International Journal of Grid and High Performance Computing
Volume 16 • Issue 1

Pre-Cutoff Value Calculation Method for Accelerating Metric Space Outlier Detection

Honglong Xu, Foshan University, China
https://orcid.org/0000-0002-8645-9028
Zhonghao Liang, Foshan University, China
Kaide Huang, Foshan University, China*
Guoshun Huang, Foshan University, China
Yan He, Foshan University, China

ABSTRACT

Outlier detection is an important data mining technique. In this article, the triangle inequality of distances is leveraged to design a pre-cutoff value (PCV) algorithm that calculates the outlier degree pre-threshold without additional distance computations. This algorithm is suitable for accelerating various metric space outlier detection algorithms. Experimental results on multiple real datasets demonstrate that the PCV algorithm reduces the runtime and number of distance computations for the iORCA algorithm by 14.59% and 15.73%, respectively. Even compared to the new high-performance algorithm ADPOD, the PCV algorithm achieves 1.41% and 0.45% reductions. Notably, the non-outlier exclusion for the first data block in the dataset is significantly improved, with an exclusion rate of up to 36.5%, leading to a 23.54% reduction in detection time for that data block. While demonstrating excellent results, the PCV algorithm maintains the data type generality of metric space algorithms.

KEYWORDS

distance triangle inequality, index, metric space, outlier detection, pre-cutoff value

INTRODUCTION

The continuous expansion of data volumes and innovative applications propels the burgeoning growth of the big data industry. The cumulative volume of data places significant stress on data storage capabilities. However, simply increasing storage devices is not a sustainable solution. Viable strategies include timely data analysis and mining, as they alleviate the pressure on storing raw data and maximize data value.

As a core step in data processing, data mining is an active academic research field, giving rise to various techniques, such as classification, clustering, association analysis, and outlier detection. While most data mining techniques focus on discovering regular patterns within datasets, non-regular

DOI: 10.4018/IJGHPC.334125 *Corresponding Author

patterns are sometimes equally valuable. Outlier detection algorithms constitute a vital branch of data mining designed to identify these non-regular patterns. They enable the discovery of distinctive data points within a dataset, known as outliers or anomalies. Thanks to outlier detection, we can promptly identify exceptional events within data, such as network attacks (Catillo et al., 2023), fraudulent transactions (Hilal et al., 2022), or equipment malfunctions (Nesa et al., 2018), thus reducing losses. Moreover, outlier detection contributes to enhancing data quality (Larson et al., 2019), recognizing exceptional outcomes within datasets, and even unearthing new knowledge.

Research into outlier detection algorithms focuses on either specialized or generic algorithms. Specialized algorithms are tailored and optimized for the characteristics of data in various domains, making full use of available information to expedite outlier detection. However, in the era of big data, the ever-expanding and diverse data types pose significant challenges to the design capacity of specialized algorithms. In such circumstances, generic outlier detection algorithms have emerged. Generic algorithms abstract and unify commonalities across data types in different domains, performing searches and mining using only partial information from the dataset, thereby enabling the application of a single algorithm across diverse data types. While there may be some performance trade-offs, generic algorithms significantly reduce data mining systems' development and maintenance costs, bringing them significant attention in academic and industrial circles in recent years.

Fortunately, most data types can be designed with distance functions that adhere to the triangle inequality, allowing them to be mapped to metric spaces (Mao et al., 2015). This, in turn, facilitates the use of metric space algorithms for retrieval, analysis, and mining. Metric space outlier detection utilizes a definition of outliers entirely based on distance and detection algorithms that are also fully reliant on distance. It eschews the use of any information other than distance, making it applicable to a wide range of data types.

Metric space outlier detection algorithms belong to the distance-based outlier detection algorithm class. Much like their counterparts, they employ distance triangle inequality for pruning to eliminate non-outliers and accelerate the outlier detection process. This step heavily depends on the cutoff value of the outlier degree, where a higher value leads to improved efficiency in non-outlier exclusion. However, existing metric space outlier detection algorithms face a critical issue when detecting the first data block, where the available outlier degree cutoff value is set to 0, causing significant delays in detecting that block and severely impeding overall detection efficiency. To address this problem, we propose a pre-cutoff value calculation method that can be used to accelerate metric space outlier detection. By making full use of the distance triangle inequality, this method calculates an initial outlier degree cutoff value, hereafter referred to as the "pre-cutoff value," through minimal computations.

The main contributions of this paper are summarized as follows:

(1) Analyzing the time allocation of each data block in the outlier detection process reveals that the time cost of the first data block is significantly greater than that of other blocks, and we analyze the reasons behind this.
(2) Introducing a pre-cutoff value calculation method to accelerate metric space outlier detection, utilizing the distance triangle inequality, and designing a method that does not require additional distance computations.
(3) Proving mathematically that the pre-cutoff value–based accelerated outlier detection algorithm guarantees correct results.

The subsequent sections of this paper are outlined as follows: The next section provides an overview of distance-based outlier detection algorithms, including some metric space outlier detection algorithms. After that, we introduce the pre-cutoff value calculation method for accelerating metric space outlier detection and then analyze and prove its correctness. The following section presents the experimental results and provides an analysis. Finally, the last section summarizes the paper's work and outlines potential future work.

RELATED WORK

Distance-based outlier definitions primarily fall into three categories: DB(p,d) outliers (Knorr & Ng, 1998; Knorr et al., 2000), k-nearest neighbor (kNN) distance sum outliers (Angiulli & Pizzuti, 2005), and k-distance outliers (Ramaswamy et al., 2000). These three definitions are all nearest neighbor outlier definitions, measuring the outlierness of data points based on the number or distance of their nearest neighbors.

DB(p,d) outliers, for instance, are defined as data points in the dataset *DS* for which at least a proportion *p* of points in *DS* have a distance greater than *d* from them. An equivalent definition states that a point is an outlier if the number of points within a distance *R* from it does not exceed *k*; such a point is known as a *k-R* outlier. According to this definition, a point is either a regular point or an outlier with no in-between status.

By contrast, k-NN distance sum outliers are defined by considering the sum of the distances to the k nearest neighbors for each object in the dataset as their outlier degree. The top *n* objects with the highest outlier degree are then considered the top-n outliers. In other words, the average distance to the *k* nearest neighbors is defined as the outlier degree, and the top-n outliers are ranked accordingly.

Distance-based outlier detection algorithms, based on the aforementioned distance-based outlier definitions, can be traced back to 1998 when Knorr and Ng (1998) first introduced distance-based outlier definitions and simultaneously proposed three categories of distance-based detection algorithms: index-based, nested-loop, and cell-based algorithms (Knorr et al., 2000).

In 2003, Bay and Schwabacher (2003) presented the ORCA algorithm, which partitions the dataset and applies a simple pruning rule. Once an object's current outlier degree falls below the threshold for the top-n outliers, that object is no longer considered a potential outlier. This approach achieved near-linear detection speed, establishing itself as a state-of-the-art algorithm in the field of outlier detection.

