【大数据与区块链专栏】

大规模数据的随机森林算法

李 扬[1a,1b,1c]，祁 乐[1b,1c]，聂佩芸[2]

（1. 中国人民大学 a. 应用统计科学研究中心，b. 统计学院，c. 统计咨询研究中心，北京 100872；
2. 腾讯公司 国际业务部，广东 深圳 518057）

摘要：信息技术的高速发展提升了人们生产、收集数据的能力，越来越多的数据呈现出海量化、高维化的特征。这类大规模数据的出现给统计分析带来计算效率方面的挑战。为有效解决计算效率较低的问题，研究者结合“分治”思想提出了一种分析框架，并以随机森林算法为例内嵌其中得到大规模随机森林算法（BLOCK-SDB-RF）。研究者从数据覆盖率及时间复杂度两方面对该算法的优势进行分析，同时通过数值模拟探究了BLOCK-SDB-RF算法的应用效果。数值模拟结果显示：1. 随着数据样本量、特征维度的增加，该算法在计算效率上的优势愈发明显；2. 尽管变量间的相关性对该算法的计算效率影响并不明显，但随着相关性的增加，研究者需要牺牲一部分预测精度。在实证分析中，研究者以音乐流媒体服务商KKBOX提供的日志数据为例，进一步讨论了BLOCK-SDB-RF算法在大规模高维实际数据分析中的作用。

关键词：大数据；计算效率；随机森林；分布式计算

中图分类号：O212 文献标志码：A 文章编号：1007－3116(2020)06－0024－10

一、引 言

随着信息技术的高速发展，人们生产、收集数据的能力大大提升，越来越多的数据呈现出样本海量化、高维化的趋势。譬如，阿里巴巴集团旗下的淘宝平台仅日交易数据就多达数10TB，Facebook公司每月产生的日志数据业已超过10PB[1]，又如Kaggle平台举办的数据挖掘竞赛中向参赛者提供的许多规模庞大的数据集等。大规模数据的广泛应用使传统的统计分析面临计算效率的挑战[2]。以2017年Kaggle举办的建模竞赛中音乐流媒体服务商KKBOX向参赛者提供的日志数据为例，该竞赛要求参赛者通过对日志数据合理建模，以准确预测用户在订阅到期后是否会流失。一方面，每个用户在音乐软件中通过实时点击会产生多种多样的信息，如点击的音乐入口、读取的音乐内容、在软件中停留的时间等，一个小时之内产生的日志记录可达成百上千条。特别地，音乐软件的受众较广，用户往往数以亿计，因此日志数据中包含的样本数量非常庞大，海量化的数据不仅对计算机存储空间等硬件设备提出了新挑战，也将带来较高的计算成本，进一步制约大数据处理技术的时效性；另一方面，从用户、产品以及两者的交互角度出发，日志数据中会存在大量与用户偏好相关的特征变量。同时，为更加准确地了解用户偏好，对于隐藏在用户背后无法被探知的信息，如天气，心情、环境等因素，需要人为探索、构造一些特征来刻画，最终数据的特征维数会大大增加。然而在高维情况下，一些经典的分类方法如Fisher判别分析等常常会失效，且计算效率同样面临严峻的挑战[3]。因此，针对这种样本量大、特征维数高的大规模数据，如何实现高效分析是研究者亟待解决的问题。

收稿日期：2019－11－08；修复日期：2020－02－16

基金项目：中国人民大学科学研究基金（中央高校基本科研业务费专项资金）研究品牌项目“生物医学大数据的统计方法基础研究”（15XNI011）

作者简介：李 扬，男，北京人，教授，研究方向：决策与预测，生物医学大数据；
祁 乐（通讯作者），女，河北张家口人，博士生，研究方向：调查设计与预测模型；
聂佩芸，女，江西南昌人，数据分析师，研究方向：推荐算法和用户画像。

目前，许多学者针对算法耗时长、计算效率低的问题提出了相应解决方案。针对样本量较大的情况，Bickel、Kleiner 等在自助法（Bootstrap 方法）基础上分别又提出了 m-n 自助法、Bootstrap of Little Bag 方法（BLB）[4-5]。这两种方法通过不同方式对数据进行抽样，降低了数据量级，从而在样本量过高时实现了计算效率的提升。但是上述方法会在一定程度上降低数据的变异性，且 BLB 方法在应用过程中需要同时优化两个参数，即样本子集的数量以及每个样本子集上重复抽样的次数，利用枚举法确定参数也会大大限制该方法在计算时间上的优势。因此，Sengupta 等人在此基础上将在每个样本子集上重复抽样的次数固定为 1，提出了 Subsample Double Bootstrap（SDB）方法，即在 BLB 方法的基础上以牺牲数据一些变异性为代价，进一步提高计算速度[6]。然而，不论是 BLB 方法还是 SDB 方法，它们的出发点均只考虑了数据量大的情境，对特征维度较高时如何实现高效分析并没有进行讨论，因此并不能完全解决大规模数据的计算挑战。同时，还有一类正则化方法比如通过添加惩罚项 LASSO、SCAD 等，或者在超高维的情况下事先利用边际回归筛选重要变量，在筛选变量的基础上添加惩罚项实现降维，以降低计算复杂度、提高计算的可行性。然而，这类方法也仅仅解决了特征维度方面的计算问题；特别地，当变量间具有较强的相关性时，不宜利用边际回归筛选变量，所以，在这种情况下利用该方法提升计算效率也不具合理性。除此之外，还有一种解决问题的常见思路是将复杂任务分解为多个子任务，通过并行计算节约运行时间[7]。比如钟龙申、Luo 等分别将并行化计算方式应用在随机森林算法以及带有正则化约束的支持向量机算法中以提高运算效率[8-9]；又如 Fang 和 Ma 针对大规模数据提出了一种自助加罚法（Bootstrap Penalization），该方法同时从特征维度、样本维度出发，通过将计算复杂的加罚任务拆分成多个低计算量的子任务，使其在多台计算机上并行计算，最终通过加权的方式将不同的子任务的结果合并进行决策，显著的减少了运算时间，提高了计算的可行性[10]。但是，这种方法主要针对加罚估计的问题进行讨论，且实现过程中涉及多个调节参数的选取，不同的调节参数可能会对估计结果产生不同程度的影响。随机森林是由 Breiman 在 Bagging 集成学习理论基础上提出的一种通过多棵分类回归树进行组合预测的统计算法。目前针对随机森林算法的优化研究大多还只集中在提升算法精度上，比如如何进行特征选择、参数优化以及处理样本不平衡等问题。吴琼等、曹正凤、汪桂金分别利用 NCL（Neighborhood Cleaning Rule）技术、聚类算法、SMOTE 算法等思想对随机森林算法进行改进以解决样本不平衡带来的算法性能下降的挑战[11-13]。Paul 等在随机森林算法中借鉴了强化学习的思想，提出了增强随机森林（Reinforced Random Forest）[14]，马景义等结合 Adaboost 算法和随机森林算法的优点提出了拟自适应随机森林，这两种新算法在提高分类的准确性上均取得了不错的效果[15]。Gall 等对随机森林的投票过程进行了优化改进[16]。特别地，针对面向高维数据的随机森林算法存在的不足，汪桂金还提出了基于智能算法的随机森林特征选择和参数优化，并通过搭建分布式计算平台 Hadoop 对并行化随机森林算法展开研究[13]。然而，这类并行运算的方法主要利用了外在计算平台的优势降低计算成本。

