文章编号：2096-1472(2016)-10-15-03

基于滑动窗口模型的数据流加权频繁模式挖掘算法

马连灯，王占刚
（天津工业大学计算机科学与软件学院，天津 300387）

摘 要：加权频繁模式挖掘比传统的频繁模式挖掘更加的具有实际意义，针对数据流中的数据只能扫描有限次的性质，提出了基于滑动窗口模型的数据流加权频繁模式挖掘方法WFP-SW，该算法中数据存储采用的是矩阵数据结构，通过矩阵之间的相关操作来产生加权频繁模式。实验结果显示，该算法在产生加权频繁模式的时候不产生冗余模式，比传统的频繁模式挖掘算法有更好的效率。

关键词：数据流；滑动窗口；加权频繁模式；矩阵
中图分类号：TP311.13 **文献标识码**：A

The Data Stream Weighted Frequent Pattern Mining Algorithm Based on the Sliding Window Model

MA Liandeng,WANG Zhangang
(*School of Computer Science and Software Engineering*,*Tianjin Polytechnic University*,*Tianjin* 300387,*China*)

Abstract: The weighted frequent pattern mining algorithm has more practical implication than traditional frequent pattern mining ones.According to the nature of data stream which can only scans database several times,this paper proposes a weighted frequent pattern algorithm based on the sliding window model(WFP-SW).The algorithm applies matrix data structure to store data,and then produces the weighted frequent pattern through some relative operation between matrixes. Experiment results show that the algorithm can avoid redundant patterns in the process of producing the weighted frequent pattern,and this algorithm is more efficient than the traditional frequent pattern mining algorithms.

Keywords:data stream;sliding window;the weighted frequent pattern;matrix

1 引言(Introduction)

加权频繁模式与传统的频繁模式挖掘是不同的[1-3]，它不仅取决于项集出现的次数，而且要考虑到数据库中项集重要性。在很多实际的应用中[4,5]，不同的数据项的重要程度是不同的。例如，在零售市场分析的时候，虽然贵重的商品没有在事务数据库中出现非常多的次数，但是它们却贡献了很大一部分的收入。所以，加权频繁模式挖掘比传统的频繁模式挖掘更能在现实世界中发挥更实际的作用。

本文提出了基于滑动窗口模型的数据流加权频繁模式挖掘方法WFP-SW，该算法中数据存储采用的是矩阵数据结构，通过矩阵之间的相关操作得到加权频繁模式。实验结果显示，该算法在产生加权频繁模式的时候不产生冗余模式，比传统的加权频繁模式挖掘算法有更好的效率。

2 基本概念(The basic concept)

定义1：设$I=\{I_1,I_2,\cdots,I_m\}$是项的集合，数据流DS是一个以一定速度连续到达的数据项序列$\{T_1,T_2,\cdots,T_n\}$，其中T_i表示第i个事务$(1\le i\le v)$，对于任意T_i都有$T_i\subseteq I$。每个项目中都有一个代表此项的重要性的非负实数的权值$w(j)$，$0\le w(j)\le 1$。

定义2：由数据项组成的集合定义为项集，其中，含有k个项的集合定义为k–项集。

定义3：项集的权值$ws(X)$是数据流中含有该项目的事务项集权值的汇总[6]。

定义4：设加权最小支持度为min_sup，如果项集X是频繁项集，则加权支持度大于或等于min_sup，即$FWI=\{X\subseteq I|ws(X)\geq min_sup\}$。

定义5：滑动窗口w的起点与终点都没有清晰的限制，w的终点就是当前的时间点。w的大小是窗口中事务的多少，这个值是提前设置好的。每当有一个新的事务到达时，就滑动一次窗口。新的事务连续进入窗口，同时，旧的事务被删除，滑动窗口一直被更新。

定义6：全序关系$\prec$。根据字母在字典中的顺序，如果X小于Y，则有$X\prec Y$，比如$A\prec B\prec C$[7]。同理，可以给出项集在字典中的顺序为$\prec$，比如$A\prec ABC\prec CD$。

