

基于云计算的 SPRINT 算法研究

杨 洁, 黄 刚

(南京邮电大学 计算机学院, 江苏 南京 210003)

摘 要: 决策树是数据挖掘中非常重要的一种技术, 常用来做数据分析和预测。传统的决策树算法在处理海量数据挖掘时, 受到 CPU 和内存的限制, 导致算法存在消耗时间过长, 容错性差, 存储量小的缺点。面对海量数据的处理, 云计算在这方面具有非常多的优势。针对决策树中优秀的 SPRINT 算法, 首先对 SPRINT 算法进行了优化, 然后为了让优化后的算法更好地应用于云计算, 对算法实现了并行化。传统的 SPRINT 算法在生成决策树时, 会发生多值偏向问题, 在生成一个节点时, 通过计算两层的 Gini 指数来降低多值偏向的影响。在算法并行化时, 通过将数据分发到各个处理器执行, 然后进行汇总处理, 从而减少算法执行的总时间。实验结果表明: 基于云计算平台的 SPRINT 改进算法具有更好的分类正确率, 同时算法的执行速度也得到了明显的提高。

关键词: 云计算; MapReduce; SPRINT 算法; Gini 指数

中图分类号: TP301.6

文献标识码: A

文章编号: 1673-629X(2017)03-0108-05

doi: 10.3969/j.issn.1673-629X.2017.03.022

Research on SPRINT Algorithm Based on Cloud Computing

YANG Jie, HUANG Gang

(School of Computer, Nanjing University of Posts and Telecommunications,
Nanjing 210003, China)

Abstract: Decision tree is a very important technology in data mining, which is often used for data analysis and forecasting. When the traditional decision tree algorithm is dealing with massive data mining, the CPU and memory is limited, resulting in its shortcomings like long time-consuming, poor fault tolerance and small storage capacity. Faced with massive data processing, cloud computing has a lot of advantages in this respect. It places emphasis on the good algorithm of SPRINT. First of all, it is optimized, and then parallelized in order to make the optimized algorithm better applied to cloud computing. When traditional SPRINT algorithm generates the decision tree, multi-valued bias problem will happen, and when it generates a node, through the calculation of Gini index of two layer, the effects of multi-valued bias is reduced. In parallel algorithm, through the distribution of data to the processor execution, then collecting and processing, the total time of execution is reduced. The experimental results show that the improved SPRINT algorithm based on cloud computing platform has better classification accuracy, and at the same time, its execution speed gets obvious improvement.

Key words: cloud computing; MapReduce; SPRINT algorithm; Gini index

0 引 言

数据挖掘发展至今, 已经产生了许多实用的数据挖掘方法, 包括决策树算法、贝叶斯方法、人工神经网络方法、支持向量机方法、聚类分析方法等。SPRINT 算法是决策树算法中的一种经典算法。SPRINT 算法是对 SLIQ 算法的进一步优化, 在内存的数据量驻留方面, 改进了决策树的数据结构。由于种种优点, 越来越多的人投身到 SPRINT 算法的研究中, 主要集中在对算法的本身进行改进。但是, 如今的数据量越来越

大, 无疑给算法工程师带来了许多新的艰巨挑战。单纯对算法本身的改进已经不能满足于现在的需求, 而云计算的出现解决了这个问题。然而, 仅仅将传统的 SPRINT 算法应用于云计算平台, 并不能充分发挥云计算的计算能力。

文中对 SPRINT 算法进行改进, 通过计算两层的 Gini 指数来产生一个决策节点。改进后的算法虽然增加了计算量, 但产生的分割属性优于未改进的 SPRINT 算法, 即产生了更优的决策树。

收稿日期: 2016-04-12

修回日期: 2016-08-10

网络出版时间: 2017-01-10

基金项目: 国家自然科学基金资助项目 (GZ211018)

作者简介: 杨 洁 (1991-), 男, 研究方向为计算机云计算与大数据应用; 黄 刚, 教授, 研究方向为计算机软件理论及应用。

网络出版地址: <http://www.cnki.net/kcms/detail/61.1450.TP.20170110.1028.070.html>

1 云计算平台

1.1 云计算简介

云计算这个新名词出现在2007年的第3季度,不到半年,它受到的关注程度就远远超过了先前大热的网格计算,并且在今天也没有任何减弱的趋势^[1]。云计算具有以下特点:“云”具有很大的规模性;云计算为了方便用户接入,使用了虚拟化技术,该技术可以让用户在任意地点使用各种不同的终端设备从“云”获得自己需要的服务;作为商业模型,“云”在创建阶段和维护阶段,在开销方面,也具有前所未有的性价比^[2]。

