Information Sciences 607 (2022) 901–920

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Efficient density and cluster based incremental outlier detection in data streams

Ali Degirmenci, Omer Karal [*]

Ankara Yildirim Beyazit University, Ayvali Mah. 150, Sok. Etlik-Kecioren Ankara, Turkey

ARTICLE INFO

Article history:
Received 10 October 2021
Received in revised form 1 May 2022
Accepted 4 June 2022
Available online 11 June 2022

Keyword:
LOF
DBSCAN
Outlier detection
Core KNN
Incremental learning
Data stream

ABSTRACT

In this paper, a novel, parameter-free, incremental local density and cluster-based outlier factor (iLDCBOF) method is presented that unifies incremental versions of local outlier factor (LOF) and density-based spatial clustering of applications with noise (DBSCAN) to detect outliers efficiently in data streams. The iLDCBOF has many advanced advantages compared to previously reported iLOF-based studies: (1) it is based on a newly-developed core k-nearest neighbor (CkNN) concept to reliably and scalably detect outliers from data streams and prevent the clustering of outliers; 2) it uses a newly-developed algorithm that automatically adjusts the value of the k (number of neighbors) parameter for different real-time applications; and 3) it uses the Mahalanobis distance metric, so its performance is not affected even for large amounts of data. The iLDCBOF method is well suited for different data stream applications because it requires no distribution assumptions, it is parameterless (determined automatically), and it is easy to implement. ROC-AUC and statistical test analysis results from extensive experiments performed on 16 different real-world datasets showed that the iLDCBOF method significantly outperformed benchmark methods.

1. Introduction

The current ease of access to technology has led to the emergence of many different types of data. In addition to meaningful and useful data, unwanted abnormal data points (outliers) also can be observed in the data that are collected. Identifying outliers is a very difficult task because their sources usually are not known. Since the presence of outliers results in a negative effect on the performance of machine learning algorithms, extensive research has been conducted in recent years to detect outliers quickly and accurately [1,2]. Studies on the detection of outliers can be categorized broadly into three different types, i.e., density-based, distance-based, cluster-based types. Density-based approaches are attractive because they do not require any assumptions about the underlying distribution of the data. In addition, considering the similarity of any data point to its nearest neighbors, the extent to which outliers exist can be determined, and this is referred to as the local outlier factor (LOF) of that data point [3]. Generally, data points with high LOF and low density are considered to be outliers. Therefore, LOF has been a popular approach, and it has inspired many studies to detect local outliers [4–6].

Most of the LOF-based outlier detection methods that have been developed are designed to operate in a batch-learning mode in which all samples are accessible before employing. However, the need to quickly extract reliable information from

[*] Corresponding author.
E-mail address: karal@ybu.edu.tr (O. Karal).

https://doi.org/10.1016/j.ins.2022.06.013
0020-0255/© 2022 Elsevier Inc. All rights reserved.

data streams requires real-time data processing. Three different techniques have been proposed for this purpose to alleviate the limitation in LOF-based approaches [7]. The first technique is the periodic implementation of the LOF algorithm when a certain amount of data is obtained. The major drawback of this approach is that the algorithm cannot access other batches in the dataset, and, due to the lack of this information, some of the normal points may be detected as outliers or outliers may be identified as normal data points. The second technique is to determine the k-distance, local reachability density (lrd) and LOF values, assuming that all previously known samples are training data. Then, the LOFs of incoming samples are computed based on the pre-determined values of the training data. Since the LOF values of the existing data are not updated, changes in the behavior of the data cannot be detected, and new samples cannot be identified correctly. The last and simplest technique is to apply the LOF algorithm from scratch for each incoming instance. Although the performance of this technique is the same as LOF operating in batch mode, the processing time is longer than it is for the other two techniques. To reduce the processing time and simultaneously achieve the same performance as LOF, Pokrajac et al. introduced the instance incremental LOF (iLOF) algorithm, which only updates the LOFs of existing data points influenced by the new incoming point [7]. As a small portion of the dataset is affected by the arrival of a new point, and the time required to compute its LOF value is reduced significantly. This has made the iLOF approach one of the most popular approaches in the literature, and this has resulted in the emergence of several different density-based incremental versions [8,9].

Although the proposed iLOF-based approaches provide significant improvements in detecting outliers in data streams, they still have three serious problems. First, it is difficult to accurately determine the parameter k (number of nearest neighbors) suitable for the distribution of the data. Depending on the k value, a more local or more global outlier can be identified. Usually, local outliers are detected when the k values are small, but global outliers are detected as the value of k increases. Considering that the distribution of data in many real-life applications changes rapidly over time, it is clear that determining the appropriate value of k will be very difficult [10,11]. Second, as the dimensionality of the data increases, the ability to detect outliers decreases gradually because each feature in high dimensional data usually has a different weighting effect on the results, and it is difficult to recognize outliers when they appear sequentially. The iLOF-based approaches were developed based on the assumption that inliers are located in the high-density regions while outliers are in the low-density regions [7]. However, when outliers occur sequentially, the density of the region that contains outliers inevitably increases, resulting in the clustering of the outliers. In this situation, it is no longer possible to detect outliers in the data stream.

Recently, cluster-based outlier detection algorithms have been proposed to address clusters of outliers. The density-based spatial clustering of applications with noise (DBSCAN) approach is quite remarkable in that it can identify and cluster noisy samples (outliers) in the dataset [12]. The ability of the DBSCAN approach to recognize clusters of arbitrary shapes gives it a significant advantage. Only two input parameters are required for it to distinguish high-density regions from low-density regions, i.e., radius (Eps) and the minimum number of points (minPts). In the past, finding the proper values for these parameters was difficult, but now they can be determined automatically using all instances of the dataset [13–15]. This facility of the DBSCAN algorithm only applies to batch mode learning; it cannot be used directly in data stream applications. However, DBSCAN's minPts parameter performs a similar task to iLOF's k parameter, which specifies the density of the processing data point. Automatically determining this parameter for the data stream can solve iLOF's most important problem. Most of the studies in the literature have chosen only one of these two methods for the detection of outliers and made improvements [16,17]. However, combining these two methods will reduce the drawbacks of each approach when applied separately and will reveal different benefits. In this paper, a novel, parameter-free, incremental local density and cluster-based outlier factor (iLDCBOF) that unifies the incremental LOF and DBSCAN is proposed to detect outliers efficiently in data streams. Extensive experiments are performed on 16 different real-world datasets to measure the effectiveness of the proposed system.

The main contributions of this paper are summarized as follows:

1. A new incremental clustering and density-based outlier detection method is proposed that simultaneously performs both clustering and outlier detection.
2. To the best of our knowledge, this is the first study to combine the concepts of incremental DBSCAN (iDBSCAN) and iLOF to detect outliers from streaming data.
3. To minimize the negative effects of the selection of parameters, iLDCBOF automatically adjusts its own hyperparameters for different, real-time applications.
4. To detect outliers from data streams and prevent their clustering, a newly-developed, core kNN (CkNN) concept is introduced.
5. The incremental Mahalanobis metric is used in all distance computations to reduce the impact of the data dimensions in both iLOF and iDBSCAN.

The structure of this paper is organized as follows. In Section 2, a literature review regarding to the different type of incremental outlier detection methods and then their shortcomings are briefly discussed. Section 3 provides information on inspired methods iLOF and iDBSCAN and then iLDCBOF method is presented in detail in the following section. In Section 5, experimental results on real-world datasets are reported and performance of the iLDCBOF is compared with the other unsupervised outlier detection methods. In Section 6, concluding remarks and future work is given.

2. Related work

There are many articles in the literature concerning on the detection of outliers, but most of them focus on static environments where all of the data points are accessible. In dynamic environments where large volumes of data are generated rapidly, the existing approaches to outlier detection can be divided into three categories, i.e., (1) distance-based approaches, (2) density-based approaches, and (3) cluster-based approaches.

Distance-based outlier detection approaches measure the outlierness degree based on the relationship of a data point to its nearest neighbors. If the distance between the point and its nearest neighbors exceeds some threshold value, it is called an outlier. Kontaki et al. proposed an outlier detection method using sliding windows depending on whether the data streams contain at least k points at distance R [18]. Since sensitive to user defined parameter values, it is difficult to obtain the desired outlier detection results without prior knowledge of the data. Yao et al. combines kNN, reverse kNN (RkNN), and shared kNN (SkNN) of the sample called composite kNN (CNN) to measure the outlierness degree in the data stream [19]. Including more nearest neighbor (NN) concepts to calculate the outlierness degree improves performance but leads to increased computation time. Bah et al. presented a hybrid distance-based outlier detection method by combining the strength of MCOD and Thresh_LEAP methods [20]. However, it suffers from the complexity that differentiates between strong and trivial inliers. Zhu et al. developed a kNN based outlier detection framework for IoT streaming data [21]. Outliers are determined by k skyband based method. Still, memory consumption and processing time are highly dependent on window size and k parameters. Dai and Ding proposed an online average weighted distance to detect outliers in wireless sensor networks [22]. Two different models are presented to detect outlier data: traditional Top-n and adaptive Top-n. If there are more outliers than the predefined value, the performance of traditional Top-n decreases. On the other hand, the performance of adaptive Top-n is poor in cases where there are many outliers or they are overly distributed.

In density-based outlier detection approaches, data points with lower density than neighboring points are defined as outliers. iLOF is a well-known density-based algorithm for the detection of local outliers in dynamic data [7]. However, iLOF suffers from high process time, as well as large memory requirements. Salehi et al. suggested a MiLOF to reduce the memory requirement of the LOF by summarizing the processed data points [23]. While process time and memory requirements are reduced, it mostly underperforms iLOF at area under the receiver operating characteristic curve (ROC-AUC) scores in real-world datasets. Na et al. proposed a window based incremental LOF named DiLOF to reduce memory requirements and avoid detection of the outlier sequence [24]. Although the ROC-AUC score is greater than MiLOF, it still cannot exceed the iLOF and the window size selection affects the performance of the algorithm. Huang et al. proposed another density summarizing and window based incremental LOF method called TADiLOF [25]. It mostly performs better than MiLOF and DiLOF but requires more input parameters. Chen et al. presented a CELOF method to reduce the high dimensionality and memory problems of the LOF by utilizing entropy and weighting distance [26]. However, CELOF's performance is highly dependent on tuning hyperparameters. Gao et al. developed a cube-based iLOF (CB-iLOF) method to improve the time and additional memory load of the iLOF [27]. Compared to the iLOF, CB-iLOF provides significant acceleration and comparable accuracy over time while using less memory with lower runtime. It has been observed that the performance of CB-iLOF is affected by cube size in terms of accuracy, runtime, memory consumption and duration, but parameter k has little effect other than time.

