孤立森林算法研究及并行化实现

王 诚,狄 萱
(南京邮电大学 通信与信息工程学院,江苏 南京 210003)

摘 要:异常检测是近年来数据挖掘中热门的研究课题之一,孤立森林算法是一种高效的无监督的异常检测算法,可以很好地处理高维大规模数据。针对孤立森林算法在计算测试样本的异常值时,计算的是测试样本在孤立森林下的平均路径长度,忽略了孤立二叉树间检测异常能力的差异性以及大规模数据下构建大量孤立二叉树需要耗费大量内存时间这两点不足,提出一种并行化改进孤立森林算法。利用每棵孤立二叉树的路径长度标准差对其进行加权计算异常值,并基于 Spark 平台实现并行化。通过在公开数据集上进行的对比实验及多种参数配置的并行性能对比实验表明,并行化改进孤立森林算法能够提高异常检测的精确度,同时具有很好的并行性能,能够高效处理需要构建大量孤立二叉树的大规模数据集。

关键词:异常检测;孤立森林算法;孤立二叉树;Spark;并行化
中图分类号:TP301.6 **文献标识码**:A **文章编号**:1673-629X(2021)06-0013-06
doi:10.3969/j.issn.1673-629X.2021.06.003

Research and Parallelization of Isolation Forest Algorithm

WANG Cheng, DI Xuan
(School of Telecommunications & Information Engineering, Nanjing University of Posts and Telecommunications, Nanjing 210003, China)

Abstract: Anomaly detection is one of the hot research topics in data mining in recent years. Isolation Forest algorithm is an efficient unsupervised anomaly detection algorithm that can handle high-dimensional large-scale data well. When Isolation Forest algorithm calculates the outliers of test samples, it calculates the average path length of test samples in Isolation Forest, ignoring the difference in the ability to detect abnormalities between isolation trees and the large amount of memory and time needed to construct a larger number of isolation trees under large-scale data. For these two deficiencies, an improved parallelized Isolation Forest algorithm is proposed. The standard deviation of the path length of each isolation tree is used to weight the outliers, and the parallelization is implemented based on the Spark platform. The comparison experiments on public datasets and parallel performance comparison experiments with multiple parameter configurations show that the proposed algorithm can improve the accuracy of anomaly detection with excellent parallel performance, and can effectively deal with large-scale data sets that need to build a large number of isolation trees.

Key words: anomaly detection; Isolation Forest algorithm; isolation tree; Spark; parallelization

0 引 言

异常检测是近年来数据挖掘中热门的研究课题之一,被广泛应用于医保欺诈[1]、网络入侵[2]、医疗诊断[3]等领域。随着物联网、云计算等技术的不断发展,数据量日渐增长,传统单机版异常检测算法难以对大规模数据进行高效检测。因此,针对大规模数据设计相应算法,具有重要的应用价值。近年来,已有的异常检测算法分为三类,分别是基于统计[4]、密度[5]和聚类[6]的算法。Liu 等[7-8]提出的孤立森林算法,通过计算测试数据在已构建的孤立森林模型下的平均路径长度代入公式求其异常值,该算法大大减少了计算时间,适用于高维数据。Liao 等[9]提出一种基于信息熵的改进孤立森林算法,该算法通过计算数据集中各个特征下的信息熵,优先选择信息熵小的特征作为切割特征,并且改进了路径长度的计算公式,对噪声特征具有较强的抵抗性。Yang 等[10]提出了一种基于孤立的嵌入式无监督特征选择(IBFS)算法,该算法通过计算孤立森林在训练过程中每个特征的得分,选出表现优异

收稿日期:2020-06-19 **修回日期**:2020-10-20
基金项目:江苏省自然科学基金项目(BK20141428)
作者简介:王 诚(1970-),男,副教授,硕导,研究方向为互联网大数据挖掘;狄 萱(1996-),女,硕士研究生,CCF 会员(C2232G),研究方向为数据挖掘。

