文章编号:1671-9352(2008)11-0054-04

基于集合运算的频繁集挖掘优化算法

娄兰芳,潘庆先
(烟台大学计算机科学技术学院,山东 烟台 264000)

摘要:挖掘关联规则是数据挖掘中一个重要的课题,产生频繁项目集是其中的一个关键步骤。提出了一种基于集合运算的频繁项目集挖掘算法,并将该算法与经典算法 Apriori 进行比较。该算法只需要对数据库扫描一遍。实验表明该算法的效率较好。

关键词:数据挖掘;关联规则;频繁项目集
中图分类号:TP311.132 **文献标志码**:A

An improved algorithm based on sets operation for mining frequent itemsets

LOU Lan-fang, PAN Qing-xian
(School of Computer Science and Technology, Yantai University, Yantai 264000, Shandong, China)

Abstract: Mining association rules is an important issue in data mining and one of its key steps is generating frequent itemsets. A frequent itemsets mining algorithm based on sets operation was presented and compared with the classical algorithms apriori. This algorithm needs only to scan the database once. Experiments indicate the new algorithm is very efficient.

Key words: data mining; association rules; frequent itemsets

数据挖掘即数据库中的知识发现。是从大型的数据库中发现潜在的、有价值的、能被用户理解的概念和信息的过程。在数据挖掘研究中关联规则挖掘是一个非常重要的研究领域[1,2]。关联规则挖掘起源于对超级市场"购物篮"问题的研究,主要是发现交易数据库中项与项之间的关联关系。关联规则挖掘研究起初主要应用于诸如市场分析、决策制定、商业管理等领域,但随着研究的不断深入,关联规则挖掘研究的应用越来越广泛,已扩展到了网络分析、天文学和生物学等领域。

传统的关联规则挖掘算法采用的是通过多次扫描数据库来生成频繁项集的策略。这类算法最有代表性的就是 R-Agrawal 在 1994 年提出的 Apriori 算法[3]。本文利用集合运算来发现频繁项集,只须扫描原事务数据库一次,并且不会产生大量的候选集,试验表明该算法是有效的。

1 问题描述

1.1 关联规则的挖掘

关联规则的形式化描述[4]如下:

定义 1 集合 $I=\{i_1,i_2,\cdots,i_m\}$ 为标识符的集合,其中 m 为整数,$i_k(k=1,2,\cdots m)$称为项目。

顾客的一次购物可以用该顾客所购买的所有商品的名称来表示,称为事务,用符号 T 来表示,很显然 $T\leqslant I$(T 是 I 的子集)。一条事务仅包含其涉及到的项目,而不包含项目的具体信息。所有事务的集合就构成了关联规则挖掘的数据集,称为事务数据库,用符号 D 来表示,事务数据库的记录数用 $|D|$ 来表示。

定义 2 项目集是由 I 中项目构成的集合。若项目集包含的项目数为 k,则称此项目集为 k 项目

收稿日期:2008-09-12
基金项目:烟台大学青年基金资助项目(DP0721)
作者简介:娄兰芳(1971-),女,硕士研究生,讲师,主要感兴趣的研究领域为数据挖掘与分析.Email:loulanfang@126.com
潘庆先(1979-),男,硕士研究生,讲师,主要感兴趣的研究领域为数据挖掘与分析.Email:pqx@ytu.edu.cn

集。

定义 3 对任意项目集 X,若数据库 D 中 $s\%$ 的事务包含项目集 X,则项目集 X 的支持率为£,记为 $\text{support}(X)=s\%$,其中包含项目集 X 的事务数称为项目集 X 的频度,记为 $\text{count}(X)=c$。若项目集 X 的支持率大于或等于用户指定的最小支持率,则项目集 X 称为频繁项目集或大项目集,否则项目集 X 称为非频繁项目集或小项目集。

