文章编号:1001 - 9081(2005)01 - 0039 - 03

基于 SPRINT 方法的并行决策树分类研究

魏红宁
(西南交通大学 校长办公室, 四川 成都 610031)
(weihn@swjtu.edu.cn)

摘 要:决策树技术的最大问题之一就是它的计算复杂性和训练数据的规模成正比,导致在大的数据集上构造决策树的计算时间太长。并行构造决策树是解决这个问题的一种有效方法。文中基于同步构造决策树的思想,对 SPRINT 方法的并行性做了详细分析和研究,并提出了进一步研究的方向。

关键词:数据挖掘;SPRINT 决策树分类;并行性
中图分类号: TP311.12 **文献标识码**:A

Study on the parallelism of decision tree classification based on SPRINT

WEI Hong-ning
(Administrative Office, Southwest Jiaotong University, Chengdu Sichuan 610031, China)

Abstract: One of the greatest problems with the decision tree technique is that the computational complexities are normally proportional to the size of training data set, so the computing time of constructing decision tree on large data sets is prohibitive. Parallelism is one effective solution to this problem. In this paper, parallel formulation of SPRINT decision tree algorithm based on synchronous tree construction approach was presented and a proposal of further study was given.

Key words: data mining; SPRINT decision tree classification; parallelism

分类是数据挖掘领域中一个非常重要的问题。至今已提出许多种分类方法,包括神经网络、遗传算法和决策树技术。其中,决策树技术最受欢迎。决策树分类技术的最大问题之一就是在大的数据集上构造决策树的计算时间太长。文献[1]的研究表明:找到一个优化决策树的计算复杂性是一个 NP 问题。当前,所有的决策树算法,如:C4、5[2]、SLIQ[3]、SPRINT[4]等,都使用局部启发式算法来减少计算复杂性。这些算法的计算复杂性从 $O(AN\log N)$ 到 $O(AN(\log N)^2)$,其中,A 是属性数目,N 是训练数据集中的记录数。它们在 N 相对较小时,构造分类决策树的速度够快,但在非常大的数据集上构造决策树的计算时间会很长,以至缺乏实际意义。并行性机制是解决这种问题的一种较好方法。

当前,已提出几种并行性分类决策树方法[5]。归纳起来,它们可以分为两类:一类方法为同步构造决策树方法,在此方法中,所有处理器通过传递和接受局部类分布信息来同步构造决策树;另一类方法为分组构造树方法,在此方法中,不同的处理器工作于决策树的不同部分,特别是,如果多于一个处理器共同扩展同一节点,那么这些处理器就要分组来处理该节点的后续子孙节点。

在并行数据挖掘研究中,一种主要的研究方法就是对现有的数据挖掘算法采用并行性处理,并在多处理器上运行。在当前主要的决策树算法中,SPRINT 算法是一种非常实用有效的决策树算法。本文就是针对 SPRINT 算法,运用同步构造决策树思想对 SPRINT 决策树的并行性进行研究。

1 SPRINT 算法

SPRINT 方法创建决策树的过程分为构建树和剪枝两个主要阶段。由于剪枝阶段所占的运算时间比例很小,本文对此不作详述。SPRINT 构建树阶段的主要过程如图 1 所示:

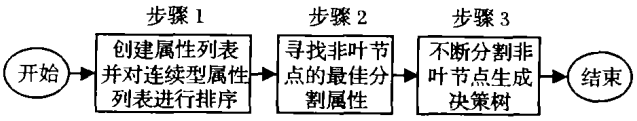

图 1 SPRINT 方法建树过程

本节的下面将对 SPRINT 算法采用了哪些主要关键技术及这些技术的特点做出分析,这种分析对于研究并行 SPRINT 方法非常有益。

1.1 数据结构

在 SPRINT 方法中有两个主要的数据结构:属性列表和类直方图(Class Histogram)。属性列表解决了以前决策树方法中需要训练数据完全驻留内存及训练数据多次排序的问题。而类直方图的作用主要用于计算对应节点的最佳分割属性。类直方图被用于表示决策树中一个节点上的记录所属类分布情况。对于每一个连续属性,都有两个相关的类直方图,分别为 C_{below} 和 C_{above},它们和属性列表中的相应扫描位置有关。C_{below} 类直方图表示符合 $A <= C$ 的所有记录的类分布情况,C_{above} 直方图表示符合 $A > C$ 的所有记录的类分布情况,其中 A 为某个连续属性,C 为一常量。对于种类属性,仅仅需要一个类直方图,它包含该种类属性每个值所对应的类分布情况。

