基于 Spark 的 CVFDT 分类算法并行化研究

庄 荣,李玲娟
(南京邮电大学 计算机学院,江苏 南京 210023)

摘 要:以提升流数据的分类挖掘效率为目标,研究将概念适应快速决策树算法(CVFDT)部署到流数据计算平台 Spark 上进行并行化的方案。设计了 CVFDT 基于 Spark 的并行化实现方案,首先对 CVFDT 算法进行属性间并行化改造,即分割点计算过程中的并行化;然后基于 Spark 在 CVFDT 的建树过程中将节点的所有属性列表转化为 Spark 特有的弹性分布式数据集 RDD,通过计算由每个 RDD 生成的并行化任务,汇总并且比较每个最佳分割点,再计算 Hoeffding 边界作为节点分裂条件找到最佳分割点,从而递归创建决策树。实验结果表明,在 Spark 集群环境下,CVFDT 算法的分类效率相对于单机环境有显著提高,改进后的并行化 CVFDT 算法对大规模流数据处理有良好的适应能力,而且合理设定 RDD 过滤可使分类效率进一步提高。

关键词:数据流;CVFDT;并行化;Spark;弹性分布式数据集
中图分类号:TP301.6 **文献标识码**:A **文章编号**:1673-629X(2018)06-0035-04
doi:10.3969/j.issn.1673-629X.2018.06.008

Research on Parallelization of Concept-adapting Very Fast Decision Tree Classification Algorithm Based on Spark

ZHUANG Rong, LI Ling-juan
(School of Computer, Nanjing University of Posts and Telecommunications, Nanjing 210023, China)

Abstract: Aiming at increase of classification and mining efficiency for stream data, we study a parallelization scheme of deploying the CVFDT (concept-adapting fast decision tree) to the stream data computing platform Spark and design a implementation scheme of CVFDT based on Spark. Firstly, the CVFDT should be parallelized among attributes, that is the parallelization of the splitting point calculation. Then in the process of building decision trees of CVFDT based on Spark, all the attribute lists of the node are transformed into Spark's unique resilient distributed datasets (RDD), and through calculation of parallel task from each RDD, each optimal splitting point is summarized and compared. The Hoeffding boundary is calculated as the node splitting condition to find the optimal splitting point, and the decision tree is recursively created. The experiment shows that the classification efficiency of CVFDT in the Spark cluster environment relative to the stand-alone environment has improved significantly. The improved parallel CVFDT has better adaptability to large-scale stream data processing and the reasonable setting of RDD filtering can further improve the classification efficiency.

Key words: data streams; CVFDT; parallelization; Spark; resilient distributed datasets

0 引 言

面对实时到达、连续、无限的流数据[1],传统的数据挖掘算法难以满足流数据挖掘的需求,因而流数据的分类挖掘[2]研究一直是热门话题并且具有重大意义。概念适应快速决策树算法(concept-adapting very fast decision tree, CVFDT)[3]是经典的流分类算法之一。CVFDT 主要克服了 VFDT(very fast decision tree)算法[4]对于数据样本的不断变化而不能更换模型的缺点,并且可以有效地解决概念漂移[5]的问题。与传统的静态大数据处理平台 Hadoop[6]不同,Spark[7]扩展了广泛使用的 MapReduce[8]模型,提出了基于内存的并行计算框架,通过将中间结果缓存在内存中以减少 I/O 磁盘操作,从而更高效地支持多次迭代式计算模式。为此,文中研究了基于 Spark 的 CVFDT 分类算法并行化,用以提高 CVFDT 算法对流数据的分类效率。

收稿日期:2017-07-10 **修回日期**:2017-11-15 **网络出版时间**:2018-02-24
基金项目:国家自然科学基金(61302158,61571238)
作者简介:庄 荣(1992-),男,硕士研究生,CCF 会员(E200052408G),研究方向为数据挖掘;李玲娟,教授,通讯作者,CCF 会员(E200015),研究方向为数据挖掘、信息安全、分布式计算。
网络出版地址:http://kns.cnki.net/kcms/detail/61.1450.TP.20180224.1521.070.html