定义 4 关联规则是形如 $X\Rightarrow Y$ 的规则,其中 X、Y 为项目集且 $X\cap Y=\phi$。

定义 5 在数据库 D 中,若 $s\%$ 的事务包含 $X\cup Y$,则关联规则 $X\Rightarrow Y$ 的支持度为 $s\%$;在数据库 D 中,若 $c\%$ 的包含项目 X 的事务也包含项目 Y,则关联规则 $X\Rightarrow Y$ 的置信度为 $c\%$。

定义 6 若关联规则 $X\Rightarrow Y$ 的支持度和置信度分别大于或等于用户指定的最小支持度和最小置信度,则关联规则 $X\Rightarrow Y$ 为强关联规则;否则称为弱关联规则。

1.2 Apriori 算法

Apriori 算法经过逐层迭代来计算出数据库中的频繁项集。第 I 次迭代计算出所有 I-项频繁集。它的每一次迭代均经过 3 个步骤:

(1) 联结产生候选集,利用$(k-1)$项频繁数据项集 F_{k-1}中产生 k-项候选数据项集 C_k。

(2) 剪枝。如果候选集中至少有一个子集不是频繁数据项集,则删除该数据项集合。

(3) 计算支持数。

1.3 集合论

1.3.1 集合的基本概念

集合[5]是不能被精确定义的基本的数学概念。一般认为一个集合指的是一些可确定的可分辨的事务构成的整体。对于给定的集合和事物,应该可以断定这个特定的事物是否属于这个集合。如果属于,就称它为这个集合的元素。集合可以有各种类型的事物构成。

一般说来,集合的元素可以是任意类型的事物,一个集合也可以作为另一个集合的元素。

1.3.2 集合的基本运算

集合的基本运算包括并、交、补、差等运算。下面就给出这些运算的定义[5]。

设 A,B 为集合,E 是全集。

$A\cup B$ 的并集定义为:

$$A\cup B=\{x\mid x\in A \text{ 或 } x\in B\}。$$

$A\cap B$ 的交集定义为:

$$A\cap B=\{x\mid x\in A \text{ 且 } x\in B\}。$$

$\bar{A}$ 的补集定义为:

$$\bar{A}=\{x\mid x\in E \text{ 且 } x\notin B\}。$$

$A-B$ 的差集定义为:

$$A-B=\{x\mid x\in A \text{ 且 } x\notin B\}。$$

性质 1 频繁数据项集的所有子集是频繁的。

性质 1 是频繁数据项集上交运算封闭性的结果。这条性质为在自下向上搜索频繁数据项集的过程中,采用删除策略提供了理论基础。

在挖掘 2-项集及 k-项集$(k>2)$时,所有频繁项集的非空子集也一定是频繁的,即频繁项集的任何非空子集都有可能有最小支持度。所以 Apriori 算法使用了一种联结规则来产生候选集,通常可以这样定义这种规则运算:

$$\begin{aligned}L_k * L_k=\{&p\cup q \text{ 其中},p,q\in L_k,p.\text{item}_1=\\&q.\text{item}_1,\cdots,p.\text{item}_{k-1}=\\&q.\text{item}_{k-1},p.\text{item}_k<q.\text{item}_k\}\end{aligned}\quad(1.1)$$

当 $k=1$ 时,该运算表示单联结。

性质 2 设 X 和 Y 是 2 个数据项集,其中 $X\subseteq Y$,那么 $\text{support}(X)\geqslant\text{support}(Y)$。

如果数据项集 X 是数据项集 Y 的子集,那么 Y 的支持数一定等于或小于 X 的支持数。

2 改进的算法及其描述

2.1 改进算法

改进算法主要是针对发现频繁集这一阶段进行改进的。改进算法思想主要是基于集合论的。下面给出改进算法的形式化描述:

$L_1=\{\text{large } 1-\text{sets}\}$;//此处 L_1 由 Itemset 集、Tset 集和支持数 support 组成。

For $(k=2;\ L_{k-1}\neq\varnothing;\ k++)$ do begin

$C_k=\text{gen}(L_{k-1})$;//产生 k-项频繁集

end

Answer $=\bigcup k\ L_k$;