的特征集合进行异常检测,提高了异常检测精度。Yong 等[11]根据 Hadoop 平台业调度机制和孤立森林算法的思想,将孤立森林算法的模型构建和异常预测两个过程进行了并行化设计,但其运算过程中需要多个 MapReduce 操作完成并行运算,多次读写硬盘,耗费大量时间。

孤立森林算法异常检测不需要计算距离和密度,避免了大量的计算,因此能够更好地支持高维数据的异常检测,同时孤立森林的每棵孤立二叉树的构建过程都是独立的,能够利用分布式平台对孤立森林算法进行并行化设计。孤立森林算法存在一些不足之处:

(1)在深入研究孤立森林算法过程中发现,孤立森林算法在计算测试样本的异常值时,计算的是测试样本在孤立森林的平均路径长度,而孤立森林算法的核心思想是:在一棵孤立二叉树中,若某个叶子节点的路径长度短,则认为该节点是异常点。当某棵孤立二叉树没有相对短的路径的叶子节点时,则说明其难以区分异常点。

(2)Yong 等[11]指出,孤立森林算法异常检测的精度与孤立二叉树的数量有关,随着数据量的增多,所需构建孤立二叉树的数量也相应增多,从而导致耗费大量内存和时间,影响算法效率。

针对第一点不足,对孤立森林中每棵孤立二叉树的路径长度的标准差进行归一化加权,计算异常值。若标准差大,则说明该棵孤立二叉树中各叶子节点间的路径长度差异大,具有较好的异常检测能力,应赋予较高的权值,否则赋予较低的权值。通过加权计算测试样本在孤立森林的异常值,以提高异常检测的精确度。针对第二点不足,利用 Spark 平台实现改进算法的并行化。Spark 平台是基于内存设计的,避免了 Hadoop 平台 MapReduce 操作需要频繁读写磁盘,能够加快整体的异常检测速度。

1 相关工作

1.1 孤立森林算法

孤立森林是由多棵孤立二叉树组成的,孤立二叉树的构建过程是算法的核心,孤立二叉树的构建过程如下:

(1)从数据集 D 中随机选择 m 个样本点作为生成本棵孤立二叉树的样本集 D_s。

(2)从样本集 D_s 中随机选择一个特征 f 和一个切割值 p。若节点 N 包含的所有样本在特征 f 下的最大值和最小值分别为 $f\_\max$ 和 $f\_\min$,则有 $p \in [f\_\min, f\_\max]$。

(3)若样本的特征 f 的值小于切割值 p,则将该样本分到节点 N 的左孩子;否则,分到右孩子。

(4)重复(2)、(3)两步,分别对节点 N 的左右孩子节点进行切分,生成孤立二叉树。当孩子节点中有多条相同的数据或只有一条数据或孤立二叉树已达到设置的最大高度时,停止生成孤立二叉树。

(5)孤立森林最终由用户指定数目的孤立二叉树组成。根据样本点 d 在每棵孤立二叉树中的路径长度 $h(d)$,利用公式(1)计算 d 的异常值,从而评价其异常情况。

$$s(d,m) = 2^{\frac{-E(h(d))}{c(m)}} \tag{1}$$

其中,m 为样本集 D_s 的样本点总数,$E(h(d))$ 为所有路径长度 $h(d)$ 的平均值,$c(m) = 2H(m-1) - 2(\frac{m-1}{m})$ 是树的高度额归一化,$H(k) = \ln k + \xi$,$\xi = 0.577\ 215\ 664\ 9$ 为欧拉常数。

1.2 Spark 大数据计算框架

Spark[12-13]是加州大学伯克利分校的 AMP 实验室开发的一款基于内存的并行分布式计算框架。相较于 Hadoop 框架中的分布式计算模块 MapReduce 需要频繁读写磁盘 I/O 操作,Spark 框架基于内存的设计,将每一轮的输出结果都缓存在内存中,避免了从 HDFS 中读取数据,更适合需要多次迭代的算法。Spark 的运行架构模型如图 1 所示,其基本运行流程是:

