



大数据中效用挖掘的快速单阶段算法

刘君强¹, 周青峰¹, 王文慧², 时磊¹

(1. 浙江工商大学 杭州 310018; 2 浙江水利水电学院 杭州 310018)

摘要:现有数据挖掘算法的缺点是在挖掘大数据时会出现大量候选模式,从而造成可伸缩性瓶颈,个别算法虽然不生成候选模式,但是计算代价高昂,缺乏有效剪裁,运行效率存在瓶颈。为此,提出一个全新的单阶段不生成候选模式的数据挖掘算法,其创新性有3点:一是基于前缀生长的模式枚举和基于效用上限值评估的剪裁策略;二是基于稀疏矩阵和虚拟投影的效用信息表达;三是节省存储空间深度优先搜索方法。大量实验表明,新算法的时间效率比现有算法高5倍以上,并且内存使用量比现有算法少20%~60%,可伸缩性高。

关键词:大数据;效用挖掘;高效用模式;频繁模式

doi: 10.11959/j.issn.1000-0801.2015100

Fast Single Phase Algorithm for Utility Mining in Big Data

Liu Junqiang¹, Zhou Qingfeng¹, Wang Wenhui², Shi Lei¹

(1. Zhejiang Gongshang University, Hangzhou 310018, China;

2. Zhejiang University of Water Resources and Electric Power, Hangzhou 310018, China)

Abstract: Most of the latest works on utility mining generates a huge number of candidates in dealing with big data, which suffers from the scalability issue. Some work does not generate candidates, but suffers from the efficiency issue due to lack of strong pruning and high computation overhead. A novel algorithm that finds high utility patterns in a single phase without generating candidates was proposed. The novelties lie in a prefix growth strategy with strong pruning, and a sparse matrix based representation of transactions with pseudo projection. The proposed algorithm works in a depth first manner and does not materialize high utility patterns in memory, which further improves the scalability. Extensive experiments on synthetic and real-world data show that the proposed algorithm outperforms the latest works in terms of running time, memory overhead, and scalability.

Key words: big data, utility mining, high utility pattern, frequent pattern

1 引言

效用模式挖掘^[1-6]是近年来发展起来的大数据分析技

术,不仅考虑数据统计显著性,而且也考虑用户兴趣和目标^[7]。例如,传统频繁模式挖掘技术^[8-11]只能从销售数据中挖掘出购买频率较高的产品组合,而效用模式挖掘技术可

收稿日期:2014-12-09;修回日期:2015-04-10

基金项目:国家自然科学基金资助项目(No.61272306),浙江省自然科学基金资助项目(No.LY12F02024)

Foundation Items: The National Natural Science Foundation of China (No.61272306), The Zhejiang Provincial Natural Science Foundation of China (No.LY12F02024)

论文引用格式:刘君强,周青峰,王文慧. 大数据中效用挖掘的快速单阶段算法. 电信科学, 2015100

Liu J Q, Zhou Q F, Wang W H. Fast single phase algorithm for utility mining in big data. Telecommunications Science, 2015100

以从中发现利润回报较高的产品组合。效用模式挖掘不仅是各种挖掘问题的基础^[12-14],也可以直接应用于各种大数据分析。例如,网络传媒的点击率和转化率分析、价值链分析、网购的消费者行为理解和预测等。

然而,效用模式挖掘技术还不成熟,只有很少量成果。由于效用模式不具有反单调性,即一个低效用模式的超集可能是高效用的,挖掘高效用模式要比挖掘频繁模式困难得多,因为很难剪裁搜索空间。现有挖掘算法大多数采用两阶段法,即先在第1阶段从原始数据中挖掘出候选模式,再在第2阶段从候选模式中进一步挖掘出效用模式。其缺点是挖掘大数据时会产生大量候选模式,造成存储空间开销过大,形成可伸缩性瓶颈,并最终导致运行的时间效率低下。

不少工作^[3,5]试图降低第1阶段生成候选模式数量。然而,对于海量的数据库或给定较小的效用阈值时,候选模式数量还是巨大的。这不仅导致第1阶段的伸缩性瓶颈,对于第2阶段也是如此,并最终导致效率瓶颈。个别算法^[15]虽然不生成候选模式,但是依赖于纵向数据表达形式和计算代价高昂的交集运算,缺少有效剪裁方法,运行效率比二阶段算法还差^[5],不适合于大数据处理。为了克服这些问题,本文提出一个全新算法,在单个阶段直接挖掘出效用模式。主要贡献如下。

首先,提出一个全新算法 SPUP (single phase utility pattern mining)来挖掘效用模式。具体地讲,提出稀疏矩阵来表达事务效用的完整信息,使得单阶段挖掘成为可能。这种稀疏矩阵表达比基于 Fptree^[9]的表达方法^[1,2,5]结构更紧凑,避免多遍扫描数据库^[3,4]。

第二,提出前缀生长策略及相应的前缀扩展树,用于引导效用模式的挖掘过程,并得到效用模式搜索空间剪裁技术的支撑,即通过估算任意子空间的模式效用上限值,有效地剪裁前缀生成树。

第三,采用人工数据和真实数据,对比已有最先进算法^[1,4,5,15]的实验结果表明,本文提出的 SPUP 算法在运行时间和空间开销上均优于其他算法。换言之,SPUP 算法是效率最高、可伸缩性最佳的效用挖掘算法。

