SDCOR: Scalable density-based clustering for local outlier detection in massive-scale datasets

Sayyed Ahmad Naghavi Nozad [a,*], Maryam Amir Haeri [b], Gianluigi Folino [c]

[a] *Department of Computer Engineering, Amirkabir University of Technology, Tehran, Iran*
[b] *Learning, Data-Analytics, and Technology Department, University of Twente, Enschede, Netherlands*
[c] *ICAR-CNR, Rende, Italy*

ARTICLE INFO

ABSTRACT

This paper presents a batch-wise density-based clustering approach for local outlier detection in massive-scale datasets. Unlike the well-known traditional algorithms, which assume that all the data is memory-resident, our proposed method is scalable and processes the input data chunk-by-chunk within the confines of a limited memory buffer. A temporary clustering model is built at the first phase; then, it is gradually updated by analyzing consecutive memory loads of points. Subsequently, at the end of scalable clustering, the approximate structure of the original clusters is obtained. Finally, by another scan of the entire dataset and using a suitable criterion, an outlying score is assigned to each object called SDCOR (Scalable Density-based Clustering Outlierness Ratio). Evaluations on real-life and synthetic datasets demonstrate that the proposed method has a low linear time complexity and is more effective and efficient compared to best-known conventional density-based methods, which need to load all data into the memory; and also, to some fast distance-based methods, which can perform on data resident in the disk.

© 2021 Elsevier B.V. All rights reserved.

1. Introduction

Outlier detection, which is a noticeable and open line of research [1–4], is a fundamental issue in data mining. Outliers refer to rare objects that deviate from the well-defined notions of expected behavior, and discovering them is sometimes compared with searching for a needle in a haystack because the rate of their occurrence is much lower than normal objects. Outliers often interrupt the learning procedure from data for most of the analytical models, and thus, capturing them is very important because it can enhance the model accuracy and reduce the computational load of the algorithm. However, outliers are not always annoying, and sometimes, they become of particular interest for the data analyst in many problems such as controlling cellular phone activity to detect fraudulent usage, like stolen phone airtime. Outlier detection methods could also be considered as a preprocessing step, which is useful before applying any other advanced data mining analytics, and it has a wide range of applicability in many research areas, including intrusion detection,

activity monitoring, satellite image analysis, medical condition monitoring, etc. [1,5].

Outliers can be generally partitioned into two different categories, namely global and local. Global outliers are objects that show significant abnormal behavior compared to the rest of the data, and thus in some cases, they are considered point anomalies. On the contrary, local outliers only deviate significantly w.r.t. a specific neighborhood of the object [2,6]. In [7,8], it is noted that the concept of the local outlier is more comprehensive than that of the global outlier; i.e., a global outlier could also be considered as a local one, but not necessarily vice versa. This is the reason that makes finding the local outliers much more cumbersome.

1.1. Motivation

In recent years, advances in data acquisition have made massive collections of data, which contain valuable information in diverse fields like business, medicine, society, government, etc. As a result, the common conventional software methods, including but not limited to [7,9–29], for processing and management of such massive amount of data will no longer be efficient, because most of these methods assume that the data is memory-resident and their computational complexity for large-scale datasets is really expensive [4].

One of the usual ways to solve this issue is the use of parallel/distributed computing techniques [30–36]. In such a strategy

---

   * Corresponding author.
   E-mail addresses: sa_na33@aut.ac.ir (S.A. Naghavi Nozad),
m.amirhaeri@utwente.nl (M. Amir Haeri), gianluigi.folino@icar.cnr.it (G. Folino).

in which both the hardware and software facilities are extensively employed, a big or even an intolerable task for a single computing module is divided into a pack of smaller assignments, as each of them is appointed to a distinct processing unit. Therefore, more work can be carried out simultaneously by exerting more devices than the condition in which the task was about to be done at once through an isolated computer.

However, distributed solutions come along with some inevitable downsides like being more complicated to implement and troubleshoot, requiring additional resources, colossal power consumption, and the principal need for hardware cooling technologies in some cases; furthermore, such dispersed resolutions commonly accomplish an activity with a less efficacy compared to their serial counterparts on account of communication and coordination overheads.

The motivation behind this study is to propose a scalable in-memory algorithm that utilizes merely a single stand-alone computer while delivering competing results with the single-run state of the data mining algorithm on the entire dataset.

1.2. Contribution

In this paper, we propose a new scalable and density-based clustering method for local outlier detection in massive-scale datasets that cannot be loaded into memory at once, employing a chunk-by-chunk load procedure. In practice, the proposed approach is a clustering method for huge datasets in which outlier identification comes after that as a side effect, and it is inspired by a scaling clustering algorithm for very large databases named after its authors, BFR [37]. However, BFR has some weak spots: it needs to know the actual number of original clusters in data; it has a strong assumption on the structure of existing clusters, which ought to be Gaussian distributed with uncorrelated attributes; and more importantly, it is not introducing noise. In short, this clustering algorithm works as follows. First, it reads the data as successive (preferably random) samples so that each sample can be stored in the memory buffer, and then it updates the current clustering model over the contents of the buffer. Based on the updated model, singleton data are classified into three groups: some of them can be discarded over the updates to the sufficient statistics (Discard Set, DS); some can be moderated through compression and abstracted as sufficient statistics (Compression Set, CS); some demand to be retained in the buffer (Retained Set, RS).

Like BFR, our proposed method operates within the confines of a limited memory buffer. Thus, assuming that an interface to the database allows the algorithm to load an arbitrary number of requested data points, whether sequentially or randomized, we are forced to load the data chunk-by-chunk so that there is enough space for both loading and processing each chunk at the same time. The proposed approach is based on clustering, and therefore, it must avoid outliers being able to play any influential role in forming and updating clusters. After processing each chunk, the algorithm should combine its approximate results with those of the previous chunks in a way that the final approximate result will compete with the same result obtained by processing the entire dataset at once. An algorithm, which is capable of handling data in such a gradual way and finally provides an approximate result, from an operational perspective, is called a scalable algorithm [38]. Moreover, from an algorithmic point of view, scalability means that algorithm complexity should be nearly linear or sublinear w.r.t. the problem size [39].

In more detail, the proposed method comprises three steps. In the first step, a primary random sampling is carried out to create an abstraction of the whole data on which the algorithm works. Then, an initial clustering model is built, and some information

required for the next phase of progressive clustering will be acquired. In the second step, on the basis of the currently loaded chunk of data into the memory, a scalable density-based clustering approach is executed in order to identify dense regions, which leads to building incrementally some clusters, named miniclusters or subclusters. When all chunks are processed, the final clustering model will be built by merging the information obtained through these miniclusters. Finally, by applying a Mahalanobis distance criterion [40,41] to the entire dataset, an outlying score is assigned to each object.

In summary, the main contributions of our proposed method are listed as follows:

- There is no need to know the real number of original clusters.
- It works better with Gaussian clusters having correlated or uncorrelated features, but it also works well with convex-shaped clusters following an arbitrary distribution.
- It has a linear time complexity with a low constant.
- In spite of working in a scalable manner and operating on chunks of data, in terms of detection accuracy, it is still able to compete with conventional density-based methods, which maintain all the data in the memory; and also, with some fast distance-based methods, which do not require to load the entire data into the memory at their training stage.

The paper is organized as follows: Section 2 discusses some related work in the field of outlier detection. In Section 3, we present the detailed descriptions of the proposed approach. Section 4 delineates the experimental design of our analysis for the proposed method and the other contending techniques. In Section 5, the experimental results on various real and synthetic datasets are provided. Finally, conclusions are given in Section 6.

2. Related work

Outlier detection methods can generally be divided into the following eight categories [42–45]: extreme value analysis, probabilistic methods, distance-based methods, density-based methods, clustering-based methods, graph-based methods, information-theoretic methods, and isolation-based methods. Besides, after considering these categories, in a discussion, we will look upon the capabilities of the corresponding methods to operate on massive-scale data.

2.1. Extreme value analysis

In extreme value analysis, the overall population is supposed to have a unique probability density distribution, and only those objects at the very ends of it are considered outliers. In particular, these types of methods are useful to find global outliers [42,46].

2.2. Probabilistic methods

In probabilistic methods, we assume that the data were generated from a mixture of different distributions as a generative model, and we use the same data to estimate the parameters of the model. After determining the specified parameters, outliers will be those objects with a low likelihood of being generated by this model [42]. Schölkopf et al. [47] propose a supervised approach, in which a probabilistic model w.r.t. the input data is provided so that it can fit normal data in the best possible way. In this manner, the goal is to discover the smallest region that contains most of the normal objects; data outside this region are supposed to be outliers. This method is, in fact, an extended version of Support Vector Machines (SVM), which has been improved to cope with imbalanced data; the other name for this method

in the literature is one-class SVM, or in brief, OCSVM [48]. In practice, in this way, a small number of outliers is considered as belonging to the rare class and the rest of the data as belonging to normal objects.

2.3. Distance-based methods

In distance-based methods, distances among all objects are computed to detect outliers. An object O is assumed to be a distance-based outlier if at least fraction p_0 of the objects in the dataset has a distance greater than d_0 from O [49,50]. Another definition for the distance-based outliers denotes that concerning the two input parameters, k as a positive integer and R as a positive real number, an object O is reported as an outlier if less than k objects in the dataset rest in the distance R from O [51]. In [9], a Local Distance-based Outlier Factor (LDOF) is proposed to find outliers in scattered datasets, which employs the relative distance from each data object to its nearest neighbors.

Bay and Schwabacher [52] propose an optimized nested-loop algorithm based on the k Nearest Neighbors (kNN) distances among objects, that has a near-linear time complexity and is shortly named ORCA (Optimal Reciprocal Collision Avoidance). ORCA shuffles the input dataset in random order using a disk-based algorithm and processes it in blocks, as there is no need to load the entire data into the memory. It keeps looking for a set of the user-defined number of data points as potential anomalies, as well as for their anomaly scores. The cut-off value is set as the minimum outlierness score of the set, and it will get updates if there is a data point having a higher score in other blocks of data. For data points that obtain a lower score than the cut-off, they should be pruned; this pruning scheme will only expedite the process of distance computation in the case of data being ordered in an uncorrelated manner. ORCA's worst-case time-complexity is $O\left(n^2\right)$, and the I/O cost for the data accesses is quadratic. For the anomaly definition, it can use either the distance to the kth nearest neighbor or the average distance of kNN.

Angiulli and Fassetti [51] propose DOLPHIN (Detecting Out-Liers PusHing objects into an INdex), a distance-based outlier mining method that is particularly dedicated to operating on disk-resident data and functions efficiently in terms of CPU and I/O cost two at a time. Furthermore, both theoretical and practical proofs are presented that the proposed method occupies the memory space as much as a small portion of the dataset. DOLPHIN obtains its efficiency certainly through combining three policies in an integrated plan, namely: (1) careful selection of objects to be retained in the buffer; (2) employing decent pruning strategies; (3) applying effective similarity inspection approaches, which is especially achieved without requiring prior indexing the entire input data, differently from the other competing techniques. Moreover, DOLPHIN is capable of being applied on any kind of records attributed to either metric or non-metric scopes.

S_p [53] is a simple and rapid distance-based method that utilizes the nearest neighbor distance on a small sample from the dataset. It takes a small random sample of the entire dataset and then assigns an outlierness score to each point, as the distance from the point to its nearest neighbor in the sample set. Therefore, this method enjoys a linear time complexity concerning each one of the essential variables, viz the number of objects, the number of dimensions, and the number of samples; furthermore, S_p has a constant space complexity, which makes it ideal for analyzing massive datasets.

2.4. Density-based methods

In density-based methods, the local density of each object is calculated in a specific way and then is utilized to define the outlier scores. Given an object, the lower its local density compared to its neighbors, the more likely it is that the object is an outlier. Density around the points could be calculated by using many techniques, which most of them are distance-based [7,42]. For example, Breunig et al. [7] propose a Local Outlier Factor (LOF) that uses the distance values of each object to its nearest neighbors to compute local densities. However, LOF has a drawback: the scores obtained through this approach are not globally comparable between all objects in the same dataset or even in different datasets. The authors of [10] introduce the Local Outlier Probability (LoOP), which is an enhanced version of LOF. LoOP gives each object a score in the interval [0,1], which is the probability of the object being an outlier and is widely interpretable among various situations. Moreover, two distributed versions of LOF are presented in [35] and [36] to operate on very large datasets. In [35], a distributed LOF pipeline framework (DLOF) along with a data assignment strategy are introduced to efficiently handle the required information exchange among various machines and elevating the independence of every processing unit; this will reduce the number of data replications and save more resources. In [36], a novel and cautious removal technique is proposed to effectively eliminate those objects that are not potential to be reported as top-ranked outliers and also not capable of being contemplated as possible neighbors of other points residing on other machines.

An INFLuenced Outlierness (INFLO) score is presented in [11], which adopts both kNN and Reverse Nearest Neighbors (RNN) of an object to estimate its relative density distribution. Furthermore, Tang and He [12] propose a local density-based outlier detection approach, concisely named RDOS, in which the local density of each object is approximated with the local Kernel Density Estimation (KDE) through nearest neighbors of it. In this approach, not only the kNN of an object are taken into account, but also the RNN and the Shared Nearest Neighbors (SNN) are considered for density distribution estimation as well. Moreover, a parameter-free density-based algorithm for both clustering and outlier detection purposes is presented in [54], which utilizes two novel neighborhood concepts, namely Unique Closest Neighbor (UCN) and Unique Neighborhood set (UNS), depending on the underlying data distribution. This proposed approach does not follow random or repetitive solutions to culminate in the best candidate outcome, and hence, is computationally cost-effective.

A non-parameter outlier detection algorithm based on Natural Neighbors (NaN) is proposed in [13]. In this parameter-free method, first, a novel search algorithm using KD-tree data structure [55] is introduced for detecting NaN instead of kNN for every object. The great benefit of NaN over kNN is that the neighbor search method is free of any input parameter, and the calculation is upon the intrinsic manifold of data. In addition, the NaN concept is scale-free, which means that the number of neighbors for various objects is not essentially the same. Furthermore, the new concepts of Natural Influence Space (NIS) and Natural Neighbor Graph (NNG) are proposed to finally compute the Natural Outlier Factor (NOF) for each individual data point. A higher NOF value for an object indicates the higher possibility for it to be identified as an outlier.

Wahid and Rao have recently proposed three different density-based methods, shortly named RKDOS [14], ODRA [15] and NaNOD [16]. RKDOS is mostly similar to RDOS, in which by using both kNN and RNN of each point, the density around it is estimated employing a Weighted Kernel Density Estimation (WKDE) strategy with a flexible kernel width. Moreover, the Gaussian

kernel function is adopted to corroborate the smoothness of estimated local densities, and also, the adaptive kernel bandwidth enhances the discriminating ability of the algorithm for better separation of outliers from inliers. ODRA is specialized for finding local outliers in high dimensions and is established on a relevant attribute assortment approach. There are two major phases in the algorithm, which in the preliminary one, unimportant features and data objects are pruned to achieve better performance. In other words, ODRA firstly tries to distinguish those attributes which contribute substantially to anomaly detection from the other irrelevant ones; also, a significant number of normal points residing in the dense regions of data space will be discarded for the sake of alleviating the detection process. In the second phase, like RDOS, a KDE strategy based on kNN, RNN, and SNN of each data object is utilized to evaluate the density at the corresponding locations, and finally, assign an outlying degree to each point. In NaNOD, the concept of Natural Neighbors (NaN) is used to adaptively attain a proper value for k (number of neighbors), which is called here the Natural Value (NV), while the algorithm does not ask for any input parameter to acquire that. Thus, the parameter choosing challenge could be mitigated through this procedure. Moreover, a WKDE scheme is applied to the problem to assess the local density of each point, as it takes the benefit of both kNN and RNN categories of nearest neighbors, which makes the anomaly detection model adjustable to various kinds of data patterns.

Besides, in recent times, Xie et al. [17] have proposed LGOD, an outlier model founded upon Newton's theory of gravitation [56]. In this model, every individual point is supposed as an element with mass, associated with a Local Resultant Force (LRF) engendered by its neighboring points. LGOD first determines the LRFs on each data object; then, regarding its tolerance to proximity parameter k, by aggregating the varying LRF values out of the fluctuating proximity parameter, outlying scores are deduced.

2.5. Clustering-based methods

Clustering-based methods use a global analysis to detect crowded regions, and outliers will be those objects not belonging to any cluster [42]. A Cluster-Based Local Outlier Factor (CBLOF) in [20], and a Cluster-Based Outlier Factor (CBOF) in [21] are presented. In both of them, after the clustering procedure is carried out, in regard to a specific criterion, clusters are divided into two large and small groups; it is assumed that outliers lie in the small clusters. At last, the distance of each object to its nearest large cluster is used in different ways to define the respective outlier score. Furthermore, a distributed strategy for both clustering and outlier detection tasks is proposed in [34], in which the classic k-means/median clustering techniques are fundamentally studied to become operable on datasets disseminated across various sites. The proposed method is efficient considering both time and required communications among multiple machines and has a satisfactory estimation level, especially on global outliers.

Jobe and Pokojovy [22] propose a data-intensive computing approach that is established upon cluster analysis and employs a reweighted variant of Rousseeuw's minimum covariance determinant method [57]. Adopting the powerful squared Mahalanobis distance statistic is the core idea of this study, although the detection capability of this criterion is substantially declined with the increasing number of outliers in data. Thus by applying a clustering-based multi-stage algorithm in this method, the potential outlying points which could be misclassified through the Mahalanobis distance measure are primarily distinguished from true inliers. Furthermore, Huang et al. [23] propose a novel outlier cluster detection method called ROCF, which does not require the count of top-N outliers as an input parameter. Using the mutual neighbors concept, ROCF firstly builds a neighborhood graph entitled MUNG (MUtual Neighbors Graph). Then, regarding the truth that outlying clusters are usually smaller in size than the normal clusters, and also by automatically unraveling the approximate outlier ratio of the input dataset, both singular outliers and anomalous minuscule clusters are discovered.

2.6. Graph-based methods

Graph-based methods are among the most robust approaches in the outlier mining area. They have called immense consideration in recent time because they can substantially express various data conditions [58,59] and extraordinarily grasp the wide-ranging interrelationships within the data objects [60,61]. In general, each individual object is characterized as a graph node, and the connections and dependencies among data elements are represented as the linking edges between the nodes. Then by assessing the graph structure — whether locally or globally — or even through evaluating the changing process of the graph or other suitable criteria, the anomalousness degree for every data object will be defined.

Moonesignhe and TAN [24,25] propose OutRank, which is among the very first graph-based approaches for outlier detection in datasets comprising of multidimensional points. OutRank is a stochastic algorithm in which, at first, a graph representation of the input data is constructed based on the similarity among objects and the count of shared neighbors between them. Then, employing the Markov chain model established regarding the attained graph, every object in data will attain an outlying score. One of the major problems with the graph-based methods is that they overlook the local information in each node vicinity, and unfortunately, this entails so many inliers be mistakenly identified as outliers, which is referred to as a high false-positive rate. In [26], a local information graph is built upon combining the graph delineation of the input data with the local information around each data object to overcome this issue. By utilizing the local information graph, the mismatching inter-dependencies among different kinds of objects are captured. Finally, the outlying scores are defined by applying a random walk on the graph. Wang et al. [27] propose another graph-based method that utilizes multiple neighborhood graphs to obtain diverse local information from different vantage points. Then, some stationary distribution vectors are acquired by executing a random walk process on these neighborhood graphs that are developed through distinctive neighbor sizes. Finally, outlier scores are derived over a novel designed scoring function established upon various change patterns of the corresponding values in the stationary distribution vectors.

Moreover, in [28], Wang et al. again, by integrating the local information with the latent associations in the graph rendering of the genuine dataset, propose a modern graph-based method called Virtual Outlier Score (VOS). In VOS, at the start, a similarity graph is initiated through employing the top-k analogous neighbors of each individual object; furthermore, a k-virtual graph is determined using the new concepts of virtual nodes and groups of virtual edges, which captures both local and global information among objects within the dataset. Then, by applying a well-suited version of the Markov random walk process on the highly associated virtual graph, under the fact that for accessing the likely anomalies in data, they should obtain more weight by the random walker than the other inliers, the algorithm reaches equilibrium or a stationary distribution in its operation. Finally, the virtual outlier ranks will be inferred from this stationary situation. Amil et al. [29] propose two methods that begin with a graph structure and are more useful for anomaly detection in high-dimensional data. Both methods depend on a reliable definition for the distance between pairs of data elements; thus, they can deal with

non-specific data types. The nodes and weighted edges in the representing graph correspond to data objects and distances among them, respectively. In the first method, an anomaly degree is specified for each item concerning the graph fragmentation. The other method employs the famous non-linear dimensionality reduction approach named IsoMap [62] and defines the anomaly scores regarding the variations between the geodesic distances and the distances in the acquired subspace.

2.7. Information-theoretic methods

Information-theoretic methods could be particularly deemed at almost the same level as distance-based and other deviation-based models. The only exception is that in such methods, a fixed concept of deviation is determined at first, and then, anomaly scores are established through evaluating the model size for this specific type of deviation; unlike the other usual approaches, in which a fixed model is determined initially and employing that, the anomaly scores are obtained by means of measuring the size of the deviation from such model [42].

Wu and Wang [18] propose a single-parameter method for outlier detection in categorical data using a new concept of Holoentropy, and by utilizing that, a formal definition of outliers and an optimization model for outlier detection is presented. According to this model, a function for the outlier factor is defined, which is solely based on the object itself, not the entire data, and it could be updated efficiently. There are two proposed greedy Information-Theory-Based (ITB) algorithms for this method, namely ITB-SP (Single Pass) and ITB-SS (Step-by-Step), that essentially can operate only on nominal space. However, the authors claim that they can be adapted to numerical space through either the extension of holoentropy or employing a suitable and viable discretization approach. Moreover, Dang et al. [19] propose an information-theoretic method named LODI in which besides identifying local outliers in continuous data, the anomalousness reason for any of them is presented too. In LODI, the quadratic entropy is investigated in an appropriate manner to select a neighboring set per each point. Then, following a learning method based on matrix eigen-decomposition, an optimal subspace is defined in which an outlier candidate is as much as possible isolated from its neighborhood. The revealed features in the corresponding subspace are critical to interpreting the outstanding characteristics of outliers.

2.8. Isolation-based methods

Isolation-based methods were firstly introduced in [63,64] and the authors named their novel method iForest. The main idea behind iForest was elicited from a well-known ensemble method called Random Forests [65], which is mostly employed in classification problems. In this approach, firstly, it is essential to build an ensemble of isolation trees (iTrees) for the input dataset, then outliers are those objects with a short average path length on the corresponding trees. For making every iTree, after acquiring a random sub-sample of the entire data, it is recursively partitioned by randomly choosing an attribute and then randomly specifying a split value among the interval of minimum and maximum values of the chosen attribute. Since this type of partitioning can be expressed by utilizing a tree structure, thus the number of times required to partition the data for isolating an object is equal to the path length from the starting node to the ending node in the respective iTree. In such a situation, the tree branches which bear outliers have a remarkably lower depth because these outlying points fairly deviate from the normal conduct in data. Therefore, data instances that possess substantially shorter paths from the root in various iTrees are more potential to be outliers.

