一种基于时间衰减模型的数据流闭合模式挖掘方法

韩 萌[1),2)] 王志海[1)] 原继东[1)]

[1)](北京交通大学计算机与信息工程学院 北京 100044)

[2)](北方民族大学计算机科学与工程学院 银川 750021)

摘 要 数据流是随着时间顺序快速变化的和连续的,对其进行频繁模式挖掘时会出现概念漂移现象.在一些数据流应用中,通常认为最新的数据具有最大的价值.数据流挖掘会产生大量无用的模式,为了减少无用模式且保证无损压缩,需要挖掘闭合模式.因此,提出了一种基于时间衰减模型和闭合算子的数据流闭合模式挖掘方式TDMCS(Time-Decay-Model-based Closed frequent pattern mining on data Stream).该算法采用时间衰减模型来区分滑动窗口内的历史和新近事务权重,使用闭合算子提高闭合模式挖掘的效率,设计使用最小支持度-最大误差率-衰减因子的三层架构避免概念漂移,设计一种均值衰减因子平衡算法的高查全率和高查准率.实验分析表明该算法适用于挖掘高密度、长模式的数据流;且具有较高的效率,在不同大小的滑动窗口条件下性能表现是稳态的,同时也优于其他同类算法.

关键词 事务数据流;数据流挖掘;频繁模式挖掘;闭合模式挖掘;时间衰减模型;概念漂移

中图法分类号 TP311 **DOI 号** 10.11897/SP.J.1016.2015.01473

Efficient Method for Mining Closed Frequent Patterns from Data Streams Based on Time Decay Model

HAN Meng[1),2)] WANG Zhi-Hai[1)] YUAN Ji-Dong[1)]

[1)](*School of Computer and Information Technology, Beijing Jiaotong University, Beijing* 100044)

[2)](*School of Computer Science and Engineering, Beifang University of Nationalities, Yinchuan* 750021)

Abstract A data stream is a continuous, unbounded and time changed sequence of data elements. Therefore, there will be a concept drift of mining frequent patterns on data stream. In some data stream applications, the information embedded in recent transactions is of particular value. Mining frequent pattern on data stream always generates useless and redundant patterns. In order to obtain the result set of lossless compression, generating closed pattern is needed. This paper proposed a method TDMCS for mining closed frequent patterns on data stream efficiently based on time decay model and closure operator. In order to distinguish between recent and historical transactions in the sliding window, time decay model is used in algorithm TDMCS. The frame of minimum support-maximal error rate-decay factor is designed to avoid concept drift. An average decay factor is designed to balance high Recall with high Precision. The performance of proposed method is evaluated via experiments, and the results show that the proposed method is efficient and steady-state, it applies to high density and long patterns data streams, it is suitable for different sizes of sliding windows, and it is also superior to other analogous algorithms.

Keywords transactional data stream; data stream mining; frequent pattern mining; closed pattern mining; time decay model; concept drift

---

收稿日期:2013-09-04;最终修改稿收到日期:2015-01-11.本课题得到国家自然科学基金(71061001)、北京市自然科学基金(4142042)和国家民委科研项目(14BFZ008)资助.**韩 萌**,女,1982 年生,博士研究生,副教授,主要研究方向为数据挖掘与机器学习.E-mail:compute2006_2@126.com;11112081@bjtu.edu.cn.**王志海**,男,1963 年生,博士,教授,博士生导师,主要研究领域为数据挖掘与机器学习.**原继东**,男,1989 年生,博士研究生,主要研究领域为数据挖掘与机器学习.

1 引 言

数据流(Data Streams)作为一种新型数据模型广泛出现在多种应用领域.与传统的数据集不同,数据流是按时间顺序的、快速变化的、海量的和潜在无限的.频繁模式的挖掘是数据挖掘的热点问题,在传统数据库中挖掘频繁模式这个问题已经被广泛地研究和应用.然而,在流数据的环境下挖掘频繁模式给研究者带来更大的机遇和挑战.

近年来,针对数据流的频繁模式挖掘算法被陆续提出.算法 Sticky Sampling[1]、Lossy Counting[1]和 FDPM[2]挖掘数据流中满足误差界和最小支持度的频繁模式,这是一种近似结果.这类算法没有区分新旧数据,没有考虑最新数据的重要性.MSW[3]和 SWP-Tree[4]等算法采用了时间衰减模型设置新旧事务的权重,强调新事务的重要性,可以得到更加合理的结果集.但是这些算法用于挖掘完整的模式结果集,会产生大量的无用的模式.为了减少模式的数量,需要挖掘精简模式,包括最大模式、闭合模式和 top-k 模式等.为了进行无损压缩,通常的方式是挖掘闭合模式.如算法 Moment[5]、CloStream[6]、Stream_FCI[7]、TMoment[8]、IncMine[9]和 CloStream*[10]等采用滑动窗口挖掘数据流的闭合频繁模式.这类算法的不足在于:(1)采用最小支持度阈值进行频繁模式挖掘,未处理概念漂移问题;(2)虽然采用了滑动窗口,但是窗口内的数据赋予相同的重要性.

为了解决概念漂移,强调新旧事务不同的重要性,更加高效的发现压缩频繁模式,本文设计一种基于时间衰减模型的数据流闭合模式挖掘方法.主要的工作与创新点在于:(1)已有设置衰减因子 f 的方法包括:仅考虑 100%查全率(Recall)或 100%查准率(Precision)[2-4]时得到的上下边界值或在 f 的取值范围(0,1)内设置随机值[11-16].前者会使得对应算法的查准率或查全率较低,而随机值则使得算法的性能不稳定.本文设计一种平均衰减因子取值方式,目的是使设计的算法可以在高查全率和高查准率之间得到平衡,且得到的算法性能是稳定的;(2)算法中采用最小支持度-最大误差率-衰减因子的框架,可以解决概念漂移,避免丢失可能的频繁模式;(3)使用闭合算子[6,10]提高算法执行的效率;(4)设计了基于时间衰减模型的闭合频繁模式挖掘方法,可以得到压缩的无损的模式集合.使用时间衰减模型[3-4,11-12,17-26]增加了最新事物的权重,同时降低了历史事务的重要性.通过对数据流处理得到的查准率进行比较得出,与已有方式相比能得到更准确的结果集.

本文第 2 节介绍数据流闭合模式挖掘的预备知识,包括闭合算子和时间衰减模型的介绍;第 3 节详细介绍基于时间衰减模型的闭合模式挖掘算法;第 4 节通过实验验证衰减因子选择的合理性,并且给出本文提出算法的实验性能分析;第 5 节进行总结.

2 预备知识

本节主要介绍数据流闭合模式挖掘的相关知识,包括闭合算子和闭合频繁模式的定义,时间衰减模型介绍,并提出概念漂移问题的解决方式.

数据流 $DS=\langle T_1,T_2,\cdots,T_n,\cdots\rangle$ 是一个有时间顺序的、连续的、无限的事务(Transaction)序列,其中 $T_n(n=1,2,\cdots)$ 是第 n 个产生的事务,如表 1 所示.由于数据流是无限的和不断流动的,模式 P 的支持数定义为已出现的 N 个事务中包含模式 P 的事务数据个数,记为 $freq(P,N)$[3].频繁模式、临界频繁模式和非频繁模式的定义如定义 1、定义 2 所示.

