分布式并行关联规则挖掘算法研究

王智钢[1,2] 王池社[1,2] 马青霞[1]
[1](金陵科技学院信息技术学院 江苏 南京 211169)
[2](江苏省信息分析工程实验室 江苏 南京 211169)

摘 要 关联规则挖掘算法 FP-Growth 虽然效率比 Apriori 要快一个数量级,但存在频繁模式树可能过大而内存无法容纳和数据挖掘过程串行处理等两大缺点。提出一种分布式并行关联规则挖掘算法,该算法针对分布式应用数据架构,不需要产生全局 FP-tree,避免全局 FP-tree 可能过大而内存无法容纳的问题,算法在各个主要步骤上都实现了并行处理。算法测试结果和分析表明,与传统的关联规则挖掘算法 FP-Growth 相比,该算法通过多节点分布式并行处理显著提高了执行效率和处理能力。

关键词 数据挖掘 关联规则 频繁模式 并行算法

中图分类号 TP311 **文献标识码** A DOI:10.3969/j.issn.1000-386x.2013.10.031

RESEARCH ON DISTRIBUTED PARALLEL ASSOCIATION RULE MINING

Wang Zhigang[1,2] Wang Chishe[1,2] Ma Qingxia[1]
[1](*School of IT, Jinling Institute of Technology, Nanjing 211169, Jiangsu, China*)
[2](*Information Analysis Engineering Laboratory of Jiangsu Province, Nanjing 211169, Jiangsu, China*)

Abstract In association rule mining, though the FP-Growth algorithm is approximately one order of magnitude faster than the Apriori algorithm, but it has two disadvantages: the first is that its frequent pattern tree may be too big to be created in the memory; the second is its serial processing approach. In this paper we propose a kind of distributed parallel association rule mining algorithm. It is for the distributed applied data framework, does not need to create the global FP-tree so avoids the problem of too big the global FP tree that fills the memory to excess. In all its principal steps the algorithm achieves parallel processing. Test result and analysis of the algorithm show that compared with conventional association rule mining algorithm FP-Growth, this one significantly improves the executing efficiency and the processing ability by multi-node distributed parallel processing.

Keywords Data mining Association rule Frequent pattern Parallel algorithm

0 引 言

关联规则是由 Agrawal 等于 1993 年在文献[1]中首先提出的一个重要的数据挖掘研究课题,Apriori 算法是 R. Agrawal 和 R. Srikant 于 1994 年提出的布尔型关联规则原创性挖掘算法[2],在 Apriori 算法基础上还派生出了很多改进算法,但 Apriori 类算法有如下不足[3]:(1) 算法必须耗费大量时间处理规模巨大的候选项目集;(2) 算法必须多次重复扫描数据库,对候选项目集进行模式匹配。针对 Apriori 类算法的不足,Han 等人提出了关联规则挖掘算法 FP-Growth[4],研究表明,FP-Growth 算法的效率约比 Apriori 要快一个数量级。

近年来,随着社会信息化的加速,各种海量数据应用不断出现,虽然 FP-Growth 算法在效率上比 Apriori 类算法要高,但仍然难以胜任针对海量数据的关联规则挖掘任务,一是可能出现生成的频繁模式树过大而内存无法容纳的问题,二是串行处理所耗费的时间也难以让人满意。另外,许多基于分布式应用数据架构的关联规则挖掘问题,为避免大量数据的传输和转储,也对分布式的数据挖掘提出了现实需求。

为实现分布式或并行关联规则挖掘,文献[5]提出了数据划分,文献[6,7]采用多线程内存共享,文献[8]在得到全局 FP-tree 之后将其分解为子树进行并行处理,文献[9]谈到由于 FP-Growth 算法挖掘每个条件模式库的过程是彼此独立的,故可将每个条件模式库的挖掘看成一个子任务,总的频繁模式挖掘任务可被划分为数目与频繁项目数相等的若干个子任务,然后将这些子任务分配给计算机集群中的各个节点分别执行。

本文提出一种分布式并行关联规则挖掘算法,为便于表述,称之为 P-FP-Growth 算法,该算法针对分布式的应用数据架构,不需要产生全局 FP-tree,在算法的各个主要步骤上都实现了分布式并行处理。

收稿日期:2012-06-28。江苏省现代教育技术研究项目(2011-R-19470);江苏省高校自然科学基金项目(11KJD520006)。**王智钢**,副教授,主研领域:数据挖掘,网络信息系统,软件工程。**王池社**,副教授。**马青霞**,硕士。

