Highlights

SDCOR: Scalable Density-based Clustering for Local Outlier Detection in Massive-Scale Datasets

Sayyed-Ahmad Naghavi-Nozad,Maryam Amir Haeri,Gianluigi Folino

- We introduce a novel scalable density-based clustering approach for local outlier detection in massive data.
- The input data is processed in chunks, hence, there is no need for it to be entirely loaded in memory.
- Our assessments prove that the proposed method has a linear time complexity with a low constant.
- Experiments on real and synthetic data show that our proposed approach is more effective and efficient than traditional density-based methods, which require all the data be resident in memory; and also compared to some state-of-the-art distance-based methods which do not need the data be totally present in memory during the training stage.

SDCOR: Scalable Density-based Clustering for Local Outlier Detection in Massive-Scale Datasets

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

[a]*Department of Computer Engineering and Information Technology, Amirkabir University of Technology, Tehran, Iran*
[b]*ICAR-CNR, Via P.Bucci 7/C, Univ. della Calabria 87036 Rende (CS), Italy*

ARTICLE INFO

Keywords:
local outlier detection
massive-scale datasets
scalable
density-based clustering
anomaly detection

ABSTRACT

This paper presents a batch-wise density-based clustering approach for local outlier detection in massive-scale datasets. Differently from well-known traditional algorithms, which assume that all the data is memory-resident, our proposed method is scalable and processes the data chunk-by-chunk within the confines of a limited memory buffer. At first, a temporary clustering model is built, then it is incrementally updated by analyzing consecutive memory loads of points. Ultimately, the proposed algorithm will give an outlying score to each object, which is named 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 the data into memory; and also some fast distance-based methods which can perform on the data resident in the disk.

1. Introduction

Outlier detection, which is a noticeable and open line of research [12, 23, 50], 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 smaller 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 model accuracy and reduce the computational loads of the algorithm. However, outliers are not always annoying and sometimes, they become of special interest for the data analyst in many problems such as controlling cellular phones activity to detect fraudulent usage, like stolen phone airtime. Outlier detection methods could also be considered as a preprocessing step, 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 [23, 4].

Outliers can be generally divided into two categories, namely global and local. Global outliers are objects, which show significant abnormal behavior in comparison to the rest of the data, and thus in some cases they are considered as point anomalies. On the contrary, local outliers only deviate significantly w.r.t. a specific neighborhood of the object [12, 20]. In [10, 14], it is noted that the concept of local outlier is more comprehensive than that of global, i.e. a global outlier could also be considered as local, but not vice versa. This is the reason that makes finding local outliers much more cumbersome.

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 traditional software methods for processing and management of this 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.

To overcome the above-mentioned issues, 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 very large datasets, in which outlier identification comes after that as a side effect; it is inspired by a scaling clustering algorithm for very large databases, named after its authors, BFR [8]. BFR 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 a memory buffer, and then it updates the current clustering model over the content of this buffer. Based on the updated model, singleton data are classified in three groups; some of them can be discarded with updates to the sufficient statistics (discard set DS), some can be reduced via compression and summarized as sufficient statistics (compression set CS), and some need to be retained in the buffer (retained set RS).

Like BFR, our proposed method operates within the confines of a limited memory buffer; thus, by 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 data chunk-by-chunk, so that there is enough space for both loading and process-

[*]Corresponding author

sa_na33@aut.ac.ir (Sayyed-Ahmad Naghavi-Nozad); haeri@aut.ac.ir (Maryam Amir Haeri); gianluigi.folino@icar.cnr.it (Gianluigi Folino)
ORCID(s):

ing each chunk at the same time. The proposed approach is based on clustering, and therefore, it must avoid that the outliers play any 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 an incremental way and finally provides an approximate result, from an operational perspective is called a scalable algorithm [48]. Moreover, from an algorithmic point of view, scalability means that algorithm complexity should be nearly linear or sublinear w.r.t. the problem size [46].

In more detail, the proposed method includes three steps. In the first step, a primary random sampling is carried out to create an abstract 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 incremental clustering will be acquired. In the second step, a scalable density-based clustering algorithm is executed in order to identify dense regions, on the basis of the chunk of data currently loaded in memory, and to build incrementally some clusters (named mini-clusters or sub-clusters). When all the chunks are processed, the final clustering model will be built by merging the information obtained through these mini-clusters. Finally, by applying a Mahalanobis distance criterion [36, 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 the original clusters.
- It works better with Gaussian clusters with correlated or uncorrelated features, but it can also work well with convex-shaped clusters with 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 memory; and also with some fast distance-based methods which do not require to load the entire data into memory at their training stage.

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

2. Related Works

Outlier detection methods can be divided into following six categories [2]: extreme value analysis, probabilistic methods, distance-based methods, information-theoretic methods, clustering-based methods, and density-based methods.

In extreme value analysis, the overall population is supposed as having a unique probability density distribution, and only those objects at the very ends of it are considered as outliers. In particular, these types of methods are useful to find global outliers [2, 11].

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 [2]. Schölkopf et al. [42] 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 find 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; in practice, a small number of outliers is considered as belonging to a rare class and the rest of data as normal objects.

In distance-based methods, the distances among all objects are computed to detect outliers. An object is assumed to be a distance-based outlier, iff it has d_0 distance away from at least fraction p_0 of other objects in the dataset [31]. Bay and Schwabacher [7] propose an optimized nested-loop algorithm based on the k nearest neighbor distances among objects, that has a near linear time complexity, and is shortly named ORCA. 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 memory. It keeps track of a set of user-defined number of data points as potential anomalies along with their anomaly scores. The minimum anomaly score of the set is used as a cut-off, and will be updated if there is a point with a higher score in other blocks. If a point has a lower score than the cut-off, the point will be pruned. This pruning process only speeds up the distance calculation if the ordering of data is uncorrelated. 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 kth nearest neighbor or average distance of k nearest neighbors. Furthermore, in [49], a Local Distance-based Outlier Factor (LDOF) is proposed to find outliers in scattered datasets, which uses the relative distance from an object to its neighbors. S_p [44] 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 has a linear time complexity in the number of objects, dimensions, and samples, and also a constant space complexity, which makes it ideal for analyzing massive datasets.

Information-theoretic methods could be considered as almost equivalent to distance-based and other deviation-based models, except that the outlier score is defined by the model size for a fixed deviation, rather than the deviation for a fixed model [2]. Wu and Wang [47] 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 of 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.

Clustering-based methods use a global analysis to detect crowded regions and outliers will be those objects not belonging to any cluster [2]. A Cluster-Based Local Outlier Factor (CBLOF) in [22], and a Cluster-Based Outlier Factor (CBOF) in [16] are presented, which in both of them, after the clustering procedure is carried out, due to a specific criterion, clusters are divided into two large and small groups; and it is assumed that outliers lie in small clusters. At last, the distance of each object to its nearest large cluster is used in different ways to define the outlier score.

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 that of 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 [10, 2]. For example, Breunig et al. [10] 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 which is that 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 [32] 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. An INFLuenced Outlierness (INFLO) score is presented in [26], which adopts both neighbors and reverse neighbors of an object to estimate its relative density distribution. Moreover, Tang and He [45] propose a local density-based outlier detection approach, 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 k-nearest-neighbors of an object are taken into account but, in addition, the reverse-nearest-neighbors and the shared-nearest-neighbors are considered for density distribution estimation as well.

Beyond the mentioned six categories of outlier detection methods, there is another state-of-the-art ensemble method based on the novel concept of isolation, named *i*Forest [34, 35]. *i*Forest derives its motivation from another ensemble technique known as Random Forests [9], which are commonly used in classification. In this case, the data is recursively partitioned by axis-parallel cuts along randomly selected attributes, so as to isolate different kinds of instances from one another. In such cases, the tree branches containing outliers are noticeably less deep, because these data points are quite different from the normal data. Thus, data points which have noticeably shorter paths in the branches of different trees are more likely to be outliers. One major challenge of using such an approach is that when the dimensionality of the data increases, an incorrect choice of attribute for splitting at the higher levels of the tree is more likely to mislead the detection approach. Nevertheless, the use of isolation makes it possible for *i*Forest to exploit sub-sampling to an extent that is not feasible in existing methods, creating an algorithm which has a linear time complexity with a low constant and a low memory requirement [3].

Remark 1. According to the fact that during the scalable clustering, we use the Mahalanobis distance measure to assign each object to a mini-cluster; and besides, the size of the temporary clusters is much smaller than that of the original clusters, it would be worth mentioning an important matter here. With respect to [41, 18, 25, 5], 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; hence, the corresponding Mahalanobis distance will no longer be reliable.

Therefore, to overcome such problem, in a preprocessing step, we need to resort to dimensionality reduction approaches. However, due to the serious dependence of some dimensionality reduction methods like PCA [38] 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 use of a random basis projections [27, 13, 1]. The main characteristic of these types of methods is that they will approximately preserve the pairwise euclidean distances between data points, and, in addition, the dimension of the transformed space is independent of the original dimension, and only depends logarithmically on the number of data points. Finally, after such preprocessing step, we can be optimistic that the singularity problem will not be present during the clustering procedures or, in the case of it would be present, 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 [19] in almost all of its clustering procedures. In addition to this drawback of the K-means algorihtm, which is being seriously depending on foreknowing the true number of the original clusters, in the case of the presence of outliers, K-means performs poorly and, therefore, we need to resort to a density-based clustering approach, like DBSCAN [17][1].

However, DBSCAN is strongly dependent on the choice of its parameters. Thus, we are forced to utilize some optimization algorithm to find the optimal values for these pa-

[1] Although we will demonstrate that, sometimes, even DBSCAN may fail during the scalable clustering, to form regular mini-clusters and hence, we will be forced to use the same K-means to fix the issue.

rameters. Here, we prefer to use the evolutionary algorithm PSO [30].

Remark 3. Another important difference between the proposed method and BFR concerns the volume of the structural information, which they need to store for the clustering procedure. 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, hence, this matrix will be literally the most prominent property of each sub-cluster. But according to the high computational expense of computing Mahalanobis distance in high-dimension spaces, thus, as in [18, 37], we will use properties of principal components in the transformed space. Therefore, the covariance matrix of each mini-cluster will become diagonal and by transforming each object to the new space of the mini-cluster, like BFR, we can calculate the Mahalanobis distance without the need to use matrix inversion.

