

# 基于时间间隔和点击量的 Prefixspan 改进算法

王娜娜, 陈立潮, 潘理虎, 张英俊

(太原科技大学 计算机科学与技术学院, 山西 太原 030024)

**摘要:**数据挖掘算法过程中对客户行为的实时性是分析客户网络消费行为的重要要素之一,但是 Prefixspan 数据挖掘算法挖掘过程中并未对此问题予以考虑,因此,在时间间隔序列模式概念的基础上,提出了一种基于时间间隔和点击量的 Prefixspan 改进算法。在该算法中,引入了频繁度和时间属性的概念,并加入了时间间隔和点击量等要素,从而使挖掘到的信息具有实时性的特点,并且提高了对挖掘对象的侧重性。通过实验验证,与原来的 Prefixspan 算法相比较后表明,改进算法用于具有时间特性的数据集时获得的挖掘结果更精确,挖掘效率得到了有效的提高。

**关键词:**时间间隔;点击率;序列模式;数据挖掘

**中图分类号:**TP301.6

**文献标识码:**A

**文章编号:**1673-629X(2011)10-0081-04

## An Improved Prefixspan Algorithm Based on Time Interval and Click Quantity

WANG Na-na, CHEN Li-chao, PAN Li-hu, ZHANG Ying-jun

(School of Computer Sci. and Tech., Taiyuan University of Sci. and Tech., Taiyuan 030024, China)

**Abstract:** The real-time character of customer behavior is one of the main factors for analyzing customer's internet consumption behavior. But it was ignored in the data mining algorithm of Prefixspan, so based on the concept of time interval sequence pattern, an improved algorithm integrated with time interval and click quantity was presented. In this algorithm, the concept of the frequent degree and time attribute was imported and the factors of time interval and click quantity was added, which made the mined dates had the real-time character, and improved the emphasis on sex of the mining object. The experiment shown that compared with the original algorithm, the improved algorithm was more precise, when used to mine the data set with real-time character, at the same time the mining efficiency has been improved effectively.

**Key words:** time interval; click quantity; sequence patterns; data mining

## 0 引言

数据挖掘是指从大量的、不完全的、有噪声的、模糊的、随机的数据中,提取隐含在其中的、人们事先未知的有用信息或知识的过程。数据挖掘技术已应用于许多领域<sup>[1,2]</sup>,如客户流失分析、交叉销售、欺诈检测、风险管理、客户细分、广告定位、销售预测等。

在当今竞争激烈和快速变化的商业环境中,如果一个企业可以在正确的时间提供正确的商品或服务给正确的消费者,那么它就可以比其它企业获得更多的盈利机会。因此,对企业来讲,深层次地挖掘用户的行为是非常重要的。序列模式挖掘方法是数据挖掘方法中一种非常重要的挖掘方法,在从大型数据库中高效

挖掘频繁子树、格、子图等结构模式方面发挥重要作用。它由 Agrawal 和 Srikant<sup>[3]</sup>在1995年首次提出,用于发现序列数据库中的频繁序列模式。近年来,序列模式挖掘的研究有了很大进步,产生了很多高性能算法<sup>[4]</sup>。其中比较典型的算法有 Apriori 算法<sup>[5]</sup>、GSP 算法<sup>[6]</sup>、基于模式扩展的 Prefixspan 算法<sup>[7-11]</sup>及 FreeSpan 算法<sup>[12]</sup>等。

Pei 等人提出了模式扩展的概念<sup>[11]</sup>,并研究出 FreeSpan 算法、Prefixspan 算法等。Prefixspan 算法由 FreeSpan 算法演化而来。Prefixspan 算法采用分治的思想,不断产生序列数据库的多个更小的投影数据库,然后在各个投影数据库上进行序列模式挖掘。此算法不需要产生候选序列模式,因此显著缩减了检索空间。相对于原始的序列数据库而言,其投影数据库的规模不断减小。但由于传统的 Prefixspan 算法只是针对用户的访问节点进行挖掘,得到的频繁序列较多,结果较笼统,信息可靠性不足。

