一种基于FP阵列技术的频繁模式挖掘算法

谭 军[1,2] 卜英勇[2] 杨 勃[2]
(中南林业科技大学计算机学院 长沙 412006)[1] (中南大学机电工程学院 长沙 410083)[2]

摘 要 在FP-growth算法中,为了产生条件FP树,必须扫描FP树两次。提出一种新颖的FP阵列技术,将FP-tree数据结构与FP阵列有效地结合起来,直接从FP阵列得到频繁项的计数,从而省略了第一次扫描,节省了时间。尤其对于稀疏数据库,该算法在执行时间上比原FP-growth算法具有更优的性能。

关键词 频繁模式增长算法,频繁模式树,稀疏数据库,FP阵列
中图法分类号 TP311.13 **文献标识码** A

FP-array-based Improved FP-growth Algorithm

TAN Jun[1,2] BU Ying-yong[2] YANG Bo[2]
(College of Computer Science, Central South University of Forestry and Technology University, Changsha 412006, China)[1]
(College of Mechanical&Electrical Engineering, Central South University, Changsha 410083, China)[2]

Abstract In FP-growth algorithm, two traversals of FP-tree are needed for constructing the new conditional FP-tree. A novel FP-array technique was presented that greatly reduced the need to traverse FP-trees. A improved FP-growth algorithm was presented which uses the FP-tree data structure in combination with the FP-array technique efficiently. Experimental results show that the new algorithm works especially well for sparse datasets.

Keywords FP-growth algorithm, FP-tree, Sparse databases, FP-array

挖掘关联规则的核心是有效地挖掘频繁项集。挖掘算法主要分为两大类:一类以Apriori[1]算法为代表,采用宽度优先的策略,逐层产生频繁项集,其主要缺点在于需要多次扫描数据库,扫描的次数取决于最大频繁模式的长度,尤其对于一个大规模数据库,扫描要耗费大量时间;另一类以频繁模式增长(FP-growth)算法[2,3]为代表,采用频繁模式树(FP-tree)的数据结构,不用产生候选集。把数据库压缩到一个只存储频繁项的树结构,它与Apriori相比,只需要扫描数据库两次,第一次扫描统计所有频繁项的次数,第二次扫描构造初始FP-tree,包含原始数据库所有的频繁项集,挖掘数据库就变成了挖掘FP-tree。

学者们提出了很多改进的FP-growth算法。例如,Top-down FP-growth[4]算法按自上往下的顺序搜索FP-tree,不用产生条件模式基和子FP-tree,节省了时间和空间。Pei[5]等学者扩展了FP-tree结构,采用Hyper-结构,实验表明该算法针对很多类型的数据有较高的性能。PatriciaMine[6]算法使用类似于FP-tree的Patricia Tries数据结构表示频繁项集信息。本文提出一种新颖的FP阵列技术(FP-array)。把FP-tree数据结构与FP阵列技术有效地结合起来,提出了一种改进的FP-growth算法,记为FP-growth#。实验结果表明,针对稀疏数据库,本算法无论在执行时间上还是内存消耗和可扩展性方面都比FP-growth算法具有更优的性能。

1 FP-阵列技术

1.1 概念及构造

FP-growth算法的核心是在从原始数据库构造FP树后遍历FP树,从而构造新的条件FP树。大量的实验结果表明,算法执行时间的80%用于遍历FP树,因此,能不能减少遍历时间成为提高算法效率的关键。

可以通过使用一个简单的附加数据结构来达到这个目的。对于一个条件FP树 T_X 的项头表里的每个项 i,为了构造新的条件FP树 $T_{X\cup\{i\}}$,需要遍历 T_X 树两次。第一次遍历是在 $X\cup\{i\}$ 的条件模式基里发现所有频繁项,构造条件FP树 $T_{X\cup\{i\}}$ 的项头表。第二次遍历构造新的条件FP树 $T_{X\cup\{i\}}$。在建立 T_X 树的同时,构造一个频繁项对的阵列 A_X,作为 T_X 树的一个属性,通过 A_X 直接找出所有频繁项,从而忽略 T_X 树的第一次遍历。

定义 假设 T 是一个条件FP树,$I=\{i_1,i_2,\cdots,i_m\}$ 是 T 的项头表的项集。T 的频繁项对的阵列是一个 $(m-1)*(m-1)$ 的矩阵,矩阵的每个元素是相应的频繁项对的计数。显然,没有必要同时设置频繁项对 (i_j,j_k) 和 (i_k,i_j) 的计数,所以在矩阵里只给出频繁项对 (i_k,i_j) 的计数 $(k<j)$ 就可以了。

