一个改进项目的加权关联规则挖掘算法

刘 燕
(闽江学院 计算机科学系, 福建 福州 350108)

摘要: 提出了一个改进的项目加权关联规则挖掘算法. 该算法利用一个加权频繁项目集必须满足的加权支持度下界, 对加权频繁候选项目集进行剪枝, 该下界计算简便, 可以减少挖掘的计算量. 理论分析和实验表明本算法和 *MINWAL (W)* 相比, 具有生成候选集数量少、挖掘效率高等特点, 特别在项目权值相差不大时, 本算法的优势更明显.

关键词: 数据挖掘; 关联规则挖掘; 交易数据库
中图分类号: 311. 13 文献标识码: A 文章编号: 1007- 855X(2008) 04- 0034- 04

An Improve Mining Algorithm of Weighte Association Rule

LIU Yan
(Department of Computer Science, Minjiang University, Fuzhou 350108, China)

Abstract: An improved mining algorithm is introduced to handle the problem of weighted association rule. A lower support bound of frequent item sets with weights is applied to this algorithm to prune the number of the candidate item sets, which can increase its simplicity and decrease the number of calculation in the mining process. It is shown through analysis and examples that the algorithm is efficient and performs better than *MINWAL (W)*, especially when the difference between the weights of items is not so large.

Key words: data mining; association rule mining; transaction database

0 引 言

在交易数据库中, 涉及到的交易项目通常很多, 由于用户并不是对每个项目都感兴趣, 因而, 在考虑问题时他们总是把自己感兴趣的项目作为优先或重点研究的对象, 希望加强这些项目的作用, 对那些不感兴趣的项目, 把它们放在次要的位置, 以降低其重要性. 在关联规则挖掘中, 为了达到上述目的, 通常对项目设置不同的权值, 对用户感兴趣的项目设置较高的权值, 对用户不感兴趣的项目设置较低的权值, 这种情况本文称为项目加权. 项目加权前提条件是假定考虑的项目与时间无关, 权值反映的是项目对用户的重要程度, 其权值的大小不受时间的影响. 以交易数据库 T 为例, $T = \{T_1, T_2, \cdots, T_k\}$. 其中, T_j 为某一笔交易, 设 $I = \{i_1, i_2, \cdots, i_m\}$ 是 T 中所涉及的项目集合, i_j 为任意一个项目, m 为项目总数. 对任意的 T_i, 均满足 $T_i \subseteq I$, $1 \leqslant i \leqslant k$, k 为 T 的交易数. 在项目加权中, 为了表示项目集合 I 中某一项目 i_j (其中 $j = 1, 2, \cdots, m$) 对于用户的重要程度, 给其分配一个权值 w_j, 其中 j 与 i_j 中的 j 对应.

1 相关定义

在关联规则挖掘过程中, 除了利用支持度和可信度之外, 如何利用项目权值有效地挖掘出用户感兴趣的规则, 文献 [1] ~ [6] 等对此问题进行了研究, 本文将以这些文献为基础对加权关联规则挖掘算法作进一步的讨论和分析.

为了描述问题的方便, 结合前面项目权值的概念, 对加权关联规则 $X \Rightarrow Y$, 有如下定义:

定义 **1** 加权支持数定义为: $countw(X \Rightarrow Y) = w_{X \Rightarrow Y} * count(X \Rightarrow Y)$, 其中 $w_{X \Rightarrow Y}$ 为关联规则 $X \Rightarrow Y$ 的平

收稿日期: 2007- 12- 25. 基金项目: 福建省教育厅项目 (项目编号: JB07172).
作者简介: 刘燕 (1966-), 女, 硕士, 高级工程师. 主要研究方向: 数据库、数据挖掘算法. E- mail: liuyan_fj_cn@ 126. com

均加权 $W_{X\Rightarrow Y} = (\sum_{i=1}^{m} W_i)/m$, 式中 w_i 对应 $(X \cup Y)$ 中项目的权值, m 为 $(X \cup Y)$ 中的项目数, $count(X\Rightarrow Y)$ 为交易集 D 中包含项目 $(X \cup Y)$ 的交易数.

