一种基于压缩矩阵的 Apriori 改进算法①

An Improved Apriori Algorithm Based on Reducing Matrix

李卫华　(中南大学 信息科学与工程学院 湖南长沙 410083)
(惠州学院 计算机科学系 广东惠州 516001)
刘卫国　(中南大学 信息科学与工程学院 湖南长沙 410083)

摘　要： 通过对 Apriori 算法挖掘过程进行分析，提出一种基于压缩矩阵的 Apriori 改进算法。该算法通过压缩矩阵和减少扫描次数来提高挖掘的速度和减少数据库的 I/O 操作时间的开销，有效提高了关联规则的挖掘效率。并用实例说明该算法是一种有效的关联规则挖掘方法。

关键词： 关联规则 Apriori 算法 频繁项集 矩阵

数据挖掘是一种从大量的、不完全的、有噪声的、模糊的、随机的数据中，提取出隐含在其中潜在有用的信息和知识的过程。它主要包括分类、预测、关联和聚类等基础概念和技术[1]。

关联规则提取发现大量数据中项集之间有趣的关联或相关关系。最经典的关联规则提取算法是 Apriori 算法，其他大部分关联规则挖掘算法都是在该算法的基础上加以改进或扩展得来的。本文提出一种基于压缩矩阵的 Apriori 改进算法是将事务数据库转换为矩阵，并利用 Apriori 性质逐步压缩矩阵。这种方法只需扫描一次数据库，从而大大提高了算法的效率。

1　Apriori 算法及其分析

设 $I=\{I_1,I_2,\cdots\cdots\}$ 是项的集合，D 是数据库事务的集合，其中每一个事务 T 是项的集合，使得 $T\subseteq I$。每一个事务有一个标识符，称作 TID。设 A 是一个项集，事务 T 包含 A 当且仅当 $A\subseteq T$。关联规则是形如 $A=>B$ 的规则，其中 $A=A_1\wedge\cdots\cdots\wedge A_m$，$B=B_1\wedge\cdots\cdots\wedge B_n$，$A_i$，$B_i$ 是属性值对。$A\subset I$，$B\subset I$，并且 $A\cap B=\varnothing$。关联规则具有支持度 s 和置信度 c 这两个重要的阈值：

support$(A=>B)=P(A\cup B)$，即 A 和 B 这两个项集在事务集 D 中同时出现的概率；

confidence$(A=>B)=P(B|A)$，即在出现项集 A 的事务集 D 中，项集 B 也同时出现的概率。

同时满足最小支持度阈值(min_sup)和最小置信度阈值(min_conf)的规则称为强规则。关联规则的挖掘是一个两步的过程：

(1) 找出所有的频繁项集。这些项集出现的频繁性至少和预定义的最小支持度一样。

(2) 由频繁项集产生强关联规则。这些规则必须满足最小支持度和最小置信度。

这两步中，第(2)步最容易。挖掘关联规则算法的总体性能由第(1)步决定，因此大部分关联规则挖掘算法着重研究第(1)步，即频繁项集挖掘算法。

Apriori 使用一种称作逐层搜索的迭代方法，k－项集用于探索(k+1)－项集。首先，找出频繁 1－项集。该集合记作 L_1，L_1 用于找频繁 2－项集 L_2，而 L_2 用于找 L_3，如此下去，直到不能找到频繁 k－项集。找每个 L_k 需要一次数据库扫描。

Apriori 性质：频繁项集的所有非空子集都必须也是频繁项集。将 Apriori 性质用于找频繁项集分成两个过程：连接和剪枝。在连接部分，L_{k-1} 与 L_{k-1} 连接产生可能的候选项集，剪枝部分使用 Apriori 性质删除具有非频繁子集的候选项集。

从 Apriori 算法的步骤不难看出，该算法有三个缺点：①在每一步产生的候选项集过多，没有排除不应该

① 基金项目：基金项目(编号)；科研项目(编号)

参与组合的元素;②每次计算子项集的支持度时,都要进行了一遍数据库扫描比较,大大增加系统的 I/O 开销,并且数据库有些可以删除的项或事务被多次扫描。③连接程序中相同的项重复比较太多。

