均衡时空挖掘数据流中频繁项集

宋奎勇 任永功 寇香霞
(辽宁师范大学计算机与信息技术学院 大连 116029)

摘 要 数据流具有流动性、连续性以及项分布不均衡性等特点,挖掘数据流中频繁项集是一项意义重大且具有挑战性的工作。提出一种均衡时空挖掘数据流中频繁项集算法——Bala_Tree,Bala_Tree 实现一遍扫描数据流、快速簇更新、周期树结构重构以及基于经典算法挖掘频繁项集。实验表明,此算法能快速扫描和更新数据,合理利用内存以及精确获得频繁项集,Bala_Tree 算法优于其他同类算法。

关键词 数据流,频繁项集,均衡,Bala_Tree

Balanced Space-time Frequent Itemsets Mining over Data Stream

SONG Kui-yong REN Yong-gong KOU Xiang-xia
(School of Computer and Infomation Technology, Liaoning Normal University, Dalian 116029, China)

Abstract Data stream has characteristics of the flow, continuity, and the unbalanced distribution of item. Minging frequent itemsets over data stream is a significant and challenging work. Presented a balanced space-time algorithm for mining frequent itemsets over data stream——Bala_Tree. The algorithm can only scan data stream once, make rapid cluster updates, periodical tree reconstruction and mine frequent itemsets based on classical algorithm. Experiments show that the algorithm can quickly scan and update data, realize the rational use of memory, accurate access to frequent itemsets. Bala_Tree algorithm is superior to other algorithms.

Keywords Data stream, Frequent itemsets, Balance, Bala_Tree

1 引言

随着通信技术的迅猛发展,各种高端设备如传感器网络、监测环境、天气预测卫星以及网络服务器等以流形式实时传送大量数据,若此数据流是连续数据序列,并且数据元素高速到达,则传统静态数据库挖掘算法对此无能为力。探索一种高效数据流挖掘方法是当前数据挖掘研究的难点之一。

为了处理不断到来的数据,时间窗口被广泛应用,根据文献[1]挖掘数据流有3种窗口模式:界标窗口、衰减窗口、滑动窗口。界标窗口关注整个数据流中的数据,并通过对整个历史数据分析得到全局性频繁模式,由于数据流的无限性,此种窗口模式必须结合快速的归纳技术及合适的数据淘汰技术才能真正适合数据流挖掘,其代表性算法是 Lossy counting[3]。在衰减窗口处理模式中,每个事务对应一个权值,而且这种权值随时间的增加而减少。使用权值表明科研人员对最新到达的数据更感兴趣,认为最新到达的数据包含更重要的信息,其代表性算法是 FP-stream[4]。滑动窗口关注最近流入窗口的所有事务,它的挖掘结果是某段时间内的局部频繁模式。滑动窗口处理模式易于理解、设计简单,得到了广泛研究和应用,其典型算法是 Moment[5]。

近年来,大量数据流挖掘算法被提出,通常分成两大类:精确算法和近似算法。精确算法能够挖掘出支持度大于最小支持度的所有频繁模式,如 Moment、DSTree[6] 和 CPS-Tree[7]。近似算法为了保证挖掘效率,损失了可容忍精确度,大部分挖掘全局模式算法是近似算法,如 Lossy counting、FP-Stream 和 Msw[8]。

DSTree 算法用模式树维护数据流最近时间窗口内的模式信息,模式树节点包括多个计数域,计数域记录最近时间窗口数据分段内的节点支持数,该方法可以精确挖掘滑动窗口内数据的频繁模式。Msw 算法可以挖掘任意滑动时间窗口的频繁模式,该方法周期性删除滑动窗口树上过期及不频繁的模式分支,从而降低滑动窗口树的空间复杂度和维护代价。CPS-Tree 介绍一种动态重构建树方法,在单遍扫描前提下建立按照支持度降序排列的树,节省了内存开销,提高了挖掘效率。