定义 **2**　加权支持度定义为: $supp\,w(X\Rightarrow Y) = countw(X\Rightarrow Y)/|D| = w_{X\Rightarrow Y} * count(X\Rightarrow Y)/|D|$, 其中 $|D|$ 为交易集 D 的交易数.

定义 **3**　给定加权关联规则 $X\Rightarrow Y$ 的最小加权支持数阈值 (mincountw), 最小加权支持度阈值为 minsupp w, 它们的关系满足:

$$\text{minsupp}\,w = \text{mincount}\,w/|D|$$

定义 **4**　项目集 X 的加权支持度不低于最小加权支持度阈值时, 即: $supp\,w(X) \geqslant \text{minsupp}\,w$ 或者 $countw(X) \geqslant \text{minsupp}\,w * |D|$, 称 X 为加权频繁项目集.

2加权关联规则挖掘算法的改进

由于项目权值的引入, 一般的关联规则挖掘算法不再适应于基于定义 2的加权支持度计算方法的加权关联规则挖掘, 因此, 必须寻找新的方法来解决项目加权关联规则的挖掘问题.

定理 **1**[1]　如果 $s-$ 项集 Y 是任何频繁 $k-$ 项集 X 的子集, 那么有 $count(Y) \geqslant countmin(X)$.

$$countmin(X) = \left[\text{minsuppw} * |D|/\left(\left(\sum_{i=1}^{s} W_i + \sum_{j=1}^{k-s} W_{rj}\right)/k\right)\right] \quad (1)$$

(1) 中 $\sum_{i=1}^{s} W_i$ 为 Y 中项目的权和, $\sum_{j=1}^{k-s} W_{rj}$ 为 X 项目集中除 Y 中 s 个项目之外余下的 $(k-s)$ 个最大权值之和.

(1) 意味着, 如果 Y 项目集是任何频繁 $k-$ 项集 X 的子集, 那么有 $count(X) \geqslant countmin(X)$ 由于 Y 的支持数 $count(Y) \geqslant count(X)$, 所以 $count(Y) \geqslant countmin(X)$ 成立, 即有:

$$count(Y) \geq \left[\text{minsuppw} * |D|/\left(\left(\sum_{i=1}^{s} W_i + \sum_{j=1}^{k-s} W_{rj}\right)/k\right)\right]$$

定理 **2**[1]　对于任意一个项目集 $X = \{x_1, x_2, \cdots, x_n\}$, $x_1, x_2, \cdots, x_n$ 是按权值 w_i 的升序排列的项目集 X 的项目, w_i 是对应项目 x_i 的权值, $0 < w_i < 1$, 项目集 X 的权值为 $W_x = (\sum_{i=1}^{s} W_i)/n$, 对于 X 的任何子集 Y, $Y = \{x_i, x_{i+1}, \cdots, x_n\}$, 则 $W_Y = (\sum_{i}^{n} W_i)/(n-i+1)$, 显然, $w_Y \geqslant w_x$, 若 X 为加权频繁项目集, 则 Y 也是加权频繁项目集.

定理 **3**[7]　假定 $k-$ 项目集 X 是加权频繁项目集, 若存在 Y, 满足 X 是 Y 的真子集, 且 Y 也是加权频繁项目集, 则 $supp(X)$ 有下界 $(k * \text{minsupp}\,w)/\sum_{i=n-k}^{n} W_i$, 其中 $\sum_{i=n-k}^{n} W_i$ 为 Y 中项目权值按升序排列的后 $(k+1)$ 个权值之和, $w_i \leqslant w_{i+1}$.