1 问题描述

设 $I=\{i_1,i_2,\cdots,i_n\}$ 是所有项目的集合。事务 T 是一个由 I 中若干个项目组成的集合，$T\subseteq I$，每一个事务具有唯一的事务标志。D 是由多个事务组成的数据库，且 $D=\sum_{j=1}^{m}d_j$，d_j 分布在不同的存储节点 M_j 上，$C_k(k=1,2,\cdots,p)$ 是具有很强计算能力的一组计算节点。在物理上，存储节点和计算节点可以完全重合、部分重合或者完全不同。现需充分利用计算节点的计算能力，尽可能快地找出全局事务数据库 D 中支持度大于给定阈值的频繁模式，并进而得出关联规则。

2 算法描述

设最小支持度计数为 min_sup，中心计算节点名为 C_c，ParDo( )表示并行执行，Do( )表示中心计算节点 C_c 的单独执行，现将分布式并行关联规则挖掘算法 P-FP-Growth 描述如下：

(1) ParDo(Count(d_1),Count(d_2),…, Count(d_m))

各存储节点 M_j 分别对 d_j 中的项目进行计数

(2) ParDo(Send(1),Send(2),…,Send(m))

各存储节点 M_j 将计数结果发送到 C_c

(3) Do(Sum_sort_del_assign_send( ))

C_c 完成项目计数汇总，事务中项目按计数降序排序并删除计数小于最小支持度计数 min_sup 的项目，为项目分配计算节点，最终得到带计算节点分配信息的全局 1 - 项频集 L，再将 L 分发到各节点。

(4) ParDo(Build_T(d_1), Build_T(d_2), …, Build_T(d_m))

各存储节点 M_j 根据表 L，为 d_j 建立局部 FP-tree，建立的方法同传统的 FP-Growth 算法。

(5) ParDo(Send(1),Send(2),…,Send(m))

各存储节点 M_j 将项目和它们在局部 FP-tree 中的条件模式基发送到 L 中指定的计算节点。

(6) ParDo(Mining(1), Mining(2),…, Mining(p))

各计算节点 C_k 按项目汇总得到各个项目的条件模式库，并进行频繁模式挖掘。

(7) ParDo(Send(1),Send(2),…,Send(p))

各计算节点 C_k 将得出的频繁模式汇集到中心计算节点 C_c

(8) Do(Get_Rule( ))

中心计算节点 C_c 汇总频繁模式，并最终得出全局关联规则集合。

3 对算法描述的说明

算法描述中第(3)步，为了后续能按照项目将频繁模式挖掘任务分配到各个计算节点实现并行挖掘，中心计算节点需要为全局 1 - 项频集 L 中的每个项目分配计算节点，并将分配结果记录在 L 中，最终得到的 L 如表 1 所示。

表 1 带计算节点分配信息的全局 1 - 项频集 L

项目	计数	计算节点
…	…	…
I_7	3745	C_2
I_{12}	2506	C_6
I_{35}	1985	C_8
…	…	…

表 1 中第三行表示 I_7 共出现了 3 745 次，将被分配到计算节点 C_2 进行频繁模式挖掘。

为 L 中的项目分配计算节点的策略可以有很多种，既可事先静态分配，适合于远程分布式计算，也可在通信非常便利的情况下，采用动态分配。

静态分配策略可以是按照项目计数顺序依次分配。例如：设项目按照计数降序排列后依次称为第 1,2,…,n 个，计算节点共有 p 个，依次名为 $C_1,\cdots,Cp$，则满足 $n \bmod p=i$ 的第 n 个项目分配给 C_i+1。

动态分配策略可以是：先从 L 的末端开始，为每个计算节点分配一个项目的频繁模式挖掘任务，然后在某计算节点完成任务之后再从 L 中由后往前分配一个项目挖掘任务。

算法描述中第(4)步，是在全局 1 - 项频集 L 约束下，为各个局部数据库建立局部 FP-tree，建立的方法同传统 FP-Growth 算法，主要就是对数据库中的事务进行三步操作：① 过滤掉 L 中没有的项目；② 将剩下的项目按照 L 中的计数降序排列；③ 将该事务加入本节点的 FP-tree。本步骤处理结束后，可得到多个在全局 1 - 项频集 L 约束下的局部 FP-tree 和 Header Table。

算法描述中第(6)步，各计算节点按项目汇总各个项目的条件模式库并进行频繁模式挖掘时，如果某个项目的条件模式库异常大，远远大于其他项目，可递归调用本算法，对该项目的频繁模式挖掘任务进一步进行分解，然后再进行并行处理。