Cluster-based outlier detection approaches generally follow a two-step procedure: clustering data and treating data points not belonging to any cluster as outliers. Li et al. proposed a clustering-based anomaly detector regarding the amplitude and shape of multivariate sequences [28]. They computed the anomaly score using sliding window sub-sampling, extended FCM clustering, and reconstruction criteria. Then, particle swarm optimization was employed to improve the performance of the algorithm. While this approach has the benefit of being less dependent on previous training data, it has high time and space complexity. Habeeb et al. presented a cluster-based real-time outlier detection framework called streaming sliding window local outlier factor coreset clustering (SSWLOFCC) algorithm using the sliding window [29]. It detects outliers using LOF, principal component analysis, agglomerative clustering, and k-means clustering. While SSWLOFCC performs well on simulated network data, it has not been tested on different real-world datasets. Yin et al. developed an online clustering method for outlier detection with synopsis data structure of the clusters [30]. Its performance is affected from the threshold value and initial clusters. Cluster-based outlier detection approaches require a large number of parameters to detect outliers from data streams, which are ideally difficult to tune, especially in high-dimensional real-world data.

As can be seen from the existing methods proposed in different categories in the literature, hyperparameters are the most important factors that affect the performance of detecting outliers in data streams. Thus, the success of existing methods depends significantly on the proper tuning of hyperparameters. Currently, the parameters of the DBSCAN approach can be determined automatically using all instances of the dataset [13,14]. If this property of the DBSCAN can be applied to data streams, the most important problem of iLOF, i.e., finding the appropriate k value for different data streams, will be solved because it has a parameter that is similar to that of iDBSCAN. To this end, a novel, parameter-free iLDCBOF that unifies iLOF and iDBSCAN is proposed to detect outliers efficiently in data streams.

3. Background

Considering that the proposed method is motivated mainly by the LOF [3] and DBSCAN [12] algorithms, we will introduce these two algorithms first and then introduce their incremental versions so readers can acquire a comprehensive understanding of the proposed approach.

3.1. Incremental Mahalanobis distance metric

Distance metrics help quantify the similarity between samples in the data set. It is very important to use a proper similarity metric as it significantly affects the performance of machine learning methods. Many distance metrics have been proposed in the literature. The most widely known of which is the Euclidean metric because it is easy to implement and conceptualize. In Euclidean distance, each dimension of the feature vectors is treated equally and correlations among them are ignored. However, in real-life data, the units of measurement in the features are not equal and correlations may occur between the features. Therefore, before computing the Euclidean distance, the data should be normalized to avoid skewness based on the features. However, in data stream applications, normalization is not preferred because future values may exceed the maximum and minimum values and lead to erroneous results. Moreover, the performance of Euclidean distance is lower in high-dimensional datasets and sensitiveness to the geometrical shape of clusters. Mahalanobis distance can be used to reduce these shortcomings of Euclidean distance and to measure similarity between samples. Since it considers unequal variances as well as correlations between features, it will adequately evaluate the distance by assigning different weights or importance factors to the features of data. It manages the correlation between features and is unaffected by the units of measure in the features [31,32]. In the proposed iLDCBOF, Mahalanobis distance is chosen instead of Euclidean due to these advantages in high-dimensional data.

The pairwise Mahalanobis distance between samples x_p and x_q is computed as [33]:

$$dist(x_q, x_p) = \sqrt{(x_q - x_p)^T \cdot C^{-1} \cdot (x_q - x_p)} \tag{1}$$

where, **C** is the covariance matrix. Since incremental learning is implemented in the study, it must be updated with each incoming data. An incremental version of the Mahalanobis distance is presented by Han et al. [34].

Assume $D = \{x_1, x_2, \ldots, x_n, \ldots\}$ is infinite data stream and n is the number of samples processed so far. In the incremental learning, mean and covariance matrix are updated for the new coming sample (x_{n+1}) is defined by

$$\hat{\mu}_{n+1} = \frac{\sum_{i=1}^{n} x_i + \hat{x}_{n+1}}{n+1} \tag{2}$$

$$C_{n+1} = \frac{\sum_{i=1}^{n} x_i^T x_i + \hat{x}_{n+1}^T \hat{x}_{n+1} - \hat{\mu}_{n+1}^T \hat{\mu}_{n+1}}{n+1} \tag{3}$$

In these Equations, $\hat{x}_{n+1}$ is the weighted sample, $\hat{\mu}_{n+1}$ is the incremental mean, and $\hat{C}_{n+1}$ is the covariance matrix based on the new incoming sample.

3.2. LOF and iLOF

LOF takes two user defined parameters which are the number of nearest neighbors (k) and the distance metric. Then, it computes the local outlierness degree of each point based on them. Some definitions and concepts necessary for a clear understanding of the LOF algorithm are presented as follows [3].

Definition 1. (k-distance of a point p) is the distance between the point p and q with the user defined positive integer k in the dataset D. It is demonstrated as $k - distance(p)$ and satisfies the following conditions:

1. for at least k points $q' \in D \setminus \{p\}$ it holds that $dist(p, q') \leqslant dist(p, q)$, and
2. for at most k-1 points $q' \in D \setminus \{p\}$ it holds that $dist(p, q') < dist(p, q)$

Definition 2. ($k - distance$ neighborhood of a point p) is denoted as $N_{k-dist(p)}$, and also defined as k-nearest neighbors of p. It consists of points that are not greater than the k-distance of the point p and defined as

$$N_{k-dist(p)}(p) = \{q \in D \setminus \{p\} | dist(p, q) \leqslant k - distance(p)\} \tag{4}$$

Definition 3. (reachability distance of a point p with respect to point q) is the maximum of the $k - distance(q)$ and the distance between the p and q points. It is given by

$$reach - dist_k(p, q) = \max\{k - distance(q), dist(p, q)\} \tag{5}$$

Definition 4. (local reachability density of a point p) is equals to the inverse of the mean k-nearest neighbors' reachability distance, and it is defined as

$$lrd_k(p) = \frac{1}{\left(\frac{\sum_{q \in N_k(p)} reach\text{-}dist_k(p,q)}{|N_k(p)|}\right)} \tag{6}$$

where, $N_k(p)$ is the set of k nearest neighbors of point p.

Definition 5. (local outlier factor of a point p) is given by

$$LOF_k(p) = \frac{\sum_{q \in N_k(p)} \frac{lrd_k(q)}{lrd_k(p)}}{|N_k(p)|} \tag{7}$$

LOF computes the ratio between average lrd of the kNN points and the lrd of the processed data point. In other words, it is a measure of the closeness of the point to the NN.

Incremental version of LOF is a density-based method so a limited number of data are affected by the incoming data point. Based on the defined distance measure, the kNN of the incoming point are determined, then reach-dist, lrd, and LOF are computed. RkNN of the data points is determined from the previously processed point in the dataset and influenced data points are figured out. Subsequently, these data points' reach-dist, lrd, and LOFs are updated. Thus, instead of revising all previously processed data in the system, only a small portion of it is updated, achieving the same performance as the batch mode LOF.

3.3. DBSCAN and iDBSCAN

DBSCAN is a density-based clustering algorithm that is widely preferred in many real-world applications due to its simplicity, efficiency, and robustness against outliers. It takes two input parameters to separate high density regions from low densities: Eps and minPts. To understand the DBSCAN algorithm clearly, definitions and concepts are introduced as below [12].

Definition 6. The Eps-neighborhood $(N_\varepsilon(p))$ of the data point p contains the data points which are in the range of Eps distance and defined by $\{q \in D \,|\, dist(p,q) \leqslant Eps\}$. Where, D is the dataset and $dist(p,q)$ is the distance between points p and q.

Definition 7. A data point p is a core point if it has more than or equal to minPts points in the Eps distance. In Fig. 1, black circles represent the Eps distance of the core points.

Definition 8. A data point p is a noise if it neither contains minPts points in the Eps neighborhood nor is Eps neighborhood of the core point. Dashed red circles represent the Eps distance of the noise points, as shown in Fig. 1.

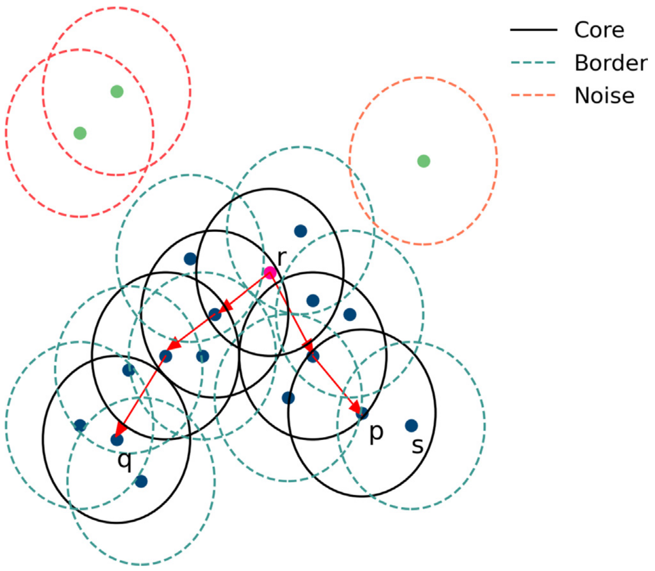


Fig. 1. Illustration of DBSCAN definitions with respect to the minPts = 4, Eps = 1.

Definition 9. A data point p is a border point if it contains fewer than minPts points in the Eps neighborhood and it is within the Eps neighborhood of the any core point. Eps distance of the border points are demonstrated with dashed green circles in Fig. 1.

Definition 10. A data point p is directly density-reachable from point q if p is within the Eps neighborhood of q, and q satisfies the core point condition. Directly density reachable is not symmetric for border and core objects while it is symmetric between core points. In Fig. 1, point s is directly density from p, while p is not direct density reachable from s. Because, p satisfies the core point condition while s is a border point.

Definition 11. A data point p is density-reachable from the point q with respect to Eps and minPts, if there is a chain of points $p_1, \ldots p_n, p_1 = q$ and $p_n = p$ such that p_{i+1} is directly density-reachable from p_i with respect to Eps and minPts.

Definition 12. A data point p is density connected to a point q with respect to Eps and minPts, if there is a point r in the dataset in which both p and q are density-reachable from r with respect to Eps and minPts. In Fig. 1, p and q are density connected, because p and q are density reachable from the point r.

Definition 13. A cluster C which is a subset of dataset D, has to satisfy maximality and connectivity conditions. These are defined as

1. Maximality: $\forall p, q$: *if* $p \in C$ and q is density-reachable from p with respect to Eps and minPts, then $q \in C$
2. Connectivity: $\forall p, q \in C$: p is density-connected to q with respect to Eps and minPts.

With the concepts and definitions described above, DBSCAN divides the dataset into three different categories as core, border, and noise. The algorithm can start with a random query point selected from the dataset, and then the Eps neighborhood of that point is found. A query point is defined as a core point if it contains at least minPts in the Eps neighborhood. Thus, the core point and its Eps neighborhood points construct a new cluster. Then, it is checked whether the points in the neighborhood of the core point meet the conditions of being a core point, respectively. If the core point condition is satisfied, the cluster expands. If not, and the query point is not in Eps neighborhood of any core point, it is marked as noise. These processes continue until all points in the dataset have been checked.

Instance incremental version of the DBSCAN algorithm is first proposed by Ester et al. [35]. Four different situations can be observed with the arrival of the new point. With the arrival of the new point, four different situations can be observed: (i) if the core condition is not met it may be marked as noise; (ii) may lead to the formation of a new cluster; (iii) may be assigned to one of the existing clusters; (iv) may cause the merger of two different clusters.