因此结合前人研究，为了在尽量保证算法有效性的前提下同时解决大规模数据样本量大、特征维度高带来的计算问题，本文利用“分治”思想提出了一种分析框架，并以随机森林算法为例内嵌其中提出大规模随机森林算法（BLOCK-SDB-RF），以探索机器学习算法在大规模高维实际数据分析中的作用。

二、方　法

（一）基础算法

随机森林是一种以决策树为基学习器的集成算法，通过样本维度的 Bootstrap 重抽样与特征维度的随机抽样制造出更多的随机性，以实现减少预测方差的目的。在决策树的建立过程中，通过测度输出变量的异质性指标找到决策树节点的最佳分组变量与最佳分割点。令 X 表示自变量，Y 表示因变量，对于分类问题，一般采用 Gini 指数或信息熵测度输出变量的异质性，见式（1）、式（2），其中 p_k 表示样本属于第 k 类的概率，$k=1,2,\cdots,|\mathcal{Y}|$，$|\mathcal{Y}|$ 表示数据集 D 中包含的类别数。

$$\mathrm{Gini}(D)=\sum_{k=1}^{|\mathcal{Y}|}p_k(1-p_k)=1-\sum_{k=1}^{|\mathcal{Y}|}p_k^2 \tag{1}$$

$$\mathrm{Ent}(D)=-\sum_{k=1}^{|\mathcal{Y}|}p_k\log_2 p_k \tag{2}$$

对于回归问题，方差常常被用来测量输出变量的异质性，见式（3）。

$$\mathrm{var}(D)=\frac{1}{n-1}\sum_{j=1}^{n}(y_j-\bar{y})^2 \tag{3}$$

异质性指标越高，意味着数据的杂乱程度越深。因此，为了使决策树更好的实现分类（回归），往往通过最大化输出变量的异质性指标减少量构建决策树。以分类问题中的 Gini 指数为例，需要优化的具体模型如式（4）。

$$\Delta\text{Gini}(t) = \text{Gini}(t) - \frac{|N_r|}{|N|}\text{Gini}(t_r) - \frac{|N_l|}{|N|}\text{Gini}(t_l) \tag{4}$$

其中，t 表示决策树节点，$\text{Gini}(t)$ 和 N 表示分组前输出变量的 Gini 指数和样本量，N_r、$\text{Gini}(t_r)$、N_l、$\text{Gini}(t_l)$ 分别代表分组后右子树、左子树的 Gini 指数及样本量。最佳分组变量与最佳分割点应是使 $\Delta\text{Gini}(t)$ 最大的变量与分割点。

在此基础上，假定有一组来自某个未知联合分布 F 的独立同分布随机样本 $\chi_n = (\mathbb{X}_1, \mathbb{X}_2, \cdots, \mathbb{X}_n)^{\mathrm{T}}$，其中 $\mathbb{X}_i = (\mathbb{X}_{i1}, \mathbb{X}_{i2}, \cdots, \mathbb{X}_{ip})^{\mathrm{T}}$，为便于区分，令 $\chi_p = (X_1, \cdots, X_p)$ 表示 p 维特征变量。对该组样本 χ_n 进行 R 次 Bootstrap 抽样得到 R 个样本集合 $\chi_n^k = (\chi_n, W_n^k)$，其中 $W_n^k = (w_1^k, w_2^k, \cdots, w_n^k)$，$k = 1, 2, \cdots, R$，表示 χ_n 中的 n 个样本在第 χ_n^k 中出现的频率。在每一个 χ_n^k 上，从特征 $\chi_p = (X_1, X_2, \cdots, X_p)$ 随机抽取一定比例（一般为 1/2）的变量 $\chi_p^k = (X_{t_1}, X_{t_2}, \cdots, X_{t_{p/2}})$，构建决策树 $T(\chi_n^k, \chi_p^k)$，其中 $t_1, t_2, \cdots, t_{p/2}$ 为集合 $Z = (1, 2, 3, \cdots, p)$ 的一个随机子集。最终合并多棵树进行决策（比如在回归问题中，多为取 R 棵决策树预测结果的平均值；在分类问题中一般使用“投票制”决定最终的分类结果），从而实现随机森林算法。

（二）大规模随机森林算法

BLOCK-SDB-RF 利用“分治”思想处理大规模数据，算法设计主要解决由样本量大、特征维数高导致的计算效率低的问题。通过对样本维度与特征维度同时进行处理，BLOCK-SDB-RF 方法可以有效减少计算时间，提高计算效率。

同样地，令 $\chi_n = (\mathbb{X}_1, \mathbb{X}_2, \cdots, \mathbb{X}_n)^{\mathrm{T}}$ 表示来自某联合分布 F 的独立同分布随机样本，其中 $\mathbb{X}_i = (\mathbb{X}_{i1}, \mathbb{X}_{i2}, \cdots, \mathbb{X}_{ip})^T$，$\chi_p = (X_1, X_2, \cdots, X_p)$ 表示 p 维特征变量。研究者希望基于该组随机样本 χ_n 构建模型以实现分类（回归）的功能。为有效提高运算效率，针对特征维度，将 $\chi_p = (X_1, X_2, \cdots, X_p)$ 随机划分为 c 组 $\chi_p^{*i} = (X_{t_{i1}}, X_{t_{i2}}, \cdots, X_{t_{im}})$，$i = 1, 2, \cdots, c$。然后从 χ_n 中随机抽取比例 q 的样本构成样本集合 $\chi_{n,q}$，在 $\chi_{n,q}$ 上分别利用特征 $\chi_p^{*i} = (X_{t_{i1}}, X_{t_{i2}}, \cdots, X_{t_{im}})$，$i = 1, 2, \cdots, c$ 构建决策树 $T(\chi_{n,q}, \chi_p^{*i})$，计算得到 χ_p^{*i} 中每个变量的重要性（通过 Gini 指数或信息熵减少量度量）$W^{(i)} = (w_{t_{i1}}^{(i)}, w_{t_{i2}}^{(i)}, \cdots, w_{t_{im}}^{(i)})$，进而以此作为权重从 $\chi_p^{*i} = (X_{t_{i1}}, X_{t_{i2}}, \cdots, X_{t_{im}})$，$i = 1, 2, \cdots, c$ 中共抽取 z 维特征变量 $\chi_p^{(f)} = (X_{t_1'}, X_{t_2'}, \cdots, X_{t_z'})$ 进入到模型构建中。