然而,DSTree 算法有几方面限制。首先,它以字母表顺序存储事务,不能保证一个高度压缩树结构,通常我们考虑以支持度降序树结构有最大压缩比,既可避免大量存储开销,又可减少搜索空间,提高挖掘效率。其次,如 DSTree 描述,在更新节点计数阶段,为了节省时间开销,没有轮转树中所有节点,这样树中保留一些无效和错误的垃圾节点,随着数据不断流入,模式树空间复杂度将迅速增加,算法可扩展性差。对于

到稿日期:2011-01-28 返修日期:2011-04-22 本文受国家自然科学基金(60603047),辽宁省科技计划项目(2008216014),大连市优秀青年科技人才基金(2008J23JH026),教育部留学回国人员科研启动基金项目资助。

宋奎勇(1979—),男,硕士生,主要研究方向为数据挖掘;任永功(1972—),男,博士,主要研究方向为数据挖掘、图像处理技术等,E-mail:rengy@dl.cn(通信作者);寇香霞(1980—),女,硕士生,主要研究方向为数据挖掘。

Msw 算法，一方面它也是按照字母表顺序压缩树结构；另一方面，算法通过记录最近包含该节点的事务 Tid 来对树结构定期剪枝，然而此方法并不能把所有过期事务都剪掉，从而造成结果不准确。CPS-Tree 算法按照支持度降序建树，但是算法树中间结点和叶子节点结构不一样，造成重构方法过于复杂，不易理解。

由此，本文提出一种均衡时间空间开销算法 Bala_Tree：利用滑动窗口策论，对节点增加多个计数域；建立项列表 I（按字母表顺序）对节点滑动更新，当簇更新后，对树结构按照 I_{sort}（按支持度顺序）中项顺序重构；最后用改进 FP-Growth[2] 算法对 CPS-Tree 树挖掘，保证其挖掘精度并提高了效率。

2 预备知识

定义 1 数据流 $DS=\{T_1, T_2, \cdots, T_i, \cdots\}$ 是一个大小无限的事务数据序列，其中 T_i 代表第 i 个事务。每一个事务包含唯一事务标识 Tid 和一些项。

定义 2 一个滑动窗口 SW 对应一个连续的簇序列 $\langle P_1, P_2, \cdots, P_k\rangle$，它所容纳的簇数目是一个定值 k，簇将数据流 DS 分段，每一簇对应一个数据流子序列且事务数一定。随着新数据到来，滑动窗口以簇为单位不断更新，每进入一个新簇，最旧簇被删除，滑动窗口中数据不断变化和更新。如表 1 所列，一簇包含 3 个事务，滑动窗口 SW 可设定任意簇数。随着数据流动，新簇流入，旧簇流出。

表 1 数据流

Pane	tid trans	direct
P1	1 ace 2 bdf 3 abd	
P2	4 bde 5 bd 6 ac	DataStreams
P3	7 d 8 de 9 bdf	
	.....	

定义 3 项集 X 在 SW 中出现的频率（即 SW 中包含 X 的事务 T 的个数）称为 X 在 SW 中的支持度。若项集 X 的支持度不小于用户给定的最小支持度，则称 X 为频繁项目集。

3 均衡频繁项集挖掘算法 Bala-Tree

首先介绍 Bala-Tree 基本结构，接着说明如何在 Bala-Tree 中插入、删除簇中事务，以及如何重构 Bala-Tree，最后解释用改进 FP-Growth 算法挖掘频繁项集。

3.1 Bala-Tree 的结构

Bala-Tree 包括项表 I 和一棵树，I 包括 3 个域：item，count，node-links。Item 域列出当前滑动窗口中的所有项，count 域记录滑动窗口中的项支持度，当簇被删除或者新簇被插入，count 会相应改变，Bala-Tree 每次重构都会按照 count 从高到低重新排序。node-links 域链接树中和 item 相同的节点，把树中相同节点项按照插入顺序链接起来，既用于簇更新，又用于挖掘频繁项集。树中节点都有一个计数列表，计数列表用来记录相应簇中节点项支持度，只是每个节点较 DSTree 中节点多一个指针域，这个指针域的作用是把树中所有相同节点项和项表 I 中相同的项链接起来，这样在进行计数列表更新的时候，项表 I 中的项就可以更新树中所有相同节点项。我们把表 1 中簇 P_1 和簇 P_2 的事务插入 Bala-Tree 中，如图 1 所示，按照字母表顺序插入簇 P_1 中 3 个事务，节点簇计数列表第一列值相应变化。项表 I 中 count 域相应变化。按照字母表顺序插入簇 P_2 中 3 个事务，节点簇计数列表第二列值相应改变，项表 I 中项 count 域等于树中相同项计数总和。

