智能系统学报
CAAI TRANSACTIONS ON INTELLIGENT SYSTEMS

基于同步频繁树的时间序列关联规则分析

李海林， 龙芳菊

引用本文:
李海林, 龙芳菊. 基于同步频繁树的时间序列关联规则分析[J]. 智能系统学报, 2021, 16(3): 502–510.
LI Hailin, LONG Fangju. Association rules analysis of time series based on synchronization frequent tree[J]. *CAAI Transactions on Intelligent Systems*, 2021, 16(3): 502–510.

在线阅读 View online: https://dx.doi.org/10.11992/tis.202008012

DOI: 10.11992/tis.202008012

基于同步频繁树的时间序列关联规则分析

李海林[1,2], 龙芳菊[1]

（1. 华侨大学 信息管理系, 福建 泉州 362021; 2. 华侨大学 现代应用统计与大数据研究中心, 福建 厦门 361021）

摘　要: 针对经典算法 Apriori 和频繁模式增长算法 (frequent pattern growth, FP-growth) 不能直接对时间序列数据进行关联规则挖掘的问题, 提出一种同步频繁树算法（synchronize frequent tree, SFT）。利用时间序列的时间属性具有一维性的特点, 定义趋势项−位置表示法表示时间序列数据, 将首条时间序列构建成一棵基础树, 通过计算树叶子节点与列表项的信息交集, 可判断其是否与该树枝中的所有节点构成频繁 K 项集。在 SFT 算法中, 用趋势项−位置表示的数据内存占用情况要优于原始数据, 并且在挖掘过程中不会产生候选频繁项集, 使得算法在整个挖掘过程中表现出较好的时间性能。基于商品数据和股票数据的数值实验表明, SFT 算法所得结果不仅与其他 5 种对比算法的结果一致, 在各量级的数据和不同的支持度计数中, 其时间复杂度都要优于对比算法。

关键词: 时间序列; 线性分段; 趋势项−位置; 事务集表示; 频繁项集; 同步频繁树; 关联规则; 时间效率

中图分类号: TP311.13　　**文献标志码:** A　　**文章编号:** 1673−4785(2021)03−0502−09

中文引用格式: 李海林, 龙芳菊. 基于同步频繁树的时间序列关联规则分析 [J]. 智能系统学报, 2021, 16(3): 502–510.

英文引用格式: LI Hailin, LONG Fangju. Association rules analysis of time series based on synchronization frequent tree[J]. CAAI transactions on intelligent systems, 2021, 16(3): 502–510.

Association rules analysis of time series based on synchronization frequent tree

LI Hailin[1,2], LONG Fangju[1]

(1. Department of Information Systems, Huaqiao University, Quanzhou 362021, China; 2. Research Center of Applied Statistics and Big Data, Huaqiao University, Xiamen 361021, China)

Abstract: In this paper, a synchronization frequent tree (SFT) algorithm is proposed to solve the problem that the classic algorithms apriori and FP-growth can not directly mine the association rules of time series data. By making use of the time attribute of time series, which has one-dimensional characteristics, we define the trend item-position representation method to represent the time series data, construct a basic tree for the first time series, and then find the information between the leaf nodes of the tree and the list items by intersection, and then judge whether the item and all the nodes in the branch constitute a frequent K itemsets. In the SFT algorithm, the memory occupancy of the data represented by the trend item-location is better than that of the original data, and candidate frequent itemsets will not be generated during the mining process, which makes the algorithm show better time performance in the entire mining process. Numerical experiments based on commodity data and stock data show that the results of the SFT algorithm are consistent with the results of the comparison algorithm, and what's more, in all levels of data, its time complexity is better than that of the comparison algorithm.

Keywords: time series; linear segmentation; trend item-location; transactionset representation; frequent itemsets; synchronize frequent trees; association rules; time efficiency

时间序列数据是指一系列时间及其对应属性值组成的序列集合, 常见于医学、金融、水文等领域[1]。通过分析这些数据, 如疾病[2]、股票[3-4] 和水文数据[5] 等, 研究者可以发现相关问题的潜在信息, 进而为相关部门或企业的工作提供指导性建议。关联规则是由 Agrawal 等[6] 首次提出的, 先找出频繁项集, 再通过项集的支持度和置信度等指标, 分析被研究对象间的关联关系。例如, 购物篮分析案例就是关联规则的一个经典应用。Apriori 算法是由 Agrawal 等[7] 提出的, 在挖掘频繁项集的过程中, 该算法不仅要多次扫描数据库, 还会产生大量的候选频繁项集, 因而导致算

收稿日期: 2020−08−12.

基金项目: 国家自然科学基金项目 (71771094, 61300139); 福建省自然科学基金项目（2019J01067）; 福建省社会科学规划一般项目（FJ2020B088）.

通信作者: 李海林. E-mail: hailin@hqu.edu.cn.