One chief issue about using such an isolation-based method is that with the increase in the number of dimensions, if an incorrect attribute choice for the splitting matter is made at the higher levels of the iTree, then the probability of detection outcomes getting misled, grows potentially. Nevertheless, the advantage of iForest is that by considering the idea of isolation, it can utilize the sub-sampling in a more sophisticated way than the existing methods and provide an algorithm with a low linear time complexity and a low memory requirement [43]. Moreover, in [66,67], an enhanced version of iForest, named iNNE is proposed, which tries to overcome some drawbacks of the primary method, including the incapability of efficiently detecting local anomalies, anomalies with a low amount of pertinent attributes, global anomalies that are disguised through axis-parallel projections, and finally anomalies in a dataset containing various modals.

Discussion

Almost all of the mentioned studies here for outlier identification assume the input data as static, centralized, and memory-resident. Therefore, they incur an expensive computational cost for large-scale datasets; i.e., they cannot simply scale well to massive data.

Extreme value analysis is commonly applied to some specific cases, where outliers are acknowledged to be existing at the margins, not the sparse internal areas of the dataset; hence, it is not employed for local outlier identification in typical KDD applications [43]. OCSVM is a semi-supervised outlier model and is mostly utilized in "novelty detection" applications, where the input data contains only good (normal) samples, and the primary purpose is to investigate whether new incoming observations — maybe through a stream — match the existing model. Furthermore, there are some cases for OCSVM in which the training data is contaminated with some abnormal instances (anomalies) to evaluate the algorithm robustness with regard to a regularization parameter meaningfully greater than the anticipated outliers portion [44]. However, this does not change the original method essence and its crucial difference with unsupervised models — which are of our main interest here — where there is no need for any information about object labels, and the input data may contain outliers that require to be identified. This study focuses on "outlier detection" in static numerical applications where the outlier labels are only employed while assessing the final detection outcomes.

Information-theoretic techniques are generally established upon analyzing the individual data attributes and the corresponding correlations between pairs of them to build the detection model; this leads to expensive computations as in the best case, the associated computational complexity is $O(np)$, a multiplication of the cardinality by the dimensionality of the data, or even higher [68]. Hence, such approaches mostly fail at processing huge datasets unless they are exerted through a sophisticated manner to cope with massive scales.

Distance-, density-, and clustering-based methods all fall under the proximity-based outlier models. In such methods, the key idea is to recognize outliers as specific points far away from the rest of the data. This causes the critical complexity stress be on the nearest neighbors calculations which may require $O(n^2)$ time, unless an indexing strategy is exerted to expedite the computations and reduce the complexity to, e.g., $O(n \log(n))$ [43]. Nevertheless, even indexing approaches degenerate in high dimensions, and the situation can become aggravated if the underpinning data patterns are not in support of effective pruning rules [51,52]. Moreover, for the graph-based methods, as they rely on the computation of nearest neighbors for each data object, too, the same scenario is repeated.

As a density-based technique, ODRA has specifically a complexity equal to the product of cardinality and dimensionality, which is not suitable for large scales. Cluster-analysis-based work in [22] is introduced as a computer-intensive algorithm, hence not apt for processing huge data sizes. Furthermore, the time complexity of [20] is outlined by the authors as linear with solely the dataset cardinality; however, the first step of the proposed clustering-based algorithm — named *FindCBLOF* — is merely applying the *Squeezer* clustering approach [69] to the data, with a computational complexity being linear with the number of points, the number of dimensions, and the final count of clusters. Consequently, *FindCBLOF* suffers from much more complexity and thus cannot scale favorably with the problem size. Moreover, the distributed works in [34–36] require multiple resources for execution; thus, they are out of our concern in this study as we are concentrating on operating on a single processing unit.

Isolation-based methods enjoy a linear time complexity with a low constant while requiring a minimal amount of memory. However, they are out of the context of the proximity-based models as being solely based upon isolating instances without employing any distance or density gauge to calculate the anomaly scores; hence, they do not fit in our experimental analysis. ORCA, DOLPHIN, and S_p are among the enhanced distance-based techniques for large-scale and disk-resident data that, while requiring a relatively small portion of memory space, claim competing linear arithmetical complicacies. Here, we present a scalable proximity-based algorithm to function on massive data that enjoys a low linear time complexity and operates reasonably well over a restricted memory space.

Remarks

Remark 1. According to the fact that during scalable clustering, we use the Mahalanobis distance measure to assign each object to a minicluster, and besides, the size of temporary clusters is much smaller than that of original clusters, it would be worth mentioning an important matter here. Concerning [41,70–72], in the case of high-dimensional data, classical approaches based on the Mahalanobis distance are usually not applicable. Because when the cardinality of a cluster is less than or equal to its dimensionality, the sample covariance matrix will become singular and not invertible [73]; hence, the corresponding Mahalanobis distance will no longer be reliable.

Therefore, to overcome such a problem, we need to resort to dimensionality reduction approaches in a preprocessing step. However, due to the serious dependence of some dimensionality reduction methods like PCA [74] to the original attributes and the consequent high computational load because of the huge volume of the input data, we need to look for alternative methods to determine a basis for data projection.

A simple and computationally inexpensive alternative is the exercise of random basis projections [75–77]. The main characteristic of these types of methods is that they will approximately preserve pairwise Euclidean distances between data points; in addition, the dimension of the transformed space is independent of the original dimension and only relies logarithmically on the number of data points. Finally, after such a preprocessing step, we can be optimistic that the singularity problem will not be present during the clustering procedures; in the event of happening, we would have a suitable mechanism to handle it.

Remark 2. As stated earlier, our proposed approach is inspired by BFR. However, BFR, by default, uses the K-means algorithm [78] in almost all of its clustering procedures. In addition to this drawback of the K-means algorithm, which is being enormously

dependent on foreknowing the actual number of original clusters, in the case of outliers presence, K-means performs poorly; therefore, we need to resort to a clustering approach that is resistant to anomalies in data. Here in this paper, we prefer to employ the density-based clustering method, DBSCAN [79].[1] Other noise-tolerant clustering algorithms [54,80–94] could be utilized as a substitute option, too, but definitely with their own specific configurations.

However, DBSCAN is strongly reliant upon the choice of its parameters, namely the minimal number of neighbors including itself, *MinPts*, within the range *Eps* — with a random distance measure, which herein is chosen as the Euclidean distance. Thus, we are forced to utilize, e.g., some heuristic or optimization algorithm to find the optimal values for these parameters. Here, we prefer to use an evidence-based approach proposed firstly in the DBSCAN original paper, or a combination of this approach with the Particle Swarm Optimization (PSO) algorithm [95] to locate the optimal parameters. More details can be found in Appendix A.

Remark 3. Another important difference between the proposed method and BFR concerns the volume of structural information, which they need to store for clustering procedures. As the proposed method, differently from BFR, can handle Gaussian clusters with correlated attributes too, thus the covariance matrix will not always be diagonal and could have many non-zero elements. Therefore, the proposed method will consume more space than BFR for building the clustering structures.

Since the Mahalanobis distance criterion is crucially based on the covariance matrix, this matrix will be literally the most prominent property of each subcluster. However, according to the high computational expense of computing the Mahalanobis distance in high-dimension spaces, as in [70,96], we will use the properties of Principal Components (PCs) in the transformed space. Therefore, the covariance matrix of each minicluster will become diagonal, and by transforming each object to the new space of the minicluster, like BFR, we can calculate the Mahalanobis distance without the need to use matrix inversion.

According to [97], when the covariance matrix is diagonal, the corresponding Mahalanobis distance becomes the same normalized Euclidean distance. Moreover, we can establish a threshold value for defining the Mahalanobis radius. If the value of this threshold is, e.g., 4, it means that all points on this radius are as far as four standard deviations from the mean; if we just denote the number of dimensions by p, the size of this Mahalanobis radius is equal to $4\sqrt{p}$.

3. Proposed approach

The proposed method consists of three major phases. In the first phase, a preliminary random sampling is conducted in order to obtain the main premises on which the algorithm works, i.e., some information on the original clusters and some parameters useful for the progressive clustering. In the second phase, a scalable density-based clustering approach is carried out in order to recognize dense areas on the basis of the currently loaded chunk of data points in the memory. Clusters built incrementally in this way are called miniclusters or subclusters, and they form the temporary clustering model. In more detail, after loading each chunk of data, according to the points already loaded in the memory and those undecided from the previous chunks, and by employing the Mahalanobis distance measure and in respect to

---

[1] However, we will demonstrate that sometimes throughout scalable clustering, even DBSCAN may produce some miniclusters that cannot abide outliers because of some data assignment restraints; hence, we will be forced to use the same K-means algorithm to fix the issue.
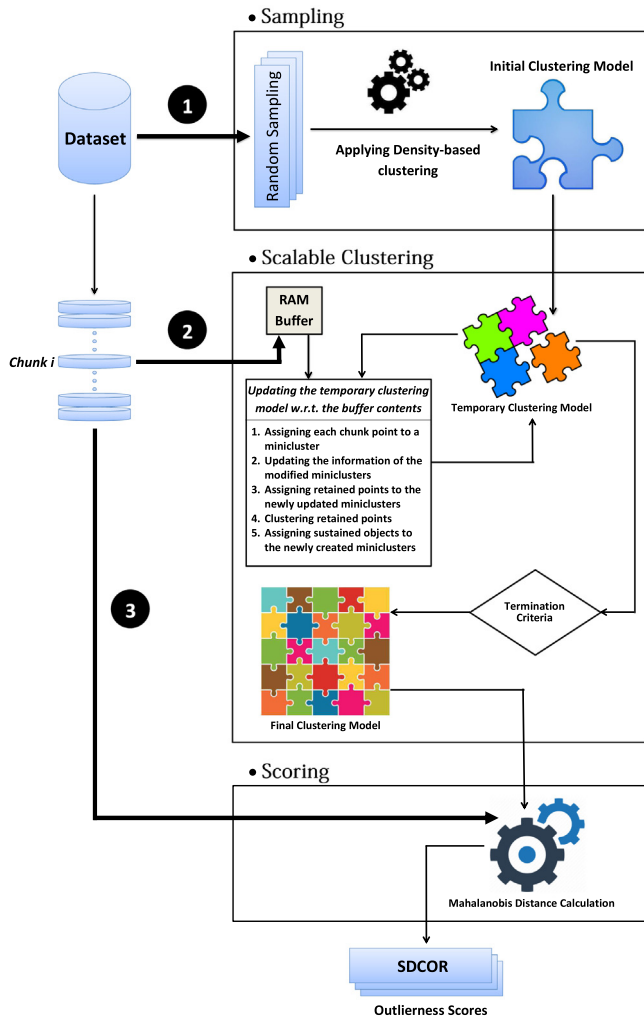
Fig. 1. Software architecture of the proposed approach.

Table 1
Major notations.

Notation	Description
$[\mathcal{X}]_{n \times p}$	Input dataset $\mathcal{X}$ with n objects and p dimensions
$x \in \mathcal{X}$	An instance in $\mathcal{X}$
$\mathcal{S} \subset \mathcal{X}$	A random sample of $\mathcal{X}$
$\mathbb{X} \subset \mathcal{X}$	A set of some points in the buffer
$\mathcal{Y} \subset \mathcal{X}$	A chunk of data
Ω	A partition of some data points in the memory into some miniclusters as $\{\mathbb{X}_1, \ldots, \mathbb{X}_{\Bbbk}\}$
$n_{\mathcal{C}}$	Number of objects in $\mathcal{Y}$
$\{\Gamma\}_{8 \times \mathcal{L}}$	Information array of temporary clusters with 8 properties for each minicluster
$\{\Gamma_i\}_{8 \times 1} \in \Gamma$	Information standing for the i-th minicluster in Γ, $1 \leq i \leq \mathcal{L}$
X_i	Minicluster points associated with Γ_i, which are removed from the buffer
$\Re$	Retained set of objects in the RAM buffer
$\Re_i$	The i-th minicluster of retained points in the buffer, discovered through DBSCAN
$\Re_\zeta$	Retained set of objects in the buffer introduced as noise by DBSCAN
Γ	List of indices to recently created or updated miniclusters associated with Γ, to be checked on for membership
γ'	Temporary list of indices to recently created or updated miniclusters associated with Γ, to be checked on for membership
m	Current number of objects associated with Γ_i
p'	Current Number of superior components associated with Γ_i
e_i	The i-th PC coefficient
λ_i	The i-th PC variance
$\mathcal{L}$	Current number of temporary clusters associated with Γ
$\Bbbk$	Number of miniclusters which are about to be added to the temporary clustering model
$\mathbb{K}$	K-means parameter for the number of clusters
$\mathbb{K}'$	Number of obtained subclusters out of the retained set, discovered through DBSCAN
$\mathbb{K}''$	Number of obtained subdivided subclusters out of the retained set, discovered through K-means
$\mathcal{T}$	True number of original clusters in $\mathcal{X}$
$\nu \in \{1, \ldots, \mathcal{T}\}$	Index to the nearest initial minicluster for a newly discovered subcluster
η	Random sampling rate
Λ	PC total variance ratio for temporary clusters
α	Membership threshold for temporary clusters
β	Pruning threshold for final clusters
Eps	DBSCAN parameter for neighborhood radius
$MinPts$	DBSCAN parameter for minimum number of neighbors involving the query point itself
μ_X	Mean location of cluster X
Σ_X	Covariance structure of cluster X
$\mathbb{S}_X$	Scatter matrix of cluster X
$\mathcal{A}_X$	Transformation matrix of cluster X
z	Object x in the space of eigenvectors
μ'_X	Mean location of cluster X in the space of eigenvectors
$MD(x, X)$	Mahalanobis distance of object x from cluster X
$SingCheck(\Sigma_X)$	A function that checks on the singularity of Σ_X, and outputs 1 in case of being singular and 0 otherwise
$CohrCheck(X)$	A function that checks on the coherence of the input data X, so that it checks whether only one dense cluster will be discovered through DBSCAN. Its output is 1 in case of being coherent, and 0 otherwise
det_{Σ_X}	Covariance determinant of cluster X
$\vec{\delta}$	Vector of maximum covariance determinant condition for miniclusters discovered through scalable clustering
$\{F\}_{2 \times \mathcal{T}}$	Information array of the final clustering model with 2 properties for each final cluster
$\mathcal{M}_i$	The i-th final cluster, comprising of the associated temporary clusters
μ_f	Mean location of a final cluster
Σ_f	Covariance structure of a final cluster
$\mathcal{U}$	A set of regenerated points
$|\cdot|$	Cardinality of a set of objects
Φ	The empty set
κ	A low constant near zero

the density-based clustering criteria, we update the temporary clustering model, which consists of making some changes to the existing miniclusters or adding new subclusters.

Note that, in the whole scalable clustering procedure, our endeavor aims not to let outliers participate actively in forming and updating any minicluster, and thus, after processing the entire chunks, there will be some objects in the buffer remained undecided. Some of these data are true outliers, while others are inliers, which, due to constraints, have failed to play an influential role in forming a subcluster. Finally, all these undecided points are cleared from the buffer, while only the structural information of the temporary clusters is maintained in the memory. Then, at the last part of scalable clustering, depending on the miniclusters associated with each initial minicluster out of the "Sampling" stage, we combine them to obtain the final clusters, which their structure will be approximately the same as of the original clusters.

At last, in the third phase of the proposed approach, w.r.t. the final clustering model gained out of the second phase, once again, we process the entire dataset in chunks to give each object an outlying score, according to the same Mahalanobis distance criterion. Fig. 1 illustrates the software architecture of the approach. Moreover, in Table 1, the main notations used in the paper are summarized.

The framework of the proposed approach is presented in Algorithm 1, which consists of three main phases, including: (1) Sampling; (2) Scalable Clustering; and (3) Scoring. All these phases will be described in detail in the next subsections.

Algorithm 1: Framework of SDCOR

> **Input** : $[\mathcal{X}]_{n \times p}$ - The n by p input dataset $\mathcal{X}$; η - Random sampling rate; Λ - PC total variance ratio; α - Membership threshold; β - Pruning threshold
>
> **Output**: Outlying scores for every object in $\mathcal{X}$

1 Phase 1 — Sampling:

2 *Step 1.* Take a random sample $\mathcal{S}$ from $\mathcal{X}$ according to the sampling rate η

3 *Step 2.* Find the optimal values for the DBSCAN parameters, *Eps* and *MinPts*, required for clustering $\mathcal{S}$, w.r.t. Appendix A

4 *Step 3.* Run DBSCAN on $\mathcal{S}$ using the obtained optimal parameters, and reserve the count of discovered miniclusters as $\mathcal{T}$, as the true number of the original clusters in data

5 *Step 4.* Build the very first array of miniclusters information (temporary clustering model) out of the result of step 3, w.r.t. Algorithm 2

6 *Step 5.* Reserve the covariance determinant values of the initial subclusters as the vector $\vec{\delta} = [\delta_1, \cdots, \delta_{\mathcal{T}}]$, for the maximum covariance determinant condition

7 *Step 6.* Clear $\mathcal{S}$ from RAM and maintain the initial temporary clustering model in the buffer

8 Phase 2 — Scalable Clustering:

9 Prepare the input data to be processed chunk by chunk so that each chunk could be fit and processed in the RAM buffer at the same time

10 *Step 1.* Load the next available chunk of data into RAM

11 *Step 2.* Update the temporary model of clustering over the contents of the buffer, w.r.t. Algorithm 3

12 *Step 3.* If there is any available unprocessed chunk, go to step 1

13 *Step 4.* Build the final clustering model, w.r.t. Algorithm 7, using the temporary clustering model acquired out of the previous steps

14 Phase 3 — Scoring:

15 According to the final clustering model, for each data point $x \in \mathcal{X}$, employ the Mahalanobis distance criterion to find the closest final cluster, and lastly, assign x to that cluster and use the criterion value as the object outlierness score

---

3.1. Sampling

In this phase, we generate a random sample of the entire dataset. As it might initially seem simple, but in actuality, it could be a complicated task since it is not guaranteed that in certain massive databases, records are not following a specific order in some attribute; this could lead to costly scans of the complete dataset to build an efficient sampling. Above all, without a practical random sampling, the modeled clustering structure may not represent the original one; after that, outliers could be misclassified over scalable clustering. In the following, the percentage of the sampled data is indicated as η.

3.1.1. Clustering the sampled data

After obtaining the sampled data, we conduct the DBSCAN algorithm to cluster them w.r.t. the calculated optimal parameters explained in detail in Appendix A. Here, as we apply "uniform random sampling" — also known as "simple random sampling" or "random sampling without replacement" — on the entire data, hence there is equal probability for every individual point to be incorporated in the sample [98]; for this reason, we assume that the number of sampled clusters through this sampling scheme is the same as the number of original clusters, $\mathcal{T}$, in the main dataset.[2] Most of all, this is an accepted truth as also observed in [37,51,53,63,64,66,67,99–101]. Besides, we presume that the location (centroid) and the shape (covariance structure) of such subclusters are so close to the original ones. In Section 5, we will show that even by using a low rate of random sampling, the mentioned properties of a sampled cluster could be quite similar

to those of the original one.[3] The idea behind making these primary subclusters — that are so similar to the original clusters in the input dataset in terms of the basic characteristics — is that we intend to determine a Mahalanobis radius, which collapses a specific percentage of objects belonging to every original cluster and let other subclusters be created around this folding area during successive memory loads of points. Ultimately, by merging these miniclusters, we will obtain the approximate structure of the original clusters.

3.1.2. Building the initial clustering model

To build the initial clustering model, we need to extract some information from the sampled clusters obtained through DBSCAN and store them in a special array. As stated earlier about the benefit of using the properties of principal components for high-dimensional data, we need to find those PCs that give higher contributions to the cluster representation. To this aim, we sort the PCs on the basis of their corresponding variances in descending order, and then we choose the topmost PCs having a share of the total variance at least equal to Λ percent. We call these PCs superior components and denote their number as p'. Let x be an object among the total n objects in the dataset $[\mathcal{X}]_{n \times p}$, belonging to the temporary cluster $[\mathbb{X}_i]_{m \times p}$, then the information about this subcluster as $\{\Gamma_i\}_{8 \times 1}$, in the array of temporary clustering model $\{\Gamma\}_{8 \times \mathcal{L}}$, is as follows:

1. Mean vector in the original space, $\mu_{\mathbb{X}_i} = \frac{1}{m} \sum_{x \in \mathbb{X}_i} x$
2. Scatter matrix in the original space, $\mathbb{S}_{\mathbb{X}_i} = \sum_{x \in \mathbb{X}_i} (x - \mu_{\mathbb{X}_i})^t (x - \mu_{\mathbb{X}_i})$
3. p' superior components, $[e_1, \ldots, e_{p'}]$, derived from the covariance matrix $\sum_{\mathbb{X}_i} = \frac{1}{m-1} \mathbb{S}_{\mathbb{X}_i}$, which form the columns of the transformation matrix $\mathcal{A}_{\mathbb{X}_i}$
4. Mean vector in the transformed space, $\mu'_{\mathbb{X}_i} = \mu_{\mathbb{X}_i} \mathcal{A}_{\mathbb{X}_i}$
5. Square root of the top p' PC variances, $[\sqrt{\lambda_1}, \ldots, \sqrt{\lambda_{p'}}]$
6. Size of the minicluster, m
7. Value of p'
8. Index to the nearest initial minicluster, $\nu \in \{1, \ldots, \mathcal{T}\}$

Algorithm 2 demonstrates the process of obtaining and adding this information per each minicluster attained through either of the "Sampling" or "Scalable Clustering" stages to the temporary clustering model. As for the eighth property of the information array, for every sampled cluster, it is set as the corresponding original cluster number, and for any of the subclusters achieved out of scalable clustering, it is set as the corresponding index to the nearest initial subcluster.[4]

3.1.3. Establishing the scalable clustering criteria

Regarding the major deviation between the sampling and original distributions, DBSCAN cannot be applied to both situations with the same setting. Hence, as it is detailed in Appendix A, w.r.t. the insignificant effect of the *MinPts* parameter on the final clustering results, we use the same value for it in both sampling and original conditions; however, *Eps* is far more alert to small changes, and thus as it is experimentally justified, we decide to

---

[2] We reserve $\mathcal{T}$ for later use.

[3] As we are working in massive scale, there is no need to be worried about minuscule clusters in data which may not have enough sampled points to represent a relatively similar structure to the genuine one; this case generally happens with not very large datasets. Besides, in most cases, tiny bunches of points tend to be outlier classes that should not be able to initiate a cluster.

[4] It is clear that all those sampled points, which the initial clustering model is built upon them, will be met again throughout scalable clustering. Nevertheless, as they are randomly sampled, it could be asserted that they will not impair the final clustering model structure.

Algorithm 2: $[\Gamma]$ = MiniClustMake($\Gamma, \Omega, \Lambda, \nu$)

Input : Γ - Current array of miniclusters information;
$\Omega = \{\mathbb{X}_1, \cdots, \mathbb{X}_{\Bbbk}\}$ - A partition of some data points in the memory into some miniclusters; Λ - PC share of the total variance; ν - Index to the nearest initial minicluster

Output: Γ - Updated temporary clustering model

1   $c \leftarrow \mathcal{L}$
2   **foreach** *minicluster* $\mathbb{X}_i, 1 \leq i \leq \Bbbk$ **do**
3   Apply PCA on $\mathbb{X}_i$ and obtain its PC coefficients and variances. Then choose p' as the number of those top PC variances for which their share of the total variance is at least Λ percent
4     $\Gamma\{1, c + i\} \leftarrow$ Mean vector of $\mathbb{X}_i$
5     $\Gamma\{2, c + i\} \leftarrow$ Scatter matrix of $\mathbb{X}_i$
6     $\Gamma\{3, c + i\} \leftarrow$ Top p' PC coefficients corresponding to the top p' PC variances
7     $\Gamma\{4, c + i\} \leftarrow$ Transformed mean vector, as $\Gamma\{1, c + i\} \cdot \Gamma\{3, c + i\}$
8     $\Gamma\{5, c + i\} \leftarrow$ Square root of the top p' PC variances
9     $\Gamma\{6, c + i\} \leftarrow$ Number of objects in $\mathbb{X}_i$
10    $\Gamma\{7, c + i\} \leftarrow$ Value of p'
11   **if** $c \equiv 0$ **then**   /* "Sampling" stage */
12      $\Gamma\{8, c + i\} \leftarrow i$
13   **else**   /* "Scalable Clustering" stage */
14      $\Gamma\{8, c + i\} \leftarrow \nu$
15   **end**
16 **end**

---

use half of the utilized quantity in the sampling case for the original one during scalable clustering.

