文章编号: 1001-0920(2009)06-0832-05

一种基于变尺度滑动窗口的数据流频繁集挖掘算法

朱小栋[1], 黄志球[1], 沈国华[1], 袁 敏[1,2]
(1. 南京航空航天大学 信息科学与技术学院, 南京 210016; 2. 湘南学院 计算机科学系, 湖南 郴州 423043)

摘 要: 基于传统滑动窗口机制的数据流频繁集挖掘算法较多地考虑快速且精确的效果,而较少考虑数据流的时变特性. 对传统的滑动窗口机制进行改进,同时考虑数据流的海量特性和时变特性,提出一种基于变尺度滑动窗口机制的数据流频繁集挖掘算法 V-Stream. 该算法采用事务链表组的概要数据结构,能够根据数据流的数据分布变化自适应调整窗口大小. Eclipse 上的仿真实验结果表明,V-Stream 相比 Manku 算法提高了挖掘数据流频繁集的时间与空间效率.

关键词: 数据流; 数据挖掘; 滑动窗口; 频繁集; 关联规则
中图分类号: TP311; TP18 **文献标识码**: A

Variable slide window based frequent itemsets mining algorithm on large data streams

ZHU Xiao-dong[1], *HUANG Zhi-qiu*[1], *SHEN Guo-hua*[1], *YUAN Min*[1,2]
(1. College of Information Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China; 2. Department of Computer Science, Xiangnan University, Chenzhou 423043, China. Correspondent: ZHU Xiao-dong, E-mail: zhuxd@nuaa.edu.cn)

Abstract: Considering the problem that many traditional time window based data streams mining algorithms have good efficiency upon synthetic data streams but fail to be applied to real data streams in the fields such as celestial bodies movement and high energy particles outbreak, a variable slide window based frequent itemsets mining algorithm V-Stream upon data streams is proposed. The algorithm could self-adaptively adjust the size of time windows. In the algorithm, transaction list group is adopted as synopsis data structure. The experiments results in Eclipse indicate that the V-Stream algorithm is more effective than the Manku algorithm in term of temporal and spatial performance.

Key words: Data stream; Data mining; Slide window; Frequent itemsets; Association rules

1 引 言

数据流挖掘是近年智能数据分析发展的一个新的研究阶段,不仅因为许多应用领域,如传感器网络、互联网的访问、计算机网络监控、金融股市和电信部门等产生大量高速实时的数据流,使得传统的数据挖掘技术不能适应这种新的数据形式,而且,对数据流进行数据挖掘已成为这些领域的迫切需要. 起源于 20 世纪 90 年代的关联规则是数据挖掘的一个核心课题. 与基于统计回归等数学方法进行数据分析不同,关联规则显得隐蔽而难以直接发现. 挖掘关联规则的关键步骤是从数据中得到满足一定支持度的频繁项集. 经典的 Apriori 算法通过分析购物篮事务数据得到该事务数据库上的频繁项集[1]. Han 等[2]提出的 FP-Growth 算法进一步提高了挖掘频繁项集的性能. 然而,这些频繁项集挖掘算法都需要多次扫描数据,不能满足数据流挖掘的要求.

近年来,国内外学者在数据流的频繁项集挖掘上提出了许多算法,他们都是针对传统数据挖掘技术的多次扫描数据进行改进或者提出新的思路. 如 Manku 等[3]首次使用近似算法进行数据流的频繁项集挖掘,该方法仅需一次扫描数据,并且通过牺牲挖掘结果的精确性来提高挖掘的效率. 但该算法挖掘的是整个历史数据流上的频繁项集,而在许多情况下,用户仅关心最近数据上的频繁项集;且该算法使用缓存技术,占用大量的内存空间. 由于用户对最近的数据兴趣度更高,一些基于此要求的数据处理

收稿日期: 2008-06-06; **修回日期**: 2008-11-15.
基金项目: 航空科学基金项目(01I52058); 湖南省自然科学基金项目(05JJ40102).
作者简介: 朱小栋(1981—),男,安徽太湖人,博士生,从事数据仓库与数据挖掘、语义 Web 的研究; 黄志球(1965—),男,南京人,教授,博士生导师,从事工程数据库、软件度量与测试等研究.