(1)Spark 在驱动节点 Driver 端运行 main( )方法并创建 SparkContext 以构建 Spark Application 的运行环境,创建完成后的 SparkContext 向资源管理器 Cluster Manager 申请所需要的 CPU、内存等资源并申请运行多个执行节点 Executor,Cluster Manager 分配并启动各个 Executor。

(2)SparkContext 构建有向无环图 DAG 以反映弹性分布式数据集 RDD 之间的依赖关系,并将其分解成任务集 TaskSet 发送给有向无环图调度器 TaskScheduler。

(3)各 Executor 向 SparkContext 申请 Task,TaskScheduler 将 Task 分发给各个 Executor,同时 SparkContext 将打好的程序 Jar 包发送给各个 Executor,各 Executor 执行结束后将结果收集给 Driver 端[14],最终释放资源。

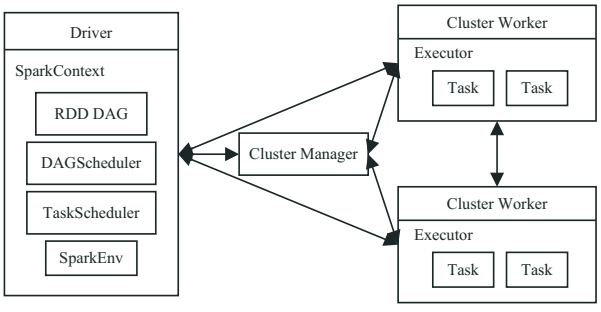


图 1 Spark 运行架构模型

2　改进孤立森林算法的并行化实现

2.1　加权孤立森林的构建算法

如公式(1)所示,孤立森林在计算测试样本异常值时,计算的是测试样本在孤立森林的平均路径长度,忽略了各孤立二叉树的异常检测能力的差异性,即每个叶子节点的路径都很长的孤立二叉树难以区分异常点,而具有短路径的叶子节点的孤立二叉树能够更好地区分异常点。如图 2 和图 3 所示,图 2 的孤立二叉树比图 3 的孤立二叉树具有更强的区分异常点的能力。因此该文通过计算每棵孤立二叉树的路径长度标准差对具有更强区分异常点能力的树进行加权,同时减小区分异常点能力较差的树的权值。

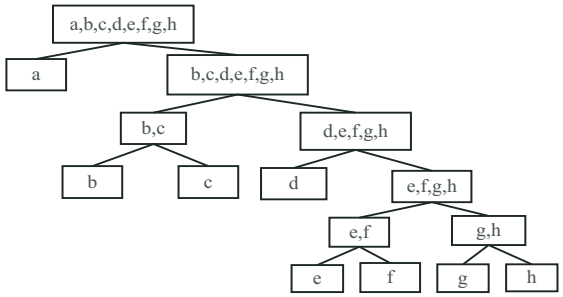


图 2　区分异常能力强的孤立二叉树

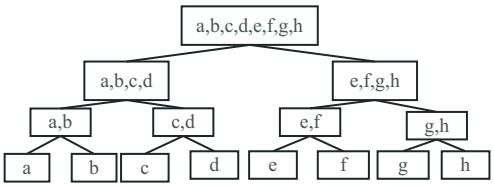


图 3　区分异常能力差的孤立二叉树

若一棵孤立二叉树的叶子节点集合为 Node = $\{node_1, node_2, \cdots, node_n\}$,叶子节点的路径长度集合为 $H_{node} = \{h_1, h_2, \cdots, h_n\}$,则该棵树的路径长度标准差 σ 的计算公式为:

$$\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(h_i - \mu)^2} \tag{2}$$

其中,n 为叶子节点的总个数,μ 为该棵树所有叶子节点路径长度的均值。

若孤立森林中所有孤立二叉树的路径长度标准差集合为 $\sigma = \{\sigma_1, \sigma_2, \cdots, \sigma_n\}$,其中最大值为 σ_{max},最小值为 σ_{min},对路径长度标准差集合进行归一化,公式为:

$$w_i = \frac{\sigma_i - \sigma_{min}}{\sigma_{max} - \sigma_{min}} \tag{3}$$

若数据集 D 中每个样本点 d 在每棵孤立二叉树中的路径长度为集合 $H_d = \{h_1, h_2, \cdots, h_n\}$,孤立二叉树的权重集 $W = \{w_1, w_2, \cdots, w_n\}$,则样本点 d 的异常值计算公式为:

$$s(d,m) = 2^{\frac{-\sum_{i=1}^{n} w_i \times h_i}{c(m)}} \tag{4}$$

具体算法实现如下:

