doi:10.3969/j.issn.1001-4616.2016.04.006

基于 Hadoop 平台的 SPRINT 算法的分析与研究

黄　刚,孙　媛

(南京邮电大学计算机学院 软件学院,江苏 南京 210003)

[摘要]　传统的决策树算法在单机平台上处理海量数据挖掘时,容易受到计算能力和存储能力的限制,所以存在耗时过长、容错性差、存储量小的缺点. 而拥有高可靠性和高容错性的 Hadoop 平台的出现为决策树算法的并行化提供了新的思路. 本文设计和实现了一种基于 Hadoop 平台的并行 SPRINT 分类算法. 实验结果表明:基于 Hadoop 平台的 SPRINT 分类算法比没有进行并行化的 SPRINT 算法具有较好的分类正确率、较低的时间复杂度和较好的并行性能,并且能明显提高算法求最佳分裂点时的执行速度.

[关键词]　Hadoop,MapReduce,数据挖掘,决策树,SPRINT 算法
[中图分类号]TP301.6　[文献标志码]A　[文章编号]1001-4616(2016)04-0025-06

Analysis and Study of SPRINT Algorithm Based on Hadoop Platform

Huang Gang,Sun Yuan

(School of Computer Science & Technology,School of Software,Nanjing University of Posts and Telecommunications,Nanjing 210003,China)

Abstract:When the traditional decision tree algorithms handle massive data mining on a single platform,due to limited computing power and storage capacity. It has the shortcomings that taking too long time,poor fault tolerance,small storage capacity. The emergence of the Hadoop platform which has high reliability and fault tolerance has provided a new way for parallelization of decision tree algorithm. In this paper,a parallel SPRINT classification algorithm based on Hadoop platform has been designed and implemented. The results show that the SPRINT classification algorithm based on Hadoop platform has better classification accuracy than the SPRINT algorithm without parallelization. It also has lower time complexity and better parallel performance. It can improve the execution speed of the algorithm for the best time of the split point significantly.

Key words:Hadoop,MapReduce,data mining,decision tree,SPRINT algorithm

数据分类[1]是数据挖掘中的一项重要分析方法. 数据分类的方法有很多种,包括决策树、统计学(如贝叶斯)、神经网络、粗糙集和最邻近这些方法. 分类方法中最知名的就是决策树:其分类的速度快,决策树模型简单、便于理解;与其他方法相比,决策树分类可以得到较高的精确度[2]. 由于决策树分类模型的精准程度直接依赖于训练数据集的大小,所以在处理海量数据时很容易就会遇到处理数据集的时间复杂度过高的瓶颈,从而使其很难得到推广. 因此,决策树分类算法的并行化研究已经成为必然趋势.

在决策树分类算法的并行化研究中,主要的研究方法之一就是对现有的决策树算法采用并行化处理,并且将数据集运行在多处理器上. MapReduce 借鉴了函数式程序设计语言的设计思想,是专门用来处理海量数据集的具有内在并行性的分布式计算模型[3]. 因为在分布式处理中,移动数据的代价总是高于转移计算的代价(Moving Computation is Cheaper than Moving Data),这是分布式计算最重要的一个设计点[4]. 简单来说就是分而治之的工作,需要将数据分而存储,本地任务处理本地数据然后归总,这样才会保证分布式计算的高效性. 本文设计和实现了一种基于 MapReduce 编程模型的并行 SPRINT 分类算法.

1　Hadoop 概述

HDFS 和 MapReduce[3]是 Hadoop 框架[5]中最核心的设计. HDFS 是 Hadoop 分布式文件系统(Hadoop

收稿日期:2015-12-16.
基金项目:国家自然科学基金(61171053).
通讯联系人:黄刚,教授,研究方向:计算机在通信中的应用、海量数据管理、移动商务平台设计开发. E-mail:huanggang@ njupt.edu.cn

Distributed File System)的缩写,为分布式计算存储提供了底层支持. 而 MapReduce 的思想可以用简单的一句话解释就是“任务的分解与结果的汇总”.