在本文中，假设全部项都是依照全序关系排序的。

3 WFP-SW原理与算法(WFP-SW principle and algorithm)

3.1 矩阵的构造

(1)事务矩阵A的构造

用矩阵的行来标识数据流中项的集合$\{I_1,I_2,\cdots,I_m\}$，用矩阵的列标识连续到达的事务$\{T_1,T_2,\cdots,T_n\}$。设滑动窗口的大小为$|w|=n$，如果项集中包含m个项，则构造一个$m\times n$的事务矩阵，同时初始化矩阵中的所有元素为0。扫描连续到达的数据流，如果窗口没有满，那么就将连续到达的事务T_i存储进矩阵中，如果项目i出现在第j条事务中，那么就设置A_{ij}为1，如果没有出现则设置为0；当窗口满的时候，首先把窗口中最旧的事务删除，然后把新到达的事务添加进去。假设事务T_i即将到达，$column$代表最旧事务的列，则最旧事务的删除方法是：$column=i\%n$。$count$用于记录每列中1的个数，即事务的长度。

(2)二项集矩阵B的构造

设项集中有m个项，那么构造的加权二项集矩阵是$(m-1)\times(m-1)$的二项集矩阵，同时初始化矩阵中的所有元素为0。对于加权频繁1−项集L_1中的两个项I_i和I_j，如果$I_i\prec I_j$，让A中的第i行与第j行参与逻辑与运算，若支持度不小于min_sup，则项集$\{I_i,I_j\}$就是加权频繁2−项集，同时把B_{ij}的值设置成1，反之，把它的值设置为0。

3.2 WFP-SW算法的基本思想

加权频繁$k-$项集的产生：$k-$项集是通过对加权频繁$(k-1)-$项集的扩展产生的。设$\{I_{i1},I_{i2},\cdots,I_{i(k-1)}\}$是加权频繁$(k-1)-$项集，在二项集矩阵$B$中，若$B[i(k-1),iu]=1\ (I_{i(k-1)}\prec I_{iu})$，且$B[i1,iu]=B[i2,iu]=\mathrm{K}=B[i(k-2),iu]=1$，则$\{I_{i1},I_{i2},\cdots,I_{i(k-1)},I_u\}$就可以扩充为$k-$项集。同时在矩阵$A$中，让对应的$k$个项的行做逻辑与运算，如果得到的结果不小于$min_sup*|w|$，则$\{I_{i1},I_{i2},\cdots,I_{i(k-1)},I_u\}$是加权频繁$k-$项集。重复这个操作，当没有新的$(k+1)-$项集产生的时候，结束算法。

3.3 WFP-SW算法描述

综合上面的分析可知，WFP−SW算法有如下关键步骤：初始窗口阶段、滑动窗口阶段、产生加权频繁模式阶段。

该算法的伪代码如下：

输入：数据流事务DS，滑动窗口大小$|w|$，每个项目权重，用户设定的最小加权支持度min_sup；

输出：加权频繁模式；

滑动窗口中的每个事务T_j

if（$w\ is\ not\ full$）//初始窗口阶段

{

for $(i=1;i\leq m;i++)$

if $(I_i\ \ in\ \ T_j)$

$A[i,j]=1;$

else $A[i,j]=0;$

}

else//滑动窗口阶段

int $column=j\%w;$

对矩阵A中第$column$列的值进行更新，其他列的值不变

扫描矩阵A中的前m行，产生L_1

构造二项集矩阵B;

//产生加权频繁模式阶段，$L=\{I_{i1},I_{i2},\cdots,I_{i(k-1)}\}$是频繁$(K-1)-$项集

if（$B[i(k-1),iu]=1$）*and*（$B[i1,iu]=B[i2,iu]=\cdots B[i(k-2),iu]=1$）

{

扩展为$K-$项集

$$if\ (sup(\{I_{i1},I_{i2},\cdots,I_{i(k-1)},I_u\})>=min\_sup)$$

$$L_k=\{I_{i1},I_{i2},\cdots,I_{i(k-1)},I_u\};$$

}

4 实验结果及分析(The experimental results and analysis)

本文中算法采用的实验平台：Windows 7操作系统，Eclipse开发工具，编程语言是java。采用IBM data generator[8]生成的数据作为实验所用的数据。本文采用稠密数据集T40I10D100K，其中D代表事务的总数，I代表最大频繁项集长度的平均，T代表事务长度的平均值，即实验中事务总数是10万条，最大频繁项集的平均长度是10，事务长度的平均值是40。