Moreover, according to [102,103], when a multivariate Gaussian distribution is contaminated with some outliers, then the corresponding covariance determinant is no longer robust and is significantly more than that of the main cluster. Following this contamination, the corresponding (classical) Mahalanobis contour line[5] will also become broader than that of the pure cluster (robust one), as it contains also abnormal data. So, it makes sense that there is a direct relationship between the value of covariance determinant of an arbitrary cluster and the wideness of its tolerance ellipse, which could also be referred to as the spatial volume of the cluster. Moreover, by being contaminated, this volume could increase and become harmful.

Since during scalable clustering, new objects are coming over time, and miniclusters are growing gradually, so, it is possible for a minicluster with an irregular shape to accept some outliers. Then, the following covariance matrix will no longer be robust, and the corresponding Mahalanobis contour line will keep getting wider too, which could cause the absorption of many other outliers. Therefore, to impede the creation process of these voluminous non-convex subclusters, which could be contaminated with myriad outliers, we have to put a limit on the covariance determinant of every subcluster, which is discovered through scalable clustering. Here, we follow a heuristic approach and employ the covariance determinant of the nearest initial subcluster obtained out of the "Sampling" stage as the limit.

This problem that outliers, through a detection procedure, could be included in normal clusters and no longer be identified is called the "masking effect". Note that we are using the mentioned constraints only when subclusters are created for the first time, not while growing over time. The justification is that while an object is about to be assigned to a minicluster, the other constraint on the Mahalanobis radius, within reason, is hindering outliers from being accepted as a member. In other words, when the Mahalanobis distance of an outlier is more than the predefined radius threshold, it cannot be assigned to the subcluster, if and only if that threshold is set to a fair value.[6] Hence, we do not

---

[5] Since the terms "Mahalanobis contour line" and "tolerance ellipse" are the same in essence, thus we will use them indifferently in this paper.

[6] The three-sigma rule of thumb or the same empirical 68-95-99.7 rule can be taken into account to determine the requisite reasonable thresholds.

---

check on the covariance determinant of subclusters while they are growing.

Here, the first phase of the proposed approach is finished, and we need to clear the RAM buffer of any sampled data and only maintain the very initial information obtained about the existing original clusters.

3.2. Scalable clustering

During this phase, we have to process the entire dataset chunk-by-chunk, as for each chunk, there is enough space in the memory for both loading and processing it all at the same time. After loading each chunk, we update the temporary clustering model according to the data points currently loaded in the memory from the present chunk or retained from the other previously loaded chunks. Finally, after processing the entire chunks, the final clustering model is built out of the temporary clustering model. A detailed description of this phase is provided as follows.

3.2.1. Updating the temporary clustering model w.r.t. the contents of the buffer

After loading each chunk into the memory, the temporary clustering model is updated upon the objects belonging to the currently loaded chunk and other ones sustained from the formerly loaded data. First of all, the algorithm checks for the possible membership of each point of the present chunk to any existing minicluster in the temporary clustering model.[7]

Then, after the probable assignments of the current chunk points, some primary and secondary information of the modified subclusters shall be updated. After this update, the structure of the altered subclusters will change, and thus they might still be capable of absorbing more inliers. Therefore, the algorithm checks again for the likely memberships of sustained points in the memory to the updated subclusters. This updating and assignment checking cycle will keep going until there is not any retained point that could be assigned to an updated minicluster.

When the membership evaluation of the present chunk and retained objects is carried out, the algorithm tries to cluster the remaining sustained objects in the memory, regarding the density-based clustering criteria, which have been constituted at the "Sampling" phase. After the new miniclusters were created out of the last retained points, there is this probability that some sustained inliers in the buffer might not be capable of actively participating in forming new subclusters because of the density-based clustering standards, though could be assigned to them, considering the firstly settled membership measures. Hence, the algorithm goes another time in the cycle of assignment and updating procedure, like what was done in the earlier steps.

Algorithm 3 demonstrates the steps necessary for updating the temporary clustering model, w.r.t. an already loaded chunk of data and other undecided objects retained in the memory from before. The following subsections will explain the details of this algorithm.

3.2.1.1. Trying to assign each tuple of the chunk to a minicluster. After loading each chunk of data into the buffer, we need to use the properties of PCs for each minicluster and transform each data tuple into the new space of that minicluster; then, like BFR, we calculate the Mahalanobis distance using the mean vector and the square root of variances, but in the space of eigenvectors. That is,

$$MD(x, X_i) = \sum_{j=1}^{p'} \left(\frac{z_j - \mu'_j}{\sqrt{\lambda_j}}\right)^2 \tag{1}$$

---

[7] After this step, the unassigned objects of the lastly and previously loaded chunks will be altogether considered as the retained or sustained objects in the buffer.

Algorithm 3: $[\Gamma, \Re] = \text{MemoProcess}(\mathcal{Y}, \Gamma, \alpha, \Re, \vec{\delta}, \Lambda)$

Input : $\mathcal{Y}$ - A chunk of data; Γ - Current array of miniclusters information; α - Membership threshold; $\Re$ - Retained set; $\vec{\delta}$ - Covariance determinant threshold; Λ - PC share of the total variance

Output: Γ - Updated temporary clustering model; $\Re$ - Modified retained set

  /* Trying to assign each tuple of a chunk to a minicluster   */
1   $\gamma \leftarrow \{1, \cdots, \mathcal{L}\}$
2   $[\Gamma, \gamma, \Re] = \text{MiniClustUpdate}(\mathcal{Y}, \Gamma, \alpha, \gamma, \Re)$
  /* Checking out the retained set   */
3   **if** $|\Re| \neq 0$ **then**
    /* Checking on the retained set membership for the recently updated miniclusters   */
4      $[\Gamma, \Re] = \text{RetSetMemb}(\Re, \Gamma, \gamma, \alpha)$
    /* Clustering the retained set   */
5   **if** $|\Re| \neq 0$ **then**
6          $l \leftarrow \mathcal{L}$
7          $[\Gamma, \Re] = \text{RetSetClust}(\Re, \Gamma, \vec{\delta}, \Lambda)$
8          $\gamma \leftarrow \{l+1, \cdots, \mathcal{L}\}$
        /* Checking on the retained set membership for the recently created miniclusters   */
9   **if** $|\Re| \neq 0$ **then**
10            $[\Gamma, \Re] = \text{RetSetMemb}(\Re, \Gamma, \gamma, \alpha)$
11   **end**
12   **end**
13 **end**

Algorithm 4: $[\Gamma, \gamma', \Re] = \text{MiniClustUpdate}(\mathbb{X}, \Gamma, \gamma, \alpha, \Re)$

Input : $\mathbb{X}$ - A set of some points in the buffer; Γ - Current array of miniclusters information; γ - List of indices to the recently created or updated miniclusters associated with Γ, to be checked on for membership; α - Membership threshold; $\Re$ - Retained set

Output: Γ - Updated temporary clustering model; $\gamma' \subseteq \gamma$ - Modified list of indices to the recently updated miniclusters; $\Re$ - Modified retained set

  /* Updating the primary information of subclusters   */
1   **foreach** $x \in \mathbb{X}$ **do**
2      $b \leftarrow \text{argmin}_{i \in \gamma} MD(x, X_i)$
3   **if** $MD(x, X_b) \leq \alpha \sqrt{\Gamma_b \{7\}}$ **then**
4          $\Gamma_b \{2\} \leftarrow \Gamma_b \{2\} + x'x$
5          $\Gamma_b \{6\} \leftarrow \Gamma_b \{6\} + 1$
6   Remove x from the RAM buffer
7   **else**
8          $\Re \leftarrow \Re \cup x$
9   **end**
10 **end**
  /* Updating the secondary information of subclusters   */
11 $\gamma' \leftarrow \Phi$
12 **foreach** $X_i, i \in \gamma$ **do**
13   **if** X_i *has accepted any new members* **then**
14   Obtain its updated covariance matrix $\sum_{X_i}$, through normalizing its updated scatter matrix, w.r.t. the current size of the minicluster as $\left(\frac{1}{\Gamma_i \{6\} - 1} \right) \cdot \Gamma_i \{2\}$
15   Apply PCA on $\sum_{X_i}$ to acquire its eigenvalues and eigenvectors, and then, update Γ_i as follows:
16          $\Gamma_i \{7\} \leftarrow$ Value of p' as the updated number of superior components
17          $\Gamma_i \{3\} \leftarrow$ Updated superior coefficients
18          $\Gamma_i \{4\} \leftarrow$ Updated transformed mean vector, as $\Gamma_i \{1\} \cdot \Gamma_i \{3\}$
19          $\Gamma_i \{5\} \leftarrow$ Square root of the updated superior variances
20          $\gamma' \leftarrow \gamma' \cup i$
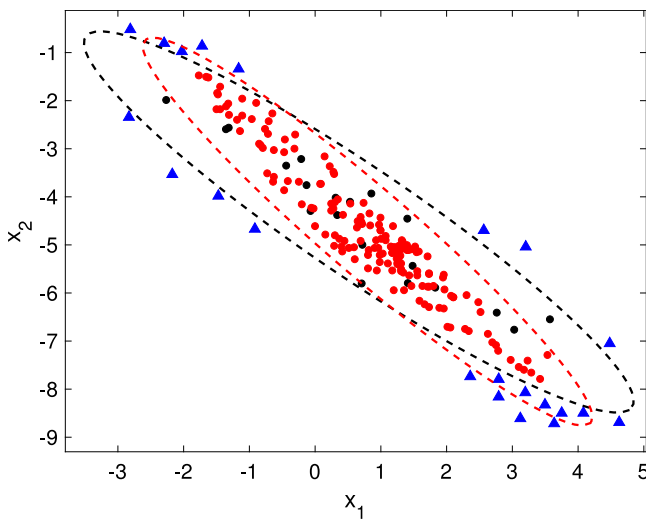21   **end**
22 **end**

Fig. 2. Assigning retained objects in the buffer to the newly updated subclusters.

Where $MD(x, X_i)$ is the Mahalanobis distance of object x from minicluster X_i; $z = x\mathcal{A}_{X_i}$ is the object in the eigenvector space of the minicluster, and z_j is its j-th component; μ'_j and λ_j are respectively the j-th components of the mean vector and the variance vector in the space of eigenvectors; finally, p' is the number of superior components associated with the minicluster. As stated above, in this style, the amount of computations is sensibly less than if we would have used matrix inversion to calculate the distance. Moreover, w.r.t. [97], the accepted Mahalanobis radius in the eigenvector space of the relevant subcluster will be the product of the membership threshold and the square root of the number of dimensions in the transformed space, as $\alpha\sqrt{p'}$.

For each data point, w.r.t. Eq. (1), we need to find the closest minicluster and check whether or not it falls in the accepted Mahalanobis threshold of that minicluster; if it does, some information connected to the corresponding subcluster shall be updated.

3.2.1.2. Updating primary and secondary information of the modified temporary clusters. For every subcluster that any new objects have been assigned to it, there are two types of information that need to be updated; primary and secondary. Primary information comprises the scatter matrix of the subcluster and its cardinality, which should be updated after each individual assignment. To update the scatter matrix, the outer product of the belonged data point with itself is added to the current scatter matrix; for the cardinality, the number of objects assigned to the subcluster is increased by one. Every object, after joining a subcluster, is removed from the buffer; otherwise, it will be retained to be decided on later.

After checking on the membership of all points and updating the primary information of the modified subclusters, for each minicluster that has accepted any new members, its PC properties, which are considered as its secondary information, must be updated too. For this purpose, according to the minicluster size, we normalize its scatter matrix in an unbiased manner to acquire its covariance matrix. Then, by applying PCA on this matrix, we update the transformation matrix, the mean vector in the space of eigenvectors, and the superior PC variances of the minicluster. Algorithm 4 demonstrates the required steps for finding the closest subcluster per every query point w.r.t. the relevant limitations, and updating the essential information of every modified subcluster.

3.2.1.3. Trying to assign retained objects in the buffer to the newly updated miniclusters. After updating the secondary information of each altered subcluster, the corresponding tolerance ellipse will rotate and might also become skewed a bit around the centroid; in other words, the following accepted Mahalanobis
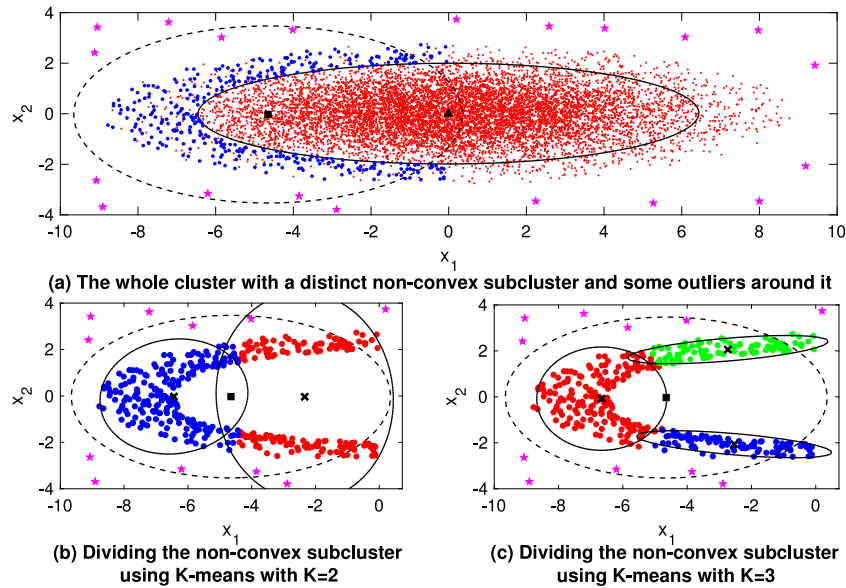
(a) The whole cluster with a distinct non-convex subcluster and some outliers around it

(b) Dividing the non-convex subcluster
using K-means with K=2

(c) Dividing the non-convex subcluster
using K-means with K=3

Fig. 3. Breaking a non-convex minicluster with a very wide tolerance ellipse into smaller pieces by the K-means algorithm.[8]

Algorithm 5: $[\Gamma, \Re] = \text{RetSetMemb}(\Re, \Gamma, \gamma, \alpha)$

Input : $\Re$ - Retained set; Γ - Current array of miniclusters information; γ - List of indices to the recently created or updated miniclusters associated with Γ, to be checked on for membership; α - Membership threshold

Output: Γ - Updated temporary clustering model; $\Re$ - Modified retained set

1 **if** $|\gamma| \neq 0$ **then**
2   **while** *true* **do**
3        $[\Gamma, \gamma, \Re] = \text{MiniClustUpdate}(\Re, \Gamma, \gamma, \alpha, \Phi)$
4   **if** $|\gamma| \equiv 0$ **then**
5 | break
6   **end**
7   **end**
8 **end**

---

neighborhood is modified. Hence, w.r.t. Algorithm 5, it would be necessary to check on the objects retained in the buffer, i.e., whether they can belong to a modified minicluster; and if it is so, the corresponding minicluster information needs to be updated, w.r.t. Algorithm 4. Nevertheless, this is not the end; by keeping up this cycle of membership checking and minicluster updating, more and more objects could be assigned to miniclusters and then be discarded. In this iterative manner, after each iteration, memory contents should be evaluated using only the updated subclusters from the last iteration; thus, the list of updated miniclusters will be shrinking over time until it becomes an empty list. This means that there is not any other sustained object in the buffer, which falls in the accepted Mahalanobis threshold of any of the updated miniclusters, or every retained object has been eventually assigned to an updated minicluster. Here, by employing this procedure, it seems that the tolerance ellipses of miniclusters are sweeping inliers through the cycle of assignment and updating.

Fig. 2 shows the scenario in which, after updating the core information of a subcluster, its Mahalanobis neighborhood is modified; some objects which were not able to belong to this subcluster now are capable of being assigned to it. Black circle points and the black dashed line tolerance ellipse respectively represent a subcluster and its Mahalanobis neighborhood. Red circle points represent objects assigned to the subcluster during the last memory process, as they reside in its accepted neighborhood.

The red dashed line represents the updated tolerance ellipse of the updated subcluster. Moreover, blue triangle points represent objects which could be assigned to the updated subcluster if they would lie in its updated Mahalanobis radius.

3.2.1.4. Clustering retained objects in the buffer. Here, after checking on the membership of every data tuple stored in the memory, we afford to cluster retained objects in the RAM buffer, using the DBSCAN algorithm again. However, as emphasized at the "Sampling" stage, according to the significant difference in the density of the sampled and original data, we have to employ a specific configuration for each situation.

Furthermore, as it was described before, it is possible that some miniclusters could be discovered during scalable clustering by DBSCAN, which are suffering from the singularity problem. Thus, for handling such a situation, there are some ways. One is to use the pseudoinverse of the covariance structure, but it is not totally accurate. The better way is to disregard such minicluster and let its points still be in the memory, to be resolved later, among other coming points. Therefore, for every discovered subcluster, we shall check on its covariance matrix, whether or not it is singular; in case of singularity, we disregard that subcluster.

Now, w.r.t. this prementioned matter that, we have to put a limit on the boundaries of the miniclusters which are being created throughout scalable clustering, we are going to demonstrate with an intuitive example that if a minicluster with a non-convex shape is formed, how outliers could be absorbed to such irregular minicluster and cause severe damage to the final clustering results.

Fig. 3a illustrates the structure of an original cluster represented with red dots, with a newly discovered non-convex subcluster shown with blue dots, and a black square and a black dashed line as its centroid and accepted tolerance ellipse respectively. The irregular minicluster is formed around the initial minicluster, while its centroid and accepted Mahalanobis radius are denoted as a black triangle and a solid black line, respectively.

---

[8] Regarding the three-sigma rule of thumb, the Mahalanobis radii equal to 1 and 2, in order, cover roughly 68 and 95 percent of the total objects in a Gaussian distribution. For convex-shaped clusters of other distributions, the amount of coverage might vary, but for non-convex-shaped clusters, it could contain objects not belonging to the distribution. Here, in all subfigures, the presented radius is equal to 1.5.

There are also some local outliers around the original cluster, which are illustrated with magenta pentagons.[9]

As it is evident, the irregular minicluster can absorb some local outliers, as its tolerance ellipse is covering a remarkable space out of the containment area by the original cluster. Moreover, the covariance determinant value for the irregular minicluster is equal to 15.15, which is almost twice that of the initial minicluster equal to 7.88. Thus, to fix this concern, by considering the proportion between the covariance determinant of an arbitrary cluster and its spatial volume, we decide to divide the irregular minicluster into smaller coherent pieces, with more minor covariance determinants and more limited Mahalanobis radii as well. Therefore, we heuristically set the threshold value for the covariance determinant of any newly discovered subcluster or subdivided subcluster, as that of the nearest initial minicluster.[10]

For the division process of an irregular subcluster, we prefer to adopt the K-means algorithm. However, K-means can cause some incoherent subdivided subclusters in such cases,[11] as shown in Fig. 3b. In Fig. 3b, two smaller miniclusters, produced as the K-means result, are represented in different colors, with covariance determinants of 1.63 and 7.71 for the coherent blue and incoherent red miniclusters, respectively. The associated centroids and tolerance ellipses are denoted as black crosses and black solid lines, respectively. It is clear that even incoherent subdivided subclusters, with a covariance determinant less than or equal to the predefined threshold though, could be hazardous as non-convex subclusters with a high value of covariance determinant, as their tolerance ellipses could get out of the scope of the following main cluster, and suck outliers in.[12]

An alternative for this is to use hierarchical clustering algorithms, which typically present a higher computational load than K-means. For this matter, we decide to adopt a K-means variant, which, for every subdivided subcluster obtained through K-means, we apply DBSCAN again to verify its cohesion. Finally, we increase the value of $\mathbb{K}$ for K-means, from 2 till a value for which,[13] three conditions for every subdivided subcluster are met: to be coherent, not to be singular, and having a covariance determinant less than or equal to δ_i.

Fig. 3c illustrates a scenario in which three smaller miniclusters are represented as the K-means output in different colors; centroids and tolerance ellipses are shown as in Fig. 3b. The covariance determinant values are equal to 0.12, 1.02, and 0.10 for respectively the green, red and blue subdivided subclusters. As it is obvious, all subdivided subclusters are coherent and not singular, with much smaller determinants than the threshold and much tighter spatial volumes as such. Ultimately, after attaining acceptable subclusters, it is time to update the temporary clustering model w.r.t. them, in regard to Algorithm 2.

---

[9] Here, for challenging the performance of our method, we are taking outliers so close to the original cluster. However, in reality, it is not usually like that, and outliers often have a significant distance from every normal cluster in data.

[10] This threshold is denoted as $\delta_i, 1 \leq i \leq \mathcal{T}$, for any subcluster discovered near the i-th initial minicluster, through scalable clustering. The proximity measure for this nearness is the Mahalanobis distance of the newly discovered subcluster centroid from the initial miniclusters. We assume any subcluster with a covariance determinant greater than such threshold, as a candidate for a non-convex subcluster, whose spatial volume could cover some significant space out of the scope of the related original cluster.

[11] As K-means focuses solely on finding the best locations for the means and does not consider the cohesion of the output clusters.

[12] However, the divided minicluster could be significantly smaller in size and determinant, though because of the lack of coherency, some PC variances could be substantially larger than others; therefore, the regarding tolerance ellipse will be more stretched in those PCs, and thus harmful.

[13] Here, the upper bound for $\mathbb{K}$ is $\lfloor |\Re_i| / (p + 1) \rfloor$, to avoid the singularity problem for every subdivided subcluster of retained objects. $|\Re_i|$ stands for the cardinality of the i-th minicluster of retained points.

Algorithm 6 shows all the steps required to cluster the data elements retained in the memory buffer. First, DBSCAN is applied to the retained set with the optimal parameters concerning the original distribution; some miniclusters, maybe along with some noisy points, will be identified. Then for every detected minicluster out of the sustained objects, we evaluate whether it qualifies for being considered as a valid minicluster conforming with the three conditions of being coherent, not being singular, and not exceeding the determinant threshold. At first, the singularity term is checked out, and in the case of being singular, the related minicluster points are discarded; otherwise, the closest initial minicluster will be found through Mahalanobis distance, and the corresponding covariance determinant will be exerted as the required constraining threshold.

If the determinant limitation is not violated, then such minicluster is qualified for being added to the temporary clustering model, and thereafter, its points in the memory will be obviated. Conversely, in the instance of determinant violation, the K-means variant is utilized for the division strategy, as for every subdivided subcluster, the mentioned three indispensable conditions shall be met. In case that K-means fails, the correspondent unbroken minicluster is discarded; oppositely, if K-means succeeds, the authentic subdivided subclusters are appended to the temporary clustering model. At last, the initial retained set passed to Algorithm 6 is replaced with the entire undecided objects after the described procedure.

