第37卷 第16期 计 算 机 工 程 2011年8月
Vol.37 No.16 Computer Engineering August 2011

·软件技术与数据库· 文章编号：1000—3428(2011)16—0042—03 文献标识码：A 中图分类号：TP311.13

一种改进的多层关联规则挖掘算法

唐 辉，吴明礼，贺玉明

(北方工业大学信息工程学院，北京 100144)

摘 要：在传统的多层关联挖掘算法中，概念层次树只提供项目的概念分层信息，导致项目查找效率不高。为此，提出一种改进的多层关联规则挖掘算法。在概念层次树的节点中增加2个域，以获取各层的频繁1-项集，通过增加1个基于Hash表的辅助存储结构，以提高项目的查找效率。实验结果表明，与传统算法相比，改进算法的挖掘时间能缩短10%左右。

关键词：多层关联规则；数据挖掘；概念层次树；频繁模式树

Improved Multi-level Association Rule Mining Algorithm

TANG Hui, WU Ming-li, HE Yu-ming

(College of Information Engineering, North China University of Technology, Beijing 100144, China)

【Abstract】 In the traditional association rule mining algorithm, concept hierarchy tree only provides concept hierarchy information of items, and the efficency of searching items is not high in this tree structure. This paper presents an improved multi-level association rule mining algorithm. It adds two fields in its nodes to help to accquire frequent 1-itemsets. It employs an auxiliary storage structure that is based on Hash table, which enhances efficency of searching items. Experimental result shows that mining time of the imrpoved algorithm is reduced by about 10% than before.

【Key words】 multiple-level association rule; data mining; concept hierarchy tree; Frequent Pattern(FP) tree

DOI: 10.3969/j.issn.1000-3428.2011.16.014

1 概述

关联规则挖掘是从大量数据中发现数据项集之间潜在关联或相关联系的一种技术。1993年，Agrawal等人首次提出Apriori算法，此后研究者们又提出如AprioriTid、FP-growth[1]等挖掘算法，这些算法的重点仍局限于在某个单一的概念层上进行频繁模式的挖掘。随着信息技术的发展，人们希望通过数据挖掘能够进行更高层次的分析，以便可以更好地利用这些数据[2]。一方面，较高概念层发现的关联规则能提供普遍意义的知识；另一方面，对于许多应用来说，特别是在电子商务的应用中，在低层或原始层的数据项之间很难找出强关联规则。

多层关联规则挖掘的方法仍基于“支持度-置信度”这一框架，通常采用自顶向下策略，从最高概念层向低层次方向进行挖掘。对每个概念层可以采用发现频繁项目集的任何算法找出该层的全部频繁项集。在多层关联规则挖掘算法中，以Cumulate和ML-T2L1最为著名。这2个算法都是通过对Apriori算法进行扩展得到的，因此，存在着与Apriori算法相同的缺陷。Ada-FP[3]是基于FP-growth算法的多层多维频繁模式挖掘算法，采用了多支持度约束。但如果原始数据只是最低抽象层的项，则无法挖掘出隐藏在层间的关联规则。研究者们还提出了许多其他的改进算法，文献[4]提出了一种多支持度挖掘算法，文献[5]提出了一种加权的关联规则挖掘方法。本文提出一种改进的多最小支持度的多层关联规则挖掘方法。

2 改进的多层关联规则挖掘算法

2.1 多层关联规则的挖掘原理及步骤

多层关联规则是一种基于概念分层[6]的关联规则挖掘方法，概念分层是一种映射，它将低层概念映射到更一般的高层概念。概念层次结构通常使用概念树表示，树的节点表示概念，树枝表示偏树，按从一般到特殊的顺序以偏序的形式排列。树中高层概念是低层概念的概括，树根是概念可能取值的最一般的描述，树叶是概念的具体描述。

多层关联规则作为单层关联规则的拓展，其挖掘原理和方法与单层关联规则基本相同，先挖掘出每一层的频繁模式，再挖掘出交叉层的频繁模式。根据Han的发现[4]，在不同的概念层上，一般应设置不同的最小支持度和最小置信度阈值，且支持度应逐层递减。

设D为事务集合，x_i和y_i分别表示第i层的最小支持度和最小置信度，A和B为事务集合D中2个项集，$\wedge$表示合取运算，则具有如下的定义。

定义1 项集A在第i层上是频繁的项集，当且仅当$x(A/D)\geqslant x_i$。

定义2 规则$A=>B$是第i层上的强关联规则，当且仅当$x(A\wedge B/D)\geqslant x_i$，$y(A\wedge B/A)\geqslant y_i$。

定义3 规则$A=>B$是第i层与第j层上的交叉关联规则，且j是比i较低的抽象层，当且仅当$x(A\wedge B/D)\geqslant x_j$，$y(A\wedge B/A)\geqslant y_j$。