表 1 事务数据库

TID	Transaction
T_1	1 3 4
T_2	2 3 5
T_3	1 2 3 5
T_4	2 3 4 5

定义 1. 频繁模式[3].令 N 为数据流中已有的事务项的个数,$\theta(\theta\in(0,1])$ 为最小支持度阈值.如果项集 P 满足 $freq(P,N)\geq\theta\times N$,则 P 为频繁模式.

定义 2. 临界频繁模式,非频繁模式[3].令 N 为数据流中已有的事务项的个数,$\theta(\theta\in(0,1])$ 为最小支持度阈值,ε 为最大允许误差阈值($\varepsilon\in(0,\theta)$).如果项集 P 满足 $\theta\times N\geq freq(P,N)\geq\varepsilon\times N$,则 P 为临界频繁模式.如果 $freq(P,N)<\varepsilon\times N$,则 P 为非频繁模式.

概念漂移是指由于数据流实时变化,之前的非频繁模式随着时间的推移可能变为频繁模式.为了减少丢失可能的模式个数.挖掘过程中不仅要维持频繁模式,还需要维持临界频繁模式.此外,为了减少维护模式的代价,需要丢失非频繁模式.丢失这些模式的可能误差不大于 ε[3-4],因此挖掘过程中采用

θ-ε 框架可以解决概念漂移问题.

频繁模式挖掘的过程中存在的最大问题在于出现大量的无用的模式，为了减少模式的数量，通常挖掘压缩模式. 闭合模式是一种常用的模式无损压缩方式，它包含完整结果集的所有信息. 为了提高发现闭合模式的速度，本文采用了闭合算子[6,10]. 采用闭合算子发现闭合模式的效率优于 CFI-Stream[27]、NewMoment[28]和 Moment[5]等经典数据流闭合模式挖掘方式[10]. 闭合算子以及采用闭合算子的闭合模式概念为定义 3～5 所示.

定义 3.　闭合算子[6,10]. 设 T 是事务集合 D 的子集，$T\subseteq D$. Y 是 D 中出现的所有项集合 I 的子集，$Y\subseteq I$. 定义函数 h 和 g：

$$h(T)=\{i\in I|\forall t\in T,i\in t\} \tag{1}$$

$$g(Y)=\{t\in D|\forall i\in D,i\in t\} \tag{2}$$

其中函数 h 的输入参数是事务集合 T，输出是 T 中所有事务都包含的一个项集. 函数 g 的输入是项集 Y，输出是包含 Y 的事务集. 函数 $C=h\circ g=h(g)$ 称为闭合算子.

定义 4.　闭合项集[10]. 如果项集 P 满足式(3)，则 P 是闭合项集；否则为非闭合模式. 其中 $C(P)$ 称为 P 的闭合.

$$C(P)=h\circ g(P)=h(g(P))=P \tag{3}$$

定义 5.　闭合频繁模式. 如果项集 P 满足 $P=C(P)$ 且其支持度不小于最小支持度，则 P 为闭合频繁模式. 如果 P 满足 $P=C(P)$ 且其支持度不小于最大误差支持度，则 P 为临界闭合频繁模式；否则 P 为非闭合频繁模式.

由于数据流的连续无限性，其中包含的知识会随着时间的推移而发生改变. 通常情况下，最近产生的事务的价值比历史事务要高的多. 因此，需要增加最近事务的权重. 时间衰减模型 TDM(Time Decay Model)是一种随着时间的推移而逐步衰减历史事务模式支持数权重的方法[3-4]. 设模式支持数在单位时间内的衰减比例为衰减因子 $f(f\in(0,1))$，记事务 T_n 到达时模式 P 的衰减支持数为 $freq_d(P,T_n)$. 则第 m 个事务 T_m 到达时，模式 P 的衰减支持数满足式(4)和(5). 即随着新事务的到达，每次模式的支持数都进行衰减. 其中如果新事务 T_m 中包含模式 P，则 r 的值为 1；否则为 0.

$$freq_d(P,T_m)=\begin{cases}r, & m=1\\ freq_d(P,T_{m-1})\times f+r, & m\geq 2\end{cases} \tag{4}$$

$$r=\begin{cases}1, & P\subseteq T_m\\ 0, & \text{其他}\end{cases} \tag{5}$$

3　TDMCS 算法描述

本节首先介绍配合闭合算子使用的数据结构，其次介绍了衰减因子的确定方法，最后详细讨论基于 θ-ε-f 三层框架的闭合频繁模式挖掘算法 TDMCS (TDM-based Closed frequent pattern mining on data Stream).

TDMCS 算法采用滑动窗口模型与时间衰减模型在数据流中挖掘闭合频繁模式，主要包括处理新事物和旧事物的两个方法：$TDMCSADD(T_{new})$ 和 $TDMCSREMOVE(T_{old})$. TDMCS 使用了与算法 CloStream[6,10]相似的 3 个数据结构，包括 *ClosedTable*[6]、*CidList*[6] 和 *NewTransactionTable*. 其中 *ClosedTable* 用于存储闭合模式相关的信息，包括 3 个字段：*Cid*、*CP* 和 *SCP*. *Cid* 用于唯一的标识每一个闭合模式 *CP*，*SCP* 是闭合模式 *CP* 对应的支持数. *CidList* 用于维护数据流中出现的每个项 *item* 和其对应的 *cid* 集合. *NewTransactionTable* 包含与新事务 T_{new} 相关的信息，包括两个字段：*TempItem* 和 *Cid*. 其中 *TempItem* 存储满足条件 $\{T_i\bigcap NewTransaction, T_i\in ClosdeTable\}$ 的项集信息.

假设衰减因子 $f=0.8$，最小支持度阈值 $\theta=0.1$，以表 1 中事务数据为例介绍 TDMCS 算法处理新事务的工作过程. 当新事务 $T_4=\{2,3,4,5\}$ 到达时，*ClosedTable*、*CidList* 和 *NewTransactionTable* 中的信息如表 2～表 4 所示. 处理事务 T_1，T_2 和 T_3 后，*ClosedTable* 中有 4 个闭合模式，如表 2 左侧所示. 当处理事务 T_4 后，*ClosedTable* 中包含 6 个闭合模式如表 2 右侧所示. 新的频繁项集如果满足最小支持度阈值和最大允许误差阈值，则需要的工作包括：

(1) 添加频繁项为新的模式，加入 *ClosedTable*. 同步更新 *CidList*.

(2) 更新已有模式.

① 如果依然是闭合模式，则更新其支持数.

② 新事务的到来使之成为非闭合模式，则删除. 同步更新 *CidList*.

具体的过程是，事务 T_4 到达时，将 T_4 存入 *NewTransactionTable*，然后比较 *CidList* 和 T_4 中的每个项 *item*，更新 *NewTransactionTable* 如表 4 所示. 然后参照 *NewTransactionTable*，在 *ClosedTable* 中添加新的模式或者更新已有的模式. 同时更新 *CidList*. 如此反复，随着数据的到达不断的更新.

