计算机研究与发展
Journal of Computer Research and Development
DOI:10.7544/issn1000-1239.2015.20131883
52(12):2834-2843,2015

基于高斯函数的衰减因子设置方法研究

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技术学院 北京 100044)
[2](北方民族大学计算机科学与工程学院 银川 750021)
(compute2006_2@126.com)

A Method to Set Decay Factor Based on Gaussian Function

Han Meng[1,2], Wang Zhihai[1], and Yuan Jidong[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 Data stream is a continuous and time changed sequence of data elements, and contained information is different over time. In some data stream applications, the information embedded in the data arriving in the new recent time period is of particular value. Therefore, time decay model (TDM) is used for mining frequent patterns on data stream. Existing methods to design time decay factor have the characteristics of randomness, so the result set is unsteady. Or, the methods just consider 100% recall or 100% precision of the algorithm, while they ignore the corresponding high precision or recall. In order to balance high recall and high precision of the algorithm and ensure the stability of the result set, a novel way to set average decay factor is designed. To further increase the weights of the latest transactions and reduce the weights of historical transactions, another novel way to design decay factor based on Gaussian function is proposed. For comparing the pros and cons of different time factors, four time decay models are researched and designed. The algorithms based on these four models are designed to discover closed frequent patterns over data streams. The performance of the proposed methods to mine the frequent patterns on the high-density or low-density data streams is evaluated via experiments. Results show that using the average time decay factor balances the high recall and high precision of the algorithm. Compared with other ways, setting decay factor based on Gaussian function gets better performance than them.

Key words decay factor; time decay model; Gaussian function; recall; precision; frequent pattern mining; data streams mining

摘 要 数据流是随着时间顺序快速变化的和连续的,其包含的知识会随着时间的改变而不同.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.因此,会采用时间衰减模型来挖掘数据流中的频繁模式.已有的衰减因子设计方式通常具有随机性,使得到的结果集具有不稳定性;或仅考虑算法的高查全率或查准率,而忽略了算法对应的高查准率或查全率.为了平衡算法的高查全率和高查准率同时保证结果集的稳定性,设计了均值衰减因子设置方式.为了更进一步地增加最新事务的权重、减少历史事务的权重,设计了采用高斯函数设置高斯衰减因子的方式.为了比较不同衰减因子设计方式的优劣,研究并设计了4种方式的时间衰减模型,并采用这4种模型挖掘数据流闭合频繁模式.通过对高密度和低密度数据流分别进行频繁挖掘的实验结果分析可以得出,采用均值衰减因子设置方式可以平衡高查全率和高查准率;采用高斯衰减因子设置方式与其他方法相比,可以得到更优的算法性能.

收稿日期:2013-12-12;**修回日期**:2014-06-04
基金项目:国家自然科学基金项目(61563001);国家民委科研基金项目(14BFZ008);北京市自然科学基金项目(4142042);北方民族大学科研基金项目(2013QZP02)

关键词　衰减因子;时间衰减模型;高斯函数;查全率;查准率;频繁模式挖掘;数据流挖掘

中图法分类号　TP18

由于数据流的不断流动和连续性,数据流中包含的知识会随着时间的改变而发生变化.通常情况下,最近产生的事务所蕴含的知识要比历史事务的知识有价值得多[1-3].因此,在数据流的频繁模式挖掘过程中,希望强调最近事务产生的频繁模式,而减少历史事务产生频繁模式的可能性.通常的方式是采用滑动窗口(sliding window)[3-11]和时间衰减模型(time decay model, TDM)[3-4,10-14]来挖掘频繁模式.前者用于强调在窗口大小内挖掘频繁模式,后者则进一步强调在窗口内也要区分最新和历史事务.采用这种方式的研究重点在于确定几个参数之间的关系:窗口大小、最小支持度阈值、最大允许误差阈值、时间衰减因子.通常情况下,难点在于确定前3个参数后如何确定时间衰减因子的取值.

近年来,衰减模型中衰减因子的设置方式分为3类:1)在衰减因子取值范围(0,1)中随机定义一个值[11-14].这类衰减因子设置方式会由于衰减因子的不同而导致挖掘结果的不稳定性.2)基于对算法100%查全率(*Recall*)或100%查准率(*Precision*)的估计,设计衰减因子取值的上下界值;然后,取衰减因子的边界值[1-2]或在二者之间随机取值来设置衰减因子[2].这类做法的缺陷在于可以得到高的查全率或查准率,而对应的算法查准率或查全率较低;或者是由于衰减值的不确定性,使得到的算法的结果不稳定.3)采用了半生命周期(half-life)方式来挖掘不确定数据流[10].这种设置方式与设置衰减因子的下界方式(假定 *Recall* 为100%)的表现相近似.总之,已有方法确定衰减因子的缺陷在于强调了查全率与查准率一方的重要性,或者是随机的设定方式导致挖掘结果的不稳定性.

为了平衡数据流频繁模式挖掘算法的高查全率和高查准率,同时保证算法挖掘结果的稳定性,即当参数:窗口大小、最小支持度阈值和最大允许误差阈值确定时,可以得到稳定的结果集.本文研究并设计了2种新的高效衰减因子设置方式,主要的工作是:

1) 设计了一种均值衰减因子设置方式.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.

2) 为了更近一步地强调最新事务的权重、减弱历史事务的权重,研究并设计了采用高斯函数设置衰减因子的方式,并重点讨论了参数方差与窗口大小之间的关联.与针对查全率和查准率的因子设置方式相比,无论是挖掘高密度还是低密度的数据流,都可以得到更优的算法性能.