1 CVFDT 算法分析

1.1 CVFDT 简介

CVFDT 属于一种增量式的分类挖掘方法,即用新到样本修正旧分类器,产生新分类器,以适应新环境。CVFDT 在树的所有节点上维持统计信息用于计算基于属性值的信息增益测试,即统计测试,并基于 Hoeffding 不等式确定叶节点变成分支节点所需的样本数目,对数据流建立分类决策树。

以 t 为时间戳,$\boldsymbol{x}_t$ 表示 t 时刻到达的数据向量,数据流可表示为$\{\cdots,\boldsymbol{x}_{t-1},\boldsymbol{x}_t,\boldsymbol{x}_{t+1},\cdots\}$[9]。CVFDT 算法的有关定义如下:

(1)信息增益[10]:叶子节点 1 中存储训练样本集 D 的统计信息,则对于样本集 D 分类所需的期望信息为:$\mathrm{Info}(D)=-\sum_{i=1}^{m}p_i\log_2(p_i)$。其中,$p_i$ 是样本集 D 中任意一条样本属于类 C_i 的概率,$p_i=|C_{i,D}|/|D|$,m 是类别属性的取值个数。对于叶子节点可能的分裂属性 A 有 v 个取值,利用属性 A 对样本集 D 进行划分的期望信息为:$\mathrm{Info}_A(D)=-\sum_{j=1}^{v}\frac{|D_j|}{|D|}\times\mathrm{Info}(D_j)$,属性 A 的信息增益为 $\mathrm{Gain}(A)=\mathrm{Info}(D)-\mathrm{Info}_A(D)$。

(2)Hoeffding[11]约束:对一个真值随机变量 r,其取值范围为 R,对 r 取了 n 个独立的观察值,平均值为 $\bar{r}$,则 Hoeffding 约束以置信度 $1-\delta$ 保证变量 r 的真值落于 $\bar{r}\pm\varepsilon$ 范围内,且 $\varepsilon=\sqrt{\frac{R^2\ln(1/\delta)}{2n}}$。由 Hoeffding 约束可知,CVFDT 决策树的某个叶节点到达 n 个新样本后,假设 X_a 和 X_b 分别是该叶节点中属性增益值最大和次大的属性,如果属性 X_a 和 X_b 的信息增益之差$>\varepsilon$,则可以证明属性 X_a 为信息增益最大的属性的置信度为 $1-\delta$。此时,该叶节点就可变为分支节点,其属性为 X_a。

1.2 CVFDT 算法流程

在拿到新的数据流样本后,从上到下遍历决策树,并在树的每个分支节点根据属性取值等判断进入不同的分支,最终到达树的叶节点。随着数据流样本的不断增多,信息增益测试为了满足一定条件,其叶节点必须要以较高的置信度确定最佳划分属性,从而变成一个分支节点,这样循环可以不断地决策学习新的叶节点。如遇到概念漂移问题,CVFDT 就会在相应分支节点上并行生成备选子树,原子树会随着备选子树的精度远远超过其本身时被替换和释放。

2 Spark 平台

Spark 不同于 Hadoop MapReduce 的是,Job 中间输出结果可以保存在内存中,从而不再需要频繁读写 HDFS[12],可以显著提高运行速度。Spark 还提供了 SparkSQL、Spark Streaming[13]、MLib[14] 等计算模式组件,更适用于分布式平台场景。Spark Streaming 是构建在 Spark 上的处理 Stream 数据的框架,基本原理是将 Stream 数据分成小的时间片断(几秒),以类似 batch 批量处理的方式来处理这小部分数据。