算法 1:加权孤立森林的构建算法。

输入:数据集 D,孤立二叉树数量 n,随机采样样本数量 m,树的高度限制 l(通常设置为 $\log_2^m$);

输出:孤立森林,权重集 W。

(1)对数据集 D 进行随机采样,抽取 m 个数据放入集合 D_s 中。

(2)调用孤立二叉树生成算法,并传入相关参数,生成单棵孤立二叉树。

(3)利用公式(2)计算当前生成的孤立二叉树的路径长度标准差。

(4)重复 1、2 两步,直到生成 n 棵孤立二叉树及路径长度标准差集合。

(5)利用公式(3)对路径长度标准差集合进行归一化,生成权重集 W。

(6)返回由 n 棵孤立二叉树组成的孤立森林及权重集 W。

算法 2:样本点异常值计算流程。

输入:数据集 D,权重集 W;

输出:数据集 D 的异常值集 N。

(1)计算数据集 D 中每个样本点在每棵孤立二叉树中的路径长度集合 H_d。

(2)利用公式(4),加权计算每个样本点的异常值,并合为异常值集 N。

(3)返回数据集 D 的异常值集 N。

2.2　改进孤立森林算法的并行化设计

该文利用 Spark 平台实现改进孤立森林算法的并行化。主要步骤包括:数据抽样、模型构建和异常预测。整体框架如图 4 所示。

(1)数据抽样。

单机版的孤立森林算法需要对原始数据集进行随机抽样,利用抽样后的数据集生成孤立二叉树。而 Spark 平台数据存储核心 RDD 是分布式数据集,如果直接对各个分区的数据进行定量随机抽样,不能保证抽样后得到的数据集是全局随机的。虽然 Spark 提供了 sample 抽样算子,但会导致非确定性大小的采样样本集。因此,该文设计从 HDFS 中读取数据文件,转化成 RDD 记为 RDD_data,在 Driver 端首先利用 count 算子计算 RDD_data 数据总量,创建 RandomData-Generator 类,根据 RDD_data 数据总量、孤立二叉树数量、随机采样样本数量,随机生成包含行号索引值的二维数组,并将其映射为(rowId(行号),treeIdArray(该行数据对应的孤立二叉树 ID 集合))的格式,最后将该形式的变量广播到各个 Executor 端,以减少 Shuffle 成本。利用 zipWithIndex 算子,为 RDD_data 添加全局索引号,在各个 Executor 端利用广播来的变量中的行

号对 RDD_data 数据内容筛选过滤,并通过 flatMap、reduceByKey 算子生成(treeId(孤立二叉树 ID),ContentArray(构建此 ID 孤立二叉树所需的数据集))格式的数据集:RDD_sample。

(2)模型构建。

将 RDD_sample 数据集 map 到各个 Executor 端进行孤立二叉树的并行同步构建并计算各个孤立二叉树的路径长度标准差,数据格式分别为:(treeId(孤立二叉树 ID),tree(孤立二叉树))、(treeId(孤立二叉树 ID),stdPath(路径长度标准差))。利用 collect 算子将多棵孤立二叉树及各自的路径长度标准差收集到 Driver 端,并将多棵树的路径长度标准差合并归一化成一个数据格式为(treeId(孤立二叉树 ID),weight(权重))的权重集。最后将构建好的模型及权重集分别存入 RDD_model 和 RDD_weight 中。

