一种基于概念格的关联规则挖掘算法

王甦菁[1]， 陈震[2]

WANG Su-Jing[1]， CHEN Zhen[2]

1.吉林大学 软件学院,吉林省 长春市 130012

2.吉林大学 计算机科学与技术学院,吉林省 长春市 130012

1.College of Software, Jilin University, Changchun 130012, China

2.College of Computer Science and Technology, Jilin University, Changchun 130012, China

E-mail: sujingwang@hotmail.com，Phn: +86-13756134800

A algorithm for mining association rules based concept lattice

Abstract: Association rule discovery is one of kernel tasks of data mining. Concept lattice, induced from a binary relation between objects and features, is a very useful formal analysis tool. It represents the unification of concept intension and extension. It reflects the association between objects and features, and the relationship of generalization and specialization among concepts. There is a one-to-one correspondence between concept intensions and closed frequent itemsets. This paper presents an efficient algorithm for mining association rules based concept lattice called Arca (Association Rule based Concept lAttice). Arca algorithm uses concept-matrix to build a part of concept lattice, in which the intension of every concept be put into one-to-one correspondence with a closed frequent itemset. Then all association rules are discovered by 4 operators which are defined in this paper performed on these concepts.

Key words: concept lattice; formal concept analysis; data mining; association rule

摘 要: 关联规则挖掘是数据挖掘中的一项核心任务，而由二元关系导出的概念格则是一种非常有用的形式化分析工具，它体现了概念内涵和外延的统一，反映了对象和特征间的联系以及概念间的泛化与例化关系。一个概念内涵与一个关联规则中的闭合项集可以一一对应。本文提出了一种新有基于概念格的关联规则挖掘算法 Arca (Association Rule based Concept lAttice)。Arca 算法通过概念矩阵构造部分概念格，使概念格中的每个概念对应一个闭合频繁项集。然后生成一些关联规则，在这些关联规则上通过定义了四个算子来生成了所有关联规则。

关键词: 概念格；形式概念分析；数据挖掘；关联规则

文献标识码: A **中图分类号**:TP18; TP311

1 引言

关联规则挖掘就是从大量的数据中挖掘出有价值描述数据项之间相互联系的有关知识。这些知识指的是一些形如 $A \rightarrow B$ 的表达式，其中 A 和 B 是特征集合。自从 Agrawal 等人提出了一种经典的提取关联规则的算法 Apriori[1]以来，有许多的关联规则挖掘算法被提出[2-10]。随着对关联规则的深入研究，关联规则的挖掘已经成为数据挖掘的一项核心任务。

国家自然科学基金(the National Natural Science Foundation of China under Grant No.50378042, 50338030)

作者简介: 王甦菁(1976 年－),男,硕士,硕士研究生,主要研究领域为机器学习,数据挖掘;**陈震**(1949 年－),男,硕士,教授,主要研究领域为数据库,信息管理,数据挖掘,决策支持.

德国的 Wille 教授在文献[11]中提出了形式概念分析，用于概念的发现、排序和显示。在形式概念分析中，概念的外延被表示为是属于这个概念的所有对象的集合，而内涵则被表示为所有这些对象所共有的特征(或属性) 集合，从而实现了对概念的哲学理解的形式化。概念格作为形式概念分析中核心的数据结构，本质上描述了对象和属性之间的联系，表明了概念之间的泛化与例化关系。由于概念格结点反映了概念内涵和外延的统一，结点间关系体现了概念之间的泛化和例化关系，因此非常适合作为规则发现的基础性数据结构。

近年来，有许多基于概念格的关联规则挖掘算法[12-16]被提出。这些算法都是构造完整的概念格，每个概念的内涵是一个闭合项集，在这些闭合项集上找出频繁闭合项集。

本文在[17]的基础上提出了一种不需要把概念格全部构造出来的关联规则挖掘算法 Arca。此算法不仅找出频繁闭合项集，而且在频繁闭合项集上挖掘出关联规则。全文组织如下：第 2 节中简单的介绍了下关联规则与概念格的一些基本定义；第 3 节讨论了 Arca 算法中用到的四个算子并给出 Arca 算法的伪码；第 4 节从实验上证明了 Arca 算法的有效性并总结全文。