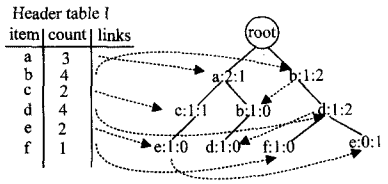


图 1 Bala-Tree 结构

算法 1 ConstructBala_Tree(P_i, root)

输入：第 i 簇事务 P_i；树根节点 root；

输出：以 I 项顺序排列树；

1 For(P_i 中每一个事务 T_k){

2 For(T_k 中每一项 I_j){

// 用二叉树结构存贮树节点，左节点存储节点的孩子，右节点存储节点的兄弟；

3 If（I_j 等于 Node 的左孩子）更新相应簇支持度；

4 Else {循环查找 Node 的右孩子，

查找成功，更新相应簇支持度，若没找到，创建新节点作为右孩子链接到尾部；并链接到 I；

5 }

6 }End For //一个事务插入完毕；

7 指针返回到头节点，等待插入第二个事务；

8}End For // P_i 簇中事务插入完毕；

3.2 树的更新和窗口滑动

Bala-Tree 把数据流划分为大小相等的簇，一簇一簇滑动。吸取 DSTree 方法的优点，对树中节点设置计数列表记录滑动窗口中每一簇中项的支持度。通过项头表 I 中 node-links 域按顺序查找树中相同节点，并且滑动每个节点计数列表，覆盖计数列表第一列，空出计数列表最后一列为最新簇滑入做准备。通过这种方式从上到下遍历 I 中每一个项，从而达到更新所有节点的目的。在此过程中，有一些快捷方法被提出，加快了更新速度，如下面性质 1 所言。

性质 1 在滑动更新过程中，若某一个节点在删除最旧簇后，支持度为零，则把以此节点为根的所有节点剪掉。

证明：无论是以字母表顺序还是支持度降序顺序建树，任何节点的支持数和时间戳都不小于其子孙节点的支持数和时间戳，若此节点支持度为零，则其子孙节点支持度必为零，证毕。

如图 2(b)中，对图 2(a)中 Bala-Tree 删除第一簇，从项表 I_{sort} 第一项 b 开始，b. links 域链向树中第一个节点，该节点 P_2 计数覆盖 P_1 计数，P_2 计数清零，为第三簇插入做好了准备。处理完此节点，通过 next 域找到下一个节点进行同样操作，直到 next 域为 null。若覆盖后，节点支持度为零，则删除此节点，如图 2(a)中 e 节点直接删除。完成后转向项表 I_{sort} 中下一项进行同样操作，直到 I_{sort} 最后一项。插入新簇如图 2(c)所示，按照图 2(b)中 I_{sort} 顺序插入第三簇事务，若树中相应位置存在此节点，节点簇计数列表第二列值相应改变，若不存在，建立新的节点并且链接到树中和列表中。

3.3 树重构

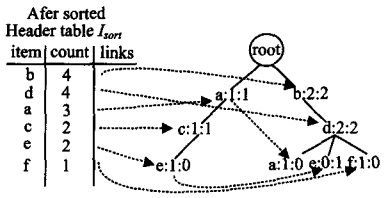
让更多的节点共享前缀，按照支持度降序建树是最节省

空间的，但是一般来说，按照支持度建树需要扫描数据库两次，第一次扫描统计项支持度，第二次扫描建树。对于数据流来说扫描两次不现实，可以通过树重构思想扫描一遍就可以达到按支持度降序建树的目的，相关论文提出两种方法：一种是分支排序方法 PAM[10]；另一种是路径调整方法 BSM[9]。

