文章编号: 1001-0920(2009)09-1310-06

一种新的动态关联规则及其挖掘算法

沈 斌[1,2], 姚 敏[2]
(1. 浙江大学 宁波理工学院, 浙江 宁波 315100; 2. 浙江大学 计算机学院, 杭州 310027)

摘 要: 在分析原有定义不足的基础上,提出一种新的动态关联规则,其支持度向量和置信度向量与经典定义相吻合,能更好地反映规则随时间变化的动态信息. 进一步提出两种新的动态关联规则挖掘算法: ITS 和 EFP-growth. 其中:两阶段 ITS 算法具有较好的可理解性;基于扩展 FP-树的 EFP-growth 算法适宜于高密度海量数据的挖掘. 实验结果表明,该算法具有较好的挖掘性能和可扩展性,适用于动态关联规则的有效挖掘.

关键词: 动态关联规则; 扩展 FP-树; 频数向量; 挖掘算法
中图分类号: TP18 **文献标识码**: A

A new kind of dynamic association rule and its mining algorithms

SHEN Bin[1,2], *YAO Min*[2]
(1. Ningbo Institute of Technology, Zhejiang University, Ningbo 315100, China; 2. College of Computer Science and Technology, Zhejiang University, Hangzhou 310027, China. Correspondent: SHEN Bin, E-mail: tsingbin@zju.edu.cn)

Abstract: In order to improve the shortcomings of the original dynamic association rule (DAR for short), this paper presents a new kind of DAR, which can reflect the dynamic information of rule with time better. Its defintions of support vector and confidence vector are accord with the classical defintions of support and confidence. Two new mining algorithms are also provided, which are ITS algorithm and EFP-growth algorithm. ITS algorithm has two stages, and can be well comprehended. EFP-growth algorithm is based on extended FP-tree and suitable for large volumes of data with high density. The experiment results show that the proposed algorithm has good performance and is fit for mining DARs.

Key words: Dynamic association rules; Extended FP-tree; Frequence vector; Mining algorithms

1 引 言

传统的关联规则挖掘算法认为:所发现的关联规则是静态和永恒有效的,但在实际上,关联规则和数据的特性会随着时间而发生改变. 以某超市若干年的历史销售数据为例,通过关联规则挖掘,可能得出"顾客在购买香烟的同时也会购买礼品"这样的规则. 仔细研究支持这个关联规则的相关数据,发现这些相关数据大量集中于春节前后几个月,而在平时则支持度较小. 如果使用这样的规则去指导平时的日常销售,则会失去实际意义,而在春节期间,该规则则有较强的指导价值.

为描述关联规则这种随时间变化的性质,Liu 等[1,2] 提出了使用支持度向量 S_V 和置信度向量 C_V 来描述规则的动态性. 在 S_V 和 C_V 的基础上,可充分利用柱状图分析、时间序列分析[1,2]、灰色 Markov 模型[3] 等,得到有关规则的详尽信息. 动态关联规则可应用于规则有效性随时间变动较大的领域,如包含大量季节性商品的销售数据库、股票购买/销售数据库等.

其他相关工作有: Au 等[4] 通过构建模糊决策树,发现关于规则变化的模糊元规则; Dong 等[5] 在一对数据集中发现差异性的关联规则频繁模式; Vreeken 等[6] 使用 MDL 准则,构建用于描述一对数据集差异的不相似性度量; Van 等[7] 研究了数据流中检测变化的算法; Au 等[8] 使用进化策略发现历史数据中的变化模式; Deepa 等[9] 使用遗传算法从

收稿日期: 2008-08-28; **修回日期**: 2008-12-01.
基金项目: 国家自然科学基金项目(60533040, 60525202); 浙江省自然科学基金重点项目(Z104267); 浙江大学宁波理工学院科研启动基金项目.
作者简介: 沈斌(1980—),男,浙江上虞人,讲师,博士,从事人工智能的研究; 姚敏(1954—),男,安徽桐城人,教授,博士生导师,从事人工智能等研究.

动态数据集中发现频繁项.

本文针对原定义的不足,提出一种新的动态关联规则,并阐述了挖掘算法 ITS 和 EFP-growth.