2 大数据分析中的效用模式挖掘问题

大数据分析的特点之一是数据来源的多样性。多个不同的数据来源为每个被分析的对象提供了丰富多样的属性,但如何融合多源的数据却是一个难题。效用挖掘通过

引入效用模型,为融合多个属性并从中提炼有价值信息提供一种通用方法。因此,效用挖掘在大数据分析中有着广泛的应用。

简要讨论一个移动电子商务应用案例^[12]。移动通信运营商保存大量移动设备轨迹数据,单纯依赖轨迹数据只能分析“高频繁度的轨迹”,例如“访问 A”以后很可能“访问 B”,其商业价值并不高。然而,如果将移动运营商的轨迹数据与消费者购物的支付数据结合起来进行大数据分析,通过融合网点、商品、商品价值等属性进行效用挖掘,就可以找出“高价值的轨迹”,例如“访问 A 购买唇膏”以后很可能“访问 B 购买钻戒”。另一个相似案例是电商平台的大数据分析,通过效用挖掘分析客户在线购物的消费行为,找出“高价值的网购轨迹”,用于改进电商平台设计和指导网上促销活动。

效用挖掘通过效用的概念将具体应用问题中多个属性融合起来表达所研究对象的价值,并从中挖掘高价值的对象,即高效用模式。下面给出高效用模式挖掘的定义。

$I = \{i_1, i_2, \dots, i_m\}$ 是所有项目的集合, $D = \{t_1, t_2, \dots, t_n\}$ 为数据库,即事务记录的集合。每个事务记录 t 是一个项目集,即 $t \subseteq I$ 。而 $u(i, t) = iu(i, t) \cdot eu(i)$ 则是项目 i 在事务记录 t 中的效用,其中 $iu(i, t)$ 是项目 i 在事务记录 t 中的份额, $eu(i)$ 是项目 i 独立于任何事务记录的外在效用。模式 X 是 I 的一个子集,如果模式 X 中的任意项目 i 在事务 t 中的份额非 0,即 $iu(i, t) \neq 0$,则模式 X 被事务记录 t 所支持,即 $X \subseteq t$ 。

- $u(X, t)$ 是事务记录 t 所支持的模式 X 的效用,取值

$$\text{为 } u(X, t) = \sum_{i \in X} u(i, t).$$

- $TS(X)$ 则是支持模式 X 的所有事务记录的集合。

- $u(X)$ 是 X 的总效用,即所有支持 X 事务记录效用和,

$$u(X) = \sum_{t \in TS(X)} u(X, t) = \sum_{t \in TS(X)} \sum_{i \in X} u(i, t).$$

- 模式 X 称作(高)效用模式,如果其效用值不低于给定阈值 $minutil$,即 $u(X) \geq minutil$ 。

- 效用模式挖掘就是发现所有(高)效用模式,即求解

$$HUPset = \{X \mid X \subseteq I, u(X) \geq minutil\}.$$

例如,表 1 给定项目集合 $I = \{a, b, c, d, e, f, g\}$,数据库 D 由 5 条事物记录组成,其中 t_1 包含 3 个项目 a, c, e ,并且 $iu(a, t_1) = 1, iu(c, t_1) = 1, iu(e, t_1) = 1$ 。表 2 是所有项目外在效用汇总。由 $eu(a) = 1$,可以得到 $u(a, t_1) = iu(a, t_1) \times eu(a) = 1$ 。假定效用模式 $X = \{b, c\}$,由表 1 可知: $TS(X) = \{t_2, t_3\}$, X 的总效用 $u(X) = u(X, t_2) + u(X, t_3) = u(b, t_2) + u(c, t_2) + u(b, t_3) +$

表 1 事务数据库 D

tid	item						
	a	b	c	d	e	f	g
t_1	1		1		1		
t_2	6	2	2			5	
t_3	1	1	1	2	6		5
t_4	3	1		4	3		
t_5	2	1		2		2	

表 2 效用表 UT

item	profit(\$)
a	1
b	3
c	5
d	2
e	2
f	1
g	1

$u(c, t_3) = 24$ 。给定最小效用阈值 $minutil = 30$, 可以得到高效用模式的集合为 $\{ \{a, b, c\}, \{a, b, d\}, \{a, d, e\}, \{a, b, d, e\}, \{b, d, e\}, \{d, e\}, \{a, b, c, d, e, g\} \}$ 。

效用模式不具有反单调性, 如上例中 $u(\{a, c\})=28$ 、 $u(\{a, b, c\})=31$ 、 $u(\{a, b, c, d\})=13$ 。因此, 挖掘高效用模式要比挖掘频繁模式困难得多。

3 相关工作

效用模式挖掘是频繁模式挖掘的发展。频繁模式挖掘算法以 Agrawal 和 Srikant 的 Apriori 算法^[8]以及 Han 等人的 FP-Growth 算法^[9]最为著名。Apriori 算法充分利用支持度的反单调性: 任何频繁模式的子集都是频繁模式。FP-Growth 算法采用基于树的结构存储事务数据库, 也利用反单调性。

近几年, 效用模式挖掘受到越来越多关注。Yao 等人^[7]指出效用指标不具有反单调性。因此, 大量算法采用生成候选模式的二阶段挖掘方法。Liu 等人^[4]提出事务加权效用 TWU 向下闭合性质, 设计了两阶段算法 TP, 第 1 阶段先找出具有高 TWU 的模式(候选模式), 第 2 阶段再次扫描数据库来计算各个候选模式的实际效用, 从而找出高效用模式。Li 等人^[3]提出孤立项剔除策略, 用于逐层挖掘候选模式的第 1 阶段, 以减少多余候选模式。

