第35卷 第19期 计 算 机 工 程 2009年10月
Vol.35 No.19 Computer Engineering October 2009
·软件技术与数据库· 文章编号：1000—3428(2009)19—0062—03 文献标识码：A 中图分类号：TP391

关联规则的快速更新算法

邵 勇[1]，陈 波[1]，史宝东[2]，刘长征[1]
(1. 大连大学信息工程学院，大连 116622；2. 华能日照发电厂，日照 276826)

摘 要：针对基于支持度变化的最大频繁项集维护问题、频繁项集与最大频繁项集互转换时的维护问题，提出3种相应算法。在挖掘最大频繁项集的过程中不断调节支持度大小，以实现其快速更新。基于最大频繁项集子集的支持计数，将现有最大频繁项集转换为频繁项集。
关键词：最大频繁项集；数据挖掘；更新；频繁模式树

Fast Update Algorithm for Association Rule

SHAO Yong[1], CHEN Bo[1], SHI Bao-dong[2], LIU Chang-zheng[1]
(1. Institute of Information Engineering, Dalian University, Dalian 116622; 2. Rizhao Power Plant, Rizhao 276826)

【Abstract】 Aiming at the maintenance problems of maximum frequent itemsets based on support change and interconversion between frequent itemsets and maximum frequent itemsets, this paper proposes three relevant algorithms. It adjusts the support rating constantly during the process of mining maximum frequent itemsets to realize fast update of maximum frequent itemsets. Based on support counting of the subsets of maximum frequent itemsets, existing maximum frequent itemsets are transformed into frequent itemsets.
【Key words】 maximum frequent itemsets; data mining; update; frequent pattern tree

1 概述

现有最大频繁项集的挖掘算法包括Max-Miner[1], Pincer-Search[2], DMFI[3], DMFIA[4], IDMFIA[5]和增量更新算法FUMFIA[5]等。上述最大频繁项集更新算法只涉及事务库的增加，没有考虑支持度变化。鉴于此，本文提出一种基于支持度变化的最大频繁项集高效更新挖掘算法(Maximum Frequent Itemsets Updating Algorithm Support + , MFIUAS+)。

在实际操作中，人们有时需要将事务库的频繁项集转化为最大频繁项集或把最大频繁项集转换为频繁项集。后者的关键是最大频繁项集子集支持计数的计算。因此，本文提出频繁项集到最大频繁项集的转换算法(Frequent itemsets Change into Maximum Frequent itemsets, FCMF)和最大频繁项集到频繁项集的转换算法(Maximum Frequent itemsets Change into Frequent itemsets, MFCF)。

2 问题描述

关联规则挖掘问题的形式化描述如下：设 $I=\{i_1,i_2,\cdots,i_n\}$ 是项的集合，与任务相关的数据 D 是数据库事务的集合，其中每个事务 T 是项的集合，使 $T\subseteq I$。每个事务有一个标识符，称为 TID。设 A 是一个项集，事务 T 包含 A 当且仅当 $A\subseteq T$。关联规则是形 $A\Rightarrow B$ 的蕴涵式，其中，$A\subset I$, $B\subset I$ 且 $A\cap B=\Phi$。规则 $A\Rightarrow B$ 在事务 D 中成立，具有支持度 s，其中，s 是 D 中事务 $A\cup B$ 的百分比，即概率 $P(A\cup B)$。项的集合称为项集。项集的出现频率是包含项集的事务数，简称为项集的频率支持计数。如果项集满足最小支持度，则称它为频繁项集。候选 k-项集的集合通常记作 C_k，频繁 k-项集的集合通常记作 L_k，所有频繁项集集合通常记作 L，设 X 为项集，则 $Support_D(X)$表示 X 在事务库 D 中的支持度，$Support_count_D(X)$表示 X 在 D 中的支持数，min_sup 为设定最小支持度，$|X|$表示项集 X 的项数。事务库 D 对应的频繁模式树记为 FP-tree$_D$，所有频繁项目的集合记为 F_D，FP-tree$_D$ 中以项目 x 为后缀项目的路径 P 上相应的项目集记为 $I_P(x)$，其中的项目保持原有顺序，以 x 为后缀项目的路径集合记为 $Paths(x)$。

