

一种基于矩阵和权重改进的 Apriori 算法

边根庆, 王 月

(西安建筑科技大学 信息与控制工程学院, 陕西 西安 710055)

摘 要: 提出基于矩阵和权重的一种改进算法——MW_Apriori 算法. 该算法首先通过扫描一次事务数据库来构造 0-1 事务矩阵, 其次赋予项和事务权重, 并计算项的权重支持度, 从而得到频繁项集. 实验结果表明, MW_Apriori 算法避免了数据库的重复扫描, 使得时间和空间的耗费显著减少, 同时能有效地挖掘出隐藏的、更有价值的事件.

关键词: 关联规则; MW_Apriori 算法; 事务矩阵; 权重支持度

中图分类号: TP31

文献标识码: A

文章编号: 1000-7180(2017)01-0136-05

An Improved Apriori Algorithm Based Matrix and Weight

BIAN Gen-qing, WANG Yue

(School of Information and Control Engineering, Xi'an University of Architecture and Technology, Xi'an 710055, China)

Abstract: An improved algorithm, which is called MW_Apriori algorithm is proposed based on the matrix and weight in this paper. Firstly, build the 0-1 transaction matrix by scanning transaction database. Then items and transactions are assigned to weights, and the weighted support of items are caculated, accordingly gettign the frequent itemsets. Experiments show that MW_Apriori algorithm avoids rescanning the database, making the cost of time and space significantly reduced, at the same time can effectively mine the hidden and valuable rare events.

Key words: association rule; MW_Apriori algorithm; transaction matrix; weighted support

1 引言

数据挖掘是挖掘出蕴藏在海量数据中的有趣的、有价值的模式和知识^[1]. 数据挖掘能为我们提供有价值的、重要的信息或知识, 从而产生不可估计的经济效益.

通常, 数据挖掘的功能主要包括概念描述、关联规则分析、聚类分析、分类与回归、趋势分析等^[2]. 其中关联规则挖掘是数据挖掘的一个重要研究方面, 频繁项集挖掘^[3]在关联规则挖掘中扮演着非常重要的角色, Agrawal 在 1993 年和 1994 年分别提出频繁项集挖掘和 Apriori 算法^[4]. Apriori 算法在挖掘过程中, 若最小支持度设置得太高, 较低频率的物品有可能被忽略, 从而导致规则的损失, 而这些规则中可能含有关键的信息; 若它被设置得太低, 可能会挖

掘出大量的无意义的规则. 因此, 在一些情况下, 若完全依赖于支持度^[5]和置信度^[6]这两个阈值是不理想的. 此外, Apriori 算法在产生频繁项集的迭代过程中, 必须反复扫描数据库, 产生大量的候选项集, 导致计算量很大, 耗费大量的时间和空间.

综上, Apriori 算法主要存在两方面的不足: (1) 时间和空间耗费很大; (2) 在挖掘过程中, 没有考虑到项和事务的重要性, 最终产生的关联规则不理想. 本文鉴于此, 提出基于矩阵和权重改进的 Apriori 算法, 目的是进一步提高该算法的性能.

2 Apriori 算法分析

Apriori 是经典的挖掘频繁项目集算法. Apriori 算法基于频繁项集性质的先验知识运用逐层搜索的迭代方法^[7]逐层搜索出候选项集, 最终目

收稿日期: 2016-04-01; **修回日期:** 2016-05-18

基金项目: 国家自然科学基金项目(61073196, 61272458); 陕西省自然科学基金基础研究计划项目(2014JM2-6119); 榆林科技计划项目(2014CXY-12)

标是迭代地挖掘出频繁项集. 其中频繁 1-项集用于探索频繁 2-项集,如此循环,直到不能找到新的频繁项集时,算法结束,之后根据最小支持度和最小置信度便可产生强关联规则^[8]. 采用此方式产生的频繁项集既不会遗漏也不会重复,但由于找出每个频繁 k -项集时都需一次完整的数据库扫描,所以为了提高频繁项集逐层产生的效率,Apriori 算法使用先验性质的重要性来压缩搜索空间^[9].