3) 研究了采用4种衰减因子设置方式设计的时间衰减模型:设定衰减因子 $f=f\_recall$,即假定 *Recall* 为100%时得到的边界值;设定 $f=f\_precision$,即假定 *Precision* 为100%时得到的边界值;设定 $f=f\_average$,在2个边界值之间取均值,设置均值衰减因子;设定 $f=f\_gauss$,采用高斯函数的形式设置高斯衰减因子.

4) 采用4种衰减模型设计数据流闭合频繁模式挖掘算法,分别挖掘高密度和低密度数据流,比较它们之间的优劣性.

1　预备知识

为了研究衰减因子设置方式的优劣,本文采用多种时间衰减模型挖掘数据流来进行比较.本节介绍相关的知识,主要包括:频繁模式挖掘和时间衰减模型,并给出了相关示例.

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.每个事务包含一个唯一的事务标识 *TID*,如表1中第1列所示.由于数据流是无限地不断流动的,频繁模式 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[1-2],记为 $freq(P,N)$.

Table 1　Transaction Data Stream

表1　事务数据流

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

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[1-2].设模式支持数在单位时间内的衰减比例为衰减因子 f,其中 $f\in(0,1)$.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 的支持数满足式(1)和式(2).其中如果事务 T_m 包含 P,则 $r=1$,否则 $r=0$.

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

$$r=\begin{cases}1, & P\subseteq T_m,\\ 0, & \text{otherwise}.\end{cases} \tag{2}$$

通过式(1)(2)可以分析出使用了衰减因子后模式的支持数小于非衰减时的支持数,且随着 f 的不同得到的值差异较大.如当 $f=0.8$ 时,采用式(3)可以得到模式的支持数小于 $1/(1-f)=1/(1-0.8)=5$;而当 $f=0.999$ 时,得到的支持数小于 1 000.

$$freq_d(P,T_m)=freq_d(P,T_{m-1})\times f+r=\sum_{i=1}^{\infty}r_i\times f^{m-i}=r_1\times f^{m-1}+r_2\times f^{m-2}+\cdots+r_m\leqslant f^{m-1}+f^{m-2}+\cdots+1\leqslant\frac{1}{1-f}. \tag{3}$$

因此,按照常规的最小支持数进行挖掘可能会丢失一些频繁模式.为了解决这个问题,使得采用时间衰减模型后正确率接近于常规模式挖掘,需要设置最大允许误差阈值 ε.即挖掘过程发现的是满足定义 1 的频繁模式和临界频繁模式,丢弃非频繁模式.

定义 1[1]. 令 N 为数据流中已有事务项的个数,$\theta(\theta\in(0,1])$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 $freq_d(P,N)\geqslant\theta N$,则 P 为频繁模式;如果项集 P 满足 $\theta N\geqslant freq_d(P,N)\geqslant\varepsilon N$,则 P 为临界频繁模式;否则,P 为非频繁模式.

例 1. 假定数据流如表 1 所示,设置最小支持度阈值 $\theta=0.5$,最大允许误差阈值 $\varepsilon=0.1\times\theta$,设定 $f=0.8$.使用式(1)生成衰减支持数,挖掘过程如图 1 所示.*ClosedTable* 表包括 3 个字段〈*Cid*,*CP*,*SCP*〉.其中 *Cid* 表示产生的闭合频繁模式的序号,是唯一的标识;*CP* 为闭合频繁模式;*SCP* 为模式的支持数.采用式(1)生成模式的频繁数过程是:每当

ClosedTable (Before T_1)

Cid	*CP*	*SCP*
0	{0}	0

FrequentTable (Before T_1)

T	$freq_d(CP)$
Null	Null

(a) Before T_1

ClosedTable (After T_1)

Cid	*CP*	*SCP*
0	{0}	0
1	{1 3 4}	1

FrequentTable (After T_1)

T	$freq_d(CP)$
T_1	$freq_d(\{1\,3\,4\})=r^{(1)}=1$

(b) Process T_1

ClosedTable (After T_2)

Cid	*CP*	*SCP*
0	{0}	0
1	{1 3 4}	0.8
2	{2 3 5}	1
3	{3}	1.8

FrequentTable (After T_2)

T	$freq_d(CP)$
T_2	$freq_d(\{1\,3\,4\})=1\times f=0.8$ $freq_d(\{2\,3\,5\})=r^{(2)}=1$ $freq_d(\{3\})=freq_d(\{1\,3\,4\})+r^{(2)}=1.8$

(c) Process T_2

ClosedTable (After T_3)

Cid	*CP*	*SCP*
0	{0}	0
1	{1 3 4}	0.64
2	{2 3 5}	1.8
3	{3}	2.44
4	{1 2 3 5}	1

FrequentTable (After T_3)

T	$freq_d(CP)$
T_3	$freq_d(\{1\,3\,4\})=0.8\times f=0.64$ $freq_d(\{2\,3\,5\})=1\times f+r^{(3)}=1.8$ $freq_d(\{3\})=1.8\times f+r^{(3)}=2.44$ $freq_d(\{1\,2\,3\,5\})=r^{(3)}=1$

(d) Process T_3

ClosedTable (After T_4)

Cid	*CP*	*SCP*
0	{0}	0
1	{1 3 4}	0.512
2	{2 3 5}	2.44
3	{3}	2.952
4	{1 2 3 5}	0.8
5	{2 3 4 5}	1
6	{3 4}	1.512

FrequentTable (After T_4)

T	$freq_d(CP)$
T_4	$freq_d(\{1\,3\,4\})=0.64\times f=0.512$ $freq_d(\{2\,3\,5\})=1.8\times f+r^{(4)}=2.44$ $freq_d(\{3\})=2.44\times f+r^{(4)}=2.952$ $freq_d(\{1\,2\,3\,5\})=1\times f=0.8$ $freq_d(\{2\,3\,4\,5\})=r^{(4)}=1$ $freq_d(\{3\,4\})=freq_d(\{1\,3\,4\})+r^{(4)}=1.512$

(e) Process T_4

Fig. 1 Mining frequent patterns on data stream based on TDM.