定理 **4**[7]　设 $D = \{T_1, T_2, \cdots, T_k\}$ 为交易数据库, T_j 为某一笔交易, $I = \{i_1, i_2, \cdots, i_m\}$ 是 D 中所涉及的项目集合, 其中项目 i_j 是按权值 w_j $(1 \leqslant j \leqslant m)$ 升序排列的项目, w_j 是对应项目 i_j 的权值, m 为项目总数. 设任意 s 维项目集 $Y \subseteq I$, 和 $s1$ 维项目集 $X \subseteq I$, 且 $Y \subset X$, 则有 $s * \text{minsupp}\,w/\sum_{i=m-s}^{m} w_i$ 是所有 $s-$ 加权频繁项集 Y 的支持度下界. 其中分母 $\sum_{i=m-s}^{m} w_i$ 是项目集 I 中项目的 $(s+1)$ 个最大权和.

基于上述定义和定理, 本文提出如下加权关联规则挖掘算法:

supp w Algorithm (minsupp w, D, W)　// minsupp w: 加权支持度阈值; D: 数据库; W: 项目权表

(1) max − size = found w (D);

(2) C_i, L_i 初始化为 Φ　// L_i: 加权频繁 $i-$ 项集集合; C_i: 加权频繁项集的候选集; i 为不大于 max−size的自然数.

(3) C_1 = search (D, W);

(4) L_1 = counting (D, W, minsupp w);

(5) for ($k = 2; L_{k-1} \neq \Phi; k++$) do

(6) $\{C_k = join - gC_k(C_{k-1}, L_{k-1})$;

(7) 计算 C_k 中所有项目集 X 的下界并将不小于其下界的项目集保留在 C_k 中;

(8) $L_k = \{c \in C_k \mid supp\ w(C) \geq minsupp\ w\}$;

(9) $C_k = Prune(C_k)\}$

(10) $Answer = \bigcup_k L_k$;

算法函数功能说明:

found w(): 扫描数据库 D, 计算出数据库中每个交易(项目集)的维数, 并求出最大维数, 由此确定该数据库中加权频繁项目集的最大可能的维数 $max - size$.

search(): 对每个 1-项目集 Y, 根据定理 4计算其 $supp(Y)$ 的下界 $(s^* minsupp\ w) / \sum_{i=m-1}^{m} W_i$, 其中 $w_i \leqslant w_{i+1}$, 若 $supp(Y)$ 不小于其下界则将 Y 放入 C_1 中.

counting(): 对 $C1$中的每个 1-项目集 Y, 计算其支持度 $supp(Y)$ 和加权支持度 $supp\ w(Y)$, 若 $supp\ w(Y) \geqslant minsupp\ w$, 则该 1-项目集 Y就是加权频繁候项目集, 将其加入到 L_1中. 对 L_1 中的项目集按其项目权值升序排列. 按上述(1)计算 $countmin(X)$ 值对 C_1 剪枝, X 为包含 Y的任何 k维项目集, $k = 1, 2, \cdots, max - size$, Y是 1-项目集, 故 $S = 1$, 只要存在一个 k值, 使 $count(Y) \geqslant countmin(X)$, 那么, 该 1-项目集 Y就是加权频繁候选项集, 应保留在 C_1 中. 并对 C_1 中的项目集按其项目权值升序排列.

根据 *search()* 函数, 1维加权频繁候选项目集 C_1 包含了全部 k维加权频繁项目集 L_1, 而且 C_1 中的项目集一定包含了所有的大于 1维的加权频繁项目集的的 1维子集. 同时根据定理 2, 那么 L_1一定是 C_2的真子集, 而且 C_2的项目集中的项目不可能包含不属于 L_1 和 C_1 项目集项目. 依此类推, 于是有利用 $(k-1)$ 维大项集 L_{K-1} 和 $(k-1)$ 维候选项目集 C_{k-1} 产生新的 K 维候选项目集 C_k 的函数 $join - gC_k()$, 具体描述如下:

join - $gC_k()$

(1) $C_k = \Phi$;

(2) For all itemsets $X \in L_{K-1}$ and $Y \in C_{K-1}$ do

(3) If $y_i < x_1$ then //x_1 和 y_i 分别是项目集 X 的第 1项和 Y的第 i项

(4) begin

(5) 取项目集 Y的第 i项和项目集 X 的所有项组成一个新的项目集 C

(6) 将 C 加入到 C_k;

(7) end;

Prune(): 计算所有满足下界的项目集 Y对应的 $countmin(X)$ 值, X 为包含 Y的任何 R维项目集, $R = 1, 2, \cdots, max - size$, Y是 S维项目集, $S < R$若对于所有的 R 值, 都有 $count(Y) < countmin(X)$ 成立, 那么, 该 S-项目集 Y就不可能是加权频繁候选项集, 应将该项目集 Y从 C_K 中移去. 然后对 C_K 中的项目集按其项目权值升序排列.

