分布式关联规则挖掘研究

王治和, 景永霞, 杜 辉

(西北师范大学数学与信息科学学院, 甘肃 兰州 730070)

[摘要] 针对分布式关联规则挖掘算法 FDM 可能造成频繁项集丢失的缺点, 提出了一种改进的分布式环境下关联规则挖掘算法. 该算法采用全局-局部通信模式, 通过对候选项集建立对应的频繁标记, 把频繁标记和频繁项集的支持计数作为各局部站点和全局站点之间的传输内容. 该算法不仅保证了数据挖掘结果的完整性和正确性, 同时也减少了站点间的通讯量.

[关键词] 分布式环境, 数据挖掘, 关联规则, FDM

[中图分类号] TP301. 6 [文献标识码] A [文章编号] 1001-4616(2010) 04-0114-05

Study of Distributed Association Rule Mining

Wang Zhihe, Jing Yongxia, Du Hui

(College of Mathematics and Information Science, Northwest Normal University, Lanzhou 730070, China)

Abstract Aiming at the shortcoming of FDM algorithm that may cause to lose the frequent itemsets, an improved algorithm of distributed association rules mining (IADM) based on distributed environment was presented in this paper. This algorithm uses global-local communication mode, builds a frequent mark for each candidate itemset, transmits frequent marks and support counts of frequent itemsets between each local site and global site. It not only guarantees the integrity and accuracy of the data mining results, but also cuts down the communication overhead among sites.

Key words distributed environment; data mining; association rules; FDM

分布式数据挖掘是从多个在物理位置上分布的数据库中进行数据挖掘的过程, 现已成为数据挖掘领域一个倍受关注的热点问题. 在分布式数据环境下进行关联规则挖掘, 其主要工作是汇总各局部站点上的局部频繁项集, 从中找出满足全局最小支持度的全局频繁项集, 从而通过它们生成相应的关联规则. 现有的分布式关联规则挖掘算法大多都是基于 Apriori算法思想的, 如 CD 算法和 FDM 算法就是在 Apriori算法的基础上实现的分布式关联规则挖掘算法.

CD算法是最典型的 Apriori算法的并行化, 但由于 CD算法不管候选项集是否频繁, 站点之间都传递候选项集的信息, 严重浪费了通讯资源的带宽, 导致在候选项集过多的时候会在很大程度上影响算法的执行效率, 并造成通讯量的过载. FDM 算法是 Cheung D W 为了减少各个分区间的通讯量而提出的, 但该算法有可能丢失频繁项集.

本文算法就是针对 FDM 算法在计算分布式数据库的频繁项目集时, 可能导致全局频繁项目集丢失这个问题而提出的.

1 FDM 算法的不足

为了减少各个分区间的通讯量, Cheung D W 在 1996年提出了 FDM (Fast Distributed association rules Mining)算法[1, 2], FDM 算法是一种在无共享计算机间实现的并行 Apriori算法, 每个站点都有它自己独立的数据库且独立地进行数据库扫描. FDM 算法引入了分布式剪枝技术, 它基于以下定理.

收稿日期: 2010-06-10

基金项目: 西北师范大学 2007~ 2010年度重点学科基金 (2007C04).

通讯联系人: 王治和, 教授, 研究方向: 数据挖掘. E-mail: wangzh@ nwnu. edu. cn

定理 **1** 若 X 是全局频繁项目集, 则必定存在一个站点 S_i ($1 \leqslant i \leqslant n$), 使得 X 及其子集在站点 S_i 上是局部频繁的.

FDM 算法的基本思想: 首先在每个迭代过程的开始, 每个站点并行运行类 Apriori算法找出其局部频繁项集, 之后各站点同步一次来完成各局部站点之间的局部频繁项集支持计数的交换, 用以计算出全局频繁项集, 并把结果广播到各局部站点, 由其生成候选项集进入下一次迭代过程, 直到所有站点的候选项集为空时算法结束. 在生成候选项集和同步通信的过程中, 需充分利用局部频繁项集和全局频繁项集之间的关系, 对需要局部处理的数据事先进行过滤, 以减少站点间的信息传输量.