首先综合 HDFS 和 MapReduce 来看 Hadoop 的结构(图 1):

在 Hadoop 的系统中,包括一台 Master 和多台 Slave. Master 主要负责 NameNode 以及 JobTracker 的工作,JobTracker 主要负责启动、跟踪和调度各个 Slave 的任务执行. 而每一台 Slave 通常具有 DataNode 的功能并负责 TaskTracker 的工作. TaskTracker 根据应用要求结合本地数据执行 Map 任务以及 Reduce 任务.

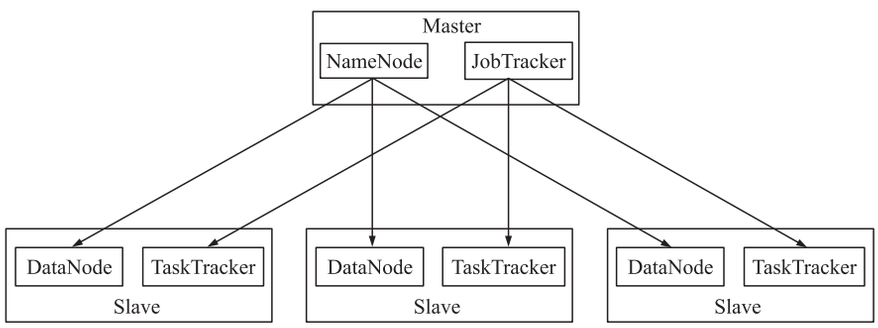

图 1　Hadoop 结构示意图

Fig. 1　The structure diagram of Hadoop

1.1　HDFS 分布式文件系统

HDFS 分布式系统以主从结构创建节点[6],其中主节点为 NameNode,其他从节点为 DataNode. 文件以数据块的形式存储在 DataNode 中. 下面就简单介绍一下 HDFS 包含的节点和功能:

(1)NameNode. NameNode 在分布式文件系统中所承担的责任是管理文件系统的命名空间,维护文件系统的文件树及所有文件和目录的元数据. 这些信息存储在 NameNode 维护的 2 个本地磁盘文件:命名空间镜像文件和编辑日志文件. 同时,NameNode 中还保存了每个文件与数据块所在的 DataNode 的对应关系,其他功能组件查找所需文件资源的数据服务器时便会用到这些信息.

(2)Secondary NameNode. 在一个 Hadoop 集群环境中,只有一个 NameNode 节点,所以 NameNode 成了整个 HDFS 系统的关键故障点,一旦发生故障将影响整个系统的运行. 为了避免这样的问题出现,Hadoop 设计了 Secondary NameNode 节点,它一般在一台单独的物理计算机上运行,与 NameNode 保持通信,按照一定时间间隔保持文件系统元数据的镜像. 当 NameNode 真的发生故障时,系统管理者就可以通过手工处理的形式将保存的元数据按照快照恢复到重新启动的 NameNode 中,这样就可以降低数据丢失的风险.

(3)DataNode. DataNode 是 HDFS 文件系统中文件存储的基本单元. HDFS 中的文件通常被分割为多个数据块,以冗余备份的形式存储在多个 DataNode 中. DataNode 定期向 NameNode 报告其存储的数据块列表,以备使用者通过直接访问 DataNode 获取所需信息.

(4)Client. Client 是 HDFS 文件系统的使用者,通过调用 HDFS 提供的 API 对系统中的文件进行读写操作. 但是在进行读写操作时,Client 需要先从 NameNode 获得文件存储的元数据信息,然后才能与相应的 DataNode 进行数据读写操作.

1.2　MapReduce 并行编程框架

MapReduce[7]从它名字上来看就大致可以看出缘由,两个动词 Map 和 Reduce,Map(展开)就是将一个任务分解成为多个任务,然后由 Reduce 将分解后的多任务处理的结果汇总起来,得出最后的分析结果.

在处理之前,首先将数据集分为若干个独立的数据块,分发至各个节点作为任务可以并行处理的前提. 处理时,每个节点就近读取本地存储的数据进行处理(map),由 Map 任务(task)以完全并行的方式处理它们,将处理后的数据进行合并(combine)、排序(shuffle and sort)后作为 reduce 函数的输入,这样可以避免大量数据的传输,提高处理效率. MapReduce 集群可以是由普通 PC 机构成的无共享式架构,表明各个任务彼此之间没有依赖性. MapReduce 框架为大数据处理提供了一种可以利用底层分布式计算进行并行处理的计算模式.