初始的属性列表来自根节点对应的训练数据。在根节点处,对于连续型属性,属性列表通过属性值排序。在树的建立过程中,节点被不断分割,对应的属性列表也被分割,但属性列表的顺序被保留,因此,如果根节点的属性列是有序的,则在后续节点中,该属性列表也是有序的。SPRINT 就是通过这种方法解决以前决策树方法的多次排序问题。

1.2 最佳分割属性的选择

在 SPRINT 方法中,如何最好的分割某个节点是采用基尼指数(gini index)来度量的,这种度量方式可以简单叙述如

收稿日期:2004 - 06 - 25;修订日期:2004 - 11 - 24
作者简介:魏红宁(1966 -),男,高级工程师,主要研究方向:CAD/CAE、数据挖掘、数据库.

下：

如果集合 T 包含 n 个类别的 m 条记录，那么其基尼指数就是：

$$gini(T) = 1 - \sum_{j=1}^{n} P_j^2 \quad (1)$$

P_j 为类 j 出现的频率。

如果集合 T 分成两部分 $T1$ 和 $T2$，分别对应 $m1$ 和 $m2$ 条记录，那么这个分割的基尼指数就是：

$$gini_{分割}(T) = \frac{m1}{m}gini(T1) + \frac{m2}{m}gini(T2) \quad (2)$$

则提供最小分割的基尼指数就被选择作为集合 T 的最佳分割。

计算基尼指数的输入信息就是 1.1 节所谈到的类直方图，类直方图的信息来源于对每个属性列表的扫描。对于连续属性和种类属性的处理有所不同。

1.3 哈希表在节点分割中的使用

对于某个节点，在最佳分割属性确定之后，该属性可以直接被分割到相应子节点。而其他属性列表不能被直接分割，因为不同属性列表的记录排列顺序是不一样的。因此，在分割其他属性列表之前，首先要从该节点的最佳分割属性的分割信息中生成一个哈希表。哈希表指示该节点上的每一条记录属于哪一个子节点。利用哈希表，就可以对其他属性进行分割了。在 SPRINT 方法中采用哈希表，克服了 SLIQ 方法需要类列表常驻内存的缺陷。

2 SPRINT 算法的并行性研究

由于决策树剪枝阶段所占时间较少，因此，在本文中，只针对构造树阶段进行并行性研究。根据图 1 所示 SPRINT 构造树算法的主要步骤，对它的并行性研究主要集中在以下三方面：

（1）训练数据集在多处理器中的分布；

（2）并行确定节点的最佳分割属性；

（3）并行分割节点的属性列表到相应子节点。

2.1 训练数据集在多处理器中的分布

训练数据集在多处理器中如何分布对于并行 SPRINT 方法的性能、负载均衡以及处理器之间的通信费用影响较大。

Age	Class	Rid
23	Win	1
28	Linux	2
43	Win	3
38	Win	4
32	Win	5
20	Linux	6

(a) 训练数据集

Age	Class	Rid	Gender	Class	Rid	
20	Linux	6	F	Win	1	Processor0
23	Win	1	M	Linux	2	
28	Linux	2	F	Win	3	Processor1
32	Win	5	M	Win	4	
38	Win	4	F	Win	5	Processor2
43	Win	3	M	Linux	6	

(b) 多处理器中的数据分布

图 2 并行 SPRINT 算法中数据初始分布

在 SPRINT 方法中，主要的数据结构是属性列表和类分布图，它们从训练数据集中产生。在多处理器环境中，为了尽可能达到负载均衡，一个自然的想法就是通过水平分割的方法把 N 条训练数据集均匀地分配到 P 个处理器中，每个处理器得到 N/P 条记录。利用这种方法，训练数据集中的种类属性可以均匀地分组，无需进一步处理。但对于连续型属性列表，SPRINT 算法要求它是经过排序的。而在根节点处，由于训练数据集中连续型属性是无序的，因此，通过水平平均分割后，每个处理器中的连续型属性列表必须排序和重新分组，形成相邻的有序属性列表。这种排序分组算法可以采用文献[6]中的并行排序算法。图 2 显示了在三个处理器环境下的属性列表的初始分布的例子。

2.2 并行确定节点的最佳分割属性

怎样利用多个处理器并行确定节点的最佳分割属性是实现并行 SPRINT 算法的关键步骤。在 SPRINT 算法中，种类属性和连续型属性的分割特性的计算所需要的信息来源是不相同的。连续型属性需要两个类直方图 C_{below} 和 C_{above} 信息，种类属性只需要一个反映该属性全局类分布信息的直方图。因此，在并行计算某个节点最佳分割属性时，对这两类属性要分别进行处理。在某个节点的每个属性的最佳分割特性计算出来之后，每个处理器之间相互通信，交换信息，得到该节点的最佳分割属性。下面将分别讨论这两类属性分割特性的并行性计算方法。

