第 34 卷 第 2 期 计 算 机 工 程 2008 年 1 月
Vol.34 No.2 Computer Engineering January 2008
·软件技术与数据库· 文章编号：1000—3428(2008)02—0060—02 文献标识码：A 中图分类号：TP301.6

一种改进的 Conf-H-Mine 算法

霍其润[1]，宋培卿[2]
(1. 首都师范大学信息工程学院，北京 100037；2. 中国兵器工业信息中心，北京 100089)

摘 要：对关联规则和约束关联规则的算法进行了研究和分析，基于候选集的约束算法需要反复扫描数据库，并产生大量的候选集，在挖掘低支持度、长模式的规则时效率低下。针对算法的缺陷，该文提出了一种 Conf-H-Mine 算法，采用 Conf-H-Struct 结构存储事务集合，不产生候选集，优化了关联规则的挖掘。实验结果证明了该算法的有效性。

关键词：数据挖掘；关联规则；项目约束挖掘

Improved Conf-H-Mine Algorithm

HUO Qi-run[1], SONG Pei-qing[2]
(1. College of Information Engineering, Capital Normal University, Beijing 100037;
2. Information Center, China North Industries Group Corporation, Beijing 100089)

【Abstract】 After analyzing and studying constraint-based data mining algorithms, there are great flaws in constraint algorithm based on candidate sets, the algorithm needs multiple scanning, produces lots of candidate sets, and has low efficiency when mining low support threshold, long rules. This paper introduces a new algorithm Conf-H-Mine which produces no candidate sets, and optimizes association rules mining. Experimental results show the algorithm is effective.

【Key words】 data mining; association rule; constraint-based mining

数据挖掘就是从大量数据中提取和挖掘知识，又称为数据库知识发现[1]。关联分析发现关联规则是数据挖掘中最活跃的领域[2-3]。在很多情况下，由于用户关心的只是关联规则的一个子集，因此引入了约束关联规则的概念。笔者提出了一种新的Conf-H-Mine算法，该算法仅扫描 2 遍数据库，并且采用模式增长的方法，不产生候选集。采用Conf-H-Struct结构存储事务集合，减少了空间的开销。该算法也适用于大型数据集，在进行数据集划分时能够进行有效的空间度量。

1 现有项目约束算法的缺点

Multijoins, Reorder, Direct项目约束算法都是Apriori-like类型的数据挖掘算法，尽管在算法中引入了项目约束缩小产生的“候选集规模”，并使之适合用户对挖掘结果的要求，但是Apriori-like类型算法的固有缺陷也继承下来了，即产生过多的候选集，并且需要对交易数据库进行多次扫描；Direct算法在缩小候选项集的同时并未对交易数据集按照项目约束条件B进行裁剪，导致在计算项集的最小支持度时仍然需要扫描大量的不满足用户给定约束条件的事务。Apriori-like类型的算法在规模较小的数据集上进行挖掘时比较有效，但在挖掘低支持度、长模式的关联规则时具有很大的局限性，因此，这类算法的挖掘效率有待提高[4]。

另一类典型的模式增长算法如 FP-Growth 算法，利用了分治法的思想，并不产生候选集，而采用将事务数据库压缩进一棵频繁模式树(FP-tree)的方法，根据频繁模式划分事务数据库为多个条件频繁模式树，并递归地挖掘频繁模式树内的频繁集，最后组合成全局的频繁集。但是这类算法需要大的存储空间，在面对大型数据集时，递归产生的开销巨大，一棵频繁模式树也不一定可以放入主存中，很容易产生频繁在主存和辅存交换的现象(颠簸现象)。

J Pei 和 J Han 等人提出了 H-Mine 方法，该方法克服了 Apriori-like 类型的候选集算法的缺点，在运行时间上提高一个数量级。同时也尽量避免了另一类关联规则挖掘算法——模式增长算法的缺点。H-Mine 算法对低支持度、长模式的关联规则非常适合，但是并没有将约束引入挖掘过程。在将项目约束引入 H-Mine 算法的过程中，提出了 Conf-H-Mine 算法，该算法针对大规模事务数据库，将事务数据库划分为可放入主存子事务数据库，并进行挖掘。

2 基本概念

在描述算法前，对基本概念和数据结构进行如下定义：

定义 1 最大共同前缀。给定项目约束条件B，B以析取范式DNF的形式表示，即$D_1 \vee D_2 \vee \cdots \vee D_n$，每一个$D_i$以$K_1 \wedge K_2 \wedge \cdots \wedge K_n$形式表示。如果$D_i$和$D_j$最多有$m$个析取项相同，称这$m$个析取项为$D_i$和$D_j$最大共同前缀$P_{ij}$。