2 关联规则与概念格的定义

$\mathcal{I}=\{i_1,i_2,\cdots,i_m\}$ 是 m 个项的集合，一个事务数据库 $\mathcal{T}=\{t_1,t_2,\cdots t_n\}$ 是 n 个事务的集合，每个事务都用唯一的 TID 来标识。每个事务 t 都是由中 $\mathcal{I}$ 的一个子集 I 构成。如果 $|I|=k$，那么称 I 为 k 项集。如果 $I\subseteq t$，那么称项集 I 包含于事务 t 中，或者称事务 t 包含项集 I。$\mathcal{T}$ 中包含项集 I 的事务占事务总数的百分比称为项集 I 的支持度。人为给定一个最小支持度 *minsupp*，如果项集 I 的支持度大于等于 *minsupp*，则称项集 I 为频繁项集。

例如：$\mathcal{I}=\{A,B,C,D,E\}$，表 1 是一个事务数据库的例子。

表 1 一个事务数据库

TID	Items
1	A C D
2	B C E
3	A B C E
4	B E
5	A B C E

定义 1 一个数据挖掘背景是一个三元组 $\mathcal{D}=(\mathcal{T},\mathcal{I},\mathcal{R})$，其中 $\mathcal{T}$ 是事务集合，$\mathcal{I}$ 是项集。$\mathcal{R}$ 是 $\mathcal{T}$ 与 $\mathcal{I}$ 之间的一个二元关系，即 $\mathcal{R}\subseteq\mathcal{T}\times\mathcal{I}$。

在一个数据挖掘背景 $\mathcal{D}=(\mathcal{T},\mathcal{I},\mathcal{R})$ 中，如果 $(t,i)\in\mathcal{R}$，我们用 1 表示，否则用 0 表示。例如：表 2 是表 1 中事务数据库对应的数据挖掘背景。我们可以把它看成一个由 0 和 1 组成的矩阵：

$$\begin{array}{c} \\ 1\\2\\3\\4\\5 \end{array}\begin{array}{c} \begin{array}{ccccc} A & B & C & D & E \end{array}\\ \begin{pmatrix} 1&0&1&1&0\\0&1&1&0&1\\1&1&1&0&1\\0&1&0&0&1\\1&1&1&0&1 \end{pmatrix} \end{array}$$

表 2 与表 1 中事务数据库对应的数据挖掘背景

	A	B	C	D	E
1	1	0	1	1	0
2	0	1	1	0	1
3	1	1	1	0	1
4	0	1	0	0	1
5	1	1	1	0	1

定义 2 设 $\mathcal{D}=(\mathcal{T},\mathcal{I},\mathcal{R})$ 是一个数据挖掘背景，集合 $T\subseteq\mathcal{T}$，记：

$f(T)=\{i\in\mathcal{I}\mid\forall t\in T,(t,i)\in\mathcal{R}\}$，表示事务 T 的共同项集，

相应的，对于集合 $I\subseteq\mathcal{I}$，记：

$g(I)=\{t\in\mathcal{T}\mid\forall i\in I,(t,i)\in\mathcal{R}\}$，表示所有包含 I 的事务的集合，

记：$h(I)=f(g((I))$

定义 3 设 $\mathcal{D}=(\mathcal{T},\mathcal{I},\mathcal{R})$ 是一个数据挖掘背景，$T\subseteq\mathcal{T}$，$I\subseteq\mathcal{I}$，如果 $f(T)=I$ 且 $g(I)=T$，则称 $C=(T,I)$ 为 $\mathcal{D}$ 的一个概念。此时称 T 为 C 的外延，本文用 *Extent(C)*表示 C 的外延；称 I 为 C 的内涵，本文用 *Intent(C)*表示 C 的内涵。用 $B(\mathcal{D})$ 记 $\mathcal{D}$ 的所有概念组成的集合。

定义 4 设 $\mathcal{D}=(\mathcal{T},\mathcal{I},\mathcal{R})$ 是一个数据挖掘背景，$C_1=(T_1,I_1)$，$C_2=(T_2,I_2)$ 是 $\mathcal{D}$ 的两个概念，规定：$C_1\le C_2\Leftrightarrow T_1\subseteq T_2$

