基于等价类的关联规则挖掘矩阵算法

王鸿铭[1],沈夏炯[1,2],李国雁[1],臧国轻[1]
(1.河南大学 计算机与信息工程学院,河南 开封 475004;
2.河南大学 数据与知识工程研究所,河南 开封 475004)

摘 要:关联规则挖掘算法中的Apriori算法利用查找频繁项集来发现数据集中的关联规则,算法思路简单容易实现;但在由 $k-1$ 次频繁项集生成 k 次频繁项集时需反复查找数据库,效率较低,在寻找高次频繁项集时其低效性更加明显;矩阵算法是通过直接查找高次频繁项集,避免了反复查找数据库,但要存储大量的非频繁项集,且查找低次频繁项集速度较慢。文中提出的矩阵等价类算法,利用等价关系进一步降低矩阵算法的时间空间复杂度,然后通过项目相似度直接求取所有最大频繁项集。实验结果证明了算法的可行性、高效性。

关键词:等价类;项目相似度;矩阵算法;Apriori 算法
中图分类号:TP301.6 **文献标识码**:A **文章编号**:1673-629X(2008)04-0055-04

An Association Mining Matrix Algorithm Based on Equivalence Class

WANG Hong-ming[1],SHEN Xia-jiong[1,2],LI Guo-yan[1],ZANG Guo-qing[1]
(1.College of Computer and Information Engineering,Henan University,Kaifeng 475004,China;
2.Institute of Data and Knowledge Engineering,Henan University,Kaifeng 475004,China)

Abstract:Apriori algorithm,one of associated rules of mining algorithms,utilizes searching results on frequent itemsets to find associated rules in datasets. Though the algorithm is easy to implement,it spends enormous time and becomes much more unefficient on searching in the databases, especially when k times frequent itemsets are created by $k-1$ ones and searching for the high level frequent itemsets. The matrix algorithm avoids these situation but needs store substantive unfrequent itemsets and is low velocity when searching for low level frequent itemsets. The matrix-equivalence genus algorithm mentioned in this paper, is proved that can reduce the complexity on time and room effectively on finding all most frequent itemsets.

Key words:equivalence class;item similitude degree;matrix algorithm;Apriori algorithm

0 引 言

关联规则是一个重要的数据挖掘研究课题,它反映了事务集合中项集之间有价值的关联或相关联系。从事务集合中发现所有满足用户给定的最小支持度的频繁项集是挖掘关联规则的关键步骤,这一步骤几乎集中了关联挖掘工作所有的计算量[1]。关联挖掘的算法研究中,Agrawal 等人提出的 Apriori[2]算法最为著名,其后的数据挖掘算法大多以 Apriori 算法为基础进行改进,如 AprioriTid、AprioriHybird 等[2,3],这类算法从一维频繁项集开始对数据库进行多次扫描,对任何一个 k 维的频繁项集 F,算法必须产生 F 所有的 2^k 个子集,当 k 很大时,运算量巨大。据此文献[4]提出了关联挖掘矩阵算法,该算法跨越了由低次到高次逐步查找频繁子集的限制,在不知道低次 $k-1$ 频繁项集的前提下可以直接计算 k 项集[4],但要存储非频繁项集,低次频繁项集的查找速度较慢,并且在给定 k 值而找不到频繁项集的情况下,若低于 k 维的频繁项集存在,也无法找到。

文中提出新的算法——矩阵等价类算法,先构造数据矩阵并且对数据矩阵进行行列删除,然后通过等价关系对数据矩阵进一步约简,以降低算法的时间空间复杂度,之后利用项目相似度直接求取所有的最大频繁项集。实验证明了该算法的高效性。

1 矩阵等价类算法的定义及性质

矩阵等价类算法中,有如下定义及性质:

定义1 最大频繁项目集(Maximal Frequent Itemset):某频繁项目集说是最大频繁项目集,如果它不是其他任何频繁项目集的子集。

收稿日期:2007-07-23
基金项目:河南省自然科学基金项目(0311011700)
作者简介:王鸿铭(1979-),男,硕士研究生,研究方向为数据挖掘;沈夏炯,副教授,硕士生导师,研究方向为形式概念分析、知识发现。