机制,如滑动窗口方法[4,5]和数据衰减方法[6]被成功地应用于数据流挖掘.文献[7]提出一种改进的 FP(Frequent Pattern)树结构,能够有效挖掘具有长频繁模式的数据流.

然而,目前数据流挖掘算法的特点是注重快速实时,大都忽略了真实数据流的"时变"特性.数据流的"时变"特性体现在,数据流到达的速率并不是固定不变的,相同历史时间段内的元组数目不一定相同,有时甚至相差巨大.例如,探测太空中高能粒子与天体运动的规律,或者地壳运动的粒子波动规律,需要在特定区间内维护一个滑动窗口来计算这些事件的数量.但高能粒子的爆发可能持续几个毫秒,也可能持续几小时、几天或更长时间,因而窗口大小很难确定.此外,人们往往仅对最近历史时期的数据流兴趣更大,所以,目前许多基于滑动窗口的数据流挖掘算法在人工输入的数据流上具有较好的特性,而在真实数据流上往往效率很低.

从上述角度出发,本文提出一种基于变尺度滑动窗口机制的频繁集挖掘算法 V-Stream.该算法根据数据本身的变化在多个尺度时间窗口上进行自适应调整,采用近似技术一次扫描数据快速得到频繁项集,同时采用事务链表组的概要数据结构.理论分析和仿真实验均表明,V-Stream 在时间和空间效率上得到了提高.

2　基本概念和技术

2.1　变尺度滑动窗口

对于数据流挖掘,在极端情况下,无界输入流需要无界的内存,显然这是不实际的.因此希望在实际的无界流处理中采用一种"窗口策略"以限制每个输入流存储下来的元组数量.滑动窗口是指在数据流上设定的一个区间,该区间只包括数据流最近的部分数据.随着新数据的到来,窗口向前移动,用新数据替换旧数据.因此,滑动窗口可以看作是数据流有限部分的历史性快照,同时利用滑动窗口的特点来满足数据流的实时性和无界的要求.

滑动窗口作为数据流有限部分的历史性快照,可以将滑动窗口划分为基于时间的滑动窗口和基于元组的滑动窗口.这两种滑动窗口通常假设窗口大小不变.基于时间的滑动窗口根据其更新方式又可以分为连续更新滑动窗口和周期更新滑动窗口.设 $s[t-T,t]$ 为数据流 DS 上的一个滑动窗口,如果 t 始终保持为 DS 中最新元素到达的时刻,则称 $s[t-T,t]$ 为连续更新滑动窗口.设 $s[t-T,t]$ 为数据流 DS 的在 t 时刻的滑动窗口,Δt 是一个时间间隔,如果 $s[t-T,t]$ 仅在每个 Δt 时间间隔的结束时刻发生改变,并且第 k 个 Δt 时间间隔结束时刻变为 $s[t+k\Delta t-T,t+k\Delta t]$,则称 $s[t-T,t]$ 为周期更新滑动窗口.

在以往数据流挖掘算法中,很多算法基于不变的滑动窗口机制,在时空复杂度等性能方面取得了较大成果.但这些技术大都忽略了具体问题中数据流的时变特性,当数据分布特征不断变化时,很难实现模型的自适应调整[4,5,8].Giannella 等[9]提出了对数时间窗口,它是一种特殊的变尺度时间窗口.其主要优点是使用了不同粒度的时间窗口,且时间窗口的大小由用户指定,如一年为单位的粗粒度时间窗口和一刻为单位的细粒度时间窗口.相比自然等长时间窗口需要 35 136 个单位,该方法仅需要 17 个窗口单位.这种对数时间窗口不能根据数据流的时变特性改变窗口大小,而是预先设定的对数窗口,故不是真正意义的变尺度窗口.

