一个基于剪枝的非递归生成关联规则的算法

陈 耿[1,5],郭 兵[2],文登敏[3],黄 倩[4],邓 莉[1]

(1.成都师范学院 计算机科学学院,成都 611130;
2.四川大学 计算机学院,成都 610065;
3. 西南交通大学 信息科学与技术学院,成都 610031;
4. 四川大学 华西公共卫生学院,成都 610041;
5. 成都师范学院 计算智能与信息技术研究所,成都 611130) *

摘 要: 关联规则挖掘是最常用的数据挖掘方法之一,通常采用基于频繁项集的挖掘技术来进行。在得到频繁项集之后,如何快速产生关联规则也是影响关联规则挖掘耗时的一个重要因素。首先介绍了在事务数据集上基于 Apriori 算法的频繁项集挖掘算法实现方法,然后说明了由频繁项集生成关联规则的基本过程,分析了基于剪枝的关联规则生成算法对效率提升的原理。然后,提出了一种非递归的、基于剪枝的由频繁项集生成关联规则的方法。在同样的软硬件环境下,经过在多个阶的频繁项集上进行对比实验,新方法能正确完成关联规则生成,并且执行的时间耗费相比原来的方法大约平均减少了 6%左右。

关键词: 数据挖掘;频繁项集;关联规则;规则生成;剪枝

doi:10.3969/**j.issn.**2095-5642.2018.11.083

中图分类号:TP301.6 **文献标志码**:A **文章编号**:2095-5642(2018)11-0083-09

1 引言

关联规则挖掘是经常用到的一种数据挖掘方法,可以从大量的历史数据里发现很多意想不到的有趣的模式。关联规则反映一个事物(或事件)与其它事物(或事件)之间的共同存在的模式、规律。如果两个或两个以上事物之间存在一定的关联关系,那么,通过检测到一个或多个事物(或事件)的存在,就能够预测其他事物(或事件)也以一定的概率存在或发生。虽然关联规则并不代表事物或事件之间存在因果关系,例如:公鸡鸣叫和天明经常同时发生,不能说天明是由于公鸡鸣叫引起的,但是,公鸡鸣叫可以预告天明这个事件会发生。

这些共同存在的有趣的模式,在专业人员的识别和进一步加工后可以成为“知识”,帮助制定不同的工作规则,指导今后的工作,从而提升工作效果、降低工作成本。关联规则的使用可以用于安全入侵检测、疾病预测、公共事务预测、广告推广甚至金融风险防范等众多领域。

面对海量的历史数据,通常,大家更关注于如何快速挖掘出频繁项集。经典的频繁项集挖掘有两大类方

* **收稿日期**:2018-05-16

基金项目:四川省科技厅应用基础研究项目“基于符号计算的程序形式化验证”(2014JY0030);四川省教育厅理工类重点项目“基于符号计算的程序验证及其应用”(14ZA0301);成都师范学院 2017 年国家级大学生创新创业训练项目“智能商品推介软件”(201714389006)

作者简介:陈 耿(1974—),男,四川遂宁人,副教授,硕士,研究方向:数据挖掘、电子政务;
文登敏(1962—),男,山西运城人,副教授,硕士生导师,硕士,研究方向:软件工程、软件构件技术等;
郭 兵(1970—),男,山东泰安人,教授,博士,博士生导师,研究方向:嵌入式实时系统、绿色计算、大数据等;
黄 倩(1979—),女,四川成都人,博士研究生,讲师,研究方向:公共卫生数据分析;
邓 莉(1995—),女,陕西汉中人,本科,研究方向:计算机应用技术。

式：Apriori 和基于它的改进算法；FP－树关联规则挖掘算法。

然而，当频繁项集包含的项的数量很庞大的情况下，特别是包含了比较长的频繁项的情况下，如何由已经得到的频繁项集快速产生关联规则，也是制约关联规则生成速度的不可忽略的因素。

2 关联规则相关名词

2.1 频繁项集与关联规则

关联规则是指在数据集中支持度和信任度分别满足给定阈值的规则，通常记作：

$$A \Rightarrow B[\text{Support}=\text{sup}\%, \text{Confidence}=\text{conf}\%] \quad (1)$$

