第 31 卷 第 3 期
2009 年 2 月

武 汉 理 工 大 学 学 报
JOURNAL OF WUHAN UNIVERSITY OF TECHNOLOGY

Vol. 31 No. 3
Feb. 2009

DOI:10.3963/j.issn.1671-4431.2009.03.023

基于数据流的 top-k 频繁项集挖掘

张 蕊
(武汉理工大学计算机科学与技术学院,武汉 430070)

摘 要: 对数据流进行频繁项集挖掘具有重要意义。然而传统的办法是由用户设定合适的支持度阈值,这在数据流环境中非常困难。更实际的办法是由用户设置一个参数 k,输出最频繁的 K 个项集。讨论了数据流的 top-k 频繁项集的挖掘,给出了相关定义,分析了挖掘中的相关技术和性质,提出了一个数据流 top-K 频繁项集挖掘算法 LIONET,并分析了算法的优越性。

关键词: 数据流; 算法; top-k 频繁项集
中图分类号: TP 311 **文献标识码**: A **文章编号**:1671-4431(2009)03-0087-04

Mining Accurate Top-K Frequent Itemsets from Data Streams

ZHANG Rui
(School of Computer Science and Technology, Wuhan University of Technology, Wuhan 430070, China)

Abstract: Frequent pattern mining on data streams is of great significance. Though a minimum support threshold is assumed to be available in classical mining, it is hard to determine it in data streams. It is practical for users to give a parameter K and K "interesting" frequent item sets can be returned as desired. Definition of top-K frequent item sets is presented and the relative natures are discussed. An algorithm to mine accurate top-K frequent item sets from data streams is proposed. The superiorities of this algorithm are analyzed.

Key words: data stream; algorithm; top-k frequent item set

对数据流进行频繁项集挖掘具有重要意义。然而,传统的频繁项集挖掘要求用户设定最小支持度,根据该阈值来判定项集是否频繁。而这个阈值的设定非常有技巧性。如果设计得过小,将会产生大量的频繁项集;如果过大,又可能无法产生频繁项集。在传统的静态数据集上,可以通过反复尝试来调整阈值。但是在数据流环境中,数据几乎只能被读一次,因此这个办法不可行。另外,从用户的角度,一个自然的想法是指定一个参数 k,输出最频繁的 k 个项集。这样,返回结果的数量在用户可控的范围内。这更符合实际需要。

1 top-K 频繁项集

给出相关问题定义如下。

定义 1 l 项集:含有 l 个项的集合。

定义 2 k-th 频繁 l 项集:假定将 l 项集按支持度降序排列,注意支持度相同的项集相互顺序任意,那么在这个有序序列中处于第 k 个位置的项集就是 k-th 频繁 l 项集[1,2]。

定义 3 top-k 频繁 l 项集:给定项集的最大大小 $\max\_l$,对于给定 $l(1 \leq l \leq \max\_l)$,所有支持度大于等于 k-th 频繁 l 项集的支持度的 l 项集的集合就是 top-k 频繁 l 项集。

定义 4 top-k 频繁项集:给定项集的最大大小 $\max\_l$,对于任意 $l(1 \leq l \leq \max\_l)$,所有 top-k 频繁 l 项集的并集就是 top-k 频繁项集。

收稿日期:2008-09-11.
作者简介:张 蕊(1977-),女,讲师. E-mail:zhangrui@whut.edu.cn

注意:因为可能存在支持度相同的情况,这里 *top-k* 频繁项集的个数不一定精确等于 $k \times \max\_l$ 个,可能比它大。

2 算法思想与描述

2.1 算法的基本思想与相关工作

算法主要解决 2 个问题:1)怎样在数据流上进行频繁项集挖掘;2)怎样找出频繁项集。尽管研究第 1 个问题的很多,但解决第 2 个问题的并不多。主要有 Metwally 等[2]、Wang 和 Fu 等[3,4]所做的工作。其中,Metwally 等仅考虑挖掘 top-k 频繁项,而不是项集。Wang 和 Fu 的工作和作者接近,但其提出的是假设 Zipfian 分布或 Chernoff 分布下的非精确算法,而文中提出的是基于滑动窗口的精确算法,记做 LIONET(aLgorithm to fInd tOp-k frequeNt itEmsets from sTreams)。

