第34卷 第23期 计 算 机 工 程 2008年12月
Vol.34 No.23 Computer Engineering December 2008
·人工智能及识别技术· 文章编号：1000—3428(2008)23—0196—03 文献标识码：A 中图分类号：TP301.6

# Apriori算法的一种优化方法

钱光超，贾瑞玉，张 然，李龙澍
(安徽大学计算机科学与技术学院，合肥 230039)

**摘 要：**介绍关联规则挖掘中的经典算法——Apriori算法的关键思想。针对传统Apriori算法效率上的不足，提出一种改进的Apriori算法——En-Apriori算法。该算法采用矩阵的方法，只须扫描一遍数据库，同时优化了连接操作，较好地提高了算法的效率。实验结果表明，En-Apriori算法优于Apriori算法，具有较好的实用性。

**关键词：**关联规则；频繁项集；Apriori算法；En-Apriori算法

# One Optimized Method of Apriori Algorithm

QIAN Guang-chao,, JIA Rui-yu, ZHANG Ran, LI Long-shu
(School of Computer Science and Technology, Anhui University, Hefei 230039)

**【Abstract】** This paper introduces the principle and efficiency of the Apriori algorithm which is the classical algorithm of association rules mining, and proposes an improved Apriori algorithm——En-Apriori algorithm which is aimed at the disadvantage of Apriori algorithm. This algorithm adopts a matrix method and need scan the database only once and optimizes the join operation, so it is more efficient. The experiment shows that the En-Apriori algorithm outperforms Apriori algorithm, and gets a good practicality.

**【Key words】** association rules; frequent itemsets; Apriori algorithm; En-Apriori algorithm

## 1 概述

关联规则挖掘发现大量数据中项集之间有趣的关联或相关关系[1]。从大量商务事务记录中发现有趣的关联关系，可以帮助制定许多商务决策。有一个关联规则的例子就是“70%的顾客在购买面包和黄油的同时也会购买牛奶”，其直观意义为顾客在购买某些商品的时候有多大倾向会购买另外一些商品。

最经典的关联规则挖掘算法是Apriori算法[2]。它是一种挖掘单维、单层、布尔关联规则的算法，由Rakesh Agrawal Rama和krishnan Skrikant于1993年提出的，其思想是利用已知的高频数据项集推导其他高频数据项集。挖掘关联规则的对象是含有大量事务的数据库，所以如何设计一种高效的算法，提高计算效率，降低扫描数据库的次数，是研究关联规则的主要课题。

Apriori算法存在以下3点不足[3]：(1)由$k$阶频繁集生成$K+1$阶候选频繁集时，在$K+1$阶候选频繁集中过滤掉非频繁集的策略值得进一步改进。(2)连接程序中相同的项目重复比较太多，因而其效率值得进一步改进。(3)在回扫数据库时有许多不必比较的项目或事务重复比较。根据上述3点不足，本文在Apriori算法的基础上提出了一种优化的En-Apriori算法。

## 2 Apriori算法及其分析

Apriori算法主要包含以下3个步骤[3]：

(1)连接步骤：连接($k$-1)-频繁项集生成$k$-项候选集。可以连接的条件是2个($k$-1)项的前($k$-2)项相等并且第1个($k$-1)项集的第($k$-1)项比第2个($k$-1)项集的第($k$-1)项小。记$l_i[j]$为$l_i$中的第$j$个项，则条件即为

$l_1[1]=l_2[1]\wedge l_1[2]=l_2[2]\wedge\cdots\wedge l_1[k-2]=l_2[k-2]\wedge l_1[k-1]<l_2[k-1]$

其中，$l_1[k-1]<l_2[k-1]$可以确保不产生重复的$k$-项集。

(2)删除步骤：利用Apriori性质对$k$-项候选集进行剪枝。剪枝的规则是：若一个$k$-项候选集的任一子集(($k$-1)-项集)不属于($k$-1)-项频繁集$L$，那么该候选$k$-项集就不可能成为一个频繁$k$-项集，可以将其删除。

(3)计数步骤：扫描交易数据库，累加$k$-项候选集在交易数据库中出现的次数。对于一条交易记录和一个候选项集，若交易记录包含该候选项集，则该候选项集出现的次数就加1个$l$。最后根据给定的最小支持度阈值生成$k$-项频繁集。