4. System overview

4.1. Problem of existing approaches

LOF is a density-based approach that computes the outlierness degree for each data point in the dataset to identify the outliers. The iLOF uses an insertion algorithm to compute the LOF of each incoming data point and update the LOFs of the affected points. The parameter k, which determines the number of points that can be affected by the incoming data point, has a significant effect on the result. When the parameter k is not selected properly, the algorithm may treat a large number of inlier points as outliers or a portion of outliers as inliers. Many methods have been proposed in the literature for the proper selection of the parameter k [10,11]. While these methods increase the accuracy of the algorithm to some extent, they also require more computation time and the associated cost.

The essence of LOF is to identify the level of the anomaly of each sample based on the relative density of the samples. This indicates whether the samples are distributed over high-density regions. Since each variable in high-dimensional data usually has a different weighting effect on the results, the computational efficiency will decrease gradually as the dimensionality of the data increases. In addition, when outliers occur consecutively, the density of the region of outliers inevitably increases, resulting in a cluster of outliers that no longer can be detected in the data stream.

The iLOF limitations mentioned above are overcome with the iLDCBOF method and the CkNN metric proposed in the study.

4.2. Nearest neighbor concepts and core k nearest neighbors (CkNN)

In density-based outlier detection approaches, the degree of outlierness is measured based on the similarity of a data point to its NN. The definition of NN has a significant role in local outlier detection algorithms because it directly affects performance. The most extensively used NN-based approach is kNN, which finds the k nearest samples with a given distance metric. However, the kNN approach has some disadvantages that affect its performance, such as the difficulty of correctly determining the parameter k, the decrease in performance as the size of the data increases, and its being affected by noisy and outlier data. For these reasons, various NN concepts have been proposed in the literature, e.g., RkNN, SkNN, and mutualkNN (MkNN), and they also are used in the detection of outliers [11]. In the RkNN approach, a point q is an RkNN of query point p if query point p is one of the kNNs of q. The SkNNs of a data point p are those points that share at least one NN with the point p. It also redefines the similarity between the points by computing number of NN the two points share. Similarity between the two points is measured by number of NN they share. In the MkNN approach, both points should be in the kNN of each other. RkNN, SkNN, and MkNN are scale-free neighbor concepts compared to kNN, the number of NN is not fixed to any value, so it can be none or more depending on the density of the point. The formal definitions of the aforementioned kNN, RkNN, SkNN, and MkNN approaches are given in Table 1.

The NN-based approaches mentioned above provide useful information about data points, but kNN is the only one that can be used alone to determine the degree of an outlier because for some points, especially the most likely outlier points, an empty set may return. To solve this problem and improve the performance of the algorithm, combining one or more NN concepts is used in the computation of the degree of outlierness [19,36]. While finding the NN of the query point, the aforementioned NN approaches do not use the cluster information of the dataset. Yet, using cluster information in computing the outlierness degree will increase the robustness of the algorithm. In this regard, a new NN concept that we call CkNN is proposed in this study. In CkNN, the k closest core points of the DBSCAN to the query point constitute the neighborhood set, where k is the number of nearest core points as determined by the user. If the query point does not belong to any cluster and/or there are not enough k core points in the cluster, the core points in the closest cluster that satisfy these conditions are considered. The description of CkNN is given in the last row of Table 1, where p is the query point and C_i is the i^{th} cluster.

Illustrative example of the NN approaches and the proposed CkNN is given in Fig. 2. The synthetic 2D dataset in Fig. 2a includes a local outlier (q_1), inlier cluster (C_1), and outlier cluster (C_2). In Fig. 2a, kNN, RkNN, SkNN, and MkNN of the two query points (q_1 and q_2) are plotted with different colored circles (for $k = 3$). When only the kNN of the local outlier point q_1 is considered in the outlierness degree computation (especially in the LOF), it will be similar to the points in cluster C_1 and it cannot be detected. On the other hand, exploiting more NN approaches along with kNN can make it even easier to distinguish local outliers that may exist near dense regions such as q_1. However, even if all NN approaches are used together, they may fail to detect the outlier, as can be seen from the point q_2 in the small outlier cluster C_2. In Fig. 2b, the proposed CkNN metric provides robust distance measures in distinguishing points that belong to both local outliers and small outliers cluster. For example, according to the DBSCAN (minPts = 4) clustering result, the blue points correspond to the core, the green points to the border, and the magenta points to the noise. Here, the CkNN of query points q_1 and q_2 are shown with red arrows ($k = 3$). Accordingly, the CkNN of the q_1 is larger than other NN approaches, hence the outlierness degree will be higher. On the other hand, since there are not enough core points in the small outlier cluster C_2, the CkNN of query point q_2 is determined from the core points in C_1. Therefore, the outlierness degree of points in C_2 (including q_2) will be much higher compared to inliers (C_1).

4.3. Local density and clustering based outlier factor (LDCBOF) with CkNN

In the proposed iLDCBOF method, CkNN is used to compute the local outlierness degree of the samples instead of kNN, so definitions of the LOF needs to be changed as follows.

- Determine core k-distance:

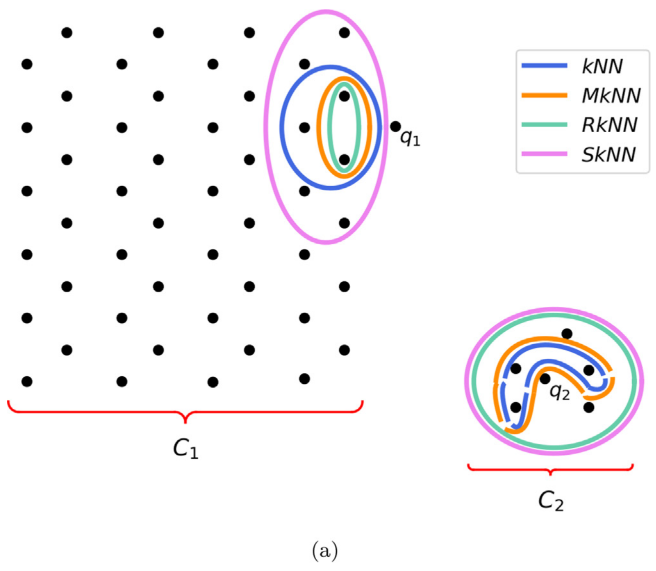
Core k-distance of an instance p is the distance between the p and q in the dataset D, which comply the following conditions:

1. at minimum k instances $q' \in D \setminus \{p\}$ it holds that $dist(p,q') \leqslant dist(p,q) \wedge q' \in C_i$ and q' is Core,
2. for at most $k-1$ instances $q' \in D \setminus \{p\}$ it holds that $dist(p,q') < dist(p,q) \wedge q' \in C_i$ and q' is Core

where, C_i is the i^{th} cluster.

Table 1
Nearest Neighbor concepts in Outlier Detection.

Concept	Definition
k Nearest Neighbors	$kNN(p) = \{r \in D \mid \forall q \in D : dist(p,r) \leqslant dist(p,q)\}$
Reverse k Nearest Neighbors	$RkNN(p) = \{q \in D \mid p \in kNN(q)\}$
Shared k Nearest Neighbors	$SkNN(p,q) = \{kNN(p) \cap kNN(q)\}$
Mutual k Nearest Neighbors	$MkNN(p) = \{q \in D \mid q \in kNN(p) \wedge p \in kNN(q)\}$
Core k Nearest Neighbors	$CkNN(p) = \{r \in D \mid \forall q \in D : dist(p,r) \leqslant dist(p,q) \wedge q \in C_i\}$



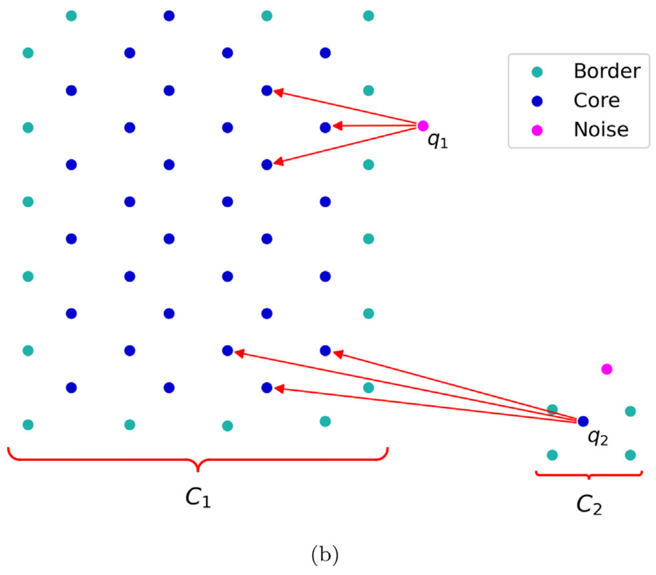


Fig. 2. Local neighbors (a) existing NN concepts (b) CkNN.

- Determine core reachability distance:

Core reachability distance of an instance p with respect to instance q is the maximum of the $core-k-distance(q)$ and the distance between the p and q, it is defined by

$$core-reach-dist(p,q) = \max\{core-k-\text{distance}(q), dist(p,q)\} \tag{8}$$

- Determine core local reachability density

Core local reachability density of an instance p is denoted as $core-lrd(p)$ and given by

$$core-lrd(p) = \frac{1}{\left(\frac{\sum_{q\in CkNN(p)} core-reach-dist(p,q)}{|CkNN(p)|}\right)} \tag{9}$$

- Determine LDCBOF

LDCBOF of a sample p is computed as

$$LDCBOF(p) = \frac{\sum_{q \in CkNN(p)} \frac{core\text{-}lrd(q)}{core\text{-}lrd(p)}}{|CkNN(p)|} \tag{10}$$

LDCBOF computes the outlierness degree by comparing the average density of the CkNN and query point. Accordingly, the LDCBOF value of the query point is smaller at the points close to the inlier clusters (high density regions), while it goes up as the distance to the inlier clusters (core points) rises. The bigger the LDCBOF, the higher the probability of an outlier. In brief, LDCBOF computes the outlierness degree based on the closeness between the query point and CkNN.

4.4. Automatic determination of Eps and minPts parameters

The proper determination of parameters in the proposed iLDCBOF approach is critical as in other unsupervised methods. A new approach has been developed based on the knowledge gained from previous experience on DBSCAN. The DBSCAN algorithm has two user-defined parameters, i.e., Eps and minPts, and they have a significant impact on the performance of the algorithm. In recent studies, they can be determined automatically from the thinnest group (excluding noise and outlier clusters) in the static dataset [12–15]. Since the DBSCAN algorithm will be applied in incremental mode in the study, these suggested methods cannot be used directly. Therefore, the mentioned parameters are determined automatically as follows.