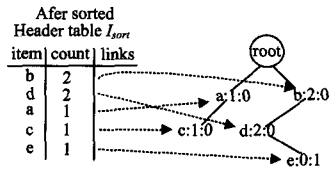
PAM(Path Adjusting Method)通过递归交换相邻节点来达到重新排序目的，也就是使用冒泡排序方法。PAM 方法时间复杂度是 $O(n^2)$，适合于较小和失调不严重的树。

BSM(Branch Sorting Method)使用数组来对树中每一个分支进行重新排序，从根节点开始，根节点每一个孩子都是一个分支。首先把一个分支路径所有节点取出放在一个临时数组里，树中相应节点支持度减 1，若支持度为零则删除此节点，把数组中节点重新排序后插回树中，如有已排序重合路径，则结合到一起。若发现某一个分支，路径不需要重新排序，则滑过，并且扩展到整个分支路径。依次处理分支，最后得到按支持度降序排序的树。BSM 方法时间复杂度是 $O(n\log_2 n)$，更适合于较大和失调严重的树，Bala-Tree 选择 BSM 方法对树进行重构。

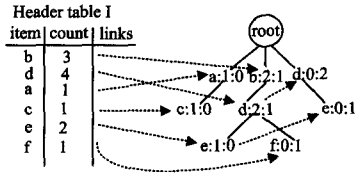
图 2(a)是第一次重构，首先对图 1 中 I 重新按照支持度由高到低排序，排好序后为 $\{b,d,a,c,e,f\}$，如图 2(a)中 I_{sort} 所示。第一个分支路径 ace，ace 和 I_{sort} 顺序相同不用取出，分支路径 abd 和 I_{sort} 顺序不同，取出放入数组中，相应节点支持度减 1。排序后为 bda 插入树中，第二个分支 bdf、bde 为排好序，不用取出和 bda 合并。完成了第一次重构。图 2(d)是第二次重构，首先重新排序图 2(c)中 I 得到图 2(d)中 I_{sort}，第一个分支 ac 满足 I_{sort} 顺序，第二个分支取出 bde，排序为 dbe 插入树中，bdf 同样需要取出并以 dbf 插入，第三分支不用取出。



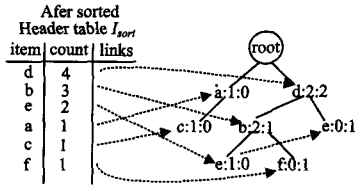
(a) Bala-Tree 第一次重构



(b) Bala-Tree 删除第一簇



(c) Bala-Tree 插入第三簇



(d) Bala-Tree 第二次重构

图 2　Bala-Tree 树构建过程

Bala-Tree 有效地利用了计数列表和树重构技术，不但实现了数据流实时更新，而且在一遍扫描数据流前提下构建了按支持度排序的压缩树。但是考虑到树重构要花销不少时间，所以适合的重构次数是十分必要的，这里提出两种策论：一种是按照簇重构；另一种是给定一个空间阈值，可以根据实际情况选择重构时间。

算法 2　Restruct_Tree(T, I)

输入：T(未重构树)，I(未排序项表)；

输出：T_{sort}(已重构树)和 I_{sort}(已排序项表)；

1 按照支持度对 I 降序排列得到 I_{sort}；
2　For(T 中每一个分支 B_i){
3　　For(B_i 中每一个未处理的路径 P_j)
4　　If(P_j 是分支 B_i 中一个已排序路径)
5　　{不用取出，扩展到分支 B_i 的其他路径，逐层递归调用，若需要排序调用 Sort_Path(P_j)；}
6　　Else Sort_Path(P_j)；
7　　End If
8　End For
9 End For
10 输出 T_{sort} 和 I_{sort}

Sort_Path(Q)

1 从树中取出路径 Q，相应树中簇计数减 1，删除计数为 0 的节点；
2 把路径 Q 中节点放入数组，按照 I_{sort} 重新排序；
3 把排序后 Q 插入树中；

