快速开采最大频繁项目集*

路松峰, 卢正鼎

(华中理工大学 计算机学院,湖北 武汉 430074)

E-mail: songflu@public.wh.hb.cn

http://songfeng_lu.home.chinaren.com

摘要: 发现最大频繁项目集是多种数据开采应用中的关键问题.提出一种快速开采最大频繁项目集的算法 DMFI(discovery maximum frequent itemsets).该算法把自底向上和自顶向下的搜索策略进行了合并.通过其独特的排序方法和有效的剪枝策略,大大减少了候选项目集的生成,从而显著地降低了 CPU 时间.

关键词: 数据开采;知识发现;关联规则;最大频繁项目集;最小支持度

中图法分类号: TP311 **文献标识码**: A

发现频繁项目集是关联规则开采[1]和顺序模式开采[2]等数据开采应用中的关键技术和步骤.典型的发现频繁项目集的算法多为 Apriori[1]算法或者其变种,这些算法采用自底向上的搜索方法,穷举每一个频繁项目集.对一个维数为 L 的频繁项目集 F,由于 F 的每个子集都是频繁项目集,这就意味着算法必须产生 F 所有的 2^L-1 个子集,当 L 很大时,这将是一个 NP 难度问题.计算项目集的支持度是发现频繁项目集中最耗时的工作,降低候选项目集的数量是降低开销的最好的手段,由于最大频繁项目集(见第 1.1 节)已经隐含了所有频繁项目集,所以可把发现频繁项目集的问题转化为发现最大频繁项目集的问题.另外,某些数据开采应用(例如发现最小主键等)仅需发现最大频繁项目集,而不必发现所有的频繁项目集.因而发现最大频繁项目集对数据开采具有重大意义.本文提出一种快速发现最大频繁项目集的算法 DMFI(discovery maximum frequent itemsets).

1 相关概念

1.1 关联规则和最大频繁项目集

关联规则[1]的开采就是发现支持度和置信度分别大于用户指定的最小支持度 minsup(minimum support)和最小置信度(minimum confidence)的规则.支持度不小于 minsup 的项目集叫频繁项目集(frequent itemsets),反之,称为非频繁项目集(infrequent itemsets).项目集中项目的数量叫做项目集维数或长度,项目集 X 的支持度记作 $\sup(X)$.有关项目集具有如下性质:

性质 1. 如果 X 是频繁项目集,那么 X 的任何子集都是频繁项目集.

性质 2. 如果 X 是非频繁项目集,那么 X 的任何超集都是非频繁项目集.

如果频繁项目集 F 的所有超集都是非频繁项目集,那么称 F 为最大频繁项目集,所有 F 的集合称为最大频繁项目集合,记作 MFS(maximum frequent sets).显然,任何频繁项目集都是最大频繁项目集的子集,所以,发现所有频繁项目集的问题可以转化为发现所有最大频繁项目集的问题.

* 收稿日期: 1999-07-30; 修改日期: 1999-12-08

基金项目: 国家"九五"国防预研基金资助项目

作者简介: 路松峰(1968-),男,河南巩义人,博士生,讲师,主要研究领域为数据开采;卢正鼎(1945-),男,湖北武汉人,教授,博士生导师,主要研究领域为数据库,软件复用.

1.2 集合枚举树

用如图1所示的集合枚举树来描述项目集,可以方便地枚举所有可能的项目组合,从而数据开采过程可以转化为集合枚举树的搜索过程.图1表示 $D=\{a,b,c,d\}$ 的完全项目集合枚举树.树的每个结点 n 由两个项目集来表示,第1个项目集叫首(head),记作 $h(n)$,由树当前结点的枚举项目集表示;第2个项目集叫尾(tail),记作 $t(n)$,由当前结点的子结点的所有项目除去当前结点所包含的项目后排序组成.例如,对结点 a,$h(a)=\{a\}$,$t(a)=\{b,c,d\}$.把结点 n 的父结点记作 n_p,子结点记作 n_c.n_c 的生成方法是:$h(n_c)=h(n)\cup\{i\}$,$i\in t(n)$;$t(n_c)=\{j\mid j\in t(n),j>i\}$.例如,结点 a 有3个子结点 ab,ac 和 ad,对结点 ac 有 $h(ac)=h(a)\cup\{c\}=\{a,c\}$,$t(ac)=\{d\}$.

