基于时间窗口的增量式关联规则更新技术*

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系 合肥 230027)

摘要 文章提出了基于时间窗口的增量式关联规则更新技术.该方法不仅可以利用在先前发现过程中已经获得的结果,而且利用时间窗口,还可以在最近的数据集中进行知识发现.

关键词 知识发现,关联规则,增量式更新,时间窗口.

中图法分类号 TP311

在数据库中发现知识(knowledge discovery in databases,简称KDD),亦称数据发掘(data mining),是当今国际上人工智能和数据库研究的一个主要热点课题[1].在KDD研究中,人们较多地侧重于知识发现技术与工具的研究,而对已发现的知识的更新、维护问题则较少注意[2].

关联规则是Rakesh Agrawal等人首先提出的一个重要的KDD研究课题[3].在现实世界数据库中可以发现各种各样的时态数据,例如,超市的交易记录有时间标记、病员的病历数据记录、天气数据日志文件等等.时态数据的出现使得有必要在知识发现过程中考虑时间因素.为此,本文在交易数据库的背景下,研究了时态关联规则的维护问题.该问题的关键是由于数据随时间的变化而变化,当前已发现的某些关联规则可能不再有效,而可能存在的新的有效关联规则有待进一步去发现.商业机构中的交易是一个不断进行的过程,交易行为的模式很可能随时间呈现出某种发展趋势或周期性.这种发展趋势或周期性对于市场计划与分析是很有价值的.本文集中讨论时态关联规则的更新与存储.一种方法是接受所有的新数据,连同过去的旧数据一起,重新运行普通的关联规则发现算法.该方法的缺点是要重新处理已经处理过的数据,不能有效地利用已经获得的结果.另一种方法是随着新数据的产生增量式地更新关联规则集,尽可能地只处理新数据.显然,后者更可取,因而也是本文所采取的方法.

目前,在增量式更新关联规则方面已有一些工作[2,4,5],但均未考虑时间因素,因而未能明确提出过时数据的淘汰策略.这样,就不能适应时态关联规则的更新维护需要.为此,我们提出基于时间窗口的增量式关联规则更新技术.该算法不仅可以利用在先前发现过程中已经获得的结果,而且利用时间窗口,还可以在最近的数据集中进行知识发现.

1 基于时间窗口的增量式更新技术

所谓时间窗口是指这样一个时间区间,在该区间之外的交易数据均认为是过时的,不用于当前关联规则的发现过程.这样,发现算法便可集中在最近的数据上,提高了发现结果的时效性.常规关联规则发现算法将所有规则保存在一个集合中,这些规则适用于整个数据库;而带时间约束的关联规则是在当前时间窗口中有效的关联规则.

令当前时间窗口为Cur_Window,其起止时间分别为T_{start}和T_{end},DB为在当前时间窗口Cur_Window中的交易所构成的交易数据库,D为其中的交易数,L为交易数据库DB的频繁项目序列集,s为最低支持,c为最低

---

* 本文研究得到国家自然科学基金和国家教委博士点基金资助.作者**欧阳为民**,1964年生,博士,副教授,主要研究领域为KDD,机器学习,人工智能及其应用.**蔡庆生**,1938年生,教授,博士生导师,主要研究领域为机器学习,知识发现,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1998-02-12收到原稿,1998-04-16收到修改稿

信任. 注意，s 和 c 均为百分数. 假定对每个项目序列 $X\in L$，其支持数 $X.support$（即在 DB 中包含 X 的交易数）是可以利用的.

设从时间 T_{end} 开始到 T_{now} 结束，对交易数据库作了某些更新，新的交易所构成的集合为 db，d 为 db 中的交易数. 新时间窗口为 New_Window，其起止时间分别为 $T_{now}-(T_{end}-T_{start})$ 和 T_{now}. 这样，在时间 $T_{now}+T_{start}-T_{end}$ 之前的交易数据不在新时间窗口 New_Window 中，应予淘汰. 记在时间 $T_{now}+T_{start}-T_{end}$ 之前的交易所构成的集合为 $retire$，其交易数记为 r. 我们记对交易数据库更新后在新时间窗口 New_Window 中的交易所构成的交易数据库为 $NewDB$，那么 $NewDB=DB\cup db/retire$. 对同样的最低支持 s，如果某项目序列 X 在新时间窗口中的支持不低于 s，即 $X.support\geqslant s\times(D+d-r)$，那么，$X$ 在 $NewDB$ 中就是频繁的.

