Journal of Computer Applications
计算机应用,2019,39(3):719 -727
ISSN 1001-9081
CODEN JYIIDU
2019-03-10
http://www.joca.cn

文章编号:1001-9081(2019)03-0719-09
DOI:10.11772/j.issn.1001-9081.2018081712

数据流频繁模式挖掘综述

韩　萌*,丁　剑
(北方民族大学 计算机科学与工程学院,银川 750021)
(*通信作者电子邮箱 compute2006_2@126.com)

摘　要:一些先进应用如欺诈检测和趋势学习等带来了数据流频繁模式挖掘的发展。不同于静态数据,数据流挖掘面临着时空约束和项集组合爆炸等问题。对已有数据流频繁模式挖掘算法进行综述并对经典和最新算法进行分析。按照模式集合的完整程度进行分类,数据流中频繁模式分为全集模式和压缩模式。压缩模式主要包括闭合模式、最大模式、top-k 模式以及三者的组合模式。不同之处是闭合模式是无损压缩的,而其他模式是有损压缩的。为了得到有趣的频繁模式,可以挖掘基于用户约束的模式。为了处理数据流中的新近事务,将算法分为基于窗口模型和基于衰减模型的方法。数据流中模式挖掘常见的还包含序列模式和高效用模式,对经典和最新算法进行介绍。最后给出了数据流模式挖掘的下一步工作。

关键词:数据流;数据流挖掘;频繁模式挖掘;序列模式挖掘;高效用模式挖掘
中图分类号:TP18　　**文献标志码**:A

Survey of frequent pattern mining over data streams

HAN Meng*, DING Jian
(*School of Computer Science and Engineering, North Minzu University, Yinchuan Ningxia* 750021, *China*)

Abstract: Advanced applications such as fraud detection and trend learning lead to the development of frequent pattern mining over data streams. Data stream mining has to face more problems than static data mining like spatio-temporal constraint and combinatorial explosion of itemsets. In the paper, the existing frequent pattern mining algorithms over data streams were reviewed, and some classical algorithms and some newest algorithms were analyzed. According to the completeness of pattern set, frequent patterns of data stream could be divided into complete patterns and compressed patterns. Compressed patterns include closed frequent patterns, maximal frequent patterns, top-k frequent patterns and combinations of them. Between them, only closed frequent patterns are losslessly compressed. And constrained frequent pattern mining was used to narrow the result set obtained, satisfying the user's demand more. Algorithms based on sliding window model and time decay model were used to better handle recent transactions which occupy an important position in data stream mining. Moreover, two of the common algorithms, sequential pattern mining and high utility pattern mining algorithms were introduced. At last, further research direction of frequent pattern mining over data streams were discussed.

Key words: data stream; data stream mining; frequent pattern mining; sequential pattern mining; high utility pattern mining

0　引言

在一些新兴的应用场景下,例如智能城市、大型基础设施监控、物联网等,数据产生的速度越来越快。数据流(data stream)被认为是高速率数据,通常被认为是大数据,它是无限的、快速的、变化的和有序的。在某些环境下,数据流的处理方法必须快速且能适应变化。数据流模型面临的主要约束[1]包括:

1)数据量巨大,可以认为是无限的。因此,无法存储所有的数据。合理的方法是存储数据的概要信息。

2)数据到达的速度快。因此,需要实时处理数据,且处理后数据即被丢弃。

3)数据项的分布可能随着时间而变化。因此,历史数据会变得无用甚至有害。

在进行数据流挖掘时,需要考虑这些约束。近年来,研究者关注数据流中分类、聚类以及模式挖掘等问题的研究。频繁模式是指在数据集中出现的次数高于用户定义的最小支持数/度阈值的项的集合。数据流频繁模式挖掘方法通常可以分为两类。第一类是基于统计来估计模式频度。如算法 Sticky Sampling[2]采用统计抽样技术来估计项集的支持数。它是挖掘数据流频繁模式的近似算法,是基于概率统计的,丢失可能频繁模式的概率不高于用户定义的参数值。算法 Lossy Counting[2]是数据流频繁模式挖掘经典方法之一,它给定了错误参数来挖掘数据流中的频繁模式。第二类是基于草图来近似估计模式频度。草图是一种概率数据结构,它用于处理项出现频度。最经典的算法是 CountSketch[3],它使用有

收稿日期:2018-08-17;**修回日期**:2018-11-09;**录用日期**:2018-11-12。
基金项目:国家自然科学基金资助项目(61563001);宁夏自然科学基金资助项目(NZ17115)。
作者简介:韩萌(1982—),女,河南商丘人,副教授,博士,CCF 会员,主要研究方向:大数据分类、模式挖掘;　丁剑(1977—),男,宁夏固原人,副教授,主要研究方向:大数据分类、模式挖掘。

限的存储空间来估计数据流中频繁项，主要依赖于草图数据结构。Pyramid 框架[4]使用草图数据结构来发现数据流中频繁模式，该算法在算法正确率以及算法速度方面有一定的优势。

挖掘数据流时研究者认为新的事务比历史事物重要得多。使用窗口模型和衰减模型可以很好地处理新事务。经典的基于窗口模型的算法包括 Lossy Counting[2]和 DSM_FT[5]。最常见的窗口模型是滑动窗口模型，如算法 MSW[6]、MFI-CBSW[7]、MFI-TimeSW[8]和 TMFI[9]使用滑动窗口模型挖掘数据流中频繁模式。经典的基于衰减模型的算法有 estDec[10]和estDec+[11]。除此之外，根据数据流处理方法，可以分为模式增长方法、假阳性和假阴性方法等；根据挖掘模式结果集合的精简程度，可以分为全集模式挖掘方法和压缩模式挖掘方法等；根据模式的特征，可以分为频繁模式、频繁序列模式、频繁树模式、频繁图模式挖掘方法等。这些内容会在下文进行介绍。

1 基本概念

数据流通常定为连续到达的事务项(item)集合，由于数据量巨大，需要在线实时处理。给定数据流 $DS = \langle T_1, T_2, \cdots, T_i, \cdots \rangle$ 是连续到达的项集(itemset)/实例(example)/事务(transaction)序列。变量 $T_i(i = 1,2,\cdots)$ 是第 i 时刻产生的第 i 条的事务。每个 T_i 包含一个唯一的标识 TID，如表 1 所示。表 1 是包含 5 条事务的数据流。由于数据流的无限特征，频繁项(item)/项集(itemset)/模式(pattern) P 的频度定义为最新的 n 条事务数据中包含模式 P 的事务个数，记为 $freq(P)$。模式 P 的支持度表示为 $support(P) = freq(P)/n$。数据流中的频繁模式概念如定义 1 所示。

定义 1 频繁模式。给定数据流 S 包含最新的 n 条事务，给定最小支持度阈值 $\theta(\theta \in (0,1])$。如果项集 P 满足 $support(P) = freq(P)/n \geqslant \theta$，则 P 为频繁模式。

如表 1 中数据流所示，设 $n = 5$，令 $\theta = 0.5$，即频繁模式需要满足最小频度为 $n \times \theta = 5 \times 0.5 = 2.5$，则可以得到频繁模式集合{{a(4)},{b(3)},{c(4)},{d(3)},{a,c(3)},{c,d(3)}}，其中{c,d(3)}表示频繁模式{c,d}的频度为 3。这 6 条项集都是满足最小支持度的频繁模式，因为它们出现的频度都大于 2.5。