Building on this, in 2011, Bhaduri et al. (2011) introduced the iORCA algorithm. It randomly selects a pivot from a dataset, calculates the distances between all objects and the pivot, and builds an index based on the sorted distances in descending order. When detecting outliers, it employs a "spiral" search of the k-nearest neighbors based on their distance in the index, terminating the process early using termination rules and achieving correct results. The iORCA algorithm boasts a low index construction time and high detection efficiency, making it a model for distance-based outlier detection algorithms.

With the Knorr outlier definition, Tao et al. (2006) carefully designed the memory allocation and devised the scan with prioritized flushing (SNIF) algorithm, which, assuming a memory capacity of 1% of the dataset, requires only three scans to complete outlier detection. After recognizing that actual database management system memory capacity tends to be larger, they further optimized memory allocation to create an algorithm requiring only two scans of the dataset.

Vu and Gopalkrishnan (2009) combined clustering and the nested-loop algorithm to propose the MultI-Rule Outlier (MIRO) algorithm. This algorithm conducts outlier detection in two stages. It initially uses the K-means algorithm for clustering and eliminates non-outlier categories based on the clustering results. Subsequently, it applies the nested-loop algorithm to detect outliers within the

Table 1. Outlier definitions and their characteristics

Outlier definition	Characteristics of outlier definition
k-R outliers (DB(p,d))	The objects for which there are less than k other objects within distance R.
kNN distance sum outliers	Top *n* objects whose sum of distance to the *k* nearest neighbors is greatest.
k-distance outliers	Top *n* objects whose distance to the *k*th nearest neighbor is greatest.

Table 2. Outlier detection algorithms and their characteristics

Algorithm	Characteristics of algorithm
NL	Original nested loop algorithm without index and optimization.
ORCA	Processes data in random order and uses a pruning rule that once an object's current outlier degree falls below the threshold for the top-n outliers, that object is no longer considered a potential outlier.
iORCA	Randomly selects a pivot from the dataset, calculates the distances between all objects and the pivot, and then sorts them in descending order for outlier detection.
SNIF	Assumes a memory capacity of 1% of the dataset, requires only three scans to complete outlier detection.
MIRO	Uses the K-means algorithm for clustering and eliminates non-outlier categories based on the clustering results, then applies the nested-loop algorithm to detect outliers within the remaining dataset.
K-means--	Simultaneously perform K-means clustering and outlier detection.
RCS	Optimizes the update of cutoff value and the utilization of memory.
DOLPHIN	Only requires two scans of the dataset by storing a portion of the dataset in the main memory.
SolvingSet Algorithm	Distributed outlier detection algorithm; uses a subset of the data set, called the solving set, to predict if new unseen objects are outliers.
DOoR	Distributed outlier detection algorithm for multicore computer clusters connected on a ring topology.
SFC	Uses space-filling curve indexing to search for approximate reverse k-nearest neighbors, resulting in nearly linear detection speed.
LEAP	Outlier detection algorithms designed for high-volume data streams (Cao et al., 2014); applicable to all three major distance-based outlier definitions.
HIOD	By selecting multiple support points, the dataset is mapped to the support point space, and then a Hilbert index is established, prioritizing the detection of data blocks in relatively sparse regions.
ADPOD	An adaptive cutoff distance-based density peak pivot selection method applied to metric space outlier detection.
KFC	Neighborhood consistency-based parameter k search aimed at selecting k for kNN-based outlier.
LOF	Local outlier factor as the ratio of the average reachability density of an object's neighborhood to its own reachability density.
UKOF	Defines a KDE-based outlier factor (KOF) to measure the local outlierness score; effective for high-volume data streams.
mRMRD	An efficient unsupervised density-based subspace selection for outlier detection in the projected subspace; effective for high dimensional data.
BLDOD	Local outlier detection method that can draw the neighborhood boundaries of the data points via Chebyshev inequality; effective for high dimensional data.

remaining dataset. Since the second stage focuses on a significantly smaller dataset, the algorithm exhibits favorable time complexity.

In contrast to MIRO's two-stage approach, Chawla and Gionas (2013) concentrated on simultaneously conducting K-means clustering and outlier detection. He introduced the k-means-- algorithm, which addresses the challenging interplay between K-means clustering and outlier detection.

Szeto and Hung (2010) conducted a meticulous probabilistic analysis of the ORCA algorithm. They optimized the TOP n outlier threshold and the utilization of trimmed memory recycling, focusing on the size of data blocks loaded into memory buffers. These optimizations resulted in a substantial improvement in algorithm performance.

Various researchers have focused extensively on dealing with large-scale data. Angiulli and Fassetti (2007) introduced an algorithm called DOLPHIN. This algorithm maintains a certain number of data objects in memory and establishes an index, accelerating nearest-neighbor searches and

achieving near-linear time complexity and I/O overhead reduction. In addition, Angiulli et al. (2006) extended their earlier Solving Set algorithm to propose a distributed outlier detection algorithm on a computer cluster architecture with a master node. The experiments conducted on an 11-node cluster yielded promising results.

Researchers at NASA Ames Research Center, including Bhaduri et al. (2011), proposed two distributed outlier detection algorithms for multicore computer clusters connected on a ring topology. They achieved favorable experimental results.

Additionally, Schubert et al. (2015) utilized space-filling curve indexing to search for approximate reverse k-nearest neighbors, resulting in a nearly linear detection speed. Cao et al. (2014) investigated outlier detection algorithms designed for high-volume data streams. In the context of growing data volumes, such outlier detection algorithms exhibit strong potential for practical applications.

Furthermore, optimizations for the iORCA algorithm have been proposed in recent years. Xu et al. (2016) proposed a fast metric space outlier detection algorithm based on Hilbert indexing. By selecting multiple pivots, they reduced spatial distortion and built a Hilbert index, prioritizing the detection of data blocks in relatively sparse regions. Subsequently, Xu et al. (2019) addressed the issue of unstable detection performance caused by the random selection of support points in the iORCA algorithm. They designed an adaptive cutoff distance-based density peak pivot selection algorithm and applied it to metric space outlier detection, achieving stable outlier detection performance with minimal support point selection time overhead.

For distance-based outlier detection algorithms, the choice of the parameter k, representing the number of nearest neighbors, is crucial. Yang et al. (2023) introduced the "neighborhood consistency-based parameter *k* search" method to address this. This method is independent of other parameters, has linear time complexity, and exhibits good generality for various datasets and detectors.

Density-based outlier detection algorithms are essentially a subset of distance-based outlier detection algorithms. The key difference is that they target local outliers rather than global outliers. Breunig et al. (2000) introduced the concept of local outliers and the corresponding local outlier factor (LOF) detection algorithm. They argued that in some cases, local outliers are more important than global outliers, and conventional outlier detection algorithms are only effective in detecting global outliers, resulting in lower accuracy in detecting local outliers. They represented the local outlier factor as the ratio of the average reachability density of an object's neighborhood to its own reachability density. Subsequently, they ranked the top *n* local outliers.

Liu et al. (2020) addressed the issue of imbalanced data distribution in different subsets of high-throughput data streams, proposing a method for local outlier detection in large-scale high-throughput data streams based on kernel density estimation. They designed effective pruning strategies, significantly improving outlier detection speed.