为克服逐层生成候选模式时多遍扫描数据库^[3,4]的缺点, 以便第 1 阶段能高效率地生成候选模式, 多个研究小组提出基于树的效用模式挖掘算法^[1,2,5]。Erwin 等人^[2]提出 CTU-

PROL 算法, 运用事务加权效用 TWU 向下闭合性质^[4]、基于效用模式树 CUP-tree、FP-Growth 算法^[9]进行挖掘。Ahmed 等人^[1]提出一种基于树的 IHUP 算法, 采用 IHUP-tree 来存储各个事务的 TWU 信息, 改进 FP-Growth 算法^[9]来挖掘效用模式的候选模式集。Tseng 等人^[5]设计基于树的二阶段算法 UPG, 利用 UP-tree 压缩存储 TWU 信息, 并提出递减 TWU 的策略, 因而生成为较少数量的候选模式。

Liu 等人^[15]提出 HUIMiner 算法, 直接计算各模式的效用信息, 避免产生和计算候选模式, 但是所依赖的纵向数据表达形式不适用于大数据, 同时代价高昂的交集运算使得其运行效率比二阶段算法还差^[5]。

本文提出剪裁搜索空间的有效办法和节省内存的数据结构, 解决可伸缩性与时间效率的瓶颈。

4 基于前缀生长的模式枚举和剪裁策略

挖掘效用模式最基本的办法是穷举项目集合 I 的每一个子集 X , 但是组合爆炸的问题使得这种穷举法行不通。因此, 在挖掘效用模式过程中很有必要使用剪裁技术。本节首先引入一种新颖的枚举效用模式的策略, 然后提出剪裁技术。

4.1 前缀生长的模式枚举策略

为了避免重复列举效用模式, 需要为数据库中的所有项目指定一个排列顺序。这样, 一个模式也可以表达为一个序列。因此, 集合表达式(如 $\{a, b, c\}$)与序列表达式(如 $\langle a, b, c \rangle$)可以混合使用, 集合“并”操作也可以用于两个序列的连接, 如 $\langle a, b, c \rangle \cup \langle d \rangle = \langle a, b, c, d \rangle$ 。下面提出一种前缀生长策略。

- $prefix(X, t)$ 是事务记录 t 相对于模式 $X = \langle i_k, \dots, i_l \rangle$ 的前缀, 即 t 中按照指定顺序排列在 i_k 前的项目组成的一个子序列。
- $preEXT(X, t) = prefix(X, t) \cup X$ 是模式 X 相对于事务记录 t 的前缀扩展。
- 前缀生长策略就是将空模式拼接前缀得到长度为 1 的模式, 将长度为 1 的模式拼接前缀得到长度为 2 的模式, 以此类推。

假定所有项目按照字典序进行排列, 那么 $\langle a \rangle$ 是空模式 $\langle \rangle$ 的一个前缀扩展, $\langle a, b \rangle$ 是模式 $\langle b \rangle$ 的一个前缀扩展。对于表 1 数据库 D , 可以得到: $preEXT(\langle c \rangle, t_1) = \langle a, c \rangle$, $preEXT(\langle c \rangle, t_2) = \langle a, b, c \rangle$ 。

根据所提出的前缀生长策略, 枚举效用模式可以看作使一颗前缀扩展树不断生长的过程。在前缀扩展树中, 根



元素是 $u(i, t)$, 即项目 i 在事务记录 t 中的效用。

这种表达法有 3 个长处:一是能够完整地表达效用信息,克服了二阶段法^[1-5]只能表达 TWU 的缺陷;二是存储开销低,实际上比基于 Fptree^[9]的表达方法^[1,2,5]更紧凑;三是计算开销低,这种表达法是按事务记录展开的横向表达法,通过穿线的办法可以快速确定各个模式的事务记录集,计算开销远低于按项目展开的纵向表达法^[15]所采用的交集运算。

表达一个模式 X 的事务记录集 $TS(X)$ 的效用信息的稀疏矩阵也称为 X 的效用信息矩阵,记作 $uMatrix(X)$ 。图 2 是根据表 1 所示数据库 D 和效用表 UT 确定的空模式 $\{\}$ 的效用信息矩阵 $uMatrix(\{\})$ 。因此,事务记录集 $TS(X)$ 和效用信息矩阵 $uMatrix(X)$ 可以视为同一个数据结构,下面不再加以区分。

5.2 基于虚拟投影的模式效用信息计算

通过搜索前缀扩展树来枚举模式时,树节点 N 所代表模式的效用信息矩阵是其父亲节点 P 所代表模式的效用信息矩阵的子矩阵,可以通过虚拟投影获得。

具体来讲,对于节点 N 的标记项目 i 的节点 C ,可以从 $uMatrix(pat(N))$ 中选择那些含有 i 的事务记录,然后得到模式 $pat(C) = (\{i\} \cup pat(N))$ 对应的效用信息矩阵 $uMatrix(pat(C))$ 。这种操作被称作投影,即节点 C 对应的效用信息矩阵 $uMatrix(pat(C))$ 通过投影其父亲节点 N 对应的效用信息矩阵 $uMatrix(pat(N))$ 得到。实际上,除了根节点之外,任何节点对应的效用信息矩阵都嵌套在其父亲节点对应的效用信息矩阵中。例如,从 $uMatrix(\{\})$ 可得到 $uMatrix(\{c\})$,从后者可得到 $uMatrix(\{b,c\})$ 。