表 2 *ClosedTable*($\theta=0.1$)

Before T_4			After T_4		
Cid	*CP*	*SCP*	*Cid*	*CP*	*SCP*
0	{0}	0	0	{0}	0
1	{1 3 4}	0.64	1	{1 3 4}	0.512
2	{2 3 5}	1.8	2	{2 3 5}	2.44
3	{3}	2.44	3	{3}	2.952
4	{1 2 3 5}	1	4	{1 2 3 5}	0.8
			5	{2 3 4 5}	1
			6	{3 4}	1.512

表 3 *CidList*

item	*CidSet*
{1}	{1,4}
{2}	{2,4}
{3}	{1,2,3,4}
{4}	{1}
{5}	{2,4}

表 4 *NewTransactionTable*

TempItem	*Cid*
{2 3 4 5}	0
{3 4}	1
{2 3 5}	2
{3}	3

上述示例中设置最小支持度阈值 $\theta=0.1$,挖掘过程中发现的所有模式都被保留了下来.这样可能会产生大量无用的模式.如果设置最小支持度阈值 $\theta=0.3$,当事务 T_4 到达时需要满足的最小支持数为 $4\times0.3=1.2$.产生的 *ClosedTable* 如表 5 所示.通过对比表 2 和表 5 可以发现满足最小支持数的频繁模式{3 4}丢失.

表 5 *ClosedTable*($\theta=0.3$)

Cid	*CP*	*SCP*
0	{0}	0
1	{2 3 5}	2.44
2	{3}	2.952

从示例中可以分析出,使用了衰减因子后模式的衰减支持数远小于非衰减时的支持数.如当 $f=0.8$ 时,采用式(6)可以得到模式的支持数小于:$1/(1-f)=1/(1-0.8)=5$.因此,按照常规的最小支持数进行挖掘会丢失可能的频繁模式.为了解决这个问题,使得采用时间衰减模型后正确率等于常规模式挖掘,需要设置最大允许误差阈值 ε.即挖掘过程中保存频繁模式和临界频繁模式,而不仅仅保存频繁模式.

$$\begin{aligned} freq_d(P,T_m)&=freq_d(P,T_{m-1})\times f+r\\ &=\sum_i r_i\times f^{m-i}\\ &=r_1\times f^{m-1}+r_2\times f^{m-2}+\cdots+r_m\\ &\le f^{m-1}+f^{m-2}+\cdots+r_m\le\frac{1}{1-f} \end{aligned} \tag{6}$$

给定最小支持阈值和最大允许误差阈值之后,如何确定衰减因子 f 的值?已有的方式采用假定 100% 的查全率和 100% 的查准率来估计[3-4].即设定 Recall 为 100%时,f 应满足式(7),称为下界值.设定 Precision=100%时,f 满足式(8),称为上界值.现有的 f 估计方式是采用满足式(7)和(8)的上下边界值之一.这种方式的最大不足是仅考虑了查全率或查准率,而忽略了对应的查准率或查全率.由于 Recall 和 Precision 不可能同时为 100%,所以选择 f 是应考虑对二者的平衡.为此本文提出了均值衰减因子设置方式,即设置 f 为上下边界的平均值,标记为 f_{average}.

$$f\ge\sqrt[(2N-\theta N-1)]{[(\theta-\varepsilon)/\theta]^2},\ \text{when Recall}=100\% \tag{7}$$

$$f<\frac{(\theta-\varepsilon)N-1}{(\theta-\varepsilon)N},\ \text{when Precision}=100\% \tag{8}$$

例如,假设 $N=10\,\text{K}$,设定 θ,ε 如表 6 所示.其中 f_{recall} 为假定 Recall=100% 时得到的下界值.$f_{\text{precision}}$ 为假定 Precision=100%时得到的上界值.在得到 f_{recall} 和 $f_{\text{precision}}$ 后,如何选择 f 的值?可以有 3 个策略,如式(9)所示.以 $\theta=0.025$,$\varepsilon=0.05\times\theta$ 为例,可以选定

$$\begin{aligned} &f=f_1=f_{\text{recall}}=0.999995,\\ &f=f_2=f_{\text{precision}}=0.995789\ \text{或}\\ &f=f_3=f_{\text{average}}=(0.999995+0.995789)/2\\ &\qquad=0.997892. \end{aligned}$$

表 6 设置最佳的衰减因子

θ	ε	f_{recall}	$f_{\text{precision}}$	f_{average}
0.05	$0.05\times\theta$	0.999995	0.997895	0.998945
0.05	$0.1\times\theta$	0.999989	0.997778	0.998884
0.05	$0.5\times\theta$	0.999929	0.996	0.997965
0.025	$0.05\times\theta$	0.999995	0.995789	0.997892
0.025	$0.1\times\theta$	0.999989	0.995556	0.997773
0.025	$0.5\times\theta$	0.99993	0.992	0.995965

$$\begin{aligned} &f_1=f_{\text{recall}},\\ &f_2=f_{\text{precision}},\\ &f_3=f_{\text{average}}=(f_{\text{recall}}+f_{\text{precision}})/2 \end{aligned} \tag{9}$$

通过实验验证(第 4 节中)选择设置 f 为 f_{average} 不仅可以得到更加合理的模式结果集的个数,且得到结果集的查全率与查准率更平衡.因此本算法中设计衰减因子 f 的值为 f_{recall} 和 $f_{\text{precision}}$ 的平均值 f_{average} 是合理的.

从滑动窗口中移除旧事务的核心问题在于如何对已有的数据结构进行剪枝处理,已有的方式常采用滑动一步剪枝一步,这样消耗较大.为了增加算法

处理的效率，采用移动步长 $STEPM$ 进行处理. 即处理旧事务时先设置删除标记 $removeTAG$，当窗口滑动 M 条事务后再进行实际的剪枝操作.

为了区分历史事务与新事务的权重，从而提高模式发现的准确性，并且为了避免丢失可能的频繁模式，本文提出了基于 θ-ε-f 框架的闭合模式挖掘算法 TDMCS. 该算法采用 $ClosedTable$，$CidList$ 和 $NewTransactionTable/OldTransactionTable$ 存储数据信息，使用时间衰减模型估计模式的支持数，挖掘出满足 θ-ε 的闭合频繁和临界闭合频繁模式. 算法的描述如算法 1 所示.

算法 1.　$TDMCS()$.