2.2.1 并行计算连续型属性分割特性

根据 SPRINT 方法，在多处理器环境中，并行计算连续型属性分割特性主要由以下几个步骤完成：

（1）初始化 C_{below} 和 C_{above}。

由于每个处理器只有该属性局部列表信息，因此，每个处理器在初始化 C_{below} 和 C_{above} 时，必须和其他处理器交换信息，使得 C_{below} 能够反映在该处理器之上所有数据段的全局类分布情况，同样，C_{above} 能够反映在该处理器之下所有数据段的全局类分布情况。图 3 为图 2 中 *processor*1 的 C_{below} 和 C_{above} 初始化情况。

Processor1

Age	Class	Rid
28	Linux	2
32	Win	5

← Position0

Position0	Linux	Win
C_{below}	1	1
C_{above}	1	3

图 3 对应图 2 中 *processor*1 的 C_{below} 和 C_{above} 初始化情况

（2）扫描该属性列表的每条记录，更新 C_{below} 和 C_{above}。同时，根据当前 C_{below} 和 C_{above} 信息，计算对应分割点的基尼指数。

在这一步，并行 SPRINT 方法和串行 SPRINT 算法没区别。每个处理器之间无需通信，可以并行独立扫描自己局部记录信息，更新本处理器的 C_{below} 和 C_{above}，图 4 表示的是图 3 中 *processor*1 扫描一条记录后的 C_{below} 和 C_{above} 情况。

Processor1

Age	Class	Rid
28	Linux	2
32	Win	5

← Position1

Position1	Linux	Win
C_{below}	2	1
C_{above}	0	3

图 4 *processor*1 扫描一条记录后的 C_{below} 和 C_{above} 情况

同时，根据当前 C_{below} 和 C_{above} 的信息，计算对应分割点的基尼指数。如：图 4 中 *position*1 的基尼指数计算如下：

$$GINI(position1) = (3/6) * (1 - (2/3)^2 - (1/3)^2) + (3/6) * (1 - (3/3)^2) = 2/9 \quad (3)$$

同理,对应 *processor*1 中的 *position*0 的基尼系数为:

$$GINI(position0) = (2/6)*(1-(1/2)^2-(1/2)^2)+(4/6)*(1-(1/4)^2-(3/4)^2) = 5/12 \quad (4)$$

比较 $GINI(position1)$ 和 $GINI(position0)$,可以得到 *processor*1 所处理的 Age 属性的局部最佳分割点为 *position*1 点。同理,*processor*0 和 *processor*2 可以计算出它们局部的最佳分割点。

(3) 选取该属性的最佳分割点。

比较各处理器局部最佳分割点,选取具有最小基尼指数的分割点作为该属性全局最佳分割点。

2.2.2 *并行计算种类属性分割特性*

在 SPRINT 算法中,种类属性的最佳分割点的计算是由该属性全局类分布信息决定的。因此,在多处理器环境中,如何并行得到该属性全局类分布信息是计算该种类属性最佳分割点的关键。具体算法如下:

(1) 每个处理器根据自己局部的记录信息建立该属性局部的类分布信息 C_{local}。

(2) 选取一个处理器作为协调处理器(coordinator),从每个处理器中收集该属性局部类分布信息 C_{local} 相加得到该属性全局类分布信息 C_{global},如图 5 所示。

Processor0

	Linux	Win
F	0	1
M	1	0

Processor1

	Linux	Win
F	0	1
M	0	1

Processor2

	Linux	Win
F	0	1
M	1	0

Coordinator

	Linux	Win
F	0	3
M	2	1

图 5　并行环境下种类属性类分布信息处理

(3) 基于协调处理器中该属性全局类分布信息 C_{global},计算它的最佳分割点。

2.3 并行分割节点的属性列表到相应子节点

根据 SPRINT 算法,在最佳分割属性被分割到子节点后,其他属性如何分割是根据哈希表的。哈希表是基于最佳分割属性的分割信息产生的。

在多处理器环境中,每个处理器中的任一属性记录都可能分割到子节点中的任何一个。因此,每个处理器如何分割其余属性列表必须基于全局哈希表。以下就是并行分割属性列表的方法。

(1) 基于最佳分割点信息,每个处理器独立分割自己局部的最佳分割属性列表,并创建局部的哈希表;