针对样本维度，首先将样本 χ_n 随机等量划分为 b 个大小为 s 但不重叠的子样本集 $\chi_{n,s}^{*j} = (X_{j1}, X_{j2}, \cdots, X_{js})$，$j = 1, 2, \cdots, b$。基于每个子样本集 $\chi_{n,s}^{*j}$ 进行一次 Bootstrap 抽样将子样本集的样本量恢复到原数据集大小 n，得到 $\chi_{n,s}^{'j} = (\chi_{n,s}^{*j}, W_{n,s}^{*j})$。其中 $W_{n,s}^{*j} = (w_1^j, w_2^j, \cdots, w_s^j)$ 表示子样本集 $\chi_{n,s}^{*j}$ 中的 s 个样本在 $\chi_{n,s}^{'j}$ 中出现的频率。然后在每个 $\chi_{n,s}^{'j}$ 上，利用特征 $\chi_p^{(f)} = (X_{t_1'}, X_{t_2'}, \cdots, X_{t_z'})$ 构建决策树 $T(\chi_{n,s}^{'j}, \chi_p^{(f)})$，最终将 $T(\chi_{n,s}^{'j}, \chi_p^{(f)})$，$j = 1, 2, \cdots, b$ 的结果合并进行决策。

显然地，BLOCK-SDB-RF 方法对特征维度的分组降维解决了随机森林算法在特征维数较高时带来的过拟合及信息损失的问题。同时，以特征的重要性为权重在每组特征中随机抽取一部分（所有组总计抽取 z 维特征）作为最终建模的变量，这样既保证了所有特征变量均有进入到模型的可能，又增大了重要特征被抽中的可能性，从而减少算法的信息损失。另一方面，通过对样本维度的随机分块，减少了构建每棵决策树所需样本中不重复样本的数量，有利于降低每棵决策树的运行时间。即便最终子集的样本量恢复到与原数据集相同，但考虑到子集中不重复的样本数至多只有 s 个，计算时只需通过对样本加权即可用较短的时间得出结果，并不会因此增加运算成本。算法的整体思路与流程见图 1。

（三）时间复杂度及数据覆盖率分析

1. 时间复杂度分析

算法的时间复杂度需要从样本维度和特征维度两方面考虑。从样本维度看，随机森林算法需对原数据集进行 R 次 Bootstrap 抽样，所得的每一个子样本集至多包含 n 个不同的样本，其样本维度的时间复杂度可以表示为 $(R+1)\times t(n)$；BLOCK-SDB-RF 在 b 个大小为 s 的子样本集上分别进行一次有放回抽样，所得样本集最多只包含 s 个不同的样本，相应样本维度的时间复杂度可记为 $2b\times t(s)$。从特征维度看，由于随机森林算法对每一个 Bootstrap 子样本集都从 p 维特征中抽取比例为 1/2 的变量建立一棵决策树，因此特征维度的时间复杂度为 $(R+1)\times t(p)$；同理，考虑到 BLOCK-SDB-RF 方法通过

分组降维将变量维数降低至 z，相应特征维度的时间复杂度为 $2b\times t(z)$。因此，BLOCK-SDB-RF 的总时间复杂度可记为 $2b\times t(sz)$，随机森林算法的总时间复杂度记为 $(R+1)\times t(np)$。在单机运算的前提下，运算成本的对比情况会受到 Bootstrap 抽样次数 R 和子样本集数量 b 的大小的影响。然而 Sengupta 等提到，在 BLB 方法中推荐 R 取值为 100，b 根据情况推荐取值在 2 到 10 之间，因此一般来讲，BLOCK-SDB-RF 方法的时间成本更加低廉。在并行计算的前提下，由于 s、z 相较 n、p 更小，显然有 $t(sz)<t(np)$，当计算资源无限充足时，BLOCK-SDB-RF 方法的时间复杂度更低，可以大大减少计算时间；当计算资源有限，比如只有 v 核 CPU 时，BLOCK-SDB-RF 方法和 RF 方法的时间成本分别为 $2b/v\times t(sz)$，$(R+1)/v\times t(np)$，如果按照 Sengupta 等的推荐取值，两种方法进行对比时也是 BLOCK-SDB-RF 方法的时间复杂度更低。BLOCK-SDB-RF 算法：

输入：

样本数据 $\chi_n=(\tilde{X}_1,\tilde{X}_2,\cdots,\tilde{X}_n)^T$；样本子集数量 b；样本子集大小 s；变量维数 p；样本量 n；变量的分组数 c；进入随机森林的变量数 z。

输出：分类结果或回归结果。

过程：

1. 将 $\chi_p=(X_1,X_2,\cdots,X_p)$ 随机划分为 c 组：

$\chi_p^{*i}=(X_{t_{i1}},X_{t_{i2}},\cdots,X_{t_{im}})\quad i=1,2,\cdots,c$

2. 从 χ_n 中随机抽取比例 q 的样本构成样本集合 $\chi_{n,q}$

3. for $i\to 1$ to c

在 $\chi_{n,q}$ 上分别利用特征 χ_p^{*i} 构建决策树 $T(\chi_{n,q},\chi_p^{*i})$，计算变量的重要性度量：

$W^{(i)}=(w_{t_{i1}}^{(i)},w_{t_{i2}}^{(i)},\cdots,w_{t_{im}}^{(i)})\quad i=1,2,\cdots,c$

end

4. 在每一个特征集合 χ_p^{*i} 中，以 $W^{(i)}=(w_{t_{i1}}^{(i)},w_{t_{i2}}^{(i)},\cdots,w_{t_{im}}^{(i)})$ 为权重随机抽取特征，共抽取 z 个特征：

$\chi_p^{(f)}=(X_{t'_1},X_{t'_2},\cdots,X_{t'_z})$

5. 将样本 $\chi_n=(\tilde{X}_1,\tilde{X}_2,\cdots,\tilde{X}_n)^T$ 随机划分为 b 个大小为 s 但不重叠的子样本集：

$\chi_{n,s}^{*j}=(X_{j1},X_{j2}\cdots,X_{js})\quad j=1,2,\cdots,b$

6. for $j\to 1$ to b

在 $\chi_{n,s}^{*j}=(X_{j1},X_{j2}\cdots,X_{js})$ 上进行一次 Bootstrap 抽样，得到 $\chi_{n,s}^{'j}=(\chi_{n,s}^{*j},W_{n,s}^{*j})$