定义 2 Conf-H排序。给定项目约束条件B，B以析取范式DNF的形式表示，即$D_1 \vee D_2 \vee \cdots \vee D_n$，每一个$D_i$以$K_1 \wedge K_2 \wedge \cdots \wedge K_n$形势表示。Conf-H排序对每一个项目item按照如下规则排序：(1)在所有的约束项目中寻找最大共同前缀P_{ij}，P_{ij}按照字典排序。(2)将所有的P_{ij}按照出现次数排序。 (3)调整所有的事务，按照在所有的最大共同前缀出现的顺序、在约束中出现的其他项目、不在约束中出现的项目顺序排序。(4)调整所有的事务不在约束中出现的项目，并按照字典排序。

定义 3 约束位图(constraint bit map)。给定经过 Conf-H 排序的约束条件 B，为一个子约束条件给出一个数据结构，称为约束位图。约束位图是一个位数组，每一个在 B 中出现

作者简介：霍其润(1976－)，女，讲师、硕士，主研方向：计算机应用技术；宋培卿，工程师、硕士

收稿日期：2007-03-30 **E-mail：**jzhqr@163.com

的项目按照 Conf-H 顺序占用 2 bit。如果项目未出现在子约束条件中，其值为 00；如果项目存在约束，其值为 01；如果项目是排斥约束，其值为 10。

定义 4 约束位图的运算。对 2 个约束位图，定义运算 $\oplus$ 为：从第 1 位开始，每 2 位进行二进制进位加法，第 3 个进位舍弃。运算结果不是一个约束位图。

性质 约束位图的运算性质。如果D_1和D_2运算结果对应某项的位为 11，则D_1和D_2不相容。即满足D_1的事务不可能满足D_2。如果对应某项的位不是 11 且发生了改变，则D_1和D_2中均含有该约束项。

3 Conf-H-Mine 算法

3.1 基本概念和问题描述

为了有效地挖掘项目约束关联规则，当数据集比较大时，可以采用划分的方法解决主存空间不够的问题。

设*TDB*是待挖掘数据集，规模是*n*，*min_sup*是最小支持度。(1)进行第 1 遍扫描，可以将数据库裁剪，得到含有频繁项的数据集。(2)将*TDB*平均划分为*k*个部分，TDB_1, TDB_2,…, TDB_k，以使在每个部分中的事务规模适合在主存中处理，每个划分含有n_i个事务，且 $\sum_{i=1}^{k} n_i = n$ 。

现在可以在每个划分TDB_i上进行H-Mine挖掘工作，支持度为

$$min\_sup_i = \left\lfloor min\_sup \times \frac{n_i}{n} \right\rfloor$$

即每个划分数据集以相同最小支持度挖掘，并与全局数据集保持相同的支持度。

假设$F_i(1 \le i \le n)$是TDB_i中的频繁模式的集合，根据划分数据库的性质，如果不存在F_i使一个频繁模式$P \in F_i$，则P不可能在全局数据集*TDB*上满足最小支持度*min_sup*，可以在各个划分数据集TDB_i上挖掘F_i中的模式P并且通过再次扫描全局数据集计算P的全局支持度。

这种划分的方法与基于 Apriori 的划分方法的思想类似，在基于 Apriori 的划分算法中，数据集被划分为子集，并在子集上用 Apriori 算法挖掘模式，最后通过再次扫描数据集来计算全局支持度。Conf-H-Mine 算法与其有本质上的差别：

(1)基于 Apriori 的划分算法很难找到一个普遍适用的、有效的划分规则和预测 Apriori 算法的空间复杂度。但是对于基于 H-Mine 的划分算法来讲，由于其附属结构很小，并且可以很容易地进行空间消耗的估计，因此可以将数据集划分。

(2)H-Mine 首先寻找全局频繁模式集，在挖掘划分数据集时，H-Mine 只考察全局频繁的数据。因为在划分子数据集中，有很多局部频繁的模式，所以 H-Mine 将不再花费代价计算，但 Apriori-like 算法无法排除这些频繁模式。

H-Mine 更高效地从局部频繁模式导出全局频繁模式。

3.2 Conf-H-Mine 算法挖掘过程

一个大型数据集 *TDB* 被划分成 4 个部分：*P*1, *P*2, *P*3, *P*4。项目约束规则是 *B*，假设最小支持度是 100，则本地支持度是 25。在这 4 个划分数据子集中分别用 H_Mine 来挖掘满足条件 *B* 的本地频繁模式，得到的结果如表 1 所示。