3.4　挖掘频繁项集

通过重构 Bala-Tree 得到按照支持度降序排序的树，这种树结构类似于经典 FP-Growth 算法。FP-Growth 算法通过项头表频繁项找到树中所有相同的节点，构造此频繁项的条件模式基，通过条件模式基产生条件模式树，挖掘模式树产生所有频繁项集。结合 Bala-Tree 的特点，对 FP-Growth 算法稍做改进，如性质 2 所言。

性质 2　给定一个数据流 DS，基于滑动窗口 BW 上的 Bala-Tree 以及一个给定的最小支持度 §，则 I_{sort} 中支持度小于 § 的项及此项后面的所有项不可能是频繁的，不属于当前滑动窗口频繁项集空间，可直接从 I_{sort} 中删除。

证明：滑动窗口上频繁项集每一项肯定是频繁的，每一项支持度必须大于等于最小支持度 §，Bala-Tree 是按照支持度降序排列的，只有最后一项支持度大于等于最小支持度，才能保证此项前缀所有项都是大于等于最小支持度的，证毕。

4　实验结果与分析

用 JAVA 实现了 DSTree、CPS-tree 和 Bala_Tree 算法，所有的实验在相同环境下进行。实验环境为：Windows XP 系统，2.66GHz 奔腾四处理器，512MB 内存。试验中，人工数据集 T40I10D100k 和真正数据集 Connect-4 来自文献[12]，T40I10D100k 包含 100,000 个事务，942 个项，是一个相对稀疏数据集。Connect-4 包含 67,557 个事务，129 个项，是一个稠密数据集。在试验中主要评估 Bala_Tree 算法的时间和空间有效性。

首先考虑空间开销，设定 $p=10$k 事务，对于数据集 T40I10D100k，设定滑动窗口大小为 3、5、7、9，如图 3 所示，算法 DSTree 需要内存最多，CPS-tree 次之，Bala_Tree 所需内存最少，其中算法 DSTree 要多出很多。对稠密数据集 Connect-4

的实验如图4所示，与数据集 T40I10D100k 的测试结果一样，Bala_Tree 所需内存还是最少。DSTree 算法利用字母表顺序建树，并且树中保留大量过期垃圾节点，造成空间开销巨大。CPS-tree 和 Bala_Tree 算法都利用了重构技术，然而 Bala_Tree 算法更简洁，所以在任意大小滑动窗口下，无论稀疏或者稠密数据集，算法 Bala_Tree 都显示出空间优越性。

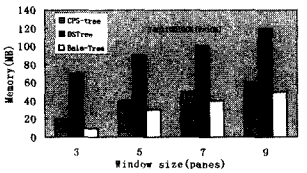


图3 在数据集 T40I10D100k 下比较内存大小

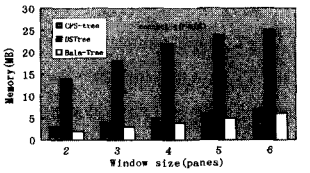


图4 在数据集 Connect-4 下比较内存大小

接下来试验比较 DSTree、CPS-tree 和 Bala_Tree 算法的时间开销。在图5中，对于数据集 T40I10D100K，设定滑动窗口 $w=4$，支持度分别为(25,20,15,10)；在图6中，对于稠密数据集 Connect-4，设定滑动窗口 $w=2$，支持度分别为(99,97,95,93,91)，可以看到无论是在较高支持度还是较低支持度下，Bala_Tree 算法时间开销都小于 DSTree 和 CPS-tree 算法，并且随着支持度降低差距越来越大。虽然 Bala_Tree 算法重构树花费了额外花销，然而实验中发现频繁项集挖掘阶段是最耗费时间的，基于支持度降序排列树利用 FP_growth 算法挖掘频繁项集效率非常高，在重构次数有限的情况下完全可以忽略重构时间花销，所以 Bala_Tree 算法总计时间花销明显小于 DSTree 算法。

