基于网格划分加权的分布式离群点检测算法

梅 林[1,2]，张凤荔[1*]，王瑞锦[1]，高 强[1]

(1. 电子科技大学网络与数据安全四川省重点实验室 成都 610054；2. 西南民族大学计算机科学与技术学院 成都 610225)

【摘要】分布式计算被广泛应用于离群点检测问题，但分布式环境中节点计算性能的差异带来了数据计算性能的下降问题。针对面向大尺度高维数据离群点分布式计算的负载均衡问题，该文提出了一种加权分布式离群点检测方法。首先根据数据节点的计算性能确定数据节点的权值，然后将数据空间划分为若干个网格，最后设计了一种基于网格划分的加权分配算法 WGBA，将这些网格分配到数据节点中，实现并行计算。实验验证了该方法的有效性。

关 键 词 基于密度的离群点检测； 分布式算法； 网格划分； 局部异常值因子

中图分类号 TP393 **文献标志码** A doi:10.12178/1001-0548.2020202

A Weighted Distributed Outlier Detection Algorithm Based on Grid Partition

MEI Lin[1,2], ZHANG Feng-li[1*], WANG Rui-jin[1], and GAO Qiang[1]

(1. Network and Data Security Key Laboratory of Sichuan Province, University of Electronic Science and Technology of China Chengdu 610054;
2. School of Computer Science and Technology, Southwest Minzu University Chengdu 610225)

Abstract Outlier detection as one of the hot issues in data mining area aims to discover the objects with abnormal behaviors from the original data distribution. And it can generate many valuable applications, e.g., bank fraud, network instruction and etc. Currently, distributed computing has been widely applied in outlier detection. However, it still brings the lower performance of data computing since there are computing differences in compute nodes of distributed environment. To solve the problem of load balancing in distributed computing-based outlier detection with respect to large scale and high dimensional data, a weighted distributed outlier detection method has been proposed. First, we tend to ascertain the weight of data node based on computing performance of data node, whereafter dividing the data space into several grids. At last, for the purpose of parallel computing, a weighted grid-based allocation algorithm based on grid dividing is proposed, which allocates the grids to configured data nodes. The extensive experiments verify the effectiveness of proposed method, and demonstrate its better performance.

Key words dense-based outlier detection; distributed algorithm; grid partition; local outlier factor

海量数据井喷式爆发及计算机硬件技术的发展使降维技术、分布式计算技术、云计算技术等在数据挖掘领域大量运用。离群点检测作为大数据时代的重要研究方向之一，已获得研究人员的广泛关注。目前，离群点检测技术已应用于防止银行诈骗[1-2]、网络入侵检测[3-4]、视频监控[5-6]、恶意软件检测[7-8]、时间序列异常检测[9-10]等热门领域。离群点检测可以帮助发现有价值的知识和异常模式，帮助人们分析数据中存在的异常行为、不规则信息特征等。大规模离群点数据处理对计算性能提出了更高的时间限制，而利用分布式计算解决离群点检测问题受到学术界和工业界的广泛关注。

目前离群点检测没有一个被广泛接受的泛化定义。关于离群点的解释大多基于文献 [11] 的基础作出基于策略方法的具体描述。文献 [12-13] 对近年来离群点检测的方法做出了综述。其中关于邻近性的方法研究长盛不衰，主要包括基于距离的模型[14]和基于距离模型引申而来的基于密度的模型[15]。基

收稿日期：2020－04－27；修回日期：2020－09－03

基金项目：西南民族大学中央高校基本科研业务费 (2017NZYQN26)；国家自然科学基金 (61802033，61472064，61602096)；四川省科技计划 (2018GZ0087，2019YJ0543)

作者简介：梅林 (1983－)，男，博士生，主要从事离群点检测、分布式计算等方面的研究.

