



基于位运算和倒排索引的关联规则挖掘算法

黄玉蕾¹, 罗晓霞², 林 青¹

(1. 西安培华学院中兴电信学院 西安 710125;

2. 西安科技大学计算机科学与技术学院 西安 710054)

摘 要:提出了一种改进的 Apriori 关联规则挖掘算法,称为 Apriori-BR。该算法首先通过扫描两次数据库建立各个频繁项目集到事务的倒排索引,并对倒排索引按照事务长度进行分组,然后在挖掘过程中,利用位运算加快子集的检测,并在必要时动态删除无效的低维事务。实验结果表明,相比于经典的 Apriori 算法和已有文献中的改进算法,本文所提的 Apriori-BR 算法显著提高了挖掘效率。

关键词: Apriori 算法;位运算;倒排索引

doi: 10.11959/j.issn.1000-0801.2015230

An Association Rule Mining Scheme Based on Bit Operation and Reverse Index

Huang Yulei¹, Luo Xiaoxia², Lin Qing¹

(1. ZTE Telecommunications College, Xi'an Peihua University, Xi'an 710125, China;

2. School of Computer Science and Technology, Xi'an University of Science and Technology, Xi'an 710054, China)

Abstract: An improved Apriori algorithm for association rule mining called Apriori-BR was proposed, which was based on bit operation and reverse index. Specifically, the reverse index from frequent itemsets to transactions was constructed firstly by scanning twice of database, and the reverse index was grouped by the length of transactions. Then in the mining process, bit operation was adopted to accelerate subset detection together with the dynamical elimination of invalid low-dimensional transactions. The numerical results show that the Apriori-BR proposed can substantially improve mining efficiency when compared with the conventional Apriori algorithm and the improved ones in the literature.

Key words: Apriori algorithm, bit operation, reverse index

收稿日期:2015-07-13;修回日期:2015-09-08

基金项目:陕西省教育厅科学研究基金项目资助项目(No. 15JK2091),国家自然科学基金资助项目(No. 41472234),西安培华学院教育教学改革研究项目(No. PHY1515)

Foundation Items: Scientific Research Fund Project from Education Department of Shaanxi (No. 15JK2091), The National Natural Science Foundation of China (No. 41472234), Educational Teaching Reform Research Project from Xi'an Peihua University (No. PHY1515)

论文引用格式:黄玉蕾, 罗晓霞, 林青. 基于位运算和倒排索引的关联规则挖掘算法. 电信科学, 2015230

Huang Y L, Luo X X, Lin Q. An association rule mining scheme based on bit operation and reverse index. Telecommunications Science, 2015230



1 引言

关联规则挖掘最早是 Rakesh A^[1]等人在 1993 年 SIGMOD 大会上提出的一个数据挖掘的重要研究课题,用于发现隐藏在大型数据集中令人感兴趣的联系。关联规则在仓储规划、销售追加以及基于购买模式的客户划分等方面得到了广泛应用。上述应用场景中数据量极其庞大,因此,如何高效地从数据库中获取潜在信息是关联规则挖掘的关键环节。目前提出的关联规则挖掘算法大致可以分为两类:第一类是 Apriori 算法^[1]以及相应的改进算法,这类算法实现简单,但需要多次扫描数据库;第二类算法是 FP-Growth 算法^[2]以及相应的改进算法,这类算法不产生候选项目集,并采用 FP-Tree 压缩数据库,其算法效率高于 Apriori 算法,缺点是在挖掘大型数据库的过程中产生大量复杂的 FP-Tree 分支,会占据大量的存储空间,而且反复地搜索 FP-Tree 需要大量的指针,内存开销大。本文旨在继承 Apriori 算法简单易行的优点,同时提高其算法的运行效率。

Apriori 算法^[1,3]是关联规则挖掘算法中最具有影响力的算法,其能够在数据库中找出满足用户给定最小支持度和最小置信度要求的关联规则,但是,经典的 Apriori 算法存在以下不足。

- 判断一个候选项目集的子集是否存在时,需要进行多次比较。
- 每个候选项目集都需要全量扫描数据库一次才能获得该候选项目集的支持度。
- 在生成候选项目集过程中,需要进行多次比较以判断连接条件是否满足。