输入：数据流 S，滑动窗口大小 N，剪枝步长 $STEPM$，衰减因子 f，最小支持度 θ，最大允许误差率 ε

输出：频繁闭合模式

Algorithm $TDMCS(S,N,STEPM,f,\theta,\varepsilon)$

```
FOR T_new IN S DO
  CALL TDMCSADD(T_new)
  IF SLIDING THEN CALL TDMCSREMOVE(T_old)
END FOR
IF NEEDED THEN OUTPUT PATTERNS
ENDIF
END TDMCS
```

过程 1.　处理新事务 $TDMCSADD()$

输入：新事务 T_{new}，闭合项集表 $ClosedTable$，滑动窗口大小 N，衰减因子 f，最小支持度 θ，最大允许误差率 ε

输出：闭合项集表 $ClosedTable$

Algorithm $TDMCSADD(T_{new})$

```
1. ADD T_new TO NewTransactionTable
   setcid(T_new)={⋃CidSet(item_i), item_i∈T_new}
2. FOR cid IN setcid(T_new) DO
     interS=T_new⋂ClosedTable(cid)
     FOR TempItem IN NewTransactionTable DO
       IF interS∈ClosedTable THEN
         support(interS)=support(interS)×f+r
       ELSE ADD 〈interS,cid〉 TO ClosedTable
       ENDIF
     END FOR
   END FOR
3. FOR 〈TempItem,cid〉 IN NewTransactionTable DO
     IF (TempItem==ClosedTable(cid))
     THEN newsupport(ClosedTable(cid))=
                  support(ClosedTable(cid))×f+r
     ELSE newsupport(TempItem)=
                  support(ClosedTable(cid))×f+r
       IF (newsupport(TempItem)>=ε×N)
       THEN ADD(TempItem,newsupport(TempItem))
            TO ClosedTable
       ENDIF
     ENDIF
   END FOR
4. IF item∈T_new AND item IS NOT IN CidList
   THEN ADD item TO CidList
   ENDIF
END TDMCSADD
```

过程 2.　处理历史事务 $TDMCSREMOVE()$.

输入：历史事务 T_{old}，闭合项集表 $ClosedTable$，滑动窗口大小 N，剪枝步长 $STEPM$，最小支持度 θ，最大允许误差率 ε

输出：闭合项集表 $ClosedTable$

Algorithm $TDMCSREMOVE(T_{old})$

```
1. ADD T_old TO OldTransactionTable
   setcid(T_old)={⋃CidSet(item_i), item_i∈T_old}
2. FOR cid IN setcid(T_old) DO
     interS=T_old⋂ClosedTable(cid)
     FOR TempItem IN OldTransactionTable DO
       IF inters∈ClosedTable AND support(interS)<
         ε×N
       THEN ADD removeTAG ON interS
       ENDIF
     END FOR
3. IF WINDOW MOVE STEPM
   THEN PRUNNING
     REMOVE items(WITH removeTAG) FROM
            closedTable
     UPDATE cidlistMap
   ENDIF
END TDMCSREMOVE
```

4　实验分析

实验运行环境的 CPU 为 2.1GHz，内存为 2GB，操作系统是 Win7，所有的实验采用 Java 实现. 实验中采用两类数据，一是真实数据流 msnbc 来自 UCI[①]，此数据描述的是 1999 年 9 月 28 日访问 msnbc.com 网站的用户信息. 用户访问的页面按照 URL 分类，并按照时间顺序记录. 包括 989 818 个事务序列，平均事务长度 5.7，数据重复项多，为高密度数据流. 二是采用 IBM 模拟数据生成器产生不同平均

① Frank A, Asuncion A. UCI Machine Learning Repository [EB/OL]. Irvine, CA: University of California, School of Information and Computer Science. http://archive.ics.uci.edu/ml, 2010

事务长度和不同平均模式长度的数据.模拟数据流包括 T5I5D1000K、T10I4D1000K、T10I5D1000K、T10I10D1000K、T20I5D1000K 和 T20I20D1000K.这些数据流用于分析不同事务长度和不同模式长度时算法的性能.其中 T10I5D1000K 表示数据的平均事务长度为 10,平均模式长度为 5,数据事务个数为 1000 K.

TDMCS 算法中设置最大误差率 $\varepsilon=0.1\times\theta$,衰减因子 f 为 f_{average}.实验中设置滑动窗口 N 的大小为 0.1 M,0.2M,0.3M,0.4M,0.5M,0.7M 和 0.8M,设置最小支持度 θ 的范围为[0.06,0.1],衰减因子 f 的取值如表 7 中所示.

表 7 衰减因子 f 的值

f_{id}	θ	ε	N/M	f
f_1	0.06	0.006	0.1	0.990686
f_2	0.06	0.006	0.2	0.995343
f_3	0.06	0.006	0.3	0.996895
f_4	0.06	0.006	0.4	0.997672
f_5	0.06	0.006	0.5	0.998137
f_6	0.06	0.006	0.7	0.998669
f_7	0.06	0.006	0.8	0.998836

首先分析设置 f 为平均衰减因子的合理性.实验从两个角度进行比较:一是比较得到结果集的模式个数;二是比较设置不同衰减因子得到的算法查全率和查准率.

表 8 中是对数据流 msnbc 进行处理得到的闭合模式的个数.选择最小支持度 θ 为 0.025 和 0.05,最大误差率 $\varepsilon=0.05\times\theta$.表中第 2 列是衰减因子 f,排列顺序是 f_{recall}、$f_{\text{precision}}$ 和 f_{average}.以 $\theta=0.025$ 为例,可以得到

$$PatternNumber(f_{\text{recall}})=47253>$$
$$PatternNumber(f_{\text{average}})=47223>$$
$$PatternNumber(f_{\text{precision}})=47219.$$

当 $\theta=0.05$ 时也可以得到相同的结论.因此从得到的模式数据量角度而言,算法中选择 f_{average} 作为衰减因子相比上下边界值而言更加合理.

表 8 3 种衰减因子的比较

θ	f	♯*pattern*
0.025	0.999999(f_{recall})	47253
0.025	0.99598($f_{\text{precision}}$)	47219
0.025	0.99799(f_{average})	47223
0.05	0.99993(f_{recall})	37031
0.05	0.98($f_{\text{precision}}$)	36941
0.05	0.998995(f_{average})	37020

从算法的查全率与查准率角度进行分析.以数据流 msnbc 为例,比较 f 设置为 f_{recall},$f_{\text{precision}}$ 和 f_{average} 算法性能的优劣.比较不同窗口大小时算法的平均性能,表现如图 1 所示.从算法的查全率可以看出设置 $f=f_{\text{recall}}$ 可以得到几乎 100% 的 Recall,而 $f=f_{\text{precision}}$ 得到的 Recall 值最低,$f=f_{\text{average}}$ 得到的 Recall 值介于二者之间.接着比较算法的查准率.设置 $f=f_{\text{precision}}$ 和 $f=f_{\text{average}}$ 时得到的查准率几乎相同,而 $f=f_{\text{recall}}$ 得到的值最低.因此,可以得出设置衰减因子为 f_{average} 可以得到比 $f_{\text{precision}}$ 和 f_{recall} 更优的算法性能:即得到的算法的查全率和查准率更加的平衡.