4 算法设计说明

FP-Growth 算法存在频繁模式树可能过大而内存无法容纳和数据挖掘过程串行处理等两大缺点。

虽然 FP-Tree 是一种压缩数据结构，但当全局数据库非常大时，生成的全局 FP-Tree 仍然会很大，甚至可能内存无法容纳。可以采用多个局部 FP-Tree 来代替全局 FP-Tree，限于篇幅，下面仅简要说明采用多个局部 FP-Tree 和采用一个全局 FP-Tree 是等效的。

(1) 设有事务 T，显然只要是在相同的全局 1 - 项频集 L 约束下，对 T 中的项目进行过滤和排序后得到的事务 T' 是唯一的。

(2) 对于 T'，在加入到 FP-Tree 中时，不管是采用全局 FP-Tree 还是局部 FP-Tree，它所对应形成的一条 FP-Tree 路径 r 必定是相同的。原因很简单，因为事务在 FP-Tree 上对应的路径仅由事务包含的项目和项目的排列次序决定。

(3) 对某一条既定的 FP-Tree 路径 r 而言，不管它是在全局 FP-Tree 还是局部 FP-Tree 上，r 上各节点的前缀路径是相同的。

各节点的前缀路径实际上就是各节点的条件模式基。由(1)、(2)、(3)可知，因为对于一个事务 T 而言，只要是在相同的全局 1 - 项频集 L 约束下，不管是采用全局 FP-Tree 还是局部 FP-Tree，它为事务中项目给出条件模式基时是一样的，没有丝

毫区别,所以全局 FP-Tree 和多个局部 FP-Tree 是等效的。

基于自然的分布式数据架构,或者通过数据划分,可在事务项目计数、过滤和排序等事务预处理环节实现并行处理。通过将每个条件模式库的挖掘看成独立子任务,可在频繁模式挖掘环节实现并行处理。

在前人的基础之上,本文不仅在事务预处理环节和条件模式库挖掘环节可进行并行处理,而且由于采用多个局部 FP-Tree 来代替全局 FP-Tree,在生成 FP-tree 环节也实现了并行处理。

5　算法测试结果

实验用计算机配置为,CPU:Intel Celeron D Processor 3.456 GHz;内存:1GB;操作系统:Microsoft Windows XP Professional。实验数据采用 T10I4D100K. dat (http://fimi. ua. ac. be/data/T10I4D100K. dat),数据文件大小为 3.93 MB,该数据集共有 10 万个事务,项目最小编号为 0,最大编号为 999,计数大于零在事务中实际出现过的项目为 871 个,所有事务中项目出现的总次数为 1 010 221,每个事务的平均长度为 10.1。

采用局域网多节点并行处理来进行算法测试。由于本算法中事务项目计数、过滤和排序以及生成 FP-tree 等数据量大的环节都是在数据存储节点本地完成,需要在网络上传输的主要是条件模式基,考虑到广域网尤其是行业、企业、部门专用网络的带宽也都在不断增大,这样的实验设计基本能够代表算法的实际应用环境。

设最小支持度为 1%,即最小支持度计数为 1 000,P-FP-Growth 算法与 FP-Growth 算法的测试结果对比如表 2 所示。

表 2　算法执行时间对比(单位:秒)

	FP-Growth	P-FP-Growth(10 个节点)
所有事务项目计数	5	0.6
各事务项目过滤和排序	112	12
生成 fp-tree	289	30
频繁模式挖掘	729	78
其他辅助工作	0	3
算法总时间	1 135	123.6

P-FP-Growth 算法分别运行在 1 ~ 10 个节点上时测试结果如图 1 所示。

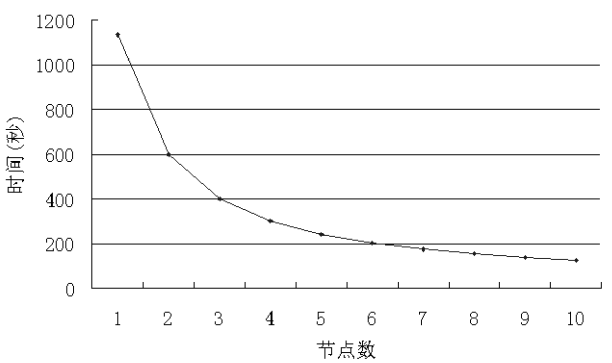

图 1　算法执行时间

6　算法分析

由图 1 可知,本算法在单机上运行时,所需时间和 FP-Growth 算法基本相当,结合算法分析可知,从算法总的计算量来说,本算法和 FP-Growth 是一样的。

