基于随机森林的文本分类并行化

彭 徵 王灵矫 郭 华
(湘潭大学信息工程学院 湖南 湘潭 411105)

摘 要 文本分类是信息检索的核心技术。传统的文本分类系统由于单机的计算与存储能力有限,已经不适用于大数据时代。在 Spark 大数据平台上并行地运行算法对文本进行分类,以数据和任务的并行化来提高算法的效率具有现实性和紧迫性。文中提出了改进的不平衡数据随机森林算法,通过对训练样本的多数类进行欠取样且对少数类进行有放回取样从而形成新训练样本的方法来减少不平衡数据对随机森林的影响。实验结果表明,新算法在处理不平衡数据集上的少数类时提高了分类的正确率。

关键词 文本分类,Spark,随机森林,不平衡数据,并行化
中图法分类号 TP311 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2018.12.023

Parallel Text Categorization of Random Forest

PENG Zheng WANG Ling-jiao GUO Hua
(The College of Information Engineering, Xiangtan University, Xiangtan, Hunan 411105, China)

Abstract Text categorization is one of the core technologies of information retrieval. Because of the limited computing performance and storage capacity in a computer, the traditional text categorization method can't be suitable for big data era nowadays. It is realistic and urgent to execute algorithms for classifying the text in parallel to improve the efficiency of algorithm by the parallelization operation of data and tasks on the big data platform of Spark. This paper proposed an improved random fo-rest algorithm for the imbalanced data. It can reduce the impact of imbalanced data on random forests by under-sampling the majority class samples and back-sampling the minority class samples to make up new training samples. The experimental results show that the new algorithm improves the categorization accuracy of the minority classes when handling imbalanced data sets.

Keywords Text categorization, Spark, Random forest, Imbalanced data, Parallelization

1 引言

文本分类[1]的关键问题就是如何准确地判断文本的类型,管子并研究这一问题后总结了许多算法:KNN 算法[2]、贝叶斯分类算法[3]、支持向量机算法[4]、决策树算法[5]等。KNN 算法和贝叶斯分类算法研究和使用得最多,但随机森林算法的分类效果比其他几种算法的分类效果更好。贺捷[6]指出随机森林算法的准确率比 KNN 算法和贝叶斯算法的都高,并且维度越高效果越明显。然而,在使用随机森林算法的过程中也出现了一个问题:由于随机森林算法在训练过程中需要建立多个分类器,运算时间较长,一般情况下运行时间是其他算法运算时间的一倍以上。目前,采用并行化的方式是减少随机森林运行时间的最好方法。文献[7-8]使用 Hadoop 平台的 MapReduce 框架对算法进行并行化运算,使得算法的运行效率远高于普通单机算法。Spark 分布式并行计算框架的运行效率高于 Hadoop 框架[9],将随机森林算法集成到 Spark 框架上进行并行化的运算效果会更好。

随机森林算法是最好的分类算法之一,但其并没有考虑不平衡数据集的情况[10]。不平衡数据集是指在一个数据集中有些类的样本数量远小于其他类,一般称样本数量少的类为少数类,样本数量多的类为多数类。在非平衡数据集中,多数类与少数类的样本数相差非常大,使用传统的随机森林算法进行分类会降低其分类性能,即少数类的分类准确率很低而多数类的准确率则较高。文献[11]提出了一种不平衡随机森林变量选择方法,通过对各决策树所求得的特征重要性度量加权求和来获得最终的特征重要性序列,再根据此序列选择特征的数量由高至低进行特征选择。此算法得到了较好的分类效果,但特征权重的计算和权重排序需要花费大量的时间,运行时间较长从而导致运算效率降低。文献[12]对 SMOTE 算法[13]进行了改进,使用基于概率分布的过采样方法为少数类建立一些伪样本,使各类的样本数达到平衡。而文本分类的样本分类属性非常多,且各类别分布不规律,这会导致建立的伪样本根本不属于自己的类别,对分类性能有较大的影响。本文提出了一种改进型的不平衡数据随机森林算