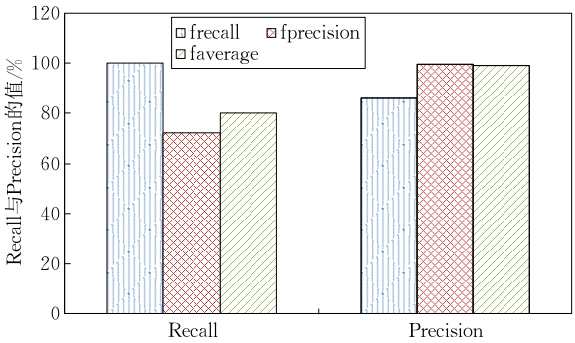


图 1 设置衰减因子为 f_{recall},$f_{\text{precision}}$,f_{average} 得到的算法性能比较(其中纵向坐标为算法得到的 Recall 与 Precision 百分比)

接着比较设置 f 为 f_{average} 与随机值的优劣.为了使设置的随机衰减因子更加合理,取其范围为(0.9,1),标记为 f_{random},采用 Java 中 *Math.random*() 函数生成.随机生成 5 个衰减因子值.设置 f 为 f_{average} 与 f_{random} 得到的算法性能如图 2 所示.从中可以看出,设置 f 为 f_{average} 与 f_{random} 得到的查准率差别不大.而采用 f_{random} 得到的算法查全率差别较大,即得到的结果集的性能不稳定.设置 f 为 f_{average} 的表现明显优于设置为 f_{random},且其得到的结果集是稳定的.

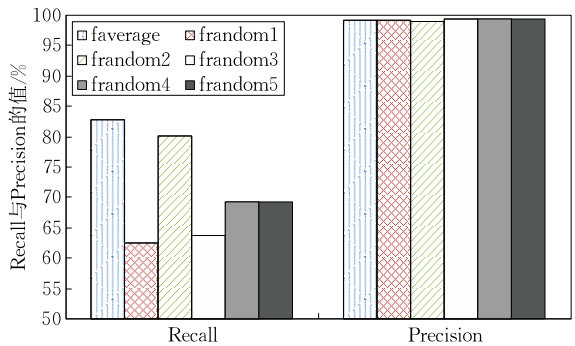
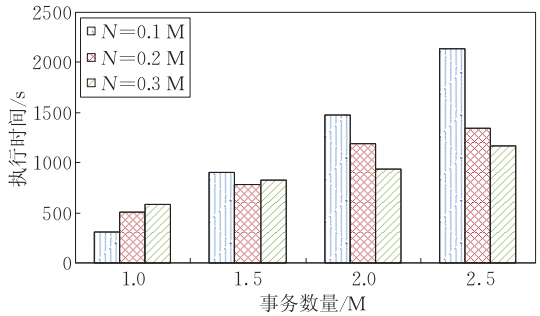


图 2 设置衰减因子为 f_{average} 和随机值得到的算法性能比较(其中纵向坐标为算法得到的 Recall 与 Precision 的百分比)

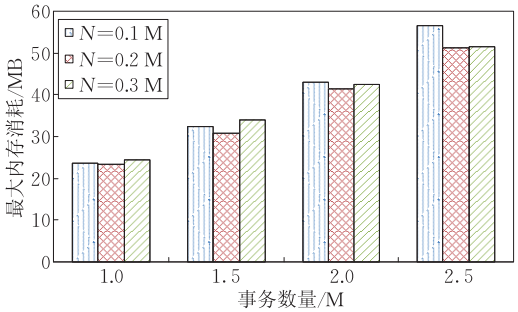
对模拟数据进行处理时也可以得到相似的结论.因此,本文中提出的算法使用时间衰减模型时设置平均衰减因子是合理的.

其次,分析窗口大小对 TDMCS 算法的影响.图 3 和图 4 所示是窗口大小 N 为 0.1 M、0.2M 和

0.3M 时，算法 TDMCS 在真实数据流 msnbc 与模拟数据流上的执行时间和占用最大内存空间的比较. 实验设置衰减因子 f 的值如表 7 中 $f_1 \sim f_3$ 所示. 设置剪枝步长为 0.1 M[3-4]，即数据流每滑动 0.1 M 条事务时进行实际剪枝操作.

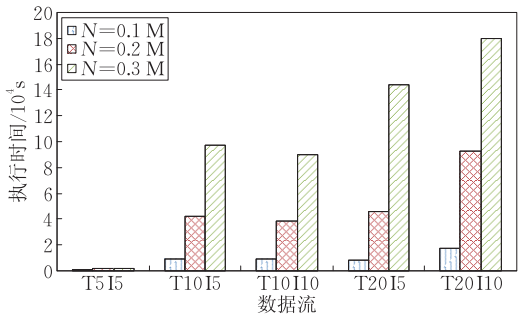


(a) 执行时间

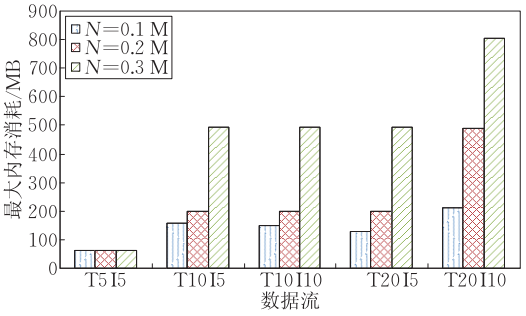


(b) 最大内存消耗

图 3　在不同窗口大小下算法 TDMCS 在 msnbc 上的比较(其中横向坐标为处理的事务数量)



(a) 执行时间



(b) 最大内存消耗

图 4　在不同窗口大小下算法 TDMCS 在不同模拟数据流上的比较(其中横向坐标为不同特征的模拟数据流)

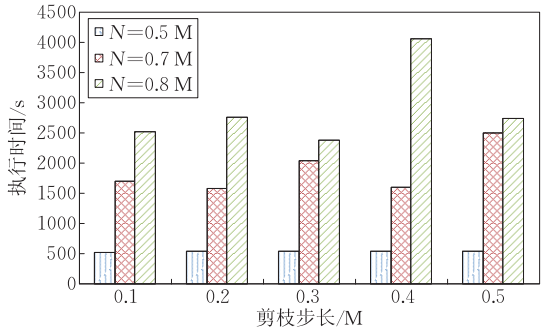
图 3 是采用 TDMCS 算法对数据流 msnbc 处理 1 M，1.5 M，2 M 和 2.5 M 条事务进行分析. 图 3(a)显示的是 TDMCS 算法在数据流 msnbc 上的执行时间. 可以看出处理事务数量较少时，窗口大小的增大会导致执行时间小幅度增加. 但是随着处理事务数量的增加，采用大的滑动窗口时间消耗反而更小. 如当处理 1 M 条事务时，随着 N 的增加，时间消耗增加. 而当处理 2 M 或 2.5 M 条事务时，随着 N 的增加，时间消耗反而减少. 从图 3(a)中可以看出，采用大的滑动窗口处理数据时，随着数据量的增加时间消耗增加的幅度比小的滑动窗口少. 如当 $N=0.3$ M 时，处理 2.5 M 事务与处理 1 M 事务相比，处理的数据量增加了 150%，而时间消耗增加了约 99.9%. 而当 $N=0.1$ M 时，相同条件下，时间消耗增加了 586.5%. 因此，采用不同的滑动窗口大小对数据流进行处理时，时间消耗差别较大.