2　问题描述及其分析

2.1　动态关联规则原定义

动态关联规则[1,2] 是一种能描述自身特性随时间变化的关联规则,具体描述如下:

设项集合 $I = \{i_1, i_2, \cdots, i_m\}$,任务相关的事务数据集 D 是在时间段 t 内收集到的,可分为不相交长度为 n 的时间序列,即有 $t = \{t_1, t_2, \cdots, t_n\}$. 根据 t 的划分,整个数据集可分为 n 个数据子集 $D = \{D_1, D_2, \cdots, D_n\}$,其中数据子集 $D_i (i \in \{1,2,\cdots,n\})$ 的数据是在 $t_i (i \in \{1,2,\cdots,n\})$ 时间段内收集的. 项集 T 满足 $T \subseteq I$. 若 X 和 Y 为项集, $X \subset I, Y \subset I$,且 $X \cap Y = \varnothing$,则有如下定义:

定义 1　动态关联规则 $X \Rightarrow Y$(或项集 $X \cup Y$)的支持度向量具有如下形式:

$$S_V = [s_{(X\cup Y)_1}, s_{(X\cup Y)_2}, \cdots, s_{(X\cup Y)_n}],$$

$$\text{s.t. } s_{(X\cup Y)_i} = f_{(X\cup Y)_i}/M,\ i \in \{1,2,\cdots,n\}. \quad (1)$$

其中:$f_{(X\cup Y)_i}$ 为 $X \cup Y$ 在 D_i 中出现的频数, M 为 D 中的事务数.

在上述定义下, $X \Rightarrow Y$ 支持度 s 可由下式得到:

$$s = s_{(X\cup Y)} = f_{(X\cup Y)}/M = \sum_{i}^{n} f_{(X\cup Y)_i}/M = \sum_{i}^{n} s_{(X\cup Y)_i}. \quad (2)$$

其中:$f_{(X\cup Y)}$ 为项集 $X \cup Y$ 在 D 中出现的频数, $s_{(X\cup Y)}$ 为项集 $X \cup Y$ 在 D 中的支持度.

定义 2　动态关联规则 $X \Rightarrow Y$ 的置信度向量为

$$C_V = [c_{(X\cup Y)_1}, c_{(X\cup Y)_2}, \cdots, c_{(X\cup Y)_n}],$$

$$c_{(X\cup Y)_i} = s_{(X\cup Y)_i} / \sum_{i}^{n} s_{X_i} = s_{(X\cup Y)_i}/s_X,\ i \in \{1,2,\cdots,n\}. \quad (3)$$

其中:$s_{(X\cup Y)_i}$ 和 s_{X_i} 分别为项集 $X \cup Y$ 和项集 X 的 S_V 中第 i 个元素, s_X 为项集 X 的支持度.

关联规则 $X \Rightarrow Y$ 的置信度 c 可由下式得到:

$$c = \frac{s_{(X\cup Y)}}{s_X} = \sum_{i}^{n} s_{(X\cup Y)_i}/s_X = \sum_{i}^{n} c_{(X\cup Y)_i}. \quad (4)$$

定义 3　一条完整的动态关联规则具有支持度向量 S_V,置信度向量 C_V 和支持度 s,置信度 c 共 4 个参数,它有如下表示形式:

$$X \Rightarrow Y(S_V, C_V, s, c), \quad (5)$$

其中 S_V, C_V 和 c, s 可分别根据式(1),(3) 和(2),(4) 得到,并一起用于描述规则的动态性质.

2.2　原定义的不足之处

$s_{(X\cup Y)_i}$ 和 $c_{(X\cup Y)_i}$ 用于描述规则在对应 D_i 中的动态性质,因此有必要给出合理的定义.

在原定义中, $s_{(X\cup Y)_i}$ 为 $f_{(X\cup Y)_i}/M$. 因为 M 为固定值,所以 $s_{(X\cup Y)_i}$ 并不能反映 $X \cup Y$ 在 D_i 中支持度的度量,而仅仅是频数度量 $f_{(X\cup Y)_i}$.

原定义 $c_{(X\cup Y)_i}$ 为 $s_{(X\cup Y)_i}/s_X$,对于确定的 $X \Rightarrow Y$, s_X 为固定值. 因此它仅是前一度量的 $1/s_X$,同样不能反映 D_i 中置信度的度量. 从信息论的角度看,后者的度量不能提供任何新的信息,因而是冗余的.