在这里，把关联规则中的 A 称为前件或左部(LHS)，B 称为后件或右部(RHS)。与关联规则密切相关的两个度量是：支持度和置信度。

支持度是前件和后件同时存在的数量在整个数据中所占的比例。

置信度是前件和后件同时存在的事务数量在有前件存在的事务数量中所占的比例，即可以理解为前件存在的时候后件也同时存在的比例。

如果不考虑一定的支持度和置信度要求，那么可以得到庞杂的关联规则，因此只有满足最小支持度阈值和最小置信度阈值的关联规则才是我们感兴趣的强关联规则，才具有实际参考价值。

设 $I=\{i_1, i_2, \cdots, i_n\}$ 是项的集合，I 包含有 n 个元素就称为 n 阶项集。如果 $(A \cup B) \subseteq I$，且 $(A \cup B)$ 存在的数量与整个数据事件的数量之比大于等于最小支持度，称 $(A \cup B)$ 为频繁项集，其阶数由 $(A \cup B)$ 包含的频繁项数量决定。因此，所有强关联规则的前件与后件的并集必然是频繁项集。由此，也可以知道，高阶频繁项集的全部子集必然都是频繁项集。Apriori 算法就是利用这个道理，逐步由 1 阶频繁项项集依次产生高阶频繁项集。文献[1]提出的 Apriori 方法是最早提出的频繁项集挖掘方法，避免了枚举所有低阶频繁项集的组合来产生高 1 阶的频繁项集，极大地避免了无谓的时间耗费，并且这个剪枝的思想在后续生成规则的时候还将继续作用，如图 1。

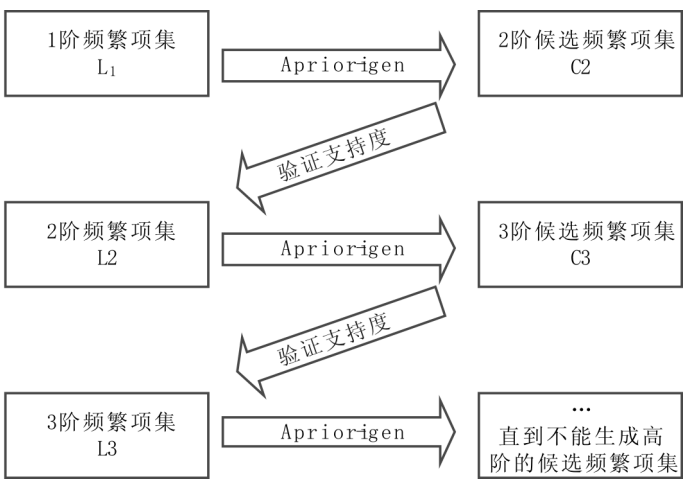

图 1　频繁项集的生成流程

2.2 主－从表型数据的频繁项集挖掘

在当前学习数据集中，数据是以主－从表形式存放，以事务标识 TID 进行关联，形式如图 2。

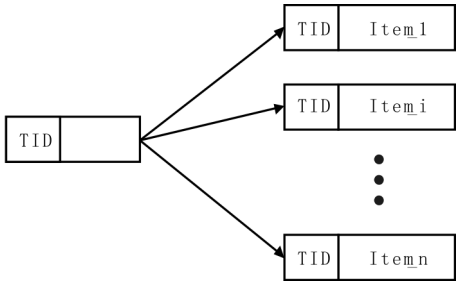

图 2　数据形式

因此,需要把纵向存储的项集转换为适合 Apriori 算法处理的横向存储形式。

同时,为了减少计算消耗,不必每次去计算候选频繁项集的支持度,而是只计算一次最小支持计数。设支持度最小支持计数 ,对于每个候选候选频繁项集,只需要利用其在数据集中出现的计数是否满足来判断是否是频繁项集,而不必每次都去进行除法计算。