## 3 优化的Apriori算法——En-Apriori算法

从以上对Apriori算法的分析中可得出，Apriori算法需要大量进行的3个操作是判断2个$k$-是否前$k$-项相同且最后一项不同，和判断一个项集是否为另一个项集的子集以及扫描数据库。第1项操作存在于合并步骤中，第2项操作存在于删除步骤和记数步骤中，第3项操作用在判断候选项集是不是频繁项集中。

这3项操作占了程序运行的大部分时间，尤其是多次扫描数据库。如果能减少这3个操作执行的次数，就可以提高Apriori算法的运行效率。本文提出一种基于Apriori算法的改进算法——En-Apriori，只须扫描一遍数据库，并可以大大减少连接操作。

### 3.1 连接策略优化

(1)减少判断次数

由于在Apriori算法中各交易记录中各项均是已经按字典排序的，因此由Apriori算法可以得出生成的候选项集也是有序的，并且等式的任意2个候选项集之间对位比较也是有序

**基金项目：**安徽省教育厅科研基金资助项目(2005kj056)
**作者简介：**钱光超(1982－)，男，硕士研究生，主研方向：智能软件；贾瑞玉，副教授；张 然，硕士研究生；李龙澍，教授、博士生导师
**收稿日期：**2008-04-17 **E-mail：**qianguangchao1@163.com

的。对于生成的频繁项集也是如此[4]。因此，在实现Apriori算法时可以利用项集之间有序的特点来减少上述 2 个操作的次数，从而提高算法的运行效率。

**定义 1** 若对 2 个$k$-项有序集$l_x$和$l_y$之间对位比较，即$l_x[i]$和$l_y[i]$比较，若有$l_x[1]=l_y[1]\wedge\cdots\wedge l_x[i\text{-}1]=l_y[i\text{-}1]\wedge l_x[i]<l_y[i]$，$1\le i\le k$，则称$l_x$比$l_y$小，记为$l_x<l_y$。若$m$个$k$-项有序项集$l_1, l_2,\cdots,l_m$，有$l_1<l_2<\cdots<l_m$，则称这$m$个$k$-项集之间有序。

对于这 $m$ 个 $k$-项集有如下性质：

**性质 1** 对于其中的任一个 $k$-项集 $l$，有 $l[1]<l[2]<\cdots<l[k]$。

证明：因为 $l$ 是有序项集，所以项集中的每个项都比前一个项大，有 $l[1]<l[2]<\cdots<l[k]$。证毕。

**性质 2** 对于其中的任意 3 个$k$-项集$l_x,l_y,l_z,1\le x<y<z<m$，若有$l_x[1]=l_y[1]\wedge\cdots\wedge l_x[i\text{-}1]=l_y[i\text{-}1]\wedge l_x[i]<l_y[i],l\le i\le k$，则存在一个$j,1\le j\le I$，使得$l_x[j]<l_z[i]$。

证明：因为$y<z$，所以 $1_y<l_z$。若$l_y[i]\le=l_z[i]$，则$j=i$，使得$l_x[i]<l_z[i]$；若$l_y[i]>l_z[i]$，则存在一个$P$，使得$l_y[p]<l_z[p]$，$l\le p\le i$，取$j=p$，有$l_x[j]<l_z[j]$。所以存在一个$j$，$1\le j\le i$，使得$l_x[j]<l_z[j]$。证毕。

利用性质 1 和性质 2，对连接步骤进行优化。

对于 2 个($k$-1)-项频繁集$l_x$和$l_y$，若$l_x$不能和$l_y$连接，则$l_x$与$l_y$之后的所有($k$-1)-项集都不能满足连接条件。所以，只要$l_x$与$l_y$不能连接，就不需要再判断$l_x$与$l_y$之后的所有($k$-1)-项集是否能连接。算法描述如下：

```
for each lx∈Lk-1
{foreach ly∈Lk-1
  {if (lx[1]=ly[1]∧ly[2]…∧lx[k-2]=ly [k-2]∧lx[k-1]<ly[k-1])
    {C=lx ⊗ ly; //将 2 个项集连接到一起
     Ck=Ck ⋃ {C};}
   else break; //1y后面的项集都不能与lx进行连接操作
  }
}
```

(2)修剪频繁集

频繁项集具有如下性质：

**性质 3** $k$-维数据项目集是频繁项目集的必要条件是它的所有$k$-1 维子集均是频繁项目集[4]。

**性质 4** 若在$k$-维数据项目集$X=\{i_1,i_2,\cdots,i_k\}$中，存在一个$j\in X$，使得$|L_{k-1}(j)|<k\text{-}1$，则$X$不是频繁项目集。其中，$|L_{k-1}(j)|$表示($k$-1)维频繁项目集的集合$L_{k-1}$中包含$j$的个数[5]。

