基于双空间搜索的频繁项挖掘方法*)

Dual Space Search Algorithms for Mining Frequents

王晓峰[1,2] 王天然[2]

(沈阳化工学院计算机系 沈阳110021)[1] (中国科学院沈阳自动化所 沈阳110003)[2]

Abstract The famous Apriori algorithm proposed by Agrawal has become a classic algorithm for mining association rules, it is widely applied to various fields such as trade decision-making, bank evaluating credit, finance insurance etc. This approach is an effective down-top algorithm for mining frequent, but it will fall across time-consuming huge computing problems in mining long pattern frequent itemsets (as 100 items). We propose a dual space search algorithm for mining frequent items. Some of new conceptions such as transaction space and items space, mutuality set mapping and dual space search etc. are proposed, Proposed algorithms can better solve problem of mining long frequent, the validity of proposed algorithms is proved through theory analysis. Some examples of computation also given. Using transaction and itemsets information for pruning useless candidate frequent itemset, dual space search algorithm has higher efficiency than Apriori algorithm.

Keywords Dual space search, Data mining, Frequent item, Associate rules, Mutuality sets

1 引言

1998年 Roberto J. 和 Bayardo Jr.[7]利用自底向上搜索和项目集排序的方法建立了一种挖掘长型频繁项的 Max-Miner 算法；Lin D. 和 Z. Kedem[6]提出了一种双向钳形搜索 Pincer-Search 方法，利用自底向上搜索产生的非频繁项集来约束和修剪自顶向下方向的最大候选频繁项集，候选频繁项集来自于 Apriori 方法。这两种方法虽然细节有所不同，但修剪最大频繁项的思想类似。假设{{1,2,3,4,5,6}}是最大候选频繁项 MFCS(maximum-frequent- candidate-set)，{1,6}和{3,6}是新发现的非频繁项集，对 MFCS 考虑{1,6}得{{1,2,3,4,5}，{2,3,4,5,6}}。再用{3,6}更新这个 MFCS：由于{3,6}是{2,3,4,5,6}的子集，用{2,3,4,5}和{2,4,5,6}取代{2,3,4,5,6}，但{2,3,4,5}是{1,2,3,4,5}的子集，所以应删除，最后 MFCS 为{{1,2,3,4,5}，{2,4,5,6}}。在 Max-miner 中使用 Hass 树提高搜索效率；在 Pincer-Search 方法中用超集(Superset)和非频繁项集建立自顶向下的最大候选频繁项集。

Jawei Han 等人针对挖掘大频繁项的困难提出了一种 FP-树(Frequent Patterns tree)方法[10]，主要是采用分而治之的思想，利用扩展前缀树这种数据结构存储经过压缩的频繁项关键信息，按 FP-树的生长趋势划分和投影数据库。

据文[10,11]介绍，这种方法对长、短频繁项都有较好的挖掘效果。不足之处是：(1)对大型数据库，FP-树仍嫌过大且要扫描两次数据库；(2)每次挖掘都要事先选择一个合适的最小支持度，构造一次 FP-树，这是一件很困难的事情，最小支持度对挖掘效果和算法的效率有直接的影响，每次构造 FP-树需要耗去大量的存储空间和时间。

在文[14]中我们根据相关集合提出了一种基于集合交集运算的支持度和信任度的定义，其主要思路如下：

事务数据库(也称事务集)是事务的集合，一个事务 T 是一个对象(或记录)，每个事务有交易时间、标识符、顾客购置的商品(X,Y,…)等项内容。设 $D=\{t_1,t_2,\cdots,t_m\}$ 是事务的集合，t_i 是一个具体的事务，每一个事务有唯一的标识，记作 TID。这样，一种具体的商品 X 是事务 T 的一个属性值，反过来对指定的事务集来说，X 也可看作为由 T 组成的集合。例如，$X=\{T_1,T_2,\cdots,T_n\}$，由所有含商品 X 的 n 个事务构成的集合，每个元素是一个包含有商品 X 的事务标识。所以，X 是所有与 X 有关系的事务集合，简称相关集合。$|X|$是含有商品 X 的事务数。

关联规则是形如 $X\Rightarrow Y$ 的蕴涵式，X,Y 代表交易商品，同时 X,Y 也是事务集合，$X\subseteq D$，$Y\subseteq D$。基于集合的支持度和可信度定义如下：

规则 X⇒Y 在事务数据库 D 中的支持度是事务库中同时包含 X 和 Y 的事务数与所有事务数之比，记为 support(X⇒Y)，即

$$\text{support}(X\Rightarrow Y)=|X\cap Y|/|D|$$

规则 X⇒Y 在事务数据库中的可信度是指包含 X 和 Y 的事务数与包含 X 的事务数之比，记为 confidence(X⇒Y)，即

$$\text{confidence}(X\Rightarrow Y)=|X\cap Y|/|X|$$

给定一个事务数据库 D，挖掘关联规则问题就是产生支持度和可信度分别大于用户给定最小支持度和最小可信度的关联规则。

注意1：这里，X 和 Y 是事务的集合而不是商品集，所以一般 $X\cap Y\neq\phi$。

注意2：X 和 Y 本身又代表具体的商品，是一种或几种商品标识构成的符号串，X 和 Y 不重复。例如 X=x1x2代表标识符为 x1、x2的两种商品，从集合上说是 X 又是 x1、x2两个事务集的交集。

比较关于支持度和信任度的两种定义不难发现，虽然表面上看两种定义不一样，但从定义的含义(逻辑结果)上看，其本质是一样的。两种定义的主要差别在于支持度、信任度的计

*)获辽宁省自然科学基金(9910200205)和辽宁省教育厅高校科研基金(20012073)资助。王晓峰 博士生，研究方向：智能信息处理，数据挖掘。王天然 研究员，博士生导师，研究方向：智能机器人。

算过程，比如支持度计算，在前者包含 X 和 Y 的事务数的统计是按集合的定义计算的，在后者包含 X 和 Y 的事务数是以两个集合的交集形式给出来的，但最终结果完全一样[14]。