According to [33], 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 the points on this radius are as far as four standard deviations from the mean, and 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 incremental clustering. In the second phase, a scalable density-based clustering algorithm is carried out in order to recognize dense areas, on the basis of currently loaded chunk of data points in memory. Clusters built incrementally in this phase, are called mini-clusters or sub-clusters, and they form the temporary clustering model. After loading each chunk of data, according to the points already loaded in memory and those undecided from previous chunks; and by employing the Mahalanobis distance measure and the density-based clustering criteria; we update the temporary clustering model, which consists of making some changes to existing mini-clusters or adding new sub-clusters.

Note that, in the whole scalable clustering procedure, our endeavor is aimed to not let outliers participate actively in forming and updating any mini-cluster, and thus, after processing the entire chunks, there will be some objects in buffer remained undecided. Some of these data are true outliers, while some others are inliers, which, due to constraints, have failed to play any effective role in forming a sub-cluster. Finally, all these undecided points are cleared from the buffer, while only the structural information of the temporary clusters is maintained in memory. Then, at the last part of the scalable clustering algorithm, we utilize another clustering-based approach to combine the mini-clusters and 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.

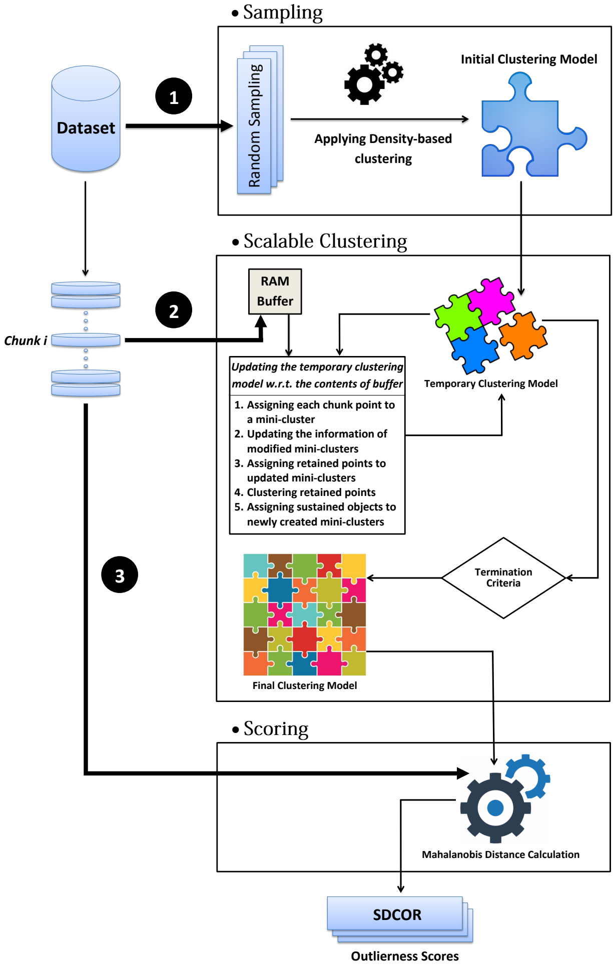

• Sampling
Dataset
1
Random Sampling
Applying Density-based clustering
Initial Clustering Model
• Scalable Clustering
2
Chunk i
RAM Buffer
Updating the temporary clustering model w.r.t. the contents of buffer
1. Assigning each chunk point to a mini-cluster
2. Updating the information of modified mini-clusters
3. Assigning retained points to updated mini-clusters
4. Clustering retained points
5. Assigning sustained objects to newly created mini-clusters
Temporary Clustering Model
Termination Criteria
Final Clustering Model
3
• Scoring
Mahalanobis Distance Calculation
SDCOR
Outlierness Scores

Figure 1: Software architecture of the proposed approach

The framework of 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 details in the next subsections.

3.1. Sampling

In this phase, we generate a random sample of the entire dataset. It is not always easy to obtain a sufficient number of

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 points
$\mathcal{Y} \subset \mathcal{X}$	A chunk of data
Ω	A partition of mini-clusters points as $\left\{\mathbb{X}_1, \cdots, \mathbb{X}_{\Bbbk}\right\}$
$n_{\mathfrak{C}}$	Number of objects in $\mathcal{Y}$
$\{\Gamma\}_{7\times\mathcal{L}}$	Information array of temporary clusters with 7 properties for each mini-cluster
$\left\{\Gamma_i\right\}_{7\times 1} \in \Gamma$	Information standing for the ith mini-cluster in $\Gamma, 1 \leq i \leq \mathcal{L}$
X_i	mini-cluster points associated with Γ_i, which are removed from buffer
$\mathfrak{R}$	Retained set of objects in RAM buffer
$\mathfrak{R}_i$	The ith mini-cluster of retained points in buffer, discovered through DBSCAN
$\mathfrak{R}_\zeta$	Retained set of objects in buffer introduced as noise by DBSCAN
γ	List of indices to recently created or updated mini-clusters associated with Γ, to be checked on for membership
γ'	Temporary list of indices to recently created or updated mini-clusters 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 ith PC coefficient
λ_i	The ith PC variance
$\mathcal{L}$	Current number of temporary clusters associated with Γ
$\Bbbk$	Number of mini-clusters which are about to be added to temporary clustering model
$\mathbb{K}$	K-means parameter for number of clusters
$\mathbb{K}'$	Number of retained set sub-clusters discovered through DBSCAN
$\mathbb{K}''$	Number of retained set subdivided sub-clusters obtained out of K-means
$\mathcal{T}$	True number of original clusters in $\mathcal{X}$
$\{F\}_{2\times\mathcal{T}}$	Information array of final clustering model with 2 properties for each ultimate cluster
$[\mathcal{M}]_{\mathcal{L}\times p}$	Means matrix of whole mini-clusters
$\mathcal{M}_i$	The ith final cluster, comprising of some temporary means
μ_f	Final mean location of an ultimate cluster
Σ_f	Final covariance structure of an ultimate cluster
$\mathcal{U}$	A set of regenerated points
η	Random sampling rate
Λ	PC total variance ratio for temporary clusters
α	Membership threshold for temporary clusters
β	Pruning threshold for final clusters
ε	DBSCAN parameter Epsilon
ψ	DBSCAN parameter MinPts
$\varepsilon_{\mathcal{S}}$	DBSCAN parameter ε required for clustering $\mathcal{S}$
$\psi_{\mathcal{S}}$	DBSCAN parameter ψ required for clustering $\mathcal{S}$
$\mathbb{C}_\varepsilon$	Coefficient of $\varepsilon_{\mathcal{S}}$ necessary for clustering $\mathcal{X}$
$\mathbb{C}_\psi$	Coefficient of $\psi_{\mathcal{S}}$ necessary for clustering $\mathcal{X}$
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
$SingCheck\left(\Sigma_X\right)$	A function that checks on the singularity of Σ_X, and outputs 1 in the case of being singular and 0 if not
$CohrCheck(X)$	A function that checks on the coherence of input data X, so that whether or not only one dense cluster will be discovered through DBSCAN. It outputs 1 in the case of being coherent, and 0 if not
det_{Σ_X}	Covariance determinant of cluster X
$\vec{\delta}$	Vector of maximum covariance determinant condition for mini-clusters discovered through scalable clustering
$\lvert\cdot\rvert$	Cardinality of a set of objects
Φ	The empty set
κ	A low constant near zero

Algorithm 1: Framework of SDCOR

```
Input  : $[\mathcal{X}]_{n\times p}$ - The $n$ by $p$ input dataset $\mathcal{X}$; $\eta$ - Random
         sampling rate; $\Lambda$ - PC total variance ratio; $\alpha$ -
         Membership threshold; $\beta$ - Pruning threshold; $\mathbb{C}_\varepsilon$ -
         Sampling Epsilon coefficient; $\mathbb{C}_\psi$ - Sampling MinPts
         coefficient
Output : The outlying score for each object in $\mathcal{X}$
Phase 1 — Sampling:
Step 1. Take a random sample $\mathcal{S}$ of $\mathcal{X}$ according to the sampling
    rate $\eta$
Step 2. Employ the PSO algorithm to find optimal values for the
    DBSCAN parameters $\varepsilon_{\mathcal{S}}$ and $\psi_{\mathcal{S}}$, required for clustering $\mathcal{S}$
Step 3. Run DBSCAN on $\mathcal{S}$ using the obtained optimal
    parameters, and reserve the count of the discovered mini-clusters
    as $\mathcal{T}$, as the true number of original clusters in data
Step 4. Build the very first array of mini-clusters information
    (temporary clustering model) out of result of step 3, w.r.t.
    Algorithm 2
Step 5. Reserve the covariance determinant values of the initial
    sub-clusters as a vector $\vec{\delta} = \left[\delta_1, \cdots, \delta_{\mathcal{T}}\right]$, for the maximum
    covariance determinant condition
Step 6. Clear $\mathcal{S}$ from RAM and maintain the initial temporary
    clustering model in buffer
Phase 2 — Scalable Clustering:
Prepare input data to be processed chunk by chunk, so that each
    chunk can be fit and be processed in the RAM buffer at the same
    time
Step 1. Load the next available chunk from data into RAM
Step 2. Update the temporary model of clustering over the
    contents of buffer, w.r.t. Algorithm 3
Step 3. If there is any available unprocessed chunk, go to step 1
Step 4. Build the final clustering model, w.r.t. Algorithm 7, using
    the temporary clustering model obtained out of the previous
    steps
Phase 3 — Scoring:
According to the final clustering model, for each data point $x \in \mathcal{X}$,
    use the Mahalanobis distance criterion to find the closest cluster,
    and finally assign $x$ to that cluster and use the criterion value as
    the object outlierness score
```

samples from each original cluster, and thus the clustering structure obtained after applying the sampling may not be representative of the original one. Hence, the outliers could be misclassified during the scalable clustering. In such a situation, obtaining a satisfactory random sample requires processing the entire dataset. In the following, the percentage of sampled data, is indicated as η.