定义4 设A是一个频繁模式，若其同时包含项目b和b的祖先c，则称A为冗余频繁模式。

2.2 电子商务数据的特点

在电子商务交易数据中，所有的商品是根据一定的分类标准存放的。这些分类标准是有层次关系的，而这些带有层次关系的分类标准就形成了一棵商品的概念层次树。

一个事务数据库中可以存在多个概念层次树，图1为某服装销售公司的一棵商品概念层次树。由于每条交易所包含的商品都属于某个具体的小类，因此都能映射到该商品层次树上。

作者简介：唐 辉(1977－)，男，硕士研究生，主研方向：数据挖掘；吴明礼，讲师、博士；贺玉明，硕士研究生

收稿日期：2011-02-28 **E-mail：**tanghuiluck@gmail.com

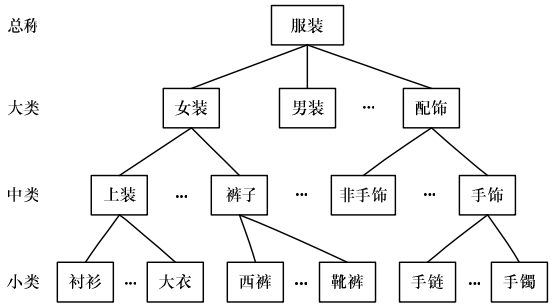

图 1　服装商品概念层次树

2.3　改进算法的策略

本文主要是从以下 2 个方面进行改进：

(1)对概念层次树的改进。传统的概念层次树主要由一个根节点和作为根节点孩子的子树集合组成，树中的每个节点包含 3 个域：节点名，父节点及孩子节点链。概念层次树与 FP(频繁模式)树的区别：每个项目在概念层次树只出现一次，而在 FP 树中则可以重复出现多次，还有就是概念层次树没有项目头表。由于多层关联规则的挖掘过程中会频繁地访问概念层次树，因此提高概念层次树的访问效率将有助于改善挖掘的效率。

1)在节点中再增加 2 个域：节点层数和节点计数。节点层数用来记录该节点所在的层，在创建概念层次树时直接赋上对应的值，于是当遍历树中的节点时就可以立刻知道每个节点所在的层数。节点计数有 2 个作用：①在获得频繁 1-项集时，用来统计该节点在所有事务中出现的次数。当判断某个节点是否为频繁项时，就可以直接根据该节点的节点计数与所在层数的最小支持数进行比较。②根据所获得的频繁 1-项集，在创建 FP 树的项目头表时，用于记录该项目在项目头表中位置。

2)增加一个基于 Hash 表的辅助存储结构。Hash 表是一种根据要访问元素关键字的值，通过哈希函数(一种将关键字映射到存储空间中特定位置的函数关系)计算出该关键字所存储的位置，然后直接进行访问的数据结构，它的查找时间复杂度为 $O(1)$。这种辅助存储结构以“键-值”对的方式存储信息，是一个数组和链表的结合体。“键”用于保存项目名称，“值”用于指向概念层次树中与该键同名的节点。当往这个结构插入项目时，先根据项目名计算出它的 Hash 值，再根据这个 Hash 值得到这个项目在数组中的位置，如果数组该位置上已经存放有其他项目了，那么在这个位置上的项目将以链表的形式存放，反之则直接将项目放到该位置上。在挖掘过程中，当需要查找某个项目或其祖先项时，就可以从这个结构中快速地访问到对应的节点。

(2)对 FP 树的改进。在 FP 树的创建过程中，会频繁地从项目头表中进行查询。由于项目头表需要按支持度降序顺序存储，只能采用顺序存储结构。顺序存储结构的查询效率较低，其查找时间复杂度为 $O(n)$，但可以进行随机读取。

本文先用一个数组依次存储排序后的全部 1-频繁项集作为项目头表，然后把项目在项目头表中的位置存储到概念层次树上对应节点的节点计数上。当需要从项目头表中查询项目时，先以这个项目为键在辅助存储结构中查找，取出该项目在项目头表中的位置，然后根据这个位置再从项目头表中随机读出。通过这样的改进，既保证了项目头表的顺序结构，又利用了 Hash 表的优点，可以快速查找频繁项目，从而提高了 FP 树的创建效率。