Apriori 算法的总体思想是基于利用 1 阶频繁项集 L1 去产生 2 阶候选频繁项集 C2;从 C2 中找出满足支持度要求的 2 阶频繁项集 L2;依次类推,逐步由低阶频繁项集产生高阶候选频繁项集,验证后得到高阶频繁项集,直到频繁项集为空。挖掘频繁项集的主流程如图 3。

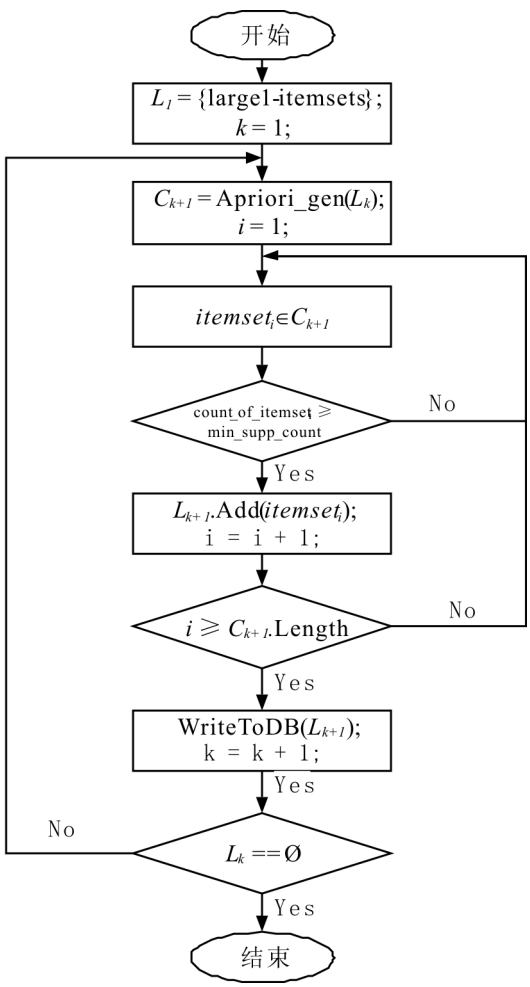

图 3　生成频繁项集的主流程

其中，Apriori_gen(L_k)的处理流程如图 4，返回(k+1)阶候选频繁项集 C_{k+1}。

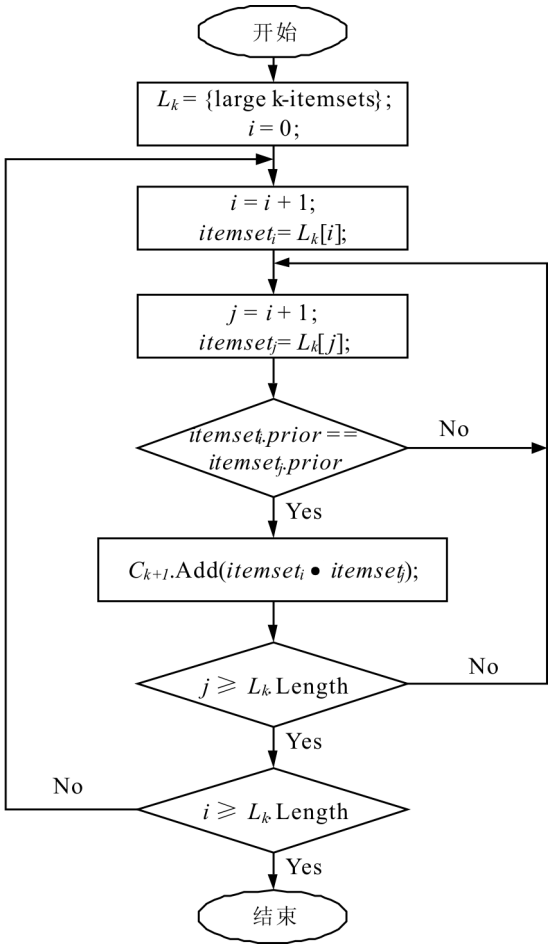

图 4　Apriori_gen 的处理流程

根据文献[1]，由低阶频繁项集拼接成高阶候选频繁项集的规则为：(1)1 –阶频繁项集直接两两拼接成 2 –阶候选频繁项集；(2)对于 k –阶频繁项集($k \geqslant 2$)，只有前 k –1 项是相同的频繁项集才能拼接 k –阶候选频繁项集，且 $itemset_i$ • $itemset_j$ 的运算方法为：$itemset_i$ 连接 $itemset_j$ 的最后一个项。