此时，C_2 称为 C_1 的超概念(superconcept)，称 C_1 为 C_2 的子概念(subconcept)。

由定义可知对于 C_2 的任何一个子概念 C_1，都有 $T_1\subseteq T_2$，即 $|T_1|\le|T_2|$。

定义 5 称偏序集 $\mathcal{L}_D = (B(\mathcal{D}), \leq)$ 称为概念格。

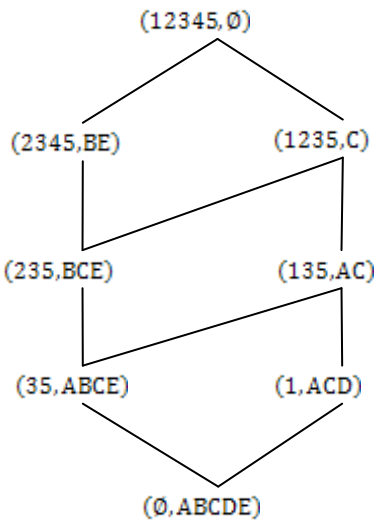

图 1 与表 2 对应的概念格

定义 6 设 $\mathcal{D} = (\mathcal{T}, \mathcal{I}, \mathcal{R})$ 是一个数据挖掘背景，$\mathcal{L}_D$ 中两个不同的结点 $C_1 = (T_1, I_1)$ 和 $C_2 = (T_2, I_2)$，如果满足 $C_1 \leq C_2$，则记为 $C_1 < C_2$。

定义 7 设 $\mathcal{D} = (\mathcal{T}, \mathcal{I}, \mathcal{R})$ 是一个数据挖掘背景，$\mathcal{L}_D$ 中两个不同的结点 $C_1 = (T_1, I_1)$ 和 $C_2 = (T_2, I_2)$，如果 C_1 是 C_2 的子概念且不存在其它的结点 C_3 满足 $C_1 < C_3 < C_2$，则称 C_1 是 C_2 的子结点(直接后继)，而称 C_2 是 C_1 的父结点(直接前驱)。记为 $C_1 \prec C_2$

由定义可知对于 C_2 的任何一个子结点 C_1，都有 $T_1 \subset T_2$，因为如果 $T_1 = T_2$ 时，C_1 和 C_2 是同一个结点。

定义 8 设 $\mathcal{D} = (\mathcal{T}, \mathcal{I}, \mathcal{R})$ 是一个数据挖掘背景，$I \subseteq \mathcal{I}$，项集 I 的支持度定义为

$$support(I) = \frac{|g(I)|}{|\mathcal{T}|}$$

定义 9 设 $\mathcal{D} = (\mathcal{T}, \mathcal{I}, \mathcal{R})$ 是一个数据挖掘背景，$I \subseteq \mathcal{I}$，并且 $support(I) \geq minsupp$，称 I 为频繁项集。

定义 10 设 $\mathcal{D} = (\mathcal{T}, \mathcal{I}, \mathcal{R})$ 是一个数据挖掘背景，对于某个事务 $i \in \mathcal{I}$，在 $\mathcal{D}$ 的矩阵中，如果 i 所对应的列有 n 个 1 时我们称该项的秩为 n，记为 $r(i) = n$。记 $m = \max\{r(i) \mid i \in \mathcal{I}\}$，称为 m 数据挖掘背景的秩。

定义 11 设 $\mathcal{D} = (\mathcal{T}, \mathcal{I}, \mathcal{R})$ 是一个数据挖掘背景，$C = (T, I)$ 是 $\mathcal{D}$ 中的一个概念，对于 T 中的每一个元素 t，从数据挖掘背景矩阵中取出 t 对应的行，组成新的矩阵 M，则称 M 为概念 C 的概念矩阵。例如概念 $(135, AC)$ 的矩阵为：

$$\begin{array}{c} \\ 1 \\ 3 \\ 5 \end{array} \begin{array}{c} \begin{array}{ccccc} A & B & C & D & E \end{array} \\ \begin{pmatrix} 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 1 \end{pmatrix} \end{array}$$

定义 12 设 $\mathcal{D} = (\mathcal{T}, \mathcal{I}, \mathcal{R})$ 是一个数据挖掘背景，$C = (T, I)$ 是 $\mathcal{D}$ 中的一个概念，在 C 的概念矩阵中，如果 i 所对应的列有 n 个 1 时我们称在概念 C 中该项的秩为 n，记为 $R_C(i) = n$，记 $m = \max\{R_C(i) \mid i \in \mathcal{I}\}$，称 m 为概念 C 的秩。

性质 1 概念 C 的所有不同于 C 的子概念的外延个数最大的是 m

证明 假设存在 $C_1 = (T_1, I_1)$，满足 $C_1 < C$ 且 $m < |T_1|$。那么至少存在一个项 i，在 C 的概念矩阵中，i 所对应的列中是 1 的位置对应的行的集合就是 T_1，所以 i 所对应的列有 $|T_1|$ 个 1，这与 C 的秩是 m 矛盾。所以概念 C 的所有不同于 C 的子概念的外延个数最大的是 m。