After obtaining the sampled data, we conduct DBSCAN algorithm to cluster them, and assume that the number of mini-clusters obtained through this, is the same as number of original clusters $\mathcal{T}$ in the main dataset. We reserve $\mathcal{T}$ for later use. Besides, we presume that the location (centroid) and the shape (covariance structure) of such sub-clusters are so close to the original ones. In Section 4, we will show that even by using a small rate of random sampling, the mentioned properties of sampled clusters are similar enough to the original ones. The idea behind making these primary sub-clusters, which are so similar to the original clusters in the input dataset in terms of basic characteristics, is that, we intend to determine a Mahalanobis radius, which collapses a specific percentage of objects belonging to each original cluster; and let other sub-clusters be created around this folding area during successive memory-loads of points; and ultimately, by merging these mini-clusters, we will obtain the approximate structure of the original clusters.

As we adopt PSO to attain optimal values of parameters $\varepsilon_{\mathcal{S}}$ and $\psi_{\mathcal{S}}$ for DBSCAN, it would be essential to stipulate this truth that as the density of the sampled distribution is much less than that of the original data, we cannot use the same parameters for the original distribution, while applying DBSCAN on objects loaded in memory, during the scalable clustering. Thus, we have to use a coefficient in the interval [0,1] for each parameter. It is also necessary to mention that ε is much more sensitive than ψ, as with a slight change in the value of ε, we may observe a serious deviation in the clustering result, but this does not apply to ψ. We show the mentioned coefficients with $\mathbb{C}_\varepsilon$ and $\mathbb{C}_\psi$, for $\varepsilon_{\mathcal{S}}$ and $\psi_{\mathcal{S}}$ respectively, and their values could be obtained through the user, but the best values gained out of our experiments are 0.5 and 0.9, for $\mathbb{C}_\varepsilon$ and $\mathbb{C}_\psi$ respectively.

Now, before building the first clustering model, we need to extract some information out of the sampled clusters obtained through DBSCAN, and store them in a special array. As stated earlier about the benefit of using properties of principal components (PCs) for high-dimensional data, we need to find those PCs that give higher contributions to the cluster representation. To this aim, we sort PCs on the basis of their corresponding variances in descending order, and then we choose the topmost PCs having their share of 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 total n objects in dataset $[\mathcal{X}]_{n\times p}$, belonging to the temporary cluster $\left[\mathbb{X}_i\right]_{m\times p}$; then the information about this sub-cluster as $\left\{\Gamma_i\right\}_{7\times 1}$, in the array of temporary clustering model $\{\Gamma\}_{7\times\mathcal{L}}$, 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, $\mathbb{S}_{\mathbb{X}_i} = \sum_{x\in\mathbb{X}_i}(x-\mu_{\mathbb{X}_i})^t\cdot(x-\mu_{\mathbb{X}_i})$
3. p' superior components, $\left[e_1,\cdots,e_{p'}\right]$, 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, $\left[\sqrt{\lambda_1},\cdots,\sqrt{\lambda_{p'}}\right]$
6. Size of the mini-cluster, m
7. Value of p'

Algorithm 2 demonstrates the process of obtaining and adding this information per each mini-cluster to the temporary clustering model. We also use this algorithm while adding the information of new discovered mini-clusters out of the scalable clustering to the temporary clustering model.

```
Algorithm 2: [Γ] = MiniClustMake(Γ, Ω, Λ)
Input  : Γ - Current array of mini-clusters information;
         $\Omega = \left\{\mathbb{X}_1,\cdots,\mathbb{X}_{\Bbbk}\right\}$ - Partition of mini-clusters points; Λ -
         PC share of total variance
Output : Γ - Updated temporary clustering model
$c \leftarrow \mathcal{L}$
foreach mini-cluster $\mathbb{X}_i, 1 \le i \le \Bbbk$ do
    Apply PCA on $\mathbb{X}_i$ and obtain its PC coefficients and
      variances. Then choose $p'$ as the number of the top PC
      variances, for which their share of total variance is at least
      $\Lambda$ percent
    $\Gamma\{1, c+i\} \leftarrow$ Mean vector of $\mathbb{X}_i$
    $\Gamma\{2, c+i\} \leftarrow$ Scatter matrix of $\mathbb{X}_i$
    $\Gamma\{3, c+i\} \leftarrow$ Top $p'$ PC coefficients corresponding to the
      top $p'$ PC variances
    $\Gamma\{4, c+i\} \leftarrow$ Transformed mean vector, as
      $\Gamma\{1, c+i\}\cdot\Gamma\{3, c+i\}$
    $\Gamma\{5, c+i\} \leftarrow$ Square root of the top $p'$ PC variances
    $\Gamma\{6, c+i\} \leftarrow$ Number of objects in $\mathbb{X}_i$
    $\Gamma\{7, c+i\} \leftarrow$ Value of $p'$
end
```

According to [24, 28], 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 Mahalanobis contour lines[2] will also become wider than that of the real clusters, 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 ellipses, which could be referred to as the spatial volume of the cluster. Moreover, by being contaminated, this volume could increase and become harmful.

Since during the scalable clustering, new objects are coming over time and mini-clusters are growing gradually, so, it is possible for a mini-cluster with an irregular shape to accept some outliers. Then, the following covariance matrix will no longer be robust, and the corresponding Mahalanobis contour lines 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 sub-clusters, which could be contaminated with myriad outliers, we have to put a limit on the covariance determinant of every sub-cluster, which is discovered through scalable clustering. Here, we follow a heuristic approach and employ the covariance determinant of the nearest initial sub-cluster obtained out of the "Sampling" stage, as the limit.

This problem that outliers can be included in the clusters and could no longer be detected, is called masking effect. Note that we are using the mentioned constraints only when sub-clusters are created for the first time, not while they are growing over time. The justification is that, while an object is about to be assigned to a mini-cluster, the other constraint on the Mahalanobis radius, within reason, is hindering outliers to be 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 sub-cluster, if and only if that threshold is set to a fair value. Hence, we do not check on the covariance determinant of sub-clusters, while they are growing.

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

Here, the first phase of the proposed approach is finished, and we need to clear 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 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, which are currently loaded in memory from the current chunk or retained from 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. Contents of Buffer

After loading each chunk into memory, the temporary clustering model is updated on the basis of objects coming from the currently loaded chunk and the other ones sustained from the previously loaded data. First of all, the algorithm checks for the possible membership of each point of the currently loaded chunk to any existing mini-cluster in the temporary clustering model[3].

Then, after the probable assignments of the current chunk points, there are some primary and secondary information of the modified sub-clusters that shall be updated. After this update, the structure of the altered sub-clusters 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 memory to updated sub-clusters. This updating and assignment checking cycle will keep going until there is not any retained point that could be assigned to an updated mini-cluster.

When the membership evaluation of the present chunk and retained objects is carried out, the algorithm tries to cluster the remaining sustained objects in memory, regarding the density-based clustering criteria which have been constituted at the Sampling phase. After the new mini-clusters were created out of the last retained points, there is this probability that some sustained inliers in the buffer might not be capable of participating actively in forming new sub-clusters, because of the density-based clustering standards, though could be assigned to them considering 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 needed for updating the temporary clustering model, w.r.t. an already loaded chunk of data and other undecided objects retained in memory from before. The following subsections will explain the details of this algorithm.

Algorithm 3: $\left[\Gamma, \mathfrak{R}\right] = \text{MemoProcess}(\mathcal{Y}, \Gamma, \alpha, \mathfrak{R}, \vec{\delta}, \varepsilon_{\mathcal{S}}, \psi_{\mathcal{S}}, \mathbb{C}_{\varepsilon}, \mathbb{C}_{\psi}, \Lambda)$

```
Input  : $\mathcal{Y}$ - A chunk of data; $\Gamma$ - Current array of mini-clusters
         information; $\alpha$ - Membership threshold; $\mathfrak{R}$ - Retained
         set; $\vec{\delta}$ - Covariance determinant threshold; $\varepsilon_{\mathcal{S}}$ - Sampling
         Epsilon; $\psi_{\mathcal{S}}$ - Sampling MinPts; $\mathbb{C}_{\varepsilon}$ - Sampling Epsilon
         coefficient; $\mathbb{C}_{\psi}$ - Sampling MinPts coefficient; $\Lambda$ - PC
         share of total variance
Output : $\Gamma$ - Updated temporary clustering model; $\mathfrak{R}$ - Modified
         retained set

/* Trying to assign each datum of a chunk to a mini-cluster */
$\gamma \leftarrow \{1, \cdots, \mathcal{L}\}$
$\left[\Gamma, \gamma, \mathfrak{R}\right] = \text{MiniClustUpdate}(\mathcal{Y}, \Gamma, \gamma, \alpha, \mathfrak{R})$
/* Checking out retained set */
if $|\mathfrak{R}| \neq 0$ then
    /* Checking on retained set membership for recently updated
       mini-clusters */
    $\left[\Gamma, \mathfrak{R}\right] = \text{RetSetMemb}(\mathfrak{R}, \Gamma, \gamma, \alpha)$
    /* Clustering retained set */
    if $|\mathfrak{R}| \neq 0$ then
        $l \leftarrow \mathcal{L}$
        $\left[\Gamma, \mathfrak{R}\right] = \text{RetSetClust}(\mathfrak{R}, \Gamma, \vec{\delta}, \varepsilon_{\mathcal{S}}, \psi_{\mathcal{S}}, \mathbb{C}_{\varepsilon}, \mathbb{C}_{\psi}, \Lambda)$
        $\gamma \leftarrow \{l+1, \cdots, \mathcal{L}\}$
        /* Checking on retained set membership for recently created
           mini-clusters */
        if $|\mathfrak{R}| \neq 0$ then
            $\left[\Gamma, \mathfrak{R}\right] = \text{RetSetMemb}(\mathfrak{R}, \Gamma, \gamma, \alpha)$
        end
    end
end
```

3.2.1.1. Trying to Assign Each Tuple of the Chunk to a mini-cluster

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

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

Where $MD(x, X_i)$ is the Mahalanobis distance of object x from mini-cluster X_i; and $z = x \cdot \mathcal{A}_{X_i}$ is the object in the eigenvector space of the mini-cluster, and z_j is its jth component; μ'_j and λ_j are respectively the jth components of the mean vector and the variance vector in the space of eigenvectors; and finally, p' is the number of superior components associated with the mini-cluster. 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. [33], the accepted Mahalanobis radius in the eigenvector space of the relevant sub-cluster will be the product of membership threshold and square root of number of dimensions in the transformed space, as $\alpha \cdot \sqrt{p'}$.