弹性分布式数据集 RDD[15] 是 Spark 的核心和基础。RDD 是一种分布式的内存抽象,表示只读的分区记录的集合。它只能通过在稳定物理存储中的数据集或其他已有的 RDD 上执行一些确定性操作(并行操作中的转换操作)来创建,并且 RDD 仅支持粗粒度转换,即在大量的记录上执行单一的操作,因此省去了大量的磁盘 I/O 操作,对于需要多次迭代计算的机器学习算法、交互式数据挖掘来说,效率得到了极大地提升;同时它具有非常出色的容错机制和调度机制,能够有效确保系统的长时间稳定运行。所以 Spark 能够更高效地支持交互式查询以及迭代式计算等多种计算模式。

3 CVFDT 基于 Spark 的并行化方案设计

3.1 CVFDT 的分割点计算过程的并行化

在 CVFDT 建树过程中计算最佳分割点时,需要将 Hoeffding 边界作为节点分裂条件找到最佳分割点,其首要任务是计算并比较各个属性的最佳基尼分割指数。在面对含多种属性类别的数据集时,线性串行的计算模式会大大降低运行效率。

因此,针对 CVFDT 算法的分割点计算过程,考虑到每个属性的基尼分割指数求解是完全独立计算,可以对这些计算进行同步,设计了如图 1 所示的属性间并行化方案。

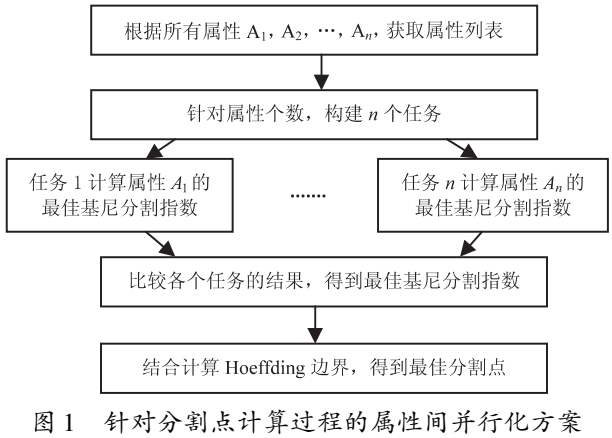

图 1 针对分割点计算过程的属性间并行化方案

如图 1 所示,首先计算每个任务的最佳基尼指数和 Hoedding 边界,从而找到每个任务的最佳分割点;然后在每个任务计算完成后进行比较,获取最佳分割点。这种改进的计算模式可以有效地降低时间复杂度。

3.2　基于 Spark 的 RDD 实施 CVFDT 的并行化

1. Spark 的并行计算过程简述。

RDD 为了让海量数据分散在不同的计算节点上进行并行处理,会横向地拆分成 N 个分区,因此对 RDD 进行计算操作时,集群会对每个分区进行计算,然后由相应的集群控制器将结果进行汇总,最终统计整个 RDD 结果。因此,Spark 的并行属于“横向”并行化。

2. 针对建树过程的基于 Spark 的横向并行化。

在 3.1 提出的属性间并行化的基础上,基于 Spark 的横向并行化,针对 CVFDT 算法的建树过程做如下并行化改造:

(1)If(都是同类属性的数值||获取的属性列表个数不超过阈值)

(2)将含有同属性最多数值的类复制给节点 N 的 Decision 类并且返回;

(3)Else

(4)得到节点 N 的属性列表 AttList,将所有属性列表转化为对应的 RDD;

(5)计算由每个 RDD 生成的并行化任务,汇总并比较每个最佳分割点;

(6)再计算 Hoeffding 边界产生节点分裂条件,找到最佳分割点;

(7)在 AttList 中找到该分割点相应属性的属性列表并删除,然后对其余属性表进行分裂,得到属性表 $\text{Attlist}_1, \text{Attlist}_2, \cdots, \text{Attlist}_N$;

(8)新的子节点 $N_1, N_2, \cdots, N_n$ 由节点 N 生成,并将属性列表 $\text{Attlist}_1, \text{Attlist}_2, \cdots, \text{Attlist}_N$ 分别赋给 $N_1, N_2, \cdots, N_n$;

