Bootstrap样本大数据模型和分布式集成学习方法

罗凯靖[1]，张育铭[1]，何玉林[2]，黄哲学[1,2]
1. 深圳大学计算机与软件学院大数据技术与应用研究所，广东 深圳 518060；
2. 人工智能与数字经济广东省实验室（深圳），广东 深圳 518107

摘要

传统Bootstrap抽样和Bagging集成学习通常以串行方式实现，计算效率低，且存在样本不可重用、扩展性差等问题，不适合高效的大规模Bagging集成学习。从大数据分布式计算的思维入手，提出新的Bootstrap样本划分（BSP）大数据模型和分布式集成学习方法。BSP数据模型通过分布式生成算法将训练数据表达成分布式Bootstrap样本集的集合，存储成HDFS分布式数据文件，为后续的分布式集成学习提供数据支持。分布式集成学习方法从BSP数据模型中随机选取多个BSP数据块，读入集群各个节点的虚拟机，用串行算法对选取的数据块并行计算统计量或训练建模，再将所有的计算子结果回传至主节点中，生成最终的集成学习结果，此过程中可加入对子结果的质量选择以进一步提高预测效果。BSP数据模型的生成和分布式集成学习采用非Map-Reduce计算范式进行，每个数据块的计算独立完成，减少了计算节点间的数据通信开销。提出的算法在Spark开源系统中以新的算子方式实现，供Spark应用程序调用。实验表明，新方法可以高效地生成训练数据的BSP数据模型，提高数据样本的可重用性，在基于有监督机器学习算法构建的大规模Bagging集成学习实验中，计算效率能提高50%以上，同时预测精度进一步提高约2%。

关键词

Bootstrap抽样；Bagging集成学习；分布式集成学习；Spark

中图分类号：TP319 文献标志码：A doi: 10.11959/j.issn.2096-0271.2024002

Bootstrap sample partition data model and distributed ensemble learning

LUO Kaijing[1], ZHANG Yuming[1], HE Yulin[2], HUANG Zhexue[1,2]
1. Big Data Institute, College of Computer Science and Software Engineering, Shenzhen University, Shenzhen 518060, China
2. Guangdong Laboratory of Artificial Intelligence and Digital Economy (SZ), Shenzhen 518107, China

Abstract

A sequential implementation of Bootstrap sampling and Bagging ensemble learning is computationally inefficient and not scalable to build large Bagging ensemble models with a large number of component models. Inspired by distributed big data computing, a new Bootstrap sample partition (BSP) big data model and a distributed ensemble learning method

for large-scale distributed ensemble learning were proposed. The BSP data model extended a dataset as a set of Bootstrap samples stored in Hadoop distributed file system. Our distributed ensemble learning method randomly selected a subset of samples from the BSP data model and read them into Java virtual machines of the cluster. Following this, a serial algorithm was executed in each virtual machine to process each sample data and build a machine learningmodel on each sample data independently and in parallel with other virtual machines. Eventually, allsub-results were collected and processed in the master node to produce the ensemble result, optionally adding a sample preferences trategy for the BSP data blocks. The BSP data model generation and the component model building were computed using a non-MapReduce computing paradigm. All component models were computed in parallel without data communication among the nodes. The algorithms proposed in this paper were implemented in spark as internal operators that can be utilized in Spark applications. Experiments have demonstrated that BSP data model of a dataset can be generated efficiently through the new distributed algorithm. It improves the reusability of data samples and increases computational efficiency by over 50% in large-scale Bagging ensemble learning, while also increasing prediction accuracy by approximately 2%.

Key words
Bootstrap sampling, Bagging, distributed ensemble learning, Spark

0 引言

Bagging集成学习方法[1]的基础是Bootstrap抽样[2]。在Bagging方法中，利用Bootstrap方法从原始数据集中采取有放回随机抽样得到多个记录条数与原始数据集相等的Bootstrap样本集，分别用不同的Bootstrap样本集来训练出多个子模型，对每个子模型的结果进行集成得到最终结果[3-4]，其精度和鲁棒性均高于由原始数据集建立的单个模型[4]，它主要用于分类或者回归问题以获取比较稳定的结果，同时降低模型的过拟合程度[3]。

Bagging集成学习算法常用较容易实现的串行程序进行计算[5-6]，基本程序架构由内循环和外循环组成，每次内循环做n次有放回抽样，产生一个样本集，建立一个子模型；外循环重复m次内循环，完成m个Bootstrap样本集的生成和m个子模型的建模，最后生成集成模型。随着数据规模的增大，传统的Bagging方法在运算时间和内存消耗等方面面临着很大的挑战。Bagging集成学习具有良好的并行性[1,5]，随着计算机硬件性能的提高，Bagging方法的并行化实现已经成为可能，只要把外循环放在m个节点或虚拟机上并行计算，然后把每个Bootstrap样本的计算结果传送到主节点做集成计算，即可得到集成结果，并行计算可以提高计算效率。

近年来出现了一些关于并行化Bagging的尝试和改进。Chen等人[7]通过分布式云平台针对大数据进行并行化决策树算法来提高训练速度和扩展性；Wei等人[8]则利用MapReduce分布式计算框架来改进随机森林模型以获得较高的分类效率；Senagi等人[9]在GPU上并行构建随机森林模型，该方法在处理高维数据方面具有良好的性能；Lyu等人[10]构建了基于在线Bagging集成方法的分类器用于进行大数据流的分类建模学习。这些工作在一定程度上提高了学习器的训练速度和扩展能力，但是仍然存在一些问题和局限性：①对于在指定较大的块数建立大的集成模型以及使用复杂建模算法时，计算效率和集成模型扩展能力受集群内存资源限制，程序运行效率不高，不适合大

规模Bagging集成学习；②对于同一训练数据集的不同分析任务，每次运行都要重复进行Bootstrap抽样操作，Bootstrap样本数据不可重用，同时，每次模型训练前都要在线生成Bootstrap样本，浪费计算资源；③目前的工作往往只聚焦于某一特定基算法来进行并行优化和改进，缺乏系统性地和工程性地提出可适用于多种不同算法的Bagging集成学习的分布式数据模型和集成框架。其他类似的并行集成学习方法存在同样问题，例如：Kleiner等人[11]提出的BLB（bag of little bootstraps）算法，首先进行无放回抽样，然后再进行Bootstrap抽样，以及Basiri等人[12]提出的FBR（frequency based replacement）并行计算改进算法等。

受人工智能预训练模型和大数据随机样本划分（random sample partition，RSP）数据模型的启发[13-15]，本文提出了Bootstrap样本划分（Bootstrap sample partition，BSP）大数据模型和分布式集成学习计算方法，其核心思想是通过执行分布式方法预先生成大量的Bootstrap样本集，称为BSP数据块，构成BSP数据模型，在不同的Bootstrap分析和集成学习任务中重用，使用分布式集成学习进行并行训练，提高Bagging集成学习的计算性能和建模能力。该方法的优点如下：