在 $\chi_{n,s}^{'j}$ 上，利用特征 $\chi_p^{(f)}=(X_{t'_1},X_{t'_2},\cdots,X_{t'_z})$ 构建决策树 $T(\chi_{n,s}^{'j},\chi_p^{(f)})$

end

7. 结合 b 棵决策树的分类结果或回归结果进行综合决策

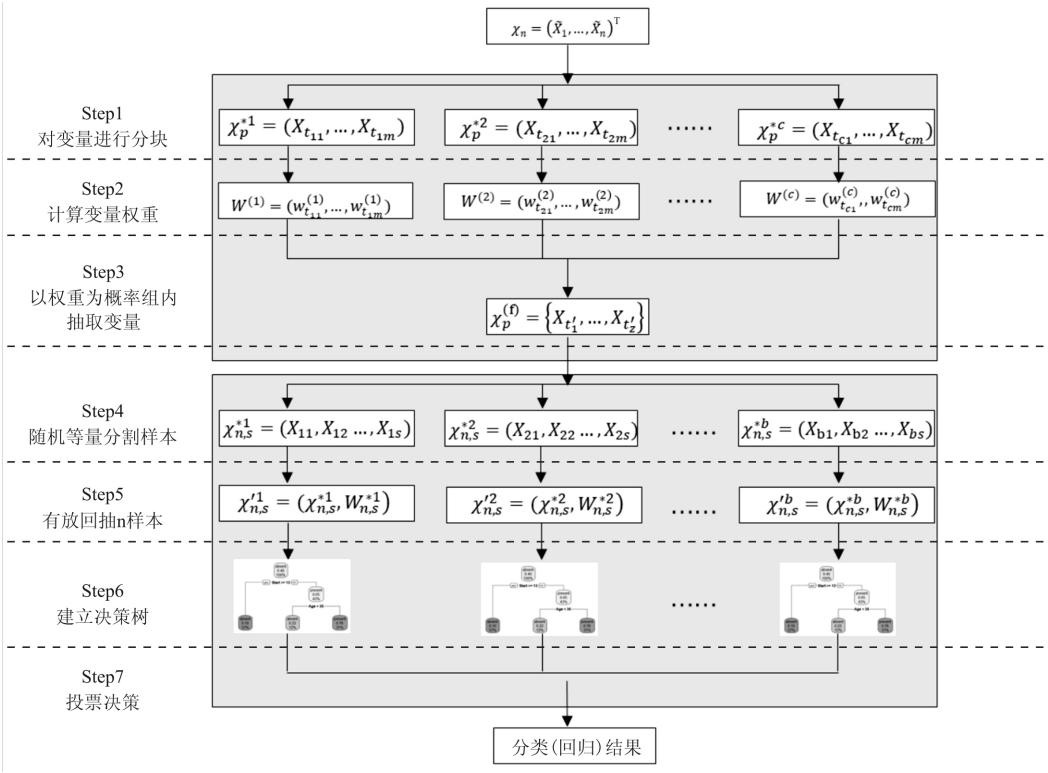

图 1　BLOCK-SDB-RF 算法结构图

2. 数据覆盖率分析

为了在数据覆盖率方面对 BLOCK-SDB-RF 方法、RF 方法实现更加公平比较，研究者在给定相同的时间成本下，分别计算两种方法对数据的覆盖情况。同时，为了便于比较，提出了两个假设条件：

条件 1：不失一般性，假定 RF 方法 Bootstrap 抽样次数 $R=1$；

条件 2：假定时间复杂度 $t(np)$、$t(sz)$ 满足条件 $t(np)\geqslant bt(sz)$。

提出条件 1 的目的是为了限制时间成本，在给定 RF 方法 Bootstrap 抽样次数 $R=1$ 的前提下，相应的时间成本为 $(R+1)\times t(np)$。条件 2 的提出给定了时间复杂度 $t(np)$、$t(sz)$ 的大小关系，该条件的成立意味着在一个样本量为 n、特征维数为 p 的数据集上建立一棵决策树的耗时大于或等于在一个样本量为 s、特征维数为 z 的数据集上建立决策树耗时的 b 倍。这也是为了实现数据覆盖率的对比而提出的条件，事实上，两个时间复杂度的大小未必总是符合线性关系的条件。从数据覆盖率看，当满足以上两个条件时，在相同的时间成本下 BLOCK-SDB-RF 方法能覆盖到更多的数据。事实上，在随机森林算法的实现过程中，每一次利用 Bootstrap 抽取子样本集时，由于单个样本的入样概率为 $1/n$，重复抽取 n 次后，大约有 $(1-1/n)^n$ 的样本在整个算法中不会参与训练模型。当 n 趋于无穷大时，约有 $1/e$ 的样本从未被抽中，即在随机森林算法的一次 Bootstrap 抽样中，样本覆盖率约为 $1-1/e$。同样地，在 BLOCK-SDB-RF 中每一个子集的 Bootstrap 抽样大约有 $s/n\times(1-1/s)^n$ 的样本在整个算法中不会参与训练模型。当数据集的样本量特别高时，可以认为 BLOCK-SDB-RF 方法的一个子集样本 Bootstrap 抽样覆盖率是 $s/n\times[1-(1/e)^b]$。当 Bootstrap 抽样次数 $R=1$ 时，RF 方法的数据覆盖率约为 $1-1/e$，在相同的时间成本下利用 BLOCK-SDB-RF 方法进行计算时，由于 $t(np)\geqslant bt(sz)$，可以覆盖到 $b\times\{s/n\times[1-(1/e)^b]\}$ 的数据，即此时 BLOCK-SDB-RF 方法的数据覆盖率为 $1-(1/e)^b$，显然高于随机森林算法。

三、数值模拟

为证明 BLOCK-SDB-RF 方法在不同类型实际数据中具有良好的应用效果，本文以随机森林方法（以下记为 RF）为参照，分别设置了不同样本量、不同特征维数场景下的模拟分析，同时考虑到变量间往往并不完全相互独立，增加第三种模拟场景以探究在变量间相关性不同的情况下两种方法的效果对比。