(3)异常预测。

将构建好的模型 RDD_model 广播给各个 Executor,测试数据集并行遍历每一棵孤立二叉树,得到每一个测试数据样本在模型下的路径长度集合,集合中元素的数据格式为(treeId(孤立二叉树 ID),pathLength(路径长度)),利用权重集 RDD_weight 对每一个测试数据样本的路径长度集合进行加权计算,得到每一个测试数据样本的异常值,从而评价其异常情况。

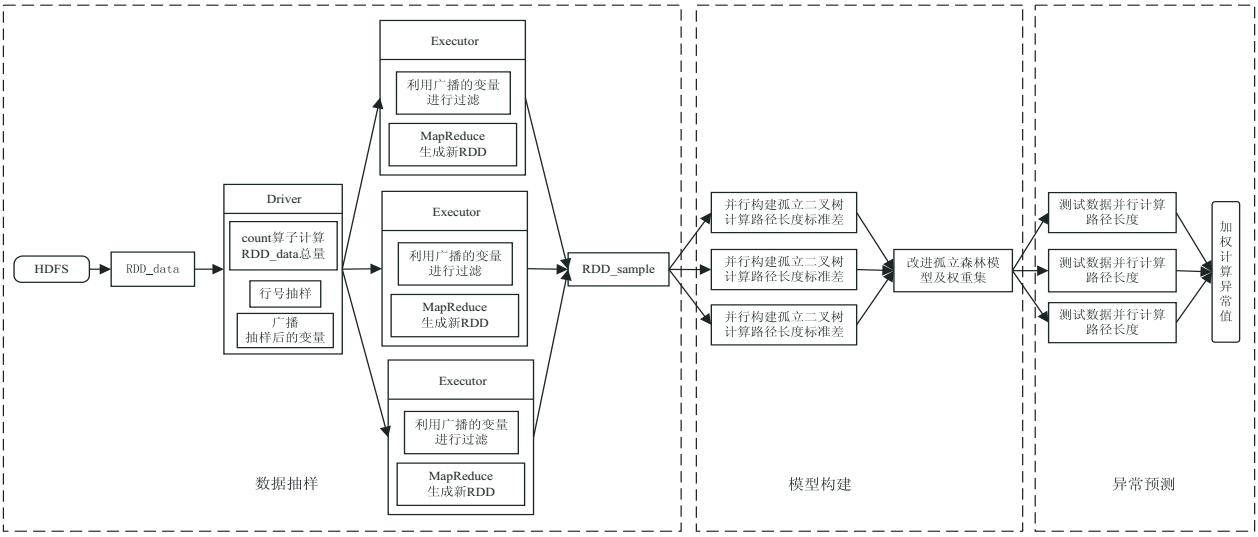


图 4 改进孤立森林算法并行化框架

3 实验设计与结果分析

3.1 实验数据集

实验数据集选取自威斯康星州数据集 Breastw、UCI 数据集 Shuttle 和 KDD CUP 99 数据集 Http[15],数据集的具体信息如表 1 所示。

表 1 数据集具体信息

数据集	样本数	特征数	异常点数
Breastw	683	9	239
Shuttle	49 097	9	2 270
Http	567 497	3	3 437

3.2 异常检测精确度

该文使用表 1 中的三个数据集对三种算法进行实验对比分析。其中,三个算法的参数均为:孤立二叉树数量 $n = 100$,随机采样样本数量 $m = 256$,树的高度限制 $l = 8$。采用 ROC 曲线下的面积 AUC 指标来评价算法异常检测精确度。AUC 值的范围为[0,1],其值越接近 1 则说明算法的检测效果越好。具体的实验结果如表 2 所示。

