文章编号:0494-0911(2004)11-0004-04 中图分类号:P208 文献标识码:B

面向 LBS 动态数据管理的 Apriori 算法研究

程新文[1], 吴中忠[1], 陈秀万[2], 张飞舟[2], 吴才聪[2], 陈军峰[2]
(1.中国地质大学 工程学院,湖北 武汉 430074; 2. 北京大学 遥感与地理信息系统研究所,北京 100871)

Research on Apriori Algorithm for LBS Dynamic Data Management

CHENG Xin-wen, WU Zhong-zhong, CHEN Xiu-wan, ZHANG Fei-zhou, WU Cai-cong, CHEN Jun-feng

摘要:LBS(Location Based Services)动态数据管理面临的一个严峻的挑战是需要在一个有限带宽、移动的、不稳定的环境中为用户提供相对稳定的服务。基于关联规则挖掘的思想,试图从业已形成的海量服务数据中发现潜在的规律,以指导服务器有效地完成数据分发工作。但经典的 Apriori 算法并不适合时序数据的处理,而现有的时序关联规则挖掘算法又对服务的关联时间阈值考虑不够,故对经典的 Apriori 算法进行改进,使之适应动态数据管理的需要,从而为解决 LBS 动态数据管理问题提出新的解决思路。

关键词:LBS;动态数据管理;Apriori 算法;时序关联规则挖掘

一、引 言

随着卫星导航、电子地图和无线传输等技术的迅猛发展,基于位置的服务(Location Based Services,简记 LBS)广泛地应用到人们日常生活的各个方面,并且积累了海量的事务信息。

但由于无线通道的带宽有限、不够稳定;加之用户处于移动环境之下,容易掉线,如何使用户在不够稳定的移动环境中使用较连续的服务,即动态数据管理问题,已成为 LBS 发展的瓶颈。当前一种有效的解决办法是利用推动式的数据分发机制[1],即服务器在客户端没有请求的情况下,重复地向客户端广播数据,客户端对服务器发送过来的数据进行缓存。系统借助一定的机制实现缓存内容一致性检查,并对缓存进行更新[2]。如何使服务器能在合适的时间把合适的数据推向客户端,成为解决动态数据管理问题的关键。

关联规则挖掘(Mining Association Rules)是数据挖掘(Data Mining)的重要组成成分,它可以从海量的数据中发现潜在有用的关联或相关关系。Apriori 算法是关联规则挖掘中的一个简洁高效的算法[3],但是经典的 Apriori 算法对于时序数据并不能进行很好的处理。同时由于客观条件的限制,在动态数据管理中必须考虑服务关联时间阈值问题,而现有有关时序关联规则算法的研究并没有对此给予充分的考虑[4,5]。

本文基于关联规则挖掘的思想,对经典的 Apriori 算法进行改进,为 LBS 动态数据管理问题的解决提供一种新的思路。

二、Apriori 算法分析

经典的 Apriori 算法是由 Agrawal 在 1993 年提出的[3],其主要贡献在于提出了频繁集的概念,使得关联规则挖掘不再是个 NP 问题,而成为工程上可以实现的问题[6]。

Apriori 算法输入为事务数据库 D 和最小支持度阈值 min_sup,输出为 D 中的频繁项集 L

算法描述如下[7]:

```
1. L1 = find_frequent_1-itemsets(D);
//产生频繁 1-项集
2. for (k = 2; Lk-1 ≠ φ; k++)
{ //产生频繁 k - 项集,直到候选集为空
3. Ck = apriori_gen(Lk-1, min_sup);
//产生候选 k - 项集
4. for each transaction t∈ D
{ //扫描数据库以计算候选集的支持度
5. Ct = subset(Ck; t);
//得到事务 t 的候选子集
6. for each candidate c ∈ Ct
//判断 c 是否隶属于 Ct
7. c.count++;
//增加相应的支持度计数
8. }
9. Lk = {c ∈ Ck | c.count ≥ min_sup}
//得到频繁 k - 项集
10. }
```