通信作者：张凤荔，E-mail：fzhang@uestc.edu.cn

于距离的模型是一种全局离群点检测方法，在数据集中，处于低密度区域和高密度区域之间的对象更有可能是离群点，但却难以检出。而基于密度的模型因为引入了对象邻域的概念，在众多应用场景中或检测准确率上高于基于距离的模型。随后，基于密度的模型相继产生了很多变种[16-17]，并在适应性方面进行了扩展。然而，随着数据尺度的增加，集中式的检测算法对服务器的计算性能、端口的吞吐量都有很高的要求。例如针对大型超市、证券交易中心而言，每天都会产生大量的交易信息。如果使用传统的集中式算法计算离群值，则需要花费数小时甚至数天的时间，无法保证时间有效性，在容错性上也难以满足要求。因此设计并行算法非常必要，该算法可用多台计算机来加速离群值计算。

文献 [18] 利用 Hadoop MapReduce 架构提出了一种基于属性值频率 AVF(attribute value frequency) 的在分类数据集上快速而简单检测离群点的方法 MR-AVF(mapReduce- attribute value frequency)，由于不适合在数值属性上使用，因此应用范围较为有限。文献 [19] 实现了一种基于并行 KD-Tree 的离群值检测算法 (parallel KD-tree, PKDTree)，实验显示其在大尺度数据集中的情况下有良好的检测性能。文献 [20] 提出了一种混合了类标号属性以及连续属性的数据的离群点定义，然后设计出一种分布式方法挖掘离群点。文献 [21] 提出了一种分布式环境下的 Top-n 离群点检测方法。文献 [22] 采用主从 (master-slave) 结构实现了 LOF (local outlier factor) 方法的并行计算。从节点中独立计算每个数据点的局部邻域，然后将所有数据点及其邻域转移到主节点，并在主节点上计算最终结果。因为最终所有的数据点被转移到主节点，主节点上的工作负载相当繁重。因此，当数据尺度较大时，这种方法性能将急剧下降。在文献 [22] 基础上，文献 [23] 提出了一种改进的架构。该架构由一个协调器和多个数据节点组成，并设计出相应的 GBP(grid-based partition) 算法。其中，协调器只负责整个调度，每个数据节点负责存储和计算数据子集的 LOF 值。经过比较，由于该架构的协调器不参与 LOF 值的计算，而将计算量分散到各数据节点中，因此计算速度得到很大提升。

然而，GBP 模型是按照所有数据节点是均匀的，即运算性能相同这一前提条件设计的。现实中，经常面对的环境是数据节点性能各异。例如，数台高性能计算机和数台低性能计算机搭配。GBP 模型不考虑数据节点的性能差异，往往造成性能低下的数据节点分配了过多的计算量，影响了系统的整体性能。

因此，本文提出了一种考虑数据节点性能差异的改进方法，首先确定各数据节点的性能权值，再设计分配算法考虑运算负载和网络负载的均衡性。与 GBP 相比，该方法可在非对称分布式环境中布置基于密度的离群点检测算法，实验证明了该方法的有效性。

1　问题定义与相关工作

1.1　LOF 算法的基本概念

为了克服基于距离方法计算全局离群点的局限性，提出了以 LOF 作为度量，用以捕捉一个对象邻域的局部密度的方法。首先，计算得到对象的 K-距离邻域，进而得到对象的局部可达密度，再通过同样的方法得到其邻域对象的局部可达密度，最后计算出对象的 LOF 值。LOF 值描述了对象的离群程度，值越大，该对象更倾向于离群点。该方法替代了以往方法直接给出判定结果的方式，因此通过指定阈值，可得到不同的结果。根据文献 [15]，给出 LOF 的相关定义：

定义 1　对象 a 的第 k 距离

给定正整数 k，对象 a 的 k 距离记作 k-distance (a)。在数据集 D 中，对象 a 与对象 b 的距离记作 $d(a,b)$。关于“距离”的定义可以根据具体应用调整，通常不特别指出，其代表欧氏距离。满足以下条件，则取 k-distance(a) 等于 $d(a, b)$：

1) 至少存在 k 个数据对象 $b' \in D-\{a\}$ 满足 $d(a, b') \leqslant d(a,b)$;