定义 13 设 $\mathcal{D} = (\mathcal{T}, \mathcal{I}, \mathcal{R})$ 是一个数据挖掘背景，$C = (T, I)$ 是 $\mathcal{D}$ 中的一个概念，对于集合 $I_1 \subseteq \mathcal{I}$，记：$g_C(I_1) = \{t \in T \mid \forall i \in I_1, (t, i) \in \mathcal{R}\}$，表示 T 中具有共同项集 I_1 的事务的集合。

定义 14 设 $\mathcal{D} = (\mathcal{T}, \mathcal{I}, \mathcal{R})$ 是一个数据挖掘背景，$C = (T, I)$ 是 $\mathcal{D}$ 中的一个概念。如果 $|T| \geq |\mathcal{T}| \times minsupp$，则称为频繁概念。

性质 2 如果 $C = (T, I)$ 是频繁概念，则 I 是频繁项集。

证明 $|T| \geq |\mathcal{T}| \times minsupp$，所以 $\frac{|T|}{|\mathcal{T}|} \geq minsupp$。

又 $T = g(I)$，所以 $\frac{|g(I)|}{|\mathcal{T}|} \geq minsupp$。所以 I 是频繁项集。

定理 1 设 $\mathcal{D} = (\mathcal{T}, \mathcal{I}, \mathcal{R})$ 为一数据挖掘背景，$C = (T, I)$ 是 $\mathcal{D}$ 中的一个概念，m 概念 C 的秩，对于 $\forall i \in \{i \mid R_C(i) = m, i \in \mathcal{I}\}$，记 $C_1 = (g_C(i), f(g_C(i)))$，则 C_1 是 C 的子结点

证明 由定义 12 和定义 13 可知 $|g_C(i)| = m$，

如果存在一个不同于 C 和 C_1 的结点 $C_2=(T_2,I_2)$，使得 $C_1<C_2$，则有 $m=|g_C(i)|<|T_2|<|T|$，又有性质 1 可知概念 C 的所有不同于 C 的子概念的外延个数最大的是 m，所以 $|T_2|\le m$，这与 $m<|T_2|$ 矛盾。所以 C_1 是 C 的子结点。

定理 2 设 $\boldsymbol{D}=(\boldsymbol{I},\boldsymbol{J},\boldsymbol{R})$ 是一个数据挖掘背景，$C=(T,I)$ 是 $\boldsymbol{D}$ 中的一个概念，m 为概念 C 的秩，$C_1=\left(g_C(i_1),f\left(g_C(i_1)\right)\right)$ 是 C 的子概念，其中 $i_1\in\boldsymbol{J}$，且 $R_C(i_1)=m_1>0$，如果对于 $\forall C_2\notin C_2(=,T_2)\ I_2\quad C_2\lesssim$，$|C|$，都有 $g_C(i_2)\not\subset T_2$，则 C_1 是 C 的子结点。

证明 假设存在 $C_3=(T_3,I_3)$，使得 $C_1<C_3<C$，则 $m_1=|g(i_1)\cap T|<|T_3|<|T|$，所以 $C_3\in\{C_2=(T_2,I_2)\mid C_2<C,\ m_1<|T_2|\}$，于是 $g_C(i_1)\not\subset T_3$。又因为 $C_1<C_3<C$，所以 $g_C(i_1)\subset T_3$，这与 $g_C(i_1)\not\subset T_3$ 矛盾。所以 C_1 是 C 的子结点。

定义 13 设 $I\subseteq\boldsymbol{J}$，对于给定的最小支持度 $minsupp\in[0,1]$ 和最小置信度 $minconf\in[0,1]$，项集 I 的支持记数为：

$$support(I)=\frac{|g(I)|}{|\boldsymbol{I}|},$$

如果 $support(I)\ge minsupp$，我们称 I 是频繁项集。

定义 14 设 $I\subseteq\boldsymbol{J}$，称 $I_1\to I_2$ 是一个关联规则，其中 $I_2\neq\varnothing$，I_1,I_2 分别称为规则的前件和后件。一个关联规则 $r=I_1\to I_2$ 的支持度和置信度分别定义为

$$support(r)=\frac{|g(I_1\cup I_2)|}{|\boldsymbol{I}|},$$

$$confidence(r)=\frac{support(I_1\cup I_2)}{support(I_2)}$$

关联规则的挖掘就是要找出所有 $support(r)\ge minsupp$ 并且 $c\quad(o)\ge$ 的所有关联规则。

3 Arca 算法

我们在生成概念格时，只生成这些概念 $C=(T,I)$，这里的 $|T|\ge mins$，例如当 $mins u=p0p.$ 时，所生成的概念格如图 2 所示。