- 原始数据T的BSP数据模型只需要一次性生成，可重复使用；
- 利用分布式计算的优势，可以指定一个很大的块数M来高效生成BSP数据模型，支持高效的大规模集成模型建模和Bootstrap统计分析，提高分析结果的精度和稳定性；
- BSP数据模型支持对T的多种不同的分析和建模任务，每次算法的运行可选用不同的多个BSP数据块；
- 可加入对BSP数据块的质量选择策略，进一步提高分析的精度。

新的分布式集成学习计算方法已经在ApacheSpark开源软件[16]上通过扩展其分布式计算框架和开发新算子实现。创新性工作包括：①BSP数据模型的分布式生成算法和转换算子toBSP；②BSP数据块的随机读取方法BspRead、样本优选方法selectPartitions和内存模型BspRDD；③单个样本集串行的分析和建模算法函数；④非Map-Reduce分布式计算范式，减少了算法执行时节点间的数据通信，提高了计算效率，BSP数据块直接采用高优化的高效的串行机器学习算法进行训练，不再需要将程序算法并行实现。通过Spark应用程序完成对数据集的BSP数据模型生成、Bootstrap统计分析和集成模型建模，具体实例在后面章节给出。仿真数据和真实数据的实验结果表明，新的分布式集成学习计算方法在计算效率、数据规模、模型扩展能力、应用的灵活性等方面比现有的串行软件有很大提升；同时，集成模型比单样本模型的精度和鲁棒性都有提高，分布式计算的效率也大幅提高。

1 BSP数据模型及其分布式生成算法

1.1 BSP数据模型

原始样本数据集$T=\{X_1,X_2,\cdots,X_n\}$的BSP数据模型定义为M个独立生成的Bootstrap样本集的集合，记作BSP(T)=$\{T_1^*,T_2^*,\cdots,T_M^*\}$，其中，$T_i^*$是对$T$做一次Bootstrap抽样后得到的一个有$n$条记录的Bootstrap样本集。对$T$重复做$M$次Bootstrap抽样，生成$M$个Bootstrap样本集，当$M$很大时，BSP($T$)可以看成一个大

数据集，则 $T_1^*,T_2^*,\cdots,T_M^*$ 是大数据集的一个划分，BSP(T)被称作 T 的Bootsrap样本划分大数据模型，简称BSP数据模型，而每个 T_i^* 被称作BSP数据块。

在Hadoop分布式文件系统（Hadoop distributed file system，HDFS）[17]中，BSP数据模型的存储方式表示为一个HDFS文件目录，BSP数据模型的每个BSP数据块为目录下的一个子目录。如**图1**所示，BSP(T)记作一个HDFS文件目录BSP-HDFS/；$T_1^*,T_2^*,\cdots,T_M^*$ 是BSP-HDFS/目录下的M个子目录，记作/part1/、/part2/、……、/part$M$/。在每个BSP数据块子目录下存储一个或多个不超过128 MB的数据文件，称作BSP数据块的数据文件，每个文件有唯一的块ID。在读取BSP数据块时，每个子目录下的所有数据文件同时读入弹性分布式数据集（resilient distributed datasets，RDD）[18]的同一分区（partition）中。

后续进行统计分析和集成学习操作时，从HDFS文件目录中随机选取m(m<<M)个子目录，m个子目录的数据文件读入RDD的m个分区中，以分区为单位并行计算统计量或建模，将每个分区的子结果传送到集群主节点做集成计算，得到最终的集成结果。

1.2 BSP数据模型生成算法

在Spark开源软件上实现了BSP数据模型的生成算法和分布式存储。首先定义Spark中代表BSP数据模型的BspRDD，每个分区对应一个Bootstrap样本集。从数据集T生成M个Bootstrap样本集的生成算法步骤如下：

- 将T读入初始RDD数据结构HadoopRDD，HadoopRDD的分区数量根据文件T的大小和Spark集群资源配置自动确定；
- 用coalesce算子将多个分区的HadoopRDD转成一个分区的CoalescedRDD；
- 用union算子将CoalescedRDD进行复制操作生成M个相同的RDD，并链接合并成UnionRDD；
- 用mapPartitions算子将Bootstrap放回抽样函数bootstrapFunc分发到每个分区进行独立Bootstrap抽样，生成M个Bootstrap样本集，结果存储在BspRDD；
- 将代表一个完整的BSP数据模型的BspRDD按照**图1**所示结构写入HDFS分布式文件。

其中，bootstrapFunc函数是用Scala

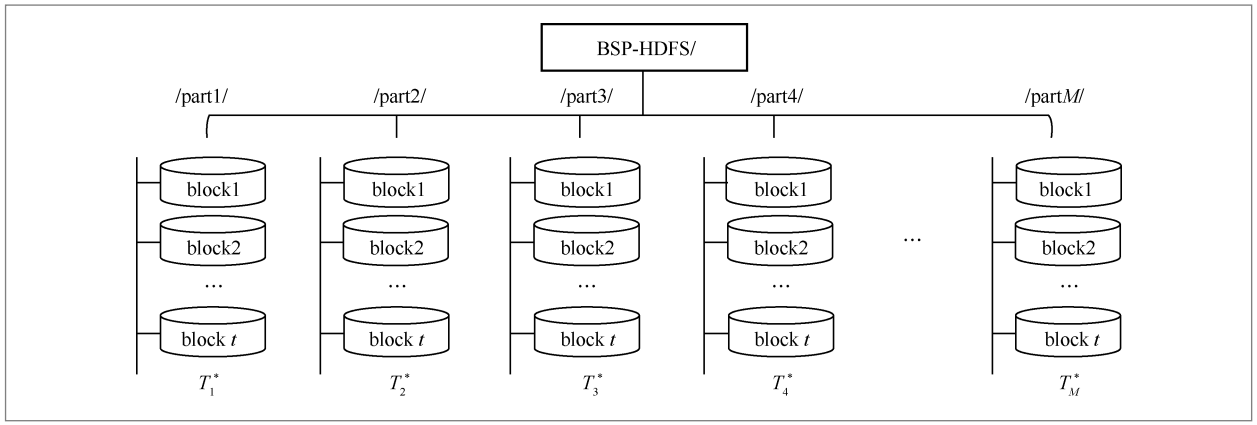

图1 BSP数据模型的分布式存储结构

语言实现的Bootstrap有放回抽样函数，每个虚拟机执行bootstrapFunc函数时采用独立的随机数种子（seed），生成独立的Bootstrap样本集。

BSP数据模型生成算法如**图2**所示，输入是原始数据集T文件，输出是BSP数据模型的BSP-HDFS文件。操作流程从左到右完成，每个方框表示RDD数据结构，方框上方列出对应的Spark操作，方框下方对应RDD数据结构名。一系列的RDD数据转换通过Spark算子完成，遵循标准的分布式进程[18-19]。

将BSP数据模型生成算法封装成Spark内部的转换算子toBSP的伪代码如算法1所示。

算法1 toBSP转换算子伪代码

输入：$\mathbb{T}$rdd：初始的HadoopRDD，M：需要生成的BSP数据块的块数.