图 3(b)为算法执行使用的空间大小. 可以看出，不同的窗口大小对使用的内存空间影响很小；随着处理事务数量的增加，内存消耗增加幅度较小. 综合时间和内存消耗可以得出结论：对数据流 msnbc 进行处理时，相同的事务数量下，TDMCS 算法使用的执行时间会随着 N 的不同而不同，而使用的存储空间受窗口大小 N 的影响不大. 因此，从空间复杂度角度而言，该算法适用于挖掘任意大小滑动窗口内的频繁模式.

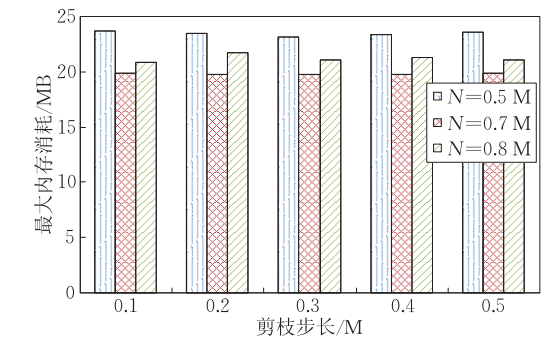
图 4 比较不同滑动窗口大小 N 时，TDMCS 在不同事务长度和不同模式长度的数据流上的性能. 从图 4 中可以看出随着 N 的增大，算法执行时间成倍增加，内存消耗有较小幅度的增加. 首先分析事务长度对算法性能的影响. 通过比较两组数据流：T5I5，T10I5 和 T20I5，T10I10 和 T20I10 可以看出随着平均事务长度的增加，算法的执行时间增加幅度较大，使用内存幅度增加相对较小. 接着分析模式长度对算法性能的影响. 通过比较两组数据流：T10I5 和 T10I10，T20I5 和 T20I10 可以看出随着平均模式长度的增加，算法执行时间和内存消耗增加，但增加的幅度不大. 如当 $N=0.3$ 时，在数据流 T10I10 上的执行时间甚至比在 T10I5 上略微减少，而内存消耗几乎不变. 可以得出结论，TDMCS 算法处理不同的数据流时，受到事务长度的影响较大，而受到模式长度的影响较小. 这表明该算法适用于挖掘长模式的数据流.

第三，分析剪枝步长对 TDMCS 算法性能的影

响.设定滑动窗口大小 N 为 0.5 M,0.7 M 和 0.8 M,剪枝步长 $SETPM$ 为 0.1 M～0.5 M($SETPM \leq N$),设定衰减因子 f 的取值为表 7 中 $f_5 \sim f_7$.图 5(a)为不同滑动窗口条件下采用不同的剪枝步长时算法执行的时间.从图中可以看到,当 $N=0.5$ M 时,剪枝步长对算法的处理时间影响不大.当 $N=0.7$ M 时,设定 $SETPM$ 为 0.2 M 时算法执行过程运行时间最少,而 $SETPM=0.5$ M 时运行时间最多,后者比前者运行时间增加了 57%.当 $N=0.8$ M 时,$SETPM=0.3$ M 时算法执行时间最少,$SETPM=0.4$ M 时运行时间最多,后者比前者运行时间增加了 70.4%.从执行时间的实验结果可以得到结论:(1)最优剪枝步长与滑动窗口大小 N 相关;(2)随着 N 的增大,不同的 $SETPM$ 带来的执行时间差增大.图 5(b)为采用不同剪枝步长和不同滑动窗口大小时算法执行需要的最大存储空间.从图中可以看出最大存储空间消耗受剪枝步长大小的影响不大.从图 5 中可以得出结论,当设定滑动窗口较小时,剪枝步长的大小对算法的执行时间和使用内存影响不大.而当滑动窗口较大时,对执行时间的影响较大,对内存使用影响甚微.因此当设定参数 $N=0.1$ M 时,设定 $SETPM=0.1$ M 是比较合理的.同样的,通过对图 5(b)的分析,进一步证明了 TDMCS 算法适用于挖掘任意大小的滑动窗口内的频繁模式.



(a) 执行时间



(b) 最大内存消耗

图 5 在不同窗口大小下采用不同剪枝步长时,算法 TDMCS 在数据流 msnbc 上的比较(其中横向坐标为剪枝步长)

最后,分析不同窗口大小时,TDMCS 算法与经典算法 CloStream*,MSW 和 SWP 的性能比较.设定衰减因子 f 的取值为表 7 中 $f_1 \sim f_5$ 所示.为了更合理的比较查全率和查准率,对算法 MSW 和 SWP 稍作修改,使其挖掘闭合模式.

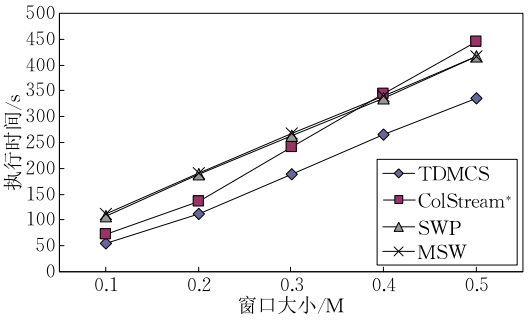
算法在数据流 msnbc 的性能表现如图 6 所示.其中图 6(a)为不同的算法在数据流 msnbc 上的时间消耗.从图中可以看出采用 TDMCS 算法消耗的平均时间与其他三者相比用时最少.

图 6(b)为不同算法在数据流 msnbc 上执行的最大内存消耗.从中可以看出 TDMCS 算法的平均内存消耗大约比 CloStream* 减少了 65%.并且随着 N 的增加,两种方法占用的内存消耗的差距增加.与 MSW 和 SWP 相比也有一定程度的减少.

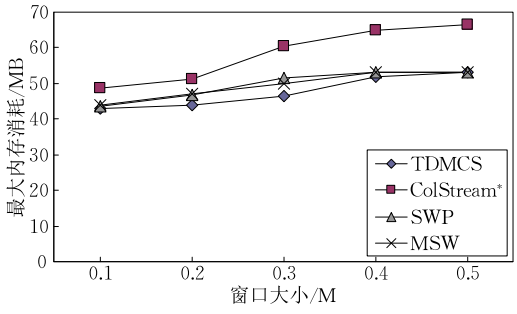
图 6(c)是算法之间的查全率比较.从中可以看出 CloStream* 得到的查全率是最高的,这是由于这个算法没有进行衰减处理.其次是 MSW 和 SWP 得到的查全率较高,因为二者采用的是下界衰减值,即设置条件是假定 100% 的查全率.相比较而言,TDMCS 得到的查全率较低,比 CloStream* 减少了 5%,比 MSW 和 SWP 减少了约 2%,这是由于它的衰减程度是最高的.

图 6(d)是对数据流 msnbc 进行处理时得到的查准率比较.可以看出采用 TDMCS 进行数据流处理时得到的算法的 Precision 明显高于其他三者.与 MSW 和 SWP 相比提高了约 7%,与 CloStream* 相比提高了约 11%.从图(c)和(d)的比较可以得出,采用均值衰减因子设置方式可以得到较为平衡的查全率和查准率.