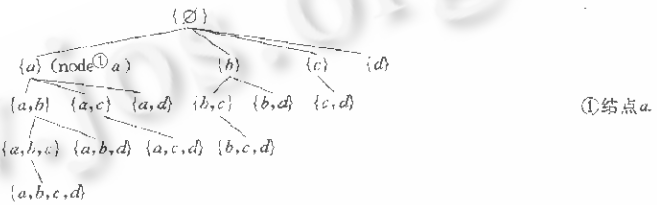


Fig.1 Complete enumeration tree for 4-dimension itemset $D=\{a,b,c,d\}$

图1 4维项目集合 $D=\{a,b,c,d\}$ 完全枚举树

另外,我们约定在进行第 k 次搜索时,所生成的任一候选项目集 A 可以表示为 $A_h\cup A_t$,其中 A_h 为 A 的前 k 个元素,A_t 为 A 的后 $L-k$ 个元素,L 为 A 的维数.

2 发现最大频繁项目集的算法DMFI

2.1 剪枝策略

为了减少生成不必要的项目集,发现频繁项目集的算法都利用性质1或性质2来减少候选项目集的数量.自底向上的搜索方法只能利用性质2,自顶向下的搜索方法只能利用性质1.为了加速搜索进程,DMFI算法采用自底向上和自顶向下相结合的搜索策略,利用每个搜索方向搜索到的信息对集合枚举树进行剪枝.

在DMFI算法中,我们设置了两个数组MFCS(maximum frequent candidate sets)和 C_k(candidate sets),分别用以保存自顶向下方向和自底向上方向上生成的候选项目集.在进行第 k 遍搜索时,C_k 中每个项目集的维数为 k,而MFCS包含枚举树第 k 层中所有结点首项目集的最长的超集,例如在图1中,当 $k=1$ 时,4个结点 a,b,c 和 d 的首项目集所对应的最长超集分别为 $\{a,b,c,d\}$,$\{b,c,d\}$,$\{c,d\}$ 和 $\{d\}$,此时,MFCS=$\{\{a,b,c,d\},\{b,c,d\},\{c,d\},\{d\}\}$.

如果在自顶向下的方向上发现某个频繁项目集 F,由于 F 的所有子集都是频繁项目集,且都不可能是最大频繁项目集,从而可以把 C_k 中是 F 子集的元素剪掉.反之,如果在自底向上的方向上发现某个非频繁项目集,由性质2可知它所有的超集都是非频繁项目集,从而可以对MFCS进行剪枝.由于第 $k+1$ 次的MFCS是由第 k 次的MFCS所生成(见第2.2节),所以对MFCS剪枝时不能简单地利用性质2.例如,如果 $\{a\}\in C_1$ 是非频繁项目集,若MFCS=$\{\{a,b,c,d\}\}$,虽然 $\{a,b,c,d\}$ 是非频繁项目集,但它的子集不一定是非频繁项目集,因而不可简单地把 $\{a,b,c,d\}$ 剪掉.C_k 对MFCS的剪枝方法如下:如果在第 k 遍搜索时,项目集 $A\in C_k$ 为非频繁项目集,$A=\{a_1,a_2,\ldots,a_k\}=h(A_p)\cup\{a_k\}$;对于任一 $m\in$MFCS,$m=m_h\cup m_t=\{m_1,m_2,\ldots,m_{k-1}\}\cup\{m_k\}\cup m_t$,由 C_k 的生成方法(见第2.2节)可知,一定存在某些 $m\in$MFCS,使得 $\{m_1,m_2,\ldots,m_{k-1}\}=h(A_p)$,并且 $\{m_k\}\cup$

$m_t \supseteq \{a_k\}$. 把 a_k 从 $\{m_k\} \cup m_t$ 中去掉即可.

2.2 候选项目集的生成

C_k 可由 MFCS 中的非频繁项目集生成,$C_k = \{m_h \cup \{i\} \mid m \in \text{MFCS}, i \in m_t, \sup(m) < \text{minsup}\}$. 对 MFCS 来讲,在进行第 k 遍搜索时,应包含枚举树第 k 层中所有结点首项目集的最长超集,所以对于任一个非频繁项目集 $m \in \text{MFCS}$,其生成下一层 MFCS 的方法如下:$m' = m'_h \cup m'_t$,其中 $m'_h = m_h \cup \{i\}, i \in m_t; m'_t = \{j \mid j \in m_t, j > i\}$.

