文章编号:1001 - 9081(2005)06 - 1360 - 02

一种有效的挖掘关联规则更新方法

王 新
(云南民族大学 数学与计算机科学学院,云南 昆明 650031)
(wxkmyn@yahoo.com.cn)

摘 要:在挖掘关联规则过程中,用户往往需要多次调整(增加或减少)最小支持度,才能获得有用的关联规则。给出一个利用已存信息有效产生新候选项目集的 PSI 算法,结果表明每次扫描数据库时能有效地减少候选项目集的数。

关键词:数据挖掘;关联规则;Apriori 算法;PSI 算法
中图分类号: TP311.13 **文献标识码**:A

Efficient updating method for mining association rules

WANG Xin
(*School of Mathematics and Computer Science, Yunnan University of the Nationlities, Kunming Yunnan* 650031, *China*)

Abstract: In mining association rules, an user may required to tune (increase or reduce) the value of the minimum support many times before a set of useful association rules could be obtained from the transaction database. In this paper, an algorithm PSI was given for efficient generation of new candidate itemsets using prestored information. It can significantly reduce the number of candidate itemsets in each database scan.

Key words: data mining; association rules; Apriori algorithm; PSI algorithm

挖掘关联规则的关键问题是求出大项目集,为提高挖掘关联规则的效率,在 Apriori 算法[1]的基础上,已提出了许多改进算法,如 AprioriTid、AprioriTidList、AprioriHybrid 等。Apriori 算法存在的主要问题是时间开销大,主要反映在两个方面,一是对海量数据库的多次扫描,另一个是用链接产生大项目集。虽然后来的改进算法对此做了积极的工作,但是,改进算法仍沿用了生成大项目集的策略,因此效率提高不明显。在实际应用中,我们发现用户在大型事务数据库中挖掘关联规则时,往往需要通过调整(增加或减少)最小支持度,对同一事务数据库进行多次挖掘,从分析、比较中获取有用的关联规则。传统的方法是只要改变了最小支持度的值,就必须重新利用 Apriori 算法或其改进算法,求出大项目集,而没有考虑利用上一次挖掘得到的有用信息。

本文讨论了如何利用已存信息,减少挖掘关联规则的时间消耗,给出一种有效求出新候选项目集的更新方法——PSI 算法,并与 Apriori 算法进行了简单的比较。

1 挖掘关联规则的基本方法

令 $I = \{i_1, i_2, \cdots, i_m\}$ 为项目集,D 为事务数据库,其中每个事务 T 是一个项目子集($T \subseteq I$),并具有一个唯一的标识 TID。设 A 是一个由项目构成的集合,称为项集。如果项集的支持度超过用户给定的最小支持度阈值,称该项集是频繁项目集(大项目集)。关联规则是形如 $X \Rightarrow Y$ 的逻辑蕴含式,其中 $X \subset T, Y \subset T$,且 $X \cap Y = \varnothing$。如果事务数据库中 s %的事务包含 $X \cup Y$,那么我们说关联规则 $X \Rightarrow Y$ 的支持度为 s,记 $sup(X \Rightarrow Y) = sup(X \cup Y) / \| D \|$。如果事务数据库里包含 X 的事务中 c %的事务同时也包含 Y,那么我们说关联规则 $X \Rightarrow Y$ 的置信度为 c,记 $conf(X \Rightarrow Y) = sup(X \Rightarrow Y) / sup(X)$。挖掘关联规则的问题可分解为以下两步:第一步是产生所有的频繁项目集(大项目集),即找出所有支持度不低于用户指定的最小支持度(MinSup)的项目集;第二步是从第一步得到的大项目集中构造置信度不低于用户指定的最小置信度(MinConf)的规则。

挖掘关联规则关键是第一步,即计算大项目集。下面先给出基本的挖掘关联规则的 Apriori 算法。在 Apriori 算法中,项目是按字典排序,L_k 记为大 k-项目集构成的集合,C_k 记为候选 k-项目集构成的集合。

```
Apriori 算法:
L1 = {大 1 - 项集};
for (k = 2; Lk-1 ≠ ∅; k++) do begin
  Ck = apriori - gen (Lk-1);
  //从 Lk-1 产生新的候选 k - 项目集
  for all transactiona t ∈ D do begin
    Ct = subset (Ck, t);              //产生 t 中的候选子集
    for all candidates c ∈ Ct do
      c. count++;
end
Lk = {c ∈ Ck | c. count ≥ minsup};
end
answer = ∪k Lk;
```