2 基于压缩矩阵 Apriori 算法的改进

针对 Apriori 算法的缺点,有关学者对此做了改进。文献[2-4]从不同的角度提出了矩阵与关联规则挖掘结合的改进算法,将事务数据库转换为基于内存的矩阵,在矩阵上找出所有的频繁项集,从而大大减少了数据库的扫描次数,提高了算法的效率。但是,这些算法各有优缺点。文献[2]中,没有进行矩阵的压缩。文献[3]中的矩阵数据结构较为合理,也进行了矩阵压缩,但压缩的不彻底。文献[4]对矩阵的压缩比较彻底,同时减少了算法循环的次数,但是矩阵数据结构不合理,并且在求频繁项集时需要将参与运算的向量拷贝到另外开设的一个矩阵,增加了空间复杂度。本文改进了矩阵的数据结构:在一个单纯的事务矩阵中,添加两个辅助行和一个辅助列,方便进行矩阵压缩;同时为了配合查找频繁 k-项集(k>=2)的运算,设置一个简单的辅助二维数组,用来记录下标组合情况。文中还对 Apriori 性质进行引申和利用,进行彻底的矩阵压缩,提高算法的效率。

定义 1　布尔矩阵:矩阵元素的值为"0"或"1"的矩阵。

定义 2　每个项 I_i 的支持度为 $support(I_i)=\sum_{i=1}^{m} r_{ij}$($r_{ij}$ 为布尔矩阵 R 中的元素),即该列向量中"1"的个数。

定义 3　k-项集支持度:对布尔矩阵 R 中的任意 k 列向量进行对位(同行的元素)"与"运算,运算结果中"1"的个数称为 k-项集支持度。

根据 k-项集支持度的定义,结合 Apriori 性质,我们可以得到以下性质。

性质 1　X_k 是 k-项集,如果频繁(k-1)-项集 L_{k-1} 中包含 X_k 的(k-1)-子项集的个数小于 k,则 X_k 不可能是 k 维最大频繁项集。[5]

证明见文献[5]。进一步可以推出,频繁项集 X_k 中每一个元素在(k-1)维频繁集的出现次数不小于 k-1。因此在布尔矩阵 R_{m*n} 中某个项 I_i 在 L_{k-1} 中出现的次数小于 k-1,则可以从该布尔矩阵中删除 I_i 对应的列。裁剪后的布尔矩阵中,每个项 I_i 在(k-1)-频繁项集中出现次数都为 k,也就是说所有包含 I_i 的 k-1 维子集都在(k-1)-频繁集中,所以对该矩阵中项的 k 组合不需要做剪枝。

性质 2　设 L_k 为 k 项频繁项集,若 $T\subseteq I$,且 $|T|=k$,则在后续挖掘中 T 为数据库中可删除事务。

该结论显然成立。因为 k-项集对于生成频繁(k+1)-项集是没有用的,所以布尔矩阵 R_{m*n} 中某行的事务数等于 k,则求(k+1)项集支持度时,该行可删除。

性质 3　对于频繁 k-项集的集合 L_k,如果 $|L_k|<k+1$,则被挖掘的事务数据库,最大频繁项集的次数为 k。其中 $|L_k|$ 是指 I_k 的频繁 k-项集的个数。

证明:对于频繁(k+1)-项集 $I_x=\{I_1,I_2,\cdots,I_{k+1}\}$,一定有 k+1 个频繁 k-项子集,若频繁 k-项集的集合 L_k 元素个数小于 k+1,则被挖掘的事务数据库不存在频繁(k+1)-项集。利用该性质可以提前终止搜索频繁集的循环。

具体的算法描述如下:

(1) 扫描数据库 D,建立矩阵 R

对于包含 m 个事务 n 个项目的事务数据库 $D=\{T_1,T_2,\cdots,T_m\}$,项目集 $I=\{I_1,I_2,\cdots,I_n\}$,建立布尔矩阵 R。扫描事务数据库 D,如果 $I_j\in T_i(1\leqslant j\leqslant n,1\leqslant i\leqslant m)$,则 $R_{ij}=1$,否则 $R_{ij}=0$。为了方便后面的计算,我们为矩阵添加一列 sum_r,记录每个事务的事务数,即累计每行的项数;为矩阵添加一行 sum_c,记录每个项的项支持度;为矩阵添加一项行,记录项的编码。生成布尔矩阵 $R_{(m+2)*(n+1)}$,具体算法如下:

```
for (j=1;j<n;j++)
  { R[0][j]=j ; //填充项编号行
    R[m+1][j]=0 ; //填充 sum_c 行,初值为0}
for (i=1;i<=m;i++)
  { sum_r=0; //项支持度初值为 0
    for (j=1;j<=n;j++)
    if (Ij in Ti)
    { R[i][j]=1;
      sum_r++; //统计项支持度
      R[m+1][j]++ ; //统计行的项数}
       else R[i][j]=0;
```