3 由频繁项集生成关联规则的基本原理

如前述，生成关联规则的基本方法是把一个频繁项集的元素分成两个部分(前件和后件)，计算前件发生的情况下后件同时发生的条件概率，表达形式为：$a \Rightarrow (l - a)$。这里 l 就是挖掘出来的频繁项集，且至少是 2 阶的频繁项集；a 为关联规则的前件，是频繁项集 l 的非空子集($a \subset l$)；$(l - a)$是关联规则的后件，是 l 中不包括 a 集合元素的子集。关联规则解读为：在 a 出现的时候$(l - a)$也以一定的机率同时出现。

我们真正需要的关联规则应该是强关联规则，不仅事件 a 和$(l - a)$在整个学习数据空间出现的机率要足够大(即必须均是频繁项集)，并且前件出现的时候后件出现的机率也要足够大，即关联规则的前部和后部两部分频繁项集同时出现的机率必须大于等于阈值要求。这个条件可以表述为置信度要求。可以用条件概率的计算公示来表示两部分同时发生的机率，置信度计算方法为：

$$\text{conf} = \frac{\text{support}(1)}{\text{support}(a)}, \text{且 conf} \geqslant \text{conf}_{\min}。$$

因此,生成关联规则的办法就是为枚举所有频繁项集 $l_k(k>1)$ 的非空子集 a,产生候选关联规则的前件 a 和后件$(l-a)$,依次去验证置信度是否满足阈值要求。

当频繁项集的阶数较高时,会产生大量的候选关联规则。k－阶频繁项集的全部候选关联规则的数量与阶数是指数函数关系,达到$(2^{10}-2)$个(排除前件为空和前件为整个频繁项集这两个情况)。假如一个频繁项集有 10 个频繁项元素,那么总共的前件和后件的组合数量将是:$2^{10}-2=1022$(个)。

如何提前筛选出不可能满足置信度阈值要求的候选关联规则,避免不必要的关联规则置信度计算,是提高生成规则的速度的主要手段。在数据项规模大的时候,会带来显著的效率提高。Rakesh Agrawal 等在文献[1]中提出了利用频繁项属性进行剪枝的规则生成算法,其原理如下:

设:l_k 是 k -阶频繁项集,a 是 l_k 的非空子集,$\tilde{a}$ 是 a 的非空子集$(\tilde{a}\subset a)$。

因为 $\mathrm{Support}(a)\leq\mathrm{Support}(\tilde{a})$,可以得到 $\dfrac{\mathrm{Support}(l)}{\mathrm{Support}(a)}\geq\dfrac{\mathrm{Support}(l)}{\mathrm{Support}(\tilde{a})}$。即可以得出结论:若 $a\Rightarrow(l-a)$ 不成立,那么 $\tilde{a}\Rightarrow(l-\tilde{a})$ 也不会成立[1]。

由此,可以从 l_k 的$(k-1)$－阶子集开始枚举,当高阶的前件能够成立,再去枚举更低阶的前件,从而避免无谓的计算。

例如:对于 4 -阶频繁项$\{a,b,c,d\}$,当 3 -阶前件规则$\{b,c,d\}\Rightarrow\{a\}$成立之后,再去验证$\{b,c,d\}$的 2 -阶子集为前件的规则:$\{c,d\}\Rightarrow\{a,b\}$、$\{b,d\}\Rightarrow\{a,c\}$、$\{b,c\}\Rightarrow\{a,d\}$是否成立。

4 LoopGenRules 规则生成算法

4.1 算法原理

按照第 2.1 节介绍的生成规则的算法,在使用中发现仍然会产生重复的规则,说明进行了重复的规则验证,还有继续优化的必要。例如 4 -阶频繁项集$\{a,b,c,d\}$,规则$\{b,c,d\}\Rightarrow\{a\}$、$\{a,b,c\}\Rightarrow\{d\}$都成立的情况下,都会去验证$\{b,c\}\Rightarrow\{a,d\}$这个规则。这就造成了时间上的多余耗费。