变尺度时间窗口是根据数据流的流速变化以及数据分布的变化自适应地调整时间窗口大小,以达到最小的内存空间和处理时间的消耗.

2.2　问题形式化描述

数据流是以时序不断出现的有序的项目序列,与传统的静态数据不同,数据流是连续的、无边界的,通常高速地出现,并且随时间变化数据分布特征不断变化.在一般的变尺度滑动时间窗口下,数据流频繁项集挖掘可以形式化描述如下:

令数据流 $DS=B_{a_1}^{b_1},\cdots,B_{a_i}^{b_i},B_{a_{i+1}}^{b_{i+1}},\cdots,B_{a_n}^{b_n}$ 是由无限的事务块构成的序列,每个事务块关联一个时间窗口 $[a_k,b_k]$,令 $B_{a_n}^{b_n}$ 是最近的事务块.每个事务块 $B_{a_k}^{b_k}$ 是由一组事务构成的集合,$B_{a_k}^{b_k}=[T_1,T_2,\cdots,T_m]$,假设每个块的事务数不一定相等.因此,数据流在时间域 $[a_i,b_n]$ 上的长度 length(DS) 是 $\text{length(DS)}=|B_{a_i}^{b_i}|+|B_{a_{i+1}}^{b_{i+1}}|+\cdots+|B_{a_n}^{b_n}|$,其中 $|B_{a_k}^{b_k}|$ 表示集合 $B_{a_k}^{b_k}$ 的基数.给定最小支持度 s,一个项集 X 在时间域 $[a_i,b_i]$ 上是频繁项集,当且仅当 $\sum_{t=a_i}^{b_i}\text{support}(X)\geq s\times|B_{a_i}^{b_i}|$.所以给定一个最小支持度,数据流上频繁项集挖掘问题规约到使用尽可能少的时间和空间消耗来发现一定时间域上所有的频繁项集.变尺度滑动窗口的窗口尺寸是根据数据流的流速变化来确定的.当流速很快时,及时缩小滑动窗口的长度;而当流速很慢时,实时地增长滑动窗口的长度.

2.3　概要数据结构

数据流挖掘在考虑时间效率的同时,也需要尽可能少地使用内存空间.Borgelt[10] 在人工合成数据集和真实数据集上的实验表明,事务链表组数据结构能表现出比 Apriori 算法[1] 和 FP-growth 算

法[2] 更好的空间性能. 为此,本文在 V-Stream 算法中,引入一种基于事务链表组的概要数据结构,记为 TLG. 相关定义如下:

1) 事务链表组中的每个元素 entry 定义为(item, f, Δ, list). 其中:item 为数据流中的一个单项, f 记录了 item 的频繁度, Δ 代表 f 中最大可能的出错数,而 list 记录的是以 item 为首项的项集链表. 这些 entry 在本文中记为表头元素.

2) list 中的每个元素 entry 定义为(itemset, f, Δ). 其中:itemset 为数据流中以该事务链头元素为首项的项集, f 记录了 itemset 的频繁度, Δ 代表 f 中最大可能的出错数.

例如一个头元素 entry 为 $(a,5,5,\text{list})$,list 中每个项集的 entry 又为 $(acd,5,3)$, $(abd,10,3)$ 等,如图 1 所示.

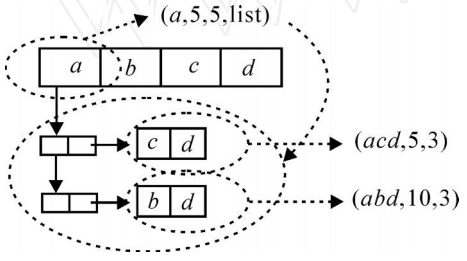


图 1 事务链表组 TLG示例

为进一步降低算法运行时的空间消耗,在 V-Stream 算法中,进一步改进 V-Stream 的概要数据结构,记为 TLG Ⅱ,如图 2 所示. 其思路是基于自然数中素数的特征,在该数组中,每个项用一个唯一的素数表示,每个项集用该项集所组成的项所对应的素数乘积表示. 在建立事务链表组 TLG Ⅱ时,按照表 1 转换相应的项与项集.