例如,根据时间 $t = \{t_1, t_2\}$ 划分事务数据集 $D = \{D_1, D_2\}$,假设 D_1 包含 990 条事务记录,其中支持 $X \cup Y$ 和 X 的事务数均为 10; D_2 包含 10 条事务记录,支持 $X \cup Y$ 和 X 的事务数分别为 9 和 10. 对于规则 $X \Rightarrow Y$,根据原定义(1) 和(3),有

$$S_V = [s_{(X\cup Y)_1}, s_{(X\cup Y)_2}] = [0.01, 0.009],$$

$$C_V = [c_{(X\cup Y)_1}, c_{(X\cup Y)_2}] = [0.5, 0.45].$$

从上述 S_V 和 C_V 的值不难发现以下问题:

1) 单独考虑 D_1, $X \Rightarrow Y$ 在 D_1 的支持度 $S_{D_1} = 10/990 = 0.01$;单独考虑 D_2, $X \Rightarrow Y$ 在 D_2 的支持度 $s_{D_2} = 10/10 = 1$. 所以 $s_{D_2} > s_{D_1}$, $s_{(X\cup Y)_1} > s_{(X\cup Y)_2}$. 上述 S_V 定义与经典支持度定义矛盾.

2) $c_{(X\cup Y)_1} = s_{(X\cup Y)_1} \times 50$, $c_{(X\cup Y)_2} = s_{(X\cup Y)2} \times 50$. C_V 与 S_V 具有相同的比例,不能提供新的度量信息.

3　动态关联规则的新定义

下面给出更为恰当的 S_V 和 C_V 的定义:

定义 4　动态关联规则 $X \Rightarrow Y$(或项集 $X \cup Y$)的支持度向量具有如下形式:

$$S_V = [s_{(X\cup Y)_1}, s_{(X\cup Y)_2}, \cdots, s_{(X\cup Y)_n}],$$

$$\text{s.t. } s_{(X\cup Y)_i} = f_{(X\cup Y)_i}/|D_i|,\ i \in \{1,2,\cdots,n\}. \quad (6)$$

其中:$f_{(X\cup Y)_i}$ 为 $X \cup Y$ 在 D_i 中出现的频数, $|D_i|$ 为 D_i 中的事务数.

上述定义中, $s_{(X\cup Y)_i}$ 反映了 $X \cup Y$ 在 D_i 中支持度的度量. 此时,式(2) 不再成立,则 $X \Rightarrow Y$ 的支持度 s 可由下式得到:

$$s = s_{(X\cup Y)} = \frac{f_{(X\cup Y)}}{M} = \sum_{i}^{n} f_{(X\cup Y)_i}/M, \quad (7)$$

其中 M 是 D 中的事务数.

定义 5　动态关联规则 $X \Rightarrow Y$ 的置信度向量为

$$C_V = [c_{(X\cup Y)_1}, c_{(X\cup Y)_2}, \cdots, c_{(X\cup Y)_n}],$$

$$\text{s.t. } c_{(X\cup Y)_i} = s_{(X\cup Y)_i}/s_{X_i},\ i \in \{1,2,\cdots,n\}. \quad (8)$$

其中:$s_{(X\cup Y)_i}$ 为项集 $X \cup Y$ 的 S_V 中第 i 个元素, s_{X_i} 为项集 X 的 S_V 中第 i 个元素.

上述定义中, $c_{(X\cup Y)_i}$ 反映了 $X \cup Y$ 在 D_i 中置信度的度量,则 $X \Rightarrow Y$ 的置信度 c 可由下式得到:

$$c = s_{(X\cup Y)}/s_X. \quad (9)$$

定义 6 一条完整的动态关联规则具有支持度向量 S_V,置信度向量 C_V 和支持度 s,置信度 c 共 4 个参数,它有式(5)的表示形式. 其中 S_V, C_V 和 s, c 可分别根据式(6),(8)和(7),(9)得到,并一起用于描述规则的动态性质.

新的 S_V 和 C_V 与经典的支持度和置信度定义相吻合,能更好地反映规则随时间变化的动态性质.