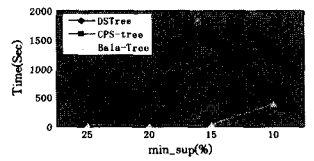


图5 在数据集 T40I10D100k 下比较时间大小

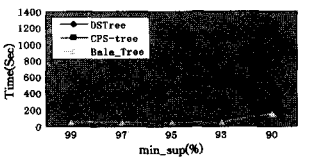


图6 在数据集 Connect-4 下比较时间大小

结束语 提出一种新颖的数据流中挖掘频繁项集的算法 Bala_Tree，Bala_Tree 算法对树中节点设置簇计数器，并且引入树重构思想得到一棵基于支持度降序树，不但节省了内存开销，而且适合利用 FP_growth 方法挖掘频繁项集。Bala_Tree 算法达到了时间和空间相对均衡，优于其他同类算法。

参 考 文 献

[1] 刘旭，毛国君，孙岳，等. 数据流中频繁闭项集的近似挖掘算法[J]. 电子学报，2007，35(5)

[2] Han J，Pei J，Yin Y. Mining Frequent Patterns without Candidate Generation[C]// Proc. ACM-SIGMOD Int'l Conf. Management of Data. 2000

[3] Manku G S，Motwani R. Approximate frequency counts over data streams[C]// Proceedings of the 28th international conference on very large data bases. Hong Kong，2002

[4] Giannella C，Han J，Pei J，et al. Mining frequent patterns in data streams at multiple time granularities[M]. Data Mining：Next Generation Challenges and Future Directions. AAAI/MIT Press，2004

[5] Chi Y，Wang H，Yu P，et al. Moment：maintaining closed frequent itemsets over a stream sliding window[C]// Proceedings of the 4th IEEE international conference on data mining. Brighton，UK，2004：59-66

[6] Leung C K-S，Khan Q I. DSTree：a tree structure for the mining of frequent sets from data streams[C]// Proc. ICDM. 2006：928-932

[7] Khairuzzaman S，Ahmed C F，Jeong B-S，et al. sliding window-based frequent pattern mining over data streams[J]. Information Sciences，2009，179：3843-3865

[8] 李国徽，陈辉. 挖掘数据流任意滑动时间窗口内频繁模式[J]. 软件学报，2009，10(19)：2585-2596

[9] Tanbeer S K，Ahmed C F，Jeong B-S，et al. CP-tree：a tree structure for single-pass frequent pattern mining[C]// Proc. PAKDD. 2008，5012：1022-1027

[10] Koh J-L，Shieh S-F. An efficient approach for maintaining association rules based on adjusting FP-tree structures[C]// Proc. the DASFAA. 2004：417-424

[11] Blake C L，Merz C J. UCI Repository of Machine Learning Databases[D]. University of California-Irvine，Irvine，CA，1998

(上接第166页)

[9] 中华人民共和国教育部. 中华人民共和国国家标准 GB2312-80《信息技术、信息交换、用汉字编码字符集、基本集》[S]. 1980

[10] 左双勇. 基于汉字字型结构的文本数字水印算法[J]. 计算机与现代化，2010，1006-2475，08-0029-04，29-32

[11] Sion R，et al. Watermarking Relational Databases[R]. CERIAS. 2002

[12] 张金永. 基于非数值属性的数据库数字水印算法研究[D]. 兰州大学，2010

[13] Ido D Lillian L，Fernando C N P. Similarity-based Models of Word Co-occurrence Probabilities[J]. Machine Learning，1999，34：43-69

[14] Jiang J，Conrath D. Semantic Similarity Based on Corpus Statistics and Lexical Taxonomy[C]// Proceedings of International Conference Research on Computational Linguistics. 1997：19-33

[15] 黄果，周竹荣. 基于领域本体的概念语义相似度计算研究[J]. 计算机工程与设计，2007，28(10)：2460-2463

[16] 吴健，吴朝晖，李莹，等. 基于本体论和词汇语义相似度的 Web 服务发现[J]. 计算机学报，2005，28(4)：595-602

[17] 夏天. 汉语词语语义相似度计算研究[J]. 计算机工程，2007，33(6)：191-194

[18] 陈杰，蒋祖华. 领域本体的概念相似度计算[J]. 计算机工程与应用，2006，42(33)：163-166

[19] 赖院根，等. 概念语义相似度计算与参数估计[J]. 情报杂志，2009，28(8)：148-152