考虑到 l 是频繁项集,a 是 l 的非空子集,$\tilde{a}\subset a$,则有:必须$(l-\tilde{a})\Rightarrow\tilde{a}(l-\tilde{a})\Rightarrow\tilde{a}$才会有$(l-a)\Rightarrow a$ 成立,原理如下:

设$\tilde{a}\subset a$,那么有 $support(l-a)\geq support(l-\tilde{a})$,从而得到 $\dfrac{support(l)}{support(l-\tilde{a})}\geq\dfrac{support(l)}{support(l-a)}$。

因此,若低阶的后件不能生成强关联规则的话,那么包含该低阶频繁项集的更高阶的频繁项集做后件,也不可能生成强关联规则。据此,可以避免验证重复的后件,提高规则的生成效率。

由此,可以用频繁项集 l_k 的 1－阶子集做后件开始,验证规则是否成立,删除不能生成强规则的 1－阶子集,把能形成强规则的 1－阶子集保留下来。继续仿造 Apriori 算法拼接 2－阶候选子集,验证后删除不能生成强规则的 2－阶子集,保留能生成强规则的 2－阶子集,以便继续生成更高一阶的子集。依此方法递归生成关联规则,直到后件集合为空或到 k－阶停止。

因为是从 1－阶频繁项子集逐步验证到高阶频繁项子集,这种方法可以避免重复验证候选关联规则。

在文献[1]中,给出了基于递归的逐步从低阶频繁项子集拼接到高阶频繁项子集的基本算法。我们进一步提出了基于循环的快速关联规则生成算法——LoopGenRules,避免递归方式算法的运行空间分配与回收带来的效率损耗。其基本处理思路为:利用当前频繁项集的低阶子集的元素,验证是否能产生强关联规则,从当前频繁项集合里剔除不能产生强规则的元素,再去生成更高一阶的频繁项集。LoopGenRules 的基本处理流程如图 5。

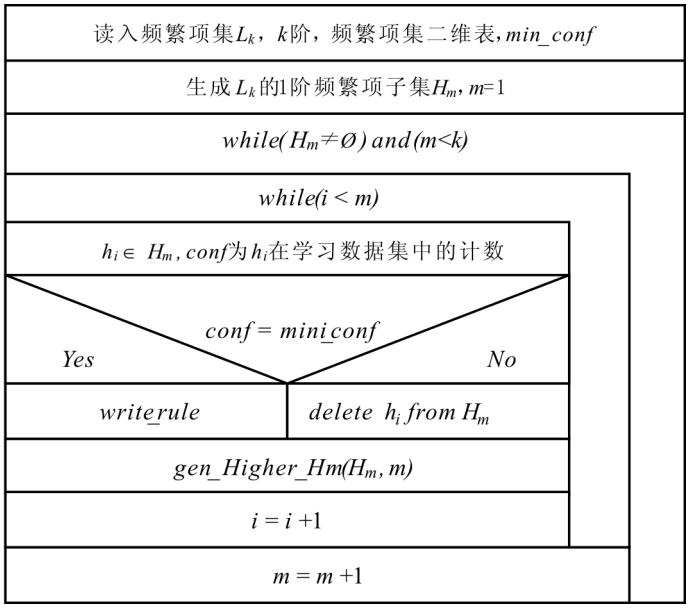

图 5　LoopGenRules 算法流程

在关联规则挖掘中，频繁项集 l_k 和 l_k 的非空子集都是频繁项，各个频繁项集的支持度以二维数据结构表示，预先加载到内存中用于计算候选关联规则的置信度。

4.2 LoopGenRules 算法执行过程样例

以一个 4 阶频繁项集生成强关联规则的过程来演示 LoopGenRules 执行的过程。

设 $l_4=\{a, b, c, d\}$。

第一次循环：由 l_4 得到 1 阶候选后件集合 $H_1=\{\{a\},\{b\},\{c\},\{d\}\}$；依次产生候选规则 $\{b,c,d\}\Rightarrow\{a\}$、$\{a,c,d\}\Rightarrow\{b\}$、$\{a,b,d\}\Rightarrow\{c\}$、$\{a,b,c\}\Rightarrow\{d\}$，并验证这四条规则是否满足置信度要求。对于满足置信度要求的关联规则记录到知识库中；否则，从集合 H_1 中删除该元素 h_i。