表 2 三种算法的 AUC 值

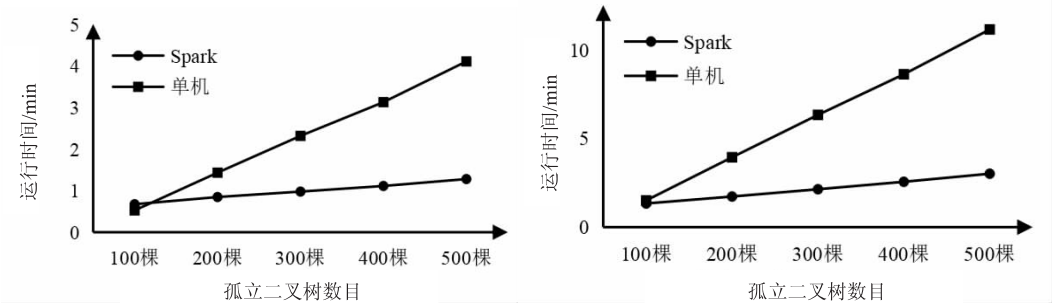
数据集	并行化改进孤立森林		改进孤立森林		原始孤立森林	
	AUC	运行时间/s	AUC	运行时间/s	AUC	运行时间/s
Breastw	0.962 1	16.74	0.961 0	0.10	0.938 8	0.10
Shuttle	0.966 2	19.79	0.969 4	5.43	0.944 3	5.27
Http	0.939 0	30.82	0.938 4	17.84	0.921 9	17.28

从表 2 中可以看出,对于 Breastw、Shuttle 和 Http 三个数据集,改进孤立森林算法相对于原始孤立森林算法的 AUC 值分别提高了 2.22%、2.51%、1.65%,这是因为改进孤立森林算法改进了异常值计算公式,为高异常检测能力的孤立二叉树赋予更高的权值,从而让异常点更为突出。并行化改进孤立森林算法与改进孤立森林算法的 AUC 值没有明显差异,因此并行化改进孤立森林算法在异常检测精度上能与改进孤立森林算法保持一致。同时,改进孤立森林算法由于需要计算路径长度标准差并归一化为权重集,相对于原始孤立森林算法增加了些许时间开销,但在小规模数据集上可以忽略不计。并行化改进孤立森林算法相较于单机的改进孤立森林算法耗费了更多的时间,这是由于数据规模小时,集群初始化、任务调度及节点间的数据通信时间远大于算法本身的运算时间。

3.3　并行性能实验

实验环境是基于 Spark 平台的计算集群,该集群共有 4 个节点,每个节点的 CPU 核数为 1 核,内存为 2 G,硬盘为 30 G,Java 版本为 1.8.0,Scala 版本为 2.11.0,Hadoop 版本为 2.7.6,Spark 版本为 2.4.0。实验数据集是由 Breastw 数据集构造的行数为 100 万行、300 万行、500 万行、800 万行的大规模数据集。分别对这四个大规模数据集进行对比实验,其中自变量为孤立二叉树数目,分别为 100 棵、200 棵、300 棵、400 棵、500 棵。同时计算当孤立二叉树数量为 100 时,改进孤立森林算法在 Spark 集群下的加速比,评价其并行性能。实验结果如图 5 ~ 图 6 所示。

从图 5 中可以看出,在不同数据规模下,随着孤立二叉树数目的不断增加,单机和并行化的孤立森林算法的运行时间都呈线性增加,单机算法的增幅更陡峭,并行算法的增幅平缓。当数据量达到 800 万行、孤立二叉树数目为 500 棵时,单机运行时间是并行化后的运行时间的 4.34 倍。从图 6 中可以看出并行化改进孤立森林算法总体上有很好的加速比。随着数据量的不断增大,加速比随着节点数的增加而明显增大,当数据量达到 800 万行、节点数为 4 时,加速比提升到了 2.88。因此,并行化改进孤立森林算法能够更高效地处理需要构建大量孤立二叉树的大规模数据集。