频繁模式挖掘的一个很大的不足在于全集频繁模式的数量巨大，尤其当最小支持度阈值很小或事务长度很长时，挖掘出来的模式呈指数级增长。为了减少模式的数量，可以挖掘压缩模式，如最大模式、闭合模式和 top-k 模式等，如定义 2 ~ 定义 4 所示。

定义 2 闭合频繁模式。给定数据流 S，给定最小支持度阈值 $\theta(\theta \in (0,1])$。给定项集 P 和 Q 满足 $support(P) \geqslant \theta$，$support(Q) \geqslant \theta$。若不存在 Q 满足 $P \subset Q$ 且 $freq(P) = freq(Q)$，则 P 为闭合频繁模式。

定义 3 最大频繁模式。给定数据流 S，给定最小支持度阈值 $\theta(\theta \in (0,1])$。给定项集 P 和 Q 满足 $support(P) \geqslant \theta$，$support(Q) \geqslant \theta$。不存在 Q 满足 $P \subset Q$，则 P 为最大频繁模式。

定义 4 top-k 频繁模式。将所有已挖掘出的项集按照支持度由高到低的顺序排序，令 θ' 为第 k 个项集的支持度。则对于频繁项集 P，若满足条件 $support(P) \geqslant \theta'$，则称 P 为 top-k 频繁模式。

除此之外，约束挖掘也是减少模式数量的有效方法且可以提高模式挖掘的效率。它可以得到更少的但是更有趣的模式结果集，这些模式更有利于用户的使用；并且还可以利用这些约束的性质进一步地减小算法的搜索空间，最终提高算法的运行效率。约束概念可以表示为定义 5 所示，其中 I 是项的集合，即 $I = \{i_1, i_2, \cdots, i_m\}$。因此挖掘出基于约束的频繁模式 P，则 P 应满足表达式 $(support(P) \geqslant \theta) \wedge (Constraint(P) = \text{true})$。

约束概念可以表示为定义 5 所示，其中 I 是项的集合，即 $I = \{item_1, item_2, \cdots, item_m\}$。

定义 5 约束[12]。约束 $Constraint$ 是 I 的幂集，即 $Constraint: 2^I \rightarrow \{\text{true}, \text{false}\}$。对任意一个频繁项集 P，如果 $Constraint(P) = \text{true}$，则称 P 满足约束 $Constraint$。

示例 1 包含 5 个事务的数据流如表 1 所示。令 $n = 5$，$\theta = 0.4$，则得到全集模式、最大模式、闭合模式和 top-2 模式如表 2 所示。

表 1 数据流

Tab. 1 Data stream

TID	事务	TID	事务	TID	事务
1	{a,b}	3	{b,c,d,f}	5	{a,c,f}
2	{a,c,d,e}	4	{a,b,c,d}		

表 2 模式集合

Tab. 2 Pattern sets

模式类型	模式集合

全集模式	a(4);b(3);c(4);d(3);f(2);
	a,b(2);a,c(3);a,d(2);b,c(2);b,d(2);
	cd(3);cf(2);b,c,d(2)

闭合模式	a(4);b(3);c(4);a,b(2);a,c(3); a,d(2);cd(3);cf(2);b,c,d(2)
最大模式	a,b(2);a,c(3);a,d(2);cf(2);b,c,d(2)
top-2 模式	a(4);b(3);c(4);d(3);a,c(3);cd(3)

从表 2 中可以看出，压缩模式数量明显小于全集模式。在 $\theta = 0.4$ 条件下可以得到 13 个全集模式，9 个闭合模式，5 个最大模式和 6 个 top-2 模式。其中闭合模式是无损压缩形式，它包含全集模式中的所有信息，其余的为有损压缩模式。

若定义约束为 $Constraint1$ 至 $Constraint4$ 所示，则可以得到满足不同约束的频繁模式集合如表 3 所示，可以得到 4 个满足 $Constraint1$，7 个满足 $Constraint2$，4 个满足 $Constraint3$ 和 3 个满足 $Constraint4$ 的频繁模式。约束条件越复杂具体则得到的模式集合越有趣，更利于满足用户的需求。

1) 约束 $Constraint1$：对任意一个频繁项集 P，如果其包含项{a}，则称其满足约束 $Constraint1$。即 $Constraint1(P) \equiv (\{a\} \subseteq P)$。

2) 约束 $Constraint2$：对任意一个频繁项集 P，如果其长度为 2，则称其满足约束 $Constraint2$。即 $Constraint2(P) \equiv (length(P) = 2)$。函数 $length(P)$ 表示 P 的长度。

3) 约束 $Constraint3$：对任意一个频繁项集 P，如果它是满足 $Constraint1$ 的闭合模式，则称其满足约束 $Constraint3$。即 $Constraint3(P) \equiv (closed(P) = \text{true}) \wedge (Constraint1(P) = \text{true})$。函数 $closed(P)$ 用于判断 P 是否为闭合模式。

4) 约束 *Constraint*4:对任意一个频繁项集 P,如果它是满足 *Constraint*1 和 *Constraint*2 的模式,则称其满足约束 *Constraint*4。即 $Constraint4(P) \equiv (Constraint1(P) = \text{true}) \wedge (Constraint2(P) = \text{true})$。

表 3　满足不同约束的模式集合

Tab. 3　Pattern sets meeting different constraints

约束	模式集合
*Constraint*1	a(4);a,b(2);a,c(3);a,d(2)
*Constraint*2	a,b(2);a,c(3);a,d(2);b,c(2);b,d(2);cd(3);cf(2)
*Constraint*3	a(4);a,b(2);a,c(3);a,d(2)
*Constraint*4	a,b(2);a,c(3);a,d(2)

本章中常见的符号和函数的含义如表 4 所示,包含数据集合、项的取值范围,以及常用的支持数、支持度和最小支持度阈值的表示符号。

表 4　常见符号或函数的含义

Tab. 4　Meanings of symbols or functions

符号	含义	符号	含义
D/*DS*	数据集合	*closed*()	判断是否闭合模式
I	项的集合	CoFPs	全集频繁模式
freq()	支持数/频度	CFPs	闭合频繁模式
support()	支持度	MFPs	最大频繁模式
θ	最小支持度阈值	TkFPs	top-*k* 频繁模式
Constraint()	判断是否满足约束条件	CdFPs	约束频繁模式
length()	项集/模式长度		

2　数据流中频繁模式类型

本节重点介绍压缩模式和约束模式,表 5 给出了几种模式的比较。全集模式是全部模式的集合;闭合模式是无损的压缩方式,可以压缩大量的模式;最大模式是有损压缩方式,得到的模式数量少于闭合模式;top-*k* 模式仅取 *k* 组频度最高的模式,通常数量很少;约束模式是挖掘满足用户需求的模式,模式数量也会明显小于模式全集。

表 5　各类型模式的比较

Tab. 5　Comparison of different patterns

模式类型	是否有损	压缩程度	模式类型	是否有损	压缩程度
CoFPs	N	无压缩	TkFPs	Y	强于 CoFPs
CFPs	N	强于 CoFPs	CdFPs	Y	强于 CoFPs
MFPs	Y	强于 CFPs			

2.1　全集频繁模式

数据集合中出现次数不低于用户定义阈值的所有频繁模式集合称为全集频繁模式(Complete Frequent Patterns, CoFPs)。一般来说,CoFPs 中包含的模式数量巨大,且包含有大量短的无意义的模式,因此不利于用户的使用。