根据数据流的应用环境,算法要能处理增量更新,并且扫描次数最好是一次。因此我们采用了一个适合增量挖掘的树结构——CanTree(CANonical-order TREE)[5]。它根据事先由用户指定的某一顺序对项进行排序,通过一次数据扫描就可以构建。同时,通常最近的数据往往比"老"的数据影响更大。因此算法采用按数据单元计算的滑动窗口技术。换句话说,算法设计了一个带滑动窗口的树结构来进行频繁项集的挖掘。

表 1 数据流的部分事务

	事务编号	事务内容
数据流 块 1	t_1	{a,f,b,d,c}
	t_2	{e,d,c,a}
	t_3	{a,b}
数据流 块 2	t_4	{c,e,a}
	t_5	{d,a,b,c}
	t_6	{a,c,e}}
数据流 块 3	t_7	{c,d,a}
	t_8	{a,e,f}
	t_9	{a,d}

算法主要有 3 部分。首先,根据数据流在滑动窗口中的初始块进行树的构建,其实质是对数据流初始块中的事务的插入过程。然后随着窗口的滑动,对树进行维护,以使结点的支持度和滑动窗口中的数据保持一致,并剪掉"无用"的子树以节省空间。另外就是对于构建好的树进行 *top-k* 频繁项集挖掘。建树的过程如表 1 和图 1 所示。下面主要讨论如何对构建好的树进行 *top-k* 频繁项集挖掘。

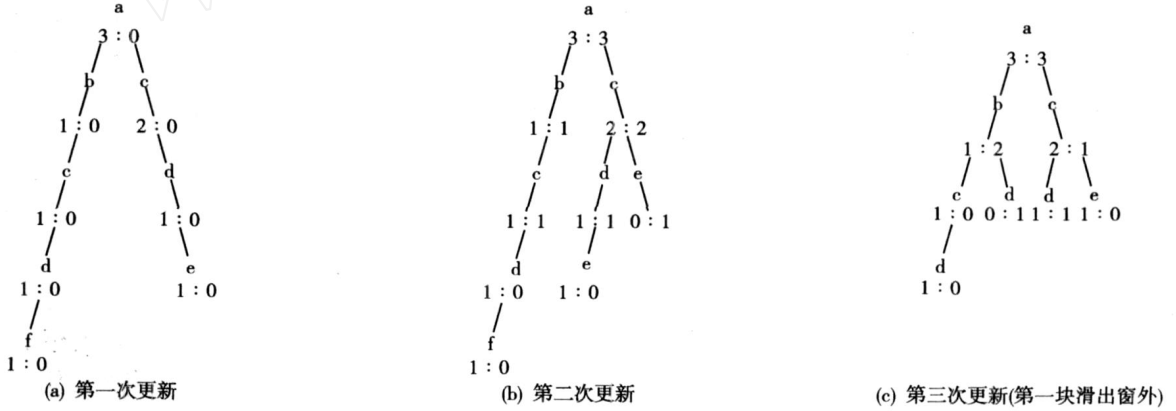

图1 树的构建与维护(滑动窗口大小为2)

2.3 有效挖掘的方法与算法描述

给定整数 $\max\_l$ 和 k,要找出 *top-k* 频繁项集,即要找出所有 *top-k* 频繁 l 项集($1 \le l \le \max\_l$)。显然,对于不同长度的项集,要找到不同的支持度阈值。对于任意 l($1 \le l \le \max\_l$),*k-th* 频繁 l 项集的支持度记作 ξ_1,则所有支持度大于 ξ_1 的 l 项集组成的集合就是 *top-k* 频繁 l 项集。另外,结合前人的工作[1],注意到以下事实:

定理 1 对于 $1 \le l_1 < l_2 \le \max\_l$,*k-th* 频繁 l_1 项集的支持度 ξ_1 不小于 *k-th* 频繁 l_2 项集的支持度,即 $\xi_{l_1} \ge \xi_{l_2}$。

证明略。根据这个定理,需要一个全局的支持度阈值 ξ 来支持挖掘。且 $\xi = \min(\xi_1, \xi_2, \cdots, \xi_{\max\_l})$。容易得到下面的推论:

推论 1 在挖掘过程中,$\xi = \xi_{\max\_l}$。

根据这个推论,可以仅在 $\xi_{\max\_l}$ 更新时更新 ξ。

算法 LIONET 的挖掘过程是在用户指定阈值 max_l 和 k 时,找出所有 top-k 频繁项集。挖掘的数据基础是如图 1 所示的带滑动窗口的 CanTree,注意这不同于文献[1 中静态数据上建立的 FP 树。根据用户给定的 max_l,挖掘中动态维护支持度阈值 ξ 和相应 k-th 频繁 l 项集的支持度 $\xi_1, \xi_2, \cdots, \xi_{max\_l}$ 以及结果集合 $result_1, result_2, \cdots, result_{max\_l}(1 \leq l \leq max\_l)$。挖掘时调用 mine_topk_itemset (Tree, NULL, $\xi, \xi_1, \xi_2, \cdots, \xi_{max\_l}$),对从根结点开始的子树(即整棵树)进行挖掘。挖掘思想类似 FP-growth[6],从初始后缀模式开始构造它的条件模式基。然后,构造条件模式树,并递归地在该树上挖掘。但是,这棵树是对滑动窗口中的数据块建立并维护的,它含有当前滑动窗口中的所有项,并且按某指定序(比如字典序)而不是按支持度大小排列,每个结点的支持度序列会随着数据流的更新而维护,甚至被剪枝。而且这里没有一个固定的支持度阈值 ξ。实际上 $\xi, \xi_1, \xi_2, \cdots, \xi_{max\_l}$,初始都是 0。而 ξ_1 为 0 意味着 $result_l$ 中包含所有的 l 项集($1 \leq l \leq max\_l$)。不过随着挖掘的进行,ξ_1 会动态更新,全局的支持度阈值 ξ 也会适时更新。挖掘的具体过程描述如下:

mine_topk_itemset($Tree, \alpha, \xi, \xi_1, \xi_2, \cdots, \xi_{max\_l}$){
　update_sup_l(intl){
　if $Tree$ 含单个路径 P{
　　if $result_l.count < k$
　for 路径 P 中结点的每个组合 β
　$\xi_l = 0$
　产生项集 $\beta \cup \alpha$,其支持度 sup $port$ = β 中结点的最小支持度;
　else if $result_l.count = k$
　if $\xi_{|\beta \cup \alpha|} \leq$ sup $port${
　ξ_l = $result_l$ 中项集的最小支持度;
插入 $\beta \cup \alpha$ 到 $result_{|\beta \cup \alpha|}$;
　update_sup_l($|\beta \cup \alpha|$)
　else if $result_l$ 中支持度为 ξ_l 的项集只有一个
　if $result_{|\beta \cup \alpha|}$ 含有支持度小于 $\xi_{|\beta \cup \alpha|}$ 的项集 X
　$\zeta_l = l.$ sup $port$;
　}
　update_sup_total(int l){
if l = max_l
从 $result_{|\beta \cup \alpha|}$ 删除 X
　}
　$\xi = \xi_1$
　}
　}
　else{
　for Tree 的头表中的每个 a_i
　　产生项集 $\beta = a_i \cup \alpha$,其支持度 sup $port = a_i.$ sup $port$;
　　用 ξ 构造 β 的条件模式基,然后构造 β 的条件模式树 $Tree\beta$
　　if $Tree\beta \neq \Phi$
　　mine_topk_itemset($Tree\beta, \beta, \xi, \xi_1, \xi_2, \cdots, \xi_{max\_l}$)。

2.3 算法结果与分析

为了评估结果的质量,用召回率(recall)和准确率(precision)2 个度量进行评价。假设 A 是符合要求的频繁项集,B 是算法输出的频繁项集,那么,召回率 = $\frac{|A \cap B|}{|A|}$,准确率 = $\frac{|A \cap B|}{|B|}$。如果召回率为 1,说明返回了所有的符合要求的频繁项集,但可能有其他不符合要求的项集也返回了;如果准确率为 1,说明返回的频繁项集都是符合要求的,但不一定返回了完整的符合要求的项集。如果两者同时为 1,说明返回的项集恰

