基于注意力机制的高效关联规则算法

管博伦,张立平,董 伟,李闰枚,朱静波,孔娟娟,汪 焱
安徽省农业科学院 农业经济与信息研究所,合肥 230001

摘 要:目的 针对事务间相似度高且频繁出现的数据集,在频繁项集挖掘过程中产生的大量冗余,分析部分已有关联规则算法在挖掘频繁项集方面的不足,提出一种使用注意力机制进行剪枝的关联规则挖掘算法。**方法** 该算法结合垂直数据格式和注意力机制的优点,依次求出频繁 k 项集中的注意力权重,利用注意力权重过滤掉交集计算中的冗余,生成精简频繁项集,最后将精简频繁项集和注意力权重进行合并,得到频繁项集。**结果** 实验结果表明:在 Data 数据集上该算法比 Eclat、Apriori、FP-Growth 的时间最大提升 94.6%、73%、95.5%,比 FP-Growth 最多节省空间 61.991MB;在 Accident 数据集上,该算法比 Eclat、Apriori、FP-Growth 的时间最大提升 93.4%、69%、85.7%,比 FP-Growth 最多节省空间 58.786 MB。**结论** 通过引入注意力机制增强了算法的泛化能力和稳定性,减少了挖掘频繁项集产生的冗余,提高了算法速度,降低了挖掘过程中内存的开销,尤其是在 AFI 指数较大、支持度较低的数据集上,该算法在时间和空间开销上更具备优势。

关键词:关联规则;频繁项集;冗余;注意力机制;精简频繁项集
中图分类号:TP311.13 **文献标识码:**A doi:10.16055/j.issn.1672-058X.2026.0003.012

An Efficient Association Rule Algorithm Based on Attention Mechanism

GUAN Bolun, ZHANG Liping, DONG Wei, LI Runmei, ZHU Jingbo, KONG Juanjuan, WANG Yan
Institute of Agricultural Economics and Information, Anhui Academy of Agricultural Sciences, Hefei 230001, China

Abstract: Objective Aiming at the large amount of redundancy generated during the mining process of frequent item sets in datasets with high similarity and frequent occurrence among transactions, this paper analyzes the deficiencies of some existing association rule algorithms in mining frequent item sets and proposes an association rule mining algorithm that uses an attention mechanism for pruning. **Methods** This algorithm combined the advantages of the vertical data format and the attention mechanism. It sequentially calculated the attention weights in frequent k-item sets, used these weights to filter out the redundancy in intersection calculations, generated simplified frequent item sets, and finally merged the simplified frequent item sets with the attention weights to obtain the frequent item sets. **Results** The experimental results showed that on the Data dataset, the proposed algorithm improved time efficiency by up to 94.6%, 73.0%, and 95.5% compared with Eclat, Apriori, and FP-Growth, respectively; it also saved up to 61.991 MB of space compared with FP-Growth. On the Accident dataset, it achieved maximal time efficiency improvements of 93.4%, 69.0%, and 85.7% over the same three baselines, respectively, with a maximum space saving of 58.786 MB against FP-Growth. **Conclusion** By introducing the attention mechanism, the generalization ability and stability of the algorithm are enhanced, the redundancy generated during frequent item set mining is reduced, the algorithm speed is improved, and the memory overhead during

收稿日期:2024-05-09 **修回日期:**2024-09-20 **文章编号:**1672-058X(2026)03-0099-09
基金项目:国家自然基金面上项目资助(32171888).
作者简介:管博伦(1993—),男,安徽蚌埠人,硕士,助理研究员,从事农业信息化和计算机视觉研究.
通信作者:董伟(1983—),男,安徽淮南人,硕士,副研究员,从事农业信息化研究. Email:dw06@163.com.
引用格式:管博伦,张立平,董伟,等.基于注意力机制的高效关联规则算法[J].重庆工商大学学报(自然科学版),2026,43(3):99-107.
Guan Bolun, Zhang Liping, Dong Wei, et al. An efficient association rule algorithm based on attention mechanism[J]. Journal of Chongqing Technology and Business University (Natural Science Edition), 2026, 43(3): 99-107.

the mining process is lowered. Especially for datasets with a large AFI index and low support, this algorithm has more advantages in terms of time and space consumption.

Keywords: association rule; frequent item set; redundancy; attention mechanism; simplified frequent item set

随着信息技术的发展,计算机技术、通信技术、网络技术正改变着人类的社会生活。现在,人类生活在一个信息爆发式增长的时代,这些信息是随机的,包含大量噪声,又蕴含着一定的规律,它们涉及社会的各个领域[1]。从大量信息中挖掘出所需要的有价值信息成为重点。知识发现(简称 KDD)又叫做数据挖掘,受到广泛的关注[2]。

关联规则算法是数据挖掘中的一个重点分支,其主要任务是在大量数据中发现背后隐藏的人们感兴趣的有规律数据[3]。关联规则最早是由 Agrawal 与 Rehman 等[4-5]提出的 Apriori 算法,用于挖掘数据集中事务之间的相互关系。算法分为两步,第一步是设置最小支持度,挖掘出事务数据集中大于最小支持度的频繁项集;第二步是设置最小置信度,挖掘出频繁项集中大于最小置信度的强关联规则。算法中最关键的第一步是挖掘满足最小支持度的频繁项集[6],这个过程需要反复扫描数据库,消耗了大量的时间。Apriori 算法思想简单,操作简单,没有复杂的推导过程,但是其缺点也很明显。由于算法要反复扫描数据库,导致该算法运行时间过长,效率低下[7],其次实验者往往需要根据先验知识来调整不同的支持度阈值[8]。改进挖掘频繁项集方法是算法的关键[9]。