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

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

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

```
Input : $\mathbb{X}$ - A set of points; $\Gamma$ - Current array of mini-clusters
        information; $\gamma$ - List of indices to recently created or
        updated mini-clusters associated with $\Gamma$, to be checked
        on for membership; $\alpha$ - Membership threshold; $\mathfrak{R}$ -
        Retained set
Output : $\Gamma$ - Updated temporary clustering model; $\gamma' \subseteq \gamma$ -
        Modified list of indices to recently updated mini-clusters;
        $\mathfrak{R}$ - Modified retained set

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

3.2.1.2. *Updating Primary and Secondary Information of Temporary Clusters*

Related information to a sub-cluster which needs to be updated after the objects assignment are twofold, primary and secondary. Primary information comprises of the scatter matrix of the sub-cluster 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; and for the cardinality, the number of objects assigned to the sub-cluster is increased by one. Each object, after joining a sub-cluster is removed from the buffer; otherwise will be retained to be decided on later.

After checking on the membership of all points and updating the primary information of sub-clusters, for each mini-cluster which has accepted any new members, its PC properties, which are considered as its secondary information, must be updated too. For this purpose, due to the size of the mini-cluster, 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 mini-cluster. Algorithm 4, demonstrates the required steps for finding the closest sub-cluster due to relevant limitations; and updating its information.

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

```
Input : $\mathfrak{R}$ - Retained set; $\Gamma$ - Current array of mini-clusters
        information; $\gamma$ - List of indices to recently created or
        updated mini-clusters associated with $\Gamma$, to be checked
        on for membership; $\alpha$ - Membership threshold
Output : $\Gamma$ - Updated temporary clustering model; $\mathfrak{R}$ - Modified
        retained set

if $|\gamma| \neq 0$ then
    while true do
        $[\Gamma, \gamma, \mathfrak{R}] = \text{MiniClustUpdate}(\mathfrak{R}, \Gamma, \gamma, \alpha, \Phi)$
        if $|\gamma| \equiv 0$ then
            break
        end
    end
end
```

3.2.1.3. *Trying to Assign Retained Objects in Buffer to Newly Updated mini-clusters*

After updating the secondary information of each sub-cluster, the corresponding tolerance ellipses will rotate smoothly around the centroid, and in other words, their accepted Mahalanobis neighborhood is modified. Hence, w.r.t. Algorithm 5, it would be necessary to check on the objects retained in buffer, i.e. whether they can belong to a modified mini-cluster, and if it is so, the corresponding mini-cluster information needs to be updated, w.r.t. Algorithm 4. But, this is not the end; by keeping up this cycle of membership checking and mini-cluster updating, more and more objects could be assigned to mini-clusters and then be discarded. In this iterative manner, after each iteration, memory contents should be evaluated using only updated sub-clusters from the last iteration; and thus, the list of updated mini-clusters 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 mini-clusters; or every retained object has been eventually assigned to an updated mini-cluster. Here, by employing this procedure, it seems that tolerance ellipses of mini-clusters are sweeping inliers through the cycle of assignment and updating.

Fig. 2 shows the scenario in which, after updating the core information of a sub-cluster, its Mahalanobis neighborhood is modified; and some objects which were not able to belong to this sub-cluster, now are capable of being assigned to it. Black circle points and black dashed line tolerance ellipse respectively represent a sub-cluster and its Mahalanobis neighborhood. Red circle points represent objects assigned to the sub-cluster during the last memory process; as they reside in the accepted neighborhood of it. The red dashed line represents the updated tolerance ellipse of the updated sub-cluster. And finally, blue triangle points represent objects which could be assigned to the updated sub-cluster; if they lie in the updated Mahalanobis radius of it.

3.2.1.4. *Clustering Retained Objects in Buffer*

Here, after checking on the membership of each tuple stored in the

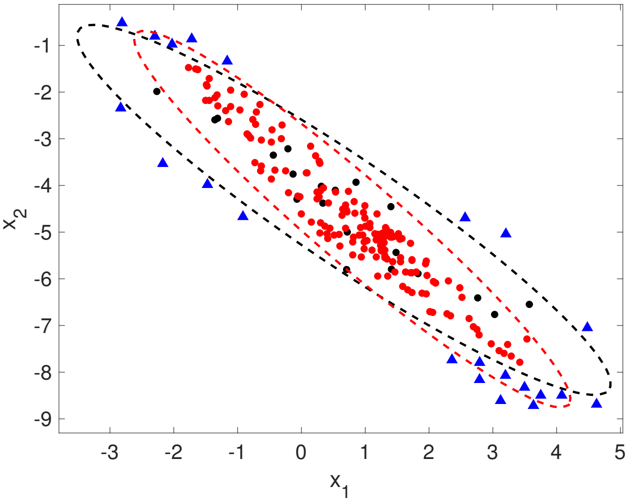
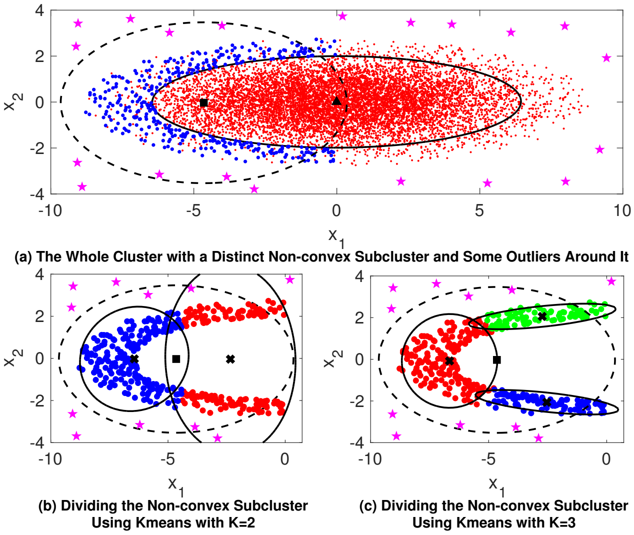


Figure 2: Trying to assign retained objects in buffer to newly updated sub-clusters



(a) The Whole Cluster with a Distinct Non-convex Subcluster and Some Outliers Around It

(b) Dividing the Non-convex Subcluster Using Kmeans with K=2

(c) Dividing the Non-convex Subcluster Using Kmeans with K=3

Figure 3: Breaking a non-convex mini-cluster with a very wide tolerance ellipse into smaller pieces by the K-means algorithm[4]

memory, we afford to cluster retained data in RAM buffer, using again the DBSCAN algorithm. However, as emphasized earlier at the "Sampling" stage, according to the significant difference in the density of sampled and original data, we have to use predefined coefficients for the obtained optimal parameters out of the sampled data.

Furthermore, as it was described before, it is possible that some mini-clusters could be discovered during the scalable clustering by DBSCAN, which are suffering from singularity problem. Thus, for handling such 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 mini-cluster and let its points still be in memory, to be resolved later. Therefore, for every discovered sub-cluster, we shall check on its covariance matrix, whether or not it is singular; and in the case of singularity, we disregard that sub-cluster.

Now, w.r.t. this prementioned matter that, we have to put a limit on the boundaries of the mini-clusters which are being created during the scalable clustering, we are going to demonstrate with an intuitive example that if a mini-cluster with a non-convex shape is formed, how outliers could be absorbed to such irregular mini-cluster and cause serious 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 sub-cluster shown with blue dots; and a black square and a black dashed line as its centroid and accepted tolerance ellipse, respectively. The irregular mini-cluster is formed around the initial mini-cluster; while its centroid and accepted Mahalanobis radius are denoted as a black triangle and a black solid line, respectively. There are also some local outliers around the original cluster, which are illustrated with magenta pentagons[5].

As it is evident, the irregular mini-cluster 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 mini-cluster is equal to 15.15, which is almost twice that of the initial mini-cluster equal to 7.88. Thus, for fixing this concern, by considering the proportion between the covariance determinant of an arbitrary cluster and its spatial volume, we decide to divide the irregular mini-cluster to smaller coherent pieces, with smaller covariance determinants; and also, more limited Mahalanobis radii as well. Therefore, we heuristically set the threshold value for the covariance determinant of any newly discovered sub-cluster or subdivided sub-cluster, as that of the nearest initial mini-cluster[6].

For the division process of an irregular sub-cluster, we prefer to adopt K-means algorithm. However, K-means can cause some incoherent subdivided sub-clusters in such cases[7], as shown in Fig. 3b. In Fig. 3b, two smaller mini-clusters, produced as K-means result, are represented in different colors; with covariance determinants of 1.63 and 7.71 for the coherent blue and incoherent red mini-clusters respectively. The associated centroids and tolerance ellipses are denoted as

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

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

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

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

black crosses and black solid lines, respectively. It is clear that even incoherent subdivided sub-clusters, with a covariance determinant less than or equal to the predefined threshold though, could be hazardous as non-convex sub-clusters with a high value of covariance determinant, as their tolerance ellipses could get out of the scope of the main cluster, and suck outliers in[8].

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 sub-cluster 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[9], three conditions for every subdivided sub-cluster are met: not to be singular, not having a covariance determinant less than or equal to δ_i, and to be coherent.

Fig. 3c illustrates a scenario in which, three smaller mini-clusters, are represented as K-means output in different colors; and centroids and tolerance ellipses shown as in Fig. 3b. The covariance determinant values are equal to 0.12, 1.02 and 0.10 for the green, red and blue subdivided sub-clusters respectively. As it is obvious, all subdivided sub-clusters are coherent and not singular, with much smaller determinants than the threshold, and much tighter spatial volumes as such. Ultimately, after attaining acceptable sub-clusters, it is time to update the temporary clustering model w.r.t. them, due to Algorithm 2. Algorithm 6 shows all the steps required to cluster data points retained in memory buffer.

3.2.1.5. *Trying to Assign Retained Objects in Buffer to Newly Created mini-clusters*

After checking on retained objects in buffer in the case of being capable of forming a new mini-cluster, it would be necessary to examine the remaining retained objects once more, w.r.t. Algorithm 5; whether or not they could be assigned to newly created mini-clusters, in the same cycle of membership checking and mini-cluster updating, like what was done in subsection 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 sub-clusters; 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 situation, in which, some objects, according to the density restrictions cannot be assigned to a cluster; in consideration of they lie in the accepted Mahalanobis radius of that cluster. Objects that have had the competence to form a cluster are shown

Algorithm 6: $\left[\Gamma, \mathfrak{R}\right] = \text{RetSetClust}(\mathfrak{R}, \Gamma, \vec{\delta}, \varepsilon_{\mathcal{S}}, \psi_{\mathcal{S}}, \mathbb{C}_{\varepsilon}, \mathbb{C}_{\psi}, \Lambda)$