Apriori 算法的具体步骤如下:

- ① 通过扫描事务数据库,记录每个项的个数,与最小支持度比较,得到满足最小支持度的频繁 1-项集 L_1 ;
- ② 为找出 L_k ,用 L_{k-1} 进行自连接产生长度为 k 的候选项集 C_k ;
- ③ C_k 是 L_k 的超集,换句话说,所有的频繁 k -项集将包含在候选项集 C_k 中,但是 C_k 也可以包含非频繁项集. 此时我们可以利用先验性质压缩 C_k ,如果候选 k -项集的 $(k-1)$ 子集不在 L_{k-1} 中,则该 k 项集也不是频繁的,可从 C_k 中直接删除.

3 基于矩阵和权重改进的 Apriori 算法

由于找出每个频繁项集时都需一次完整的数据库扫描,在面对大型事务库的时候,算法开销很大,会严重降低 Apriori 算法的运行效率. 此外,由于 Apriori 算法并没有考虑挖掘出的规则的价值,所以可能挖掘出大量无意义的规则,所以为了减少扫描数据库次数,以及反映不同项和事务的重要性,挖掘出更有价值的信息,本文对 Apriori 进行改进,提出基于矩阵(Matix)和权重(Weight)的 MW_Apriori 算法.

3.1 MW_Apriori 算法思想

假设事务数据库 D 中,包括 m 个事务和 n 个事务项,则可构造 m 行 n 列的 0-1 事务矩阵 M 如下:

$$M = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}$$

式中, $a_{ij} = \begin{cases} 1, a_{ij} \in T_i \\ 0, a_{ij} \notin T_i \end{cases}$, 式中, $i = 1, 2, \cdots, m, j = 1, 2, \cdots, n$ 上述矩阵 M 的行与事务数据库中的事务相对应,列则对应该事务所包含的项.

用 $I = \{I_1, I_2, \cdots, I_n\}$ 表示事务数据库 D 所包含的项集, n 表示项的个数. I_j 在事务数据库中出现

的概率是 $P(I_j)$, 计算见式 (1), I_j 的权重记为 $w(I_j)$, 与 $P(I_j)$ 有关, $w(I_j)$ 的计算见式 (2).

$$P(I_j) = l/m \tag{1}$$
$$w(I_j) = 1/P(I_j) \tag{2}$$

式中, l 表示 I_j 在事务集中出现的次数,即上述矩阵中第 j 列中 1 的个数, m 是事务记录的个数.

事务 T_k 指事务数据集中的第 k 条记录,它的权重指该事务中所包含的项的平均权重,记为 $w_t(T_k)$,即对 $a_{kj} = 1$ 的所有 $w(I_j)$ 求平均值,其中, $j = 1, 2, \cdots, n$ 计算见式 (3).

$$w_t(T_k) = \frac{\sum_{j=1}^{n \& I_j \in T_k} w(I_j)}{|T_k|} \tag{3}$$

式中, $|T_k|$ 表示事务 T_k 中包含的项的个数.

项的权重支持度记为 $wsupport$, 权重支持度表示包含该项的事务权重占有所有事务权重的比例,根据矩阵易知哪些事务包含了该项,计算见式 (4).

$$wsupport(S) = \frac{\sum_{k=1}^{m \& S \subseteq T_k} w_t(T_k)}{\sum_{k=1}^m w_t(T_k)} \tag{4}$$

式中, S 表示事务数据库中的任意项.

3.2 MW_Apriori 算法描述

基于上述描述的矩阵和权重,对 Apriori 算法进行改进,具体步骤如下:

- (1) 扫描事务数据库,构造 0-1 事务矩阵,并根据事务矩阵计算出每个项和事务的权重,即 $w(I_j)$ 和 $w_t(T_k)$.
- (2) 根据事务矩阵得到候选 1-项集 C_1 ,计算 C_1 中每个项的权重支持度 $wsupport(S)$,找出满足最小支持度的频繁 1-项集 L_1 ;
- (3) 为找出 L_k ,用 L_{k-1} 进行自连接产生长度为 k 的候选项集 C_k ,根据事务矩阵和权重支持度计算方法计算出 C_k 中每个项的权重支持度,找出满足最小支持度的频繁 k -项集 L_k ,直到不能找到新的频繁项集算法结束.

算法流程见图 1 所示.

3.3 MW_Apriori 算法实例

以事务数据表 1 为例,该事务数据表有 4 个事务,5 个事务项,假设最小支持度为 0.30.