到稿日期:2017-10-22 返修日期:2018-01-28
彭 徵(1992—),男,硕士生,主要研究方向为数据挖掘;王灵矫(1971—),男,硕士生导师,主要研究方向为宽带 IP 网络技术,E-mail:xtu_wlj@126.com(通信作者);郭 华(1976—),女,高级实验师,主要研究方向为电力系统自动化。

法(ID-RF),通过对训练样本的多数类进行欠取样,对少数类进行有放回取样,来减少不平衡数据对随机森林算法分类效果的影响。

2　相关技术

2.1　文本预处理

在文本分类的过程中,无论采用随机森林还是其他分类算法进行分类,都必须对文本进行预处理,将文本转化为适合分类处理的中间形式[14]。文本中的预处理包括文本分词、去除停用词、特征词提取、词频统计、文本的定量等。预处理文本即去除所有没有实际意义的词和一些对文本分类没有贡献的词,对文本分类算法的分类速度和精度有一定程度的改进。

2.2　特征权重的计算

在使用向量空间模型表示文本时,特征权重的计算是极其关键的环节,它决定了能否利用向量空间模型来准确表示文本。文本能否被正确表示将影响文本分类的性能,因为文本只有被正确表示才能产生正确的文本分类结果。

词频-逆向文本频率权重(TF-IDF)是常用的特征权重表示方式[15]。一个特征 i 在文本 D 中的词频表示为特征 i 在文本 D 中的词频权重(TF)。

$$TF(i)=\frac{f(i)}{p} \tag{1}$$

其中,$f(i)$表示特征 i 在文本 D 中的频率,p 表示所有特征在文本 D 中频率的总和。

逆向文本频率(IDF)是一种用于描述特征 i 在文本集 C 中普遍性的度量函数。

$$IDF(i)=\log(\frac{|C|}{N(i)}) \tag{2}$$

其中,$|C|$表示文本集的总文件,$N(i)$表示包含特征 i 的文本数量。

$$TF\text{-}IDF(i)=TF(i)*IDF(i) \tag{3}$$

2.3　决策树

决策树是一种类似于流程图的树结构,每个内部结点(非树叶结点)表示在一个属性上的测试,每个分支代表该测试的一个输出,而每个树叶结点(或终端结点)存放一个类标号。树的最顶层结点是根结点。决策树分为 ID3,C4.5 和 CART[16] 3 种算法,本文使用的是 ID3 算法。

决策树在创建树时需要对属性进行选择,ID3 算法使用信息增益作为属性选择度量。设结点 N 代表存放分区 D 的样本,选择具有最高信息增益的属性作为结点 N 的分裂属性,对 D 中的样本进行分类时所需要的期望信息如式(4)所示:

$$Info(D)=-\sum_{i=1}^{m}p_i\log_2(p_i) \tag{4}$$

其中,p_i 是 D 中任意样本属于类 C_i 的非零概率;$Info(D)$是识别 D 中样本的类标号所需要的平均信息量,又称为 D 的熵。

假设按某属性 A 划分 D 中的样本,其中属性 A 根据训练数据的观测具有 v 个不同值$\{a_1,a_2,\cdots,a_v\}$。可以用属性 A 将 D 划分为 v 个分区或子集$\{D_1,D_2,\cdots,D_v\}$,其中,D_j 包含 D 中的样本,它们的 A 值为 a_j。

$$Info_A(D)=\sum_{j=1}^{v}\frac{|D_j|}{|D|}\times Info(D_j) \tag{5}$$

其中,项$\frac{|D_j|}{|D|}$充当第 j 个分区的权重;$Info_A(D)$是基于按 A 划分对 D 的样本分类所需要的期望信息,需要的期望信息越小,分区的纯度越高。

信息增益定义为原来的信息需求与新的信息需求之间的差,即:

$$Gain(A)=Info(D)-Info_A(D) \tag{6}$$

选择具有高信息增益 $Gain(A)$的属性 A 作为结点 N 的分裂属性。

2.4　Spark 框架

Spark 是由 UC Berkeley 的 AMP Lab 开发的一种与 Hadoop 相似的分布式计算框架,支持基于内存的分布式数据集,为大数据交互式查询和迭代式计算提供了快速的计算环境。它支持 Scala,Java,Python 语言,提供支持 SQL 和结构化数据处理的 Spark SQL[17],以及用于机器学习的 MLib[18]、图形处理的 Graph X[19] 和流式计算的 Spark Streaming[20]。Spark 结构如图 1 所示。