```
Input  : $\mathfrak{R}$ - Retained set; $\Gamma$ - Current array of mini-clusters
         information; $\vec{\delta}$ - Covariance determinant threshold; $\varepsilon_{\mathcal{S}}$ -
         Sampling Epsilon; $\psi_{\mathcal{S}}$ - Sampling MinPts; $\mathbb{C}_{\varepsilon}$ -
         Sampling Epsilon coefficient; $\mathbb{C}_{\psi}$ - Sampling MinPts
         coefficient; $\Lambda$ - PC share of total variance
Output : $\Gamma$ - Updated temporary clustering model; $\mathfrak{R}$ - Modified
         retained set

Apply DBSCAN algorithm to cluster $\mathfrak{R}$ w.r.t. the two parameters
  $\mathbb{C}_{\varepsilon} \cdot \varepsilon_{\mathcal{S}}$ and $\mathbb{C}_{\psi} \cdot \psi_{\mathcal{S}}$ for Epsilon and MinPts respectively.
  Consider the result of such clustering as $\left\{\mathfrak{R}_1, \cdots, \mathfrak{R}_{\mathbb{K}'}\right\} \cup \mathfrak{R}_{\zeta}$
/* Adding the information of the newly discovered mini-clusters to the
   temporary clustering model                                          */
foreach $\mathfrak{R}_i, 1 \leq i \leq \mathbb{K}'$ do
    if $SingCheck\left(\Sigma_{\mathfrak{R}_i}\right) \equiv 1$ then        /* Singularity check */
        $\mathfrak{R}_{\zeta} \leftarrow \mathfrak{R}_{\zeta} \cup \mathfrak{R}_i$
        continue
    end
    $b \leftarrow \text{argmin}_{h \in \{1, \cdots, \mathcal{T}\}} MD(\mu_{\mathfrak{R}_i}, X_h)$
    if $det_{\Sigma_{\mathfrak{R}_i}} > \vec{\delta}(b)$ then        /* Irregular mini-cluster */
        Apply K-means with the number of clusters
          $2 \leq \mathbb{K}'' \leq \lfloor \left|\mathfrak{R}_i\right| / (p+1) \rfloor$ on $\mathfrak{R}_i$. Find the minimum
          value for $\mathbb{K}''$ as by which, for every subdivided
          sub-cluster $\mathfrak{R}_{i,j}, 1 \leq j \leq \mathbb{K}''$, we have
          $SingCheck\left(\Sigma_{\mathfrak{R}_{i,j}}\right) \equiv 0, CohrCheck\left(\mathfrak{R}_{i,j}\right) \equiv 1$
          and $det_{\Sigma_{\mathfrak{R}_{i,j}}} \leq \vec{\delta}(b)$
        if such $\mathbb{K}''$ is not found then
            $\mathfrak{R}_{\zeta} \leftarrow \mathfrak{R}_{\zeta} \cup \mathfrak{R}_i$
            continue
        end
        $[\Gamma] = \text{MiniClustMake}(\Gamma, \left\{\mathfrak{R}_{i,1}, \cdots, \mathfrak{R}_{i,\mathbb{K}''}\right\}, \Lambda)$
        Remove $\left\{\mathfrak{R}_{i,1}, \cdots, \mathfrak{R}_{i,\mathbb{K}''}\right\}$ from RAM buffer
    else                                                   /* Regular mini-cluster */
        $[\Gamma] = \text{MiniClustMake}(\Gamma, \mathfrak{R}_i, \Lambda)$
        Remove $\mathfrak{R}_i$ from RAM buffer
    end
end
/* Setting unresolved points as retained set                           */
$\mathfrak{R} \leftarrow \mathfrak{R}_{\zeta}$
```

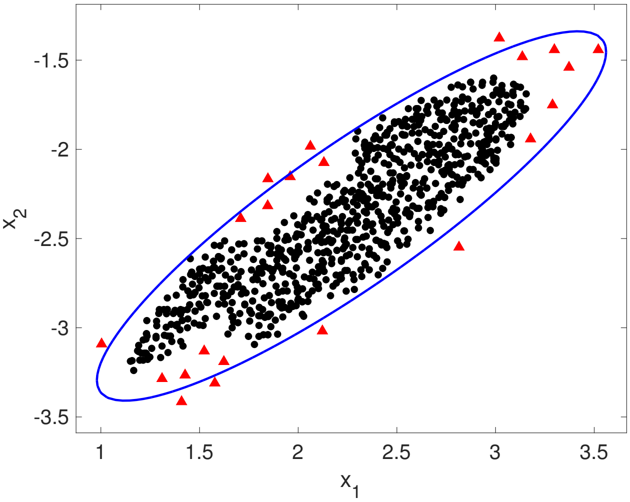
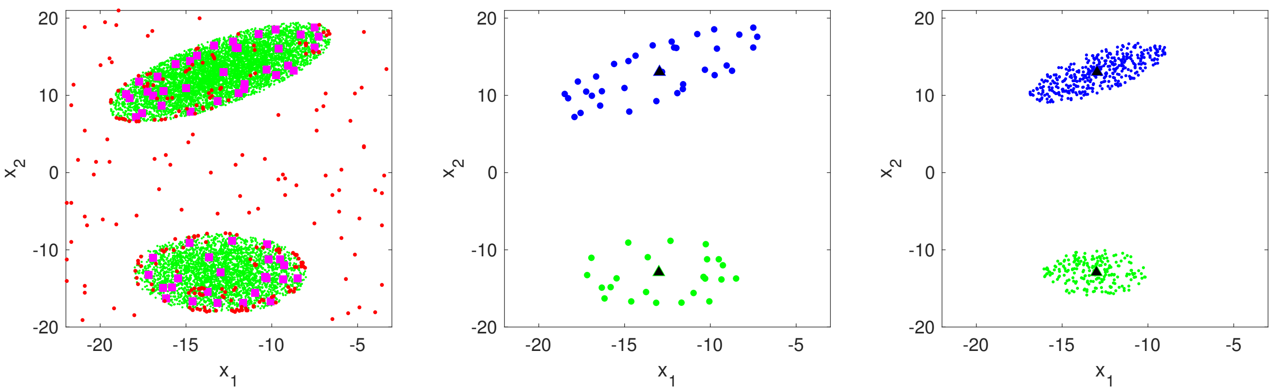


Figure 4: Trying to assign retained objects in buffer to newly created sub-clusters

[8] However, the divided mini-cluster could be significantly smaller in size and determinant, though because of lack of coherency, some PC variances could be very larger than others. Therefore, tolerance ellipses will be more stretched in those PCs, and thus harmful.

[9] Here, the upper bound for $\mathbb{K}$ is $\lfloor \left|\mathfrak{R}_i\right| / (p+1) \rfloor$, to avoid singularity problem for every subdivided sub-cluster of retained objects. $\left|\mathfrak{R}_i\right|$ stands for the cardinality of the ith mini-cluster of retained points.



(a) After Processing the Last Chunk

(b) Temporary Means and Final Means

(c) Regenerated Objects and Final Means

Figure 5: Proposed method appearance at the last steps of the scalable clustering

with black solid circles; and those which are not a part of the cluster, but reside in its accepted Mahalanobis radius, which is represented by a blue solid line, are denoted as red triangles.

Thus, if retained objects could belong to a newly created sub-cluster, the corresponding information of that sub-cluster will be modified w.r.t. Algorithm 4; and if not, such data will be still retained in buffer for further process. Although if it was the last chunk which was processed, all these retained objects will 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 which have not found the competence of forming a mini-cluster or being assigned to one, due to applied restrictions. However, at last, all of these true and untrue anomalies which are maintained in 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 the scalable clustering, it is time to construct the final clustering model or the same approximate structure of $\mathcal{T}$ original clusters, w.r.t. the temporary clustering model, with $\mathcal{L}$ mini-clusters. Hence, we follow Algorithm 7, as at the first step, K-means algorithm is carried out to cluster the centroids of temporary clusters[10].

After clustering the temporary means, we shall merge the information of the associated mini-clusters in each of such clusters to obtain 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, in the case of containing more than one temporary cluster, we utilize the sizes and the centroids of the associated mini-clusters, to obtain the final mean.