其中 gen 函数是以 L_{k-1}作为输入的,其结果可直接生成 k-项频繁集。gen 函数实现如下:

gen $(L_{k-1})\{$

$C_k=\{L_{k-1}.\text{Itemset} * L_{k-1}.\text{Itemset}\}$//根据定义(1.1)产生候选集

$T_k=\{L_{k-1}.\text{Tset}\cap L_{k-1}.\text{Tset}\}$//事务支持集合作交运算

For all candidates $c\in C_k$ do

$c.\text{support}=\text{Count}(c.T_k)$;//计算每一项的支

持数
If c.support < minsup then
delete c
end
end
}

2.2 应用实例

以表 1 中的事务数据库为例进行阐释说明。

表 1　一个简单的事务数据库
Table 1　A simple transaction database

事务 TID	项集 Itemset
T_1	1,3,4
T_2	1,2,3,4,5
T_3	2,3,4
T_4	1,2,3,6
T_5	4,5,6
T_6	2,3,4,6

(1) 扫描一遍事务数据库,建立如下数据库表 2。之所以要生成这样的表,最大的优势在于方便计算候选集项目的支持数。

表 2　新建立的数据库表
Table 2　New creation database

项集 Itemset	事务支持集 Tset
{1}	$\{T_1, T_2, T_4\}$
{2}	$\{T_2, T_3, T_4, T_6\}$
{3}	$\{T_1, T_2, T_3, T_4, T_6\}$
{4}	$\{T_1, T_2, T_3, T_5, T_6\}$
{5}	$\{T_2, T_5\}$
{6}	$\{T_4, T_5, T_6\}$

(2) 根据库表 2,计算事务支持集 Tset 中各行元素的个数(即事务支持数),然后删除那些支持数小于给定阈值的项集.这样就可得到 1-项频繁集。表 2 中因为计算项目集{5}的支持数为 2 达不到阈值,故将其删除。结果见表 3。

表 3　1-项频繁集
Table 3　One length of large item sets

项集 Itemset	事务支持集 Tset	支持数 Support
{1}	$\{T_1, T_2, T_4\}$	3
{2}	$\{T_2, T_3, T_4, T_6\}$	4
{3}	$\{T_1, T_2, T_3, T_4, T_6\}$	5
{4}	$\{T_1, T_2, T_3, T_5, T_6\}$	5
{6}	$\{T_4, T_5, T_6\}$	3

(3) 根据得到的 1-项频繁集,项目集两两作并运算,其对应的事务支持集 Tset 两两作交运算求出交集,然后计算每项事务支持集元素的个数,删除那些支持数小于给定阈值的项集,这样就得到 2-项频繁集。最终得到表 4。

表 4　2-项频繁集
Table 4　Two length of large item sets

项集 Itemset	事务支持集 Tset	支持数 Support
{1,3}	$\{T_1, T_2, T_4\}$	3
{2,3}	$\{T_2, T_3, T_4, T_6\}$	4
{2,4}	$\{T_2, T_3, T_6\}$	3
{3,4}	$\{T_1, T_2, T_3, T_6\}$	4

(4) 根据 Apriori 算法中的定义计算 $L_2 * L_2$,可得到 3-项候选集,其对应的事务支持集作交运算。实质上,这时 $L_2 * L_2$ 就是项集作集合的交与并的运算。在本例中,仅有{2,3}与{2,4}符合定义(3.1),{2,3}与{2,4}作并运算可得 3-项候选集{2,3,4};其对应的事物支持集为$\{T_2, T_3, T_4, T_6\} \cap \{T_2, T_3, T_6\} = \{T_2, T_3, T_6\}$,计算支持数为 3,等于给定最小支持数阈值,故应保存该项集;最终得到 3-项频繁集如表 5 所示。