因此,只需要在内存中保留根节点对应的效用信息矩阵,其他节点的效用信息矩阵只是父亲节点的效用信息矩阵的子矩阵,不需要开辟独立的内存空间。从节点 N 的 $uMatrix(pat(N))$ 推导出节点 C 的 $uMatrix(pat(C))$ 的操作称为虚拟投影,采用虚拟投影能进一步提高算法的伸缩性。

6 单阶段挖掘高效用模式

本节提出单阶段高效用模式挖掘算法 SPUP。SPUP 算法将第 4.1 节的前缀生长策略、第 4.2 节的基于效用上限值剪裁策略、第 5 节的基于稀疏矩阵的效用信息表达方法相结合,并应用到按深度优先搜索高效用模式的过程中。SPUP 算法采用深度优先搜索时,只需要在内存中保存当

前正在生长的前缀扩展树分支,极大地节省了内存空间,进一步提高算法的伸缩性。下面是 SPUP 算法的伪代码,大致分为 5 步,可以结合例子进行解释。

算法 SPUP($D, UT, minutil$)

(1) 创建根节点 N 并令 $pat(N) = \{\}$;

(2) 根据 D 和 UT 计算 $uMatrix(\{\})$;

(3) $Subtree(N, uMatrix(pat(N)), minutil)$ 。

函数 $Subtree(N, uMatrix(pat(N)), minutil)$

(4) for $uMatrix(pat(N))$ 的每列 i do

(5) 计算 $u(\{i\} \cup pat(N))$ 和 $ubound(\{i\} \cup pat(N))$;

(6) if $(u(\{i\} \cup pat(N)) \geq minutil)$ then 输出 $\{i\} \cup pat(N)$;

end if

(7) if $(ubound(\{i\} \cup pat(N)) \geq minutil)$ then

(8) 创建节点 C 并令 $pat(C) = \{i\} \cup pat(N)$;

(9) 通过虚拟投影得到 $uMatrix(C)$;

(10) $Subtree(C, uMatrix(pat(C)), minutil)$;

(11) end if

(12) end for

(13) 清除 N 为根的子树。

步骤 1 (第 1~3 行):创建前缀扩展树的根节点。通过扫描数据库 D ,过滤掉全部无关的项目,得到根节点(空模式)对应的效用信息矩阵 $uMatrix(\{\})$ 。

步骤 2 (第 4~5 行):对于节点 N 对应的模式 $pat(N) = \langle i_k, \dots, i_1 \rangle$,根据定理 1,从 $uMatrix(pat(N))$ 中过滤掉无关的项目。然后,对于按指定顺序排在 i_k 前面的非无关项目 i ,根据定理 2,计算模式 $\{i\} \cup pat(N)$ 的效用值 $u(\{i\} \cup pat(N))$ 和效用上限值 $ubound(\{i\} \cup pat(N))$ 。

例如,图 3 是根据表 1 的数据库 D 和表 2 的外在效用表 UT 生成的部分前缀扩展树,其中所有项目按照字典序排序,最小效用阈值 $minutil = 30$ 。每个节点 N 都标明其效用信息矩阵 $uMatrix(pat(N))$ 以及各项目 i 的效用值(u)和上限值(uB),即 $u(\{i\} \cup pat(N))$ 和 $ubound(\{i\} \cup pat(N))$ 。

步骤 3 (第 6 行):如果效用值 $u(\{i\} \cup pat(N)) \geq minutil$,那么模式 $\{i\} \cup pat(N)$ 是一个高效用模式,将它输出保存。如图 3 所示,有一个节点对应模式 $\{b, c\}$,其节点对应模式 $\{a\} \cup \{b, c\}$ 的效用值 $u(\{a\} \cup \{b, c\}) = 31 > minutil$,因此 $\{a, b, c\}$ 是一个高效用模式,将它输出。

步骤 4 (第 7~11 行):如果 N 的节点 C 对应模式 $\{i\} \cup pat(N)$ 的效用上限值 $ubound(\{i\} \cup pat(N)) \geq minutil$,需