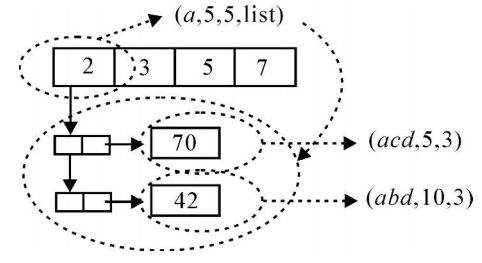


图 2 事务链表组的改进 ——TLG Ⅱ

表 1 事务数据项与素数的映射

项/项集	a	b	c	d	…	ab	…
素数/素数积	2	3	5	7	…	6	…

这种采用素数映射的方法在许多文献中被采用,如文献[11]. 在 TLG Ⅱ中,项与项集用相应的素数和素数积表示,对于大量持续的数据流而言,能更有效地减少内存空间的消耗. 当然,随着项的数目的增加,素数积会逐渐增大. 如果整数存储为 4 个字节,则 TLG Ⅱ的方式可支持长度为 10 的项集,足够保证常规的频繁集挖掘. 但当项集长度超过 10 时,则不能采用 TLG Ⅱ的数据结构,而应选用 TLG 的方式.

3 V-Stream 算法

V-Stream 算法的主要思想是:对持续不断的数据流,根据流速确立不同尺度的时间窗口,建立并维护一个事务链表组的概要数据结构(如后面的图 3(a) 所示). 同时对数据流中的每个事务计数,然后对事务链表组进行更新和修剪. 一旦从客户端发送对频繁项集的请求时,V-Stream 算法将根据事务链表组输出挖掘结果. 以 TLG 作为数据流的概要数据结构,V-Stream 算法描述如下:

Input: DS = $B_{a_i}^{b_i}, B_{a_{i+1}}^{b_{i+1}}, \cdots, B_{a_n}^{b_n}$;

user-defined support threshold s; error rate ε.

Output: A set of frequent items and itemsets

1) Initialization: TLG ← ∅; Items ← ∅; Itemsets ← ∅; $b_{\text{current}} \leftarrow 1$

2) Foreach Batches

$e \leftarrow$ next itemset;

3) UPDATE($e, f, \Delta, s, \varepsilon$);

4) if BatchEnd do

PRUNE(TLG, b_{current});

5) $b_{\text{current}} \leftarrow b_{\text{current}} + 1$

6) endif

算法中的变量含义如下:s 是用户定义的支持度阈值或称最小支持度;ε 是近似计数的误差,一般设为最小支持度的十分之一或百分之一;b_{current} 是桶的标记;变量 Item 和 Itemset 分别表示输出的频繁项和频繁项集. 下面给出算法的详细过程.

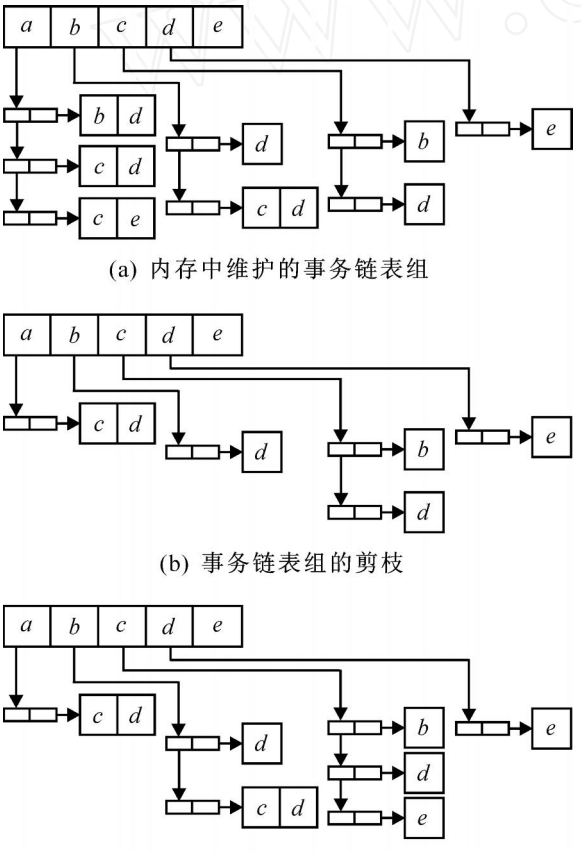
Step1: 每个不同尺度的时间窗口对应一个桶,每个桶存放一个事务数据块,如 $B_{a_i}^{b_i}$. 将读入的事务数据流 $B_{a_i}^{b_i}, B_{a_{i+1}}^{b_{i+1}}, \cdots, B_{a_n}^{b_n}$ 存到该时间窗口所对应的桶中. 每个桶的宽度不需要相同,每个桶最多存放 $w = [1/\varepsilon]$ 个事务. 每个桶有唯一的标志 b_{current},初始值为 1. 当跳转到下一个时间窗口时,桶的标志增 1.