法的挖掘效率低。为解决这一问题, 很多学者从不同角度提出相应的方法。魏玲等[8] 借鉴文献 [9] 的 MapReduce 框架, 提出了基于 MapReduce 的 Apriori 改进算法 (MapReduce算法), 算法的基本思想是将频繁 $K-1$ 项集的前 $K-2$项作为键, 将最后一项作为值, 并将具有相同键的频繁 $K-1$ 项集合并, 以实现快速挖掘出候选频繁 K 项集。此外, 他们还提出性能更高的基于 Bigtable 与 MapReduce 的 Apriori 改进算法 (BM_Apriori 算法), 算法以事务集序号记录每个项出现的位置, 通过求频繁 $K-1$ 项集间的序号列表交集, 即可快速获取候选频繁 K 项集。Zhang[10]基于概率论知识, 通过参数 a 和 b 估算数据项集同时出现的概率, 进而确定频繁项集, 最终实现对 Apriori 算法的改进, 但是该算法存在频繁项集缺失的可能性。Tran 等[11] 为了减少 Apriori 算法扫描数据库的次数, 将事务集转化成事务矩阵, 但是在矩阵运算过程中需要消耗较长时间。杨秋翔等[12] 基于 1 和 0 表示数据项出现和未出现的数据集表示法创建权值向量矩阵, 提出可以在数据挖掘过程中不断缩减矩阵的算法 (WV_Apriori 算法), 以此达到提高挖掘频繁 K 项集速率的目的, 但是当数据量越来越大时, 创建的矩阵也会随之扩大, 进而降低挖掘速率。Han 等[13] 提出能快速挖掘频繁项集的 FP-growth 算法, 该算法将原数据存储到 FP-tree 中, 从中挖掘频繁项集, 从而不会产生候选频繁项集, 但是该算法不能给出频繁项集的支持度和置信度。此外, 上述算法不能直接用于时间序列数据的关联规则挖掘。Das 等[14] 于 1998 年首次提出挖掘时间序列数据的关联规则, 此后该研究成为数据挖掘领域的热门方向。Velumani 等[15] 在数据预处理阶段, 先将时间序列数据转为多个事务集, 再用 Apriori 算法挖掘关联规则。赵益[16] 提出了 Improved-Apriori 算法, 算法通过计算位置列表的方式可避免多次扫描数据库。然而这 2 个算法均是基于 Apriori, 它们都会产生大量的候选频繁项集, 会导致其挖掘效率不高。针对时间序列数据, 其他学者也做出了相应的努力。Das 等[14] 先将一条时间序列等分成多条子序列, 再挖掘多个时间序列的项间关联规则, 但是他并没有给出 3 支及以上股票的实验结果。Chen 等[17] 所提的 CEMiner 算法基于区间数据, 发现多个项间的闭合时态模式。Ruan 等[18] 提出一种可以在大规模区间型时态数据上并行和定量挖掘时间序列模式的算法。然而, 这 2 种方法不能给出频繁项集的支持度和置信度。由于树形结构分支明确, 直观易懂, 因而树形存储结构是一种较为常用的存储形式, 许多学者也将该存储结构应用到数据挖掘中。Schlüter等[19] 提出利用 2 种树结构挖掘时间序列的关联规则, Rashid 等[20] 基于树结构, 采用模式增长方法挖掘时态模式并给出关联规则, Pankaj 等[21] 也用到了 FP-tree结构。另外, 马慧等[22] 利用 FP-tree 的优势并考虑项间具有不同的有效时间, 提出一种基于 FP-tree 挖掘时态关联规则算法。

鉴于上述文献的理论研究及其所存在的问题, 本文针对多条时间序列数据, 通过数据降维并将符号化后的降维数据用趋势项−位置表示, 再利用树结构找出频繁 K 项集, 该过程通过求树的叶子节点与列表项间的信息交集, 便可判断该项是否与该树枝中的所有结点构成频繁 K 项集, 无需产生大量的候选频繁项集, 此外, 算法还能给出频繁项集的支持度和置信度。由于在某些情况下需要考虑多条时间序列在同时区内的关联关系, 例如: 在带有时间属性的零售商品销售数据中, 顾客常会同时购买 A 和 B 等商品, 若仅知道商品 A 的需求变化趋势但不知道商品 B 的变化趋势, 此时可以通过挖掘商品间的销售量变化趋势的关联规则, 进而预测商品 B 的需求变化趋势。因此, 在同时区内, 本文提出一种基于同步频繁树的时序数据关联规则算法 (synchronize frequent tree, SFT)。

1　时间序列特征表示

挖掘多条时间序列数据间的频繁项集, 需要先对原数据进行特征表示。本文定义了 2 种表示法, 即趋势项−位置表示法和事务集表示法。经典算法 Apriori、FP-growth 以及近些年提出的 MapReduce、BM_Apriori 和 WV_Apriori 算法都是基于事务集表示的数据, 而 SFT 算法则是基于趋势项−位置表示的数据。

1.1　事务集表示法和趋势项−位置表示法

Apriori、FP-growth、MapReduce、BM_Apriori 和 WV_Apriori 等算法只能挖掘事务集数据的频繁项集, 对于时间序列数据, 需要将其转换为事务集才可以使用。本文将时间序列转换为事务集的方法定义为事务集表示法, 其转换规则为: 多条时间序列数据在同时区内的趋势项组合表示一个事务。例如: $T_A=(a_1,a_2,a_3)$、$T_B=(b_1,b_2,b_3)$ 和 $T_C=(c_1,c_2,c_3)$ 是 3 条符号化后的时间序列, 其转换为事务集的结果为 $[(a_1,b_1,c_1),(a_2,b_2,c_2),(a_3,b_3,c_3)]$。

趋势项−位置表示法是为了提出 SFT 算法而定义的一种时间序列转换方法, 其关键在于考虑到时间序列数据的时间属性具有一维性的特点, 表示规则为趋势项+位置列表, 其中趋势项是由时间序列进行线性分段后的斜率确定, 共分上

升、平稳和下降 3 种，用 q_1、q_2、q_3 表示，q 表示时间序列名。例如：T_A=(a_2,a_2,a_1,a_3,a_1,a_3,a_1,a_1) 是一条符号化后的时间序列数据，将其用趋势项−位置表示为 list_A=[{a_1:(2,4,6,7)},{a_2:(0,1)},{a_3:(3,5)}]，其中，{a_2:(0,1)}表示趋势项 a_2 在第 0 和第 1 个时区内出现。

1.2　性能比较与分析

显然事务集表示法并没有减少原数据量，而趋势项−位置表示法只保留每个趋势项，相同趋势项则用位置索引代替。由于特征表示数据是各类算法挖掘工作的基础，其对算法运行效率具有很大影响，因此有必要从转换时效和转换后数据的内存占用情况对上述 2 种表示法的性能进行比较和分析。实验采用 python 程序将后文使用的 3 条股票时间序列数据分别进行事务集表示和趋势项−位置表示，每条数据量为 5 343，共 16 029 个数据。实验结果如表 1 所示。

表 1　2 种表示法的性能比较

Table 1　Performance comparison of two representations

性能	事务集表示法	趋势项−位置表示法
转换时效/s	0.038	0.039
内存占用/B	16 560	96

从转换时效方面，2 种表示法处理的数据量相同，因而差异性不大；从内存占用方面，由趋势项−位置表示的数据要远低于事务集表示的数据，因为趋势项−位置表示法以数值型 (int) 数据记录趋势项的变化趋势，而事务集表示法则以字符型 (str) 数据记录，由于 int 占用的内存要低于 str 占用的内存，因此用前者表示的数据，内存占用情况要优于后者。