下面给出带有结点内部剪枝技术的 FDM 算法所采取的主要技术: 生成候选项集、候选数据集的局部剪枝和全局剪枝、合计数轮流检测. 其主要过程如下:

(1) 候选项集的生成. 根据在局部数据库 DB_i 经过 $k-1$ 次迭代生成的全局大的数据集 LG_{ik-1}, 利用公式 $C_{ik} = \text{apriori\_gen}(LG_{ik-1})$ 生成 C_{ik};

(2) 本地剪枝. 对于每一个数据集 $X \in C_{ik}$, 扫描每个站点 S_i 上的局部数据库 DB_i, 以计算本地支持度 $X.\text{support}_i$, 如果 X 在局部数据库 DB_i 是局部大的, 那么将其加入到站点 S_i 的局部频繁 $k-$ 项目集的集合 L_{ik} 中;

(3) 广播挖掘结果. 将站点 S_i 的局部频繁 $k-$ 项目集的集合 L_{ik} 向其他站点广播, 最终合成全局频繁 $k-$ 项集 L_k;

(4) 计算 LG_{ik}. 站点 S_i 根据 L_k 的信息, 对 L_{ik} 进行剪枝, 根据 $LG_{ik} = L_{ik} \cap L_k$ 得出 LG_{ik}.

对 FDM 算法的工作流程进行仔细分析之后, 我们可以看到该算法的不足在于[3,4]:

(1) 在每次迭代过程中, 各局站点 S_i 都会扫描本地数据库中的原始数据. 产生频繁 $k-$ 项集就需要 k 次扫描数据库, 尤其是产生长频繁项集时扫描次数更多. 而在后续的扫描时, 这些原始的数据源中会存在很多非频繁项集, 使得每次循环计算候选项集支持计数时都会检查这些非频繁项集. 而且, 随着支持度阈值的提高, 非频繁项集的个数会成比例增长. 随着 k值的增加, 候选项集的集合 C_k 的长度增加, 如果仍然需要每次扫描整个数据库来计算候选项集的支持计数的话, 效率会变低. 该算法没有充分利用计算所得的频繁项集结果去修剪数据库以减小其大小及平均事务记录长度.

(2) 在 FDM 算法中, 各局部站点 S_i 在进行本地剪枝时对于每一个数据集 $X \in C_{ik}$ 扫描每一个局部数据库 DB_i, 以计算本地支持度 $X.\text{support}_i$, 只有 $X.\text{support}_i \geqslant \min\_\text{sup}$ 时才将其加入到 DB_i 的局部频繁 $k-$ 项集的集合 L_{ik} 中, 并广播它们的支持计数, 对于不是局部频繁项集的候选项集仅作简单的剪枝处理, 这样就有可能造成某些本应该是频繁项集的项集由于忽略某些站点上的支持计数而被认为是非频繁的.

(3) 在各局部站点 S_i 上, 局部候选频繁 $k-$ 项集 C_{ik} 的产生是根据本地的全局大频繁项集 LG_{ik-1} 利用公式 $C_{ik} = \text{apriori\_gen}(LG_{ik-1})$ 来计算得到的, 而有些全局频繁 $(k-1)-$ 项集在有些站点是不频繁的 (即, $X \in L_{k-1}$, 而在站点 S_i 上, $X \notin LG_{k-1}$), 因而在该站点产生候选频繁 $k-$ 项集 C_{ik} 时, 这些全局频繁 $(k-1)-$ 项集将不被考虑, 这样也可能会导致局部候选 $k-$ 项集的集合 C_{ik} 中元素的缺失, 从而影响最终全局频繁项集的完整性.

2 现有改进算法

综合上面发现的 FDM 算法不足, 文献[5-7] 描述的算法都是提出了新增频繁项目集 (新增频繁项目集是指分布式系统中, 在其他站点是频繁项集而在本地不是频繁项集的项目集) 的概念, 用来弥补 FDM 算法在关联规则挖掘过程中有可能造成频繁项集丢失的不足.

文献[5] 提出了一个关联规则分布式挖掘系统 DAM INER 的实现方案, 它是基于分布式数据库基础上的全局-局部站点的数据挖掘模式, 该系统中用到的核心算法 ARDM 是利用新增频繁项目集支持计数的传递来保证频繁项集的完整性. ARDM 由局部站点和全局站点协同完成关联规则的分布式挖掘任务. 局部站点的挖掘任务是: 为全局站点提供局部频繁项目集 L_{ik}, 同时根据 L_{ik} 提取局部关联规则并保存在局部规则库中; 全局站点的任务是: 综合所有局部站点的 L_{ik}, 从中筛选出全局频繁项目集 L_k, 生成全局关联规则且保存在全局规则库中, 以供局部站点访问. 算法将频繁项集本身及其支持计数作为各局部站点和全局站点之间的传输内容.