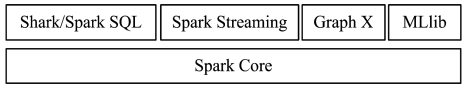


图 1　Spark 结构

Fig. 1　Structure of Spark

Spark 中的 RDD 是弹性分布式数据集[21],能够适应当前大部分数据并行算法和编程模型。使用 RDD 可以实现很多现有的集群编程模型以及一些以前不支持新应用的模型。RDD 能够在这些模型中取得与专有系统同样的性能,还能提供包括容错处理、滞后结点处理等这些专有系统所缺乏的特性。

Spark 生态系统兼容 Hadoop 生态系统。虽然 Spark 通用引擎在一定程度上是用来取代 MapReduce 系统的,但 Spark 能完美兼容 Hadoop 生态中的 HDFS 和 YARN 等组件,使现有 Hadoop 用户能够十分容易地迁移到 Spark 系统中。

Spark 支持 3 种集群管理员:Apache Mesos、Hadoop YARN 和独立集群管理,本文通过 Hadoop YARN 框架来管理 Spark 引用所需的资源[22]。若要将 Spark 集群运行在 YARN 模式下,首先需要部署一个 YARN 集群。Spark 集群由 Master 结点和 Worker 结点构成,用户程序通过与 Master 结点交互来申请所需的资源,Worker 结点负责 Executor 的启动运行。

3　基于随机森林的文本分类算法及其并行化

3.1　ID-RF 分类算法

随机森林分类算法的数据集默认为平衡数据,即各个类别的样本是相等的。然而,网络中文本的多样性导致不同文本类别的数量差距较大,因此文本分类的数据集属于不平衡数据。本文对随机森林算法进行了改进,使其适用于不平衡数据。

随机森林算法在建树的过程中,使用有放回抽样的方式从原始数据集中抽取相同数目的样本形成训练集来构建决策树。在训练集中有一些样本是重复的,从除去重复的样本训练集中抽取的样本大约占原始数据集样本总数的 63.2%。ID-RF 算法的步骤如下:

1)设原始训练数据集为 D,总样本数为 d,训练集中有 M 个属性,需要构建 k 棵决策树,分类个数为 c,每类的样本数为 C_i。以 d/c 为阈值,将样本数量大于 d/c 的类分为多数类,小于 d/c 的类分为少数类。

2)对各多数类样本采用欠取样方法获取 k 个样本数为 $0.632C_i$ 的多数样本集 $M_i(i=1,2,\cdots,k)$(如 $0.632C_i<d/c$,对训练样本继续有放回抽样,将采样样本数增加到 d/c)。

3)将少数类样本进行有放回取样,获得 k 个样本数为 d/c 的少数样本集 $S_i(i=1,2,\cdots,k)$。将少数样本集 S_i 与多数样本集 M_i 随机组合成 k 个平衡数据集 $D_i(i=1,2,\cdots,k)$。

4)构建随机森林模型。从 M 个属性中随机抽取 $F(F\ll M)$个属性作为分类属性集,然后采用 ID3 方法构建决策树,不限制每棵决策树的生长,且不进行剪枝。以此反复创建 k 棵决策树。

5)投票。随机森林分类器采取多数投票法进行决策,利用所构建的 k 棵决策树对某个数据进行分类,最后进行投票,将得票最多的类别作为随机森林的最终输出。

不平衡数据随机森林算法为了使少数类与多数类平衡,采用对少数类进行有放回取样、对多数类进行欠取样的方法。在对多数类进行取样时加入了一个最小阈值 63.2%,使得取样后的样本数与普通随机森林取样的不重复样本数相同,多数类训练样本不会出现失真,从而保持了多数类的正确率[23]。