针对经典 Apriori 算法的上述缺点,多种改进措施已被提出:对数解码和剪枝^[4]、减少连接次数^[5]、删减事务^[6]。参考文献[7]提出一种基于 MapReduce 编程模型的分布式关联规则挖掘算法 MCM-Apriori,采用分布式来提升算法的可靠性与性能。参考文献[8]中算法只扫描原始数据库一次,将原始数据库变为压缩矩阵,在挖掘过程中只采用逻辑运算,提升了挖掘效率。参考文献[9]中算法在挖掘过程中生成二进制形式的候选项目集,采用和 Apriori 一样的搜索策略,能很好地胜任短频繁项目集的挖掘。参考文献[10]基于 AFOP 树型结构,提出一种自顶向下的多层关联规则挖掘算法,在挖掘过程中寻找每个项集的前缀时,可以从 AFOP 树中快速获取到关键信息,从而有效地节约了计算开销。目前已有的改进算法^[4-10]仅仅针对 Apriori 算法的单

一缺陷进行改进的,提升效果有限,因此,本文提出一种基于位运算和倒排索引的 Apriori-BR 算法,其不但减少数据库的扫描次数^[4-6],而且降低了候选项目集的比较次数^[7-10],其全面考虑了经典 Apriori 算法的上述 3 方面缺陷,从而最大程度提升了挖掘效率。

2 关联规则挖掘算法

2.1 基本概念

关联规则挖掘问题^[3]定义为:设 $I=\{i_1, i_2, \dots, i_N\}$ 是包含 N 个不同项目的集合,简称为项集。 $D=\{d_1, d_2, \dots, d_M\}$ 是包含 M 个事务的数据库,其中每个事务由若干个属于 I 的项目组成,不同的事务通过唯一的 T_{id} 进行标识,对于事务 d_m 而言,用 $J(d_m)$ 表示其包含的项集,用 $T_{id}(d_m)$ 表示其唯一的标识,用 Y 表示 D 中所有事务包含的项目集构成的集合,即 $Y=\bigcup_{n \in \{1, 2, \dots, M\}} J(d_n) \subseteq I$ 。为了方便后续的描述,首先给出以下定义。

定义 1 关联规则表示为 $X \Rightarrow Y$ 的蕴涵式,这里 $X \subseteq I, Y \subseteq I, X \cap Y = \emptyset$ 。 $X \Rightarrow Y$ 的支持度与置信度分别定义为: $S(X \Rightarrow Y) = P(X \cup Y)$ 和 $C(X \Rightarrow Y) = P(Y|X)$,其中 $P(\cdot)$ 运算是求取相应事件的概率。支持度和置信度是关联规则中的两个指标,代表规则的有用性和确定性。

定义 2 如果项集 $X \subseteq J(d_m)$,则称事务 d_m 包含项集 X 。

定义 3 项集 X 在 D 中的支持数 $A(X)$ 是 D 中包含 X 的事务数目,即 $A(X) = |\{d_m \in D | X \subseteq J(d_m)\}|$,其中 $|\cdot|$ 运算是求取集合中元素的数目,而项集 X 在数据库 D 中的支持度 $S(X)$ 则为上述支持数与 D 中事务总数目的比例,即 $S(X) = A(X)/M$ 。

定义 4 对于项集 X ,如果 X 中包含有 k 个项目,即 $|X|=k$,则称项集 X 为 k -项集。

定义 5 当项集 X 的支持度 $S(X)$ 满足预定义的最小支持度阈值 θ 时,即 $S(X) \geq \theta$,则称 X 是频繁项集。与此同时,若项集 X 为 k -项集,则称其为 k -项频繁项集。

2.2 经典的 Apriori 算法

Apriori 算法是一种自底向上的规则挖掘方法,算法的基本思想是:找出所有出现的频繁项集;利用频繁项集产生满足最小支持度的关联规则;筛选出其中满足用户给定最小置信度的规则。具体地说,Apriori 算法利用逐层搜索的迭代方法找出所有满足最小支持度和最小置信度的频繁项集,即利用 k -项频繁项集间两两比较、判断、连接产生 $(k+1)$ -项候选项集,并通过扫描数据库计算候选项集的

支持度,根据最小支持度筛约束选出候选项集中 $(k+1)$ -项频繁项集。为了提高效率,Apriori 算法利用频繁项集的重要性质:频繁项集中任何子集均为频繁项集,从而可以降低算法的搜索空间。其详细步骤见参考文献[1]。