针对传统算法的不足之处,很多专家学者对其进行了改进[10]。Mudumba 等[11]提出一种关于多个独立数据源的关联规则方法,用于发现跨分布的频繁模式;Mokkadem 等[12]采用递归方法来挖掘频繁项集,在项集的递归性方面体现了优点,能高效地应用于数据库中添加或者删除事务方面;Bui-Thi 等[13]基于多个关联规则组建立了一种基于关联规则的可解释分类模型,提高了模型规则数量和预测精度。基于 Apriori 的改进算法,需要多次扫描数据库,生成频繁项集候选集,这是这类算法的瓶颈。为了降低生成频繁项集候选集的时间开销,Han 等[14]提出一种不需要产生频繁项集候选集的 FP-growth 算法,该算法利用树结构,由下而上构造频繁项集,加快了运算速度。随着硬件设备的升级,厍向阳等[15]提出使用 Hadoop 平台改进 FP-growth,在 Hadoop 平台下采用负载均衡和数据碎片化计算方法,再次提高了算法的运行速度,利用并行运算克服了 Apriori 算法串行运算速度慢的瓶颈;毛伊敏等[16]提出使用信息熵和遗传算法来合并相似数据项,动态获取最优支持度阈值,解决了 Can 树占用空间过大,无法动态选择支持度阈值的问题;Mahdi 等[17]设计了一种新的 FR-Tree 算法来挖掘数据库中具有高置信度的罕见关联规则。但是基于树的算法在建立 FP 树或者 Can 树时也需要花费额外的时间开销。

Savasere 等[18]提出的 Partition 算法是一种基于垂直数据格式的算法,并使用交集来计算项集的支持度;Zaki[19]提出的 Eclat 算法采用等价类垂直格式的关联挖掘,但在面对含有大量事物列表的数据集时,同样会占用大量的内存空间;Zaki 等[20]改进了 Eclat 算法,提出 Diffsets 是一种对稠密数据的纵向表示法,该算法巧妙利用垂直数据项交集产生的差集来产生频繁项集,进一步减少了内存空间。垂直格式挖掘算法在对数据集进行交集操作时,可以忽略掉非频繁项集,按照交集操作生成的候选集都是频繁项集[21]。数据库系统和开发工具对并集操作都有着良好的支持,底层封装好的算法,让这种并集操作速度更快[22]。使用交集挖掘可以直接挖掘出数据项所在数据集中的位置,更有利于挖掘流式数据项背后隐含的模式。在基于垂直数据分布的关联规则算法中,支持度计算和候选项集的产生方式与 Apriori 算法相似,只是在发现候选集的过程中就开始计数。利用垂直数据分布来挖掘频繁项集比 Apriori 水平格式挖掘算法快几个数量级,但是面对不均匀数据集时,会导致算法效率严重下降。

综上所述,基于 Apriori 算法的关联规则操作简单,没有复杂的过程,但是该类型的算法要多次扫描数据库,挖掘频繁项集,增加了时间开销。基于 FP-Growth 算法的关联规则利用树结构优势,减少了扫描数据库的次数,但是在挖掘支持度较低的频繁项集时,构建树的时间会更长。基于垂直格式的关联规则算法通过事务间的并集操作提高了计算效率,但是在面对事务间相似度高的不均匀数据集时,会产生大量冗余,影响算法的性能。

本文针对上述关联规则挖掘算法运算过程中产生大量中间结果集降低算法效率的缺点,详细分析了垂直数据格式的优点,在此基础上,定义了注意力权重数组,通过注意力机制对频繁项集进行剪枝操作、动态加权,减少中间结果集,降低候选频繁项集的并集和扫描操作次数,提高了算法挖掘效率。最后的实验结果表明,该算法在时间性能方面比上述关联算法有较大提高。

1　相关工作

1.1　概念定义

定义 1　设数据集 $D=\{i_1,i_2,i_3,\cdots,i_n\}$ 是所有数据项的集合,n 为自然数。事务 T 是由一系列具有相同唯一标识 TID 的数据项组成的项集,其中每一个事务 T 都是数据集 D 上的一个子集,即 $T\subseteq D$。事务 T 可以表示为 $T=\{t_1,t_2,t_3,\cdots,t_m\}$,$m\leqslant n$,$m$ 为自然数,表示在事务 T 中所包含的数据项数。如果事务 T 中包含 k 个项集,则称之为 k-项集。关联规则可以表示为 $X\Rightarrow Y$ 的逻辑蕴含关系,其中 $X\subseteq I$,$Y\subseteq I$,且 $X\cap Y=\varnothing$。

定义 2　支持度。关联规则 $X\Rightarrow Y$ 的支持度 Supprot($X\Rightarrow Y$)表示在数据集 D 中,同时出现事务 X 和事务 Y 的百分比。最小支持度(Minsup)由用户自己设定,计算公式为

$$\text{Support}(X\Rightarrow Y)=P(X\cup Y) \tag{1}$$

其中,$P(X\cup Y)$为事务 X 和事务 Y 在数据集 D 中同时出现的概率。

定义 3　置信度。关联规则 $X\Rightarrow Y$ 的置信度 Confidence($X\Rightarrow Y$)表示在数据集 D 中,同时出现事务 X 和事务 Y 的数量与包含事务 X 的数量的百分比,既发生在 X 上又发生 Y 的条件概率。最小置信度(Minconf),由用户自己设定,公式为

$$\text{Confidence}(X\Rightarrow Y)=\frac{\text{Support}(X\cup Y)}{\text{Support}(X)} \tag{2}$$

定义 4　强关联规则。关联规则 $X\Rightarrow Y$ 满足其支持度大于最小支持度 Minsup,且其置信度大于最小置信度 Minconf。其中最小支持度和最小置信度由用户根据实际情况或者经验给定。

定义 5　频繁项集。数据项的频度为该数据项在整个数据集 D 中出现的次数。项集的频度为包含该项集的事物在整个数据集 D 中出现的次数。如果项集 L 的支持度满足最小支持度阈值,置信度满足最小置信度阈值,则称项集 L 为频繁项集,频繁 k-项集记作 L_k。