定义 设 X 为事务库 D 中的频繁项集，如果 X 的任何真超集在 D 中均为非频繁项集，则称 X 为最大频繁项集。将所有最大频繁项集的集合记为 MFS(Maximum Frequent Sets)，将所有候选最大频繁项集的集合最大频繁候选项集集合记为 $MFCS$(Maximum Frequent Candidate Sets)。

性质1[5] 如果 X 为频繁项集，那么 X 的任何非空子集都是频繁项集。

性质2[5] 如果 X 为非频繁项集，那么 X 的任何超集都是非频繁项集。

定理1 设 L 为最小支持度 min_sup 事务库 D 的所有频繁项集的集合，MFS 为最小支持度 min_sup 事务库 D 的所有最大频繁项集的集合，则一定有 $MFS\subseteq L$。

证明：对任意 $X\in MFS$，由定义可知 $Support_D(X)\geq min\_sup$，因此，$X\in L$，可得 $MFS\subseteq L$。

由定理1可知，可以用 L 作为最大频繁候选项集集合 $MFCS$。

定理2 令 $MFCS=L$，并将 $MFCS$ 中的项集按项数多少降序排序，设此时 $MFCS$ 中的第1个项集为 X_1。令 $MFCS^1=\{X|X\in MFCS\text{-}X_1 \text{ and } X\not\subset X_1\}$，并保持原有排序策略，如果 $MFCS^1$ 不为空，则设此时 $MFCS^1$ 中的第1个项集为 X_1^1。依此类推，令 $MFCS^i=\{X|X\in MFCS^{i-1}\text{-}\{X_1, X_1^1,\cdots, X_1^{i-1}\} \text{ and } X\not\subset X_1 \text{ and } \cdots \text{ and } X\not\subset X_1^{i-1}\}$，并保持原有排序策略。如果 $MFCS^i$

基金项目：辽宁省教育厅青年基金资助项目(20040052)
作者简介：邵 勇(1968－)，男，硕士研究生，主研方向：数据挖掘；陈 波，教授、博士后；史宝东，学士；刘长征，硕士研究生
收稿日期：2009-04-29 **E-mail：**shaoyongshao@sohu.com

不为空，则设此时 $MFCS^i$ 中的第 1 个项集为 X_1^i，则 $X_1 \in MFS$, $X_1^1 \in MFS$,…, $X_1^i \in MFS$。

证明：$X_1 \in MFCS$，即 $X_1 \in L$(由定理 1 可知)，$Support_D(X_1) \geq min\_sup$，即 X_1 为频繁项集，如果存在 X_1 的真超集 Y 为频繁项集，则$|Y|>|X_1|$，而 L 中频繁项集的项数最大为$|X_1|$，即 Y 为非频繁项集，与假设矛盾，因此，$X_1 \in MFS$。由 $X_1^1 \in MFCS^1$ 得出 $X_1^1 \in L$，即 X_1^1 为频繁项集，如果存在 X_1^1 的真超集为频繁项集，则只可能是 X_1，因为 $MFCS^1$ 中除 X_1^1 外的所有其他频繁项集的项数不可能大于$|X_1^1|$。而 $X_1^1 \not\subset X_1$，即 X_1^1 的所有超集均为非频繁的，所以 $X_1^1 \in MFS$，依此类推，可以证明 $X_1^i \in MFS$。

定理 3 设 x 为 FP-tree$_D$ 中频繁项目头表最后一项的项目，若从 FP-tree$_D$ 中删除此项目，则只要把每个路径 $P \in Paths(x)$ 中的 x 剪去即可。

定理 3 的证明可以由 FP-tree$_D$ 构造过程得到。

定理 4 设支持度提高 *min_sup'*后的最大频繁项集集合为 *MFS'*，如果 X 为 *MFS'*中的最大频繁项集，则 X 必定为 *MFS* 中某个最大频繁项集的子集。

证明：由 $X \in MFS'$可得 $Support_D(X) \geq min\_sup' \geq min\_sup$，所以 $X \in L$，可得 *MFS* 中一定存在至少一个最大频繁项集包含 X。

3 频繁项集的快速更新算法

3.1 FCMF 算法

FCMF 算法实现在事务库和最小支持度不变的情况下，将频繁项集集合转换为最大频繁项集集合，具体描述如下

输入 事务库 D 中频繁项集 L

输出 D 中的最大频繁项集 *MFS*

MFCS= L　　//根据定理 1

将 MFCS 中的项集按项数降序排序

MFS= Φ

while MFCS ≠ Φ　do begin

MFS= MFS ∪ MFCS[1]　//根据定理 2

MFCS= MFCS-MFCS[1]