其中,Apriori-gen 是以大(k - 1) - 项目集 L_{k-1} 为自变量的候选生成函数。该函数返回所有大 k-项目集的超集,分链接(join)和修剪(prune)两步执行:

```
函数 Apriori-gen ( Lk-1 ):
insert into Ck
select p[1], p[2], ···, p[k - 1], q[k - 1]
from Lk-1 p, Lk-1 q
where p[1] = q[1], p[2] = q[2], ···, p[k - 2] = q[k - 2],
  p[k - 1] < q[k - 1];// 链接(join)
```

收稿日期:2004 - 11 - 12;**修订日期**:2005 - 01 - 21 **基金项目**:云南省教育厅科研基金资助项目(022Y165)
作者简介:王新(1963 -),男,云南人,副教授,主要研究方向:知识发现与数据挖掘.

```
For all itemsets c ∈ C_k do
  For all (k - 1) - subset s of c do
    If (s ∉ L_{k-1}) then
      delete c from C_k;                    // 修剪(prune)
  end
end
answer = ∪ {c ∈ C_k};
```

利用大项目集生成关联规则算法为:

```
for all 大 k 项集 lk, k ≥ 2 do begin
  H_1 = {L_k 中规则的后件,该规则的后件中只有一个项目}
  call ap - genrules(l_k, H_1)
end
procedure ap - genrules(l_k:大 k 项集,
  H_m:m 个项目的后件的集合)
if (k > m + 1) then begin
  H_{m+1} = Apriori - gen(H_m)
  For all h_{m+1} ∈ H_{m+1} do begin
    Conf = supp(l_k)supp(l_k - h_{m+1})
    If (conf ≥ minconf) then
      output 规则(l_k - h_{m+1}) = > h_{m+1},
      置信度 = conf,支持度 = supp(l_k)
    else
      从 H_{m+1} 中删除 h_{m+1}
  end
  call ap - genrules(l_k, H_{m+1})
end
```

2　利用已存信息挖掘关联规则

当用户调整(增加或减少)最小支持度,对同一事务数据库多次挖掘关联规则时,利用最近挖掘的已存信息减少每次扫描数据库计算候选项目集数。已存储信息包括最近一次生成的大项目集和候选项目集。设 s_1 和 s_2 分别为最近挖掘和当前挖掘的最小支持度,M_1 和 M_2 相应地表示最近挖掘和当前挖掘大项目集。分别考虑以下两种情况:

1) 如果 $s_2 > s_1$,那么,M_2 能容易地从 M_1 中通过选取支持度不小于 s_2 的项目集得到。

2) 如果 $s_2 < s_1$,那么,M_2 能通过组合 M_1 和支持度在 $[s_2, s_1)$(如 $s_2 \leqslant x.count < s_1$)的项目集 x 的集合得到。

对于情况 2),关键是求出支持数在 $[s_2, s_1)$ 的大项目集。下面给出利用已存信息求出全部未知大项目集的 PSI 算法。

设 L_k 和 $\tilde{C}_k$ 分别表示最近挖掘的大 k-项目集和非大 k-项目集的候选集,假如 L_k 和 $\tilde{C}_k$ 中的项目已存在磁盘上,用 $\tilde{L}_k$ 表示 $\tilde{C}_k$ 中满足 $s_2 \leqslant x.count < s_1$ 条件的全部项目集 c 的集合。为说明方便,将算法 PSI 分为两部分:第一部分是为当前挖掘求出大 1 - 项目集,由于全部 1 - 项目集属于 $L_1 \cup \tilde{C}_1$,L_1 和 $\tilde{C}_1$ 分别更新为 $L_1 \cup \tilde{L}_1$ 和 $\tilde{C}_1 - \tilde{L}_1$。第二部分为产生候选项目集,求出大 k-项目集,类似于 Apriori 算法,PSI 也通过更新 L_{k-1} 产生候选 k-项目集,不同的是 PSI 仅产生未知支持数的候选项目集,称这些候选项目集为新候选项目集,记为 $\bar{C}_k$。新候选项目集 $\bar{C}_k$ 的产生由 *Procedure New_Candidate*$(L_{k-1}, T_k, \bar{C}_k)$ 给出描述。记 T_k 表示 $L_k \cup \tilde{C}_k$,其中 L_k 和 $\tilde{C}_k$ 分别表示最近挖掘存储的信息,其含义为在 T_k 中的每个项目集 x 的支持数已知,将 x 从 $\bar{C}_k$ 中删除。每次扫描,计算出全部新的候选项目集的支持数,设 $\bar{L}_k$ 为 $\bar{C}_k$ 中具有支持数在 $s_2 \leqslant x.count < s_1$ 的全部项目集 c 的集合。显然,支持数在 $[s_2, s_1)$ 的大项目集的集合是 $\tilde{L}_k \cup \bar{L}_k$,然后,$L_k$ 和 $\tilde{C}_k$ 分别更新为 $L_k \cup \tilde{L}_k \cup \bar{L}_k$ 和 $(\tilde{C}_k - \tilde{L}_k) \cup (\bar{C}_k - \bar{L}_k)$。