定义2 最大频繁项集集合(Maximum Frequent Itemset Set,MFS):集合中每一个元素都是最大频繁项目集,MFS = $\{FS_1, FS_2, \cdots, FS_i\}$, $i = 1,2,\cdots, n$; n 为集合中最大频繁项集的个数。

定义3 项目相似度对于BOOL型矩阵 $A_{m*n} = \{A_{ij} \mid$ 其中 $A_{ij} = 0$ 或 $1, i = 1,2,3,\cdots, m; j = 1,2,3,\cdots, n\}$,定义 A 中任意两列 A_j 和 A_k 中第 $i(1 \leqslant i \leqslant m)$ 行的值均为1的行的个数称为 A_j 和 A_k 的项目相似度,项目相似度用符号 ItemSim(A_j, A_k) 表示,其中 A_j, A_k 为 A 中的两个任意不同的列,即 $j \neq k$。

根据项目相似度的定义,可知项目相似度表示 A_j, A_k 所代表的项目同时出现在不同事务中的频度,从而由定义易得到如下性质:

性质1 布尔矩阵 A_{m*n} 中的任意两列的项目相似度等于这两列的对应行的乘积的和,即 ItemSim$(A_j, A_k) = \sum A_{ij} * A_{ik}$,其中 $i = 1, 2, 3, \cdots, n, A_j, A_k$ 为 A 中的两个任意不同的列,即 $j \neq k$。

性质2 假定最小支持度为 min_ support,事务数为 g,对于BOOL型矩阵 $A_{m*n} = \{A_{ij} \mid$ 其中 $A_{ij} = 0$ 或 $1, i = 1, 2, 3, \cdots, m; j = 1,2,3, \cdots, n\}$,若矩阵中的两列 A_j, A_k 的项目相似度为 s,如果 $s/g * 100\% >$ min_ support,则 A_j 和 A_k 所代表的项目存在于同一频繁项集中。这也给出了频繁项集的计算方法。

矩阵等价类算法利用事务 value 值的相等关系进行事务的等价类划分,进而约简数据矩阵。其相关定义如下:

定义4 给定事务集 D 和项集 I,对于 $\forall t \in D$,定义 Value(t) = BinToDec(A_{ij}),其中 $A_{ij} = 0$ 或 $1, i = 1,2,3,\cdots, \mid D \mid; j = 1,2,3,\cdots, \mid I \mid$,称为事务 t 的 value,A_{ij} 的顺序为矩阵中的当前顺序;BinToDec 函数将二进制字符串转化为对应的十进制值[5]。如在表3中,value(2000) = 56, value(5000) = 19。

定义5 设 R 为定义在集合 A 上的一个关系,若 R 是自反的、对称的和传递的,则 R 称为等价关系。

定义6 设 R 为集合 A 上的等价关系,对任何 $a \in A$,集合 $[a]_R = \{x \mid x \in A, aRx\}$ 称为元素 a 形成的 R 等价类。

由等价类的定义可知 $[a]_R$ 是非空的,因为 $a \in [a]_R$,因此任给集合 A 及其上的等价关系 R,可写出 A 上各个元素的等价类[6]。

定义7 给定事务集 D 和项集 I,对于 $\forall t \in D$,R 为 D 中 value 值相等关系,称 D/R 为 D 中的一个划分,即 $D/R \subseteq D \times D: \{(t_1, t_2) \mid t_1, t_2 \in D$ 且 Value (t_1) = Value $(t_2)\}$。

如在表3中,根据等价关系,事务集被划分为4个等价类 $[56]_R$、$[48]_R$、$[36]_R$ 和 $[19]_R$。

性质3 等价类中事务具有相同的0、1表示序列。证明略。

2 矩阵等价类算法的基本原理及实现步骤

对于事务集 $D = \{T_1, T_2, \cdots, T_g\}$,设 g 为其事务数 $\mid D \mid$。挖掘其中潜在的关联规则,先构造数据矩阵,利用频繁项集的性质进行矩阵的行列删除,再利用等价类进一步简化矩阵,之后通过项目相似度直接求取最大频繁项目集,进而找出其中的关联规则。其实现步骤如下:

(1)构建数据矩阵。

根据如下步骤构建数据矩阵:

① 求出项集的最大集合 I 和集合中元素的个数 f。

$I = \{I_k \mid k = 1, 2, 3, \cdots, f$ 并且 $I_k \in T_i$,其中 $i = 1, 2, 3, \cdots, n\}$