For automatic computation of the Eps parameter, the k-distance graph of the dataset usually is drawn, and the elbow point is determined from this graph either by inspection or by knee/elbow detection methods. However, since the number

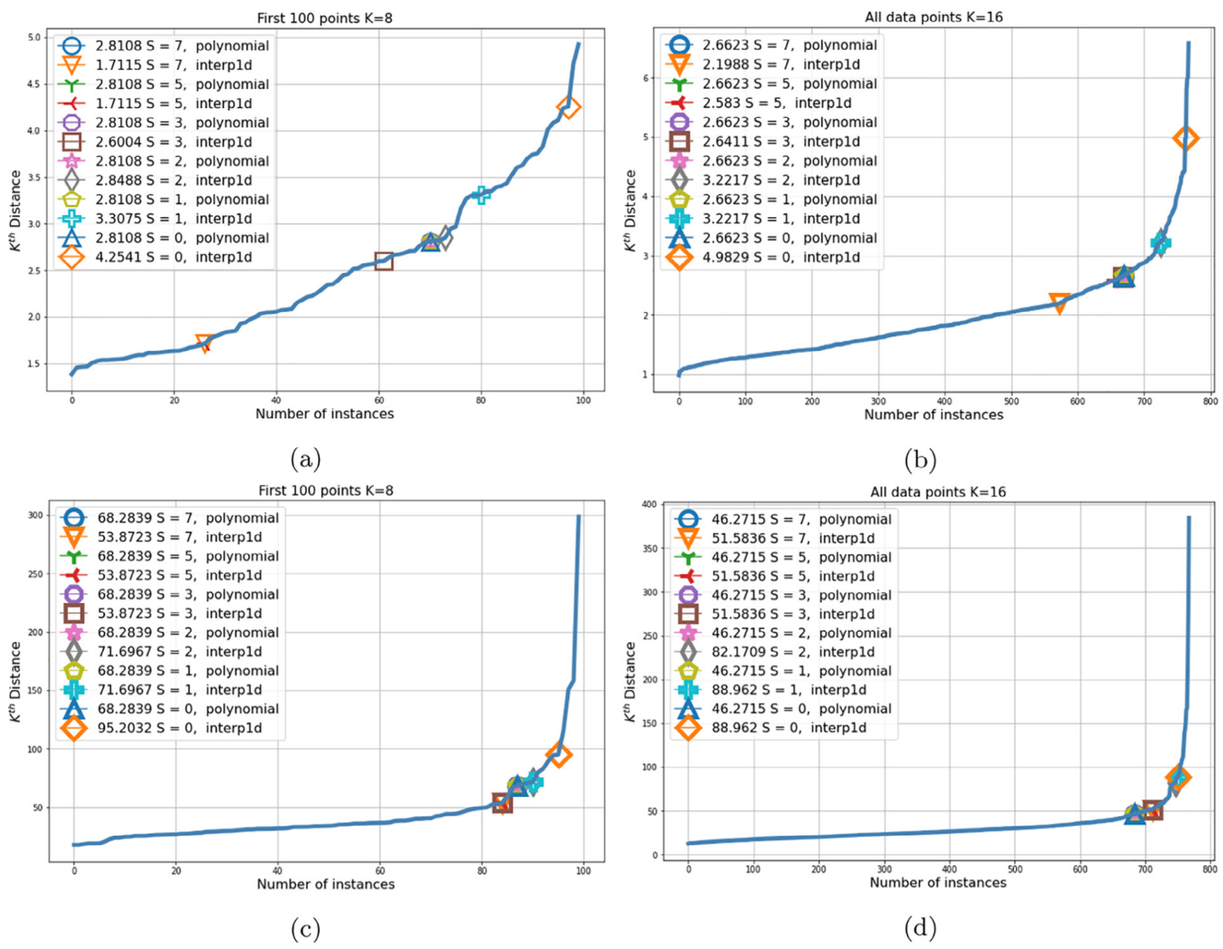


Fig. 3. Elbow point for the Pima (768x8) dataset (a) Mahalanobis distance k = 8, (b) Mahalanobis distance k = 16, (c) Euclidean distance k = 8, (d) Euclidean distance k = 16.

of instances is not known in advance in any real-time application, it is not suitable for use in streaming data. Moreover, the Eps distance must be adjusted adaptively to different data streams. Therefore, algorithms that can compute the minimum number of instances to obtain remarkable results cannot be applied. However, statisticians have concluded that at least 100 samples are required to produce meaningful results [37]. With this in mind, the first 100 data points are used in this study to compute the Eps parameter of the iLDCBOF.

To automatically determine the Eps parameter in a data stream, the k^{th} nearest neighbor distance of the first 100 data points is computed by the Mahalanobis or Euclidean metric and sorted in ascending order. Then, the elbow point is determined from the graph obtained using the "kneedle" algorithm, as it performs well on noisy data and also can work on discrete datasets [38]. From the results of the k-distance plot on real-world Pima dataset (768×8), it can be observed that the first 100 points (k = feature_num) and all points (k = 2*feature_num) in the dataset give almost the same Eps values with respect to the Mahalanobis metric, as shown in Figs. 3a and 3b. The Eps values obtained by using the Mahalanobis distance metric in the polynomial interpolation technique with different S values are 2.8108 (first 100 points), and 2.6623 (all data points). The difference between them is only 0.1485. However, the Euclidean metric does not work properly on some datasets. For example, as can be observed in Figs. 3c and 3d, the Eps values are quite different from each other in the graphical representation of the elbow point for all samples in the Pima dataset and for the first 100 samples. The Eps values obtained by using the Euclidean distance metric in the polynomial interpolation technique with different S values are 68.2839 (first 100 points), and 46.2715 (all data points), respectively. The difference between them is 22.0124, which is quite large compared to Mahalanobis.

In addition, Fig. 3 shows that the polynomial interpolation methods give the same performance under different S values compared to the one-dimensional interpolation method. However, larger S values also increase the processing time. Considering stability, effectiveness, the author's recommendations [38], and the observed results, it is appropriate to use $S = 1$ and the polynomial interpolation technique to find the elbow point in Fig. 3.

The minPts parameter is set to twice the number of features (*feature_num*) in the dataset (i.e., $minPts = (2 * feature\_num)$). In the literature, this property is recommended for static datasets, and the Eps parameter is determined accordingly [16]; however, it cannot be applied directly to dynamic data streams because not all of the data is accessible. Thus, determining the NN of the query point is a time-consuming process in incremental learning. Both iDBSCAN (for clustering) and iLOF (for outlier detection) utilize a similar NN concept, although they were proposed for different purposes. The proposed iLDCBOF method exploits this common NN concept in the study to reduce the processing time. For this purpose, minPts is set to the *feature_num* in the first 100 data points and is limited to 15 if the *feature_num* is greater than 15. From the 101st incoming data point, the minPts value is set to $2 * feature\_num$ for data streams no more than 15, otherwise 30. Accordingly, the parameter k in LDCBOF is set to minPts-1, and the strategy for determining this is presented in Section 5.6.

4.5. Proposed iLDCBOF in detail

The iLDCBOF method consists of three stages and its pseudo code is depicted in Algorithm 1.

Algorithm 1: incremental Local Density and Clustering Based Outlier Factor

```
Input: infinite data stream D = $\{x_1, x_2, \ldots, x_n, \ldots\}$
  $x_i \in R^N$, where, $N$ corresponds to the dimensionality of
  dataset D
Output: LDCBOF value, Cluster label $C = \{C_1, C_2, \ldots, C_n\}$
1   D_sub = take first 100 instances in D
2   if num_features ≤15 then
3     minPts = 2 * num_features
4   else
5     minPts = 30
6   end if
7   Eps = kneedle(D_sub, sorted_distances)
8   Run DBSCAN on the D_sub with minPts and Eps
9   Compute LDCBOF value of the samples in the D_sub
10  for each query instance $x_q$ do
11    Compute $kNN(x_q)$ and $k - distance(x_q)$
12    Find the distance $d$ to the closest cluster $C_p$ in $C$ such that
    $dist(x_q, d)$ is the smallest
13    If $(dist(x_q, d)$ is minimum) && $(x_q \leqslant Eps)$ && $(size(C_p \geqslant minPts)$ then
14      $C_p = C_p \cup x_q$
```

* (*continued*)

Algorithm 1: incremental Local Density and Clustering Based Outlier Factor

```
15        else-if $dist(x_q \neq \text{minimum})$ && $(x_q > Eps)$ && $(size(C_p < \text{minPts}))$ then
16          $x_q \rightarrow Outlier(O_i)$
17        else-if $Count(O_i) \geqslant \text{minPts}$ then
18          Form new cluster
19        $L_{Update}$ = Label the samples that change their status to core
20        end if
21        $S_{Update}$ = $k$ Reverse Nearest Neighbors $x_q$
22        for $\forall x_i \in kNN(x_q)$ do
23          Compute $core-reach-dist(x_q, x_i)$ using Eq. 8
24        end for
25        for $\forall x_i \in S_{Update}$ and $\forall x_j \in kNN(x_i)$ do
26          Update $core-k-distance(x_i)$
27          Update $core-reach-dist(x_j, x_i)$ using Eq. 8
28          if $x_i \in kNN(x_j)$ then
29            $S_{Update} = S_{Update} \cup \{x_j\}$
30          end if
31        end for
32        Compute $core-lrd(x_i)$ using Eq. 9
33        for $\forall x_i \in S_{Update}$ do
34          Update $core-lrd(x_i)$ using Eq. 9
35          Update $LDCBOF(RkNN(x_i))$ using Eq. 10
36        end for
37        Compute $LDCBOF(x_q)$ using Eq. 10
38      end for
```

The first stage (lines 1–7) is to automatically determine the hyperparameters.

- The first 100 data points from the data stream are accumulated and then the minPts and Eps parameters are automatically determined with respect to these first 100 points.

The second stage (lines 8–20) is clustering the dataset with the iDBSCAN method and computation of the first 100 data points LDCBOF value.

- The DBSCAN algorithm is executed for the first 100 data points. According to the DBSCAN clustering results, the CkNN and LDCBOF of the first 100 data are computed.
- The incremental learning process starts with the incoming 101st data point and its distance from the existing data in the system is determined.
- The affected data are updated together with the incoming data point by applying the iDBSCAN algorithm. In addition, data points whose status changes from “border”/“noise” to “core” are specified and stored in the table.

The third stage (lines 21–37) is to compute the LDCBOF value of the incoming point and update the affected data.

- To achieve the same result in batch and incremental mode, the data affected by the incoming data point must be identified and updated, which are the kNN, RkNN of the incoming point and the points that change their status in the iDBSCAN.
- “core-k-distance, core-reach-dist, core-lrd” values of the incoming point is computed. If the affected point(s) are arised, their values are also updated.
- Finally, the LDCBOF of the incoming point is computed. If there are any affected data, their LDCBOFs are also updated.