简而言之,Hadoop 提供了一个稳定的共享存储和分析系统. 存储由 HDFS 实现,分析由 MapReduce 实

现. 虽然 Hadoop 还有其他功能,但这些功能是 Hadoop 的核心所在.

2　SPRINT 算法

SPRINT 算法[8]是决策树分类算法中比较典型的一个算法. SPRINT 算法的目的就是彻底地解决内存容量受限制的问题,能够处理其他任何算法都不适用的超大规模训练样本集,并且能有效地生成决策树.

(1)创建根节点 N;

(2)if T 都属于同一类别则返回 N 为叶结点;

(3)for each T 中的属性 A,执行 A 上的所有可能划分,找出最佳划分将 T 分割为 T_1,T_2;

(4)调用 sprintformtree(T_1);

(5)调用 sprintformtree(T_2).

本文只讨论 SPRINT 算法的决策树的构建阶段,对剪枝阶段不作讨论,因为剪枝阶段所占的时间比例很小.

2.1　SPRINT 算法的数据结构

SPRINT 算法定义了 2 个其他决策树算法所没有的新的数据结构[9]:属性表和直方图,它们都是产生于训练集,附属于决策树结点上. 属性表主要包含了 3 个字段:记录的索引 rid、属性值、类别标识. 建立好属性表后,如果是连续型属性,则属性表要按属性值进行预排序,离散属性表则不用预排序. 初始化完毕之后,所有属性表都关联到决策树的根节点. 属性表解决了之前的决策树算法中训练数据需要完全驻留内存及需要多次排序的问题. 在树的建立过程中,当结点分裂时,属性表也随之分裂,并关联到相应的子结点中. 划分属性表时,并没有改变属性记录的顺序,SPRINT 算法就是通过这种方法解决之前的决策树算法需要多次排序的问题.

直方图是用来描述结点上的属性的所属类别的分布情况. 当描述连续型属性时,对每个决策树结点都维护 2 个统计直方图,分别是 C_{above} 和 C_{below},用于表示属性记录在给定结点上的类分布情况,其中 C_{above} 记录的是未处理过的类,C_{below} 则是记录已处理过的类,遍历属性表时,直方图也随之改变. 而离散型属性同样要有一个直方图,用来记录属性值所对应的类的分布情况. 直方图的作用主要用于计算每一种分裂方案的 gini index 值,即寻找最佳分割.

2.2　计算最佳分裂

SPRINT 算法依旧采用 gini 指标[10]作为评价结点分裂质量的参数.

对于有 n 种类别的数据集 S,

$$\mathrm{Gini}(S)=1-\sum p_1^2. \tag{1}$$

式中,p_j 为 S 种类别 j 的相对频率.

如果一个分割将 S 分为 S_1 和 S_2 两个子集,则被分裂后的 gini 参数 $\mathrm{gini}_{split}(S)$ 为:

$$\mathrm{Gini}_{split}(S)=\frac{n_1}{n}\mathrm{gini}(S1)+\frac{n_2}{n}\mathrm{gini}(S2). \tag{2}$$

为了找出划分结点的最佳分裂点,要遍历这个结点上所有的属性表,并且评价这个属性的分裂好坏. 在求得的 gini 参数中找出最小值,则最小的 gini 参数所对应的属性被选作这个结点的分裂属性. 然后就是要确定属性的分裂点.

对于连续属性而言,分裂点是训练集中两个连续属性值的中间点. 为了求结点的分裂点,开始时 C_{below} 被初始化为 0,而 C_{above} 则初始化为所有记录在根结点的类分布情况,并且在初始化排序时随之获得. 对其他结点的类分布在新建结点的时候才可以确定. 一次只读取一个属性记录,每读取一次都要更新相应的 C_{below} 和 C_{above}. 一旦读入一个记录值后,就要对此值间(即属性的记录之间)的分裂进行评估. C_{below} 和 C_{above} 中已经包含了求解 gini 参数所需的全部信息,由于连续属性表中的记录都是已排序的,所以属性的每个候选分裂点的求解只需顺序扫描一次属性表即可求出,如果在扫描过程中找到了成功的分裂点,便将其保存下来,并在处理下一个属性前释放 C_{below} 和 C_{above}.