然后，从生成的概念格提取出一些关联规则出来。我们把这些关联规则叫做基本关联规则。具体方法如下：对于每一对 $C_1=(T_1,I_1)$ 和 $C_2=(T_2,I_2)$，$C_1\prec C_2$ 来说，如果 $\frac{|T_1|}{|T_2|}\ge minconf$，那么 $I_2\to I_1-I_2$ 就是一条基本的关联规则。且这条规则是最小置信度（也就是说，这条规则的置信度总是大于等于它的最小置信度的。）是 $\frac{|T_1|}{|T_2|}$。

下面我们定义 4 个关于关联规则的算子，基本关联规则和这 4 个算子可以算出所有要求的关联规则。

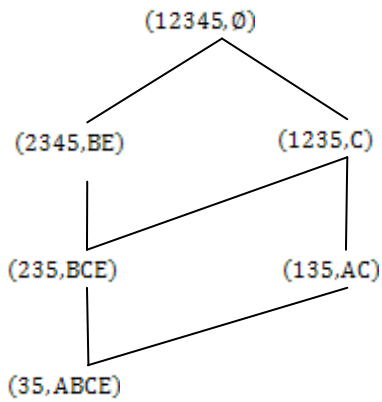

图 2 $minsupp=0.4$ 时，所生成的概念格

表 3 $minconf=0.5$ 从图 2 提取的基本关联规则

	基本关联规则	最小置信度
①	$\varnothing\to BE$	4/5
②	$\varnothing\to C$	4/5
③	$BE\to C$	3/4
④	$C\to BE$	3/4
⑤	$C\to A$	3/4
⑥	$BCE\to A$	2/3
⑦	$AC\to BE$	2/3

定义 15 设 $r_1 = I_1 \to I_2$，$r_2 = I_3 \to I_4$，最小置信度分别为 $conf_1, conf_2$。如果 $I_1 \cup I_2 = I_3$，那么 r_1 和 r_2 可进行“+”运算。$r_1 + r_2 = I_1 \to I_4$，最小置信度 $conf_{12} = conf_1 \times conf_2$。设 $r_3 = I_5 \to I_6$，最小置信度为 $conf_3$。如果 $I_1 \cup I_2 = I$ 并且 $I_3 \cup I_4 = I$，那么 r_1、r_2 和 r_3 可进行“+”运算。$r_1 + r_2 + r_3 = I_1 \to I_6$，最小置信度 $conf_{123} = conf_1 \times conf_2 \times conf_3$。同样可以定义 n 个规则的“+”运算。

定义 16 设 $r_1 = I_1 \to I_2$，$r_2 = I_3 \to I_4$，最小置信度分别为 $conf_1, conf_2$。如果 $I_1 \cup I_2 = I_3$，那么 r_1 和 r_2 可进行“⊕”运算。$r_1 \oplus r_2 = I_1 \to I_2 \cup I_4$，最小置信度 $conf_{12} = conf_1 \times conf_2$。设 $r_3 = I_5 \to I_6$，最小置信度为 $conf_3$。如果 $I_1 \cup I_2 = I$ 并且 $I_3 \cup I_4 = I$，那么 r_1、r_2 和 r_3 可进行“⊕”运算。$r_1 \oplus r_2 \oplus r_3 = I_1 \to I_2 \cup I_4 \cup I_6$，最小置信度 $conf_{123} = conf_1 \times conf_2 \times conf_3$。同样可以定义 n 个规则的“⊕”运算。