基于时间窗口的关联规则更新的关键在于发现更新后在新时间窗口中的数据库 $NewDB$ 中的频繁项目序列集 $NewL$. 注意，原频繁序列集 L 中的频繁序列 X 在更新后的数据库 $NewDB$ 有可能不再是频繁的，即 $X\in L$，但 $X\notin NewL$ 却是可能的；而不在 L 中的频繁序列 X 在更新后的数据库 $NewDB$ 也有可能变为频繁的，即 $X\notin L$，但 $X\in NewL$ 却是可能的.

本文余下部分采用如下记号. 数据库 DB 中长度为 k 的频繁项目序列（称为频繁 k-项目序列）的集合记为 L_k，更新后的数据库 $NewDB$ 中的新的频繁 k-项目序列集记为 $NewL_k$，C_k 为算法 IWUP 第 k 次循环中长度为 k 的候选集. 另外，$X.support_D$，$X.support_r$ 和 $X.support_d$ 以及 $X.support_N$ 分别为项目序列 X 在 DB，$retire$，db 和 $NewDB$ 中的支持数. 下面，我们首先讨论 TWUP 算法的第 1 次循环，接着讨论后继的各次循环. 限于篇幅，这里略去完整的算法描述.

1.1　频繁 1-项目序列集的更新

在推导更新后数据库存 $NewDB$ 中的频繁 1-项目序列时，如下性质是非常有用的.

引理 1. 某原频繁 1-项目序列 $X\in L_1$ 在更新后数据库存 $NewDB$ 中是非频繁的，当且仅当 $X.support_N<s\times(D+d-r)$.

证明：由最低支持和频繁 1-项目序列的定义可直接推知.

引理 2. 对某原非频繁 1-项目序列 $X\notin L_1$，该项目序列 X 在更新后数据库 $NewDB$ 中是频繁 1-项目序列的必要条件是 $X.support_d\geqslant s\times(d+k-r)$，其中 $s\times k$ 为 X 在淘汰数据库 $retire$ 中的支持数.

证明：既然 $X\notin L_1$，那么 $X.support_D\leqslant s\times D$. 假定 $X.support_d<s\times(d+k-r)$，那么 $X.support_N=X.support_D+X.support_d-s\times k<s\times D+s\times(d+k-r)-s\times k<s\times(D+d-r)$. 这样，$X$ 在更新后数据库 $NewDB$ 中就是非频繁的. 于是，引理 2 得证.

基于上述两引理，在 $NewDB$ 中发现频繁 1-项目序列集 $NewL_1$ 可按如下步骤进行：

(1) 遍历淘汰数据库 $retire$，计算所有项目序列 $X\in L_1$ 在 $retire$ 中的支持数，记为 $X.support_r$；

(2) 遍历新增数据库 db，计算所有项目序列 $X\in L_1$ 在 db 中的支持数，记为 $X.support_d$，从而得到 L_1 中所有项目序列 X 在更新后数据库 $NewDB$ 中的支持，$X.support_N=X.support_D+X.support_d-X.support_r$. 检查 $X.support_N$ 的大小，如果 $X.support_N$ 低于 $s\times(D+d-r)$，根据引理 1，X 在更新后数据库 $NewDB$ 中就是非频繁的，因而予以淘汰. 经过这一过滤后，L_1 中剩下的在更新后数据库 $NewDB$ 中就是频繁 1-项目序列.

(3) 在对 db 和 $retire$ 作上述遍历的同时，根据 db 中的每一交易 t 构造不在 L_1 中的候选 1-项目序列集 C_1，分别计算各候选在 db 和 $retire$ 中的支持. 按照引理 2，对 C_1 中的任一项目序列 X，如果 $X.support_d<s\times(d+k-r)$，那么 X 在更新后数据库 $NewDB$ 中就必是非频繁的. 因此，可将 X 从 C_1 中删除. 这样，我们便可对 C_1 进行修剪，删除其中所有那些在 db 中的支持低于 $s\times(d+k-r)$ 的候选.

