第34卷 第17期 计 算 机 工 程 2008年9月
Vol.34 No.17 Computer Engineering September 2008
·软件技术与数据库· 文章编号：1000—3428(2008)17—0047—03 文献标识码：A 中图分类号：TP311

高维大数据集中频繁闭合模式的挖掘

余光柱[1,2]，王 亮[3]，易先军[4]，邵世煌[1]
(1. 东华大学信息学院，上海 201600；2. 湖北省荆州市公安局，荆州 434000；
3. 湖北省荆州市环境管理信息中心，荆州 434000；4. 北京大学计算机软件与微电子学院，北京 102600)

摘 要：高维大数据集对现有的数据挖掘算法提出了挑战。该文把挖掘任务分解为挖掘频繁长模式与短模式2个子问题，提出一种在高维大数据集中挖掘长项集的算法，即inter-transaction。该算法利用了高维数据中长事务相交迅速变短的特性，通过事务的交集运算直接得到长闭合模式，同时采用新的减枝策略，优化了事务交集运算的方法。实验表明，该方法对高维大数据集非常有效。

关键词：高维大数据集；频繁闭合模式；减枝策略

Frequent Closet Pattern Mining in Large High Dimensional Dataset

YU Guang-zhu[1,2], WANG Liang[3], YI Xian-jun[4], SHAO Shi-huang[1]
(1. College of Information Science and Technology, Donghua University, Shanghai 201600; 2. Jingzhou Public Security Bureau, Hubei Province, Jingzhou 434000; 3. Information Center of Jingzhou Environment Management, Hubei Province, Jingzhou 434000;
4. School of Software and Microelectronics, Peking University, Beijing 102600)

【Abstract】 High dimensional large data has posed great challenges to most existing algorithms for frequent patterns mining. This paper decomposes the mining task into two parts: mining short frequent itemsets and long frequent itemsets, and proposes a new algorithm, i.e., inter-transaction, to find all long frequent closet patterns in large high dimensional dataset. The new algorithm utilizes the characteristic that the intersection of long transactions is usually a very short itemset, and can find long closet patterns directly via intersecting relevant transactions. In addition, the algorithm adopts a new pruning strategy to cut down search space and optimizes the performance of intersection of transactions. Experiments on synthetic data show that this method achieves high performance in large high dimensional dataset.

【Key words】 large high dimensional dataset; frequent closet pattern; pruning strategy

1 概述

随着各学科研究的深入，出现了大量的高维数据集。以频繁模式的挖掘为例，现有的挖掘算法如Apriori等一般以枚举各列(维)的组合为搜索空间，能处理的维数(属性数)一般不超过100[1]。近年来出现的CARPENTER[1], TD-CLOSET[2]算法虽然能处理高维生物数据，但都假设数据的大小(行数)远小于其维数(列数)。但现实中存在着许多高维大数据集，这种假设限制了它们的广泛使用。即使是生物数据，它们的大小也会因为研究的深入而迅速增长，CARPENTER, TD-CLOSET赖以工作的条件会逐步消失。同时，将多个生物数据存放于一个库中可能会构成一个维数更高、规模更大的数据集，在这样一个高维大数据集上挖掘频繁集是一件既具有挑战性又很有意义的工作。为此，本文根据高维数据集的特点，把复杂的挖掘任务分解为挖掘频繁长模式与频繁短模式2个子问题，再选择合适的方法分别予以解决。鉴于现有的许多算法在挖掘短模式时是有效的，提出一种专门挖掘闭合长模式的算法即inter-transaction。

2 背景知识

设 $I=\{i_1, i_2,\cdots, i_m\}$ 为项目集合，$P=\{T_1,T_2,\cdots,T_n\}$ 为事务数据库。每一事务 $T_q(T_q\in P)$ 都是 I 的一个子集，即 $T_q\subseteq I$。为简便，有时把集合 $\{i_1, i_2,\cdots, i_k\}$ 写为 $i_1\,i_2\,\cdots\,i_k$。

项集/事务的大小在本文中称为长度。给定一项集/事务，其长度若超过用户定义的阈值 *minlen*，称此项集/事务为长项集/事务，否则称短项集/事务。类似的方法可定义：长模式，长闭合模式，长频繁模式，长频繁闭合模式等。给定一组事务，其事务编号(transaction identifier, tid)的集合记为 *tidlist*；由 *tidlist* 指定的一组事务的交集称交集事务，记为 $T(tidlist)$；当 $|tidlist|=K$ 时，$T(tidlist)$ 称 K-交集事务($1\le K\le N$，N 为事务数)；当 $|tidlist|=1$ 时，$T(tidlist)$ 是事务数据库中的一条事务记录。例如，设 T_1=ABDF, T_2=ADFG，则 *tidlist*={1, 2}，2-交集事务 $T(1,2)=T_1\cap T_2$=ADF。没有歧义时，简称事务。