举例说明FP阵列的构造。假设最小支持度阈值是10%,在对原始数据库的第一次扫描后,得到频繁项:a:10,d:

到稿日期:2008-08-14 返修日期:2008-10-21 本文受国家自然科学基金项目(50474052)资助。
谭 军(1971-),男,博士生,讲师,主要研究领域为数据仓库、数据挖掘,E-mail:tanjun007@vip.sina.com;卜英勇 教授,博士生导师;杨 勃 博士生。

8,b:8,f:7,g:5,c:3,e:2。在对数据库的第二次扫描中,将构造 FP 树 $T_{\odot}$及 FP 阵列 $A_{\odot}$。FP 阵列将存储所有频繁项对的计数,每个元素就是一个频繁项对的计数,初始化为 0。例如 $A_{\odot}[a,b]$是项集$\{a,b\}$的计数。

在第二次扫描中,当处理每个事务时,首先提取事务中的所有频繁项,按项头表的项目顺序组成项集 I。然后把项集 I 插入到 $T_{\odot}$。如果频繁项对$\{i,j\}$包含在 I 中,$A_{\odot}[i,j]$的值递增 1。例如,对于第 3 个事务,$\{a,d,g,f\}$被提取(项 h 是非频繁的),并插入到 $T_{\odot}$中。同时,$A_{\odot}[a,d]$,$A_{\odot}[a,g]$,$A_{\odot}[a,f]$,$A_{\odot}[d,g]$,$A_{\odot}[d,f]$,$A_{\odot}[g,f]$的值都递增 1。在第二次扫描后,FP 阵列 $A_{\odot}$包含了所有频繁项对的计数,如图 1(a)所示。

在构造完 FP 树 $T_{\odot}$及 FP 阵列 $A_{\odot}$后,对于 $T_{\odot}$项头表的每个项目 i,递归调用 FP-growth 算法挖掘频繁项集。但是现在就不用遍历 $T_{\odot}$得到 i 的条件模式基的所有频繁项,$A_{\odot}$已经给出了 i 的所有频繁项。例如,检查 $A_{\odot}$的第三行,得出项 a,d,b 就是 f 的条件模式基的频繁项。因此,阵列 $A_{\odot}$省略了 $T_{\odot}$的第一次扫描,条件模式树 $T_{(i)}$可以直接从阵列 $A_{\odot}$初始化。

同样,对于条件 FP 树 T_X,在构造新的条件 FP 树 $T_{X\cup(i)}$的同时构造新的 FP 阵列 $A_{X\cup(i)}$。例如,如果构造条件 FP 树 $T_{(f)}$,那么同时构造的 FP 阵列 $A_{(f)}$将如图 1(b)所示。构造过程如下:从阵列 $A_{\odot}$得出在$\{f\}$的条件模式基里的频繁项是 a,b,d。顺着 f 的链接表,得到第一个节点$\{b,d\}$:1,插入到 FP 树 $T_{(f)}$中的同时 $A_{(f)}[b,d]$递增 1。将链接表的第二个节点$\{a,d\}$:1 插入到 FP 树 $T_{(f)}$中,同时 $A_{(f)}[a,d]$递增 1。将链接表的第三个节点$\{a,b\}$:2 插入到 $T_{(f)}$中,同时 $A_{(f)}[a,b]$递增 2。依次类推,直到 $T_{(f)}$构造完成,$A_{(f)}$也同时完成,将用于条件 FP 树的下一层构造。

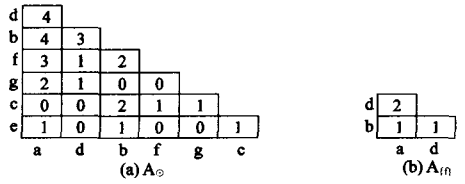

图 1 两个 FP 阵列例子

1.2 技术分析

假设初始 FP 树的频繁项的数目是 n,相关 FP 阵列的规模是$\sum_{i=1}^{n-1} i=n(n-1)/2$。从初始 FP 树构造的条件 FP 树的频繁项更少,相关的 FP 阵列的规模也减少。因为 FP 阵列是 FP 树的一个属性,所以随着 FP 树的存储空间释放,FP 阵列的存储空间也随之释放。