定义 7 设有动态关联规则 $X \Rightarrow Y(S_V, C_V, s, c)$,最小支持度阈值为 min_sup,最小置信度阈值为 min_conf. 如果 $s \geq$ min_sup, $c \geq$ min_conf,则称 $X \Rightarrow Y$ 为强动态关联规则.

4 动态关联规则挖掘算法

为有效挖掘该规则,首先提出频数向量的概念.

定义 8 动态关联规则 $X \Rightarrow Y$ 的频数向量为

$$F_V = [f_{(X \cup Y)_1}, f_{(X \cup Y)_2}, \cdots, f_{(X \cup Y)_n}], \quad (10)$$

其中 $f_{(X \cup Y)_i}$ 为 $X \cup Y$ 在 D_i 中出现的频数.

于是可将整个过程分成两个阶段:第 1 阶段计算频繁项集 L 及其频数向量 F_V;第 2 阶段由 L 产生动态规则,由 F_V 产生 S_V 和 C_V.

第 2 阶段直截了当,可由规则生成函数 Rule-generation 得到.

对于第 1 阶段,Liu 等[1,2]提出了两种挖掘算法. 其中:Liu 算法 1 扫描数据库的次数与规则数量成正比,当规则数目庞大时,其开销很大,不具有实用价值;Liu 算法 2 则建立在 Apriori 的基础上.

为进一步提高算法效率,本文提出两种新算法:第 1 种是改进的两阶段挖掘算法 ITS,它继承了 Liu 算法 1 可理解性强的优点,同时具有更高的执行效率;第 2 种是基于扩展 FP 树(EFP 树)的 EFP-growth,它适宜于高密度海量数据的挖掘,比 Liu 算法 2 具有更好的性能.

4.1 改进的两阶段动态关联规则挖掘 ITS 算法

ITS 是一种改进的两阶段算法,其优点是具有较强的可理解性,可充分利用已有的高性能关联规则挖掘算法;其缺点是需要单独的过程生成频数向量.

设整个数据集为 D,并分割为 n 个数据子集 $D_1 \sim D_n$;全体频繁项集为 L, $l_j \in L$;事务项为 t, t 包含的频繁项集所构成的集合为 $L_{temp} = \text{subset}(L, t)$; l_j 在 D_i 中出现的频数为 $f_{(X \cup Y)_{ij}}$;频繁项集 l_j 的频数向量为 F_{V_j}.

算法包含两个阶段:第 1 阶段使用高性能算法,如 FP-growth,MAFIA 等,找到所有的频繁项集 L;第 2 阶段对数据库进行一次扫描,得到 l_j 在 D_i 上的频数 $f_{(X \cup Y)_{ij}}$,再由 $f_{(X \cup Y)_{ij}}$ 合成 F_{V_j}. 具体描述如下:

ITS 算法

输入:数据集 D 和子集 $D_1 \sim D_n$, min_sup

输出:频繁项集 L 及其对应的频数向量 F_V 和 s

1) (L, s) = High-performance-association-mining-algorithm// 得到频繁项集及其支持度

2) for each D_i do{

3) for each transaction $t \in D_i$ do { // 扫描数据库

4) L_{temp} = subset(L, t)// 得到 t 所包含的是频繁项集的子集

5) for each frequent item-set $l_j \in L_{temp}$ do

6) $f_{(X \cup Y)_{ij}}$ ++}}

7) 对于任一频繁项集 $l_j \in L$,由 $f_{(X \cup Y)_{ij}}$ 构成 F_V

8) return L with their corresponding F_V and s.

该算法在第 2 阶段仅需扫描 1 次数据库便可得到 F_V,因此比 Liu 算法 1 具有更好的性能.

4.2 基于 EFP 树的 EFP-growth 算法

Liu 算法 2[1,2]是基于 Apriori 框架的算法,当面临高密度海量数据时,执行效率较低. 为此,本文提出了基于 FP-growth 框架的 EFP-growth. 该算法在 FP 树的基础上进行扩展,提出了用于动态关联规则挖掘的 EFP 树,使之在有效生成频繁项集的同时,对频数向量进行存储和计算. 该算法的优点在于可充分利用 FP-growth 的高时空效率,具有较高的执行效率.

定义 9(EFP 树) 一棵 EFP 树包括:

1) 一个标记为 null 的根结点,根结点的子代为若干个前缀子树和一个频繁项表头.