在事务数据库中挖掘频繁项，就是要发现满足用户最小支持度 minsupp 的所有项目集，特别是发现最大频繁项集。这一任务包含两方面的要求：事务组合数的要求和项目组合的要求；

支持度的本质是对事务组合数的要求，最大频繁项是对项目组合的要求。挖掘最大频繁项就是发现满足条件：

（事务项的组合下界≥minsupp）∩（项目组合长度>0）的所有项目集合。

所以挖掘频繁项，特别是挖掘大频繁项要充分利用事务项集和项目集两个集合的有用信息，压缩频繁项的搜索空间，才能提高挖掘效率。本文主要讨论这种基于两空间搜索的挖掘方法。

2 主要概念与定义

搜索空间：设 t 是由 n 个事务项（标识）组成的一个事务项集合，称其幂集 $P(t)$ 为 T 搜索空间，简称事务空间或 T 空间；设 x 是由 m 个物品项目（标识）组成的一个项目集合，称其幂集 $P(x)$ 为 X 搜索空间，简称项目空间或 X 空间。

X→T 映射：给定项目空间 X 和事务空间 T，称映射 f：X→T 是 X 到 T 的集合映射，简记为 X→T 映射。

T→X 映射：给定项目空间 X 和事务空间 T，称映射 g：T→X 是 T 到 X 的集合映射，简记为 T→X 映射。

注意：从上面定义看，映射 f 和 g 是互逆的两个映射，但是一般情况下，T 和 X 包含的元素数是不同的。若 x、t 的元素个数分别是 m、n，那么 f 是一个为 2^m 到 2^n 的集合映射，g 是一个 2^n 到 2^m 的集合映射。

项目集的含义与本质：若 x 是一个实际长度为 p、支持度为 s 的物品项目集（简称项目集），那么，x 可以被看作为项目 $x_1x_2\cdots x_p$ 的组合，是 X 空间的一个 p 维向量，也是一个 $x=\{x_1,x_2,\cdots,x_p\}$ 的项目集合；同时，在 T 空间中有 s 个事务支持 x，$f(x)=\{t_1,t_2,\cdots,t_s\}$ 是 x 所对应的事务集合。这里，$f(x)$ 是 X 到 T 的集合映射。

事务集与项目集的关系：任给一个项目集合，利用 X 到 T 的映射 $f(x)$，必有一个事务集与之对应；反之给定一事务项集 $t=\{t_1,t_2,\cdots,t_q\}$，利用 T→X 映射可以求出 t 对应的所有相关项目集 x：$g(t)=\{x_1,x_2,\cdots,x_p\}$。其中，p 为项目集长度，q 为支持度，$g(t)$ 是 T 到 X 的映射。

事务数据库的本质：一个给定的事务数据库本质上是建立在 T、X 两个空间上的一个关系。设项目集 $X=\{x_1,\cdots,x_m\}$，事务集 $T=\{t_1,\cdots,t_n\}$，那么事务数据库就是定义在 $T\times X$ 上的一个关系 R，$R\subseteq T\times X$。给定 $t_i\in T$，$x_j\in X$，有 $\langle t_i,x_j\rangle\in R$。

事务相关集和项目相关集：设项目集 $X=\{x_1,\cdots,x_m\}$，事务集 $T=\{t_1,\cdots,t_n\}$，事务数据库 $R\subseteq T\times X$。给定 $t_i\in T$，有 $x_j\in X$，$\langle t_i,x_j\rangle\in R$ 成立。把所有与 t_i 有关系的项目用集合表示出来，有集合 $[t_i]_R=\{x_1,\cdots,x_p\}$，称为事务 t_i 的相关集合；同理，把所有与 x_j 有关系的事务项用集合表示出来，得 $[x_j]_R=\{t_1,\cdots,t_q\}$ 集合，被称为项目 x_j 的相关集合。

事务数据库的 T、X 集合映射：给定一个事务数据库，如果项目集 $x=\{x_1,x_2,\cdots,x_p\}$，每个项目相关集 $[x_j]_R=\{t_{j1},\cdots,t_{jr}\}$，$j=1,\cdots,p$，令 $f(x)=[x_1]_R\cap[x_2]_R\cap\cdots\cap[x_p]_R=\{t_1,\cdots,t_q\}$，那么 $\{t_1,\cdots,t_q\}$ 是 x 的一个相关集合，f 是事务数据库的 X→T 映射；

同理，如果 $t=\{t_1,t_2,\cdots,t_s\}$，令 $g(t)=[t_1]_R\cap[t_2]_R\cap\cdots\cap[t_s]_R=\{x_1,\cdots,x_k\}$，则 $\{x_1,\cdots,x_k\}$ 是事务集 t 的一个相关集合，g 是事务数据库的 T→X 映射。

比较前面关于 T 与 X 空间映射的定义可知，这是两个空间映射的具体实现。

事务数据库，T、X 两空间及其集合映射之间的关系，概括起来如图1所示。

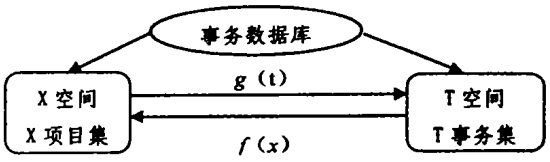

图1 事务数据库，T、X 两空间及其集合映射之间的关系

事务组合构成了一个事务搜索空间（或事务域），用 T 表示；项目组合构成了一个项目搜索空间（或项目域），用 X 表示。挖掘频繁项要在 T、X 两个空间中同时搜索满足条件的目标。所以应该同时压缩两个搜索空间才能获得最佳搜索路径。