交集事务实际上是一项集，但它们又有区别。例如，K-项集指明项集中项目数为 K，没有指明项集的支持度是多少，而 K-交集事务不能指明生成的项集有多长，但它指明了生成的项集的支持度不小于 K。它类似于文献[1]中行支持集(row support set)的概念。若支持度阈值为 *minsup*，局部支持度阈值定义为 *minsup*/*n*，*n* 为划分数。其他概念，如局部频繁集、全局候选集等见文献[3]。

3 算法

3.1 基本思想

数据库中所有项集可通过事务间的交集运算产生。设库中事务数为 N，任意2事务的交(共 C_N^2 个)能产生所有支持度不小于2的项集(2-交集事务)。如此下去，任意 m 个事务的

基金项目：高等学校博士学科点专项科研基金资助项目(20060255006)
作者简介：余光柱(1969－)，男，博士研究生，主研方向：数据挖掘；王 亮，工程师；易先军，博士研究生；邵世煌，教授、博士生导师
收稿日期：2007-09-25 **E-mail：**ygz@mail.dhu.edu.cn

交集能产生所有支持度不小于 m 的项集(m-交集事务，共 C_N^m 个)。理论上，可通过事务间的交集运算得到所有闭合模式(项集)的支持度[1]。与 CARPENTER 和 TD-CLOSE 相似，inter-transaction 也是基于枚举所有事务组合的。这种方法的一个优点是可直接得到频繁长项集，避免了从短项集开始逐步扩展到长项集的冗长过程。

在实际应用中，事务数 N 常达数百万之多，2^N 次交集运算往往较为困难。为此，本文采用了划分的方法[3]，将数据库平分为 n 份，先求出每一划分 P_i 中的局部频繁项集，再构造一个全局候选集，最后检验每一候选项集的支持度。但是，即使每一划分内事务数 N 很小(例如 N=1 000)，通过枚举所有的交集事务来求得局部频繁集的方法仍不可行。

3.2 减枝策略

由于 inter-transaction 的目标是挖掘长项集，而短事务对长项集的支持度没有影响，且短事务与其他事务的交也是短的，因此可采取如下减枝策略来控制(交集)事务的数量：去掉所有短(交集)事务。具体办法是：划分 P_i 中任意 2 事务相交后产生 C_N^2 个 2-交集事务，去掉所有短交集事务，合并相同的长交集事务并检验每一个长交集事务的支持度，再让所有长交集事务两两相交，如此反复，直到没有长交集事务。这样，生成的短交集事务可及时被剪除，计算量大幅度减少。

与基于大项目集性质的减枝策略相比，新的减枝策略充分利用了高维数据集中长模式相对稀疏的特性，避免了大量的短交集事务的生成，提高了算法的性能。一般来讲，频繁模式的密度随模式长度的增长而变得稀疏。而且，长事务/项集间相同的项目往往很少，这意味着长事务/项集的交集很可能是一个短项集。而且，参与交集运算的事务/项集越多，得到的项集越短。相反，短事务/项集过于稠密，让这些短事务/项集两两相交，会产生过多的交集事务，计算量太大。这正是笔者不能枚举所有交集事务，而将整个挖掘任务分解为挖掘长模式与短模式 2 个子问题的主要原因。

3.3 inter-transaction 算法

inter-transaction 算法描述如下：

输入：数据库 P，阈值 minsup, minlen

输出：所有频繁长项集

步骤：(1)扫描数据库，将数据库 P 划分为 n 等份；(2)对每一划分 $P_i \in P$，调用 gen-lf-itemsets，求出局部频繁长项集；(3)合并所有局部频繁长项集，组成全局候选集 C；(4)再次扫描数据库，对所有候选项集 $c \in C$ 进行统计计数，得到其支持度。如果候选项集 c($c \in C$)的支持度 c.support≥minsup，输出项集 c。

Inter-transaction 与 partition 算法[3]的框架相似，但调用的子程序不同。partition 算法通过调用一个完整的算法如 Apriori 求得划分中所有局部频繁集，Inter-transaction 则通过事务的交集运算求得局部频繁长项集。另一区别是划分大小的选取。partition 算法主要根据内存大小确定划分大小，inter-transaction 算法则在减少划分内交集运算数与提高局部支持度阈值之间取得平衡。

子程序 gen-lf-itemsets 负责产生划分内所有局部频繁长项集。该子程序利用事务编号集合 *tidlist* 记录哪些事务参与了交集运算，用新的减枝策略压缩搜索空间。算法描述如下：