输出：$\mathbb{D}$rdd：生成的BspRDD.

```
1：val RDD1:CoalescedRDD = Trdd.coalesce(1)// 将HadoopRDD的所有分区合并生成只有1个分区的CoalescedRDD
2：val RDD2:UnionRDD = RDD1.union(Trdd，M) //对CoalescedRDD并行复制成M个RDD，再将M个RDD链接合并成1个UnionRDD
3：val Drdd ： BspRDD= RDD2.mapPartitions (bootstrapFunc) // 对于RDD2的每个分区应用Bootstrap抽样算法生成Bootstrap样本
4：return Drdd
5：end
```

用toBSP算子生成数据集T的BSP数据模型通过一个简单的Spark应用程序完成，应用程序举例如**图3**所示。

2 基于BSP数据模型的分布式集成学习方法

2.1 基于BSP数据模型的分布式集成学习框架

在上一步中获得数据集T的BSP数

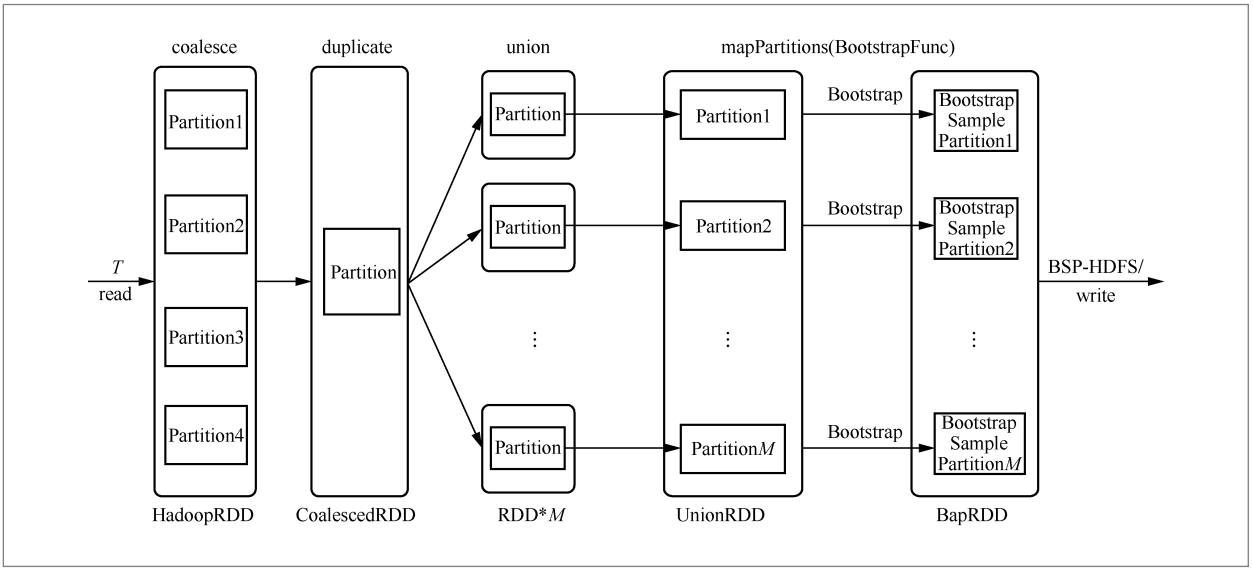

图 2 BSP 数据模型生成算法

```
import org.apache.spark.sql.BspContext._
import org.apache.spark.bsp.BspRDD

val spark = SparkSession.builder().master("yarn").getOrCreate() //初始化Spark上下文环境
val rdd: RDD[Row] = spark.sparkContext.textFile(T_Path)         //读取指定路径原始数据集T
val bspRDD: BspRDD[Row] = rdd.toBSP(M)                          //进行BSP数据块的生成，数量为M
bspRDD.saveAsTextFile(BSP-HDFS_Path)                            //进行HDFS存储为BSP-HDFS/
```

图 3 算子 toBSP 生成 BSP 数据模型的 Spark 应用程序示例

据模型后，对T做多种Bootstrap统计分析和集成学习建模时，不再需要生成新的Bootstrap样本集。基于BSP数据模型的分布式集成学习框架如**图4**所示，底层是分布式存储的BSP数据模型，包含M个BSP数据块，作为预先生成的Bootstrap样本集。对于T的一个Bootstrap统计分析或集成学习建模任务用下面4个步骤完成：

- 从BSP数据模型中随机选取m（$m<<M$）个BSP数据块，读入m个分区的BspRDD；
- 每个分区的数据用统计分析或建模算法在节点虚拟机内完成计算结果，m个分区独立并行计算完成；
- m个分区的计算结果上传到集群主节点；
- 在主节点用Spark应用程序完成m个上传子结果的集成计算。

与Map-Reduce分布式计算框架[20]不

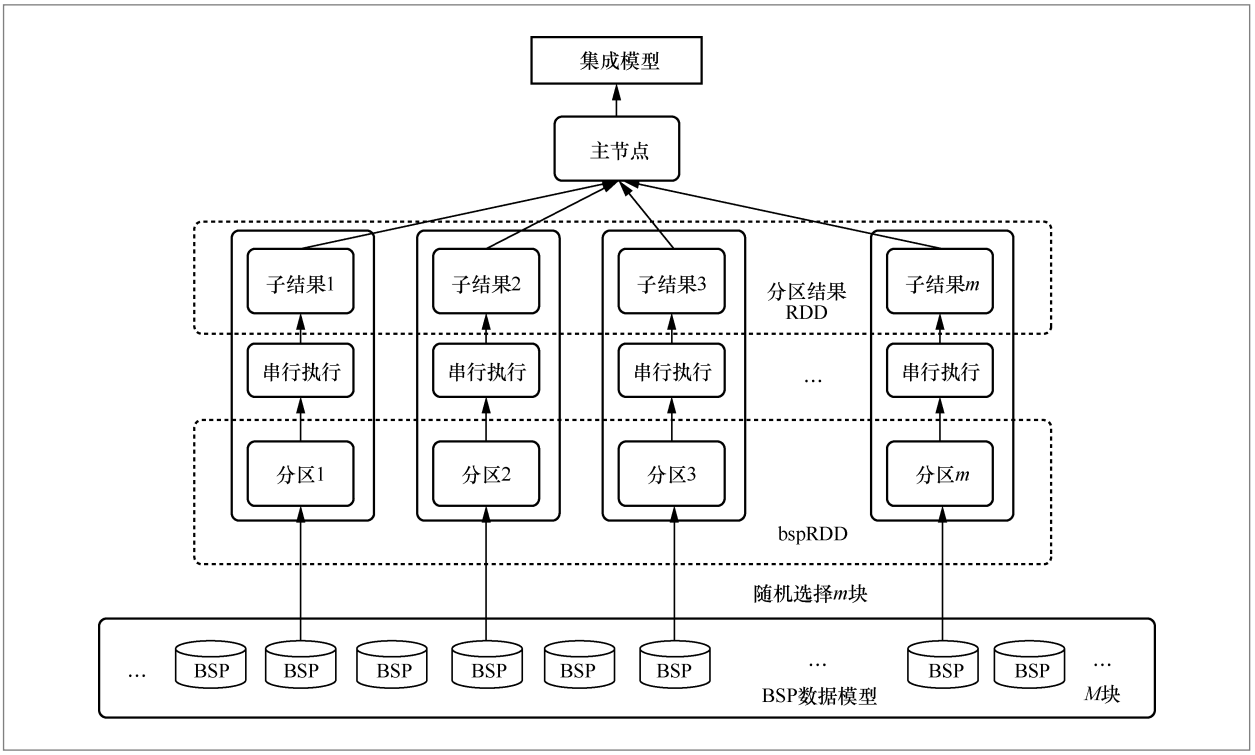

图 4 基于 BSP 数据模型的分布式集成学习框架

同，本框架第二步采用同一串行算法在m个分区独立并行计算，不需要节点间的数据通信，称作非Map-Reduce计算范式[21]。其有两个显著的优点，一是复杂迭代算法在m个分区上并行计算效率高；二是串行算法不需要按Map-Reduce的范式重写，很多难以并行的串行算法可以在此框架下并行计算。