实验对WFP-SW算法和FIM-SW[9]算法进行对比。其中后者是利用Apriori性质产生频繁K-项集，并且在频繁项集产生的过程中，需要进行连接和剪枝操作，所以算法的时间效率比较低。WFP-SW算法在产生加权频繁项集的时候，没有产生大量的候选项集，这样就省去了连接和剪枝的操作，算法的效率显著提高。图1给出了在窗口大小$|w|=1000$，$sup=0.02$的前提下，WFP-SW算法和FIM-SW算法随事务数变化的挖掘时间比较；图2给出了在$|w|=50000$，挖掘五万条事务的前提下，WFP-SW算法和FIM-SW算法随支持度变化的挖掘时间比较。

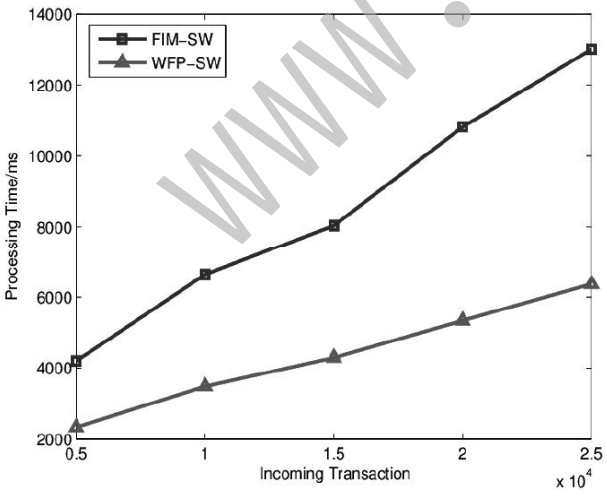

图1 不同事务长度下算法的运行时间比较

Fig.1 The running time of the algorithm under different length of transaction

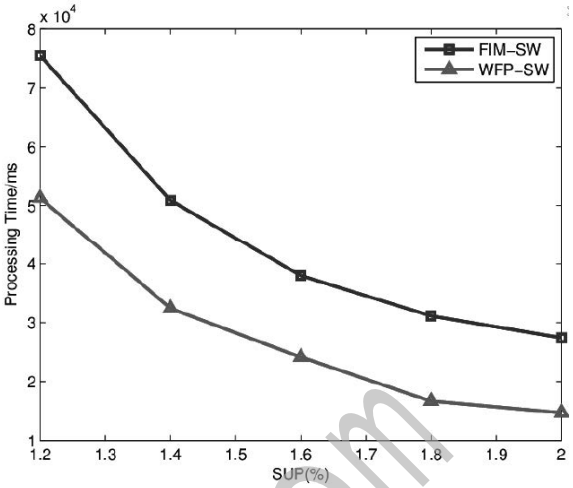

图2 不同支持度下算法的运行时间比较

Fig.2 The running time of the algorithm under different support

5 结论(Conclusion)

本文提出了基于滑动窗口模型的数据流加权频繁模式挖掘算法WFP-SW，该算法只需扫描一次数据流，数据存储采用的是矩阵数据结构，通过矩阵之间的相关操作来产生加权频繁模式。同时该算法在产生加权频繁模式的时候不产生冗余模式，通过与算法FIM-SW的对比，验证了WFP-SW算法具有更高的效率。

参考文献(References)

[1] G.Lee,U.Yun,H.Ryang.Mining Weighted Erasable Patterns by Using Underestimated Constraint-based Pruning Technique[J]. Intell.Fuzzy Syst.,2015,28(3):1145-1157.

[2] G.Lee,U.Yun,K.H.Ryu.Sliding Window Based Weighted Maximal Frequent Pattern Mining Over Data Streams,Expert Syst.Appl,2014,41(2):694-708.

[3] U.Yun,G.Pyun,E.Yoon.Efficient Mining of Robust Closed Weighted Sequential Patterns Without Information Loss[J].International Journal on Artificial Intelligence Tools,2015,24(1):01-28.

[4] 张晴,高广银.贾波数据挖掘技术在超市营销系统中的应用[J].软件工程,2016,19(5):35-38.

[5] 孙黎明.探索软件工程数据挖掘技术[J].软件工程,2015,18(5):

(下转第8页)