计算机科学2004Vol. 31№. 3

多元时间序列中跨事务关联规则分析的高效处理算法

董泽坤[1] 李 辉[2] 史忠植[2]

(中国科技大学研究生院计算机学部 北京100039)[1]

(中科院计算技术研究所智能信息重点实验室 北京100080)[2]

摘 要 用挖掘跨事务关联规则的方法分析多元时间序列,可以找到序列中不同采样点观察值之间相互影响的关系。本文为实现这一目的,提出一种新的分析方法:ES-Apriori。此方法通过减少数据库扫描次数,优化内存分配,能够高效地分析多元时间序列之间的关联规则。试验表明,用此方法分析中国证券市场的股票时间序列非常有效。

关键词 关联规则,跨事务,多元时间序列,ES-Apriori

An Efficient Algorithm for Mining Inter-transaction Association Rules in Multiple Time Series

DONG Ze-Kun[1] LI Hui[2] SHI Zhong-Zhi[2]

(University of Science and Technology of China, Beijing 100039)[1]

(Key Laboratory of Intelligent Information Processing, Institute of Computing Technology, Chinese Academy of Sciences, Beijing100080)[2]

Abstract We can use methods that mine inter-transaction association rules to analysis multiple time series for finding their relationships. In this paper, a new algorithm named ES-Apriori will be presented to mine inter-transaction association rules in multiple time series. This algorithm needs to search the whole database only one time, with the rational arrangement of memory usage it can successfully mine the association rules in time series. Experiments have shown that this method can efficiently analyze the time series of Chinese Stock Market.

Keywords Association rules, Inter-transaction, Multiple time series, ES-Apriori

1 引言

关联规则是数据挖掘领域的一项重要研究内容。关联规则的挖掘算法首先由 Agrawal 等在文[1]中提出。随后关联规则挖掘有多种扩充,如多层关联规则挖掘[3,4],多维关联规则挖掘[5],多表间关联规则挖掘[7]等等。在传统的关联规则(Intra-transaction association rules)挖掘算法中,挖掘的关联项都出现在同一事务或同一序列中,用这些方法无法挖掘存在于不同事务中的关联关系[6],而"跨事务"的关联分析(Inter-transaction association rules)可以解决这一问题[6]。

跨事务性可以帮我们找到多笔交易记录中不同事件之间的关联规则。把这种特性应用到多元时间序列中,可以分析多个采样点中不同时间序列的观察值之间的关联规则。例如,在多元股票时间序列中,可以发现类似如下规则:

R1:深发展第0天涨,浦发银行第1天跌->民生银行第2天涨(20%,80%)

由于R1中加入了时间因子,各项之间体现的关系已经不再是事件是否发生在同一时间,而是不同时间序列中各事件之间的前后关系。很明显,规则R1对时间序列的预测很有贡献。

如何运用特殊数据结构来节省对数据库扫描的次数以提升"跨事务性"关联规则算法效能,仍然是一个需要解决的问题。针对这一问题,本文从如下两个方面对多元时间序列的跨事务性关联分析做出了改进:(1)减少数据库扫描次数,(2)对读入内存的数据采用"分而治之"的策略。由此思想设计的算法命名为:ES-Apriori。

用 ES-Apriori 算法设计的程序对中国证券市场五年近500支股票的时间序列分析,表明 ES-Apriori 是一种行之有效的多元时间序列的跨事务关联规则分析方法。此算法产生所有频繁项集的过程中只需要扫描一次数据库;同时,一次调入内存的数据量大大降低。这两个特性对提高系统的效能有非常大的贡献,它们有效地缓解了因数据爆炸而耗尽内存的问题。

本文第2部分介绍多元时间序列的跨事务关联规则,第3部分给出 ES-Apriori 算法。试验在第4部分给出,最后是结束语。

2 多元时间序列的跨事务关联规则

多元时间序列合并集:设时间序列的集合 $S=\{s_1, s_2, \cdots, s_u\}$,$T_i$是在时刻 i 对 S 的观察值集合,$T_i=\{s_1(i), s_2(i), \cdots, s_u(i)\}(1\leqslant i\leqslant n)$,多元时间序列合并集 D 定义为:$D=\{T_1, T_2, \cdots, T_n\}$。$D$ 中每组观察值各分配一个识别号 TID。