图 1 基于 TDM 的数据流频繁模式发现示意图

一个事务到达时,则更新整个 *ClosedTable* 表,具体的计数过程如 *FrequentTable* 所示.例如当事务 T_3 到达时,工作过程是先更新表中已有的模式:

$$freq_d(\{1\ 3\ 4\}) = 0.8 \times f + r^{(3)} = 0.64,$$

其中,$r^{(3)}=0$;

$$freq_d(\{2\ 3\ 5\}) = 1 \times f + r^{(3)} = 1.8,$$

$$freq_d(\{3\}) = 1.8 \times f + r^{(3)} = 2.44,$$

其中,$r^{(3)}=1$;

添加新的模式:

$$freq_d(\{1\ 2\ 3\ 5\}) = r^{(3)} = 1,$$

如此反复,生成全部模式的支持数.

给定最小支持度后,当事务 T_4 到达时需要满足的最小支持数为 $4\times0.5\times0.1=0.2$,产生的闭合模式为 6 个,如图 1(e)所示.假定不使用最大允许误差阈值,则需要满足的最小支持数为 $4\times0.5=2$.从图 1 可以看出,仅能发现 2 个闭合模式:{2 3 5}和{3},丢失了 4 个可能的频繁模式.因此使用衰减模型发现数据流中的频繁模式时需要满足 $\theta-\varepsilon$ 框架.

2　时间衰减因子的研究

给定参数滑动窗口大小 N、最小支持度阈值 θ 和最大允许误差阈值 ε 之后,如何确定衰减因子 f 的值?已有的方式通常采用随机设置或者采用 100%查全率和 100%查准率来确定.例如 MSW 算法等[1-2]设定衰减因子方式为:假定 $Recall=100\%$ 时,f 应满足式(4);假定 $Precision=100\%$ 时,f 满足式(5).

$$f \geqslant \sqrt[(2N-\theta N-1)]{[(\theta-\varepsilon)/\theta]^2},\ Recall = 100\%. \quad (4)$$

$$f < \frac{(\theta-\varepsilon)N-1}{(\theta-\varepsilon)N},\ Precision = 100\%. \quad (5)$$

通常选择衰减因子的方式是设置其为上下限值,如式(6)中 f_1 和 f_2 所示;或者选择上下限之间的随机值.选择上下界时会得到较好的 *Recall* 或者 *Precision*,而对应的 *Precision* 或 *Recall* 则较低.而随机选定衰减因子得到结果的优劣程度不定.

$$f_1 = f\_recall = \sqrt[(2N-\theta N-1)]{[(\theta-\varepsilon)/\theta]^2},$$
$$f_2 = f\_precision = \frac{(\theta-\varepsilon)N-1}{(\theta-\varepsilon)N}. \quad (6)$$

为了更好地研究衰减因子的取值方式,对式(1)进行修改.引入衰减因子函数 $factor(f,x)$,如式(7)所示.其中当事务 T_i 到达时,参数 $x=m-i,i=1,2,\cdots,n,\cdots,m$,参数 f 是衰减因子,T_m 是最新到达的事务.函数 $factor(f,x)$可以看作是每个 r_i 的权值,如式(8)所示,是参数 f 的 x 幂.令集合 D 表示堆积的衰减因子,则当 T_m 到达时其取值为式(9)所示.

$$freq_d(P,T_m) = freq_d(P,T_{m-1}) \times f + r = \sum_{i=1}^{\infty} r_i \times f^{m-i} = \sum_{i=1}^{\infty} r_i \times factor(f,m-i), \quad (7)$$

$$factor(f,x) = f^x, \quad (8)$$

$$D_m = \{factor(f,m-1),\cdots,factor(f,1),factor(f,0)\} = \{f^{m-1},f^{m-2},\cdots,f^1,1\}. \quad (9)$$

由于 *Recall* 和 *Precision* 不可能同时为 100%,所以选择 f 时应考虑对二者的平衡.本文首先提出一种设定衰减因子为上下界的平均值方法,如式(10)中 f_3 所示.它是 *f_recall* 和 *f_precision* 的平均值,从理论上而言可以平衡算法的查全率和查准率,同时也避免了衰减因子设置的随机性.

例 2. 假设 $N=10000$,设定 θ,ε 如表 2 所示.其中 *f_recall* 为假定 $Recall=100\%$ 时得到的下界阈值,*f_precision* 为假定 $Precision=100\%$ 时得到的上界阈值.在得到 *f_recall* 和 *f_precision* 后,如何选择 f 的值?可以有 3 个策略,如式(6)和式(10)所示.以 $\theta=0.025$,$\varepsilon=0.05\times\theta$ 为例,可以选定:

$$f = \begin{cases} f_1 = f\_recall = 0.999995, \\ f_2 = f\_precision = 0.995789, \\ f_3 = f\_average = 0.997892. \end{cases}$$

Table 2　Set Decay Factors

表 2　设置衰减因子

θ	ε	*f_recall*(*Recall*=100%)	*f_precision*(*Precision*=100%)	*f_average*
0.05	$0.05\times\theta$	0.999 995	0.997 895	0.998 945
0.05	$0.1\times\theta$	0.999 989	0.997 778	0.998 884
0.05	$0.5\times\theta$	0.999 929	0.996	0.997 965
0.025	$0.05\times\theta$	0.999 995	0.995 789	0.997 892
0.025	$0.1\times\theta$	0.999 989	0.995 556	0.997 773

从实验分析发现,设计 f 的值为均值衰减因子 $f\_average$ 时可以得到更加平衡的或更优的算法性能.

