分层相联规则中求强项集的并行算法*)

A Parallel Algorithm for Finding Large Itemsets in Multiple level Association Rules

叶阳东

(郑州大学计算机科学系 郑州450052)

摘 要 Finding large itemsets is a key problem for mining multiple level association rules in large database of transaction. In this paper we present a parallel algorithm ML_TALA that is more efficient than existing parallel multiple_level association rule mining algorithms. First, the algorithm scans an encoded transaction database once to find each large l itemset in all concept levels, then uses the large l itemsets to filter the database, so it can produce the thoroughly filtered transaction database TA.

关键词 Association rule, Large itemset, Coocept hierarchy, Transaction database

相联规则发现算法的研究在提高算法效率、发现多种形式的规则两方面都取得了较快的发展,但较多的研究工作是围绕着具体事务的关联性的,也就是对概念级相联规则的研究。近期 J. Han 在算法中引入面向属性的概念分层树,达到对事务进行一般化的目的,可进行多层次概念相联规则的发现。本文所给出的算法,一方面仍沿用算法 ML_T2LA[1]中的并行策略和编码技术,通过对概念分层树的编码,使得事务基本项的编码中直接含有其高层一般化概念的信息,达到对事务数据库进行代码化,这就为并行求得多层相联规则的强项集奠定了良好的基础;另一方面本算法在一次扫描事务数据库并行求得所有概念层的强一项集后,利用这些强项集对事务数据库进行了过滤,这样在求以后的多项集时,相对于 J. Han 的 ML_T2LA、ML_T1LA 算法[1]来说,效率有所提高。

一、基础知识

事务 事务基本项集 $I=(i_1,i_2,\cdots,i_m)$是由 m 个不同点组成的集合,i_k 称为事务基本项。事务是 I 上的一个子集,也就是由一组点 $i_j,i_k,\cdots,i_p\in I$ 构成,每个事务均有一个唯一标识符 Tid。不同事务一起构成了相联规则发现的事务数据库 D。

概念分层 许多问题的研究和分析焦点并不是基本事务之间的相互关联性,而可能是对某类事务或满足于一定条件的事务属性之间的关联性进行分析,这可由领域专家用概念分层结构对事务中的基本项进行一般化,使关联性的分析随具体的概念分层分布在各个层上。概念分层根据一般到特殊的原则,将一般化的概念分多层次按树状结构进行组织,树的最高层是最一般化的概念,树的叶子结点是最基本的事务项。本算法中的概念分层树是平衡的,关联性分析是针对同层概念而进行的。

项集 X 的支持度(Support) 根据概念分层,将事务基本项集 I 中所有结点的先驱结点都加入以后的项集称之为 I 的扩展项集 EI。Support(X)表示项集 X 的重要性,其中:项集 $X\subset EI$,并且 X 中的项是同层概念。很显然,若 Tid 支持 I 中基本项 i_k,则必然支持 i_k 的先驱结点。若 B=D 中支持项集 X 的事务数量,A=D 中所有事务的数量,则项集 X 的支持度 Support(X)=B/A。

最小支持度 发现任务所要求的项集的最小支持度。只有支持度大于等于最小支持度的项集才有可能在相联规则中出现,这些项集称之为强项集(large itemset)。最小支持度可根据具体问题进行分层定义,很显然高层的支持度要大些,最底层的叶子结点的支持度最小。

规则置信度(Confidence) 规则可靠性的度量。

*)本文得到国家自然科学基金项目和河南省自然科学基金项目的部分资助

对于规则 R:X⇒Y,其中 X、Y⊂EI 并且 X、Y 是同层概念项集,规则 R 的置信度为:confidence(R)=support(X∪Y)/support(X)。

相联规则 形式是 X⇒Y{F},其中 X、Y⊂EI 并且 X、Y 是同层概念项集,X∩Y=∅。规则中的 X 称为先决条件,Y 称为规则的结果,F 为置信度。若 X、Y⊂I,所发现的规则为面向事务的基本规则,否则所发现的规则就是面向概念分层树的同层一般化概念相联规则。

二、事务数据库的代码化

本算法是通过对概念分层树进行编码,达到对事务数据库进行代码化。算法中的编码技术使得事务的编码直接含有事务的分类信息和其一般化概念的信息。例如:在某超级市场的事务数据库中,对食品类的物品所做的概念分层树如下图所示:

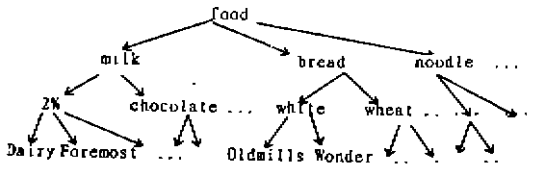

图1 食品相关的概念分层树

图中的结点 milk、bread、noodle、2%、chocolate、white、wheat 的编码分别为1＊＊、2＊＊、3＊＊、11＊、12＊、21＊、22＊,树中的叶子结点为事务的基本项属于项集 I,分别为 Dairy、Foremost…、oldmilla、Wonder…,编码为111、112…、211、212…。用以上的数字序列编码对事务数据库进行代码化后的数据库在本算法中用 T 表示,它具有以下的特点:

1. 事务基本项的编码既含有分层信息又含有其一般化概念的编码信息,这样在求不同层相联规则时不用再查询概念分层树。例如,基本项编码1231,说明所分析问题的概念分层树的有效层数是4,该结点的先驱结点的编码分别是123＊、12＊＊、1＊＊＊。

2. 代码化后的数据库有利于不同层项集支持度的计算。对于事务 TID 所支持的不同层项集,可采用至下而上的方法一次性求出。例如,事务〈TID_k,{112,122,231}〉所支持的高层概念项有:11＊、12＊、23＊、1＊＊、2＊＊。

3. 代码化后的事务数据库,进行弱项和小事务的过滤很方便。

三、发现分层相联规则的步骤

针对具体问题的要求和概念分层树,首先进行事务数据库的预处理和代码化,再根据各层的最小支持度 minsup[1..max_level],求出各层的强项集 LL[1..max_level],最后找出满足于最小置信度的各层相关联的规则并解释输出相应的规则,具体可分为如下几步:

步骤1 预处理与发现任务有关的数据。根据具体问题的要求对数据库进行相应的操作,从而构成规格化后的事务数据库 D。

步骤2 准备相应问题的概念分层并按上述的编码技术进行编码,从而产生代码化后的事务数据库 T。

步骤3 针对 T,用 ML_TALA 算法求出所有各层的强项集 LL[1..max_level]。若某层概念项的数量是 m,则该层可能的项集数量就是2^m个,由此可见分层相联规则中 LL[1..max_level]的求得是本发现算法的核心问题。

步骤4 用各层的强项集 LL[1..max_level]和各层的最小支持度 minconf[1..max_level],求出各层的相联规则 Rset[1..max_level]。

步骤5 解释并输出 Rset[1..max_level]。

说明:步骤3中,由于大型事务数据库中不同项集的数量非常多,若对所有项集都进行支持度的计算,几乎是不可能的。因此在求项集时,本算法为了最大可能地减少项集的组合,在产生强项集时仍采用 Apriori Algorithm 中的方法[2],其遵循原则是:任何强项集的子集都是强项集;任何弱项集的超集都是弱项集。这样本算法在求第 i 层的强项集 LL[i]时,首先求出该层项数为一项的强项集 L[i,1],再由 L[i,1]产生项数为二的候选项集 C2,扫描数据库计算支持度求出 L[i,2],依次类推产生 Ck,扫描数据库求出 L[i,k]。另一方面,本算法在力求对项集进行充分剪支的同时,尽最大可能地使用一次扫描事务数据库的信息,也就是说在一次扫描数据库的同时,并行产生所有层的强一项集,再扫描数据库产生所有层的强两项集,依次类推,最后并行产生各层项数最多的强项集。由此可见,本算法扫描数据库的次数是最少的。

四、并行算法 ML_TALA 与比较

本文中的算法在使用现有编码技术的基础上,

根据概念分层树在一次扫描事务数据库的过程中，并行求得各层项数相同的强项集。本算法和 J. Han 的 ML.T2LA、ML.T1LA 算法的不同之处是：在并行求得所有概念层的强一项集后，利用这些强项集对事务数据库进行过滤，从而产生高度过滤后的事务数据库 TA，这样在求以后的多项集时效率必然要高一些。