```
Algorithm 7: [$\mathcal{F}$] = FinalClustBuild($\Gamma, \eta, \beta$)
Input  : $\Gamma$ - Current array of mini-clusters information; $\eta$ -
         Random sampling rate; $\beta$ - Pruning threshold
Output : $\mathcal{F}$ - Final clustering model

Consider the mean vectors of the whole temporary clusters
  $X_j$'s, $1 \leq j \leq \mathcal{L}$, as a matrix $[\mathcal{M}]_{\mathcal{L} \times p}$, and apply K-means with
  $\mathbb{K} = \mathcal{T}$ to cluster them. Assume the result of such clustering as
  $\{\mathcal{M}_1, \cdots, \mathcal{M}_\mathcal{T}\}$
$\mathcal{F} \leftarrow \Phi$
foreach $\mathcal{M}_i, 1 \leq i \leq \mathcal{T}$ do
    if $|\mathcal{M}_i| \equiv 1$ then                       /* Isolated mini-cluster */
        Use the same mean location and covariance structure of
          the isolated sub-cluster, as for those of the final cluster
    else                                                  /* Group of mini-clusters */
        /* Calculating the final mean location                                   */
        $\mu_f \leftarrow \frac{\sum_{j:\, X_j \in \mathcal{M}_i} [\Gamma_j\{6\} \cdot \Gamma_j\{1\}]}{\sum_{j:\, X_j \in \mathcal{M}_i} \Gamma_j\{6\}}$
        /* Calculating the final covariance structure                            */
        $\mathcal{U} \leftarrow \Phi$
        foreach $X_j \in \mathcal{M}_i$ do
            Regenerate $\eta \cdot \Gamma_j\{6\}$ number of points, with
              Gaussian distribution, due 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
            Add these regenerated points to $\mathcal{U}$
        end
        Calculate the Mahalanobis distance of the points in $\mathcal{U}$
          based upon the sample mean $\mu_\mathcal{U}$, and the sample
          covariance matrix $\Sigma_\mathcal{U}$
        Prune $\mathcal{U}$ by discarding those points with Mahalanobis
          distance more than $\beta \cdot \sqrt{p}$ and recalculate $\Sigma_\mathcal{U}$
        $\Sigma_f \leftarrow \Sigma_\mathcal{U}$
        Remove $\mathcal{U}$ from buffer
        /* Adding information to the final clustering model                      */
        $\mathcal{F}\{1, i\} \leftarrow \mu_f$
        $\mathcal{F}\{2, i\} \leftarrow \Sigma_f$
    end
end
```

For acquiring the final covariance matrix, for each of the associated mini-clusters 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 mini-cluster equal to the product of

[10] Since, the original clusters are convex and besides, mini-clusters are developed among their space, thus, using K-means here is reasonable. Therefore, there is no need to utilize a density-based clustering method, for the case of non-convex clusters.

the sampling rate (which was used at the "Sampling" stage) and the cardinality of that mini-cluster, as $\eta \cdot \left|X_j\right|$. And this is necessary for saving free space in memory, while regenerating the approximate structure of an original cluster. We consider all regenerated objects of all sub-clusters belonging to a final cluster, as an 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 more than $\beta \cdot \sqrt{p}$, 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 which consists 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 the scalable clustering and before building the final clustering model. The green dots are normal objects belonged to a mini-cluster. The red dots are temporary outliers, and the 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 pruned regenerated data points for every final cluster beside 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 each object, a score of outlierness, w.r.t. the final clustering model obtained out of the scalable clustering.

3.3. Scoring

At this phase, w.r.t. the final clustering model which was 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 [2] 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"[11].

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 memory after loading each chunk, and the application of PCA to each mini-cluster 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 lie in the accepted Mahalanobis neighborhood of the current temporary clusters and are being assigned to them (and this will escalate over time by the increasing number of sub-clusters, which are being created during each memory process); and also by utilizing an indexing structure for k-NN queries, the time complexity of 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. Applying PCA on mini-clusters is $O\left(min\left(p^3, n_{\mathfrak{C}}^3\right)\right)$ [28], as p stands for the dimensionality of the input dataset. But according to our strong assumption that $p < n_{\mathfrak{C}}$, thus, applying PCA will be $O\left(p^3\right)$. Hence, the two first phases of the algorithm will totally take $O\left(max\left(\kappa n_{\mathfrak{C}} \log\left(n_{\mathfrak{C}}\right), p^3\right)\right)$. The last phase of the algorithm, w.r.t. this concern that only consists of calculating the Mahalanobis distance of any of the total n objects in the input dataset to $\mathcal{T}$ final clusters, is $O(n\mathcal{T})$; and regarding that $\mathcal{T} \ll n$; hence, the time complexity of this phase will be $O(n)$. The overall time complexity is thus at most $O\left(max\left(\kappa n_{\mathfrak{C}} \log\left(n_{\mathfrak{C}}\right), p^3\right) + n\right)$. However, it is evident that both p and $n_{\mathfrak{C}}$ values are negligible w.r.t. n, and therefore, we can state that the time complexity of our algorithm is linear.

Analysis of the algorithm space complexity is twofold. First, we consider stored information in memory for the clustering models; and second, the amount of space required for processing the resident data in RAM. With respect to 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 linear space complexity of these methods; hence, the overall space complexity will be $O\left(n_{\mathfrak{C}} + \mathcal{L}p^2\right)$.

4. Experiments

In this section, we conduct a suite of experiments aimed to analyze the accuracy, the robustness and the scalability of the proposed method. All the experiments were executed on a laptop having a 2.5 GHz Intel Core i5 processor and 6 GB of memory. The code was implemented using MATLAB 9 and, for the sake of reproducibility, it is published on GitHub[12].

To test the accuracy, we compare our method with some state-of-the-art density-based methods, namely LOF and

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

[12] https://github.com/sana33/SDCOR

LoOP; and a fast K-means variant optimized for clustering large-scale data, named X-means [39]; and two state-of-the-art distance-based methods, namely S_p and ORCA, on some synthetic and real-life datasets. In addition, experiments were conducted on some synthetic datasets in order to test how the final accuracy varies when the number of outliers is increased. Finally, the scalability of the proposed algorithm and the effect of random sampling rate on the final detection accuracy were tested.

4.1. Accuracy and Stability Analysis

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

4.1.1. Real Datasets

Some public and large-scale real benchmark datasets, taken from UCI [15], and preprocessed and labeled by ODDS [40], are used in our experiments. They are representative of different domains in science and humanities. Table 2 shows the characteristics of all test datasets, namely the numbers of objects (#n), attributes (#p) and outliers (#o). In addition, for the outliers in each dataset, their share of total objects in the corresponding dataset is reported in percentage terms. In all of our experiments, the Area Under the Curve (AUC) (curve of detection rate and false alarm rate) [12, 23] is used to evaluate the detection performance of compared algorithms. The AUC and runtime results[13] of different methods are summarized in Table 3 and Table 4, respectively. The **bold-faced** AUC and runtime indicate the best method for a particular dataset.

Table 2
Properties of the Datasets Used in the Experiments

	Dataset	#n	#p	#o (‰)
Real Datasets	Shuttle	49,097	9	3,511 (7.15%)
	Smtp (KDDCUP99)	95,156	3	30 (0.03%)
	ForestCover	286,048	10	2,747 (0.96%)
	Http (KDDCUP99)	567,498	3	2,211 (0.38%)
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%)

For competing density-based methods, the parameters are set as suggested, i.e. $MinPtsLB = 10$, $MinPtsUB = 50$ in LOF and $k = 30$, $\lambda = 3$ in LoOP. For X-means, the minimum and maximum number of clusters are set to 1 and 15 respectively. Maximum number of iterations and number of times to split a cluster are set to 50 and 6 respectively as well. In ORCA, the parameter k determines the number of nearest neighbors, and increasing it also increases the runtime. We use ORCA's default setting of $k = 5$ in our experiments. The parameter N determines how many anomalies are about to be reported. If N is small, ORCA increases the running cut-off rapidly and thus, more searches will be pruned off, which will result in a much faster runtime. Hence, as the true number of anomalies is not supposed to be known in the algorithm, we set $N = \frac{n}{8}$, as a reasonable value, where n stands for the cardinality of the input data. For S_p, the sample size s is set to the default value as suggested by the author, equal to 20 in our experiments. Each dataset is divided into 10 chunks, and for the algorithms that have random elements, including S_p and the proposed method SDCOR, their results are reported in the format of $\mu \pm \sigma$, over 40 independent runs, as μ and σ stand for the mean value and the standard deviation of AUC values.

Moreover, as long as we are trying to not let outliers participate actively in the process of forming and updating mini-clusters during the 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 $\vec{\delta}$, the boundary on the volume of mini-clusters at the time of creation; and the membership threshold α, which is useful to restrain the volume of mini-clusters, while they are incrementally growing over time.