以上 3 种衰减因子的设置方式得到的函数都是 f 的 x 幂形式,得到的累积因子的取值集合 D 的变化趋势如图 2 所示,它们的变化比较平缓. 为了加重新近事务的权重、降低历史事务的权重,本文提出第 2 种设计 f 的方式:即采用高斯函数形式,如式(11)所示. 其中为了使 f_4 满足衰减因子的特性:$f\in(0,1)$,增加常数值 A. 为了保证当最新事务 T_m 到达时 f 的值为 1,且与滑动窗口大小 N 产生关联,所以设定参数:均值为 0,方差的平方值为 $B\times N$,B 为常量值.

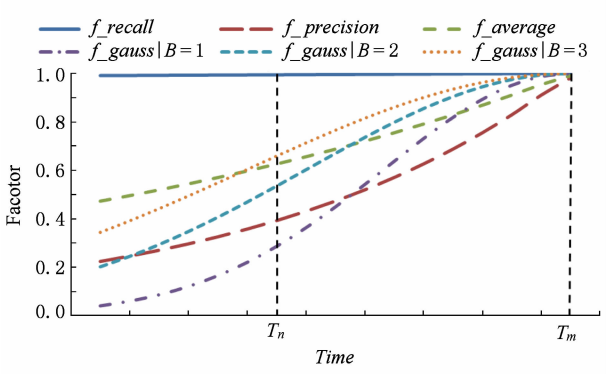


Fig. 2 Comparison of D with four decay factors.

图 2 不同衰减因子的累计衰减值变化趋势的比较

重新整理衰减因子函数,可以得到式(12). 函数 $factor(f,x)$ 有 4 种取值,1)当 f 取值为 f_recall,$f\_precision$ 和 $f\_average$ 时,$factor(f,x)$ 为 f 的幂形式;2)当 $f=f\_gauss$ 时,$factor(f,x)$ 为高斯函数形式. 画出 4 种衰减因子的取值曲线,如图 2 所示. 从图 2 可以看出,通常情况下设置 $f=f\_recall$ 时,随着时间的改变历史事务的权重衰减得微小. 设置 $f=f\_precision$ 时,随着时间的改变历史事务的权重衰减的程度较大;而 $f=f\_average$ 在二者之间. 当设置 $f=f\_gauss$,$B=1$ 时,得到的最近事务的衰减程度低于 $f\_precision$ 和 $f\_average$,而得到历史事务的衰减程度高于其他三者. 换言之,采用高斯函数设置衰减因子,会更加强调新近事务的重要性,降低历史事务的重要性.

$$f_3 = f\_average = (f\_recall + f\_precision)/2 = \frac{\sqrt[(2N-\theta N-1)]{[(\theta-\varepsilon)/\theta]^2} + \dfrac{(\theta-\varepsilon)N-1}{(\theta-\varepsilon)N}}{2}. \tag{10}$$

$$f_4 = f\_gauss = A\frac{1}{\sigma\sqrt{2\pi}}e^{-\frac{(x)^2}{2\sigma^2}}. \tag{11}$$

$$factor(f,x) = \begin{cases} f\_recall^x, & f = f\_recall; \\ f\_precision^x, & f = f\_precision; \\ f\_average^x, & f = f\_average; \\ \dfrac{A}{\sqrt{2\pi BN}}e^{-\frac{x^2}{2BN}}, & f = f\_gauss. \end{cases} \tag{12}$$

例 3. 假设 $N=10^4$,设定最小支持度 $\theta=0.05$,最大允许误差 $\varepsilon=0.1\times\theta$,高斯衰减参数 B 分别取值为 1,2,3. 则可以得到:

$$f = \begin{cases} f\_recall = 0.999989, \\ f\_precision = 0.997778, \\ f\_average = 0.998884, \\ f\_gauss \mid B=1 = 0.9995, \\ f\_gauss \mid B=2 = 0.99975, \\ f\_gauss \mid B=3 = 0.99983. \end{cases}$$

当有 4 个事务到达时,得到事务的衰减权值 $factor(f,x)$ 满足式(12),累积因子集合 D 如表 3 所示. 当 T_{50} 到达时可以看出,设定 $f=f\_recall$ 得到的函数值为 0.999 45,几乎没有变化;而 $f=f\_gauss|B=1$ 时得到的函数值是 0.286 506,远低于其他三者. 设置不同的 B 相比,随着 B 值的增大,衰减的幅度变缓.

Table 3 Values of D with Different Decay Factors

表 3 设置不同衰减因子得到的堆积衰减值 D

T_m	f_recall	$f\_precision$	$f\_average$	$f\_gauss\|B=1$	$f\_gauss\|B=2$	$f\_gauss\|B=3$
T_1	0.999 989	0.997 778	0.998 884	0.999 5	0.999 750 031	0.999 833 347
T_2	0.999 978	0.995 561	0.997 769	0.998 002	0.999 000 5	0.999 333 556
T_3	0.999 967	0.993 349	0.996 656	0.995 51	0.997 752 529	0.998 501 124
T_4	0.999 956	0.991 142	0.995 543	0.992 032	0.996 007 989	0.997 336 886
⋮	⋮	⋮	⋮	⋮	⋮	⋮
T_{50}	0.999 45	0.894 739	0.945 699	0.286 505	0.535 261 429	0.659 240 63

使用堆积衰减值 D 生成频繁模式的过程如图 3 所示,其中每组数据表示的是〈{CP} (SCP)〉. 假定数据流如表 1 所示,添加新事务 T_5={1 3 4 7}. 与例 1 中生成模式的支持数过程不同,不需要每次更新整个 *ClosedTable* 表,而是更新与 T_m 相关的模式. 例如,在事务 T_1 到达后生成的模式〈{1 3 4}(1)〉,仅需在事务 T_5 到达后进行支持数更新. 采用的是堆积因子 D 中的 f^4 作为权值进行更新:〈{1 3 4}($1\times f^4+1$)〉. 这种方式与例 1 相比可以降低算法的时间复杂度.