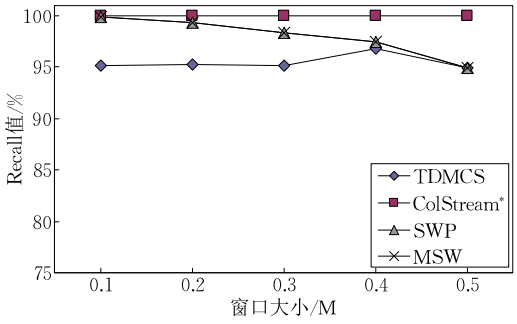
算法在模拟数据流上的比较如图 7 所示,是对不同窗口下的算法性能取平均得到的.使用事务长度和模式长度不等的 4 组数据:T10I4,T10I5,T10I10 和 T20I5.图 7(a)是算法执行时间的比较,整体而言,TDMCS 和其他 3 个算法相比,时间使用较短.CloStream* 由于不进行衰减处理操作,时间消耗也较短.二者的时间消耗低于 MSW 和 SWP.图 7(b)比较算法的内存消耗,总体而言,TDMCS 的内存消耗是最低的,但整体之间的差距不是太大.图 7(c)和图 7(d)比较的是算法的查全率和查准率.由于算法 MSW 和 SWP 使用的是相同衰减因子设置方式,因此仅与 SWP 做性能比较.CloStream* 算法不做衰减处理,其得到的查全率是最高的,但得到的查准率是最差的.SWP 设置衰减因子为边界值,得到的查全率和查准率居中.TDMCS 得到的查全



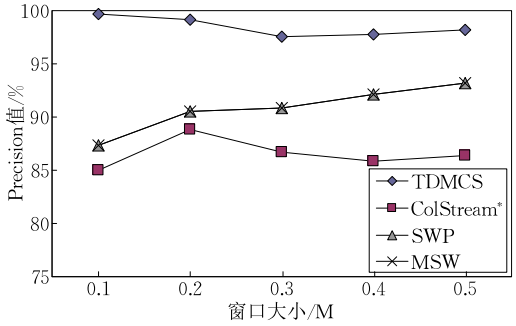
(a) 执行时间



(b) 内存消耗

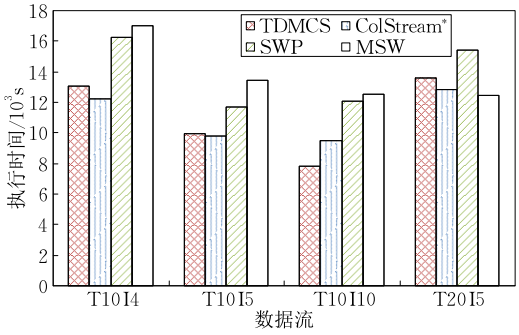


(c) 查全率

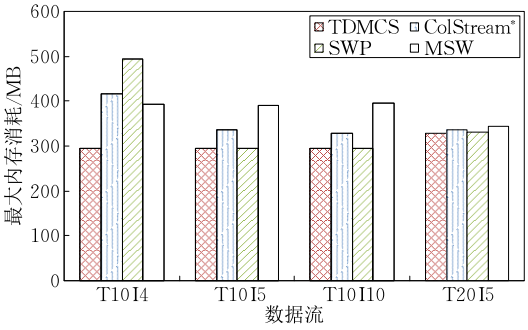


(d) 查准率

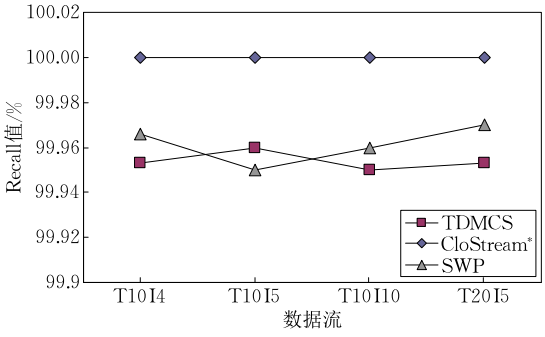
图 6　算法在 msnbc 上的性能比较(其中横向坐标为滑动窗口大小)



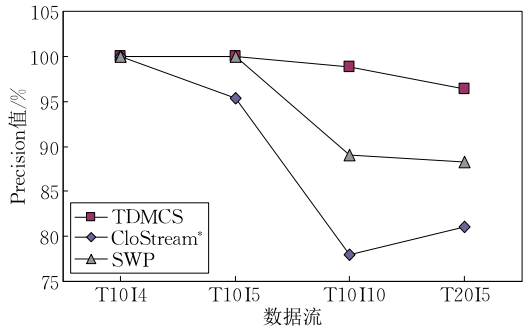
(a) 执行时间



(b) 内存消耗



(c) 查全率



(d) 查准率

图 7　算法在模拟数据流上的平均性能比较(其中横向坐标为不同特征的模拟数据流)

率与其余二者相比减少的不足 1%. 但得到的查准率与 CloStream* 相比增加了约 10%,与算法 SWP 相比增加了约 4%. 因此,从平衡算法的查全率和查准率角度而言,TDMCS 算法的表现更好.

从图 7 的分析可以看出,在不同模式长度的数据流上进行比较,如 T10I4,T10I5 和 T10I10,TDMCS 使用的时间和内存消耗随着模式长度的增加,与其余 3 个算法相比减少较明显,且得到的查准率的优势也很明显. 比较不同事务长度的数据流,如 T10I5 和 I20I5 可以分析出,TDMCS 得到的查准率优势同样较明显. 因此,TDMCS 算法相比而言更适用于长模式和长序列的数据流处理.

5 总 结

数据流不同于常规的数据库,它具有流动性、连续性和无限性.数据流包含的知识会随着时间的发展而发生改变.所以挖掘数据流频繁模式时需要考虑到概念漂移问题.通常情况下,新近的事务比历史事务包含更重要的信息.为此,本文提出了一种基于时间衰减模型的闭合模式挖掘算法 TDMCS.该算法采用了闭合算子提高闭合模式挖掘的效率;通过设置平均衰减因子得到更加平衡的算法查全率和查准率;采用了最大误差阈值配合衰减模型使用,可以有效避免概念漂移,用于挖掘更加合理的闭合模式结果集.大量实验验证得出 TDMCS 算法具有较高的效率,适用于挖掘高密度,长序列和长模式的数据流,适用于不同大小的滑动窗口,且该算法优于其他同类算法.