对于离散属性而言,只需要对属性表做一次扫描求出直方图的统计数据即可. 一旦完成了扫描,可以

将所有属性值看作可能的分裂点来求出对应的 gini 参数,要注意的是,直方图中已包含全部计算任意子集分裂所需的 gini 参数的信息.

2.3 执行结点的分裂

在最佳分裂点计算出后,需要创建新的子结点,同时将属性表分割到各子结点上. 最佳属性表的划分,可以按照求出的分裂点将记录分别放入每一个子节点的属性表中,其他非分支的属性表可以通过一个新的数据结构进行划分. 通过引入哈希表[11]并且利用属性表中的 rid(序号)进行划分,当划分分支属性表的时候,把属于左子结点的记录的 rid 插入哈希表中,于是在划分支属性表时,就可以通过查看记录的 rid 值是否在哈希表中,来决定将该记录归于哪个子结点. 分裂时还要为每一个新的叶子结点创建直方图,用于下一步评价连续属性的最佳分裂时初始化 C_{above}.

3 基于 Hadoop 的 SPRINT 算法的并行化

由于 SPRINT 算法能较好地支持并行处理即允许多处理器并行工作,所以可以利用 Hadoop 在并行处理方面的优势对 SPRINT 算法以并行的方式计算以减少串行计算时所带来的过大的时间复杂度[12]. 由此可以得到新的 SPRINTbH(SPRINT based on Hadoop)算法.

在这个新的 SPRINTbH 中,主要考虑的问题[13]就是如何在 MapReduce 并行建树阶段求得好的分裂点,并且用求得的分裂点对数据集进行正确划分. 而每分裂一次结点就调用一次 MapReduce 函数,因为要通过 MapReduce 函数确定分裂点,直到满足决策树的结束条件.

SPRINTbH 算法描述:

(1)首先根据训练数据集创建初始属性表,如果是连续属性,则要对属性表进行预排序.

(2)由于 MapReduce 的函数的输入输出都会被存储在文件系统. 所以输入是存放在文件中的已经预排序的属性表,在交给 Map 函数处理前,文件会被分割成多个文件块,每个文件块被分配存储到数据节点上,Map 函数处理时,每个节点就近读取本地存储的数据处理.

(3)如果是连续属性,那么每个 Map 函数获得的就是以属性值大小排序的属性表,同时生成直方图. 而属性值排序的顺序则是以 Map 函数读取的顺序. 然后每个 Map 函数对自己获得的属性表进行最佳分裂点计算,即计算出在各自的 Map 函数处理环节的所有的 gini 值,实时更新直方图(对于离散属性,无需排序,直方图无需更新),并且找出其中的最优方案,即最小的 gini 值. 对于分类属性,每个 Map 函数分别统计各自的属性记录的类的分布信息.

(4)每个 Map 函数在计算出各自的最优方案后,在输出过程中,会将所有具有相同属性的属性表求得的 gini 值合并,作为 Reduce 函数的输入,Reduce 函数找出这所有的 gini 值中最小的值,确定全局的最优方案,即最佳分裂属性和分裂点. Reduce 函数同时还要将 Map 函数中统计出的类的分布信息进行相加,得到全局的类分布频率.

(5)确定了最优方案以后,将 Reduce 的输出作为新的 Map 函数的输入,每个 Map 函数将各自的属性表划分到相应的子结点中. 可以通过将划分到左子结点的属性的 rid 记录到事先建立好的哈希表中,即每个 Map 函数可以将自己统计的记录到哈希表的左子结点中的属性作为输出,传递给 Reduce 函数作为输入. Reduce 函数统计所有的 Map 函数的输出,得到所有的被划分到左子结点的属性. 然后确定在这个结点中的所有的左子结点和右子结点的属性.

(6)重复(2)~(5)步骤,直到满足决策树结束条件.

4 实验结果分析

把 SPRINTbH 算法在 Hadoop 平台上用打高尔夫球的训练数据集做实验,对这个算法的有效性进行验证,其分布如表 1 所示.