Algorithm 6: $[\Gamma, \Re] = \text{RetSetClust}(\Re, \Gamma, \vec{\delta}, \Lambda)$

Input : $\Re$ - Retained set; Γ - Current array of miniclusters information; $\vec{\delta}$ - Covariance determinant threshold; Λ - PC share of the total variance

Output: Γ - Updated temporary clustering model; $\Re$ - Modified retained set

1 Apply the DBSCAN algorithm to $\Re$ concerning the optimum parameters for the original distribution explained in Appendix A. Consider the result of such clustering as $\{\Re_1, \cdots, \Re_{\mathbb{K}'}\} \cup \Re_\zeta$
   /* Adding the information of the newly discovered miniclusters to the temporary clustering model   */

2 **foreach** $\Re_i, 1 \leq i \leq \mathbb{K}'$ **do**

3   **if** $SingCheck(\Sigma_{\Re_i}) \equiv 1$ **then**   /* Singularity check */

4       $\Re_\zeta \leftarrow \Re_\zeta \cup \Re_i$

5   continue

6   **end**

7    $\nu \leftarrow \text{argmin}_{h \in \{1, \cdots, \mathcal{T}\}} MD(\mu_{\Re_i}, X_h)$

8   **if** $det_{\Sigma_{\Re_i}} > \vec{\delta}(\nu)$ **then**   /* Irregular minicluster */

9   Apply K-means with the number of clusters $2 \leq \mathbb{K}'' \leq \lfloor |\Re_i| / (p + 1) \rfloor$ to $\Re_i$. Find the minimum value for $\mathbb{K}''$, as by which for every subdivided subcluster $\Re_{i,j}, 1 \leq j \leq \mathbb{K}''$, we have $CohrCheck(\Re_{i,j}) \equiv 1$, $SingCheck(\Sigma_{\Re_{i,j}}) \equiv 0$ and $det_{\Sigma_{\Re_{i,j}}} \leq \vec{\delta}(\nu)$

10   **if** $such$ $\mathbb{K}''$ is not $found$ **then**

11          $\Re_\zeta \leftarrow \Re_\zeta \cup \Re_i$

12   continue

13   **end**

14       $[\Gamma] = \text{MiniClustMake}(\Gamma, \{\Re_{i,1}, \cdots, \Re_{i,\mathbb{K}''}\}, \Lambda, \nu)$

15   Remove $\{\Re_{i,1}, \cdots, \Re_{i,\mathbb{K}''}\}$ from the RAM buffer

16   **else**   /* Regular minicluster */

17       $[\Gamma] = \text{MiniClustMake}(\Gamma, \Re_i, \Lambda, \nu)$

18   Remove $\Re_i$ from the RAM buffer

19   **end**

20 **end**
   /* Setting the unresolved points as the retained set   */

21 $\Re \leftarrow \Re_\zeta$

3.2.1.5. Trying to assign retained objects in the buffer to the newly created miniclusters. After checking on retained objects in the buffer in case of being capable of forming a new minicluster, it would be necessary to examine the remaining retained objects once more, w.r.t. Algorithm 5; i.e., it ought to be inspected whether or not they could be assigned to the newly created
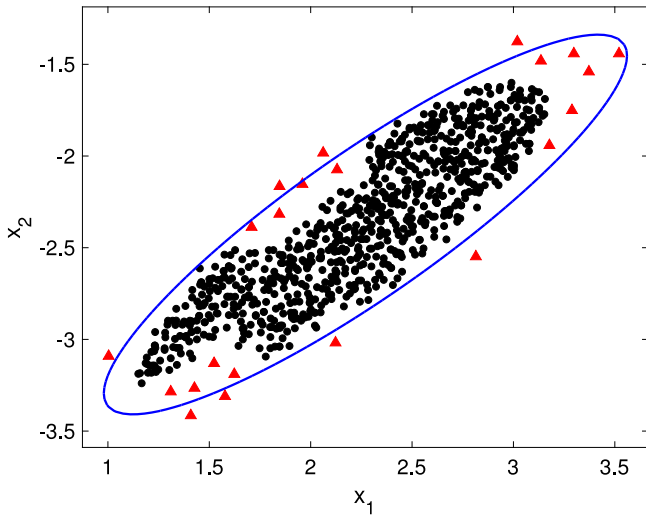
Fig. 4. Assigning retained objects in the buffer to the newly created subclusters.

miniclusters, in the same cycle of membership checking and minicluster updating, like what was done in Section 3.2.1.3. The reason for this concern is that, due to limitations connected to the utilized density-based clustering algorithm, such objects may not have been capable of being an active member of any of the newly created subclusters, even though they lie in the associated accepted Mahalanobis radius. Hence, it becomes essential to check again the assignment of these latter retained objects.

Fig. 4 demonstrates an intuitive example of such a situation in which some objects according to the density restrictions cannot be assigned to a cluster, even though they lie in the accepted Mahalanobis radius of that cluster. Objects that have had the competence to form a cluster are shown with solid black circles; those that are not a part of the cluster, but reside in its accepted Mahalanobis radius represented by a solid blue line, are denoted as red triangles.

Thus, if retained objects could belong to a newly created subcluster, the corresponding information of that subcluster will be modified w.r.t. Algorithm 4; otherwise, such data will be still retained in the buffer for further process. However, if it were the last chunk that was processed, all these retained objects would be marked as temporary outliers. But, all of these temporary outliers are not true outlying points; as stated earlier, some of them are normal objects that have not found the competence of forming a minicluster or being assigned to one due to applied restrictions. Anyway, at last, all of these true and untrue anomalies which are maintained in the buffer will be discarded, and this is only the temporary clustering model which is remained after all.

3.2.2. Building the final clustering model

Here, at the end part of scalable clustering, we follow Algorithm 7 to construct the final clustering model or the same approximate structure of $\mathcal{T}$ original clusters, w.r.t. the temporary clustering model with $\mathcal{L}$ miniclusters.

For every individual minicluster in the temporary clustering model, we have already realized to which original cluster in the main distribution it belongs; thus, to obtain the final clustering model, we just need to merge the information of the relevant miniclusters to each of the original clusters to acquire the final clusters. Here, we presume that the core information of each final cluster only consists of a centroid μ_f, and a covariance matrix Σ_f. Hence, w.r.t. Algorithm 7, if a final cluster contains only one temporary cluster, then the final centroid and the final covariance matrix will be the same as for the temporary cluster. Otherwise,

---

Algorithm 7: $[\mathcal{F}] = \text{FinalClustBuild}(\Gamma, \eta, \beta)$

Input : Γ - Current array of miniclusters information; η - Random sampling rate; β - Pruning threshold
Output: $\mathcal{F}$ - Final clustering model

1   Consider the arrangement of the entire temporary clusters X_j's, $1 \leq j \leq \mathcal{L}$ w.r.t. the original clusters in the main distribution as $\{\mathcal{M}_1, \cdots, \mathcal{M}_\mathcal{T}\}$; $\mathcal{M}_i$, $1 \leq i \leq \mathcal{T}$ stands for the i-th final cluster, consisting of the associated temporary clusters as every temporary cluster possesses a property declaring the original cluster it belongs to

2   $\mathcal{F} \leftarrow \Phi$

3   **foreach** $\mathcal{M}_i$, $1 \leq i \leq \mathcal{T}$ **do**

4   **if** $|\mathcal{M}_i| \equiv 1$ **then**   /* Isolated minicluster */

5   Use the same mean location and covariance structure of the isolated subcluster, as for those of the final cluster

6   **else**   /* Group of miniclusters */
      /* Calculating the final mean location   */

7       $\mu_f \leftarrow \dfrac{\sum_{j:X_j \in \mathcal{M}_i} [\Gamma_j\{6\} \cdot \Gamma_j\{1\}]}{\sum_{j:X_j \in \mathcal{M}_i} \Gamma_j\{6\}}$
      /* Calculating the final covariance structure   */

8       $\mathcal{U} \leftarrow \Phi$

9   **foreach** $X_j \in \mathcal{M}_i$ **do**

10   Regenerate $\eta \cdot \Gamma_j\{6\}$ number of fresh data points, with Gaussian distribution, with regard to $\Gamma_j\{1\}$ as the temporary mean and $\left(\frac{1}{\Gamma_j\{6\}-1}\right) \cdot \Gamma_j\{2\}$ as the temporary covariance matrix

11   Add these regenerated points to $\mathcal{U}$

12   **end**

13   Calculate the Mahalanobis distance of the points in $\mathcal{U}$ based upon the sample mean $\mu_\mathcal{U}$, and the sample covariance matrix $\Sigma_\mathcal{U}$

14   Prune $\mathcal{U}$ by discarding those points with a Mahalanobis distance more than $\beta\sqrt{p}$, and recalculate $\Sigma_\mathcal{U}$

15       $\Sigma_f \leftarrow \Sigma_\mathcal{U}$

16   Remove $\mathcal{U}$ from the buffer
      /* Adding information to the final clustering model   */

17       $\mathcal{F}\{1, i\} \leftarrow \mu_f$

18       $\mathcal{F}\{2, i\} \leftarrow \Sigma_f$

19   **end**

20   **end**

---

in the case of containing more than one temporary cluster, we utilize the sizes and centroids of the associated miniclusters to obtain the final mean.

For acquiring the final covariance matrix of a final cluster comprised of multiple temporary clusters, for each of the associated miniclusters and w.r.t. its centroid and covariance structure, we afford to regenerate a specific amount of fresh data points with Gaussian distribution. We define the regeneration size of each minicluster equal to the product of the sampling rate — which was used at the "Sampling" stage — and the cardinality of that minicluster, as $\eta |X_j|$; this is necessary for saving some free space in the memory while regenerating the approximate structure of an original cluster. We consider all regenerated objects of all subclusters belonging to a final cluster as a unique and coherent cluster and afford to obtain the final covariance structure out of it.

But before using the covariance matrix of such regenerated cluster, we need to mitigate the effect of some generated outliers, which could be created unavoidably during the regeneration process, and can potentially prejudice the final accuracy outcomes. For this purpose, we need to prune this transient final cluster, according to the Mahalanobis threshold β, obtained through the user. Thus, regenerated objects having a Mahalanobis distance of more than $\beta\sqrt{p}$ from the regenerated cluster will be obviated. Now, we can compute the ultimate covariance matrix out of such pruned regenerated cluster and then remove this transient cluster. This procedure is conducted in sequence for every final cluster, consisting of more than one temporary cluster.

Fig. 5a demonstrates a dataset consisting of two dense Gaussian clusters with some local outliers around them. This figure

is, in fact, a sketch of what the proposed method looks like at the final steps of scalable clustering and before building the final clustering model. Green dots are normal objects belonged to a minicluster, and red dots are temporary outliers; magenta square points represent the temporary centroids. Fig. 5b demonstrates both temporary means and final means, represented by solid circles and triangles respectively, with a different color for each final cluster. Fig. 5c colorfully demonstrates the pruned regenerated data points for every final cluster besides the final means, denoted as dots and triangles, respectively.

Now, after obtaining the final clustering model, the second phase of the proposed approach is finished. In the following subsection, the third phase, named "Scoring" is presented, and we will describe how to give every object a score of outlierness, w.r.t. the final clustering model obtained out of scalable clustering.

3.3. Scoring

In this phase, w.r.t. the final clustering model obtained through scalable clustering, we give each data point an outlying rank. Therefore, like phase two, once more, we need to process the whole dataset in chunks and use the same Mahalanobis distance criterion to find the closest final cluster to each object. This local Mahalanobis distance [42] is assigned to the object as its outlying score; the higher the distance, the more likely it is that the object is an outlier. Here, we name such score obtained out of our proposed approach, SDCOR, which stands for "Scalable Density-based Clustering Outlierness Ratio".[14]

3.4. Algorithm complexity

Here, at first, we analyze the time complexity of the proposed approach. For the first two phases, "Sampling" and "Scalable Clustering", the most expensive operations are the application of DBSCAN to the objects residing in the memory after loading each chunk, and the application of PCA to each minicluster to obtain and update its secondary information.

Let $n_{\mathfrak{c}}$ be the number of data points contained in a chunk. Considering the three-sigma rule of thumb, in every memory load of points, the majority of these points lies in the accepted Mahalanobis neighborhood of the current temporary clusters and is being assigned to them — and this will escalate over time by the increasing number of subclusters, which are being created during every memory process — and also, by utilizing an indexing structure for the kNN queries, the time complexity of the DBSCAN algorithm will be $O\left(\kappa n_{\mathfrak{c}} \log\left(n_{\mathfrak{c}}\right)\right)$; where κ is a low constant close to zero. Note that exercising an indexing structure to decrease the computational complexity of answering the kNN queries is only applicable in lower dimensions; for high-dimensional data, we need some sort of sequential scans to handle the kNN queries, which leads to average time complexity of $O\left(\kappa n_{\mathfrak{c}}^2\right)$ for DBSCAN.

Applying PCA on miniclusters is $O\left(min\left(p^3, n_{\mathfrak{c}}^3\right)\right)$ [103], as p stands for the dimensionality of the input dataset. But according to our necessary assumption that $p < n_{\mathfrak{c}}$, thus applying PCA will be $O\left(p^3\right)$. Hence, the first two phases of the algorithm will totally take $O\left(max\left(\kappa n_{\mathfrak{c}}^2, p^3\right)\right)$.

The last phase of the algorithm, w.r.t. this concern that only consists of calculating the Mahalanobis distance of the total n objects in the input dataset to $\mathcal{T}$ final clusters, is $O\left(n\mathcal{T}\right)$; regarding that $\mathcal{T} \ll n$, hence, the time complexity of this phase will be $O\left(n\right)$. The overall time complexity is thus at most

$O\left(max\left(\kappa n_{\mathfrak{c}}^2, p^3\right) + n\right)$. However, in addition to κ being a truly tiny constant, it is evident that both p and $n_{\mathfrak{c}}$ values are negligible w.r.t. n; therefore, we can state that the time complexity of our algorithm is linear.

Analysis of the algorithm space complexity is twofold. First, we consider the stored information in the memory for the clustering models, and second, the amount of space required for processing the resident data in RAM. Concerning the fact that the most voluminous parts of the temporary and of the final clustering models are the scatter and covariance matrices, respectively, and that $\mathcal{L} \gg \mathcal{T}$, thus, the space complexity of the first part will be $O\left(\mathcal{L}p^2\right)$. For the second part, according to this matter that in each memory load of points, the most expensive operations belong to the clustering algorithms, DBSCAN and K-means, and regarding the linear space complexity of these methods, hence, the overall space complexity will be $O\left(n_{\mathfrak{c}} + \mathcal{L}p^2\right)$.

4. Experimental evaluation

This section supplies details on the efficacy, efficiency, and other specific tests employed in our experiments. For the effectiveness test, we analyze the accuracy and stability of the proposed method on some real-life and synthetic datasets, compared to some other state-of-the-art and well-known distance-, density-, and clustering-based methods, which are all subsets of the proximity-based methods for outlier identification. The representative competing strategies are namely ORCA, DOLPHIN, and S_p as the distance-based methods; LOF, LoOP, and an ensemble version of LOF (introduced in [67]) called EnLOF[15] as the density-based techniques; and finally, a fast K-means variant optimized for clustering large-scale data, named X-means [105] as the only clustering-based approach in our analysis,[16] for which we will report a special benchmarking against SDCOR w.r.t. some exclusive clustering robustness criteria.

Moreover, for the competing methods that are based on some sort of random sampling and present non-deterministic results over various iterations — namely S_p, EnLOF, and SDCOR — we provide some statistical significance test to find differences among them across several attempts. Consequently, for the efficiency test, experimentation is conducted on some artificial datasets to assess how the final accuracy varies when the number of outliers increases. Besides, more investigations are carried out to appraise the scalability of the proposed algorithm and the effect of random sampling rate on the final detection validness.

All the experiments were executed on a laptop having a 2.5 GHz Intel Core i5 processor, 6 GB of memory, and a Windows 7 Ultimate (Service Pack 1) operating system. Furthermore, in what follows, first of all, we explain in detail the utilized evaluation metrics and the analyzed real-world and artificial datasets in our analysis. Moreover, we provide some ins and outs on the algorithms implementations and the user-defined parameters for all contending methods.

---

[14] Due to the high computational load associated with calculating the Mahalanobis distance in high dimensions, one can still gain the benefit of using the properties of principal components for computing the outlying scores, like what was done during scalable clustering.

[15] It should be noted that another ensemble version of LOF has been presented by Zimek et al. [104], though with different parameterization. However, this approach produces similar outputs to EnLOF, it is empirically proven in [67] that this peculiar assembly adaptation of LOF suffers from a much higher arithmetical cost, even in most cases in comparison with the original LOF method.

[16] Actually, X-means is not an anomaly recognition technique in essence as it assumes the input data free of noise. Therefore, after obtaining the final clustering outcome through this method, the Euclidean distance of every object to its closest centroid is assigned to it as an outlier score; hence, the following assessment measures could be calculated.
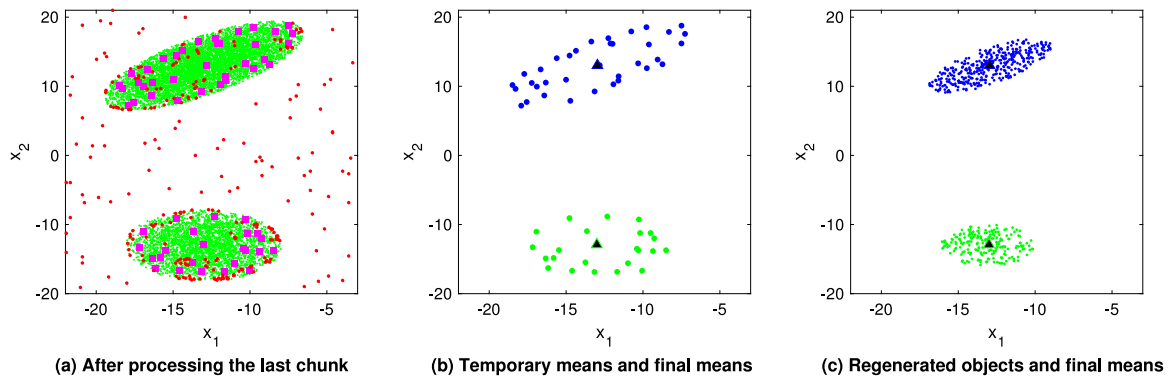
Fig. 5. Proposed method appearance at the last steps of scalable clustering.

4.1. Evaluation metrics

4.1.1. General-purpose metrics

We evaluate the functionality of the competing outlier detection algorithms in general by cross-checking two widely-used metrics in this area [44,106]. One is the Receiver Operating Characteristic, also the Receiver Operating Curve (ROC), or the same curve of detection rate and false alarm rate (true positive rate against false positive rate), calculated concerning Eqs. (2) and (3); the other is the Precision–Recall (PR) curve, computed in respect of Eqs. (2) and (4). In both metrics, the positive class (class of interest) stands for the outliers, while the negative class typifies the normal samples. The examination is then established on the Area Under the Curve (AUC) for both metrics, which are called, in order, AUROC (Area Under Receiver Operating Curve) and AUPRC (Area Under Precision–Recall Curve); more importantly, it should be pointed out that throughout the literature, AUROC is much more common and popular than AUPRC for comparing the anomaly identification tasks. The related equations for computing the referred measures are as follows:

$$TPR\ (Recall) = \frac{true\ positives}{true\ positives + false\ negatives} \quad (2)$$

$$FPR = \frac{false\ positives}{false\ positives + true\ negatives} \quad (3)$$

$$Precision = \frac{true\ positives}{true\ positives + false\ positives} \quad (4)$$

4.1.2. Clustering validity metrics

Besides the above-mentioned two popular assessment measures for unsupervised outlier detection methodologies, as the proposed method is essentially founded upon the cluster analysis, and also, X-means is the only clustering-based competing method in our experiments, hence we desire to compare these two techniques, in particular, utilizing some external clustering validity measures employed in [107,108].[17]

Here, we exert five specific measures: Purity [109], the Mirkin metric [110], F-measure [111], Entropy [112], and Variation of Information (VI) [113] to weigh up the difference among the clustering results of SDCOR and X-means. These measures are divided into two separate collections. In one category — containing Purity, the Mirkin metric, and F-measure — it depends on computing the number of pairs of points on which two clustering outcomes coincide/collide with each other. The other category — including Entropy and VI — is information-reliant, and more precisely, it is the homogeneity of a clustering solution that goes under evaluation. Furthermore, each one of these measures happens at a certain value as its best condition; some fall into a minimum equal to 0, and some drive into a maximum equal to 1 in the ideal case. Consider that dataset $\mathcal{X}$ with n objects composes, in reality, of classes $D = (D_1, \ldots, D_l)$, and the result of applying an arbitrary clustering approach on $\mathcal{X}$ is as $C = (C_1, \ldots, C_k)$; in both the real situation and the clustering output, anomalies, like a group of normal objects, are categorized as a single group of elements[18]. More details on the noted external clustering accuracy metrics are presented in the following.

4.1.2.1. Purity. Every output cluster $C_r, 1 \leq r \leq k$ may include elements associated with multiple target classes. The Purity of an arbitrary cluster is the ratio of the size of the domineering class in the cluster to the cluster cardinality, delineated as follows:

$$Purity\ (C_r) = \frac{1}{|C_r|} \max_{s=1,\ldots,l} |C_r \cap D_s|, \quad 1 \leq r \leq k \quad (5)$$

The Purity value of a single cluster always falls in the interval $\left[\frac{1}{l}, 1\right]$; higher Purity rates denote that the relevant cluster is a more "pure" version of the domineering class. The Purity of the entire assortment of the detected clusters is computed as a weighted aggregation of the respective clusters purities, defined as follows:

$$Purity\ (C) = \sum_{r=1}^{k} \frac{|C_r|}{n} Purity\ (C_r) =$$
$$\frac{1}{n} \sum_{r=1}^{k} \max_{s=1,\ldots,l} |C_r \cap D_s| \quad (6)$$

In the best condition, Purity is equal to 1. Greater values for this measure imply that the equivalent clustering result is more accurate; terrible clusterings possess Purity values near 0. However, Purity is not totally dependable for the accuracy evaluation of a clustering method, as a deficient result cannot absolutely lead to the inefficacy of the related method.

---

[17] The distinction between the internal and the external clustering validity metrics is that the internal indices assess the inherent unification/separation of the output clusters, while the external indices appraise the similarity/dissimilarity between the clustering solution out of a peculiar clustering function and the expected results or the same "ground truth" of the corresponding dataset.

[18] SDCOR and X-means both provide anomaly scores for all the normal and abnormal elements in the input data, and even anomalies are finally assigned to a discovered cluster. Therefore, for distinctly defining the resultant outliers cluster, we utilize the existing ground truth on the actual number of outliers (o) in the correspondent dataset; scores are sorted in descending order, and then the top-o indices will be separated as the anomaly cluster.

4.1.2.2. Mirkin metric.
The Mirkin metric is computed as follows:

$$M(C, D) = \frac{1}{n^2}\left(\sum_{r=1}^{k}|C_r|^2 + \sum_{s=1}^{l}|D_s|^2 \right.$$
$$\left. -2\sum_{r=1}^{k}\sum_{s=1}^{l}|C_r \cap D_s|^2\right) \tag{7}$$

Concerning this formula, the Mirkin metric is limited to the range of [0, 1], and in the best case where the clustering solution and the ground truth are identical, it is equal to 0.