表 1 事务数据

Tid	事务
100	A,C,D
200	B,C,E
300	A,B,C,E
400	B,E

用 MW_Apriori 算法进行挖掘,过程如下:

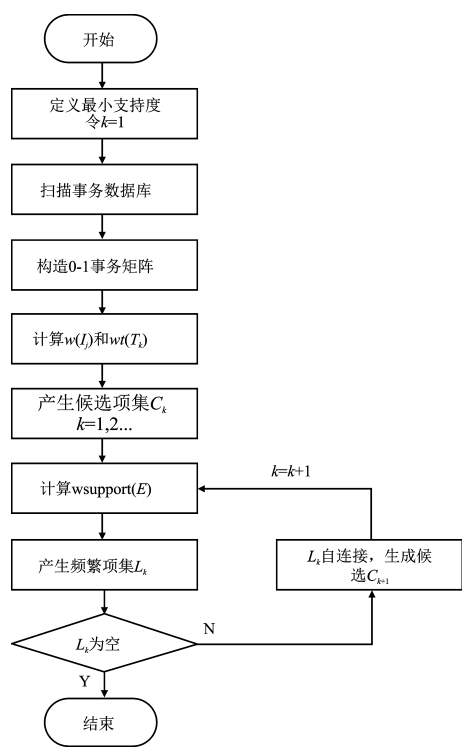


图1 MW-Apriori 算法流程

① 根据事务数据库可构造 4 行 5 列的 0-1 事务矩阵 M 如下:

$$M = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 \end{bmatrix}$$

根据式(1)和式(2), $P(A) = 1/2, w(A) = 2.00$, 用同样的方法计算出其他项的权重, 见表 2.

表 2 项的权重

I_j	$w(I_j)$
A	2.00
B	1.33
C	1.33
D	4.00
E	1.33

根据式(3), 以数据库中的第一记录为例, 由事务矩阵可知, 该事务包含 A, C, D 三项, $wt(T_1) = (2.00 + 1.33 + 4.00)/3 = 2.44$, 可用同样的方法计算出其他事务的权重见表 3.

表 3 事务权重

Tid	$wt(T_k)$
100	2.44
200	1.33
300	1.50
400	1.33

② 根据式(4), 以第一个项 A 为例, 由事务矩阵可知第一个事务和第三个事务包含该项, $wsupport(A) = (2.44 + 1.50)/(2.44 + 1.33 + 1.50 + 1.33) = 0.597$, 用同样的方法计算出 C_1 中每个项的权重支持度见表 4.

表 4 候选 1-项集的权重

C_1	$wsupport(S)$
A	0.597
B	0.630
C	0.798
D	0.370
E	0.630

与最小支持度比较后得到 $L_1 = \{A, B, C, D, E\}$

③ 将 L_1 进行自连接, 得到候选项集 C_2 , 根据事务矩阵和式(4) 计算出每个候选项的权重见表 5.

表 5 候选 2-项集的权重

C_2	$wsupport(S)$
AB	0.227
AC	0.597
AD	0.370
AE	0.227
BC	0.429
BD	0.000
BE	0.630
CD	0.370
CE	0.429
DE	0.000

与支持度比较得到 $L_2 = \{AC, AD, BC, BE, CD, CE\}$, 如此循环, 直到不能找到新的项集, 算法结束.

4 实验及算法分析

本文采用两个实验来测试改进后的算法性能. 实验一测试 MW_Apriori 将算法在运行时间上的优越性. Apriori 算法、文献[10]中的 M_Apriori 算法以及 MW_Apriori 算法用同样的实验环境进行测试. 假定最小支持度为 0.25, 用两种算法挖掘不同数量的数据记录, 时间消耗对比如图 2 所示.

由图 2 的实验结果可知, 在处理相同数量的事务记录时, MW_Apriori 算法消耗的时间明显少于

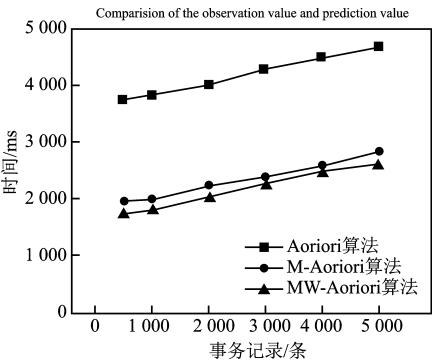


图 2 MW_Apriori 与 M_Apriori、Apriori 算法运行时间对比

Apriori 算法所需的时间,较 M_Apriori 算法相比,有小幅度的减少.

