2018 IEEE International Conference on Data Mining Workshops (ICDMW)

Outlier Detection Based on Low Density Models

Félix Iglesias Vázquez
Inst. of Telecom.
TU Wien
Vienna, Austria
Email: felix.iglesias@nt.tuwien.ac.at

Tanja Zseby
Inst. of Telecom.
TU Wien
Vienna, Austria
Email: tanja.zseby@tuwien.ac.at

Arthur Zimek
Dep. of Math. and Comp. Science
University of Southern Denmark (SDU)
Odense, Denmark
Email: zimek@imada.sdu.dk

***Abstract*—Most outlier detection algorithms are based on lazy learning or imply quadratic complexity. Both characteristics make them unsuitable for big data and stream data applications and preclude their applicability in systems that must operate autonomously. In this paper we propose a new algorithm—called SDO (Sparse Data Observers)—to estimate outlierness based on low density models of data. SDO is an eager learner; therefore, computational costs in application phases are severely reduced. We perform tests with a wide variation of synthetic datasets as well as the main datasets published in the literature for anomaly detection testing. Results show that SDO satisfactorily competes with the best ranked outlier detection alternatives. The good detection performance coupled with a low complexity makes SDO highly flexible and adaptable to stand-alone frameworks that must detect outliers fast with accuracy rates equivalent to lazy learning algorithms.**

I. Introduction

Discovering outliers in data, either for removal (i.e., data cleaning) or for a careful analysis of anomalies, has become an almost indispensable step in any application that works with data. Although from different perspectives, methods commonly establish outlier scores by measuring distances or densities in the multidimensional data space. Among the diverse techniques proposed in the literature, outlier detection based on the k-nearest neighbors (k-nn) [1] or the Local Outlier Factor (LOF) [2], despite many variations and evolutions [3], are still the ones that stand out due to their reliability [4].

Fundamentally, the challenges that outlier detection algorithms face are:

1) *Time complexity*. Algorithms perform costly object-to-object comparisons to estimate distances and densities [5, 6]. If not comparing extensively, they alleviate calculations by defining volumes, ratios, or neighborhoods that involve additional parameters.
2) *Uncertainty of non-extreme objects*. Without an external reference or benchmark, outlierness is defined solely based on the context of the data under test. This fact induces uncertainties since *normality* (for the application context) is not necessarily defined in terms of density or distance, or it is not homogeneously defined. Whereas extreme values are clear, intermediate values might be sometimes difficult to interpret [7].
3) *Parameterization*. Performances often depends on parameters that might be difficult to estimate, are not intuitive, or demand readjustment if data changes [4].
4) *Embedding in decision-making processes*. The most accurate outlier detection methods are usually based on lazy learners, meaning that high computational efforts are required when new objects are evaluated (i.e., in application phases). Moreover, it also involves large space to store old data. These are two relevant drawbacks that hinder the embedding of outlier detection in systems that are expected to work autonomously and make fast decisions based on continuously collected data.

In this paper we present the SDO (Sparse Data Observers) algorithm for outlier detection and ranking. SDO builds a low density data model formed by *observers*. An observer is a data object placed within the data mass and ideally equidistant to other observers within the same cluster. The outlierness of a data object is evaluated based on the distance to its x-closest observers (a visual example of the SDO ranking for a 2-dimensional dataset is shown in Figure 1).

SDO obtains very good performances in experiments with synthetic cluster-like datasets generated with highly variable structures, parameters, and dimensions. Moreover, SDO shows equivalent performances to the methods analyzed by Campos et al. [4] when tested with the same datasets and indices as in their benchmark repository. This repository [4] is probably the most recent (2016) and exhaustive comparison among algorithms for outlier ranking and detection.

SDO addresses all the challenges listed above:

1) *Time complexity*. SDO is lightweight and shows linear

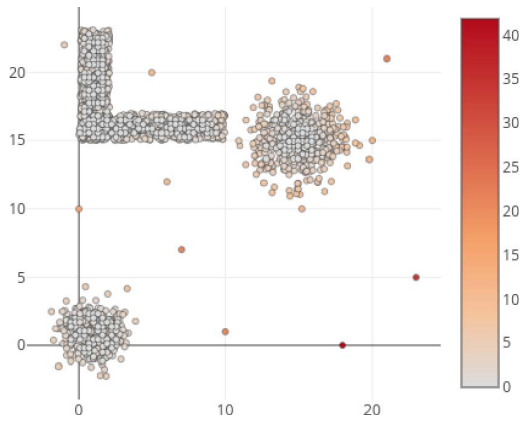


Fig. 1. SDO scoring in a 2-dimensional example.

