大型事务数据库中的一种快速的规则挖掘算法*)

A Fast Algorithm on Association Rules in Large Transaction Databases

朱玉全　孙志挥

(东南大学计算机科学与工程系　南京 210096)

Abstract Discovery of association rules is an important data-mining task. Several algorithms have been proposed to solve this problem. But most of them are based on the Apriori algorithm that requires repeated passes over the large transaction databases. In this paper, the authors propose a new database schema and present a relevant algorithm. Comparing with Apriori and Apriori-like algorithms, the authors also offer some experiments to show that the new algorithm is more efficient.

Keywords Data mining, Association rules, Transaction databases

1 引言

数据挖掘(Data Mining),也称为数据库中知识发现KDD,是指发掘隐藏在堆积如山的数据中的真知灼见,这基本上正在变成一种商业上非做不可的事情。关联规则[1](Association Rules)是数据挖掘的重要研究内容,目前的绝大部分关联规则挖掘算法一般都分为两个阶段:①频繁项目集的发现;②规则的产生。算法的计算工作量主要集中在第一阶段上,因此,如何快速确定频繁项目集是算法效率的关键,在这方面已有许多工作与成果[2~6]。但总的来讲,许多研究[2~6]都是在 Apriori 算法或其派生算法的基础上进行的。这些算法或多或少存在如下两个问题:①算法必须耗费大量的时间处理规模巨大的候选项目集;②算法必须多次重复机械地扫描大型事务数据库,对候选项目集进行模式匹配。这两个问题是数据挖掘的难点也是热点,并已成为约束系统性能的瓶颈。

针对上述问题,本文首先提出了事务数据库的一种新的表示方法,其次设计出一种相应的关联规则挖掘算法 Fast-Apriori 算法,最后与 Apriori 类算法进行了性能比较。

由于篇幅所限,上下文中所用述语、符号以及其含义除非特别说明,均与文[1]相同,在此不作详细介绍。

2 模式定义及其构造方法

数据挖掘大体上可以分为三个阶段:①数据预处理;②挖掘算法的实现;③模式评价和知识验证。绝大部分情况下,经预处理后得到的数据库模式为:〈顾客号 TID,项目 1,项目 2,…,项目 m〉。其实我们在构造布尔型关联规则(事实上对于定量关联规则,我们仍将其转换成布尔型关联规则)的挖掘算法时,对于每一个项目只需考虑购买与否两种情况,即 0/1 变量,也就是说每一个项目只需占二进制数的一位即可。因此,经过预处理后得到的数据库模式可以表示为:〈顾客号 TID,字段 1,字段 2,…,字段 p〉。其中字段 i(i=1,…,p)的类型均为无符号整数,并且各字段的每一位分别一一对应于一个项目。

为了方便地将原模式下的数据库 D 转换成新模式下的数据库 D',定义如下转换函数。

定义 1　转换函数 $Y=F(X)$ $X=(x_m,x_{m-1},\cdots,x_1)^T$

$Y=(y_p,y_{p-1},\cdots,y_1)^T$　$F(X)=(f_p,f_{p-1},\cdots,f_1)^T$

$$f_i=f_i(x_{i*k},x_{i*k-1},\cdots,x_{(i-1)*k+1})=\sum_{j=1}^{k}x_{(i-1)*k+j}*2^{j-1},$$

$i=1,2,\cdots,p$

其中 k 为无符号整数所占的字节数,m 为项目数,$p=[m/(k*8)]$为新模式下的字段数。一般情况下,k=4,因此,$p\ll m$。

算法 1　新模式下的数据库 D' 生成算法。

```
输入:事务数据库 D。
输出:新模式下的数据库 D'。
方法:
(1)for all transactions t∈D do begin
(2)   x_i=0 (i=1,2,…,m);
(3)   for i=1 to m
(4)     if(item_i∈t) then
(5)       x_i=1;
(6)   Y=(f_p,f_{p-1},…,f_1)^T;
(7)   Write record Y=(y_p,y_{p-1},…,y_1)to database D';
(8)end
```

算法 2　算法 1 的逆算法。

```
输入:数据库 D'。
输出:数据库 D。
方法:
(1)for all records t'∈D'
(2)       i=1; t=Φ;
(3)       while(t'≠0)do begin
(4)       if((t' and 01F)≠0)then //表明 t' 的第 i 位为 1
(5)         t=t∪{item_i}; //事务 t 包含项目 item_i
(6)     shr(t');i++;
(7)   end
(8)   Write t to database D;
(9)end
```

显然,在数据库 D 与数据库 D' 中记录是一一对应的,并且通过算法 1 和算法 2 可以很方便地实现它们之间的转换,即它们之间是信息等价的,也就是说数据库 D' 保留了原有数据库 D 的一切信息,对原有数据库的一切操作均可以在数据库 D' 上实现,反之亦成立。因此我们可以删除原有数据库 D,从而节约了大量的存储空间。另外如果经预处理以后得到的数据库模式已符合要求,那么可以跳过算法 1。