多元时间序列的跨事务关联规则:设 $\sum=\{e_1(0), \cdots, e_1(w-1), e_2(0), \cdots, e_2(w-1), \cdots, e_u(0), \cdots, e_u(w-1)\}$是事件的集合,这些事件是多元时间序列合并集 D 中各序列观察值的属性,w 是 D 的滑动时间窗口。以时刻 s $(1\leqslant s\leqslant n-w+1)$ 为 D 的参考时间基准点,如果事件 $e_i(1\leqslant i\leqslant u)$ 出现在时刻 $s+x(0\leqslant x\leqslant w-1)$,则标记 $e_i(x)$ 在 T_s 中。每一个 $e_i(x)$ 分配一个识别号 IID。多元时间序列的跨事务关联规则是形如 X

董泽坤 硕士,研究方向:数据挖掘,机器学习。**李 辉** 博士后,研究方向:人工智能、数据挖掘、模式识别。**史忠植** 博士生导师,研究员,研究方向:人工智能、智能主体、机器学习。

$\Rightarrow Y$ 的蕴涵式，并且满足以下条件：

1. $X\subset\Sigma, Y\subset\Sigma$，

2. $\exists\ e_i(0)\in X,\ 1\leqslant i\leqslant u$，

3. $\exists\ e_j(q)\in X,\ 1\leqslant j\leqslant u, ((i=j)\wedge(0<q<w-1))\vee((i\neq j)\wedge(0\leqslant q<w-1))$，

4. $\exists\ e_i(p)\in Y,\ 1\leqslant i\leqslant u,\ max(q)<p\leqslant w-1$，

5. $X\cap Y=\Phi$

多元时间序列的跨事务关联规则的支持度和置信度与传统关联规则的定义相似。

3 多元时间序列的跨事务关联规则算法

和传统关联规则算法比较，跨事务关联规则算法要更复杂：

(1)要处理的数据超过算法能承受的范围后，频繁项集的数目将变得巨大而无法处理；

(2)候选集数目的增加导致更频繁的数据库扫描动作。

"跨事务性"关联规则算法 E-Apriori 和 EH-Apriori 在文[6]中提出，算法的思想源于 Apriori。与传统关联规则分析方法不同，这两个算法构造的 C_2 都包括 $e_i(0)|(1\leqslant i\leqslant u)$ 项。EH-Apriori 算法在产生 L_1 时采用了哈希表技术，它的效率要比 E-Apriori 算法高。

E(H)-Apriori 算法找每个 L_k 需要一次数据库扫描，当得到所有频繁项集后再生成规则。此算法不但频繁访问数据库，而且将其应用到实际股票数据的预测分析中，当加大涨跌幅度分段数、加大滑动时间窗口时，效果并不理想。

3.1 ES-Apriori 算法

为了减少数据库扫描次数，减少程序运行消耗的内存，本文在 Apriori 算法的基础上，按照"分而治之"的思想提出了一种新算法。并把它命名为 ES-Apriori(扩展的分步 Apriori 算法)。

算法 ES-Apriori

```
Step 1
1  C1={{ e1(x)} | (e1(x)∈Σ)∧(0≤x≤w-1)}
2  for each inter-time series transaction T1 in D
3       for each candidate c:e1(x)∈C1(e1∈T1+x)
4            {c. TID}∪=T1. TID
5  L1={(c. IID,{c. TID}),c:{e1(x)}|(c∈C1)∧c. IID= e1(x). IID
       ∧(|c. TID |≥support)}
Step 2
6  for each item: e1(0)∈L1{
7       for each item: ek(x)∈L1((x≠0)∨(x=0∧i<k))
8            C'2= {{ e1(0),ek(x)}}
9       for each candidate c∈C'2: { e1(0), ek(x)}{
10                {c . IID} = {e1(0). IID}∪{ek(x). IID} ;
11                {c . TID} = {e1(0). TID} ∩ {ek(x). TID}
12      }
13      L'2={({c. IID},{c. TID}),c:{e1(0),ek(x)}|(c∈C'2)∧(|
        c. TID |≥support)}
14      for(k=3;L'k-1≠Φ;k++){
15           Ck = ES-Apriori-Gen (L'k-1); // join L'k-1 with L'k-1
                without prune step
16           for each candidate c: { e1(0),..,ek(x)}{
17                {c . IID} = {e1(0). IID}∪…∪{ek(x)
                              . IID} ;
18                {c. TID } = {e1(0). TID }∩…∩{ ek(x)
                              . TID } ;
19           }
20           L'k ={({c. IID,{c. TID}),c:{ e1(0),…,ek(x)}|c∈C'k
                ∧(|c. TID |≥support)}
21      }
22      L'=∪k L'k(k>1)
23      R'=GenerateRule (L'∪L1);
24 }
25 R=∪R'
```