假设第 1 步的候选规则中有 $\{a,b,c\}\Rightarrow\{d\}$ 不满足置信度要求，$\{d\}$ 会被从 H_1 中删除，则最后 1－阶候选后件集合 $H_1=\{\{a\},\{b\},\{c\}\}$。然后用 H_1 拼接 2－阶候选后件集合 $H_2=\{\{a, b\},\{a, c\},\{b, c\}\}$。

第二次循环：依次利用 2－阶候选后件集合 $H_2=\{\{a, b\},\{a, c\},\{b, c\}\}$ 的元素产生候选规则 $\{c,d\}\Rightarrow\{a,b\}$、$\{b,d\}\Rightarrow\{a,c\}$、$\{a,d\}\Rightarrow\{b,c\}$、$\{a,d\}\Rightarrow\{b,c\}$，并验证这三条规则是否满足置信度要求。

假设第 2 步的候选规则中有 $\{a,d\}\Rightarrow\{b,c\}$ 不满足置信度要求，$\{b,c\}$ 会被从 H_2 中删除，则最后 2－阶候选后件集合 $H_2=\{\{a, b\},\{a, c\}\}$。然后用 H_2 拼接 3－阶候选后件集合 $H_3=\{\{a, b,c\}\}$。

第三次循环：利用 3－阶候选后件集合 $H_3=\{\{a, b,c\}\}$ 的元素产生候选规则 $\{d\}\Rightarrow\{a,b,c\}$，验证此规则是否满足置信度要求，根据结果判断是否写入规则知识库。

H_3 只比输入的频繁项集阶数低一阶，无法再生成 4－阶候选后件集合，终止循环，见表 1。

表 1　LoopGenRules 的执行过程样例

循环序数	循环初始候选后件集合 H	候选规则	循环结束时候选后件集合 H
1	$\{\{a\},\{b\},\{c\},\{d\}\}$	$\{b,c,d\}\Rightarrow\{a\}$（成立） $\{a,c,d\}\Rightarrow\{b\}$（成立） $\{a,b,d\}\Rightarrow\{c\}$（成立） $\{a,b,c\}\Rightarrow\{d\}$（不成立）	$\{\{a\},\{b\},\{c\}\}$
2	$\{\{a,b\},\{a,c\},\{b,c\}\}$	$\{c,d\}\Rightarrow\{a,b\}$（成立） $\{b,d\}\Rightarrow\{a,c\}$（成立） $\{a,d\}\Rightarrow\{b,c\}$（不成立）	$\{\{a,b\},\{a,c\}\}$
3	$\{\{a,b,c\}\}$	$\{d\}\Rightarrow\{a,b,c\}$（成立）	$\{\{a,b,c\}\}$

至此,利用$\{a, b, c, d\}$这个频繁项集得到满足置信度要求的关联规则,并且对不满足置信度要求的后件遍历路径进行了剪枝,没有重复去验证包含$\{d\}$、$\{b, c\}$做子集的后件的规则,减少了时间耗费。

5 性能优化

在计算置信度时,需要计算前件与整个频繁项集的支持数的比值。为了进一步提高规则生成时计算置信度的速度,一次性把所有的频繁项集和支持计数读入内存,以散列表的形式记录,在查询指定的频繁项集的支持度时,不再执行耗时的磁盘 I/O 操作。定义内存散列表格式为:第一个参数为字符串类型,存储频繁项集,是 key;第二个参数为整型,存储支持度计数,是 value。

定义 $dict\_FQI$:

$Dictionary<String, int> dict\_FQI = new\ Dictionary<String, int>()$;

使用频繁项集 l_k 作为键值,在内存散列表 dic_FQI 中检索到其对应的计数数值,进行置信度计算:$int\ nLC = dict\_FQI[l\_k]$。