在模拟数据中，$X_{n\times p}$ 从均值为零向量的 p 元正态分布中生成，该多元正态分布的协方差矩阵为分块对角矩阵，如式(5)所示。协方差矩阵由虚线划分为 4 部分，其中设定第一部分大小为 10×10 的子矩阵中元素非零，即除前 10 个变量间具有系数为 cor 的相关关系以外，其他变量间均不具有相关性。$Y_{n\times1}$ 由线性回归模型 $Y=X\beta+\varepsilon$ 生成，然后通过 Sigmoid 变换将其转化为 0－1 变量。其中，随机干扰项 ε 服从标准正态分布 $N(0,1)$，自变量系数 β 设为 $\{a,\cdots,a,-a,\cdots,-a,0,\cdots,0\}$，$a$ 与 $-a$ 的数量均为 10，且 a 从正态分布 $N(3,0.5)$ 中产生。

$$\left[\begin{array}{ccc:ccc} 1 & \cdots & \text{cor} & 0 & \cdots & 0 \\ \vdots & \ddots & \text{cor} & \vdots & \cdots & 0 \\ \text{cor} & \cdots & 1 & 0 & \cdots & 0 \\ \hdashline 0 & \cdots & 0 & 1 & \cdots & 0 \\ 0 & \cdots & 0 & 0 & \cdots & 0 \\ 0 & \cdots & 0 & 0 & \cdots & 1 \end{array}\right] \tag{5}$$

研究采用以下指标以度量 BLOCK-SDB-RF 方法的应用效果。其中 TPR 为召回率，表示所有正类中，有多少被预测成正类（正类预测正确）。TNR 表示所有反类中，有多少被预测成反类（反类预测正确）。预测正确率 Accuracy 表示所有样本中分类正确的比例，指标 G-mean 的含义比 TPR、TNR 更加综合，具体定义为：

$$\text{G-mean}=\sqrt{\text{TPR}\times\text{TNR}}$$

与预测正确率 Accuracy 的区别在于当正负样本存在不平衡时 G-mean 的参考价值较大。四类指标均为越高越好。

（一）不同样本量

在第一个模拟场景中，设定特征维数 $p=1\,000$，变量间相关系数 cor=0.5。分别考虑数据样本量为 40 000、60 000、80 000、100 000 的情况下，BLOCK-SDB-RF 方法和 RF 方法的应用效果，结果见图 2。图 2 中 (a)、(b)、(c)、(d) 的横轴名称为 t，表示计算耗费的时间，纵轴为指标 G-mean，用以衡量预测结果的准确率，该指标越大意味着预测结果越准确；图 2 中实线表示 BLOCK-SDB-RF 方法，虚线表示 RF 方法。为便于对两种方法进行对比，借鉴 Sengupta 等的思路考虑在相同的计算时间内哪种方法能以更快的速度达到合理准确的结果，此处

结合模拟场景将时间限定为 800 秒[6]。

由图 2（a）可见，BLOCK-SDB-RF 方法的 G-mean指标在 30 秒附近急剧增加并很快达到稳定状态，较短时间内完成了计算；而 RF 方法的 G-mean指标在近 200 秒附近才有相似的变化，800 秒内并没有完成计算。可见与 RF 方法相比，BLOCK-SDB-RF 方法的计算以更快的速度达到收敛，图 2(b)、(c)、(d)的情况类似。同时，与图 2(b)、(c)、(d)对比发现，随着数据的样本量不断增加，BLOCK-SDB-RF 与 RF 两种方法的 G-mean 指标达到一定程度的耗时差距越来越大，表明 BLOCK-SDB-RF 方法在数据的样本量越大时计算速度的优势越突出。

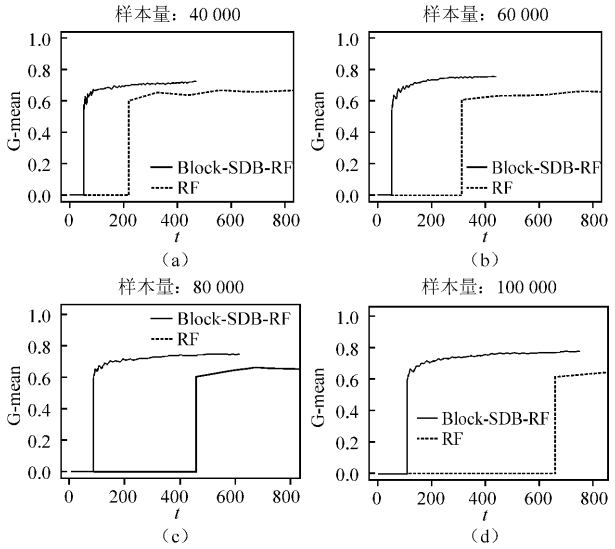

图 2　不同数据量下的方法对比图

另一方面，为了进一步比较两种算法的准确性，给出四种指标 G-mean、TPR、TNR，预测正确率的对比情况，见表 1。其中，TPR、TNR 分别表示正、负样本中预测正确的比例，G-mean、预测正确率度量了总体的预测准确率，四类指标均为越高越好。由表 1 见，在不同的样本数量下，BLOCK-SDB-RF 方法的表现基本上优于 RF 方法。

表 1　算法准确性对比

样本量	方法	G-mean	TPR	TNR	预测正确率
n=40 000	RF	0.664	0.606	0.729	0.667
	BLOCK-SDB-RF	0.723	0.684	0.765	0.724
n=60 000	RF	0.654	0.579	0.739	0.660
	BLOCK-SDB-RF	0.754	0.716	0.793	0.755
n=80 000	RF	0.636	0.555	0.729	0.641
	BLOCK-SDB-RF	0.730	0.734	0.726	0.730
n=100 000	RF	0.652	0.662	0.643	0.652
	BLOCK-SDB-RF	0.760	0.739	0.782	0.761

（二）不同特征维数

第二种场景下，设定数据的样本量 n=40 000，变量间相关系数 cor=0.5。比较 RF 和 BLOCK-SDB-RF 两种方法在特征维数分别为 1 000，2 000，3 000，5 000时的表现，结果见图 3。考虑到变量维数增加，为了更清晰的展示结果，将时间限定为 1 600 秒。