近年来,挖掘 CoFPs 的方法有很多。如算法 FIS_EDS[13] 采用界标窗口模型挖掘数据流中的 CoFPs。该算法设计数据结构 Bit-Table 存储模式信息。使用 Bit-Table 可以比较容易地得到模式频度并且避免了解析整个窗口中的事务信息。CanTree-GTree 算法[14] 基于 Hadoop 框架挖掘数据流中的 CoFPs。该算法使用投影树结构 CanTree 和 GTree 存储模式信息,并采用自顶向下的遍历方式搜索整棵树。该算法可以得到准确完整的模式集合。SysTree 算法[15] 采用并行技术挖掘数据流中的 CoFPs。该算法基于树结构,且分别采用界标窗口和滑动窗口来处理事务。当频繁项数量少时,该算法可以实现很准确的挖掘过程;当频繁项数量大时,挖掘过程是近似的且是非假阳性的。PFIMoS 算法[16] 挖掘不确定数据流中的概率频繁模式。该算法是深度优先的,且设计了一种索引树结构 PFIT 存储数据概要信息。由于以上策略,与已有算法相比,该算法可以减低时间和空间消耗。

2.2　闭合频繁模式

闭合频繁模式(Closed Frequent Patterns, CFPs)是强大的频繁模式的表现方式,因为它们消除冗余信息。一般来说,闭合频繁模式比全集频繁模式中的模式数量少得多,且闭合模式包含了全集频繁模式中的全部信息,是一种无损压缩模式。

近年来,在数据流中挖掘闭合频繁模式的方法有很多。经典算法包括 Moment[17]、CloStream +[18] 和 TMoment[19] 等采用滑动窗口挖掘数据流的闭合频繁模式。Moment 算法使用事务滑动窗口挖掘数据流中的 CFPs。该算法中设计一种新的数据结构 CET(Closed Enumeration Tree)来存储 CFPs 和临界 CFPs。使用 CET 相比前缀树而言,可以减少大量的树节点,降低插入和删除模式时的时间和空间消耗。CloStream + 算法设计一种新的数据结构挖掘数据流中的 CFPs。算法设计两个新的数据结构 ClosedTable 和 CidList,前者存储与闭合模式有关的信息,后者存储与数据流中出现的项(item)有关的信息。二者的交叉点是都存储事务的标识 TID。该算法不需要消耗大量时间进行空间搜索,但是相对应地会付出更多的时间消耗。TMoment 算法使用 TCET(Transaction translate Closed Enumeration Tree)数据结构挖掘数据流中的 CFPs。TCET 是一种前缀树结构,每个节点表示一个闭合项集。当滑动窗口更新数据时,算法使用深度优先策略遍历该树来更新闭合频繁模式信息。TCET 存储窗口中内容和闭合模式信息时需要的内存空间比较少。

AFPCFI-DS(Algorithm based on FP-tree for mining Closed Frequent Itemsets in Data Stream)算法[20] 使用 FP 树结构挖掘每个滑动窗口中的 CFPs。每次处理新的滑动窗口时,算法会首先更新头表(head table),然后依据头表更新 FP-tree。每当添加或删除事务时,算法使用局部更新策略来降低时间消耗。TDMCS 算法[21] 采用滑动窗口模型和时间衰减模型挖掘可变数据流中的 CFPs。为了提高效率,设定的闭合频繁模式满足支持度-误差度-衰减因子这三层结构。该算法设计一种均值衰减衰减因子,使得到的模式结果集合具有高且均衡的查全率和查准率。这个算法可以有效地处理具有概念漂移问题的数据流模式挖掘过程。FLMCFI(Fast and Lossless Mining of Closed Frequent Itemsets)算法[22] 提出一种新的搜索机制挖掘数据流中的 CFPs。该搜索机制可以用于解决数据流中观察到的概念漂移现象。

2.3　最大频繁模式

最大频繁模式(Maximal Frequent Patterns, MFPs)也称为最长频繁模式。如果一个频繁模式没有父模式,则它是最大频繁模式。最大频繁模式的目的是取长度最长的模式集合。最大频繁模式集合中模式数量会明显低于全集模式中的模式数量。但是由于它是一种有损压缩方式,因此无法从最大模式集合中反推出全集模式集合。

经典的 MFPs 挖掘算法之一是estDec+[11]。该算法使用压缩前缀树 CP-tee 挖掘 MFPs,与前缀树结构相比可以降低内存消耗。CP-tree 上的每个节点都存储多个项集的信息。因此可以通过合并或分类节点来控制 CP-tree 的大小。该算法可以使用有限的内存空间存储大量的项集信息。WMFP-SW 算法[23]使用滑动窗口模型发现数据流中的权重 MFPs。一种新的树结构 WMFP-SW-tree 和阵列结构 WMFP-SW-array 用于存储权重模式信息,可以有效提高挖掘的效率。挖掘出的 MFPs 需要满足最小支持度阈值和权重约束。TV 和 iTV 算法[24]基于 spark 挖掘数据流中的 MFPs。该文中提出一种 ASP-Tree 结构存储模式信息,与模式增长算法相比可以降低内存消耗,同时可以减少搜索时间。WOB 点阵算法[25]同样使用滑动窗口模型挖掘数据流中的权重 MFPs。该算法使用 FP 树存储模式信息,由于使用点阵技术可以避免反复重建整个树机构。

2.4 top-k 频繁模式

top-k 频繁模式(Top-k Frequent Patterns, TkFPs)用于挖掘数据流中出现频度最高的 k 组模式。一般而言,MFPs 和 TkFPs 的压缩程度是强于 CFPs,且都是有损压缩。已有的算法挖掘 TkFPs 或闭合 TkFPs,如算法 FSS[26]、TKRES[27]挖掘 TkFPs。为了限定模式结果集的大小,算法 Top-k-FCI[28]、FCI_max[29]、TFRC-Mine[30]挖掘数据流中的闭合 TkFPs。

TFRC-Mine 算法采用最小长度策略挖掘闭合的 TkFPs。该算法不需要设置支持度阈值。为了处理项之间的冗余和兴趣度关联,算法关注闭合模式且设置了最小长度约束。算法中设计了一种新的压缩位向量表示形式,用于剪枝无趣的候选项。TKRES 算法挖掘信号数据流中的 TkFPs。挖掘出的模式是连续的形式,可以称之为片断(episode)。这样更利于监督者对得到的 TkFPs 结果集进行分析,从而作出合理的判断。TKRES 算法采用 k 树结构来保存 top-k 片断和它们的出现信息。TwMinSwap[31]算法采用项抽样策略挖掘数据流中的 TkFPs。核心思想是采用时间权重来统计模式的重要性,随着时间的推进历史模式权重会减少。该算法仅需要 $O(k)$ 内存空间存储模式信息。算法采用最小长度策略挖掘闭合的 TkFPs。FTK 算法[32]使用任意大小的滑动窗口模型挖掘 TkFPs。该算法使用的内存空间仅与窗口大小成对数关系,而不是线性关系;因此,与已有方式相比得到的模式结果集合更准确,且时间消耗会明显减少。

2.5 约束频繁模式

约束可以用于筛选数据或结果集,可以令用户找到真正有兴趣的内容。在频繁模式挖掘过程中,约束挖掘可以减少模式的数量,提高模式集合的利用率。约束挖掘(constrained mining)是指用户仅对频繁模式挖掘结果的一部分感兴趣,即模式需要满足用户定义的约束,这样挖掘出的模式称为约束频繁模式(Constrained Frequent Patterns, CdFPs)。

例如,最小支持度阈值约束就是反单调约束,即如果一个项集是非频繁的,则其父项集也是非频繁的。如一个项集 P 满足约束 $ConstraintA(P) \equiv (\{i,j\} \subseteq P)$,其中 i, j 为项。则 P 的父项集也满足约束 $ConstraintA$。即 $ConstraintA$ 是满足单调性的。如果项集 P 不满足约束 $ConstraintA$,其父项集也可能满足约束 $ConstraintA$。