6 实验内容与分析

为了验证和测试我们的 LoopGenRules 关联规则产生算法的执行时间耗费,在 i5－2520M 双核处理器、12GB 内存、1TB 5400rpm 机械硬盘、64 位 Windows7 的实验环境下,用 6 至 11 个一阶频繁项分别生成了 6 组人工测试数据,分别最高产生了 6 阶到 11 阶的人工频繁项集,然后由各组最高阶的频繁项集生成强关联规则。在不同阶的人工频繁项集下基于文献[1]的 ap－genrules 算法与本文 LoopGenRules 算法处理对应频繁项集数据在相同置信度要求下,各自执行 10 次生成关联规则,记录时间耗费数据,最后得到平均时间耗费数据见表 2(置信度阈值均为 20%):

表 2　不同阶的人工频繁项集上执行的时间耗费

频繁项集阶数	ap－genrules 算法耗时均值(miliseconds)	LoopGenRules 算法耗时均值(miliseconds)
6 阶	30.89	26.66
7 阶	54.04	44.99
8 阶	109.46	97.44
9 阶	228.36	198.68
10 阶	436.01	406.30
11 阶	850.64	819.45

从实验运行效果看来，在全部的实验数据下，采用 LoopGenRules 方法来生成关联规则的平均时间耗费均短于 ap－genrules 关联规则生成方法，平均节约大约 6％的时间耗费，见图 6。

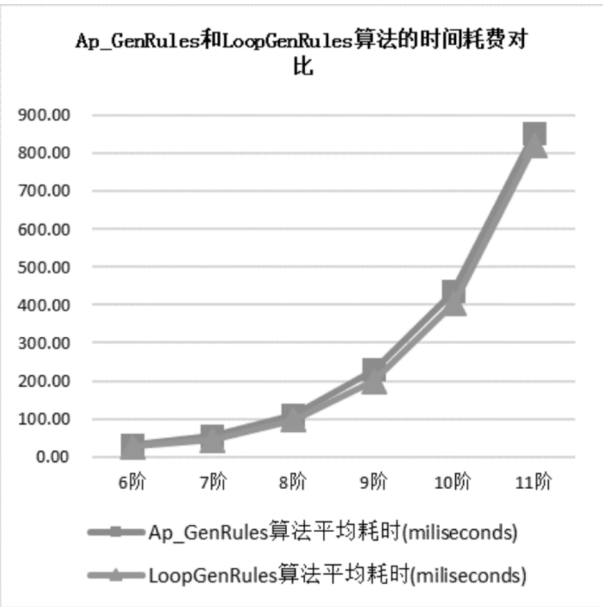

图 6　在不同阶频繁项集下 LoopGenRules 算法时间耗费

我们利用 LoopGenRules 算法对某商业企业一年的销售历史数据进行关联规则挖掘，取得了较好的效果，针对学习数据集的关联规则挖掘的结果表示如图 7 所示。

商品编码P		商品编码R	支持度	置信度
花生汤圆*,鲜肉水饺*	——>	黑芝麻汤圆*	0.1901	0.9295
花生汤圆*	——>	黑芝麻汤圆*	0.2360	0.8774
鲜肉水饺*	——>	黑芝麻汤圆*	0.3926	0.8141
黑芝麻汤圆*,花生汤圆*	——>	鲜肉水饺*	0.1901	0.8057
花生汤圆*	——>	鲜肉水饺*	0.2045	0.7605
花生汤圆*	——>	黑芝麻汤圆*,鲜肉水饺*	0.1901	0.7069
黑芝麻汤圆*	——>	鲜肉水饺*	0.3926	0.6667
黑芝麻汤圆*,鲜肉水饺*	——>	花生汤圆*	0.1901	0.4843
鲜肉水饺*	——>	花生汤圆*	0.2045	0.4241
黑芝麻汤圆*	——>	花生汤圆*	0.2360	0.4007
鲜肉水饺*	——>	黑芝麻汤圆*,花生汤圆*	0.1901	0.3942
黑芝麻汤圆*	——>	花生汤圆*,鲜肉水饺*	0.1901	0.3228