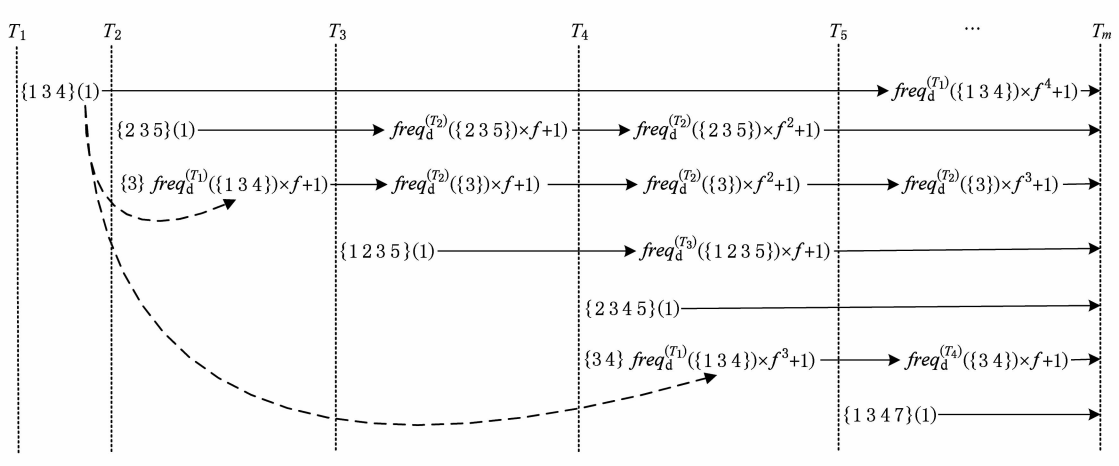


Fig. 3　Discover frequent patterns with D.

图 3　使用堆积衰减值 D 生成频繁模式示意图

3　实验分析

实验运行环境的 CPU 为 2.1 GHz,内存为 2 GB,操作系统为 Win7,所有的实验均采用 Java 实现. 为了比较多种衰减因子设置方法的优劣,采用了 2 类真实数据流:1)高密度数据流 msnbc,挖掘出的模式具有较高的频度. 它来自 UCI[15],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. 2)数据流 kosarak,它是低密度数据集合,从中发现的模式频度较低. 它来自 SPMF[16],包含了 990 000 条事务序列,是匈牙利门户新闻网站的点击流序列.

3.1　参数设置

实验中设置衰减因子的取值为 4 类:1)取下界值. 假定 $Recall$=100%,标记为 $f=f\_recall$;2)取上界值. 假定 $Precision$=100%,标记为 $f=f\_precision$;3)为了平衡算法的 $Recall$ 和 $Precision$,设置 f 为均值衰减因子,标记为 $f=f\_average$;4)为了强调最新事务的重要性、降低历史事务的权重,设置为高斯衰减因子,其中设置均值 $\mu=0$,方差 $\delta^2=B\times N$,标记为 $f=f\_gauss$.

实验中设置最小支持度 θ 为 0.06 和 0.000 5,前者用于发现高密度数据流中的频繁模式,后者用于发现低密度数据流中的频繁模式. 设置最大允许误差率 $\varepsilon=0.1\times\theta$,窗口大小 N 取值为 1 000,2 000,3 000 和 4 000.

3.2　实验分析

首先对高斯函数作为衰减因子时参数 δ^2 进行分析,主要比较参数 B 对算法性能的影响. 通过对高密度数据流 msnbc 进行处理,可以得到的实验结果如图 4 所示. 其中分别设置 B 取值为 1,2,3,即讨论参数 $\delta^2=N,2N,3N$ 的算法性能比较. 从图 4 可以分析出,当设置 $B=1$ 时可以得到最优的查准率,但是相对而言查全率最差;当 $B=3$ 时可以得到最优的查全率,而对应的查准率稍差. 通过对不同窗口大小条件下算法的整体性能分析,设置 $B=3$ 时,得到的算法 $Recall$ 和 $Precision$ 是最均衡且算法性能最优,其次是 $B=2$,而 $B=1$ 得到算法的 $Recall$ 和 $Precision$ 之间的差距最大. 通过对低密度数据流 kosarak 进行处理,也可以得到相似的结论.

接着比较不同的衰减因子设置方式的优劣. 对高密度数据流 msnbc 进行分析. 通过实验 1 的分析,设置参数 $B=3$. 设置 4 类衰减因子,在不同的窗口大小条件得到的算法查全率和查准率如图 5 和图 6 所示. 图 5 是对同一窗口时,设置不同衰减因子得到的 $Recall$ 或 $Precision$ 进行比较. 从图 5 中可以看出:

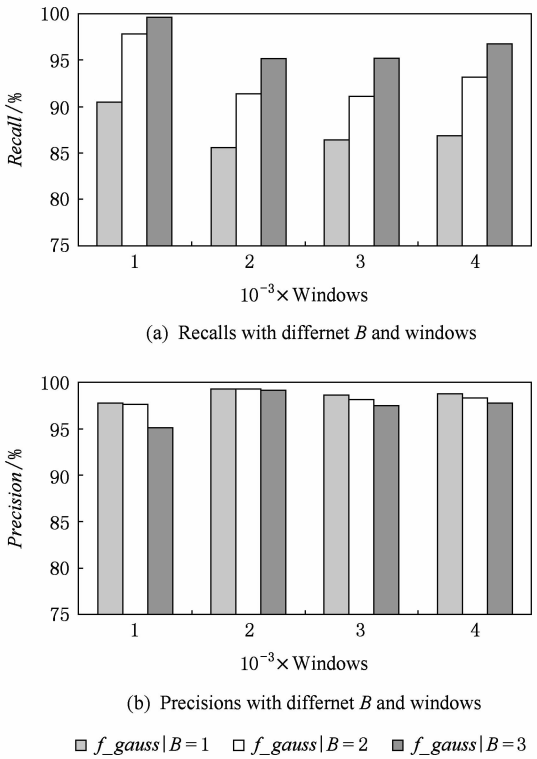


Fig. 4 Performances of algorithms with different B and windows on msnbc.