从表 2 可以看出,本算法在事务项目计数、过滤及排序,生成 FP-tree 和频繁模式挖掘等算法的主要步骤上都实现了并行处理,通过 10 个节点分担负载每个步骤所需时间都只略大于原来的十分之一,而为实现并行处理所需增加的辅助工作耗时却非常少,在算法测试中只有 3 秒,仅占算法总耗时的 2.43%。对比两个算法的总执行时间可知,在 10 个节点情况下,并行算法可以节约耗时 89.11%。所以可以说,并行化之后的算法 P-FP-Growth 执行效率得到显著提高。

从图 1 可以看出,设 FP-Growth 算法执行某一关联规则挖掘任务所需时间为 ϕ,那么大体上而言,本算法在同型号的 n 台计算机上执行同一关联规则挖掘任务所需时间仅略大于 ϕ/n。之所以会略大于 ϕ/n,一是因为任务在分配到各个节点时无法做到完全平均,二是因为为实现并行处理所做的辅助性工作会略有耗时。

本算法的改进之处主要体现在:

(1) 采用多个局部 FP-Tree 代替全局 FP-Tree,可避免全局 FP-tree 可能过大而内存无法容纳的问题,突破了 FP-Growth 算法的处理能力瓶颈。

(2) 不仅在事务项目计数、过滤、排序环节和条件模式库挖掘环节可进行并行处理,而且在生成 FP-tree 环节也实现了并行处理。

(3) 基于分布式数据架构,特别适用于当前应用日益广泛的分布式数据挖掘。

7　结　语

本文提出一种分布式并行关联规则挖掘算法,该算法针对分布式的应用数据架构,不需要产生全局 FP-tree,在算法各个主要步骤上都实现了分布式并行处理。实验结果表明,在有 n 个节点并行处理的情况下,算法耗时只略大于传统 FP-Growth 算法的 $1/n$。从算法的适用范围来说,由于不需要全局 FP-tree,能够突破 FP-Growth 算法的处理能力瓶颈,本算法非常适用于针对海量数据的关联规则挖掘,另外由于采用分布式处理,本算法也非常适用于针对分布式数据架构的关联规则挖掘。

参　考　文　献

[1] Agrawal R, Imielinski T, Swami A. Mining association rules between sets of items in large databases[C]//Proceedings of ACM SIGMOD International Conference on Management of Date, 1993:207 - 216.

[2] Agrawal R, Srikant R. Fast algorithms for mining association rules [C]//Proceedings of the 1994 International Conference on Very Large Data Bases, 1994:487 - 499.

[3] 朱玉全,孙志挥,季小俊. 基于频繁模式树的关联规则增量式更新算法[J]. 计算机学报,2003,26(1):91 - 96.

[4] Han J, Pei J, Yin Y. Mining Frequent Patterns Without Candidate Generation[C]//Proceedings of ACM SIGMOD International Conference on Management of Data, 2000:1 - 12.

[5] Pramudiono I, Kitsuregawa M. Parallel FP-Growth on PC cluster[C]// Proceedings of International Conference on Internet Computing, 2003: 467 - 473.

(下转第 119 页)

3 σ 的范围内。

③ 函数曲线下 99.993666% 的面积在均值左右四个标准差 4 σ 的范围内。

④ 函数曲线下 99.999976% 的面积在均值左右五个标准差 5 σ 的范围内。

根据以上正态分布的性质,可以设定:

① 当图像中 0.3% 的像素点在平均值左右三个标准差 3 σ 以外的范围,则判定轴承有缺陷。

② 当图像中 0.007% 的像素点在平均值左右四个标准差 4 σ 以外的范围,则判定轴承有缺陷。

③ 当图像中 0.00001% 的像素点在平均值左右五个标准差 5 σ 以外的范围,则判定轴承有缺陷。

采用以上的方法对轴承质量进行判定,能有效地检测出缺陷轴承,对于凹坑、孔洞、断裂等明显缺陷,缺陷部分的像素点大部分会在 5 倍标准差以外,由此可知该方法能检测出一些细小孔洞。很多锈斑、污渍等缺陷的灰度值接近与背景颜色,通过 3 个标准差划分阈值时,还是能将大部分缺陷与背景分离开来,而且这类缺陷出现时,基本都是有一定面积的,通过 3 倍标准差,能有效检测出一些浅锈斑缺陷的轴承。如图 12 所示,为轴承图像分别在 3 σ、4σ、5σ 这三个尺度下分割的图像,从左到右分别是原图像、3σ 尺度分割的图像、4σ 尺度分割的图像、5σ 尺度分割的图像,从上大小分别为轴承 1、轴承 2、轴承 3。轴承 1 为合格轴承;轴承 2 为小缺陷轴承;轴承 3 为大缺陷轴承。对应的分割出的白色区域占总面积的百分比如表 2 所示。可以看出,该方法能避免轴承表面纹理所带来的影响,使判定更准确。