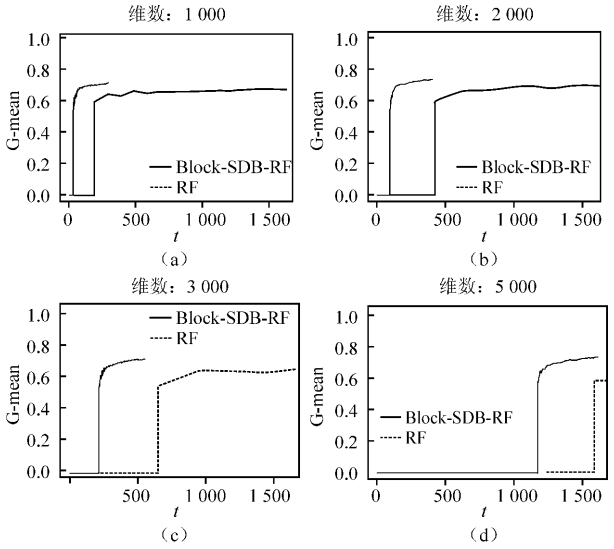

图 3　不同特征维数下的方法对比图

与第一种模拟场景相同，图中横轴为 t，表示计算耗费的时间，纵轴为指标 G-mean，实线表示 BLOCK-SDB-RF 方法，虚线表示 RF 方法。结果显示，在特征维数为 1 000 时，BLOCK-SDB-RF 方法的 G-mean 指标 50 秒左右达到了稳定状态，仅在 500 秒内就完成了计算；而 RF 方法在 200 秒附近 G-mean 指标才有显著的增长，具体参见图 3(a)。同时由图 3(b)、(c)、(d)可见，不论特征维数高低，BLOCK-SDB-RF 方法总是可以在更短时间完成计算，并以比 RF 方法更快的计算速度使结果的 G-mean指标达到稳定。同时，随着特征维数的不断增加，两种方法所需的计算时间也越长。特别地，BLOCK-SDB-RF 达到一定预测精度的耗时短于 RF 方法的程度越来越大，表明 BLOCK-SDB-RF 在特征维数不断增长时计算速度的优势越来越明显。

表 2　算法准确性对比

维数	方法	G-mean	TPR	TNR	预测正确率
p=1 000	RF	0.681	0.632	0.733	0.683
	BLOCK-SDB-RF	0.723	0.684	0.765	0.724
p=2 000	RF	0.682	0.611	0.762	0.686
	BLOCK-SDB-RF	0.721	0.712	0.731	0.721
p=3 000	RF	0.665	0.671	0.659	0.665
	BLOCK-SDB-RF	0.720	0.676	0.766	0.721
p=5 000	RF	0.656	0.665	0.646	0.656
	BLOCK-SDB-RF	0.703	0.697	0.709	0.703

同样的，从分类准确性的角度来看，不论特征维度高低，BLOCK-SDB-RF 方法的表现也基本上优于 RF 方法。特别的，随着特征维度的增加，RF 方法的 G-mean 和预测正确率指标有逐步下降的趋势，而 BLOCK-SDB-RF 方法的准确率相对更加稳定。

（三）不同变量相关性

在实际中，变量间往往并不是完全相互独立的，为探究变量间的不同相关性对两种方法计算效果的影响，本节设定数据的样本量 $n=40\ 000$，$p=1\ 000$，比较在变量间相关性分别为 0.1，0.3，0.6，0.9 四种情况下，BLOCK-SDB-RF 和 RF 方法在计算速度方面的差异。同样地，为便于对两种方法进行对比，此处结合模拟场景将时间限定为 1 000 秒，结果见图 4。

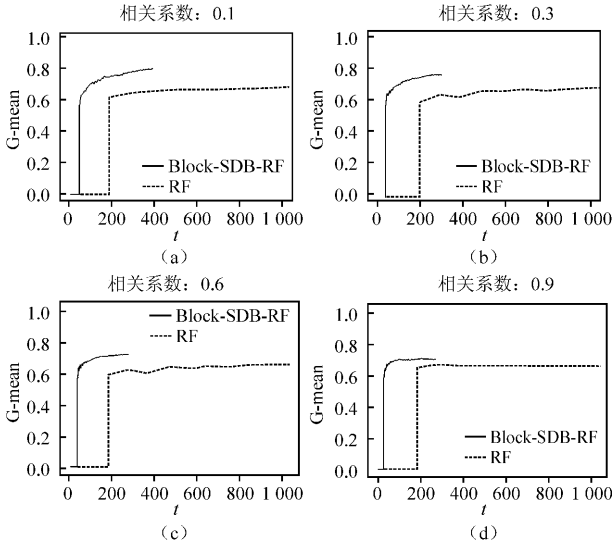

图 4　不同变量间相关性的方法对比图

表 3　算法准确性对比

相关性	方法	G-mean	TPR	TNR	预测正确率
cor=0.1	RF	0.696	0.695	0.697	0.696
	BLOCK-SDB-RF	0.813	0.782	0.844	0.813
cor=0.3	RF	0.701	0.702	0.699	0.701
	BLOCK-SDB-RF	0.779	0.741	0.818	0.779
cor=0.6	RF	0.664	0.670	0.657	0.664
	BLOCK-SDB-RF	0.724	0.709	0.739	0.724
cor=0.9	RF	0.656	0.665	0.646	0.656
	BLOCK-SDB-RF	0.703	0.697	0.709	0.703

图 4(a)显示，在变量间相关系数为 0.1 时，BLOCK-SDB-RF 方法的 G-mean 指标在 40 秒左右快速增长，200 秒左右就完成了运算；而 RF 方法在限定的 1 000 秒内未完成计算，且在 200 秒附近G-mean 才开始急剧增加，再一次表明与 RF 相比，BLOCK-SDB-RF 方法具有更高的计算效率。同时，由图 4 的(b)、(c)、(d)发现，变量间的不同相关性并不影响 BLOCK-SDB-RF 在计算速度方面的优势，BLOCK-SDB-RF 始终以明显较快的速度达到收敛。然而由表 3 可见，随着变量间相关性的增加，在对全部特征进行分组降维时受到的干扰也越来越大，因此 BLOCK-SDB-RF 方法的预测准确率会有明显下降，RF 方法的分类准确率也会受到一定程度的影响。但从总体上讲，相比 RF 方法，BLOCK-SDB-RF 的分类准确率更高。

四、实证分析

本文的实证数据是由音乐流媒体服务商 KKBOX 提供的日志数据，来源于 2017 年 Kaggle 平台举办的一场数据挖掘竞赛。该竞赛向大众提供了训练集合表、测试集合表等按时间顺序划分的日志数据与歌曲信息表、用户信息表、歌曲补充信息表共 5 个表格。为便于分析，本文对该竞赛提供的 5 个表格数据进行了清洗与整合，并结合特征工程从用户、歌曲及两者交互三个维度提取不同的特征变量，最终得到样本量为 100 000、特征维数为 1 089 的大规模数据集（除去歌曲 id、用户 id 及因变量“是否重复听歌”三个变量）。数据集的整体结构如图 5 所示，部分变量及含义见表 4。

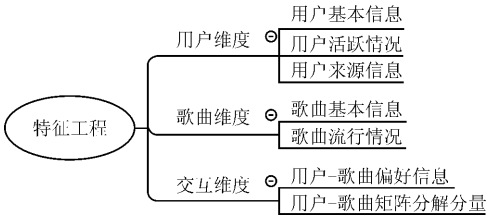

图 5　特征维度示意图

表 4　数据特征示例（部分）

维度	特征简称	特征简介	特征编号
用户	Source system tab	用户点击的系统按钮	X_3
	Source screen name	用户点击的界面名称	X_4
	City	用户城市	X_{35}
	Gender	用户性别	X_{37}
	Msno active cnt	用户活跃次数	X_{87}
	Active time avg	平均活跃时间	X_{89}
歌曲	Song length	歌曲长度	X_6
	First artist name	歌手	X_9
	Issue country	发行国家	X_{12}
	Issue press	出版社	X_{13}
交互	Mnso language prb	某用户偏好语种比例	X_{27}
	Mnso first genre id prb	某用户偏好曲风比例	X_{28}
	Members partial (i)	用户-歌曲矩阵，SVD 分解后用户矩阵分量	$X_{40}-X_{86}$