图 4 不同 B 值和窗口大小对应的算法性能比较

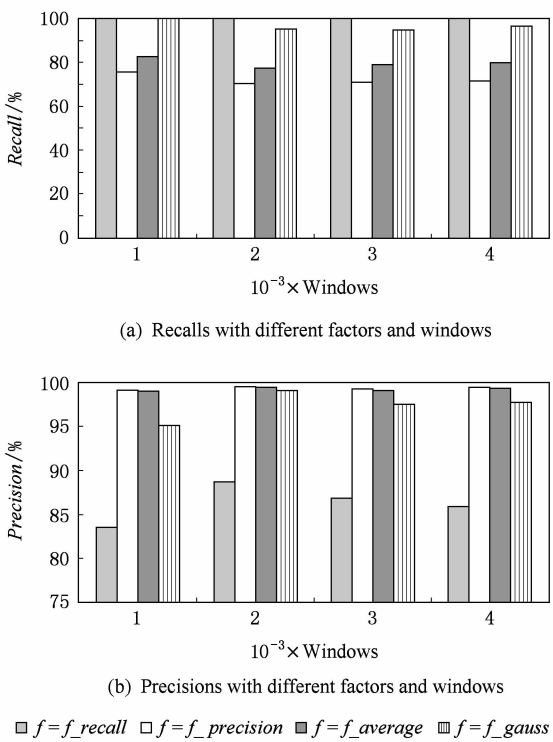


Fig. 5 Performances of algorithms with different decay factors and windows on msnbc.

图 5 比较不同条件下算法在 msnbc 上的性能

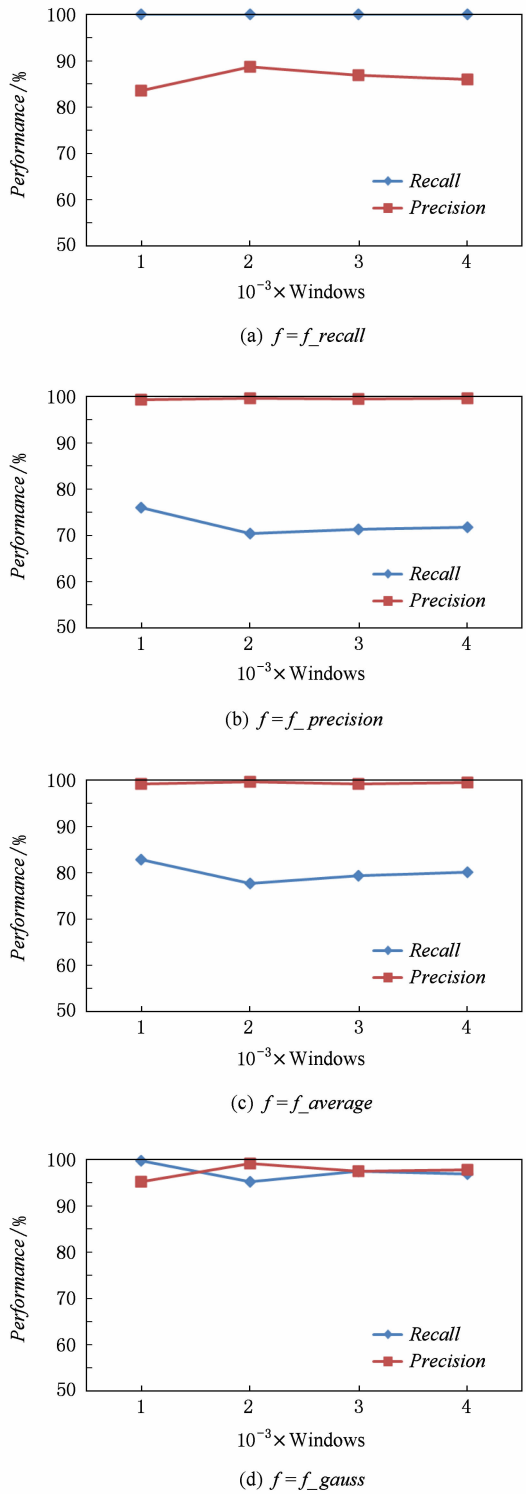


Fig. 6 *Recall* and *Precision* of algorithm with different decay factors on msnbc.

图 6 在 msnbc 上比较算法性能

1) 设置 $f=f\_recall$ 时,可以得到几乎 100%的 *Recall*,但是得到的 *Precision* 是 4 种方式中最低的;

2) 设置 $f=f\_precision$ 时,可以得到平均最高的 *Precision* 值,但是得到的 *Recall* 值是最低的;

3) 设置 $f=f\_average$ 时,得到的 *Recall* 在 $f=f\_recall$ 与 $f\_precision$ 之间,而得到的 *Precision* 高于 $f=f\_recall$ 和 $f\_precision$,所以验证了设置 $f=f\_average$ 与设置 $f=f\_recall$ 或 $f\_precision$ 相比可以平衡算法的高 *Recall* 和高 *Precision*;

4) 通过设置 $f=f\_gauss$,与前三者相比得到的算法 *Recall* 和 *Precision* 的平均值是最优的;

5) 设置不同衰减因子时,得到的查全率和查准率的优劣关系受到窗口变化的影响很小.