1.2 MapReduce 编程模型

MapReduce最初来源于Google的几篇论文,是一种用于处理海量数据挖掘的并行编程模型^[3]。

MapReduce的运行模型如图1所示。

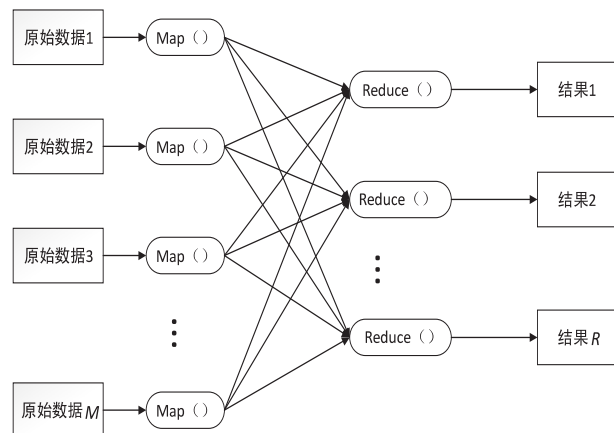


图1 MapReduce的运行模型

MapReduce降低了编程人员的要求,在编程时他们不需要关心程序运行的具体细节,只需要编写两个主要的函数:

Map: $(in_key, in_value) \rightarrow \{ (key_i, value_i) \mid i = 1, 2, \dots, k \}$

Reduce: $(key, [value_1, value_2, \dots, value_m]) \rightarrow (key, out_value)$

Map函数需要的输入参数和Reduce函数所输出的结果会根据不同的应用发生变化。 in_key 和 in_value 是Map的输入参数,它们指明了Map操作所要处理的原始数据是什么。Map操作结束后,会产生一些中间结果,它们以键值对的形式存在,即 $(key, value)$ 对。在Map阶段结束后,系统会对所有的中间结果进行整理,使具有相同key的value集结在一起,也就是说,Reduce的输入参数是 $(key, [value_1, value_2, \dots, value_m])$ 。Reduce的工作就是对这些具有相同键key的value值进行归并处理,最终产生 (key, out_value) 的结果。其中每一个Reduce都会产生自己的结果,最后,将所有Reduce的结果合在一起就成了最终结果^[4]。万方数据

2 SPRINT 算法

2.1 基本思想

SPRINT算法创建决策树的过程:

输入:节点 n ,数据集 D ,分割方法CL;

输出:一棵决策树,根节点为 n ,决策树基于数据集 D 和分割算法CL。

Procedure BuildTree(n, D, CL)

初始化根节点:

以CL为分割依据,计算 D 中的数据,生成节点信息;

If(节点 n 满足分割条件)

选择最好的效果将 D 分为 D_1, D_2 ;

创建节点 n 的子节点 n_1, n_2 ;

TDTree(n_1, D_1, CL);

TDTree(n_2, D_2, CL);

endif

end

SPRINT算法的终止条件一般存在3种情况:

(1)当前生成的节点中,所有的样本数据如果都属于同一个类,那么这个节点就是一个叶节点,即算法终止计算,同时用这个共同的类作为该叶节点的类标记。

(2)所有属性都已经用完,没有多余的属性用作测试属性。

(3)当前节点内的样本数少于用户规定的阈值,则用该节点内占大多数的类作为该叶节点的类标记^[5]。

2.2 SPRINT 算法的数据结构

SPRINT算法定义了两个特殊的数据结构:属性列表和类直方图。在SPRINT算法之前的决策树方法中,数据都必须长时间驻留在内存,限制了大数据的处理,SPRINT使用属性列表解决了这个问题,并取消了数据的多次排序。属性列表用一个三元组表示,即 $\langle \text{属性值}, \text{类别标识}, \text{行索引} \rangle$ 。其中行索引是自动生成的。初始化根节点时,对于离散属性,属性列表没有什么强制要求,而对于连续属性,属性列表则需要对它进行排序处理。在算法生成决策树的过程中,随着数据的不断分割,节点不断的生成,与节点相对应的属性列表也在被分割,然而属性列表顺序却没有改变,也就是说该属性列表也是有序的,因此,不用对属性列表进行重排^[6]。