算法 PSI:

```
s_1 = 最近最小支持数;
s_2 = 当前最小支持数;
/* 第一部分 */
L̃_1 = {c | s_2 ≤ x.count ≤ s_1, c ∈ C̃_1};
L_1 = L_1 ∪ L̃_1;
C̃_1 = C̃_1 - L̃_1;                          /* 更新 L_1 和 C̃_1 */
/* 第二部分 */
for (k = 2; | L_{k-1} | > 1; k++) do begin
  T_k = L_k ∪ C̃_k;
  C̄_k = φ;
  New_Candidate(L_{k-1}, T_k, C̄_k);
  /* 产生新候选 k-项目集 */;
  forall transactions t ∈ DB do
  forall c ∈ C̄_k and c is contained in t do
  c.count++;
  L̃_k = {c | s_2 ≤ x.count ≤ s_1, c ∈ C̃_k};
  L̄_k = {c | s_2 ≤ x.count ≤ s_1, c ∈ C̄_k};
  L_k = L_k ∪ L̃_k ∪ L̄_k;
  C̃_k = (C̃_k - L_k) ∪ (C̄_k - L̄_k); /* 更新 L_k 和 C̃_k */
end
```

产生新候选项目集函数:

```
Procedure New_Candidate(L_{k-1}, T_k, C̄_k);
forall c = {c_p[1], c_p[2], ···, c_p[k-1], c_q[k-1]},
  C_p, C_q ∈ L_{k-1},
c_p[n] = C_q[n], 1 ≤ n ≤ k - 2, and c_p[k-1] < c_q[k-1] do
if c ∉ T_k and all (k - 1) - subsets s of c satisfies s ∉ L_{k-1} then
  C̄_k = C̄_k ∪ {c};
end Procedure
```

3　举例

表 1

TID	Items
001	BEG
002	ACDF
003	BFG
004	BDE
005	CF
006	BDF
008	ACE
009	BCE
010	ACE

设已知事务数据库 D 如表 1 所示。下面利用算法 PSI 说明产生大项目集的过程。设最近挖掘的最小支持数,大项目集和具有其支持数的非大项目集的候选集等如下信息已存在磁盘上。

$s_1 = 3$,

$L_1 = \{\{A\}_4, \{B\}_5, \{C\}_5, \{D\}_4, \{E\}_6, \{F\}_4\}$

$\tilde{C}_1 = \{\{G\}_2\}$,

$L_2 = \{\{A,C\}_3, \{A,E\}_3, \{B,E\}_3, \{C,E\}_3\}$,

$\tilde{C}_2 = \{\{A,B\}_0, \{A,D\}_2, \{A,F\}_1, \{B,C\}_1, \{B,D\}_2, \{B,F\}_2, \{C,D\}_1, \{C,F\}_2, \{D,E\}_2, \{D,F\}_2, \{E,F\}_0\}$,

$L_3 = \phi$,

$\tilde{C}_3 = \{\{A,C,E\}_2\}$,

此时 $M_1 = \{\{A,C\}_3, \{A,E\}_3, \{B,E\}_3, \{C,E\}_3\}$。其中,$X_s$ 表示一个支持数为 s 的项目集。