(2) 每个处理器之间互相交换自己的局部哈希表信息,并在每个处理器中独立构建全局的哈希表;

(3) 基于全局的哈希表,每个处理器独立地分割自己局部的非分割属性到相应子节点中。

3 算例

本算例选择 PC CLUSTER 作为实验平台,包含 8 个节点。每个节点只使用两个 CPU,256MB 内存和一个局部硬盘。每个节点之间通过 100 BASE-TX 网络相连。操作系统使用 Linux。采用 MPI 为消息传递。文献[7]中 synthetic 数据库中的 function 2 数据集作为本实验的 benchmark 数据。

在实验中,输入数据首先被放在每个节点的局部硬盘中,然后进行初始属性列表生成、排序等前处理工作。本实验的目的主要是看随着处理节点的增加生成决策树的时间变化情况,所以,本文忽略前处理时间。

实验结果如图 6 所示:

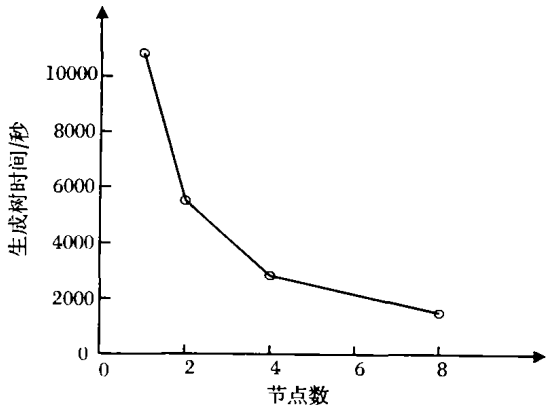

图 6　生成树时间和处理节点树之间的关系

从图 6 可以看出,随着处理节点的增多,决策树产生时间在下降。可以分析得到,如果本算例采用千兆以太网来连接每个节点并且每个节点采用多个 CPU 方式,本算法性能将会得到较大的提高。

4 结语

决策树方法的并行性是当前解决大规模数据分类问题的研究重点。SPRINT 方法是一种非常流行的决策树方法,它不依赖于任何集中存放和驻留内存的数据结构,因此,SPRINT 算法的并行性实现就非常自然和容易实施。

但在我们研究 SPRINT 算法的并行性实现过程中发现,简单地对 SPRINT 算法进行并行性实现时,有些问题还需要进一步处理。例如,在分割某个节点的非分割属性列表时,反映该节点分割信息的完整哈希表需要在所有处理器中存储。而该哈希表的大小和该节点的记录数 N 成正比,为 O(N),并不依赖处理器的数目。这样就使得每个处理器所需要的内存也为 O(N)。而且,产生该哈希表需要每个处理器之间都要进行局部哈希表信息交换,这种通信费用也为 O(N)。因此,简单地将 SPRINT 算法进行并行性实现,在运行时间和内存需求方面所表现的伸缩性能并不好。所以,在进一步对 SPRINT 算法并行性研究工作中,这些问题将是今后研究的重点。

参考文献:

[1] HAN EH, SRIVASTAVA A, KUMAR V. Parallel formulation of inductive classification learning algorithm[R]. Minneapolis, USA: University of Minnesota, 1996.

[2] QUINLAN R. C4.5: Programs for Machine Learning[M]. San Mateo, CA: Morgan Kaufmann, 1993.

[3] MEHTA M, AGRAWAL R, RISSANEN J. SLIQ: A Fast Scalable Classifier for Data Mining[A]. Proceedings of EDBT-96[C]. Berlin, Germany: Springer Verlag, 1996. 18 - 32.

[4] SHAFER J, AGRAWAL R. MEHTA M. SPRINT: A Scalable Parallel Classifier for Data Mining[A]. Proceedings Of the 22nd International Conference on Very Large Databases[C]. San Mateo, CA: Morgan Kauffman, 1996. 544 - 555.

[5] SRIVASTAVA A, HAN EH, KUMAR V, *et al.*. Parallel Formulations of Decision Tree Classification Algorithms[J]. Data Mining and Knowledge Discovery, 1999, 3(3): 237 - 261.

[6] DEWITT D, NAUGHTON J, SCHNEIDER D. Parallel Sorting on a Shared-nothing Architecture using Probabilistic Splitting[A]. Proceedings First international Conference on Parallel and Distributed Information Systems[C]. New York: IEEE Press, 1992. 280 - 291.

[7] AGRAWAL R, IMIELINSKI T, SWAMI A. Database Mining: A Performance Perspective[J]. IEEE Transaction on Knowledge and Data Engineering, 1993, 5(6): 914 - 925