关联规则挖掘的Apriori算法的改进

黄 进* 尹治本

(西南交通大学计算机与通信工程学院 成都 610031)

【摘要】提出一种将Apriori算法与散列技术和事务压缩技术相结合的改进算法，研究了散列函数的构造及其对算法效率的影响，分析了事务压缩技术的原理及其实现方法，用实例给出了原算法与改进算法的实现步骤，结果表明，新算法减小了存储空间，提高了算法的效率，并改进了数据挖掘技术的性能。

关 键 词 数据挖掘; 关联规则; Apriori算法; 散列; 事务压缩

中图分类号 TP311 文献标识码 A

Improvement of Apriori Algorithm for Mining Association Rules

Huang Jin Yin Zhiben

(College of Computer and Communications Engineering, Southwest jiaotong University ChengDu 610031)

Abstract This paper puts forward an enhanced algorithm which associates Apriori with hash technique and transaction reduction technique. The construction and the influence to algorithm's efficiency of hash function is studied.The theory and realizable method of transaction reduction technique are also analyzed.And then the realizable steps of the foold algorithm and the enhanced algorithm are made out through an example. The result shows that the new algorithm promotes the algorithm' efficiency and at the same time improves the performance of the data mining technique through cutting down the store space.

Key words data mining; association rule; Apriori algorithm; hash; transaction reduction

文献[1]提出了Apriori算法，该算法是挖掘布尔关联规则频繁项集的最有影响的数据挖掘算法之一，它运用Apriori性质并且通过连接、剪枝等步骤，在产生较少侯选项集的情况下产生频繁项集，进而产生强关联规则。由于数据挖掘本身决定其面临的是海量数据，因此在许多情况下仍会产生大量的侯选项集尤其是侯选2_项集，从而严重影响挖掘的效率。文献[2]提出的结合散列技术的关联规则挖掘方法提供了一种思路，但其缺乏有效的散列函数。本文提出将Apriori算法与散列技术和事务压缩技术相结合的改进算法，并通过构造一个有效的散列函数大大提高了Apriori算法产生频繁项集的有效性。

1 Apriori算法

Apriori算法是挖掘海量数据关联规则频繁项集的有效算法[3]，其有效性主要体现在Apriori性质及其连接、剪枝操作步骤上。

Apriori性质：频繁项集的所有非空子集必定是频繁的，或非频繁项集的所有超集必定是非频繁的。若存在项集I，I不是频繁的，即$P(I)<$min_sup，则项i添加到项集I的结果项集$i\cup I$必定也不是频繁的，即$P(i\cup I)<$min_sup(min_sup为最小支持度)，这说明Apriori性质满足反单调性。

2002年8月28日收稿

* 男 25岁 硕士 主要从事数据挖掘和知识发现方面的研究

Apriori算法的主要步骤：首先扫描事务数据库TDB，根据最小支持度min_sup产生频繁1_项集的集合L_1；由L_1执行连接($L_1 \infty L_1$)和剪枝操作，产生侯选2_项集的集合C_2，并根据min_sup产生频繁2_项集的集合L_2；再由L_2产生L_3，如此重复，直到$L_k=\varnothing$，结束。

连接操作：设$L_k=\{l_1,l_2,l_3,\cdots,l_n\}$，$l_i$、$l_j$ $(1\le i\le n, 1\le j\le n, i\ne j)$ 是其中的两个元素；$l_i=\{l_i[1], l_i[2], l_i[3], \cdots, l_i[k]\}$，$l_i[m](1\le m\le k)$是项集$l_i$中的第$m$个项。$L_k$中的两个元素是可连接的当且仅当这两个元素的前$k-1$个元素相同，即当$(l_i[1]=l_j[1])\wedge(l_i[2]=l_j[2])\wedge(l_i[3]=l_j[3])\wedge\cdots\wedge(l_i[k-1]=l_j[k-1])\wedge(l_i[k]\ne l_j[k])$时，$l_i \infty l_j=\{l_i[1], l_i[2], l_i[3], \cdots, l_i[k], l_j[k]\}$，$l_i \infty l_j \in C_{k+1}$。

剪枝操作：设$c_k \in C_k$，即c_k是一个侯选k_项集，c_{k-1}是c_k的一个$(k-1)$_项子集。由Apriori性质可知，任何非频繁的$(k-1)$_项集不可能是频繁k_项集的子集。即若$c_{k-1}\notin L_{k-1}$，则$c_k\notin L_k$，即侯选k_项集c_k应该从侯选k_项集的集合C_k中删除。

例　ElecTech公司是一个跨国公司，它的一个事务数据库TDB如表1所示，通过Apriori算法依次产生的侯选项集的集合C_i如表2所示(假设最小事务支持计数为2，即min_sup=2/10=20%%)。