3 Fast-Apriori 算法描述

3.1 有关性质及定理

定义 2　对于任意 $x\in D'$,$sitem_x$ 定义为 x 所对应的项目

*)本文得到国家自然科学基金资助(项目编号 79970092)。朱玉全　博士生,主要研究领域为数据库系统。孙志挥　教授,博士生导师,主要研究领域为计算机集成系统、数据库系统等。

集。

性质 1 对于任意 $x \in D'$，如果 x 的第 $j_1, j_2, \cdots, j_m$ 位为 1，那么它所对应的项目集为 $sitem_X = \{item_{j1}, item_{j2}, \cdots, item_{jm}\}$。

定义 3 Length(x)定义为 x 中 1 的个数。

定理 1 对于$\forall$ x，$\forall$ y，如果 length(x)=length(y)=k−1，那么项目集 $sitem_X$ 与项目集 $sitem_Y$ 有且只有 k−2 个元素相同当且仅当 length(x and y)=k−2。

证明：因为项目集 $sitem_X$ 与项目集 $sitem_Y$ 有 k−2 个元素相同，所以在 x 与 y 中必有 k−2 位值为 1 的位相同，这 k−2 位的“与”值为 1，另外还有两位各不相同，它们的“与”值必为 0，其它位均为 0，“与”值为 0，因此 length(x and y)=k−2。反之，如果 length(x and y)=k−2，那么在 x 及 y 中对应于这 k−2 位的位值都为 1，其它位均互不相同，由于 length(x)=k−1，length(y)=k−1，因此在其余位中有且只有各一位值为 1，而这两位是不可能同时为 1 的，故在这两个数中有两位互不相同，余下的位值都为 0，所以项目集 $sitem_X$ 与项目集 $sitem_Y$ 有且只有 k−2 个元素相同。

定理 2 对于$\forall$ x，$\forall$ y，如果 x and (not y)=0，那么 $sitem_X \subseteq sitem_Y$。

证明：因为 x and (not y)=0，所以 x 与(not y)的各位均不相同。假设 x 的第 $j_1, j_2, \cdots, j_m$ 位为 1，那么(not y)的第 $j_1, j_2, \cdots, j_m$ 位也为 0，即 y 第 $j_1, j_2, \cdots, j_m$ 位为 1，也就是说，如果 x 的第 i 位为 1 必有 y 的第 i 位也为 1，因此定理 2 成立。

3.2 关联规则的挖掘算法 Fast-Apriori 算法

本算法的基本步骤与 Apriori 算法一样，但具体实现与 Apriori 算法完全不同。算法的第一步是简单统计所有含一个元素项集出现的频率，来决定最大的一维频繁项目集。在第 k 步，分三个阶段，首先调用函数 Fast-Apriori-Gen(L_{k-1})来生成候选项目集 C_k，其次调用函数 Fast-Count-Support(C_k)计算候选项目集 C_k 中元素的支持度，最后统计候选项目集 C_k 中的 k-维频繁项目集。

算法 2 挖掘频繁项目集算法 Fast-Apriori。

```
输入：数据库 D'，minsup。
输出：频繁项目集 L。
方法：
(1)C1={candidate 1-item sets};
(2)L1={c∈C1|c.support≥minsup};
(3)For(k=2;Lk-1≠φ; k++)do begin
(4)   Ck=Fast-Apriori-Gen(Lk-1); // 调用函数 Fast-Apriori-Gen(Lk-1)
(5)   call Fast-Count-Support(Ck); // 调用函数 Fast-Count-Support(Ck)
(6)   Lk={c∈Ck|c.support≥minsup};
(7)end
(8)L=∪kLk;
Function Fast-Apriori-Gen (Lk-1)
(9)For all p1,p2∈Lk-1 do begin
(10)  temp1= p1 and p2; temp2=temp1;
        //temp1,temp2为临时变量
(11)  temp3=1; //temp3为临时变量，统计 temp1中1的个数
(12)  temp4=1; //temp4为临时变量，统计右移的次数
(13)  while (temp2≠0 )do begin
(14)    if((temp2 and 01F)≠0)then//表示 temp2的最低位为1
(15)    temp[temp3++]=temp4;
          //表示 temp1的第 temp4位值为1
(16)    shr(temp2); temp4++; //temp2右移一位，右移次数加1
(17)  end
(18)  tag=0;
(19)  if(temp3=k-2)then begin //见定理1
(20)    for(i=1;i≤k;i++)do begin
(21)      if (temp1 and not(2^temp[i]))∉ Lk-1 then begin //见说明②例
(22)        tag=1; break;
(23)      end
(24)    end
(25)  if (tag=0)then insert temp1 into Ck; //Ck 的所有 k-1维子集均属于 Lk-1
(26)  end
(29)end
Function Fast-Count-Support (Ck)//见定理2
(30)For all t'∈D' do begin
(31)  t'=not(t');
(32)  for all c∈Ck do begin
(33)    if((c and t')=0)then
(34)c.count++;
(35)  end
(36)end
```