2) 至多存在 $k-1$ 个数据对象 $b' \in D-\{a\}$ 满足 $d(a, b') < d(a, b)$。

定义 2　数据对象 a 的 k-距离邻域

根据定义 1 可知对象 a 的 k 距离，对象 a 的 k-距离邻域是所有到 a 的距离小于等于 a 的 k-距离的对象的集合，记作：

$N_k(a) = \{a' | d(a, a') \leqslant k\text{-distance}(a)\}$。

定义 3　对象 a 相对于对象 b 的可达距离记作：

$\text{Reachdist}_k(a,b) = \max\{k\text{-distance}(b), d(a,b)\}$

定义 4　局部可达密度

把对象 a 的局部可达密度记作：

$$\text{Lrd}(a) = 1 \Big/ \left(\frac{\sum b \in N_k(a) \text{Reachdist}_k(b,a)}{|N_k(a)|} \right)$$

定义 5 局部离群因子

把对象 a 的局部离群因子记作：

$$\mathrm{LOF}_k(a)=\sum_{b\in N_k(a)}\frac{\mathrm{Lrd}(b)}{\mathrm{Lrd}(a)}\Big/|N_k(a)|$$

根据上述定义，LOF 算法需要一个超参数 k(确定邻域的范围)。k 的选择会引起 LOF 值的波动，k 值太小虽能减少计算量，但会降低检测准确率，k 值太大，计算开销会急剧增大，因此需要在实际应用中认真考察，以确定 k 值。

1.2 GBP 算法的相关概念

文献 [22] 采用主从 (master-slave) 结构实现了 LOF 方法的并行计算 (parallel LOF algorithm，PLOFA)。然而，在从节点上仅仅计算每个数据点的 k-距离邻域，而最终的 LOF 值计算则交给主节点，因此主节点计算负担重，而导致计算性能瓶颈。GBP 算法在此基础上作了进一步的工作，提出由一个协调器和多个数据节点组成的架构替代主从结构。GBP 算法最大的贡献是将计算量进一步下放到各数据节点。与 master 进程不同，协调器只负责整个分配安排。因此，即使数据尺度很大，协调器也不会成为瓶颈。在实际硬件条件允许的情况下，可通过增加数据节点以扩展系统的性能，而算法本身只需做微小的修改。

在该过程中主要采用以下 2 个步骤：

1) 在数据空间中对数据集进行网格化分。例如数据维度 d=2，每一维平均划分数 s=4，显然划分产生的网格数16(s^d)。

2) 以网格为基本单位将数据点分配给各数据节点。分配算法考虑以下两个因素：① 每个数据节点被分配的数据点数量尽量平均，以获得系统计算负载的均衡性；② 根据 1.1 节中定义可知，计算一个数据点 LOF 值时，需要计算其 k-距离邻域。例如给定 k=3，计算图 1 中g_1网格中 B 点的 LOF 值时，其 k-距离邻域中的对象除了 C点，还包含邻近网格g_2中的 D、E。假如g_1和g_2网格被分配给了不同的数据节点，则计算 B 点的 LOF 值时，协调器需要将网格g_2中的 D、E 传输给 B 点分配的数据节点，由此产生网络传输开销。因此邻近网格要尽量分配给相同数据节点，以减少传输数据点时产生的网络负担。最终分配算法 GBP 很好地平衡了两方面的性能需求。

然而，GBP 模型并没有考虑数据节点的性能差异，因此其设计适于布置在一个严苛的环境中。例如，某科研单位统一采购的一批计算机，或拥有一批相同型号性能计算机的机房等等。因此，该模型在应用时缺乏灵活性。一个极端的情况是当性能最低下的数据节点被分配了最多的数据点，则系统的整体性能会受到很大的影响。

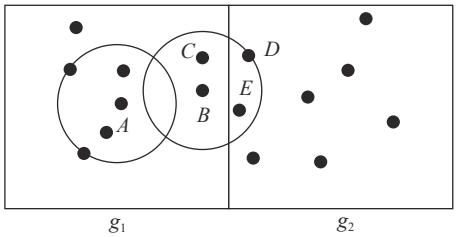


图 1 邻近网格 LOF 计算示例 (k=3)