(9)执行 buildTree(n_1),buildTree(n_2),…,buildTree(n)操作,递归建立决策树。

4　实验结果与分析

选择传统单机 CVFDT 算法和基于 Spark 集群的 CFVDT 并行化算法对实验数据集进行分类操作,算法运行环境是:

集群硬件环境:1 个 Master 节点,2 个 Slave 节点。

集群操作系统:centos6.6。

集群软件环境:JRE1.7.0_13、Scala_2.11.6、Spark1.3.1。

单机环境:eclipse_4.5.0、JRE1.7.0_13、Windows7、2.13 GHz、4 GB 内存。

目的是借助流数据处理平台提高 CVFDT 算法的执行效率。为了验证所设计的 CVFDT 算法基于 Spark 的并行化方案的可行性和有效性,对比了单机和集群并行环境下,CVFDT 算法处理不同数据量所需的时间。

实验使用的数据集源自 Kaggle 比赛,基于美国 UGC 网站 Stumbleupon 提供的历史数据,设计分类模型,预测该网站提供的网页是否长期流行。训练集样本数目为 10 706 个,测试集样本大小为 5 171 M。

4.1　建树效率的比较

为了考虑算法的实用性,选取了 200k、300k、500k、800k、1 000k 条这五种不同规模的数据集,测试结果如表 1 所示。

表 1　建树时间测试结果

数据大小/条	现有算法/s	并行化算法/s	综合效率提高/%
200k	67	66.9	0.13
300k	110	106.4	3.37
500k	197	187.1	5.28
800k	310	289.6	7.01
1 000k	401	367.4	9.14

当选取 200k 条规模数据集时,算法(建树)效率提升微乎其微,并且 10 000 条数据规模下,建树时间反而增加。原因是资源管理、网络传输会伴随着 Spark 运行集群而产生额外开销。当数据规模达到 300k 以上时,或者海量数据规模时,基于 Spark 的 CVFDT 并行化有明显的效率提升。

4.2　数据处理时间的比较

图 2 对比了单机和 Spark 集群环境下在数据规模分别为 200 M、300 M、500 M、800 M、1 000 M 时所需的时间。

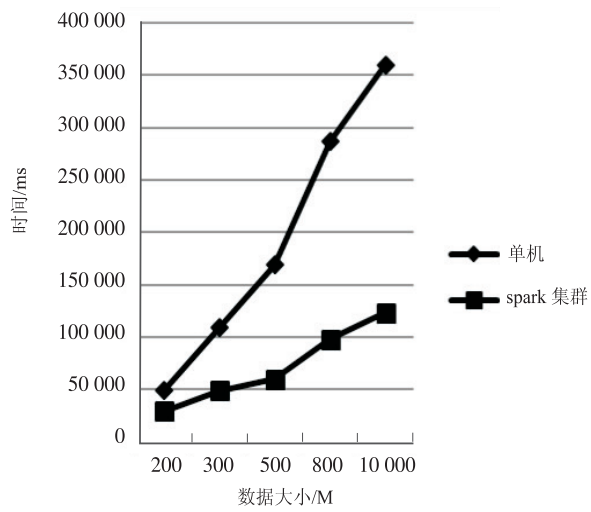

图 2　对应不同数据量的处理时间测试结果

在单机环境下,随着数据规模的扩大,数据处理时间急剧增加;在 Spark 集群环境下,在 200 M 数据规模下,处理时间提升不明显,除了上一节提到的原因之外,主要还存在求解每个分裂条件的基尼系数时,会依

照分裂条件对 RDD 进行过滤,然后再调用 count( ) 函数来统计个数,每一次的 count 操作都是在 Spark 集群中完成,然后将计算结果传输给虚拟机。当属性列表数据条数 N 不是很大时,Spark 的优势无法体现。300 M 数据量之后,可以看到并行化算法的数据处理时间明显减少,当数据量到 1 000 M 时,处理时间缩减了 66.6% 。