ES-Apriori 算法在第一阶段遍历数据库并构造 C_1 频繁候选集(1～5行)，同时记录每个 C_1 的 IID，包含此 IID 的 TID 集合和 TID 集合的计数值。计数值满足最小支持度阈值的 C_1 将加入到 L_1 中。在第二阶段，挑出所有的 $e_i(0)$ 项，并以每个 $e_i(0)$ 项为一步，每步构造所有包含这个 $e_i(0)$ 项的频繁项集(6～24行)。新产生的频繁 k-项集的 $\{IID\}$ 由两个 $k-1$-项集的 $\{IID\}$ 的并集组成，$\{TID\}$ 由两个 $k-1$-项集的 $\{TID\}$ 的交集组成。然后，输出包含 $e_i(0)$ 项的关联规则(23行)。将每一步得到的关联规则集合并(25行)，就是所有关联规则的集合。

3.2 ES-Apriori 算法特性

ES-Apriri 算法从两个方面提高了系统的性能。

3.2.1 *仅扫描一次数据库* 在文[2]中，一种命名为 AprioriTid 的算法被提出。AprioriTid 通过构造一个不断收缩的临时数据库，减少了扫描数据库的开销。但是，AprioriTid 算法只是降低了单次的数据库扫描空间，而总的扫描次数没有降低。ES-Apriori 算法继承并拓展了 AprioriTid 的思想。通过改变它的数据结构，将 $\langle TID,\{IID\}\rangle$ 改进为 $\langle\{IID\},\{TID\}\rangle$ 使算法扫描数据库的开销进一步减少。

ES-Apriori 算法仅在产生 C_1 时扫描一次数据库。以多元时间序列合并集 D 的第 T_i 个观察值集合为参考起始点，如果 T_{i+x} 集合中包含项 e_i，则在记录包含 $e_i(x)$ 项的 TID 链表中加入 T_i 的 TID。如果 TID 的计数值超过最小支持度计数值，将 $e_i(x)$ 项加入到 L_1 频繁项集。由于所有数据库信息保存在 $L_1\{\langle IID,\{TID\}\rangle\}$ 中，以后的所有操作都可以脱离数据库。由 L_1 构造 $C_2\{\langle\{IID_{l_1}^1, IID_{l_1}^2\},\{TID_{l_1}^1\cap TID_{l_1}^2\}\rangle\}$ 时直接连接两个 $L_1: l_1\infty l_2$，并求 l_1 和 l_2 的 $\{TID\}$ 的交集，计算交集中 TID 的个数就可以得到 $l_1\infty l_2$ 的支持度计数。这样，不用再次扫描数据库就可以直接生成候选集 C_k，近而得到频繁项集 L_k。