挖掘关联规则的一般步骤可以理解为给定一个数据集 D,挖掘在数据集中满足最小支持度阈值和最小置信度阈值的过程,可分为两个过程[23]:挖掘满足最小支持度的所有频繁项集,在所有频繁项集中挖掘出满足最小置信度的强关联规则。在以上两个过程中,最关键的是第一部分,它直接关系到算法的挖掘效率[24]。

1.2　垂直数据分布关联规则算法

水平分布是数据集中常见的一种格式,由很多事务 T 组成,对应唯一的事务标识 T_{ID},每个事务又有若干数据项构成。垂直分布的数据可定义为数据集 D 由一系列的数据项构成,每个数据项又由若干个出现在相应事务中的事务标识 T_{ID} 组成,即包括该数据项所有的 T_{ID}。将所有事务标识组成的列表称为 Tidset。

水平分布的数据格式按照事务 T_{ID} 来记录,符合人们日常产生数据的规律,垂直分布的数据格式按照 Tidset 来记录,更方便于数据的处理。通过不同的 Tidset 进行交集运算便可以计算频繁项集的支持度。当把垂直分布的每个数据项进行并集处理时,会产生许多由共同前缀组成的数据项集。为了描述和计算方便,将它们划分为不同的等价类,看做不同的子问题来处理[25]。下面给出等价类的定义:

$$S_a=[a]=\{b[k-1]\in L_1\mid a[1:k-2]=b[1:k-2]\} \tag{3}$$

其中,$a[1:k-2]=b[1:k-2]$且 $a[1:k-2]<b[1:k-2]$。例如令 $L=\{AB,AC,AD,AE,BC,BD,CD,CE\}$,其可以产生的等价类 $S_A=[A]=\{B,C,D,E\}$,$S_B=[B]=\{C,D\}$,$S_C=[C]=\{D,E\}$,由上述 S_A、S_B、S_C 3 个等价类相互连接生成的集合为 $\{AB,AC,AD,AE\}$、$\{BC,BD\}$、$\{CD,CE\}$,这 3 个项集相互独立。使用等价类可以将搜索空间划分为独立的子空间,在实际操作中将带有相同前缀的等价类看成一个独立的子问题,不同的前缀等价类看成每个新问题来解决。

垂直数据分布关联规则的挖掘过程充分利用了垂直数据格式的优势,使用交集来计算数据项集的支持度。同时该算法利用等价类的概念将大问题转化为几个独立的子问题,分别计算各个部分子问题的频繁项集,最后通过合并得到整个数据集的全局频繁项集。由于交集计算性能好,垂直数据分布的关联规则挖掘比 Apriori 算法运算速度更快[26]。但该算法在面对相似度较高的事务大量、频繁出现的非均匀分布数据集时,等价类计算的中间过程会产生大量频繁项集,这些中间过程频繁项集会带来较高的计算冗余,增加挖掘时间开销,并且这些局部频繁项集会占用过多的内存空间,导致算法性能下降。

2　注意力机制关联规则算法

2.1　相关定义

通过大量实验可以发现,面对相似度较高的不均匀分布数据集,垂直数据分布关联规则算法会在计算子问题频繁项集时产生部分 Tidset 相同但数据项集不重复的事务。我们把这种类型的记录称为 Tidset 重现(Reappear Tidset)记录,简称 reTidset。为了解决上述问题,这里对垂直数据分布的关联规则算法进行改进,提出注意力机制关联规则挖掘(Attention Mechanism Association Rule Mining),简称 AM_ARM 算法。在 AM_ARM 算法中,为了

减少处理 reTidset 产生的冗余计算时间,定义了注意力权重。下面给出相关概念和定义。

定义 6 注意力权重(Attention Mechanism Weighted)$(X_1,X_2,\cdots,X_j)=[1,2,3,\cdots,n]$表示在数据集中出现了 reTidset 的现象,数据项 $X_1,X_2,\cdots,X_j$ 的 Tidset 相等,权重值为该数组的长度。

定义 7 注意力聚焦指数(Attention Focus Index),简称 AFI,表示在某个支持度下,数据集中注意力能够聚焦的程度。公式如下:

$$I_{\mathrm{AF}}=\frac{1}{n}\sum_{i=1}^{n}\left(\frac{F_i-f_i}{F_i}+\cdots+\frac{F_n-f_n}{F_n}\right) \tag{4}$$

其中,F_i 是原始频繁项集的长度,f_i 是经过注意力权重裁剪之后的精简频繁项集长度,n 是频繁 n 项集。由式(4)可知注意力聚焦指数表示某一个数据集中注意力的聚焦程度,AFI 值越大,注意力集中的程度越大。AFI 的取值范围是[0,1),当 AFI 为 0 时表明在该数据集中注意力没有任何集中。

2.2 算法描述

一种基于注意力机制关联规则挖掘算法的算法描述如下所述:

(1) Input 数据集 D, Minsup
(2) Output Frequen k-item
(3) For itemset in D
(4) If itemset is reTidset
(5) AMweight(itemsets)
(6) If count(itemset) ≥ Minsup
(7) Itemset join in L_1
(8) Intersation(T_i)
(9) For $X_i \in L_k, j<i$
(10) Begin
(11) If itemset is reTidset
(12) AMweight(itemsets)
(13) Tidset(R) = Tidset(X_i) ∩ Tidset(X_j)
(14) $R=X_i \cup X_j$
(15) If count(R) ≥ Minsup
(16) Tidset(R) join in L_{k+1}
(17) $T_i=T_i \cup R$, T_i 初始值为空
(18) If ($T_i \neq \varnothing$)
(19) Intersation(T_i)
(20) Else end