MFCS= MFCS-{X|X ∈ MFCS and X ⊆ MFCS[1]}

return　MFS

end

3.2 MFCF 算法

MFCF 算法实现在事务库和最小支持度不变的情况下，将最大频繁项集集合 *MFS* 转换为频繁项集集合 L，其关键是频繁项集支持数的计算，具体描述如下：

输入 事务库 D 中频繁项集 *MFS*，FP-tree$_D$ 和 Htable$_D$

输出 D 中的最大频繁项集 L

L= Φ

for(i=1;i≤|MFS|,i++)

令 n=| MFS[i]|将 MFS[i]做如下处理

C_k= MFS[i]所有 k 项组合-L; //(1≤k≤n)

C= $C_1 \cup C_2 \cup \cdots \cup C_n$

for(j=1;j≤| C |,j++)

调用过程 ComputeCount(FP-tree$_D$, Htable$_D$, C[1])

将得到支持数的频繁项集 C[1]并入 L 中

end for

end for

return L

3.3 MFIUAS+算法

在 FP-tree 中，每个节点由 4 个域组成：节点名称 node_name，节点计数 node_count，节点链 node_link 和父节点指针 node_parent。为方便树遍历，创建了一个频繁项目头表 Htable，它由 2 个域组成：项目名称 item_name 和项目链头 item_head，项目链头指向 FP_tree 中与之名称相同的第 1 节点。

当支持度变大后，事务数据库 D 中的频繁一项目集 L_1 中的某些项目变成了非频繁项目，原先支持度下的 FP-tree$_D$ 必须做相应改变。其项目头表中项目为非频繁项目的项被删除，设这些非频繁项目构成的集合为 L_{n1}，$|L_{n1}|$表示 L_{n1} 的项数。

设 FP-tree$_D$ 的频繁项目头表为 Htable$_D$，Htable$_D$[i]表示 Htable$_D$ 中的第 i 项，|Htable$_D$|表示 Htable$_D$ 的总项数。当支持度变大后，FP-tree$_D$ 的更新构造算法如下：

if $L_{n1} \neq \Phi$　then

for(i=| Htable$_D$ |;i>| Htable$_D$ |-|L_{n1}|;i--)

根据 Htable$_D$[i].head 找到树中节点名称为 Htable$_D$[i].item_name 的节点 $nd_1,\cdots,nd_h$

直接删除节点 $nd_1,\cdots,nd_h$

end for

else

do nothing

end if

支持度变大时的最大频繁项集更新算法 MFIUAS+描述如下：

输入 原支持度下的最大频繁项集集合 *MFS*，更新后的 FP-tree$_D$，最小支持度 *min_sup'*和 Htable$_D$

输出 最小支持度为 *min_sup'*时的最大频繁项集集合 *MFS'*

扫描 L_1 得到小于 min_sup'的项目的集合 L_{n1}

MFCS=MFS //定理 4

for(i=1;i≤| MFCS |;i++)

MFCS [i]= MFCS [i]- L_{n1}

end for

将 MFCS 按项数大小降序排列，MFS'= Φ

while(MFCS ≠ Φ)

MFCS=MFCS- MFCS [1]

调用过程 ComputeCount(FP-tree$_D$, Htable$_D$, MFCS [1])

if　Support$_D$(MFCS [1])≥min_sup' then

MFS'= MFS' ∪ MFCS [1]

else

for all e ∈　MFCS [1]

if MFCS [1]- e 不是 MFS'并且不是 MFCS 的子集　then

MFCS= MFCS ∪ {MFCS [1]- e}

end if

end for

end if

将 MFCS 按项数大小降序排列，

end while

return MFS'

Procedure ComputeCount(FP-tree$_D$, Htable$_D$, MFCS[1])

/*计算 MFCS[1]在 D 中的支持数，由于 MFCS[1]已按项目支持数降序排序，并设最后一个项目为 i，只需根据父指针向根节点搜索即可*/

搜索项目头表的项目名称域 item_name，设 Htable$_D$ [j].item_name=i

根据 Htable$_D$[j].head 找到树中节点名称为 Htable$_D$[j]. item_name 的节点 $nd_1,\cdots,nd_h$

根据 $nd_1,\cdots,nd_h$ 及其前缀节点的父节点指针域找到包含 Htable$_D$[j].item_name 的所有路径 $P_1,P_2,\cdots,P_h$;

for(k=1; k≤h; k ++)