收稿日期: 2004-01-06
作者简介: 程新文(1955-),男,湖北武汉人,教授,研究方向为摄影测量与遥感。

11. return $L = U_k L_k$;//得到频繁项集的集合

procedure apriori_gen(L_{k-1}:频繁 $(k-1)$ - 项集; min_sup:最小支持度)

1. for each itemset $l_1 \in L_{k-1}$

2. for each itemset $l_2 \in L_{k-1}$

3. if $(l_1[1] = l_2[1]) \wedge (l_1[2] = l_2[2]) \wedge ::: \wedge (l_1[k-2] = l_2[k-2]) \wedge (l_1[k-1] < l_2[k-1])$ then {

4. $c = l_1[1] l_2[2] ::: l_1[k-1] l_2[k-1]$//连接步:产生候选

5. if has infrequent_subset(c; L_{k-1}) then

6. delete c;

// 剪枝步:删除非频繁候选

7. else add c to C_k;

8. }

9. return C_k;

procedure has_infrequent_subset(

c: 候选 k - 项集; L_{k-1}: 频繁 $k-1$ - 项集);

1. for each $(k-1)$ - subset s of c

//根据 Apriori 性质判断是否有非频繁子集

2. if $s \in L_{k-1}$ then

3. return TRUE;

4. return FALSE;

经典的 Apriori 算法的流程见图 1。

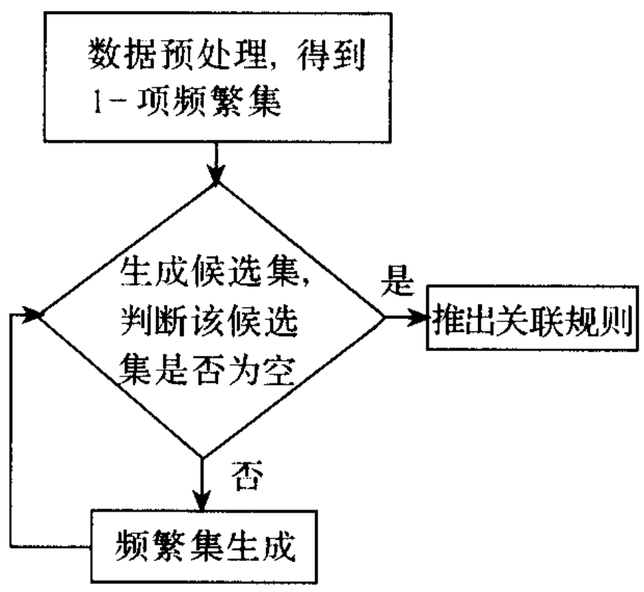

图 1　Apriori 算法流程

经典的 Apriori 算法主要适用于交易集类型的关联规则挖掘,即在一次交易过程中不考虑交易项的先后顺序。而在动态数据管理中,需要面对的都是具有时间顺序的数据,即隐含了先后顺序关系的数据。由于客户端设备的缓存有限,另外若在客户端缓存中存储时间间隔较长的服务数据,将导致用于数据一致性维护和数据更替的费用加大,所以需要考虑服务关联时间阈值,即在多长的时间间隔内认为两项服务是有关联的。经典的 Apriori 算法将一次交易过程中的所有交易看为一项,现有有关时序关联规则挖掘的算法研究[4,5]是将用户在不同时间的交易数据归为一项,均未考虑用户服务之间有关联的时间阈值问题。

三、Apriori 算法的改进

针对动态数据管理的特殊性,本文主要在候选集建立和频繁集生成两方面对 Apriori 算法进行改进,同时在数据的预处理方面考虑服务关联时间阈值问题。

1. 数据预处理

取一时间阈值 tlimit,认为某项服务后 tlimit 之内发生的服务都是与该项服务有关联的。例如用户在得到 a 项服务后的 tlimit 时间内又先后申请了 b 和 c 两项服务,即认为 b, c 两项服务是与 a 有关联的。