R[i][0] = sum_r;
}

(2) 计算最小事务支持度 min_supsh = ceiling (min_sup * m) (ceiling 函数用于求不小于自变量的最小整数)。对比矩阵 sum_c 行,如果某元素值小于 min_supsh,则删除矩阵中该元素对应的列,余下的项都是频繁 1 - 项集。重新计算矩阵的 sum_r 列,如果某元素值等于 0,则删除矩阵中该元素对应的行。

(3) k = 2;

(4) 压缩矩阵。按照性质 1,删除在频繁 (k-1) - 项集中出现次数小于 (k-1) 的项 I_i 对应的矩阵列。重新计算矩阵的 sum_r 列,按照性质 2,如果某元素值小于 k,则删除矩阵中该元素对应的行;重新计算矩阵的 sum_c 行,如果某元素值小于 min_supsh,则删除矩阵中该元素对应的列;反复压缩矩阵,直到不能再压缩为止。

(5) 求频繁 k - 项集。若压缩后事务矩阵 R 列数为 c_num (k < c_num),对事务矩阵中的后 c_num - 1 列进行 k 维组合,则生成 $p = C_{c\_num-1}^{k-1}$ 个组合对,每个组合对中的 k 个列向量进行"与",对结果向量求 k 项集支持度,k 项集支持度不小于 min_supsh 的组合对所对应的项集为频繁 k - 项集。求频繁 k - 项集的主要代码如下:

```
// R 为压缩后的事务矩阵
// r_num 为 R 的行数,c_num 为 R 的列数
// Lk: 频繁 k - 项集
//Wp*k数组存放 p 个 k 维项组合对
for (g=0; g<p; g++) //对每个 k 维组合求 k 项集支持度
{ supnum=0;//k 项集支持度初值为 0
  for(i=1;i<r_num-1;i++)
    { cj=1;
      for (j=1; j<=k;j++)
        { cj=cj*R[i][W[p][j]]; //"与"运算
          if (cj==0) break; }
      supnum=supnum+cj; // 统计 k - 项集支持度
      }
if supnum >=min_supsh // 为频繁集
  { for (j=0; j<k;j++)
      L[j]=R[0][ W[p][j]];//求 k 维数组合的项集
      Lk = Lk∪L;}
  }
```

(6) 统计 $|L_k|$,如果 $|L_k| >= k+1$,$k = k+1$,跳转到 (4);否则停止算法。

3 实例分析

根据改进的算法,下面通过一个具体的例子进行分析,事务数据库由表 1 给出,假定 min_sup = 25%,最小事务支持计数为 min_supsh = ceiling (10 * 25%) = 3。

表 1

TID	项集	TID	项集
T1	I1,I2,I4,I5,I6	T6	I2,I3,I5,I6
T2	I2,I3,I5,I6	T7	I3
T3	I2,I3	T8	I2,I4,I5,I6
T4	I3,I4,I5	T9	I3,I4,I5
T5	I1,I3,I5,I6	T10	I5,I6

(1) 将事务数据库转换为布尔矩阵如表 2。

表 2

	sum_r	1	2	3	4	5	6
T1	5	1	1	0	1	1	1
T2	4	0	1	1	0	1	1
T3	2	0	1	1	0	0	0
T4	3	0	0	1	1	1	0
T5	4	1	0	1	0	1	1
T6	4	0	1	1	0	1	1
T7	1	0	0	1	0	0	0
T8	4	0	1	0	1	1	1
T9	3	0	0	1	1	1	0
T10	2	0	0	0	0	1	1
sum_c		2	5	7	4	8	6

