一种用于挖掘正、负关联规则的改进 Apriori 算法*)

赵海丰　邢永康　杨华丽　秦　鹏　卢俊杰

(重庆大学计算机学院　重庆 400044)

摘　要　本文提出一种传统的关联规则挖掘主要着眼于正关联规则，即形如 A→B 的规则的挖掘，而对负关联规则的研究非常有限，然而实践表明在关联规则的各个应用领域中，负关联规则同正关联规则有着同样的重要性。Apriori 算法是挖掘关联规则的一个经典算法，但是它只局限于挖掘正关联规则，本文对该算法进行改进提出了 Ex-Apriori 算法，新算法不仅能挖出负关联规则，而且由于兴趣度的引进，能够剔除大量无趣的关联规则。实验表明该种算法有效且可行。

关键词　正关联规则，负关联规则，兴趣度

1　引言

关联规则挖掘是数据挖掘的一类，它用来在一个大的事务集中发现各个项间的隐含关系，1993 年由 R. Agrawal 首先提出[1]，此后大量的学者对其进行了深入的研究，现在关联规则已经广泛地应用于各个领域，如诊断决策、电信、入侵检测等。传统关联规则挖掘得出的是形如 A→B 的蕴涵式，它所表示的意思就是在一个事务中如果出现了 A，那么也就极有可能出现 B。R. Agrawal 于 1994 年提出了一种基于支持度－置信度模式的关联规则挖掘算法[2]即 Apriori 算法。XinDong Wu 在传统关联规则的基础上进行了扩展，提出了负关联规则[3]，即 A→¬ B，¬ A→B，¬ A→¬ B，同传统关联规则相比，负关联规则对事务集中项的状态进行了扩展，它不仅研究各项出现之间的关系，还研究各项出现与不出现的关系。

尽管在应用中负关联规则同正关联规则具有同等的重要性，但是由于研究起步晚且难度较大，负关联规则的挖掘还没有能够出现一种像 Apriori 算法那样得到广泛认可的算法。Brin 等人在文[4]提出了基于 Chi-squared 测试的两个频繁集的相关度的方法，1998 年 Savasere 等人在文[5]提出了一种挖掘了强负关联规则的算法，但是由于限制条件太多，很难实际应用。XinDong Wu 在文[3]中正式提出负关联规则的同时还提出了一种能同时挖掘正、负关联规则的算法，该种算法在生成频繁项目集时会造成丢失。

传统关联规则的挖掘是基于支持度－置信度模型的，根据这一模型挖掘会得出大量的正关联规则，这些关联规则是否都是有意义的呢？根据 Piatetsky-shapiro 提出的兴趣度标准，我们发现大量的规则都是无趣的。

针对以上问题，本文在原有 Apriori 算法的基础上进行了改进，又引入了兴趣度标准，提出了一种能够同时挖掘正负关联规则的算法 Ex-Apriori。论文的其余部分是这样安排的：第 2 部分是问题描述，第 3 部分是算法描述，第 4 部分是算法设计，第 5 部分是算法分析，通过实验对本文所述算法同传统算法进行比较，最后是结论和下一步工作展望。

2　问题描述

设 $I=\{i_1, i_2, \cdots, i_m\}$ 是 m 个不同项目的集合。给定一个事务数据库 D，其中每一个事务 T 是 I 中一组项目的集合，即 $T \subseteq I$。如果对于 I 中的一个子集 X，有 $X \subseteq T$，我们就说一个事务 T 包含 X。负关联规则就是形如 $X \rightarrow \neg Y$，$\neg X \rightarrow Y$，$\neg X \rightarrow \neg Y$ 的蕴含式，其中 $X \subset I$，$Y \subset I$，而且 $X \cap Y = \emptyset$。

负关联规则中的支持度和置信度有着同正关联规则相同的意义和形式：

$\neg X$ 的支持度 $\mathrm{supp}(\neg X) = |\{T | T \in D, \neg X \subseteq T\}| / |D|$

负规则 $\neg X \rightarrow Y$ 的置信度

$\mathrm{confidence}(\neg X \rightarrow Y) = \mathrm{supp}(\neg X \cup Y) / \mathrm{supp}(\neg X)$