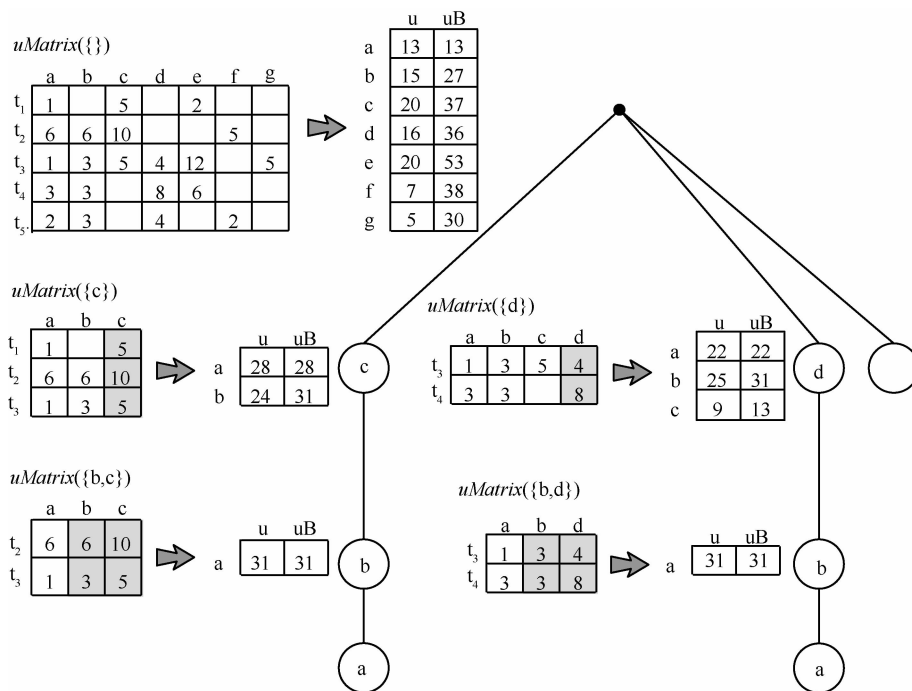


图3 前缀扩展树以及各个节点的效用信息矩阵

要通过虚拟投影得到节点 C 对应的效用信息矩阵 $uMatrix(\{i\} \cup pat(N))$, 然后深度优先搜索以节点 C 为根的子树。如图3所示, $\{a\}$ 、 $\{b\}$ 和 $\{c\}$ 都是根节点 $\{\}$ 的节点, 由于 $ubound(\{a\})$ 和 $ubound(\{b\})$ 都小于 $minutil$, 项目 a, b 对应的节点被剪裁掉。与此同时, 由于 $ubound(\{c\}) = 37 > minutil$, 项目 c 对应的节点被创建, 然后继续深度优先搜索该节点对应的子树。

步骤5 (第12~13行): 深度优先搜索完节点 N 的所有后, 以节点 N 为根的子树就被清除, 然后回溯到节点 N 的父亲节点 P , 继续深度优先搜索 P 的下一个节点, 以此类推。如图3所示, 搜索模式 $\{a, b, c\}$ 相应的节点后, 回溯到根节点, 然后继续深度优先搜索根节点的项目 d 对应的节点。

7 实验评价

本节通过实验对比, 评估效用挖掘的效果, 并且评估所提出的单阶段模式挖掘算法 SPUP 的效率和可伸缩性。实验用到两个数据集: 第一个是 IBM Quest 人工数据集 T2016N1kD1mL6k, 有 100 万条事务记录, 事务记录平均长度为 20 个项目, 最大频繁模式平均长度为 6 个项目, 每一条事务记录中各个项目份额服从 1~5 的均匀分布, 每一个项目对应的外在效用值服从 0.01~10 的对数正态分布; 第二个是美国加州一所大型百货超市连锁店销售的真实数据集^[4], 有 1 112 949 条销售事务记录, 每一条事务记录由商品和商品销售数量组成, 表示某一个消费者在某个时间

段购买商品情况。事务记录平均长度为 7.2, 效用表描述每种商品的利润。实验环境为 3 GHz CPU / 3 GB RAM 的 PC 一台, 运行于 Windows 7 系统的 Cygwin。

7.1 效用挖掘的效果评价

本小节通过对比本文 SPUP 算法挖掘得到的 TopK 高效用模式和 CAP 算法^[10]挖掘得到的 TopK 闭合频繁模式, 评估效用挖掘的效果。具体采用两个指标: 第一个是覆盖度 (recall), 定义为 TopK 高效用模式被 TopK 闭合频繁模式所覆盖 (包含) 的比例; 第二个是精确度 (precision), 定义为 TopK 闭合频繁模式中包含 TopK 高效用模式的比例。如果覆盖度和精确度比较低, 则表明挖掘高效用模式和挖掘频繁模式有着不同的效果。

图4(a)是人工数据集的 TopK 闭合频繁模式和 TopK 高效用模式的挖掘效果对比。当 TopK 从 1 000 变到 10 000 时, 覆盖度均低于 18.2%, 精确度均低于 45.6%。例如, 前 5 000 个闭合频繁模式只覆盖 5.6% 的前 5 000 个高效用模式, 44.8% 的前 5 000 个闭合频繁模式包含前 5 000 个高效用模式。其中, 精确度大于覆盖度的原因是多个闭合模式包含同一个高效用模式。这个结果表明, 大部分高效用模式不是频繁模式。

图4(b)是真实数据集的挖掘效果对比。当 TopK 从 1 000 变到 10 000 时, 覆盖度均低于 6%, 精确度均低于 12.1%。例如, 前 5 000 个闭合频繁模式只覆盖 2.5% 的前

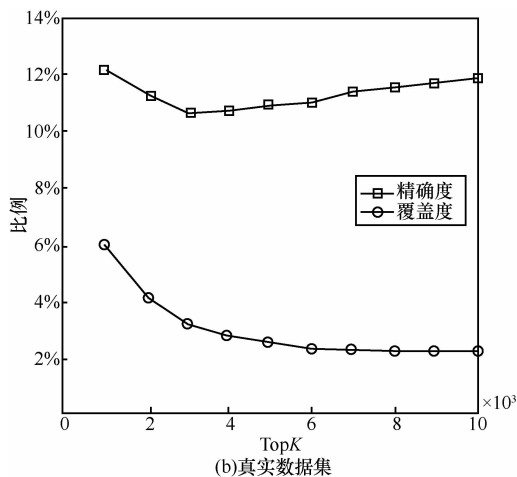
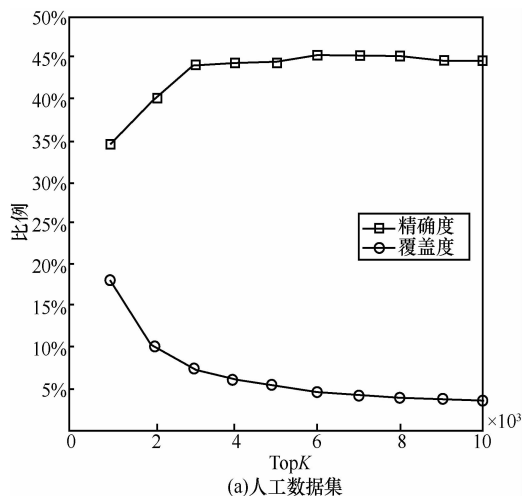