由于模式 *ab* 在所有的子集中都“本地频繁”，因此 *ab* 是全局频繁的。全局支持度就是 *ASC*=280。模式 *ac* 和 *ad* 与之情况相同，也是全局频繁的。它们的全局支持度分别是 320 和 260。模式 *abc* 在除 *P*2 的所有划分子数据集中都频繁，因为它的 *ASC*=120>100，所以它是全局频繁的，只需扫描 *P*2，得到在 *P*2 中的本地支持度与之 *ASC* 相加来计算其全局支持度。模式 *abcd* 在除 *P*2, *P*3 外的子集中本地频繁，并且 *ASC*=50 小于全局支持度，需要扫描 *P*2, *P*3 来计算全局支持度。

接下来第 3 次扫描 H-Mine，先扫描 *P*2，得到 *ab* 的本地支持度 10 和 *abcd* 的本地支持度 20。考虑到最小本地支持度是 25，因为 *abcd* 在 *P*3 中不本地频繁，所以 *abcd* 在 *P*3 中的支持度不可能大于 25，*abcd* 已不可能全局频繁，不需要扫描 *P*3，第 3 次扫描结束。

表 1 划分子数据集中的本地频繁模式

本地频繁集	划分子数据集	*ACS*(支持度之和)
ab	*P*1,*P*2,*P*3,*P*4	280
ac	*P*1,*P*2,*P*3,*P*4	320
ad	*P*1,*P*2,*P*3,*P*4	260
abc	*P*1,*P*3,*P*4	120
abcd	*P*1,*P*4	50
…	…	…

3.3 算法分析与评价

从上面的叙述可以看出，Conf-H-Mine 算法采用了几种优化的方法来产生最终的全局频繁模式。(1)计算每个局部频繁模式的本地支持度的和(*ASC*)。(2)只在以后扫描，使某个模式不进行本地频繁划分子集的操作。(3)利用本地最小支持度阈值和已得到的本地模式的 *ASC* 来推断模式全局频繁的可能性，只考虑那些有可能成为全局频繁的模式。

利用上述优化方法，需要重新计算的模式可以大大地缩减。即使在数据集的关联比较稀疏的情况下，只有约 20%的本地频繁模式需要在第 3 次扫描中进行考察。

上述介绍的 Conf-H-Mine 算法空间开销小，并且可以有效地在主存中挖掘带项目约束的关联规则。从目前的内存容量来看，一个中等规模的数据集可用 Conf-H-Mine 进行有效的处理。如果是比较大的数据集，带项目约束的关联规则挖掘总可以通过 Conf-H-Mine 算法经过 3 步扫描来实现。第 1 遍扫描得到频繁 1-项集；第 2 遍扫描将数据集划成可以在主存中处理的小规模数据集；第 3 遍扫描用来产生全局频繁模式集。由于每个子集都可以用 H-Mine 来有效挖掘，因此整个数据集的挖掘也极具效率。

与基于 Apriori 的划分方法比较，Conf-H-Mine 方法要多一次扫描的过程。但是整个规则挖掘过程的主要工作在于对每个划分子集所进行的挖掘。由于 Conf-H-Mine 产生较少的划分子集并且得益于在每个子集上的高效挖掘，因此 Conf-H-Mine 要比基于 Apriori 的划分与 Direct 算法相结合的方法总体上更有效率。

4 算法的实现与比较

为了评价 Conf-H-Mine 算法的执行效果，可以进行一系列对比试验。由于挖掘的主要开销在于发掘模式，因此只对基于主存的算法进行了实现。实验平台是 2000 MHz Pentium，内存 256 MB 的 PC 机上，操作系统是 Windows 2000。Conf-H-Mine 算法和 Direct 算法由 Visual C++6.0 实现。

笔者改进了 Direct 算法，使它可以将事务数据库装入到主存中扫描。这里主要列出了在主存中挖掘的执行数据，有关 Conf-H-Mine 的运行时间数据包括构造 Conf-H-Struct 的运行时间和挖掘项目约束模式的运行时间，还包括所有的 CPU 开销和 I/O 开销。

测试数据集是 Gazelle 数据集。Gazelle 数据集是一个稀疏数据集，内容是 Gazelle.com 的用户点击序列，包括 60 000 个事务，每个事务包含 267 条以上的项目。 (下转第 65 页)