3 T、X 集合映射的性质

3.1 T、X 集合映射的基本性质

设 R 是事务数据库，t 是事务集合，x 是项目集合。T、X 集合映射的基本性质如下：

性质1 如果 $x=x_1\cup x_2$，那么 $f(x)=f(x_1)\cap f(x_2)$。

证明：设 $x_1=\{x_{11},x_{12},\cdots,x_{1p}\}$，$x_2=\{x_{21},x_{22},\cdots,x_{2q}\}$，那么 $x=x_1\cup x_2=\{x_{11},x_{12},\cdots,x_{1p},x_{21},x_{22},\cdots,x_{2q}\}$，$f(x)=f(x_1\cup x_2)=f(\{x_{11},x_{12},\cdots,x_{1p},x_{21},x_{22},\cdots,x_{2q}\})=[x_{11}]_R\cap[x_{12}]_R\cap\cdots\cap[x_{1p}]_R\cap[x_{21}]_R\cap[x_{22}]_R\cap\cdots\cap[x_{2q}]_R=([x_{11}]_R\cap[x_{12}]_R\cap\cdots\cap[x_{1p}]_R)\cap([x_{21}]_R\cap[x_{22}]_R\cap\cdots\cap[x_{2q}]_R)=f(x_1)\cap f(x_2)$，其中，$f(x_1)=[x_{11}]_R\cap[x_{21}]_R\cap\cdots\cap[x_{1p}]_R$，$f(x_2)=[x_{21}]_R\cap[x_{22}]_R\cap\cdots\cap[x_{2q}]_R$。

性质2 如果 $t=t_1\cup t_2$，那么 $g(x)=g(t_1)\cap g(t_2)$。

证明略。

性质3 如果 $x_1\subseteq x_2$，那么 $f(x_2)\subseteq f(x_1)$；反之如果 $f(x_2)\subseteq f(x_1)$，那么 $x_1\subseteq x_2$。

证明略。

显然，T、X 空间存在一种对偶关系，一个项目集如果在 X 空间中增大其长度，那么在 T 空间中的支持度将减小；反之，如果在 T 空间事务项增长，那么对应 X 空间中的项目长度将减小。

定理1 设 R 是事务数据库，t 是事务集合，x 是项目集合。如果 $t\neq\phi$，$g(t)=x\neq\phi$，那么 $t\subseteq f(x)=f(g(t))$；反之，如果 $x\neq\phi$，$f(x)=t\neq\phi$，那么 $x\subseteq g(t)=g(f(x))$。

证明：(1)证明 $t\subseteq f(x)=f(g(t))$。设 $t=\{t_1,t_2,\cdots,t_s\}$，$g(t)=[t_1]_R\cap[t_2]_R\cap\cdots\cap[t_s]_R=\{x_1,\cdots,x_k\}\neq\phi$。则，$\{x_1,\cdots,x_k\}\subseteq[t_1]_R$，$\{x_1,\cdots,x_k\}\subseteq[t_2]_R$，$\cdots$，$\{x_1,\cdots,x_k\}\subseteq[t_s]_R$，任意一个 t_i 与 $x_1,\cdots,x_k$ 都有 R 关系（$i=1,\cdots,s$）。反过来说，每一个 x_j 都与 $t_1,\cdots,t_s$ 有 R 关系（$j=1,\cdots,k$）。所以，$\{t_1,t_2,\cdots,t_s\}\subseteq[x_1]_R$，$\{t_1,t_2,\cdots,t_s\}\subseteq[x_2]_R$，$\cdots$，$\{t_1,t_2,\cdots,t_s\}\subseteq[x_k]_R$，即：$t=\{t_1,t_2,\cdots,t_s\}\subseteq[x_1]_R\cap[x_2]_R\cap\cdots\cap[x_k]_R$；$[x_1]_R\cap[x_2]_R\cap\cdots\cap[x_k]_R=f(x)$，所以 $t\subseteq f(x)=f(g(t))$ 成立。

(2)证明 $x\subseteq g(t)=g(f(x))$。设 $x=\{x_1,x_2,\cdots,x_p\}$，f

$(x)=[x_1]_R\cap[x_2]_R\cap\cdots\cap[x_p]_R=\{t_1,\cdots,t_s\}\neq\phi$。则$\{t_1,\cdots,t_s\}\subseteq[x_1]_R,\cdots,\{t_1,\cdots,t_s\}\subseteq[x_p]_R$，$t$ 中的每一个元素 t_i 都与 $x_1,\cdots,x_p$ 有 R 关系$(i=1,\cdots,s)$。所以，$\{x_1,x_2,\cdots,x_p\}\subseteq[t_1]_R$，$\{x_1,x_2,\cdots,x_p\}\subseteq[t_2]_R,\cdots,\{x_1,x_2,\cdots,x_p\}\subseteq[t_s]_R$，即：$x=\{x_1,x_2,\cdots,x_p\}\subseteq[t_1]_R\cap[t_2]_R\cap\cdots\cap[t_s]_R$。但，$[t_1]_R\cap[t_2]_R\cap\cdots\cap[t_s]_R=g(t)$，所以 $x\subseteq g(t)=g(f(x))$成立。证毕。

定理1说明在 T、X 集合映射过程中支持度不会减小，项目长度也不会减短。也就是说，若 x 不变，则$|f(x)|$是 x 的最大支持度；若 t 不变，$g(t)$是 t 对应的最大项目集。下面根据定理1，考虑 T、X 集合映射与频繁项的关系。

3.2 T、X 集合映射与频繁项的关系

给定一个事务数据库的项目集合 $x=\{x_1,x_2,\cdots,x_p\}$：

(1)若 $f(x)=[x_1]_R\cap[x_2]_R\cap\cdots\cap[x_p]_R=\{t_1,\cdots,t_q\}=t$，$t$ 是 x 的一个相关事务集合，那么 q 是 x 的最大支持度，x 是具有最大支持度 q 的项目集，t 是支持 x 的最大事务集。但 x 与 t 不是1—1的，也可能有 y，使得 $f(y)=t$ 成立。

(2)若用户给定一个最小支持度 *minsupp*，那么根据 *minsupp* 的大小有下面几种情况：①*minsupp*＞|f(x)|，因 x 的最大支持度小于给定最小支持度，所以 x 不是频繁项；② *minsupp*＝|f(x)|，x 的最大支持度刚好满足给定最小支持度，x 是频繁项。如果 x 不再增大，那么 x 是一个最大频繁项。③ *minsupp*＜|f(x)|，此时，x 是频繁项，但通常不是最大频繁项。这意味着有包含 x 的频繁项集存在。