Performance comparison of the proposed iLDCBOF algorithm and the base methods is demonstrated in Fig. 4. As shown in Fig. 4a, the 2D dataset contains inlier clusters (C_1, C_2), small outlier clusters (C_3, C_4), local outliers (p_1, p_2), and global outliers (p_3, p_4). The result of the DBSCAN algorithm with parameters of minPts = 4, Eps = 1.5 is shown in Fig. 4b. DBSCAN algorithm perfectly label both local outlier points (p_1, p_2) and global outlier points (p_3, p_4). However, it fails to detect outlier cluster C_3 and C_4. To overcome this, minPts or Eps can be increased, but in this case the inlier points in C_1 and C_2 can be labeled as

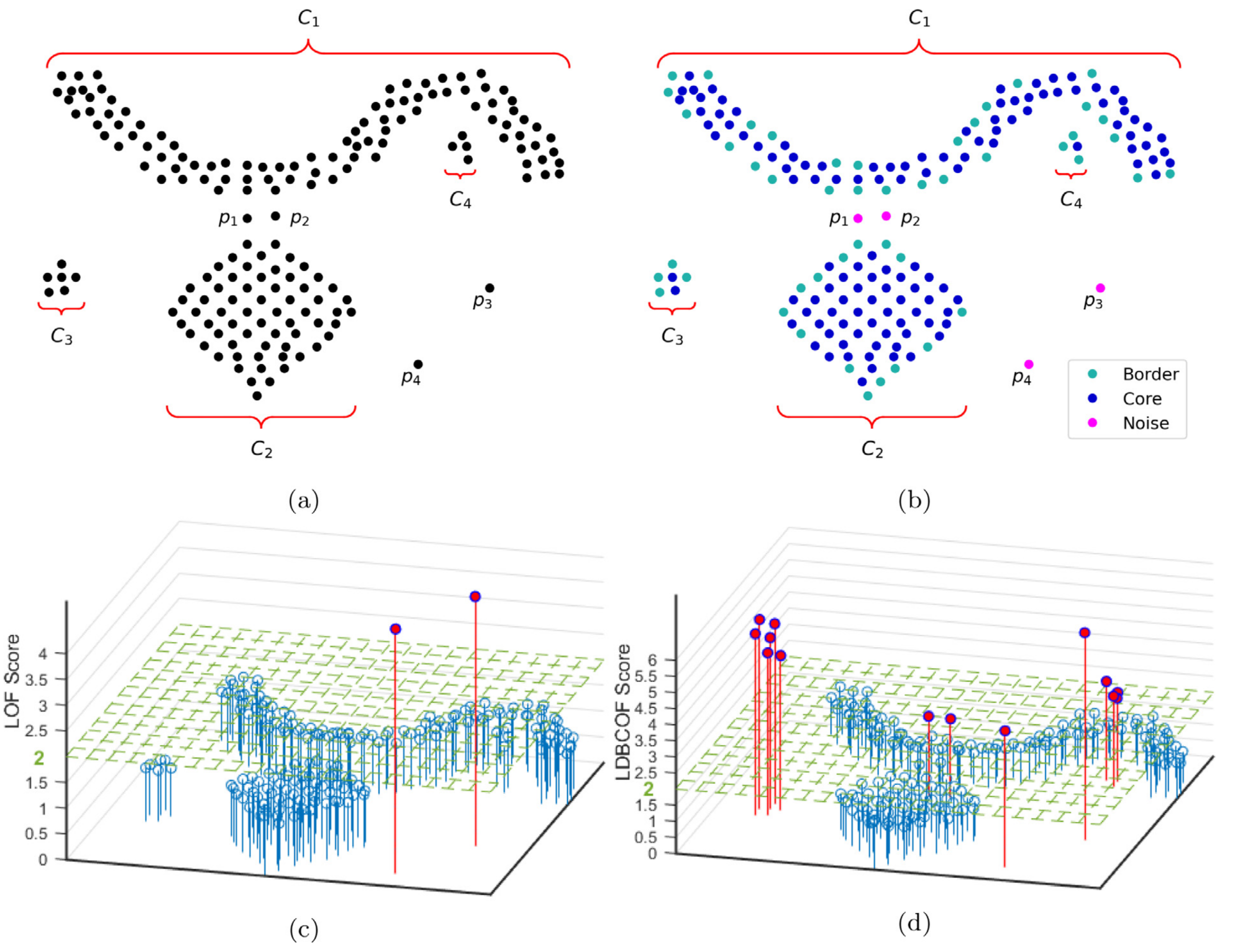


Fig. 4. Illustration of the artificial 2D dataset and performance of the algorithms (a) Scatter plot of the dataset, (b) Clustering result of DBSCAN (minPts = 4, Eps = 1.5), (c) LOFs (k = 3), (d) LDCBOFs of the samples (minPts = 4, Eps = 1.5, CkNN = 3).

outliers. In LOF-based algorithms, the outliers are determined by applying a specific threshold value to the LOFs of the samples. The LOFs of all points in Fig. 4a are shown in Fig. 4c according to the their positions. As can be seen from the Fig. 4c, only two global outliers (p_3 and p_4) can be detected by applying a threshold value 2 to the LOFs. The points in the outlier clusters C_3, C_4 and local outlier points (p_1 and p_2) cannot be detected, because their LOFs are too close to the inlier points. Specifically, cluster C_3 and C_4 contains enough samples to reduce local outlierness degree, besides C_4 is close to the inlier cluster C_1. Local outlier points p_1 and p_2 cannot be detected in LOF due to they are very close to the clusters C_1 and C_2. Because closest points in the dataset without considering the clustering information can be used to computation of the LOFs. While base methods cannot detect the points in outlier clusters C_3 and C_4, the proposed iLDCBOF method successfully determine the all outliers (global, local, and cluster). LDCBOFs of the points are demonstrated in Fig. 4d. To compute LDCBOF of the any samples in clusters C_3 and C_4, the core samples inside C_2 and C_1 needs to be utilized, respectively. For points p_1 and p_2, the CkNN of the points is determined from the nearest core points belonging to the same cluster. Therefore, the iLDCBOF method performs better in cases (C_3, C_4, p_1, and p_2) where iLOF fails, thanks to the use of CkNN to compute the local outlierness degree. The highest LDCBOF value of the inlier point is the 1.5216 while the lowest LDCBOF of the outlier point is 2.4608. Clearly, the iLDCBOF method provides a more apparent distinction between inliers and outliers. This is because formation of outlier sequence (outlier clusters) is prevented by CkNN, and the outlierness degree of points closer to the clusters are more distinctive than the LOF. In a nutshell, the iLDCBOF method demonstrates robust behavior against sequence of outliers and local outliers compared to the iLOF and iDBSCAN without requiring any parameter adjustment.

iLDCBOF method uses the advantage of the clustering information of the DBSCAN and computes the local outlierness degree of the points with CkNN. The intuition behind using CkNN of the samples instead of kNN for outlierness degree computation is to make it more robust against local outliers and detection of outlier clusters in which kNN often underperforms. Because the outlierness degree of samples in high-density regions (core and border) remains similar, while the outlierness degree in low-density regions (noise and small outlier cluster samples) increases. In this manner, the distinction between the outlierness degree of the inlier and outlier samples becomes more explicit.

5. Experimental results

In this section, the effectiveness of the proposed iLDCBOF method is depicted and compared with commonly used unsupervised outlier detection methods. All experiments are conducted on a computer with Windows 10 operating system running on an Intel Core i7-6900 K CPU@3.2 GHz processor, 64 GB of RAM. The datasets, experimental settings and evaluation criteria used in the study are explained in detail below. In addition, the performance of the proposed iLDCBOF against benchmarking methods is analyzed and the observed findings are reported.

5.1. Dataset

Only real-world datasets obtained from different fields are used to assessment of the performance. Because the cause of the outliers is not known in advance and this varies according to the application area of the collected data. 13 real-world datasets (Balance Scale, Thyroid, Annthyroid, Ecoli, Pima, Glass, Vowels, Heart Disease, Hepatitis, Cardio, Ionosphere, Satimage, and Musk) are originally derived from the UCI machine learning library [39], two datasets {Biomed and Boston House-price (Boston_HP)} from the Statlib collection [40] and Mammography from OpenML [41]. Characteristics of the real-world datasets are summarized in Table 2.

5.2. Experimental setup

Python version 3.8.5 is used to carry out the proposed iLDCBOF and benchmarking methods. The python outlier detection (PyOD) toolbox is used for the LODA method [42]. Publicly available source codes are used for the SOS[1], SO-GAAL[2], and MO-GAAL2 methods. Since there is no publicly open source code, iLOF[7], iCOF [8], INFLO [4], LDOF [5], and LoOP [6] methods are developed and implemented by us. Batch and incremental versions of unsupervised methods are expected to yield the same outlierness degree when the same data points are processed. However, the iLoOP method does not provide the same performance as the batch mode version as it does not update the old affected samples in the data [9]. Therefore, the batch mode LoOP method is used in the benchmarking. Due to the use of Mahalanobis distance metric in the process of computing Eps distance from the first 100 points, singular matrix errors are encountered in some datasets. To alleviate this situation, the number of points used to compute the Eps distance is increased until there is no singular matrix error.

5.3. Performance criteria

Different performance metrics are proposed to evaluate the effectiveness of the algorithms. These metrics are mainly derived from the confusion matrix as shown in Table 3.

Although outlier detection can be defined as a binary classification problem, most of the outlier detection methods computes the outlierness degree of the each instance. In addition, outliers constitute a small percentage of datasets. Therefore, accuracy which is the ratio of the correctly classified instances to the number of the instances in the dataset, is not an ideal performance metric for the evaluation of outlier detection method. Receiver operating characteristics (ROC) and the ROC-AUC is widely used performance metric. Because ROC, unlike accuracy and F-score evaluation metrics, is not affected by the distribution of classes in the dataset. It also allows different models to be compared simultaneously with various threshold values. To plot the ROC curve and find out the ROC-AUC, sensitivity, and specificity metrics need to be computed. Sensitivity and specificity are defined as

$$Sensitivity = \frac{TPs}{TPs + FNs} \tag{11}$$

$$Specificity = \frac{TNs}{TNs + FPs} \tag{12}$$

where, sensitivity/specificity measures the ratio between correctly classified outliers/inliers and all outliers/inliers, respectively. A ROC graph demonstrates the tradeoff between sensitivity and the specificity. In some cases, it is not easy to compare the ROC graph between different methods. So, a scalar value is required to compare the effectiveness of predictive models. ROC-AUC, which summarizes the ROC curve into a single value, enables easier comparison between prediction models. In a nutshell, the ROC-AUC score measures how well the predictive model can characterize the difference between classes.

Recently, statistical tests have been widely used to make a detailed comparison between a set of machine learning algorithms and to reveal the significance of reported differences among classifiers. There are two types of statistical test in the literature: parametric and nonparametric tests. If the assumptions required for parametric tests are valid, parametric tests are always preferred to non-parametric ones as they will have lower Type I errors and higher power.

One of the non-parametric and widely used tests in the literature is the Friedman test. It is applied to investigate whether there is a significant difference between the results of the methods compared on the data sets. The null hypothesis for this

[1] https://github.com/jeroenjanssens/scikit-sos
[2] https://github.com/leibinghe/GAAL-based-outlier-detection

Table 2
Properties of the real-world datasets.

Dataset	Description Inlier vs. Outlier	Dimension	Total # of instance	# of Outlier
Balance Scale	Left vs. Balanced	4	302	14
Biomed	Healthy vs. Diseased	5	194	67
Thyroid	Normal vs. Hyperfunction and Subnormal functioning	6	3772	93
Annthyroid	Normal vs. Hyperfunction and Subnormal functioning	6	7200	534
Mammography	Non-calcification vs. Calcification	6	11183	260
Ecoli	(cp, im, imU, om, pp) vs. (imS, imL, omL)	7	336	9
Pima	Healthy vs. Diabetes	8	768	268
Glass	Building Windows Float Processed vs. Rest	9	214	144
Vowels	(6, 7, 8) vs. (1)	12	1456	50
Heart Disease	Presence vs. Absence	13	297	137
Boston House-price	$Price < 35k \; vsPrice > 35k$	13	506	48
Hepatitis	Live vs. Die	19	155	32
Cardio	Normal vs. Pathologic	21	1831	176
Ionosphere	Good vs. Bad	33	351	126
Satimage	(1, 3, 4, 5, 7) vs. (2)	36	5803	71
Musk	(j146, j147, 252) vs. (213, 211)	166	3062	97

Table 3
Confusion matrix for outlier detection.