Step2: 为数据流中到当前时刻出现的每个单项创建一个单项数据链表 e. 每个 entry 为(item, f, Δ, list),list 中保存的是以首项为前缀的项集,list 中元素 entry 定义为(itemset, f, Δ).

Step3: 当一个新的项集 e 到达时,如果事务链表组中存在 e,则找到 e 的首项所对应的事务链,使 e 的频度加 1,并将 e 的 Δ 值更新为事务链表中 e 的超集的最小 Δ 值. 当一个新的项集 e 到达时,如果事务链表组中不存在 e,则找到 e 的首项所对应的事务链,将 $(e, f, b_{\text{current}} - 1)$ 加入事务链表组中. 另外在当

前事务链中更新现存的 e 的子项集频度. 如果 e 是唯一单项,则需要建立新的头元素.

Step4: 剪枝操作在时间窗口跳转时发生,对于任何项 (e, f, Δ),如果 $f + \Delta \leq b_{\text{current}}$,则删除. 删除方法为扫描事务链表组 TLG 的每个事务链,如果某事务链中有项集 e 的 entry,且满足 $f + \Delta \leq b_{\text{current}}$,则将 e 从 TLG 中脱离. 把该事务链中的所有脱离的项集去除首项,并以该条事务链的所有脱离项集来创建一个临时的事务链表组 TLG′,同时更新频度. 一条事务链表扫描完毕后,把 TLG′合并到 TLG,在合并过程中更新相应项集的频度. 合并完后回收 TLG′占用的内存,最后得到新的事务链表组 TLG. 依次循环,直到挖掘完最后一个事务链表后结束. 事务链表组的剪枝与合并如图 3(b),3(c) 所示.



(a) 内存中维护的事务链表组

(b) 事务链表组的剪枝

(c) TLG与TLG′的合并

图 3　事务链表组 TLG

Step5: 当收到对挖掘频繁项集结果的请求后,输出满足 $f \geq (s - \varepsilon) \mid B_{a_i}^{b_i} \mid$ 的 entry 的项集. 输出方法为扫描事务链表组的一个内存副本 TLG^* 的每个事务链,如果某事务链中有项集 e 的 entry,且满足 $f \geq (s - \varepsilon) \mid B_{a_i}^{b_i} \mid$,则输出频繁项集. 项集输出后将 e 从 TLG^* 中脱离,并以该条事务链的所有脱离项集来创建一个临时的事务链表组 TLG′,同时更新频率. 一条事务链表扫描完毕后,把 TLG′合并到 TLG^*,在合并过程中更新相应项集的频率. 合并完后回收 TLG′所占内存,最后得到新的事务链表组 TLG^*. 依次循环,直到挖掘完最后一个事务链表后结束,此时已经输出了所有挖掘出的频繁项集,最后回收副本 TLG^* 的内存.