图 6 是对某一衰减因子值对应的 *Recall* 和 *Precision* 进行比较.从平衡算法的查全率和查准率的角度比较,图 6 可以看出 *Recall* 与 *Precision* 差距最小的是图 6(d),即 $f=f\_gauss$ 时,其次是 $f=f\_recall$ 时差距较小,差距最大的是 $f=f\_precision$. 也就是说,设置 f_guass 可以得到最优的算法性能,设置 $f\_precision$ 相比而言会得到最差的算法性能.

从图 5 和图 6 分析算法的整体表现可知,采用不同的衰减因子设置方式得到的性能比较结果由优到差的顺序是:$f\_gauss\rightarrow f\_recall\rightarrow f\_average\rightarrow f\_precision$.

接着对低密度数据流 kosarak 进行处理,算法的性能表现如图 7 和图 8 所示.可以看出得到的结论与图 5 和图 6 的实验结论比较类似.

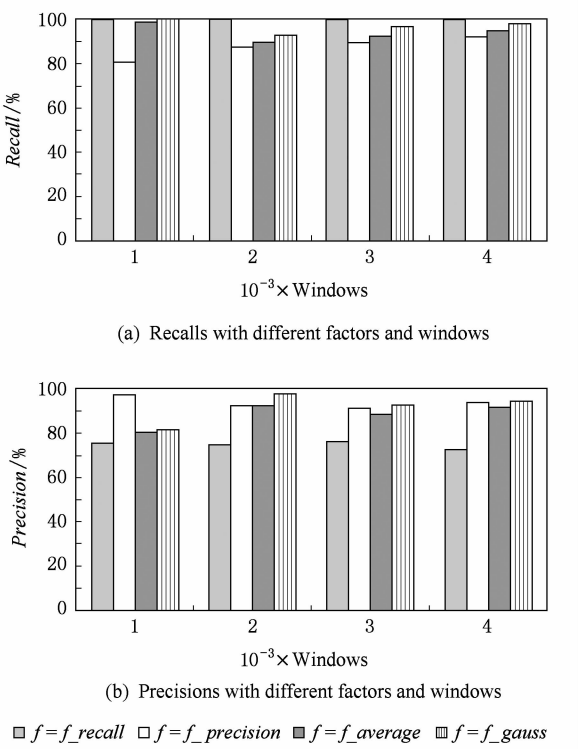


Fig. 7　Precisions of algorithm with different factors and windows on kosarak.

图 7　比较不同条件下算法在 kosarak 上的性能

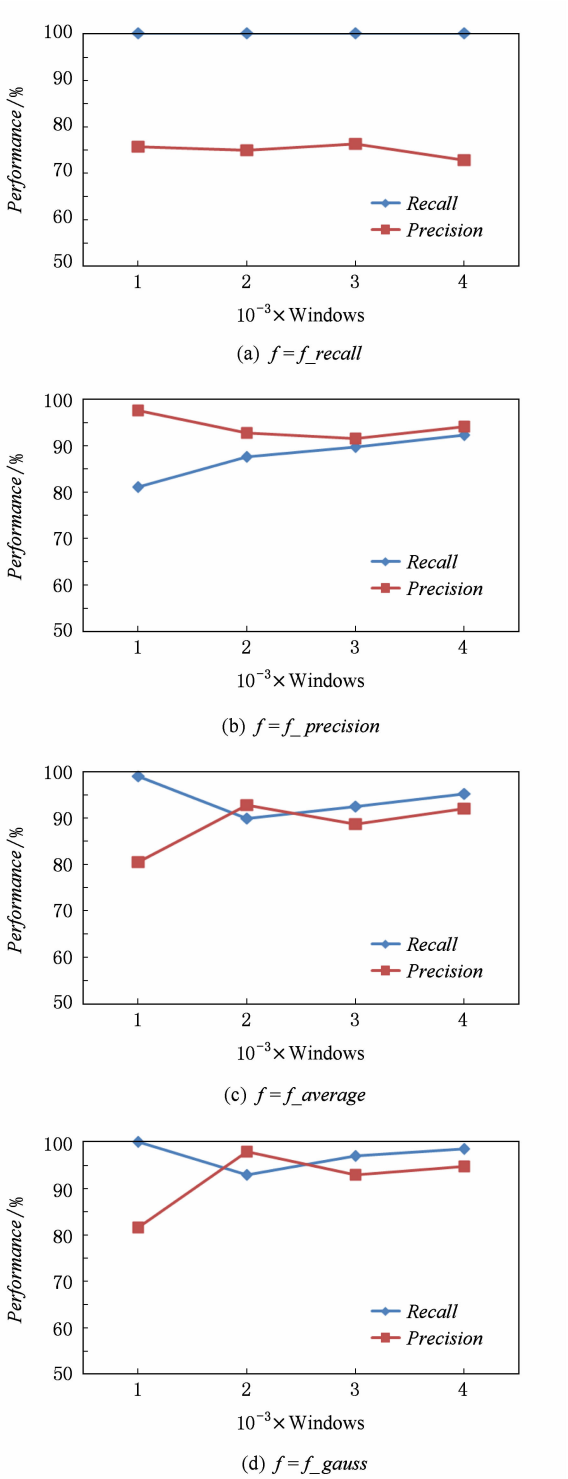


Fig. 8　*Recall* and *Precision* of algorithm with different decay factors on kosarak.

图 8　在 kosarak 上比较算法性能

1) 设置 $f=f\_recall$ 时,可以得到几乎 100% 的 *Recall*,但是 *Precision* 是四者中最低的.