本实验主要是为了验证并行化以后的 SPRINTbH 算法的高效性和可扩展性,用来判断某种天气是否适合打高尔夫. 由于这个数据集属于天气领域的,所以包含了 Outlook、Temperature、Humidity 和 Windy 这 4 个属性. 没有缺省值,所以被用来完成分类任务.

该数据集包含 13 条记录和 4 个属性(其中 Temperature 和 Humidity 为连续型属性,Outlook 和 Windy

为离散型属性).

对于离散属性值超过 2 个的,可以将其组合成 2 个后再进行树的分裂. 在该数据集中 Outlook 有 3 个属性值,分别是 Sunny、Rain 和 Overcast,所以可以将 Sunny 和 Rain 组合在一起,然后再进行 gini 计算.

表 1　打高尔夫球的训练数据集

Table 1　The training data set of Golfing

Outlook	Temperature	Humidity	Windy	Class:Play	rid
Sunny	85	85	False	No	0
Sunny	80	90	True	No	1
Overcast	83	78	False	Yes	2
Rain	70	96	False	Yes	3
Rain	68	80	True	No	4
Rain	65	70	True	No	5
Overcast	64	65	False	Yes	6
Sunny	72	95	False	No	7
Sunny	69	70	False	Yes	8
Rain	75	80	True	Yes	9
Sunny	75	70	True	Yes	10
Overcast	72	90	True	Yes	11
Overcast	81	75	False	Yes	12
…	…	…	…	…	…

通过实验验证得到在创建根节点时的经过预排序的属性表(如表 2 和表 3 所示). 由 Reduce 函数进行处理得到的在确定根节点阶段的最优方案是在 $gini_{split}$(Outlook)时的值最小,所以将 Outlook 属性作为决策树的第一次分裂. 创建的根节点处的哈希表如图 2 所示.

表 2　预处理以后的 Outlook 和 Windy 属性表

Table 2　The attribute table of outlook and windy after pretreatment

Outlook	Class:Play	rid	Windy	Class:Play	rid
Sunny	No	0	False	No	0
Sunny	No	1	True	No	1
Overcast	Yes	2	False	Yes	2
Rain	Yes	3	False	Yes	3
Rain	No	4	True	No	4
Rain	No	5	True	No	5
Overcast	Yes	6	False	Yes	6
Sunny	No	7	False	No	7
Sunny	Yes	8	False	Yes	8
Rain	Yes	9	True	Yes	9
Sunny	Yes	10	True	Yes	10
Overcast	Yes	11	True	Yes	11
Overcast	Yes	12	False	Yes	12
…	…	…	…	…	…

表 3　预处理以后的 Temperature 和 Humidity 属性表

Table 3　The attribute table of temperature and humidity after pretreatment

Temperature	Class:Play	rid	Humidity	Class:Play	rid
64	Yes	6	65	Yes	6
65	No	5	70	No	5
68	No	4	70	Yes	8
69	Yes	8	70	Yes	10
70	Yes	3	75	Yes	12
72	No	7	78	Yes	2
72	Yes	11	80	No	4
75	Yes	9	80	Yes	9
75	Yes	10	85	No	0
80	No	1	90	No	1
81	Yes	12	90	Yes	11
83	Yes	2	95	No	7
85	No	0	96	Yes	3
…	…	…	…	…	…

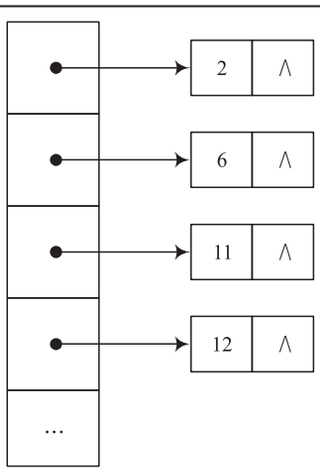

图 2　记录在哈希表上的划分至根节点的左子结点的属性的 rid

Fig. 2　The rid of properties that recorded on the has table which partition to the left child node of the root