时间阈值的选择需要综合考虑 LBS 服务的有效时间、客户端设备缓存大小和数据挖掘目的等因素。在数据预处理时,从数据库中顺序读取每个用户的服务信息,结合时间阈值,将每个用户的服务数据进行分块,形成服务集(形式上的交易集)。在每个交易集中存储的都是用户在时间阈值 tlimit 内得到的相关联服务的类型编码串。例如上面提到的 3 次有关联的服务,就形成了一个交易集$\{a, b, c\}$。

与传统交易集不同的是现在形成的交易集中可以存在重复的项。例如,一个内容为$\{a, b, a, c\}$的交易集,表示的是用户在得到 a 项服务后的 tlimit 时间内又先后申请了 b, a, c 3 项服务。

2. 隶属子集的判断

在候选集生成的剪枝步操作和频繁集生成过程中均需要进行隶属子集的判断。在经典 Apriori 算法中,判断集 A 是否属于集 B,只需考虑集 A 中的每一项是否均出现在集 B 中即可。然而在动态数据管理中,服务发生具有先后顺序,不能颠倒,否则就会得到错误的结论。就隶属子集判断的问题本文提出以下方法。

对 A 中的第 1 项服务,从 B 中第 1 项开始查找最先出现的和其相同的项,返回该项在 B 中的位置,若找不到就返回 0。A 中以后各项在 B 中的查找都是从上次查找返回的位置之后开始,该次查找返回值是 B 中最先出现与其相同的项的位置。若找不到,返回上次查找返回的位置。一旦发现本次返回的位置不是在上次返回的位置之后,即可判断 A 中的服务项在 B 中不是顺序出现,即集 A 并不隶

属于集 B。只有当 A 中各项均通过了判断,才判断 A 隶属于 B。

算法描述如下:

1. $loc_0 = 0$;

//对 A 中第 1 项的查找是从 B 的首项开始

2. for each item a_i in itemset A

3. $loc_i = Find(B, a_i, loc_{i-1})$;

//从上次查找返回的位置之后开始查找

4. if($loc_i \leqslant loc_{i-1}$)

5. return "Not In"

6. return "In"

procedure Find(B:目标集,a:待寻找项,loc:开始的位置)

1. for each item b_i in B behind loc

2. if($b_i = = a$) return i

3. return loc;

该算法确保了集 A 中的项在集 B 也是按原来的顺序出现。例如,集$\{a,b,c\}$隶属于集$\{a,d,b,e,c\}$,却不隶属于$\{b,a,c,e\}$。

3. 候选集的生成

经典的 Apriori 算法在候选集 C_k 的生成过程中选择 C_{k-1}中满足只有一项不同的任意两个交易集,进行连接操作。而考虑时序关系后,选取出来进行连接操作的两个集 A,B,必须满足集 A 的首项与集 B 的尾项不同,而剩余部分对应相同的条件,即集 A 的第 2 项与集 B 的第 1 项相同,…,集 A 的最后一项和集 B 的倒数第 2 项相同。

算法描述如下:

1. if(Left(A,1) = = Right(B,1))

//判断 A 的首项和 B 的末项是否相同

2. Not Join

3. else

4. if(Right(A, size(A) - 1)! = Left(B, size(B) - 1))

//判断 A 的首项之后的部分和 B 的末项之前的部分是否对应相同

5. Not Join

6. else

7. Join

例如:$\{a,b,d,e\}$与$\{b,d,e,f\}$可以进行连接操作,而$\{a,b,d,e\}$与$\{a,b,c,e\}$不可以进行连接操作。在进行剪枝步操作时,只需要依次剔除连接步产生候选集的中间一项,即可生成该候选集的所有需要判断的子集。例如对于$\{a,b,d,e\}$与$\{b,d,e,f\}$连接产生的候选集$\{a,b,d,e,f\}$只需判断$\{a,d,e,f\}$,$\{a,b,e,f\}$和$\{a,b,d,f\}$是否在频繁集中出现就可完成剪枝步操作。