2.2 质量选择策略

基于统计学原理，每个Bootstrap样本是随机生成的，从样本计算的统计量或建立的模型可以定义为随机变量[22]。当样本足够大时，该统计量呈正态分布。根据这一原理，当m个BSP数据块结果传送到主节点后，我们可以估计统计量的分布参数，计算其置信区间；根据置信区间，选择落入置信区间内的BSP数据块结果做集成计算，提高集成结果的泛化性。

设$S=\{D_1,D_2,\cdots,D_m\}$为从BSP数据模型随机抽取的m个BSP数据块，$Z=\theta(T)$是待估计的统计量，$Z_i=\theta(D_i),i\in\{1,2,\cdots,m\}$是$Z$的$m$个估计值，具有共同的数学期望$\mu$和方差$\sigma^2$。当$m$很大时，$Z$服从正态分布。$Z$的置信区间可以用式(1)计算。

$$\bar{Z}\pm z_{\alpha}\frac{s}{\sqrt{m}} \qquad (1)$$

其中，$\bar{Z}$是m个Z_i的均值，s是相应的标准差，z_{α}在给定显著性水平α后，通过标准正态$N(0,1)$得到。

得到置信区间后，只选择落入置信区间的BSP数据块结果来计算最后的集成结果，称作截尾的统计量结果（trimmedstatistics）。根据上述策略，基于Spark构造了样本优选方法selectPartitions。

2.3 基于BSP数据模型的分布式集成学习方法

基于BSP数据模型的分布式集成学习分为两个阶段：给定训练数据T，生成T的BSP数据模型，存储在分布式文件中；对T做Bootstrap统计分析或建立集成学习模型。对于同一训练数据T，第一阶段只需执行一次，BSP数据模型可以在第二阶段不同的分析任务中重复使用，具体分析任务通过编写和执行不同的Spark应用程序完成。

在Spark系统实现中，对于单个BSP数据块的串行分析算法，结合mapPartitions算子封装成新的Spark算子，供应用程序调用。在应用程序执行过程中，用于单个样本集分析和建模的串行算法通过mapPartitions算子分发到各节点或虚拟机中，对其相应BSP数据块的分区数据进行独立计算，节点间不需要数据通信。当各节点的计算完成后，其计算结果传送到主节点做集成，集成操作由应用程序完成。

以建立分类集成模型为例，假设训练数据T的BSP数据模型存储在HDFS数据文件中，主要操作步骤如下：

- 调用随机选择读取方法bspRead读m个BSP数据块到各节点内存；
- 用mapPartitions算子调用串行分类算法，分发到各虚拟机，对各个BSP数据块独立建立分类子模型；
- 将各子模型传送到主节点，调用selectPartitions方法，以模型精度作为统计量，构造置信区间，筛选出落入置信区间的子模型建立集成模型；
- 将测试数据集读入主节点内存，采用投票方法对测试数据分类，计算集成模型精度；
- 将集成模型和测试结果写入磁盘。

上述步骤的Spark应用程序伪代码如算法2所示。

算法2：BSP分布式集成学习伪代码

输入：trainFilePath：用于训练的BSP数据模型路径，testFilePath：用于测试的数据路径，

$\mathbb{L}$：某串行版本分类算法，m：随机选取的BSP数据块的个数；

输出：bspPredictions：预测结果，evaluations：评价指标结果；

```
1：valTrainRDD = spark.bspRead (trainFilePath,m).rdd // 随机读取m个BSP数据块进内存
2：valtestRDD = spark.read.parquet (testFilePath).rdd // 读取测试集
3：val models = TrainRDD.map Partitions (sample =>𝕃.fit(sample)) // 训练
4：valselectedModels = select Partitions(models) // 对m个BSP数据块训练结果进行筛选
5：val predicts = selectedModels. predict (testRDD) // 测试
6：valbspPredictions = predicts. map(vote) // 对结果进行集成
7：evaluations = evaluate (bsp Predictions.collect()) // 结果进行评估得到集成精度
8：returnbspPredictions , evaluations
9：end
```

基于BSP数据模型的分布式集成学习方法的特点如下：①各子模型的训练互不影响，可以减少大量节点间的网络通信传输开销；②可多次重复进行不同的统计分析和建模任务，节省了Bootstrap样本集生成的时间开销；③可以加入BSP数据块的样本优选策略，进一步提高分析的精度。

3 实验结果分析

实验结果从两个方面验证和展示基于BSP数据模型的分布式集成学习的优势：一是计算效率，二是有监督集成学习模型的精度。与已有串行方式实现的集成学习算法相比，本文提出的方法在计算效率和数据以及模型扩展性方面有明显的提升。与单个模型相比，集成模型的精度也有进一步提高。由于可以高效地计算上千个子模型，其集成模型的精度也高于常用的少量子模型的集成模型。同时，BSP数据块的优选策略进一步提高了集成模型的精度。

3.1 实验环境与数据集

实验在 Spark集群上进行。集群配置有32台服务器节点，其中24台服务器配置两个16核、2.6 GHz的Intel Xeon E5-2650 CPU，8台服务器配置两个12核、2.6 GHz的Intel Xeon E5-2630 CPU，所有服务器的内存为128 GB，每个服务器配30 TB 的硬盘。集群的软件设置见**表1**，Spark的运行参数配置见**表2**。

实验数据采用UCI的两个数据集，SUSY数据集有500万个对象记录，每个对象有18个特征和1个标签值，对象分成两类，用于建立分类模型；Wine-quality数据集包含Wine-quality-red数据子集和Wine-quality-white数据子集，它们分别有1 599个、4 898个对象记录，两个数据子集都有11个特征和1个输出值，用于建立回归模型。

3.2 衡量标准

计算效率以程序运行的时间长短衡

量，以秒为单位。

分类模型的性能用准确率（Accuracy）和F1值（F1-Score）衡量。准确率定义为：

$$\mathrm{ACC}=\frac{n}{m} \tag{2}$$

其中，m是测试数据的样本对象数，n是模型正确分类的对象数[23]。

F1值计算式如下：

$$\mathrm{F1}=\frac{2\times\mathrm{Recall}\times\mathrm{Precision}}{\mathrm{Precision}+\mathrm{Recall}} \tag{3}$$