经典的 Apriori 算法存在以下不足之处:(1) 判断一个候选项集的子集是否存在时,需要进行多次比较;(2) 每个候选项集都需要全量扫描数据库一次才能获得该候选项集的支持度;(3) 在生产候选项目集过程中,需要进行多次比较以判断连接条件是否满足。

3 基于位运算和倒排索引的改进算法 Apriori-BR

3.1 算法改进思想

为了更好地说明本算法的改进思想,在算法描述中结合具体的事务数据库进行说明,见表 1。设最小支持度阈值为 50%,即最小支持数为 3,则 1-项频繁项集为 $L_1=\{f(5),c(4),d(4),b(3),e(3)\}$,其中括号内数字表示所对应的项目在 D 中所有事务中出现的总次数,即项目的支持数。

表 1 事务数据库 D

T_{id}	包含项目	T_{id}	包含项目	T_{id}	包含项目
1	$c\ d\ f\ p$	2	$b\ c\ d\ e\ f$	3	$d\ e\ f\ q$
4	$b\ f\ r$	5	$c\ d\ f\ s$	6	$b\ c\ e\ t$

3.1.1 位运算

针对经典 Apriori 算法中的问题(1),引入了位运算操作来减少在候选项集子集的判断过程中所需的比较次数,从而提升候选项集的子集检测效率。

为了实现上述目标,对于事务数据库 D 中的任意项目 $i_n, n \in \{1, 2, \dots, N\}$ 定义了一个模为 M 的两元列向量 $R_n=\{t_{1n}, t_{2n}, \dots, t_{Mn}\}^T$,其中 M 表示数据库 D 中的事务总数,如果事务 d_m 中包含项目 i_n ,则 $t_{mn}=1$,否则 $t_{mn}=0$ 。引入上述两元向量存储后,可利用位运算来进行支持数和子集检测的运算。比如,当计算项目 i_j 和 i_k 的支持数,其等价于计算向量 $R_j \& R_k$ 结果中包含“1”的个数,后续使用 $(R_j \& R_k).count$ 代表上述操作,此外 $\&$ 表示按位与运算。而当判断向量 R_j 是

否为 R_k 的子集时,其等价于判断 R_j 是否等于 $R_j \& R_k$ 。

3.1.2 倒排索引

传统的数据库一般采用正向索引,以表 1 中的数据库为例,对于 $T_{id}=1$ 的事务而言,可以获得相应的项目 c, d, f, p 。采用这种方式,在计算候选频繁项目的支持度时,由于无法知道每个项目是否包含在某个事务中,必须对数据库进行全库扫描,并对所有事务进行逐个比对,极大地增加了算法的复杂度,即经典 Apriori 算法中的问题(2)。假设数据库中事务的平均长度为 L ,当有 C 个候选项目时,计算支持度所需的算法复杂度为 $O(M \cdot L \cdot C)$ 。因此,本文引入倒排索引技术,对每个项目,获取其对应的 T_{id} ,并按 T_{id} 对应事务的项目长度进行分组,建立倒排索引。表 1 事务数据库 D 所对应的倒排索引如图 1 所示。

本文构建的倒排索引由 3 部分组成:第一部分是各个频繁项目的位权,用于标记这个频繁项目用二进制位表示时的位置;第二部分是频繁项目头节点,其内容包括频繁项目及其支持数(项目后括号内的数字);第三部分是倒排索引节点,由项目长度(仅涵盖频繁项目)和事务 T_{id} 列表组成。对 T_{id} 按照事务长度进行分桶保存。采用倒排索引,对于包含个项目的候选集,每个候选集只需扫描长度为 N 的头节点表,并取出头节点表中事务集合,设头节点对应事务的平均个数为 K ,则算法复杂度为 $O(K \cdot N \cdot C)$ 。虽然 $N > L$,但由于 $K \ll M$,采用倒排索引将大大提升扫描数据库效率。

3.1.3 快速连接

Apriori 算法在执行连接步骤生成 $(k+1)$ -项候选项集 L_{k+1} 的过程中,需要从 L_k 中找到两个项集 S_1 和 S_2 ,记符号 $S_1[t]$ 代表 S_1 的第 t 项,如果 S_1 和 S_2 的前 $k-1$ 项均相同,并且满足 $S_1[k] < S_2[k]$,则表示 S_1 和 S_2 是可连接的,连接 S_1 和 S_2 产生的项集是 $S_1 \triangleright S_2 = \{S_1[1], S_1[2], \dots, S_1[k], S_2[k]\}$ 。从这个过程中可以看到,为了生成一个 $(k+1)$ -项候选项集,最优的情况需要比较 k 次,最差的情况是一个 k -项频繁项目项和 L_k 匹配失败,而该频繁项目项需要与 L_k 中每个项集都进行 k 次比较,算法的效率很低,即经典 Apriori 算法中的问题(3)。