4.1.2.3. F-measure.
F-measure is a widely-used measure in the clustering accuracy appraisal tasks, which is intrinsically dependent on the information-retrieval concepts. For comparing two partitions of clusters and classes through this metric, an easy way would be calculating the Precision (P), Recall (R), F-value (F), and ultimately the consequent F-measure for every cluster w.r.t. the regular classes, and at last, we just need to compute the total F-measure for the entire clusters collection. The formulation of this metric is as follows:

$$P(C_r, D_s) = \frac{|C_r \cap D_s|}{|C_r|}, \quad 1 \le r \le k, 1 \le s \le l \tag{8}$$

$$R(C_r, D_s) = \frac{|C_r \cap D_s|}{|D_s|}, \quad 1 \le r \le k, 1 \le s \le l \tag{9}$$

Concerning the already-mentioned formulas, it can be understood that Precision and Recall describe, in order, how homogeneous and complete the cluster C_r is with regard to the class D_s. After computing the Precision and Recall values for a specific cluster with reference to various classes, the correspondent F-values will be the harmonic mean of the respective Precisions and Recalls, in the following terms:

$$F(C_r, D_s) = \left(\frac{\frac{1}{P(C_r, D_s)} + \frac{1}{R(C_r, D_s)}}{2}\right)^{-1} =$$
$$\frac{2P(C_r, D_s)R(C_r, D_s)}{P(C_r, D_s) + R(C_r, D_s)}, \quad 1 \le r \le k, 1 \le s \le l \tag{10}$$

The succeeding F-measure[19] quantity for the cluster C_r is the maximal F-value attained at the whole category of classes $D = (D_1, \ldots, D_l)$. That is:

$$F_1(C_r) = \max_{s=1,\ldots,l} F(C_r, D_s), \quad 1 \le r \le k \tag{11}$$

Finally, the F-measure value for the entire clusters collection is calculated as the weighted sum of the distinctive clusters F-measures, denoted in this way:

$$F_1(C) = \sum_{r=1}^{k}\frac{|C_r|}{n}F_1(C_r) \tag{12}$$

F-measure is always positive, and at the worst condition, it falls into 0; it happens at 1 as its best case, and high values for it connote better clusterings. This measure has an essential superiority over the Purity and the Entropy criteria, as it takes into account both the homogeneity (conformity) and the completeness (integrity) of the clustering solution.

4.1.2.4. Entropy.
The Entropy metric is an information-theoretic benchmark, and particularly, it is more reliable than the Purity metric. The Entropy of the cluster C_r is defined as follows:

$$E(C_r) = -\frac{1}{\log(l)}\sum_{s=1}^{l}\frac{|C_r \cap D_s|}{|C_r|}\log\left(\frac{|C_r \cap D_s|}{|C_r|}\right),$$
$$1 \le r \le k \tag{13}$$

---

19 Here as we employ the traditional form of F-measure which is the harmonic mean of Precision and Recall, we denote it as F_1.

Table 2
Properties of the datasets used in the efficacy experiments.

	Dataset	#n	#p	#o	%o
Real datasets	Mammography	11,183	6	260	2.32
	Adult	38,323	6	1,168	3.05
	Shuttle	49,097	9	3,511	7.15
	Smtp	95,156	3	30	0.03
	Skin	199,283	3	5,085	2.55
	CreditCardFraud	284,807	29	492	0.17
	ForestCover	286,048	10	2,747	0.96
	Http	567,498	3	2,211	0.39
	Hepc	2,003,171	7	5,123	0.26
Synth. datasets	Data1	500,000	30	5,000	1.00
	Data2	1,000,000	40	10,000	1.00
	Data3	1,500,000	50	15,000	1.00
	Data4	2,000,000	60	20,000	1.00

Note that for conditions that x equals 0 in $x \log(x)$, we consider the result as 0. Furthermore, as it was remarked, Entropy is a more all-embracing measure than Purity because it considers the distribution of the entire classes in a certain cluster. The precedent term in Eq. (13), $-\frac{1}{\log(l)}$, is a normalization term to restrict the Entropy values to the interval [0, 1]; however, the Entropy of a random cluster is contrary to its Purity, as a Purity of 1 indicates that the cluster is identical to the respective class, but for Entropy, it is reversed. Finally, the global Entropy of the corresponding clustering product is computed as below:

$$E(C) = \sum_{r=1}^{k}\frac{|C_r|}{n}E(C_r) \tag{14}$$

The global Entropy quantities also fall in the range of [0, 1]. As it is obvious, a flawless clustering is the one in which every single cluster incorporates elements from only a unique class; the global Entropy in such an ideal condition is equivalent to 0. Generally, the lower the value of Entropy, the more unerring the clustering outcome is.

4.1.2.5. Variation of Information (VI).
This measure is another information-theoretic-based tool like Entropy to evaluate clustering solutions. VI appraises the proportion of information that one gains and loses while transferring from one partition to another. VI is delineated in the literature in various forms, though we use the following formula:

$$VI(C, D) =$$
$$\frac{1}{n\log(n)}\sum_{r=1}^{k}\sum_{s=1}^{l}|C_r \cap D_s|\log\left(\frac{|C_r||D_s|}{|C_r \cap D_s|^2}\right) \tag{15}$$

The maximum value of VI relies on n, although in the optimum state, it comes to 0.

4.2. Datasets description

Our evaluation employs various real-world and artificial data in diverse situations. These data collections range in cardinality from about 11,000 to more than 2,000,000 points and in dimensionality from 3 to 60 features. In the following, all of the datasets utilized in our analysis are explained in detail.

4.2.1. Real-life datasets
Some public and large-scale real benchmark datasets, mostly taken from the UCI repository [114], and some others from the OpenML [115] and Kaggle [116] libraries are used in our experiments; they are representatives of different domains in science, society, and humanities. The upper part of Table 2 gives a summary of the properties of these real-world data series.

This table shows the characteristics of all the important test datasets, namely the numbers of objects (#n), attributes (#p), and outliers (#o); in addition, for outliers in each dataset, their share of total objects in the corresponding dataset is reported in percentage terms (%o). Among the noted nine real datasets, except for *Mammography* and *CreditCardFraud* which can be attained through OpenML and Kaggle respectively, the other seven can be directly reached through UCI.[20] Moreover, *Mammography*, *Shuttle*, *Smtp*, *ForestCover*, and *Http* are preprocessed and labeled by ODDS library [117]; the other 4 have been curated under our reflections which will be clarified in subsequent.

The *Adult* dataset is extracted out of the 1994 Census database, and the prediction problem is to define whether a person earns over $50 thousand a year. This dataset contains 14 attributes in total which we consider the six continuous ones here to represent the data; the >$50 K class is down-sampled to 10% to exemplify the outlier class.

The "Skin Segmentation Data Set", which herein is referred to as the *Skin* dataset, is collected out of various face images of diverse age and race groups, and different genders. Two classes are available, namely skin and non-skin, in which the skin class is down-sampled to 10% to typify the anomaly class.

The "Credit Card Fraud Detection" data collection[21] which here is denoted as the *CreditCardFraud* dataset, comprises transactions made in two days under credit cards in September 2013 by European cardholders. The dataset is exceedingly unbalanced, and the anomaly class (frauds) stands for 0.17% of all transactions; furthermore, we have discarded the first attribute, which represents the time of the transaction occurrence w.r.t. the first transaction in the dataset.

The "Individual household electric power consumption Data Set" which herein appears as the *Hepc* dataset, holds mensuration of some electrical equipment in a house beside the date and time of the measurement. The dataset consists of 9 features, including the two temporal ones, which we omit here; about 1.25% of the samples suffer from missing values which are discarded consequently. This dataset does not possess any labeling; hence we follow a subtle heuristic as employed in [104] to label data objects as inliers/outliers. For an arbitrary dataset with p dimensions, the Mahalanobis distances of all points in each cluster w.r.t. the corresponding sample mean and sample covariance matrix, follow a χ^2 distribution with p degrees of freedom. The points exhibiting a distance more than the theoretical 0.975 quantile are indicated as candidate outliers; then, the nominated outliers pack is down-sampled to 10% to represent the anomaly class.

4.2.2. Synthetic datasets

Various experiments have been conducted on variant simulated datasets in our study. All of these artificial data series have been created under an ideal setting and follow the strong assumptions of our algorithm, including the one that the structure of existing clusters should be Gaussian due to the use of the Mahalanobis distance measure, and also, the meaningful proportion between the dimensionality and cardinality of every dataset to evade the singularity problem throughout the clustering course of action. Moreover, the generated outliers are typically more outstanding than those in the real data, and the outliers "truth" can be utilized to evaluate whether an outlier algorithm is eligible to locate them.

In more detail, for each dataset having p dimensions, we build Gaussian clusters with arbitrary mean vectors so that they are quite far away from each other to hinder possible overlappings among the multidimensional clusters. As for the covariance matrix, first, we create a matrix $[\mathbb{A}]_{p \times p}$, whose elements are uniformly distributed in the interval $[0,1]$. Then, we randomly select half the elements in $\mathbb{A}$ and make them negative. Finally, the corresponding covariance matrix is obtained in the form of $\left[\sum\right]_{p \times p} = \mathbb{A}^T \mathbb{A}$. Now, w.r.t. the mean vector (location) of each cluster and its covariance matrix (shape of the cluster), we can generate an arbitrary number of data points from a p-variate Gaussian distribution. Moreover, to eliminate marginal noisy objects in each cluster, we can exploit the Mahalanobis distance criterion and eliminate those objects outside the Mahalanobis radius, set to, e.g., 1.

For injecting local outliers to each cluster, first, we consider a hypercube covering the boundaries of the corresponding cluster in every dimension, with the same centroid as of the cluster and having a specific amount of vacant space around it. Then, we randomly generate records in this space and accept them as local outliers if and only if they fall in the accepted Mahalanobis distance interval of the cluster, e.g., $\left[4\sqrt{p}, 6\sqrt{p}\right]$. Since the hypercube volume increases so rapidly by p for high dimensions, we need to extend the accepted interval for generated points to save time. Therefore, some of the synthesized outliers in this way could be global. However, as mentioned earlier, the concept of local outliers covers global ones; hence, each global outlier could be presumed as a local outlier too, but not vice versa.

4.3. Algorithms implementations and parameters

Except for ORCA, DOLPHIN, and X-means which are implemented by the genuine authors in C/C++, we have implemented all of the other five competing methods, including the proposed method, in MATLAB R2016b (version 9.1).[22] In what follows, for every method in our analysis, some essential matters on the parameter setting are presented.

4.3.1. ORCA

In ORCA, the parameter k denotes the number of nearest neighbors, which by using higher values for that, the execution time will also increase. Here, the suggested value for k, equal to 5, is utilized in all of our experiments. The parameter N specifies the maximum number of anomalies to be reported. If N is set to a small value, then ORCA increases the running cut-off quickly, and therefore, more searches will be pruned off, which will result in a much faster runtime. Hence, as the correct number of anomalies is not assumed to be foreknown in the algorithm, we set $N = \frac{n}{8}$, as a reasonable value, where n stands for the cardinality of the input data; in the case of the efficiency test where the outliers proportion is foreknown as much higher than usual, we set $N = \frac{n}{2}$.

However, ORCA does not report an anomaly score for the rest of the data, and this is as long as for computing the AUC values, there is an indispensable need to report an anomaly score for every instance; thus, we set a score equal to zero for other non-anomaly reported objects.

4.3.2. DOLPHIN

For executing the DOLPHIN method on any query data, we need two distinct input parameters specialized for every dataset and two other general parameters that could be established globally in all experiments. The two specific parameters are, namely, R, the neighborhood radius, and k, the minimum neighborhood cardinality required for an object to be identified as an inlier. The other two general parameters are viz $p_{inliers}$, the fraction of granted inliers to be maintained in the indexing structure, and h, the number of histogram bins used to approximate the nearest neighbors distribution for every point; these parameters are in connection to the pruning rules employed by the method and are determined equal to 0.05 and 16, respectively.

To define k under each dataset, we have set it to 1% of the dataset size. However, for R, we followed the *DolphinParamEstim* procedure stipulated in the original paper. Concerning this procedure, the parameter R directly correlates with the expected ratio of outliers, *alpha*, which is anticipated to be detected by DOLPHIN.

It should be noted that DOLPHIN is the only method in our evaluations that does not provide any anomaly scores for the data elements, and its output is all and solely the definite list of potential outliers. In such a case, the ROC and PR curves will not be appealingly smooth, and the following AUC values will not be very reliable either,[23] For this reason, we decided to run the method by various R values, which are in accordance with different *alpha* values[24] and rank the detected outliers in the entire iterations w.r.t. the sum of their appearance times in diverse iterations. This heuristic strategy would lead to some sort of outlier ranking, in which every potential outlier gains a positive integer score with a direct relationship to its anomalousness degree, while non-outlier objects attain a score of zero.

4.3.3. S_p

S_p is the simplest distance-based method in our analysis and only requires one single parameter, which even with its default value proposed by the authors, promising outcomes could be achieved over multiple datasets. For this reason, we follow the same procedure as suggested in the original paper and set the sample size, s, equal to 20 in our experiments.

4.3.4. LOF and LoOP

These two traditional state-of-the-art density-based methods follow a unique structure to build the premise of their calculations. The first thing they need to compute the outlier scores is the materialization matrix, which contains the kNN category for every data instance and the associated distances.[25]

For LOF, there are two input parameters determining the lower and upper proximity bounds in kNN evaluation. As stated by the authors, unfortunately, the LOF values attained over variant k values do not demonstrate a fixed and predictable behavior; i.e., LOF neither rises nor falls monotonically under the neighborhood

parameter. Therefore, in all experiments, we follow the implied manner in the original article and set these two parameters as $MinPtsLB = 10$ and $MinPtsUB = 50$.

For LoOP, as it is inspired by LOF, there is solely one proximity parameter which we consider here equal to 30, as the average value of the LOF related lower and upper vicinity limits. There is another input parameter, λ, which is in accordance with the empirical 68-95-99.7 rule and specifies the strictness level on defining anomalies; we set it equal to 3, as suggested.

4.3.5. EnLOF

EnLOF is an ensemble derivative of LOF, which is mainly influenced by the nearest neighbor of every object among the sampled instances. This method, like iNNE, is essentially inspired by the iForest method and thus enjoys an adequate number of subsamples (t) with a specific size (ψ) to determine the anomaly scores for every object. Here, we follow the same premise as iForest and set the two parameters as suggested, i.e., $t = 100$ and $\psi = 256$.

4.3.6. X-means

For X-means, the minimum and the maximum number of clusters are set to 1 and 15, respectively; the number of times to split a cluster and the maximum number of iterations are set to 6 and 50, respectively, as well. The other interesting parameters for building the KD-tree data structure are *max_leaf_size* and *min_box_width* that are set as suggested by the authors; for the datasets with a cardinality less than 100 thousand points, values equal to 40 and 0.03 are employed, respectively, and for those with greater size, in order, 80 and 0.1 are utilized.

4.3.7. SDCOR

SDCOR enjoys a small number of input parameters that could be either set as recommended or meaningfully acquired through simple operations. For the DBSCAN parameters, in Appendix A, it is explained well in detail how to practically obtain the optimum values for the *Eps* and *MinPts* parameters through two different approaches; one, heuristic, and the other, evolutionary.

More importantly, as long as we are trying not to let outliers take part in the process of forming and updating miniclusters during scalable clustering, two of the input parameters of the proposed algorithm are more critical, listed in the following. The random sampling rate, η, which influences the parameter δ, the boundary on the volume of miniclusters at the time of creation; the membership threshold, α, which is useful to restrain, again, the volume of miniclusters, but while they are progressively growing over time.

Note that the sampling rate should not be set too low, as by which the singularity problem might happen during the "Sampling" phase, or even some original clusters may not take the initial density-based form, for the lack of enough data points. Hence, one can state that there is a linear relationship between the random sampling rate and the actual number of original clusters in data. In other words, for datasets with a high frequency and variety of clusters, we are forced to take higher ratios of random sampling to avoid both problems of singularity and misclustering in the "Sampling" stage. Here, regarding our preknowledge about the real and synthetic data, we have set η to 0.5%, unless specified otherwise; e.g., the *Adult* dataset was an exception as its inherent manifold forced us to exercise a larger ratio for random sampling equal to 3%, because, for the lower rates, the singularity issue was bothering us while processing various individual runs.[26]

---

[23] In fact, in more convenient computational conditions, DOLPHIN mostly leads to non-promising detection results. On the other hand, in more compelling parameter settings, i.e., lower values for R and greater values for k, in spite of higher data-processing costs, the DOLPHIN algorithm outputs incline to be quite deterministic and auspicious in all cases; thus, the subsequent detection accuracy outcomes will be more dependable.

[24] In all assessments, *alpha* takes ten different values; 1 to 10 with the step length of 1 for the effectiveness experimentation, and 5 to 50 with the step length of 5 in the case of the efficiency test where the outliers ratio in the input data is much higher than usual.

[25] It better be noted that while working with LOF and LoOP, for the sake of saving time and space, we decided to omit ties among the nearest neighbors; ties or the same overlapping neighbors, are those with a distance from the query point equal to the kth smallest distance. Only one from such category will be chosen as the kth nearest neighbor. Usually, this does not hurt the final detection results too much.

[26] The random sampling rate could be much greater and equal to, e.g., the chunk size. Nonetheless, in the present report, to substantiate the excellent and stable performance of our proposed method, we decided to set it to minimal values.

The same problem concerns α, as a too low value could bring to the problem that the number of subclusters created over scalable clustering will become too large, leading to a much higher computational load. In addition, a too high value for α brings the risk of outliers getting joined to normal subclusters, and it gets worse when this misclassification escalates over time, subsequently increasing the "False Negative" rate. Herein, for all experiments, α is set to 2.[27] Moreover, the pruning threshold, β, could be determined the same as α, and as observed over our analysis, modifying it in the range of meaningful values does not significantly affect the final output quality.

Finally, for Λ, the PC share of total energy (variance), we decided to exert the full capacity in the case of real datasets, as none of them are, in essence, high-dimensional.[28] However, for the synthetic datasets, in the case of more than 30 dimensions, lower quantities like 90% or 80% were employed.

5. Experimental results

5.1. Accuracy and stability analysis

Here, the evaluation results of the experiments conducted on various real-life and synthetic datasets, with a diversity of size and dimensionality, are presented in order to demonstrate the accuracy and stability of the proposed approach.

For SDCOR, every dataset is divided into ten chunks to simulate the algorithm capability for scalable processing. Furthermore, as we know the true structural characteristics of all the real and synthetic data, we compute the accurate anomaly score, based on the Mahalanobis distance criterion, for each object and report the following optimal AUC next to the attained result by SDCOR. Moreover, for the algorithms that have random elements, including S_p, EnLOF, and the proposed method, their detection results are reported in the format of $\mu \pm \sigma$, over 40 independent runs, as μ and σ stand for the average and the standard deviation of the correspondent AUC values.

The AUROC and AUPRC results, and the subsequent runtimes of different methods are summarized in Tables 3, 4, and 5, respectively. Every **bold-faced** AUC and runtime denote the best method for a specific dataset.

5.1.1. Real datasets

As for the real datasets, Tables 3 and 4 show that in terms of both AUROC and AUPRC, SDCOR is more accurate than all the other competing methods. Further elaborated, in terms of AUROC, SDCOR outperforms ORCA, DOLPHIN, S_p, and X-means in seven out of the nine real datasets, and compares favorably with them in the other two viz *Adult* and *Smtp*; for LOF, LoOP, and EnLOF, SDCOR excels them in eight out of the nine real-life data, and only in the case of the *Smtp* dataset, it falls behind these three density-based methods, though not with a remarkable difference. In terms of AUPRC, SDCOR surpasses all the other techniques in all datasets, except for ORCA in *Mammography*, *Adult*, and *Smtp*; for DOLPHIN in *Mammography* and *Adult*; for S_p in *Adult*, *Smtp*, and *Skin*; for EnLOF in only *ForestCover*; for X-means in *Adult* and *Smtp*. However, the superior methods over SDCOR concerning AUPRC have either a quite equivalence or an insignificant difference with it in all the superiority cases.

Moreover, it is obvious that in the context of AUROC, the attained results by the proposed approach are almost the same as the optimal ones. However, in the case of AUPRC, the deviation from the optimum case is not trivial in most cases,[29] but it is satisfactory as SDCOR generally stays at the top w.r.t. the other contending methods. Additionally, the *CreditCardFraud* dataset is an exception concerning the optimal AUC levels, in which both the AUROC and AUPRC outcomes outstrip the corresponding ideal conditions; for AUROC, the distinction is superficial, though, for AUPRC, it is considerable. Such an unusual condition could be caused by some adopted suboptimal clustering parameters; moreover, sometimes, it could come off because of the closeness of some outliers to the original normal clusters in certain real situations, along with the slight variations in the acquired ultimate Mahalanobis contour lines on account of the minor randomness in the functionality of the proposed method.

In addition, the average lines indicate that SDCOR performs overall much better than all the other methods. More importantly, SDCOR is effective on the largest datasets, *Http* and *Hepc*, with rather perfect results. In addition, by considering the standard deviations of AUC values for S_p, EnLOF, and SDCOR, it is evident that SDCOR is much more stable than the other two random-based outlier identification methods,[30] except for EnLOF in terms of AUPRC, which suffers somewhat from a lower average variance than SDCOR. For S_p, it should be pointed out that the outstanding values of standard deviation for it are due to using only one tiny sample in every execution, which causes the algorithm to go through large variations on the final accuracy.

Besides, Table 5 reveals that SDCOR performs much better than the other competing methods in terms of execution time, except for S_p that accomplishes the tasks slightly faster than the proposed algorithm. Although S_p is the fastest among the compared algorithms, as it was noted, its AUC results suffer from large variations, and overall is lower than SDCOR. Moreover, w.r.t. the two conventional and one novel density-based methods, it is evident that there is a huge difference in the consuming time between SDCOR and these counterparts from the density realm; it is due to the fact that differently from LOF and LoOP, in SDCOR, it is not required to compute the pairwise distances of the total objects in a dataset, and in contrast to EnLOF, SDCOR is not established on some ensemble calculations which despite being efficient in certain aspects, could incur high computational expenses.

Finally, it is worthwhile to mention some valuable comments here. As stated earlier at the beginning of this paper, the strong assumption of SDCOR is on the structure of existing clusters in the input dataset, which should have Gaussian distribution. In practice, though, w.r.t. [119], with appreciation to the Central

---

[27] However, with the expense of higher arithmetical stress, one can be more conservative and set α to a lower value, e.g., 1, which is reasonable too.

[28] Besides, the distribution of variance among the real attributes may not be fair enough, which could cause some unexpected results if fewer PCs are taken into account; furthermore, it is not easy to notice as it requires intensive computing, especially in the case of large-scale data. Therefore, it would be more prudent to avoid opting for lower shares of the total variance while working with not very high dimensions.

[29] Despite this inescapable deviation, it is crystal clear that even the optimum values for AUPRC are in general far away from the perfect state, and this is quite natural while analyzing anomaly identification approaches. Further detailed, anomalies are typically in the minority, thus making the correspondent dataset highly-skewed in the class distribution; i.e., the number of negative elements (inliers) substantially outmatches the number of positive elements (outliers). On the other hand, w.r.t. the ROC and PR formulations, the impact of the number of false positives is inferior in the ROC analysis, though gains great superiority while estimating PR; such distinction mainly originates in the incorporated Precision criterion by PR. As a result, the referred data imbalance could incur, as a rule, considerably lower values for AUPRC than the related AUROC [118].

[30] In the context of AUROC which is more prevalent than AUPRC for analyzing anomaly detection approaches, the *Mammography* and *Adult* datasets come down with further variances, and this is as a consequence of the adopted low sampling rates for them. These two data series have the smallest cardinality among all, and thus the utilized sampling ratios are not effective in the best way. Furthermore, as stipulated earlier, we do this to manifest the capability of the proposed method to operate fairly well even with shallow levels of random sampling; however, for higher sampling ratios, more consistent results were observed.

Table 3

AUROC results for SDCOR and its competitors on the real and synthetic datasets.