文献[6]提到的改进方案也是基于新增频繁项目集支持计数的传递,只是在接收到各局部站点 S_i 上的局部频繁项目集 L_{ik} 之后,直接将得到的候选数据集广播到各局部站点 S_i,由各局部站点独立地计算出新增频繁项目集和局部支持计数,然后发送至全局站点.

文献[7]提出了一种折衷方案,以增加通讯量为代价来防止由于某些站点没有广播非频繁候选项集的支持计数而造成的频繁项集丢失问题,该算法仍然采用迭代的方法求解全局频繁项目集 L_k.

3　分布式关联规则挖掘算法 *IADM*

分布式关联规则挖掘,时间开销主要体现在:频繁项目集的确定和网络的通讯量.

本文提出的分布式关联规则挖掘算法 IADM采用了全局 - 局部的通信模式,该算法是针对 FDM算法在计算分布式数据库 DB 的频繁项目集时,可能导致全局频繁项目集的丢失这个问题而提出的. IADM算法在文献[5]提出的 ARDM算法的基础上作了改进,进一步减少了各站点之间的通讯量,并且避免了频繁项目集的丢失,保证了数据挖掘结果的完整性和正确性.

IADM算法主要是通过对候选项集建立对应的频繁标记,其在全局站点的初始值都为 false,同时保证各局部站点和全局站点之间的候选 k - 项集顺序的一致性,然后把频繁标记和频繁项集的支持计数作为各局部站点和全局站点之间的传输内容,显然,通讯量要小于文献[5]提出的 ARDM算法传输频繁项集本身及其支持计数的通讯量,这样就可以降低各站点间的通讯量.

3.1　算法思想

各个局部站点 S_i 扫描本地数据库 DB_i 产生候选 k - 项集,并根据本地最小支持计数(本地最小支持计数 = 本地数据库记录数 × min_sup)判断这些项集是否是频繁项集,并为它们打上频繁标记 $Freq_{ik}$.然后,将频繁标记 $Freq_{ik}$ 发送至全局站点,全局站点根据所有站点的频繁标记进行逻辑或运算求出全局频繁 k - 项集的候选集合 L'_k 的频繁标记 $Freq'_k = \vee\ Freq_{ik}\ (i = 1, 2, \cdots, n)$,然后将 L'_k 的频繁标记 $Freq'_k$ 发送到各个局部站点,各个局部站点 S_i 收到全局站点的数据后,将 L'_k 中项目集的支持计数发送到全局站点,由全局站点计算每个项集的全局支持合计数,并最终确定全局频繁项目集 L_k 及其频繁标记 $Freq_k$,利用公式 C_{k+1} = apriori_gen(L_k) 计算候选 $k+1$项集 C_{k+1}.

如果 $C_{k+1} \neq \phi$,将 L_k 中项集的频繁标记 $Freq_k$ 发送到各个局部站点,进行下一轮挖掘;

如果 $C_{k+1} = \phi$,通知各个局部站点挖掘算法结束,各局部站点删除局部站点项集文档中与本次挖掘相关的信息.

各个局部站点 S_i 根据 $Freq_k$ 获取 L_k 的值,并独立地计算出候选 $(k+1)$ - 项集的集合 C_{k+1}.

最后根据在全局站点求得的全局频繁项目集和用户给定的最小置信度(min_conf),生成全局关联规则,并把它保存到全局知识库中,从而完成关联规则的整个挖掘过程.

3.2　算法详细描述

在 IADM算法中,每个频繁项集有以下几个属性,其中 Item存储项集的标识, count存储该项集的支持计数, freq用来标识该项集是否频繁,若频繁, freq = true;反之, freq = false.

```
string Item;          //项集标识
int count;            //支持计数
boolean freq;         //频繁标记
```

算法的基本流程如图 1所示.

算法 IADM的详细过程:

(1) 全局站点接到用户数据挖掘请求(所带参数为站点信息)后,根据站点信息,对 C_1 进行排序.

(2) 根据各个站点的 URL启动多个线程分别调用各局部站点 $S_i\ (i = 1, 2, \cdots, n)$ 的 Web服务(参数为 k 和 $Freq_{k-1}$).