如果包含 MFCS[1]，那么 MFCS[1]的支持数增加 nd_k.node_count

end for

4 示例与算法分析

例 1(FCMF 算法) 设事务数据库 D 如表 1 所示，最小支持度为 50%，求最大频繁项集集合 *MFS*。

表 1 事务数据库 *D*

TID	项集
101	i_1, i_2, i_3, i_4
102	i_1, i_3
103	i_1, i_2, i_3
104	i_1, i_2, i_4

对于 D 中的频繁单项集集合 L_1={{i_1:4},{i_2:3}, {i_3:3}, {i_4:2}}、全体频繁项集 L={{i_1:4}, {i_2:3}, {i_3:3}, {i_4:2}, {i_1 i_2:3}, {i_1 i_3:3}, {i_1 i_4:2}, {i_2 i_3:2}, {i_2 i_4:2}, {i_1 i_2 i_3:2}, {i_1 i_2 i_4:2}}，运用 FCMF 算法将事务数据库 D 的频繁项集集合转化为最大频繁项集集合，具体步骤如下：

(1)令 *MFCS*=*L*, *MFS*= Φ ；

(2)将 *MFCS* 按项数降序排序得到 *MFCS*={{i_1 i_2 i_3:2}, {i_1 i_2 i_4:2}, {i_1 i_2:3}, {i_1 i_3:3}, {i_1 i_4:2}, {i_2 i_3:2}, {i_2 i_4:2}, {i_1:4}, {i_2: 3}, {i_3:3}, {i_4:2}}；

(3)*MFS*=*MFS* ∪ {{i_1 i_2 i_3:2}}={{i_1 i_2 i_3:2}}；

(4)*MFCS*={{i_1 i_2 i_4:2}, {i_1 i_4:2}, {i_2 i_4:2},{i_4:2}}；

(5)*MFS*=*MFS* ∪ {{i_1 i_2 i_4:2}}}={{i_1 i_2 i_3:2}, {i_1 i_2 i_4:2}}；

(6)*MFCS*= Φ ；

(7)*MFS*={{i_1 i_2 i_3:2}, {i_1 i_2 i_4:2}}；

(8)结束。

例 2(MFCF 算法) 设事务数据库 D 如表 1 所示，最小支持度为 50%，最大频繁项集 *MFS* {{i_1 i_2 i_3:2}, {i_1 i_2 i_4:2}}，求频繁项集集合 L。相应的频繁模式树 FP-tree$_D$ 如图 1 所示。

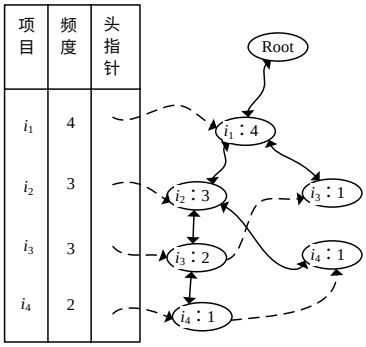

图 1 表 1 所描述 *D* 对应的 FP-tree$_D$

运用 MFCF 算法将事务数据库 D 的最大频繁项集集合 *MFS* 转化为频繁项集集合 L。

(1)*L*= Φ ；

(2)*MFS*[1]={{i_1 i_2 i_3:3}}；

(3)对其进行处理得到 *C*={{i_1}, {i_2}, {i_3}, {i_1 i_2}, {i_1 i_3}, {i_2 i_3}, { i_1 i_2 i_3}}；

(4)以项集{i_1 i_3}为例，求项集支持度，取出该项集最后一项{i_3}，查 Htable$_D$ 找到该项头指针，求出路径集合 *Paths*(i_3)={$i_1 \rightarrow i_3$, $i_1 \rightarrow i_2 \rightarrow i_3$}，将路径中{$i_3$}项的支持数相加，即 1+1=2 为该项集支持数；

(5)得到 *L*={{i_1:4}, {i_2:3}, {i_3:3}, {i_1 i_2:3},{i_1 i_3:3}, {i_2 i_3:2}, {i_1 i_2 i_3:2}}；