文中所提出的算法在挖掘的过程中加入了时间特

收稿日期:2011-03-01;修回日期:2011-06-05

基金项目:山西省自然科学基金资助项目(2009011022-1)

作者简介:王娜娜(1985-),女,硕士研究生,研究方向为人工智能及应用;陈立潮,博士,教授,硕士生导师,中国计算机学会高级会员,主要研究方向为智能软件、人工智能。

性和用户访问量的因素,并提出了频繁度的概念,使其具有实时性和精确性的特点,从而使得挖掘出来的信息更加准确、可靠。

## 1 改进算法中的问题定义

关于序列模式挖掘的经典应用是研究消费者购买行为。虽然消费者购买行为序列模式包括一系列项目,但是在该序列中,哪些项目是消费者比较喜欢的,或者是什么时间预计要购买的,这就需要考虑到点击量和时间等特性。

在消费者购买实例中,消费者采购清单中的商品列表是代表性的序列数据,每一个商品都带有它交易的时间,同时根据这些数据,还可以计算出每种商品的交易次数。商品序列  $A$  可以表示为  $((w_1 t_1)(w_2 t_2)(w_3 t_3) \cdots (w_n t_n))$ , 其中,  $w_j$  表示项目,  $t_j$  表示项目发生的时间点,  $1 \leq j \leq n, t_{j-1} \leq t_j, 2 \leq j \leq n$ 。如果在序列中商品交易的时间相同,则按它们的交易次数的大小排列。

两个有效项目发生的时间差被称为时间间隔  $T$ , 随后给定常量  $k$ , 其中  $0 < k < r$ ; 所以时间间隔被分为  $r-1$  个时间段,  $I_0$  代表时间间隔  $t=0$ ;  $I_1$  代表时间间隔  $0 < t \leq T_1$ ;  $I_j$  代表时间间隔  $T_{j-1} < t \leq T_j$ , 其中  $1 < j < r-1$ ;  $I_r$  代表时间间隔  $T_{r-1} < t < \infty$ 。定义时间间隔  $TI = \{I_0, I_1, I_2, \dots, I_r\}$ 。则基于时间间隔的序列模式定义如下:

定义 1: 令  $W = \{w_1, w_2, \dots, w_m\}$  为一系列项目,  $TI = \{I_1, I_2, \dots, I_r\}$  为一组时间间隔, 则序列  $a = \{b_1, \&_1, b_2, \&_2, \dots, b_{s-1}, \&_{s-1}, b_s\}$  为一基于时间间隔的序列模式, 其中,  $b_i \in W, 1 \leq i \leq s, \&_j \in TI, 1 \leq j \leq s-1$ 。序列中项目的个数称为序列的长度。长度为  $K$  的序列称为  $K$ -序列。

每一项事务被表示为  $\langle \text{sid}, s \rangle$ ,  $\text{sid}$  表示此项事务的身份, 即数据序列识别标识, 不能缺失。  $s$  表示数据序列, 允许缺失。一个序列数据库  $S$  由一系列  $\langle \text{sid}, s \rangle$  记录构成。

定义 2: 点击量是指访问项目  $a_i$  的独立  $\text{sid}$  数或者是浏览人数。此处并不考虑顺序问题。  $a_i$  的访问率记为  $\text{count}(a_i)$ 。

定义 3: 给定一基于时间约束的序列模式  $a$ , 它在数据库  $S$  中的支持度计算被如下定义:

$\text{support\_count}_t(a) = |\{(\text{sid}, s) \mid (\text{sid}, s) \in S \cap a \text{ 被包含于 } s\}|$