DataBase

TID	IID
100	1 3 4
200	2 3 5
300	1235
400	25

C_1

IID	TID
1	100 300
2	200 300 400
3	100 200 300
4	100
5	200 300 400

L_1

IID	TID
1	100 300
2	200 300 400
3	100 200 300
5	200 300 400

C_2

IID	TID
1 2	300
1 3	100 300
1 5	300
2 3	200 300
2 5	200 300 400
3 5	200 300

L_2

IID	TID
13	100 300
23	200 300
25	200 300 400
35	200 300

C_3

IID	TID
235	200 300

L_3

IID	TID
235	200 300

图2 ES-Apriori 算法中候选集和频繁项集的产生

在图2所示的数据库中，假设有5项 Item，4笔交易，最小支持度计数为2。如图中所示，L_1 由满足支持度计数的 C_1 构成，C_2 由 L_1 连接构成。L_1 中的 $IID=1$ 和 $IID=2,3,5$ 分别连接可以构成{1,2}、{1,3}、{1,5}三项，它们的 TID 分别为{100,300}∩{200,300,400}={300}，{100,300}∩{100,200,300}={100,300}，{100,300}∩{200,300,400}={300}。由于{1,2}、{1,5}两项的 TID 计数值小于最小支持度计数值2，所以不再加入到 L_2 中。如此类推，L_2 由4项组成，即：{1,3}、{2,3}、{2,5}、{3,5}。L_2 连接形成 C_3，C_3 只有一项，且满足最小支持度计数，所以〈{2,3,5},{200,300}〉加入到 L_3，算法结束。

3.2.2 *减少了单次调入内存的数据量* ES-Apriori 的所有大项集保留了各项的数据库信息，从而减少了数据库扫描次数，但是它的代价是使用大量内存。所以，如何合理分配内存，是 ES-Apriori 算法的另一重点。

利用多元时间序列的跨事务关联规则性质(如下)，可以分步构造 L_k，使每一步需要扫描的空间大幅缩减，从而降低内存的开销。

多元时间序列的跨事务关联规则性质：所有规则都可以在各时间序列观察值的参考时间基准项 $e_i(0)$ 的基础上产生。

此性质可以由多元时间序列的跨事务关联规则定义中的条件2(频繁项集的 X 子集必定存在相对地址值为0的元素)推出。这一性质为 ES-Apriori 的分步策略提供了理论依据。

如图3所示，假设有2个时间序列的事件 A、B，滑动时间窗口=3，它的扩展1-项集为 A0、A1、A2、B0、B1、B2。考察6-项集{A0,A1,A2,B0,B1,B2}，它包含的规则有且仅有：(1)A0，B0→A1,A2，B1,B2；(2)A0，A1,B0，B1→A2，B2。在计算这两条规则的置信度时，只需要搜索由 A0构造的频繁项集空间，并不需要搜索由 B0构造的频繁项集空间。因为这个6-项集的符合多元时间序列跨事务关联规则条件的所有 X 子集只有{A0，B0}、{A0，A1,B0，B1}，这两项都是在 A0的基础上构造产生的("连接步"中，重复出现的项集不再构造，所以，不在 B0的基础上再次构造)。同理，5项集{B0，A1,A2,B1,B2}的 X 子集{B0}、{B0，A1，B1}只需搜索由 B0构造的频繁项集空间。

基本项	1-项集	2-项集	3-项集	4-项集	5-项集	6-项集
A	A0	A0A1	A0A1A2	A0A1A2B0	A0A1A2B0B1	A0A1A2B0B1B2
			A0A1B0	A0A1A2B1	A0A1A2B0B2	
			A0A1B1	A0A1A2B2	A0A1A2B1B2	
			A0A1B2	A0A1B0B1	A0A1B0B1B2	
				A0A1B1B2		
				A0A2B0B1		
		A0A2	A0A2B0	A0A2B0B2	A0A2B0B1B2	
			A0A2B1	A0A2B1B2		
			A0A2B2	A0B0B1B2		
		A0B0	A0B0B1			
			A0B0B2			
		A0B1	A0B1B2			
	A1A2	A0B2				
B	B0	B0A1	B0A1A2	B0A1A2B1	B0A1A2B1B2	
		B0A2	B0A1B1	B0A1A2B2		
		B0B1	B0A1B2	B0A1B1B2		
		B0B2	B0A2B1	B0A2B1B2		
			B0A2B2			
	B1B2		B0B1B2			

图3 跨时间序列关联规则性质

从上面的分析得出，挖掘所有规则可以分成 u 步运行：每步只挖掘包含 $e_i(0)|(1\leqslant i\leqslant u)$ 的关联规则。这样，一次调入内存的数据可降低为全部调入的1/u，当有大量数据项参与运算时，此方法也能顺利运行。