在垂直格式数据集基础上对数据集进行扫描,在扫描过程中利用最小支持度阈值进行剪枝,得到由满足最小支持度的数据项集形成的频繁 1-项集。

在频繁 1-项集的基础上计算注意力权重,得到权值为 2 的注意力权重。利用权重对数据项进行剪枝,在频繁 1-项集中删除冗余项集,得到精简频繁 1-项集。对精简频繁 1-项集中的数据项进行交集合并,在合并过程中删除支持度小于最小支持度阈值的 2-项集,利用最小支持度进行剪枝,得到频繁 2-项集。对频繁 2-项集进行注意力权重计算,得到权值为 3 的注意力权重。利用权重对数据项进行剪枝,删除频繁 2-项集中的冗余频繁项集,得到精简频繁 2-项集。

继续循环挖掘,直至该精简频繁 k-项集的个数为 1 或者精简频繁 k-项集交集后得到的频繁$(k+1)$-项集个数为 0,循环终止。按照等价类的计算方法,将加权数组按照权值由小到大的顺序,结合精简频繁 k-项集进行展开,最终得到频繁 k-项集。

3 实验与结果分析

3.1 数据集

实验选用的是 Data、Accident 和 Groceries 数据集,其中 Data 数据集事务记录有 200 条,数据项种类 26 种,Accident 数据集事务记录有 150 条,Groceries 数据集事务记录有 10 075 条。在 Data 和 Accident 数据集中相似度较高的事务记录大量出现。Data 数据集如表 1 所示,由于数据集记录较多,这里只展示前 10 条。

表 1 Data 数据集

Table 1 Data dataset

ID	记 录
1	A, B, C, D, F, H
2	A, E, H, K, L, M, N, O, Y
3	C, D, M, N, P, S, T
4	B, C, D, E, G, I, J, K, M
5	B, C, D, E, F, G, I, J, K, M, N, O, R, W, Z
6	A, C, D, E, F, I, J, K, L, N, Q, P, X, Y
7	A, D, E, F, I, J, K, L, N, P, R, S, T
8	B, C, D, E, F, G, H, I, J, K, M, N, O, P, Q, R, S, T, U, V
9	A, B, C, E, F, O, Q
10	A, B, C, G, L, M, N, S, X, Z

3.2 实验结果

3.2.1 实验环境与参数设置

实验所用的环境是 Windows10 企业版 64 位操作系统,处理器型号为英特尔酷睿 i7-7700,频率为 3.60 GHz,内存存储容量为 16 GB。该实验对比使用垂直数据格式关联规则 Eclat、Apriori、FP-Growth 和 AM_ARM 算法在 Data、Accident 和 Groceries 数据集上不同支持度下的时间开销和内存开销。

3 个数据集在不同支持度下频繁 1-项集、频繁 2-项集和频繁 3-项集的数量变化如表 2 所示。

表 2　支持度分析

Table 2　Support analysis

数据集	支持度	频繁 1-项集	频繁 2-项集	频繁 3-项集
Data	0.1	16	71	76
	0.2	13	13	1
	0.3	6	3	0
	0.4	5	0	0
	0.5	3	0	0
Accident	0.1	70	1 509	16 516
	0.2	53	862	7 258
	0.3	40	562	4 080
	0.4	33	403	2 365
	0.5	32	289	1 233
	0.6	24	168	549
	0.7	18	93	240
	0.8	13	47	85
	0.9	6	14	16
Groceries	0.01	86	216	30
	0.02	60	58	2
	0.03	43	18	0
	0.04	32	8	0
	0.05	29	3	0
	0.06	20	1	0
	0.07	18	1	0
	0.08	13	0	0
	0.09	10	0	0

通过表 2 可以看到:Data 数据集的频繁项集长度在支持度范围为 0.01~0.1 之间显示出随 Minsup 的变化而变化的趋势更明显;Accident 数据集在 0.3 到 0.8 支持度范围上,频繁项集较平滑;Groceries 数据集在 0.01 到 0.02 支持度范围上,频繁项集较明显。表 3 展示了 Eclat 算法分别在 Data 数据集、Accident 数据集和 Groceries 数据集上的频繁 1-项集、频繁 2-项集和频繁 3-项集的数量。其中 Data 数据集的 Minsup 范围是[0.01,0.1],Accident 数据集的 Minsup 范围是[0.3,0.75],Grocerise 数据集 Minsup 的范围是[0.01,0.019]。

表 3　Eclat 不同支持度下的频繁项集

Table 3　Frequent itemsets of Eclat under different support levels

数据集	支持度	频繁 1-项集	频繁 2-项集	频繁 3-项集
Data	0.01	26	314	2 053
	0.02	26	278	1 466
	0.03	26	245	1 030
	0.04	25	197	637
	0.05	23	168	448
	0.06	23	125	273
	0.07	22	93	139
	0.08	22	84	111
	0.09	17	65	72
	0.1	16	51	50

续表(表3)

数据集	支持度	频繁 1-项集	频繁 2-项集	频繁 3-项集
Accident	0.3	40	562	4 080
	0.35	36	467	3 069
	0.4	33	403	2 365
	0.45	32	345	1 713
	0.5	32	289	1 233
	0.55	26	216	824
	0.6	24	168	549
	0.65	20	117	340
	0.7	18	93	240
	0.75	13	61	138
Groceries	0.01	88	216	30
	0.011	80	176	21
	0.012	76	155	18
	0.013	76	137	11
	0.014	74	119	9
	0.015	72	101	7
	0.016	70	92	4
	0.017	69	87	3
	0.018	65	76	2
	0.019	61	70	2