② 创建矩阵 A_{m*n},其中 $m = g + 2, n = f + 3$,元素 $A_{1i} \in I, 2 \leqslant i \leqslant f + 1$ 且 $\cup A_{1i} = I; A_{j1} \in D$,$2 \leqslant j \leqslant g + 1$ 且 $\cup A_{j1} = D; A_{1n-1}$ = sum,A_{1n-1} 记录该列左边的各行中1的个数;A_{1n} = value,A_{1n} 记录 sum 列左边各行01码转换成十进制的值;A_{m1} = sup,A_{m1} 记录该行上面对应列中1的个数。

(2)填充数据。在矩阵 A 中各行各列填入相应的值。填入原则如下:

$A_{ij} = \{A_{ij} \mid$ 若 $I_i \in T_j$,则 $A_{ij} = 1$,否则,$A_{ij} = 0$,其中 $i = 2,3,\cdots, f+1, j = 2,3,\cdots, g+1\}$,并在最后一行(sup 行)中汇总对应列1的个数,A_{1n-1} 列汇总对应行1的个数。

(3)删除不符合条件的行、列。一般不研究1项集,因此去掉 sum 中值为1的行,由于 sup$[m]$ 值代表 I_m 在交易中出现的频度,所以可根据 t = sup $[m]/g * 100\%$ 的值是否大于 min_ Support 进行行删除,若 $t <$ min_ Support,则删除 m 列,其中 $m = 2,3, \cdots, f+1$。重新计算矩阵中项集的集合 I 及项目的个数 f_1 和事务集 D 及事务的个数 g_1。

(4)初始化最大频繁项集集合 MFS。即用矩阵中每一个1维频繁项集,初始化 MFS 中一个子集。

(5)根据 value 的定义,即 Value(t) = BinToDec (A_{ij}),其中 $A_{ij} = 0$ 或 $1, i = 2,3,\cdots, m-1; j = 2,3, \cdots, n-2$,计算各行的 value 值,然后根据 value 值进行等价类划分。

(6)等价类划分。根据等价关系和等价类的概念,设 R 为集合 D 具有相同 Value 值的关系,显然 R 是 D

的等价关系。通过 R 关系可以确定 D 上的一个划分，其中的每一个分块即为 D 集合上的 R 等价类。为了描述方便，设 E_i 为 D/R 集合集中第 i 个等价类。由于在事务集中会出现大量的事务具有相同的项目，因此，把这些事务划为一类，在求取高维频繁项集时，用其中一个做代表会极大地降低算法时间复杂度。为了计算方便，定义 E_i 的数据结构如下：

```
Type struct T{int n;//记录中事务的个数
    String itemstring;//等价类事务的项目字符串}
T E[m];
```

根据 value 的值进行等价类划分，对每个等价类 E_i，仅保留其中一个事务所对应的行，删除其余事务所对应的行。重新计算矩阵中事务集 D 及事务的个数 g_1，并更新矩阵。算法如下：

```
k=0;//k 表示 D/R 中第 k 个等价类
for (i=2;i<g+1;i++){Val=value[i,n];//val 存放 value 的临时值
  k=k+1;
  //getTrsstr 函数求取事务 A[i,1]中的项目字符串
  E[k].itemstring=getTrsstr(A[i,1]);
  for (j= i+1;j<= g+1;j++){
    if (A[j,n]=val) {E[k].n=E[k].n+1;//等价类中事务个数加 1
        Delete(j);//删除第 j 行}}}
```

(7)求最大频繁项集的集合 MFS。从矩阵的第二列开始，依次求取与其后各列的项目相似度，即求取 t = ItemSim(I_j, I_k)，其中，$j = 1,2,\cdots,f_1$；$k = 1,2,\cdots,f_1$，且 $j \neq k$。若 $t/g * 100\%$ 大于 min_ support，则把 I_k 加入到 FS_j 中。其算法如下：

```
//求最大频繁项集的集合 MFS
for (i=2;i<=f1;i++){
  S=A[1,i]//表示把项目 A[1,i]给项目变量 S
  for (j=2;j<= f1;j++){
    if (i<>j){t=ItemSim(S,A[1,j]);//求两项目的相似度
    若等价类字符串中存在这两项目，则 t 的值再加上所在等价类中事务数减 1;}
Get(temp);//temp 保存项目 S、A[1, j]做相似度计算后产生的临时项目
//若 t/g*100%>min_support 则把 A[1,j]加入到 A[1,i]所在的频繁项目集中
if (t/q*100%>min_support) {
  A[1,j]into FS1;// 频繁项集 FS1⊆MFS
  S=Temp;// 求临时项目 temp 与后面项目的相似度}}}
```