2.4　改进算法的步骤

2.4.1　频繁 1-项集的生成

在频繁 1-项集的生成过程中，与传统方法不同的是，在遍历事务数据库中的每条事务时，根据概念层次树进行扩充，把每个项的所有祖先项都添加进来，去掉重复项。然后遍历剩余的每个项，在概念层次树上找到对应的节点并让节点计数加 1。遍历完事务数据库后，概念层次树上的每个节点的节点计数就为该节点的支持数。本文采用层交叉单项目过滤策略，即第 i 层上的项目被考察，当且仅当它在(i–1)层的父节点是频繁的。从概念层次树根开始广度优先遍历概念层次树，由于每个节点上都有层数信息，就可以直接与对应层的最小支持度比较，如果节点的节点计数大于或等于对应层的最小支持度阈值，则保留，否则删除掉该节点及其所有的孩子节点。算法伪代码如下：

```
getFreq1Item(D, minSups, CT)
输入 事务数据库 D，各层最小支持度数组 minSups，概念层次树 CT
输出 频繁 1-项集 L
(1)创建用于存储扫描结果的 L;
(2)for each(Transaction t in D)
(3)    for each(Item e in t)
(4)        从 CT 中找到 e 的全部祖先，加入到 t 中;
(5)        删除 t 中重复的祖先项;
(6)    end for;
(7)    for each (item e in t)
(8)        在 CT 中查找节点 e，该节点的节点计数加 1;
(9)    end for;
(10)end for;
(11)把 CT 的根节点的孩子节点全部添加队列 Q 中;
(12)while(Q 不为空)
(13)    弹出队顶元素 x；与该元素所在层的最小支持度比较;
(14)    if(x 的节点计数≥所在层的最小支持数)
(15)        x 添加到 L 中，x 的全部孩子节点添加到 Q 中;
(16)    else 删除 x 节点及所有孩子节点;
(17)end while;
```

说明：这样的改进既避免了大量非频繁项的比较，又可以很方便获得各层的频繁 1-项集，也不用预先对事务进行编码或扩充。在获得各层的频繁 1-项集的同时还减掉概念层次树中的非频繁项，减小概念层次树的节点数量，提高了查询效率。

2.4.2　FP 树的创建

创建项目头表。先对 L 按支持数降序排序，然后遍历 L 创建项目头表，同时把项目在项目头表中的位置存储到概念层次树上对应节点的节点计数中。这样就避免了在项目头表中进行查询，大大提高了查询效率。

创建 FP 树。在创建 FP 树过程中，需要对事务数据库中每条事务进行扩充，去掉非频繁项，对剩余的项再按项目头表中的顺序排列。由于在获得频繁 1-项集后，概念层次树上只剩下频繁项，而且每个节点上还存储了该项目在项目头表中的位置，因此在对事务数据库中的事务进行处理和扩充时，直接从概念层次树上进行查找即可，不用从项目头表中进行查询。具体的算法伪代码如下：

```
buildFPTree(D, itmTB, CT)
输入 事务数据库 D，项目头表 itmTB，概念层次树 CT
输出 FP 树 T
(1)创建一棵只包含根节点的 FP 树 T，并把根节点置为当前父节点;
```

(2)for each(Transaction t in D)

(3) for each(Item e in t)

(4) 在 CT 中查找 e，如果不存在，则为非频繁项，直接舍弃；

(5) 从 CT 中找到 e 的全部祖先，把它们加入到 t 中；

(6) 删除 t 中重复的祖先项；

(7) 将剩余项按照项目在 itmTB 中的顺序排列；

(8) end for;

(9) for each(Item e in t)

(10) if (T 的当前父节点的孩子节点中存在项目 e)

(11) 将 T 中的节点名为 e 的节点的支持数增加 1；

(12) 将 T 中的节点名为 e 的节点置为当前父节点；

(13) else

(14) 将 e 插入 T 中；

(15) 将新插入的节点 e 连接到 itmTB 中 e 的同名节点链上；

(16) 将 T 中新插入的节点 e 置为当前父节点；

(17) end for

(18)end for

说明：在对事务的扩充过程中，概念层次树既可以帮助滤掉非频繁项，又能获得项目在项目头表中的位置信息，然后直接按这个值进行排序，大大减少了项目的查询次数。

2.4.3 挖掘关联规则

频繁模式的生成过程与传统的 FP-growth 算法没有什么区别，由于挖掘出的频繁项集中可能包含了不同概念层的项，因此需要进行冗余检查，避免产生过多的冗余关联规则。条件子树的生成与挖掘过程也与上述过程类似。

3 实验及性能分析

实验的运行环境如下：CPU 为 Intel Core2 2.8 GHz，内存为 2 GB，操作系统为 Windows XP，使用的编程语言为 Java，编程环境为 MyEclipse。实验使用的数据为某电子商务公司多年的销售数据，该交易事务数据库中包含了 103 497 条订单交易数据、315 061 条交易商品数据。概念层次树的最大层数为 4，分别是：商品大类 9 个，商品中类 19 个，商品小类 69 个，具体商品 5 236 个，各层支持度阈值由上往下分别设为 2、1.5、1、0.5。