(6)*MFS*[2]={{i_1 i_2 i_4:3}}，因为{i_1}, {i_2}, {i_1 i_2}在 L 中已存在，所以减去；

(7)对其进行处理得到 *C*={{i_4},{i_1 i_4}, {i_2 i_4}, {i_1 i_2 i_4}}；

(8)根据项头表查 FP-tree$_D$ 得到 *C* 中项集的支持度并加入 L 中，*L*=*L* ∪ {{i_4:2},{i_1 i_4:2},{i_2 i_4:2},{i_1 i_2 i_4:2}}={{i_1:4},{i_2:3}, {i_3:3}, {i_4:2},{i_1 i_2:3},{i_1 i_3:3},{i_1 i_4:2},{i_2 i_3:2}, {i_2 i_4:2},{i_1 i_2 i_3:2},{i_1 i_2 i_4:2}}；

(9)结束。

例 3(MFIUAS+算法) 设事务数据库 D 如表 1 所示，最小支持度为 60%，原支持度下最大频繁项集集合 *MFS*={{i_1 i_2 i_3:2},{i_1 i_2 i_4:2}}，求新支持度下的最大频繁项集 *MFS'*。

根据新的最小支持度，得到变为非频繁一项集集合为 L_{n1}={{i_4:2}}，通过 FP-tree 构造更新算法得到如图 2 所示的新 FP-tree，主要步骤如下：

(1)*MFS'*= Φ ；

(2)*MFCS*=*MFS*={{i_1 i_2 i_3:2},{i_1 i_2 i_4:2}}；

(3)将 *MFCS* 中的每个项集去除非频繁项目得到 *MFCS*=*MFS*={{i_1 i_2 i_3},{i_1 i_2}}并按项数降序排列；

(4)*MFCS*=*MFCS*-*MFCS*[1]={{i_1 i_2}}, *MFCS*[1]={i_1 i_2 i_3}；

(5)调用过程 ComputeCount(FP-tree$_D$, Htable$_D$, *MFCS*[1])，计算 *MFCS*[1]支持数等于 2；

(6)判断 *MFCS*[1]支持度小于新给定的最小支持度，所以，对其进行分解，加入 *MFCS* 中并排序，*MFCS*=*MFCS* ∪ {{i_2 i_3}, {i_1 i_3}}={{i_1 i_2},{i_2 i_3},{i_1 i_3}}；

(7)*MFCS*=*MFCS*-*MFCS*[1]={{i_2 i_3},{i_1 i_3}}, *MFCS*[1]={i_1 i_2}；

(8)调用过程 ComputeCount(FP-tree$_D$, Htable$_D$, *MFCS*[1])，计算 *MFCS*[1]支持数等于 3；

(9)*MFS'*=*MFS* ∪ {{i_1 i_2:3}}={{i_1 i_2:3}}；

(10)*MFCS*=*MFCS*-*MFCS*[1]={{i_1 i_3}}, *MFCS*[1]={i_2 i_3}；

(11)调用过程 ComputeCount(FP-tree$_D$, Htable$_D$, *MFCS*[1])，计算 *MFCS*[1]支持数等于 2，不满足最小支持度；

(12)对其进行分解，加入 *MFCS* 中并排序，*MFCS*=*MFCS* ∪ Φ ={{i_1 i_3}}；

…

(13)*MFS'*={{i_1 i_2:3}, {i_1 i_3:3}}；

(14)结束。

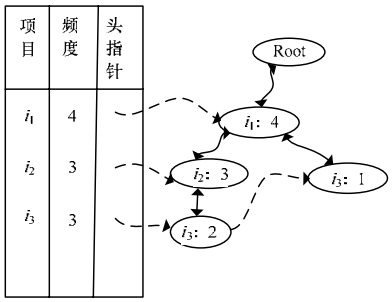

图 2 表 1 所描述 *D* 对应的新 FP-tree$_D$

由以上 2 种算法的描述及其示例分析可得如下结论：

(1)FCMF 算法利用现有频繁项集集合，解决了将频繁项集集合转化为最大频繁项集集合的问题。它运用原来挖掘的频繁项集集合 L，对其按项数多少降序排序，快速确定最大频繁项集。若不采用此算法，则必须对事务库利用最大频繁项集生成算法重新生成，其效率很低。

(2)MFCF 算法利用现有最大频繁项集集合，解决了将最大频繁项集集合转化为频繁项集集合的问题，它运用原来挖掘的最大频繁项集集合 *MFS*，通过频繁模式树，计算出最大

（下转第 68 页）