类直方图的作用是记录节点上数据的属性类别的分布情况,这些数据用于计算对应节点的最佳分割属性。决策树节点在处理连续属性时,需要维护2个统计直方图,分别为 C_{below} 和 C_{above} 。其中, C_{below} 表示 $A \leq C$ 的类分布情况, C_{above} 表示 $A > C$ 的类分布情况, A 代

表数据中该连续属性的一个值, C 代表当前选取的连续属性分割值。随着树的创建, 直方图不断更新, 以寻找最佳分割点。对于每一个离散属性, 节点只需要维护 1 个直方图, 该直方图也被称为统计矩阵, 用来对这个离散属性具有的所有离散值分布情况进行记录^[7]。

2.3 分割指数

SPRINT 算法采用 Gini 指数作为分割标准。在属性选择标准中, Gini 指数算法是采用数据集的不纯度作为属性选择标准的。一个节点 t 的 Gini 指数计算公式为:

$$\text{Gini}(t) = 1 - \sum_{i=0}^{C-1} [P(i|t)]^2 \quad (1)$$

其中, $P(i|t)$ 表示给定节点 t 中属于类 i 的记录所占的比例; C 表示所有类的总个数。

如果以 A 作为集合, 将集合划分成节点 B_1 和节点 B_2 , 则分割后的 Gini 指数计算方法为:

$$\text{Gini}_{\text{split}}(A) = \left(\frac{n_1}{n}\right) * \text{Gini}(B_1) + \left(\frac{n_2}{n}\right) * \text{Gini}(B_2) \quad (2)$$

其中, n, n_1, n_2 分别为 A, B_1, B_2 的记录数。

SPRINT 算法在所有的属性中, 选择 $\text{Gini}_{\text{split}}(A)$ 具有最小值的 A^* 作为分割标准, 因为 $\text{Gini}_{\text{split}}(A)$ 越小表明分割属性选择得越好^[8]。

3 SPRINT 算法存在的缺点和改进方案

SPRINT 算法通过 Gini 值来选择分割属性, 然而这会引发多值偏向问题。多值偏向会导致生成非最优甚至是错误的决策树, 原因在于多值偏向在选择分割属性时, 往往会选择取值个数较多的属性作为分割属性, 这就将分割结果与属性的取值个数关联到了一起, 这种关联是没有道理的。文献[9]首先通过理论分析, 研究了多值偏向问题, 然后直接通过算法表达式进行推理, 证明了选择 Gini 指数作为分割标准确实会引发多值偏向问题^[9]。

针对多值偏向问题, 文中提出了一种对 SPRINT 算法的改进方案。假设 A 为测试属性, 属性 A 具有 2 个属性值, 每个属性值对应的概率为 p_1, p_2 。算法先对属性 A 进行第一次分割, $\{B_1, B_2\}$ 为这两个节点的属性, 分别对应的不纯度为 $\text{Gini}_{\text{split}}(B_1)$ 和 $\text{Gini}_{\text{split}}(B_2)$, 则:

$$\text{Gini}'_{\text{split}}(A) = p_1 * \text{Gini}(B_1) + p_2 * \text{Gini}(B_2) \quad (3)$$

算法选择属性 A' 作为分割属性的标准时, A' 使得 $\text{Gini}'_{\text{split}}(A')$ 最小。

算法的详细步骤如下:

(1) 存在属性候选集 Q , 算法依次从 Q 中选取每一个属性作为候选属性 A , 对 A 先进行一次二元分割,

生成两个新节点 $\{B_1, B_2\}$, 并且每个节点的属性值概率分别为 p_1 和 p_2 , 此时不用计算两个节点的 Gini 值。然后算法对 B_1 和 B_2 节点再次进行分割, 又分别产生两个节点, 共计四个节点。计算 $\text{Gini}_{\text{split}}(B_1)$ 和 $\text{Gini}_{\text{split}}(B_2)$ 。