The "curse of dimensionality" poses a severe challenge to outlier detection performance in high-dimensional data. Riahi-Madvar et al. (2021) utilized density-based unsupervised subspace selection for outlier detection in projected subspaces. They calculated the LOF for data points in the corresponding subspace. Experimental results demonstrated that this algorithm not only reduced computational complexity and execution time but also improved the accuracy of outlier detection.

Aydın (2023) employed inequalities to delineate neighborhood boundaries for data points and detected outliers by quantifying the density of their neighborhoods, yielding effective results. Table 2 summarizes these outlier detection algorithms and their characteristics.

PRE-CUTOFF VALUE CALCULATION FOR METRIC SPACE OUTLIER DETECTION

As previously discussed, existing metric space outlier detection algorithms involve a limitation when detecting the first data block: They use a default outlier degree cutoff value of 0, which does not effectively exclude non-outliers. In this section, we introduce a pre-cutoff value acceleration method

for metric space outlier detection. Leveraging the distance triangle inequality, we designed a pre-cutoff value calculation method that does not require additional distance calculations.

Algorithm 1 consists of two functions: getKthNN from lines 1 to 26 and getPreCutoffValue from lines 27 to 41. In the provided pseudocode, index[] is a one-dimensional sorted array with a length of *size*, *k* is the *k*-th nearest neighbor parameter, and *tmp* is a temporary variable passed from outside the function, reducing unnecessary computation overhead.

The getKthNN function effectively applies an optimized binary search algorithm to a one-dimensional sorted array. As shown in lines 3–4, if the object *id* to be processed is located at the beginning or end of the array, i.e., its index is 0 or size-1, then we can directly determine the position of its *k*-th nearest neighbor, which is at the index abs(*id* - *k*), where abs is the absolute value function.

Table 3. Pre-Cutoff value calculation for metric space outlier detection

Algorithm 1: Pre-Cutoff Value Calculation Method	
Input: index[], k, n, size **Output**: c	
1:	**getKthNN**(index[], size, id, k, tmp)
2:	start1←0, end1←0, start2←0, end2←0;
3	**if** (id==0 || id==size-1) **then**
4	**return** index[abs(id-k)];
5	start1←id-1, end1←id-k, start2←id+1, end2←id+k;
6	**if** (id<k) **then** end1←0;
7	**if** (id>size-1-k) **then** end2←size-1;
8	Len1←start1-end1+1;
9	Len2←end2-start2+1;
10	**if** (len1>=k && dist(index[id],index[end1])<tmp || len2>=k && dist(index[id],index[end2])<tmp) **then**
11	**return** 0;
12	klen←k;
13	i←start1-min(len1, k/2)+1;
14	j←start2-min(len2, k/2)-1;
15	**while** (klen>1)
16	**if** (index[i]>index[j]) **then**
17	start2←j+1;
18	len2←end2-start2+1;
19	klen←klen-(j-start2+1);
20	**else**
21	start1←i-1;
22	len1←start1-end1+1;
23	klen←klen-(i-start1+1);
24	**if** (klen==1) **then**
25	**return** min(index[start1], index[start2]);
26	**end**
27	**getPreCutoffValue**(index[], size, k, n)
28	c←0, tmp←0, tmpDis←0;
29	build minheap;
30	**for** id = 0 **upto** size-1 **do**
31	tmpDis ←getKthNN(index[], size, id, k, tmp);
32	**if** (tmpDis>0)
33	**if** (minHeap.size<n)
34	minheap.push(tmpDis);
35	**else if** (tmpDis>minheap.top)
36	minheap.pop;
37	minheap.push(tmpDis);
38	tmp←minHeap.top;
39	c ←minheap.top;
40	**return** c;
41	**end**

In addition, it is necessary to calculate the search range before and after *id*, i.e., on its smaller side, from *start1* to *end1*, and on its larger side, from *start2* to *end2*. It should be noted that, generally, the order is from small to large: *end1*, *start1*, *id*, *start2*, *end2*. Additionally, cases where id approaches the array boundary and does not have *k* elements must be handled (lines 5–9).

Next, in lines 10–11, the current pre-cutoff value *tmp* passed from outside the function comes into play. If the distance between id and any *k*-th nearest neighbor on either side is less than *tmp*, *id* can be excluded directly (by simply setting its pre-outlier degree to 0 and returning it), as its one-dimensional outlier degree cannot be greater than *tmp*. The search length *klen* is then initialized as *k* (line 12), and comparisons are made at *k/2–1* on both sides of *id* because the *k*-th nearest neighbor must be on the smaller side. The search length *klen* is updated, the search range is adjusted, and the binary search continues (lines 13–23) until *klen* = 1, at which point the *k*-th nearest neighbor is found (lines 24–25).

The getPreCutoffValue function primarily builds a min heap with a length of *n* (line 29), calling the getKthNN function to calculate the one-dimensional outlier degree (line 31). Objects with a one-dimensional outlier degree greater than 0, when heap size is less than *n*, will be pushed into the heap sequentially (lines 32–34). Otherwise, when the length reaches *n*, only those with a one-dimensional outlier degree greater than the minimum element of the heap can be pushed into the heap. At the same time, the smallest element is removed to maintain the heap size of *n*, and the *tmp* value is updated (lines 35–38). After scanning the entire index[], the pre-cutoff value *c* is returned.

Algorithm Analysis

Time Complexity Analysis

The getKthNN function employs a variant of a binary search to search for the *k*-th nearest neighbor of the element at index id in the ordered array. Its time complexity is O(log *k*). The getPreCutoffValue function consists of two main operations: Firstly, it iteratively calls the getKthNN function, attempting to find the *k*-th nearest neighbor of each element in the index array. Since the array length is *size*, the time complexity of this part of the operation is O(*size* * log *k*). Secondly, it involves building a min-heap to maintain the n elements with the largest distances from their *k*-th nearest neighbors, i.e., the *n* elements with the highest pre-cutoff values. Because it requires traversing the array, and each element trying to enter the heap requires O(log *n*), this part of the operation would require at most O(*size* * log *n*). The combined time complexity of these two operations is O(*size* * log *k* + *size* * log *n*). However, given that *k* and *n* are typically very small constants, the pre-cutoff value (PCV) algorithm can be approximated to O(*size*), which implies linear time complexity.

Proof of Correctness of Outlier Detection Results

Overall Approach. Due to the triangular inequality of distances, calculating distances between each object in the dataset and a reference point maps them to one-dimensional space. In this one-dimensional space, distances between pairs of objects (one-dimensional space distances) are less than or equal to their actual distances in the multi-dimensional space. As a result, when searching for the *k* nearest neighbors of an object s_a in one-dimensional space, all *k* nearest neighbors' one-dimensional space distances (computed using $pdist()$) are less than or equal to their multi-dimensional distances (computed using $dist()$). This leads to the conclusion that s_a's one-dimensional outlier degree is less than or equal to its multi-dimensional outlier degree. By extension, it is clear that the one-dimensional outlier degrees of all objects are less than their multi-dimensional outlier degrees. By taking the *n* objects with the highest one-dimensional outlier degrees and using the smallest one-dimensional outlier degree as the pre-cutoff value *c*, it can be similarly proven that *c* is less than or equal to the cutoff value of the multi-dimensional outlier degree. Using *c* to exclude non-outliers will not mistakenly exclude outliers, ensuring the correctness of the final results.