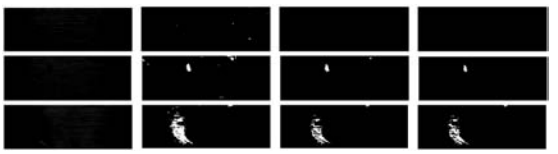

图 12　轴承多尺度阈值分割对照图

表 2　轴承决策判定表

分割尺度	阈值	轴承 1	轴承 2	轴承 3
3 σ	0.3%	0.06%	0.7755%	8.1639%
4 σ	0.007%	0%	0.3888%	4.3321%
5 σ	0.00001%	0%	0.3210%	2.3545%
判定结果		通过	不通过	不通过

6　实验与分析

系统采用 Matlab 2010a 软件仿真。离线实拍 233 张轴承图像中,63 张合格轴承图像,170 张缺陷轴承图像。取 10 张合格轴承图像作为训练样本,剩余的 213 张作为测试样本。测试结果如表 3 所示。总体正确率达到 98.2%。具有较高的识别率。

表 3　实验结果表

	被检数量	正确检测数量	误检数量	正确率
合格轴承	53	50	3	94.3%
缺陷轴承	170	169	1	99.4%
总数	223	219	4	98.2%

通过实验得到,一张轴承图片的平均检测速度为 20 ms 左右。理论上一个轴承的检测时间平均只需 60 ms,能满足工业现场流水线对系统实时性的要求。

7　结　语

本文探讨轴承表面缺陷检测系统的研究与开发,在前人研究的基础上进一步改进,得出了更具通用性和可靠性的检测系统。系统采用单方向的梯度法对轴承进行定位。拟合图像灰度分布曲线,实现快速灰度变换;提出基于样本图像的灰度均值、方差、以及被测图像的灰度均值动态选择全局阈值的方法并进行多尺度阈值分割图像及决策判断。本方法具有高实时性、高准确性、高稳定性、高工程实用价值。但在灰度变换的过程,对光照的稳定性要求较高,光照变化后的图像进行灰度变化存在一定的失真,仍然有待提高。

参　考　文　献

[1] 赵妍妍. 基于计算机视觉的微小轴承外圈表面缺陷检测技术研究[D]. 长春:吉林大学机械工程与工程学院,2008.

[2] Kurokawa K, Morimoto M, Fujii K. On-stream defect detection of metal artifacts from line CCD image[C]//Automation Congress, 2008. WAC 2008. World Publication, 2008:1-6.

[3] 梁琳,何卫平,雷蕾,等. 光照不均图像增强方法综述[J]. 计算机应用研究,2010,27(5):1625-1628.

[4] Park Changhyun, Won Sangchul. An automated web surface inspection for hot wire rod using undecimated wavelet transform and support vector machine[C]//Industrial Electronics, 2009. IECON'09. 35th Annual Conference of IEEE, 2009(3-5):2411-2415.

[5] 景振毅,等. MALTLAB 7.0 实用宝典[M]. 北京:中国铁道出版社,2009.

[6] Carsten Steger, Markus Ulrich, Christian Wiedemann. 机器视觉算法与应用[M]. 杨少荣,吴迪靖,段德山,译. 北京:清华大学出版社,2008.

[7] Rafael C Gonzalez, Richard E Woods. 冈萨雷斯数字图像处理[M]. 阮秋琦,阮宇智,译. 北京:电子工业出版社,2010.

[8] Steven M Kay. 统计信号处理基础——估计与检测理论[M]. 罗鹏飞,等译. 北京:电子工业出版社,2011.

[9] 曹菊生,魏国强. 概率统计与数据处理[M]. 苏州:苏州大学出版社,2011.

(上接第 115 页)

[6] Zaïane O R, Mohammad E H, Lu P. Fast parallel association rule mining without candidacy generation[C]//Proceedings of 1st IEEE International Conference on Data Mining, 2001:665-668.

[7] Liu L, Li E, Zhang Y, et al. Optimization of frequent item-set mining on multiple-core processors[C]//Proceedings of 33rd International Conference on Very Large Data Bases, 2007:1275-1285.

[8] 谈克林,孙志挥. 一种 FP 树的并行挖掘算法[J]. 计算机工程与应用,2006(13):155-157.

[9] 陈敏,李徽翡. 集群系统中的 FP-Growth 并行算法[J]. 计算机工程,2009(20):71-75.