其中，Precision是精确率，Recall是召回率。

回归模型用均方根误差RMSE和决定系数R^2衡量。均方根误差计算式为：

$$\mathrm{RMSE}=\sqrt{\frac{\sum_{i=1}^{m}(y_i-\hat{y}_i)^2}{m}} \tag{4}$$

其中，$\hat{y}_i$是对象i的模型预测值，y_i是对象i的实际观测值。

决定系数R^2的计算式如下：

$$R^2=1-\frac{\sum_{i=1}^{m}(y_i-\hat{y}_i)^2}{\sum_{i=1}^{m}(y_i-\bar{y})^2} \tag{5}$$

其中，$\bar{y}$是所有预测值的均值，R^2是回归模型常用的度量拟合效果的参数，取值范围为0到1，R^2越大，模型的拟合效果越好。

3.3 实验及结果分析

实验1：BSP数据模型生成算法的计算效率。本实验通过比较4种算法从不同大小的训练数据集T生成不同数量BSP数据块的运行时间，展示本文提出的BSP数据模型生成算法的优势。4种算法包括：本文提出的BSP数据模型生成算法toBSP、基于Spark广播分发的BSP数据块生成算法、单机串行方式的Bootstrap样本生成算法和单机多核并行方式的Bootstrap样本生成算法。训练数据T采用了4种文件大小，分别是50 MB、250 MB、500 MB和1 GB。图5和图6分别展示了4种方法生成500个和1 000个BSP数据块的运算时间，以秒为单位。时间柱顶部的数字表示运行时间，OT表示该算法在该数据集上运行时间超过1 800 s，无法在30 min内生成给定数量的BSP数据块，程序终止。时间柱所表示算法的顺序从左到右依次是toBSP、Spark广播、单机多核并行、单机串行。

从图5和图6的结果看，只有toBSP算法可以在30 min内生成指定数量的BSP数据块。尽管训练数据集T的增长不是线性的，但toBSP的运算时间增长呈线性的趋势，在25 min内生成了1 000个1 GB训练数据集的Bootstrap样本集。Spark广播方法对小数据集有效，对稍大的数据集

表1 实验环境各软件版本设置表

软件	参数
NodeOS	OperatingSystem = CentOS Version = 7.5.1804
Spark	SparkVersion = 2.4.0 HadoopVersion = 3.0 JavaVersion = 1.8.181 ScalaVersion = 2.11.12 HDFSReplicationFactor = 3 HDFSBlockSize = 128 MB

表2 Spark实验运行参数配置信息表

参数	含义	设定值
master	Spark运行模式	yarn
deploy-mode	driver部署模式	client
driver-memory	driver端内存大小	16 GB
num-executor	Spark运行时executor数量	100
executor-core	每个executor占的CPU数	4
executor-memory	每个executor占的内存大小	16 GB