为了提升效率,将 k -项频繁项集分为两部分进行存

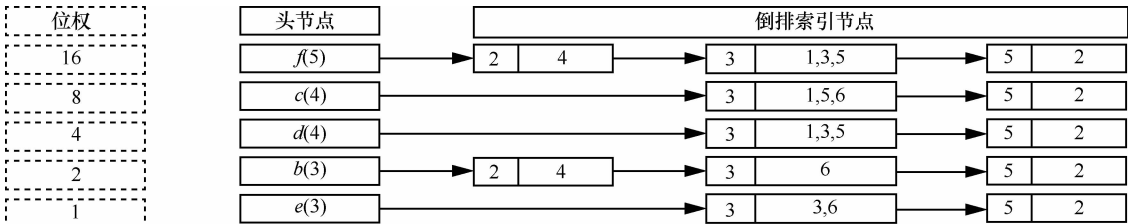


图 1 表 1 中事务数据库所对应的倒排索引



储,两部分结构分别命名为 first 和 second,其中 first 存储前 $k-1$ 项的向量表示形式,second 存储第 k 项。例如对频繁项集 $A=\{f,d,b\}$,根据图 1 中的位权信息可知,其 first 为 (10100),second 为 2(位权);频繁项集 $B=\{f,d,e\}$,则 first 为 (10100),second 为 1(位权);进行连接时,如果两个项集的 first 相等(编程语言支持对两个二进制位串的比较),并且 second 的位权不相等,则直接生成一个新的候选项集,其中 second 存储两相集中位权较小 second。在此例中新的候选项集是 $\{f,d,b,e\}$,其 first 为 (10110),second 为 1(位权)。采用基于二进制位串表示方式后,在生成候选项集时,对前面部分比较只需一次运算即可完成,无需对每个项进行单独的查找、比较,从而提高连接效率。

3.2 相关性质和优化策略

性质 1 在事务数据库 D 中若有一事务 T ,其长度小于 $(k+1)$,则对于长度大于 k 的候选项目集,事务 T 无需扫描。

性质 2 一个项集 X 在数据库中的支持数,等于这个项集的所有项的列向量,即 $R_x, \forall x \in X$,按位“与”操作之后“1”的个数。

由性质 1 和性质 2,提出如下的优化策略。

策略 1 列向量生成过程中,在挖掘 k -项频繁项目集时,只需要从每个项目的事务长度不小于 k 的倒排索引列表中生成。

策略 2 在计算 k -项候选项集的支持数时,如果在某两个项的列向量进行按位“与”运算之后“1”的个数小于 k ,则这个候选项的支持数小于 k ,不是频繁项目集,可以直接删除。

策略 3 采用位运算加速连接运算。

3.3 算法描述

Apriori-BR 算法的处理流程为:全量扫描数据库一次,获得 1-项频繁项集 L_1 ,再次扫描数据库,生成倒排索引。然后执行如下步骤:从 $k=1$ 开始,迭代产生长度为 $k+1$ 的候选项集的集合 C_{k+1} ,每个迭代包含以下两步:第一步, L_k 自连接,从 L_k 中选择项集连接生成长度为 $k+1$ 的候选项集,参与连接的项集必须满足 $(k-1)$ -前缀相同,即 GenCandidate 子函数。第二步是剪枝,在将候选项目集加入 C_{k+1} 时,如果其 k 个长度为 k 的子集不都在 L_k 中,则删除此候选项目集。在计算长度为 $k+1$ 的候选项集的支持度时,采用第 3.2 节策略 2 进行剪枝,即图 2 主函数中 for 循环内部的 while 和 if 语句部分。剪枝完毕后,挖掘出所有 $(k+1)$ -项频繁项集被加入 L_{k+1} 中。此过程直至 L_{k+1} 等于空

集时结束。

输入:事务数据 D ,最小支持数阈值 θ 。

输出:频繁项集 L 。

扫描数据库两次,得到 L_1 并建立从频繁项目集到事务的倒排索引;初始化 $L=\emptyset, k=1$;