3.2 基于 Spark 文本分类的并行化设计

整个文本分类过程中有两个部分可以进行并行化处理。文本预处理部分对每个文本进行预处理和特征选择、特征权重计算,各个文本均相互独立,可以对其进行并行化的处理。随机森林算法的建树过程、每棵决策树的构建过程与其他树相互之间没有影响,所有随机森林的建树过程也可以并行化。图 2 为随机森林算法的并行化文本分类流程,其中实线部分表示训练阶段,箭头虚线表示测试阶段,虚线方框表示并行部分。

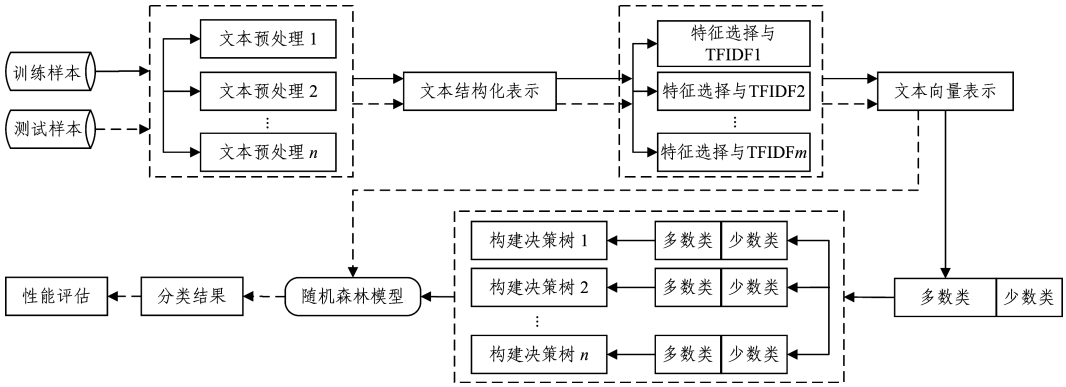


图 2 随机森林算法的并行化文本分类流程

Fig. 2 Parallel text categorization process of random forest

文本在 Spark 框架上进行预处理,通过并行方式减少运行的时间。文本的预处理流程如图 3 所示。

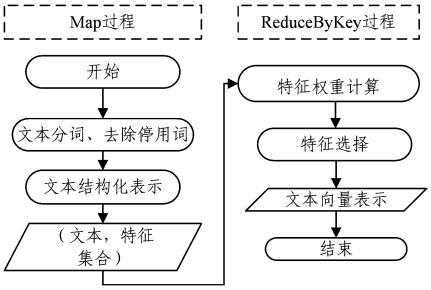


图 3 文本预处理流程

Fig. 3 Process of text pretreatment

根据图 3 所示的文本预处理过程,主要程序由 Driver 类、Mapper 类和 Reducer 类组成。Driver 类是程序的底层驱动,用于初始化程序的基础功能。Mapper 类主要是对文本进行词语划分、词干提取、去除停用词等操作,并把处理后的每个词定义为文本的索引词。每个索引词作为文本的一个特征,最后将所有特征组成集合。Reducer 类则计算文本各特征的权重,用卡方统计的文本降维方法进行特征选择,使文本变成向量的形式。

传统的随机森林算法在单机上实现大规模数据处理时,往往需要花费大量的时间,效率较低。采用嵌入 Spark 框架的随机森林并行化算法可以有效地减少建树的时间并提高算法的效率。图 4 为基于 Spark 的 ID-RF 算法的并行化流程图。

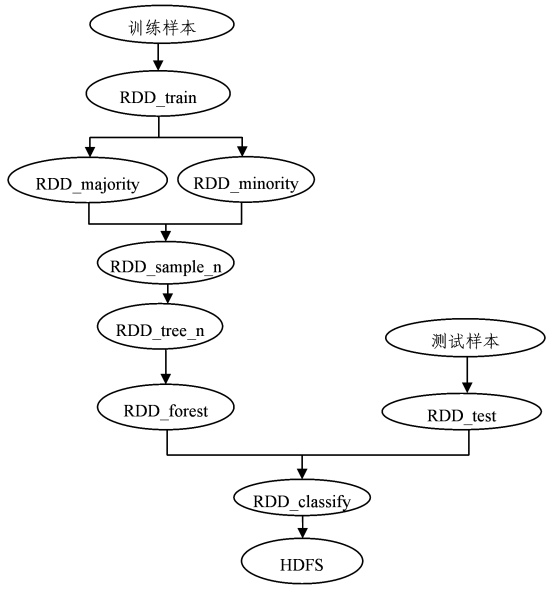


图 4 基于 Spark 的 ID-RF 算法的并行化流程

Fig. 4 Parallel ID-RF process based on Spark

ID-RF 算法在 Spark 平台上的并行化过程主要分为以下步骤:

1)将经过文本预处理的训练样本存入 RDD_train,把 RDD_train 中的训练样本分为多数类和少数类,并分别存入 RDD_majority 和 RDD_minority 中。

2)对 RDD_majority 和 RDD_minority 中的样本分别进行欠取样和放回抽样,并把结果分别存入 RDD_majority_n 和 RDD_minority_n 中。然后将 RDD_majority_n 和 RDD_minority_n 中的数据进行随机组合,并将组合好的结果存入 RDD_sample_n。

3)根据每一个 RDD_sample_n 中的数据运行决策树算法,将运行得到的决策树模型存入 RDD_tree_n,利用所有的决策树构建随机森林,并将随机森林模型存入 RDD_forest。

4)将测试样本存入 RDD_test,测试数据通过随机森林模型得到最终的分类结果,并将结果存入 RDD_classify。

5)对 RDD_classify 中的数据进行分析,并将分析结果以文档的形式存入 HDFS。

4　实验及结果

本文的实验在 3 台 PC 组成的 park 集群中完成,其中包括 1 台主结点,2 台从结点。每个结点的配置为 3.3 GHz 的 Intel 处理器,4 GB 内存,CentOs 操作系统。实验使用 Hadoop 2.4.0 的 YARN 框架来进行 Spark 资源调度,通过 Scala 2.12.1 编写算法。

4.1　分类性能

本次实验的数据源选自某网站近两个月的文本样本集,样本共分为读物、娱乐、历史、教育、社会、文化、IT、军事八大类。表 1 列出了样本中各个类别的数量。

表 1　样本数量分布

Table 1　Distribution of number of samples

类别	样本数量
读物	12000
娱乐	4332
历史	2962
教育	876
社会	5298
文化	3746
IT	1198
军事	3896

由表 1 可以看出,样本中各类的数量分布得严重不均匀,部分类的数量远小于其他类的数量。分别使用传统的随机森林算法和改进后的 ID-RF 算法对数据源进行分类,得到的结果如图 5 所示。

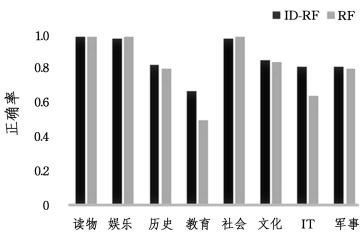


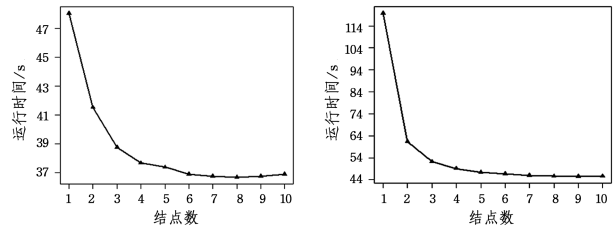
图 5　随机森林算法正确率对比

Fig. 5　Comparison of correct rate of random forest algorithms

从图 5 可以看出,传统的随机森林算法虽然总体的正确率较高,但是对于一些少数类的正确率却很低,如图 5 中的教育与 IT 这两类的正确率明显低于其他类。而改进后的 ID-RF 算法将教育类的正确率从 50%提升到 67%,将 IT 类的正确率从 64%提升到 81%,并且对其他类的正确率没有影响,这足以证明 ID-RF 算法在分类性能和平衡方面比普通随机森林算法更好。