2　同步频繁树

鉴于经典算法 Apriori 和 FP-growth 不能直接对时间序列数据进行关联规则挖掘，提出了一种可以解决上述问题的新算法，即 SFT 算法。新算法通过求叶子节点与列表项间的信息交集，便可判断该项是否与该树枝中的所有节点构成频繁项集。算法总体思路：先将时间序列用趋势项−位置表示，并去除非频繁项，再用首条时间序列构建一棵基础树，通过求叶子节点与列表项间的信息交集，便可以在树的生长过程中不断挖掘出频繁 K 项集。通过给出频繁 K 项集所有前缀项的信息量，便可以计算出频繁 K 项集的支持度与置信度。新方法除了适用于多条数时间序列数据外，在小数据和大数据中，其都能取得较优的时间效率，此外还能给出频繁项集的支持度和置信度。

X 表示叶子节点所在的列表，x 为树的叶子节点，也是列表的部分项，即 X 中的项需要满足某些条件后才能成为树的叶子节点。y_i 表示列表 Y 的项。loc_list 表示叶子节点信息表与列表项位置表的信息交集，其信息量用 loc_count 表示，data 表示仅包含频繁项的数据集，min_supc 表示最小支持度计数。

算法　SFT 算法

输入　data, min_supc;

输出　频繁 K 项集。

1) 以 Root 为根，首条时间序列的所有项为 x_t，构建一棵基础树；

2) 让所有叶子结点 x_t 与时间序列 X 后的所有 Y 序列进行匹配计算；

3) 对 x_t 与 y_i 求 loc_list，并判断 loc_count 是否不小于 min_supc：

①是，将 loc_list 作为 y_i 的节点信息表，y_i 作为叶子节点，即 x_i。x_i 所在的树枝节点构成频繁项集，频繁项数即为 x_i 所在的树层，项集个数即为 loc_count。判断 Y 是否为最后一条时间序列：

是，输出频繁 K 项集；

否，输出频繁 K 项集，重复 2)、3)；

② 否，y_i 不是叶子节点，舍弃。

3　实例分析

设最小支持度计数 min_supc 为 2，用趋势项−位置表示的时间序列数据集 data=[[{a_1: (0, 2, 3, 5, 7, 9), a_2: (4, 6, 8), a_3: (1)}],[{b_1: (2, 3), b_2: (0, 4, 5, 6, 8, 9), b_3: (1,7)}],[{ c_1: (3, 4, 6), c_2: (2, 7), c_3: (0, 1, 5, 8, 9) }]], data 可视化结果如图 1 所示，{b_1: (2, 3)}表示趋势项 b_1 在第 2 和第 3 个时区内出现过。在构建同步频繁树前，需要先去除 data 中的非频繁项。

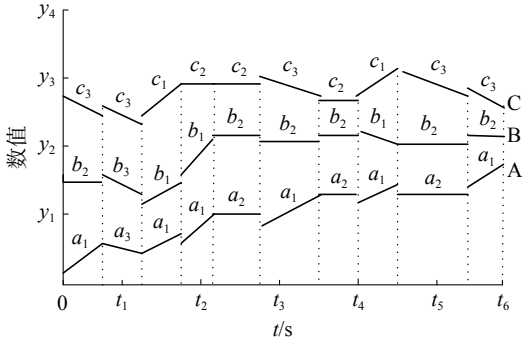


图 1　实例数据可视化

Fig. 1　Visualization of an instance data

在 data 中，由于 a_3 的数量仅为 1，小于 2，因此去掉此非频繁项。根据 SFT 算法的挖掘步骤，利用首条时间序列 [{a_1: (0, 2, 3, 5, 7, 9), a_2: (4, 6, 8) }] 中的 a_1 和 a_2 项构建一棵基础树，如图 2 所示。

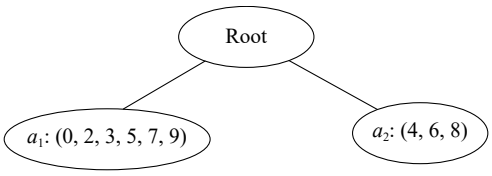


图 2　基础树
Fig. 2　Base tree

叶子节点 a_1、a_2 分别与时间序列 B 和 C 中的项进行匹配计算, 即 [{b_1: (2, 3), b_2: (0, 4, 5, 6, 8, 9), b_3: (1,7) }] 和 [{ c_1: (3, 4, 6), c_2: (2, 7), c_3: (0, 1, 5, 8, 9) }], 结果如图 3 所示。例如 a_1 与 b_2 的 loc_list 为 (0,5,9), 即 loc_count 为 3, 因而 b_2 成为叶子节点, 其信息表为 (0,5,9), 信息量为 3, 由于 b_2 位于第 2 层树, 则 a_1b_2 是频繁 2 项集, 项集个数为 3。而 a_1 与 b_3 的 loc_list 为 (7), 即 loc_count 小于 2, 因此 b_3 不是叶子节点。

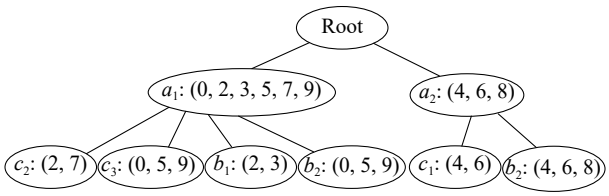


图 3　同步频繁树的生长
Fig. 3　Growth process of synchronize frequent tree

由于 C 是最后一条时间序列, 因此输出图 3 中的所有频繁 2 项集: a_1c_2、a_1c_3、a_2c_1, 其项集个数分别为 2、3、2, 但 B 并非最后一条时间序列, 先输出频繁 2 项集 a_1b_1、a_1b_2、a_2c_1、a_2b_2, 再以同样的思路将叶子节点 b_1、b_2、b_1 与时间序列 C 中的项进行匹配计算, 结果如图 4 所示, 由于叶子结点是最后一条时间序列中的项, 因此一棵完整的同步频繁树构建完成, 并输出所有的频繁 3 项集, 即 $a_1b_2c_3$、$a_2b_2c_1$。