2375-9259/18/$31.00 ©2018 IEEE
DOI 10.1109/ICDMW.2018.00140

complexity for large datasets. Data objects are compared to the observers, being the number of observers—in absence of pre-knowledge (i.e., the worst case)—estimated based on statistical sampling calculations. Therefore, being m and k respectively the number of data objects and observers, k converges asymptotically as a function of m, becoming a constant: $O(mk)|_{m\to\infty} = O(m)$.

2) *Uncertainty of non-extreme objects*. SDO models data by redrawing the whole data picture with only a few points. Up to a certain degree, the model keeps density differences if they are representative of the dataset normality. This trait partially removes uncertainty in the evaluation of intermediate objects. Moreover, in cases where a ground truth is available, SDO accuracy highly improves since models are more representative.
3) *Parameterization*. SDO depends on three parameters. We propose some rules of thumb for a default parameterization. Experiments demonstrate rules of thumb robustness, yet pre-knowledge and deep data exploration are always recommended for the fine tuning.
4) *Embedding in decision-making processes*. SDO is an eager learner that creates a data model, so new data is compared with a model instead of with a dataset formed by previous samples. This fact makes SDO fast and suitable for embedding in systems, monitors, and decision-making phases, nicely adapting to the requirements of machine learning, stream data, and big data. If evolving scenarios are expected, SDO models must be periodically retrained.

In the remainder, we revisit popular outlier detection techniques in Sect. II. The SDO methodology is depicted and discussed in Sect. III. Diverse validation experiments are conducted in Sect. IV. Finally, conclusions are drawn in Sect. V. The reader can resort to Table I for clarification about the symbols used throughout this paper.

TABLE I
NOTATION

symbol	description
S	set of training or application objects
s_i	any element of S
O	set of observers
o_j	any element of O
n	number of dimensions
m	number of elements in S
k	number of elements in O (parameter)
k_{act}	number of *active* observers
D	matrix with distances between S and O
I	matrix with indices of the x-closest observers to every s_i in S
P	observation array
p_j	observations of the o_j observer
y_i	outlierness score of s_i
x	active-observers per object (parameter)
q	idle-observers threshold (parameter)
Z	Z-value, related to CI
ϵ	error
σ	standard deviation
$M(.)$	average function
$d(.)$	distance function
$Q_\rho(.)$	quantile for the cumulative probability ρ

II. OUTLIER DETECTION ALGORITHMS

Outlier detection is a main field of study within the scope of data mining and machine learning. It is usually tackled by statisticians, computer scientists, and engineers, although outliers commonly affect almost any application that trusts in data to describe or to monitor a phenomenon or a system. Extensive explanation and discussion about outlier analysis and detection are provided in diverse surveys [8, 9, 10, 11, 3].

A well-known definition by Hawkins states that *"an outlier is an observation which deviates so much from the other observations as to arouse suspicions that it was generated by a different mechanism"* [12]. It is important to notice that *outlier* and *outlierness* are ambiguous and debatable concepts. Outlierness is a flexible peculiarity intimately bound to a *context* and, therefore, defined *in comparison with other data* and, in practice, *in connection with an application*. In the related literature we find different classes of outliers (e.g., global, local, subspace, extreme values, noise, clusters) as well as redefinitions of the main concepts [13, 14, 15].

In any case, given a set of objects that populate a multidimensional space, it is possible to establish object-outlierness scores in an unsupervised manner solely based on the inherent, a priori unknown structure of the global picture drawn by such objects. Under these considerations, diverse techniques have been proposed in the literature. Existing methods lay in three main branches: (a) statistical methods (b) methods based on similarity and neighborhoods, and (c) clustering-based methods.

A. Statistical methods

Statistical methods use probability or distribution models to fit data [16]. Singular objects located far from model shapes are identified as outliers. Statistical methods can be parametric (when the distribution model is assumed and parameters are adjusted to maximize the fitness) or non-parametric (when no distribution is assumed). Among statistical methods, we find techniques supported by plain statistics [17], distribution fitting [18], regression techniques [19], histograms [20], or methods based on probability densities [21].

Outlier detection based on statistics work well for low dimensional spaces (typically one dimension). They are descriptive, i.e., they explain the structure of the data and provide a mathematical summarization to understand it based on the assumption of a distributional model. Moreover, unless some pre-knowledge ensures that the *mass* of the data will approximately fit a given distribution, a statistical method will be inaccurate or probably will not work. This issue is particularly pertinent to parametric models.

B. Neighborhood-based methods

In this group we find methods that use distance between objects or subspaces, definitions of neighborhood for objects

and subspaces, and derived calculations of density. Such methods offer good outlierness evaluations, do not need to estimate data distributions and satisfactorily scale to multi-dimensional spaces. However, they show a strong dependency on parameters that define the meaning, limits, and reach of the applied concepts of *neighborhood* and *density*. The parameterization is not always intuitive and introduces subjectivity into the algorithmic interpretation of outlierness. Complexity is typically quadratic due to neighborhood queries. Establishing outlierness thresholds might be hard in scenarios with density variations. The *curse of dimensionality* affects distance-based methods in high-dimensional spaces [22] since sparsity increases and distances between objects tend to equalize. Sophisticated density-based approaches improve accuracy and robustness, but at the expense of computational costs.