算法输入：(1)代码化的事务数据库 T〈TID，Itemset〉；(2)各层的最小支持度 minsup[1..maxlevel]。

算法输出：各层强项集的集合 LL[1..maxlevel]。

算法ML.TALA 的描述　(算法的语法类似 C 和 Pascal，其中，max.level：最大的层数，max.length：最大的项数)：

```
(1) for all transaction t ∈ T do
(2) for (i:=1;i<max.level;i++) do
(3) do support 1.itemset(i);//对 t 所支持的各层一项集进行计数
(4) for(i:=1;L[i,1]<>∅ and i<max.level;i++) do
(5) get(L[i,1]);//产生所有层次的强一项集
(6) TA:=get.filtered.t.table(T,L[1..max.level,1]);//由所有层次的强一项集，一次性过滤 T，产生 TA
(7) for (k:=2;k<max.length;k++) do
(8) {for(i:=1;i<max.level;i++) do
(9) if L[i,k-1]<>∅ then
(10) CK[i]:=gen.candidate.set(L[i,k-1]);//求各层的候选项集
(11) for each transeaction t ∈ TA do
(12) for (i:=1;i<max.level;i++)do
(13) if L[i,k-1]<>∅ then
(14) {Ct[i]:=gen.subsets(Ck[i],t);
(15) for each candidate c ∈ Ct[i] do support++
(16) };//对 t 所支持的各层 k 项集进行计数
(17) for (i:=1;i<max.level;i++) do
(18) if L[i,k-1]<>∅ then
(19) L[i,k]:={c|c ∈ Ck and c.support>=minsup[i]}
(20) else L[i,k]:=∅;
(21)
(22) for (i:=1;i<max.level;i++) do
(23) LL[i]:=UkL[i,k] //求各层的强项集
(24) }
```

说明：(10)中 Ck[i]的产生仍采用 Apriori Algorithm 中的方法[2]，具体的过程是：

第一步：对 L[i,k-1]中的每一个项集按项进行排序；

第二步：对于 L[i,k-1]中不同的项集 L.k-1.q 和 L.k-1.p 结合产生 Ck 的方法描述如下：

```
insert into Ck[i]
select p.item1,p.item2,…,p.itemk-1,q.itemk-1
    from L.k-1.p,L.k-1.q
    where p.item1=q.item1,p.item2=q.item2,…,p.itemk-2=q.itemk-2,p.itemk-1<q.itemk-1
```

本文的 ML.TALA 算法和 ML.T1LA、ML.T2LA 算法的比较：J. Han 的 ML.T2LA 算法仅在求得 L[1,1]后，利用 L[1,1]进行一次过滤产生 T2，由此可见 T2相对于 TA 来说，精简的程度不足；而算法 ML.T1LA 是无过滤的并行算法。因此，与 ML.TALA 算法相比，这两个算法在扫描数据库进行支持度的计算时，CPU 和 IO 的负载将要大一些。

算法 ML.TALA 和非并行算法 ML.T2L1的比较：算法 ML.T2L1除了在事务数据库的过滤方面不足以外，它是采用逐层多次扫描事务数据库求得强项集，这样 ML.T2L1总共扫描 T1：2次、扫描 T2：$\sum_i k_i - 1$次，其中，k_i 是第 i 层非空强项集的最大项数；而 ML.TALA 算法总共扫描 T：2次、扫描 TA：$\max(k_1..k_{maxlevel})-1$次。由此可见本算法比 ML.T2L1算法扫描数据库的次数要少得多。但是，要在一次扫描数据库并行求各层的强项集，必然要求很大的存储空间去存放 Ck[1..maxlevel]，这可能引起页转储，从而增加了 IO 的负载。

结束语　由于本算法采用了并行策略求强项集，大大减少了扫描事务数据库的次数，再加之利用并行求得各层强一项集对数据库进行了充分的过滤，使得发现算法的效率有了明显的改善。但该算法仅适用于同层相联规则的发现，研究适用范围更广的相联规则发现算法是我们今后一段时间所要做的工作之一。

参考文献

[1] J. Han, Y. Fu. Discovery of Multiple-Level Association Rules from Large Databases. In Proc. of the 21th Intl. Conf. on VLDB, Swizerland, 1995

[2] R. Agrawal, R. Srikant, Fast Algorithms for Mining Association Rules. In Proc. of the 20th Intl. Conf. on VLDB, Santiago, Chile, 1994