给定一个事务数据库 g 的事务项集合 $t=\{t_1,t_2,\cdots,t_s\}$：

(1)若 $g(t)=[t_1]_R\cap[t_2]_R\cap\cdots\cap[t_s]_R=\{x_1,\cdots,x_p\}=x$，$x$ 是 t 的一个相关项目集合，那么 s 是支持度，x 是支持度至少为 s 的项目集，t 是支持 x 的最小事务集。注意 x 与 t 不是1—1的，也可能有 t'，使得 $g(t')=x$ 成立。

(2)若用户给定一个最小支持度 minsupp，那么根据 minsupp 的大小有下面几种情况：④minsupp＞$|t|$，x 的最小支持度小于给定最小支持度。如果 t 是整个事务空间，则 x 不是频繁项；如果 t 只是事务集的一部分，则 x 不一定不是频繁项；⑤minsupp＝$|t|$，x 的最小支持度刚好满足给定最小支持度，x 是频繁项。如果 t 是整个事务空间，x 是一个最大频繁项，否则不定；⑥minsupp＜$|t|$，x 是频繁项。即使 t 是整个事务空间，也不一定是最大频繁项。这意味着可能有包含 x 的频繁项集存在。

注：x 是最大频繁项意味着 x 的任意子集也都是频繁项，任何包含 x 的项目集的支持度都要小于 minsupp。即不存在包含 x 的频繁项集。

3.3 T、X 集合映射与最大频繁项的计算

定理2 设 R 是事务数据库，x 是 R 的一个频繁项，如果$|f(x)|$＝minsupp，那么 $g(f(x))$是含有 x 的最大频繁项，并且是唯一的最大频繁项。其中，$f(x)$和 $g(t)$分别是 X→T 映射和 T→X 映射。

证明：(1)首先证明 $g(f(x))$是含有 x 的频繁项。设 minsupp 是用户给定的最小支持度，$x=\{x_1,x_2,\cdots,x_p\}$是事务数据库的一个频繁项。已知 $f(x)=[x_1]_R\cap[x_2]_R\cap\cdots\cap[x_p]_R=\{t_1,\cdots,t_s\}=t$，$s=$minsupp；令 $g(f(x))=g(t)=[t_1]_R\cap[t_2]_R\cap\cdots\cap[t_s]_R=\{x_1,\cdots,x_q\}=x'$，$x'$的支持度最小为$|t|=s$，所以$\{t_1,\cdots,t_s\}$是支持 x'的最小事务项集，x'是频繁项。又根据定理1有 $x\subseteq g(f(x))=g(t)=x'$，所以 x'是含有 x 的频繁项。

(2)证明 $g(f(x))$是最大频繁项。

假设 $g(f(x))=x'$不是最大频繁项，那么有频繁项集 y 存在，使得 $y=x'\cup z$，$z\cap x'=\phi$，$z\neq\phi$ 成立。设 $y=\{x_1,x_2,\cdots,x_q,z_1,\cdots,z_w\}$，因为 $f(x)=[x_1]_R\cap[x_2]_R\cap\cdots\cap[x_p]_R=\{t_1,\cdots,t_s\}=t$，$x\subseteq x'\subseteq y$，所以 $f(y)=[x_1]_R\cap[x_2]_R\cap\cdots\cap[x_q]_R\cap[z_1]_R\cap\cdots\cap[z_w]_R=[x_1]_R\cap[x_2]_R\cap\cdots\cap[x_p]_R\cap\cdots\cap[x_q]_R\cap[z_1]_R\cap\cdots\cap[z_w]_R\subseteq\{t_1,\cdots,t_s\}\cap[z_1]_R\cap\cdots\cap[z_w]_R\subseteq t$；但$|f(x)|=|t|=$minsupp，所以$|f(y)|\leqslant|f(x)|$。如果$|f(y)|<|f(x)|$，则$|f(y)|<$minsupp，$y$ 不是频繁项，这与 y 是频繁项集矛盾。

如果$|f(y)|=|f(x)|$，那么因 $f(y)=f(x'\cup z)=f(x')\cap f(z)=\{t_1,\cdots,t_s\}\cap[z_1]_R\cap\cdots\cap[z_w]_R\subseteq t$，$|f(y)|=|\{t_1,\cdots,t_s\}\cap[z_1]_R\cap\cdots\cap[z_w]_R|=|\{t_1,\cdots,t_s\}|$，所以 $f(y)=t$，$\{t_1,\cdots,t_s\}\subseteq[z_1]_R\cap\cdots\cap[z_w]_R$，$\supseteq\{t_1,\cdots,t_s\}\subseteq[z_j]_R(j=1,\cdots,u)$，即：$t_i(i=1,\cdots,s)$与 $z_1\cdots z_w$ 相关，注意到 $g(t)=x'=\{x_1,\cdots,x_q\}$，每一个 $t_i(t_i\in t,i=1,\cdots,s)$又都与 $x_1\cdots x_q$ 相关，所以 $g(t)=[t_1]_R\cap[t_2]_R\cap\cdots\cap[t_s]_R=g(f(y))$。但$\{x_1,\cdots,x_q,z_1,\cdots,z_w\}\subseteq g(t)$，比较 $g(t)=[t_1]_R\cap[t_2]_R\cap\cdots\cap[t_s]_R=\{x_1,\cdots,x_q\}=x'$，得：$z=\{z_1,\cdots,z_w\}=\phi$，$x'=y$。这与 $z\neq\phi$ 矛盾，因此，$g(f(x))=\{x_1,\cdots,x_q\}=x'$是最大频繁项。