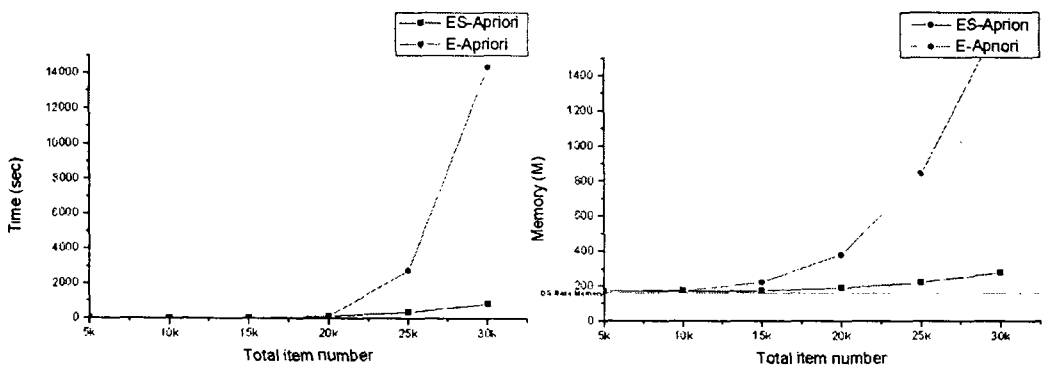

图4 ES-Apriori 和 E-Apriori 的时间/空间性能对比

ES-Apriori 算法分割了频繁项集空间，降低了一次调入内存的数据量，从而缓解了因数据爆炸而耗尽内存的问题。

4 实验

本文使用中国证券市场1997～2001年共1125个交易日近500支股票的收盘价时间序列作为测试集，比较了 E-Apriori 和 ES-Apriori 算法的性能。

实验使用 PIII667，512M 内存，操作系统是 Win2000 Server 的计算机。每支股票的涨跌幅度分成两段，频繁项最大长度为6，时间窗口为3，最小支持度为1%，分别使用含有5k～30k 个数据项的数据测试，结果如图4所示。

由图中可知，当项的总数小于20k 时，E-Apriori 和 ES-Apriori 的执行效率都很高。但是随着数据的增加，E-Apriori 的内存使用量将急速增加，导致运算时间骤然变长；而 ES-Apriori 无论在内存上还是在时间上都呈现平稳增加的态势。在试验中，当项的个数大于30k 后，E-Apriori 会耗尽计算机内存而无法继续运行；而 ES-Apriori 却可以顺利运行。实验结论证明，分析较大数据量的多元时间序列的跨事务关联规则时，ES-Apriori 算法在时间/空间性能上要优于 E-Apriori 算法。

结论 本文提出了一种新的多元时间序列的跨事务关联规则分析方法 ES-Apriori。由于 ES-Apriori 在计算中使用了分步策略，使得每步计算的搜索空间急剧下降，内存的开销也较 EH-Apriori 小很多。所以，在增加数据、增大时间窗口、加大涨跌幅度分段数时，算法仍然能够顺利运行。本文使用中国证券市场1997～2001年间1125个交易日，近500支股票的时间序列作为实验数据，证明 ES-Apriori 非常有效。

参 考 文 献

1 Agrawal R, Imielinski T, Swami A. Mining assocation rules between sets of items in large databases. In: Proc. of the ACM SIGMOD Conf. on Management of Data, 1993

2 Agrawal R, Srikant R. Fast algorithms for mining association rules. In: Proc. of the 20th Conf. on Very Large Data Bases, 1994

3 Han J, Fu Y. Discovery of multiple-level association rules from large databases. In: Proc. of the 21th Conf. on Very Large Data Bases, 1995

4 Srikant R, Agrawal R. Mining generalized association rules. In: Proc. of the 21th Conf. on Very Large Data Bases, 1995

5 Kamber M, Han J, Chiang Y. Metarule-guided mining of multi-dimensional association rules using data cubes. In: Proc. of the Knowledge Discovery and Data Mining, 1997

6 Lu H, Han J, Feng L. Stock movement and n-dimensional inter-transaction association rules. In: Proc. of the SIGMOD Wrkshop on Research Issues on Data Mining and Knowledge Discovery, 1998