4.2　运行时间

本文选用了两组样本集进行实验,样本集 A 含有 1 万多个样本,样本集 B 含有 6 万多个样本。依次将 Spark 结点从 1 增加到 10,记录两个样本集的运行时间。实验结果如图 6 所示。



(a)样本集 A　　(b)样本集 B

图 6　运行时间

Fig. 6　Running time

图 6 中,当结点数从 1 增加到 10 时,ID-RF 算法的平均运行时间明显缩短,图 6(a)从原来的 48 s 缩短为 36 s,图 6(b)从 119 s 缩短为 46 s。因为 Spark 的结点数越多,运行 ID-RF 算法的进程就越多,运行时间随之缩短。若结点数增加,结点间的数据通信也会增加,运行时间的缩短量会越来越小。当结点数增加到 9 和 10 时运行时间基本没有变化,甚至有增加的趋势。

由图 6 可以发现,普通的单机运算就是图中结点数为 1 的情况,并行化运算可以根据训练样本选择最佳的结点数。从图中可以看出,结点数为 1 的运行时间比其他结点数的运行时间长得多。因此,在 Spark 平台上并行化运行 ID-RF 算法的效率比在单机上运行的高。

4.3　加速比

加速比主要用于衡量算法在 Spark 平台上的并行化性能和效果,其数值为结点数为 1 的运行时间与结点数为 n(n=1,2,…,10)的运行时间之比。加速比的理想图是一条斜率为 1 的直线。根据样本集 A 和样本集 B 在不同结点数时的运行时间可以得到两个样本集的加速比,如图 7 所示。

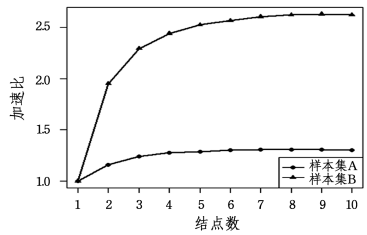


图 7　加速比

Fig. 7　Speedup ratio

由于结点间通信的影响,图 7 中两样本加速比的增长趋势随着结点数量的增加而减弱。样本集 B 的加速比明显大于样本集 A 的加速比,可以看出样本集 B 的并行化效果更好。

样本的数据量越大，算法运行的时间就越长，而结点间的通信时间占总运行时间的比例就越小，对加速比的影响也越小。

结束语 本文提出一种基于文本分类的随机森林改进算法——ID-RF 算法，采用对训练样本的多数类进行欠取样、对少数类进行有放回取样的方式使各样本数达到平衡。该算法适用于非平衡的文本分类数据源，使少数类的正确率得到很大的提升。本文将 ID-RF 算法运行在 Spark 环境下，使算法在多个结点上并行化运行，以缩短算法的运行时间。实验结果表明，ID-RF 算法的平衡性比普通的随机森林算法更好，在 Spark 平台上运行 ID-RF 算法的时间远短于在单机上运行的时间。

接下来将进一步改进随机森林算法，通过对特征选择、投票权重等方面的改进来提升算法的分类效果。此外，进一步完善程序使整个文本分类过程在 Spark 平台上运行得更加高效。

参 考 文 献

[1] YIN C Y, XI J W. The Research of Text Classification Technology Based on Improved Maximum Entropy Model[C]// International Conference on Computational Intelligence Theory, Systems and Applications. 2015: 142-145.

[2] LIU J, JIN T, PAN K J. An Improved KNN Text Classification Algorithmbased on Simhash[C]// International Conference on Cognitive Informatics & Cognitive Computing. 2017: 92-95.

[3] SHARMA N, SINGH M. Modifying Naive Bayes Classifier for Multinomial Text Classification[C]// International Conference on Recent Advances and Innovations in Engineering. 2016: 1-7.

[4] WANG X L, WANG J, YANG Y. Labeled LDA-Kernel SVM: A Short Chinese Text Supervised Classification Based on SinaWeibo[C]// International Conference on Information Science and Control Engineering. 2017: 429-432.

[5] BĂDULESCU L A. Data Mining Classification Experiments with Decision Trees over the Forest Covertype Database[C]// International Conference on System Theory, Control and Computing. 2017: 236-241.