图4 TopK模式的覆盖度 recall 和精确度 precision

5 000 个高效用模式,10.9%的前 5 000 个闭合频繁模式包含前 5 000 个高效用模式。相对于人工数据集,真实数据集的 TopK 闭合频繁模式比 TopK 高效用模式的覆盖度和精确度更低,这更说明高效用模式挖掘在实际应用中的价值远远高于频繁模式挖掘。

7.2 算法效率评价

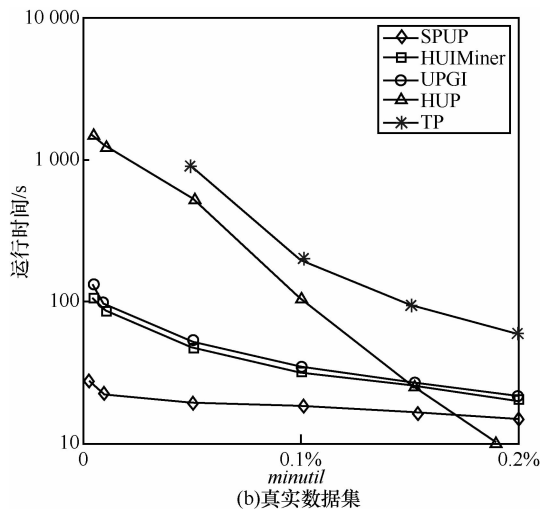
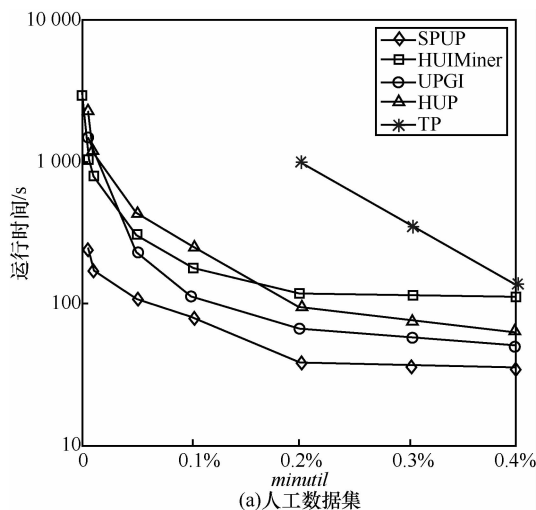
本小节通过对比现有最先进效用模式挖掘算法 TP^[4]、IHUP^[1]、UPG^[5]、HUIMiner^[15],评估所提出的单阶段挖掘算法 SPUP 的效率,所有算法都采用 C++ 实现。

图 5(a)是 5 种算法挖掘人工数据集的运行时间,其中横坐标表示最小效用阈值 $minutil$, 范围是 0.005%~0.4%,纵坐标表示运行时间。当效用阈值 $minutil = 0.2\%$ 时,SPUP 算法运行 39 s,UPG 算法运行 67 s,IHUP 算法运行 98 s, HUIMiner 算法运行 118 s, 而 TP 算法的运行时间达到 1 009 s。当效用阈值 $minutil < 0.2\%$ 时,TP 算法不能运行。当效用阈值 $minutil = 0.005\%$ 时,SPUP 算法运行 248 s,

HUIMiner 算法运行 1 052 s,UPG 算法运行 1 430 s,IHUP 算法运行 2 346 s。

图 5(b)是挖掘真实数据集的运行时间, $minutil$ 范围是 0.005%~0.2%。当效用阈值 $minutil \geq 0.2\%$ 时,高效用模式数量很少,此时 HUIMiner 算法运行最快;当 $minutil$ 低于 0.2% 时,HUIMiner 算法的运行时间增长很快。当效用阈值 $minutil = 0.05\%$ 时,SPUP 算法运行 20 s,UPG 算法运行 52 s, IHUP 算法运行 55 s,HUIMiner 算法运行 630 s,而 TP 算法运行时间达到 1 117 s;当效用阈值 $minutil = 0.005\%$ 时,SPUP 算法运行 26 s,UPG 运行 116 s,IHUP 算法运行 154 s, HUIMiner 算法运行 1 797 s,TP 算法不能运行。

综合比较可知,SPUP 算法效率总是最高的,算法 UPG^[5]、

图5 各算法随着 $minutil$ 变动的运行时间比较

HUIMiner^[15]和 IHUP^[1]在不同情况下效率互有高低,SPUP 算法的效率比后 3 个算法高 5 倍以上,TP 算法^[4]效率总是最低的,SPUP 算法比 TP 算法高 70 倍。