		Actual Class	
		Outlier	Inlier
Class	Outlier	True Positives (TPs) (Correct)	False Positives (FPs) (Incorrect)
Predicted	Inlier	False Negatives (FNs) (Incorrect)	True Negatives (TNs) (Correct)

test assumes that there is no significant difference between the methods. If the computed p-value is less than the chosen significance level, the null hypothesis is rejected, which specifies that at least two methods are significantly different from each other [43]. The Friedman test first ranks the performance achieved for each dataset/operation combination and then compute the overall ranking for each operation, $R_j = \frac{1}{g}\sum_i r_i^j$.

Under the null hypothesis, which states that all the methods behave similarly and thus their ranks R_j should be equal, the Friedman statistic

$$\chi_F^2 = \frac{12g}{m(m+1)}\left[\sum_j R_j^2 - \frac{m(m+1)^2}{4}\right] \tag{13}$$

is distributed according to χ_F^2 with $m-1$ degrees of freedom, when g and m are large enough (usually, $g > 10$ and $m > 5$), where g is the number of datasets and m is the number of methods. Iman and Davenport (1980) demonstrated that Friedman's χ_F^2 takes a conservative stance and derived a better statistic

$$F_F = \frac{(g-1)\chi_F^2}{g(m-1) - \chi_F^2} \tag{14}$$

which is distributed according to the F-distribution with $m-1$ and $(m-1)(g-1)$ degrees of freedom.

The Nemenyi post hoc test is used to make pairwise comparisons between benchmarking methods. In this test, if the mean ranks of two methods is greater than or equal to a critical distance (CD), the performance of the two classifiers is significantly different. CD is defined as

$$CD = q_\alpha\sqrt{\frac{m(m+1)}{6g}} \tag{15}$$

where, the critical values (q_α) are based on the Studentized range statistic divided by $\sqrt{2}$.

5.4. Complexity analysis

This section reports the time complexity analysis of the proposed iLDCBOF method. iLDCBOF is a combination of incremental LOF and DBSCAN algorithms. LOF is a density-based outlier detection method while DBSCAN is a density-based clustering method. Both methods use NN queries to compute density information, and then algorithm-specific operations are performed. Therefore, the computational complexity of both methods is mainly in NN queries. The time complexity of the NN search is $O\left(N^2\right)$ and can be reduced to $O(N \cdot \log N)$ with indexing structures such as R-tree and KD-tree; N is the number of samples in the data set.

DBSCAN has a time complexity of $O(N \cdot \log N))$. In iDBSCAN, the time complexity is higher compared to the batch mode version and is equal to $O(\gamma \cdot N \cdot \log N)$, where γ is the speed up factor, and has been proven theoretically and empirically by Ester et al. [35].

As with batch mode DBSCAN, the time complexity of batch mode LOF is $O(N \cdot \log N)$ [7]. The asymptotic time complexity of an iLOF for inserting a new sample is defined as:

$$T_{iLOF\_insertion} = O\left(k \cdot F \cdot T_{kNN} + k \cdot F \cdot T_{RkNN} + F^2 \cdot k + T_{insert}\right) \tag{16}$$

where, k is the user defined NN number, F is the maximum number of RkNN of the new sample, T_{kNN} and T_{RkNN} represents the time complexities of kNN and RKNN range queries, respectively. T_{insert} corresponds to the required time to insert a new sample into the data. The time complexities of T_{kNN}, T_{RkNN}, and T_{insert} are equal to $O(\log N)$ with indexing structures. The time complexity of the insertion of a new sample is

$$T_{iLOF\_insertion} = O\left(k \cdot F \cdot \log N + F^2 \cdot k\right) \tag{17}$$

After all the required updates are performed the time complexity of the iLOF is $O(N \cdot \log N)$. Therefore, if the iDBSCAN and then the iLOF method were applied separately in the proposed approach, the time complexity of iDBSCAN-iLOF would be as follows:

$$\underbrace{\quad} O(\gamma \cdot N \cdot \log N)_{iDBSCAN} + \underbrace{\quad} O(N \cdot \log N)_{iLOF} \tag{18}$$

However, in the proposed study, since the NN query for each new incoming sample is performed together in both methods, it is used to compute both clustering and outlier score at the same time. Hence, after necessary simplifications in Eq. 18, the overall complexity of iLDCBOF results in $O(\gamma \cdot N \cdot \log N)$. As a result, the time complexity of iLDCBOF is almost the same as iDBSCAN, but both clustering and outlier detection are performed simultaneously.

5.5. Analysis of the parameter Eps

To demonstrate the effectiveness of automatically determining the Eps distance, Table 4 presents the analysis results for the first 100 data points (with $k = feature\_num$) and for all of the data points (with $k = (2 * feature\_num)$) for both Euclidean and Mahalanobis metrics in 16 different real-world datasets. Euclidean distance is a well-known metric that is used in different machine learning algorithms, from clustering to classification. However, the performance of Euclidean distance is lower on high-dimensional datasets, as it gives equal importance to the features and assumes that the features are uncorrelated. According to the results, the performance of the Euclidean metric varies from one dataset to another. If any feature of the dataset possesses a more highly skewed distribution than the rest, the distance between the two instances is quite large due to the Euclidean metric. The Biomed, Glass, Ionosphere, Pima, Musk, and Satimage datasets are prime examples of this situation. Normalizing the data can be seen as a solution, but it is not suitable for use in data stream applications. Also, since the future value of the features is not known in advance, only local maximum and minimum values can be computed, not global values, and this may cause errors in any additional computations. However, as can be seen in Table 4, the performance of the Mahalanobis metric is consistent, even across different datasets. In other words, approximately the same elbow values are obtained for both the first 100 data points and for all of the data points with the Mahalanobis metric. The experimental results indicate that the Mahalanobis distance is more suitable for real-world applications, since it is scale-invariant and takes into account correlations between features.

5.6. Analysis of the parameter k

Since the parameter minPts directly affects the size of clusters in the dataset and the number of noise (outlier) data, it is very difficult to determine accurately. Also, it has a fundamental role in detecting outliers with the iLDCBOF method. To specify the appropriate minPts, a wide grid search must be performed in all datasets against various values of k, as shown in Table 5. Based on the results, four out of eleven datasets, i.e., Balance Scale, Glass, Vowels, and Boston_HP, with data dimensions up to 15 provide the best ROC-AUC results for the formula $k = (2 * feature\_num - 1)$. There are slight variations in the

Table 4
Elbow points in the real-world datasets.

Dataset	Mahalanobis		Euclidean	
	First 100, k = feature_num	All data, k = 2*feature_num	First 100, k = feature_num	All data, k = 2*feature_num
Balance Scale	0.9871	1.0847	1.4142	1.4142
Biomed	1.9870	2.1183	34.3520	85.2477
Thyroid	2.1918	1.2695	0.1789	0.1014
Annthyroid	2.2063	1.1153	0.0865	0.0334
Mammography	2.5111	0.7486	1.5878	0.6139
Ecoli	1.9265	2.1506	0.2381	0.2728
Pima	2.8108	2.6633	68.2839	46.2715
Glass	3.3944	3.6019	0.9141	1.9115
Vowels	4.5653	3.3368	2.2444	2.3933
Heart Disease	4.8532	4.5971	52.0707	46.0292
Boston_HP	4.8366	4.3814	68.3565	72.3592
Hepatitis	5.9124	6.0853	106.2100	114.9200
Cardio	5.3440	4.5368	4.5158	4.1880
Ionosphere	5.5018	4.6909	2.1557	3.0797
Satimage	7.8818	6.8751	60.9426	43.1166
Musk	16.4602	17.0353	859.9510	782.5700
Average	4.5856	4.1431	78.9689	75.2826

7 remaining datasets, i.e., Biomed, Thyroid, Annthyroid, Mammography, Ecoli, Pima, Heart Disease, with little impact on performance. Similarly, setting the *k* value to 29 in datasets with data dimensions larger than 15 results in a negligible difference in performance. For example, Table 5 shows that four out of five datasets, i.e., Hepatitis, Cardio, Ionosphere, and Satimage, have a decrease of 0.01, while k has no effect on the Musk dataset as shown in Table 5. In addition, since choosing a *k* value that is too high will increase the processing time [25,44], limiting *k* to 29 will be a suitable choice, especially for real-time-applications.

5.7. Performance comparison

The efficiency of the proposed iLDCBOF method was compared against the nine commonly-used, unsupervised outlier detection methods that operate in the incremental mode (iLOF, iCOF, and LODA) and the batch mode (INFLO, LDOF, LoOP, SOS, SO-GAAL, and MO-GAAL).

In order to allow a fair comparison with the proposed iLDCBOF method, the same parameters are used as much as possible in the benchmarking methods. The iLOF, iCOF, INFLO, LDOF, and LoOP methods take *k* as an input parameter. In these methods, *k* is set manually to $k = [(2 * \mathit{feature\_num}) - 1]$ because it is adusted automatically to this value in the iLDCBOF method for each dataset. As suggested in the reference paper of the SOS [45], the perplexity parameter, *h*, is set to 4.5. In a similar way, the parameters of the LODA method adjusted to the number of bins = 10 and the number of random cuts = 100 [46]. In the SO-GAAL and MO-GAAL methods, the values of the parameters recommended in the reference article are used [47], i.e., Number of Epochs (E):20, Discriminator Learning Rate (DLR):0.01, Generator Learning Rate (GLR):0.0001, Decay (Dec):1e-6, SGD Momentum (P):0.9, MO-GAAL:Number of Sub-generators:10, E:25, DLR:0.01, GLR:0.0001, Dec:1e-6, P: 0.9).

The ROC-AUC scores of the proposed iLDCBOF and the benchmarking methods with the specified hyperparameter are presented in Table 6. The ROC-AUC scores of the best performing methods in each dataset are highlighted in bold. To obtain more reliable and scalable results in the LODA, SO-GAAL, and MO-GAAL methods, the average of 10 consecutive runs is demonstrated in Table 6. The ROC-AUC score of these methods may change in every implementation.

As can be seen in Table 6, according to the experimental results, the proposed iLDCBOF methods performed best in 13 out of 16 datasets. In the remaining three datasets (Mammography, Cardio, and Satimage), it achieves the second-best ROC-AUC scores. Although the best performance in the Satimage dataset is achieved with LODA (0.98), it only exceeded iLDCBOF (0.96) by 0.02. In particular, iLDCBOF exceeds the second-best performance by at least 20% in Hepatitis and by at least 28% in the Boston_HP datasets. Furthermore, it demonstrated the optimal ROC-AUC score in the Musk dataset.