DB(k,λ)-Outlier is one of the first distance-based methods for outlier detection [23]. Ramaswamy et al. [1] use k-nn algorithms to measure object isolation in the dataset space. The method Grid-ODF [24] performs space partitioning and considers both, distance and neighboring density evaluations. Some methods take reverse neighborhood into account [25, 26].

With a deeper conception of object-density, an outstanding progress in the outlier analysis field was initiated by the LOF (Local Outlier Factor) method [2]. Some popular variants that try to enhance diverse aspects of LOF are: COF (Connectivity-based Outlier Factor) [27], LOCI (Local Outlier Correlation Integral) [28], INFLO (Influenced Outlierness) [29], LDF (Local Density Factor) [30], or LoOP (Local Outlier Probabilities) [31], among others. ABOD (Angle-Based Outlier Detection) [32] is based on angles vectors, does not need parameterization and is robust against the *curse of dimensionality*. A density-based method that uses reference points (RP) to alleviate algorithm calculations was presented by Pei et al. [33].

C. Clustering-based methods

In clustering-based methods, algorithms first find clusters and subsequently discover and rank outliers based on the clustering solution. This seems advantageous as clustering and outlier detection are complementary tasks and often required together. On the other hand, the main disadvantage is the dependence on the clustering phase itself. Clustering algorithms must be carefully parameterized as they can easily fail and obtain misleading abstractions of the data; moreover, many times data do not follow cluster-like shapes or structures that are comprehensible for the applied algorithm. Thus, if the algorithm does not deal properly with classic clustering problems (e.g., density differences, local minima, correct number of clusters), the outlier evaluation is not only downgraded but can be completely wrong. Moreover, outliers tend to distort clustering solutions, resulting in a vicious circle.

Some clustering algorithms are more robust to outliers than others, or even incorporate outlier detection per se. For instance, CLARANS [34], CURE [35], BIRCH [36], or DBSCAN [37] are known to be robust in the presence of outliers. The ROF (Resolution-based Outlier Factor) method [38] ranks outliers by means of performing a density-based clustering and predefining some distance-based thresholds. Jaing et al. [39] propose a *two-phase clustering* (modified k-means and minimum spanning tree) for outlier detection. CBOD [40] is also a two-phase method with time complexity close to linear.

Some works play with the ambiguous definition of outlierness and address specific kinds of singularity. For example, Duan et al. [15] claim that clusters can also be outliers and propose LDBSCAN for their detection. He et al. [13] redefine "outlierness" and present the CBLOF (Cluster-Based Local Outlier Factor) index and an algorithm (FindCBLOF) for the discovery. Similarly, a pattern-based definition of outlierness is given by He et al. [14], also a score called FPOF (Frequent Pattern Outlier Factor) and the respective algorithm (FindFPOF). The popular k-means clustering has also a version that embeds outlier detection, called k-means-- [41]. GLOSH (Global-Local Outlier Scores from Hierarchies) operates on hierarchical density-estimates as used in HDBSCAN* clustering [42]. Finally, [43] uses subspace clustering to define patterns against which sample outlierness is evaluated.

III. The SDO Method

A. The Algorithm

As a modeling algorithm, SDO consists of two separate phases: training and application (a scheme with an example is shown in Figure 2). Whenever performing outlier analysis for a single static dataset, data in training and application are the same. In stream data applications, models must be periodically retrained. In this paper, we explore the case of detecting anomalies in a single, static dataset.

Training:

1) *Initialize observers*: Given S as a set of m data objects, O is a subset of k objects from S taken at random ($k \ll m$). Objects in O are called *observers*. k is ideally selected to guarantee that O keeps the statistical properties of the data mass distribution in S. Methods to select observers are discussed later in Section III-B.
2) *Observe samples*: A distance matrix D is created by measuring the distance between observers and data objects. Therefore, $D_{i,j} = d(s_i, o_j)$, where $d(.)$ is the Euclidean distance, $i \in \{1, ..., m\}$, and $j \in \{1, ..., k\}$. D is reduced to the observation matrix I, which stores the identifiers of the x-closest observers to every object in S (x is an SDO design parameter).
3) *Remove idle observers*: During the training phase, each observer o_j observes p_j objects from S; i.e., p_j is the number of appearances of o_j in I. Therefore, P is an array of length k that contains the number of observations of every observer. Whenever an observer did not observe at least q objects, it becomes an *idle* observer and is removed (q is an SDO design parameter). This removal prevents the selection of outliers as observers. In other words, it ensures that observers coexist in medium-to-high density zones or regions that are representative for the data. Therefore, the model is constructed only by *representative* data points. After removing idle observers, the remaining k_{act} observers are called *active* observers.