文章以“用户重复听歌”事件作为准则判断用户对歌曲是否喜爱，即如果用户在第一次点击某首歌曲之后的一个月里触发“重复听歌”事件，则认为用

户喜爱该歌曲，将因变量记为 1，否则记为 0。考虑到数据集样本量大，特征维数高，本文利用 BLOCK-SDB-RF 方法在训练集上建模，并在测试集上对用户是否会重复听歌进行预测，以期更好地了解用户的听歌偏好。

（一）初步探索

在建模之前，对用户、歌曲及两者交互三个维度下变量间的相关关系进行初步探索。考虑到变量间的关系并不能由简单的线性相关进行刻画，文中采用了秩相关系数，部分变量间的相关关系如图 6 所示。图 6 中带斜线阴影部分表示负相关，未带斜线阴影部分代表正相关，颜色的深浅表示变量间相关关系的强弱，颜色越深，意味着相关关系越强。

结果显示，用户点击的来源信息（X_{26}，X_{25}）即用户通过点击电台、专辑或者其他方式进入该歌播放的比例、用户最后听歌时间（X_{91}）、歌曲流行程度（X_{23}）对因变量 Y 是否重复听歌的影响比较显著。歌曲—用户交互维度中，特定用户对歌手（X_{29}）、作曲家（X_{30}）、作词人（X_{31}）的喜好也对 Y 有影响，但是相关性较之用户维度、歌曲维度的变量更弱。另一方面，不同特征间也具有一定程度的相关关系，比如 X_{20}用户对发行年份的偏好、X_{21}某歌曲计数、X_{22}某用户计数三个特征间具有较强的正相关关系等。但是考虑到在数值模拟部分发现，特征变量间的相关性并不会影响 BLOCK-SDB-RF 方法在计算时间方面的优势，因此仍然可以使用该方法对数据进行建模、预测。

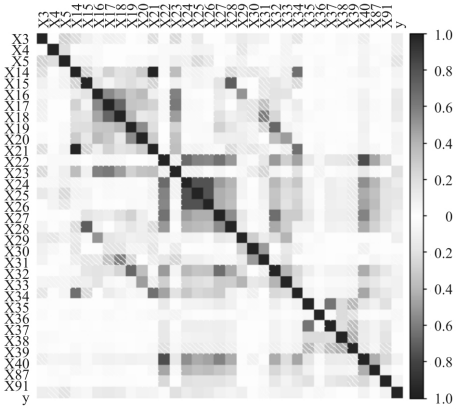

图 6　部分变量间的秩相关图

（二）实证结果

将清洗后的数据集按照 3∶1 划分为训练集与测试集。在训练集上分别利用 RF、BLOCK-SDB-RF 两种方法建立模型，然后在相应测试集上进行预测，得到实证结果如下。

1. 相同时间内计算结果对比

为节约时间成本并方便结果对比，同样借鉴 Sengupta 等的思路，考虑在 1 000 秒时间内哪种方法能以更快的速度达到合理准确的结果，最终结果如图 7 所示[6]。图 7 中横轴为 time，表示计算消耗的时间；纵轴为指标 G-mean，实线表示 BLOCK-SDB-RF 方法，虚线表示 RF 方法。结果显示，RF 方法在测试集预测结果的 G-mean 指标于 700 秒附近开始急剧增加并达到一定高度，而 BLOCK-SDB-RF 方法在 400 秒附近就发生了类似变化，表明 BLOCK-SDB-RF 方法在计算时间方面具有一定优势。

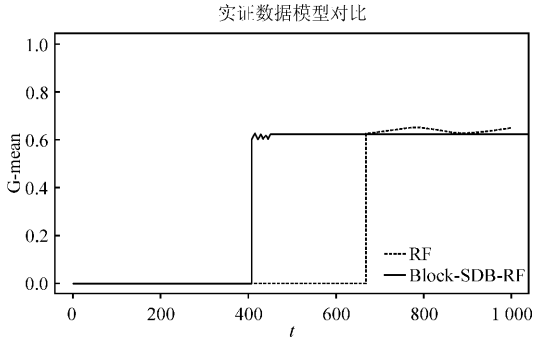

图 7　实证数据结果对比图

2. 不限制计算时间的结果对比

当不限制计算时间时，分别使用 RF、BLOCK-SDB-RF 方法对测试集做出预测，其中 RF 方法共建立 200 棵决策树，得到预测结果见表 5。结果显示，BLOCK-SDB-RF 方法预测的总体准确率为 75.9%，G-mean 为 59.5%，与 RF 方法得到的计算结果基本一致。但是从计算时间角度看，BLOCK-SDB-RF 方法共消耗计算时间 13.53 分钟，与 RF 方法建立 200 棵决策树需要耗费的 20.19 小时相比，大大降低了计算成本。特别是在更加大型的推荐系统中该方法的优势将更加明显。

表 5　预测结果

方法	G-mean	TPR	TNR	预测正确率
RF	0.623	0.903	0.430	0.762
BLOCK-SDB-RF	0.595	0.917	0.386	0.758

3. 变量重要性分析

在 BLOCK-SDB-RF 实施过程中，将分组考察各个特征变量对分类结果的贡献，并以此作为该特征变量的重要性度量。表 6 为重要性排名前 10 的特征变量及其与因变量 Y 的秩相关系数。

由表 6 可见，BLOCK-SDB-RF 方法下重要性排名前 10 位的特征变量，比如 X_{23}某歌曲计数（即歌曲的流行程度）、X_{25}某用户不同界面来源比例、X_{26}某用户 app 上音乐入口比例等，与因变量 Y 的秩相

关系数也较高，分别为 0.184，0.060，0.134。这与建模之前初步探索得出的结果大体一致，即与因变量 Y 越相关的变量越能解释 Y 中包含的信息。相比而言，用户、歌曲维度下的变量比两者交互维度的变量更重要。