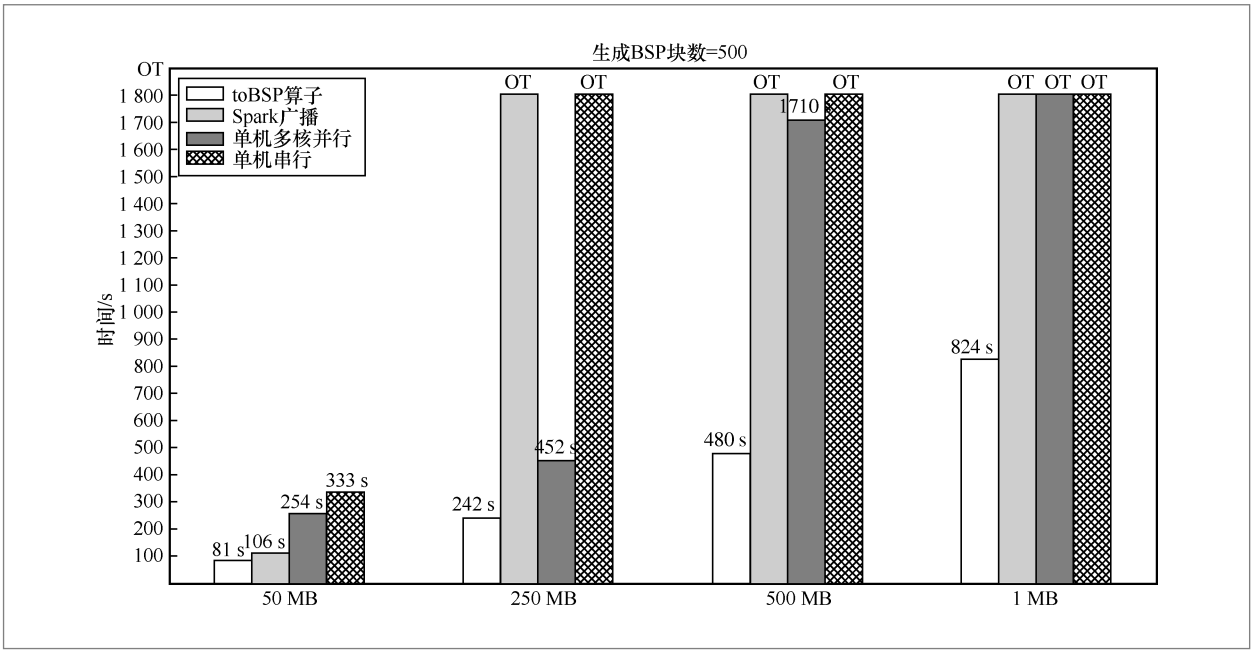

图 5 各方式生成 500 个 BSP 块实验效率对比

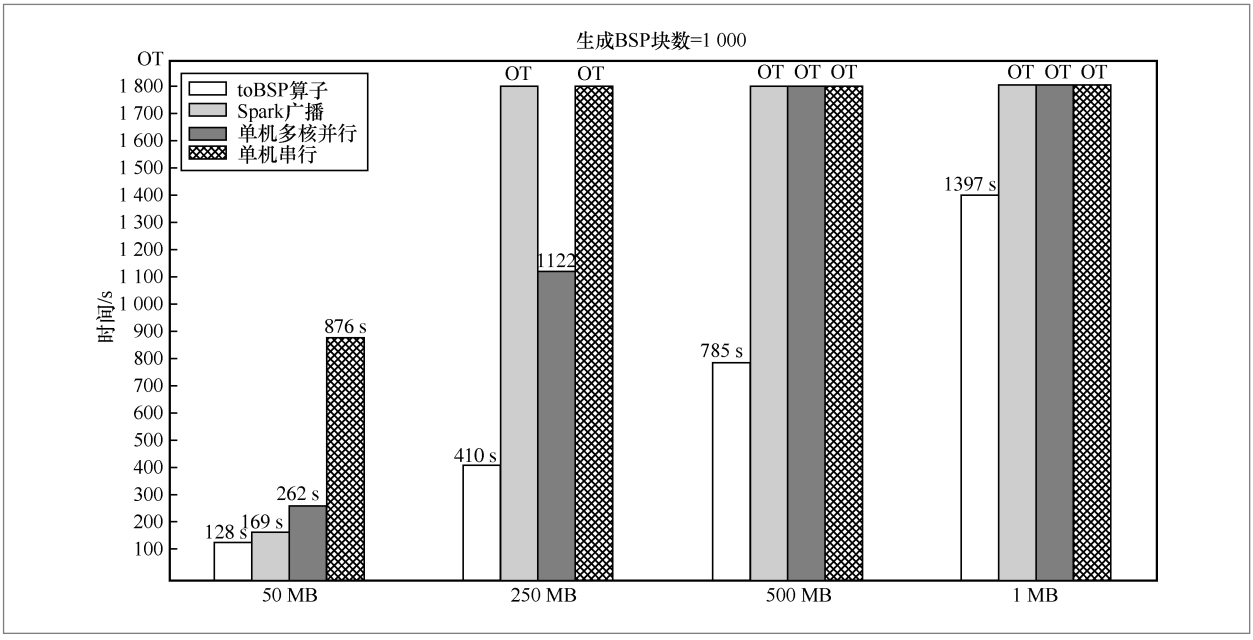

图 6 各方式生成 1 000 个 BSP 块实验效率对比

（如250 MB）非常耗时，无法在指定的时间内生成给定数量的BSP数据块。单机串行算法只适合小的训练数据，受内存限制，运算时间随着BSP数据块数量的增加而增加。单机多核并行优于单机串行，但运行效率仍受内存和CPU限制，不适合大的训练数据集和大量BSP数据块的生成。由此可以看出，本文提出的toBSP算法可以高效地生成训练数据集的大量BSP数据块，为Bootstrap统计分析和集成学习提供了生成

好的Bootstrap样本集，从而提高后续的数据分析效率和灵活性。例如，在**图6**的展示中，13 min内即可生成1 000个500 MB的Bootstrap样本集，其他算法则无法做到。

实验2：基于BSP数据模型的分布式集成学习的计算效率。本实验比较串行方式集成学习算法建立集成模型的运算时间与本文提出的分布式集成学习算法建立集成模型的运算时间，展示本文提出的方法在集成模型建模上的效率和模型扩展性能。本实验采用的原始训练数据为250 MB，是SUSY数据集的一个子集。建模算法为决策树算法。**图7**展示两种集成算法建立不同数量子模型的集成模型的运算时间，子模型数量由1增长到1 000。串行算法每次需要重新Bootstrap抽样后再进行训练。

从**图7**的时间增长曲线可以看出，串行方式集成学习算法随着子模型数量的增加，运算时间快速增加。而本文提出的分布式集成学习算法每增加100个子模型，运算时间只有微小的增加，时间维持在相对恒定水平；当子模型数据超过500个时，运算时间的增加才略微加大，这是由于本实验所使用集群中的物理核心用满后，即使具备超线程技术，也无法通过增加工作核心数来加速分布式训练，但运算时间增速不大，直到1 000个子模型全部生成。这样的效果是由本方法对每个子模型独立计算，大量的子模型并行计算，且几乎没有节点间的通信开销取得的。100个模型的建模时间与1个模型的建模时间相差不大，运算时间的增长是由子模型向主节点的传送和主节点的集成计算产生的。另外，本实验使用的集群共有480个核，当所有核用满后，Spark对虚拟机的计算做任务调度，增加了子模型批量训练的时间，分布式训练的提速达到瓶颈，训练时间的增速也因此加快了。

对于串行方式集成学习算法，由于运行在单个节点上，计算200个子模型的集成模型需要运行5个多小时，效率极低。再增加子模型数量，就达到了内存的极限，从**图7**中的垂直虚线可以看出串行算法的时间开销增长速度在明显加快，因此在图中截断显示，在实验中即使是使用单机多线程，在CPU资源有限的情况下仍无法使训练时间增长趋势放缓。本文算法相比于串行方式在时间效率上有了巨额提升。因为其只需要取对应划分好的BSP数据块进行训练，无须进行再次划分生成，同时BSP分布式集成学习方法将划分好的BSP数据块读取并行训练，训练时间并不随着模型增加而受到影响。下面分析两种训练方式的时间复杂度，设单个数据块的运算时间为TA，使用的核心总数（即并行度）为P，它们都是相当于一个定值，使用的块数为m，因此BSP集成方法的时间复杂度为$O((m/P)\times TA)$，而串行训练方式的时间复杂度显然为$O(m\times TA)$。从中可以看出，本文提出的分布式集成学习方法和算法可以极大地提高集成学习和Bootstrap统计分析的计算效率，同时也可以高效地建立大规模的集成学习模型，提高了集成学习的模型扩展能力和性能。

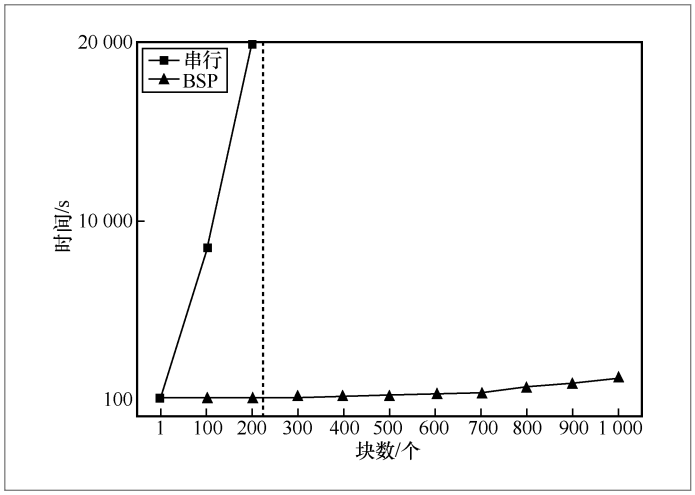

图 7 BSP 分布式集成学习与串行训练时间效率对比

实验3：基于BSP数据模型的分布式

集成学习模型精度。本实验用250 MB的SUSY数据集子集，对比使用3种不同的实验方法，即非集成方法、BSP集成方法、加入了质量选择策略的BSP集成方法来训练4种有监督机器学习算法，包括：决策树、随机森林、逻辑回归和支持向量机，建立了4种不同规模的集成模型，分别是500、1 000、1 500和2 000个子模型，用独立的测试数据对每个算法的单个模型和集成模型进行测试，计算其分类精度ACC和F1值，结果如**图8**、**表3**和**表4**所示。同时，采用线性回归算法训练单个线性回归模型和4种规模的集成模型。用衡量标准RMSE和R^2计算模型的性能，结果见**表3**和**表4**。

从**图8**可以看出，除随机森林外，其他算法的集成模型精度都高于单个模型的精度（图中水平直线所示）。除支持向量机的集成模型外，其他算法的集成模型精度随着子模型数量的增加而增加，最后稳定在某个子模型数量上。例如：决策树稳定在1 000个子模型，其提升最大，提高了约2%，在该数据集上的准确率能达到80%以上，在达到1 000个BSP数据块前，其增速较为明显，在达到1 000个BSP数据块时，增速开始放缓，准确率也逐渐趋于稳定。逻辑回归稳定在1 500个子模型，而支持向量机在500个子模型时就已经稳定，后面随块数的增加准确率变化不大。随机森林