4. 关联规则的推出

根据经典 Apriori 算法计算,算出频繁集之后,就是根据最小可信度进行判断,得出结论。对于频繁集,需要分别计算不同顺序的结果,例如频繁集$\{a,b\}$就需要分别计算 $a \rightarrow b$ 和 $b \rightarrow a$ 的可信度,并与最小可信度进行比较。而在动态数据管理中就只用按从前向后的顺序进行计算。在上述的例子中只需要计算 $a \rightarrow b$ 的可信度,并与最小可信度进行比较即可。

四、实 验

1. 实验数据的生成

为了验证本文对 Apriori 算法改进的有效性,本文生成了一系列的随机模拟试验数据,对算法进行评价。模拟数据的生成主要参考了以下几个参数(见表 1)。

表 1 模拟数据的参数

参数	注释
P	总用户数
S	每个用户最大服务量
N	现有的服务类型
tlimit	时间阈值
min_sup	最小支持度
min_conf	最小可信度

本文为了使试验结果具有可比性,将时间阈值取为 1 h,服务类型取为 6 种,每个用户的最大服务量取为 300 次。这样随着总用户数的取值变化,就可以产生不同量级的模拟数据。

2. 关联规则挖掘的过程

依次读入每个用户的服务信息,按照设定的时间阈值(取为 1 h),将某项服务发生后时间阈值 tlimit 范围之内发生的服务,进行合并,作为每个用户服务集中的一项。同时统计每种类型的服务发生的次数,结合最小支持度 min_sup 参数产生 1 - 项频繁集。

依照改进后的 Apriori 算法进行计算,并结合最小支持度 min_sup 得出频繁集的结果。最后计算频繁集中各项的可信度,与 min_conf 进行比较,得出试验数据中有效关联的结果集。