2) 频繁项表头中的结点包含 4 个域:频繁项名、支持度计数、频数向量和结点头指针. 其中:频数向量的元素个数为数据子集的个数 n,它存放该频繁项在各个数据子集中的频数;结点头指针指向 EFP 树的第 1 个相同频繁项目结点.

3) 前缀子树中的结点包含 4 个域:频繁项名、支持度计数、频数向量和结点指针. 其中:支持度计数表示在该分支上包含该频繁项所有交易的数目;频数向量中的元素个数为数据子集的个数 n,其元素 $f_i (i \in \{1, 2, \cdots, n\})$ 表示在该分支上数据子集 D_i 所包含的该频繁项的交易数目;结点指针指向下一个相同频繁项目,可为 null.

下面基于 EFP 树,提出 EFP-growth 算法.

EFP-growth 算法

输入:数据集 D 和子集 $D_1 \sim D_n$, min_sup

输出:频繁项集 L 及其对应的频数向量 F_V 和 s

1) Tree ← EFP-construction (D, $D_1 \sim D_n$,

min_sup)

2) 创建长度为 0 的频繁项集 ϕ,设置其支持度为 D 的事务数,其频数向量为 $D_1 \sim D_n$ 的事务数

3) $(L, s, Fv) \leftarrow$EFP-growth(Tree, null)

Function: EFP-construction (D, $D_1 \sim D_n$, min_ sup)

4) 依次扫描 $D_1 \sim D_n$,得到频繁项目集 F,支持度计数及其频数向量 Fv_F,按支持度计数降序排列,生成频繁项列表,记为 L

5) 创建 Tree 的根结点 T,标记为 null,依次对 D_i ($i \in \{1, 2, \cdots, n\}$) 中的每个事务执行 6) 和 7) 操作

6) 选出此事务中出现在 L 中的所有频繁项,将它们按 L 的次序排列,记为 $[p \mid P]$, p 为第 1 个元素, P 是余下元素. 调用 insert_tree($[p \mid P]$, T, i)

7) 函数 insert_tree($[p \mid P]$, T, i) 执行如下:若 T 有子代结点 N,且 N.item-name = p.Item-name,则 N 的支持度计数加 1, N 的频数向量中第 i 个元素 f_{N_i} 加 1;否则,创建新结点 N,设其支持度为 1,频数向量中第 i 个元素 f_{N_i} 为 1,其余元素为 0,由其父链接指向 N,其结点链接指向相同项名的下一结点. 若 P 非空,则递归调用 insert_tree($[p \mid P]$, T, i)

Function: EFP-growth(Tree, α)

8) for each $\alpha_i \in$ Tree 的频繁项表头 do {

9) $L \leftarrow \{\phi\}$,输出频繁模式 $\beta \leftarrow \alpha_i \cup \alpha$, β.sup $\leftarrow \alpha_i$.sup, β $Fv \leftarrow \alpha_i$. Fv

10) $L \leftarrow L \cup \beta$,构造 β 的条件模式基和条件 EFP-tree Tree$_\beta$

11) if Tree$_\beta$ 含单路径 P, then {

12) for each 路径 P 上结点的组合 β do {

13) 输出频繁模式 $c \leftarrow \beta \cup \alpha$, c.sup $\leftarrow$minimum support of nodes in β

14) for each c. Fv 中的元素 c. Fv do

15) c. $Fv_i \leftarrow$minimum Fv_i of nodes in β

16) $L \leftarrow L \cup c$}}

17) else

18) if Tree$_\beta \neq \phi$, then{

19) $(L_t, s, Fv) \leftarrow$EFP-growth(Tree$_\beta$, β), $L \leftarrow L_t \cup L$}}

20) return L with their corresponding Fv and s.

在该算法中,频数向量的值在频繁项集生成过程中得到保存和计算,无需单独扫描计算生成.