证明：设$X$是$k$维频繁项目集，则它的$C_k^{k-1}=k$个$k$-1 维子集均在$L_{k-1}$中。则在由$X$生成的$k$个$k$-1 维子集中，每一个项目$i\in X$共出现$k$-1 次，因此对 $\forall\, i\in X$，均有$|L_{k-1}(j)|>k\text{-}1$，这与条件矛盾，所以$X$不是频繁项目集。

上述性质说明：若$L_{k-1}$中有一元素$c$包含一个项目$i$使得$|L_{k-1}(j)|<k\text{-}1$，则所有$L_{k-1}$中不同于$c$的元素与$c$连接而生成的候选$k$-维数据项目集均不可能是频繁项目集(假设$c$由生成的一个候选$k$-维数据项目集$X$是频繁项目集，则$c$显然是$X$的一个$k$-1 维子集。由于$i\in c$，因此$i\in X$，由性质 2 的证明可知$|L_{k-1}(i)|>k\text{-}1$，与条件矛盾)。

既然由 $c$ 生成的一个候选 $k$ 维数据项目集一定不是频繁项目集，那么就不需要让 $c$ 参加连接。因此，有如下修剪频繁集优化策略：

先计算$|L_{k-1}(j)|$，其中$j\in L'=\{\cup A|,A\in L_{k-1}\}$，即计算$L_{k-1}$中所有项目的频度，再找出那些频度小于$k$-1 的项目，记为$L''=\{i||L_{k-1}(i)|<k\text{-}1\}$，然后在$L_{k-1}$中去掉所有包含$L''$中任一元素的频繁项目集，从而得到一个新的更小的$k$-1 维频繁项目集的集合$L'_{k-1}$，再由$L'_{k-1}$与自身相连接生成候选$k$-维数据项目集的集合$C'_k$。这样生成的候选$k$维数据项目集合$C'_k$一般比原来的由$L_{k-1}$生成的集合$C_k$要小一些。

### 3.2 数据库优化策略

在 Apriori 算法中，多次扫描数据库需要消耗较多的时间。如何在这步操作中提高效率，本文的算法中使用了布尔矩阵来记录数据库中事务信息的方法，该方法只须扫描一次数据库，大大提高了算法的效率。

设$I=\{i_1,i_2,\cdots,i_m\}$是项的集合，任务相关的数据$\boldsymbol{D}$是数据库事务的集合，其中每个事务$T$是项的集合，使得$T\subseteq I$。每个事务有一个标识符，称作$TID$。

**定义 2** 项集 $I$ 的矩阵记为

$$\boldsymbol{D}=(\boldsymbol{D}_1,\boldsymbol{D}_2,\cdots,\boldsymbol{D}_n)=\begin{bmatrix} d_{11} & d_{12} & \cdots & d_{1n} \\ d_{21} & d_{22} & \cdots & d_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ d_{p1} & d_{p2} & \cdots & d_{pn} \end{bmatrix}$$

其中，$p$ 是事务数量；$n$ 是项的数目。

**定义 3** 每个项$I_j$的向量定义为

$$\boldsymbol{D}_j=\begin{bmatrix} t_{1j} \\ t_{2j} \\ \vdots \\ t_{pj} \end{bmatrix}$$

其中，$t_{ij}=\begin{cases}0 & I_j\notin T_i\\ 1 & I_j\in T_i\end{cases}$，因此，$I_j$的支持度计数为

$$support\_count(I_j)=\sum_{i=1}^{p}t_{ij}$$

**定义 4** 2-项集$\{I_i,I_j\}$的向量定义为

$$\boldsymbol{D}_{ij}=\boldsymbol{D}_i\wedge\boldsymbol{D}_j=\begin{bmatrix} d_{1i}\wedge d_{1j} \\ d_{2i}\wedge d_{2j} \\ \cdots \\ d_{pi}\wedge d_{pj} \end{bmatrix}$$

其中，“∧”是逻辑“与”运算符。因此，2-项集$\{I_i,I_j\}$的支持度计数$support_count\{I_i,I_j\}=\sum_{k=1}^{p}d_{ki}\wedge d_{kj}$。