3.2.2　结果分析

在 Groceries 数据集中,AM_ARM 挖掘产生的 AFI 指数为 0,产生的频繁项集个数与 Eclat 算法产生的个数相同,这里就不再列表说明。表 4 和表 5 分别为使用 AM_ARM 算法在 Data 数据集和 Accident 数据集上挖掘的结果。在 Data 数据集中,当 Minsup 为 0.01 时,精简比重最大,AM_ARM 计算产生的精简频繁 2-项集的数量为 194,比频繁 2-项集的数量减少 120 个,精简频繁 3-项集的数量为 360,比频繁 3-项集的数量减少 1 693 个,最大精简占比约为 82.5%;当 Minsup 为 0.08 时,精简比重最小,精简频繁 3-项集的数量为 93,比频繁 3-项集的数量减少 18 个,最小精简占比约为 16.2%。在 Accident 数据集中,当 Minsup 为 0.35 时,精简比重最大,AM_ARM 计算产生的精简频繁 2-项集数量为 210,比频繁 2-项集的数量减少 257 个,精简频繁 3-项集的数量为 848 个,比频繁 3-项集的数量减少 2 221 个,最大精简占比约为 72.4%;当 Minsup 为 0.65 时,精简比重最小,AM_ARM 计算产生的精简频繁 2-项集数量为 70,比频繁 2-项集的数量减少 47,精简频繁 3-项集的数量为 130,比频繁 3-项集的数量减少 210,最小精简占比约为 61.8%。表 3 和表 4 中的数据表明,算法使用注意力权重对 reTidset 事务进行剪枝,是有效果的。

表 6、表 7、表 8 分别是 Eclat、Apriori、FP-Growth 和 AM_ARM 4 种算法分别在 3 个数据集上不同支持度下挖掘频繁项集的时间开销和空间开销。

表4 AM_ARM不同支持度频繁项集(Data)

Table 4 Frequent itemsets of AM_ARM under different support levels (Data)

支持度	频繁1-项集数量	加权数组数量(w=2)	精简频繁2-项集的数量	加权数组数量(w=3)	精简频繁3-项集数量	频繁3-项集数量	AFI指数
0.01	26	10	194	136	360	2 053	0.603
0.02	26	9	189	121	358	1 466	0.538
0.03	26	8	182	98	318	1 030	0.474
0.04	25	5	157	67	271	637	0.388
0.05	23	4	139	53	231	448	0.328
0.06	23	1	124	40	179	273	0.176
0.07	22	0	93	18	110	139	0.104
0.08	22	0	84	12	93	111	0.081
0.09	17	0	65	9	59	72	0.090
0.1	16	0	51	9	38	50	0.120

表5 AM_ARM不同支持度频繁项集(Accident)

Table 5 Frequent itemsets of AM_ARM under different support levels (Accident)

支持度	频繁1-项集数量	加权数组数量(w=2)	精简频繁2-项集的数量	加权数组数量(w=3)	精简频繁3-项集数量	频繁3-项集数量	AFI指数
0.3	40	12	308	83	1 211	4 080	0.577
0.35	36	15	210	54	848	3 069	0.636
0.4	33	14	181	37	690	2 365	0.629
0.45	32	14	156	24	516	1 713	0.622
0.5	32	14	133	19	363	1 233	0.622
0.55	26	10	100	9	240	824	0.622
0.6	24	8	79	8	154	549	0.624
0.65	20	5	70	11	130	340	0.509
0.7	18	5	52	3	93	240	0.526
0.75	13	5	32	6	53	138	0.545

表6 不同算法在Data数据集上的开销

Table 6 Cost of different algorithms on the Data dataset

算法	开销项	Minsup									
		0.01	0.02	0.03	0.04	0.05	0.06	0.07	0.08	0.09	00.1
Eclat	时间开销/s	1.169	0.679	0.393	0.191	0.118	0.057	0.041	0.021	0.021	0.024
Apriori	时间开销/s	0.234	0.190	0.172	0.109	0.080	0.077	0.063	0.048	0.052	0.034
FP-Growth	时间开销/s	1.415	1.008	0.160	0.051	0.026	0.012	0.009	0.006	0.003	0.003
AM_ARM	**时间开销/s**	**0.063**	**0.056**	**0.053**	**0.040**	**0.030**	**0.029**	**0.024**	**0.024**	**0.011**	**0.019**
Eclat	空间开销/MB	28.364	28.262	27.852	27.238	27.033	27.033	26.521	26.726	26.419	26.521
Apriori	空间开销/MB	32.768	31.949	30.822	30.003	29.389	28.744	28.672	28.569	28.262	28.262
FP-Growth	空间开销/MB	89.277	41.740	29.375	18.250	15.726	14.578	14.066	14.011	13.832	13.761
AM_ARM	**空间开销/MB**	**27.136**	**27.136**	**27.033**	**27.033**	**26.931**	**26.828**	**26.726**	**26.624**	**26.828**	**26.316**

表7 不同算法在Accident数据集上的开销

Table 7 Cost of different algorithms on the Accident dataset

算法	开销项	Minsup									
		0.3	0.35	0.4	0.45	0.5	0.55	0.6	0.65	0.7	0.75
Eclat	时间开销/s	7.002	3.848	2.294	1.345	0.776	0.487	0.230	0.146	0.079	0.082
Apriori	时间开销/s	1.022	0.818	0.657	0.552	0.423	0.318	0.225	0.161	0.130	0.100
FP-Growth	时间开销/s	8.030	1.767	0.684	0.307	0.150	0.071	0.050	0.020	0.013	0.006
AM_ARM	**时间开销/s**	**0.667**	**0.253**	**0.225**	**0.141**	**0.091**	**0.062**	**0.034**	**0.058**	**0.021**	**0.012**
Eclat	空间开销/MB	47.104	45.772	41.984	39.936	37.888	35.840	32.768	30.720	29.696	28.672
Apriori	空间开销/MB	30.310	29.593	29.491	29.286	28.979	28.569	28.262	28.262	28.057	28.057
FP-Growth	空间开销/MB	95.240	71.576	64.512	37.376	24.268	19.558	16.384	15.360	14.336	14.336
AM_ARM	**空间开销/MB**	**36.454**	**36.454**	**35.430**	**34.611**	**33.484**	**31.846**	**29.696**	**29.696**	**28.672**	**27.648**