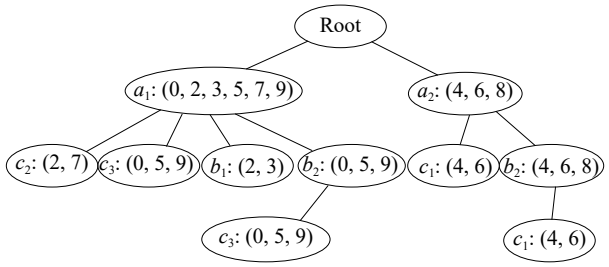


图 4　完整的同步频繁树
Fig. 4　Complete synchronized frequent tree

4　数值实验

为了验证 SFT 算法具有普适性, 使用零售商品数据和股票数据分别进行实验。分别与基于时间序列事务集的 Apriori[7]、FP-growth[14]、MapReduce[9-10]、BM_Apriori[9] 以及 WV_Apriori[13] 算法的挖掘结果进行比较和分析, 以检验 SFT 算法的有效性和性能。

4.1　数据介绍

零售商品数据是从 UCI 网站下载的 Online_Retail_II, 取其中代号为 20 725、20 727 和 20 728 的销售量。每条数据的时间在 2010 年 12 月 2 日−2011 年 12 月 9 日, 共 225 天, 675 个数据量。股票数据是从预测者网中获取, 冠城大通股票 (600067)、中船科技股票 (600072) 和上汽集团股票 (600104) 的日收盘价, 每支股票的时间在 1997 年 12 月 25 日−2021 年 3 月 4 日, 共 5 343 个工作日, 16 029 个数据量。为了获取多条时间序列间的变化趋势规则, 需要先将每条时间序列分割成多个趋势项, 然后再借鉴文献 [23] 的对齐思想将多条时间序列的趋势项对齐。例如, 在时区 [0, 3] 内, 时间序列 A 的趋势项是 a_1, 而时间序列 B 在 [0,1] 和 [1, 3] 区间内的趋势项为 b_1 和 b_2, 为了与 B 对齐, 序列 A 的 [0, 3] 时区被分成 [0,1] 和 [1, 3], 对应的趋势项是 a_1、a_1。本文使用的分割策略为基于滑动窗口的线性回归, 分割所得线段可以保留原数据的局部变化趋势。以上汽集团股票数据为例, 图 5 是它的原始变化趋势, 图 6 展示的是时间在 2020 年 8 月 14 日−2020 年 9 月 24 日该支股票的数据分割过程, 蓝色表示原始时间序列, 红色表示分割后的局部拟合线段, 表 2 则给出部分具体的原始数据。

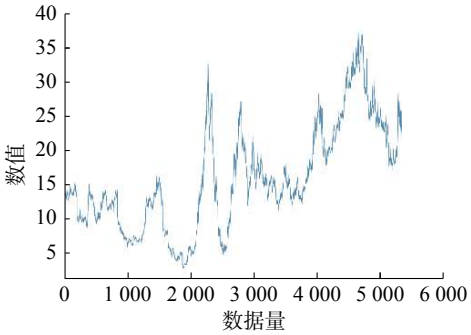


图 5　上汽集团历年的股票数据
Fig. 5　Stock data of Shangqi group over the years

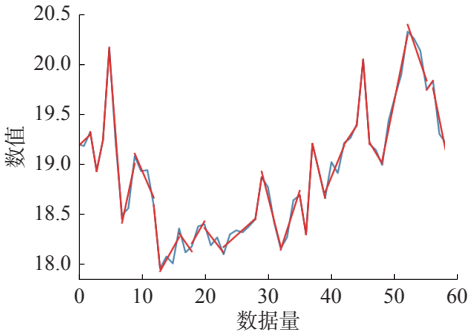


图 6　时间序列线性分段 (上汽集团股票)
Fig. 6　Linear segmentation of time series(Shangqi group)

表 2　原始数据 (上汽集团股票)

Table 2　Raw data(Shangqi group)

日期	收盘价/元	日期	收盘价/元
2020-09-01	19.02	2020-09-14	19.43
2020-09-02	18.91	2020-09-15	19.67
2020-09-03	19.21	2020-09-16	19.89
2020-09-04	19.26	2020-09-17	20.32
2020-09-07	19.40	2020-09-18	20.25
2020-09-08	20.04	2020-09-21	20.13
2020-09-09	19.20	2020-09-22	19.74
2020-09-10	19.14	2020-09-23	19.83
2020-09-11	18.99	2020-09-24	19.30

4.2　时序数据关联分析

本次实验共分为 2 组，第 1 组使用商品销售数据集，第 2 组使用股票数据集。实验采用 SFT 算法，除了与经典算法 Apriori 和 FP-growth 进行比较外，实验还给出了近年来提出并且具有较好性能的 MapReduce、BM_Apriori 和 WV_Apriori 算法的挖掘结果进行比较。由于近年来从时间序列角度研究商品销售关联性成为热门方向[24-25]，因此本文将给出详细的商品销售数据挖掘结果，并用股票数据的部分结果说明新方法具有普适性。第 1 组实验结果如图 7 所示，图 7 中展示了基于不同的 min_supc，实验算法和对比算法的频繁 2、3 项集的个数。实验表明，无论 min_supc 取什么值，这些算法的挖掘结果都是相同的，进而说明 SFT 方法能够取得同样精度的效果。