2 基于网格划分的加权分配算法 WGBA

2.1 计算架构

计算架构如图 2 所示，本文利用一个分布式框架，该框架由一个协调器和多个性能各异的数据节点 (设有$|N|$个数据节点的集合N表示为$\{n_1,n_2,\cdots,n_{|N|}\}$)组成。每个数据节点将存储完整数据集的一部分。框架使用协调器负责调度，所有的实际计算都分配给数据节点。首先，使用基于网格划分的加权分配算法将数据集分割成几个子集，并将它们分配到数据节点。然后，数据节点负责计算本地数据点的 LOF 值，在计算数据点 LOF 值时，如其 k-距离邻域包含异地数据点，则协调器负责协调网络传输。

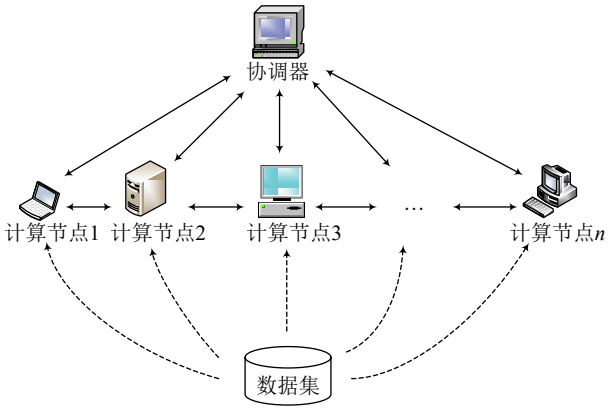


图 2 计算架构

2.2 数据节点的权重

由于不同的数据节点具有不同的计算性能，因此需要对它们的性能进行评估，确定权值。对数据节点进行评级的一种简单快速的方法是使用某种性能测试软件。然而，该方法不能准确评估数据节点在异常检测任务中的性能。因此，本文提出一种更精确的确定权重的方法。

定义 6　数据节点 i 的权重

首先，生成一个较小的数据集 S 作为测试集，该测试集大小适中。如果过小，则测试结果波动较大；过大，则会耗费过多的时间，因此本文建议该集中的数据点数量约为 10 000，维度值为 5。接下来，每个数据节点在测试集上运行 LOF 算法并获得运行时间t_i(数据节点 i 在测试集 S 的运行时间)，则数据节点 i 的权重记作：

$$w_i = \frac{t_i}{t_1 + t_2 + \cdots t_{|N|}}$$

2.3　WGBA 算法

与文献 [23] 的网格划分方法类似的，设数据集维度为 d，尝试将整个 d 维空间分割成几个等距网格。将每个维度分割成几个相等的段 (段的数量用 s 表示)。在此之后，空间被划分为s^d个网格。记$g_{x_1,x_2,\cdots,x_d}$为维度 i 在位置x_i处的网格，$i \in [1,d]$。给出邻近网格的定义：

定义 7　网格$g_{x_1,x_2\cdots,x_d}$的邻近网格记作：

$$N(g_{x_1,x_2,\cdots,x_d}) = \{g_{y_1,y_2,\cdots,y_d} |$$

$$\max(1,x_i-1) \leqslant y_i \leqslant \min(s,x_i+1), g_{y_1,y_2,\cdots,y_d} \neq g_{x_1,x_2,\cdots,x_d}\}$$