表1　事务数据库TDB

TID	项集合
T001	a, b, d
T002	a, b, c, e
T003	a, c
T004	a, b, c
T005	a, c, d
T006	a, b, d, e
T007	a, b
T008	a, c, f
T009	b, c
T010	c, g

表2　侯选项集的集合C_i

C_1	C_2	C_3	C_4
$\{a\}$	$\{a, b\}$	$\{a, b, c\}$	$\varnothing$
$\{b\}$	$\{a, c\}$	$\{a, b, d\}$	
$\{c\}$	$\{a, d\}$	$\{a, b, e\}$	
$\{d\}$	$\{a, e\}$		
$\{e\}$	$\{b, c\}$		
$\{f\}$	$\{b, d\}$		
$\{g\}$	$\{b, e\}$		
	$\{c, d\}$		
	$\{c, e\}$		
	$\{d, e\}$		

2　算法改进

由表2可以看出，尽管运用Apriori性质可以压缩侯选k_项集，但对于频繁项集尤其是频繁2_项集的产生仍需要占用大量的存储空间。对频繁2_项集的产生，剪枝操作不起任何作用。若频繁1_项集的规模为$O(n)$，则侯选2_项集的规模将达到$O(n^2)$。例如，若有10^3个频繁1_项集，则将产生多达$C_{10^3}^2=10^3(10^3-1)/2\approx 10^6/2$个侯选2_项集。

2.1　散列技术

当扫描事务数据库，对侯选k_项集的集合C_k中的侯选k_项集计数时，同时产生每个事务的所有$(k+1)$_项集，通过散列函数把其散列到散列表中并计数。在连接产生侯选$(k+1)$_项集的同时考察散列表计数，若该计数对应的支持度小于min_sup，则把该$(k+1)$_项集从侯选侯选$(k+1)$_项集中删除，否则，继续进行剪枝操作。由于剪枝操作对侯选2_项集的产生不起作用，因此，在产生侯选2_项集的时候可以不进行剪枝操作。

侯选k_项集散列函数的构造为

$$\mathrm{hash}(x_1,x_2,x_3,\cdots,x_k)=\sum_{i=1}^{k}(\mathrm{order}(x_i)(2^r)^{k-i}) \mod (\mathrm{prime}(C_n^k\overline{E}\overline{D})=$$

$$(\mathrm{order}(x_1)(2^r)^{k-1}+\mathrm{order}(x_2)(2^r)^{k-2}+\cdots+\mathrm{order}(x_k)(2^r)^0)\bmod(\mathrm{prime}(C_n^k\overline{E}\overline{D})$$

式中　函数order(x_i)返回项x_i在C_1表中的编号，$r(r\in N)$是基规模度，n是侯选1_项集的个数$|C_1|$，C_n^k是没有支持度限制的侯选k_项集的组合数，$\overline{E}$ $(0<\overline{E}\le 1)$是事务项组合存在均度，$\overline{D}$ $(0<\overline{D}\le 1)$是事务项组合稠密均度，函数prime (x)返回不大于x的最大素数。

散列函数应该包含侯选k_项集中每一个项的特征，而侯选k_项集中的每一个项都可以在侯选1_项集的集合C_1表中找到，因此可以在C_1中给每一个项编号以标识其自身，用函数order(x_i)返回项x_i的这个编号。通常，C_1用一维数组存储，用其项集的地址来表示这个编号不仅简单而且效率很高，即order(x_i)= $i(i\in N)$或order(x_i)=$|C_1|-i+1(i\in N)$。

对每一个项的编号求和可以反映出这个项集的总体特征，但简单的求和容易产生散列冲突，这里把项的编号order(x_i)与$(2^r)^{k-i}$相乘再求和，与2的幂相乘可在计算机中通过移位操作实现，其效率很高($m\times 2^{100}$只需把m向左移动100位即可)。

定义1　称$r(r\in N)$为基规模度。这里以2^r为基，r的取值应以侯选1_项集的个数$|C_1|$为参考标准且r与$|C_1|$

成正比例关系，即$|C_1|$大则r较大，$|C_1|$小则r较小。设$|C_1|=n$，若$\mathrm{order}(x_i)=i(i\in N)$，则侯选$k$_项集的项编号与$(2^r)^{k-i}$的积的和值sum的范围为$1\times(2^r)^{k-1}+2\times(2^r)^{k-2}+\cdots+k\times(2^r)^0 \le \mathrm{sum} \le (n-k+1)\times(2^r)^{k-1}+(n-k+2)\times(2^r)^{k-2}+\cdots+n\times(2^r)^0$；若$\mathrm{order}(x_i)=n-i+1(i\in N)$，则侯选$k$_项集的项编号与$(2^r)^{k-i}$的积的和值sum的范围为$k\times(2^r)^{k-1}+(k-1)\times(2^r)^{k-2}+\cdots+1\times(2^r)^0 \le \mathrm{sum} \le n\times(2^r)^{k-1}+(n-1)\times(2^r)^{k-2}+\cdots+(n-k+1)\times(2^r)^0$。散列表的长度$\mathrm{length}=\mathrm{prime}(C_n^k\ \overline{E}\ \overline{D})$，主要由$C_n^k$决定。若length＜sum，则散列表占用的存储空间就会较小，同时存储空间的利用率也会较高。通常，取r=1。若$|C_1|$很大，可适当增大r的值。