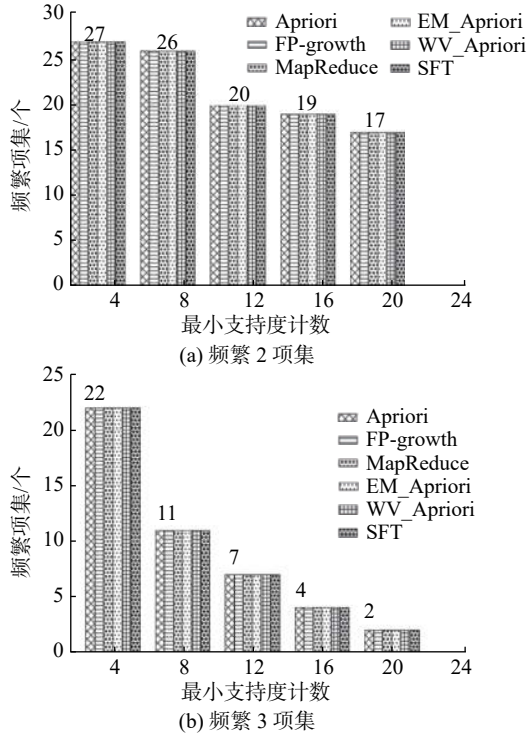
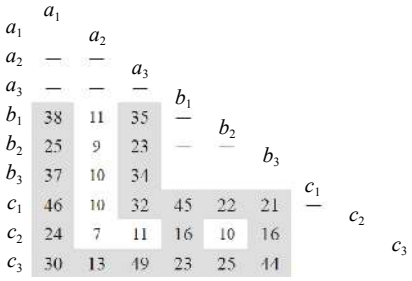


图 7　频繁项集个数对比 (商品数据)

Fig. 7　Comparison of frequent itemsets(commodity data)

为说明实验结果的真实可靠性，图 8 给出了当 min_supc =12 时，频繁 2 项集的详细数据。其中，阴影部分的数字表示满足 min_supc 的频繁项集个数，例如频繁项集 a_1b_1 共有 38 个，非阴影部分的数字表示不满足 min_supc 的项集个数，‘—’则表示该频繁项集不存在，因为本文挖掘的是不同时间序列间的关联关系，所以 a_1a_2 等不是要找的频繁 2 项集。



	a_1	a_2	a_3	b_1	b_2	b_3	c_1	c_2	c_3
a_1									
a_2	—								
a_3	—	—							
b_1	38	11	35						
b_2	25	9	23	—					
b_3	37	10	34	—	—				
c_1	46	10	32	45	22	21			
c_2	24	7	11	16	10	16	—		
c_3	30	13	19	23	25	44	—	—	

图 8　频繁 2 项集及其个数

Fig. 8　Frequent 2 itemsets and their number

鉴于 FP-growth 算法不能给出频繁项集的支持度和置信度，而 MapReduce、BM_Apriori 和 WV_Apriori 是为挖掘频繁项集而提出的算法，故本文仅给出 SFT 和 Apriori 算法所挖掘频繁项集的支持度和置信度。当 min_supc=12 时，2 个算法均给出如表 3 所示的结果，其中 Rule 表示规则，Sup 表示支持度，Conf 表示置信度。以 $a_1b_1 \Rightarrow c_1$ 为例，其表明增加购买 20 725 商品、20 727 商品和 20 728 商品的规则数占总规则数的 0.09；当都增加购买 20 725 和 20 727 时，20 728 的购买量会增加的概率为 0.57。

表 3　SFT 与 Apriori 算法挖掘关联规则

Table 3　Association rules mined by SFT and Apriori

项集	Rule	Sup	Conf	Rule	Sup	Conf
频繁2项集	$a_1 \Rightarrow b_1$	0.17	0.38	$a_1 \Rightarrow b_2$	0.11	0.25
	$a_1 \Rightarrow b_3$	0.16	0.37	$a_1 \Rightarrow c_1$	0.20	0.46
	$a_1 \Rightarrow c_2$	0.10	0.24	$a_1 \Rightarrow c_3$	0.13	0.3
	$a_2 \Rightarrow c_3$	0.05	0.43	$a_3 \Rightarrow b_1$	0.15	0.38
	$a_3 \Rightarrow b_2$	0.10	0.25	$a_3 \Rightarrow b_3$	0.15	0.36
	$a_3 \Rightarrow c_1$	0.14	0.34	$a_3 \Rightarrow c_3$	0.22	0.53
	$b_3 \Rightarrow c_1$	0.20	0.53	$b_1 \Rightarrow c_2$	0.07	0.19
	$b_1 \Rightarrow c_3$	0.10	0.27	$b_2 \Rightarrow c_1$	0.09	0.37
	$b_2 \Rightarrow c_3$	0.11	0.44	$b_3 \Rightarrow c_1$	0.09	0.25
	$b_3 \Rightarrow c_2$	0.07	0.19	$b_3 \Rightarrow c_3$	0.19	0.54
频繁3项集	$a_1\, b_1 \Rightarrow c_1$	0.09	0.57	$a_1 \Rightarrow b_1\, c_1$	0.09	0.22
	$a_1\, b_2 \Rightarrow c_1$	0.06	0.6	$a_1 \Rightarrow b_2\, c_1$	0.06	0.15
	$a_1\, b_3 \Rightarrow c_3$	0.07	0.45	$a_1\, b_3 \Rightarrow c_3$	0.07	0.17
	$a_3\, b_1 \Rightarrow c_1$	0.08	0.54	$a_3 \Rightarrow b_1 c_1$	0.08	0.20
	$a_3\, b_1 \Rightarrow c_3$	0.05	0.34	$a_3 \Rightarrow b_1\, c_3$	0.05	0.13
	$a_3\, b_2 \Rightarrow c_3$	0.06	0.60	$a_3 \Rightarrow b_2\, c_3$	0.06	0.15
	$a_3\, b_3 \Rightarrow c_3$	0.10	0.67	$a_3 \Rightarrow b_3\, c_3$	0.10	0.25

为了说明 SFT 算法具有普适性, 第 2 组实验使用股票数据。由于挖掘步骤与第 1 组实验相同, 因此图 9 直接给出挖掘结果, 其表明 6 种算法的挖掘结果相同, 因而说明 SFT 算法具有较强的普适性。