在本文的挖掘算法中由于非频繁集的引入，使得候选的关联规则数目大幅增多，为了剔除其中无用的规则，算法中利用了兴趣度来对候选的关联规则进行筛选。兴趣度就是描述关联规则前件和后件间联系或影响紧密程度的一个度量，1991 年由 Piatetsky-Shapiro 首先提出，即如果关联规则 X→Y 满足：

*)本研究得到国家自然科学基金青年基金资助(编号：60403009)。赵海丰　硕士生，主要研究领域为人工智能。邢永康　博士后，副教授，硕士生导师，主要研究领域为人工智能。

$$supp(X \cup Y) \approx supp(X) * supp(Y) \quad (1)$$

或 $supp(X \cup Y)/supp(X)supp(Y) \approx 1$

那么 X 和 Y 近似独立，规则 X→Y 无趣。在本文的算法中将传统兴趣度形式稍加变化得到了

$$interest(X, Y) = supp(X \cup Y) - supp(X) * supp(Y)$$

分析 interest(X, Y)有三种可能的取值情况：

(1)interest(X, Y)>0，表示 X 和 Y 正相关，事件 X 出现得越多，事件 Y 出现得也越多；

(2)interest(X, Y)=0，表示 X 和 Y 相互独立，事件 X 的出现与事件 Y 无关；

(3)interest(X, Y)<0，表示 X 和 Y 负相关，事件 X 出现得越多，事件 Y 出现得越少。

负关联规则的挖掘就是挖掘出满足最小支持度、最小置信度和最小兴趣度的形如 X→¬Y，¬X→Y，¬X→¬Y 的规则，且 X 和 Y 均为频繁项目集。同正关联规则挖掘相比，负关联规则挖掘的难度要大得多，XingDong Wu 在文[3]所举实例中，一个只有 6 个项，5 个事务的小数据库，产成了 49 个非频繁集，只是非频繁集(ABCDEF)一项，就可能产生 110 个负关联规则，而所有 49 个非频繁集则可能产生最多 818 个关联规则。由此可见，负关联规则的挖掘确实是一项很艰难的工作。

3 算法描述

基于上面讨论的问题，本文提出了一种能同时挖掘出正负关联规则的算法即 Ex-Apriori 算法，该算法在生成频繁集阶段模仿了 Apriori 算法，不同之处是在生成一阶频繁集后对项的状态进行了扩展，引入了非项，从而保证了得到包括非项的频繁集。然后在生成关联规则阶段又采用了兴趣度对候选规则进行筛选，从而保证了最终得到结果的有效性。

在挖掘负关联规则的过程中，由于对非频繁集的考虑，使得对于支持数的计算变得非常复杂。在文[7]中，提出了一种新颖的方法，它将事务对项目的支持判断转化成两个二进制数的或操作。定义 1 和定理 1 对文[7]所述方法作了概述。

定义 1 给定 $I=\{i_1, i_2, \cdots, i_m\}$，事务数据库 D，事务 $T \in D$(项目集 X)，且 $T \subseteq I(X \subseteq I)$。

称 $b=b_1 b_2 \cdots b_{m-1} b_m$ 为事务 T(项目集 X)所对应的二进制数，记为 $B_t(B_x)$，其中

$b_j \in \{0,1\}$，且如果 $i_j \in T(i_j \in X)$，则 $b_j=1$，否则 $b_j=0$，$j=1,2,\cdots,m$。

定理 1 给定 $I=\{i_1, i_2, \cdots, i_m\}$，事务数据库 D 中的事务 T，项目集 c。如果 $B_t orB_C = B_t$，则 T 支持 c，反之亦然。

例 1 给定一事物数据库 D，$I=\{1,2,3,4,5,6,7,8\}$，事务 $T=\{1,3,5,7\}$，项目集 $c=\{1,5,7\}$，则 $B_t=10101010$，$B_c=10001010$，因为 $B_t orB_c = B_t$，所以 T 支持 c。

由于定理较为简单，且在文[7]中有详细的证明，在本文中就不加详述。为了能够将这种方法应用到本文的算法中，我们对定义 1 和定理 1 进行了扩展。

定义 2 给定 $I=\{i_1, i_2, \cdots, i_m\}$，事务数据库 D，事务 $T \in D$(项目集 X)，且 $T \subseteq I(X \subseteq I)$。