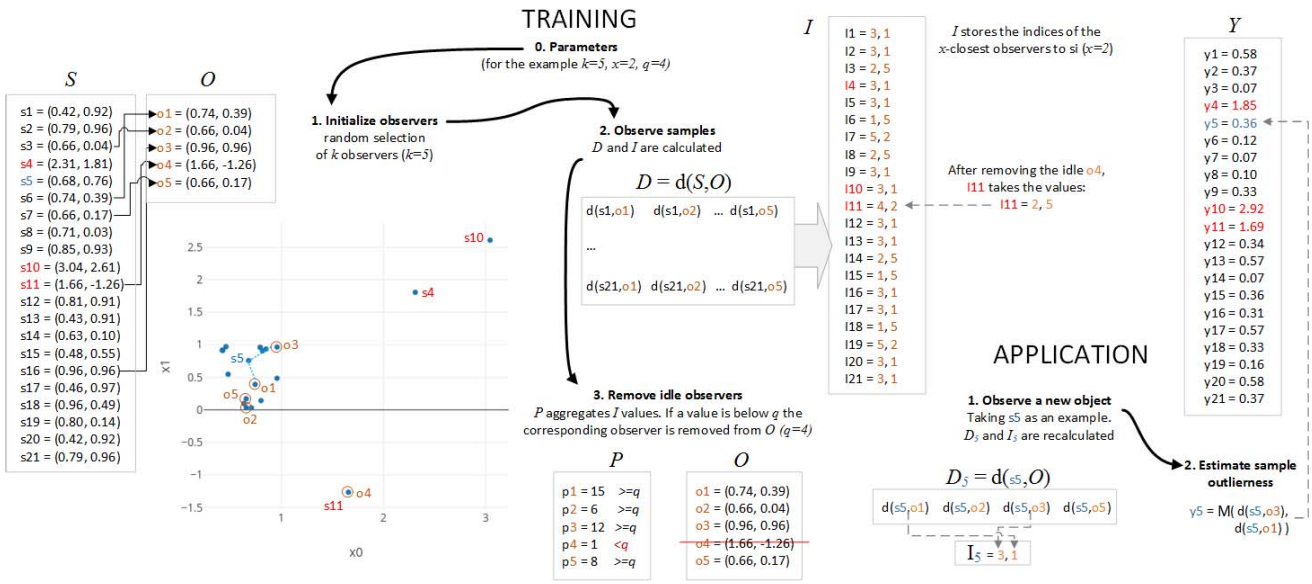


Fig. 2. Scheme and example of SDO operation.

Application:

1) *Observe a new object*: During the application phase, objects in S or new objects from a consistent dataset (we equally refer to them as S for the sake of clarity) are evaluated by the model formed by O. Therefore, given an object s_i, a new distance array D_i of length k_{act} is calculated, being $d(s_i, o_j)$ the value of the j element in D_i. As before, the identifiers of the x closest observers to s_i are stored in the observation array I_i.
2) *Calculate sample outlierness*: The outlier score is estimated as the average distance between the object s_i and its x closest observers. Therefore, the score y_i is:

$$y_i = M[d(s_i, o_{I_{i_1}}), \ldots, d(s_i, o_{I_{i_x}})]$$

with some average function $M(.)$. The average function used in the experiments of this paper is the median.

B. Selection of observers

Given a suitable k—how to establish k is discussed in Section III-C—the trivial approach to select observers is assuming that a sample of k random objects of S is already representative from a statistical perspective. This straightforward approach works satisfactorily in most cases. Nevertheless, SDO obtains optimal performances when observers are selected in a way that they form an n-dimensional lattice or skeleton that, in addition to matching the data shape, shows vertices uniformly distributed in zones with equivalent density. Building such an ideal low density model can be challenging and computationally intensive. A satisfactory solution is—before selecting observers—to simplify the dataset by means of an histogram-based discretization (i.e., similar to performing lossy compression), but it requires additional parameters to correctly fit the histogram binning.