While 1

 初始化 $L_{k+1}=\emptyset$, 生成候选项目集 $C_{k+1}=\text{GenCandidate}(L_k)$;

 For 遍历 C_k 中的所有元素 $c \in C_{k+1}$

 生成候选项集的向量形式,即对于 $\forall c \in C_{k+1}$, 得到 R_c 。

$v=R_c[1], n=1$;

 While($n < |R_c|-1$ && $(v, R_c[n+1]).count \geq \theta$)//其中, $v.count$ 表示向量 v 中元素为“1”的个数

$n++$;

 if $n == |R_c|-1$ && $v.count \geq \theta$ //判断候选项目 $c \in C_{k+1}$ 是否为频繁项目

$L_{k+1}=L_{k+1} \cup \{c\}$;

 If $L_{k+1} == \emptyset$

 break;

 Else

 更新 $L=L \cup L_{k+1}, k=k+1$;

Function $C_{k+1}=\text{GenCandidate}(L_k)$ //生成候选集

 初始化 $C_{k+1}=\emptyset$;

 For 对于 L_k 中每个项目 $i_1 \in L_k$

 For 对于 $L_k-\{i_1\}$ 中每个项目 $i_2 \in L_k$

 If($i_1.first == i_2.first$ and $i_1.second < i_2.second$

$c=i_1 \triangleright i_2$;//采用第 3.1.3 节的快速连接;

 If check(c, L_k)

$C_{k+1}=C_{k+1} \cup \{c\}$;

Function check(c, L_k)//判断是否应该动态删除该低维事务

 For 对于集合 c 中所有包含 k 个元素的子集 s

 If $s \notin L_k$

 Return false;

 Return true;

3.4 算法演示

针对表 1 中的数据库,其关联规则挖掘过程如下:首先扫描 2 次数据库,生成如图 1 所示的倒排索引,并获得 1-项频繁项目集 $L_1=\{f(5),c(4),d(4),b(3),e(3)\}$ 。然后开始连

接,生成 2-项候选项目集 C_2 。在生成 C_2 时,以项目 f 和 c 为例。 $f.first=(0\ 0\ 0\ 0\ 0)$, $f.second=16$, $c.first=(0\ 0\ 0\ 0\ 0)$, $c.second=8$, 满足 $f.first=c.first$, 则连接生成 2-项候选项目集 $\{fc\}$, $\{fc\}.first=(1\ 0\ 0\ 0\ 0)$, $\{fc\}.second=8$ 。同理,生成其他所有候选项目集 $C_2=\{fc,fd,fb,fe,cd,cb,ce,db,de,be\}$ 。在计算支持度时,以 $\{fc\}$ 和 $\{fb\}$ 为例进行说明,对于 $\{fc\}$ 而言,项目 f 的倒排索引是(4,1,3,5,2),相应的列向量是 $R_1=(1\ 1\ 1\ 1\ 1\ 0)^T$,项目 c 的倒排索引是(1,5,6,2),相应的列向量是 $R_2=(1\ 1\ 0\ 0\ 1\ 1)^T$, $R_1 \& R_2=(1\ 1\ 0\ 0\ 1\ 0)^T$, $R_1 \& R_2$ 结果中“1”的个数,即 $(R_1 \& R_2).count$, 为 $3 \geq M \cdot \theta = 3$, 属于频繁项目;而对于 $\{fb\}$ 而言,项目 b 的倒排索引是(4,6,2),对应的列向量是 $R_3=(0\ 1\ 0\ 1\ 0\ 1)^T$, $R_1 \& R_3=(0\ 1\ 0\ 1\ 0\ 0)^T$,不同于 $\{fc\}$ 中的情况,此处 $(R_1 \& R_3).count=2 < M \cdot \theta = 3$,故不是频繁项目,则删除。依次进行挖掘,获得 2-项频繁项目集 $L_2=\{fc(3),fd(4),cd(3)\}$;以此方法挖掘出 3-项频繁项目集 $L_3=\{fcd(3)\}$,算法结束。

4 实验结果与分析

为了测试优化算法的性能,在 CPU i5-4200u 2.5 GHz、2 048 MB DDR RAM、Windows XP Professional SP2 的环境下进行测试,用 VC++6.0 实现了经典的 Apriori、Apriori-B、Apriori-R 和 Apriori-BR 4 种算法,其中 Apriori-B 表示仅仅在经典的 Apriori 算法中加入了位运算和快速连接,其类似于参考文献[7~10]中算法的效果,而 Apriori-R 表示仅仅在经典的 Apriori 算法中加入了倒排索引,其类似于参考