对改进后的 FP-growth 算法和 Ada-FP 算法的执行效率进行了测试，实验的结果对比如图 2 所示。

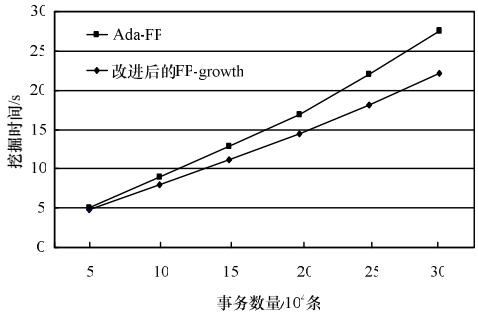

图 2 实验结果比较

各层的支持度保持不变，同时再把置信度置为 50，对全部事务进行挖掘测试。挖掘结果如图 3 所示，总共挖掘出 227 条关联规则，其中交叉层关联规则为 106 条，剩余的为各抽象层中的关联规则。

请设置最小支持度和置信度

大类层支持度：2 中类层支持度：1.5 小类层支持度：1

商品层支持度：0.5 最小置信度：50 挖 掘 重 置

挖掘结果

序号	规则内容	支持度	置信度	类型
1	[连衣裙]==>[上装]	10.39%	63.58%	交叉层
2	[裙子, 内衣/家居服]==>[上装]	1.65%	74.77%	交叉层
3	[连衣裙, 配饰]==>[上装]	1.70%	71.44%	交叉层
4	[非首饰]==>[女装]	9.71%	88.26%	交叉层
5	[其它]==>[女装]	2.42%	92.65%	商品中类层
6	[吊坠]==>[上装]	2.26%	66.11%	交叉层
7	[牛仔裤]==>[上装]	1.98%	74.23%	交叉层
8	[衬衫, 短裙]==>[针织衫]	1.42%	66.07%	具体商品层
9	[七分裤]==>[针织衫]	2.83%	51.63%	具体商品层
10	[家居服]==>[女装]	2.45%	75.61%	交叉层

首页 上页 下页 末页 本页显示第1条到第10条记录，总共227条记录

图 3 挖掘结果

从测试结果可以看出，改进后的 FP-growth 算法的挖掘效率有了很大的提高，特别是在 1-频繁项数量较大时，效果更加显著。其主要原因是由于哈希表的快速查找性能，因此在频繁项较多，事务数量越大的情况下，节省的时间也就越多。另一方面，对于概念层次树的改进，提高了访问概念层次树的效率，也有助 FP-growth 算法性能的改进。实验结果证明了改进算法的有效性，同时也具有一定的实用性。

4 结束语

本文提出多层关联规则的改进算法，通过对概念层次树的改进，既充分利用了概念层次树的特点，又提高了频繁 1-项集的生成效率，同时还减少了项目查询次数。另外，引入一个基于 Hash 表的辅助存储结构来存储频繁 1-项集，虽然在一定程度上增加了空间开销，但是显著地改进了项目的查询效率。再有，改进后的 FP-growth 算法既保持了原算法的特点，又利用了 Hash 表的快速查找性能，从而进一步提高算法的执行效率。由于不同概念层的项之间具有父子关系，因此在确定不同概念层的最小支持度阈值时，应考虑相互的关系和制约因素，这样才能不至于遗漏或产生太多的冗余频繁项集。在将来的工作中，将进一步对各概念层的最小阈值设置进行研究，使得多支持度的设置更合理、科学、简便。

参考文献

[1] Han Jiawei, Pei Jian, Yin Yiwen, et al. Mining Frequent Patterns Without Candidate Generation: A Frequent-pattern Tree Approach[J]. Data Mining and Knowledge Discovery, 2004, 8(1): 53-87.

[2] 毛国君, 段立娟, 王 实, 等. 数据挖掘原理与算法[M]. 北京: 清华大学出版社, 2005.

[3] Mao Runying. Adaptive-FP: An Efficient and Effective Method for Multi-level Multi-dimensional Frequent Pattern Mining[D]. [S. l.]: Simon Fraser University, 2001.

[4] 王振宇, 白石磊, 熊范纶. 多最小支持度策略的关联规则挖掘方法[J]. 小型微型计算机系统, 2002, 23(8): 971-973.

[5] 李成军, 杨天奇. 一种改进的加权关联规则挖掘方法[J]. 计算机工程, 2010, 36(7): 55-57.

[6] Han Jiawei, Kamber M. 数据挖掘: 概念与技术[M]. 范 明, 孟小峰, 译. 北京: 机械工业出版社, 2001.

编辑 任吉慧