Leung 等[33]设计一种基于树结构的算法发现不确定数据流中的 CdFPs。为了发现满足用户需求的模式,加入了对类值取值范围的约束。设计树结构存储满足约束的项集,对第一批数据直接存储内容。树中的节点包含两个字段——项名和支持度。算法从树中挖掘满足用户定义支持度的频繁模式。Silva 等[12]设计了多个约束发现数据流中的 CdFPs。算法将约束加入压缩前缀模式树结构从而生成满足用户不同约束的模式。树中的每个节点包含一个项,一个近似支持度和最大误差值。树中的边连接同时出现的项用于创建模式。这样每个节点对应一个近似模式,模式由根节点到这个节点中的项组成。这样可以从全集模式中筛选出可能的模式,会降低内存和时间消耗。Hu 等[34]提出算法用于挖掘数据流中的闭合 CdFPs。首先将数据流分成片段,然后使用树结构存储可能的约束闭合频繁项集。对于每一段到达的数据,首先建立相应的局部树数结构,然后更新和剪枝全局树结构用于挖掘约束闭合频繁模式。该算法定义了频繁项集需要包含的项内容的约束,最终目的是筛选频繁模式集合,找到更有用的关联规则。Cuzzocrea 等[35]在不确定数据流中发现满足用户约束的 CdFPs。该算法处理的是无限信号网络数据流,设计了两种约束-简洁反单调约束和简洁非反单调约束-的频繁模式挖掘。算法是基于树结构的分布挖掘系统,用于发现满足简洁约束的频繁模式,首先发现满足约束的局部频繁项集,然后发现满足约束的全局频繁项集。Kiran 等[36]设计一种模式增长算法,采用周期频繁树结构发现大数据中的周期 CdFPs。这些模式满足最小支持度阈值和最大到达间隔的约束。定义了局部周期性和周期性用于找到局部和全局频繁模式。如果一个模式的局部周期性不满足用于定义的最大周期阈值,则它是非周期的频繁模式,可以用于避免对这些频繁模式进一步搜索。因此,使用局部周期性可以降低找到周期频繁模式的计算成本。Cagliero 等[37]分析无线传感网络中的历史信号数据流,以识别其中读数出现异常趋势的传感器组合。作者设计基于距离的约束来发现信号数据流中的非频繁权重模式。这种约束可以用于分析在不同环境或者不同条件下获取的传感器测量结果。

3 数据流中频繁模式挖掘关键技术

数据流是海量、快速、有序的数据序列,因此在有限的时间和空间中挖掘数据流中的频繁模式面临很大的挑战。研究者提出了许多在数据流中挖掘频繁模式的方法,包括窗口方法、衰减方法、先验方法、模式增长方法、精确方法、近似方法、假阳性方法、假阴性方法、在线方法和离线方法等,本文对几种主要方法介绍。表 6 给出了几种经典数据流频繁模式挖掘算法概述,从使用的窗口模型、衰减制度、使用方式、采用何种近似算法以及发现的模式类型等方面进行介绍。

表 6 数据流频繁模式挖掘算法特征

Tab. 6 Features of frequent pattern mining algorithms

算法	窗口	衰减	方法	模式集合
Lossy-Counting	landmark	N	lossy counting	CoFPs
estDec	damped	Y	prefix tree	CoFPs
estDec+	damped	Y	prefix tree	MFPs
Moment	sliding	N	FP-growth	CFPs
NewMoment	sliding	N	FP-growth	CFPs
TMoment	sliding	N	prefix tree	CFPs
MSW	sliding	Y	prefix tree	CoFPs
SWP-tree	sliding	Y	prefix tree	CoFPs
CloStream	sliding	N	Closed Table	CFPs

3.1　窗口方法

数据流处理常用的窗口模型有三种:界标窗口(landmark window)、滑动窗口(sliding window)和倾斜窗口/衰减窗口(damped window)。界标窗口固定窗口的起始点 s,窗口的另一端 e 随着数据的不断到达而增长,不断地把得到的结果输出。算法处理 T_s 和 T_e 之间的最新事务数据。滑动窗口对窗口的起始与结束都没有明确的定义,定义的是窗口的长度 N。即算法处理 $T_{new-N+1}$ 和 T_{new} 之间的最新事务数据。当新事物 T_{new+1} 到达时,历史事务 $T_{new-N+1}$ 会移除窗口。处在窗口内的事务具有相等的重要性。窗口保持一定的长度在数据流上进行滑动,不断地把得到的结果输出。衰减窗口是由固定的时间起点 s,窗口的另一端 e 随着数据流的到达不断增长。但是不同时间段的数据具有不同的权重。即历史数据所占权重很小,而新数据权重大。算法处理 T_s 和 T_e 之间的最新事务数据。

数据流挖掘最常用的窗口模型是滑动窗口模型(Slidng Window Model, SWM)。滑动窗口模型包括固定 SWM 与可变 SWM。在任意时刻,前者中窗口内最新事务的个数是固定的;而后者的最新事务个数是可变的。算法设计对窗口内的最新事务进行处理。

图 1 是采用固定滑动窗口处理新事务的过程。图 1(a)中处理最新事务 T_{new},采用的滑动窗口大小为 N。当处理新事务 $T_{new'}$ 时,由于滑动窗口大小 N,则事务 $T_{new-N+1}$ 和 $T_{new'-N+1}$ 之间的 $|new'-new|$ 个事务将被移出窗口,如图 1(b)所示。移出的方法可以是单个移出或批量移出。为了避免出现窗口内的事务出现概念漂移现象,一般窗口的大小不会很大。

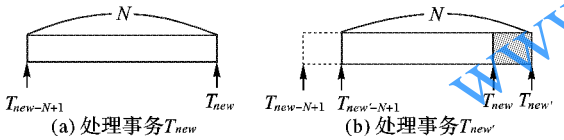

图 1　固定大小滑动窗口的移动

Fig. 1　Moving the fixed size sliding window

以采用概念漂移检测器调整滑动窗口为例介绍窗口的变化,当出现概念漂移则收缩窗口,否则扩展窗口。假设给定窗口大小 N,图 2 表示了可变滑动窗口扩展和收缩的过程,其中 $|new'-new|$ 表示 $T_{new'}$ 与 T_{new} 之间的实例个数或者时间的距离。图 2(b)表示没有发生概念漂移时,滑动窗口会扩展窗口的尺寸,从 N 扩展到 $N+|new'-new|$。图 2(c)表示发生概念改变时,滑动窗口会缩减尺寸,窗口大小会从 N 收缩至 N'。