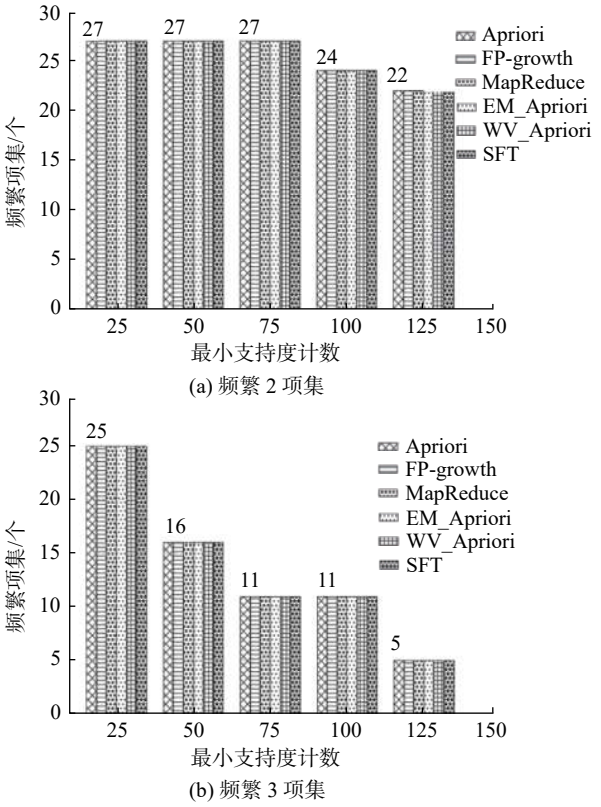


图 9　频繁项集个数对比（股票数据）

Fig. 9　Comparison of frequent itemsets (stock data)

4.3　时间效率分析

时间效率是衡量算法好坏的重要指标之一。由于挖掘频繁项集所耗时间占整个挖掘过程的多数时间, 因此图 10 给出 6 种算法在挖掘商品数据和股票数据频繁项集时所消耗的时间可视化图, 表 4 是具体的数值结果。实验表明, 无论在数据量为 675 的商品数据, 还是在数据量为 16 029 的股票数据, SFT 算法的时间效率在不同最小支持计数的取值下都要优于其他算法, 因而说明 SFT 算法具有较强的稳健性。

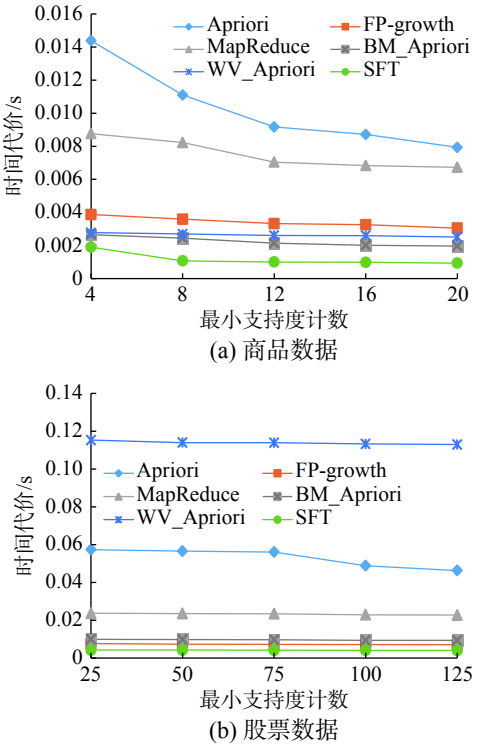


图 10　时间复杂度对比

Fig. 10　Comparison of time complexity-commodity data

表 4　6 种算法挖掘频繁项集的时耗

Table 4　Time consumption of six algorithms for mining frequent itemsets　　s

数据集	Min_sup	Apriori	FP-growth	MapReduce	BM_Apriori	WV_Apriori	SFT
商品数据	4	0.014 386	0.003 870	0.008 764	0.002 653	0.002 778	0.001 897
	8	0.011 101	0.003 590	0.008 223	0.002 444	0.002 693	0.001 069
	12	0.009 170	0.003 321	0.007 040	0.002 141	0.002 609	0.001 000
	16	0.008 713	0.003 251	0.006 831	0.002 012	0.002 598	0.000 982
	20	0.007 933	0.003 049	0.006 371	0.001 961	0.002 505	0.000 933
股票数据	25	0.057 053	0.007 691	0.024 362	0.009 790	0.115 257	0.004 307
	50	0.056 445	0.007 544	0.023 703	0.009 662	0.113 917	0.004 247
	75	0.055 805	0.007 434	0.023 542	0.009 632	0.113 889	0.004 138
	100	0.048 232	0.007 375	0.023 023	0.009 398	0.113 221	0.004 035
	125	0.043 928	0.007 243	0.022 827	0.009 105	0.112 937	0.003 988

由图 10 和表 4 的定量结果分析可知, SFT 算法要优于其他 5 种对比算法, 因而有必要从定性角度进一步分析实验结果。本文认为影响 SFT 算法取得较好性能的主要因素有: 1) 在生成频繁

K 项集的过程中，SFT 算法不会产生候选频繁项集；2) SFT 算法只需要对叶子节点和列表趋势项求一次信息交集，即可快速判断出频繁 K 项集；3) 由 1.2 节分析得出，由趋势项−位置表示的数据，其所占内存要远低于事务集表示的数据，因而导致在程序运行过程中，SFT 算法的处理速度要快于其他算法。