C. Parameter discussion

SDO depends on three main parameters:

- k: total number of observers.
  k is a parameter closely related to statistical sampling. Whenever facing datasets that are unknown, k can be adjusted based on the population size. However, pre-analysis and pre-knowledge of the data are usually possible and prevent the overestimation of k. We recommend using pre-knowledge to adjust k whenever possible.
  In absence of pre-knowledge, k can be set by using statistical parameter estimation. By applying PCA (Principal Component Analysis) on the dataset, the resulting most important parameter (i.e., the first principal component) will be a representative random variable as it contains the maximum variance. Later, k can be estimated as the sample size in a case of finite populations for matching the mean of the PCA most important parameter. An assumed error ϵ and CI (Confidence Intervals, represented by a Z-value) are required.[1] For example:

$$k = \frac{m^2\sigma^2}{(m-1)\epsilon^2 + Z^2 * \sigma^2} \quad (1)$$

- x: number of active observers per object.
  x determines how many observers are consulted to evaluate the location of every single object of S. x is a robust parameter that commonly takes values within 3 and 10 without considerably affecting the score. However, it must be defined considering k and scenario peculiarities. Low k with high x leads to oversimplifications of the model and the outlierness estimations; high k with low x reduces stability and tends to overfitting.

[1]In our experiments we assume a CI=95% (Z=1.96) and an error $\epsilon = 0.1\sigma$.

Algorithm 1 SDO prototype (MATLAB-Octave)

```
q=q_0; %SDO parameter
x=x_0; %SDO parameter
k=k_0; %SDO parameter

[m,n]=size(S); %'S' is the input dataset

% random sampling of observers
index=randperm(m);
O = S(index(1:k),:);

% TRAINING
% observation matrix calculation
for i=1:m
    D_tr=dist(S(i,:),O');
    [v indices]=sort(D_tr);
    I_tr(i,:)=indices(1:x);
end
% observations array calculation
for j=1:k
    Aj=I_tr==j;
    P(j)=sum(sum(Aj));
end

% idle-observers removal
O(P<q,:)=[];

% APPLICATION
for i=1:m
    D_app_i=dist(S(i,:),O');
    [v indices]=sort(D_app_i);
    I_app_i=indices(1:x);
    Bi=O(I_app_i,:);
    y(i)=median(dist(S(i,:),Bi'));
end
```

- q: idle-observer threshold, or minimum number of observations expected for an *active* observer.
  This parameter is intended to clean the model and remove potential outliers and irrelevant observers from O. By default, empirical tests with $q \approx Q_\rho(P)$ with $\rho = 0.3$ leads to satisfactory results, where $Q_\rho(.)$ is a quantile function that calculates the cut-off value for the cumulative probability ρ in the interval [0,1]. Note that this step reduces the number of observers and keeps the ones that performed more observations (i.e., the most active ones). In any case, the final value of q must be based on design criteria, since q is sensitive to density variations. A high q can lead to oversimplified scenarios where little clusters are ignored, which could be appropriate when outliers occur in clumps [50]. Low q can induce outliers among observers, adding noise to the model and blurring final scores. Thus q can be seen as a fine/coarse adjustment for the granularity of the model.

D. Code prototype

Algorithm 1 offers a simplified version of the SDO MATLAB implementation. Complete MATLAB scripts are available online [51][2].

[2]In [51] an additional *fast* version of SDO is provided, in which `for` loops are vectorized. In Listing 1 we show the *normal* version for the sake of clarity and intuitive understanding.

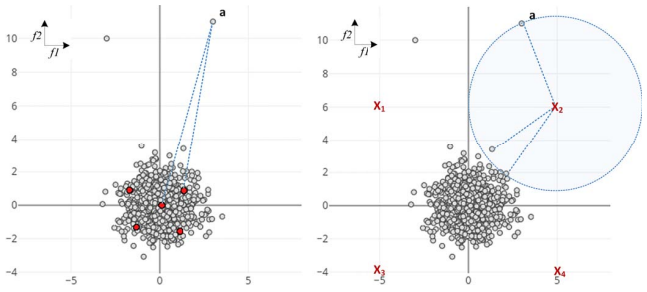


Fig. 3. *Left plot* (SDO): In the example, outlierness of point a is calculated based on the distance to their 2-closest observers (red dots). *Right plot* (RP-method): Outlierness of point a is calculated based on the minimum density measured with respect to every reference point, i.e., $\{x_1, x_2, x_3, x_4\}$. In the example, the density of a for x_i is established based on the 2-nearest neighbours of a in the one-dimensional set formed by the ordered distances of every data object to x_i.

E. Comparison with similar approaches

From the reviewed literature, the approach that shows most similarities to SDO belongs to the *neighborhood-based* group and is proposed by Pei et al. [33] (henceforth referred as RP-method). Our *observers* might be deemed analogous to the *reference points* that the RP-method uses. However, in contrast, reference points do not try to build a model of the data, instead, they are space locations taken as anchors to measure relative k-neighborhoods and densities. After checking all reference points, the outlierness of a specific data object is evaluated based on the reference point that computes the minimum density for such specific object. Therefore, it is irrelevant if reference points are within the data mass or in low-density zones; in fact, authors recommend to establish them as vertices of a grid that evenly splits the complete data space. This approach could be seen as a variant of *approximating* outlier scores by vantage-points (other methods use, e.g., locality-sensitive hashing [44], random projections [45], or space-filling curves [46]). Figure 3 compares SDO and the RP-method.