如图 3，可以看到在维度$d=2$，$s=4$的例子下，网格$g_{2,2}$的邻近网格数量为$3^d-1=8$。

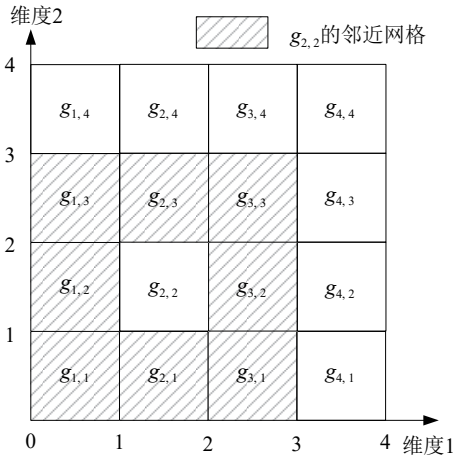


图 3　网格划分案例

为了加快异常检测的计算速度，本文考虑以下两个因素提出一种分配方法。

1) 为了获得高并行性，根据不同数据节点的计算性能来分配数据点 (均衡负载)。

2) 计算每个数据点的 k-距离邻域，如果将相邻网格尽可能分配到相同的数据节点，则可以减少网络开销。

最后，在每个数据节点上计算所分配数据点的 LOF 值。所提方法的细节如算法 1 所示。

算法 1 WGBA 算法

输入：未分配数据节点的网格集合 G, 数据节点集合 N, 数据点集合 P

输出：数据点的 LOF 值

1) 根据网格中数据点的数量，对 G 中的网格升序排列
2) 根据定义 6 计算 N 中的数据节点的权值并依权值做降序排列
3) 创建一个空的网格集合 G'(已分配数据节点的网格集合)
4) for N 中的每个数据节点 n 中的数据点数量不大于$w_i \times |p|$ do
5)　for G 中数据点最少的网格 g do
6)　将网格 g 分配给数据节点 n，并将网格 g 从 G 移至 G'
7)　将交集$N(g) \cap G$中的网格升序排列，并依次分配给数据节点 n，再将已分配的网格从 G 移至 G'
8) if $G \neq \emptyset$ then
9)　for G 中每个网格 g do
10)　在 N 中选择被分配了最多$N(g)$的数据节点，并把 g 分配给它
11) 在每个数据节点上，根据定义 2 计算其所分配的数据点的 k-距离邻域
12) 在每个数据节点上，根据定义 3 和定义 4 计算其所分配的数据点的局部可达密度
13) 在每个数据节点上，根据定义 5 计算其所分配的数据点的 LOF 值。

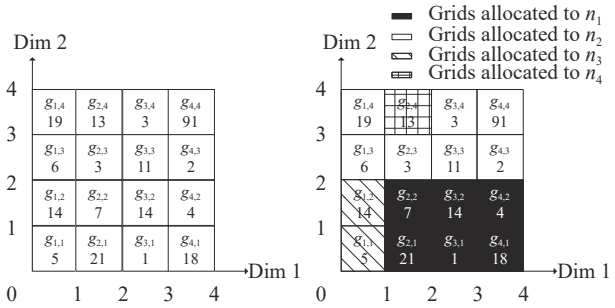
2.4　算法示例

以示例说明本文算法的过程。首先，定义 2 维空间中的数据集$P(|P|=150)$如图 4a 所示。假设已有数据节点的数量为 4 个，根据 2.2 节的方法得到数据节点的权值为：[0.4,0.3,0.2,0.1]。设置 s=4，并将空间分割为 16 个网格。相关的数据点数量显示在每个网格的底部。

根据算法 1，首先对 G 中这 16 个网格根据所包含的数据点数升序排列：$g_{3,1},g_{4,3},g_{3,4},g_{2,3},g_{4,2},g_{1,1},g_{2,2},g_{4,4},g_{3,3},g_{1,3},g_{2,4},g_{1,2},g_{3,2},g_{1,4},g_{4,1},g_{2,1}$；接着对 N 中节点根据权值降序排列，记为：n_1,n_2,n_3,n_4 (行 1)～行 2))；初始化一个空集 G' (行 3))。接下来进行第 1 次分配，先通过计算得到$w_1 \times |p|$=0.4×150 = 60再依次把网格$g_{3,1},g_{4,2},g_{2,2},g_{3,2},g_{4,1}$分配给$n_1$(行 4)～行 7))。通过重复上述过程，进行 3 次分

配，把$g_{4,3}$,$g_{3,4}$,$g_{4,4}$, $g_{3,3}$,$g_{2,3}$,$g_{1,3}$分配给n_2；把$g_{1,1}$,$g_{1,2}$分配给n_3；把$g_{2,4}$分配给n_4。此时，还有$g_{1,4}$,$g_{2,1}$两个网格未分配数据节点，因为$g_{1,4}$的邻近网格$N(g_{1,4})$中有$g_{1,3}$,$g_{2,3}$2 个网格被分配给了n_2，而只有$g_{2,4}$1 个分配给了n_4，所以把$g_{1,4}$分配给n_2；同理，把$g_{2,1}$分配给n_1(行 8)～行 10))。最后每个数据节点负责计算分配给它的数据点的 LOF 值 (行 11)～行 13))。