限于篇幅有限并且考虑到各类因素对不同算法的影响程度不同，因而对于每个对比算法，仅给出必要的分析和解释：1) WV_Apriori 算法在数据量较少的商品数据中，具有较好的时间效率，而在数据量较大的股票数据中，其时间效率明显降低，表现出很差的稳健性。导致这种结果的原因：①在创建 0-1 矩阵的过程中，每项表示一列，每个事务表示一行，当数据量越来越大时，其创建的矩阵会越来越大，因而将会消耗更多的时间；②由于频繁项集的挖掘是基于 0-1 矩阵，当矩阵很大时，除了遍历矩阵需要较多的时间外，矩阵占用较多内存也会影响程序的运行速率；③ 由于 WV_Apriori 算法只需要遍历一次数据库，并且在挖掘更高的频繁项集时，不断缩小矩阵，因此当数据量较小时，其效率要高于 Apriori、FP-growth 和 MapReduce 算法，正因为该算法需要遍历完整个矩阵后才能判断项间是否构成频繁项集，因此其运行速率不及 BM_Apriori 和 SFT 算法。2) 由于 BM_Apriori 算法直接求 2 个频繁 K−1 项之间的序号列表交集，即可快速找出它们之间的所有项集个数，但是该算法的基础数据用事务集表示，因而在数据量较少的商品数据中，其时间效率要优于除了 SFT 算法以外的其他算法，然而在数据量较大的股票数据中，该算法会产生候选频繁 K 项集，因而其时间效率略低于 FP-growth 算法。3) MapReduce 算法通过合并具有相同键的频繁 K−1 项集，即可快速产生候选频繁 K 项集，因而其挖掘速率要快于 Apriori 算法，但是正因它还会产生较多的候选项集，因此导致其运行效率不及余下的其他算法。4) FP-growth 算法只需要扫描 2 次数据库，并通过头指针可以快速找到其他树枝上的相同项，因而在数据量较大的股票数据中，其挖掘速率要优于除了 SFT 以外的其他算法，但是由于其挖掘的基础数据用事务集表示，此外，在重新对事务集进行排序、构建 FP-tree、提取条件模式基以及构建条件 FP-tree 过程中，消耗了较多时间，因此导致其运行速率不及 SFT 算法。5) 当最大频繁项集是 K 时，Apriori 算法需要扫描 K 次数据库，并从大量候选频繁项集中挖掘出频繁 K 项集，因而导致其具有较低的挖掘速率。

表 5 分别从 6 个层面，概括总结 6 种算法的区别与联系：除了 SFT 算法外，其他算法都是基于用事务集表示的数据；SFT 和 BM_Apriori 算法分别用项的位置和事务的序号代替原数据，而其他算法仅仅是通过不同的方法表示保留原数据；SFT 算法与 FP-growth 和 WV_Apriori 算法一样不会产生候选频繁项集；仅有 SFT 算法和 BM_Apriori 算法可以直接判断频繁 K 项集，而其他算法则需要遍历完整个数据库才能判断；SFT 和 Apriori 算法能给出频繁项集的支持度与置信度，而其他算法不能；SFT 算法与 FP-growth、Mapreduce 和 BM_Apriori 算法都能应用于大数据中。

表 5 6 种算法对比

Table 5 Comparison of six algorithms

对比项	Apriori	FP-growth	Mapreduce	BM_Apriori	WV_Apriori	SFT
基础数据表示	事务集	事务集	事务集	事务集	事务集	趋势项-位置
挖掘基础	$C_{(k-1)}$	条件模式基	$C_{(k-1)}$	事务的序号	布尔矩阵	项的位置
候选频繁项集	有	无	有	有	无	无
直接判断频繁K项	否	否	否	是	否	是
有无sup,conf	有	无	无	无	无	有
适用大数据	否	是	是	是	否	是

显然，无论在定量分析还是定性分析中，SFT 算法都要优于其他算法。

5 结束语

针对经典算法 Apriori 和 FP-growth 不适用于时间序列数据，提出了一种挖掘时间序列数据关联规则的新方法，即 SFT 算法，并给出了近些年来提出的，且具有较好性能的 MapReduce、BM_Apriori 和 WV_Apriori 算法挖掘结果作对比。SFT 算法的总体思路是先将时间序列数据用趋势项−位置表示，再通过求叶子节点与列表项间的信息交集，进而可以快速挖掘出频繁 K 项集。

本文具的创新点: 1) 新定义的事务集表示法可以将多条时间序列数据转换成事务集, 该转换法让仅适用于挖掘无序数据的关联规则算法也可以挖掘时间序列数据, 此外还定义了趋势项−位置表示法, 用该方法表示的数据, 其内存占用要低于原数据所占的内存, 而内存占用情况会影响算法运行速率; 2) SFT 算法可以挖掘多条时间序列数据间的关联规则, 弥补了 Apriori、FP-growth 等算法不适用于时间序列数据的缺点; 3) 通过计算树叶子节点与列表项间的信息交集, 便可快速判断频繁 K 项集, 并且不会产生候选频繁项集, 因而提高了算法的挖掘速率; 4) SFT 算法充分利用时间序列具有一维性特点, 并采用了直观易懂的树型结构, 这为时间序列数据关联分析的研究提供了一种新的研究思路和视角。实验表明: 1) 基于趋势项−位置表示法表示的时间序列数据, 其内存占用要远低于用事务集表示的时间序列数据; 2) 利用 SFT 算法挖掘出的频繁项集与经典算法 Apriori、FP-growth 以及近些年来所提的, 并且具有较好性能的 MapReduce、BM_Apriori 和 WV_Apriori 算法的挖掘结果相同; 3) 无论在大数据还是小数据中, SFT 算法的时间效率都要优于对比算法; 4) 对于关联规则, SFT 算法给出的结果与 Apriori 的结果相同。

SFT 算法具有较高较优的时间性能, 使其有较强的普适性。然而, SFT 算法只考虑到同时区内不同时间序列间的关联关系, 挖掘在不同时区内不同时间序列间的关联关系, 将是进一步需要研究的工作。

参考文献:

[1] 陈海燕, 刘晨晖, 孙博. 时间序列数据挖掘的相似性度量综述 [J]. 控制与决策, 2017, 32(1): 1–11.
CHEN Haiyan, LIU Chenhui, SUN Bo. Survey on similarity measurement of time series data mining[J]. Control and decision, 2017, 32(1): 1–11.

[2] ACHEBAK H, DEVOLDER D, BALLESTER J. Trends in temperature-related age-specific and sex-specific mortality from cardiovascular diseases in Spain: a national time-series analysis[J]. The lancet planetary health, 2019, 3(7): e297–e306.

[3] 李海林, 梁叶. 基于关键形态特征的多元时间序列降维方法 [J]. 控制与决策, 2020, 35(3): 629–636.
LI Hailin, LIANG Ye. Dimension reduction for multivariate time series based on crucial shape features[J]. Control and decision, 2020, 35(3): 629–636.