文献[4~6]中算法的效果。实验数据分两种,真实数据(accidents 和 mushroom)以及合成数据(T70I20D100K),其中,真实数据取自 FIMI 的数据库^[11],而合成数据则来自 IBM 公司 Almaden 实验室的生成器。Accidents 数据集包含 1991-2000 年 Flanders 地区的交通事故记录。该数据集的大小为 34 678 KB,共 340 184 条事务,有 572 个项目,平均事务长度为 45。Mushroom 为 557 KB,共有 8 124 条事务,有 119 个项目,平均事务长度为 23。T70I20D100K 中共 100 000 条事务,有 70 个项目,平均事务长度为 20。

从实验结果可以看出,同一数据集上,算法的运行时间随最小支持度的变小而增长,这是由于最小值支持度变小会使得候选项集数目增加而导致的。在图 2 中可以看出,传统 Apriori 算法的运行时间在最小支持度较小时,比 Apriori-B 和 Apriori-R 两个算法大很多,原因在于当支持度较小时,Apriori 生成大量的候选项目集,且每个候选项集都需要全量扫描数据库才能获得支持度,将大量时间消耗在候选项集的连接,子集测试和支持度的获取中。对比图 2(a)与图 2(c)发现,在 accidents 数据集上 Apriori-B 算法运行时间略小于 Apriori-R,而在 T70I4D100K 数据集上运行结果相反,原因在于 accidents 数据集中项目数相对较多,候选项集数目巨大,而候选项集间连接消耗大量时间,Apriori-B 算法优化效果明显,而当事务数相对较少,扫描数据库次数远小于连接次数时,Apriori-R 算法优化效果有限。在 T70I4D100K 数据集中,项

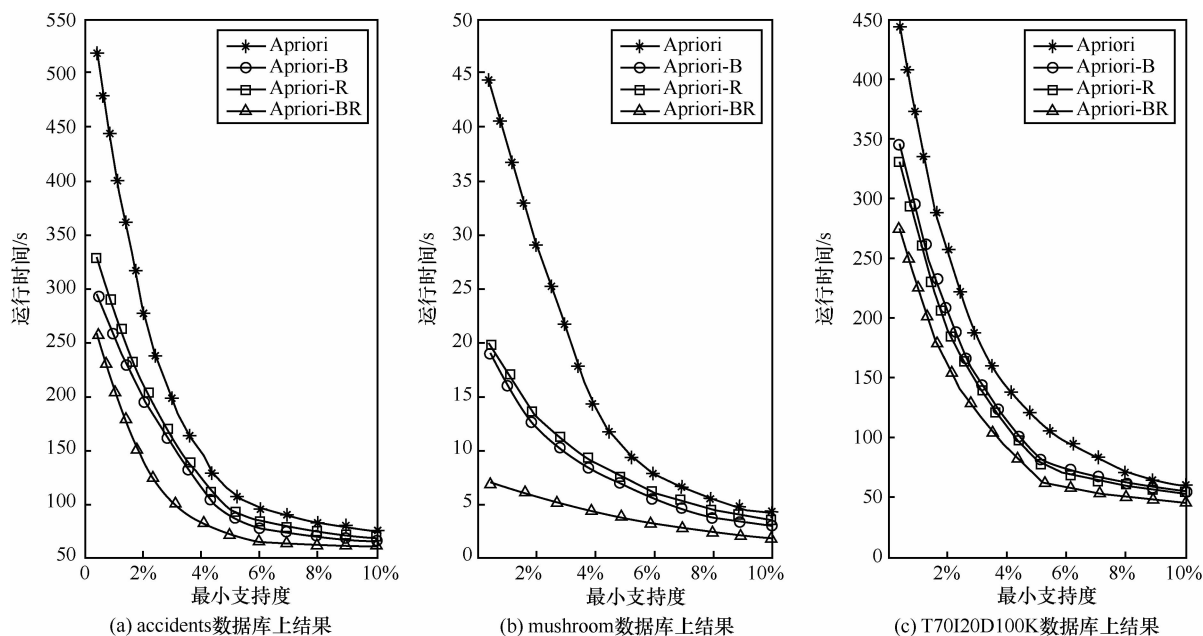


图2 不同算法运行时间随最小支持度需求的对比结果



目较少,候选项集数目相对较少,而数据库中事务较多,因此,Apriori-R 算法相对于 Apriori-B 算法取得了更好的性能。在图 2(b)中,由于数据集规模较小,运行时间远小于其余两图,并且由于项集较少,Apriori-BR 算法在候选项集间连接、扫描数据库的优化效果相较于其余两子图来说更好。综合图 2 中 3 组对比可以看出,Apriori-BR 算法结合了 Apriori-B 与 Apriori-R 算法,在不同数据集中都取得了较高的挖掘效率。

