# 发现广义序贯模式的增量式更新技术*

欧阳为民[1,2] 蔡庆生[2]

[1](安徽大学计算中心 合肥 230039)

[2](中国科学技术大学计算机科学系 合肥 230027)

**摘要** 提出一种称为FAST的增量式更新技术,以处理因最低支持的改变而引起的广义序贯模式的维护问题.其主要思想是再次利用在对旧的最低支持度进行处理时所获得的结果.

**关键词** 在数据库中发现知识,概念层次,广义序贯模式,增量式更新.

**中图法分类号** TP311

在数据库中,发现知识KDD(knowledge discovery in databases)亦称为数据发掘(Data Mining),是当今国际人工智能和数据库研究的十分活跃的新兴领域.[1]

序贯模式(Sequential Pattern)的发现是R. Agrawal[2]首先提出来的.设有一个交易数据库D,每个顾客可在不同时间购买不同物品,每次购买活动称为交易(Transaction).这里,顾客、交易时间和所购物品分别以Customer-ID, Transaction-Time和Itemset标识.如果我们以Customer-ID为第一关键字,以Transaction-Time为第二关键字对数据库D排序,那么,对每位顾客而言,他进行的所有交易是按交易时间的升序排列的,从而构成了一个序列.我们称这种序列为顾客序列CS(customer-sequence).一般地,令某顾客的各次交易时间为$T_1,T_2,T_3,\ldots,T_n$,该顾客在交易时间$T_i$购买的物品集记为$itemset(T_i)$.于是,该顾客的CS序列为$itemset(T_1),itemset(T_2),itemset(T_3),\ldots,itemset(T_n)$.相应地,我们也可以认为上述交易数据库$D$已转换为顾客序列数据库.

如果某序列$s$包含在某顾客的CS序列中,那么我们称该顾客支持(Support)该序列$s$.某序列的支持度为支持该序列的顾客数与顾客序列数据库中顾客总数之比.由于数据库中的顾客总数是一定的,所以,为方便计,我们一般只考虑支持某序列的顾客数,以此来代表序列的支持度.序贯模式(Sequential Pattern)就是在上述顾客序列数据库中满足用户指定最低支持的最长的序列.[2]每个这样的最长序列均代表一个序贯模式.

显然,上述序贯模式是建立在原始概念级上的,即是单层次的.然而,在大多数情况下,物品之间是存在概念层次关系的.为此,我们结合物品之间的概念层次关系,提出了广义(多层次)序贯模式及其相应算法.[3]其基本思想是,对每个物品生成一个概念层次编码,然后根据概念层次关系,利用R. Agrawal的算法,自顶向下逐层递进地在不同概念层发现相应的序贯模式.

但是,不论是单层次的,还是多层次的序贯模式发现算法,它们都有两个共同的前提:(1)交易数据库中的元组数不变;(2)最低支持度预先指定,也不变.如果这两个前提不成立,我们应该怎么办?这样一种数据更新问题很少有人涉及.[3]D. W. Cheung等人提出了在交易数据库增加新元组的情况下,关联规则的增量式更新技术.[3]本文提出另一种增量式更新技术来处理最低支持度发生变化的情形,其主要思想是,再次利用在对旧的最低支持度进行处理时所得到结果.本文第1节介绍广义序贯模式的有关概念;第2节介绍广义序贯模式的更新;第3节提出并描述广义序贯模式的增量式更新技术;第4节小结,并指出今后的有关工作.

## 1 广义序贯模式的有关概念

本节介绍广义序贯模式的有关概念.假设我们有交易数据库$D$:(*Customer_ID*, *Customer-Time*, *Itemset*),其中*Customer_ID*, *Customer-Time*和*Itemset*分别为顾客标识、交易时间和物品集.$Itemset=(A_p,\ldots,A_q)$,$A_i\in T(i=p,\ldots,q)$,其中$T$为所有物品的集合.