(a)100 万行数据集　　(b)300 万行数据集

(c)500 万行数据集　　(d)800 万行数据集

图 5　不同规模数据集下运行时间对比

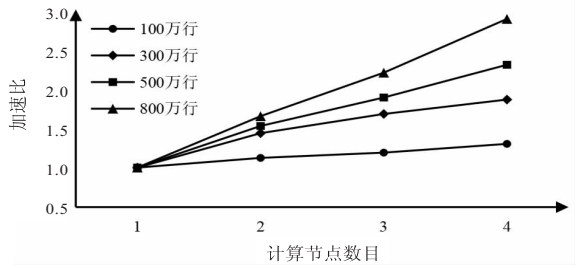


图 6　100 棵孤立二叉树下的加速比

4　结束语

该文针对孤立森林算法在计算测试样本的异常值时,忽略了孤立二叉树间检测异常能力差异性以及大规模数据下构建大量孤立二叉树需要耗费大量内存时间这两点不足进行改进,加权计算测试样本在孤立森林中的异常值并基于 Spark 平台设计实现并行化改进孤立森林算法。通过多个对比实验结果表明,并行化

改进孤立森林算法能够加快大规模数据集下的异常检测速度,同时提高了异常检测精度。下一步将把该算法与实际应用场景相结合,检验其实际应用价值。

参考文献:

[1] RAWTE V,ANURADHA G. Fraud detection in health insurance using data mining techniques[C]//2015 international conference on communication, information & computing technology (ICCICT). Mumbai:IEEE,2015:1-5.

[2] 欧阳丽. 基于优化后的随机森林算法的入侵检测技术研究[D]. 长沙:湖南大学,2018.

[3] LING C X,SHENG V S,YANG Q. Test strategies for cost-sensitive decision trees[J]. IEEE Transactions on Knowledge and Data Engineering,2006,18(8):1055-1067.

[4] BI J,FENG T,YUAN H. Real-time and short-term anomaly detection for GWAC light curves[J]. Computers in Industry,2018,97:76-84.

[5] TANG B,HE H. A local density-based approach for outlier detection[J]. Neurocomputing,2017,241(7):171-180.

[6] ZHU Li,QIU Yuanyuan,YU Shuai,et al. A fast KNN-based MST outlier detection method[J]. Chinese Journal of Computer,2017,40(12):224-238.

[7] LIU F T,TING K M,ZHOU Z H. Isolation forest[C]// 2008 eighth IEEE international conference on data mining. Italy:IEEE,2008:413-422.

[8] LIU F T,TING K M,ZHOU Z H. Isolation-based anomaly detection[J]. ACM Transactions on Knowledge Discovery from Data,2012,6(1):3.1-3.39.

[9] LIAO L,LUO B. Entropy isolation forest based on dimension entropy for anomaly detection[C]//Computational intelligence and intelligent systems. Jiujiang, China: Springer, 2019:365-376.

[10] YANG Q,SINGH J,LEE J. Isolation-based feature selection for unsupervised outlier detection[C]//Annual conference of the prognostics and health management society. USA: PHM,2019:824-831.

[11] 侯泳旭,段 磊,秦江龙,等. 基于 Isolation Forest 的并行化异常探测设计[J]. 计算机工程与科学,2017,39(2):236-244.

[12] ZAHARIA M,CHOWDHURY M,FRANKLIN M J,et al. Spark:cluster computing with working sets[C]//IEEE international conference on cloud computing technology and science. Boston,MA:USENIX Association,2010:10.

[13] 宋董飞,徐 华. DBSCAN 算法研究及并行化实现[J]. 计算机工程与应用,2018,54(24):52-56.

[14] 朱子龙,李玲娟. 基于 Spark 的密度聚类算法并行化研究[J]. 计算机技术与发展,2018,28(6):80-84.

[15] YAMANISHI K,TAKEUCHI J I,WILLIAMS G,et al. On-line unsupervised outlier detection using finite mixtures with discounting learning algorithms[J]. Data Mining and Knowledge Discovery,2004,8(3):275-300.

关于本刊变更主管单位的公告

经国家新闻出版署批准(国新出审〔2021〕715 号),《计算机技术与发展》期刊主管单位由陕西省工业和信息化厅变更为陕西省科学技术协会。其他登记事项不变。

特此公告。

《计算机技术与发展》编辑部

2021 年 6 月 7 日