Proof Process. (1) Definitions and notations used in the proof:

Let DS be the dataset, p be a randomly selected object in DS, namely pivot, and $dist()$ be the distance function. After calculating the distances between all objects in DS and p using the $dist()$ function, these distance values create a one-dimensional space based on p, referred to as pDS.

Let $pdist()$ be the one-dimensional space distance function (absolute difference), which is defined as $pdist\left(s_1, s_2\right) = \left|dist\left(s_1, p\right) - dist\left(s_2, p\right)\right|$. Due to the triangular inequality of distances, it holds that $pdist\left(s_1, s_2\right) \le dist\left(s_1, s_2\right)$.

In addition, $nn_i\left(s, DS\right)$ represents the i-th nearest neighbor of object *s* in the dataset DS, and $w_k\left(s, DS\right)$ represents the outlier degree of object *s* in dataset DS (for distinction, we refer to this as the multi-dimensional outlier degree). Correspondingly, the *k*-th nearest neighbor of object *s* in dataset DS is referred to as the multi-dimensional *k*-th nearest neighbor.

Similarly, $nn_i\left(s, pDS\right)$ represents the i-th nearest neighbor of object *s* in the one-dimensional space pDS, and $w_k\left(s, pDS\right)$ represents the outlier degree of object *s* in the one-dimensional space pDS (referred to as one-dimensional outlier degree). The *k*-th nearest neighbor of object *s* in pDS is referred to as the one-dimensional *k*-th nearest neighbor. Therefore:

Consider an object s_a, and its *k*-th nearest neighbors in pDS, denoted as s_{a1}, s_{a2}, s_{a3}, ..., s_{ak}, where $1 \le i \le k$, meaning $s_{ai} = nn_i\left(s_a, pDS\right)$.

By the triangular inequality of distances, we have: $pdist\left(s_a, s_{ai}\right) \le dist\left(s_a, s_{ai}\right)$.

Table 3 summarizes the notations and their description.

(2) First, we will prove that the one-dimensional outlier degree $w_k\left(s_a, pDS\right)$ of object s_a is less than or equal to the multi-dimensional outlier degree $w_k\left(s_a, DS\right)$.

We will prove this in three cases based on whether the multi-dimensional k-th nearest neighbor $nn_k\left(s_a, DS\right)$ is the same as the one-dimensional k-th nearest neighbor $nn_k\left(s_a, pDS\right)$:

① $nn_k\left(s_a, DS\right) = nn_k\left(s_a, pDS\right)$

Proof:

Due to the triangle inequality in the metric space, we have:

$$pdist\left(s_a, nn_k\left(s_a, pDS\right)\right) \le dist\left(s_a, nn_k\left(s_a, DS\right)\right).$$

That is, $w_k\left(s_a, pDS\right) \le w_k\left(s_a, DS\right)$ is proven.

② $nn_k\left(s_a, DS\right) \ne nn_k\left(s_a, pDS\right)$, and $nn_k\left(s_a, pDS\right) \in \left\{nn_i\left(s_a, DS\right) \mid 1 \le i \le k-1\right\}$

Proof:

Without loss of generality, assume that $nn_k\left(s_a, pDS\right) = nn_b\left(s_a, DS\right)$, where $1 \le b \le k-1$.

Then, we have $dist\left(s_a, nn_b\left(s_a, DS\right)\right) \le dist\left(s_a, nn_k\left(s_a, DS\right)\right)$, as well as

$pdist\left(s_a, nn_k\left(s_a, pDS\right)\right) \le dist\left(s_a, nn_b\left(s_a, DS\right)\right)$; therefore

$pdist\left(s_a, nn_k\left(s_a, pDS\right)\right) \le dist\left(s_a, nn_k\left(s_a, DS\right)\right)$, so that $w_k\left(s_a, pDS\right) \le w_k\left(s_a, DS\right)$ is proven.

Table 3. Notations and description

Notation	Description
k	Number of neighbors for calculating outlier degree
n	Number of outliers to detect
c	Cutoff value, equal to the outlier degree of current nth outlier
dist	Distance function
DS	Dataset
p	A randomly selected object in the dataset, also called the pivot
pDS	A one-dimensional space produced from the distances between all objects in DS and p using the $dist()$ function
$pdist()$	Distance function used in the one-dimensional space
$nn_i\left(s, DS\right)$	The i-th nearest neighbor of object s in the dataset DS
$w_k\left(s, DS\right)$	The outlier degree of object s in dataset DS
$nn_i\left(s, pDS\right)$	The i-th nearest neighbor of object s in the one-dimensional space pDS
$w_k\left(s, pDS\right)$	The outlier degree of object s in the one-dimensional space pDS
s_a	An object of pDS
s_{ai}	The i-th nearest neighbor of object in the one-dimensional space

③ $nn_k\left(s_a, DS\right) \neq nn_k\left(s_a, pDS\right)$, and $nn_k\left(s_a, pDS\right) \notin \left\{nn_i\left(s_a, DS\right) \mid 1 \le i \le k-1\right\}$

Proof:

There must be $nn_i\left(s_a, DS\right) \notin \left\{nn_j\left(s_a, pDS\right) \mid 1 \le j \le k\right\}$, where $1 \le i \le k$. Without loss of generality, let us assume that the object is $nn_b\left(s_a, DS\right)$, and further let it be the c-th nearest neighbor of s_a in pDS, $nn_c\left(s_a, pDS\right)$, namely $nn_b\left(s_a, DS\right) = nn_c\left(s_a, pDS\right)$, where $1 \le b \le k$, and $c > k$.

Then, it follows that $dist\left(s_a, nn_b\left(s_a, DS\right)\right) \le dist\left(s_a, nn_k\left(s_a, DS\right)\right)$, and $pdist\left(s_a, nn_k\left(s_a, pDS\right)\right) \le pdist\left(s_a, nn_c\left(s_a, pDS\right)\right)$, and furthermore that $pdist\left(s_a, nn_c\left(s_a, pDS\right)\right) \le dist\left(s_a, nn_b\left(s_a, DS\right)\right)$. Therefore, it holds that $pdist\left(s_a, nn_k\left(s_a, pDS\right)\right) \le dist\left(s_a, nn_k\left(s_a, DS\right)\right)$, so that $w_k\left(s_a, pDS\right) \le w_k\left(s_a, DS\right)$ is proven.

Combining the results from ① to ③, we can conclude that the one-dimensional outlier degree $w_k\left(s_a, pDS\right)$ of object s_a is less than or equal to the multi-dimensional outlier degree $w_k\left(s_a, DS\right)$.

Based on the generality of object s_a, it can be inferred that the one-dimensional outlier degree of each object is less than or equal to its multi-dimensional outlier degree.

(3) According to the results from (1), it can be similarly proven that the cutoff value of a one-dimensional outlier degree is less than or equal to that of a multi-dimensional outlier degree.

(4) The cutoff value of the multi-dimensional outlier degree, which is the outlier degree of the *n*-th multi-dimensional outlier, is used in this paper to exclude non-outliers with a pre-cutoff value less than or equal to the previously mentioned cutoff value. This will never result in false exclusions and thus guarantees the correctness of the detection results.