* 本文研究得到国家自然科学基金和安徽省教委科研基金资助.作者**欧阳为民**,1964年生,在职博士生,副教授,主要研究领域为KDD,机器学习,人工智能及应用.**蔡庆生**,1938年生,教授,博导,主要研究领域为知识发现,机器学习,人工智能.

本文通讯联系人:欧阳为民,合肥 230039,安徽大学计算中心

本文1997-07-21收到原稿,1997-09-15收到修改稿

**定义 1.1.** 某顾客的顾客序列(Customer Sequence)为序列 $itemset(T_1),itemset(T_2),itemset(T_3),\ldots,itemset(T_n)$,其中 $T_i(i=1,2,\ldots,n)$为该顾客的交易时间,且 $T_1<T_2<T_3<\ldots<T_n$,$itemset(T_i)$为该顾客在时间 $T_i$ 所购物品的集合.

**定义 1.2.** 序列的长度为该序列中所含有的物品(Item)的个数.

**定义 1.3.** 长度为 $K$ 的序列称为 $K$-序列.

**定义 1.4.** 某顾客支持序列 $s$,如果 $s$ 包含在该顾客的顾客序列中.

**定义 1.5.** 某序列在某概念层的支持为交易数据库中在该层包含该序列的顾客数.

**定义 1.6.** 在某概念层,满足在该层最低支持的序列为在某概念层,常见序列(Frequent Sequence).

**定义 1.7.** 设有序列 $A$ 和 $B$,若序列 $A$ 中的所有物品均依次出现在序列 $B$ 中,则称 $A$ 为 $B$ 的子序列.

**引理 1.** 若某 $K$-序列为常见序列,则其任一长度为$(K-1)$的子序列必是常见序列.

**定义 1.8.** 在某概念层,最长的常见序列称为在该层的广义序贯模式(Generalized Sequential Pattern).

本文采用下列记号:$Fre[K,L]$为第 $L$ 层的常见 $k$-序列的集合,$C[K,L]$为第 $L$ 层的候选 $k$-序列的集合(即潜在的常见 $k$-序列的集合).这两个集合的成员均有两个域:(1) *sequence*(即序列名);(2) *support*(即相应序列的支持).

R. Agawral 的算法需要对交易数据库作多次扫描.对每一概念层 $L$,算法的第 1 次扫描只是统计单一物品(Item)的出现次数,以确定第 $L$ 层的常见 1-序列.后续扫描,比如说第 $K$ 次扫描,分两个阶段组成.首先,采用 Apriori_gen 候选生成算法,利用第 $K$-1 次扫描所得之常见$(K-1)$-序列 $Fre[K-1,L]$来生成候选 $K$-序列集 $C[K,L]$,接着扫描交易数据库,以计算 $C[K,L]$中每个候选序列的支持.然后,挑选其支持不低于该层最低支持的候选构成第 $L$ 层的常见 $K$-序列集 $Fre[K,L]$.重复上述过程,直到第 $L$ 层的常见$(K-1)$-序列集 $Fre[K-1,L]$中仅有一个序列时为止(注意,文献[2]的终止条件为 $Fre[K-1,L]$中不含有任何序列).

## 2 广义序贯模式的更新

本文要解决的问题是:当最低支持发生变化,而交易数据库的元组数不变时,如何更新序贯模式.事实上,由于最低支持是用户自行指定的,事先并不知道怎样才是合适的,因而往往采取试探性的办法.这样,最低支持设定的不合适自然在所难免.为了发现用户真正感兴趣的序贯模式,必然要不断地调整最低支持.于是,我们有必要研制一种高效序贯模式更新技术,以处理因最低支持的改变而引起的问题.事实上,这个问题可以视为在新的最低支持下序贯模式的发现.显然,最自然的方法是以新的最低支持对交易数据库直接地再次应用 R. Agrawal 的算法.这种方法有一个明显的缺点,即不能利用在旧的最低支持下所获得的结果,因而不是一个好的选择.为此,我们提出了一种称为 FAST 的增量式广义序贯模式更新算法.