	Dataset	ORCA	DOLPHIN	S_p		LOF	LoOP	EnLOF	X-means	SDCOR (Optimal)
Real datasets	Mammography	0.703	0.723	0.818 ± 0.040		0.772	0.741	0.775 ± 0.018	0.803	$\textbf{0.820} \pm 0.037\ (0.879)$
	Adult	0.625	0.622	$\textbf{0.644} \pm 0.029$		0.557	0.566	0.509 ± 0.016	0.642	$0.590 \pm 0.036\ (0.665)$
	Shuttle	0.606	0.721	0.877 ± 0.095		0.614	0.568	0.703 ± 0.053	0.780	$\textbf{0.969} \pm 0.009\ (0.994)$
	Smtp	0.860	0.847	0.864 ± 0.025		0.881	$\textbf{0.939}$	0.835 ± 0.031	0.884	$0.779 \pm 0.008\ (0.815)$
	Skin	0.692	0.659	0.881 ± 0.065		0.798	0.060	0.136 ± 0.083	0.862	$\textbf{0.882} \pm 0.009\ (0.889)$
	CreditCardFraud	0.855	0.809	0.822 ± 0.025		0.731	0.695	0.940 ± 0.006	0.824	$\textbf{0.960} \pm 0.001\ (0.958)$
	ForestCover	0.743	0.471	0.568 ± 0.111		0.598	0.557	0.912 ± 0.017	0.779	$\textbf{0.935} \pm 0.006\ (0.950)$
	Http	0.459	0.989	0.924 ± 0.255		0.066	0.259	0.771 ± 0.028	0.055	$\textbf{0.999} \pm 0.001\ (0.999)$
	Hepc	0.975	0.961	0.964 ± 0.011		0.830	0.816	0.805 ± 0.033	0.971	$\textbf{0.989} \pm 0.003\ (0.997)$
	Real AVG	0.724	0.756	0.818 ± 0.073		0.650	0.578	0.710 ± 0.032	0.733	$\textbf{0.880} \pm 0.012\ (0.905)$
Synth. datasets	Data1	1.000	1.000	1.000 ± 0.000		0.992	0.984	1.000 ± 0.000	1.000	$1.000 \pm 0.000\ (1.000)$
	Data2	1.000	1.000	1.000 ± 0.000		0.997	0.993	1.000 ± 0.000	1.000	$1.000 \pm 0.000\ (1.000)$
	Data3	1.000	0.996	1.000 ± 0.000		1.000	0.999	1.000 ± 0.000	1.000	$1.000 \pm 0.000\ (1.000)$
	Data4	1.000	1.000	1.000 ± 0.000		1.000	1.000	1.000 ± 0.000	1.000	$1.000 \pm 0.000\ (1.000)$
	Synth. AVG	1.000	0.999	1.000 ± 0.000		0.997	0.994	1.000 ± 0.000	1.000	$1.000 \pm 0.000\ (1.000)$

Table 4

AUPRC results for SDCOR and its competitors on the real and synthetic datasets.

	Dataset	ORCA	DOLPHIN	S_p		LOF	LoOP	EnLOF	X-means	SDCOR (Optimal)
Real datasets	Mammography	0.149	$\textbf{0.151}$	0.148 ± 0.056		0.136	0.093	0.095 ± 0.013	0.123	$0.148 \pm 0.067\ (0.277)$
	Adult	0.099	0.083	$\textbf{0.110} \pm 0.018$		0.038	0.037	0.033 ± 0.002	0.109	$0.073 \pm 0.020\ (0.230)$
	Shuttle	0.178	0.314	0.608 ± 0.182		0.132	0.115	0.251 ± 0.035	0.328	$\textbf{0.617} \pm 0.049\ (0.942)$
	Smtp	0.353	0.001	$\textbf{0.496} \pm 0.076$		0.054	0.013	0.001 ± 0.001	0.293	$0.245 \pm 0.083\ (0.252)$
	Skin	0.073	0.063	$\textbf{0.115} \pm 0.043$		0.060	0.037	0.014 ± 0.001	0.083	$0.096 \pm 0.007\ (0.100)$
	CreditCardFraud	0.044	0.010	0.006 ± 0.001		0.016	0.033	0.035 ± 0.018	0.007	$\textbf{0.708} \pm 0.003\ (0.489)$
	ForestCover	0.067	0.007	0.011 ± 0.003		0.019	0.015	$\textbf{0.090} \pm 0.027$	0.020	$0.089 \pm 0.009\ (0.109)$
	Http	0.006	0.100	0.231 ± 0.084		0.002	0.002	0.008 ± 0.001	0.003	$\textbf{0.463} \pm 0.063\ (0.500)$
	Hepc	0.249	0.251	0.370 ± 0.081		0.016	0.018	0.032 ± 0.018	0.322	$\textbf{0.426} \pm 0.070\ (0.658)$
	Real AVG	0.136	0.109	0.233 ± 0.060		0.053	0.040	0.062 ± 0.013	0.143	$\textbf{0.318} \pm 0.041\ (0.395)$
Synth. datasets	Data1	1.000	0.426	1.000 ± 0.000		0.939	0.874	1.000 ± 0.000	1.000	$1.000 \pm 0.000\ (1.000)$
	Data2	1.000	0.000	1.000 ± 0.000		0.966	0.888	1.000 ± 0.000	1.000	$1.000 \pm 0.000\ (1.000)$
	Data3	1.000	0.390	1.000 ± 0.000		0.992	0.871	1.000 ± 0.000	1.000	$1.000 \pm 0.000\ (1.000)$
	Data4	1.000	0.000	1.000 ± 0.000		1.000	0.999	1.000 ± 0.000	1.000	$1.000 \pm 0.000\ (1.000)$
	Synth. AVG	1.000	0.204	1.000 ± 0.000		0.974	0.908	1.000 ± 0.000	1.000	$1.000 \pm 0.000\ (1.000)$

Table 5

Execution time (secs) results for SDCOR and its competitors on the real and synthetic datasets.

	Dataset	ORCA	DOLPHIN[a]	S_p	LOF	LoOP	EnLOF	X-means	SDCOR[b]
Real datasets	Mammography	5.909	2.750	$\textbf{0.020}$	0.972	0.423	1.956	1.141	0.701
	Adult	33.860	90.010	$\textbf{0.050}$	6.087	4.507	11.007	6.926	0.855
	Shuttle	80.367	99.670	$\textbf{0.061}$	10.058	7.836	15.277	7.385	0.907
	Smtp	142.526	162.410	$\textbf{0.110}$	6.225	1.927	8.448	38.663	0.895
	Skin	405.904	673.170	$\textbf{0.209}$	14.625	5.165	12.915	8.564	1.307
	CreditCardFraud	5,396.614	1,281.840	$\textbf{0.528}$	539.153	524.688	169.969	374.642	7.811
	ForestCover	3,618.686	3,557.930	$\textbf{0.489}$	22.865	9.577	40.152	22.246	1.643
	Http	5,570.813	5,879.130	$\textbf{0.686}$	55.660	25.305	56.319	793.285	1.848
	Hepc	92,882.000	243,583.580	$\textbf{3.247}$	45,098.631	37,145.707	643.060	7,078.436	17.632
	Real AVG	12,015.187	28,370.054	$\textbf{0.600}$	5,083.808	72.428	106.567	925.698	3.733
Synth. datasets	Data1	18,838.446	55,411.820	$\textbf{0.918}$	5,924.641	5,458.474	324.351	461.955	12.863
	Data2	101,361.263	262,633.230	$\textbf{2.075}$	33,806.998	32,672.413	856.039	1,641.932	41.162
	Data3	224,559.345	647,534.410	$\textbf{3.479}$	83,647.072	82,139.098	1,520.316	3,528.569	89.455
	Data4	389,826.856	1,210,724.637	$\textbf{4.997}$	133,835.315	132,388.373	2,267.934	5,952.274	161.932
	Synth. AVG	183,646.478	544,076.024	$\textbf{2.867}$	64,303.506	63,164.590	1,242.160	2,896.183	76.353

[a] The reported quantity is the highest execution time over various DOLPHIN runs w.r.t. different R values.

[b] The required time for finding the optimal parameters of DBSCAN is not taken into account as a part of the total runtime.

Limit Theorem, a fair amount of the real-world data follows the Gaussian distribution. Furthermore, w.r.t. [120–122], even when the original variables are not Normal (Gaussian), employing the properties of PCs for detecting outliers is possible, and the corresponding results will be reliable. Since, given that the PCs are linear functions of p random variables, a call for the Central Limit Theorem may vindicate the approximate Normality for the PCs, even in the cases that the original variables are not Normal.

Regarding this concern, it is plausible to set up more official statistical tests for outliers based on the PCs, supposing that the PCs have Gaussian distributions. Moreover, the exercise of the Mahalanobis distance criterion for outlier detection is only viable for the convex-shaped clusters. Otherwise, outliers could be assigned to an irregular (density-based) cluster under the masking effect, thus will be misclassified.

5.1.2. Synthetic datasets

The experiments are carried out on four artificial datasets with specific details presented in the lower part of Table 2; each dataset consists of 6 Gaussian clusters, and the outliers take up 1 percent of its volume.

Table 6
Friedman test results for the real datasets, on the AUROC and AUPRC outputs out of the S_p, EnLOF, and SDCOR methods.

Dataset	AUROC			AUPRC		
	S_p	EnLOF	SDCOR	S_p	EnLOF	SDCOR
Mammography	**0.818**	0.775	**0.820**	**0.148**	0.095	**0.148**
Adult	**0.644**	0.509	0.590	**0.110**	0.033	**0.073**
Shuttle	0.877	0.703	**0.969**	**0.608**	0.251	**0.617**
Smtp	**0.864**	**0.835**	0.779	**0.496**	0.001	0.245
Skin	**0.881**	0.136	**0.882**	**0.115**	0.014	**0.096**
CreditCardFraud	0.822	0.940	**0.960**	0.006	0.035	**0.708**
ForestCover	0.568	**0.912**	**0.935**	0.011	**0.090**	**0.089**
Http	0.924	0.771	**0.999**	0.231	0.008	**0.463**
Hepc	0.964	0.805	**0.989**	**0.370**	0.032	**0.426**

The excellent results in Tables 3 and 4 verify that the synthetic datasets are overall too simple for SDCOR and also for all the other competing methods except for DOLPHIN in the case of AUPRC. The chief reason for this almost-common ideal performance is that, as stipulated beforehand, in our dedicated setting for building these synthetic data collections, normal objects are in very dense areas, and outliers reside in very sparse zones far enough away from the normal clusters.

However, despite the absolute functioning of most of the other rivaling techniques, regarding the time consumption results in Table 5, apart from S_p, the other runtimes are remarkably and incompetently higher than SDCOR. Besides, for S_p, which is performing slightly quicker than our proposed method, first, considering all runtime results, its distinction from SDCOR is not significant, and second, its accuracy in the real situations, which is the most challenging contest, is still quite far away from SDCOR.

5.2. Significance analysis

As it was clarified beforehand, through our assessment, some of the compared methods are non-deterministic, namely S_p, EnLOF, and SDCOR; i.e., they present different results over diverse executions, even if the same input parameters are exerted globally. Therefore, in this subsection, we perform a significance analysis to realize whether there are major differences in the erratic results out of these three random-based methods. Here, we adopt the Friedman test [123,124] for such statistical analysis.

The Friedman test is a distribution-free statistical hypothesis examination that arranges the significance of the compared algorithms w.r.t. every dataset independently; as it is free of any input parameters, it is also called a non-parametric randomized black analysis of variance. The null hypothesis of this test is that all of the populations of the experimental results — each population corresponding to the application of one of the analyzed methods in different repetitions — have the same median value. Once the Friedman test rejects the null hypothesis, a post-hoc test is required to make pairwise comparisons and detect the couples of significantly distinctive methods.

A critical value obtained from a chi-squared distribution with two degrees of freedom and a significance level of 5% is used for this test. Furthermore, the *p*-value was adjusted by using the Holm methodology [125], and also, as for the post-hoc test, to verify the differences between each couple of methods, the Nemenyi test [124] was adopted.

The Friedman test results for the real-life data series, over the AUROC and AUPRC measurements, are reported in Table 6.[31] In

particular, in every row, if only one method is reported in **bold**, it is determined as the significantly best method among all w.r.t. the corresponding dataset; if two values are reported in **bold**, it means that there is no significant difference between the related methods, though both of them are performing better than the remaining one. Moreover, for the four synthetic datasets in the efficacy experimentation, all of the acquired results out of the three random-based methods are perfect; thus, there is no place for more evaluation.

It is evident that in terms of AUROC, SDCOR is the only winner in 4, the co-winner along with S_p and EnLOF among respectively 2 and 1, out of the nine real-life datasets; hence, SDCOR is the only winner or is amid the winners in 7 out of the total. However, S_p is the sole winner only in the case of the *Adult* dataset, while EnLOF never stands in the first place without any rivals. In terms of AUPRC, SDCOR conquers others only in two datasets, viz *CreditCardFraud* and *Http*, and in the other 6 instances, it is among the conquerors mostly along with S_p and exclusively with EnLOF just in the case of the *ForestCover* dataset; S_p excels others only in *Smtp*, whereas EnLOF never gains the absolute superiority. Overall, it is apparent that SDCOR is the conqueror and the most precise technique among all non-deterministic methods in our analysis.

5.3. SDCOR vs. X-means

The evaluation results over the real-world data collections for the two clustering-based techniques, SDCOR and X-means, obtained out of the five external clustering validity measures, namely Purity, the Mirkin metric, F-measure, Entropy, and VI, along with the following averages are presented in Table 7. Moreover, for every metric, concerning its ideal condition mentioned beneath the metric name, the best method(s) among two for a particular dataset is/are indicated in **bold-faced** print. Besides, for the four synthetic datasets in the effectiveness evaluations, as both SDCOR and X-means attain perfect results, we omit demonstrating them here.

It is perceptible from Table 7 that in all comparison cases, SDCOR is either the undoubted winner or compares favorably well with X-means. In the case of Purity and Entropy, SDCOR fails to conquer X-means overall, although with a negligible difference. However, for the Mirkin metric, F-measure, and VI, the superiority is achieved by SDCOR.

More detailed, it was noticed that in contrast to Purity and Entropy, which alone consider the conformity of the output, F-measure enjoys an essential advantage over them, as it takes into account both the conformity and integrity of the clustering solution. Now w.r.t. the X-means fundamentals in which there is no noise taken for granted beforehand, and more crucial, it is tried to split the input data to multiple clusters until it reaches a reasonable quantity for the frequency of the clusters, hence an arbitrary original cluster (class), as well as the outliers pack, can be eventually divided into several subclusters; this is the exact point which makes the major deviation between X-means and our proposed approach in which the number of output clusters is the same as the ground truth.

Concerning the conformity and integrity concepts, when a cluster contains objects solely from a unique class but does not wholly encompass it, we will have perfect conformity though imperfect integrity as output. A similar conception is intelligently embedded in the Mirkin metric and VI mathematical formulas. In the Mirkin metric, the term corresponding to the intersection of the related clusters and classes will be shallow in the case of high-conformity/low-integrity, thus producing Mirkin values more distant from 0, the optimal condition; this makes the Mirkin metric quite the same as F-measure in quality. On the other hand,

---

[31] It should be noted that for this statistical appraisal, the same results out of the 40 iterative runs mentioned in Section 5.1 are exerted. Furthermore, the following stipulated average AUC ranks in this subsection are utilized in Table 6 as representatives for the non-deterministic methods on different datasets.

Table 7
Results of the clustering validity measures for SDCOR and X-means on the real datasets.

Dataset	Purity (max = 1)		Mirkin (min = 0)		F-measure (max = 1)		Entropy (min = 0)		VI (min = 0)	
	SDCOR	X-means	SDCOR	X-means	SDCOR	X-means	SDCOR	X-means	SDCOR	X-means
Mammography	0.977	**0.978**	**0.071**	0.784	**0.963**	0.299	0.149	**0.107**	**0.022**	0.223
Adult	**0.970**	**0.970**	**0.105**	0.836	**0.945**	0.207	0.195	**0.169**	**0.026**	0.232
Shuttle	0.954	**0.976**	**0.088**	0.091	**0.954**	0.918	0.221	**0.076**	0.028	**0.027**
Smtp	**1.000**	**1.000**	**0.001**	0.894	**1.000**	0.190	**0.002**	**0.002**	**0.000**	0.207
Skin	0.974	**0.975**	**0.097**	0.831	**0.950**	0.232	0.170	**0.073**	**0.019**	0.188
CreditCardFraud	**0.999**	0.998	**0.001**	0.654	**0.999**	0.473	**0.006**	0.018	**0.001**	0.116
ForestCover	**0.990**	**0.990**	**0.036**	0.906	**0.982**	0.144	0.078	**0.061**	**0.009**	0.212
Http	0.999	**1.000**	**0.002**	0.814	**0.999**	0.305	0.009	**0.001**	**0.001**	0.133
Hepc	**0.997**	**0.997**	**0.005**	0.901	**0.997**	0.173	0.017	**0.014**	**0.002**	0.172
Real AVG	0.984	**0.987**	**0.045**	0.746	**0.977**	0.327	0.094	**0.058**	**0.012**	0.168

in VI, as an enhanced version of Entropy, it is stipulated that the gained/lost portion of information while moving from one partition to another is computed. More precisely, the input to the logarithm term, which in the ideal state equals 1, will tend to greater values in non-paradigmatic conditions. This causes the following measured quantity for VI to become farther away from the optimum case equal to 0. Such quality of VI addresses both the conformity and integrity concepts.

Finally, for the real case of the evaluated data series, as in all instances, there is merely a normal cluster and some outliers around it, X-means, w.r.t. the employed parameter setting, inevitably breaks the normal cluster along with the outliers pack down to various subclusters; however, as stated earlier, as required in our analysis, identified outliers, despite being scattered over various subclusters, are after all distinguished as a single cluster, and this is carried out by sorting the related outlier scores and cutting off the topmost indices. This cluster chopping procedure done by X-means leads us to the high-conformity/low-integrity situation, and the following deficiencies in the final clustering cogency evaluations through the Mirkin metric, F-measure, and VI; for Purity and Entropy though, as the outliers are separated well from the inliers in the final clustering solution, hence the acquired measurements for these two conformity-based metrics are generally satisfactory. However, for the four artificial datasets, each of them containing 6 clusters and some distant anomalies dispersedly in the space, it is lucky that the normal clusters are not broken into smaller pieces, and it has occurred just for the outliers pack due to their much greater dispersion; therefore, regarding the flawless anomaly detection results for these synthetic datasets, the following clustering validness measures are all perfect too.

5.4. Tolerance to a high number of outliers

To analyze the accuracy when the number of outliers is increased, we follow the same procedure used to generate the synthetic datasets, described in detail in Section 4.2.2. More precisely, every dataset is established upon a fixed manifold concerning the normal elements, comprising 4 pruned Gaussian clusters with a total of 20,000 objects, in a predefined dimensionality of 2. For the sake of adding noise, the percentage of injected outliers is increased from 50 to 150 percent of the volume of the normal objects, with a step length of 10 percent; alternatively stated, with a fixed number of 20,000 normal elements in every dataset, the number of outliers is raised from 10,000 to 30,000 with the step length of 2,000. Moreover, as we observe this presumption that these data series hold very high rates of noise, sizeable *MinPts* values for finding the DBSCAN optimum parameterization, along with a random sampling rate equal to 10% in all tests, are exerted.

Fig. 6 reveals that, in terms of both AUROC and AUPRC, SDCOR is noise-tolerant, as by increasing the percentage of outliers,

always perfect AUC results are achieved out of our method. Concerning the other methods, their behavior under both AUROC and AUPRC is somewhat the same. ORCA and DOLPHIN are the closest rivals to the proposed method; S_p and X-means generally perform at a lower level but are relatively good. However, for LOF, LoOP, and EnLOF, they are entirely misclassifying outliers in all complex conditions.[32]

The reason for the favorable functionality of SDCOR in bearing much noise is that the basis for forming miniclusters during scalable clustering is the fulfillment of the DBSCAN requirements, which are obtained out of the "Sampling" stage. Simply put, as all of the normal objects follow a Gaussian distribution and outliers are injected using a continuous uniform distribution, hence the local density of the normal points is much higher than that of the outliers; therefore, the acquired optimal parameters for the density-based clustering are obtained proportional to the dense regions containing only the sampled inliers. For this reason, in every memory process, the probability of a subcluster being formed by the outliers is much lesser than that of the normal objects, and thus, our approach based on DBSCAN obtains very satisfactory results.

Fig. 7 shows the DBSCAN result on the sampled data from the test dataset with 150 percent of injected outliers. Four discovered Gaussian clusters and noises are represented with dots in different colors and empty red circles, respectively. As it is evident, even in this highly noisy situation, outliers cannot satisfy DBSCAN constraints on forming a minicluster.

Here, it is worth remarking that w.r.t. our non-reported evaluations on other efficiency tests, when the numbers of normal objects and attributes are increasing (like what we did in the "percentage of outliers" test), under the strong assumptions of the proposed method, one can still anticipate significant detection results out of SDCOR. However, for some of the methods in the contest, w.r.t. their corresponding high computational load, it would be a tedious task to obtain the accuracy results out of them for the very large datasets employed in our experiments.