(3) 证明唯一性。

设 $g(f(x))=x'$，若 y 是另外一个包含频繁项 x 的最大频繁项，$y\neq g(f(x))$。不妨设 $y=\{y_1,\cdots,y_q\}$，$x'=\{x_1,\cdots,x_p\}$，那么 $y\neq x'$，$y\cap x'=x\neq\varphi$；令 $y=x\cup\{y_1,\cdots,y_m\}$，因为 $g(f(x))=x'$是包含 x 的最大频繁项，其支持度＝$|f(x)|=|t|=$minsupp，设 $x'=x\cup\{x_1,\cdots,x_m\}$，$f(x')=f(x\cup\{x_1,\cdots,x_m\})=f(x)\cap f(\{x_1,\cdots,x_m\})=t\cap f(\{x_1,\cdots,x_m\})\subseteq t$；$|f(x')|=|t\cap f(\{x_1,\cdots,x_m\})|\leqslant|t|$；由于 x'是最大频繁项，其支持度≥minsupp＝$|t|$，因此$|f(x')|=|t|$，注意到 $f(x')\subseteq t$ 有 $f(x')=t$。同理，$f(y)=t$，所以 $f(x')=f(y)=t$。x'与 y 对应同一组相关事务集。因此，$g(f(x'))=g(t)=g(f(y))=g(f(x))=x'$，由定理1$y\subseteq g(f(y))$，$y\subseteq x'$，则 y 不是一个包含频繁项 x 的最大频繁项，与假设矛盾，所以 $y=g(f(x))=x'$，$g(f(x))=x'$是唯一一个满足用户最小支持度的最大频繁项。证毕。

定理3 设 R 是事务数据库，x_1，x_2是 R 的2个频繁项集，如果 $f(x_1)\subseteq f(x_2)$，那么 $x_1\cup x_2$是一个更大的频繁项集。其中，$f(x)$是 X→T 集合映射。

证明：设 $x_1=\{x_{11},x_{12},\cdots,x_{1p}\}$，$x_2=\{x_{21},x_{22},\cdots,x_{2q}\}$，则 $x=x_1\cup x_2=\{x_{11},x_{12},\cdots,x_{1p},x_{21},x_{22},\cdots,x_{2q}\}$，$f(x)=f(x_1\cup x_2)=f(x_{11},x_{12},\cdots,x_{1p},x_{21},x_{22},\cdots,x_{2q})=[x_1]_R\cap[x_2]_R=f(x_1)\cap f(x_2)$。因为 $f(x_1)\subseteq f(x_2)$，所以 $f(x)=f(x_1)\cap f(x_2)=f(x_1)$。又因为 x_1，x_2是频繁项集，support(x_1)大于用户给定的最小支持度，所以$|f(x)|=|f(x_1)|$大于最小支持度，$x=x_1\cup x_2$是频繁项。

推论1 设 R 是事务数据库，x_1，x_2是 R 的2个频繁项集，如果 $f(x_1)\subseteq f(x_2)$，$x=x_1\cup x_2$，那么 $g(f(x_1))$是支持度为$|f(x_1)|$且含有 $x_1\cup x_2$的大频繁项。其中，$f(x)$是 $X\to T$ 集合映射。

证明：由定理3的证明过程可知，$f(x)=f(x_1)\cap f(x_2)=f(x_1)$，所以 $g(f(x_1))=g(f(x))$。由定理2知 $g(f(x))$是含有 x 的频繁项，所以 $g(f(x_1))$是含有 $x_1\cup x_2$的大频繁项。

进一步应用这个推论可知：如果 x_1是频繁项，那么 $g(f(x_1))$是包括了所有支持度大于$|f(x_1)|$，且包含 x_1的频繁项集合。也就是说，在支持度为$|f(x_1)|$的条件下，所有包含 x_1

的频繁项的一定都在 $g(f(x_1))$中。这实际上是定理2的推广，对快速发现长频繁项具有重要意义。只要发现长频繁项的一部分，就可以通过 g 映射发现全部。

定理4 设 R 是事务数据库，t_1，t_2是 R 的2个事务项集，如果 $g(t_1)\subseteq g(t_2)$，那么 $t_1\cup t_2$是支持 $g(t_1)$的事务项集，或者说 $g(t_1)$的支持度为$|t_1\cup t_2|$。其中，$g(t)$是 T→X 集合映射。

证明：设 $x_1=g(t_1)$，$t=t_1\cup t_2$。因为 $g(t_1\cup t_2)=g(t_1)\cap g(t_2)$，$g(t_1)\subseteq g(t_2)$，所以 $g(t)=g(t_1\cup t_2)=g(t_1)\cap g(t_2)=g(t_1)$，所以 $g(t_1)$是 t_1的相关项目集，也是 $t=t_1\cup t_2$的相关项目集。因此 $x_1=g(t_1)$的支持度为$|t_1\cup t_2|$。

推论2 设 R 是事务数据库，t_1，t_2是 R 的2个事务项集，如果 $x=g(t_1)\cap g(t_2)\neq\phi$，那么 $t_1\cup t_2$是支持 $x=g(t_1)\cap g(t_2)$的事务项集，或者说 x 的支持度为$|t_1\cup t_2|$。其中，$g(t)$是 T→X 集合映射。

与定理2相对应的是如下定理：

定理5 设 R 是事务数据库，t_1是 R 的1个事务项集，如果$|f(g(t_1))|=$minsupp，那么 $g(t_1)$是由 t_1确定的最大频繁项，并且是唯一的最大频繁项。其中，$f(x)$和 $g(t)$分别是 X→T 映射和 T→X 映射。

推论3 设 R 是事务数据库，t_1是 R 的1个事务项集，如果$|f(g(t_1))|>$minsupp，那么 $g(t_1)$是由 t_1确定的频繁项，最大支持度是$|f(g(t_1))|$。其中，$f(x)$和 $g(t)$分别是 X→T 映射和 T→X 映射。

推论4 设 R 是事务数据库，t 是 R 的1个事务项集，$|f(g(t))|=$minsupp，$x=g(t)$是频繁项，如果 t_1是 R 的1个事务项集，t_2是包含 t_1的任意事务项集，$x_1=g(t_1)$，$x_2=g(t_2)$，如果 $x_1\subseteq x$，$x=g(t)$，$|f(g(t))|\geqslant$minsupp，那么 x 是包含 x_2的大频繁项集，也就是说 x 包含了由 t_1生成的所有频繁项集。

证明：因为 $x=g(t)$，$|f(g(t))|\geqslant$ minsupp，所以 x 是频繁项。又 $x_1\subseteq x$，x_1也是频繁项，又 $t_1\subseteq t_2$，得 $g(t_2)\subseteq g(t_1)$，所以 $x_2\subseteq x_1\subseteq x$，即 x_2是频繁项，x 是包含 x_2的大频繁项集。因为 t_2是包含 t_1的任意事务项集，也就是说所有含有 t_1的任意事务项集 t_2所对应的项目集 $g(t_2)$都是 x 频繁项的子集，x 包含了由 t_1生成的所有频繁项集。证毕。