为表达方便,我们记第 $L$ 层的旧的最低支持为 $S[L]$,$C[K,L]$是第 $L$ 层的候选 $L$-序列集$(K=1,2,3,\ldots,m$,其中 $m$ 为最长的序贯模式的长度),$Fre[K,L]$是第 $L$ 层的常见 $K$-序列集$(K=1,2,3,\ldots,m$,其中 $m$ 为最长的序贯模式的长度).与此同时,我们记第 $L$ 层的新的最低支持为 $NewS[L]$,$NC[K,L]$是第 $L$ 层的候选 $K$-序列集$(K=1,2,3,\ldots,n$,其中 $n$ 为最长的序贯模式的长度),$NFre[K,L]$是第 $L$ 层的常见 $K$-序列集$(K=1,2,3,\ldots,n$,其中 $n$ 为最长的序贯模式的长度).

对概念层 $L$,最低支持的改变可分为如下两种情形.

(1) $NewS[L]>S[L]$,此时在 $Fre[K,L]$中的某些 $K$-序列将不再是常见的;

(2) $NewS[L]<S[L]$,此时不在 $Fre[K,L]$中的某些 $K$-序列将可能成为常见的.

在第 1 种情形下,因为原来不是常见 $K$-序列,不可能变为常见的,而原来是常见 $K$-序列,则有可能变为非常见的,即在新的最低支持 $NewS[L]$下,所有可能的第 $L$ 层的常见 $K$-序列一定在 $Fre[K,L]$中,且所有可能的第 $L$ 层的常见 $K$-序列的支持均已记录在各自的 *support* 域中,所以处理起来十分方便.对概念层 $L$,相应的更新算法如下:

(1) for $(K=1;K\leqslant m;m++)$ {

(2) $\quad NFre[K,L]=\{x\in Fre[K,L]\mid x.support\geqslant NewS[L]\}$;

(3) }

(4) Answer $Set=\bigcup_i NFre[K,L]$;

第 2 种情形较为复杂,我们在第 3 节重点加以讨论.

## 3 增量式更新算法 FAST

FAST 算法是在 Apriori 算法的基础上变化而来的,所不同的是候选生成算法.因为第 $L$ 层的新的最低支持

$NewS[L]<S[L]$，所以，原来在最低支持 $S[L]$ 下各常见 $K$-序列集 $Fre[K,L]$ 在新的最低支持 $NewS[L]$ 下仍是常见的. 于是，在每次扫描顾客序列数据库以计算各候选 $K$-序列的支持时，就不用考虑 $Fre[K,L]$ 中的 $K$-序列. 这样，我们便可设法在生成候选 $K$-序列集 $C[K,L]$ 时不生成 $Fre[K,L]$ 中的 $K$-序列. 为此，我们需要略为修改一下 Apriori 算法，使之保留每次扫描所对应的候选 $K$-序列集和常见 $K$-序列集 $C[K,L]$ 和 $Fre[K,L]$（$K$ 为扫描次数）. 下面，我们描述 FAST 算法推导第 $L$ 概念层广义序贯模式的基本思想.

对概念层次 $L$，首先推导该层的常见 1-序列集. 由于每一单个项目的支持均已记录在 $C[L,1]$ 中，不必重新计算，我们只要做：(1) 对顾客序列数据库作一次扫描，选取支持不低于 $NewS[L]$ 的各单个项目，形成该层的常见 1-序列集 $NFre[L,1]$；(2) 从 $NFre[L,1]$ 中筛选出原来不在 $Fre[L,1]$ 中的项目，记为 $NewFre[L,1]$，即有 $NewFre[L,1]=NFre[L,1]-Fre[L,1]$.