好是符合要求的频繁项集,这时认为该算法是准确的。

在人造数据上进行了测试。用来比较的算法是 Wang 和 Fu 提出的 Zipfian[3]和 Chernoff[4]算法。部分比较结果见表 2。

表 2 算法 LIONET、Zipfian 和 Chernoff 比较的部分结果

	LIONET		Chernoff		Zipfian	
	召回率	准确率	召回率	准确率	召回率	准确率
Trans_num = 500k, k = 10, max_l = 3	1.00	1.00	1.00	1.00	1.00	0.96
Trans_num = 600k, k = 20, max_l = 4	1.00	1.00	0.98	0.98	1.00	0.88

可以看出,LIONET 算法是一个准确的算法,而 Zipfian 和 Chernoff 算法在某些情况下会出现召回率或准确率不为 1 的情况,也就是说它们不是准确的算法。

3 结 语

讨论了数据流的频繁项集的挖掘,分析了挖掘中的相关技术和性质,提出了一个采用滑动窗口技术、基于树结构的数据流频繁项集挖掘算法 LIONET,并分析了算法的优越性。

参考文献

[1] Cheung Y L, Fu A W C. Mining Frequent Itemsets Without Support Threshold: With and Without Item Constraints[A]. IEEE Transaction on Knowledge and Data Engineering[C],2004,16(9):1052-1069.

[2] Metwally A, Agrawal D, Abbadi A E. Efficient Computation of Frequent and Top-K Elements in Data Streams[A]. Proc 10th Int Conf Database Theory (ICDT'05)[C]. Heidelberg: Springer-Verlag, 2005.

[3] Wong R C W, Fu A W C. Mining Top-K Itemsets over Sliding Window Based on Zipfian Distribution[A]. Proc 2005 SIAM Int Conf Data Mining(SIAM'05)[C]. Newport Beach:SIAM Press,2005.

[4] Wong R C-W, Fu A W-C. Mining top-K Frequent Itemsets from Data Streams[J]. Data Mining and Knowledge Discovery, 2006,13(2):193-217.

[5] Leung C K-S, Khan Q I, Hoque T. CanTree: A Tree Structure for Efficient Incremental Mining of Frequent Patterns[A]. Proc 2005 Int Conf Data Mining (ICDM'05)[C]. Houston: IEEE Computer Society, 2005.

[6] Han J, Pei J, Yin Y. Mining Frequent Patterns Without Candidate Generation[A]. Proc 2000 ACM-SIGMOD Int Conf Management of Data (SIGMOD'00)[C]. Dallas: ACM Press,2000.

(上接第 72 页)

3 结 语

多核处理器的技术应用为基于共享存储的高性能计算提供了发展方向。利用了 MPI/OpenMP 混合编程模型在 SMP 集群中实现对二维结构的红外频率选择表面的 FDTD 模拟,通过分析并行系统的性能寻找解决电磁计算问题的有效方法。实验表明 MPI 和 OpenMP 的有机结合可以提高了并行程序的执行效率,为科学计算提供了一种有效的并行策略。

参考文献

[1] 赵永华, 迟学斌. 基于 SMP 集群的 MPI+OpenMP 混合编程模型及有效实现[J]. 微电子学与计算机, 2005, 22(10): 7-11.

[2] 李清宝, 张 平. 基于分布/共享内存层次结构的并行程序设计[J]. 计算机应用, 2004, 24(6): 148-150.

[3] Aversa R. Performance Prediction Through Simulation of a Hybrid MPI/OpenMP Application[J]. Parallel Computing, 2005 (31):1013-1033.

[4] 蔡佳佳, 李名世, 郑 锋. 多核微机基于 OpenMP 的并行计算[J]. 计算机技术与发展, 2007, 17(10): 87-91.

[5] 葛德彪, 闫玉波. 电磁波时域差分方法[M]. 西安:西安电子科技大学出版社, 2005.

[6] Su M F, El-Kady I, Bader D A, et al. A Novel FDTD Application Featuring OpenMP-MPI Hybrid Parallelization[A]. Proceedings of the 2004 International Conference on Parallel Processing[C]. 2004:373-379.

[7] LI Min, ZHOU Jian. Parallelization of the Electromagnetic Simulations on Grid-Enabled MPI Library[A]. Proceedings of ICCSE[C]. 2008: 792-795.