Note that the sampling rate η 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[14]. The same problem concerns α, as a too low value could bring to the problem that the number of sub-clusters which are being created during the scalable clustering will become too large, which leads to a much higher computational load. In addition, a too high value for α brings the risk of outliers getting joined to the normal sub-clusters, and this will escalate over time, which increases the "False Negative" rate. Here, in all experiments, α is set to 2.

As for the real datasets, Table 3 shows that in terms of AUC, SDCOR is more accurate than all the other competing methods, except for the $Smtp$ dataset, which LoOP has achieved the best result, though with a negligible difference. Moreover, it is obvious that the attained results by the proposed approach are almost the same as the optimal ones, and also, the average line indicates that SDCOR performs overall much better than all the other methods. More importantly, SDCOR is effective on the largest dataset $Http$. In addition, by considering the standard deviations of AUC values for S_p and SDCOR, it is apparent that SDCOR is much more stable than S_p, as the mean value of the standard deviations

[13] For the proposed method, as we know the true structural characteristics of all the real and synthetic data, we compute the accurate anomaly score, based on Mahalanobis distance criterion, for each object and report the following optimal AUC next to the attained result by SDCOR. Moreover, we do not take into account the required time to find the optimal parameters of DBSCAN, as a part of total runtime. Furthermore, for X-means, as it assumes the input data free of noise, thus after obtaining the final clustering outcome, the Euclidean distance of each object to the closest centroid is assigned to it as an outlier score, and hence, the following AUC could be calculated.

[14] Hence, one can state that there is a straight relationship between the random sampling rate, and the true 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 pre-knowledge about real and synthetic data, we have set η to 0.5%.

Table 3
AUC Results for SDCOR and its Competitors on Real and Synthetic Datasets

	Dataset	LOF	LoOP	X-means	ORCA	S_p	SDCOR (Optimal)
Real Datasets	Shuttle	0.602	0.553	0.841	0.606	0.876±0.093	**0.967**±0.010 (0.994)
	Smtp (KDDCUP99)	0.874	**0.900**	0.110	0.860	0.858±0.029	0.780±0.006 (0.815)
	ForestCover	0.598	0.550	0.503	0.743	0.552±0.132	**0.934**±0.007 (0.950)
	Http (KDDCUP99)	0.871	0.862	0.001	0.459	0.948±0.211	**0.995**±0.002 (0.999)
	real data results average	0.736	0.716	0.364	0.667	0.808±0.116	**0.919**±0.006 (0.939)
Synth. Datasets	Data1	0.992	0.926	0.639	1.000	1.000±0.000	1.000±0.000 (1.000)
	Data2	0.997	0.959	0.692	1.000	1.000±0.000	1.000±0.000 (1.000)
	Data3	0.993	0.945	0.316	1.000	1.000±0.000	1.000±0.000 (1.000)
	Data4	0.995	0.937	0.358	1.000	1.000±0.000	1.000±0.000 (1.000)
	synth. data results average	0.994	0.942	0.501	1.000	1.000±0.000	1.000±0.000 (1.000)

Table 4
Execution Time (Secs) Results for SDCOR and its Competitors on Real and Synthetic Datasets

	Dataset	LOF	LoOP	X-means	ORCA	S_p	SDCOR
Real Datasets	Shuttle	303.920	354.100	7.312	80.367	**0.031**	0.841
	Smtp (KDDCUP99)	1,206.390	1,055.630	38.545	142.526	**0.075**	1.176
	ForestCover	9,005.150	11,907.550	21.779	3,618.686	**0.269**	2.583
	Http (KDDCUP99)	57,600.452	60,231.241	792.655	5,570.813	**0.449**	4.646
	real data results average	17,028.978	18,387.130	215.073	2,353.098	**0.206**	2.311
Synth. Datasets	Data1	79,211.854	81,893.782	461.071	18,838.446	**0.778**	9.540
	Data2	345,632.412	348,243.431	1,640.049	101,361.263	**1.923**	26.640
	Data3	604,768.975	609,776.649	3,525.552	224,559.345	**3.251**	55.770
	Data4	1,209,594.368	1,221,492.918	5,947.979	389,826.856	**4.175**	101.340
	synth. data results average	559,801.902	565,351.695	2,893.663	183,646.478	**2.532**	48.323

for SDCOR is 0.006, which is really dispensable comparing to that of S_p, equal to 0.116. And this is due to using one very small sample only by S_p, which causes the algorithm go through large variations on the final accuracy. For X-means, as it is not compatible with anomalies in the input data, thus it severely fails on separating outliers from the normal clusters, and it is clear from its average AUC for real datasets, which shows that outliers are totally misclassified.

Furthermore, Table 4 reveals that SDCOR preforms much better than other competing methods in terms of execution time, except for S_p, which is slightly faster than the proposed method. Although S_p is the fastest among the compared algorithms, as it was noted, its AUC suffers from large variations and it is lower than SDCOR. Moreover, w.r.t. the two state-of-the-art density-based methods, it is evident that there is a huge difference on consuming time between SDCOR and these methods, and it is due to the fact that in SDCOR, it is not required to compute the pairwise distances of the total objects in a dataset, differently from LOF and LoOP.

As stated earlier at the beginning of this paper, the strong assumption of SDCOR is on the structure of the existing clusters in the input dataset, which should have Gaussian distribution. In practice though, w.r.t. [43], quite a lot of real world data are Gaussian distributed — thanks to the Central Limit Theorem. Furthermore, w.r.t. [29, 21, 6], even when original variables are not Normal, employing properties of PCs for detecting outliers is possible and the corresponding results will be reliable. Since, given that PCs are linear functions of p random variables, an appeal to the Central Limit Theorem may justify approximate Normality for the PCs, even when the original variables are not Normal. Following this issue, it is possible to set up more formal tests for outliers based on PCs, assuming that the PCs are normally distributed. Moreover, the use of Mahalanobis distance criterion for outlier detection is only viable for convex-shaped clusters. Otherwise, outliers could be assigned to an irregular (density-based) cluster under masking effect and thus, will be misclassified.

4.1.2. Synthetic Datasets

Experiments on synthetic datasets are conducted in an ideal setting, since these datasets are following the strong assumptions of our algorithm on the structure of existing clusters, which should be Gaussians. Moreover, the generated outliers are usually more distinctive than those in the real data, and the outliers "truth" can be used to verify whether an outlier algorithm is capable of finding them. The experiments carried out on four artificial datasets are reported at the bottom of Table 3, along with their execution times in Table 4. Each dataset consists of 6 Gaussian clusters, and outliers take up 1 percent of its volume.

In more detail, for each dataset having p dimensions, we build a Gaussian cluster with a mean vector, so that it is quite far away from the other means, 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. to 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 the objects outside of 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 iff they fall in the accepted Mahalanobis distance interval of the cluster, e.g. $\left[4\sqrt{p}, 10\sqrt{p}\right]$. Due to the fact that the volume of the hypercube 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.

The results in Table 4 verify that synthetic datasets are in general too easy for SDCOR, and also for ORCA and S_p, as they always achieve perfect results, since, in such datasets, normal objects are totally in very dense areas and outliers reside in very sparse zones and far enough away from the normal clusters. However, X-means still fails on distinguishing anomalies from normal clusters, and LOF and LoOP are attaining near-perfect results. Although ORCA and S_p are rivaling the proposed method in terms of accuracy on artificial datasets, regarding the time consumption results in Table 4, ORCA's runtime is remarkably more than SDCOR and besides, for S_p, which is performing slightly quicker than our method w.r.t. other methods, such datasets are not seriously challenging like the real ones. The execution times of LOF and LoOP are totally huge though, as it took almost two weeks for the largest synthetic dataset to obtain results for each of these state-of-the-art density-based methods.

4.2. 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 the last subsection. Hence, the percentage of outliers in a dataset is increased from 50 to 150 percent with a step length of 10 percent. The numbers of normal objects and attributes for any of these datasets are 20,000 and 2 respectively. Also, the intrinsic manifold of normal objects in all these datasets is the same, which consists of 4 Gaussian clusters.

Fig. 6a reveals that SDCOR is totally noise tolerant, as by increasing the percentage of outliers, always perfect AUC results are achieved out of our method. ORCA performs favorably well, though with increase in the outliers ratio, it loses its stability. S_p attains fairly good detection results, but it can not stand high amounts of noises. However, X-means performs completely erratic in all noisy situations, and moreover, for LOF and LoOP, they are entirely misclassifying outliers in all complex conditions. The reason for this is that in SDCOR, the basis for forming mini-clusters, during the scalable clustering, is the fulfilment of DBSCAN requirements, which are obtained out of "Sampling" stage. As all of the normal objects follow a Gaussian distribution and outliers are injected using a continuous uniform distribution, hence, the local density of normal points is much higher than that of outliers. Therefore, the acquired optimal parameters for density-based clustering are obtained proportional to the dense regions containing only sampled inliers. For this reason, in each memory process, the probability of a sub-cluster being formed by outliers is very less than that of normal objects, and thus, our approach based on DBSCAN, obtains very good results.

Fig. 6b shows DBSCAN result on the sampled data of the test dataset with 150 percent of injected outliers. Four discovered Gaussian clusters and noises are represented with dots in different colors, and red empty circles, respectively. As it is evident, even at this highly noisy situation, outliers cannot satisfy DBSCAN constraints on forming a mini-cluster.