3算法分析

表 1是一张商家的交易项目权值表, 权值的高低由利润高低来取值, 表 2是交易表:

表 1 交易项目权值表 (W)

Tab 1 Transactions item weights(W)

编号	项目名称	权值 W (利润高低)
1	刻录机	0.1
2	光盘	0.2
3	啤酒	0.3
4	尿布	0.4

表 2 交易表 (D)

Tab 2 Transactions(D)

交易编号	交易项目	交易编号	交易项目
1	啤酒, 尿布 (3, 4)	5	刻录机, 光盘, 尿布 (1, 2, 4)
2	刻录机 (1)	6	刻录机, 光盘 (1, 2)
3	刻录机, 啤酒 (1, 3)	7	啤酒, 尿布, 光盘 (2, 3, 4)
4	尿布, 光盘 (2, 4)	8	刻录机, 光盘 (1, 2)

在现实生活中, 商家往往希望获取的信息与他的高利润商品有关, 如表 1中的光盘的利润很低, 而一包尿布的利润可能是光盘的好几倍, 这样商家期望能获取这些高利润商品如啤酒、尿布的信息等. 本文以表 1和表 2为实例分析提出的挖掘算法在挖掘结果和效率等方面的有效性, 若取最小加权支持度为 0.08 进行加权关联规则挖掘, 为了表达方便, 项目集中的项目用项目编号表示, 根据算法思想, 具体步骤如下:

1) 求 C_1: 求得各个 1- 项集 {1}、{2}、{3}、{4} 对应的支持度数是 {5, 5, 3, 4}, 对应的支持度为 {0.625, 0.625, 0.375, 0.5}. 对于所有的 1- 项集 Y, 根据定理 4计算其 $supp(Y)$ 的下界约为 0.114, 由于每个 1- 项集 Y 的支持度 $supp(Y)$ 都大于其共同的下界, 所以, 将 1- 项集 Y 都加入到 C_1 中.

2) 求 L_1: 对于 1- 项集其 $supp\,w$ 值大于 0.08的有 {2}, {3}, {4}, 所以它们是频繁项目集, 应加入到 L_1 中. 然后再计算 C_1 中它们各对应的 $countmin(X)$ 值, 对 C_1 剪枝.

3) 求 C_2: C_2 由 C_1 和 L_1 产生, 将 L_1 中的项目集的第一项和 C_1 中的项目集中的第 i项按 $join-gC_k()$ 函数算法进行比较. 得出 $C_2 = \{\{1, 2\}, \{1, 3\}, \{2, 3\}, \{1, 4\}, \{2, 4\}, \{3, 4\}\}$.

4) 对 C_2剪枝: 首先计算出 C_2中的各项目集的支持度, 对于所有的 C_2中的 2- 项集 Y, 计算其 $supp(Y)$ 的共同的下界为 0.177 8, 然后将支持度小于下界的项目集 {1, 3}, {2, 3}, {1, 4} 从 C_2 中移去, 于是 $C_2 = \{\{1, 2\}, \{2, 4\}, \{3, 4\}\}$.

5) 求 L_2, 对 C_2 中的每个项目集求加权支持度, 得 $L_2 = \{\{2, 4\}, \{3, 4\}\}$.

6) 进一步对 C_2 剪枝, 计算 C_2 中它们对应的各 $countmin(X)$ 值.

7) 求 C_3: 与 3) 步方法一样, C_3 由 L_2 和 C_2 导出, $C_3 = \{\{1, 2, 4\}, \{1, 3, 4\}, \{2, 3, 4\}\}$.