表 6　变量重要性排名与秩相关系数

变量名称	编号	中文解释	重要排名	秩相关
Source type	X_5	用户 app 上播放音乐的入口	1	−0.182
Source screen name	X_4	用户点击的界面名称	2	−0.113
Source system tab	X_3	用户点击的系统按钮	3	0.108
Song id cnt	X_{23}	某歌曲计数	4	0.184
Msno source screen name prb	X_{25}	某用户不同界面来源比例	5	0.160
Member partial 0	X_{40}	SVD 分解后用户矩阵分量	6	0.077
Msno cnt	X_{22}	某用户计数	7	0.024
Msno groupby first composer	X_{17}	所有用户对作曲家的偏好	8	0.131
Msno source type prb	X_{26}	某用户 app 上音乐入口比例	9	0.134
Msno language prb	X_{27}	某用户偏好语种比例	10	0.089

五、讨 论

本文以随机森林算法为基础，结合“分治”思想提出了大规模随机森林算法，对样本量大、特征维度高的数据集进行分析。在尽量保证算法有效性的前提下，解决了大规模数据带来的计算效率问题。通过数值模拟与实证分析表明，随着数据量、特征维度的增加，该方法在计算时间上的优势愈发明显，同时变量间的相关性对该方法在计算效率方面的优势影响并不显著。然而，在应用方面 BLOCK-SDB-RF 方法仍然存在一些问题有待进一步探讨。

第一，本文以随机森林为基础算法，通过同时对样本维度和特征维度进行处理，将其内嵌到本文提出的分析框架中以应对大规模数据带来的计算效率的挑战。事实上，本文提出的大规模数据分析框架并不局限于随机森林算法，其他常用的机器学习算法比如支持向量机（SVM）等均可以嵌进其中。

第二，KKBOX 提供的日志数据实际上是一个不平衡数据，其中包含的正样本（因变量为“重复听歌”的样本）数量远远大于负样本（因变量为“未重复听歌”的样本），并且文中并没有针对这种情况做出处理，因此在实证分析部分得到的预测结果会出现真阳性率较高，真阴性率较低的情况。事实上，这种不平衡数据集在实际生活中非常普遍，比如贷款违约预测的数据、医疗数据等，针对这种数据在因变量上分布不平衡的问题，可以考虑通过数据层面或者算法层面的调整解决[17−18]，比如在数据层面上利用过采样、欠采样将不平衡数据调整为平衡数据，或者类似调整的支持向量机算法等直接对算法进行修正以适应不平衡数据。

第三，变量间的相关性虽然并不影响 BLOCK-SDB-RF 方法在计算时间方面的优势，但随着相关性的增加，预测结果的准确率会因此有所下降。即该方法在提升计算效率的同时，有时会以牺牲预测精度为代价。特别地，在实际数据中，变量间具有一定程度的相关性总是不可避免的。这一方面要求我们在预测的准确率和计算效率上做出权衡，另一方面也需要考虑当变量间具有较强的相关性时，如何减轻变量选择结果受到的干扰，为未来研究指明方向。

参考文献：

[1] Chen M, Mao S, Liu Y. Big Data: A Survey[J]. Mobile Networks and Applications, 2014, 19(2): 171-209.

[2] 李金昌. 大数据与统计新思维[J]. 统计研究, 2014, 31(1): 10-17.

[3] Fan J, Fan Y. High Dimensional Classification Using Features Annealed Independence Rules[J]. Annals of Statistics, 2008, 36(6): 2605.

[4] Bickel P J, Gotze F, Zwet W R V. Resampling Fewer than n Observations: Gains, Losses, and Remedies for Losses[J]. Statistica Sinica, 1997, 7(1): 1-32.

[5] 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.

[6] Sengupta S, Volgushev S, Shao X. A Subsampled Double Bootstrap for Massive Data[J]. Publications of the American Statistical Association, 2016, 111(515): 11.

[7] Denning P J. Parallel Computing and Its Evolution [J]. Science & Technology Review, 1988(1): 34-37.

[8] 钟龙申. 随机森林算法处理不平衡数据的改进及其并行化[D]. 广东：广东工业大学, 2016.

[9] Luo D, Ding C, Huang H, et al. Parallelization with Multiplicative Algorithms for Big Data Mining[C]. International Conference on Data Mining, 2012: 489-498.

[10] Fang K，Ma S. Analyzing Large Datasets with Bootstrap Penalization[J]. Biometrical Journal，2017，59(2)：358-376.
[11] 吴琼，李运田，郑献卫. 面向非平衡训练集分类的随机森林算法优化[J]. 工业控制计算机，2013，26(7)：89-90.
[12] 曹正凤. 随机森林算法优化研究[D]. 北京：首都经济贸易大学，2014.
[13] 汪桂金. 随机森林算法的优化改进及其并行化研究[D]. 江西：南昌大学，2019.
[14] Paul A，Mukherjee D P. Reinforced Random Forest[C]. Indian Conference on Computer Vision，Graphics and Image Processing，2016：1-8.
[15] 马景义，吴喜之，谢邦昌. 拟自适应分类随机森林算法[J]. 数理统计与管理，2010，29(5)：806-811.
[16] Gall J，Lempitsky V. Class-Specific Hough Forests for Object Detection[M]// Decision Forests for Computer Vision and Medical Image Analysis. London：Springer，2013：143-157.
[17] Ganganwar V. An Overview of Classification Algorithms for Imbalanced Datasets[J]. International Journal of Emerging Technology and Advanced Engineering，2012，2(4)：42-47.
[18] 张景肖，魏秋萍，张波. 信用评分模型中稀有事件特殊采样处理方法探讨[J]. 统计与信息论坛，2012，27(11)：15-19.

A Distributed Random Forest Algorithm for Massive Data

LI Yang[1a,1b,1c]，QI Le[1b,1c]，NIE Pei-yun[2]

(a. Center for Applied Statistics，b. School of Statistics，c. Center for Statistical Consulting，1. Renmin University of China，Beijing 100872，China；2. International Business Group，Tencent，Shenzhen 518057，China)

Abstract：The rapid development of technology has improved people's ability to produce and collect data，quantification and high dimensionality become the main characteristics for more and more data. These Large-scale data brings challenges in computational efficiency to statistical analysis. An analysis framework is proposed according to the idea of " divide and conquer"，and a distributed random forest algorithm for massive data（BLOCK-SDB-RF）is designed. The researchers analyze the advantages of the proposed algorithm from the perspective of data coverage and time complexity，and discuss the performance of the BLOCK-SDB-RF algorithm through numerical simulation. The numerical simulations show that：(1) With the increase of data sample size and feature dimension，the advantage of the algorithm in computational efficiency becomes more and more obvious；(2) Although the correlation between variables does not influence the computational efficiency of the algorithm，the researchers need to sacrifice some of the prediction accuracy as the correlation increases. In the empirical analysis，the log data is provided by music streaming service provider KKBOX as an example to further explore the role of BLOCK-SDB-RF algorithm in massive data analysis.

Key words：big data；computational efficiency；random forest；distributed computing

（责任编辑：马　慧）