## 2 Prefixspan 改进算法

本研究提出了一种新的基于时间间隔和点击量的序列挖掘方法 (Click and time-interval sequential pat-

terns CTSP), 对于序列数据库中的序列, 将访问率和支持度二者结合起来, 计算出最后的频繁度, 根据频繁度再进行序列挖掘。

改进的 Prefixspan 算法中, 利用一个三元组来存储数据元素, 记为:  $\langle \text{sid}(i), P(i), t(i) \rangle$ , 其中  $\text{sid}(i)$  表示项目  $i$  的原始交易序列号,  $\text{seq}$  表示序列信息项目,  $t(i)$  表示项目在原始序列中的发生时间,  $\text{count}(i)$  表示项目在序列中获得的点击量, 对用户的点击量也设置一个阈值, 只有大于最小点击量的才可以考虑到挖掘算法中, 如果小于点击阈值的序列, 则它的子序列的点击量也小于给定阈值。最后序列数据库中的序列的频繁度通过公式 (1) 计算得到。

频繁度计算公式利用了加权平均法。挖掘的内容不同, 设定的加权值  $\sigma \mid \sigma \mid 0 < \sigma < 1$  也不尽相同。频繁度计算公式如下所示:

$$\text{fre}(i) = [\text{count}(i) \quad \text{sup}(i)] \begin{bmatrix} \sigma \\ 1 - \sigma \end{bmatrix} \quad (1)$$

1) 专家评判法是一种最基本的指标权重确定方法。指标权重用来反映各个属性的相对重要性, 属性越重要, 赋予的权重越大, 反之则越小。在此算法中利用传统的专家评判法给出公式中的加权平均值。在研究过程中, 针对商品销售领域, 得到两个专家给出的权重, 分别为  $(0.5, 0.5)$ ,  $(0.7, 0.3)$ , 得到的权重矩阵为:

$$\begin{bmatrix} 0.5 & 0.5 \\ 0.7 & 0.3 \end{bmatrix}$$

再对专家权重进行归一化处理得  $[0.6 \ 0.4]$ 。所以在频繁度计算中, 支持度属性和点击量属性所占的权重分别是 0.6 和 0.4。

2) 算法的具体过程如下:

(1) 扫描序列数据库长度为 1 的序列, 得到各个序列的点击量和序列支持度。根据频繁度计算公式, 计算出各 1-序列的频繁度, 并根据给出的最小频繁度, 找出频繁 1-序列, 然后依照频繁度的大小, 依次排列。

(2) 由 1-序列产生长度为 2 的频繁 2-序列。2-序列是由 1-频繁序列自连接得到的, 不需要再扫描原数据库。序列依然由一个三元组表示,  $(\text{sid}(i), P(i), T(i))$ ,  $\text{sid}$  表示项目  $i$  的交易序列号,  $P(i)$  表示序列,  $T(i)$  表示两元素之间的时间间隔。通过频繁项之间的自连接来构造 2-序列, 并根据二者之间发生的时间间隔, 得到 2-序列的支持度, 进而计算出频繁度, 挖掘出数据库中的 2-频繁序列。

(3) 以此类推, 不断得到基于各个频繁前缀的投影数据库, 再在投影数据库中按照上面的方法递归地进行挖掘, 直到找到数据库中的所有频繁序列为止。

### 3 Prefixspan 算法示例

给定序列数据库数据如表 1 所示,已知  $TI = \{I_0, I_1, I_2, I_3\}$ 。其中: $I_0:t=0, I_1:0 < t \leq 3, I_2:3 < t \leq 6, I_3:6 < t < \infty$ 。

表 1 序列数据库

Sid	sequence
10	((a,1)(c,3)(a,4)(b,4)(a,6)(e,6)(c,10))
20	((d,5)(a,7)(b,7)(e,7)(d,9)(e,9)(c,14)(d,14))
30	((a,8)(b,8)(e,11)(d,13)(b,16)(c,16)(c,20))
40	((b,15)(f,17)(e,18)(b,22)(c,22))