接着分两阶段推导第 $L$ 概念层的常见 2-序列集：(1) 生成候选 2-序列集 $NC[L,2]$；(2) 计算各候选的支持，选取其支持不低于 $NewS[L]$ 的候选，形成常见 $K$-序列集 $Fre[L,2]$. 分别叙述如下：

提高算法效率的一个重要关键是尽可能少地生成候选，或者减少生成候选所需的时间. 很明显，新的候选 2-序列集 $NC[L,2]$ 一定包含原候选 $K$-序列集 $C[L,2]$，而 $C[L,2]$ 又是根据常见 1-序列集 $Fre[L,1]$ 生成的，其中包含常见 2-序列集 $Fre[L,2]$. 那么，我们就设法在生成候选 2-序列集时不生成 $C[L,2]$ 中的候选 2-序列. 办法是，对 $NewFre[L,1]$ 用 apriori_gen 算法生成候选 2-序列集，记结果为 $C'[L,2]$. 然后，再仿照 apriori_gen 算法，从 $NewFre[L,1]$ 和 $Fre[L,1]$ 中选取两个 1-序列构成候选 2-序列，由此生成的候选集记为 $C''[L,2]$. 于是，$NC[L,2]=C'[L,2]\cup C''[L,2]\cup C[L,2]$. 然后分别计算 $C'[L,2]$ 和 $C''[L,2]$ 中各候选的支持，从 $NC[L,2]$ 中选取支持不低于 $NewS[L]$ 的候选，形成新的常见 $K$-序列集 $NFre[L,2]$. 记新增加的常见 2-序列集为 $NewFre[L,2]$，则有 $NewFre[L,2]=NFre[L,2]-Fre[L,2]$.

仿照上述方法，我们可按如下方法推导第 $L$ 层的常见 $K$-序列集 $NFre[K]$（$K\geqslant 3$）. 首先构造候选 $K$-序列集 $NC[K]$. 办法是，对 $NewFre[K-1]$ 用 apriori_gen 算法生成候选 $K$-序列集，记结果为 $C'[L,K]$. 然后，再仿照 apriori_gen 算法，从 $NewFre[K-1]$ 和 $Fre[K-1]$ 中选取两个 $(K-1)$-序列构成候选 $K$-序列，由此生成的候选集记为 $C''[L,K]$. 于是，$NC[L,K]=C'[L,K]\cup C''[L,K]\cup C[L,K]$. 然后分别计算 $C'[L,K]$ 和 $C''[L,K]$ 中各候选的支持，从 $NC[L,K]$ 中选取支持不低于 $NewS[L]$ 的候选，形成新的常见 $K$-序列集 $NFre[K]$. 记新增加的常见 $K$-序列集为 $NewFre[L,K]$，则有 $NewFre[L,K]=NFre[L,K]-Fre[L,K]$.

上述过程一直反复进行到 $NFre[L,K]$ 中只有一个 $K$-序列时为止. 我们记 $NFre[L,K]$ 中 $K$-序列的个数为 $|NFre[K]|$.

现在，我们给出生成 $C''[L,K]$ 的变形的 apriori_gen[4] 候选生成算法 FAST_gen. 该算法分为如下两步：

(1) 拼接

insert into $C''[L,K]$

select $p.item_1, p.item_2, \ldots, p.item_{k-1}, q.item_{k-1}$

from $NewFre[L,K]p, Fre[L,K]q$

where $p.item_1=q.item_1, p.item_2=q.item_2, \ldots, p.item_{k-2}=q.item_{k-2}$

(2) 修剪

对 $C''[L,K]$ 中的任一候选 $c$，如果 $c$ 中存在一个不属于 $NFre[K-1]$ 的长度为 $K-1$ 的子序列，那么就从 $C''[L,K]$ 中删除该候选 $c$.

根据上述讨论，我们描述广义序贯模式的增量式更新算法 FAST 如下.

**Algorithm**: FAST (Incremental Updating Algorithm)

Input: (1) A Customer-Sequence databases CSDB, in the format of $(TID, Time, itemset)$, in which each item in itemset contains encoded concept hierachy information, and;

(2) The old minimum support is $S[L]$, whereas the new minimum support is $NewS[L]$, and;

(3) Candicate $K$-sequence set $C[L,K]$ and frequent $K$-sequence set $Fre[L,K]$ for all $K$.

Output: A set of updated generalized sequential patterns.