5.5. Scalability

To assess the scalability of the proposed method, we measure the time consumption with the increasing number of objects. For this purpose, first, a synthetic dataset containing 4 Gaussian clusters with 200,000 elements in 10 dimensions is generated. Then, we conduct random sampling with the rates of 10 to 100

---

[32] Actually, except for EnLOF which is predefined to work with only the nearest neighbor of every object, for the other two state-of-the-art density-based techniques, such deficient outcomes are quite related to the lack of the optimal parameters. More detailed, it was noted beforehand that in LOF and LoOP, locating the optimum values for the proximity options is not easy and demands a high amount of trial and error w.r.t. the existing outliers ground truth, which is not fair concerning our analytical context.
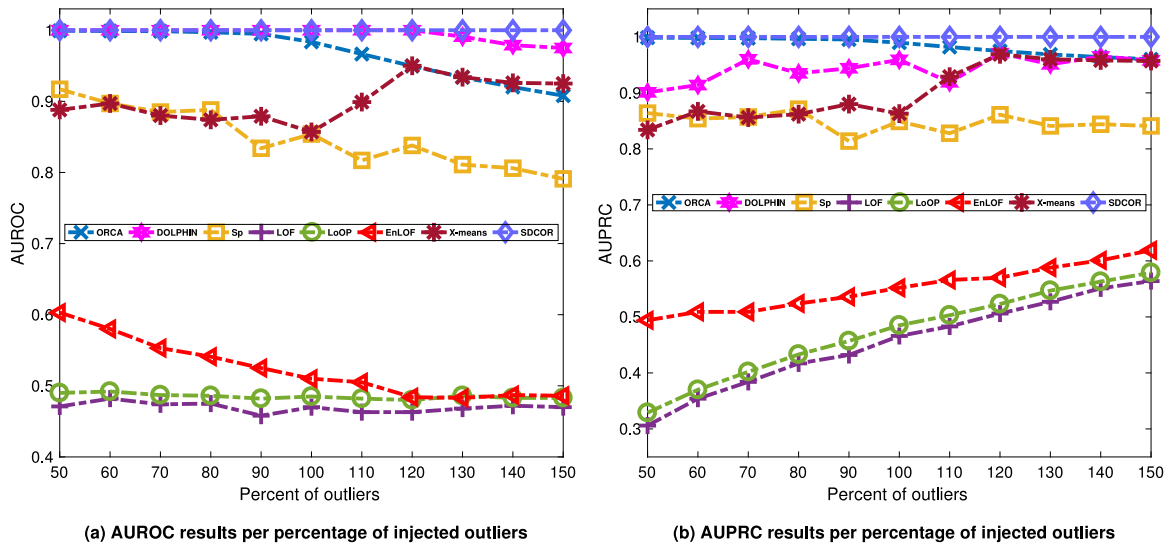
(a) AUROC results per percentage of injected outliers

(b) AUPRC results per percentage of injected outliers

Fig. 6. Robustness results out of the competing methods, over the AUROC and AUPRC measures.
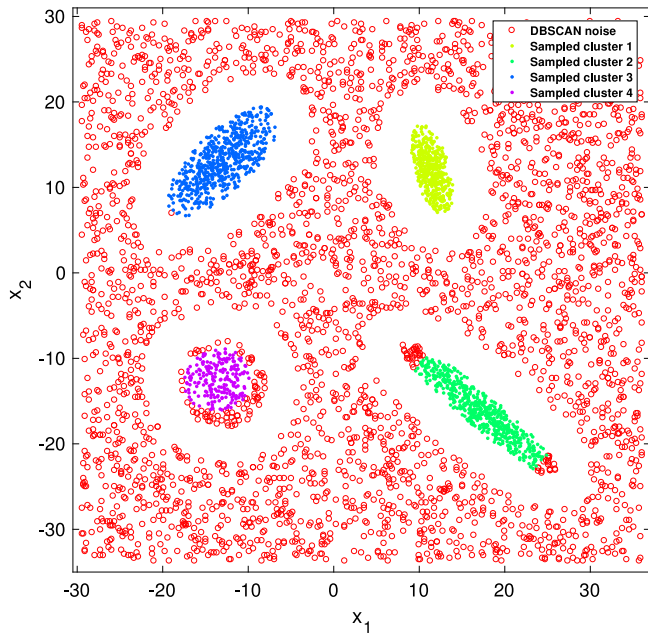
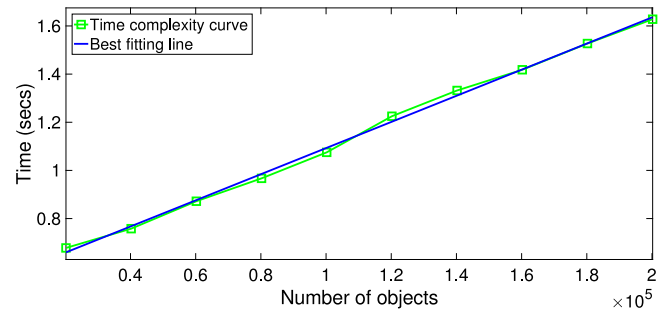**Fig. 7.** DBSCAN result with the optimal parameters on the high-noisy sampled data.

Fig. 8. Scalability test result for SDCOR.

5.6. Effect of sampling rate

Here, we examine the variation of the covariance determinant belonging to the sampled data from a unique cluster per various random sampling rates. Therefore, first of all, we create an arbitrary Gaussian cluster with 10,000 objects and two attributes. Then, we start sampling with the rate of 0.5 percent and proceed to 100 percent with the step length of 0.5 percent; subsequently, for each of these resulting sampled clusters, we calculate the corresponding covariance determinant to be incorporated in our analysis.

As Fig. 9a displays, with the increase in the random sampling rate, the corresponding covariance determinant is approaching that of the original cluster. However, as it is evident, even covariance determinants associated with very shallow sampling rates are quite close to that of the main cluster. For example, for the lowest sampling rate equal to 0.5 percent, the corresponding covariance determinant is approximately 300, which is close enough to that of the original cluster, roughly equal to 220.

Now, if we plot the tolerance ellipses for both the main cluster and the sampled cluster with the sampling rate of 0.5 percent, we observe that they are so similar to each other. Fig. 9b illustrates such a situation, in which the objects belonging to the original cluster and those associated with the sampled one are illustrated with blue dots and red squares, respectively. Moreover, tolerance ellipses of the main and sampled clusters are shown, in order, in red and black.
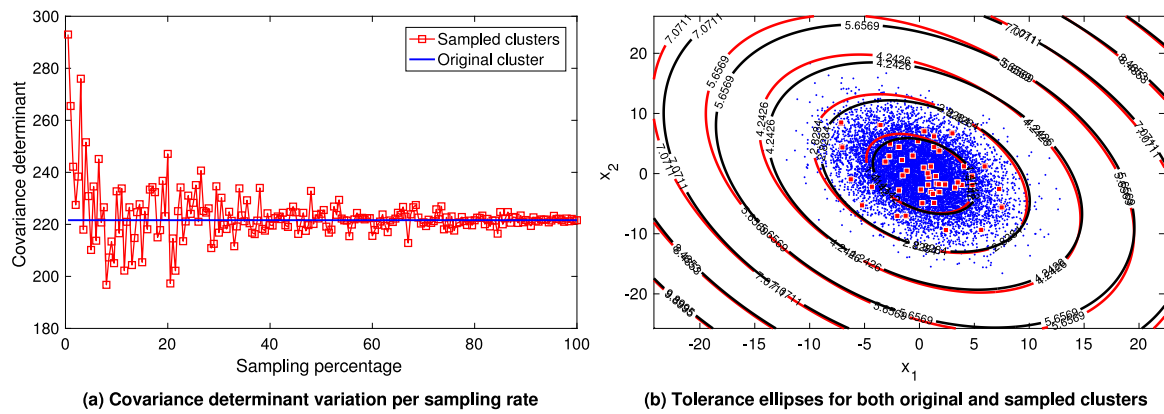
percent with a step length of 10 percent to build ten datasets for the scalability test; for each resulting dataset, we inject 200 outliers into it. In other words, we want to analyze the execution time using datasets having very similar essential characteristics, namely the location (centroid) and the shape (covariance matrix). Moreover, in all experiments, parameters were set as suggested, and the following detection results were all perfect.

Fig. 8 demonstrates that the run time of SDCOR is close to a linear function of the number of objects; this is a confirmation of the result of the analysis conducted in Section 3.4 on the algorithm complexity. In particular, for the dataset with the minimum size equal to 20,200, the run time is 0.678 s. However, when the number of objects reaches the maximum value equal to 200,200 — i.e., about ten times the minimum value — the processing time increases by only 2.4 times to 1.628 s. Accordingly, SDCOR has a low constant in its runtime complexity, and hence, is scalable.

(a) Covariance determinant variation per sampling rate

(b) Tolerance ellipses for both original and sampled clusters

Fig. 9. Covariance determinant and tolerance ellipses variation per random sampling rate.

6. Conclusion

In this paper, a new scalable density-based clustering approach for local outlier detection in massive data is proposed, which processes the input data in chunks. First of all, by obtaining a random sample of the entire dataset and applying a density-based clustering algorithm to it, the initial temporary clustering model is built, which contains the rough information of the original clusters in data. Then, this model is progressively updated by loading successive chunks of data into the memory. Ultimately, after processing the whole chunks, the final clustering model is acquired, which w.r.t. that and conducting another scan of the entire dataset, each object is given an outlying score equal to its local Mahalanobis distance.

A complete evaluation, conducted on both real-world and synthetic datasets, demonstrates the appealing performance of SDCOR in comparison with different state-of-the-art density-based outlier algorithms, which need the data to be resident in the memory; and also, with some other rapid distance-based anomaly detection methods, which can operate well on the disk-resident data. Moreover, the efficiency outcomes confirm the robustness of the proposed method comparing to other methods in very noisy conditions. In addition, the experiments substantiate that the algorithm has a linear time complexity with a low constant and that, even with a meager rate of random sampling, it is still able to satisfactorily approximate the shape of the real clusters. For future work, we would like to enhance our proposed approach to be able to cope also with the density-based non-convex clusters having various distributions.

CRediT authorship contribution statement

Sayyed Ahmad Naghavi Nozad: Project administration, Conceptualization of this study, Investigation, Methodology, Software, Data curation, Formal analysis, Validation, Visualization, Resources, Writing - original draft. **Maryam Amir Haeri:** Initial conceptualization, Data curation, Validation. **Gianluigi Folino:** Supervision, Formal analysis, Validation.

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.

Appendix A. DBSCAN optimal parameters

This Appendix describes the ways employed in this paper to determine the optimal values for the DBSCAN parameters to be applied throughout the "Sampling" and the "Scalable Clustering" phases.

A.1. k-distance graph

The first approach is a heuristic suggested by the original paper of DBSCAN [79, Section 4.2], the sorted k-dist or the same k-distance graph, which is a mapping from the corresponding dataset to the distance of every point to its kth nearest neighbor plotted in descending order. Alternatively stated, in this manner, we fix the *MinPts* parameter, as it is more convenient than the *Eps* parameter to be established, and its main purpose is to polish the density estimation; its variation often has an insignificant influence on the clustering outcomes as they "do not substantially conflict with each other" [79,126]. For most of datasets, one can keep this parameter at the predetermined value of *MinPts* = 4 [79] — which is mostly recommended for two-dimensional data — or *MinPts* = $\lfloor \ln(n) \rfloor$ [127] which is in accordance with the size of the corresponding input data; furthermore, Sander et al. [128] suggest to set it to double the number of dimensions, i.e., *MinPts* = $2p$. Besides, it would be wiser, more considerate, and helpful to set this parameter to much higher values to acquire improved results while working on datasets that are very noisy, or large-scale, or high-dimensional, or suffer from many duplicate values; any of the mentioned special conditions or more than one can be enough reason to employ a large value for *MinPts* [126].

Finally, after determining the *MinPts* value, the best value for *Eps* is in consonance with the maximal k-dist value in the "thinnest" cluster — the cluster with the lowest density. For identifying this threshold, we need to visually locate the first point in the first "valley" — "knee" or "elbow" — in the related sorted k-distance graph. By utilizing this quantity as the distribution

(a) *k*-distance plots for MinPts=18
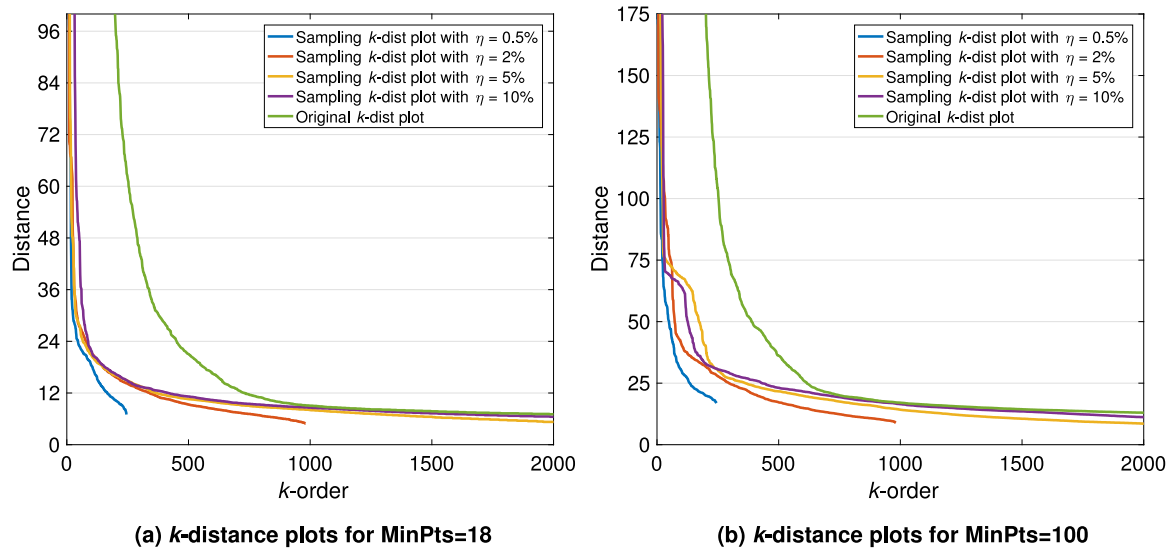
(b) *k*-distance plots for MinPts=100

Fig. 10. Sampling and original sorted *k*-distance plots for the *Shuttle* dataset with $k = 17$ (*MinPts* = $2 \times 9 = 18$) and $k = 99$ (*MinPts* = 100).

threshold, every point with a higher *k*-dist value that stands at the left of the threshold will be considered as noise or a marginal point; on the contrary, every point with a lower *k*-dist value residing at the right of the threshold will be a core point, thus attributed to some cluster.

Besides the challenges behind selecting the optimal parameter values for DBSCAN, it ought to be contemplated that the distribution of the sampled data is different from the original one, and much more critical, DBSCAN shall be applied on both; hence, it will be required to set distinct values for each situation. As *MinPts* is a less sensitive parameter than *Eps*, we decide to utilize the same quantity for it concerning both the sampled and the original distributions. However, about *Eps*, sometimes even slight variations could incur significant ups and downs in the final results. One exact point is that after sampling, the distance between every point and its *k*th nearest neighbor expands, and this is a sufficient cause to say that the optimal *Eps* value for the sampled data is meaningfully larger than the required one for the original distribution. Now, if we plot the sorted *k*-distance plot for both the sampled and original data, it could be observed that the first valley for the sampling *k*-dist plot happens earlier and with a greater value than the corresponding valley for the original plot.

Fig. 10 illustrates the zoomed-in views of the *k*-distance plots concerning several sampling rates for the *Shuttle* dataset; the blue, red, yellow, and violet lines indicate, in order, the sampling *k*-dist plots for the sampling rates equal to 0.5%, 2%, 5%, and 10%, while the green line denotes the correspondent *k*-dist graph for the original distribution. Two distinct quantities for *MinPts* w.r.t. the above heuristics are supposed as one in Fig. 10a, equal to double the *Shuttle* dimensionality ($2 \times 9 = 18$), and the other in Fig. 10b, established as a much higher value (100) to evade the noise[33] — as *Shuttle* contains the most ratio of outliers among real and artificial datasets exerted in our accuracy and stability analysis.

As it is discerned from the illustration, for both *MinPts* specific values, the sampling *k*-dist plots are following almost the same pattern and can be easily distinguished from the relevant original plot; furthermore, in both situations, the first valley for the sampling plots occurs at a higher value than the original one.

In Fig. 10a, for *MinPts* = 18, it seems most proper to define the cut around 24 for the sampled distribution, and about half of this value for the original one[34]; in Fig. 10b, for *MinPts* = 100, the best cuts for the sampling and the primary data appears to be approximately 50 and 25, respectively[35] As a consequence, as it is not possible to attain the original *k*-dist graph in the usual case because of the following computational expenses, we prefer to set the correspondent *Eps* value for the original distribution during scalable clustering to half of the optimal cut for the sampled data.

A.2. PSO evolutionary algorithm

Despite the advantages connected to utilizing the *k*-distance graph for finding the DBSCAN optimum parameters, sometimes, it is not that easy to interactively locate the first valley position in the related sorted *k*-dist graph as it might be very smooth and without any visual steps (artifacts); furthermore, in some cases, setting the most suitable value for *MinPts* requires a high amount of trial and error. Therefore, one can resort to an optimization algorithm like PSO to conquer the mentioned difficulties. PSO looks for the optimal cases in a search space having a dimensionality on a par with the number of demanded parameters by the particular problem; *Eps* and *MinPts* are the required parameters here, and their lower and upper bounds need to be determined for building the search zone employed by the PSO algorithm while being applied to the sampled data. For the original distribution requisite parameters, regarding the impossibility of putting PSO into practice for the original massive data, the same scenario will be followed as in the *k*-distance graph strategy; half of the acquired sampling optimum *Eps* for the primary data during the batch-wise clustering, and the same *MinPts* for both situations.

Inspired by the approximate analogousness of the plots in Fig. 10, and that the optimal quantity for *Eps* lies between the minimum and maximum of any of these plots — which are almost the same in various sampling conditions, and also the original graph — we decide to establish the lower and upper survey bounds for the *Eps* parameter as the minimum and maximum

---

[33] The proximity parameter, *k*, in the *k*NN search does not include the subject point, but the range query search employed in DBSCAN considers the point itself too for the density evaluation; thus, *k*, here, complies with *MinPts* = *k* + 1.

[34] To avoid the singularity problem while operating on diverse iterations of the proposed method, it is more discreet to set the sampling cut a bit greater, as different random samplings could have slightly different distributions, thus demanding varying requisites.

[35] Higher integers for *MinPts*, naturally incur larger values for *Eps* to satisfy the range query search carried out by DBSCAN.

values of the k-dist graph associated with the corresponding sampled data. For the *MinPts* parameter, w.r.t. the already mentioned heuristics, we prefer to set the lower search bound to $\lfloor \ln(n) \rfloor$ − as it is a dynamic number according to every certain dataset − and the subsequent upper bound could be asked through the user as a high-enough integer to avoid anomalies; the most suitable *MinPts* value will be lying in this range. In the following, an explanation of the PSO algorithm and the peculiar evaluating manner for its optimization plan exerted in this study will be provided.

PSO is an evolutionary population-based algorithm oriented from the corporate behavior of some animal gatherings, in particular, flocks and shoals; it is totally founded upon mathematical concepts and can solve intricate arithmetical optimization problems [129,130]. One of the greatest virtues of utilizing PSO among various optimization strategies is its dependence on a fewer number of parameters; besides, the requisite ones are extensively debated in the scientific literature.

Here, we laconically explain the classical or the same inertial version of the algorithm. Like any optimization problem, the target is to figure out a variable vector − also known as the position vector − represented as $X = (x_1, \ldots, x_n)$ which is about to minimize or maximize a specific optimization function denoted as $f(X)$ − also called the fitness, the objective or the cost function. X is an n-dimensional vector which stands for the n distinct unknown variables in the intended problem,[36] and $f(X)$ is a multi-criteria evaluation metric that appraises the reliability of a particular condition specified by the position vector X. If we consider a swarm with P particles, there would be a position vector along with a velocity vector for each particle; every position vector is a capable solution to the objective function and is updated through successive iterations according to the knowledge of the particle itself and other participating ones in the group. We illustrate particle i at iteration t of the solving procedure as $X_i^t = (x_{i1}, \ldots, x_{in})$, and the corresponding velocity vector as $V_i^t = (v_{i1}, \ldots, v_{in})$. Each particle is bounded in a distinct interval, which in total, all of these intervals build the search space required for resolving the optimization function. At each iteration, the velocity and position vectors for particle i at the j-th dimension are updated as follows:

$$V_{ij}^{t+1} = w V_{ij}^t + c_1 r_1^t \left(pbest_{ij} - X_{ij}^t \right)$$
$$+ c_2 r_2^t \left(gbest_j - X_{ij}^t \right) \tag{16}$$
$$X_{ij}^{t+1} = X_{ij}^t + V_{ij}^{t+1} \tag{17}$$

where $i = 1, \ldots, P$ and $j = 1, \ldots, n$.

Regarding Eq. (16), the updating formula of the velocity vector in every dimension, three different terms contribute to a particle motion over any iteration. In the first term, parameter w is the inertia weight constant with a positive value. This parameter handles the balance between the global search (exploration) and the local search (exploitation) of the algorithm. The second and the third terms are, in order, the individual perception term and the collective learning term. The individual perception term takes into account the difference between the particle's own previous best location ($pbest_i$) and its current position (X_i^t); in contrast, the collective learning term addresses the inclination towards the previous best-identified location among all of the involved particles ($gbest$). Parameters c_1 and c_2 are acceleration constants that weigh the personal and social understanding of every particle, respectively. Parameters r_1 and r_2 are two random quantities in the range of [0, 1] which help the optimization problem not to fall early in a local optimum resolution; these random values

are reset at every iteration. Finally, the corresponding position of every particle in each dimension will be updated w.r.t. Eq. (17), and in the case that it is out of the specified searching limits, it will get the equivalent violated bound quantity.

PSO consists of two separate sections, one initialization part, and one iterative component. In the initialization section, firstly, the position and velocity vectors for every particle get primary values, and then after evaluating the alignment of the existing positions through the fitness function, the associated best local and global positions are established. Consequently, in the iterative section, at every repetition, the velocity and position vectors are updated in all of their dimensions according to Eqs. (16) and (17); then, by employing the objective function again, the related best local and global locations are modified until a termination condition is satisfied.

For the PSO cost function in our experiments, we have utilized the aggregation of two internal clustering validity indices [131], namely the Davies–Bouldin index [132] and the CS index [133, 134], along with the ratio of detected outliers [135], which requires to be minimized. To be noted, the CS index incurs a high calculational load, especially on large datasets, yet it was very fruitful for our analysis. By the way, other indices with various arithmetic expenses − including but not limited to the Dunn index [136], the Silhouette index [137], the Banfield–Raftery index [138], and the Maulik–Bandyopadhyay index (Index I) [139] − could be employed too, although it is not guaranteed that better outcomes will undoubtedly be achieved by applying a more computationally expensive index and vice versa.

Moreover, we have defined three special conditions for the cost function as the infinite value to evade them. One is when after applying DBSCAN to the sampled data with the specific parameters out of a particular particle, no cluster is detected, and, i.e., every object is introduced as noise; such parameters are out of order w.r.t. the accepted sampling terms. The other condition is when every sampled point is assigned to some cluster without any noise being identified; this is against our presumption that the input data always contains some outliers that need to be separated. Finally, there is another unacceptable situation that happens when at least one of the discovered sampled clusters through DBSCAN is suffering from the singularity problem; with the optimum parameters in a condition conforming with the proposed method strong assumptions, there should not be any detected sampled cluster with the singularity issue. Besides, while we are applying internal cluster validation indices to a clustering result out of a particle parameters, detected noisy objects are assumed as a unique cluster, too, as they should play an active role in the arrangement evaluation of the DBSCAN outcome [140].

Appendix B. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.knosys.2021.107256.

References

[1] V. Hodge, J. Austin, A survey of outlier detection methodologies, Artif. Intell. Rev. 22 (2) (2004) 85–126.

[2] V. Chandola, A. Banerjee, V. Kumar, Anomaly detection: A survey, ACM Comput. Surv. 41 (3) (2009) 15.

[3] A. Zimek, E. Schubert, H.-P. Kriegel, A survey on unsupervised outlier detection in high-dimensional numerical data, Stat. Anal. Data Min. ASA Data Sci. J. 5 (5) (2012) 363–387.

[4] H. Wang, M.J. Bah, M. Hammad, Progress in outlier detection techniques: A survey, IEEE Access 7 (2019) 107964–108000.

[5] M. Agyemang, K. Barker, R. Alhajj, A comprehensive survey of numeric and symbolic outlier mining techniques, Intell. Data Anal. 10 (6) (2006) 521–538.

[6] J. Han, J. Pei, M. Kamber, Data Mining: Concepts and Techniques, Elsevier, 2011.

---

[36] The related variables for DBSCAN are its two input parameters, *Eps* and *MinPts*, as it was already discussed.

[7] M.M. Breunig, H.-P. Kriegel, R.T. Ng, J. Sander, LOF: identifying density-based local outliers, in: ACM Sigmod Record, ACM, 2000, pp. 93–104.

[8] T. De Vries, S. Chawla, M.E. Houle, Finding local anomalies in very high dimensional space, in: 2010 IEEE International Conference on Data Mining, IEEE, 2010, pp. 128–137.

[9] 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.

[10] 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, ACM, 2009, pp. 1649–1652.

[11] 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.

[12] B. Tang, H. He, A local density-based approach for outlier detection, Neurocomputing 241 (2017) 171–180.

[13] J. Huang, Q. Zhu, L. Yang, J. Feng, A non-parameter outlier detection algorithm based on Natural Neighbor, Knowl.-Based Syst. 92 (2016) 71–77.

[14] A. Wahid, A.C.S. Rao, RKDOS: A relative kernel density-based outlier score, IETE Tech. Rev. (2019) 1–12.