4.3 规则生成函数

规则生成函数如下:1) 调用关联规则生成函数,得到规则集;2) 将 r_j 前项和后项并集所组成频繁项集的 Fv 和 s 分别赋给 r_j 的 Fv 和 s;3) 得到支持度向量和置信度向量. 具体描述如下:

Function: Rule-generation

输入: 频繁项集 L,对应的 Fv 和 s, min_conf

输出:动态关联规则集 R 及其相应的 Sv, Cv, Fv, s, c

1) (R, c) = rule-generation-sub-algorithm//调用关联规则生成函数

2) for each rule $r_j \in R$ do

3) 将 r_j 前项和后项并集的 Fv 和 s 分别赋给 r_j 的 Fv 和 s

4) 对于任一规则 $r_j \in R$,由 $f_{(X \cup Y)_{ij}} M / \mid D_i \mid$ 得到 $s_{(X \cup Y)_{ij}}$,并构成 Sv_j

5) 对于任一规则 $r_j \in R$,由 $s(X \cup Y)_{ij} / sx_i$ 得到 $c_{(X \cup Y)_{ij}}$,并构成 Cv_j

6) return R with their corresponding Sv, Cv, Fv, s, c.

上述函数中, $s_{(X \cup Y)_{ij}}$ 和 $c_{(X \cup Y)_{ij}}$ 分别是 r_j 支持度向量 Sv_j 和置信度向量 Cv_j 的第 i 个元素, sx_i 是项集 X 在 D_i 中的支持度, M 是 D 中的事务数.

5 性能评测

使用 VC6.0 来实现算法. 其中:ITS 第 1 阶段采用 MAFIA;第 2 阶段利用 Trie 树结构生成频繁项集 Trie 树,并在该树上进行相关的累加计数操作.

Liu 算法 1 需要以所产生规则数量级的次数扫描数据库,时间开销庞大,不具实用性,因此不列入比较范围. 下面对 ITS, EFP-growth 和 Liu 算法 2 进行比较测试. 测试平台为 CPU AMD Sempron 2400 + 1.67 GHz,内存 512 MB,操作系统为 Windows XP.

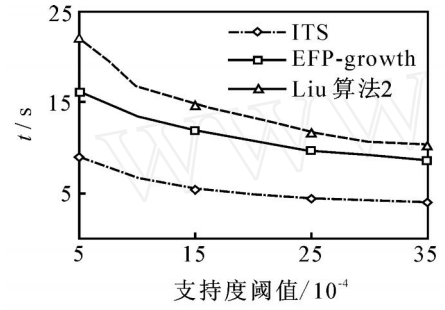
实验 1　不同数据集的算法性能.

采用 SQL Sever 2000 自带的 FoodMart 2000, T10I4D100K 和 Connect4 对算法性能进行测试. 其中:FoodMart 2000 为实际销售数据集,属稀疏型数据集;T10I4D100K 由 IBM Almaden 实验室提供的生成器产生,介于稀疏型与密集型之间;Connect4 取自 UCI 机器学习数据集,属非常密集型数据集.

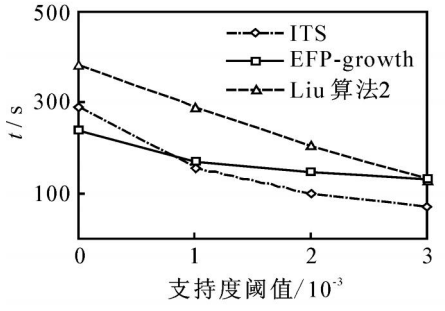
对于 FoodMart 2000,选取其中 1998 年的销售数据,将相同客户在同一时间内购买的商品视为同一购物篮,并将底层商品按商品类的层次进行概化,这样可有效避免出现底层商品种类多而支持度小的问题. 最终得到 41 011 条事务记录,将其按月分成 12 个数据子集进行测试. 对于 T10I4D100K,将其等分成 10 个子集进行测试,每个子集均包含 10 K 条事务数. 对于 Connect4,将其分成 7 个数据子集进行测试. 其中:前 6 个子集分别包含 10 K 条记录,最后一

个包含 7557 条记录.