参 考 文 献

[1] Manku Q. Approximate frequency counts over streaming data//Proceedings of the 28th International Conference on Very Large Data Bases. Hong Kong, China, 2002: 346-357

[2] Yu J X, Chong Z, Lu H, Zhou A. False positive or false negative: mining frequent itemsets from high speed transactional data streams//Proceedings of the 30th International Conference on Very Large Data Bases. Toronto, Canada, 2004: 204-215

[3] Li Guo-Hui, Chen Hui. Mining the frequent patterns in an arbitrary sliding window over online data streams. Journal of Software, 2008, 19(19): 2585-2596(in Chinese)
(李国徽, 陈辉. 挖掘数据流任意滑动时间窗口内频繁模式. 软件学报, 2008, 19(19): 2585-2596)

[4] Chen H, Shu L C, Xia J L, Deng Q S. Mining frequent patterns in a varying-size sliding window of online transactional data streams. Information Sciences, 2012, 215(12): 15-36

[5] Chi Y, Wang H X, Yu P S, Muntz R R. Catch the moment: Maintaining closed frequent itemsets over a data stream sliding window. Knowledge and Information Systems, 2006, 10(3): 265-294

[6] Yen S J, Lee Y S, Wu C W, Lin C L. An efficient algorithm for maintaining frequent closed itemsets over data stream. Next-Generation Applied Intelligence, 2009, 5579(1): 767-776

[7] Tang K M, Dai C Y, Chen L. A novel strategy for mining frequent closed itemsets in data streams. Journal of Computers, 2012, 7(7): 1564-1572

[8] Nori F, Deypir M, Sadreddini M H. A sliding window based algorithm for frequent closed itemset mining over data streams. Journal of Systems and Software, 2013, 86(3): 615-623

[9] Cheng J, Ke Y, Ng W. Maintaining frequent closed itemsets over a sliding window. Journal of Intelligent Information Systems, 2008, 31(3): 191-215

[10] Yen S J, Wu C W, Lee Y S, et al. A fast algorithm for mining frequent closed itemsets over stream sliding window// Proceedings of the 2011 IEEE International Conference on Fuzzy Systems. Taipei, China, 2011: 996-1002

[11] Nabil H M, Eldin A S, Belal M A E. Mining frequent itemsets from online data streams: Comparative study. International Journal of Advanced Computer Science and Applications, 2013, 4(7): 117-125

[12] Li Hai-Feng, Zhang Ning, et al. Frequent itemset mining over time-sensitive streams. Chinese Journal of Computers, 2012, 35(11): 2283-2293(in Chinese)
(李海峰, 章宁等. 时间敏感数据流上的频繁项集挖掘算法. 计算机学报, 2012, 35(11): 2283-2293)

[13] Li H F, Ho C C, Chen H S, Lee S Y. A single-scan algorithm for mining sequential patterns from data streams. International Journal of Innovative Computing, Information and Control, 2012, 8(3A): 1799-1820

[14] Nabil H M, Eldin A S, Belal M A E. Mining frequent itemsets from online data streams: Comparative study. International Journal of Advanced Computer Science and Applications, 2013, 4(7): 117-125

[15] Zhao Y, Zhang C, Zhang S. A recent-biased dimension reduction technique for time series data//Proceedings of 9th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD). Hanoi, Vietnam, 2005: 751-757

[16] Leung C K S, Jiang F. Frequent itemset mining of uncertain data streams using the damped window model//Proceedings of the 2011 ACM Symposium on Applied Computing(SAC'11). Taichung, China, 2011: 950-955

[17] Chang J H, Lee W S. Finding recent frequent itemsets adaptively over online data streams//Proceedings of the ACM SIGKDD Internal Conference on Knowledge Discovering and Data Mining. Washington, USA, 2003: 487-492

[18] Chang J H, Lee W S. Decaying obsolete information in finding recent frequent itemsets over data streams. IEICE Transactions on Information and Systems, 2004, E87-D(6): 1588-1592

[19] Chang J H, Lee W S. estWin: Online data stream mining of recent frequent itemsets by sliding window method. Journal of Information Science, 2005, 31(2): 76-90

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

[21] Chang J H, Lee W S. Finding recently frequent itemsets adaptively over online transactional data streams. Information Systems, 2006, 31(8): 849-869

[22] Chang J H, Lee W S. estMax: Tracing maximal frequent itemsets instantly over online transactional data streams. IEEE Transactions on Knowledge and Data Engineering, 2009, 21(10): 1418-1431

[23] Cohen E, Strauss M. Maintaining time-decaying stream aggregates//Proceedings of the 22th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Data Systems. New York, USA, 2003: 223-233

[24] Cormode G, Tirthapura S, Xu B. Time-decaying sketches for robust aggregation of sensor data. SIAM Journal on Computing, 2009, 34(9): 1309-1339

[25] HewaNadungodage C, Xia Y, Lee J J, Tu Y. Hyper-structure mining of frequent patterns in uncertain data streams. Knowledge and Information Systems, 2013, 37(1): 219-244

[26] Shie B E, Yu P S, Tseng V S. Efficient algorithms for mining maximal high utility itemsets from data streams with different models. Expert Systems with Applications, 2012, 39(17): 12947-12960

[27] Jiang N, Gruenwald L. CFI-Stream: Mining closed frequent itemsets in data streams//Proceedings of the ACM SIGKDD Internal Conference on Knowledge Discovering and Data Mining. New York, USA, 2006: 592-597

[28] Lee H F, Ho C C, Lee S Y. Incremental updates of closed frequent itemsets over continuous data streams. Expert System with Applications, 2009, 36(2): 2451-2458

HAN Meng, born in 1982, Ph. D. candidate, associate professor. Her research interests include data mining and machine learning.

WANG Zhi-Hai, born in 1963, Ph. D., professor, Ph. D. supervisor. His research interests include data mining and artificial intelligence.

YUAN Ji-Dong, born in 1989, Ph. D. candidate. His research interests include machine learning and data mining.

Background

Searching for frequent patterns in a continuous transactional data stream has become important for many applications, such as web-server log and click stream mining, online analytical processing, e-business and stock data analysis, and sensor network data analysis, et al. In most of these applications, recent transactions are more important than historical ones. Therefore, researchers have proposed the Sliding Window Model (SWM) and the Time Decay Model (TDM). Based on SWM and TDM, some methods were used to discover frequent patterns on data streams[1-26]. The drawbacks are that (1) minimum support is used to discover patterns, but concept drift is not processed; (2) Although SWM is used, the weights of transactions are same in it; (3) Although based on SWM and TDM, complete patterns set are generated which are difficult to understand; (4) In recent years, two kinds of decay factors are set in TDM. First, decay factor is set to a random value in the range of (0,1). Second, decay factor is set to the upper/lower bound with the estimation of 100% Recall or 100% Precision of algorithm. In a word, the disadvantages of existed techniques are that importance of one side of Recall and Precision is emphasized, or instable result sets with a random decay factor are generated.

In order to avoid concept drift, distinguish recent transactions from historical ones, discover compact pattern set, and apply to high dense transactions and long patterns, a novel algorithm based on time decay model is proposed in this paper. Mainly works and innovations are that: (1) a novel method to set decay factor f is proposed. In order to balance high Recall with high Precision, an average decay factor on the basis of lower bound and higher bound is raised; (2) The three layer frame: minimum support-maximum error rate-decay factor is proposed to avoid concept drift; (3) Closed operator is used to generate closed patterns efficiently; (4) A novel algorithm TDMCS based on SWM and TDM is provided to discover closed frequent patterns on data streams. Compared with some classical methods, TDMCS gets lossless, compact and more accurate result set.