通过示例完整展示了 WGBA 算法的计算过程，此时，数据节点 n_1，n_2，n_3，n_4 中数据点的个数为 61，57，19，13。如图 4b 所示，该算法在考虑数据节点性能的基础上尽量将邻近的网格分配给相同的数据节点，能很好平衡计算负载和网络负载。



a. 未分配计算节点 b. 已分配计算节点

图 4 算法 1 示例

3 实验结果与分析

本文采用 python 作为开发语言。整个分布式硬件环境设置为一台协调器 (Intel Core i5 460m @ 2.53 GHz，4 GB 内存) 和 4 个数据节点组成。数据节点如表 1 所示。经训练集测试得到其对应的权值为 0.46，0.19，0.21，0.14。如 1.2 节所述，PLOFA 方法和 GBP 方法与本文研究点相似。因此将本文的方法 WGBA 与 PLOFA、GBP 布置在上述软硬件环境中，再与集中式 LOF 计算进行对比。需要说明的是，集中式 LOF 计算采用运算性能最高的数据节点 *a* 完成，以突出分布式方法的性能优势。PLOFA 方法同样使用数据节点 *a* 作为主节点。首先采用 3 个取自 UCI 的机器学习数据库的真实数据集，对 WGBA 性能进行评估。然后通过生成数据集对 WGBA 性能进行评估。

表 1 实验环境配置

数据节点	配置
数据节点*a*	Intel Core i7 6700k @ 4 GHz，16 G 内存
数据节点*b*	Intel Core i3 3220 @ 3.3 GHz，8 G 内存
数据节点*c*	Intel Core i5 8250u @1.6 GHz，8 G 内存
数据节点*d*	Intel Core i5 2540m @2.4 GHz，4 G 内存

3.1 公开数据集

本文采用 Wisdm，Query Analytics Workloads，Tamilnadu Electricity Board Hourly Readings Data Set[24]3 个公开数据集进行对比实验。数据集的基本信息如表 2 所示。

表 2 公开数据集特征

数据集名称	数据点个数	数据维度
Wisdm	15 630 426	6
Query Analytics Workloads	260 000	8
Tamilnadu Electricity Board Hourly Readings	45 781	5

依据核心算法 LOF 的相关概念，关于 *k* 值的选取影响数据点邻域的范围过大和过小都不利于算法的准确度，一般推荐值在 10～20 之间，因此选用设定 *k*=10，测试 3 种方法。实验结果如表 3 所示。

表 3 公开数据集的实验结果

模型	运算时长/ms		
	Wisdm	Query Analytics Workloads	Tamilnadu Electricity Board Hourly Readings
PLOFA	2.38×10^6	45 814	3 758
GBP	545 788	14 793	1 632
WGBA	371 749	7 233	1 310
Centralized LOF	7.09×10^6	65 283	4 462

由表 3 可知，集中式的 LOF 计算在性能上无法与分布式方法相比。在分布式方法中，本文方法消耗更少的运算时长，随着数据量的加大，表现出更好的计算性能。

3.2 人工数据集

受到开放数据集的限制，本文同时又生成大量的人工数据集对 3 种分布式方法进行测试。具体地，随机生成多个聚类中心点，并且每个簇中的数据点遵循高斯分布。最后，将一些产生的噪声添加到数据集中。本文研究重点在于通过分布式方法提高检测速度，但 *k* 值的选取对计算量的影响也非常大，因此取多个 *k* 值进行试验，以评估 *k* 值对模型的影响。如图 5 所示，数据点个数取 200 000，维度取 5，WGBA 考虑了数据节点的性能，具有整体优势，但与 PLOFA 和 GBP 一样，随着 *k* 值增加，运算量也逐渐增大，并从曲线可知，3 种方法的增长率相似。