(4) 对原部分数据库 $DB/retire$ 进行遍历，计算 C_1 中各个候选 X 在 $DB/retire$ 中的支持数，从而得到 X 在更新后数据库 $NewDB$ 中的支持数 $X.support_N$. 通过检查 $X.support_N$ 是否不低于 $s\times(D+d-r)$，我们可以从 C_1 中发现新的频繁 1-项目序列. 更新后数据库 $NewDB$ 中的频繁 1-项目序列集 $NewL_1$ 由原 L_1 中在 $NewDB$ 中仍是频繁的项目序列和在 C_1 中发现的新频繁项目序列共同组成.

与 Apriori 算法的第 1 次循环相比，TWUP 算法首先从原频繁 1-项目序列 L_1 中排除在更新后数据库 $NewDB$ 中不再是频繁的项目序列. 经此过滤后，L_1 中剩下的在更新后数据库 $NewDB$ 中就全是频繁 1-项目序

列.而达到这一目的,仅需对新增数据库 db 和淘汰数据库 $retire$ 作1次遍历.该算法还根据引理2,对根据 db 构造出的候选集 C_1 进行修剪,排除那些不可能成为新频繁项目序列的元素.这两项工作均在对新增数据库 db 和淘汰数据库 $retire$ 作1次遍历中完成.然后,该算法对原部分数据库 $DB/retire$ 作1次遍历,以从 C_1 中发现新的频繁项目序列.与此形成对照,Apriori 算法必须以所有数据项目构成长度为1的候选集,对整个更新后数据库 $NewDB$ 进行遍历,以发现频繁1-项目序列集 $NewL_1$.显然,TWUP 算法所涉及的候选数远远少于 Apriori 算法所涉及的候选数.

1.2 频繁 k-项目序列集的更新

在推导更新后数据库 $NewDB$ 中频繁2-项目序列集 $NewL_2$ 时,下列性质将是有用的.

引理3. 如果某项目序列 $\{X_1,X_1,\ldots,X_{k-1}\}\in L_{k-1}$ 在第 $k-1$ 次循环中发现,在更新后数据库 $NewDB$ 中不再是频繁的,即 $\{X_1,X_1,\ldots,X_{k-1}\}\notin NewL_{k-1}$,那么,包含该项目序列的任何原频繁 k-项目序列 $Y\in L_k$(对 $k\geqslant 2$)在更新后数据库 $NewDB$ 中均不可能成为频繁项目序列,即 $Y\notin NewL_k$.

证明:根据频繁项目序列的任何子项目序列均必是频繁的(参阅文献[3])这一性质可以推知.

引理4. 原频繁 k-项目序列集 L_k 中的任何 k-项目序列 $\{X_1,X_1,\ldots,X_k\}$ 在更新后数据库 $NewDB$ 中是非频繁的,当且仅当 $\{X_1,X_1,\ldots,X_k\}.support_N<s\times(D+d-r)$.

证明:由最新支持和频繁 k-项目序列的定义可直接推知.

引理5. 对某原非频繁 k-项目序列 $\{X_1,X_1,\ldots,X_k\}\notin L_k$,该项目序列 X 在更新后数据库 $NewDB$ 中是频繁 k-项目序列的必要条件是 $\{X_1,X_1,\ldots,X_k\}.support_d\geqslant s\times(d+k-r)$,其中 $s\times k$ 为 $\{X_1,X_1,\ldots,X_k\}$ 在淘汰数据库 $retire$ 中的支持数.

证明:类似于引理2.

基于上述引理,在 $NewDB$ 中发现频繁2-项目序列集 $NewL_2$ 可按如下步骤进行:

(1) 根据引理3,从 L_2 中删除在 $NewDB$ 中不再是频繁的项目序列.第1次循环所识别出的不再是频繁的项目序列的集合为 L_1-NewL_1.对任何项目序列 $X\in L_2$,如果存在某子项目序列 $Y\in L_1-NewL_1$,那么,X 在更新后数据库 $NewDB$ 中就不可能是频繁的,因而可将 X 从 L_2 中删除.这样,我们便对 L_2 进行了修剪.

(2) 分别对新增数据库 db 和淘汰数据库 $retire$ 作1次遍历,计算修剪后 L_2 中每个项目序列 X 在 db 和 $retire$ 中的支持数 $X.support_d$ 和 $X.support_r$,从而计算出 X 在更新后数据库 $NewDB$ 中的支持,$X.support_N=X.support_D+X.support_d-X.support_r$.根据引理4,在 $NewDB$ 中淘汰所有非频繁项目序列.这样,L_2 中剩下的在 $NewDB$ 中就全是频繁2-项目序列.