4.3 测试结果分析

由表 1 和图 2 可以看出,基于 Spark 集群的并行化 CVFDT 算法在处理规模较大的流式数据时,运行效率有所提高,并且在数据规模增大时,其效果会越发明显。并且并行化 CVFDT 算法相对于单机环境在处理海量数据时处理效率有显著提高,而且合理设定 RDD 过滤可使处理效率进一步提高。

5 结束语

将经典的流数据分类挖掘算法 CVFDT 部署于流数据处理平台 Spark 上,借助构建在 Spark 之上的实时计算框架 Spark Streaming 来实现对流数据的并行化分类。对 CVFDT 算法进行了属性间并行化改造,并且基于 Spark 的 RDD 进行了 CFVDT 算法在建树流程上的横向化并行。测试结果证明了该设计思想的正确性和方案的有效性,也说明了基于 Spark 的并行化 CVFDT 算法对大规模流数据有良好的适应能力。

参考文献:

[1] DING Shifei, WU Fulin, QIAN Jun, et al. Research on data stream clustering algorithms[J]. Artificial Intelligence Review, 2015, 43(4): 593-600.

[2] ABURROUS M, HOSSAIN M A, DAHAL K, et al. Predicting phishing websites using classification mining techniques with experimental case studies[C]//7th international conference on information technology. Las Vegas, NV, USA: IEEE, 2010: 176-181.

[3] 王 涛,李舟军,颜跃进. 一种基于哈希链表的高效概念漂移连续属性处理算法[J]. 计算机工程与科学, 2008, 30(8): 65-68.

[4] 袁 磊,张 阳,李 梅,等. 在数据流管理系统中实现快速决策树算法[J]. 计算机科学与探索, 2010, 4(8): 673-682.

[5] MINKU L L, WHITE A P, YAO Xin. The impact of diversity on online ensemble learning in the presence of concept drift[J]. IEEE Transactions on Knowledge & Data Engineering, 2010, 22(5): 730-742.

[6] DITTRICH J, QUIANÉRUIZ J A, JINDAL A, et al. Hadoop++: making a yellow elephant run like a cheetah (without it even noticing)[J]. Proceedings of the VLDB Endowment, 2010, 3(1-2): 515-529.

[7] 黎文阳. 大数据处理模型 Apache Spark 研究[J]. 现代计算机, 2015(8): 55-60.

[8] 沈 超,邓彩凤. 论 Storm 分布式实时计算工具[J]. 中国科技纵横, 2014(3): 53.

[9] RAAHEMI B, ZHONG Weicai, LIU Jing. Peer-to-peer traffic identification by mining IP layer data streams using concept-adapting very fast decision tree[C]//Proceedings of the 20th IEEE international conference on tools with artificial intelligence. Dayton, OH, USA: IEEE, 2008.

[10] 张发扬,李玲娟,陈 煜. VFDT 算法基于 Storm 平台的实现方案[J]. 计算机技术与发展, 2016, 26(9): 192-196.

[11] YIN Chunyong, FENG Lu, MA Luyu, et al. A feature selection algorithm of dynamic data-stream based on Hoeffding inequality[C]//International conference on advanced information technology and sensor application. [s. l.]: IEEE, 2015: 92-95.

[12] 欧阳永. 运营商大数据系统建设的分析与研究[D]. 南京: 南京邮电大学, 2016.

[13] 管祥青. 大数据可视化模型的协同过滤算法研究及应用[D]. 长沙: 湖南大学, 2015.

[14] XIN R S, GONZALEZ J E, FRANKLIN M J, et al. GraphX: a resilient distributed graph system on Spark[C]//International workshop on graph data management experiences and systems. New York, NY, USA: ACM, 2013.

[15] 刘志强,顾 荣,袁春风,等. 基于 SparkR 的分类算法并行化研究[J]. 计算机科学与探索, 2015, 9(11): 1281-1294.