表8 不同算法在Groceries数据集上的开销

Table 8 Cost of different algorithms on the Groceries dataset

算法	开销项	Minsup									
		0.01	0.011	0.012	0.013	0.014	0.015	0.016	0.017	0.018	0.019
Eclat	时间开销/s	2.151	2.100	2.153	2.052	2.074	2.050	2.172	2.001	2.029	2.003
Apriori	时间开销/s	9.32	7.75	7.17	7.01	6.57	6.33	5.59	5.45	5.34	4.77
FP-Growth	时间开销/s	0.37	0.31	0.34	0.31	0.29	0.30	0.27	0.29	0.26	0.27
AM_ARM	**时间开销/s**	**2.13**	**2.06**	**2.04**	**2.01**	**2.06**	**2.03**	**2.05**	**2.00**	**1.98**	**1.96**
Eclat	空间开销/MB	34.48	34.28	33.95	33.37	33.42	33.12	32.86	32.82	32.52	32.43
Apriori	空间开销/MB	30.05	29.71	29.66	29.90	29.78	29.76	29.82	29.71	29.68	29.61
FP-Growth	空间开销/MB	28.95	28.82	28.73	28.51	28.47	28.26	27.92	27.87	27.86	27.78
AM_ARM	**空间开销/MB**	**33.91**	**33.54**	**33.26**	**33.12**	**33.01**	**32.88**	**32.86**	**32.81**	**32.50**	**32.50**

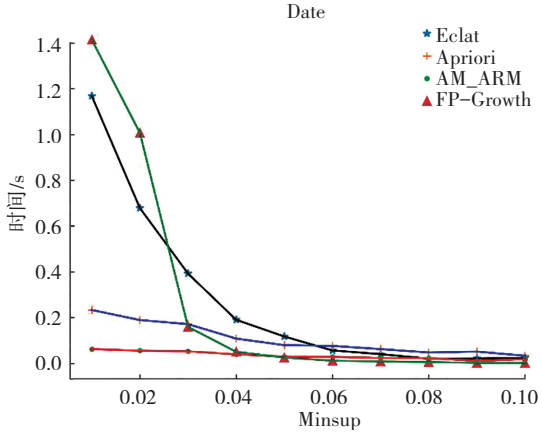
由表 6 可以看出,在 Data 数据集中,当 Minsup 为 0.01 时, AM_ARM 挖掘频繁项集的时间开销为 0.063 s,超越其他 3 种算法的比重最多,分别比 Eclat、Apriori、FP-Growth 的时间快 94.6%、73%、95.5%。在 Minsup 整个区间上,AM_ARM 算法的速度均比 Eclat 和 Apriori 的速度快,在 Minsup ∈ [0.01,0.04] 区间上,AM_ARM 算法的速度比 FP-Growth 的速度要快,其中当 Minsup 为 0.01 时,AM_ARM 的速度超越 FP-Growth 的速度最多,达到了 95.5%。AM_ARM 的空间开销基本与 Eclat 算法持平,当 Minsup 为 0.01 时,比 FP-Growth 节省空间最多,达 61.991 MB。

由表 7 可以看出,在 Accident 数据集中,当 Minsup 为 0.35 时,AM_ARM 挖掘频繁项集的时间为 0.253 s,超越其他 3 种算法的比重最多,分别比 Eclat、Apriori、FP-Growth 的时间快 93.4%、69%、85.7%。在 Minsup 整个区间上,AM_ARM 算法的速度均比 Eclat 和 Apriori 的速度快。在 Minsup ∈ [0.3,0.6] 区间上,AM_ARM 算法的速度比 FP-Growth 的速度要快,其中当 Minsup 为 0.3 时,AM_ARM 的速度超越 FP-Growth 的速度最多,达到了 91.7%。AM_ARM 的空间开销较低,当 Minsup 为 0.3 时,比 FP-Growth 节省空间最多,达 58.786 MB。

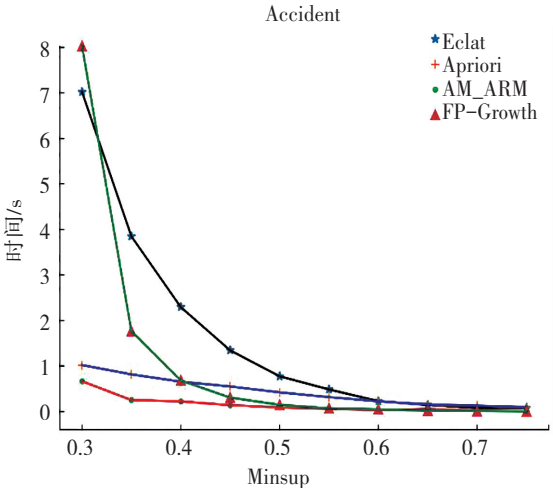
由表 8 可以看出,在 Groceries 数据集中,AM_ARM 算法的时间开销在整个 Minsup 区间上都比 Apriori 的速度快,与 Eclat 算法的时间开销持平,其内存开销与 Eclat 算法持平。

AM_ARM 算法在从 Data 数据集、Accident 数据集和 Groceries 中挖掘频繁项集时,均取得了较好的效果,在不同的数据集上其时间开销和空间开销具有比较优良的稳定性。尤其是当 Minsup 较小,且 AFI 指数较大时,其所用的时间开销和内存开销更具优势。