A noteworthy, determinant difference between these two algorithms is that the RP-method is based on lazy learning, i.e., a new object is compared with all previous data objects; instead, SDO uses eager learning, i.e., a new object is compared with a model, which represents all previous data.

A subspace outlier detection algorithm similar to SDO has been recently published by Sathe and Aggarwal [47]. Points in common with SDO are: the algorithm is straightforward, shows linear times, is based on random sampling and is easily scalable for stream data analysis since it also creates models. The main difference lies in the fact that, instead of *observers* or *reference points*, [47] uses ensemble-centric randomized subspace histograms. Every data point is therefore evaluated based on the joined scores of different histograms that work on random data samples and subspaces. The combined final score seems to be robust and generalize properly.

F. Supervised vs. unsupervised methods

Although SDO is characterized as eager by building a model during a training phase, it should be noted that it nevertheless

is unsupervised. There is no additional information beyond the data taken into account during the training phase. In this respect it can be compared to clustering-based methods, which also learn some structure in an unsupervised way and evaluate potential outliers given the cluster model. However, SDO does not derive an explicit cluster structure and thus does not suffer from the typical disadvantages of cluster-based methods.

There are also supervised methods for outlier detection (aka one-class classifiers) such as the Support Vector Data Description [48] and related methods. However, a recent experimental study did not find that supervised methods work statistically significantly better than unsupervised methods despite having the advantage of labeled training data [49]. In this paper we therefore focus on unsupervised methods for a competitive evaluation.

IV. Evaluation

For the evaluation of the SDO algorithm different experiments have been conducted by using three different approaches:

- With cluster-like synthetic datasets created with a tool for the generation of datasets for clustering and outlier detection testing [52].
- With popular datasets for testing and comparison of outlier detection methods [4].
- We additionally compare runtime performances of SDO and LOF in order to show the advantages of eager learning vs. lazy learning with growing sample size. This experiment is carried out in a controlled scenario with synthetic data [52].

We have used the same performance indices as Campos et al. [4]. We refer the reader to their work for clarification and discussion about the indices, which are:

- *P@n*, precision at the top n ranks;
- *adjP@n*, adjusted P@n;
- *AP*, average precision;
- *adjAP*, adjusted AP;
- *MaxF1*, Maximum F1 score [53];
- *AdjMF1*, adjusted MaxF1 score;
- *ROC-AUC*, area under the ROC curve.

Adjusted indices follow the procedure of Hubert and Arabie [54]. The parameter n for *P@n* and *adjP@n* has been chosen as the number of labeled outliers in the corresponding test dataset, following Campos et al. [4].

Unless specifically mentioned in the experiment, for SDO we have used the proposed rules of thumb for parameter adjustment—k and q—introduced in Section III-C. For the remaining parameters we chose $x = 5$ and $\rho = 0.3$. For the benchmark data we show additionally the performance with parameter adjustment based on pre-knowledge.

A. Synthetic Datasets

The script that runs the tests can be downloaded from [51]. Performance indices are listed in Table II. Results of SDO with synthetic data are excellent, even in spite of the variability in shapes, densities, percentage of outliers, number of clusters and number of features.

TABLE II
1000 outlier detection tests with synthetic data.

	P@n	adj P@n	AP	adj AP	Max F1	adj MF1	ROC AUC
mean	0.96	0.95	0.95	0.95	0.97	0.97	0.97
median	1.00	1.00	1.00	1.00	1.00	1.00	0.97

TABLE III
Paramaters adjusted by using pre-knowledge.

	k	q	x		k	q	x
ALOI	5000	9	2	Annthyr.	4758	6	5
Glass	92	1	1	Arrhyt.	16	225	6
Ionos.	46	9	3	Cardio.	311	79	8
KDD	2000	87	5	Heart.	12	56	7
Lympho.	26	0	3	Hepat.	8	27	4
PenDig.	5782	3	1	Intern.	848	5	5
Shuttle	282	3	1	Page.	1702	22	5
Wavef.	2337	6	4	Parkin.	52	2	3
WBC	95	12	4	Pima	339	30	9
WDBC	82	1	1	Spam	2255	1	4
WPBC	22	5	4	Stamps	72	9	1
				Wilt	3729	1	1