2) 设置 $f=f\_precision$ 时,可以得到较高的

Precision,但 *Recall* 是三者中最低的.

3) 而设置 $f=f\_average$ 与 f_recall 和 $f\_precison$ 相比,得到的 *Recall* 和 *Precison* 均在二者之间.

4) 通过设置 $f=f\_gauss$,与前三者相比得到的 *Recall* 与 *Precision* 的平均值是最高的.

5) 从平衡算法的查全率和查准率的角度分析,设置 f_guass 得到二者的差距最小. 如图 8(d)所示;其次是 $f\_average$ 和 $f\_precision$,它们的表现差别不大,如图 8(b)和图 8(c)所示;二者差距最大的是 f_recall,如图 8(a)所示.

总之,性能算法比较而言由优到劣的顺序是:$f\_gauss \rightarrow f\_average \rightarrow f\_precision \rightarrow f\_recall$.

综合上述实验,可以得出结论:

1) 窗口大小的改变对不同衰减因子得到的算法性能优劣影响较小;

2) 关注结果集的查全率时,可以设置 $f=f\_recall$,但结果集中会包含较多的非频繁模式,查准率较低;

3) 关注结果集的查准率时,可以设置 $f=f\_precision$,但是结果集中会丢失可能的频繁模式,查全率较差;

4) 采用本文提出的 $f=f\_average$ 与设置 f 为上下界值相比,确实可以得到平衡的查全率和查准率,在高密度数据流处理中算法的性能表现在设置 f 为上下界值之间,而对低密度数据流进行处理时它的表现优于二者;

5) 高斯函数作为衰减因子值时,通过实验验证:设置参数均值 $\mu=0$,方差 $\delta^2=B\times N$ 可以得到好的算法性能,因此这种参数设置方式具有一定的合理性;

6) 实验表明,无论是高密度还是低密度数据流处理时,采用本文设计的高斯函数作为衰减因子与已有方式相比皆可得到更优的算法性能.

4 总 结

时间衰减模型的使用效率主要取决于衰减因子的设置方式. 已有的衰减因子设置方式包括随机设定方式,即从(0,1)范围内随机确定衰减值,通常是接近 1 的值. 另一类常用的设计方式是假定算法具有 100%查全率或 100%查准率时得到的衰减因子边界值. 本文重点研究了已有的衰减因子 f 的设计方法,并提出 2 种新的衰减因子设计方式:1)基于 100%查全率与 100%查准率的均值衰减因子设置方式;2)基于高斯函数的高斯衰减因子设置方式,文中重点讨论了高斯函数中相关参数的设置方式. 与查全率和查准率设置方式相比,高斯衰减因子进一步强调最近事务重要性,忽略历史事务的重要性. 同时,本文研究了基于 4 种衰减因子的时间衰减模型,并采用这 4 种模型发现高密度和低密度数据流中的频繁模式. 通过对实验结果的分析,针对不同特征的数据流,本文提出的 2 种方式皆优于已有方式.

参 考 文 献

[1] Li Guohui, Chen Hui. Mining the frequent patterns in an arbitrary sliding window over online data streams [J]. 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[J]. 软件学报, 2008, 19(19): 2585-2596)

[2] Chen Hui, Shu L, Xia Jiali, et al. Mining frequent patterns in a varying-size sliding window of online transactional data streams [J]. Information Sciences, 2012, 215: 15-36

[3] Li Haifeng, Zhang Ning, Zhu Jianming, et al. Frequent itemset mining over time-sensitive streams [J]. Chinese Journal of Computers, 2012, 35(11): 2283-2293 (in Chinese)
(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)

[4] Chi Yun, Wang Haixun, 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

[5] Yen S J, Lee Y, Wu Chengwei, et al. An efficient algorithm for maintaining frequent closed itemsets over data stream [G] //Next-Generation Applied Intelligence. Berlin: Springer, 2009: 767-776

[6] Tang Keming, Dai Caiyan, Chen Ling. A novel strategy for mining frequent closed itemsets in data streams [J]. Journal of Computers, 2012, 7(7): 1564-1572

[7] Noria 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

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

[9] Yen S, Wu Chengwei, Lee Y, et al. A fast algorithm for mining frequent closed itemsets over stream sliding window [C] //Proc of 2011 IEEE Int Conf on Fuzzy Systems. Piscataway, NJ: IEEE, 2011: 996-1002

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

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

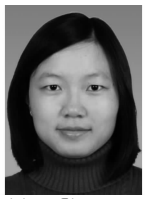
[12] Li Huafu, Ho Chinchuan, Chen H, et al. A single-scan algorithm for mining sequential patterns from data streams [J]. International Journal of Innovative Computing, 2012, 8(3): 1799-1820

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

[14] Liao Guoqiong, Wu Lingqin, Wan Changxuan. 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 (in Chinese)
(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)

[15] Frank A, Asuncion A. UCI machine learning repository [DB/OL]. Irvine, CA: School of Information and Computer Science, University of California, 2010 [2013-04-11]. http://archive.ics.uci.edu/ml

[16] Fournier-Viger P, Gomariz A, Gueniche T, et al. SPMF: A Java open-source pattern mining library [J]. Journal of Machine Learning Research, 2014, 15: 3389-3393



Han Meng, born in 1982. Received her Master degree in computer science from Beijing Jiaotong University. Currently associate professor at Beifang University of Nationalities in China and PhD candidate at Beijing Jiaotong Univeristy. Her main research interests include data mining and machine learning.

Wang Zhihai, born in 1963. Received his PhD in computer application from Hefei University of Technology. Professor in Beijing Jiaotong University. His main research interests include data mining and artificial intelligence.

Yuan Jidong, born in 1989. PhD candidate in Beijing Jiaotong University. His main research interests include machine learning and data mining.