第一步:扫描序列数据库长度为 1 的序列,得到各个序列的点击量和序列频繁度。以表 1 中的数据库为例,  $\text{count}(a) = 5, \text{count}(b) = 6, \text{count}(c) = 6, \text{count}(d) = 4, \text{count}(e) = 5, \text{count}(f) = 1$ 。再扫描项目的支持度  $\langle a \rangle : 3, \langle b \rangle : 4, \langle c \rangle : 4, \langle d \rangle : 2, \langle e \rangle : 3, \langle f \rangle : 1$ 。假设最小频繁度为 3,根据频繁序列的计算方法,则频繁序列为  $\langle a \rangle : 3.8, \langle b \rangle : 4.8, \langle c \rangle : 4.8, \langle e \rangle : 3.8$ 。 $d, f$  为非频繁序列。

第二步:由 1-序列产生长度为 2 的频繁 2-序列。2-序列是由 1-频繁序列自连接得到的,不需要再扫描原数据库。由此可得如表 2 所示。

表 2 序列扩展表

a	b	c	e
		(10, (ac), $I_1$ )	(10, (ae), $I_2$ )
	(10, (ab), $I_1$ )	(10, (ac), $I_3$ )	(10, (ae), $I_2$ )
	(10, (ab), $I_0$ )	(10, (ac), $I_2$ )	(10, (ae), $I_0$ )
a	(20, (ab), $I_0$ )	(20, (ac), $I_3$ )	(20, (ae), $I_0$ )
	(30, (ab), $I_0$ )	(20, (ac), $I_1$ )	(20, (ae), $I_1$ )
		(30, (ac), $I_3$ )	(30, (ae), $I_1$ )
		(10, (bc), $I_2$ )	
		(20, (bc), $I_3$ )	
		(30, (bc), $I_3$ )	(10, (be), $I_1$ )
b		(30, (bc), $I_3$ )	(20, (be), $I_0$ )
		(30, (bc), $I_0$ )	(20, (be), $I_1$ )
		(30, (bc), $I_2$ )	(30, (be), $I_1$ )
		(40, (bc), $I_0$ )	(40, (be), $I_1$ )
		(40, (bc), $I_3$ )	
c		(10, (ec), $I_2$ )	
		(20, (ec), $I_3$ )	
		(20, (ec), $I_2$ )	
e		(30, (ec), $I_2$ )	
		(30, (ec), $I_3$ )	
		(40, (ec), $I_2$ )	

从表中得出 2-时间间隔序列:  $\langle a, I_0, b \rangle : 3, \langle a, I_3, c \rangle : 3, \langle b, I_3, c \rangle : 4, \langle b, I_1, e \rangle : 4, \langle e, I_2, c \rangle : 4$ 。 $\text{count}(ab) = 6, \text{count}(ac) = 9, \text{count}(bc) = 9, \text{count}(be) = 5, \text{count}(ec) = 7$ 。由计算公式得,  $\langle ab \rangle : 4.2, \langle ac \rangle : 5.4, \langle bc \rangle : 6, \langle be \rangle : 4.4, \langle ec \rangle : 5.2$ 。已知最小频繁度为 3,所以,频繁 2-时间间隔序列为:  $\langle a, I_0, b \rangle, \langle a, I_3, c \rangle, \langle b, I_3, c \rangle, \langle b,$

$I_1, e \rangle, \langle e, I_2, c \rangle$ 。

第三步:以此类推,在各个频繁前缀的投影数据库中按照上面的方法递归地进行挖掘,直到找到数据库中所有的频繁序列为止。

### 4 算法性能分析及实验验证

实验环境为 Pentium(R) D CPU 3.2GHz, 1.0GB 内存, Windows XP 操作系统, 算法代码采用 C++ 编写在 Visual C++ 6.0 环境下编译。测试数据 C10k| N10| T5| I6, 其中, C 代表的是数据库中的序列数, N 代表数据库中的项目数, T 表示时间间隔为 5, I 表示潜在序列的最大长度为 6。运行 CTSP 算法和 Prefixspan 算法, 对 2 个算法的挖掘结果进行对比。