**定义 5** $k$-项集$\{I_1,I_2,\cdots,I_k\}$向量定义为

$$\boldsymbol{D}_{12\cdots k}=\boldsymbol{D}_1\wedge\boldsymbol{D}_2\wedge\cdots\wedge\boldsymbol{D}_k=(\boldsymbol{D}_1\wedge\boldsymbol{D}_2\wedge\cdots\wedge\boldsymbol{D}_{k\text{-}1})\wedge\boldsymbol{D}_k$$

因此，$k$-项集$\{I_1,I_2,\cdots,I_k\}$的支持度计数为

$$support\_count\{I_1,I_2,\cdots,I_k\}=\sum_{q=1}^{p}\{(d_{q1}\wedge d_{q2}\wedge\cdots d_{q3})\wedge d_{qk}\}$$

### 3.3 En-Apriori 算法分析

En-Apriori 算法采用矩阵来记录事务和项集的信息，从而只需要扫描一遍数据库，效率比 Apriori 算法高很多，这是一种空间换时间的提高效率的方法。其中频繁 1-项集产生的时间为 $O(pn)$，这与 Apriori 算法是一样的($p$ 是事务的数量，$n$ 是事务宽度)。

$L_{k-1}$中繁项集的个数记为$|L_{k-1}|$，$C_k$中的数据项集个数记为$|C_k|$，$C_k$中第$i$元素的子集个数记为$N_i$，$1\le i\le|C_k|$。

在做连接操作时，En-Apriori算法根据频繁项集的一些性质采用了经过缩减的频繁$k$-1 项集$L'_{k-1}$来连接生成候选频繁$k$-项集$C'_k$，$|L'_{k-1}|\le|L_{k-1}|$。在大多情况下，$|L'_{k-1}|\le|L_{k-1}|$，因此在连接操作方面，En-Apriori算法要优于Apriori算法。

Apriori算法从$C_k$中取元素，然后求该元素子集，判断该

子集是否在$L_{k-1}$中，需要进行的计算为$\sum_{j=1}^{|C_k|}\sum_{i=1}^{|N_j|}(|L_{k-1}|)$次。

En-Apriori算法通过事务矩阵$\boldsymbol{D}$($\boldsymbol{D}$是$p\times n$阶矩阵，$p$是事务计数，$n$是项目数目)，即$O(p)$的时间复杂度就可以完成候选频繁项集$C'_k$的生成和$L'_k$的生成。而Apriori算法还需要扫描数据库，判断候选项集是否是频繁项集，这一操作的复杂度是$O(p)$，再加上$\sum_{j=1}^{|C_k|}\sum_{i=1}^{|N_j|}(|L_{k-1}|)$次的计算次数，时间效率将比En-Apriori算法低得多。

## 4 实验比较分析

实验环境：P4 2.93 GMHz，512 MB 内存，WindowsXP+VS2003，实验的数据集采用 http://ftp.Ics.uci.edu/pub/ma~hine-learning-databases/mushroom/ 和 http://www.cse.cuhk.edu.hk/-kdd/data/IBM-VC++.zip 提供的蘑菇数据库(mushroom database)。该数据库有 8 124 条记录，记录了蘑菇的帽子形状、帽子颜色、颈的形状、颈的颜色、气味、生存环境、是否有毒等 23 种属性，每种属性有 2~12 个枚举值。而其中的属性 stalk-root 有 2 480 个样本缺省，另外 veil-type 属性有 2 个枚举值 partial 和 universal。

实验分别采用采用 Apriori 算法和 En-Apriori 算法对蘑菇数据库进行试验。实验前首先对原始数据进行预处理，并利用二进制数据(0 和 1)表示交易中某个项出现与否。(因 Apriori 算法和 En-Apriori 算法只处理单维、单层、布尔变量)。

实验中两者的运行时间比较如图 1 所示。由图 1 可知，在相同数据量和不同支持度下，En-Apriori 算法执行时间上明显少于 Apriori 算法，并且支持度越高算法执行效率越高。

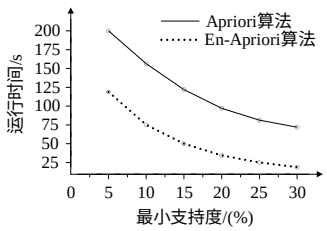


**图 1 不同支持度下 2 种算法运行时间的比较**

## 5 结束语

本文对关联规则提取中的 Apriori 算法进行了研究。Apriori 的对象常常是大型数据库或数据仓库，所面临的最大挑战是计算效率问题。文章通过对 Apriori 算法的分析，提出了一种改进的 En-Apriori 算法，该算法只需扫描一遍数据库，并对连接等操作做了优化，提高数据项集频度统计速度，从而提高算法的效率。

## 参考文献


[1] Tan Pangning, Steinbach M, Kumar V. Introduction to Data Mining[M]. 北京: 人民邮电出版社, 2006.