例如：对表 3 所示的基本关联规则进行“+”和“⊕”运算。因为①的前件∪①的后件 $= \varnothing \cup BE = BE \neq \varnothing =$ ②的前件，所以①和②不能进行“+”运算。因为①的前件∪①的后件 $= \varnothing \cup BE = BE$ ③的前件，所以①和③可以进行“+”运算。

$$①+③ = \varnothing \to C$$

因为“+”运算和“⊕”运算的条件是相同的，所以①和③可以进行“⊕”运算。

$$①⊕③ = \varnothing \to BCE$$

然后再从③开始向后逐个检查每一条基本关联规则是否可以与③进行“+”运算。④和⑤不可以，⑥可以。于是有

$$③+⑥ = BE \to A$$

$$①+③+⑥ = \varnothing \to A$$

同样，进行“⊕”运算有

$$③⊕⑥ = BE \to AC$$

$$①⊕③⊕⑥ = \varnothing \to ABCE$$

对表 3 所示的基本关联规则进行“+”和“⊕”运算的结果如表 4 所示。

表 4 对表 3 所示的基本关联规则进行“+”和“⊕”运算的结果

基本规则	“+”运算生成的规则	“⊕”运算生成的规则
① ③	$\varnothing \to C$	$\varnothing \to BCE$
③ ⑥	$BE \to A$	$BE \to AC$
① ③ ⑥	$\varnothing \to A$	$\varnothing \to ABCE$
② ④	$\varnothing \to BE$	$\varnothing \to BCE$
④ ⑥	$C \to A$	$C \to ABE$
② ④ ⑥	$C \to A$	$C \to ABCE$
② ⑤	$\varnothing \to A$	$\varnothing \to AC$
⑤ ⑦	$C \to BE$	$C \to ABE$
② ⑤ ⑦	$\varnothing \to BE$	$\varnothing \to ABCE$

[定义 3.8] 设 $r_1 = I_1 \to I_2$，对 r_1 进行“右移”运算可以产生规则集 $\left\{ I_1' \to I_2 \cup I_1'' \middle| \begin{matrix} I_1' \cap I_1'' = \varnothing, I_1' \cup I_1'' = I, I_1' \neq \varnothing, \\ I_1'' \neq \varnothing, g(I_1) \le g(I_1'') \end{matrix} \right\}$，最小置信度不变。

[定义 3.9] 设 $r_1 = I_1 \to I_2$，对 r_1 进行“分解”运算可以产生规则集 $\{ I_1 \to I_2' \mid I_2' \not\subset \varnothing, I_2' \subset I_2 \}$，最小置信度不变。

对基本关联规则运用以上 4 个算子，产生出来的规则，并对这些新产生的规则运用“右移”运算和“分解”运算，这样可以得到所有置信度小于 1 的关联规则。具体算法见算法 1。

Algorithm Arca($\mathbf{D}$, min_supp, min_conf)

输入：最小支持度*min_supp*、最小置信度*min_conf*和数据挖掘背景 $\mathbf{D} = (\mathbf{I}, \mathbf{J}, \mathbf{R})$

输出：关联规则集合*Rules*

```
1  Rules ← ∅;
2  BasicRules ←
     gen_basic_rules(D, min_supp, min_conf);
3  foreach r in BasicRules
4    Rules ← Rules ∪ r;
5    add_combination(Rules); //对 Rules 中的
     每条规则进行“加”和“组合”这两个算子操
     作
6  loop
```

```
7   foreach r in BasicRules;
8     Rules ← Rules ∪ right_move(r); //对
      Rules 中的每条规则进行“右移”算子操作
9   loop
10  foreach r in BasicRules;
11    Rules ← Rules ∪ decompose(r); //对
      Rules 中的每条规则进行“分解”算子操作
12  loop
13  return Rules;
```

算法 1 Arca 算法

算法 1 是先调用算法 2 利用概念格来生成基本规则表，然后在基本规则表上实施“加”算子和“组合”算子来生成关联规则，最后在生成的关联规则上实施“右移”算子和“分解”算子从而得到所有关联规则。

在所有的 $Rules \leftarrow Rules \cup r$ 过程中，如果 r 已经在 $Rules$ 中，则需要更新 r 的相应的规则最小置信度。把 $Rules$ 中的相应的规则最小置信度和 r 的最小置信度作比较，取较大的作为 $Rules$ 中的相应的规则最小置信度。对于 $Rules \leftarrow Rules \cup$ right_move (r) 和 $Rules \leftarrow Rules \cup$ decompose (r) 也是这样。

Algorithm gen_basic_rules($\boldsymbol{D}$, min_supp, min_conf)

输入：最小支持度*min_supp*、最小置信度*min_conf*和数据挖掘背景 $\boldsymbol{D} = (\boldsymbol{I}, \boldsymbol{J}, \boldsymbol{R})$