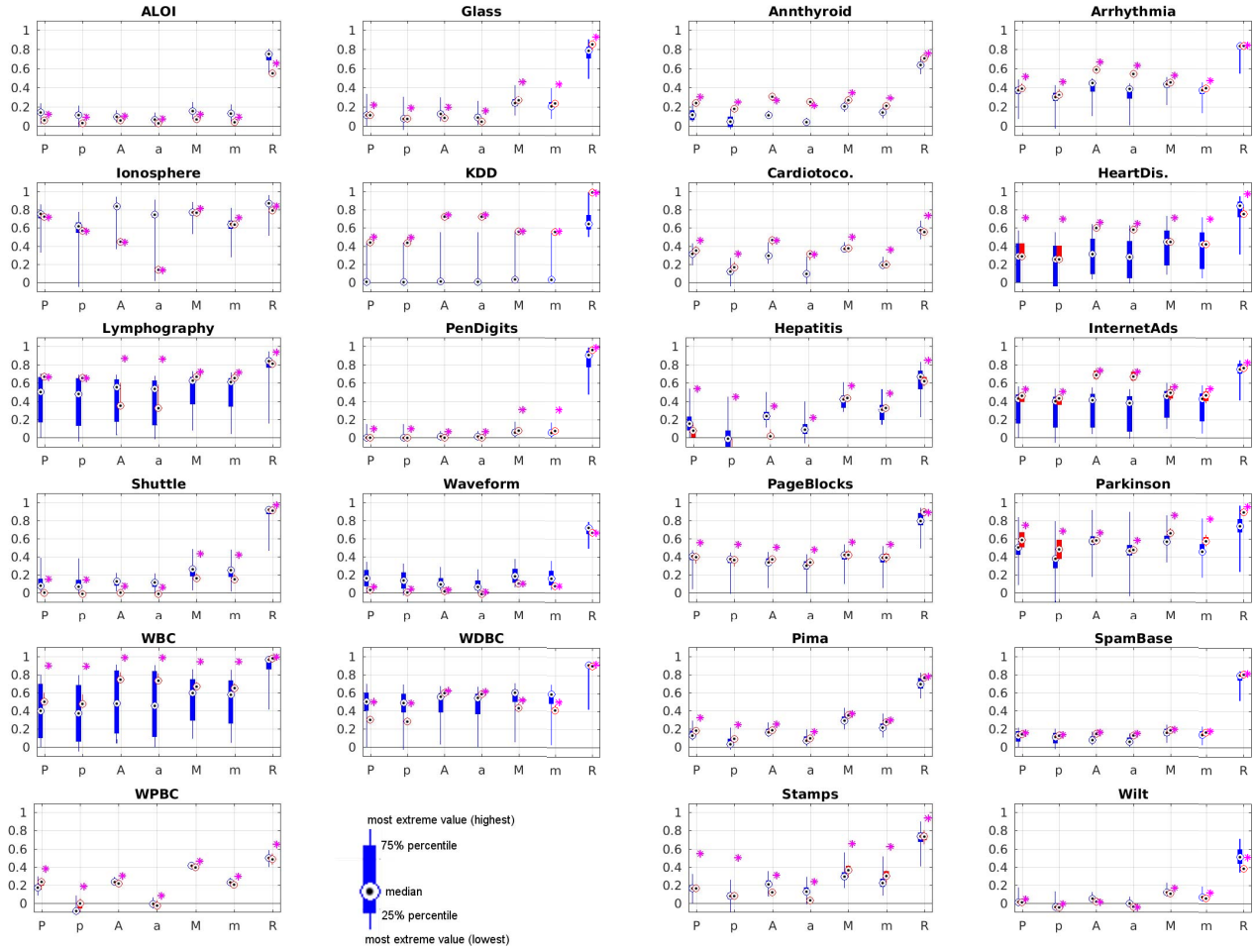
B. Benchmark Datasets

For this set of experiments we have randomly selected one dataset from each of the 23 different groups available [4], opting for the normalized versions without duplicates, as recommended by Campos et al. [4]. For each dataset, SDO has been run 10 times. We thus can compare the performance of SDO to the performance distribution of the 12 methods tested by Campos et al. [4] over up to 100 different parameters each. Figures 4(a) and 4(b) show the results, where *red* box plots correspond to SDO performances and *blue* box plots to the methods evaluated in [4], which were tested in the same conditions and are shown here for comparison. Experiments can be further consulted and downloaded for reproducibility from [51].

It should be noted that SDO parameters for these experiments as represented by the box plots have not been tuned using pre-knowledge, in spite of the fact that it is highly recommended and would improve performances in many cases. To show the effect of such tuning, *magenta* asterisks have been added to Figures 4(a) and 4(b), and correspond to SDO performances achieved when using pre-knowledge. The specific parameters for such cases are given in Table III.[3] In the figures the box plots for benchmark results condense the 12 algorithms, each tested by Campos et al. [4] over 100 different parameter settings. As a result, the top performance represented by the top values of the blue box plots can correspond to a different algorithm and parameter setting for each dataset and each performance measure. Therefore, it is reasonably fair to compare red and blue box plots medians and, on the other hand, magenta asterisks with top values of blue box plots.