现假设当前挖掘的最小支持数变为 2,即 $s_2 = 2$,利用算法 PSI,可得 $\tilde{L}_1 = \{\{G\}_2\}$。$L_1$ 和 $\tilde{C}_1$ 分别更新为 $\{\{A\}_4, \{B\}_5, \{C\}_5, \{D\}_4, \{E\}_6, \{F\}_4, \{G\}_2\}$ 和 ϕ。然后,求出新的候选项目集 $\bar{C}_2$,并通过扫描一次数据库计算它们的支持数。有 $\bar{C}_2 = \{\{A,G\}_0, \{B,G\}_2, \{C,G\}_0, \{D,G\}_0, \{E,G\}_1, \{F,G\}_1\}$,$\tilde{L}_2 = \{\{A,D\}_2, \{B,D\}_2, \{B,F\}_2, \{C,F\}_2, \{D,E\}_2, \{D,F\}_2\}$ 和 $\bar{L}_2 = \{\{B,G\}_2\}$。$L_2$ 和 $\tilde{C}_2$ 分别更新为 $\{\{A,C\}_3, \{A,D\}_2, \{A,E\}_3, \{B,E\}_3, \{C,E\}_3,$

(下转第 1372 页)

假设下一个时间序列值只与前一个值相关,然而许多电力负荷数据流的变量之间的相关性不会以较快的速率衰减,观测值经历较长时间之后仍然具有显著的相关性。ADWT 产生的预测与实际值的相关性 +42%,比 MLP 的负相关性要好。

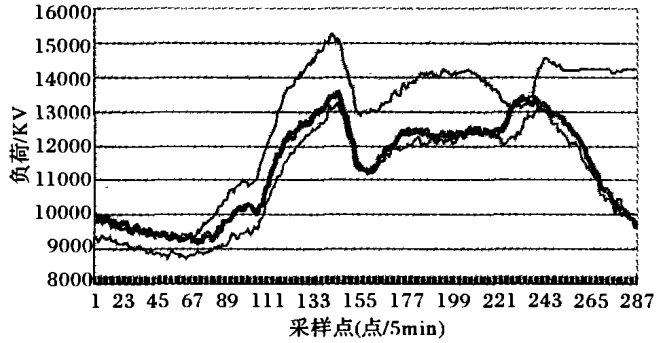

图 2 预测性能比较

5 结语

本文提出基于无偏似然估计软阈值的小波大纲构建方法,在其基础上的预测方法对分解后的平稳成分 A_L 使用线性回归预测,对各个级别系数 D_i 采用 Fourier 能量谱分析,根据数据的不同成分得到统计性信息的预测。与文献[5]相比,本文提出的基于小波大纲的数据流预测方法考虑了信号中必然存在的噪声的影响,形成的小波大纲可以有效地消除噪声,同时大幅度减少了内存中保存的小波系数的空间,实验证明利用这种小波大纲得到的预测信息在计算速度上有大幅提高,而质量并未明显降低。与神经元网络在预测中的应用相比,由于神经元网络的隐节点数目难以确定、过度拟合、训练时间长、预测精度对训练样本的质量和数量敏感等原因,不适合快速在线预测。文中所提算法可伸缩性能好,可以快速适应趋势的变化,能够广泛地应用于数据流挖掘领域。下一步工作包括实现增量更新的小波大纲预测算法,研究多数据流的相关性预测等。

参考文献:

[1] GABER MM, KRISHNASWAMY S, ZASLAVSKY A. Ubiquitous Data Stream Mining, Current Research and Future Directions[A]. Workshop Proceedings held in conjunction with The Eighth Pacific-Asia Conference on Knowledge Discovery and Data Mining[C], Sydney, Australia May 26, 2004.

[2] GILBERT AC, KOTIDIS Y, MUTHUKRISHNAN S, *et al.* Surfing wavelets on streams: One-Pass summaries for approximate aggregate queries [A]. In Proc. of the 27th Int'l Conf on VLDB[C] 2001.79-88.

[3] ZHENG H, ZHANG L. The factor analysis of short-term load forecast based on wavelet transform[J]. Power System Technology, Proceedings. PowerCon 2002. International Conference on, 2002, 2:13-17.

[4] RENAUD O, STARCK JL, MURTAGH F. Prediction Based on a Multiscale Decomposition, International Journal of Wavelets[J]. Multiresolution and Information Processing, 2003, 1(2):217-232.

[5] AHMAD S, TASKAYA-TEMIZEL T, AHMAD K. Summarizing Time Series: Learning Patterns in'Volatile' Series[A]. Proceedings of the 5th International Conference on Intelligent Data Engineering and Automated Learning (IDEAL 2004), Lecture Notes in Computer Science[C]. Heidelberg: © Springer-Verlag. 2004, 3177:523-532.