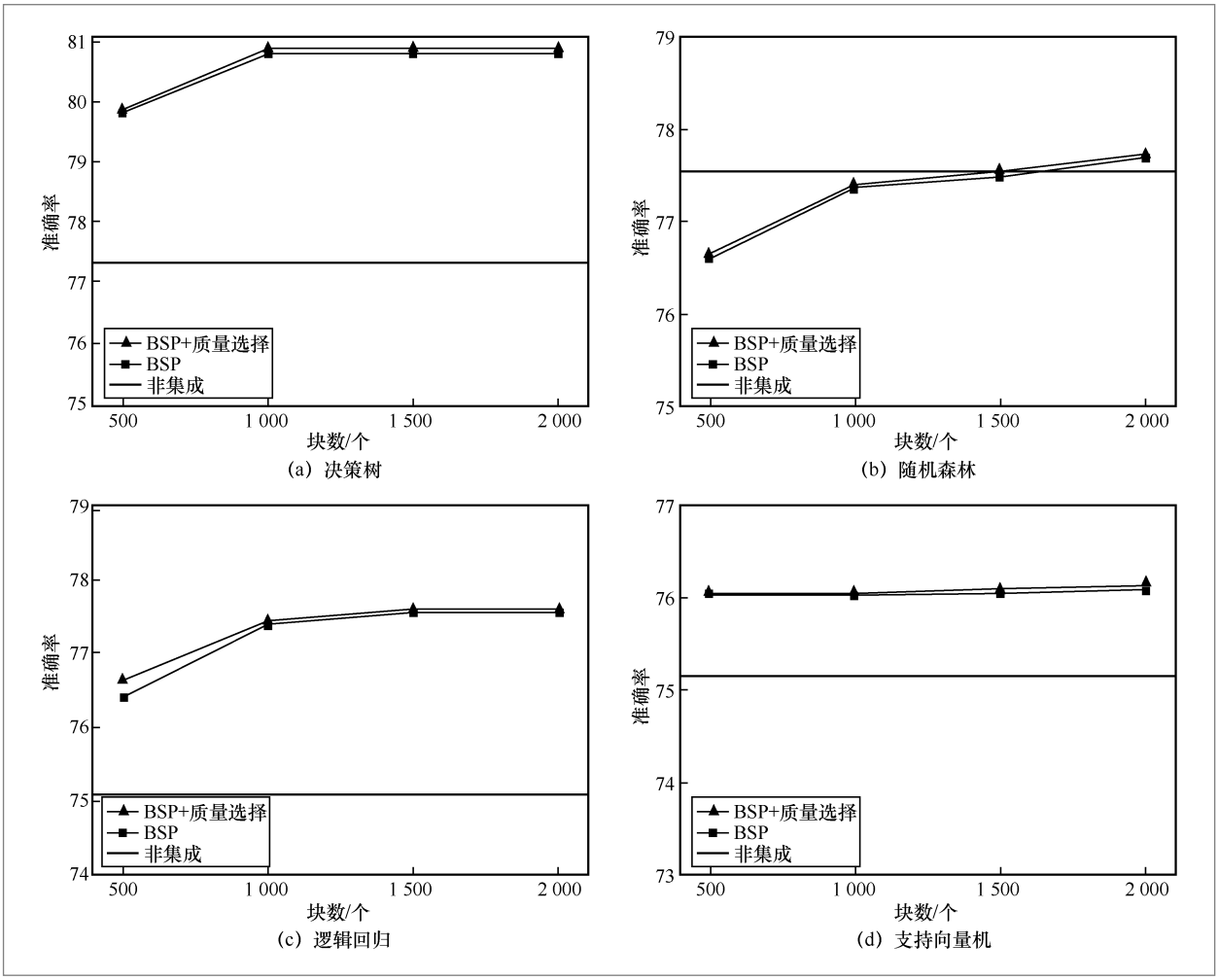

图 8 BSP 分布式集成学习与串行训练预测精度趋势

表 3 BSP 分布式集成学习分类预测效果表

有监督分类算法	方法	块数	ACC	F1
决策树	非集成方法	—	77.3744%	74.2809%
	BSP	500	79.8355%	75.0224%
		1 000	80.7834%	75.4517%
		1 500	80.7766%	75.5691%
		2 000	80.7987%	75.5712%
	BSP+质量选择	500	79.8921%	75.4444%
		1 000	80.8879%	76.4505%
		1 500	80.8886%	76.4509%
		2 000	80.8890%	76.4511%
随机森林	非集成方法	—	77.5589%	72.5875%
	BSP	500	76.5913%	71.7854%
		1 000	77.3842%	72.8892%
		1 500	77.4987%	72.9157%
		2 000	77.7025%	72.9765%
	BSP+质量选择	500	76.6595%	72.5094%
		1 000	77.4099%	72.8927%
		1 500	77.5590%	73.0127%
		2 000	77.7273%	73.1287%
逻辑回归	非集成方法	—	75.0791%	73.9617%
	BSP	500	76.3870%	74.0017%
		1 000	77.3881%	74.6258%
		1 500	77.5469%	74.6954%
		2 000	77.5689%	74.7826%
	BSP+质量选择	500	76.6107%	74.6667%
		1 000	77.4291%	74.8243%
		1 500	77.5831%	74.9023%
		2 000	77.5970%	75.1290%
支持向量机	非集成方法	—	75.1528%	71.5133%
	BSP	500	76.0346%	72.0137%
		1 000	76.0231%	72.8382%
		1 500	76.0312%	72.9002%
		2 000	76.1145%	73.2978%
	BSP+质量选择	500	76.0471%	72.2447%
		1 000	76.0431%	72.9718%
		1 500	76.0882%	72.9900%
		2 000	76.1252%	72.9909%

的结果有所区别，因为随机森林模型本身是一种集成模型，算法内部进行了集成，使用BSP算法在块数较多时对随机森林的集成效果才趋于明显。它的单个模型的精度已经很高，与其他方法的决策树和逻辑回归的集成模型的精度接近，高于支持向量机的集成模型。但是可以看出，当集成模型的规模加大时，随机森林的集成模型精度还是超过了单个模型的精度。同时，从决策树的集成模型精度也可以得出，当集

表 4 BSP 分布式集成学习回归预测效果表

有监督回归算法	方法	块数	RMSE	R^2
线性回归	非集成方法		0.661331	0.303635
	BSP	500	0.653482	0.320068
		1 000	0.652633	0.321833
		1 500	0.652594	0.321880
		2 000	0.652477	0.322156
	BSP+质量选择	500	0.652993	0.321085
		1 000	0.652508	0.322092
		1 500	0.652415	0.322208
		2 000	0.652369	0.322382

成模型的规模加大时，其精度也高于随机森林单个模型的精度。本文提出的分布式集成学习方法恰好可以提高集成学习模型的规模。

表3和**表4**给出了不同算法和学习方法的模型预测效果，可知：对于同一算法，BSP集成模型好于单个模型；规模大的BSP集成模型好于规模小的BSP集成模型；采用BSP数据块优选好于没有优选，只有支持向量机的集成模型除外，其原因有待进一步分析。同样，对于线性回归模型，BSP集成模型优于单个模型；规模大的BSP集成模型好于规模小的BSP集成模型；BSP数据块的质量选择同样可以提高集成模型的性能。这些结果与分类集成模型一致。该实验说明使用BSP集成学习比非集成学习的效果要好，同时在BSP集成学习中加入质量选择策略后，参与训练的BSP数据块的质量较好，模型的鲁棒性进一步提升，模型的预测结果也会有进一步提升。使用BSP分布式集成学习方法不仅使训练时间缩短，预测效果也得到了进一步提高。