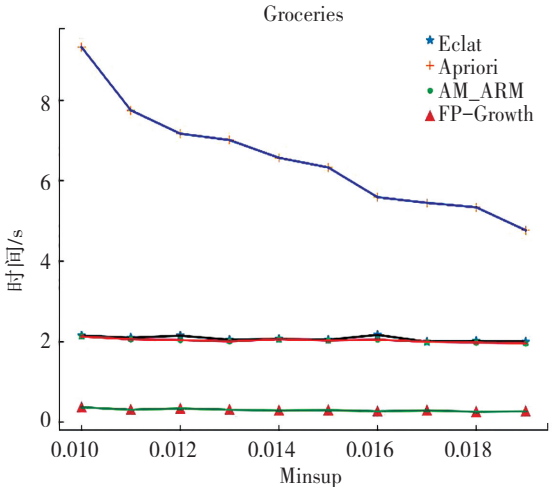
图 1 为 Eclat、Apriori、FP-Growth 和 AM_ARM 4 种算法在 Data、Accident 和 Groceries 3 个数据集中不同支持度下挖掘频繁项集产生的时间开销拟合走势图。可以看到,相比其他 3 种算法,当 AFI 指数较大,且 Minsup 较小时,AM_ARM 更具有优势,其速度更快。



(a) Data 数据集



(b) Accident 数据集

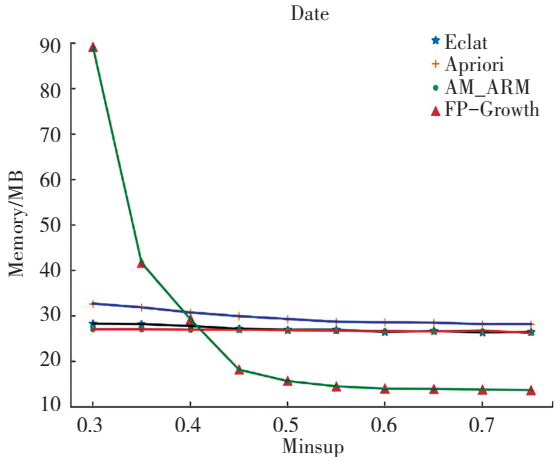


(c) Groceries 数据集

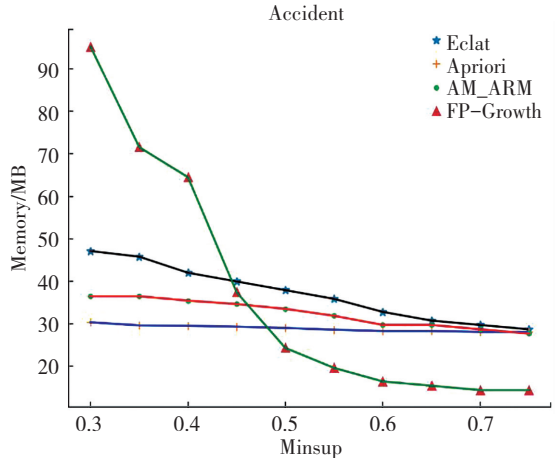
图 1　时间开销

Fig. 1　Time cost

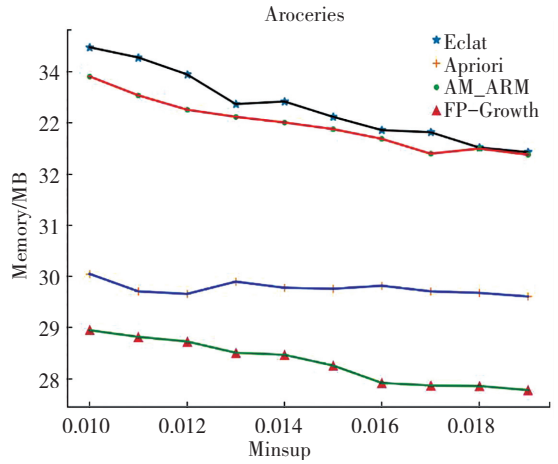
图 2 为 Eclat、Apriori、FP-Growth 和 AM_ARM 4 种算法在 Data、Accident 和 Groceries 3 个数据集中不同支持度下挖掘频繁项集产生的内存开销拟合走势图。可以看到 AM_ARM 算法并没有因为引入加权数组而增大其内存开销,由于其采用了精简频繁项集反而相较于 Eclat 算法减少了内存开销。在 Data 数据集和 Accident 数据集中,当频繁项集长度较长且 AFI 数较高时,AM_ARM 算法表现出更少的内存开销。FP-Growth 算法由于需要构建 FP 树,在频繁项集长度长且 AFI 指数高的数据集中,表现出更多的内存开销。Apriori 算法通过反复扫描事务项集,不需要引入额外的内存,在 3 种数据集中,表现出稳定的内存开销。



(a) Data 数据集



(b) Accident 数据集



(c) Groceries 数据集

图 2 内存开销

Fig. 2 Memory cost

由上述实验可见,当产生的频繁项集较多,AFI 指数较高时,AM_ARM 算法的时间开销要远小于 Eclat 算法,同时内存开销也低于 Eclat 算法。当支持度较大时,两种算法的时间开销和内存开销相近。对于事务数据集,在支持度较小的情况下,数据集 AFI 指数较大时,频繁项集和精简频繁项集的长度下降迅速,AM_ARM 算法的时间开销更低。挖掘频繁项集过程中的冗余较大,事务间相似度较高,且出现频率较高,产生了 reTidset 现象,使用 AM_ARM 算法挖掘精简频繁项集的时间开销和内存开销更低。面对一定的支持度,数据集对应的 AFI 指标相对较小时,数据间冗余较小,频繁项集和精简频繁项集的长度下降缓慢,加权数组的长度为 0,此时 AM_ARM 算法也能表现出较快的挖掘速度,占用较小的内存空间。

4 结 论

提出一种基于注意力机制的关联规则挖掘算法(AM_ARM)。该算法以垂直数据分布关联规则为基础,通过对产生 reTidset 现象的原因进行分析,进而提出使用加权数组对频繁 k-项集进行剪枝,挖掘精简频繁项集,减少算法搜索空间,减少算法的时间开销。通过实验可以看到,当数据集中事务的出现频率较高,不同事务间相似度较高时,AM_ARM 算法节省的时间开销更多,运行效率更高,对在不清楚数据分布情况下进行关联规则挖掘具有一定的应用价值。

本文提出的 AM_ARM 算法面对不同的数据集,在时间开销和空间开销上具有比较优良的稳定性。但是同时可以看到,AM_ARM 算法的内存空间开销仍偏大。因此下一步针对 AM_ARM 算法的内存开销进行深入研究,在降低其时间开销的同时,降低内存空间。

参考文献(References):

[1] FAN M, YANG G. Association rule mining algorithm for privacy protection[J]. Journal of Information Security Research, 2021, 7(11): 1007-1016.