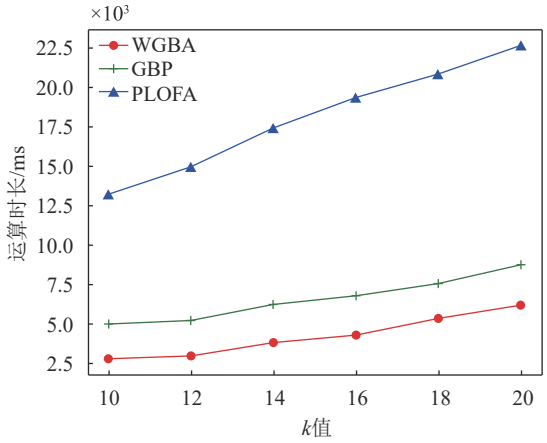


图 5　k 值选取对模型的影响

最后，考察数据尺度对模型的影响。图 6 选取 k 值为 10，维度为 5，显示了随着数据尺度的增加，由于 GBP 分配时不考虑数据节点性能的差异性，因此性能相对低下的数据节点被分配了过多的数据点，运算负担重，降低了分布式系统的整体性能。PLOFA 由于主节点具有计算瓶颈，随着数据尺度增加，3 种方法性能都有所下降，但 WGBA 具有更大的优势。

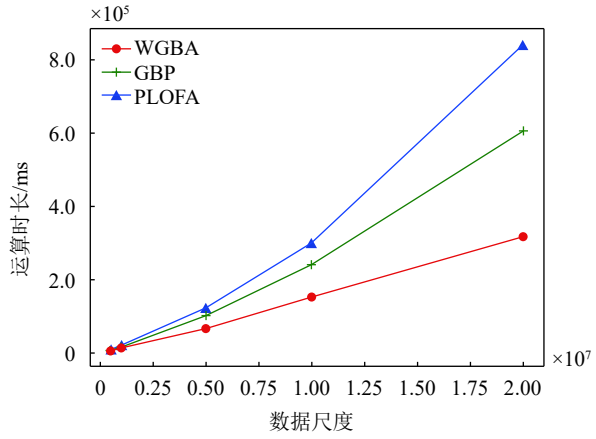


图 6　数据尺度对模型性能影响

4　结 束 语

针对大尺度数据下异常检测分布式计算负载均衡问题，本文提出了一种基于密度的分布式检测方法 WGBA。与以往技术不同，该方法考虑了数据节点的性能差异，以平衡运算负载，并且同时考虑了网络负载，因此具有很好的适应性。最后，实验结果与分析说明了本文方法的有效性。

参　考　文　献

[1] ADEWUMI A O, AKINYELU A A. A Survey of machine-learning and nature-inspired based credit card fraud detection techniques[J]. International Journal of System Assurance Engineering and Management, 2017, 8(2): 937-953.

[2] JURGOVSKY J, GRANITZER M, ZIEGLER K, et al. Sequence classification for credit-card fraud detection[J]. Expert Systems with Applications, 2018, 100: 234-245.

[3] UMER M F, SHER M, BI Y. A two-stage flow-based intrusion detection model for next-generation networks[J]. PloS One, 2018, DOI: 10.1371/journal.pone.0180945.

[4] 杨晓明, 张翔, 王佳昊, 等. 基于有限自动机的 RFID 入侵检测[J]. 电子科技大学学报, 2014, 43(5): 775-780.
YANG Xiao-ming, ZHANG Xiang, WANG Jia-hao, et al. RFID intrusion detection with finite automation[J]. Journal of University of Electronic Science and Technology of China, 2014, 43(5): 775-780.

[5] KHALEGHI A, MOIN M S. Improved anomaly detection in surveillance videos based on a deep learning method[C]//2018 8th Conference of AI & Robotics and 10th RoboCup Iranopen International Symposium (IRANOPEN). [S.l.]: IEEE, 2018: 73-81.

[6] KIRAN B R, THOMAS D M, PARAKKAL R. An overview of deep learning based methods for unsupervised and semi-supervised anomaly detection in videos[J]. Journal of Imaging, 2018, 4(2): 36.

[7] SEWAK M, SAHAY S K, RATHORE H. An investigation of a deep learning based malware detection system[C]// Proceedings of the 13th International Conference on Availability, Reliability and Security. ACM, 2018: 1-5.