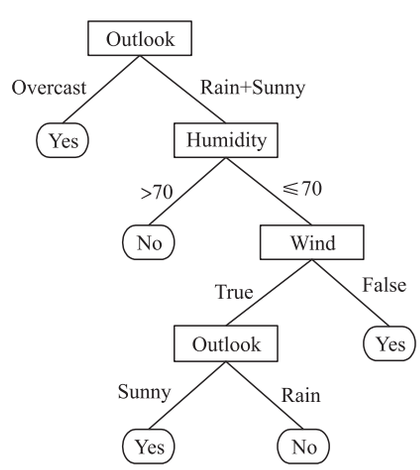

图 3　最终打高尔夫决策树

Fig. 3　The final decision tree of golfing

5 台 PC 机(其中一台主机,4 台从机),采用 SUSE 以及 Hadoop、Eclipse、JDK. 然后通过集群测试数据集在不同数量的节点上的运行时间. 运行统计结果如图 4 所示,数据的处理时间主要花费在数据的分割和记录的格式化过程,随着集群数量的增多,数据集的处理时间复杂度降低,算法的准确率也能维持在一定的水平,说明在 Hadoop 平台上对 SPRINT 算法进行并行化处理是可行的.

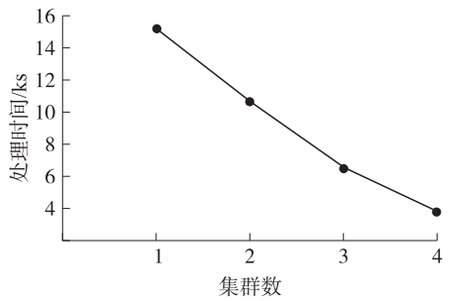

图 4　不同的集群数量处理运行结果

Fig. 4　The operation results of clustering with different numbers

5　结束语

决策树已经被应用到很多问题中:根据疾病分类患者;根据起因分类设备故障;根据拖欠支付的可能性分类贷款申请等等. 在 Hadoop 平台上搭建的 SPRINT 算法的并行化分类算法与传统 SPRINT 分类算法在处理海量数据方面相比有较大优势,能够高效地对大规模数据集进行分类处理. 实验表明,在 Hadoop 平台上搭建的 SPRINT 算法的并行化分类算法具有较好的分类正确率,较低的时间复杂度和较好的并行性能.

[参考文献]

[1] 邵峰晶,于忠清. 数据挖掘原理与算法[M]. 北京:中国水利水电出版社,2009.

[2] 王云飞. SPRINT 分类算法的改进[J]. 科学技术与工程,2008,8(23):6248-6252.

[3] TAYLOR,RONALD C. An overview of the Hadoop/MapReduce/HBase framework and its current application in bioinformatics[J]. BMC bioinformatics,2010.

[4] RANGER C,RAGHORAMAN R,PENMETSA A,et al. Evaluating mapreduce for multi-core and muti-processor systems[C]// IEEE 13th Internet Symposium on High Performance Computer Architecture. Melbourne:IEEE Australia,2007:13-24.

[5] TOM W,DOUG C. Hadoop 权威指南[M]. 周敏奇,王晓玲,金澈清,等译. 北京:清华大学出版社,2011.

[6] 刘军. Hadoop 大数据处理[M]. 北京:人民邮电出版社,2013.

[7] LU D,CHENG X. The research of decision tree Mining based on Hadoop[C]//9th Internation Conference on Fuzzy Systems and Knowledge Discovery(FSKD). Chongqing:IEEE,2012:798-801.

[8] 朱明. 数据挖掘[M]. 合肥:中国科学技术大学出版社,2008.

[9] 潘天鸣. 基于 Hadoop 平台的决策树算法并行化研究[D]. 上海:华东师范大学,2012.

[10] 刘学军. 基于最小 Gini 指标的决策树分类算法设计与研究[J]. 教育技术导刊,2009(5):56-57.

[11] 李远方,贾时银,邓世昆,等. 基于树结构的 MapReduce 模型[J]. 计算机技术与发展,2011,21(8):149-152.

[12] 朱敏,万剑怡,王明文. 基于 MR 的并行决策树分类算法的设计与实现[J]. 广西师范大学学报(自然科学版),2011,29(1):82-84.

[13] 孙媛,黄刚. 基于 Hadoop 平台的 C4.5 算法的分析与研究[J]. 计算机技术与发展,2014,24(11):83-90.

[责任编辑:黄　敏]