称 $b=b_1 b_2 \cdots b_{m-1} b_m$ 为事务 T(项目集 X)所对应的二进制数，记为 $B_t(B_x)$，其中

$b_j \in \{0,1\}$且如果 $i_j \in T(i_j \in X)$，则 $b_j=1$，否则 $b_{j=0}$，$j=1,2,\cdots,m$。给定项目集 X，$X \subseteq I$，称 $b_p = b_1 b_2 \cdots b_{m-1} b_m$ 为项目集 X 所对应的正部二进制数，即为记为 B_{xp}，其中 $b_j \in \{0,1\}$，且如果 $i_j \in X$，则 $b_j=1$，否则 $b_j=0$，$j=1,2,\cdots,m$。称 $b_n = b_1 b_2 \cdots b_{m-1} b_m$ 为项目集 X 所对应的非部二进制数，即为记为 B_{xn}，其中 $b_j \in \{0,1\}$，且如果¬ $i_j \in X$，则 $b_j=0$，否则 $b_j=1$，$j=1,2,\cdots,m$。

定理 2 给定 $I=\{i_1, i_2, \cdots, i_m\}$，事务数据库 D 中的事务 T，项目集 c。如果 $B_t orB_{cp} = B_t$ 且 $B_t andB_{cn} = B_t$，则 T 支持 c，反之亦然。

由于本文算法中的项目集引入了非项，因此以在这里将事务对项目支持的判断分为两部分判断，即正部判断和非部判断，两者都支持时方能说明事务对项目支持。由定义可知对正部的判断实际上就是定理 1 的引用，所以不加详述。下面主要证明非部判断的正确性，假设项目集包括 $i_1, i_2, \cdots, i_n$ 个非项，则 B_{cn} 的 1 到 m 位为 0，其他位为 1，因为 $B_t andB_{cn} = B_t$，所以 B_t 的 1 到 m 位也为 0，根据定义 2 可知这些非项对应的项没有在 T 中出现，所以 T 支持 c 的非部，由于 T 既支持 c 的正部又支持 c 的非部，因此说 T 支持 c。

例 2 给定一事物数据库 D，$I=\{1,2,3,4,5,6,7,8\}$，事务 $T=\{1,3,5,7\}$，项目集 $c=\{1,5,7,\neg 2,\neg 8\}$，则 $B_t=10101010$，$B_{cp}=10001010$，$B_{cn}=10111110$，因为 $B_t orB_{cp} = B_t$ 且 $B_t andB_{cn} = B_t$，所以 T 支持 c。

4 算法设计

```
Procedure GenerateAllFrequentItems //生成所有的频繁项集
Input:事务数据库 D;最小支持度 ms,
Output: L 频繁项集合
let L'1={频繁 1-项目集};
let L1=L'1∪{¬X|X∈L'1};
for(k=2 ;Lk-1≠Ø;k++)do
  Begin
    Ck=Apriori-gen(Lk-1};
    call Calculate-count(Ck);
    Lk={c|c∈Ck∧c. count>=ms};
  end;
```

let $L=\bigcup_{i=1}^{k} L_i$;

在第(2)步将传统的1-项频繁集进行了扩展，引入了非项，正是这一步变化保证了在以后的迭代中能够生成所有包括非项的频繁集。Apriori-gen用来由频繁k-1项集得到候选k项集，限于篇幅，不再详细介绍。

Procedure Calculate-count //遍历数据库，计算候选k项集中候选项的支持数

input：候选k项集的集合 C_k

output：无；

For each transaction T∈D do

For each C∈C_k do

if($B_t$or$B_{cp}=B_t$)&&($B_t$and$B_{cn}=B_t$)

c. count^{++};