基于 TLG Ⅱ的 V-Stream 算法描述与上雷同,仅需要在输入和输出时作相应的素数映射与解码操作,这里不再赘述.

4　实验结果与分析

4.1　实验环境与结果

实验使用一台 Pentium4 1.8 GHz/768 M 微机(操作系统为 Redhat Linux 9.0) 作为服务器端,一台 Celeron 2.66 GHz/256M 微机(操作系统为 Windows XP) 作为客户端,实验数据来源于 IBM 数据产生器(http://www.almaden.ibm.com/software/quest/Resources/index.shtml). 按照平均事务长度,平均频繁集长度和事务数这 3 个参数分别生成多组数据集. 取平均事务长度为 4,7,10,平均频繁集长度为 2,4,6. 误差阈值始终取值为最小支持度的 0.1.

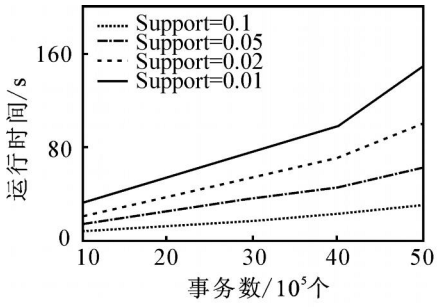
实验过程中,每个窗口的事务数有较大的差别. Manku 采用的是等宽度的时间窗口,而 V-Stream 算法采用变尺度滑动窗口,根据事务数的多少设置窗口大小. 实验中算法程序的输入就是按照不同窗口大小设置不同数量的事务数. 本算法的实现语言是 JAVA5.0,开发工具为 Eclipse.

对 V-Stream 算法随事务数变化和最小支持度变化在挖掘时间和内存消耗上进行了实验. 另外,本文还实现了 Manku 算法用来进行性能对比实验. 图 4(a) 给出了基于 TLG 的 V-Stream 算法当事务数在 10～50 万之间变化时,不同支持度下的算法运行时间的比较. 图 4(b) 给出了 V-Stream 算法与缓冲内存分别是 30M 和 40M 的 Manku 算法之间的运行时间比较. 图 4(c) 给出了基于 TLG 和 TLG Ⅱ的 V-Steam 算法随支持度变化的空间效率变化. 图 4(d) 给出了当最小支持度分别为 0.01 和 0.1 时,V-Stream 算法随事务数变化的内存消耗.

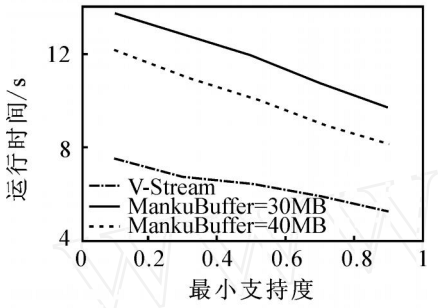
4.2　实验结果分析

从图 4(a) 中可以看出 V-Stream 算法与最小支持度和挖掘事务数之间的关系. 挖掘相同的事务数,最小支持度越低,运行时间越长. 这是因为最小支持度越低,挖掘相同事务数会产生越多的频繁集. 对于同一支持度,越多的事务数则含有越多的频繁项集,运行时间越长.

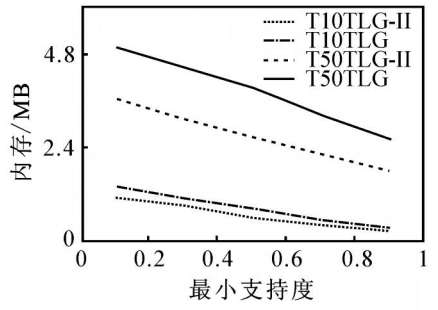
从图 4(b) 中可以看出,与 Manku 算法相比,V-Stream 算法明显提高了挖掘频繁集的时间效率. 这是因为 V-Stream 算法采用变尺度滑动窗口,根据