(2) 扫描表 2,求频繁 1 - 项集。I_1 对应的 sum_c < min_supsh,因此删除项 I_1 在矩阵中的对应列,重新计算矩阵 sum_r 列,并删除值为 0 的元素所在的行,得到的矩阵如表 3。得频繁 1 - 项集 L_1 = {I2:5, I3:7, I4:

4,I5:8,I6:6}

表 3

	sum_r	2	3	4	5	6
T1	4	1	0	1	1	1
T2	4	1	1	0	1	1
T3	2	1	1	0	0	0
T4	3	0	1	1	1	0
T5	3	0	1	0	1	1
T6	4	1	1	0	1	1
T7	1	0	1	0	0	0
T8	4	1	0	1	1	1
T9	3	0	1	1	1	0
T10	2	0	0	0	1	1
sum_c		5	7	4	8	6

(3) k=2,扫描表3,根据性质2,T7 对应的 sum_r =1<k,删除该行,找不到可删除的列,得表4。

表 1　　W 数组

	sum_r	2	3	4	5	6
T1	4	1	0	1	1	1
T2	4	1	1	0	1	1
T3	2	1	1	0	0	0
T4	3	0	1	1	1	0
T5	3	0	1	0	1	1
T6	4	1	1	0	1	1
T8	4	1	0	1	1	1
T9	3	0	1	1	1	0
T10	2	0	0	0	1	1
sum_c		5	6	4	8	6

1	2
1	3
1	4
1	5
2	3
2	4
2	5
3	4
3	5
4	5

表 2　　W 数组

	sum_r	2	3	5	6
T1	3	1	0	1	1
T2	4	1	1	1	1
T5	3	0	1	1	1
T6	4	1	1	1	1
T8	3	1	0	1	1
sum_c		4	3	5	5

1	2	3
1	2	4
1	3	4
2	3	4

计算得{I2I3:3,I2I4:2,I2I5:4,I2I6:4,I3I4:2,I3I5:5,I3I6:3,I4I5:4,I4I6:2,I5I6:6},所以频繁 2 - 项集 L_2 = {I2I3:3, I2I5:4,I2I6:4, I3I5:5,I3I6:3,I4I5:4,I5I6:6}。

(4) k=3,由于 I4 在 L_2 中出现的次数为 1<k-1,所以删除 I4 对应的列,重新计算最后一列,删除 T3,T4,T9,T10 对应的行,得表5。

计算得{I2I3I5:2,I2I3I6:2,I3I5I6:3, I2I5I6:4},所以频繁 3 - 项集 L_3 = {I3I5I6:3, I2I5I6:4},由于 $|L_3|$ =2<k+1;最大频繁集的次数为3,算法结束。

4 结束语

时间复杂度定性分析:与 Apriori 算法相比,改进算法只扫描事务数据库一次。不需要对频繁(k-1)项集进行连接和剪枝,直接通过事务矩阵的 k 维列向量计算 k 项集支持度求频繁 k-项集。并且在计算过程,对事务矩阵进行有效的剪裁,这样 k 值越大,矩阵被压缩的越小,求 k 项集支持度的计算量越小,因此效率越高。

空间复杂度的定性分析:Apriori 算法所需要存储的数据主要是交易记录,改进算法存储的也是相同的数据,但事务矩阵中存储的主要是要是布尔值"0"和"1",可以位方式进行存储,这样占用的内存空间较少。

关联规则挖掘在货篮分析、数据库营销、Web 站点优化等方面具有很好的应用价值。本文对关联规则挖掘算法 Apriori 进行了研究和改进,但是该改进算法只能挖掘布尔类型的关联规则,而不是量化关联规则,这是需要进一步研究的课题。

参考文献

1 Jiawei Han, Micheline Kamber 著,范明、孟小峰等译,数据挖掘概念与技术,北京:机械工业出版社,2001.

2 李超、余昭平,基于矩阵的 Apriori 算法改进,计算机工程,2006,32(23):68~69.

3 曾万聘、周绪波、戴劲、常桂然、李春平,关联规则挖掘的矩阵算法,计算机工程,2006, 32(2):45~47.

4 王柏盛、刘寒冰、靳书和、马丽艳,基于矩阵的关联规则挖掘算法,微计算机信息(管控一体化,2007年,24(53):143~145.

5 张素兰,一种基于事务压缩的关联规则优化算法,计算机工程与设计,2006,27(18): 3450~3453.