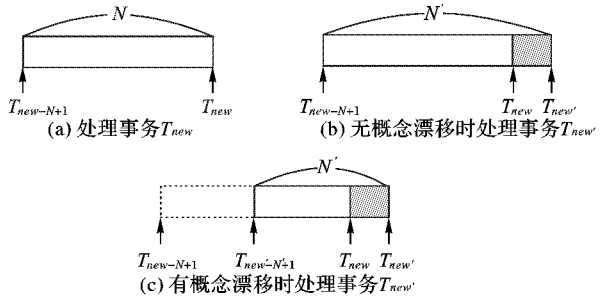

图 2　可变滑动窗口的扩展和收缩

Fig. 2　Expanding and shrinking of variable-size sliding window

固定 SWM 会按照经验给定窗口大小,且该值是固定的。如 SWCA[38]、EclatDS[39]、MSW[6]、SWP-Tree[40] 和 SA-Miner[41] 采用滑动窗口发现模式结果集。TKRES 算法[27] 使用滑动窗口模型挖掘信号数据流中的 top-k 频繁模式。窗口中包含 m 组数据,每一组数据是用户定义的时间单位内产生的数据,如一天或者一周。CanTree-GTree 算法[14] 基于固定滑动窗口和 Hadoop 框架挖掘数据流中的全集频繁模式。当一个窗口装满事务时,历史事务将被移出,新近事务会被加入。FTK 算法[32] 使用滑动窗口模型挖掘 top-k 频繁模式。该算法给出了滑动窗口大小的上限,可以处理上限范围内任意大小窗口中的事务从而得到频繁模式。该算法使用的内存空间大小与窗口大小成对数关系。算法设定滑动窗口大小范围从 3600 ~ 360000 个事务,用来验证算法的优势。虽然是任意大小,但在每次实验中,窗口大小是固定的。WOB 点阵算法[25] 使用滑动窗口模型挖掘数据流中的权重 MFPs。通过窗口的滑动删除历史事务信息增加新近事务信息,这样可以避免重建整个树结构。SysTree 算法[15] 基于树结构且分别采用界标窗口和滑动窗口来挖掘数据流中的频繁模式。该文中解释道大窗口会产生大量的模式,小窗口会产生少的模式;因此,不能说明哪个窗口大小更好,这取决于不同的用户需求。

可变滑动窗口大小的改变有多种策略。在时间衰减模型中按照高查全率假设,通过对频繁项集的衰减频度估计来缩短或扩大窗口[40]。FIMoTS 算法[42] 使用基于时间戳的滑动窗口模型,接着转化为基于事务的可变滑动窗口进行处理。窗口大小的改变受到模式的频度变化影响。VSW 算法[43] 提出一种可变大小滑动窗口发现数据流中的频繁模式。窗口大小由达到数据的概念改变数量动态决定。当概念平稳时,窗口扩展。而当概念改变时,窗口收缩。CCFPM 算法[44] 使用可变滑动窗口发现数据流中的闭合频繁模式。窗口的收缩和扩展受到概念改变的影响。

3.2　衰减方法

很多数据流频繁模式挖掘算法为每个事务赋予相同的重要性或权重。然而,一些时间敏感的数据流应用认为最新产生的事务比历史事物更重要。时间衰减模型是处理时间敏感数据流频繁模式挖掘的一种有效方法,它是一种随着时间的推移而逐步衰减历史模式支持数权重的方法,它强调新近事务产生模式的重要性。衰减方法的核心是衰减因子的设置,通常有三类设置方式:随机值、固定值、计算值。随机值设置衰减因子的方法的不足在于随机性使得得到的模式结果集合不稳定;固定值设置方法的优劣取决于专家知识;计算值会根据算法中设计的其他参数值,如窗口大小、最小支持度等进行计算,从实验验证这种方式更合理。常见的几种设置衰减因子与衰减函数的方法如表 7 所示。

如 MSW(Mining Sliding Window)算法[6] 是使用固定滑动窗口模型和时间衰减模型挖掘数据流中的全集频繁模式的经典算法之一。该算法使用一种滑动窗口树 SW-tree 存储最新的模式信息,并周期性地对树结构剪枝,去除历史频繁模式和不频繁的模式。该算法使用时间衰减模型逐步降低历史事务模式支持数的权重,并由此来区分最近产生事务与历史事务的模式。算法 SWP-Tree[40] 使用可变滑动窗口发现数据流中的频繁模式。为了强调最新频繁模式,使用时间衰减模型来区分新旧事务产生的模式;设计一种增量更新的树结构记录模式信息,从而提高事务的处理速度。DFPMiner 算法[45] 挖掘数据流中的全局频繁模式,它动态构建全局模式树,利用时间指数衰减函数对模式树中各模式的支持数进行统计,以此发现界标

窗口内的模式。PFP-growth 算法[46]用于挖掘事务不确定数据流中的频繁模式,它使用概率衰减窗口模型,通过计算各概率数据项的期望支持度以发现模式。IncSpam 算法[47]使用固定衰减因子值的时间衰减模型发现可扩展滑动窗口模型中的频繁项集。它使用可扩展排序树存储所有当前滑动窗口内的模式,可以减少时间和内存消耗。λ-Hcount 算法[48]采用时间衰减模型计算数据流中的频度,其中设定衰减因子为 0.98 ~ 1 的常量值。采用哈希函数估计数据流中项的密度值,从而发现频繁模式。TDMCS 算法[49]采用衰减模型挖掘数据流中的闭合频繁模式。提出一种均值衰减因子方法,可以得到稳定的和查全率和查准率更均衡的模式结果集合。TwMinSwap 算法[31]使用衰减方法挖掘数据流中的 top-k 频繁模式。该算法采用衰减评估模式的频度,核心思想是随着时间的推进历史模式频度会逐渐衰减。衰减因子的取值是范围是(0, 1)。

表 7 衰减方法

Tab. 7 Decay methods

算法	衰减因子 f	衰减函数
MSW	$\sqrt[2N-\theta N-1]{[(\theta-\varepsilon)/\theta]^2} < f < 1$	$f^{t_{cur}-t_i}$
SWP-Tree	$\sqrt[2N-\theta N-1]{[(\theta-\varepsilon)/\theta]^2} < f < 1$	$f^{t_{cur}-t_i}$
DFPMiner	$0 < f < 1$	$2^{-f(t_{cur}-t_i)}$
PFP-growth	$0 < f < 1$	$f^{t_{cur}-t_i}$
IncSpam	$b^{\frac{1}{h}}, b < 1, h > 1$	$f^{t_{cur}-t_i}$
λ-HCount	$0.98 < f < 1$	$f^{t_{cur}-t_i}$
TwMinSwap	$0 < f < 1$	$f^{t_{cur}-t_i}$
TDMCS	$f = (\sqrt[2N-\theta N-1]{[(\theta-\varepsilon)/\theta]^2}+1)/2$	$f^{t_{cur}-t_i}$

4 其他模式技术

数据流中挖掘出的模式除了频繁模式以外,还包含:序列模式(Sequential Patterns, SPs)[50-51],用于发现具有先后次序的复杂项集;高效用模式(High Utility Patterns, HUPs)[52-53],用于发现具有高价值或效用的项集;图模式(SubGraphs Patterns, SGPs)[54-55],用于发现满足用户阈值的子图结构;片断(EPisode, EP)[27],用于发现连续的事件(event);非频繁模式(INFrequent Patterns, INFPs)[37],用于发现出现频度低的项集;概率频繁模式(Probabilistic Frequent Patterns, PFPs)[16],用于发现不确定数据流中满足最小支持度和最小概率参数的频繁项集;以及其他类型的模式等[56]。这些模式的相同之处是它们都是频繁的,不同之处可以从项/项集是否有序、权重是否相等和是否连续三个方面区分,具体如表 8 所示。针对不同的数据特征可以挖掘出不同类型的模式,不同的模式具有不同的应用。本章主要介绍序列模式和高效用模式。

表 8 模式的区别

Tab. 8 Differences of patterns

模式	项/项集有序	项/项集权重相等	项/项集连续
FPs	N	Y	N
INFPs	N	Y	N
PFPs	N	Y	N
SPs	Y	Y	N
HUPs	N	N	N
SGPs	Y	Y	Y
EP	Y	Y	Y

4.1 序列模式

序列模式(Sequential Patterns, SPs)挖掘用于发现序列数据集合中出现次数不低于用户定义阈值的项集序列,这些频繁的项集序列称为序列模式。最早的序列模式挖掘算法是 prefixSpan[57],这是基于投影数据库和前缀树结构的静态数据集合处理方法。它可以挖掘出全集序列模式,并且可以减少候选项集数量。