[15] A. Wahid, A.C.S. Rao, ODRA: an outlier detection algorithm based on relevant attribute analysis method, Cluster Comput. (2020) 1–17.

[16] A. Wahid, C.S.R. Annavarapu, NaNOD: A natural neighbour-based outlier detection algorithm, Neural Comput. Appl. (2020) 1–17.

[17] J. Xie, Z. Xiong, Q. Dai, X. Wang, Y. Zhang, A local-gravitation-based method for the detection of outliers and boundary points, Knowl.-Based Syst. 192 (2020) 105331.

[18] S. Wu, S. Wang, Information-theoretic outlier detection for large-scale categorical data, IEEE Trans. Knowl. Data Eng. 25 (3) (2011) 589–602.

[19] X.H. Dang, B. Micenková, I. Assent, R.T. Ng, Local outlier detection with interpretation, in: Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Springer, 2013, pp. 304–320.

[20] Z. He, X. Xu, S. Deng, Discovering cluster-based local outliers, Pattern Recognit. Lett. 24 (9–10) (2003) 1641–1650.

[21] L. Duan, L. Xu, Y. Liu, J. Lee, Cluster-based outlier detection, Ann. Oper. Res. 168 (1) (2009) 151–168.

[22] J.M. Jobe, M. Pokojovy, A cluster-based outlier detection scheme for multivariate data, J. Amer. Statist. Assoc. 110 (512) (2015) 1543–1551.

[23] J. Huang, Q. Zhu, L. Yang, D. Cheng, Q. Wu, A novel outlier cluster detection algorithm without top-n parameter, Knowl.-Based Syst. 121 (2017) 32–40.

[24] H. Moonesighne, P.-N. Tan, Outlier detection using random walks, in: 2006 18th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'06), IEEE, 2006, pp. 532–539.

[25] H. Moonesinghe, P.-N. Tan, Outrank: a graph-based outlier detection framework using random walk, Int. J. Artif. Intell. Tools 17 (01) (2008) 19–36.

[26] C. Wang, H. Gao, Z. Liu, Y. Fu, A new outlier detection model using random walk on local information graph, IEEE Access 6 (2018) 75531–75544.

[27] C. Wang, H. Gao, Z. Liu, Y. Fu, Outlier detection using diverse neighborhood graphs, in: 2018 15th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP), IEEE, 2018, pp. 58–62.

[28] C. Wang, Z. Liu, H. Gao, Y. Fu, VOS: A new outlier detection model using virtual graph, Knowl.-Based Syst. 185 (2019) 104907.

[29] P. Amil Marletti, N. Almeira, C. Masoller Alonso, Outlier mining methods based on graph structure analysis, Front. Phys. 7 (2019) 1–11.

[30] L. Zeng, L. Li, L. Duan, K. Lu, Z. Shi, M. Wang, W. Wu, P. Luo, Distributed data mining: a survey, Inf. Technol. Manag. 13 (4) (2012) 403–409.

[31] E. Januzaj, H.-P. Kriegel, M. Pfeifle, Scalable density-based distributed clustering, in: European Conference on Principles of Data Mining and Knowledge Discovery, Springer, 2004, pp. 231–244.

[32] F. Angiulli, S. Basta, S. Lodi, C. Sartori, Distributed strategies for mining outliers in large data sets, IEEE Trans. Knowl. Data Eng. 25 (7) (2012) 1520–1532.

[33] J. Mao, P. Sun, C. Jin, A. Zhou, Outlier detection over distributed trajectory streams, in: Proceedings of the 2018 SIAM International Conference on Data Mining, SIAM, 2018, pp. 64–72.

[34] J. Chen, E.S. Azer, Q. Zhang, A practical algorithm for distributed clustering and outlier detection, 2018, arXiv preprint arXiv:1805.09495.

[35] Y. Yan, L. Cao, C. Kulhman, E. Rundensteiner, Distributed local outlier detection in big data, in: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2017, pp. 1225–1234.

[36] Y. Yan, L. Cao, E.A. Rundensteiner, Distributed Top-N local outlier detection in big data, in: 2017 IEEE International Conference on Big Data (Big Data), IEEE, 2017, pp. 827–836.

[37] P.S. Bradley, U.M. Fayyad, C. Reina, et al., Scaling clustering algorithms to large databases, in: KDD, vol. 98, 1998, pp. 9–15.

[38] J. Yin, Q. Ho, E.P. Xing, A scalable approach to probabilistic latent space inference of large-scale networks, in: Advances in Neural Information Processing Systems, 2013, pp. 422–430.

[39] S.-H. Teng, et al., Scalable algorithms for data and network analysis, Found. Trends® Theoret. Comput. Sci. 12 (1–2) (2016) 1–274.

[40] P.C. Mahalanobis, On the Generalized Distance in Statistics, National Institute of Science of India, 1936.

[41] K. Ro, C. Zou, Z. Wang, G. Yin, Outlier detection for high-dimensional data, Biometrika 102 (3) (2015) 589–599.

[42] C.C. Aggarwal, Data Mining: The Textbook, Springer, 2015.

[43] C.C. Aggarwal, Outlier analysis, in: Data Mining, Springer, 2015, pp. 237–263.

[44] R. Domingues, M. Filippone, P. Michiardi, J. Zouaoui, A comparative evaluation of outlier detection algorithms: Experiments and analyses, Pattern Recognit. 74 (2018) 406–421.

[45] X. Wang, X. Wang, M. Wilkes, New Developments in Unsupervised Outlier Detection, Springer Singapore, 2021.

[46] S. Cabras, J. Morales, Extreme value analysis within a parametric outlier detection framework, Appl. Stoch. Models Bus. Ind. 23 (2) (2007) 157–164.

[47] B. Schölkopf, J.C. Platt, J. Shawe-Taylor, A.J. Smola, R.C. Williamson, Estimating the support of a high-dimensional distribution, Neural Comput. 13 (7) (2001) 1443–1471.

[48] D.M. Tax, R.P. Duin, Support vector domain description, Pattern Recognit. Lett. 20 (11–13) (1999) 1191–1199.

[49] E.M. Knox, R.T. Ng, Algorithms for mining distancebased outliers in large datasets, in: Proceedings of the International Conference on Very Large Data Bases, Citeseer, 1998, pp. 392–403.

[50] S. Ramaswamy, R. Rastogi, K. Shim, Efficient algorithms for mining outliers from large data sets, in: Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, 2000, pp. 427–438.

[51] F. Angiulli, F. Fassetti, Dolphin: An efficient algorithm for mining distance-based outliers in very large datasets, ACM Trans. Knowl. Discov. Data (TKDD) 3 (1) (2009) 1–57.

[52] S.D. Bay, M. Schwabacher, 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, 2003, pp. 29–38.

[53] M. Sugiyama, K. Borgwardt, Rapid distance-based outlier detection via sampling, in: Advances in Neural Information Processing Systems, 2013, pp. 467–475.

[54] M.A. Rahman, K.L.-M. Ang, K.P. Seng, Unique neighborhood set parameter independent density-based clustering with outlier detection, IEEE Access 6 (2018) 44707–44717.

[55] J.L. Bentley, Multidimensional binary search trees used for associative searching, Commun. ACM 18 (9) (1975) 509–517.

[56] I. Newton, Mathematical Principles of Natural Philosophy, A. Strahan, 1802.

[57] P.J. Rousseeuw, K.V. Driessen, A fast algorithm for the minimum covariance determinant estimator, Technometrics 41 (3) (1999) 212–223.

[58] D.J. Cook, L.B. Holder, Graph-based data mining, IEEE Intell. Syst. Appl. 15 (2) (2000) 32–41.

[59] L. Akoglu, H. Tong, D. Koutra, Graph based anomaly detection and description: a survey, Data Min. Knowl. Discov. 29 (3) (2015) 626–688.

[60] S. Ranshous, S. Shen, D. Koutra, S. Harenberg, C. Faloutsos, N.F. Samatova, Anomaly detection in dynamic networks: a survey, Wiley Interdiscip. Rev. Comput. Stat. 7 (3) (2015) 223–247.

[61] R. Yu, H. Qiu, Z. Wen, C. Lin, Y. Liu, A survey on social media anomaly detection, ACM SIGKDD Explor. Newsl. 18 (1) (2016) 1–14.

[62] J.B. Tenenbaum, V. De Silva, J.C. Langford, A global geometric framework for nonlinear dimensionality reduction, Science 290 (5500) (2000) 2319–2323.

[63] F.T. Liu, K.M. Ting, Z.-H. Zhou, Isolation forest, in: 2008 Eighth IEEE International Conference on Data Mining, IEEE, 2008, pp. 413–422.

[64] F.T. Liu, K.M. Ting, Z.-H. Zhou, Isolation-based anomaly detection, ACM Trans. Knowl. Discov. Data (TKDD) 6 (1) (2012) 1–39.

[65] L. Breiman, Random forests, Mach. Learn. 45 (1) (2001) 5–32.

[66] T.R. Bandaragoda, K.M. Ting, D. Albrecht, F.T. Liu, J.R. Wells, Efficient anomaly detection by isolation using nearest neighbour ensemble, in: 2014 IEEE International Conference on Data Mining Workshop, IEEE, 2014, pp. 698–705.

[67] T.R. Bandaragoda, K.M. Ting, D. Albrecht, F.T. Liu, Y. Zhu, J.R. Wells, Isolation-based anomaly detection using nearest-neighbor ensembles, Comput. Intell. 34 (4) (2018) 968–998.

[68] T.M. Cover, Elements of Information Theory, John Wiley & Sons, 1999.

[69] Z. He, X. Xu, S. Deng, Squeezer: an efficient algorithm for clustering categorical data, J. Comput. Sci. Tech. 17 (5) (2002) 611–624.

[70] P. Filzmoser, R. Maronna, M. Werner, Outlier identification in high dimensions, Comput. Statist. Data Anal. 52 (3) (2008) 1694–1711.

[71] M. Hubert, P.J. Rousseeuw, K. Vanden Branden, ROBPCA: a new approach to robust principal component analysis, Technometrics 47 (1) (2005) 64–79.

[72] E. Ayyıldız, V. Purutçuoglu, E. Wit, A short note on resolving singularity problems in covariance matrices, Int. J. Stat. Probab. 1 (2) (2012) 113–118.

[73] O. Ledoit, M. Wolf, Honey, I shrunk the sample covariance matrix, J. Portfolio Manag. 30 (4) (2004) 110–119.

[74] K. Pearson, LIII. On lines and planes of closest fit to systems of points in space, Lond. Edinb. Dublin Phil. Mag. J. Sci. 2 (11) (1901) 559–572.

[75] W.B. Johnson, J. Lindenstrauss, Extensions of Lipschitz mappings into a Hilbert space, Contemp. Math. 26 (189–206) (1984) 1.

[76] S. Dasgupta, A. Gupta, An elementary proof of the Johnson-Lindenstrauss lemma, Int. Comput. Sci. Inst. Tech. Rep. 22 (1) (1999) 1–5.

[77] D. Achlioptas, Database-friendly random projections, in: Proceedings of the Twentieth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, ACM, 2001, pp. 274–281.

[78] E. Forgey, Cluster analysis of multivariate data: Efficiency vs. interpretability of classification, Biometrics 21 (3) (1965) 768–769.

[79] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, et al., A density-based algorithm for discovering clusters in large spatial databases with noise, in: Kdd, 1996, pp. 226–231.

[80] A. Rodriguez, A. Laio, Clustering by fast search and find of density peaks, Science 344 (6191) (2014) 1492–1496.

[81] M.A. Rahman, L.-M. Ang, K.P. Seng, Clustering biomedical and gene expression datasets with kernel density and unique neighborhood set based vein detection, Inf. Syst. 91 (2020) 101490.

[82] A. Bryant, K. Cios, RNN-DBSCAN: A density-based clustering algorithm using reverse nearest neighbor density estimates, IEEE Trans. Knowl. Data Eng. 30 (6) (2017) 1109–1121.

[83] A. Lotfi, P. Moradi, H. Beigy, Density peaks clustering based on density backbone and fuzzy neighborhood, Pattern Recognit. 107 (2020) 107449.

[84] R. Liu, H. Wang, X. Yu, Shared-nearest-neighbor-based clustering by fast search and find of density peaks, Inform. Sci. 450 (2018) 200–226.

[85] J. Xie, H. Gao, W. Xie, X. Liu, P.W. Grant, Robust clustering by detecting density peaks and assigning points based on fuzzy weighted K-nearest neighbors, Inform. Sci. 354 (2016) 19–40.

[86] R. Mehmood, G. Zhang, R. Bie, H. Dawood, H. Ahmad, Clustering by fast search and find of density peaks via heat diffusion, Neurocomputing 208 (2016) 210–217.

[87] R. Liu, W. Huang, Z. Fei, K. Wang, J. Liang, Constraint-based clustering by fast search and find of density peaks, Neurocomputing 330 (2019) 223–237.

[88] T. Liu, H. Li, X. Zhao, Clustering by search in descending order and automatic find of density peaks, IEEE Access 7 (2019) 133772–133780.

[89] R. Bie, R. Mehmood, S. Ruan, Y. Sun, H. Dawood, Adaptive fuzzy clustering by fast search and find of density peaks, Pers. Ubiquitous Comput. 20 (5) (2016) 785–793.

[90] X. Chen, W. Liu, H. Qiu, J. Lai, APSCAN: A parameter free algorithm for clustering, Pattern Recognit. Lett. 32 (7) (2011) 973–986.

[91] Z. Zhou, G. Si, Y. Zhang, K. Zheng, Robust clustering by identifying the veins of clusters based on kernel density estimation, Knowl.-Based Syst. 159 (2018) 309–320.

[92] M. Pavan, M. Pelillo, Dominant sets and pairwise clustering, IEEE Trans. Pattern Anal. Mach. Intell. 29 (1) (2006) 167–172.

[93] J. Hou, H. Gao, X. Li, DSets-DBSCAN: A parameter-free clustering algorithm, IEEE Trans. Image Process. 25 (7) (2016) 3182–3193.

[94] J. Hou, W. Liu, A parameter-independent clustering framework, IEEE Trans. Ind. Inf. 13 (4) (2017) 1825–1832.

[95] J. Kennedy, R. Eberhart, Particle swarm optimization, in: Proceedings of ICNN'95-International Conference on Neural Networks, vol. 4, IEEE, 1995, pp. 1942–1948.

[96] R.A. Maronna, R.H. Zamar, Robust estimates of location and dispersion for high-dimensional datasets, Technometrics 44 (4) (2002) 307–317.

[97] J. Leskovec, A. Rajaraman, J.D. Ullman, Mining of Massive Datasets, Cambridge university press, 2014.

[98] S. Thompson, Sampling, A Wiley-Inter-Science Publication, John Wiley & Sons, Inc., New York, 1992.

[99] C.R. Palmer, C. Faloutsos, Density biased sampling: An improved method for data mining and clustering, in: Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, 2000, pp. 82–92.

[100] G. Kollios, D. Gunopulos, N. Koudas, S. Berchtold, Efficient biased sampling for approximate clustering and outlier detection in large data sets, IEEE Trans. Knowl. Data Eng. 15 (5) (2003) 1170–1187.

[101] M. Wu, C. Jermaine, Outlier detection by sampling with accuracy guarantees, in: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2006, pp. 767–772.

[102] M. Hubert, M. Debruyne, Minimum covariance determinant, Wiley Interdiscip. Rev. Comput. Stat. 2 (1) (2010) 36–43.

[103] I.M. Johnstone, A.Y. Lu, Sparse principal components analysis, 2009, arXiv preprint arXiv:0901.4392.

[104] A. Zimek, M. Gaudet, R.J. Campello, J. Sander, Subsampling for efficient and effective unsupervised outlier detection ensembles, in: Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2013, pp. 428–436.

[105] D. Pelleg, A.W. Moore, et al., X-means: Extending k-means with efficient estimation of the number of clusters, in: Icml, vol. 1, 2000, pp. 727–734.

[106] G.O. Campos, A. Zimek, J. Sander, R.J. Campello, B. Micenková, E. Schubert, I. Assent, M.E. Houle, On the evaluation of unsupervised outlier detection: measures, datasets, and an empirical study, Data Min. Knowl. Discov. 30 (4) (2016) 891–927.

[107] R.M. Aliguliyev, Performance evaluation of density-based clustering methods, Inform. Sci. 179 (20) (2009) 3583–3602.

[108] R. Alguliyev, R. Aliguliyev, L. Sukhostat, Anomaly detection in big data based on clustering, Stat. Optim. Inf. Comput. 5 (4) (2017) 325–340.

[109] A.M. Rubinov, N. Soukhorokova, J. Ugon, Classes and clusters in data analysis, European J. Oper. Res. 173 (3) (2006) 849–865.

[110] B. Mirkin, Mathematical Classification and Clustering, vol. 11, Springer Science & Business Media, 1996.

[111] C.J. Van Rijsbergen, The Geometry of Information Retrieval, Cambridge University Press, 2004.

[112] F. Boutin, M. Hascoët, Cluster validity indices for graph partitioning, in: Proceedings. Eighth International Conference on Information Visualisation, 2004. IV 2004, IEEE, 2004, pp. 376–381.

[113] M. Meilă, Comparing clusterings by the variation of information, in: Learning Theory and Kernel Machines, Springer, 2003, pp. 173–187.

[114] D. Dua, C. Graff, UCI machine learning repository, 2017, URL: http://archive.ics.uci.edu/ml.

[115] J. Vanschoren, J.N. Van Rijn, B. Bischl, L. Torgo, OpenML: networked science in machine learning, ACM SIGKDD Explor. Newsl. 15 (2) (2014) 49–60.

[116] A. Goldbloom, B. Hamner, Kaggle data science company, 2010, URL: https://www.kaggle.com.

[117] S. Rayana, ODDS library, 2016, URL: http://odds.cs.stonybrook.edu.

[118] J. Davis, M. Goadrich, The relationship between precision-recall and ROC curves, in: Proceedings of the 23rd International Conference on Machine Learning, 2006, pp. 233–240.

[119] J. Shlens, A tutorial on principal component analysis, 2014, arXiv preprint arXiv:1404.1100.

[120] I. Jolliffe, Principal Component Analysis, Springer, 2011.

[121] D.M. Hawkins, Identification of Outliers, vol. 11, Springer, 1980.

[122] V. Barnett, T. Lewis, Outliers in Statistical Data, Wiley, 1974.

[123] M. Friedman, The use of ranks to avoid the assumption of normality implicit in the analysis of variance, J. Amer. Statist. Assoc. 32 (200) (1937) 675–701.

[124] J. Demsar, Statistical comparisons of classifiers over multiple data sets, J. Mach. Learn. Res. 7 (2006) 1–30.

[125] S. García, F. Herrera, An extension on "statistical comparisons of classifiers over multiple data sets" for all pairwise comparisons, J. Mach. Learn. Res. 9 (2009) 2677–2694.

[126] 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. 42 (3) (2017) 1–21.

[127] D. Birant, A. Kut, ST-DBSCAN: An algorithm for clustering spatial–temporal data, Data Knowl. Eng. 60 (1) (2007) 208–221.

[128] J. Sander, M. Ester, H.-P. Kriegel, X. Xu, Density-based clustering in spatial databases: The algorithm gdbscan and its applications, Data Min. Knowl. Discov. 2 (2) (1998) 169–194.

[129] A. Nickabadi, M.M. Ebadzadeh, R. Safabakhsh, A novel particle swarm optimization algorithm with adaptive inertia weight, Appl. Soft Comput. 11 (4) (2011) 3658–3670.

[130] B.S.G. de Almeida, V.C. Leite, Particle swarm optimization: A powerful technique for solving engineering problems, in: Swarm Intelligence-Recent Advances, New Perspectives and Applications, IntechOpen, 2019.

[131] B. Desgraupes, Clustering indices, Univ. Paris Ouest-Lab Modal'X 1 (2013) 34.

[132] D.L. Davies, D.W. Bouldin, A cluster separation measure, IEEE Trans. Pattern Anal. Mach. Intell. (2) (1979) 224–227.

[133] C.-H. Chou, M.-C. Su, E. Lai, A new cluster validity measure for clusters with different densities, in: IASTED International Conference on Intelligent Systems and Control, 2003, pp. 276–281.

[134] C.-H. Chou, M.-C. Su, E. Lai, A new cluster validity measure and its application to image compression, Pattern Anal. Appl. 7 (2) (2004) 205–220.

[135] C. Guan, K.K.F. Yuen, F. Coenen, Particle swarm optimized density-based clustering and classification: Supervised and unsupervised learning approaches, Swarm Evol. Comput. 44 (2019) 876–896.

[136] J.C. Dunn, Well-separated clusters and optimal fuzzy partitions, J. Cybern. 4 (1) (1974) 95–104.

[137] P.J. Rousseeuw, Silhouettes: a graphical aid to the interpretation and validation of cluster analysis, J. Comput. Appl. Math. 20 (1987) 53–65.

[138] J.D. Banfield, A.E. Raftery, Model-based Gaussian and non-Gaussian clustering, Biometrics (1993) 803–821.

[139] U. Maulik, S. Bandyopadhyay, Performance evaluation of some clustering algorithms and validity indices, IEEE Trans. Pattern Anal. Mach. Intell. 24 (12) (2002) 1650–1654.

[140] S. Saitta, B. Raphael, I.F. Smith, A bounded index for cluster validity, in: International Workshop on Machine Learning and Data Mining in Pattern Recognition, Springer, 2007, pp. 174–187.

Sayyed Ahmad Naghavi Nozad was born in Ghaen, South Khorasan Province, Iran, in 1988. He received the B.Sc. degree in Software Engineering from the Payame Noor University, Ghaen, Iran, in 2015, and the M.Sc. degree in Artificial Intelligence from the Amirkabir University of Technology, Tehran, Iran, in 2018. His research interests include but are not limited to Outlier Detection, Cluster Analysis, Community Detection, Novelty Detection, Information Theory, Image Processing, and Deep Learning.

Maryam Amir Haeri received her bachelor's degree in Software Engineering in 2007 and her master's degree in Information Technology in 2009, both from Sharif University of Technology, Tehran, Iran. She also received her doctorate in Artificial Intelligence from the Amirkabir University of Technology, Tehran, Iran, in 2014. She served as an assistant professor in the Computer Engineering Department, Amirkabir University of Technology, from September 2015 to January 2019. In addition, she was a research fellow at the Algorithm Accountability Lab at the University of Kaiserslautern, Germany, from February 2019 to October 2020. Since November 2020, she has been an assistant professor at the Learning, Data-Analytics and Technology Department, University of Twente, the Netherlands. Her research interests include Machine Learning, Big Data Analytics, Fairness in Machine Learning, and Complex Networks Analysis.

Gianluigi Folino holds a Ph.D. in Physics, Mathematics, and Computer Science (Radboud University, Nijmegen, Holland, 2010). Since 2001, he works as a senior researcher at ICAR-CNR (the Institute of High Performance Computing and Networking of the Italian National Research Council). He is also a lecturer at the University of Calabria and the University Magna Graecia of Catanzaro. His research interests focus on applications of distributed computing in the area of data mining, bio-inspired algorithms (particularly genetic programming and swarm intelligence), big data, and bioinformatics. Within ICAR-CNR, he has been a contributor to several national and international research/industrial projects, and since 2013, he has been the coordinator of the project "Cyber Security – Digital and electronic payment services protection". In addition, he was a visiting researcher at the University of Nottingham (United Kingdom) in 2007 and 2009, at Radboud University, Nijmegen (Netherlands), in 2008 and 2009, and finally at the University of California (UCLA), Los Angeles, in 2013.