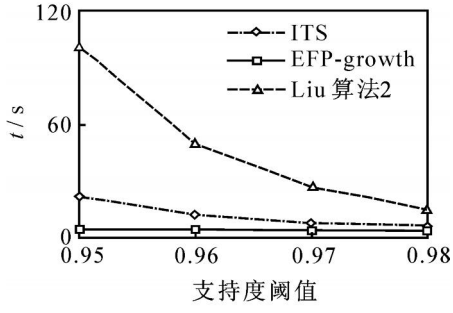
测试结果如图 1 所示,图中纵坐标为运行时间,横坐标为支持率阈值. 对于 FoodMart 2000,在高支持率阈值区域,ITS 的性能最好,Liu 算法 2 与 EFP-growth 性能较接近;在低支持率阈值区域,性能从高到低为 ITS > EFP-growth > Liu 算法 2. 对于 T10I4D100K,在高支持率阈值区域,ITS 的性能最好;在低支持率阈值区域,性能从高到低排列为 EFP-growth > ITS > Liu 算法 2. 对于 Connect4,性能差异较为明显,从高到低排列为 EFP-growth > ITS > Liu 算法 2.



(a) FoodMart 2000数据集



(b) T10I4D100K数据集



(c) Connect4数据集

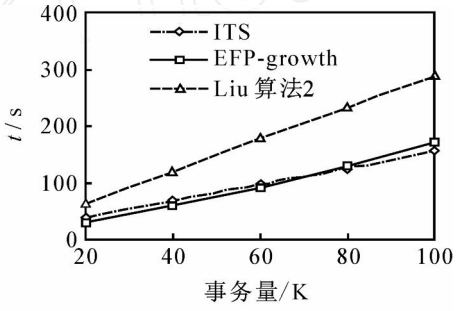
图 1 不同数据集测试 3 种算法的性能

Liu 算法 2 是基于 Apriori 框架的算法,需要耗费大量时间处理规模庞大的频繁项候选集,并要多次扫描数据库;EFP-growth 建立在高压缩 EFP-tree 的基础上,具有更好的性能优势,在高密度数据集上尤为明显;ITS 的优势在于可充分利用已有高性能挖掘算法,当频繁项集数目较少时,第 2 阶段运算量较小,因此对于稀疏型数据集和高支持度阈值,能表现出较好的执行效率.

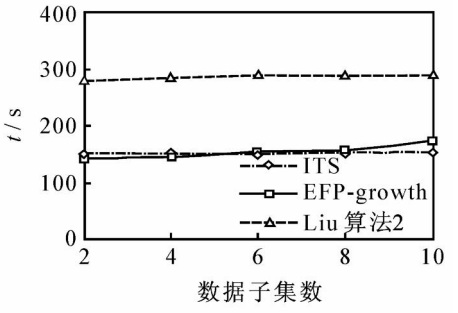
实验 2 参数变化对算法性能的影响.

支持度阈值变化对算法性能的影响较大. 从图 1 可以看出:当支持度阈值变小时,执行时间逐步增大;在高支持度阈值区域,算法执行时间变化较慢,而在低支持度阈值区域,由于频繁项集数量迅速增大,执行时间增长幅度迅速变大.

在事务量可扩展性测试中,选取 T10I4D100K 为测试集,支持度阈值为 0.001,依次选取其中的 20K,40K,60K,80K 和 100K,划分成 10 个子集. 测试结果如图 2(a) 所示. 可以发现:随着事务量的逐步增加,3 种算法的执行时间以近似线性的趋势增大,具有较好的可伸缩性.



(a) 事务量变化



(b) 数据子集个数变化

图 2 3 种算法的可扩展性测试

在数据子集个数对算法性能影响的测试中,支持度阈值为 0.001,选取 T10I4D100K 为测试集,依次等分为 2,4,6,8 和 10 个数据子集. 测试结果如图 2(b) 所示. 可以发现:随着子集个数的增加,执行时间有一定程度的增长,但幅度不大. 这表明 ITS 和 EFP-growth 对子集个数变化具有较好的可伸缩性.

6 结 论

动态关联规则采用支持度向量和置信度向量来描述规则随时间变化的情况,具有普通关联规则所不具有的优点. 本文针对原定义的不足,给出了新的动态关联规则相关定义及其挖掘算法 ITS 和 EFP-growth. ITS 先用已有关联规则算法得到频繁项集,再扫描一次数据集得到频数向量;基于 EFP-tree 的 EFP-growth 采用 FP-growth 框架. 实验结果验证了两种算法的有效性和可扩展性. 如何更好地反映

规则的动态性质,有待于进一步研究.