7.3 存储开销和伸缩性评价

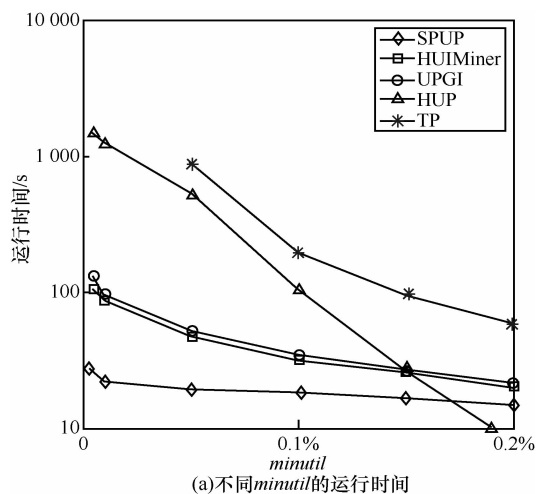
图 6(a)分别是算法 SPUP、HUIMiner、UPG 和 IHUP 挖掘人工数据集的最大内存使用量,TP 算法没有统计内存使用情况。当效用阈值 $minutil = 0.4\%$ 时,SPUP 算法消耗内存 295 MB,HUIMiner 算法消耗内存 373 MB,UPG 算法消耗内存 424 MB,IHUP 算法消耗内存 426 MB。当效用阈值 $minutil = 0.01\%$ 时,SPUP 算法消耗内存 307 MB,HUIMiner 算法消耗内存 380 MB,UPG 算法消耗内存 485 MB,IHUP 算法消耗内存 534 MB。尽管没有直接统计数据,但 TP 算法消耗内存最多,因为当效用阈值 $minutil < 0.2\%$ 时,由于缺少内存,TP 算法不能运行。

图 6(b)是挖掘真实数据集的最大内存使用量。当效用阈值 $minutil = 0.2\%$ 时,SPUP 算法消耗内存 26.8 MB,算法 UPG 和 IHUP 消耗内存 63.2 MB,HUIMiner 算法消耗内存 74.9 MB。当效用阈值 $minutil = 0.05\%$ 时,SPUP 算法消耗内存 86.2 MB,HUIMiner 算法消耗内存 122.9 MB,UPG

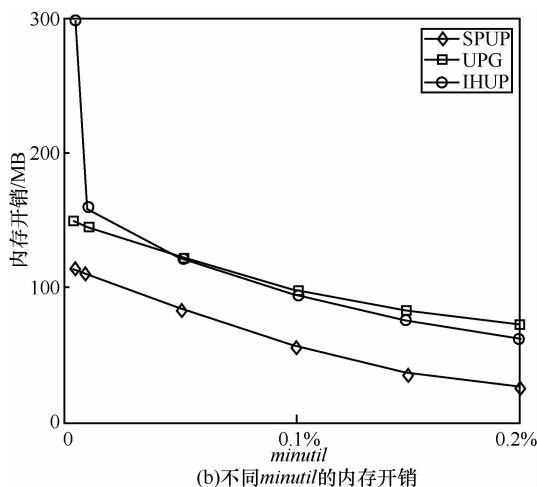
算法消耗内存 126.1 MB,IHUP 算法消耗内存 126.5 MB。当效用阈值 $minutil = 0.005\%$ 时,SPUP 算法消耗内存 116.2 MB,HUIMiner 算法消耗内存 149.9 MB,UPG 算法消耗内存 165.4 MB,IHUP 算法消耗内存 306.7 MB。TP 算法消耗内存最多,因为当 $minutil < 0.01\%$ 时,TP 算法因缺少内存而不能继续运行。

综上可知,SPUP 算法消耗内存最少,HUIMiner 算法和 UPG 算法内存消耗量次之,IHUP 算法消耗内存最多,SPUP 算法比后 3 个算法少消耗 20%~60% 的内存,TP 算法消耗内存最多。

最后,通过实验评估 SPUP 算法的可伸缩性。实验采用人工数据集 T2016N1kL6k,令记录数从 10 万条变到 100 万条,效用阈值 $minutil = 0.01\%$ 。如图 7 所示,SPUP 算法的运行时间与记录数的关系近似线性,SPUP 算法的内存开销与记录数的关系也是线性的,表明 SPUP 算法具有很好的可伸缩性。