若不考虑最小支持度限制，则侯选k_项集数$y(k)=C_n^k$ $(n=|C_1|)$，这是一个离散类抛物线(开口向下)型函数，当$k=|C_1|/2(|C_1|$为偶数)或$k=(|C_1|\pm1)/2(|C_1|$为奇数)时函数取得最大值。以$|C_1|=8$为例，$y(k)$与k的函数关系如图1曲线1所示。

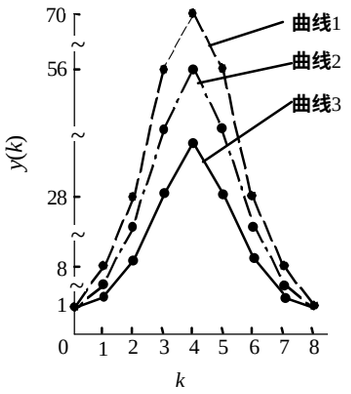

图1 $y(k)$与k的函数关系

定义2 事务项组合存在度(简称组合存在度)$E_i(1\le i\le|C_1|)$为由事务数据库推导出的侯选i_项集数与不考虑最小支持度限制而得到的侯选i_项集数的商。

例如，从表1的事务数据库可知：

$$E_1=|\{\{a\},\{b\},\{c\},\{d\},\{e\},\{f\},\{g\}\}|/C_7^1=1$$

$$E_2=|\{\{a,b\},\{a,d\},\{b,d\},\{a,c\},\{a,e\},\{b,c\},\{b,e\},\{c,e\},\{c,d\},\{d,e\},\{a,f\},\{c,f\},\{c,g\}\}|/C_7^2=0.619$$

$$E_3=|\{\{a,b,d\},\{a,b,c\},\{a,b,e\},\{a,c,e\},\{b,c,e\},\{a,c,d\},\{a,d,e\},\{b,d,e\},\{a,c,f\}\}|/C_7^3=0.257$$

$$E_4=|\{\{a,b,c,e\},\{a,b,d,e\}\}|/C_7^4=0.057,$$

$$E_5=E_6=E_7=0。$$

由此可知，E_i的值与各事务的数据项结构相关且$0\le E_i\le1$，$E_1=1$。

定义3 组合存在度E_i的均值称为组合存在均度，记为$\overline{E}$ $(0<\overline{E}\le1)$，以反映E_i的均态。受$\overline{E}$影响的$y(k)$与k的函数关系如图1曲线2所示。由于在对事务数据库进行扫描之前无法确定E_i及$\overline{E}$，因此，$\overline{E}$常由数据库管理员根据经验或已知情报估计。通常，可令$\overline{E}=1-\mathrm{min_sup}$。$\overline{E}$的大小不会影响算法的正确性，但会影响算法的效率。

定义4 事务项组合稠密度(简称组合稠密度)$D_i(1\le i\le|C_1|)$为由事务数据库推导出的满足min_sup的侯选i_项集数与由事务数据库推导出的侯选i_项集数的商。

例如，从表1的事务数据库可知：

$$D_1=|\{\{a\},\{b\},\{c\},\{d\},\{e\}\}|/|\{\{a\},\{b\},\{c\},\{d\},\{e\},\{f\},\{g\}\}|=5/7=0.714$$

$$D_2=|\{\{a,b\},\{a,c\},\{a,d\},\{a,e\},\{b,c\},\{b,d\},\{b,e\}\}|/|\{\{a,b\},\{a,d\},\{b,d\},\{a,c\},\{a,e\},\{b,c\},\{b,e\},\{c,e\},\{c,d\},\{d,e\},\{a,f\},\{c,f\},\{c,g\}\}|=7/13=0.538$$

$$D_3=|\{\{a,b,c\},\{a,b,d\},\{a,b,e\}\}|/|\{\{a,b,d\},\{a,b,c\},\{a,b,e\},\{a,c,e\},\{b,c,e\},\{a,c,d\},\{a,d,e\},\{b,d,e\},\{a,c,f\}\}|=3/9=0.333$$

$$D_4=D_5=D_6=D_7=0$$

由此可知，D_i的值与各事务的数据项相关且$0\le D_i\le1$。

定义5 组合稠密度D_i的均值称为组合稠密均度，记为$\overline{D}$ $(0<\overline{D}\le1)$，以反映D_i的均态。受$\overline{D}$影响的$y(k)$与k的函数关系如图1曲线3所示。由于在对事务数据库进行扫描之前无法确定D_i及$\overline{D}$，因此，$\overline{D}$常由数据库管理员根据经验或已知情报估计。若无法估计，可令$\overline{D}=1$。$\overline{D}$的大小不会影响算法的正确性，但会影响算法的效率。