[4] 程小林, 郑兴, 李旭伟. 基于概率后缀树的股票时间序列预测方法研究 [J]. 四川大学学报, 2018, 55(1): 61–66.
CHENG Xiaolin, ZHENG Xing, LI Xuwei. Research of stock time based on probabilistic suffix tree[J]. Journal of Sichuan University, 2018, 55(1): 61–66.

[5] 王 玲, 徐培培, 彭开香. 基于因子模型和动态规划的多元时间序列分段方法 [J]. 控制与决策, 2020, 35(1): 35–44.
WANG Ling, XU Peipei, PENG Kaixiang. Segmentation of multivariate time series with factor model and dynamic programming[J]. Control and decision, 2020, 35(1): 35–44.

[6] AGRAWAL R, IMIELIŃSKI T, SWAMI A. Mining association rules between sets of items in large databases[C]// Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data. Washington, USA, 1993: 207–216.

[7] AGRAWAL R, SRIKANT R. Mining sequential patterns[C]//Proceedings of the 11th International Conference on Data Engineering. Taipei, China, 1995: 3–14.

[8] 魏玲, 魏永江, 高长元. 基于 Bigtable 与 MapReduce 的 Apriori 算法改进 [J]. 计算机科学, 2015, 42(10): 208–210, 243.
WEI Lin, WEI Yongjiang, GAO Changyuan. Improved Apriori algorithm based on bigtable and MapReduce[J]. Computer science, 2015, 42(10): 208–210, 243.

[9] KARIM R, HOSSAIN A, RASHID M, et al. A MapReduce framework for mining maximal contiguous frequent patterns in large DNA sequence datasets[J]. IETE technical review, 2012, 29(2): 162–168.

[10] ZHANG Xiaolu. Pythagorean fuzzy clustering analysis: a hierarchical clustering algorithm with the ratio index-based ranking methods[J]. International journal of intelligent systems, 2018,33(9): 1798–1822.

[11] TRAN T N, DRAB K, DASZYKOWSKI M. Revised DBSCAN algorithm to cluster data with dense adjacent clusters[J]. Chemometrics and intelligent laboratory systems. 2013,120(15): 92−96.

[12] 杨秋翔, 孙涵. 基于权值向量矩阵约简的 Apriori 算法 [J]. 计算机工程与设计, 2018, 39(3): 690–693,762.
YANG Qiuxiang, SUN Han. Apriori algorithm based on weight vector matrix reduction[J]. Computer engineering and design, 2018, 39(3): 690–693,762.

[13] HAN Jiawei, PEI Jian, YIN Yiwen. Mining frequent patterns without candidate generation[J]. ACM sigmod record, 2000, 29(2): 1–12.

[14] DAS G, LIN K I, MANNILA H, et al. Rule discovery from time series[C]//Proceedings of the 4th International Conference on Knowledge Discovery and Data Mining. New York, USA, 1998: 16−22.

[15] VELUMANI B, UMAJOTHY P. Mining temporal association rules from time series microarray using apriori al-

gorithm[J]. Review of bioinformatics and biometrics, 2013, 2(2): 29−36.

[16] 赵益. 多时间序列上时序关联规则的挖掘 [D]. 上海: 东华大学, 2018.
ZHAO Yi. Discovery of tempopal assocition rules in multivariate time series[D], Shang Hai: Donghua University, 2018.

[17] CHEN Yicheng, PENG W C, LEE S Y. CEMiner—An efficient algorithm for mining closed patterns from time interval-based data[C]//Proceedings of the IEEE 11th International Conference on Data Mining. Vancouver, Canada, 2011: 121−130.

[18] RUAN Guangchen, ZHANG Hui, PLALE B. Parallel and quantitative sequential pattern mining for large-scale interval-based temporal data[C]//Proceedings of 2014 IEEE International Conference on Big Data (Big Data). Washington, USA, 2014: 32−39.

[19] SCHLÜTER T, CONRAD S. Mining several kinds of temporal association rules enhanced by tree structures[C]// Proceedings of the 2nd International Conference on Information, Process, and Knowledge Management. Saint Maarten, Netherland Antilles, 2010: 86−93.

[20] RASHID M M, GONDAL I, KAMRUZZAMAN J. Mining associated patterns from wireless sensor networks[J]. IEEE transactions on computers, 2015, 64(7): 1998−2011.

[21] PANKAJ G, SAGAR B B. Discovering weighted calendar-based temporal relationship rules using frequent pattern tree[J]. Indian journal of science and technology, 2016, 9(28): 1–6.

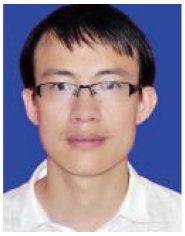
[22] 马慧, 汤庸, 潘炎. 一种基于 FP-树的时态关联规则的分区挖掘方法 [J]. 计算机工程, 2006, 32(17): 132−134.
MA Hui, TANG Yong, PAN Yan. A FP-tree based partition mining approach to discovering temporal association rules[J]. Computer engineering, 2006, 32(17): 132−134.

[23] 张建业, 潘泉, 张鹏等. 基于斜率表示的时间序列相似性度量方法 [J]. 模式识别与人工智能, 2007, 20(2): 271−274.
ZHANG Jianye, PAN Quan, ZHANG Peng, et al. Similarity measuring method in time series based on slope[J]. Pattern recognition and artificial intelligence, 2007, 20(2): 271−274.

[24] SALEM M Z. Effects of perfume packaging on Basque female consumers purchase decision in Spain[J]. Management decision, 2018, 56(8): 1748−1768.

[25] LI Hailin, WU Y J, CHEN Yewang. Time is money: dynamic-model-based time series data-mining for correlation analysis of commodity sales[J]. Journal of computational and applied mathematics, 2020, 370: 112659.

作者简介：



李海林，教授，博士生导师，主要研究方向为数据挖掘与决策支持。主持国家自然科学基金项目 2 项、省部级基金项目 4 项。发表学术论文 60 余篇。

龙芳菊，硕士研究生，主要研究方向为数据挖掘与企业管理。

[责任编辑：李雪莲]