(2) 由式(3)计算 $\text{Gini}'_{\text{split}}(A)$ 。

(3) 算法在产生的所有 $\text{Gini}'_{\text{split}}(A')$ 中, 选取最小的 A' , 作为当前产生节点的最佳分割属性。然后, 根据选出的属性, 对节点的数据进行分类, 生成真正的 B_1 和 B_2 节点。由于之前已经计算过, 因此在保留第一步的数据时, 可以迅速完成对该节点的分类和数据信息采集。

(4) 对于所有生成的节点, 如果满足算法的停止条件, 就将该节点设置为叶节点, 并用它的所属类标记, 否则, 就运用步骤(1)~(3)对节点继续进行分割, 直到算法结束。

4 基于 MapReduce 的并行化

4.1 原始数据的预处理

在云计算平台中, 考虑到负载均衡的原则, 采用水平分割对整个数据集进行分段, 将数据集平均地分配到所有处理器中。这样数据集中的种类属性能够被均匀分组, 不需要对数据再进行其他操作^[10]。由于 SPRINT 算法对于连续数据是有特别要求的, 即要求连续数据是已经排好序的。在算法开始时, 由于原始数据中的连续属性是没有经过排序的, 所以, 在对数据水平分割后, 处理器需要对其中的连续属性进行排序和重新分组操作, 使得连续属性的属性列表是一张相邻有序的表^[11]。

4.2 并行计算节点的最佳分割属性

在并行环境中, 计算当前节点的最佳分割属性时, 需要对连续属性和离散属性分别进行处理。

对于连续属性, 算法初始化 C_{below} 和 C_{above} 类直方图, 同时各个节点进行通信。 C_{below} 记录该处理器之下的数据, C_{above} 记录该处理器之上的数据^[12]。然后对该属性列表的所有数据进行检索, 同时更新两个类直方图, 并且根据直方图的信息, 计算每个连续值的 Gini 指数。在当前计算的所有连续值中选取一个具有最小 Gini 指数的连续值作为该属性的第一层最佳分割点。

对于离散属性, 每个处理器根据自身所拥有的数据建立对应该属性局部的类直方图 C , 同时获取一个协助处理器, 这个协助处理器的任务是汇总同一属性的所有局部类直方图, 将这些信息相加, 就得到了该属性的全局类直方图信息^[13]。然后根据协助处理器所记录的该属性全局直方图信息, 就能够计算出该属性

的第一层最佳分割点。

这样,就能够分别对连续和离散属性进行计算了,即完成了对 $Gini_{split}(A)$ 的计算。然后,按照第一层的计算方式,继续产生第二层最佳分割属性,即能够计算出 $Gini_{split}(B_1)$ 和 $Gini_{split}(B_2)$ 的值,然后利用式(3)计算出当前最小的 $Gini'_{split}(A)$ 。

4.3 分割节点的属性列表

完成最佳分割点的计算之后,每个处理器根据该最佳分割和行索引,对自己拥有的属性列表进行分割,同时创建局部哈希表。然后,所有处理器进行通信,共享自己刚刚建立的局部哈希表,所有哈希表进行汇总整理后,就可以创建全局哈希表。每个处理通过查找该全局哈希表,对自己的数据进行分割,分割完后,将分割的数据存入子节点中。分配完后,决策树的一个节点就创建了。重复以上步骤,直到满足节点不再需

要分割的条件,即为叶子节点^[14]。

5 实验

5.1 实验内容

实验所选取的数据集来自于 UCI 网站的 Acute Inflammations 数据集。该数据集的数据来源是一个医学专家数据集测试系统,它将执行的推断是诊断泌尿系统的两种疾病。数据集总共包含 6 个属性,其中有 1 个连续属性。数据集内的数据形式为“35, 9nonoyesyesyesyesno”,最后 2 个是类别属性,decision: Inflammation of urinary bladder 和 decision: Nephritis of renal pelvis origin。实验选择第一个类别属性。