如图 1 所示, 测试了两算法在不同支持度下所挖掘出的模式数。在改进的算法中, 挖掘出的频繁模式序列数比传统的 Prefixspan 算法挖掘出的序列数更少。

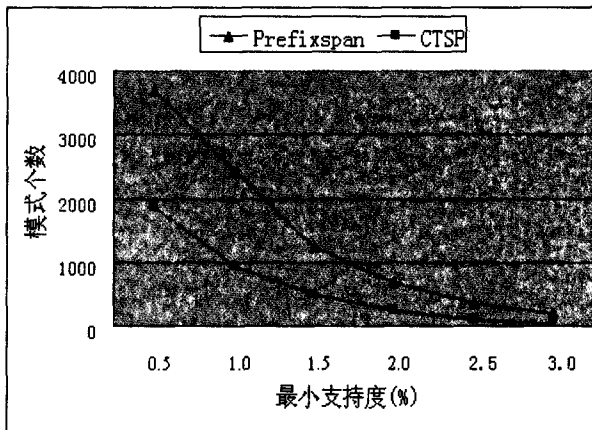


图 1 算法挖掘出的模式数

对挖掘出的序列进行进一步的分析, 根据序列挖掘的度量定义了一个评估指标: 准确率。准确率是指根据以往经验判定的序列中, 正确的序列所占挖掘结果的比例。通过对挖掘结果进行进一步的分析, 结果如图 2 所示。

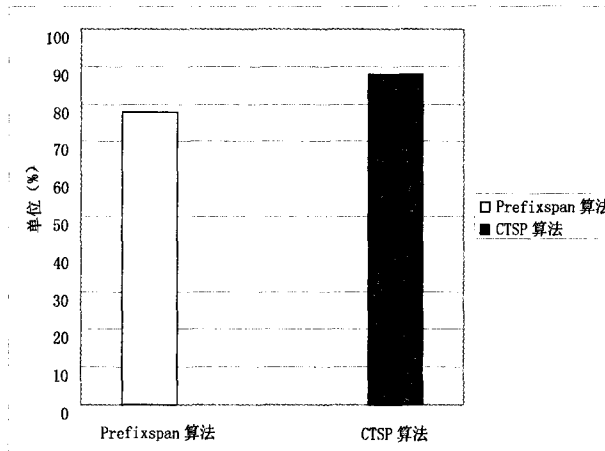


图 2 算法挖掘的准确率

如图 2 所示,测试了两算法在相同评价指标下挖掘出的序列的准确率。在改进的算法中,挖掘出的频繁模式序列的准确率要比传统的 Prefixspan 算法挖掘出的序列的准确率高约 10%。

## 5 结束语

本研究将时间属性和频繁度的概念引入到 Prefixspan 算法中,提出了基于时间属性和点击量的 Prefixspan 改进算法。在改进的算法中,除了根据用户的访问序列进行挖掘以外,同时还将用户的访问时间和点击量因素加入到挖掘信息的过程中。提高了挖掘信息的实时性与准确性,从而使互联网时代的信息更加高效和准确地服务于用户。但是,该算法的改进是以部分时间和空间作为代价的,下一步的主要工作是如何进一步改进算法的时间复杂性和空间复杂性,提高算法的整体性能。