推论4对实际计算非常有用。在事务项集合中，若已知 $x=g(t)$是频繁项，那么对所有满足 $g(t_i)\subseteq g(t)$的事务项集 t_i 都不用再增加事务项往下计算了，减少了很多不必要的搜索计算。

根据上面的定理有双空间频繁项挖掘方法。

4 双空间搜索挖掘方法

双空间是指项目空间 X 和事务项空间 T，双空间搜索挖掘就是指在 X、T 两个空间中搜索最大频繁项。根据搜索的次序可以分为项目优先搜索和事务优先搜索两种基本方法。项目优先搜索就是首先满足项目组合要求，从最小项目组合频繁项集开始，寻找最大的频繁项；事务优先搜索就是首先满足事务项组合要求，从最小事务项组合开始，寻找最大的频繁项集。下面分别介绍这两种基本方法。

4.1 项目优先挖掘算法

1 计算每一个项目的事务相关集：$[x_j]_R=\{t_{j1},\cdots,t_{jr}\}$，j=1，…，m；

2 计算每一个事务项的项目相关集：$[t_i]_R=\{x_{i1},\cdots,x_{ip}\}$，i=1，…，n；

3 设用户给定的最小支持度为 minsupp，删除$|[x_j]_R|<$ minsupp 的项目，$k=2$；

4 按 Apriori 方法把项目$[x_j]_R$组合成长度为 k 的频繁项集：$L_k=\{C_{k1},C_{k2},\cdots,C_{kq}\}$，$|L_k|=q$；

5 对 L_k 中的每个频繁项计算：$f(C_{ki})$(i=1，2，…，q)，按$|f(C_{ki})|$(频繁项所含事务项的多少)由小到大分成3类：

L_{k1}频繁项满足$|f(C_{ki})|=$minsupp(i=1，…，r1)，有 r1 个；

L_{k2}频繁项满足$|f(C_{kj})|=$minsupp+1(j=1，…，r2)，有 r2个；

L_{k3}频繁项满足$|f(C_{kl})|>$minsupp+1(l=1，…，r3)，有 r3个；

合起来 r1+r2+r3=q。

6 优化：从 L_{k2}、L_{k3}中分别减去包含 $C_{ki}(C_{ki}\in L_{k1})$的频繁项集；

```
for i =1 to r1
    for j =1 to r2
        if f(Lk1·Cki)⊆f(Lk2·Ckj)then Lk2:=Lk2−Ckj
    endfor
    for l=1 to r3
    if f(Lk1·Cki)⊆f(Lk3·Ckl)then Lk3:=Lk3−Ckl
    endfor
endfor
P:=|Lk2|  从 Lk3中减去包含 Ckj(Ckj∈Lk2)的频繁项集；
for j=1 to P
   for l=1 to r3
      if f(Lk2·Ckj)⊆f(Lk3·Ckl)then Lk3:=Lk3−Ckl
   endfor
endfor
```

7 分类处理频繁项集 L_k；

7.1 对 L_{k1}频繁项，计算 $g(f(C_{ki}))$，把映射结果添加到 L_k 中；

7.2 对 L_{k2}频繁项，如果 r2>=1那么

(1)把集合 $f(C_{kj})$组合成 r2个长度为 minsupp 的事务项(满足最小支持度的)相关集 t_i；

(2)对每个事务项集计算 $g(t_i)$(T→X 映射)，得项目相关集(频繁项)；

(3)同 L_k 中的频繁项相比较，如果结果没有新的频繁项，那么转7.3；

(4)对新频繁项，用 $f(g(t_i))$(X→T 映射)把项目相关集转换成事务相关集；

(5)再用 T→X 映射，事务相关集转换为项目相关集；并添加到 L_k 中；

7.3 对 L_{k3}频繁项，按 Apriori 方法把 L_{k3}频繁项组合成长度为 k+1的新频繁项；

8 如果新频繁项集非空，那么 k=k+1，转5，否则转9；

9 输出频繁项；

10 结束算法。

说明：

(1)项目相关集合$[x_j]_R$是所有含有项目 x_j 的事务标识符集合；事务相关集合$[t_i]_R$是所有与事务 t_i 相关的项目标识符集合。

(2)算法第5步分类根据是频繁项满足$|f(C_{ki})|=$minsupp 时，由定理2可知，此时 $g(f(C_{ki}))$是唯一的最大频繁项集。频繁项满足$|f(C_{kj})|>=$minsupp ($j=1,\cdots,r2$)时，$g(f(C_{kj}))$是频繁项但是不一定是最大的，把它一部分分解成$|f(C_{kj})|=$minsupp，直接计算最大频繁项。另一部分继续组合成更长的频繁项。

(3)优化的依据是定理2、定理3及其推论。如果 x_1是频繁

项，那么 $g(f(x_1))$是包括了所有支持度大于$|f(x_1)|$，且包含x_1的频繁项集合。也就是说，在支持度为$|f(x_1)|$的条件下，所有包含 x_1的频繁项的一定都在 $g(f(x_1))$中。

4.2 应用举例

例：设有事务数据库如图2所示，给定最小支持度为3，求所有最大频繁项集合。

TID	Items				
T1	x1	x2	x3	x4	x5
T2		x2	x3	x4	
T3			x3	x4	x5
T4	X1	x2			
T5	X1		x3		x5
T6	X1		x3	x4	x5

图2 事务数据库

解：(1)计算相关集合：

[x1]={t1,t4,t5,t6};[x2]={t1,t2,t4};

[x3]={t1,t2,t3,t5,t6};[x4]={t1,t2,t3,t6};

[x5]={t1,t3,t5,t6};

[t1]={x1,x2,x3,x4,x5};[t2]={x2,x3,x4};

[t3]={x3,x4,x5};[t4]={x1,x2};

[t5]={x1,x3,x5};[t6]={x1,x3,x4,x5};