实验对数据集使用未改进的 SPRINT 算法进行分割,产生的决策树如图 2 所示。

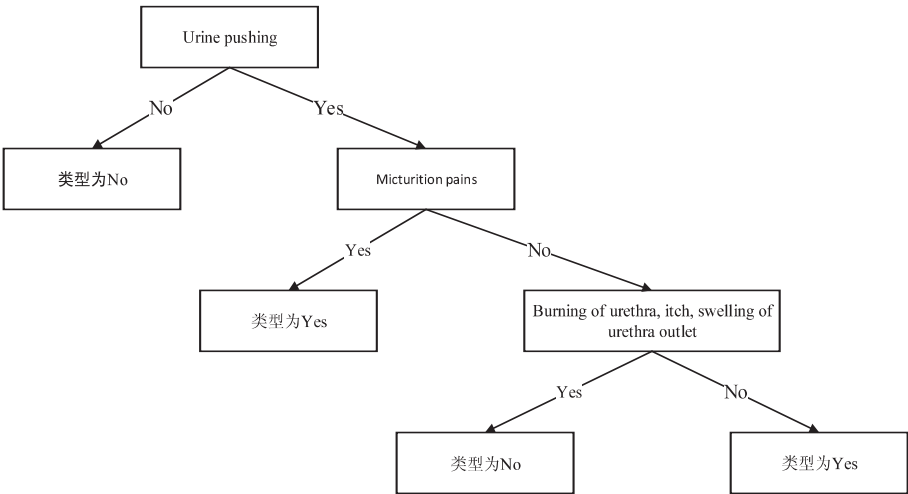


图2 原SPRINT算法生成的决策树

然后将同一数据集用改进后的算法进行分类,产生的决策树如图 3 所示。

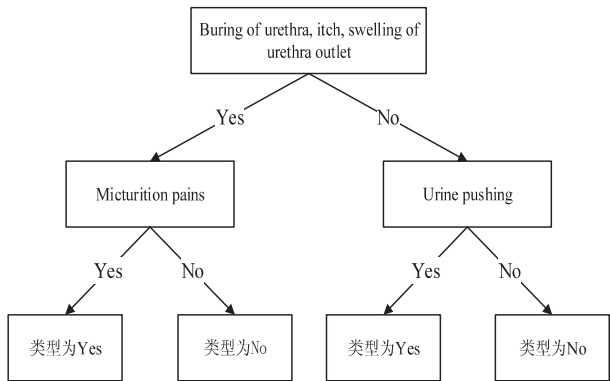


图3 改进SPRINT算法生成的决策树

5.2 实验结果分析

下面,分析改进后的算法是如何生成图 2 的决策树的,仅分析第一个分割属性的生成过程。算法依次将各个节点作为第一个分类属性,然后选取另一个分类属性进行决策。如果选择 Buring of urethra 对样本

先进行分类,接着分别选取 5 个其他的属性进行分类,左子树用 Temperature of patient 分类,结果为:

$$Ginisplit(\text{Temperature of patient} < 38.0) = 20/50Gini(\text{左}) + 30/50Gini(\text{右}) = 20/50(1 - (20/20)^2 - (0/20)^2) + 30/50(1 - (9/30)^2 - (21/30)^2) = 0.252$$

若左子树用 Occurrence of nausea 分类,结果为:

$$Ginisplit(\text{Occurrence of nausea}) = 41/50Gini(\text{左}) + 9/50Gini(\text{右}) = 41/50(1 - (20/41)^2 - (21/41)^2) + 9/50(1 - (9/9)^2 - (0/9)^2) = 0.4097$$

若左子树用 Lumbar pain 分类,结果为:

$$Ginisplit(\text{Lumbar pain}) = 20/50(1 - (20/20)^2 - (0/20)^2) + 30/50(1 - (9/30)^2 - (21/30)^2) = 0.252$$

若左子树用 Urine pushing 分类,结果为:

$$Ginisplit(\text{Urine pushing}) = 0/50(1 - (0)^2 - (0)^2) + 50/50(1 - (21/50)^2 - (29/50)^2) = 0.4872$$

若左子树用 Micturition pains 分类,结果为:

$$\text{Ginisplit}(\text{Micturition pains}) = 29/50(1 - (29/29)^2 - (0/29)^2) + 21/50(1 - (0/21)^2 - (21/21)^2) = 0$$