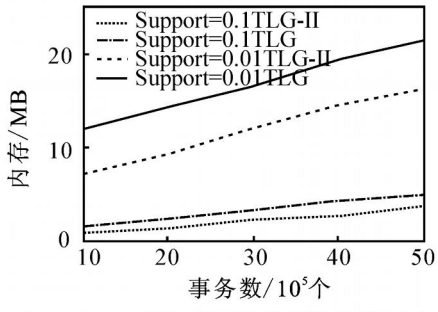
(a) 基于TLG的V-Stream的时间效率



(b) V-Stream与Manku的时间效率比较



(c) V-Stream算法随支持度变化的空间效率



(d) V-Stream算法随事务数变化的空间效率

图 4 实验结果的比较

事务数的多少设置窗口大小实验中算法程序的输入就是按照不同窗口大小设置不同数量的事务数,从而表现出较高的运行效率. TLG 的数据结构具有很高的空间优越性,它不需要缓冲存储区的辅助空间. 如实验中 Manku 算法需要 30M 和更高的缓存,而 V-Stream 仅需要几兆的内存消耗. 因此,与 Manku 算法需要较多的缓冲内存相比,V-Stream 算法的事务链表组的数据结构能有效降低内存的消耗.

在实验中还设计了两种事务链表组的数据结构,TLG 与 TLG Ⅱ. 在第 2 节里分析了 TLG Ⅱ的改进之处,从图 4(c),4(d) 的实验结果中可进一步看出 TLG Ⅱ在空间性能上的提高. 图 4(c) 给出了分别挖掘 10 万个和 50 万个事务,基于 TLG 与 TLG Ⅱ的 V-Stream 算法随最小支持度变化的内存消耗变化. 图 4(d) 给出了分别设置最小支持度为 0.01 和 0.1 时,基于 TLG 与 TLG Ⅱ的 V-Stream 算法随挖掘事务数变化的内存消耗变化. 实验结果表明,在挖掘大量的事务数时或者在最小支持度较小时,TLG Ⅱ相比 TLG 在内存开销上的优势更加明显.

5 结 论

目前大部分基于滑动窗口机制的数据流挖掘算法都假设滑动窗口的大小不变,然后进行增量式分析. 这些算法在人工数据流上具有较好的效率,但在真实数据流上往往效率较低. 本文同时考虑数据流的海量特性和时变特性,提出了一种基于变尺度滑动窗口的数据流频繁集挖掘算法 V-Stream,能在多个尺度的时间窗口上挖掘频繁项集. 算法采用近似挖掘技术,一次扫描数据快速得到频繁项集. 另外,还提出两种基于事务链表组的概要数据结构,改善了在挖掘海量数据时的内存开销. 本文在 Eclipse 上开发了基于公共 IBM 合成数据集的模拟数据流产生器,并进行了 V-Stream 算法的性能实验以及与 Manku 算法的对比实验. 实验结果表明了 V-Stream 算法在性能上的优势. 本文算法尤其适用于流速显著变化的数据流频繁集挖掘应用领域.

进一步需要研究的问题包括:1) 如何在时变环境中对数据流进行特征抽取,根据数据流的特征变化来自动调整窗口的大小;2) 在有限资源下怎样进行数据流挖掘,如数据流流速超过系统的处理能力或者 CPU,内存,带宽以及能量等资源有限情况下怎样有效进行数据分析.

参考文献(References)

[1] Agrawal R, Imielinski T, Swami A. Mining association rules between sets of items in large databases[C]. Proc of the 1993 ACM SIGMOD Int Conf on Management of Data. Washington D C, 1993: 207-216.

[2] Han J, Pei J, Yin Y. Mining frequent patterns without candidate generation[C]. Int Conf on Management of Data. Dallas, 2000: 432-444.

[3] Manku G S, Motwani R. Approximate frequency counts over data streams[C]. 28th Int Conf on Very Large Data Bases. Hong Kong, 2002: 356-357.

[4] Chang J H, Lee W S. A sliding window method for finding recently frequent itemsets over online data streams[J]. J of Information Science and Engineering, 2004, 20(4): 753-762.