EXPERIMENTAL RESULTS AND ANALYSIS

This section begins with an introduction to the datasets used in the experiments, followed by an explanation of the experimental platforms and settings. Finally, it presents the experimental results along with a detailed analysis.

Experimental Datasets

In this paper, we used four real datasets, all obtained from the UCI Machine Learning Repository and Outlier Detection DataSets (ODDS). They are detailed as follows:

KDD Cup 1999 Dataset[1]: Originally sourced from the Defense Advanced Research Projects Agency (DARPA) of the United States Department of Defense, this dataset has been processed by Professor Sal Stolfo and others at Columbia University. This paper utilizes the 10% version of the TCP dataset, which consists of 190,065 records, each containing 42 attributes.

Shuttle Dataset[2]: This dataset originates from the National Aeronautics and Space Administration (NASA) and contains space shuttle data. It comprises 58,000 records, each having nine attributes.

HTTP Dataset[3]: Derived from the original KDD Cup 1999 dataset available in the UCI Machine Learning Repository, this dataset has been crafted by researchers as an anomaly detection dataset. It comprises 567,479 records, each containing three attributes.

Mammography data set[4]: This dataset, provided by Aleksandar Lazarevic, is now publicly available in openML. It contains 11,183 samples with 260 calcifications, with each record containing six attributes.

Experimental Platform and Settings

The experimental environment utilized a Windows 11 Professional 22H2 operating system running on an Intel i9-12900K CPU with 64GB of DDR5 memory. The experimental program was developed using Visual Studio 2022 Community and compiled in Release mode.

Unless otherwise specified, to facilitate comparisons with the benchmark algorithms iORCA and ADPOD, we kept the experimental parameters in this paper consistent with those algorithms. This includes setting the number of nearest neighbors (k) to 5, the intended number of detected outliers (n) to 30, and the data block size (m) to 1,000. Notably, since the PCV algorithm employs the same outlier definition as iORCA and ADPOD, the outlier detection results are also consistent with them. As a result, we did not conduct accuracy-related experiments in this paper.

We normalized all datasets using the min-max scaling method to map the data into the [0, 1] interval. We used the following distance function:

$$dist\left(x_i, x_j\right) = \sqrt{\sum_{k=1}^{d} \delta\left(x_{ik}, x_{jk}\right)}$$

where

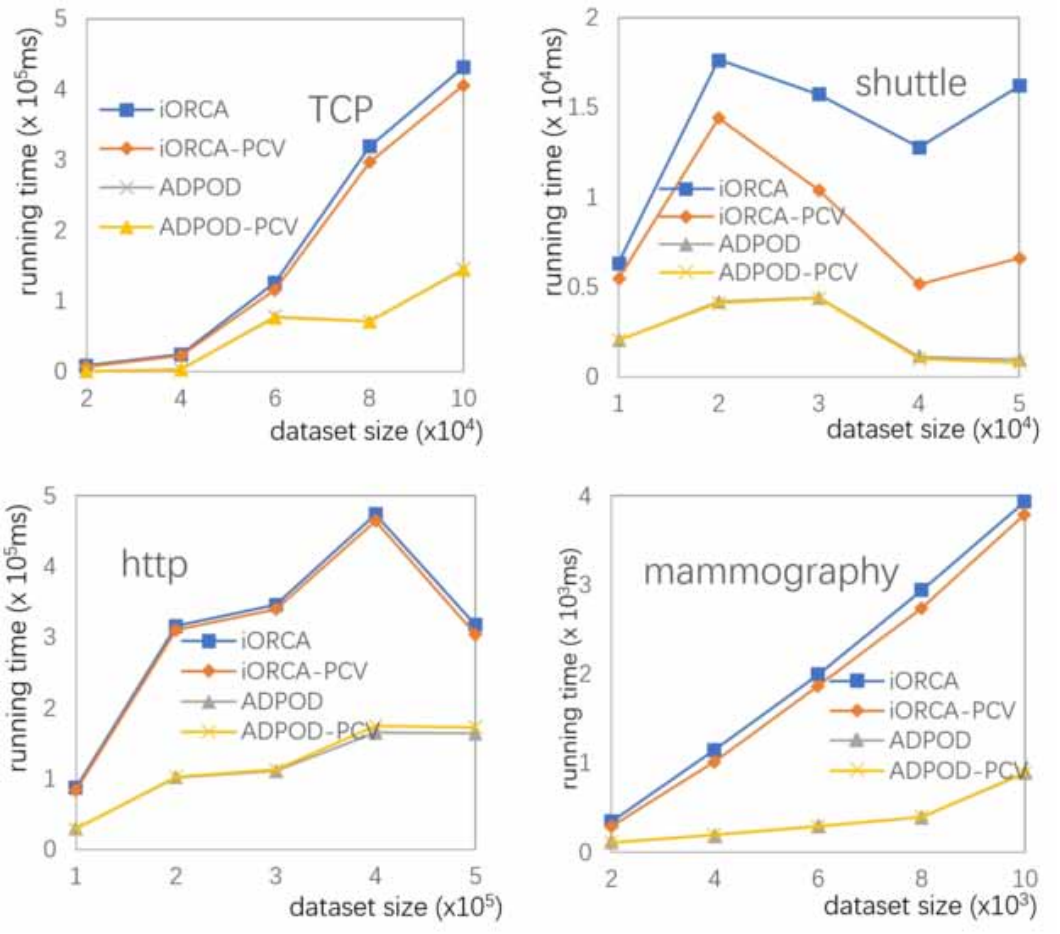
$$\delta\left(x_{ik}, x_{jk}\right) = \begin{cases} (x_{ik} - x_{jk})^2, for\, continuous\, attributes \\ x_{ik} - x_{jk}\ ?\ 0:1, \quad for\, discrete\, attributes \end{cases}$$

Experimental Results and Analysis

To reduce experimental errors, we ran each experiment in this paper ten times, taking the average value as the final result. In each of the ten experimental runs, the iORCA algorithm randomly selected support points, while the ADPOD algorithm selected support points according to its own rules. To provide a point of comparison, we also forced the PCV algorithm to use the same support points. In other words, we compared whether the iORCA and ADPOD algorithms produced different results when applying the pre-cutoff value under precisely the same conditions.