数据流中序列模式挖掘算法采用的主要数据结构之一是树结构。如 StrPMiner 算法[58]挖掘多数据流中最优的 SPs。该算法仅对多数据流扫描一次,压缩数据信息并使用 PBuilder 方法进行处理。StrPMiner 算法使用 T_0 树结构保存候选项集。BFSPMiner 算法[59]采用滑动窗口挖掘数据流中的 SPs。该算法使用倒置 T_0 树结构存储模式信息。该算法得到的模式结果集合更加准确,且可以降低运行时间。算法 MAHUSP[60]挖掘数据流中的高效用 SPs。该算法使用一个称为 MAS-tree 的树结构存储可能的模式信息。当发现一个可能的模式时则更新 MAS-tree。算法同时设计两个机制来更好地利用有效的存储空间。

4.2 高效用模式

高效用模式(High Utility Patterns, HUPs)挖掘用于发现事务数据集合中效用不低于用户定义阈值的项或项集。在真实应用中,效用可以指单位利润或者购买数量等。高效用模式挖掘方法通常可以分为一阶段和二阶段两类。

较早的 HUPs 挖掘算法是二阶段的。算法 TWU[61]提出二阶段方法挖掘数据集合中的高效用模式。该算法使用过估计概念,以满足高效用模式挖掘中的抗单调性。二阶段方法的主要问题之一是需要多次扫描数据集合产生大量候选集。数据流中挖掘 HUPs 的方法主要是一阶段的。如算法 SHU-Growth[62]使用滑动窗口技术挖掘 HUPs。为了产生更少的候选集合降低空间消耗,该算法使用基于树的结构存储 HUPs。LIHUP 算法[63]使用基于 list 的数据结构挖掘 HUPs。基于 list 的数据机构的重建依赖于一种最优排序策略,并且在重建的同时可以有效地更新效用信息。该算法也不产生候选集合,可以有效提高算法效率。Vert_top_k_DS[64]算法使用滑动窗口模型挖掘数据流中的 top-k HUPs。该算法只有一层结构,且不产生任何候选集合。算法中定义新的数据结构 iList 用于存储项的信息,且这个结构在合并和删除项时工作效率很高。HAUPM 算法[65]使用衰减窗口挖掘平均 HUPs。该算法使用衰减窗口关注新近事务中的 HUPs,使用时间因子发现显著的新近的模式信息。该算法使用新的衰减平均效用树(DAT)结构和事务效用 list(TUL)来提高挖掘效率。Choi 等[66]提出算法挖掘 Twitter 数据流中 HUPs,用于检测新兴话题。该方法使用滑动窗口技术计算每批推特中单词的效用,确定每批数据中的最小效用。该方法使用 TP 树结构从候选主题模式中提取实际主题模式。相比已有算法,该算法可以提高查全率。

4.3 其他模式

数据流中非频繁模式挖掘用于提取数据集中的稀有或者异常模式,是一种新的数据流异常检测方法[67]。Hemalatha 等[68]提出一种基于最小非频繁模式的数据流孤立点检测方

法 MIP-DS。该算法挖掘数据流中的最小非频繁模式。它首先挖掘滑动窗口内的最新模式,再生成所有的候选后筛选非频繁模式。通过实验验证,该方法可以提高孤立点检测的效率。Duong 等[69]提出算法挖掘信号数据流(不确定数据流)中的非频繁权重模式。这些模式满足基于距离的约束条件,用于分析两类信号数据:一是在相同环境下彼此相似的连续信号;二是在不同环境下或不同条件下的距离信号。该算法的优势在于在知识提取过程中使用约束,从而挖掘出更压缩的模式集合。

不确定数据流中发现的频繁模式通常是概率频繁模式,这些模式需要满足最小支持度和最小概率参数。Cuzzocrea 等[67]分析挖掘不确定数据集合中频繁模式的一些算法。这些算法通常使用 UF 树结构来存储模式,使用概率支持度的上界值来降低计算量。Li 等[16]提出算法 PFIMoS 使用滑动窗口模型挖掘不确定数据流中的概率频繁模式。该算法设计了一种压缩的数据结构-概率频繁项集树 PFIT,并采用自底向上的方式来存储数据信息,从而有效地减低搜索空间。PFIMoS 使用概率支持度估计方法来计算概率支持度的上下界值,这样可以降低计算时间。

5　进一步的研究方向

数据流挖掘中概念漂移问题的解决方法在过去十多年中得到了一定的发展,但是现有的方法仍然存在着不足之处,这为研究者提供了进一步的研究方向。

1)大规模数据流中模式挖掘结果集数量巨大,其中存在大量无用的模式。当事务长或最小支持度阈值低时,这个问题尤其严重。如何挖掘更加满足用户需求的有趣模式,且模式的数量尽可能地压缩;在模式挖掘过程中如何解决概念漂移带来的频繁与非频繁项的转变所带来的丢失可能模式的现象等问题需要进一步研究。

2)在某些应用中概念漂移是可以预期的,它沿时间轴或在不同对象中的模型化区域可能重新出现。例如粮食需求预测,可以用模糊周期性季节的影响为对象设定特定子群,或者传递不同的上下文之间的学习似乎是一个潜在的研究可预测概念漂移的路线。但是在很多的应用中,概念漂移的工作是在隐藏背景下发生,是不可观测到的。如何更好地分析这类概念漂移现象还需要进一步研究。

3)基于模式的分类方法研究。数据流中包含无限的数据,这些数据包含大量的冗余信息甚至是噪声,而模式发现可以去除数据中的冗余信息且不受噪声的影响。因此,挖掘有趣的、频繁的和有区分力的模式,可以用于有效的分类或聚类。基于模式的分类或聚类具有更高的准确性,并且可以很好地解决缺失值的问题。可以进一步对基于模式的数据分类方法进行研究。

4)由于云的出现,研究者提出了使用边缘计算和云计算来实现数据流处理,这是大数据发展带来的新计算方法[70-71]。如何更好地利用云技术来提高大数据挖掘效率也是需要继续研究的内容。

参考文献(References)

[1] BIFET A. Adaptive stream mining: pattern learning and mining from evolving data stream [C]// Proceedings of the 2010 Conference on Adaptive Stream Mining: Pattern Learning and Mining from Evolving Data Streams. Amsterdam, Netherlands: IOS Press, 2010: 1 - 212.

[2] MANKU G S, MOTWANI R. Approximate frequency counts over data streams [C]// Proceedings of the 28th International Conference on Very Large DataBases. San Francisco, CA: Morgan Kaufmann, 2002: 346 - 375.

[3] CHARIKAR M, CHEN K, FARACH-COLTON M. Finding frequent items in data streams [C]// Proceedings of the 2002 International Colloquium on Automata, Languages and Programming, LNCS 2380. 2002: 693 - 703.

[4] YANG T, ZHOU Y, JIN H, et al. Pyramid sketch: a sketch framework for frequency estimation of data streams [J]. Proceedings of the VLDB Endowment, 2017, 10(11): 1442 - 1453.

[5] LI H-F, SHAN M-K, LEE S-Y. DSM-FI: an efficient algorithm for mining frequent itemsets in data streams [J]. Knowledge Information Systems, 2008, 17(1): 79 - 97.

[6] 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,2008,19(19):2585 - 2596. (LI G H, CHEN H. Mining the frequent patterns in an arbitrary sliding window over online data streams [J]. Journal of Software, 2008, 19(10): 2585 - 2596.)

