
  }
}
```

Apriori-sort 算法中的函数 *SwapTi*(*D*,*pd*,*db*)为求事务数据库 *D* 中每一个事务 *Ti*(*i*=1,2,…,*n*)的度 *d*(*Ti*),形成对应的以度表示的事务度数据库 *db*,*SwapTi* 也是算法的关键函数之一。

```
function SwapTi(D,pd,db)
{
  while(ReadString(D,Ti)<>null) do //扫描数据库 D 的事务Ti
  {
    db[i]=0;//初始化事务度
    while(ReadString(Ti,It)<>null) do //扫描事务 Ti 中的项 It
    {
      Search(It∈pd[k])//找出 It 项在 p 中的位置 k
      db[i]=db[i]+pd[k];//求事务 Ti 的度 d(Ti)
    }
  }
```

Apriori-sort 算法中的函数 *DbSort*(*dba*,*db*)为对事务度数据库 *db* 进行排序, 排序后的结果放在有序事务度数据库 *dba* 中,*DbSort* 也是算法的关键函数之一。

```
function DbSort(dba,db) //折半插入排序
{
  bda[1]=db[1];//初始化有序事务度数据库 dba 中第一项为事务度数据库 db 第一项
  for(i=2;i<db.count;i++)//对 db 中的每一项进行折半插入排序
  {
    left=0,right=i−1;
    while(left<=right) do //利用折半查找插入位置
    {
      middle=(left+right)/2;//取中点
      if(db[i]<dba[middle])//插入值小于中点值
      {right=middle−1;}//向左缩小区间
      else
      {left=middle+1;}//否则,向右缩小区间
    }
    for(k=i−1;k>=left;k−−)
    {dba[k+1]=dba[k];}//成块移动空出插入位置
     dba[left]=db[i];//插入
  }
}
```

Apriori-sort 算法中的函数 AprioriGen(*Lk*−1,*pd*,*sum*)为产生新的以度表示项集的 *k* 项候选集 *ck*,*AprioriGen* 也是算法的关键函数之一。

```
function AprioriGen(Lk−1,pd,sum)
{
  for(i=1;i<=lk->count;i++)
  {
    for(j=0;j<sum;j++)
    {
      cki=lk−1(i)|pd[j];//形成候选集
      for(k=0;k<=ck->count;k++)//去掉候选集中有相同的项集
      {
        if(ck(k)==cki)
        {next i;}//剪去候选集中有与新项集相同的项集
      }
      for(k=0;k<sum;k++)//去掉候选集中任意子集为非频繁项集的项集(剪支)
```