若数据库是稀疏的、大规模的,则构造的 FP 树就很大,并且没有很多共享的前缀,FP 阵列节省了遍历的时间,下一层次条件 FP 树可以直接初始化。在这种情况下,通过忽略第一次遍历节省的时间就比构造 FP 阵列所需的统计计数的时间要大得多。

但是,当数据库密集时,FP 树更加紧密。对于一个紧密 FP 树的每个项,其遍历是非常快的,而构造 FP 阵列时统计计数的时间可能更多。在这种情况下,构造阵列可能就不是个好办法。

数据集(条件模式基)随着递归的层次不同而改变。为了判断一个数据集是稀疏的还是稠密的,在每个 FP 树的构造过程中统计树每一层节点的数目。根据实验我们发现,如果树的上层四分之一部分的节点数目少于总数的 15%,数据集极有可能是稠密的。如果数据集是稠密的,我们就不必构造 FP 阵列。否则,我们就要为构造下一层次条件 FP 树而构造 FP 阵列。

2 一种改进的 FP-growth 算法

基于 FP 阵列技术,我们提出一种改进的 FP-growth 算法,记为 FP-growth#。在算法中,T 表示 FP-tree,其属性有基、项头表、FP 阵列。T. base 包含了项集 X,T 是 X 的条件 FP 树,T. FP-array 包含了 FP 阵列 A_X。算法细节描述如下。

```
Procedure FPgrowth#(T)
Input:一个条件 FP-tree:T
Output: 所有的频繁项集
方法:
1. If T 是单个路径 P
2. Then for 路径 P 中节点的每个组合(记作 Y)
3.        Output 项集 Y∪T. base,其支持度 support=Y 中节点的最小支持度;
4. Else for each i 在 tree 的项头表{
5.        Output Y=T. base∪{i},其支持度=i 的支持度
6.      If T. FP-array is not NULL
7.        从 T. FP-array,构造 Y 的新条件 FP 树的项头表
8.      Else 从 T 构造新项头表;
9.      构造 Y 的条件 FP-树 TY,及 FP-阵列 AY;
10. If TY 不为空集
11.   调用 FP-growth#(TY);
12. END
```

在算法中,第六行测试当前 FP 树的 FP 阵列是否为空。如果 FP 树对应的是一个稀疏数据集,它的 FP 阵列就不是空的。第七行直接从 FP 阵列构造新条件 FP 树的项头表。与 FP-growth 相比,节省了一次 FP 树的扫描。在第九行,为了判断是构造 FP 阵列还是设置 TY. FP-array=NULL,要同时统计树不同层次节点的数目。

3 实验分析

通过在很多数据集上运行 FP-growth * 和 FP-growth,比较了它们的性能。本文给出在一个虚拟数据集和一个真实数据集上运行的结果。虚拟数据集来自 IBM 研究中心的网站[7],它包含有 10 万条事务和 1000 个项目,平均事务长度是 40,平均频繁模式长度是 10,是稀疏数据集。真实数据集从文献[8]下载,非常稠密,所以即使对于最小支持度取非常高的值,也可以挖掘出很长的频繁模式。实验环境:Windows 2003 Server 操作系统,CPU 为 Intel 2.8GHz,内存 1G。算法用 C++实现。

图 2 显示两个算法各自的运行时间。由于 FP 阵列技术的使用,对于稀疏数据集,整体上 FP-growth# 比 FP-growth 要快很多。但当最小支持度非常低时,FP 树有很好的压缩性,FP 阵列就没有太大优势,结果被图 2 所证实。对于非常低的支持度,FP-growth * 和 FP-growth 的运行时间几乎一样。

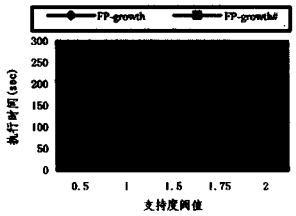

图 2 稀疏数据集上算法比较

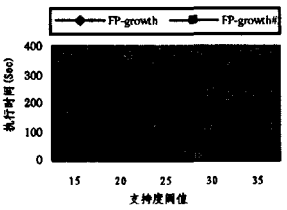

图 3 稠密数据集上算法比较

图 3 显示了两个算法在真实数据集上的运行时间。因为是非常稠密的数据集，FP 树有非常好的压缩性，两个算法的执行时间几乎一样，因此 FP 阵列技术不适合使用在这类挖掘中。

结束语 本文将 FP-tree 和 FP 阵列有效地结合起来，提出了一种改进的 FP-growth 算法。实验结果表明，对于稀疏数据库，本算法具有比 FP-growth 算法更优的性能。如何将 FP 阵列技术应用于最大频繁项集挖掘和闭频繁集挖掘，有待进一步研究。