The experiments initially tested the runtime of the iORCA and ADPOD algorithms, both before and after the application of the PCV algorithm, as a function of the dataset's scale. To facilitate comparative analysis, we also recorded the time required to build an index, but the values were very small, and after applying the PCV algorithm, there was only an average increase of 1.76%. The total

Figure 1. Running times on four datasets for various dataset sizes

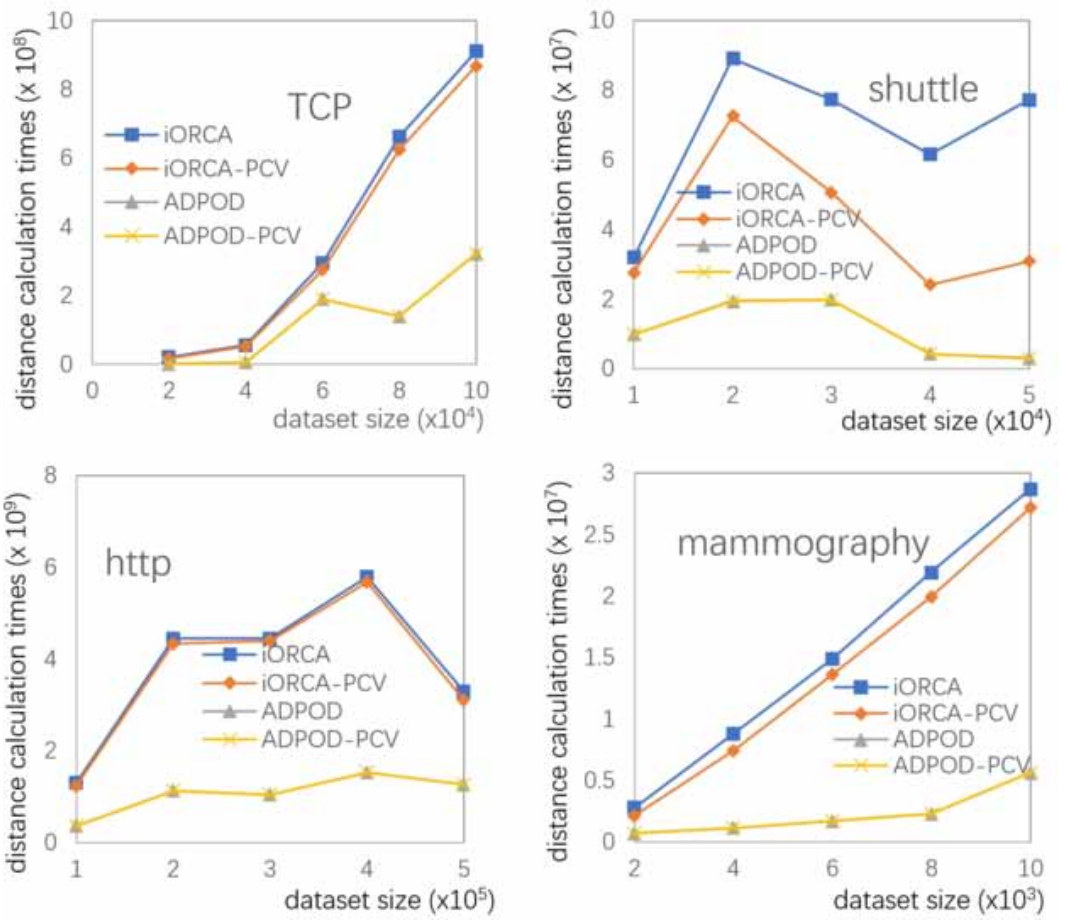


time spent on index construction only accounted for an average of 4.85% of the entire outlier detection time. Therefore, these results are not displayed in the figure.

Despite maintaining the same operating environment during the experiments, variations in operating system conditions and CPU frequency fluctuations can introduce some errors into the runtime measurements. On the other hand, for complex data types or high-dimensional data, distance calculations can be computationally expensive relative to other processing steps. As a result, we also tested the number of distance calculations for the iORCA and ADPOD algorithms, both before and after applying the PCV algorithm, in relation to changes in dataset size.

As expected, the PCV algorithm only computes the pre-cutoff value before applying the iORCA and ADPOD algorithms for outlier detection, and it does not require additional distance calculations. Therefore, when the PCV algorithm is applied alongside the outlier detection algorithms, the number of distance calculations only decreases and cannot increase. In the worst-case scenario, the number of distance calculations remains unchanged. The worst-case scenario refers to a situation in which the pre-cutoff value is too small compared to the outlier degrees of objects in the first data block of the dataset, rendering it ineffective at excluding non-outliers. Even in this worst-case scenario, the additional computational overhead incurred during index construction is minimal. The time and space requirements for outlier detection remain identical to those of the original algorithm since the PCV algorithm modifies the pre-cutoff value from 0 to a value greater than 0 during the detection process.

Figure 2. Distance calculation times on four datasets for various dataset sizes



For the less-favorable experimental results, we can identify several contributing factors:

(1) The size of the first data block as a proportion of the entire dataset was relatively low, and the algorithm did not promptly reach the stopping rule of the iORCA and ADPOD algorithms.
(2) The dataset had a high dimensionality, and the PCV algorithm's calculations were based on distance information that had been mapped to one dimension, which led to a loss of some of the original distance information while saving on computational costs.
(3) The outlier degrees of objects in the first data block were generally higher than the pre-cutoff value, rendering the PCV algorithm ineffective in excluding non-outliers.

However, the PCV algorithm imposes minimal additional runtime overhead regardless of the specific conditions. This additional runtime is negligible compared to the overall runtime for outlier detection. Hence, the PCV algorithm can be safely applied to any outlier detection algorithm because its potential benefits far outweigh the incurred cost.

To gain a deeper understanding of the performance of the PCV algorithm, this study also conducted tests on the detection time for the first data block and the number of non-outliers excluded before and after applying the PCV algorithm to iORCA and ADPOD. As shown in Figure 3 and Table 4, the experiments on the four datasets demonstrated that applying the PCV algorithm resulted in an average reduction of 23.54% in the detection time for the first data block. Moreover, it excluded an average of 365.5 non-outliers, accounting for 36.55% of the data in the first data block.

To further evaluate the performance of the PCV algorithm, we conducted experiments using the iORCA algorithm and the TCP and shuttle datasets as examples. The objective was to examine how the outlier detection time changes with varying parameters k and n before and after applying the PCV algorithm.

As seen in Figure 4, the runtime of the iORCA algorithm does not significantly increase with the increase in parameter k when applying the PCV algorithm, and in some cases, it even decreases. This behavior is related to the distribution of the dataset, as different values of k result in varying pruning efficiencies within the algorithm. For k values greater than 70, the algorithm's runtime