Procedure GenerateRules //生成关联规则

input：频繁项集 L；最小兴趣度 mi；最小置信度 mc；

output：关联规则集合 R，R=∅

for any frequent itemset X in L do

begin

for any itemset A∪B=X and A∩B=∅ do

begin

if interest(A,B)>=mi

if confidence(A→B)>=mc

R=R∪{A→B}

if confidence(B→A)>=mc

R=R∪{B→A}

end

end

5 算法分析

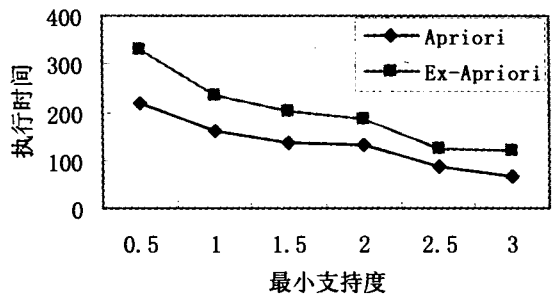
因为Ex-Apriori算法是由Apriori改进而来的，在过程GenerateAllFrequentItems中将非项加入了迭代，这没有影响Apriori算法的理论依据，所以在GenerateAllFrequentItems调用结束后能够保证生成所有的频繁项目集。

Apriori算法最大的时间消耗就花在对数据库的遍历上，而遍历的次数由最大频繁集的项数决定。如果1-项频繁集的个数为m，Apriori算法产生的最大频繁集为k项集，Ex-Apriori算法产生的最大频繁项为j项集，则有k<=j<=m，本文Ex-Apriori算法最多遍历m次数据库(因为在项目集中某一项和它的非项不可能同时出现)，所以在最糟糕的情况下Ex-Apriori算法和Apriori算法消耗的时间是基本相同的，也就是说两种算法有着同一数量级的时间复杂度，尽管Ex-Apriori算法通常会产生比Apriori算法更高阶的频繁集，从而引起对数据库遍历次数的增加，但是通过分析可以预知这部分开销是在可以接受范围内的，同时我们也可以通过调整最小支持度来控制这种开销。由于非项的引入，使得候选集中项目个数增多，从而对支持数计算的次数也要增多。但是由于文中算法采用了新颖的支持数计算方法，在硬件设备高速发展的今天，这部分增支几乎可以忽略，因此在效率上Ex-Apriori算法是可行的。

同时，由于生成的频繁集的数目增多，导致候选关联规则的数目增多，为了能够有效地剔除其中冗余的规则，Ex-Apriori算法在GenerateRules过程中引入了兴趣度，这一措施最大可能地保证了最终获得规则的有效性。

综上所述，Ex-Apriori算法在保证能够挖掘所有可能的正负关联规则的前提下，尽可能地减小了计算的复杂性，且对挖掘出的关联规则进行了有效的筛选，所以该算法是有效的可行的。

为了进一步验证文中算法的正确性和可行性，我们用VC^{++} 6.0，SQL SERVER2000在CPU为P4-2.0GHz、内存512M、操作系统为Windows 2000(professional)的机子上实现了Ex-Apriori算法和Apriori算法，测试数据库的有关参数如下：|D|表示事务数据记录的数目；|T|表示事务数据记录的平均长度；|I|表示最大频繁项目集的平均长度；|L|表示最大频繁项目集的数目；N表示事务项目集的个数。在我们实验中，N=1000，|L|=2000，|D|=50k，|T|=20，|I|=10，实验结果如图1所示。图1表明文中Ex-Apriori算法与Apriori算法的执行时间是处于同等数量级的，故是可行的。



参考文献

1 Agrawal R, Imielinski T, Swami A. Mining association rules between sets of items in massive databases. In: Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data. ACM, Washington D.C. 1993. 207～216

2 Agrawal R, Srikant R. First algorithms for mining association rules in large database [A]. In: Proceedings of the 1994 International Conference on VLDB [C]. San Francisco: Morgan Kaufmann Publishers, 1994. 487～499

3 Wu X, Zhang C, Zhang S. Mining both positive and negative association rules. In: Proc. of ICML, 2002. 658～665

4 Brin S, Motwani R, Silverstein C. Beyond market basket: Generalizing associationrules to correlations. In: Proc. of SIGMOD, 1997. 265～276

5 Savasere A, Omiecinski E, Navathe S. Mining for strong negative associations in a large database of customer transactions. In: Proc. of ICDE, 1998. 494～502

6 Piatetsky-Shapiro. Discovery, analysis, and presentation of strong rules. In Knowledge discovery in Databases. AAAI/MIT, Menlo Park, Calif., USA, 1991. 229～248

7 朱玉全，陈耿，杨鹤标. 正负关联规则挖掘算法研究. 计算机科学，2006，33(3)：188～190