其中 ItemSim 函数用于求取两项目的相似度，其主要实现过程如下：

```
int ItemSim(A[m,i],A[m,j])
{ int k=0;if (i<>j) for (m=2;m<=g1;m++) { k=k+A[m,i]*A[m,j]}return k;}
```

3　矩阵等价类算法的应用实例

假设有如下事务集，最小支持度 50%，如表 1 所示。

表 1　事务集

Transaction ID	Items Bought	Transaction ID	Items Bought
2000	A,B,C	5000	B,E,F
1000	A,B	8000	A,D
4000	A,D	7000	A,B

根据矩阵等价类算法，矩阵变化流程如下所示。

(1)根据表 1 构造矩阵，如表 2 所示。

表 2　空数据矩阵

	A	B	C	D	E	F	Sum	Value
2000								
1000								
4000								
5000								
8000								
7000								
Sup								

(2)填充数据，并计算各项值，如表 3 所示。

表 3　数据矩阵

	A	B	C	D	E	F	Sum	Value
2000	1	1	1	0	0	0	3	56
1000	1	1	0	0	0	0	2	48
4000	1	0	0	1	0	0	2	36
5000	0	1	0	0	1	1	3	19
8000	1	0	0	1	0	0	2	36
7000	1	1	0	0	0	0	2	48
Sup	5	4	2	2	1	1		

(3)删除非频繁项目列，等价类划分并删除相应行，然后计算合并，如表 4 所示。

表 4　约简后的数据矩阵

	A	B	Sum	Value
2000	1	1	3	56
1000	1	1	2	48
4000	1	0	2	36
5000	0	1	3	19
Sup	5	4		

由以上算法可知，等价类分别为 $E[1] = \{1, AB\}$，$E[2] = \{2, AB\}$，$E[3] = \{2, A\}$，$E[4] = \{1, B\}$，显然同时包含 AB 的事务数为 $2+2-1=3$，所以 AB 的最小支持度为 $3/6 * 100\% = 50\%$，满足条件，因此，最大频繁项目集为 $\{A, B\}$(去除一项最大频繁项目集)。在此基础上可求出各频繁项集的可信度，进而求得关联规则。

4　矩阵等价类算法与其它算法的比较分析

●时间复杂度定性分析：矩阵等价类算法打破了

Apriori 算法由低次到高次逐步查找频繁子集和多次读取数据库的限制，一次读取事务记录后，利用项目相似度直接求取所有最大频繁项集，若事务数为 m，项目数为 n，则时间复杂度为 $O(m\ n^2)$，同条件下，Apriori 算法的时间复杂度为 $O(m2^k)$（k 为最大频繁项集的维数），显然，k 较大时 $O(m\ n^2)$ 比 $O(m2^k)$ 小，但矩阵等价类算法需划分等价类，因此当 k 较小时，其效率比 Apriori 算法低。由于矩阵算法需要事先设定 k，低于和大于 k 维的最大频繁项目集无法求得，在此不做比较。

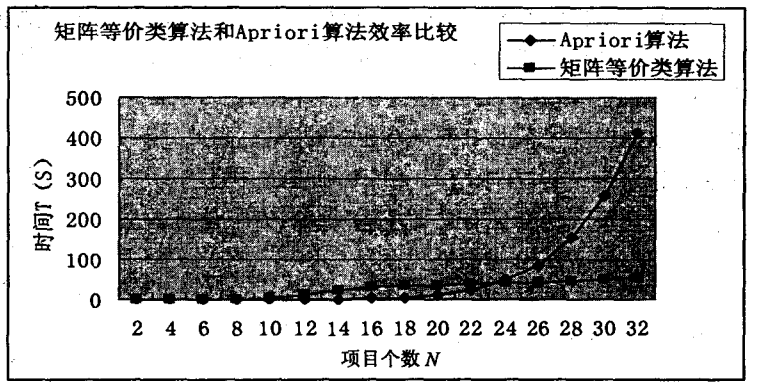

图 1 矩阵等价类算法和 Apriori 算法效率比较图

●空间复杂度定性分析：矩阵等价类算法存储的是 0 和 1 这种布尔数据，因而在事务集较大时会比 Apriori 算法占用更少的空间；由于采用等价类进一步约简数据矩阵，因此也比矩阵算法空间复杂度低。