```
        {
          ma1=cki-p[k];//求候选集的一子集
          if(ma1 ∉ lk-1)//候选集的子集不是频繁项集
          {next i;}//剪支
        }
        ck=ckUcki;//把 cki 加入候选集
      }
        return ck;//返回候选集结果
  }
```

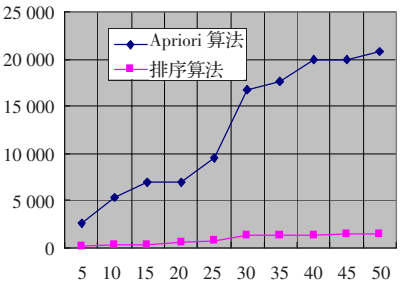

图 1　计算时间变化曲线

4　仿真实验及分析

时间复杂度的定性分析：Apriori-sort 算法查找频繁项集时，只扫描数据库 Dd 中满足条件 $d(Ck)\leqq d(Ti)$ 的事务。对扫描数据库进行了有效剪支。因此 Apriori-sort 算法比传统的 Apriori 算法的计算时间要快。

为了验证 Apriori-sort 算法的正确性和有效性，选定了 500~5 000 条左右样本数据（样本数据文件采用的是 IBM 公司 Almaden 中心提供的用于关联规则的标准仿真数据集），在相同的硬件配置条件下对 Apriori 算法和 Apriori-sort 算法的处理效率进行 10 次测试，两种算法的软硬件测试环境相同，均在 AMD Athlon(rm)64 X2 Dual Core Processor 3600+CPU、1 GB 内存、160 GB 硬盘、Windows XP sp2 操作系统环境。比较 Apriori 算法和 Apriori-sort 算法的计算时间，在所有的仿真数据计算中，Apriori 算法和 Apriori-sort 算法挖掘的结果是一样的，但 Apriori-sort 算法的计算时间远小于 Apriori 算法。实验结果如表 2 和图 1 所示。

表 2　两种算法计算时间表

事务数（100 条）	Aprior 算法/ms	Apriori-sort 算法/ms	事务数（100 条）	Aprior 算法/ms	Apriori-sort 算法/ms
5	2 538	202	30	16 807	1 278
10	5 275	256	35	17 653	1 239
15	6 956	281	40	19 966	1 322
20	6 945	548	45	19 871	1 388
25	9 505	710	50	20 807	1 486

图 1 是测试数据事务数为 500~5 000 时，Apriori 算法和 Apriori-sort 算法的计算时间随事务数量变化的曲线，横坐标为事务库中的事务数据量（单位：100 条），纵坐标为计算时间（单位：ms）。从图中可见，Apriori 算法和 Apriori-sort 算法的计算时间随事务数据量的增加而增加，但 Apriori-sort 算法的增长幅度较小，且 Apriori-sort 算法的计算时间远小于 Apriori 算法。

5　结束语

关联规则的挖掘具有重要的实用价值，在数据挖掘领域应用广泛。由于被挖掘的事务数据库规模较大，关联规则挖掘算法的运行效率就显得特别重要。本文对 Apriori 算法进行了深入的研究，设计了一种基于事务度数据库排序的 Apriori-sort 挖掘算法。该算法只需对事务数据库扫描 1 次，就可把事务数据库转换成十进制数据表示的事务度数据库，然后对事务度数据库进行挖掘。实验表明：Apriori-sort 关联规则挖掘时间效率较高。由于可对事务度数据库进行排序以达到对扫描数据库的剪支的目的，Apriori-sort 的效率较高。同时，Apriori-sort 算法可应用于任何 Apriori 算法适用的场合。

参考文献：

[1] 贾娟，亓文法，侯晓辉，等.基于不规则版面布局模型的区域划分和分区排序算法[J].计算机工程与应用，2003，39(30)：51-53.

[2] Gatos B，Mantzaris S，Perantonis S，et al.Automatic page analysis of a digital library from newspaper archives[J].International Journal of Digital Libraries，2000，3(1)：77-84.

[3] Aiello M，Monz C，Todoran L，et al.Document understanding for a broad class of documents[J].International Journal on Document Analysis and Recognition，2002，5(1)：1-16.

[4] 彭仪普，熊拥军.关联规则挖掘 AprioriTid 算法优化研究[J].计算机工程，2006，32(5)：55-57.

[5] 王德兴，胡学钢，刘晓平，等.改进购物篮分析的关联规则挖掘算法[J].重庆大学学报：自然科学版，2006，29(4)：105-141.

[6] 曾万聃，周绪波，戴劲，等.关联规则挖掘的矩阵算法[J].计算机工程，2006，32(2)：45-47.

[7] 王燕.基于等价关系的关联规则挖掘算法研究[J].计算机工程与应用，2006，42(8)：187-189.

[8] Agrawal R，Imielinski T.Mining association rules between sets of items in large databases[C]//Proc of ACM SIGMOD Conference on Management of Data，Washington DC，1993：207-216.

（上接 142 页）

参考文献：

[1] Foster I，Kesselman C.The grid 2：Blueprint for a new computing infrastructure[M].[S.l.]：Morgan Kaufmann Press，2003.

[2] Azzedin F，Maheswaran M.Integrating trust into grid resource management systems[C]//2002 International Conference on Parallel Processing(ICPP 2002).Canada：IEEE Press，2002：47-54.

[3] Foster I，Kesselman C，Tuecke S.The anatomy of the grid：Enabling scalable virtual organizations[J].Int'l Journal on Supercomputer Applications，2001.

[4] 曹蓉.基于 OGSA 的网格信任模型研究[D].合肥：合肥工业大学，2006.

[5] 陈锦.基于行为的网格信任模型研究[D].西安：西北大学，2007.