输入: 数据库划分 P_i 阈值 minsup, minlen

输出: 所有局部频繁长项集

步骤：(1)在输入的划分 P_i 中，求出任意 2 个(交集)事务的交；(2)如果没有长交集事务，子程序停止；(3)去掉所有的短交集事务，合并相同的长交集事务：如果 T(tidlist1)=T(tidlist2)，且 tidlist1≠tidlist2，则将两者合并生成新的交集事务 T(tidlist)=T(tidlist1∪tidlist2)；(4)检验每个交集事务 T(tidlist)，如果它是局部频繁的(|tidlist|≥minsup/n)，把它加入到全局候选集 C 中：C=C∪T(tidlist)，转(1)。

4 事务的交集运算

Inter-transaction 的性能基于大量的事务交集运算。一般地，事务可用水平项目向量(Horizontal Item-Vector, HIV)和水平项目列表(Horizontal Item-List, HIL)独立表示[4-5]。但无论采用哪种格式，事务交集运算的性能都会随事务长度的增加而降低[5]。若项目数为 8 000，则 HIV 格式的事务在内存中需占有 1 KB，任何 2 事务的交需 8 000 次比特运算，效率极低。

在垂直挖掘算法中，多种优化方法如 Diffsets 和 Viper 被提出以提高交集运算的性能。但这些优化方法的一个重要目标是减少内存消耗，主要通过压缩事务向量占用的内存数量来提高交集运算的速度。这些压缩后的格式不适于直接进行交集运算，需要变换之后才能进行位运算，从而影响了性能且应用范围有限。inter-transaction 则同时用 HIV 和 HIL 2 种格式表达每一事务。由于 HIL 格式的数据能说明每一项目在 HIV 格式中的位置，项目向量的交集运算只需在相应的比特上完成。例如，设项目数为 8，事务 T_1 的 HIL 表达式为{1, 3, 8}，HIV 表达式 10100001，事务 T_2 的 HIL 表达式为{1, 2, 3, 6, 8}，HIV 表达式 11100101，则 T_1 和 T_2 的交不必进行 8 次比特运算，根据较短的 HIL 表达式(此处为 T_1)提供的信息可知，只需在第 1, 3, 8 比特位置进行逻辑与运算即可。这样虽然会浪费大量内存，但它极大地提高了运算的速度，使事务交集运算的性能不再受限于事务的长度。而且，基于划分的方法节省的内存足以满足这种冗余表达方式。

5 实验及分析

实验在浪潮 XEON 服务器上进行。CPU 主频 2.4 GHz，内存 2 GB，运行 Windows 2003，程序用 Delphi 7 编写。实验用的 6 个数据集为 T40.I30.D8000K，项目数分别为 500, 1 000, 2 000, 4 000, 8 000 和 16 000，它由 IBM 的数据发生器生成，具有高维大数据集的特征。其平均事务长 40，含有许多长模式。

图 1 表明，inter-transaction 的计算时间随事务量线性增长。图 2 显示了项目数量的变化对计算时间的影响。

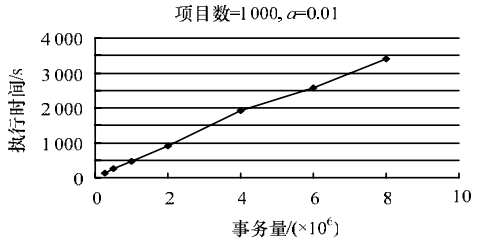

图 1 时间-事务量变化曲线

随着项目的增加，性能明显好转，这是因为交集事务在项目多时更容易缩短所致。

图 3 是工作量-项目数变化曲线，它表示在一定的考察时间内，*minlen* 随着项目数增加能达到最小值的变化。*minlen* 越小，意味着 inter-transaction 完成的工作越多，留给其他算法的工作量就越小，效率越高。但 *minlen* 太小，需枚举的交集事务会大量增加，从而影响整体性能。图 4 显示了支持度阈值 *minsup* 对算法性能的影响。若 *minsup* 太小，则候选集会迅速增长，算法性能快速降低。

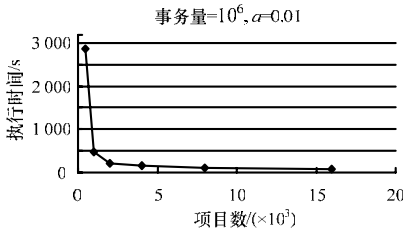

图 2　时间-项目数变化曲线

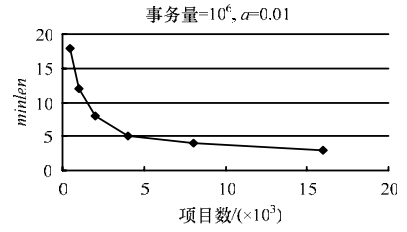

图 3　工作量-项目数变化曲线

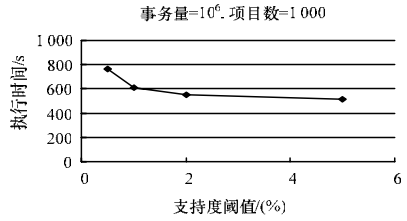

图 4　时间-支持度阈值变化曲线

6　结束语

本文把挖掘任务分解为挖掘长模式与短模式 2 个子问题，根据长、短模式的不同特点选用合适的方法加以解决。提出的 inter-transaction 算法充分利用了高维数据中长事务相交后迅速变短这一特性，适于高维大数据集中长项集的挖掘。该算法采用新的减枝策略，用冗余信息提高交集运算的速度，对内存要求较低。性能随维数的增长而提高。