(3) 在各个局部站点 S_i 上,

① $k = 1$时,对本地候选 1 - 项集 C_{i1} 按照与全局站点相同的排序算法进行排序,使其得到的局部候选 1 - 项集 C_{i1} 与全局站点的全局候选 1 - 项集 C_1 项集顺序一致;

② $k > 1$时,根据从全局站点发送的频繁标记 $Freq_{k-1}$ 的值得到 L_{k-1},然后利用公式 C_k = apriori

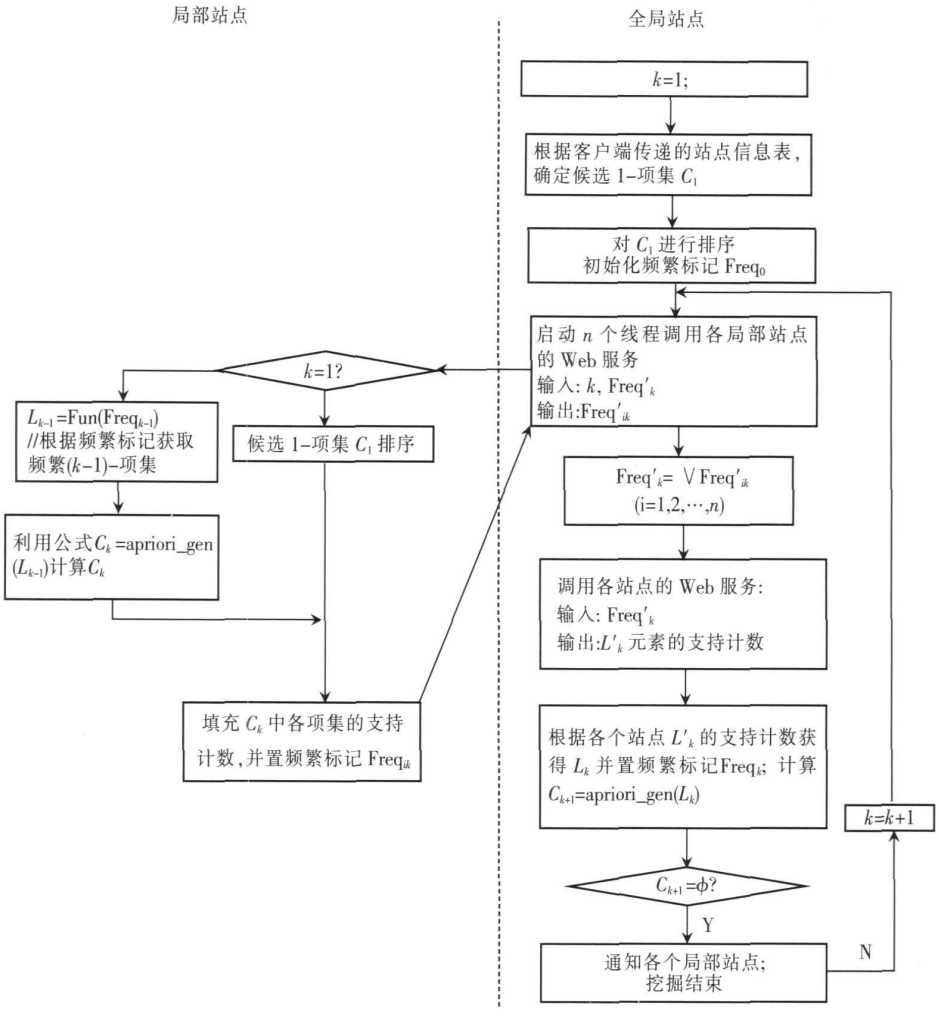

图 1 算法 IADM 的流程

Fig.1 Flow chart of the algorithm IADM

gen(L_{k-1}) 计算得到候选 $k-$ 项集 C_k.

(4) 在各个局部站点 S_i 上, 扫描各个局部数据库 DB_i 中的所有事务, 对 C_k 中每个项集的出现次数进行计数 (将其填写在 count域中), 并置频繁标记 $Freq_{ik}$ 用以标记该项集在本地是否频繁.

(5) 将各个局部站点 S_i 的候选 $k-$ 项集的频繁标记 $Freq_{ik}$ 传回给全局站点, 在全局站点对各个局部站点发送的频繁标记 $Freq_{ik}$ (元素为 boolean类型) 对应位进行或运算 (注: 全局站点上频繁标记初始值都为 false), 即, 只要该项集在某一个站点是频繁的, 那么最终运算结果中它所对应的值为 true

(6) 全局站点启动多个线程分别调用各个站点上的 Web服务 (参数为 $Freq'_k$), 得到各个站点候选频繁 $k-$ 项集 L'_k中各项集的支持计数.