虽然较 M_Apriori 算法相比,MW_Apriori 算法在运行时间只有不太明显的优越性,但在产生的关联规则有效性方面有很大的提升.为此用实验二测试 M_Apriori 算法和 MW_Apriori 算法产生的频繁项集的和对应的支持度,为方便显示实验结果,在这里选取的事务数据库包含 9 条事务记录,5 个事务属性,见表 6.

表 6 事务数据

Tid	事务
100	I_1, I_2, I_5
200	I_2, I_4
300	I_2, I_3
400	I_1, I_2, I_4
500	I_1, I_3
600	I_2, I_3
700	I_1, I_3
800	I_1, I_2, I_3, I_5
900	I_1, I_2, I_3

假设最小支持度为 0.25,实验测试 1-项集和 2-项集的支持度,结果见表 7 和表 8.

表 7 MW_Apriori 与 M_Apriori 的 1-项集支持度对比

项	MM_Apriori 算法		M-Apriori 算法	
	支持度	频繁项集	权重支持度	频繁项集
I_1	0.669	Y	0.667	Y
I_2	0.683	Y	0.778	Y
I_3	0.538	Y	0.667	Y
I_4	0.310	Y	0.222	N
I_5	0.269	Y	0.222	N

表 8 MW_Apriori 与 M_Apriori 的 2-项集支持度对比

项	MM_Apriori 算法		M-Apriori 算法	
	支持度	频繁项集	权重支持度	频繁项集
$I_1 I_2$	0.494	Y	0.444	Y
$I_1 I_3$	0.386	Y	0.444	Y
$I_1 I_4$	0.142	N	0.111	N
$I_1 I_5$	0.269	Y	0.222	N
$I_2 I_3$	0.374	Y	0.444	Y
$I_2 I_4$	0.310	Y	0.222	N
$I_2 I_5$	0.269	Y	0.222	N
$I_3 I_4$	0.000	N	0.000	N
$I_3 I_5$	0.128	N	0.111	N
$I_4 I_5$	0.000	N	0.000	N

从表 7 和表 8 中可以看出,在 1-项集中, I_4 项和 I_5 项不是 M_Apriori 算法的频繁项,而是 MW_Apriori 算法的频繁项;在 2-项集中, $I_1 I_5$ 、 $I_2 I_4$ 和 $I_2 I_5$ 不是 M_Apriori 算法的频繁项,而是 MW_Apriori 算法的频繁项,而这些项可能会出现在最后产生的关联规则中.此外,通过 MW_Apriori 算法,罕见项的支持度有增大趋势,而其他项的支持度略有下降趋势,以前隐藏的、有价值的事件可以被有效地挖掘出来.

通过分析实验结果,改进后的 MW_Apriori 算法具有以下优点:

①MW_Apriori 算法只扫描一次数据库,将事务数据库转换为 0-1 矩阵,而 Apriori 算法在每次产生频繁项集时,都需一次完整的数据库扫描,所以 MW_Apriori 算法较 Apriori 算法可显著降低运行时间,减少项集占用的空间和 I/O 操作次数.

②M_Apriori 算法较 Apriori 算法相比,虽然构建了事务矩阵和频繁状态矩阵,减少了数据库扫描次数,但在递归产生频繁项集时,M_Apriori 算法不仅需要逐行扫描频繁状态矩阵,还需扫描事务矩阵的列来判断是否满足最小支持度,最后还需更新频繁状态矩阵,而本文的 MW_Apriori 算法则只需要扫描事务矩阵的列,计算出权重支持度,这样便缩短了运行时间.

③与 Apriori 算法和 M_Apriori 算法相比,MW_Apriori 算法给项和事务赋予权重,最终使得罕见项的支持度有增大趋势,而其他项的支持度略有下降趋势,之前隐藏的、有价值的事件可以被有效地挖掘出来.