4.3. Scalability

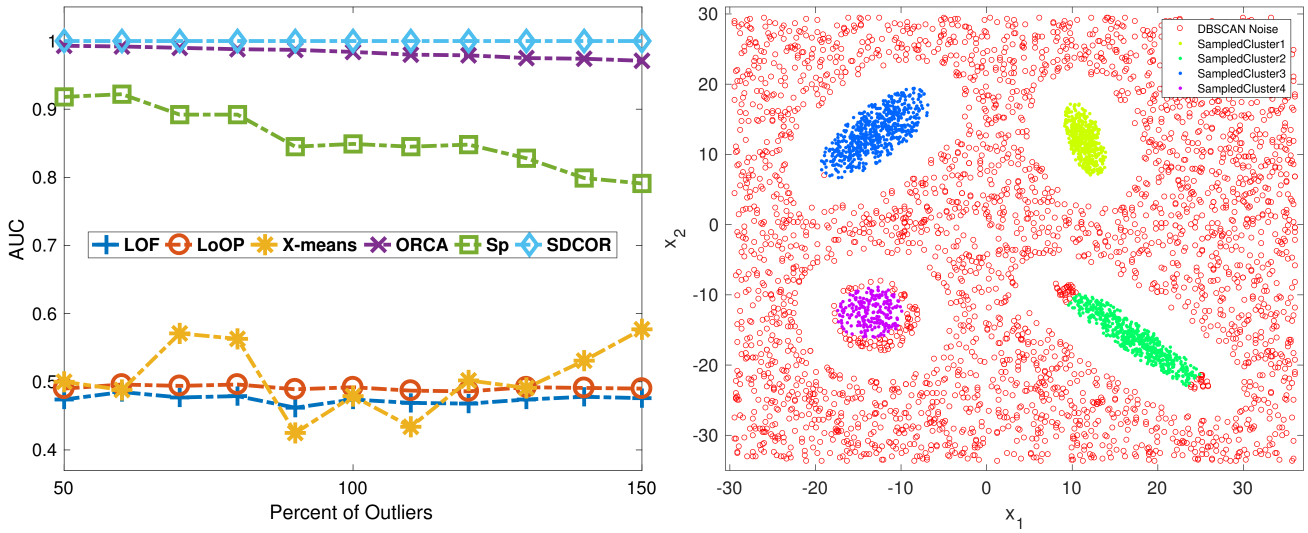
To assess the scalability of the proposed method, we measure the time consumption with increasing number of objects. First, a synthetic dataset with 200,000 normal objects having 10 attributes, containing 4 Gaussian clusters is generated. Then, we conduct random sampling with sampling rates of 10 to 100 percent with step length of 10 percent, and for each resulting dataset, we inject 200 outliers into it. In other words, we want to analyze the execution time using datasets which have very similar basic characteristics, namely the location (centroid) and the shape (covariance matrix) of each cluster. Moreover, in all experiments, parameters are set as suggested in the previous subsections, and also, in all cases, perfect detection results are achieved.

Fig. 7 demonstrates that the run time of SDCOR is close to a linear function of the number of objects; and this is a confirmation of the result of the analysis conducted in subsection 3.4 on the algorithm complexity. In particular, for the dataset with a minimum size equal to 20,200, the run time is 1.19 seconds. But when the number of objects reaches the maximum value equal to 200,200 (about ten times the minimum value), the processing time increases by about 2.2 times, to only 2.59 seconds. Therefore, SDCOR has a low constant in its runtime complexity, and hence, is scalable.

4.4. The Effect of the Sampling Rate

Here, we examine the variation of covariance determinant of sampled data of a unique cluster per various random sampling rates. Therefore, first of all, we create an arbitrary Gaussian cluster with 10,000 objects and 2 attributes. Then, we start sampling with the sampling rate of 0.5 percent and proceed to 100 percent with the step length of 0.5 percent, and then, for each of these resulting sampled clusters, we calculate the corresponding covariance determinant of the cluster.

As Fig. 8a shows, with increase in random sampling rate, the corresponding covariance determinant is approaching that



(a) AUC Results per Percentage of Injected Outliers

(b) DBSCAN Result with Optimal Parameters on the High Noisy Sampled Data

Figure 6: Robustness results

of the original cluster. However, as it is evident, even covariance determinants associated with very low sampling rates are quite close to that of the main cluster. In 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. 8b illustrates such situation in which, objects belonging to the original cluster and those belonging to the sampled one are shown with blue dots and red squares respectively. Moreover, tolerance ellipses of the main and the sampled clusters are shown in red and black respectively.

5. 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 incrementally updated by loading successive chunks of data into memory. Ultimately, after processing the whole chunks, the final clustering model was acquired, which w.r.t. that and conducting another scan of the entire dataset, each object was given an outlying score equal to its local Mahalanobis distance.

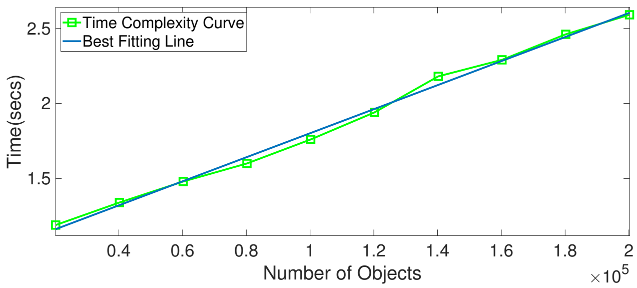
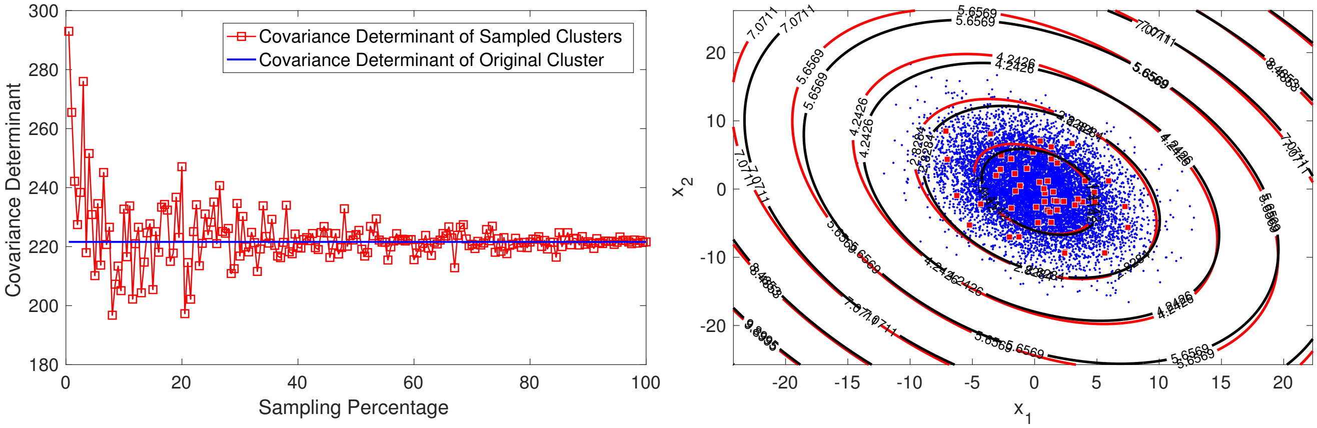


Figure 7: Scalability test

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 be resident in memory; and also, 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 confirm that the algorithm has a linear time complexity with a low constant and that, even with a very low rate of random sampling, it is still able to satisfactorily approximate the shape of the real clusters. For the future work, we would like to enhance our proposed approach to be able to cope also with density-based non-convex clusters.

Acknowledgment

Special thanks to Dr. Victoria J. Hodge (University of York, UK), for her clever advice on the title of this paper, and Dr. Mohammad-Mehdi Ebadzadeh (Tehran Polytechnic, Iran), for his valuable comments about some of the algorithms. Also, we appreciate Dr. Ali-Mohammad Saghiri (Tehran Polytechnic, Iran) for his generous review of the paper before submission.



(a) Covariance Determinant Variation per Sampling Rate

(b) Tolerance Ellipses for both Original and Sampled Clusters

Figure 8: Covariance determinant and tolerance ellipses variation per random sampling rate

References

[1] Achlioptas, D., 2001. Database-friendly random projections, in: Proceedings of the twentieth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems, ACM. pp. 274–281.

[2] Aggarwal, C.C., 2015a. Data mining: the textbook. Springer.

[3] Aggarwal, C.C., 2015b. Outlier analysis, in: Data mining, Springer. pp. 237–263.

[4] Agyemang, M., Barker, K., Alhajj, R., 2006. A comprehensive survey of numeric and symbolic outlier mining techniques. Intelligent Data Analysis 10, 521–538.

[5] Ayyıldız, E., Purutçuoglu, V., Wit, E., 2012. A short note on resolving singularity problems in covariance matrices. International Journal of Statistics and Probability 1, 113–118.

[6] Barnett, V., Lewis, T., 1974. Outliers in statistical data. Wiley.

[7] Bay, S.D., Schwabacher, M., 2003. Mining distance-based outliers in near linear time with randomization and a simple pruning rule, in: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 29–38.

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

[9] Breiman, L., 2001. Random forests. Machine learning 45, 5–32.

[10] Breunig, M.M., Kriegel, H.P., Ng, R.T., Sander, J., 2000. Lof: identifying density-based local outliers, in: ACM sigmod record, ACM. pp. 93–104.

[11] Cabras, S., Morales, J., 2007. Extreme value analysis within a parametric outlier detection framework. Applied Stochastic Models in Business and Industry 23, 157–164.

[12] Chandola, V., Banerjee, A., Kumar, V., 2009. Anomaly detection: A survey. ACM computing surveys (CSUR) 41, 15.

[13] Dasgupta, S., Gupta, A., 1999. An elementary proof of the johnson-lindenstrauss lemma. International Computer Science Institute, Technical Report 22, 1–5.

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

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

[16] Duan, L., Xu, L., Liu, Y., Lee, J., 2009. Cluster-based outlier detection. Annals of Operations Research 168, 151–168.

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

[18] Filzmoser, P., Maronna, R., Werner, M., 2008. Outlier identification in high dimensions. Computational Statistics & Data Analysis 52, 1694–1711.

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

[20] Han, J., Pei, J., Kamber, M., 2011. Data mining: concepts and techniques. Elsevier.

[21] Hawkins, D.M., 1980. Identification of outliers. volume 11. Springer.

[22] He, Z., Xu, X., Deng, S., 2003. Discovering cluster-based local outliers. Pattern Recognition Letters 24, 1641–1650.

[23] Hodge, V., Austin, J., 2004. A survey of outlier detection methodologies. Artificial intelligence review 22, 85–126.

[24] Hubert, M., Debruyne, M., 2010. Minimum covariance determinant. Wiley interdisciplinary reviews: Computational statistics 2, 36–43.

[25] Hubert, M., Rousseeuw, P.J., Vanden Branden, K., 2005. Robpca: a new approach to robust principal component analysis. Technometrics 47, 64–79.

[26] Jin, W., Tung, A.K., Han, J., Wang, W., 2006. Ranking outliers using symmetric neighborhood relationship, in: Pacific-Asia Conference on Knowledge Discovery and Data Mining, Springer. pp. 577–593.

[27] Johnson, W.B., Lindenstrauss, J., 1984. Extensions of lipschitz mappings into a hilbert space. Contemporary mathematics 26, 1.

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

[29] Jolliffe, I., 2011. Principal component analysis. Springer.

[30] Kennedy, J., 2010. Particle swarm optimization. Encyclopedia of machine learning , 760–766.

[31] Knox, E.M., Ng, R.T., 1998. Algorithms for mining distancebased outliers in large datasets, in: Proceedings of the international conference on very large data bases, Citeseer. pp. 392–403.

[32] Kriegel, H.P., Kröger, P., Schubert, E., Zimek, A., 2009. Loop: local outlier probabilities, in: Proceedings of the 18th ACM conference on Information and knowledge management, ACM. pp. 1649–1652.

[33] Leskovec, J., Rajaraman, A., Ullman, J.D., 2014. Mining of massive datasets. Cambridge university press.

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

[35] Liu, F.T., Ting, K.M., Zhou, Z.H., 2012. Isolation-based anomaly detection. ACM Transactions on Knowledge Discovery from Data (TKDD) 6, 1–39.

[36] Mahalanobis, P.C., 1936. On the generalized distance in statistics, National Institute of Science of India.

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

[38] Pearson, K., 1901. Liii. on lines and planes of closest fit to systems of points in space. The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science 2, 559–572.

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

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

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

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

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

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

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

[46] Teng, S.H., et al., 2016. Scalable algorithms for data and network analysis. Foundations and Trends® in Theoretical Computer Science 12, 1–274.

[47] Wu, S., Wang, S., 2011. Information-theoretic outlier detection for large-scale categorical data. IEEE transactions on knowledge and data engineering 25, 589–602.

[48] Yin, J., Ho, Q., Xing, E.P., 2013. A scalable approach to probabilistic latent space inference of large-scale networks, in: Advances in neural information processing systems, pp. 422–430.

[49] Zhang, K., Hutter, M., Jin, H., 2009. A new local distance-based outlier detection approach for scattered real-world data, in: Pacific-Asia Conference on Knowledge Discovery and Data Mining, Springer. pp. 813–822.

[50] Zimek, A., Schubert, E., Kriegel, H.P., 2012. A survey on unsupervised outlier detection in high-dimensional numerical data. Statistical Analysis and Data Mining: The ASA Data Science Journal 5, 363–387.