3. 规模增长试验

通过赋予测试程序不同的总用户数产生不同规模的测试数据,进行规模增长试验(其中,最小支持

度取为 10%,最小可信度取为 30%)。试验结果见图 2。

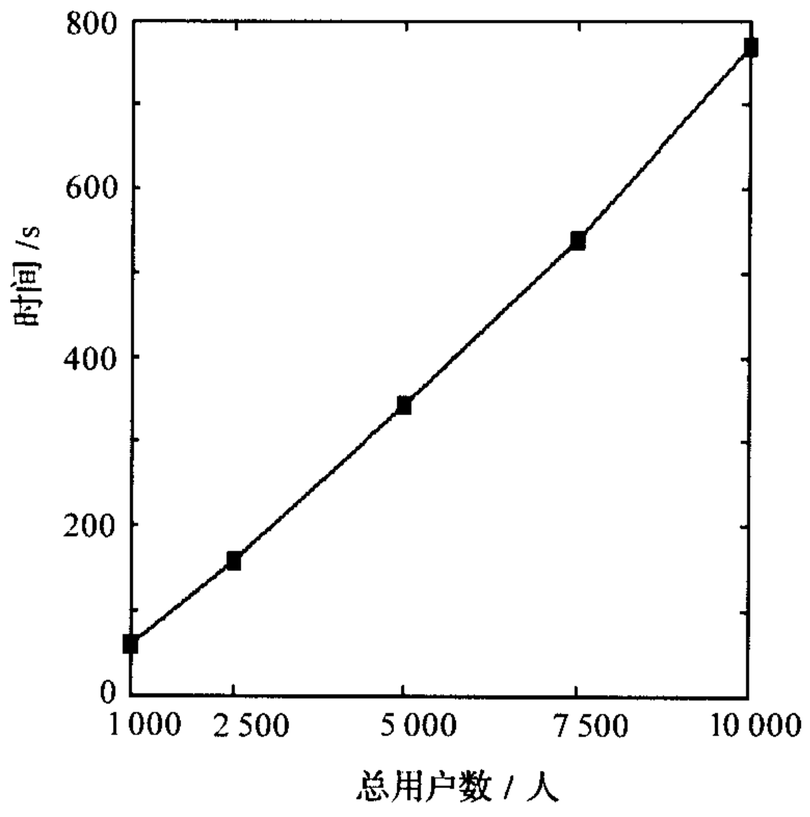

图 2　规模增长试验结果

如图 2 所示,改进后的 Apriori 算法执行时间随着问题规模的增长以线性规律增长。

五、结论与展望

本文对经典的 Apriori 算法进行了改进,使之适合时序数据的处理,同时对动态数据管理中存在的服务关联时间阈值问题给予了考虑。为 LBS 动态数据管理问题的解决提出了一种新的解决办法。

在 LBS 系统中,用户得到的服务不仅和时间有关,而且与空间分布有关联。将空间要素加入关联规则挖掘的过程中,可能发现更有效的规则。这将是下一步研究的重点。

参考文献:

[1] AGRAWAL R, CHRYSANTHIS P K. Efficient Data Dissemination to Mobile Clients in E-Commerce Applications[A]. Third International Workshop on Advanced Issues of E-Commerce and Web-Based Information Systems, 2001[C].[s.l.]:[s.n.],2001. 58-65.

[2] CHAN B Y,SI A,LEONG H V. Cache Management for Mobile Databases: Design and Evaluation[A]. Proceedings of 14th International Conference on Data Engineering[C].[s.l.]:[s.n.],1998.54-63.

[3] AGRAWAL R, IMIELINSKI T, SWAMI A N. Mining Association Rules between Sets of Items in Large Databases[A]. Proc. of the 1993 Int'l Conf. on Management of Data[C].[s.l.]:[s.n.], 1993. 207-216.

[4] LI Ying-jiu. Discovering Calendar-based Temporal Association Rules[A]. Proceedings of Eighth International Symposium on Temporal Representation and Reasoning, 2001[C].[s.l.]:[s.n.],2001.111-118.

[5] AGRAWAL R, SRIKANT R. Mining Sequential Patterns[A]. Proceedings of the Eleventh International Conference on Data Engineering 1995[C]. [s.l.]:[s.n.],1995.3-14.

[6] 段云峰,杨凤年,李剑威,等. 移动通信业务中的业务关联规则挖掘[J]. 电信科学,2001,(11): 17-19.

[7] 褚玉林,王向阳, 彭宁嵩. 利用 C++ 标准类挖掘关联规则[J]. 洛阳工学院学报,2002,23(3): 67-69.

[8] WU Shiow-yank, WU Kun-ta. Dynamic Data Management for Location Based Services in Mobile Environments [A]. Proceedings of Seventh International Database Engineering and Applications Symposium 2003[C].[s.l.]:[s.n.],2003. 180-189.

(上接第 3 页)

这些国家和地区的经济和社会发展。

由此可见,应用 ASTER 提取 DEM 的应用前景非常广阔,国内的相关研究机构和生产单位有必要尽快开发相关的应用软件和开展积极的应用。

参考文献:

[1] ERSDAC. ASTER Level 1 Data Products Specification (GDS version)[M].[s.l.]:[s.n.],2001.

[2] PARTOVI A. Suitability Study of ASTER Data Geometry to Digitize Contour Lines in ILWISSAR[D]. [s.l.]:[s.n.],2003.

[3] 张钧屏, 方艾里, 万志龙. 对地观测与对空监视[M]. 北京:科学出版社,2001.

[4] HURTADO J M. Extraction of a Digital Elevation Model from ASTER Level 1A Stereo Imagery Using PCI Geomatica OrthoEngine[M].[s.l.]:[s.n.],2002.

[5] TOUTIN T, CHENG P, DEM Generation with ASTER Stereo Data[J]. Earth Observation Magazine, 2001, 10 (6): 10-13.

[6] TOUTIN T. 3D Topographic Mapping with ASTER Stereo Data in Rugged Topography[J]. IEEE Transactions on Geoscience and Remote Sensing, 2002, 40 (10): 2 241-2 247.