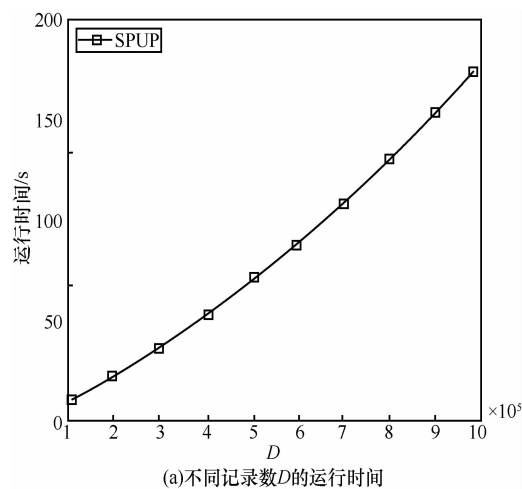


(a)不同 $minutil$ 的运行时间

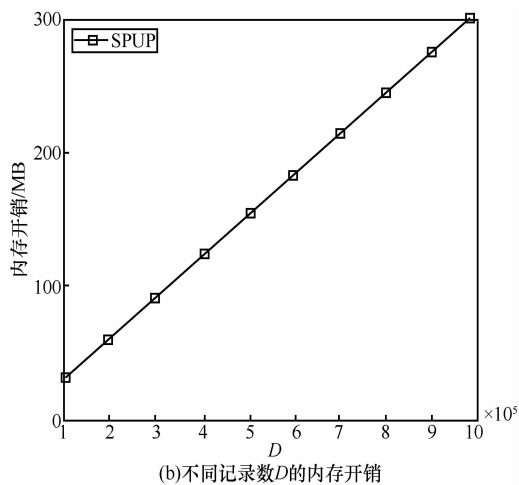


(b)不同 $minutil$ 的内存开销

图 6 各个算法随着 $minutil$ 变动的内存开销比较



(a)不同记录数 D 的运行时间



(b)不同记录数 D 的内存开销

图 7 SPUP 算法的可伸缩性实验结果



8 结束语

效用挖掘是融合多源的大数据进行综合分析的一种通用方法,其挖掘效果显著地不同于频繁模式挖掘,但是遇到效率与可伸缩性难题。为此,本文提出大数据中发现效用模式的快速单阶段挖掘算法 SPUP,采用稀疏矩阵表达法保存事务记录的完整效用信息,无须多次扫描数据库,直接挖掘高效用模式;提出前缀生长策略和评估模式效用上限值的方法,能有效地剪裁搜索空间。实验表明,SPUP 算法的效率高于现有算法 5 倍以上,内存开销减少 20%~60%,可伸缩性高。

参考文献

- 1 Ahmed C F, Tanbeer S K, Jeong B S, *et al.* Efficient tree structures for high utility pattern mining in incremental databases. *IEEE Transactions on Knowledge and Data Engineering*, 2009, 21 (12): 1708~1721
- 2 Erwin A, Gopalan R P, Achuthan N R. Efficient mining of high utility itemsets from large datasets. *Proceedings of PAKDD*, Osaka, Japan, 2008
- 3 Li Y C, Yeh J S, Chang C C. Isolated items discarding strategy for discovering high utility itemsets. *Data & Knowledge Engineering*, 2008, 64(1): 98~217
- 4 Liu Y, Liao W, Choudhary A. A fast high utility itemsets mining algorithm. *Proceedings of the Utility-Based Data Mining Workshop in Conjunction With the 11th ACM SIGKDD*, Chicago, Illinois, USA, 2005
- 5 Tseng V S, Shie B E, Wu C W, *et al.* Efficient algorithms for mining high utility itemsets from transactional databases. *IEEE Transactions on Knowledge and Data Engineering*, 2013, 25 (8): 1772~1786
- 6 Yen S J, Lee Y S. Mining high utility quantitative association rules. *Proceeding of the 9th International Conference on Data Warehousing and Knowledge Discovery*, Regensburg, Germany, 2007
- 7 Yao H, Hamilton H J, Geng L. A unified framework for utility-based measures for mining itemsets. *Proceedings of ACM SIGKDD the 2nd Workshop on Utility-Based Data Mining*, Philadelphia, PA, USA, 2006
- 8 Agrawal R, Srikant R. Fast algorithms for mining association rules. *Proceedings of the 20th International Conference on Very Large Databases*, Santiago, Chile, 1994
- 9 Han J, Pei J, Yin Y. Mining frequent patterns without candidate generation. *Proceedings of ACM SIGMOD conference*, Dallas, USA, 2000
- 10 Liu J, Pan Y. An efficient algorithm for mining closed itemsets. *Journal of Zhejiang University Science*, 2004, 5(1): 8~15
- 11 Liu J, Pan Y, Wang K, *et al.* Mining frequent item sets by opportunistic projection. *Proceedings of SIGKDD*, Edmonton, Canada, 2002
- 12 Shie B E, Cheng J H, Chuang K T, *et al.* A one-phase method for mining high utility mobile sequential patterns in mobile commerce environments. *Proceedings of IEA/AIE12*, Dalian, China, 2012
- 13 Wu C W, Lin Y F, Yu P S, *et al.* Mining high utility episodes in complex event sequences. *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Chicago Illinois, USA, 2013
- 14 Wu C W, Shie B E, Tseng V S, *et al.* Mining top-*K* high utility itemsets. *Proceedings of SIG KDD*, Beijing, China, 2012
- 15 Liu M, Qu J. Mining high utility itemsets without candidate generation. *Proceedings of CIKM*, Maui, HI, USA, 2012

[作者简介]



刘君强,男,博士,浙江工商大学教授,主要研究方向为大数据分析 with 云计算、网络信息安全与隐私保护、信息管理与软件工程。

周青峰,男,浙江工商大学硕士生,主要研究方向为大数据分析 with 云计算、数据挖掘、Web Services。

王文慧,女,浙江水利水电学院讲师,主要研究方向为数据挖掘、机器学习、复杂系统建模与控制。

时磊,男,博士,浙江工商大学讲师,主要研究方向为无线传感器网络、Ad Hoc 网络。

2015100-9

(上接 2015109-6)

王元,女,西安邮电大学硕士生,主要研究方向为 D2D 通信和异构网。

唐睿,男,西安交通大学博士生,主要研究方向为 D2D 通信和异构网。

张振伟,男,西安交通大学博士生,主要研究方向为 D2D 通信和异构网。

曲桦,男,博士,西安交通大学教授,博士生导师,主要研究方向为现代通信网、计算机通信网、计算机网络体系结构。

2015109-7