5 结束语

本文通过对经典 Apriori 算法的分析,提出了一种基于位运算和倒排索引的频繁项集挖掘算法 Apriori-BR。经典的 Apriori 算法需要反复扫描数据库以便计算候选项集的支持度,并且在生成候选项目集时也需要反复对比判断所生成的项目是否存在,影响了算法的效率。本文所提的 Apriori-BR 算法在挖掘过程中只需要扫描 2 次数据库,用于获得 1-项频繁项集,并建立频繁项集到事务的倒排索引。利用位运算加速子集的检测,提升连接速度。在获取候选项集的支持度时,无需再次扫描数据库,只需对候选项集中各个项目的二元列向量进行位运算,并且能偶在计算过程中及早发现非频繁项目,提前终止剩余运算,从而进一步提升了算法效率。最终,不同数据库下的仿真结果充分验证了 Apriori-BR 算法的优势和效果。

参考文献

- 1 Rakesh A, Tomasz I, Arun S. Mining association rules between sets of items in large databases. *Proceedings of ACM SIGMOD 1993 International Conference on Management of Data*, New York, USA, 1993: 207~216
- 2 Han J W, Pei J, Yin Y W. Mining frequent patterns without candidate generation. *Proceedings of ACM SIGMOD 2000 International Conference on Management of Data*, Dallas, TX, USA, 2000: 1~12
- 3 Wur S Y, Leu Y H. An effective Boolean algorithm for mining association rules in large databases. *Proceedings of the 6th International Conference on Database Systems for Advanced Applications (DASFAA '99)*, Hsinchu, China, 1999: 179~186
- 4 Nagpal S. Improved Apriori algorithm using logarithmic decoding and pruning. *International Journal of Engineering Research and Applications (IJERA)*, 2012, 2(3): 2569~2572
- 5 Jiao Y B. Research of an improved apriori algorithm in data mining association rules. *International Journal of Computer and Communication Engineering (IJCCE)*, 2013, 2(1): 25~27
- 6 Singh J, Ram H, Sodhi D J S. Improving efficiency of Apriori

algorithm using transaction reduction. *International Journal of Scientific and Research Publications*, 2013, 3(1): 1~4

- 7 郭健, 任永功. 云计算环境下的关联挖掘在图书销售中的研究. *计算机应用与软件*, 2014, 31(11): 50~53
Guo J, Ren Y G. Research on association rule mining in book sales under cloud computing environment. *Computer Applications and Software*, 2014, 31(11): 50~53
- 8 刘敏娴, 马强, 宁以凤. 基于频繁矩阵的 Apriori 算法改进. *计算机工程与设计*, 2012, 33(11): 4235~4239
Liu M X, Ma Q, Ning Y F. Improved Apriori algorithm based on frequent matrix. *Computer Engineering and Design*, 2012, 33(11): 4235~4239
- 9 陈耿, 朱玉全, 杨鹤标等. 关联规则挖掘中若干关键技术的研究. *计算机研究与发展*, 2005, 42(10): 1785~1789
Chen G, Zhu Y Q, Yang H B, et al. Study of some key techniques in mining association rule. *Journal of Computer Research and Development*, 2005, 42(10): 1785~1789
- 10 毛宇星, 陈彤兵, 施伯乐. 一种高效的多层和概化关联规则挖掘方法. *软件学报*, 2011, 22(12): 2965~2980
Mao Y X, Chen T B, Shi B L. Efficient method for mining multiple-level and generalized association rules. *Journal of Software*, 2011, 22(12): 2965~2980
- 11 Frequent Itemset Mining Dataset Repository. <http://fimi.ua.ac.be/data/>, 2013

[作者简介]



黄玉蕾,女,西安培华学院讲师,主要研究方向为云计算、大数据。



罗晓霞,女,西安科技大学副教授,主要研究方向为数据库、软件工程及软件开发。



林青,女,西安培华学院讲师,主要研究方向为计算机网络、仿真。