图 7　在某销售数据上进行关联规则挖掘的效果

7 结语

基于 Apriori 算法和 ap－genrules 算法改进的 LoopGenRules 算法在保存于关系数据库中的数据进行关联规则挖掘，基本不受数据规模限制。通过利用内存散列结构存储频繁项集等方法优化性能，针对某小企

业的部分历史数据(1 万条事务记录左右)在单机实验环境进行实验,生成支持度 10%及以上的全部频繁项集平均在 2000 毫秒左右,基于频繁项集再生成置信度 20%及以上关联规则的响应时间在平均在 180 毫秒左右,可以看出由频繁项集产生规则的时间耗费占总体时间耗费的比例很小了,此关联规则生成算法具有较好的实用性。

由于当前大规模的数据通常还附带有数值型或非数值型的属性,并且越来越多的基于大数据环境存储,今后要继续探索在带层次化属性的、大数据环境下关联规则的挖掘技术,并且丰富挖掘结果的图形化展示手段,以对规则的下一步使用提供更多的信息。

参考文献:

[1] RAKESH A, RAMAKRISHNAN S. Fast Algorithms for Mining Association Rules[C]//In Proc.1994 Int. Conf. Very Large Databases, 1994:487－499.

[2] 董博,王雪.关联规则算法的计算效率优化研究[J].计算机仿真,2017,34(9):247－253.

[3] 袁晓建,张岐山,甘智平,等.Apriori 算法的改进及在电子商务中的应用[J].福州大学学报(自然科学版), 2018,46(3):330－334.

[4] 刘林东,齐德昱.一种改进的关联规则挖掘算法研究[J].广东第二师范学院学报,2018,38(3):69－73.

[5] 崔妍,包志强.关联规则挖掘综述[J].计算机应用研究,2016,33(2):330－334.

[6] 刘步中.基于频繁项集挖掘算法的改进与研究[J].计算机应用研究,2012,29(2):475－477.

[7] 钱光超,贾瑞玉,张然,李龙澍.Apriori 算法的一种优化方法[J].计算机工程,2008,34(23):196－198.

[8] 王晓峰,王天然,赵越.一种自顶向下挖掘长频繁项的有效方法.计算机研究与发展,2004,41(1):148－155.

[9] HAN J W, KAMBER M, JIAN P. 数据挖掘:概念与技术[M].范明,孟小峰,等,译.北京: 机械工业出版社,2012:160－173.

[10] 陈耿.数据挖掘技术及其在用户行为分析系统中的应用[D].成都:西南交通大学,2003:44－53.

A Non-recursive Algorithm for Generating Association Rules Based on Pruning

CHEN Geng[1,5], *GUO Bing*[2], *WEN Dengmin*[3], *HUANG Qian*[4], *DENG Li*[1]

(1.School of Computer Science, Chengdu Normal University, Chengdu 611130, China;
2. School of Computer, Sichuan University, Chengdu 610065, China;
3. School of Information Science, Southwest Jiaotong University, Chengdu 610031, China;
4. West China School of Public Health, Sichuan University, Chengdu 610041, China;
5. Institute of Compute Intelligence and Info Technology,
Chengdu Normal University, Chengdu 611130, China)

Abstract: Association rule mining, one of the most commonly used data mining activities, is usually conducted by mining technology based on frequent itemsets. How to rapidly generate association rules is one of the important factors affecting the total time cost of association rules mining after frequent itemsets are obtained. The realization method of the mining algorithm of frequent itemsets based on Apriori algorithm is introduced, the primary process of frequent itemsets generating association rules is illustrated, and the principle of the efficiency improvement caused by the generating algorithm of association rules based on pruning is explained. Then, a non-recursive algorithm of association rules generated by frequent itemsets based on pruning is presented. Experiments in the same hardware and software environments on various frequent itemsets prove that the new algorithm can work correctly, and the time cost can be reduced by about 6% on the average compared to the original method.

Keywords: data mining; frequent itemset; association rule; rule generating; pruning

(实习编辑:杨晓玲　责任校对:曲　比)