(2)计算 k=2时的频繁项：L_k={x1x3,x1x5,x3x4,x3x5,x4x5};

(3)由映射 f 做 X-T 转换：

x1x3={t1,t5,t6},x1x5={t1,t5,t6},x3x4={t1,t2,t3,t6}

x3x5={t1,t3,t5,t6},x4x5={t1,t3,t6};

(4)排序、分类：L_{k1}={x1x3,x1x5,x4x5 };L_{k2}={x3x4,x3x5}。

(5)优化：因为 $f(L_{k1}\cdot x1x3)\subseteq f(L_{k2}\cdot x3x5)$，$f(L_{k1}\cdot x4x5)\subseteq f(L_{k2}\cdot x3x4)$，所以，$L_{k2}$={x3x4,x3x5 }−{x3x5}−{x4x5}=ϕ;

(6)分类处理。只有1类恰好满足最小支持度：L_{k1}={x1x3,x1x5,x4x5};

计算 T-X 映射：g(t1,t5,t6)={x1,x3,x5},g(t1,t3,t6)={x3,x4,x5};

注意：这里通过 T 空间到 X 空间的转换，直接求得了含有 x1x5,x3x4的最大频繁项：x1x3x5,x3x4x5，其效率要比 Apriori 算法高很多。

因此，最大频繁项为：{x1,x3,x5}和{x3,x4,x5};

全部频繁项为：{x1,x3},{x1,x5},{x4,x5},{x3,x4},{x3,x5},{x1,x3,x5},{x3,x4,x5}。

4.3 事务优先挖掘算法

1 计算每一个项目的事务相关集：$[x_j]_R=\{t_{j1},\cdots,t_{jr}\}$，j=1,…,m;

2 计算每一个事务项的项目相关集：$[t_i]_R=\{x_{i1},\cdots,x_{ip}\cdots\}$,i=1,…,n;

3 设用户给定的最小支持度为 minsupp，删除$|[x_j]_R|<$minsupp 的项目，k=2;

4 类似于 Apriori 方法把事务的相关项目集$[t_i]_R$组合成长度为2的事务项集：$L_k=\{T_{k1},T_{k2},\cdots,T_{kq}\}$，$|L_k|=q$;

5 对 L_k 中的每个事务项计算：$f(g(T_{ki}))$(i=1,2,…,q)，根据 $|f(g(T_{ki}))|$值($g(T_{ki})$项目的支持度)，把 L_k 由小到大分成3类：

L_{k1}中事务项满足$|f(g(T_{ki}))|<$minsupp (i=1,…,r1)，有 r1个；

L_{k2}中事务项满足$|f(g(T_{kj})|=$minsupp (j=1,…,r2)，有 r2个；

L_{k3}中事务项满足$|f(g(T_{kl}))|>$minsupp(l=1,…,r3)，有 r3个；

合起来 r1+r2+r3=q。

对于 L_{k1}事务项，如果$|g(T_{ki})|\leqslant 2$，那么 $L_{k1}:=L_{k1}-T_{kj}$;

对于 L_{k2}事务项，如果$|g(T_{kj})|<2$，那么 $L_{k2}:=L_{k2}-T_{kj}$;

6 优化：从 L_{k2}、L_{k3}中分别减去含有 $T_{ki}(C_k\in L_{k1})$的事务项集；

```
for i =1 to r2
    for j =2 to r2
        if g(Lk2·Tkj)⊆g(Lk2·Tki)then Lk2:= Lk2−Tkj
    endfor
    for l=1 to r3
        if g(Lk3·Tkl)⊆g(Lk2·Tki)then Lk3:=Lk3−Tkl
    endfor
endfor
```

7 分类处理事务集 L_k;

7.1 对 L_{k2}事务项，计算 $g(T_{ki})$，把映射结果添加到 L_k 中；

7.2 对 L_{k3}中事务项，如果 r2>=1那么

(1)集合 $f(g(T_{kj}))$分解成长度为 minsupp 的事务项(满足最小支持度的)相关集 t_i;

(2)每个事务项集计算 $g(t_i)$(T→X 映射)，得项目相关集(频繁项)；

(3)同 L_k 中的频繁项相比较，如果结果没有新的频繁项，那么转7.3；否则把新频繁项添加到 L_k 中；

7.3 对 L_{k1}中事务项，按类 Apriori 方法把 L_{k1}事务项组合成长度为 k+1的新事务项；

8 如果新事务项集的 g 映射结果非空，那么 k=k+1，转5，否则转9；

9 输出 C_k 频繁项；

10 结束算法。

很明显，事务优先挖掘方法是项目优先挖掘方法的翻版。其主要依据是定理4、5及其两个推论。其有效性从项目优先挖掘方法和两空间的对偶关系也可得到证明。这里就不再继续举例说明了。

5 讨论

项目组合优先搜索的挖掘方法在 X 空间按 Apriori 算法修剪分支；在 T 空间，对支持度在一定范围[minsupp,minsupp+l](这里 l=1)的频繁项分解成支持度等于 minsupp 的频繁项，再直接计算出最大频繁项集，超出范围的仍回到 X 空间继续挖掘最大频繁项。这样在 X、T 空间分别搜索、相互利用有用的信息，极大地提高了算法的搜索效率。前面的定理保证了挖掘方法的正确性。

Apriori 算法只是利用了频繁项的计数特征，忽略了结构特征，在遇到长频繁项时，只能一步一步地计数，不能很快地发现。双空间搜索方法利用了计数特征和结构特征两种信息，在利用事务空间信息中，我们分类处理不同事务项长度的频繁项集。当大于指定步长时再次回到 T 空间，利用 Apriori 方法寻找频繁项。在 T 空间能够轻而易举地发现各种大频繁

项、长频繁项，克服了 Apriori 算法挖掘长频繁项的不足，又获得了很高的效率。

对 T、X 空间的搜索范围分配，由[minsupp，minsupp+l]来决定。当 l 增加时，T 空间的搜索范围加大，反之减少。换一个角度来看，l 相当于步长。过大会增加 T 空间的搜索成本，降低总的搜索效率，过小可能没有发挥应有的效率。