输出：关联规则集合*Rules*

```
1   初始化一个队列 Queue;
2   BasicRules ← ∅
3   Queue.EnQueue((I, ∅));
4   do while Queue ≠ ∅
5     C = Queue.DeQueue;
6     foreach C1 in SUBNODES*(C)
7       if |Extent(C1)| / |Extent(C)| ≥ min_conf then
8         if C1 ∉ Queue then
              Queue.EnQueue(C1);
9         生成规则 r : Intent(C) → Intent(C1);
10        BasicRules ← BasicRules ∪ r;
11      end if
12    loop
13  loop
14  return BasicRules;
```

算法 2 生成基本关联规则的算法

Algorithm SUBNODES*(C, $\boldsymbol{D}$, min_supp)

输入：给定的概念 $C = (T, I)$、数据挖掘背景 $\boldsymbol{D} = (\boldsymbol{I}, \boldsymbol{J}, \boldsymbol{R})$ 和最小支持度*min_supp*

输出：概念C的所有外延里的元素个数大于等于*min_supp*的子结点集合*subnodes*

```
1   subnodes ← ∅;
2   M ← C对应的概念矩阵;
3   计算各属性在概念矩阵M中的秩;
4   m ← C的秩;
5   do while m ≥ min_supp
6     S ← 秩为m的属性集合;
7     do while S ≠ ∅
8       I1 ← 从S中取出一个属性，并且从S中取
          出和这个属性在M中具有相同列的属
          性组成的集合;
9       S ← S − I1;
10      T1 ← gC(I1);
11      I1 ← I ∪ I1;
12      if 对于 subnodes 中的每个概念
            C2 = (T2, I2) 都有 T1 ⊄ T ∩ T2 then
            subnodes ← subnodes ∪ (T1, I1);
13    loop
14    m ← m − 1;
15  loop
16  return subnodes;
```

算法 3 SUBNODES*算法

4 实验结果

为了进行实验评价,我们使用 Visual C++ 6.0 以及 STL 实现了上述算法和 Aprior 算法. 在算法的实现过程中,使用了一些特殊的数据结构对算法进行优化,以获得最好的时间性能。通过自定义的位集合类(BitSet)及其操作来提高集合之间运算的速度.

位集合类是将集合看成是内存中某个连续的空间,集合中的每个元素对应于这段内存空间中的某些位.这样,集合之间的操作就可以用内存中位运算来实现,从而提高集合运算的速度。对于 Aprior 算法的 k-项集，使用一个有序数组，在查找元素时，使用了二分查找法。

对比实验在CPU为迅驰1.7G,内存1GB,操作系统为 Windows XP SP2 的 IBM ThinkPad T42 笔记本上进行。使用的测试数据是通过随机生成的一个事务数据库，项数 1000，事务数 10000。在最小置信度为 1%的条件下，实现结果如图 3 所示。