[8] BOJAN K, ERAISHA G, WEBSTER G, et al. Empowering convolutional networks for malware classification and analysis[C]//2017 International Joint Conference on Neural Networks (IJCNN). [S.l.]: IEEE, 2017: 3838-3845.

[9] GOBINATH L, SAMARABANDU J, WANG Xian-bin. Sequence to sequence pattern learning algorithm for real-time anomaly detection in network traffic[C]//2018 IEEE Canadian Conference on Electrical & Computer Engineering (CCECE). [S.l.]: IEEE, 2018: 1-4.

[10] LÄNGKVIST M, KARLSSON L, LOUTFI A. A review of unsupervised feature learning and deep learning for time-series modeling[J]. Pattern Recognition Letters, 2014, 42: 11-24.

[11] HAWKINS D M. Identification of outliers[M]. London: Chapman and Hall, 1980.

[12] 吴镜锋, 金炜东, 唐鹏. 数据异常的监测技术综述[J]. 计算机科学, 2017(S2): 34-38.
WU Jing-feng, JIN Wei-dong, TANG Peng. Survey on monitoring techniques for data abnormalities[J]. Computer Science, 2017(S2): 34-38.

[13] 梅林, 张凤荔, 高强. 离群点检测技术综述[EB/OL]. [2020-04-29]. https://www.arocmag.com/article/02-2020-12-002.html.
MEI Lin, ZHANG Feng-li, GAO Qiang. Overview of outlier detection technology[EB/OL]. [2020-04-29]. https://www.arocmag.com/article/02-2020-12-002.html

[14] KNORR E, NG R. Algorithms for mining distance-based outliers in large datasets[C]//Proc of the 24th VLDB

Conference. San Francisco: Morgan Kaufmann Publishers Inc, 1998: 392-403.

[15] BREUNIG M M, KRIEGEL H P, NG R T, et al. LOF: Identifying density-based local outliers [C]//ACM sigmod record. New York: ACM Press, 2000: 93-104.

[16] 施化吉, 周书勇, 李星毅, 等. 基于平均密度的孤立点检测研究[J]. 电子科技大学学报, 2007, 36(6): 1286-1288, 1295.
SHI Hua-ji, ZHOU Shu-yong, LI Xing-yi, et al. Average density-based outliers detection[J]. Journal of University of Electronic Science and Technology of China, 2007, 36(6): 1286-1288, 1295.

[17] KRIEGEL H P, KRÖGER P, SCHUBERT E, et al. LOOP: Local outlier probabilities[C]//Proceedings of the 18th ACM conference on Information and knowledge management. [S.l.]: ACM, 2009: 1649-1652.

[18] KOUFAKOU A, SECRETAN J, REEDER J, et al. Fast parallel outlier detection for categorical datasets using MapReduce[C]//IEEE International Joint Conference on Neural Networks. Piscataway, NJ: IEEE Press, 2008: 3298-3304.

[19] HE Qing, MA Yun-long, WANG Qun, et al. Parallel outlier detection using kd-tree based on mapreduce[C]// IEEE Third International Conference on Cloud Computing Technology & Science. Piscataway, NJ: IEEE Press, 2012: 75-80.

[20] OTEY M E, GHOTING A, PARTHASARATHY S. Fast distributed outlier detection in mixed-attribute data sets[J]. Data Mining and Knowledge Discovery, 2006, 12(2-3): 203-228.

[21] ANGIULLI F, BASTA S, LODI S, et al. Distributed strategies for mining outliers in large data sets[J]. IEEE Transactions on Knowledge and Data Engineering, 2013, 25(7): 1520-1532.

[22] LOZANO E, ACUNA E. Parallel algorithms for distance-based and density-based outliers[C]//Proceedings of the Fifth IEEE International Conference on Data Mining. Piscataway, NJ: IEEE Press, 2005: 4.

[23] BAI MEI, WANG X, XIN Jun-chang, et al. An efficient algorithm for distributed density-based outlier detection on big data[J]. Neurocomputing, 2016, 181(C): 19-28.

[24] MERZ C J. UCI repository of machine learning databases[EB/OL]. [2020-01-23]. http://www.ics.uci.edu/~mlearn/MLRepository.html.

编 辑 税 红