7 史忠植. 知识发现. 清华大学出版社，2002

（上接第107页）

式表示的。如上下位关系使用一种缩进的书写方式来表示，对义和反义关系是分别用对义和反义关系表表示的。我们首先提取出这些隐式关系，然后将这些关系表示在类中。

总之，这个转换器就是将义原的各种关系挖掘出来，并用类表示出来。在图3中，将变量"canSubject"定义为集合类是为了它能容纳多个事件。

概念词典中有62000左右词条，我们同样使用一个转换器，以把每个概念表示为范畴属性的子类（如图5）。

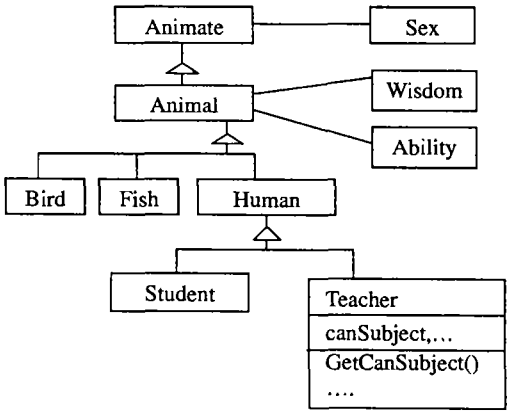

图5 概念类 Teacher 的继承关系以及部分关联

现在，假如判别"教师"是不是"动物"，如果在知网里面，需要找到"教师"的范畴属性，然后看"动物"是否是其范畴属性的上位义原，才能判断，其操作不易。但如果使用类库，其操作就在"教师"和"动物"类中进行即可完成。由此，使用对象技术表示知识后，其各种操作将得到简化并提供更高的灵活性。

另外，为便于系统的扩展，我们将为上层提供的接口从义原类库中分离出来，提供一个公共的接口，实现对各种关系的查询和判别，经过这样的设计，系统变成一个开放式的系统，便于今后的扩展和补充。

结论 本文在充分理解和分析知网的基础上，使用面向对象的技术来表示知网的语义知识，把知网中的上下位关系用继承机制表示出来，其它关系在类中设置变量来表示，进而把义原集合转化为面向对象技术中的类库，将概念词典表示为词条的 DEF 中的范畴属性类的子类。通过以上的处理，就可以将知识转化为利用面向对象技术表示的知识，使得这些知识变得容易操作，为上层的应用提供方便快捷的服务。更重要的是，通过这样的处理，说明利用面向对象的技术表示知识是确实可行的，为我们今后利用面向对象技术表示从文档中获得的知识提供基础。

利用面向对象的技术表示知网中的知识是我们从文本中提取知识的第一步，为我们今后从文本中提取知识提供丰富的资源，利用这些资源从文本中提取知识并将这些知识利用面向对象的技术表示出来，这样既可以用计算机加以处理，又为信息检索、文本过滤、信息检测提供方便使用的工具。

参 考 文 献

1 Wong Ping Wai, Yang Yongsheng. a maximum entropy approach to HowNet-Based Chinses Word Disambiguation. processing of COLING-ACL'02 . 2002

2 Yang XiaoFeng, Li Tangqu . A Study of Semantic Disambiguation Based HowNet. International Journal of computational linguistics and Chinese Language processing, 2002, 7(1): 47～78

3 董振东. 知网. http://www.keenage.com

4 周强，冯松岩. 构建知网关系的网状表示. 中文信息学报，2001，14(6):21～27

5 Wang C Y. Sense Pruning by HowNet - a knowledge-based Word Sense Disambiguation:[MPhil Thesis]. Hong Kong University of Science and Technology

6 Miller, George A. Nouns in WordNet: a lexical inheritance system. Five Papers on WordNet:[CSL Report 43]. Cognitive Science Laboratory, Princeton University, 1993

7 Baker C F, Fillmore C J, Lowe J B. The Berkeley FrameNet Projiect . In: processing of COLING-ACL'98, 1998. 86～90

8 Richardson S D, Dolan W B, Vandervende L. MindNet: acquiring and struct-uring semantic information from text. Processing of COLING-ACL'98, 1998. 1098～1092