参 考 文 献

[1] Agrawal R , Srikant R . Fast algorithm for mining association rules[A] // The International Conference on Very Large Data Bases[C]. 1994:487-499

[2] Han J, Pei J, Yin Y. Mining frequent patterns without candidate generation[A] // The 2000 ACM SIGMOD International Conference on Management of Data[C]. 2000:1-12

[3] Han Jiawei. 数据挖掘概念与技术[M]. 北京：机械工业出版社，2002:158-161

[4] Wang K, Tang L, Han J. Top down FP-growth for association rule mining[C] // Proc. of the 6th Pacific Area Conference on Knowleage Discovery and Data Mining(PAKDD). 2002

[5] Pei J, Han J, Lu H. H-mine: Hyper-structure mining of frequent patterns in large databases[A] // Proc. of IEEE Intl. Conference on Data Mining[C]. 2001:441-448

[6] Pietracaprina A, Zandolin D. Mining frequent itemsets using patricia tries[C] // Proceedings of the 1st Workshop on Frequent Itemset Mining Implementations. Melbourne, Fl, Nov. 2003

[7] http: // www. almaden. ibm. com/software/quest/resources/index. shtml

[8] http: // fimi. cs. helsinki. fi

(上接第 203 页)

表 1 手机网站的抽取结果

网站	图片正确率	价格正确率	型号正确率	召回率	时间(ms)
21cn	100% (100%)	100%(99%)	100% (100%)	99%(98%)	220 (317)
3553	99% (99%)	99% (98%)	99% (99%)	95% (93%)	332 (464)
pudou	100% (100%)	98% (97%)	100% (100%)	96% (94%)	136 (252)
shouji	99% (99%)	100% (100%)	100% (100%)	95% (95%)	253 (374)
younet	99% (99%)	100% (100%)	98% (98%)	95% (95%)	152 (231)

表 2 五金网站的抽取结果

网站	图片正确率	价格正确率	型号正确率	召回率	时间(ms)
chaolong	100%	100%	100%	99%	71
chinawj	100%	100%	100%	97%	164
Wjb2b	98%	99%	100%	94%	126
wujin. xd55	100%	99%	100%	96%	207
cn. easthardware	99%	98%	100%	95%	54

实验表明，无论商品信息在网页中以何种标签方式存储，基于 DOM 树路径的网页信息抽取方法都具有较高的精度和召回率，而且在效率上也有一定的提高。因为它不需要对 HTML 网页源文件的结构树进行简化和修剪处理，也不需要通过聚类算法来找到包含商品信息的关键块，只需要通过每个关键词组在 DOM 树中的路径来寻找最近祖先节点即可。因此基于 DOM 树路径的网页信息抽取方法是一种与站点结构无关的抽取方法，具有很强的适应性。

结束语 基于 DOM 树路径的网页信息抽取方法是对基于关键词组聚类和节点距离的网页信息抽取方法的一种简化和扩充。它利用对待抽取的网站的部分网页集来获取到关键词组，然后构建每个网页的 DOM 树结构，通过遍历 DOM 树结构来获取每个关键词在 DOM 树中的路径，这些路径的最近祖先节点包含了我们待抽取的信息块。同时对块外的信息，比如图片等等，通过比较图片节点与最近祖先节点之间的路径来获取它们之间的距离，然后利用基于节点距离的网页抽取方法来获取正确的信息。因此基于 DOM 树路径的网页信息抽取方法能够获取比较高的正确率和效率。同时该方法不局限于以〈table〉标签存放商品信息的网站，对其他各种标签存放的商品信息也能准确地抽取，因此具有很强的适应性和扩展性，在各种电子商务网站中都具有比较广阔的应用前景。

参 考 文 献

[1] Wong T-L , Lam W . Adapting Web Information Extraction Knowledge via Mining Site-Invariant and Site-Dependent Features [J]. ACM Transactions on Internet Technology, 2007, 7(1)

[2] Yang Pei, Zheng Qilun, Peng Hong, et al. A Stepwise Learning Approach to Automatic Discover Interest Data Block[C] // The third International Conference on Machine Learning and Cybernetics(ICMLC). 2004

[3] 邓健爽，郑启伦，彭宏，等. 基于关键词聚类和节点距离的网页信息抽取[J]. 计算机科学，2007，34:213-216

[4] [OL]. http: // www. w3. org/DOM/