[2] RAJ S, RAMESH D, SREENU M, et al. EAFIM: efficient apriori-based frequent itemset mining algorithm on Spark for big transactional data[J]. Knowledge and Information Systems, 2020, 62(9): 3565-3583.

[3] WANG W, LI Q, ZHU F. Association rules combined fuzzy decision quality control technology in intelligent manufacturing[J]. Intelligent Systems with Applications, 2024, 21: 200331.

[4] 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. New York: ACM, 1993: 207-216.

[5] REHMAN S U, ALNAZZAWI N, ASHRAF J, et al. Efficient top-K identical frequent itemsets mining without support threshold parameter from transactional datasets produced by IoT-based

smart shopping carts[J]. Sensors, 2022, 22(20): 8063.

[6] ALMASI M, SANIEE ABADEH M. CARs-Lands: an associative classifier for large-scale datasets[J]. Pattern Recognition, 2020, 100: 107128.

[7] NASR M, HAMDY M, HEGAZY D, et al. An efficient algorithm for unique class association rule mining[J]. Expert Systems with Applications, 2021, 164: 113978.

[8] WANG X P, LIN J X, WU J W, et al. Adaptive-association-rule mining algorithm based on determination coefficient [J]. CAAI Transactions on Intelligent Systems, 2020, 15(2): 352-359.

[9] LYU X, MA H. An efficient incremental mining algorithm for discovering sequential pattern in wireless sensor network environments[J]. Sensors, 2018, 19(1): 29.

[10] SANTOSO M H. Application of association rule method using Apriori algorithm to find sales patterns case study of indomaret tanjung anom[J]. Brilliance: Research of Artificial Intelligence, 2021, 1(2): 54-66.

[11] MUDUMBA B, KABIR M F. Mine-first association rule mining: an integration of independent frequent patterns in distributed environments[J]. Decision Analytics Journal, 2024, 10: 100434.

[12] MOKKADEM A, PELLETIER M, RAIMBAULT L. A recursive algorithm for mining association rules[J]. SN Computer Science, 2022, 3(5): 384.

[13] BUI-THI D, MEYSMAN P, LAUKENS K. MoMAC: multi-objective optimization to combine multiple association rules into an interpretable classification[J]. Applied Intelligence, 2022, 52(3): 3090-3102.

[14] HAN J, PEI J, YIN Y. Mining frequent patterns without candidate generation[C]//Proceedings of the 2000 ACM SIGMOD international conference on Management of data. New York: ACM, 2000: 1-12.

[15] 厍向阳, 张玲. 基于 Hadoop 的 FP-Growth 关联规则并行改进算法[J]. 计算机应用研究, 2018, 35(1): 109-112.
SHE Xiang-yang, ZHANG Ling. Parallel improved algorithm of FP-Growth association rules based on Hadoop[J]. Application Research of Computers, 2018, 35(1): 109-112.

[16] 毛伊敏, 邓千虎, 陈志刚. 基于信息熵与遗传算法的并行关联规则增量挖掘算法[J]. 通信学报, 2021, 42(5): 122-136.
MAO Yi-min, DENG Qian-hu, CHEN Zhi-gang. Parallel association rules incremental mining algorithm based on information entropy and genetic algorithm[J]. Journal on Communications, 2021, 42(5): 122-136.

[17] MAHDI M A, HOSNY K M, ELHENAWY I. FR-Tree: a novel rare association rule for big data problem[J]. Expert Systems with Applications, 2022, 187: 115898.

[18] SAVASERE A. Efficient algorithms for mining association rules in large databases of customer transactions[C]//Morgan Kaufmann, proceedings of the 21st VLDB conference. Zurich, 1995: 43-444.

[19] ZAKI M J. Scalable algorithms for association mining[J]. IEEE Transactions on Knowledge and Data Engineering, 2000, 12(3): 372-390.

[20] ZAKI M J, GOUDA K. Fast vertical mining using diffsets[C]//Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. New York: ACM, 2003: 326-335.

[21] FISTER I Jr, FISTER I, FISTER D, et al. A comprehensive review of visualization methods for association rule mining: Taxonomy, challenges, open problems and future ideas[J]. Expert Systems with Applications, 2023, 233: 120901.

[22] SUDIRMAN I D, BAHRI R S, UTAMA I D, et al. Using association rule to analyze hypermarket customer purchase patterns[C]//Proceedings of the Second Asia Pacific International Conference on Industrial Engineering and Operations Management. Surakarta, Indonesia, 2021: 14-16.

[23] TELIKANI A, GANDOMI A H, SHAHBAHRAMI A. A survey of evolutionary computation for association rule mining[J]. Information Sciences, 2020, 524: 318-352.

[24] 崔妍, 包志强. 关联规则挖掘综述[J]. 计算机应用研究, 2016, 33(2): 330-334.
CUI Yan, BAO Zhi-qiang. Survey of association rule mining[J]. Application Research of Computers, 2016, 33(2): 330-334.

[25] 欧阳为民, 蔡庆生. 基于垂直数据分布的关联规则高效发现算法[J]. 软件学报, 1999, 10(7): 754-760.
OUYANG Wei-min, CAI Qing-sheng. An efficient algorithm for discovering association rules based on vertical data layout[J]. Journal of Software, 1999, 10(7): 754-760.

[26] MOKKADEM A, PELLETIER M, RAIMBAULT L. SufRec, an algorithm for mining association rules: recursivity and task parallelism[J]. Expert Systems with Applications, 2024, 236: 121321.

责任编辑:李翠薇