[6] BYOUNG-KEE Y, SIDIROPOULOS N, JOHNSON T, *et al.* Online data mining for coevolving time sequences[A]. In ICDE[C], 2000. 13-22.

[7] DONOHO D, JOHNSTONE I. Adapting to unknown smoothness via wavelet shrinkage[J]. Journal of the American Statistical Association 1995, 90(9):1200-1224.

[8] STEIN CM. Estimation of the mean of a multivariate normal distribution[J]. Annals of Statistics. 1981, 9:1135-1151.

[9] GAROFALAKIS M, GIBBONS PB. Probabilistic Wavelet Synopses [J]. ACM Transactions on Database Systems, 2004, 29(1):43-90.

[10] MALLAT S. A theory for multiresolution signal decomposition, the wavelet representation[J]. IEEE Trans. Pattern Ana. and Machine Intell, 1989, 2(7).

(上接第 1361 页)

$\{B, D\}_2$, $\{B, F\}_2$, $\{C, F\}_2$, $\{D, E\}_2$, $\{D, F\}_2$, $\{B, G\}_2\}$ 和 $\{\{A,B\}_0$, $\{A, F\}_1$, $\{A, G\}_0$, $\{B,C\}_1$, $\{C, D\}_1$, $\{C, G\}_0$, $\{D, G\}_0$, $\{E, F\}_0$, $\{E, G\}_1$, $\{F, G\}_1\}$,接着,产生新的候选项目集 $\overline{C}_3$,并通过再次扫描数据库计算它们的支持数。有 $C_3 = \{\{A, D, E\}_1, \{B, D, E\}_1, \{B, D, F\}_1\}$, $\widetilde{L}_3 = \{\{A, C, E\}_2\}$ 和 $\overline{L}_3 = \phi$,L_3 和 $\widetilde{C}_3$ 分别更新为 $\{\{A, C, E\}_2\}$ 和 $\{\{A,D,E\}_1, \{B, D, E\}_1, \{B, D, F\}_1\}$,因为 $|L_3| \leq 1$,挖掘过程终止。

故 $M_2 = \{\{A, C\}_3, \{A, D\}_2, \{A, E\}_3, \{B, E\}_3, \{C, E\}_3, \{B, D\}_2, \{B, F\}_2, \{C, F\}_2, \{D, E\}_2, \{D, F\}_2, \{B, G\}_2, \{A, C, E\}_2\}$,再利用大项目集生成关联规则算法可求得关联规则。

4 结语

综上可见,利用已存储信息,PSI 算法每次扫描能减少候选项目集支持数的计算。对于表 1 事务数据库 D,表 2 表示当前最小支持度变为 2 时,PSI 和 Apriori 算法的结果比较。PSI 利用最小支持数为 3 的已存信息,只需两次扫描数据库,而 priori 需要三次扫描数据库。而且,每次扫描数据库计算支持数的候选项目集总数,priori 比 PSI 的大。最后,需要指出 PSI 算法的思想,不仅可运用到挖掘关联规则的 priori 改进算法中,而且可运用到挖掘序列模式的 prioriall 算法中,以减少重复挖掘知识过程中的时间和空间上的开销。

表 2 Apriori 与 PSI 的比较

	Apriori 数		PSI 数
C_1	7		
C_2	21	$\overline{C}_2$	6
C_3	4	$\overline{C}_3$	3
总数	32		9

参考文献:

[1] AGRAWAL R, IMIELINSKI T, SWAMI A. Mining association rules between sets of items in large databases[A]. In: Proceedings of the ACM SIGMOD Conference on Management of Data[C]. Washington D. C, 1993.207-216.

[2] AGRAWAL R, SRIKANT R. Fast algorithm for mining association rules[A]. In: Proceedings of the 20th International Conference on Very Large Databases[C]. Santiago, Chile, 1994.478-499.

[3] YUGAMI N, OHTO Y, OKAMOTO S. Fast discovery of interesting rules[A]. In: Proceeding of the 4th Pacific-Asia Conference[C]. PAKDD2000, Japan, 2000.17-27.

[4] CHEN M-S, HAN J, YU PS. Data mining: An overview from a database perspective[J]. IEEE Transactions on Knowledge and Data Engineering, 1996, 8(6):866-883.

[5] TSAI PSM, CHEN CM. Discovering knowledge from large database using prestored information Information System[J], 2001, 26(1):1-14.