(下转第 842 页)

6 结 论

当时延不超过一个采样周期、数据包丢失率一定时,正则无脉冲奇异被控对象的网络控制系统的闭环模型是由数据包丢失率约束的异步动态切换系统,且状态反馈和动态输出反馈闭环 NCS 可用统一模型描述.系统指数稳定的条件可以用一组不等式描述,在给定其中一组参数的前提下,可用 LMI 工具箱求解另一组矩阵不等式,进而可判定系统的稳定性.数据例子给出了稳定性的判定方法.

参考文献(References)

[1] Wei Zhang, Michael S Branicky, Stephen M Phillips. Stability of networked control systems [J]. IEEE Control Systems Magazine, 2001, 21(1): 84-99.

[2] Yue Dong, Han QingLong, James Lam. Network-based robust $H\infty$ control of systems with uncertainty [J]. Automatica, 2005, 41: 999-1007.

[3] Fang Huajing, Ye Hao, Zhong Maiying. Fault diagnosis networked control systems [J]. Annual Reviews in Control, 2007, 31: 55-68.

[4] Huaguang Zhang, Dedong Yang, Tianyou Chai. Guaranteed cost networked control for T-S fuzzy systems with time delays[J]. IEEE Trans on Systems, Man, and Cybernetics: Part C, 2007, 37(2): 160-172.

[5] Gao Huijun, Chen Tongwen. $H\infty$ estimation for uncertain systems with limited communication capacity [J]. IEEE Trans on Automatic Control, 2007, 52(1): 2070-2084.

[6] Dai L. Singular control systems[M]. Berlin: Spring-Verlag, 1989: 1-100.

[7] 邱占芝, 张庆灵, 杨春雨. 基于广义系统的网络控制系统的分析与建模[J]. 东北大学学报, 2005, 26(5): 409-412.
(Qiu Z Z, Zhang Q L, Yang C Y. Modeling and analysis for networked control systems based singular systems[J]. J of Northeastern Unerveisity, 2005, 26(5): 409-412.)

[8] Qiu Zhanzhi, Zhang Qingling, Zhao Zhiwu. Stability of singular networked control systems with control constraint [J]. J of Systems Engineering and Electronics, 2007, 18(2): 290-296.

[9] Rabello A, Bhaya A. Stability of asynchronous dynamical systems with rate constraints and applications [J]. IEE Proc of Control Theory Application, 2003, 150(5): 546-550.

[10] 朱其新, 胡寿松. 网络控制系统的分析与建模[J]. 信息与控制. 2003, 21(1): 5-8.
(Zhu Q X, Hu S S. Modeling and analysis of networked control systems [J]. Information and Control, 2003, 21(1): 5-8.)

(上接第 836 页)

[5] Zhu X D, Huang Z Q. Conceptual modeling rules extracting for data streams [J]. Knowledge-based System, 2008, 21(8): 934-940.

[6] Chang J H, Lee W S. Zhou A. Finding recent frequent itemsets adaptively over online data streams[C]. ACM SIGKDD Int Conf on Knowledge Discovery and Data Mining. San Diego, 2003: 487-492.

[7] Jin R M, Agrawal G. Frequent pattern mining in data streams[M]. New York: Springer, 2007: 61-84.

[8] Lin C-H, Chiu D-Y, Wu Y-H, et al. Mining frequent itemsets from data streams with a time-sensitive sliding window[C]. SIAM Int Conf on Data Mining. Newport Beach, 2005: 59-66.

[9] Giannella C, Han J, Pei J, et al. Mining frequent patterns in data streams at multiple time granularities [M]. Cambridge: AAAI/MIT Press, 2003: 191-212.

[10] Borgelt C. Finding frequent itemsets by recursive elimination[C]. Workshop Open Source Data Mining Software. Chicago, 2005: 66-70.

[11] Sivanandam S N, Sumathi D, Hamsapriya T, et al. A hybrid parallel frequent itemset mining algorithm for very large databases [J]. Academic Open Internet Journal, http://www.acadjournal.com, 2004, 13.