(7) 全局站点进行计算, 得到候选频繁 $k-$ 项集 L'_k中各个项集的支持合计数, 若项集支持合计数 $\geqslant$ min_sup* |DB|, 则将该项集加入全局频繁 $k-$ 项集 L_k, 然后置 $Freq_k$ = true

(8) 利用 C_{k+1} = apriori_gen(L_k) 计算得到候选 $k+1$项集 C_{k+1},

如果 $C_{k+1} \neq \Phi$, 则 $k = k+1$, 转 (2) 执行;

如果 $C_{k+1} = \Phi$, 挖掘算法结束.

3.3 算法的改进之处

本文所提出的算法 IADM 的改进之处:

(1) 避免某些频繁项集由于忽略某些站点的支持计数而被误认为是非频繁项集的情况.

在 IADM 算法中, 各局部站点 S_i 对于每一个数据集 $X \in C_{ik}$, 扫描每一个局部数据库 DB_i, 计算该站点上的所有候选项集的局部支持计数, 并对所有项集打上频繁标记, 然后将各局部站点 S_i 上的频繁标记发

往全局站点进行计算,得出所有局部站点频繁项集的并集 ——全局频繁项集的候选项集,然后将其频繁标记发送到各局部站点,这样全局频繁项集的候选项集在任何一个站点的支持计数都可以传送到全局站点,从而保证了候选项集支持计数的完整性,其正确性由定理 1保证.

(2) 避免局部候选频繁 k – 项目集 C_{ik} 元素的丢失.

在 IADM 算法中,各局部站点 S_i 候选频繁 k – 项目集 C_k 的产生不再是根据本地的全局大频繁项集 LG_{ik-1},而是根据全局频繁 $(k-1)$ – 项集 L_{k-1} 利用公式 $C_k = \text{apriori gen}(L_{k-1})$ 计算得到,这样就可以避免由于有些全局频繁 $(k-1)$ – 项集在有些站点是不频繁的,而使得在这些站点产生候选频繁 k – 项集 C_{ik} 时,这些全局频繁 $(k-1)$ – 项集不被考虑而导致的局部候选频繁 k – 项目集 C_{ik} 中元素的缺失.

(3) 降低了时间开销.

在分布式关联规则挖掘算法中,时间开销主要体现在两方面:

① 频繁项目集的确定. IADM 算法在判断某项集是否为全局候选项集时,只是进行简单的逻辑或运算,减少了全局候选项集的计算量,从而减少了确定频繁项目集的计算量.

② 网络的通讯量. IADM 算法是在已有算法 ARDM 的基础上改进的,在挖掘频繁 k– 项集时,各局部站点和全局站点之间只传送一次各局部站点频繁项集的支持计数,其余几次通讯都只传送候选项集的频繁标记,这样就可以减少站点之间的通讯量.

4 结束语

本文针对 FDM 算法有可能丢失频繁项集的不足,在已有改进算法的基础上,提出了分布式关联规则挖掘算法 IADM,该算法在保证频繁项集完整性的前提下,把频繁标记和频繁项集的支持计数作为各局部站点和全局站点之间的传输内容,减少了各站点之间的通讯量,从而提高了算法的效率.

[参考文献]

[1] Cheung D W. Efficient mining of association rules in distributed databases[J]. IEEE Transactions on Knowledge & Data Engineering, 1996, 8(6): 910-921.

[2] Schuster A, Wolff R, Trock D. A highr performance distributed algorithm for mining association rules[J]. Knowledge and Information Systems, 2004(8): 210-221.

[3] 刘群. 基于 CORBA 的分布式关联规则挖掘系统的研究和实现[D]. 青岛: 山东科技大学信息科学与工程学院, 2005: 15-17.

[4] 张迎春. 带补偿的快速分布式关联规则挖掘算法的研究[D]. 青岛: 山东科技大学信息科学与工程学院, 2006: 23-24.

[5] 赵斌, 吉根林. 分布式系统中关联规则挖掘研究[J]. 小型微型计算机系统, 2003(12): 2270-2271.

[6] 陈涛, 张玮. 一个改进的并行关联规则算法研究[J]. 计算机技术与发展, 2007(1): 139-141.

[7] 段红勇. 分布式关联规则算法和分布式决策树算法的对比研究[D]. 长沙: 中南林学院电子与信息工程学院, 2005: 55.

[责任编辑: 丁 蓉]