[7] MEMAR M, SADREDDINI M H, DEYPIR M, et al. A block-based approach for frequent itemset mining over data streams [C]// Proceedings of the 2011 8th International Conference on Fuzzy Systems and Knowledge Discovery. Piscataway, NJ: IEEE, 2011: 1647 - 1651.

[8] LI H-F, LEE S-Y. Mining frequent itemsets over data streams using efficient window sliding techniques [J]. Expert Systems with Applications, 2009, 36(2): 1466 - 1477.

[9] FAN G, YIN S. A frequent itemset mining algorithm based on matrix in sliding window over data steam [C]// ISDEA '13: Proceedings of the 2013 3rd International Conference on Intelligent System Design and Engineering Applications. Washington, DC: IEEE Computer Society, 2013: 66 - 69.

[10] CHANG J H, LEE W S. Finding recent frequent itemsets adaptively over online data streams [C]// KDD '03: Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. New York: ACM, 2003: 487 - 492.

[11] SHIN S J, LEE D S, LEE W S. CP-tree: an adaptive synopsis structure for compressing frequent itemsets over online data streams [J]. Information Sciences, 2014, 278: 559 - 576.

[12] SILVA A, ANTUNES C. Pushing constraints into data streams [C]// BigMine '13: Proceedings of the 2nd International Workshop on Big Data, Streams and Heterogeneous Source Mining: Algorithms, Systems, Programming Models and Applications. New York: ACM, 2013: 79 - 86.

[13] FARHAT A, GOUIDER M S, SAID L B. New algorithm for frequent itemsets mining from evidential data streams [J]. Procedia Computer Science, 2016, 96: 645 - 653.

[14] KUSUMAKUMARI V, SHERIGAR D, CHANDRAN R, et al. Frequent pattern mining on stream data using Hadoop CanTree-GTree [J]. Procedia Computer Science, 2017, 115: 266 - 273.

[15] BUSTIO-MARTÍNEZ L, CUMPLIDO R, HERNÁNDEZ-LEÓN R,

et al. On the design of hardware-software architectures for frequent itemsets mining on data streams [J]. Journal of Intelligent Information Systems, 2018, 50(3): 415 – 440.

[16] LI H, ZHANG N, ZHU J, et al. Probabilistic frequent itemset mining over uncertain data streams [J]. Expert Systems with Applications, 2018, 112: 274 – 287.

[17] CHI Y, WANG H, YU P S, et al. Catch the moment: maintaining closed frequent itemsets over a data stream sliding window [J]. Knowledge and Information Systems, 2006, 10(3): 265 – 294.

[18] YEN S-J, WU C-W, LEE Y-S, et al. A fast algorithm for mining frequent closed itemsets over stream sliding window [C]// Proceedings of the 2011 IEEE International Conference on Fuzzy Systems. Piscataway, NJ: IEEE, 2011: 996 – 1002.

[19] NORI F, DEYPIR M, SADREDDINI M H. A sliding window based algorithm for frequent closed itemset mining over data streams [J]. Journal of Systems and Software, 2013, 86(3): 615 – 623.

[20] DAI C, CHEN L. An algorithm for mining frequent closed itemsets in data stream [J]. Physics Procedia, 2012, 24(C): 1722 – 1728.

[21] HAN M, DING J, Li J. TDMCS: an efficient method for mining closed frequent patterns over data streams based on time decay model [J]. International Arab Journal of Information Technology, 2017, 14(6): 851 – 860.

[22] REDDY V S, RAO T V, GOVARDHAN A. Fast and Lossless Mining of Closed Frequent Itemsets (FLMCFI) over data streams [J]. Journal of Advanced Research in Dynamical and Control Systems, 2018, 10: 256 – 263.

[23] LEE G, YUN U, RYU K H. Sliding window based weighted maximal frequent pattern mining over data streams [J]. Expert Systems with Applications, 2014, 41(2): 694 – 708.

[24] KARIM M R, COCHEZ M, BEYAN O D, et al. Mining maximal frequent patterns in transactional databases and dynamic data streams: a Spark-based approach [J]. Information Sciences, 2018, 432: 278 – 300.

[25] CHANG Y I, LI C E, CHOU T J, et al. A weight-order-based lattice algorithm for mining maximal weighted frequent patterns over a data stream sliding window [C]// Proceedings of the 2018 IEEE International Conference on Applied System Innovation. Piscataway, NJ: IEEE, 2018: 961 – 964.

[26] HOMEM N, CARVALHO J P. Finding top-*k* elements in data streams [J]. Information Sciences, 2010, 180(24): 4958 – 4974.

[27] AMPHAWAN K, SOULAS J, LENCA P. Mining top-*k* regular episodes from sensor streams [J]. Procedia Computer Science, 2015, 69: 76 – 85.

[28] LI J, GONG S. Top-*k*-FCI: mining top-*k* frequent closed itemsets in data streams [J]. Journal of Computational Information Systems, 2011, 7(13): 4819 – 4826.

[29] TSAI P S M. Mining top-*k* frequent closed itemsets over data streams using the sliding window model [J]. Expert Systems with Applications, 2010, 37(10): 6968 – 6973.

[30] AMPHAWAN K, LENCA P. Mining top-*k* frequent-regular closed patterns [J]. Expert Systems with Applications, 2015, 42(21): 7882 – 7894.

[31] LIM Y, KANG U. Time-weighted counting for recently frequent pattern mining in data streams [J]. Knowledge and Information Systems, 2017, 53(2): 391 – 422.

[32] SONG C, LIU X, GE T. Top-*k* frequent items and item frequency tracking over sliding windows of any sizes [C]// Proceedings of the 2017 IEEE 33rd International Conference on Data Engineering. Washington, DC: IEEE Computer Society, 2017: 199 – 202.

[33] LEUNG C K-S, HAO B, JIANG F. Constrained frequent itemset mining from uncertain data streams [C]// Proceedings of the 2010 IEEE 26th International Conference on Data Engineering Workshops. Washington, DC: IEEE Computer Society, 2010: 120 – 127.

[34] HU W-C, WANG B-N, CHENG Z-L. Mining frequent closed patterns with item constraints in data streams [C]// Proceedings of the 2008 International Conference on Machine Learning and Cybernetics. Piscataway, NJ: IEEE, 2008: 274 – 280.

[35] CUZZOCREA A, LEUNG C K S, MacKINNON R K. Mining constrained frequent itemsets from distributed uncertain data [J]. Future Generation Computer Systems, 2014, 37: 117 – 126.

[36] KIRAN R U, KITSUREGAWA M, REDDY P K. Efficient discovery of periodic-frequent patterns in very large databases [J]. Journal of Systems and Software, 2016, 112: 110 – 121.

[37] CACCIERO L, CERQUITELLI T, CHIUSANO S, et al. Characterizing unpredictable patterns in wireless sensor network data [J]. Information Sciences, 2018, 467: 149 – 162.

[38] LI C-W, JEA K-F. An adaptive approximation method to discover frequent itemsets over sliding-window-based data streams [J]. Expert Systems with Applications, 2011, 38(10): 13386 – 13404.

[39] DEYPIR M, SADREDDINI M H. EclatDS: an efficient sliding window based frequent pattern mining method for data streams [J]. Intelligent Data Analysis, 2011, 15(4): 571 – 587.

[40] CHEN H, SHU L, XIA J, et al. Mining frequent patterns in a varying-size sliding window of online transactional data streams [J]. Information Sciences, 2012, 215: 15 – 36.

[41] LI C-W, JEA K-F. An approach of support approximation to discover frequent patterns from concept-drifting data streams based on concept learning [J]. Knowledge and Information Systems, 2014, 40(3): 639 – 671.