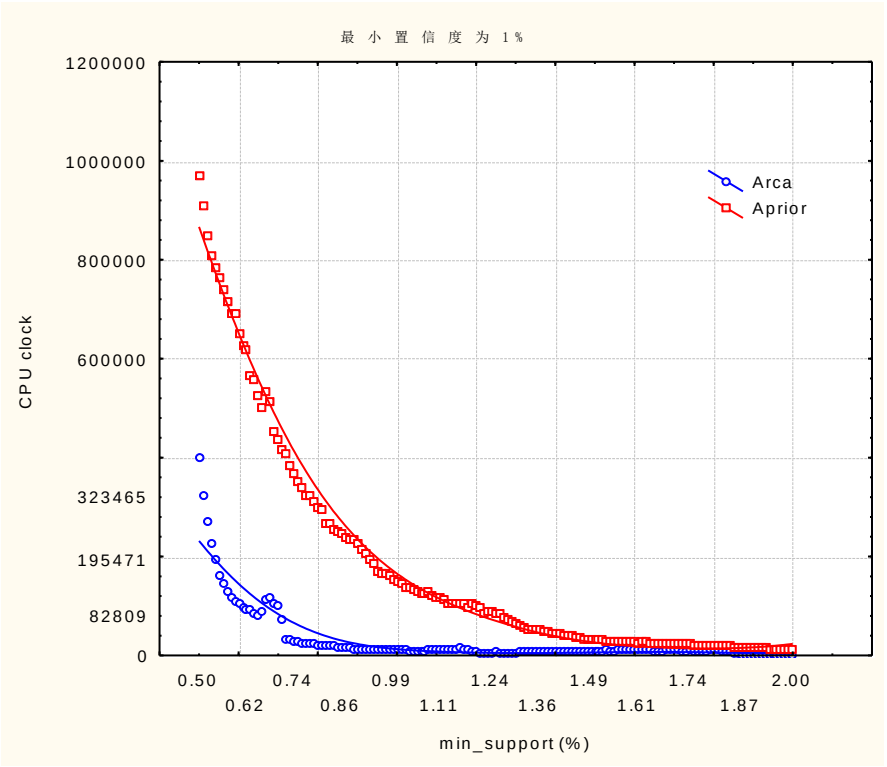

图 3 最小置信度为 1%的条件下 Arca 算法和 Aprior 算法的运行时间

参考文献:

[1] Agrawal R, Imielinski T, Swami A. Mining association rules between sets of items in large databases[J]. ACM SIGMOD Record: 1993, 22: 207-216

[2] Savasere A, Omiecinski E, Navathe S. An efficient algorithm for mining association rules in large databases[C]. Proceedings of the 21st VLDB Conference. the 21st VLDB Conference, Zurich, 1995. 432-444

[3] Park J S, Chen M S, Yu P S. An effective hash-based algorithm for mining association rules[C]. ACM SIGMOD Record . ACM Press, 1995, 24: 175-186.

[4] Zaki M J, et al. New algorithms for fast discovery of association rules[R]. University of Rochester : 1997.

[5] Shenoy P, et al. Turbo-charging vertical mining of large databases[C]. ACM SIGMOD Record. ACM Press, 2000, . 22-33..

[6] Pasquier N, et al. Discovering frequent closed itemsets for association rules[C]. Proceeding of the 7th International Conference on Database Theory . The 7th International Conference on Database Theory, Jerusalem, 1999. Berlin: Springer-Verlag, 1999, 1540: 398-416.

[7] Pei J, Han J, Mao R. An efficient algorithm for mining frequent closed itemsets[C]. 2000 ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery, Dallas, 2000. 21-30.

[8] Han J, et al. FreeSpan: frequent pattern-projected sequential pattern mining[C]. Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining. The sixth ACM SIGKDD

international conference on Knowledge discovery and data mining, 2000. Boston : ACM Press, 2000, 355-359.

[9] Han J, et al. PrefixSpan: mining sequential patterns efficiently by prefix-projected pattern growth[C]. The Proceedings of 17th International Conference. The17th International Conference, .2001. 215-224

[10] Han J, Pei J, Yin Y. Mining frequent patterns without candidate generation[C]. 2000 ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery, Dallas, 2000. 1-12.

[11] Godin R, Missaoui R. An incremental concept formation approach for learning from databases[J]. Theoretical Computer Science: 1994, 133: 387-419

[12] Valtchev P, Missaoui R, Lebrun P. A partition-based approach towards constructing Galois (concept) lattices[J]. Discrete Mathematics: 2002, 256(3):801-829.

[13] Pasquier N, Bastide Y, Taouil R, Lakhal L. Discovering frequent closed itemsets for association rules[C]. Proceeding of the 7th International Conference on Database Theory . The 7th International Conference on Database Theory, Jerusalem, 1999. Berlin: Springer-Verlag, 1999, 1540: 398-417

[14] Stumme G. Efficient data mining based on formal concept analysis[C]. Proceeding of the 13th International Conference. Database and Expert Systems Applications : 13th International Conference, DEXA 2002 Aix-en-Provence, 2002. Berlin: Springer-Verlag, 2453:534-547

[15] Pasquier N, Bastide Y, Taouil R, Lakhal L. Efficient mining of association rules using closed lattices[J]. Information System: Elsevier, 1999, .24(1): 25-46

[16] Stumme G, Taouil R, Bastide Y, Pasquier N, Lakhal L. Fast Computation of Concept Lattices Using Data Mining Techniques[C]. Proceeding of 7th Intl. Workshop on Knowledge Representation Meets Databases (KRDB'00). The 7th Intl. Workshop on Knowledge Representation Meets Databases (KRDB'00), Berlin, 2000.

[17] Wang S, Chen Z, Wang D. An Algorithm based on Concept Matrix for Building Concept Lattice with Hasse[C]. Proceeding of 2007 International Symposium on Information Systems & Management. 2007 International Symposium on Information Systems & Management, Shanghai, 2007.