然后计算右子树,得到 $\text{Ginisplit}(\text{Urine pushing}) = 0$,那么最小的 $\text{Ginisplit}'(\text{Buring of urethra}) = 50/120\text{Ginisplit}(\text{Micturition pains}) + 70/120\text{Ginisplit}(\text{Urine pushing}) = 0$ 。

用同样的方法计算其他分割点的 $\text{Ginisplit}'$,选出最小的 $\text{Ginisplit}'(A^*)$, $A^* = \text{Buring of urethra}$ 时, $\text{Ginisplit}'(A^*)$ 最小,所以选择 Buring of urethra 作为第一层的分割属性。

5.3 改进算法应用于并行化环境

实验运用 Adult 数据集。实验过程中,不断添加物理机的个数,产生的折线图如图 4 所示。

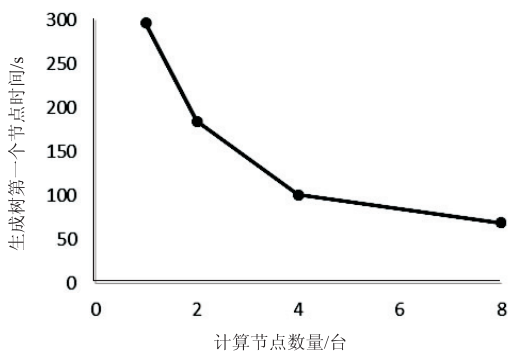


图 4 生成第一个节点和计算节点数量的关系

从图中可以看出,随着计算节点的不断增加,算法运行时间不断缩减。

6 结束语

文中对 SPRINT 算法进行了改进,当算法选取最佳分割属性时,不仅考虑该次分割的 Gini 指数,而且继续考虑下次分割带来的 Gini 指数。改进算法在层数上明显优于原算法。在算法的执行时间上,通过在云计算平台上增加计算节点,可以大大提高算法的执行速度。实验结果表明,文中提出的算法是有效的。

参考文献:

- [1] 李玲娟,张敏. 云计算环境下关联规则挖掘算法的研究[J]. 计算机技术与发展,2011,21(2):43-46.
- [2] 张晓洲. 云计算关键技术及发展现状研究[J]. 网络与信息,2011,25(9):36-37.
- [3] 王静红,王熙照,邵艳华,等. 决策树算法的研究及优化[J]. 微机发展(现更名:计算机技术与发展),2004,14(9):30-32.
- [4] Dean J, Ghemawat S. MapReduce: simplified data processing on large clusters[J]. Communications of the ACM, 2008, 51(1):107-113.
- [5] Settouti N, Aourag H. A comparative study of the physical and mechanical properties of hydrogen using data mining research techniques[J]. JOM, 2015, 67(9):1-9.
- [6] Hu Q H, Che X J, Zhang L, et al. Rank entropy-based decision trees for monotonic classification[J]. IEEE Transactions on Knowledge and Data Engineering, 2012, 24(99):1.
- [7] 王熙照,杨晨晓. 分支合并对决策树归纳学习的影响[J]. 计算机学报, 2007, 30(8):1251-1258.
- [8] Chen Jin, Luo Delin, Mu Fenxiang. An improved ID3 decision tree algorithm[C]//Proceedings of 2009 4th international conference on computer science & education. [s. l.]: [s. n.], 2009:127-130.
- [9] 韩松来,张辉,周华平. 决策树算法中多值偏向问题的理论分析[C]//全国自动化新技术学术交流会会议论文集(一). 出版地不详:出版者不详,2005.
- [10] Elyasigomari V, Mirjafari M S, Screen H R C, et al. Cancer classification using a novel gene selection approach by means of shuffling based on data clustering with optimization[J]. Applied Soft Computing, 2015, 35:43-51.
- [11] 王云飞. SPRINT 分类算法的改进[J]. 科学技术与工程, 2008, 8(23):6248-6252.
- [12] 董峰,刘远军. 数据挖掘中决策树 SPRINT 算法探讨[J]. 邵阳学院学报:自然科学版, 2007, 4(2):23-25.
- [13] 魏红宁. 基于 SPRINT 方法的并行决策树分类研究[J]. 计算机应用, 2005, 25(1):39-41.
- [14] Yang Shueng-Bien, Yang Shen-I. New decision tree based on genetic algorithm[C]//Computer control and automation. [s. l.]: [s. n.], 2010:115-118.