By analyzing the experimental results in Table 6, it is apparent that the performance of each method varies for different datasets. For instance, the iCOF method performed second best in Vowels (0.96), but it underperformed the average in Pima (0.51), Boston_HP (0.49), Cardio (0.53), and Satimage (0.56), and it performed last in the Musk (0.36) datasets. LODA performed best in Mammography (0.86), Cardio (0.84), and Satimage (0.98), and it performed well above average in Biomed (0.79), Ecoli (0.82), Pima (0.60), Glass (0.69), Hepatitis (0.62), and Musk (0.94), and it performed far below average in the Balance Scale (0.46), Annthyroid (0.47), Vowels (0.65), and Boston_HP (0.44) datasets. However, iLOF, which inspired iLDCBOF, performed second best in Ecoli (0.86) and Ionosphere (0.88), above average in Biomed (0.68), Mammography (0.68), Vowels (0.89), and Heart Disease (0.58), and it scored below average or had similar results in the rest of the datasets. In addition, iLOF's performance degraded as the data dimension increased (except for the Ionosphere dataset), but iLDCBOF virtually

Table 5
Effects of the parameter k of iLDCBOF on the ROC-AUC score in real-world datasets of different sizes.

Dataset	7	9	11	13	15	17	19	21	23	25	27	29	31	33	35	37	39	41	Min	Max
Balance Scale 302 × 4	**0.97**	0.97	0.96	0.95	0.95	0.90	0.94	0.97	0.96	0.97	0.97	0.96	0.97	0.96	0.96	0.95	0.94	0.94	0.90	**0.97**
Biomed 194 × 5	0.77	**0.80**	0.81	0.82	0.82	0.82	0.83	0.83	0.83	0.83	0.83	0.83	0.83	0.83	0.83	0.83	0.83	0.83	0.77	**0.83**
Thyroid 3772 × 6	0.79	0.81	**0.83**	0.85	0.88	0.89	0.90	0.92	0.92	0.93	0.94	0.94	0.95	0.95	0.95	0.96	0.96	0.96	0.79	**0.96**
Annthyroid 7200 × 6	0.76	0.77	**0.78**	0.78	0.79	0.79	0.80	0.80	0.80	0.80	0.80	0.80	0.80	0.80	0.80	0.80	0.80	0.79	0.76	**0.80**
Mammography 11183 × 6	0.70	0.71	**0.71**	0.73	0.74	0.75	0.76	0.76	0.75	0.76	0.77	0.77	0.77	0.77	0.78	0.78	0.78	0.78	0.70	**0.78**
Ecoli 336 × 7	0.87	0.87	0.87	**0.88**	0.89	0.88	0.89	0.88	0.88	0.87	0.87	0.87	0.87	0.87	0.87	0.87	0.87	0.87	0.87	**0.89**
Pima 768 × 8	0.59	0.61	0.62	0.63	**0.63**	0.64	0.64	0.65	0.65	0.66	0.66	0.66	0.66	0.66	0.66	0.67	0.67	0.67	0.59	**0.67**
Glass 214 × 9	0.67	0.71	0.71	0.72	0.72	**0.73**	0.73	0.73	0.73	0.73	0.73	0.73	0.72	0.72	0.72	0.72	0.73	0.72	0.67	**0.73**
Vowels 1456 × 12	0.95	0.95	0.96	0.97	0.97	0.97	0.97	**0.98**	0.98	0.97	0.98	0.98	0.97	0.97	0.97	0.97	0.97	0.97	0.95	**0.98**
Heart Disease 297 × 13	0.53	0.53	0.54	0.55	0.56	0.57	0.58	0.58	0.59	**0.60**	0.61	0.61	0.61	0.62	0.62	0.62	0.62	0.63	0.53	**0.63**
Boston_HP 506 × 13	0.58	0.56	0.61	0.64	0.65	0.66	0.68	0.68	0.69	**0.70**	0.70	0.70	0.70	0.70	0.70	0.70	0.69	0.69	0.56	**0.70**
Hepatitis 155 × 19	0.57	0.63	0.69	0.72	0.73	0.75	0.76	0.76	0.77	0.77	0.76	**0.76**	0.76	0.76	0.76	0.76	0.76	0.76	0.57	**0.77**
Cardio 1881 × 21	0.64	0.68	0.71	0.72	0.74	0.74	0.73	0.73	0.73	0.73	0.73	**0.73**	0.73	0.73	0.73	0.73	0.73	0.73	0.64	**0.74**
Ionosphere 351 × 33	0.90	0.90	0.91	0.91	0.92	0.92	0.92	0.92	0.92	0.93	0.93	**0.93**	0.93	0.93	0.94	0.94	0.94	0.94	0.90	**0.94**
Satimage 5803 × 36	0.65	0.67	0.67	0.69	0.72	0.76	0.84	0.91	0.94	0.95	0.96	**0.96**	0.96	0.97	0.97	0.97	0.97	0.97	0.65	**0.97**
Musk 3062 × 166	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.00**	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.00**

Table 6
Comparison of ROC-AUC scores with different outlier detection methods over real-world datasets. Best results are marked bold for each dataset.

Dataset	iLDCBOF	iLOF	iCOF	LODA	INFLO	LDOF	LoOP	SOS	SO-GAAL	MO-GAAL	Average
Balance Scale	**0.9673**	0.7146	0.5985	0.4633	0.7274	0.6765	0.7594	0.7681	0.9010	0.9396	0.7276
Biomed	**0.8021**	0.6802	0.6004	0.7907	0.5850	0.5209	0.6367	0.6820	0.5805	0.4867	0.6181
Thyroid	**0.8276**	0.7070	0.6070	0.7421	0.6903	0.7111	0.6401	0.6988	0.7730	0.7406	0.7011
Annthyroid	**0.7821**	0.7270	0.6885	0.4740	0.7053	0.7728	0.7291	0.7284	0.7341	0.7472	0.7007
Mammography	0.7135	0.6807	0.3563	**0.8593**	0.4523	0.7100	0.6611	0.4853	0.2597	0.3026	0.5297
Ecoli	**0.8767**	0.8589	0.8447	0.8204	0.8257	0.6347	0.7701	0.6133	0.7855	0.8536	0.7785
Pima	**0.6326**	0.5511	0.5105	0.5988	0.5063	0.4822	0.4791	0.5280	0.4893	0.4943	0.5155
Glass	**0.7264**	0.6376	0.6550	0.6940	0.6228	0.6215	0.6357	0.6451	0.4908	0.4332	0.6040
Vowels	**0.9775**	0.8914	0.9615	0.6514	0.8927	0.8256	0.9227	0.7786	0.7872	0.8153	0.8363
Heart Disease	**0.6009**	0.5787	0.5391	0.4733	0.5194	0.5078	0.5146	0.5385	0.5657	0.5117	0.5276
Boston_HP	**0.6988**	0.5108	0.4862	0.4412	0.4814	0.5436	0.4996	0.5035	0.5011	0.4983	0.4962
Hepatitis	**0.7630**	0.5742	0.5912	0.6239	0.5424	0.5663	0.5650	0.5206	0.4693	0.4730	0.5473
Cardio	0.7286	0.5610	0.5338	**0.8391**	0.5802	0.6469	0.5709	0.6188	0.3190	0.2654	0.5483
Ionosphere	**0.9276**	0.8836	0.8448	0.7891	0.8340	0.8764	0.8729	0.8080	0.6088	0.6462	0.7960
Satimage	0.9625	0.5546	0.5577	**0.9803**	0.5870	0.7039	0.5387	0.5253	0.5017	0.5222	0.6079
Musk	**0.9992**	0.4099	0.3563	0.9441	0.4907	0.7100	0.4302	0.5240	0.5000	0.5000	0.5406
Average	**0.8116**	0.6576	0.6082	0.6991	0.6277	0.6569	0.6391	0.6229	0.5792	0.5769	

was unaffected. This demonstrates that iLDCBOF performs well even in high dimensional data. The SO-GAAL and MO-GAAL methods had the second and third best performances, respectively, in the Balance Scale dataset, but they had the two lowest results in the Mammography, Glass, Hepatitis, Cardio, Ionosphere, and Satimage datasets. It is obvious from the ROC-AUC scores in Table 6 that iLDCBOF, due to newly introduced CkNN concept, produced more robust results against the outliers in data streams without requiring any user defined hyperparameters.

Finally, the performance results in the last column of Table 6 depict that iLDCBOF significantly exceeds the average ROC-AUC scores in 16 different real-world datasets. Considering the results in the last row of Table 6, the average ROC-AUC score of iLDCBOF (0.8116) in 16 different datasets was notably impressive. As an illustration, LODA demonstrated the closest performance to the iLDCBOF in the average results within the 9 methods that were compared, but iLDCBOF outperformed this method by 16.11%. In addition, given the performances of LODA, iLOF, LDOF, and LoOP, which are the best methods after iLDCBOF, it is apparent that their performances may vary. Considering the results, it is clear that the proposed iLDCBOF is suitable for data stream applications in different domains.

5.8. Statistical test analysis

Two non-parametric statistical tests are applied sequentially to determine the effectiveness of the compared methods, including the iLDCBOF. The first is the Friedman test, which is used to check whether the competence measures of all methods are equivalent. The second is the Nemenyi post hoc test, which is used to determine whether the proficiency measures of a pair of methods are equivalent. In both tests, the significance level is set to 0.05. In Friedman test, if the p-value is smaller than the significance level and the F_F value is greater than the critical value determined from the F distribution table, the null hypothesis is rejected which indicates that at least two methods are considerably different from each other. The p-value of approximately 8.9364E-7 was computed because of applying the Friedman test, indicating that the null hypothesis is highly unlikely.

In other words, the critical value of F(9,135) = 1.9499 and the computed F_F is 6.8159 for the ROC-AUC scores, thus the null hypothesis can be rejected with 95% confidence level. Since the null hypothesis is rejected due to the Friedman test, the Nemenyi post hoc test is applied, which provides a better insight to analyze the comparison methods.

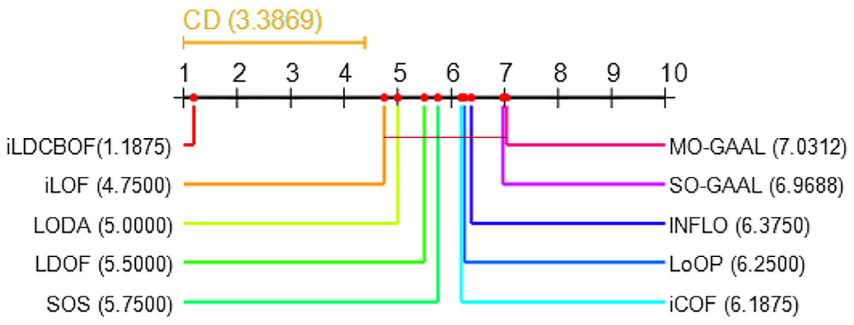


Fig. 5. Average ranks diagram comparing the benchmarking methods in terms of ROC-AUC.

At a significance level of $\alpha = 0.05$, the CD of the Nemenyi test for 10 methods in 16 datasets is equal to 3.3869. If the difference between the mean ranks of the compared methods are higher than the CD value, their performance is significantly different. Graphical visualization of the Nemenyi post hoc test is shown in Fig. 5. The farther to the left the position of the method on the coordinate axis, the better the performance of the method, and vice versa.