2.3 排序策略

项目集的支持度越高,就越有可能是最大频繁项目集的一部分,所以在生成新的 MFCS 之前,要对原有的 MFCS 中元素 m 的尾部 m_t 重新排序,以便使支持度较高的子项目集出现在更多的新项目集中,这对尽早发现最大频繁项目集有重要意义. 对每一个 $m \in \text{MFCS}$,m_t 应按照 $\sup(m_h \cup \{i\})$ 的大小来排序,其中 $i \in m_t$. 例如,$m = \{a,b,c,d,e\} \in \text{MFCS}, k = 2, m_h = \{a,b\}, m_t = \{c,d,e\}$,如果 $\sup(\{a,b,c\}) > \sup(\{a,b,d\}) > \sup(\{a,b,e\})$,则 $\{a,b,c\}$ 更可能是最大频繁项目集的一部分,所以应当使 $\{a,b,c\}$ 尽可能多地出现在新的 MFCS 中,对 m_t 重新排序后 $m_t = \{e,d,c\}$,从而 m 变为 $\{a,b,e,d,c\}$,则新生成的项目集为 $\{a,b,e,d,c\}$,$\{a,b,d,c\}$ 和 $\{a,b,c\}$.

另外,应当对 MFCS 的元素按项目集的维数进行排序,其原因是,根据性质 1,按我们的排序策略,任何项目集的子集一定在该项目集之后出现,这样,当发现某个项目集为频繁项目集时,由于其子集不可能是最大频繁项目集,因而不必计算其子集的支持度,直接从 MFCS 中把其子集删除即可,这样可以降低 I/O 和 CPU 时间.

2.4 DMFI 算法

DMFI 算法按宽度优先的方法对集合枚举树进行双向搜索. MFS 用来保存最大频繁项目集,算法描述如下:

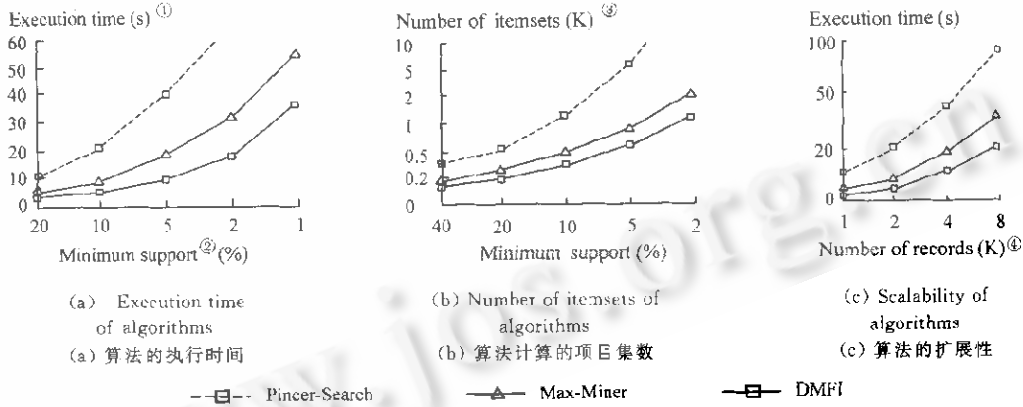
```
(1)  k=0, C_0={}, MFCS={1,2,...,n}, MFS={}   //假设项目交易集合 I={1,2,...,n}
(2)  do
(3)    delete t from m_t where m=m_h∪m_t and m∈MFCS and sup(m_h∪{t})<minsup
(4)    for x=1 to |MFCS| do
(5)      if |MFCS[x]|≤k then
(6)        MFS←MFCS[x]; delete m from MFCS where m⊂MFCS[x]
(7)      else
(8)        calculate sup(MFCS[x])
(9)        if sup(MFCS[x])≥minsup then
(10)         MFS←MFCS[x]; delete m from MFCS where m⊂MFCS[x]
(11)   create C_{k+1} and next MFCS   //方法见第 2.2 节
(12)   k++
(13) while C_k≠{}
(14) return MFS
```

3 算法分析与比较

从理论上讲,如果最长的最大频繁项目集的维数为 L,那么 DMFI 算法最多只需 L 次搜索,即可找到所有的最大频繁项目集,因为在第 L 次搜索时,对任意 $m \in \text{MFCS}$,$|m_h| = L$,如果搜索未结束,则必然存在 $m_h \cup \{i\}$ 为频繁项目集,而 $|m_h \cup \{i\}| = L+1 > L$,这与已知条件矛盾,所以最多只需搜索 L 遍.