为了测试矩阵等价类算法的效率，选定了 1 万条样本数据，矩阵中 1 的分布率为 45%，项目数量 N 从 2 到 32，最小支持度为 0.003，在相同硬件配置条件下对 Apriori 算法和矩阵等价类算法的处理效率进行 3 次测试，求平均时间 $T_{平}$。测试结果如表 5 所示（时间单位：s）。

表 5 Apriori 算法和矩阵等价类算法的测试结果

序号	N 的取值	Apriori 的 $T_{平}$	矩阵等价类的 $T_{平}$	序号	N 的取值	Apriori 的 $T_{平}$	矩阵等价类的 $T_{平}$
1	2	0.1	0.6	9	18	5.4	33.6
2	4	0.2	0.9	10	20	11.5	35
3	6	0.3	1	11	22	24	38.1
4	8	0.4	1.7	12	24	50.1	41
5	10	0.5	4	13	26	86.1	43.4
6	12	0.8	11.1	14	28	152.6	47
7	14	1.2	22.2	15	30	256.1	51.2
8	16	2.4	29.5	16	32	410.1	54.3

实验结果比较图如图 1 所示。

由实验结果不难看出，在 N 较大且频繁项集的维数 k 也较大的情况下，查找所有的最大频繁项集，矩阵等价类算法响应时间明显比 Apriori 算法的响应时间短。

5 结束语

关联规则在顾客购物分析、商品广告邮寄分析、网络故障分析等方面都有广泛应用。文中所设计的矩阵等价类算法，能快速挖掘事务集中潜在的关联规则，在处理布尔型项集且 K 较大时，具有比 Apriori 算法更高的时间和空间效率，具有良好的可行性。

参考文献：

[1] 刘大有，刘亚波，尹治东. 关联规则最大频繁项目集的快速发现算法[J]. 吉林大学学报：理学版，2004，42(2)：212－215.

[2] Agrawal R, Imielinski T, Swami A. Mining Association rules Betweensets of items in large Databases[C]//Proceedings of the 1993 ACM SIGMODInternational Conference on Management of Data (SIGMOD'93). New York: ACM Press, 1993: 207－216.

[3] Agrawal R, Srikant R. Fast Algorithms for Mining Association rules[C]//Proc. 20th Int. Conf. Very Large Data Bases (VLDB'94). [s.l.]: Morgan Kaufmann Publishers, 1994: 487－499.

[4] 曾万聃，周绪波，戴 勃，等. 关联规则挖掘的矩阵算法[J]. 计算机工程，2006(1)：45－47.

[5] 沈夏炯，贾培艳，刘宗田. 形式背景同构判定的等价类算法[J]. 计算机科学，2006(12)：148－151.

[6] 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，2006，42：187－189.

（上接第 54 页）

challenges and benefits[J]. Communications of the AIS, 1999, 1(7): 2－28.

[2] Yogesh M. Knowledge management for the new world of business[J]. Asia Strategy Leadership Institute Review, 1998(6): 58－60.

[3] Nissen M E, Levitt R E. Agent－based modeling of knowledge flows: Illustration from the domain of information systems design[C]//The 37th Hawaii IEEE Int'l Conf. System Sciences. Hawaii: [s.n.], 2004.

[4] 窦万春，刘茜萍，蔡士杰. 面向认知协作的知识流分析与研究[J]. 计算机研究与发展，2006，43(6)：1109－1114.

[5] Zhuge Hai. Knowledge flow management for distributed team software development[J]. Knowledge－Based System, 2002, 15(8): 465－471.

[6] Zhuge Hai. A knowledge flow model for peer－to－peer team knowledge sharing and management[J]. Expert System with Application, 2002, 22(4): 313－320.