### 参考文献:

- [1] 孟宪苹,宋 菲,李 俊.基于序列模式挖掘的入侵检测系统的研究[J].计算机技术与发展,2008,18(3):154-156.
- [2] 叶和平,尚 敏.一种面向入侵检测的数据挖掘算法研究[J].计算机技术与发展,2008,18(11):149-151.
- [3] Agrawal R, Srikant R. Mining Sequential Patterns[C]//Proceedings of the 11th International Conference on Data Engineering. Taipei, Taiwan, China:IEEE Press,1995.
- [4] 吴 楠,胡学刚.基于聚类分区的序列模式挖掘算法研究[J].计算机技术与发展,2010,20(6):109-112.
- [5] 陈 伟.Apriori 算法的优化方法[J].计算机技术与发展,2009,19(6):80-83.
- [6] Srikanth R, Agrawal R. Mining Sequential Patterns: Generalizations and Performance Improvements[R]. IBM Almaden Research Center,1995.
- [7] Chen Yen-Liang, Chinag Mei-Ching, Kao Ming-Yat. Discovering time-interval sequential in patterns databases [J]. Expert Systems with Application,2003,25(3):343-354.
- [8] 李锡娟,刘 峰,闫娜娜,等. Prefixspan 算法与 CloSpan 算法的分析与研究[J]. 计算机技术与发展,2008,18(1):70-73.
- [9] Tsai Chieh-Yuan, Chien Chun-Ju. An Optimized Classification Model for Time-Interval Sequences[C]//Proceedings of the World Congress on Engineering. London:[s. n.],2010.
- [10] 汪林林,范 军.基于 Prefixspan 的序列模式挖掘改进算法[J].计算机工程,2009,35(23):56-61.
- [11] Pei Jian, Han Jiawei, Mortazavi-Asl B, et al. Mining Sequential Patterns by Pattern-growth: The PrefixSpan Approach[J]. IEEE Trans. on Knowledge and Data Engineering, 2004, 16(11):1424-1440.
- [12] Han Jiawei, Pei Jian, Mortazavi-Asl B, et al. FreeSpan: Frequent Pattern-projected Sequential Pattern Mining[C]//Pro. of 2000 Int'l Conf. on Knowledge Discovery and Data Mining. New York, USA:ACM Press, 2000: 355-359.

(上接第 80 页)

## 3 结束语

介绍了开发嵌入式 Web 服务器环境的搭建和  $\mu$ Clinux 操作系统的移植,嵌入式系统的开发都是基于一定的硬件平台,如果对平台选取的合理,便于开发过程的控制。由于 Linux 和  $\mu$ Clinux 都是技术成熟的操作系统,并且源代码公开,因此课题的开发采用了这二种操作系统,并且详细地介绍了在 Redhat Linux 操作系统上搭建开发嵌入式 Web 服务器环境和在 NETARM3000 硬件平台  $\mu$ Clinux 操作系统的移植。只要完全理解论文的内容,相信可以很容易地设计一个适合多平台嵌入式操作系统的嵌入式 Web 服务器的开发环境。

### 参考文献:

- [1] 丁苍峰,薛宁静.嵌入式 Web 服务器身份认证的解决方案[J].计算机工程与设计,2008,29(24):6255-6257.
- [2] 朱玮玮,杨建明.  $\mu$ Clinux—一种嵌入式 Linux 系统[J].舰船电子工程,2003(4):56-59.
- [3] 沈 可.使用嵌入式 Linux 操作系统进行软件开发的特点及优势[J].电脑开发和应用,2001,14(7):13-15.
- [4] 徐 明.基于 S3C44B0 与  $\mu$ Clinux 构建 Web 服务器[J].可编程控制器和工厂自动化,2005(12):97-100.
- [5] 陈坚华,范 燕,冉立新,等.基于 ARM7TDMI 的  $\mu$ Clinux 移植[J].电子器件,2003,26(1):15-18.
- [6] 英海燕.基于 ARM 的嵌入式 Linux 操作系统移植[J].现代情报学,2005,25(5):155-156.
- [7] 王瑞恒.嵌入式 Internet 在 ARM 上的实现方法和研究[D].哈尔滨:哈尔滨工程大学,2008.
- [8] 郑音飞,胡大可.基于  $\mu$ Clinux 的嵌入式 Web 服务器的实现[J].电子产品世界,2003(7):50-51.
- [9] 赵 颖,周春光,吉 斌,等.嵌入式系统有限资源下的代码移植和优化[J].吉林大学学报(理学版),2007,45(6):995-998.
- [10] 袁爱君.基于 ARM 的嵌入式  $\mu$ Clinux 系统设计与 Web 服务器应用的实现[D].杭州:浙江大学,2003.
- [11] Venkateswaran S. Embedded bootloaders[J]. Embedded Systems Europe, 2004,8(55):32-34.
- [12] 孟 雷,忽海娜. ARM—Linux 嵌入式系统 BootLoader 的配置与移植[J].计算机技术与发展,2008,18(10):204-206.