其他计算最大频繁项目集的算法有 Max-Miner[3]和 Pincer-Search[4]. Max-Miner 突破了传统的自底向上的搜索策略,尽可能早地对项目集进行了剪枝.其缺陷是:① 未利用自顶向下的信息进行剪枝;② 未对 MFCS 进行适当的排序,产生了多余的候选项目集;③ 在第 k 次搜索时,由 DMFI 可知,维数不大于 k 的项目集一定是频繁项目集,不必计算它们,但 Max-Miner 将计算这些项目集的支持度. Pincer-Search 采用了自底向上和自顶向下的双向搜索策略,但其第 k 次的 MFCS 是由 $k-1$ 次的 MFCS 中的非频繁项目集去掉一个元素来生成,产生了过多的无用候选项目集,其 C_k 的生成采用类似于 Apriori 生成候选项目集的方法,对海量数据库来讲,将陷入 NP 难度的陷阱.

我们用 Powerbuilder 在 64M 内存、CPU 为 Pentium III-450M 的联想奔月 2000 机器上实现了 DMFI, Max-Miner 和 Pincer-Search 算法.我们利用 http://www.ics.uci.edu/~mlearn/MLSummary.html 上所提供的蘑菇数据库(mushroom database)来进行实验,该数据库有 8 124 条记录,记录了蘑菇的 23 种属性,每种属性有 2~12 个枚举值.图 2(a)显示了不同的最小支持度(分为 5 档:20%,10%,5%,2%,1%)下算法的性能变化,实验结果表明,最小支持度越低(候选项目集将越多),CPU 耗费时间越多;同时我们看到,DMFI 算法的执行时间比其他两种算法要少.图 2(b)显示了在不同的最小支持度(分为 5 档:40%,20%,10%,5%,2%)下算法要计算的项目集的数量,可以看出,DMFI 算法计算的项目集将更少.为了显示在不同数据量下算法的扩展性,我们随机从范例数据库中抽取了 4 个不同的记录数(1000,2000,4000 和 8000)进行测试,固定最小支持度为 2%,结果如图 2(c)所示,因为 DMFI 算法计算较少的项目集,因而具有更好的扩展性.



(a) Execution time of algorithms (a) 算法的执行时间

(b) Number of itemsets of algorithms (b) 算法计算的项目集数

(c) Scalability of algorithms (c) 算法的扩展性

①执行时间(秒),②最小支持度,③项目集数(千),④记录数(千).

Fig. 2

图 2

4 小 结

本文提出了一种有效的快速发现最大频繁项目集的算法 DMFI,有效地把自底向上和自顶向下的搜索策略进行了合并.理论和实验结果表明,相对于其他发现频繁项目集的算法,DMFI 具有更优越的性能.DMFI 为海量数据库发现频繁项目集和仅需发现最大频繁项目集的数据开采应用提供了一种有效而快速的算法.

References:

[1] Agrawal, R., Srikant, R. Fast algorithms for mining association rules. In: Bocca, J. B., Jarke, M., Zaniolo, C., eds.

Proceedings of the 20th International Conference on Very Large Databases. San Francisco: Morgan Kaufmann Publishers, 1994. 487~499.

[2] Agrawal, R., Srikant, R. Mining sequential patterns. In: Yu, P.S., Chen, A.L.P., eds. Proceedings of the 11th International Conference on Data Engineering. Los Alamitos, CA: IEEE Computer Society, 1995. 3~14.

[3] Bayardo, R. Efficiently mining long patterns from databases. In: Haas, L.M., Tiwary, A., eds. Proceedings of the ACM SIGMOD International Conference on Management of Data. New York: ACM Press, 1998. 85~93.

[4] Lin, Dao-I, Kedem, Z.M. Pincer-Search: a new algorithm for discovering the maximum frequent set. In: Schek, H.J., Saltor, F., Ramos, I., *et al*, eds. Proceedings of the 6th European Conference on Extending Database Technology. Heidelberg: Springer-Verlag, 1998. 105~119.

Fast Mining Maximum Frequent Itemsets*

LU Song-feng, LU Zheng-ding

(*College of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan* 430074, *China*)

E-mail: songflu@public.wh.hb.cn

http://songfeng_lu.home.chinaren.com

Abstract: Discovering maximum frequent itemsets is a key problem in many data mining applications. In this paper, the DMFI (discovery maximum frequent itemsets) algorithm which combines the bottom-up and top-down searches is proposed to solve this problem. Using the unique ordering method and efficient pruning strategy, the number of candidate itemsets is greatly decreased, therefore CPU time is reduced remarkably.

Key words: data mining; knowledge discovery; association rule; maximum frequent itemset; minimum support

* Received July 30, 1999; accepted December 8, 1999

Supported by the Defence Pre-Research Project of the National 'Ninth Five-Year-Plan' of China