(3) 本步骤是发现新的频繁2-项目序列.首先,利用候选生成算法 Apriori_gen[3] 根据 $NewL_1$ 生成候选集 C_2,注意,由于 L_2 中的项目序列已经处理过了,所以应从 C_2 中删除属于 L_2 的项目序列.然后,分别对 db 和 $retire$ 作1次遍历,计算每个候选 X 在 db 和 $retire$ 中的支持数 $X.support_d$ 和 $X.support_r$.接着对 C_2 作进一步修剪,即对任何 $X\in C_2$,如果 $X.support_d<s\times(d+k-r)$,那么,根据引理5,$X$ 在 $NewDB$ 中就必是非频繁的,因此可将 X 从 C_2 中删除.

(4) 对原部分数据库 $DB/retire$ 进行遍历,计算 C_2 中各个候选 X 在 $DB/retire$ 中的支持数,从而得到 X 在 $NewDB$ 中的支持 $X.support_N$.对每个候选 $X\in C_2$,如果 $X.support_N\geqslant s\times(D+d-r)$,那么 X 就是新的频繁2-项目序列.$NewDB$ 中的频繁2-项目序列集 $NewL_2$ 由原 L_2 中在 $NewDB$ 中仍是频繁的项目序列和在 C_2 中发现的新频繁项目序列共同组成.

对第 $k(k\geqslant 3)$ 次循环,应用上述算法,直到没有新的候选产生时为止.在 TWUP 算法的第 k 次循环中,各数据库仅遍历1次.对原频繁 k-项目序列,该算法只对新增数据库 db 和淘汰数据库 $retire$ 作1次遍历,就可发现在更新后数据库 $NewDB$ 中仍是频繁的 k 项目序列;而对新的频繁 k-项目序列,其候选集 C_k 根据上次循环所得到的 $NewDB$ 中的频繁 $(k-1)$-项目序列集 $NewL_{k-1}$ 生成,然后利用引理3和引理5对 C_k 进行修剪.经过如此修剪后的候选集 C_k,要远远小于在 Apriori 算法中所生成的候选集.这表明,在关联规则更新方面,TWUP 算法的性能要优于重新运行 Apriori 算法.

2 小　结

本文提出了一种基于时间窗口的增量式关联规则更新维护算法.利用时间窗口技术,我们得以提高关联规则集与当前数据的相关性,同时还可利用已经获得的结果,降低规则集更新的处理代价.TWUP 算法已在 PC 586/166(32M 内存)上用 Visual FoxPro1.0 实现,并用合成数据进行了测试.实验表明,该算法思想正确,具有良好的可扩放性.

参考文献

1 欧阳为民,蔡庆生.大型数据库中多层关联规则的元模式制导发现.软件学报,1997,8(12):920～927
(Ou-Yang Wei-min, Cai Qing-sheng. Meta-pattern guided discovery of multiple-level association rule in large databases. Journal of Software, 1997,8(12):920～927)

2 Cheung D W, Han J, Ng V *et al*. Maintenance of discovered association rules in large databases: an incremental updating technique. In: Proceedings of 1996 International Conference on Data Engineering. 1996. http://www.cs.hku.hk/~dcheung

3 Agrwal R, Srikant R. Fast algorithm for mining association rules. In: Proceedings of the 20th Very Large DataBases Conference. 1994. 487～499. http://www.almaden.ibm.com/cs/quest

4 Cheung D W, Lee S D, Kao B. A general incremental technique for updating discovered association rules. In: Proceedings of the 1997 International Conference on Databases Systems for Advanced Applications. 1997. http://www.cs.hku.hk/~dcheung

5 Feldman R, Aumann Y, Amir A *et al*. Efficient algorithms for discovering frequent sets in incremental databases. In: Proceedings of 1997 SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery. 1997. 59～66. http://www.cs.biu.ac.il/~feldman

A Time-Window Based Incremental Technique for Updating Association Rules

OU-YANG Wei-min[1]　CAI Qing-sheng[2]

[1](*Computer Center　Anhui University　Hefei* 230039)

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

Abstract　A time-window based incremental technique for updating association rules is presented in this paper, which can not only re-use the results acquired in the previous discovery process, but also focus the discovery on the recent data set using time window.

Key words　Knowledge discovery, association rule, incremental update, time-window.