函数prime(x)返回不大于x的最大素数，因为素数求模将减少散列冲突，可提高算法效率。

3.2 事务压缩技术

事务压缩技术的原理为不包含任何k_项集的事务不可能包含任何$(k+1)$_项集。若一个事务不包含任何k_项集，则该事务包含的项数必定小于k，同时必小于$k+1$，则该事务不包含任何$(k+1)$_项集。由此，在扫描事务数据库产生侯选i_项集$(i>k)$计数之前，先把该事务删除以提高扫描效率，在以后的操作则不需要该事务。

例　仍以ElecTech公司的事务数据库TDB为例，结合散列技术和事务压缩技术的Apriori算法依次产生的侯选项集的集合C_i如表3所示，事务数据库TDB的压缩过程如表4所示(假设最小事务支持计数为2，即min_sup=2/10=20%, $\overline{E}$ =1−20%, $\overline{D}$ =0.7,order(x_i)=i)。其中，用到的散列函数有

hash (x_1,x_2) = (order(x_1)×2+order(x_2)) mod (prime((1−20%)×0.7×C_7^2))=(order(x_1)×2+ order(x_2)) mod 11

hash (x_1,x_2,x_3) = (order(x_1)×2^2+order(x_2)×2+order(x_3)) mod (prime((1−20%)×0.7×C_7^3))=
(order(x_1)×2^2+ order(x_2)×2+order(x_3)) mod 19

hash (x_1,x_2,x_3,x_4) = (order(x_1)×2^3+order(x_2)×2^2+order(x_3)×2+order(x_4)) mod (prime(0.7×(1−20%)×C_7^4))=
(order(x_1)×2^3+order(x_2)×2^2+order(x_3)×2+order(x_4)) mod 19

表3　侯选项集的集合C_i

C_1	C_2	C_3	C_4
{*a*}	{*a* , *b*}	{*a* , *b* , *c*}	
{*b*}	{*a* , *c*}	{*a* , *b* , *d*}	∅
{*c*}	{*a* , *d*}	{*a* , *b* , *e*}	
{*d*}	{*a* , *e*}		
{*e*}	{*b* , *c*}		
{*f*}	{*b* , *d*}		
{*g*}	{*b* , *e*}		
	{*d* , *e*}		

表4　TDB的压缩过程

TID	项的集合	TID	项的集合	TID	项的集合
T001	*a* , *b* , *d*	T001	*a* , *b* , *d*	T001	*a* , *b* , *d*
T002	*a* , *b* , *c* , *e*	T002	*a* , *b* , *c* , *e*	T002	*a* , *b* , *c* , *e*
T003	*a* , *c*	T003	*a* , *c*	T004	*a* , *b* , *c*
T004	*a* , *b* , *c*	T004	*a* , *b* , *c*	T005	*a* , *c* , *d*
T005	*a* , *c* , *d*	T005	*a* , *c* , *d*	T006	*a* , *b* , *d* , *e*
T006	*a* , *b* , *d* , *e*	T006	*a* , *b* , *d* , *e*	T008	*a* , *c* , *f*
T007	*a* , *b*	T007	*a* , *b*		
T008	*a* , *c* , *f*	T008	*a* , *c* , *f*		
T009	*b* , *c*	T009	*b* , *c*		
T010	*c* , *g*	T010	*c* , *g*		

4　结 束 语

本文通过频繁项集的产生过程可以看出，散列技术能够有效压缩频繁项集尤其是频繁2_项集的大小，事务压缩技术能够有效压缩事务数据库的大小。尽管算法需要额外的空间存储散列技术表，但该表用一个一维整型数组即可表示，比侯选项集表要小得多，从而在减小存储空间的同时提高了Apriori算法的效率。

参 考 文 献

1　Agrawal R, Srikant R. Fast algorithms for mining association rules[C]. In Proceeding of the 20th International Conference on Very Large Databases. 1994, 487-499

2　Jong S P, Ming S C, Philip S Y. An effective hash based algorithm for mining association rules[C]. In Proceedings of the 1995 ACM SIGMOD International Conference On Management of Data. 1995, 24(2)：175-186

3　Jiawei H, Micheline K. Data mining: concepts and techniques[C]. Morgan, 2001, 149-158

编　辑　徐培红