参考文献(References)

[1] Liu Jin-feng, Rong Gang. Mining dynamic association rules in databases [C]. Proc of Int Conf on Computational Intelligence and Security. Xi'an, 2005: 688-695.

[2] 荣冈, 刘进锋, 顾海杰. 数据库中动态关联规则的挖掘[J]. 控制理论与应用, 2007, 24(1): 129-133.
(Rong G, Liu J F, Gu H J. Mining dynamic association rules in databases[J]. Control Theory & Applications, 2007, 24(1): 129-133.)

[3] 刘俊, 谢彦峰,张忠林,等. 基于灰色 Markov 模型动态关联规则的元规则挖掘[J]. 计算机应用, 2008, 28(9): 2353-2356.
(Liu J, Xie Y F, Zhang Z L, et al. Research of mining meta-association rules for dynamic association rule based on model of Grey-Markov[J]. Computer Applications, 2008, 28(9): 2353-2356.)

[4] Wai-Ho Au, Keith C C Chan. Mining changes in association rules: A fuzzy approach[J]. Fuzzy Sets and Systems, 2005, 149(1): 87-104.

[5] Dong G Z, Li J Y. Mining border descriptions of emerging patterns from dataset pairs [J]. Knowledge and Information Systems, 2005, 8(2): 178-202.

[6] Vreeken J, Van Leeuwen M, Siebes A. Characterising the difference[C]. Proc of 13th ACM SIGKDD Int Conf on Knowledge Discovery and Data Mining. San Jose, 2007: 765-774.

[7] Van Leeuwen M, Siebes A. STREAMKRIMP: Detecting change in data streams[C]. Proc of European Conf on Machine Learning and Principles and Practices of Knowledge Discovery in Data. Antwerp, 2008: 672-687.

[8] Wai-Ho Au, Keith C C Chan. An evolutionary approach for discovering changing patterns in historical data[C]. Proc of 11th IEEE Int Conf on Fuzzy Systems. Honolulu, 2002: 890-895.

[9] Deepa Shenoy P, Srinivasa K G, Venugopal K R. Dynamic association rule mining using genetic algorithms [J]. Intelligent Data Analysis, 2005, 5(9): 439-453.

(上接第 1309 页)

[8] 邵远, 何发昌, 彭健. 一种机器人非视觉多传感器信息融合方法[J]. 电子学报, 1996, 24(8): 94-97.
(Shao Y, He F C, Peng J. An approach of robot non-vision multi-sensor fusion[J]. Acta Electronica Sinica, 1996, 24(8): 94-97.)

[9] 陈守煜, 胡吉敏. 可变模糊方法及其在工件识别中的应用[J]. 系统工程与电子技, 2006, 28(9): 1325-1328.
(Chen S Y, Hu J M. Variable fuzzy method and its application in parts recognition [J]. Systems, Engineering and Electronics, 2006, 28(9): 1325-1328.)

[10] 车录锋, 周晓军, 徐志农, 等. 可拓方法在多传感器信息融合工件识别中的应用[J]. 系统工程理论与实践, 2000, 20(8): 91-94.
(Che L F, Zhou X J, Xu Z N, et al. Application of extension method in multi-sensor data fusion for parts recognition [J]. System Engineering Theory and Practice, 2000, 20(8): 91-94.)

[11] 郭文艳, 韩崇昭. 基于灰度关联的多传感器融合目标识别方法[J]. 传感器与微系统, 2007, 26(9): 115-117.
(Guo W Y, Han C Z. Method of multi-sensor fusion target recognition based on gray correlation analysis [J]. Transducer and Microsystem Technologies, 2007, 26(9): 115-117.)

[12] Yager R R. On ordered weight averaging aggregation operators in multi-criteria decision making[J]. IEEE Trans on Systems, Man and Cybernetics, 1988, 18(1): 183-190.

[13] Yager R R. Induced ordered weight averaging operators[J]. IEEE Trans on Systems, Man and Cybernetics, 1999, 29(2): 141-150.

[14] 钟嘉鸣, 李订芳. 基于粗糙集理论的属性权重确定最优化方法研究[J]. 计算机工程与应用, 2008, 44(20): 51-53.
(Zhong J M, Li D F. Research on optimization method of attribute weight determining based on rough set theory[J]. Computer Engineering and Applications, 2008, 44(20): 51-53.)