8) 对 C_3剪枝: 3- 项目集的共同支持度下界为 0.24, C_3 中的每个项目集的支持度都小于其下界值, 故 $C_3 = \Phi$, 从而 $L_3 = \Phi$, 挖掘加权频繁项目集工作结束.

本算法得出的加权频繁项目集与直接计算得出的结果一致.

可以看出, 在上述加权关联规则挖掘过程中, 项目集是否是加权频繁项目集既取决于项目集在数据库中出现的频率, 也取决于项目的权值大小.

本算法中 C_K 是在 $(K-1)$ 维加权频繁项集 L_{K-1} 和 $(K-1)$ 维加权频繁候选项目集 C_{k-1} 的基础上产生的, C_K 的大小取决于 L_{K-1} 和 C_{k-1} 的大小. 本文采用方法是以定理 2为依据的, 与文献 [1] 中的 C_K 生成算法比较, 本文产生的 C_K 小, 因为 C_{k-1} 中小于 L_j 中的 x_1 的项目肯定比数据库的项目集合 I中满足该条件的少. 由于产生的初始加权频繁候选集小, 使后面的算法步骤计算量大大减小.

另外, 文献 [1] 所使用的剪枝策略对初始生成的 C_K 中的每个项目集都要计算其最小支持度数 $countmin(X)$, 每个项目集的最多计算次数为 $(m-k)$, $m = max-size$, 即数据库中最长的项目集的维数, 平均计算次数是 $(m-k)/2$ 每次循环计算 $Countmin(X)$ 的复杂度为 $O(n^*(m-k))$. 本算法采用所有 k 维加权频繁项目集的共同支持度下界剪枝, 减少了计算次数. 在对 C_K 中的项目集进行剪枝时, 无论 C_K 中有多少个项目集都只需计算这些项目集的共同下界一次. 有了这个共同下界, 可以先将 C_K 中小于该下界的所有的项目集剪掉. 为了缩小 C_K 项目集, 降低 C_K 产生过程的复杂度, 本算法在进一步的剪枝策略中同样采用计算 $countmin(X)$ 值的方法, 但由于前面经过下界的过滤, 已大大减少参与计算 $countmin(X)$ 值的项目集. 在相同的实验条件下, 利用文献 [1] 中的算法和本文算法进行加权关联规则挖掘, 实验结果与上述理论分析结果相符, 特别在项目权值相差不大时, 该算法的优势更明显.

4结 论

关联规则挖掘的主要目标是挖掘出用户感兴趣的关联规则, 面对海量数据, 挖掘效率是评价挖掘方法的主要指标. 本文以文献 [1~ 7]等为基础, 提出了一个改进的项目加权关联规则挖掘算法. 该算法在挖掘过程中结合了用户的主观兴趣度量 (项目权值), 并利用一个加权频繁项目集必须满足的加权支持度下界, 对加权频繁候选项目集进行剪枝, 从而使挖掘过程生成较少的候选集数量, 提高了挖掘效率.

参考文献:

[1] CAI C H, ADA W C FU, CHENG C H, et al. Mining Association Rules with Weighted Items [C]. In Proc. of IDEAS Symp., August 1998.

[2] 欧阳为民, 郑诚, 蔡庆生. 数据库中加权关联规则的发现 [J]. 软件学报, 2001, 12(4): 612- 619.

[3] 陆建江, 钱祖平. 数据库中布尔型及广义模糊加权关联规则的挖掘 [J], 应用数学, 1999, 12(2): 38- 43.

[4] 路松峰, 胡和平. 加权关联规则的开采 [J]. 小型微型计算机系统, 2001, 22(3): 347- 375.

[5] 陆建江. 加权关联规则挖掘算法的研究 [J]. 计算机研究与发展, 2002, (10): 1281- 1286.

[6] 张智军, 方颖, 许云涛. 基于 Apriori算法的水平加权关联规则挖掘 [J]. 计算机工程与应用, 2003, 39(14): 197- 199.

[7] 刘燕. 一个加权频繁项目集的支持度下界 [C]. 计算机科学, 2006, 33(增刊): 317- 319.