4 结束语

本文提出的BSP数据模型和基于BSP数据模型的分布式集成学习方法采用了大数据分布式计算的策略，使BSP数据模型的生成和集成学习建模的分离，高效生成BSP数据模型，可以在不同分析和建模任务中的重用，增强了数据可重用性，支持高效的大规模集成学习建模和Bootstrap统计分析。BSP数据模型的生成和分布式集成学习采用非Map-Reduce计算范式进行计算，每个数据块的计算独立完成，减少了计算节点间的数据通信开销，整个建模过程只有一次聚合集成操作，且可加入质量选择策略。实验结果表明，本文方法在计算效率、数据规模、预测效果、应用的灵活性等方面比现有的方法有很大提升。但此方法的缺点是BSP数据模型占据很大的存储空间，在分析工作完成后，可以将BSP数据模型的分布式文件删除，只保留原训练数据，释放集群的存储空间。未来的研究工作将重点开发Spark的Bootstrap统计分析算子，增强Spark系统的统计分析功能。

参考文献:

[1] BREIMAN L. Bagging predictors[J]. Machine Learning, 1996, 24(2): 123-140.

[2] EFRON B. Bootstrap methods: another look at the jackknife[J]. The Annals of

Statistics, 1979, 7(1):1–26.

[3] DAVISON A C, HINKLEY D V. Bootstrap methods and their application[M]. Cambridge: Cambridge University Press, 1997.

[4] ROSENBLATT M. A central limit theorem and a strong mixing condition[J]. Proceedings of the National Academy of Sciences of the United States of America, 1956, 42(1): 43–47.

[5] SAGI O, ROKACH L. Ensemble learning: a survey[J]. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 2018, 8(4): e1249.

[6] KREISS J P, PAPARODITIS E. Bootstrap methods for dependent data: a review[J]. Journal of the Korean Statistical Society, 2011, 40(4): 357–378.

[7] CHEN J G, LI K L, TANG Z, et al. A parallel random forest algorithm for big data in a spark cloud computing environment[J]. IEEE Transactions on Parallel and Distributed Systems, 2017, 28(4): 919–933.

[8] XU W, HOANG V T. MapReduce-based improved random forest model for massive educational data processing and classification[J]. Mobile Networks and Applications, 2021, 26(1): 191–199.

[9] SENAGI K, JOUANDEAU N. Parallel construction of random forest on GPU[J]. The Journal of Supercomputing, 2022, 78(8): 10480–10500.

[10] YU Y X, PENG S C, YUAN Y, et al. A classifier using online bagging ensemble method for big data stream learning[J]. Tsinghua Science and Technology, 2019, 24(4): 379–388.

[11] KLEINER A, TALWALKAR A, SARKAR P, et al. A scalable bootstrap for massive data[J]. Journal of the Royal Statistical Society Series B: Statistical Methodology, 2014, 76(4): 795–816.

[12] BASIRI S, OLLILA E, KOIVUNEN V. Robust, scalable, and fast bootstrap method for analyzing large scale data[J]. IEEE Transactions on Signal Processing, 2016, 64(4): 1007–1017.

[13] 黄哲学, 何玉林, 魏丞昊, 等. 大数据随机样本划分模型及相关分析计算技术[J]. 数据采集与处理, 2019, 34(3): 373–385.
HUANG Z X, HE Y L, WEI C H, et al. Random sample partition data model and related technologies for big data analysis[J]. Journal of Data Acquisition & Processing, 2019, 34(3): 373–385.

[14] SALLOUM S, HUANG J Z, HE Y L. Random sample partition: a distributed data model for big data analysis[J]. IEEE Transactions on Industrial Informatics, 2019, 15(11): 5846–5854.

[15] MAHMUD M S, HUANG J Z, SALLOUM S, et al. A survey of data partitioning and sampling methods to support big data analysis[J]. Big Data Mining and Analytics, 2020, 3(2): 85–101.

[16] ZAHARIA M, CHOWDHURY M, FRANKLIN M J, et al. Spark: Cluster computing with working sets[J]. HotCloud, 2010, 10(10): 95.

[17] SHVACHKO K, KUANG H R, RADIA S, et al. The hadoop distributed file system[C]//Proceedings of 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST). Piscataway: IEEE Press, 2010: 1–10.

[18] ZAHARIA M, CHOWDHURY M, DAS T, et al. Resilient distributed datasets: a fault-tolerant abstraction for in-memory cluster computing[C]//Proceedings of the 9thUSENIXSymposium on Networked Systems Design and Implementation. San Jose: USENIX Association, 2012: 15–28.

[19] GUR, QI Y, WU T Y, et al. SparkDQ: efficient generic big data quality management on distributed data-parallel computation[J]. Journal of Parallel and Distributed Computing, 2021, 156: 132–147.

[20] DEAN J, GHEMAWAT S. MapReduce[J]. Communications of the ACM, 2008, 51(1): 107–113.

[21] SUN X D, HE Y L, WU D M, et al. Survey of distributed computing frameworks for supporting big data analysis[J]. Big Data Mining and Analytics, 2023, 6(2): 154–169.
[22] EFRON B, TIBSHIRANI R J. An introduction to the bootstrap[M]. Boca Raton: CRC press, 1994.
[23] KADKHODAEI H, EFTEKHARI MOGHADAM A M, DEHGHAN M. Big data classification using heterogeneous ensemble classifiers in Apache Spark based on MapReduce paradigm[J]. Expert Systems with Applications, 2021, 183: 115369.

作者简介

罗凯靖（1997– ），男，深圳大学计算机与软件学院硕士生，主要研究方向为大数据并行与分布式计算技术、大数据系统计算技术。

张育铭（1998– ），男，深圳大学计算机与软件学院硕士生，主要研究方向为数据挖掘、大数据系统计算技术。

何玉林（1982– ），男，博士，人工智能与数字经济广东省实验室（深圳）副研究员，主要研究方向为大数据智能计算、多样本统计分析、数据挖掘与机器学习算法及应用等。

黄哲学（1959– ），男，博士，深圳大学特聘教授，大数据技术与应用研究所所长，大数据系统计算技术国家工程实验室副主任。1993年获瑞典皇家理工学院博士学位。主要研究方向为数据挖掘、分布式机器学习系统、大数据并行处理与分析、大数据系统计算技术等。

收稿日期： 2023-06-25
通信作者： 黄哲学，zx.huang@szu.edu.cn
基金项目： 国家自然科学基金项目（No.61972261）；广东省自然科学基金面上项目（No.2023A1515011667）；深圳市基础研究重点项目（No.JCYJ20220818100205012）；深圳市基础研究面上项目（No.JCYJ20210324093609026）
Foundation Items: The National Natural Science Foundation of China (No.61972261), Natural Science Foundation of Guangdong Province (No.2023A1515011667), Key Basic Research Foundation of Shenzhen (No.JCYJ20220818100205012), Basic Research Foundations of Shenzhen (No.JCYJ20210324093609026)