如何减少数据偏离(data skew)、参数(如 *minlen*,划分大小)选取对算法性能的影响是进一步的研究重点。

参考文献

[1] Feng Pan, Gao Cong, Yang Jiong, et al. CARPENTER: Finding Closed Patterns in Long Biological Database[C]//Proc. of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. Washington D. C., USA: [s. n.], 2003: 413-419.

[2] Liu Hongyan, Han Jiawei, Xin Dong, et al. Mining Frequent Patterns from Very High Dimensional Data: A Top-down Row Enumeration Approach[C]//Proc. of the 6th SIAM International Conference on Data Mining. Bethesda, Maryland, USA: [s. n.], 2006: 20-22.

[3] Savasere A, Omiecinsky E, Navathe S. An Efficient Algorithm for Mining Association Rules in Large Databases[C]//Proc. of the 21st International Conference on Very Large Databases. San Francisco, USA: [s. n.], 1995: 432-444.

[4] Shenoy P, Haritsa J R, Sudarshan S, et al. Turbo-charging Vertical Mining of Large Databases[C]//Proc. of ACM SIGMOD International Conference on Management of Data. Dallas, Texa, USA: [s. n.], 2000: 22-33.

[5] Zaki M J, Gouda K. Fast Vertical Mining Using Diffsets[C]//Proc. of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining Conference on Knowledge Discovery in Data. Washington D. C., USA: [s. n.], 2003.

（上接第 38 页）

4　实验

对递归解法、动态规划解法和新解法进行了实验比较。在配置为 Genuine Intel(R) CPU T2050@1.60 GHz, 1.00 GB 内存的机器上，通过从数据文件中读取输入值，对 3 种解法进行实验的相应参数和结果如表 1 所示。

表 1　3 种解法的时间耗费对比

数据量	递归/ms	动态规划/ms	新解法/ms
n=25, c=1 000	4 735	0	235
n=25, c=10 000	4 797	16	235
n=30, c=1 000	147 610	15	422
n=30, c=10 000	150 734	32	436

如表 1 所示，因递归算法对 0-1 背包问题进行了大量重复的递归调用，动态规划解法和新解法的效率都远远优于它。接下来，采用不同的数据量，着重对动态规划解法和新解法进行了实验和比较，结果如表 2 所示。

表 2　2 种解法的时间耗费对比

数据量	动态规划/ms	新解法/ms
n=5, c=100 000	32	0
n=8, c=100 000	47	0
n=8, c=1 000 000	516	0
n=20, c=100 000	156	125
n=20, c=1 000 000	1 547	125
n=30, c=1 000 000	2 500	422

实验结果证实了 3.4 节的分析。从表 1 和表 2 可以看出，当背包容量 c 很大、2^n-1 足够小于 nc 时，新解法要优于动态规划解法。另外，随着物品个数或背包容量的增加，动态规划法的时间耗费均有较大的变化；而新解法的时间耗费受背包容量变化的影响很小，只会因物品个数的增加而受到较大的影响。因此，在实际应用中，可根据具体情况加以选择，新解法不失为一种有效的解决方案。

5　结束语

本文使用形式推导的方法，开发出了一种新的求解 0-1 背包问题的非递归算法。比起通常的分支限界法、回溯法、递归解法和动态规划解法，新解法以自顶向下的非递归方式进行计算，仅当需要某子问题的解时，才生成该子问题并把求解的结果保存起来，在避免重复求解相同子问题的同时，又减少了实际计算的子问题个数。使用先于该算法得到的循环不变式，形式化地验证了该算法的正确性。

参考文献

[1] 王晓东. 计算机算法设计与分析[M]. 2 版. 北京: 电子工业出版社, 2005.

[2] 常用算法(三)[Z]. (2004-06-06). http://www.channel7.cn/2004/10-4/191731.html.

[3] Xue Jinyun. A Unified Approach for Developing Efficient Algorithmic Programs[J]. Journal of Computer Science and Technology, 1997, 12(4): 314-329.

[4] Xue Jinyun. Two New Strategies for Developing Loop Invariants and Their Applications[J]. Journal of Computer Science and Technology, 1993, 8(2): 147-154.

[5] Dijkstra E W. A Discipline of Programming[M]. Englewood Cliffs: Prentice Hall, 1976.