Figure 3. Running times of first data block on four datasets

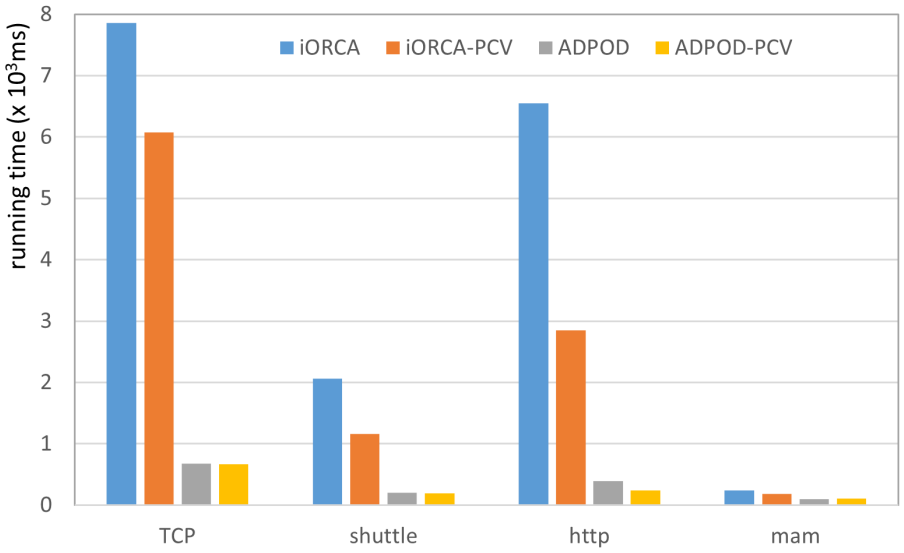
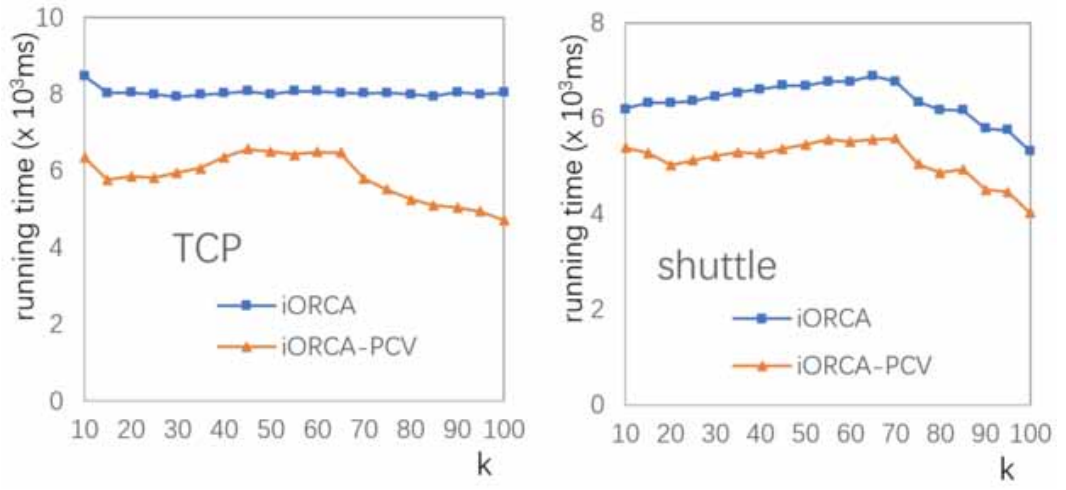


Table 4. Non-Outliers excluded from the first data block

Dataset	iORCA	iORCA-PCV	ADPOD	ADPOD-PCV
TCP	0	269.8	0	147.9
shuttle	0	491.1	0	411.9
http	0	607.6	0	591.9
mam	0	403.4	0	0

Figure 4. Running time with various k of iORCA and iORCA-PCV algorithm



even decreases in the case of the shuttle dataset. Although the iORCA algorithm exhibits relatively stable runtime in the TCP dataset, after applying the PCV algorithm, a significant decrease in runtime occurs for k values greater than 65. Furthermore, it is evident that throughout the entire range of parameter k, the PCV algorithm consistently maintains a substantial reduction in the runtime of the iORCA algorithm.

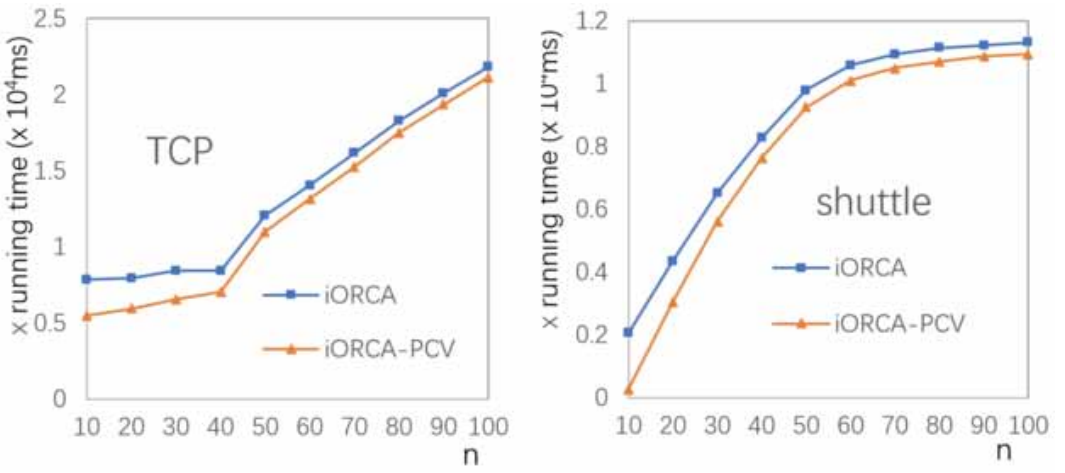
As shown in Figure 5, with the increase in parameter n, the runtime of the iORCA algorithm gradually increases both before and after applying the PCV algorithm. However, with the assistance of the pre-cutoff values provided by the PCV algorithm, the efficiency of excluding non-outliers in the outlier detection algorithm is enhanced. Consequently, this results in reduced runtime compared to not using the PCV algorithm.

SUMMARY AND FUTURE RESEARCH

Summary

Distance-based outlier detection algorithms, especially metric space outlier detection algorithms, represented by the iORCA algorithm, are highly time-consuming when detecting the first data block due to the lack of outlier degree cutoff values that effectively exclude non-outliers and reduce the number of distance calculations. To address this issue, this paper proposed a pre-cutoff value calculation method called PCV, which allows calculating pre-cutoff values using existing index data (such as the distances between a randomly chosen pivot in the iORCA algorithm

Figure 5. Running time with various n of iORCA and iORCA-PCV algorithm



and all objects in the dataset) without additional distance calculations. These pre-cutoff values are then utilized for outlier detection in the first data block. Notably, this algorithm is suitable for accelerating not only the iORCA algorithm but also other distance-based outlier detection algorithms, such as ADPOD. Even if the original algorithm did not create a one-dimensional index, it can still benefit from a small amount of time spent building an index. Experimental results demonstrate the effectiveness of the PCV algorithm, which, when applied, reduces the runtime and the number of distance calculations for the iORCA algorithm by an average of 14.59% and 15.73%, respectively. Even for the newer and more efficient ADPOD algorithm, the PCV algorithm provides reductions of 1.41% and 0.45%, respectively. The additional time spent on PCV calculations is negligible.

Future Research

In future research, we will explore using more pivots and calculating pre-cutoff values based on the distances between these pivots and all objects in the dataset in a multi-dimensional space. Utilizing more distance information in the multi-dimensional space will certainly yield larger pre-cutoff values, which are more effective in excluding non-outliers. However, this approach will require more time for calculations. Finding a balance between the yielded benefits and the associated costs is an important research question. Additionally, the choice of support points also influences the performance of pre-cutoff values, so we will investigate methods for selecting support points to enhance this performance.

AUTHOR NOTE

Dr. Kaide Huang is the corresponding author. This research was supported by NSF-China [grant number 61802063] and Guangdong Basic and Applied Basic Research Foundation [grant number 2021B1515120048]. The authors of this publication declare there are no competing interests.