As can be seen in Fig. 5, the average rank of iLDCBOF is 1.1875, and statistically its performance will differ significantly from any method with an average rank exceeding 4.5744 (=1.1875 + 3.3869). For example, the average rank of iLOF is 4.75 and is greater than 4.5744, so it is statistically significant. In summary, as can be seen from Table 6 and Fig. 5, the efficacy of iLDCBOF is more clearly confirmed by both ROC-AUC scores and statistical analysis tests.

6. Conclusion

In this study, a new incremental outlier detection method (iLDCBOF) is introduced based on clustering and density-based approaches. The iLDCBOF method eliminates the three most important disadvantages of iLOF, i.e., 1) it does not require any user-defined parameters; 2) it has outstanding performance even on high dimensional datasets; and 3) it can recognize small outlier clusters. In addition, it can perform outlier detection and clustering simultaneously in data stream applications. This success of iLDCBOF is due to its combination of clustering and density-based approaches, and the newly introduced metric, CkNN, to compute the degree of outlierness. CkNN exhibits more robust behavior against outliers because it uses both the density and clustering information of the dataset. Extensive experiments have been performed on 16 real-world datasets to demonstrate the effectiveness of the proposed method. The proposed iLDCBOF outperforms both the incremental mode (iLOF, iCOF, and LODA) and the batch mode (INFLO, LDOF, LoOP, SOS, SO-GAAL, and MO-GAAL) learning methods, based on results from statistical tests and ROC-AUC analysis.

In summary, iLDCBOF is suitable for the real-time detection of outliers and clustering data stream applications in different domains. The proposed method can be combined with machine learning methods in the future to reduce the effects of outliers. In addition, the iLDCBOF method performs the detection of outliers and clustering simultaneously. However, all samples are stored in memory, which is somewhat of a limitation. Therefore, the algorithm can be improved by adding new steps in the future (e.g., summarization, fixed memory, and adaptive windows) to reduce the consumption of memory and to reduce the processing time, especially in large data streams.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] A. Kiersztyn, P. Karczmarek, K. Kiersztyn, W. Pedrycz, Detection and classification of anomalies in large data sets on the basis of information granules, IEEE Trans. Fuzzy Syst. (2021).
[2] Y. Zhou, H. Ren, Z. Li, W. Pedrycz, Anomaly detection based on a granular markov model, Expert Syst. Appl. 187 (2022) 115744.
[3] M.M. Breunig, H.-P. Kriegel, R.T. Ng, J. Sander, Lof: identifying density-based local outliers, in: Proceedings of the 2000 ACM SIGMOD international conference on Management of data, 2000, pp. 93–104.
[4] W. Jin, A.K. Tung, J. Han, W. Wang, Ranking outliers using symmetric neighborhood relationship, in: Pacific-Asia conference on knowledge discovery and data mining, Springer, 2006, pp. 577–593.
[5] K. Zhang, M. Hutter, H. Jin, A new local distance-based outlier detection approach for scattered real-world data, in: Pacific-Asia Conference on Knowledge Discovery and Data Mining, Springer, 2009, pp. 813–822..
[6] H.-P. Kriegel, P. Kröger, E. Schubert, A. Zimek, Loop: local outlier probabilities, in: Proceedings of the 18th ACM conference on Information and knowledge management, 2009, pp. 1649–1652.
[7] D. Pokrajac, A. Lazarevic, L.J. Latecki, Incremental local outlier detection for data streams, in: 2007 IEEE symposium on computational intelligence and data mining, IEEE, 2007, pp. 504–515.
[8] D. Pokrajac, N. Reljin, N. Pejcic, A. Lazarevic, Incremental connectivity-based outlier factor algorithm, Visions of Computer Science-BCS International Academic Conference (2008) 211–223.
[9] C. Hamlet, J. Straub, M. Russell, S. Kerlin, An incremental and approximate local outlier probability algorithm for intrusion detection and its evaluation, J. Cyber Secur. Technol. 1 (2017) 75–87.
[10] J. Ning, L. Chen, C. Zhou, Y. Wen, Parameter k search strategy in outlier detection, Pattern Recogn. Lett. 112 (2018) 56–62.
[11] J. Li, Q. Zhu, Q. Wu, Z. Fan, A novel oversampling technique for class-imbalanced learning based on smote and natural neighbors, Inf. Sci. 565 (2021) 438–455.
[12] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, et al, A density-based algorithm for discovering clusters in large spatial databases with noise, Kdd 96 (1996) 226–231.
[13] D. Tang, S. Zhang, J. Chen, X. Wang, The detection of low-rate dos attacks using the sadbscan algorithm, Inf. Sci. 565 (2021) 229–247.
[14] E. Azhir, N.J. Navimipour, M. Hosseinzadeh, A. Sharifi, A. Darwesh, An automatic clustering technique for query plan recommendation, Inf. Sci. 545 (2021) 620–632.
[15] S.F. Galán, Comparative evaluation of region query strategies for dbscan clustering, Inf. Sci. 502 (2019) 76–90.
[16] E. Schubert, J. Sander, M. Ester, H.P. Kriegel, X. Xu, Dbscan revisited, revisited: why and how you should (still) use dbscan, ACM Trans. Database Syst. (TODS) 42 (2017) 1–21.
[17] O. Alghushairy, R. Alsini, T. Soule, X. Ma, A review of local outlier factor algorithms for outlier detection in big data streams, Big Data Cognitive Comput. 5 (2021) 1.
[18] M. Kontaki, A. Gounaris, A.N. Papadopoulos, K. Tsichlas, Y. Manolopoulos, Efficient and flexible algorithms for monitoring distance-based outliers over data streams, Inform. Syst. 55 (2016) 37–53.
[19] H. Yao, X. Fu, Y. Yang, O. Postolache, An incremental local outlier detection method in the data stream, Appl. Sci. 8 (2018) 1248.

[20] M.J. Bah, H. Wang, M. Hammad, F. Zeshan, H. Aljuaid, An effective minimal probing approach with micro-cluster for distance-based outlier detection in data streams, IEEE Access 7 (2019) 154922–154934.
[21] R. Zhu, X. Ji, D. Yu, Z. Tan, L. Zhao, J. Li, X. Xia, Knn-based approximate outlier detection algorithm over iot streaming data, IEEE Access 8 (2020) 42749–42759.
[22] T. Dai, Z. Ding, Online distributed distance-based outlier clearance approaches for wireless sensor networks, Pervasive Mobile Comput. 63 (2020) 101130.
[23] M. Salehi, C. Leckie, J.C. Bezdek, T. Vaithianathan, X. Zhang, Fast memory efficient local outlier detection in data streams, IEEE Trans. Knowl. Data Eng. 28 (2016) 3246–3260.
[24] G.S. Na, D. Kim, H. Yu, Dilof: Effective and memory efficient local outlier detection in data streams, in: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2018, pp. 1993–2002.
[25] J.-W. Huang, M.-X. Zhong, B.P. Jaysawal, Tadilof: Time aware density-based incremental local outlier detection in data streams, Sensors 20 (2020) 5829.
[26] L. Chen, W. Wang, Y. Yang, Celof: Effective and fast memory efficient local outlier detection in high-dimensional data streams, Appl. Soft Comput. 102 (2021) 107079.
[27] J. Gao, W. Ji, L. Zhang, A. Li, Y. Wang, Z. Zhang, Cube-based incremental outlier detection for streaming computing, Inf. Sci. 517 (2020) 361–376.
[28] J. Li, H. Izakian, W. Pedrycz, I. Jamal, Clustering-based anomaly detection in multivariate time series data, Appl. Soft Comput. 100 (2021) 106919.
[29] R.A. Ariyaluran Habeeb, F. Nasaruddin, A. Gani, M.A. Amanullah, I. Abaker Targio Hashem, E. Ahmed, M. Imran, Clustering-based real-time anomaly detection-a breakthrough in big data technologies, Trans. Emerging Telecommun. Technol. (2019) e3647.
[30] C. Yin, S. Zhang, Z. Yin, J. Wang, Anomaly detection model based on data stream clustering, Cluster Comput. 22 (2019) 1729–1738.
[31] W. Yan, Q. Sun, H. Sun, Y. Li, Joint dimensionality reduction and metric learning for image set classification, Inf. Sci. 516 (2020) 109–124.
[32] Y. Ruan, Y. Xiao, Z. Hao, B. Liu, A nearest-neighbor search model for distance metric learning, Inf. Sci. 552 (2021) 261–277.
[33] Y. Zhao, Y. Yu, Y. Li, G. Han, X. Du, Machine learning based privacy-preserving fair data trading in big data market, Inf. Sci. 478 (2019) 449–460.
[34] C.H. Tan, V.C. Lee, M. Salehi, Mir_mad: An efficient and on-line approach for anomaly detection in dynamic data stream, in: 2020 International Conference on Data Mining Workshops (ICDMW), 2020, pp. 424–431.
[35] M. Ester, H.-P. Kriegel, J. Sander, M. Wimmer, X. Xu, Incremental clustering for mining in a data warehousing environment, in: Proceedings of the 24rd International Conference on Very Large Data Bases, VLDB '98, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1998, pp. 323–333.
[36] Y. Ma, X. Zhao, J. Zhang, C. Zhang, X. Qin, Outlier detection from multiple data sources, Inf. Sci. (2021).
[37] W.H. Greene, Econometric analysis, Pearson Education India, 2003.
[38] V. Satopaa, J. Albrecht, D. Irwin, B. Raghavan, Finding a kneedle in a haystack: Detecting knee points in system behavior, in: 2011 31st international conference on distributed computing systems workshops, IEEE, 2011, pp. 166–171.
[39] A. Asuncion, D. Newman, Uci machine learning repository (2007).
[40] C. Kooperberg, Statlib: an archive for statistical software, datasets, and information, Am. Stat. 51 (1997) 98.
[41] J. Vanschoren, J.N. Van Rijn, B. Bischl, L. Torgo, Openml: networked science in machine learning, ACM SIGKDD Explorations Newsletter 15 (2014) 49–60.
[42] Y. Zhao, Z. Nasrullah, Z. Li, Pyod: A python toolbox for scalable outlier detection, arXiv preprint arXiv:1901.01588 (2019)..
[43] S. García, A. Fernández, J. Luengo, F. Herrera, Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power, Inform. Sci. 180 (2010) 2044–2064.
[44] X. Deng, L. Wang, Modified kernel principal component analysis using double-weighted local outlier factor and its application to nonlinear process monitoring, ISA Trans. 72 (2018) 218–228.
[45] J. Janssens, F. Huszár, E. Postma, H. van den Herik, Stochastic outlier selection, Tilburg centre for Creative Computing, techreport 2012–001 (2012)..
[46] T. Pevný, Loda: Lightweight on-line detector of anomalies, Mach. Learn. 102 (2016) 275–304.
[47] Y. Liu, Z. Li, C. Zhou, Y. Jiang, J. Sun, M. Wang, X. He, Generative adversarial active learning for unsupervised outlier detection, IEEE Trans. Knowl. Data Eng. (2019).