在此算法中需要说明以下几点：

①(20)～(23)实现 Apriori 算法中“修剪”功能。

②例如 temp1=10010101B(m=8)，$sitem_{temp1}=\{item_1, item_3, item_5, item_8\}$。那么经(13)～(17)步后 temp[1..4]={1,3,5,8}，表示 temp1的第1、3、5及8位均为1。temp1 and not ($2^{temp[i]}$)(i = 1 ～ 4)值分别为 10010100B、10010001B、10000101B 及 00010101B，它们所表示的项目集分别为{$item_3, item_5, item_8$}、{$item_1, item_5, item_8$}、{$item_1, item_3, item_8$}及{$item_1, item_3, item_5$}，即 $sitem_{temp1}$ 的所有3或(k−1)维子集。

③此算法中的 Fast-Apriori-Gen(L_{k-1})函数与 Apriori 算法中的 apriori-gen(L_{k-1})功能相同，但实现方法完全不同，在 Apriori 算法中一些较难实现的问题在这里得到了较好的解决，例如“修剪”功能、计算支持度等。

④此算法中计算候选项目集 C_k 中元素的支持度函数 Fast-Count-Support(C_k)远比 Apriori 算法中相应函数简单。

4 算法分析与比较

从理论上讲，如果某大型事务数据库中有 m 个项目，每个项目平均占有 1 个字节，那么每条记录的长度为 m×1×8 位，而在新的数据库模式中每条记录的长度仅占 m 位，压缩比为 1×8，从而大大地减小了访问 I/O 的次数。另外，在新的数据库模式中进行关联规则挖掘也远比 Apriori 算法简单，在这里只要进行一些“或”、“与”、“非”、“移位”等操作即可，进一步节约了 CPU 的运行时间。

为了进一步验证该算法优越性，我们给出了本算法与 Apriori 类算法的比较。实现环境是基于 Windows(professional)平台，内存64M，CPU 为 Pentium III－450MHz。实现环节所采用的语言为：Delphi5.0。所采用的数据集为：南京市某中小型企业事务数据库。该企业是江苏省规模适中的中小型企业，在过去的几年中，积累了大量的原始数据，预处理后该数据库共有812 145条记录和91种属性。为了使数据具有代表性，我们随机从该数据库中抽取了两组不同的记录数(20 000和40 000)分别对 Fast-Apriori 算法及 Apriori[1]算法进行测试，实验结果如图1、2所示，由此可见 Fast-Apriori 算法是快速的，比 Apriori[1]算法快4～10倍。

结束语 关联规则的挖掘是当前数据挖掘领域的主要研究课题，本文提出了大型事务数据库中的一种快速的关联规则挖掘算法。理论与实验结果表明，该算法比 Apriori 类算法具有较好的性能。另外 Apriori 算法的改进方法均适用于此算法，我们将对此作进一步的研究。

参 考 文 献

1 Agrawal R, Srikant R. Fast algorithms for mining association rules. In: Proc. of the 20th Int. Conf. on VLDB, Santiago, Chile, 1994. 487～499

2 Han J, Kamber M. Data Mining: Concepts and Techniques. 北京：高等教育出版社，2001

3 范明，孟小峰，等译. 数据挖掘：概念与技术. 北京：机械工业出版社，2001

(下转第69页)