REFERENCES

Angiulli, F., Basta, S., & Pizzuti, C. (2006). Distance-based detection and prediction of outliers. *IEEE Transactions on Knowledge and Data Engineering*, *18*(2), 145–160. doi:10.1109/TKDE.2006.29

Angiulli, F., & Fassetti, F. (2007). Very efficient mining of distance-based outliers. In *Proceedings of the Sixteenth ACM Conference on Information and Knowledge Management (CIKM '07).* ACM. doi:10.1145/1321440.1321550

Angiulli, F., & Pizzuti, C. (2005). Outlier mining in large high-dimensional data sets. *IEEE Transactions on Knowledge and Data Engineering*, *17*(2), 203–215. doi:10.1109/TKDE.2005.31

Aydın, F. (2023). Boundary-aware local density-based outlier detection. *Information Sciences*, *647*, 119520. doi:10.1016/j.ins.2023.119520

Bay, S. D., & Schwabacher, M. (2003). Mining distance-based outliers in near linear time with randomization and a simple pruning rule. In *Proceedings of the ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM. doi:10.1145/956750.956758

Bhaduri, K., Matthews, B. L., & Giannella, C. R. (2011). Algorithms for speeding up distance-based outlier detection. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM. doi:10.1145/2020408.2020554

Breunig, M. M., Kriegel, H.-P., Ng, R. T., & Sander, J. (2000). LOF: Identifying density-based local outliers. *SIGMOD Record*, *29*(2), 93–104. doi:10.1145/335191.335388

Cao, L., Yang, D., Wang, Q., Yu, Y., Wang, J., & Rundensteiner, E. A. (2014). Scalable distance-based outlier detection over high-volume data streams. In *Proceedings of the IEEE 30th International Conference on Data Engineering (ICDE)*. IEEE. doi:10.1109/ICDE.2014.6816641

Catillo, M., Pecchia, A., & Villano, U. (2023). CPS-GUARD: Intrusion detection for cyber-physical systems and IoT devices using outlier-aware deep autoencoders. *Computers & Security*, *129*, 103210. doi:10.1016/j.cose.2023.103210

Chawla, S., & Gionas, A. (2013). K-means–: A unified approach to clustering and outlier detection. In *Proceedings of the 2013 SIAM International Conference on Data Mining (SDM)*. SIAM. doi:10.1137/1.9781611972832.21

Hilal, W., Gadsden, S. A., & Yawney, J. (2022). Financial fraud: A review of anomaly detection techniques and recent advances. *Expert Systems with Applications*, *193*, 116429. doi:10.1016/j.eswa.2021.116429

Knorr, E. M., & Ng, R. T. (1998). Algorithms for mining distancebased outliers in large datasets. In *Proceedings of the 24th International Conference on Very Large Data Bases*. Morgan Kaufmann Publishers Inc.

Knorr, E. M., Ng, R. T., & Tucakov, V. (2000). Distance-based outliers: Algorithms and applications. *The VLDB Journal—The International Journal on Very Large Data Bases, 8*(3–4), 237–253.

Larson, S., Mahendran, A., Lee, A., Kummerfeld, J. K., Hill, P., Laurenzano, M., Hauswald, J., Tang, L., & Mars, J. (2019). Outlier detection for improved data quality and diversity in dialog systems. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies,* Volume 1. Association for Computational Linguistics. doi:10.18653/v1/N19-1051

Liu, F., Yu, Y., Song, P., Fan, Y., & Tong, X. (2020). Scalable KDE-based top-n local outlier detection over large-scale data streams. *Knowledge-Based Systems*, *204*, 106186. doi:10.1016/j.knosys.2020.106186

Mao, R., Xu, H., Wu, W., Li, J., Li, Y., & Lu, M. (2015). Overcoming the challenge of variety: Big data abstraction, the next evolution of data management for AAL communication systems. *IEEE Communications Magazine*, *53*(1), 42–47. doi:10.1109/MCOM.2015.7010514

Nesa, N., Ghosh, T., & Banerjee, I. (2018). Non-parametric sequence-based learning approach for outlier detection in IoT. *Future Generation Computer Systems*, *82*, 412–421. doi:10.1016/j.future.2017.11.021

Ramaswamy, S., Rastogi, R., & Shim, K. (2000). Efficient algorithms for mining outliers from large data sets. *SIGMOD Record*, *29*(2), 427–438. doi:10.1145/335191.335437

Riahi-Madvar, M., Akbari Azirani, A., Nasersharif, B., & Raahemi, B. (2021). A new density-based subspace selection method using mutual information for high dimensional outlier detection. *Knowledge-Based Systems*, *216*, 106733. doi:10.1016/j.knosys.2020.106733

Schubert, E., Zimek, A., & Kriegel, H.-P. (2015). Fast and scalable outlier detection with approximate nearest neighbor ensembles. In M. Renz, C. Shahabi, X. Zhou, & M. Cheema (Eds.), *Database systems for advanced applications* (pp. 19–36). Springer. doi:10.1007/978-3-319-18123-3_2

Szeto, C.-C., & Hung, E. (2010). Mining outliers with faster cutoff update and space utilization. *Pattern Recognition Letters*, *31*(11), 1292–1301. doi:10.1016/j.patrec.2010.04.002

Tao, Y., Xiao, X., & Zhou, S. (2006). Mining distance-based outliers from large databases in any metric space. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM. doi:10.1145/1150402.1150447

Vu, N. H., & Gopalkrishnan, V. (2009). Efficient pruning schemes for distance-based outlier detection. In W. Buntine, M. Grobelnik, D. Mladenić, & J. Shawe-Taylor (Eds.), *Machine learning and knowledge discovery in databases* (pp. 160–175). Springer. doi:10.1007/978-3-642-04174-7_11

Xu, H., Rong, H., Mao, R., Chen, G., & Shan, Z. (2016). Hilbert Index-based Outlier Detection algorithm in metric space. *International Journal of Grid and High Performance Computing*, *8*(4), 34–54. doi:10.4018/IJGHPC.2016100103

Xu, H., Sun, F., Tan, L., & Huang, W. (2019). *Adaptive Cutoff Distance Based Density Peak Pivot for Metric Space Outlier Detection*. doi:10.1007/978-981-13-7986-4_35

Yang, J., Tan, X., & Rahardja, S. (2023). Outlier detection: How to select k for k-nearest-neighbors-based outlier detectors. *Pattern Recognition Letters*, *174*, 112–117. doi:10.1016/j.patrec.2023.08.020

ENDNOTES

1 https://archive.ics.uci.edu/dataset/130/kdd+cup+1999+data
2 https://archive.ics.uci.edu/dataset/148/statlog+shuttle
3 https://odds.cs.stonybrook.edu/http-kddcup99-dataset/
4 http://odds.cs.stonybrook.edu/mammography-dataset/

Honglong Xu is a lecturer at Foshan University. He received his Ph.D. in Information and Communication Engineering from Shenzhen University. His research interests include data mining, big data and parallel computing.

Zhonghao Liang is a master's degree student majoring in mathematics, with a research focus on outlier detection and parallel computing.

Kaide Huang is an associate professor of computer science in Foshan University. He received his Ph.D. from Sun Yat-sen University, with research interests in artificial intelligence and machine learning.

Guoshun Huang is a professor of artificial intelligence in Foshan University. He received his Ph.D. in computer software and theory from Huazhong University of Science and Technology. His current research interests include neural networks, data mining, rough set, and granular computing.

Yan He is a research assistant of Foshan University. She does research in learning analytics, data mining.