[2] Han Jiawei, Kamber M. Data Mining: Concepts and Techniques[M]. 北京: 机械工业出版社, 2001.

[3] 区玉明, 张师超, 徐章艳, 等. 一种提高 Apriori 算法效率的方法[J]. 计算机工程与设计, 2004, 25(5): 846-848.

[4] Witten I H, Frank E. Data Mining: Practical Machine Learning Tools and Techniques[M]. 北京: 机械工业出版社, 2006.

[5] 徐章艳, 张师超. 挖掘关联规则中一种优化的 Apriori 算法[J]. 计算机工程, 2003, 29(19): 83-84.


（上接第 149 页）

1)$B$ 选取随机整数 $K$=123, $R^{'}=KG=(4132,6407)$ ；

2)$B$ 计算：

$R=KP_C=(7819,3611)$, $r=h(x)^{-1}1234(\mathrm{mod}\,4427)=264$ ；

3)$B$ 计算：

$s=K+r\sigma(\mathrm{mod}\,4427)=123+264\times2184(\mathrm{mod}\,4427)=1189$ ；

4)(264,1189,(2322,6135))是 $B$ 对消息 $m$ 的签名。

(4)代理签名的验证过程

验证人$C$收到代理签名($r$, $s$, $Q_0$)= (264,1189,(2322,6135))后，首先获取$B$的公钥$Q_0$=(2322,6135)，再进行如下运算：

1)验证 $r$=264, $s$=1189 是否在区间[1, 4426]上；

2)计算：

$X=sG-r(P_A+Q_0r_0)=(917,573)+(7359,3988)=(4132,6407)$,

$m=h(K_Cx^{'})r\,\mathrm{mod}\,n=7819\times264\,\mathrm{mod}\,4427=1234$ ；

3)$C$ 计算原始消息 $m$=1234，并检验最后一位为冗余位 4，结果正确，$C$ 接受签名。

## 5 结束语

在改进的数字签名方案中，签名人对消息的 2 次签名能够有效地抵抗生日攻击。原数字签名方案抗击生日攻击的概率为 $p=1-P_{n-1}^t/(n-1)^t$，改进的数字签名方案抗击生日攻击的概率则为 $p=[1-P_{n-1}^t/(n-1)^t]^2$，所以，提高了数字签名的安全性能。通过具体的实例可以看出，改进的数字签名方案和代理数字签名方案，具有运算量小的特征，且易于实现。

## 参考文献


[1] Koblitz N. Elliptic Curve Cryptosystems[J]. Mathematics of Computation, 1987, 48(1): 203-209.

[2] Miller V S. Use of Elliptic Curves in Cryptography[C]//Proc. of Advances in Cryptology-CRYPTo'85. New York, USA: Spring Verlag, 1986: 417-426.

[3] 杨义先, 孙 伟, 钮心忻. 现代密码新理论[M]. 北京: 科学出版社, 2002: 106-119.

[4] Peng Fei, Qiu Shuisheng, Long Min. A Secure Digital Signature Algorithm Based on Elliptic Curve and Chaotic Mappings[J]. Circuits Systems Signal Processing, 2005, 24(5): 585-597.

[5] 白国强, 黄 谆, 陈弘毅, 等. 基于椭圆曲线的代理数字签名[J]. 电子学报, 2003, 31(11): 1659-1663.

[6] Zhao Zemao, Liu Fengyu. Method of Constructing Elliptic Curve Authenticated Encryption Scheme[J]. Applied Mathematics and Computation, 2005, 168(1): 146-151.

[7] Arazi B. Communication-computation Trade-off in Executing ECDSA in a Contactless Smartcard[J]. Designs Codes and Cryptography, 2006, 38(1): 399-415.

[8] Hwang Renjunn, Lai Chih Hua, Su Fengfu. An Efficient Signcryption Scheme with Forward Secrecy Based on Elliptic Curve[J]. Applied Mathematics and Computation, 2005, 167(1): 870-881.

[9] Trappe W, Washington L G. Introduction to Cryptography with Coding Theory[M]. New York, USA: Spring Verlag, 2004: 189-205.