事务项优先搜索方法首先在 T 空间搜索，项目集合的支持度逐渐由小到大，直到满足用户给定的最小支持度。每次在 T 空间搜索后，都检查一下所产生的频繁项，如果所得到频繁项支持度大于最小支持度时分解该项目集合以便确定出最大频繁项。其效率与 X 空间优先搜索方法差不多。

同 Pincer-Search 方法相比较，双空间搜索和 Pincer-Search 方法都是双向搜索，但 Pincer-Search 方法只是在 X 空间中运用上下两个方向进行搜索，利用了 Apriori 方法中的非频繁项信息，并没有利用事务空间的结构信息；我们的双空间搜索方法除了利用 X 空间的信息外，还应用了事务空间中频繁项的结构信息，所以搜索速度要比 Pincer-Search 方法快。

Mohammed J. Zaki 给出了一种 Hybrid 搜索方法[13]，也用到了事务项集合和项目集合两方面的信息进行搜索并且取得了较好的效果。我们这里提出的两空间搜索方法同该方法比较主要有以下不同：

(1)基本出发点不同。文[13]从伪等价关系出发，讨论了图与伪等价类的关系，进而分析最大频繁项；本文是从相关集合出发，定义基于交集的支持度和信任度，分析事务数据库对应的两个搜索空间及其映射。

(2)分析方法不同。文[13]借助最大子图讨论最大频繁项；本文直接根据映射性质分析最大频繁项在不同空间中的表现。

(3)搜索方法有所不同。文[13]采取由上到下，再由下到上的混合搜索方法，基本上是在一个空间搜索，事务标识交集只是用来计算支持度；本文采用两空间搜索方法，可以分别使用不同的优先搜索策略。通过引进“步长”的概念，从计算方法上讲，更像一种迭代法。

(4)本文探讨了数据挖掘中事务数据库对应的空间对偶关系，对借鉴现有的某些方法如线性规划中的对偶问题，为进一步深入分析算法的复杂性开辟了一个新思路。

结束语 根据相关集合的支持度，可以发现事务数据库是建立在 T、X 空间之上的一种关系，最大频繁项是同时满足两个空间中特定约束条件的项目集合。T、X 空间存在一种对偶关系，一个项目集如果在 X 空间中增大其长度，那么在 T 空间中的支持度将减小；反之亦然。f、g 相关集合映射反映了这种对应关系，为 T、X 两个空间的相互转换提供了一个有力的工具。在新的频繁项挖掘方法中，从一个空间到另一个空间双向搜索比只在单个空间中搜索的效率要提高很多，充分地利用两个空间的结构信息和数值特征比只用数值特征要好。笔者认为基于相关集合的双空间搜索方法应该是一类方法。事实上，在本文基本完成后，才读到文[13]，发现有些思路竟有相似之处，不胜惊讶，但细细比较还是有很多不同。本文的 T、X 集合映射及其相关定理实质上是关于频繁项结构的一种分析与描述。当给定一个事务数据库时，准确地分析与描述出频繁项的分布结构是一个值得深入研究的问题。另外，本文主要从理论上证明了方法的正确性和有效性，通过一个实例说明该方法比著名的 Apriori 方法要好很多。具体的计算复杂度分析还有待进一步深入研究。

参 考 文 献

1 Agrawal R, Imielinski T, Swami A. Mining Association Rules between Sets of Items in Large Databases. In: Proc. of ACM SIGMOD Conf. on Management of Data, 1993. 207～216

2 Agrawal R, Srikant R. Fast Algorithms for Mining Association Rules. In: Proc. of 20th Int'l Conf. on Very Large Databases, 1994. 478

3 Park J S, Chen M S, Yu P S. An Effective Hash Based Algorithm for Mining Association Rules. IEEE Transaction on Knowledge and Data Engineering, 1997, 9(4): 813

4 Savasere A, Omiecinski E, Navathe S. An Effective Algorithm for Mining Association Rules in Large Databases. In: Proc. of the 21th Intl. Conf. on Very Large Databases, Zurich, Switzerland, 1995. 9

5 Cheung D W, et al. Maintenance of discovered association rules in large databases: an incremental updating technique. In: Proc. of the 12th Intl. Conf. on Data Engineering, New Orleans, Louisana, 1996. 106～114

6 Lin D-I, Kedem Z M. Pincer-Search: A New Algorithm for Discovering the Maximum Frequent Set, In: Proc. of the Sixth European Conf. on Extending Database Technology. 1998

7 Bayardo Jr R J. Efficiently Mining long patterns from databases. In: Proc. of the 1998 ACM-SIGMOD Int. Conf. on Management of Data (SIGMOD'98), 85～93

8 Zou Qinghua, Chiu Henry, Chu W W. Using patterns decomposition methods for finding all frequent patterns in large datasets, [UCLA CS-TR 200038]

9 Pawlak Z. Rough Sets Theoretical Aspects of Reasoning about Data. Kluwer Academic Publishers, 1991

10 Han J, Pei J. Mining Frequent Patterns by Pattern-Growth: Methodology and Implications. ACM SIGKDD Explorations (Special Issue on Scalable Data Mining Algorithms), 2000, 2(2)

11 Han J, Pei J, Yin Y. Mining Frequent Patterns without Candidate Generation. In: Proc. 2000 ACM-SIGMOD Int. Conf. on Management of Data(SIGMOD'00), Dallas, TX, May 2000

12 Wang Xiaofeng, et al. Mutuality Sets. 沈阳化工学院学报, 1999, 3: 67～76

13 Zaki M J. Scalable Algorithms for Association Mining. IEEE Transactions on Knowledge And Data Engineering, 2000, 12(3): 372～390

14 王晓峰，王天然. 相关测度与增量式信任度和支持度的计算. 软件学报，已录用待发

15 王晓峰，王天然. 一种自顶向下挖掘频繁项的有效方法. 计算机研究与发展，待发表

16 王晓峰，尹丹娜，郑诗诠. 相关集合及其在知识库化简中的应用. 清华大学学报，1998(S2): 6～9

17 王晓峰，唐忠，等. 相关集合在数据库知识发现中的应用. 南京大学学报，计算机专辑，2000, 36: 11, 52～57