5 结束语

本文基于矩阵和权重的思想,提出了 MW_Apriori 算法.该算法通过扫描一次事务数据库,构造出 0-1 事务矩阵,其中矩阵的行与事务数据库中的事务记录相对应,列则对应该事务记录所包含的项,并给项和事务赋予权重以反应出它们在事务数据库中的重要性,通过计算项的权重支持度,最终得到频繁项集.与 Apriori 算法和 M_Apriori 算法相比,该算法缩短了运行时间,减小了项集占用空间,减少了 I/O 操作次数.同时使得罕见项的支持度有增大趋势,而其他项的支持度略有下降趋势,之前隐藏的、有价值的事件可以被有效地挖掘出来.

参考文献:

[1] Baralis E,Cagliero L,Cerquitelli T, et al. NEMICO: Mining Network Data through Cloud-Based Data Mining Techniques[C]// Utility and Cloud Computing, 2014 IEEE/ACM 7th International Conference on. IEEE, 2014:503-504.

[2] Liao J,Zhao Y,Long S. MRPrePost-A parallel algorithm adapted for mining big data[C]// 2014 IEEE Workshop on Electronics, Computer and Applications. IEEE,2014:564-568.

[3] Muhammad Asif,Jamil Ahmed. Analysis of Effectiveness of Apriori and Frequent Pattern Tree Algorithm in Software Engineering Data Mining[C]// proceed-

ings of 2015 6th International Conference on Intelligent Systems, Modelling and Simulation. Shanghai, 2015: 28-33.

[4] 郑麟.一种直接生成频繁项集的分治 Apriori 算法[J]. 计算机应用与软件, 2014,31(4):297-301.

[5] 付沙,周航军. 关联规则挖掘 Apriori 算法的研究与改进[J]. 微电子学与计算机,2013, 30(9):110-114.

[6] 罗丹,李陶深. 一种基于压缩矩阵的 Apriori 算法改进研究[J]. 计算机科学,2013, 40(12):75-80.

[7] Zhang S, Du Z, Wang J T L. New techniques for mining frequent patterns in unordered trees. [J]. IEEE Transactions on Cybernetics, 2015, 45(6):1113-1125.

[8] Soltani A, Akbarzadeht M R. Confabulation-inspired association rule mining for rare and frequent itemsets [J]. IEEE Transactions on Neural Networks & Learning Systems, 2014, 25(11):83-92.

[9] 周发超,王志坚,叶枫,等. 关联规则挖掘算法 Apriori 的研究改进[J]. 计算机科学与探索, 2015,9(9): 1075-1083.

[10] 吴学雁,莫赞. 基于 Aproiri 算法的频繁项集挖掘优化方法[J]. 计算机应用,2014(6):124-129.

作者简介:

边根庆 男,(1968-),副教授. 硕士生导师,研究方向为信息管理技术.

王 月(通讯作者) 女,(1993-),硕士. 研究方向为信息安全技术.

E-mial:yuewang_711@163. com.

(上接第 135 页)

[6] 周佳宁,李荣宽. 一种低功耗 14 位 10MS/s 流水线 A/D 转换器[J]. 微电子学与计算机. 2012,29(4):49-57.

[7] Wei-Hsuan T, Tzung-Hung K. A 1. 2V 30mW 8b 800MS/s Time-interleaved ADC in 65nm CMOS[C]// IEEE Symposium on VLSI Circuits. Honolulu,2008: 18-20.

[8] Robert L, Markus K, Frank O. Implementation of a high-speed flash ADC for high-performance pipeline ADCs in an 180nm CMOS process [C] // IEEE 2015 ISPACS. Barli,2015:317-322.

[9] 贾华宇. 高性能流水线模数转换及其数字校准技术研究[M]. 西安. 西安电子科技大学出版社,2014.

[10] Sounak R, Sanjoy KD, Swapna B. A parallel pipeline ADC with a Sub-ADC employing an improvized dynamic comparator[C] // IEEE Region 10 Conference. IEEE,2009:1-6.

作者简介:

陈忠学 男,(1989-),硕士研究生. 研究方向为模拟集成电路设计. E-mail:czxue1015@qq. com.

何 全 男,(1990-),硕士研究生. 研究方向为射频模拟集成电路设计.

章国豪 男,(1964-),教授. 研究方向为射频、微波及毫米波单片电路及组件.