Begin

(1) for ($L=1; L<$max_level$; L++$) {

(2)　　$NFre[L,1]=\{1\text{-}sequence\ i \mid i.support\geqslant NewS[L]\}$;

(3)　　$NewFre[L,1]=NFre[L,1]-Fre[L,1]$;

(4)　　for ($K=2; |NFre[K-1]|>1; K++$) {

```
(5)        C'[K,L]=apriori-gen(NewFre[L,K-1]);
(6)        C''[K,L]=FAST-gen(NewFre[L,K-1],Fre[L,K-1]);
(7)        for each transaction t∈CSDB do {
(8)            CC'[t]={c|c∈C'[L,K], c is a sub-sequence of t.itemset};
(9)            CC''[t]={c|c(C''[L,K], c is a sub-sequence of t.itemset};
(10)           for each candicate c∈CC'[t] do c.support++;
(11)               for each candicate c∈CC''[t] do c.support++;
(12)           }
(13)    NC[L,K]=C[L,K]∪C'[L,K]∪C''[L,K];
(14)    NFre[L,K]={c∈NC[L,K]|c.support≥NewS[L]}
(15)    C[L,K]=NC[L,K];Fre[L,K]=NFre[L,K];   /* used for next time */
(16)    }
(17) }
(18) Answer Set=∪k NFre[L,K];
(19) End.
```

## 4 小　结

正如在专家系统的研制过程中确定产生式规则的信任度一样，为了发现用户真正感兴趣的序贯模式，需要不断地调整最低支持，直到用户满意为止. 为此，我们提出了一种称为 FAST 的增量式更新算法来处理因最低支持发生变化而引起的序贯模式的更新问题. 该算法的主要思想是，再次利用在旧的最低支持下已经获得的结果(即每一个候选 $K$-序列集 $C[L,K]$与常见 $K$-序列集 $Fre[L,K]$)，从而尽可能地减少了计算量. 当然，要做到这一点，需要保存每一个候选集 $K$-序列 $C[L,K]$与常见 $K$-序列集 $Fre[L,K]$. 显然，这是一个以空间换时间的策略. 本算法已经在 PC 486/100 (16M 内存)上用 Visual FoxPro1.0 实现，性能良好.

**致谢**　国际 KDD 研究知名学者、加拿大 Simon Fraster 大学 Han Jiawei 教授为我们提供了大量的相关资料，特此深表感谢.

### 参考文献


1　欧阳为民，蔡庆生. 在数据库中自动发现广义序贯模式. 软件学报，1997，8(11)：864～870
(Ou-Yang Wei-min, Cai Qing-sheng. Automatic discovery of generalized sequential pattern in databases. Journal of Software, 1997,8(11):864～870)

2　Agrawal R, Srikant R. Mining sequential patterns. In: Proceedings of the International Conference'95 on Data Engineering. Taibei. March 1995

3　Cheung D W, Han J, Ng V *et al*. Maintenance of discovered association rules in large databases: an incremental updating technique. In: Proceedings of the International Conference'96 on Data Engineering. New Orleans, Louisiana. Feb. 1996

4　Agrawal R, Srikant R. Fast algorithm for mining association rules. In: Proceedings of the International Conference'94 on Very Large Data Bases. Santiago, Chile. Sept. 1994. 487～499

5　Han J, Fu Y. Discovery of multiple-level association rules from large databases. In: Proceedings of the 21st VLDB Conference. Zürich, Swizerland. 1995


## An Incremental Updating Technique for Discovering Generalized Sequential Patterns


OU-YANG Wei-min[1,2]　CAI Qing-sheng[2]

[1](*The Computing Center　Anhui University　Hefei*　230039)

[2](*Department of Computer Science　University of Science and Technology of China　Hefei*　230027)



**Abstract**　An incremental updating technique called FAST is proposed in order to deal with the maintenance of discovered generalized sequential patterns resulted from the change of the minimum support. The main idea is to re-utilize the results acquired in process with the old minimum support.



**Key words**　KDD (knowledge discovery in databases), concept hierarchy, generalized sequential pattern, incremental updating.