[6] HE J. Random Forest in Application of Text Classification[D]. Guangzhou: South China University of Technology, 2015. (in Chinese)
贺捷. 随机森林在文本分类中的应用[D]. 广州：华南理工大学，2015.

[7] BECHINI A, MATTEIS A D D. Spreading Fuzzy Random Forests with MapReduce[C]// IEEE International Conference on Systems, Man, and Cybernetics. 2017.

[8] XIANG X J, GAO Y, SHANG L. Parallel Text Categorization of Massive Text based on Hadoop[J]. Computer Science, 2011, 38(10): 184-188. (in Chinese)
向小军，高阳，商琳. 基于 Hadoop 平台的海量文本分类的并行化[J]. 计算机科学，2011，38(10)：184-188.

[9] YAN J M. The Research and Application of Text Classification Based on Cloud Computing[D]. Hangzhou: Zhejiang Sci-Tech University, 2016. (in Chinese)
严嘉铭. 基于云计算的文本分类研究与应用[D]. 杭州：浙江理工大学，2016.

[10] MORE A S, RANA D P. Review of Random Forest Classification Techniques to Resolve Data Imbalance[C]// International Conference on Intelligent Systems and Information Management. 2017: 72-78.

[11] YIN H, HU Y P. An Imbalanced Feature Selection Algorithm Based on Random Forest[J]. Journal of Sun Yat-sen Universyt, 2014, 5(9): 59-65. (in Chinese)
尹华，胡玉平. 基于随机森林的不平衡特征选择算法[J]. 中山大学学报，2014，5(9)：59-65.

[12] YU H L, GAO S, ZHAO J. Classification for Imbalanced Microarray Data Based on Oversampling Technology and Random Forest[J]. Computer Science, 2012, 39(5): 190-194. (in Chinese)
于化龙，高尚，赵靖. 基于过采样技术和随机森林的不平衡微阵列数据分类方法研究[J]. 计算机科学，2012，39(5)：190-194.

[13] CHAWLA N V, BOWYER K W, HALL L O, et al. SMOTE: Synthetic minority over-sampling technique[J]. Journal of Artificial Intelligence Research, 2002, 16: 321-357.

[14] YIN C Y, XI J W, WANG J. The Research of Text Classification Technology Based on Improved Maximum Entropy Model[C]// International Conference on Computational Intelligence Theory, Systems and Applications. 2015: 142-145.

[15] GUO A Z, YANG T. Research and Improvement of feature words weight based on TFIDF Algorithm[C]// Information Technology, Networking, Electronic and Automation Control Conference. 2016: 415-149.

[16] EL HABIB DAHO M, SETTOUTI N, EL AMINE LAZOUNI M. Weighted Vote for Trees Aggregation in Random Forest[C]// International Conference on Multimedia Computing and Systems. 2014: 428-443.

[17] CUI Y, LI G Q, CHENG H. Indexing for Large Scale Data Querying based on Spark SQL[C]// International Conference on e-Business Engineering. 2017: 103-108.

[18] AKGÜN B, ÖĈÜDÜCÜ Ş G. Streaming Linear Regression on Spark MLlib and MOA[C]// International Conference on Advances in Social Networks Analysis and Mining. 2015: 1244-1247.

[19] GOMBOS G, KISS A. P-Spar(k)ql: SPARQL Evaluation Method on Spark GraphX with Parallel Query Plan[C]// International Conference on Future Internet of Things and Cloud. 2017: 212-219.

[20] PERROT A, BOURQUI R, HANUSSE N. HeatPipe: High Throughput, Low Latency Big Data Heatmap with Spark Streaming[C]// International Conference Information Visualisation. 2017: 66-71.

[21] 夏俊鸾. Spark 大数据处理技术[M]. 北京：电子工业出版社，2015.

[22] LI H, LI Z, SHE K. An Improvement of Random Forest Algorithm Based on Comprehensive Sampling without Replacement [J]. Computer Engineering & Science, 2015, 7(37): 1233-1238. (in Chinese)
李慧，李正，余堃. 一种基于综合不放回抽样的随机森林算法改进[J]. 计算机工程与科学，2015，7(37)：1233-1238.