表 5　3-项频繁集
Table 5　Three length of large item sets

项集 Itemset	事务支持集 Tset	支持数 Support
{2,3,4}	$\{T_2, T_3, T_6\}$	3

(5) 到此看到项集 3-项频繁集已无法产生 4-项频繁集,频繁集挖掘算法至此结束。

3　算法实现和性能分析

下面给出了论文改进算法在超市系统中定义的 2 个结构体。

```
Public Type Spswrs
  spidset() As String '商品项目集
  saleidset() As String '销售事务集
End Type
Public Type Spsetsup
  support As Integer '支持数
  spidset() As String '商品项目集
End Type
```

图 1 示意了数据在内存中的存储。系统定义一个结构体数组和 I, J 两个变量作为指示指针。通过对指针 I, J 进行循环操作即可模拟表的联结。

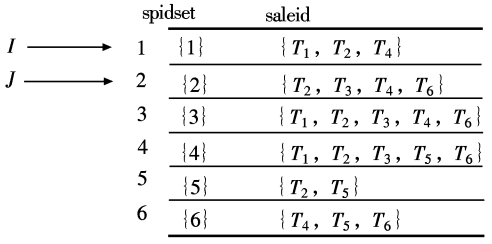

图 1　数据在内存中的存储
Fig.1　Data store in memory

表 6 是在超市管理系统中分别执行改进算法和 Apriori 算法实际测得的数据。(注:表中时间数为 0 并非表示不需要执行时间,而是执行时间小于 1s,系统以秒计时)

表 6 算法时间性能比较

Table 6 Time performance comparison table

数据量	支持度阈值为 0.01		支持度阈值为 0.1		支持度阈值为 0.2	
	改进算法	Apriori 算法	改进算法	Apriori 算法	改进算法	Apriori 算法
167 行	1	27	1	1	0	1
422 行	1	30	0	2	0	2
943 行	1	41	0	3	1	3
2199 行	2	55	1	9	1	9

4 结束语

改进的算法在执行性能上较经典 Apriori 算法有以下几点优势:

(1) 只须扫描原事务数据库一次。改进算法在扫描原事务数据库建立如表 2 所示的数据库表结构以后,在产生候选集,计算支持数等操作均不须再次访问原数据库。这样算法的时间性能可以很大地提高。

(2) 在计算候选集支持数上,改进算法只须计算其对应的事务支持集元素个数即得到支持数,较之经典算法需多次扫描数据库的做法,又在执行时间上大大缩短。

(3) 在产生候选集时,经典算法须经过联结、剪枝和计算 3 步,而改进算法不需要经过剪枝。

(4) 在产生($k-1$)-项频繁项集后,由于数据量的骤减,在由($k-1$)-项频繁集生成 k-项候选集时可不必进行数据表的联结,而是定义一个适合存放数据的结构体数组,将($k-1$)-项频繁集中项集 Itemset 和事务支持集 Tset 中的数据全部调入内存结构体数组中,定义 2 个变量指针指示数组中不同的元素,就此根据联结规则模拟数据表的联结,这样可加快执行速度。

参考文献:

[1] CHEN M S, HAN F, YU P S. Data mining: an overview from database perspective[J]. IEEE Transactions on Knowledge and Data Engineering, 1996, 8(6):866-883.

[2] AGRAWAT R, IMIETINSKI T, SWAMI A. Mining association rules between sets of items in very large databases[C]//Proceedings of the ACM SIGMOD Conference on Management of data, washington, USA, may, 1993:207-216. [2008-09-24]. http://www2. computer. org/portal/web/csdl/abs/trans/tk/1999/05/k0798abs.htm

[3] AGRAWAL R, SRIKANT R. Fast algorithms for mining association rules[C]// Proceeding of the 20th Int7 Conference on Very Large Databases, Santiago, Chile, Santiago de Chile: Morgan Kaufmann, 1994: 487-499.

[4] 冯建华,赖辉▮,周立柱,等.利用 L_1 为关联规则的采掘准备数据[J],计算机科学,2000,27(增刊):82-87.

[5] 耿素云, 屈婉玲, 张立昂.离散数学[M]. 第 2 版.北京:清华大学出版社,1999.

(编辑:孙培芹)