[42] 李海峰, 章宁, 朱建明, 等. 时间敏感数据流上的频繁项集挖掘算法[J]. 计算机学报, 2012, 35(11): 2283 – 2293. (LI H F, ZHANG N, ZHU J M, et al. Frequent itemset mining over time sensitive streams [J]. Chinese Journal of Computers, 2012, 35(11): 2283 – 2293.)

[43] DEYPIR M, SADREDDINI M H, HASHEMI S. Towards a variable size sliding window model for frequent itemset mining over data streams [J]. Computers and Industrial Engineering, 2012, 63(1): 161 – 172.

[44] 韩萌, 王志海, 丁剑. 一种频繁模式决策树处理可变数据流[J]. 计算机学报, 2016, 39(8): 1541 – 1554. (HAN M, WANG Z H, DING J. Efficient decision tree for evolving data streams based on frequent patterns [J]. Chinese Journal of Computers, 2016, 39(8): 1541 – 1554.)

[45] 吴枫, 仲妍, 吴泉源. 基于时间衰减模型的数据流频繁模式挖掘

[J]. 自动化学报, 2010, 36(5): 674 – 684. (WU F, ZHONG Y, WU Q Y. Ming frequent patterns over data stream under the time decaying model [J]. Acta Automatica Sinica, 2010, 36(5): 674 – 684.)

[46] 廖国琼, 吴凌琴, 万常选. 基于概率衰减窗口模型的不确定数据流频繁模式挖掘[J]. 计算机研究与发展, 2012, 49(5): 1105 – 1115. (LIAO G Q, WU L Q, WANG C X. Frequent patterns mining over uncertain data streams based on probability decay window model [J]. Journal of Computer Research and Development, 2012, 49(5): 1105 – 1115.)

[47] LI H, HO C, et al. A single-scan algorithm for mining sequential patterns from data streams [J]. International Journal of Innovative Computing, 2012, 8(3): 1799 – 1820.

[48] CHEN L, MEI Q. Mining frequent items in data stream using time fading model [J]. Information Sciences, 2014, 257: 54 – 69.

[49] 韩萌, 王志海, 原继东. 一种基于时间衰减模型的数据流闭合模式挖掘方法[J]. 计算机学报, 2015, 38(7): 1473 – 1482. (HAN M, WANG Z H, YUAN J D. Efficient method for mining closed frequent patterns from data streams based on time decay model [J]. Chinese Journal of Computers, 2015, 38(7): 1473 – 1482.)

[50] CHEN J, CHEN P. Sequential pattern mining for uncertain data streams using sequential sketch [J]. Journal of Networks, 2014, 9(2): 252 – 258.

[51] CHENG X, SU S, XU S, et al. Differentially private maximal frequent sequence mining [J]. Computers and Security, 2015, 55(C): 175 – 192.

[52] AHMED C F, TANBEER S K, JEONG B-S, et al. Single-pass incremental and interactive mining for weighted frequent patterns [J]. Expert Systems with Applications, 2012, 39(9): 7976 – 7994.

[53] AHMED C F, TANBEER S K, JEONG B-S, et al. Interactive mining of high utility patterns over data streams [J]. Expert Systems with Applications, 2012, 39(15): 11979 – 11991.

[54] BRAUN P, CAMERON J J, CUZZOCREA A, et al. Effectively and efficiently mining frequent patterns from dense graph streams on disk [J]. Procedia Computer Science, 2014, 35: 338 – 347.

[55] CUZZOCREA A, HAN Z, JIANG F, et al. Edge-based mining of frequent subgraphs from graph streams [J]. Procedia Computer Science, 2015, 60: 573 – 582.

[56] FOURNIER-VIGER P, LIN J C W, KIRAN R U, et al. A survey of sequential pattern mining [J]. Data Science and Pattern Recognition, 2017, 1(1): 54 – 77.

[57] PEI J, HAN J, MORTAZAVI-ASL B, et al. Mining sequential patterns by pattern-growth: the PrefixSpan approach [J]. IEEE Transactions on Knowledge and Data Engineering, 2004, 16(11): 1424 – 1440.

[58] TÖWS D, HASSANI M, BEECKS C, et al. Optimizing sequential pattern mining within multiple streams [EB/OL]. [2018-07-07]. http://cs.emis.de/LNI/Proceedings/Proceedings242/223.pdf.

[59] HASSANI M, TÖWS D, SEIDL T. Understanding the bigger picture: batch-free exploration of streaming sequential patterns with accurate prediction [C]// Proceedings of the 32nd Annual ACM Symposium on Applied Computing. New York: ACM, 2017: 866 – 869.

[60] ZIHAYAT M, CHEN Y, AN A. Memory-adaptive high utility sequential pattern mining over data streams [J]. Machine Learning, 2017, 106(6): 799 – 836.

[61] LIU Y, LIAO W, CHOUDHARY A. A two-phase algorithm for fast discovery of high utility itemsets [C]// Proceedings of the 2005 Pacific-Asia Conference on Knowledge Discovery and Data Mining, LNCS 3518. Berlin: Springer, 2005: 689 – 695.

[62] YUN U, KIM D, RYANG H, et al. Mining recent high average utility patterns based on sliding window from stream data [J]. Journal of Intelligent and Fuzzy Systems, 2016, 30(6): 3605 – 3617.

[63] YUN U, RYANG H, LEE G, et al. An efficient algorithm for mining high utility patterns from incremental databases with one database scan [J]. Knowledge-Based Systems, 2017, 124: 188 – 206.

[64] DAWAR S, SHARMA V, GOYAL V. Mining top-k high-utility itemsets from a data stream under sliding window model [J]. Applied Intelligence, 2017, 47(4): 1240 – 1255.

[65] YUN U, KIM D, YOON E, et al. Damped window based high average utility pattern mining over data streams [J]. Knowledge-Based Systems, 2018, 144: 188 – 205.

[66] CHOI H-J, PARK C H. Emerging topic detection in twitter stream based on high utility pattern mining [J]. Expert Systems with Applications, 2019, 115: 27 – 36.

[67] CUZZOCREA A, LEUNG C K. Computing theoretically-sound upper bounds to expected support for frequent pattern mining problems over uncertain big data [C]// Proceedings of the 2016 International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems, CCIS 611. Berlin: Springer, 2016: 379 – 392.

[68] HEMALATHA C S, VAIDEHI V, LAKSHMI R. Minimal infrequent pattern based approach for mining outliers in data streams [J]. Expert Systems with Applications, 2015, 42(4): 1998 – 2012.

[69] DUONG Q-H, RAMAMPIARO H, NØRVÅG K, et al. High utility drift detection in quantitative data streams [J]. Knowledge-Based Systems, 2018, 157: 34 – 51.

[70] de ASSUNÇÃO M D, da SILVA VEITH A, BUYYA R. Distributed data stream processing and edge computing: a survey on resource elasticity and future directions [J]. Journal of Network and Computer Applications, 2018, 103: 1 – 17.

[71] ur REHMAN M H, LIEW C S, WAH T Y, et al. Towards next-generation heterogeneous mobile data stream mining applications: opportunities, challenges, future research directions [J]. Journal of Network and Computer Applications, 2017, 79: 1 – 24.

This work is partially supported by the National Natural Science Foundation of China (61563001), the Natural Science Foundation of Ningxia (NZ17115).

HAN Meng, born in 1982, Ph. D., associate professor. Her research interests include big data classification, pattern mining.

DING Jian, born in 1977, associate professor. His research interests include big data classification, pattern mining.