反馈信息。

任务协调：负责系统各个模块之间的协同、通信。

系统管理：提供对用户信息库和缓存的管理，并负责用户搜索习惯的学习。

搜索代理：采用2.4节所论述的策略，进行网上源信息采集。

页面分析：对搜索代理取得的页面，进行特征提取，并使用 VSM 算法给出匹配度。

4 实验结果及评估

4.1 IR 系统的两大评价标准

1)查全率(recall)：挖掘到的文档资料数与实际相关文档资料数之比。

2)查准率(precision)：结果集合中的相关文档数与结果集文档数之比。

作为一个好的 IR 系统，在保证查准率的同时还要能有较高的查全率[6]。

4.2 兴趣度导引算法与 fish-search 算法及广度优先算法的比较结果

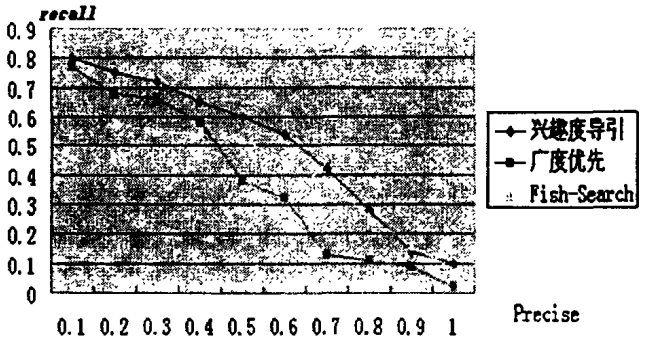

图3 评估结果 r-p 图

我们以 http://WWW.edu.cn/为搜索空间，使用 PSA 系统分别使用基于兴趣度导引的算法(取经验阈值 IT 为 0.34，IB 为0.1，m 为3)、fish-search 算法、广度优先算法，对20个查询进行试验，得到结果如图3。

测试结果表明，广度优先算法具有较高的查全率，但在查准率方面衰减非常快。Fish-Search 算法精度高，但是相应的查全率在低查准率区间表现很差。而兴趣度算法具有极好的稳定性，表现明显优于其它两种算法。

结论 本文引入数据挖掘中的感兴趣度模型，提出了基于客观感兴趣度导引的 Web 页面采集算法，及基于主观感兴趣度的相关反馈度量公式，并且给出了相应的实现系统 PSA。实验证明，以上两种策略的实施大大提高了 WWW 个性化搜索系统的精度和效率，并加强了系统对用户搜索习惯学习的能力。

参 考 文 献

1 邹涛，王继成，等. WWW 上的信息挖掘技术及实现. 计算机研究与发展，1999，36(8)

2 McBryan OA 1994. GENVL AND WWW: Tools for taming the Web Proc. Intl. World Wide Web Conference, ed. By Nierstrasz O. Genneva: CERN

3 Menczer F, Belew R K. Artificial Life Applied to Adaptive Information Agents. Communication Technology Lab, Image Science Group ETH- Zertrum, ETZ F86

4 De Bra, Post. Searching for Arbitratry Information in the WWW: the Fish-Search for Mosaic

5 杨炳儒，等. 感兴趣度的研究综述. 计算机科学，2001，28(10)：43～45

6 Salton G. The Smart Retrieval System-Experiment in Automatic Document Processing. Prentice-Hall, Englewood Cliffs, New Jersey 1971

7 Harper D, van Rijsbercen C J. An evaluation of feedback in document retrieval using co-occurrence data. Journal of Documentation, 1978, 34: 189～216

8 van Rijsbergen C J, Retrieval. London: Butterworths, 1979

(上接第60页)

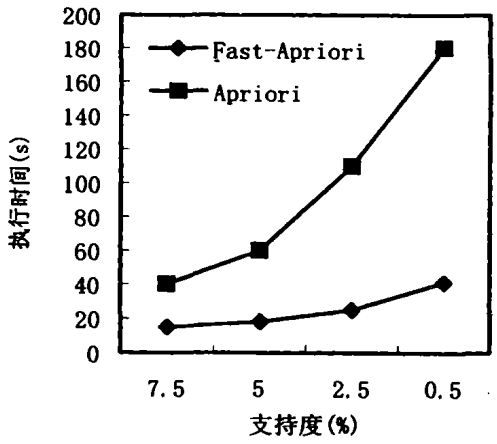

图1 算法执行时间(|T|=20 000)

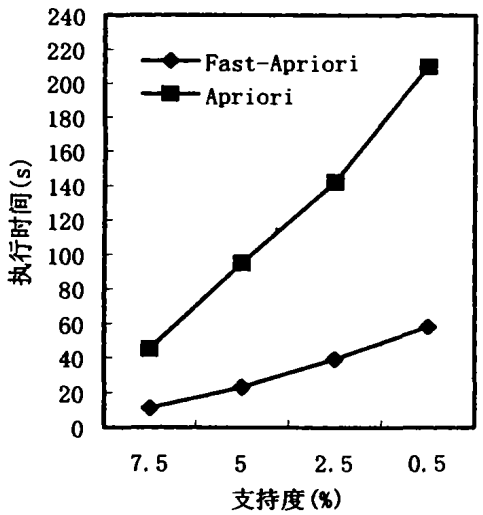

图2 算法执行时间(|T|=40 000)

4 Agrawal R, Srikant R. Fast algorithm for mining association rules. In: Proc. 20th Int. Conf. on VLDB, Santiago, Chile, 1994. 487～499

5 Houtsma M, Swami A. Set-oriented mining for association rules in relational databases. In: Yu P, Chen A, eds. Proc. of the Int. Conf. on Data Engineering, Los Alamitos, CA: IEEE Computer Society Press, 1995. 25～33

6 Savasere A, Omiecinski E, Navathe S. An efficient algorithm for mining association rules. In: Proc 21th Int. Conf. on VLDB, Zurich, Switzerland, 1995. 432～444