Results in Figures 4(a) and 4(b) show that SDO performances—when adjusted with rules of thumb and parameters by default—vary depending on the specific dataset. In

[3]The *pre-knowledge-based adjustment* was simulated by taking dataset samples and tuning SDO parameters by means of Evolutionary Search algorithms, which used sample labels for evaluation.



(a) *"Literature"* datasets.

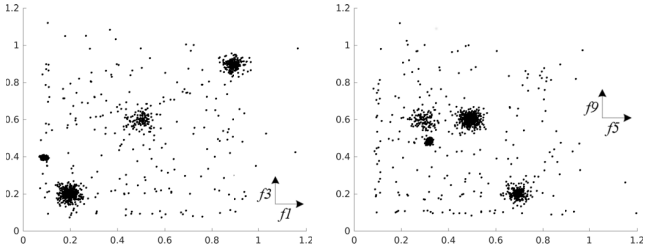
(b) *"Semantic"* datasets.

Fig. 4. Comparison of SDO performances with benchmark results [4]. 'P' stands for *P@n*, 'p' for *adjP@n*, 'A' for *AP*, 'a' for *adjAP*, 'M' for *MaxF1*, 'm' for *AdjMF1*, and 'R' for *ROC-AUC*. Blue box plots are for benchmark results [4]. Red box plots are for the SDO 10-run sets adjusted with rules of thumb. Magenta asterisks show performances achieved by SDO by using pre-knowledge. Box plots mark: medians (dots), 25% and 75% percentiles (thick lines), and high and low extreme values (thin lines).

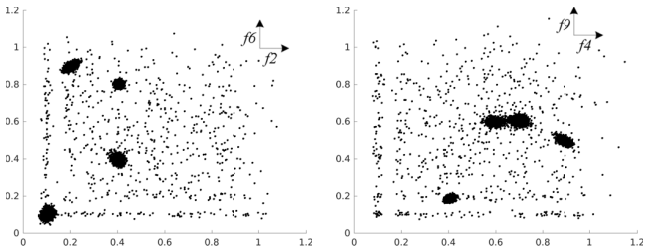
most cases SDO performs competitive with the 12 methods in various parameter settings, in several cases SDO outperforms all the other methods, in some cases SDO outperforms with a considerable margin. Only in a few cases, the average performance of SDO is below the average of the 12 methods in the benchmark. This behavior was expected and is consistent with the experiments in Campos et al. [4], where authors state that "none of the more recent methods tested offer any comprehensive improvement over those *classics*" (*classics* refers to LOF and k-nn). Note that SDO was not primarily intended to outperform LOF or k-nn algorithms (lazy learners) but to keep equivalent accuracies while implementing a fast, model-based approach (as eager learner).

It is worth mentioning that the datasets explored and provided by Campos et al. [4] are challenging scenarios with the following characteristics: (a) Outliers are not necessarily defined to be geometrically or topologically differentiated in the input space drawn by the selected features; instead, their outlier condition has been established based on some pre-knowledge from classification problems. (b) They do not represent large datasets, big data, or stream data cases. (c) They are used to test algorithms with no time constraint and without paying attention on computational resources.

The design goals of SDO slightly differ from these three conditions since SDO is mainly devised to address the problem of embedding outlier detection in autonomous frameworks that must explore large data volumes or continuous streams of data (e.g., anomaly detection in network traffic), which is hardly assumable by lazy approaches like LOF or k-nn. Therefore, assumed whenever models are used for classification, SDO might sacrifice accuracy to gain flexibility and speed. Additionally, the parameterization of SDO highly benefits from



(a) dataset 4: feature 1 vs. feature 3 (b) dataset 4: feature 5 vs. feature 9



(c) dataset 7: feature 2 vs. feature 6 (d) dataset 7: feature 4 vs. feature 9

Fig. 5. Scatter plots of different paired features for the 4th dataset with $m = 1622$ and the 7th dataset with $m = 9814$. Dimensions are randomly selected.

TABLE IV
SDO VS. LOF RUNTIME COMPARISON.

dataset	m	#out	SDO AUC	LOF AUC	SDO time	LOF time
1st	1175	155	0.93	0.75	0.08s	0.14s
2nd	1184	129	0.95	0.65	0.06s	0.08s
3rd	1297	149	0.94	0.72	0.06s	0.09s
4th	1622	219	0.93	0.57	0.07s	0.12s
5th	2198	101	0.98	0.78	0.09s	0.17s
6th	4129	148	0.98	0.76	0.16s	0.37s
7th	9814	711	0.96	0.55	0.37s	1.32s
8th	25028	2002	0.96	0.36	1.09s	6.46s
9th	61806	932	0.99	0.43	3.24s	27.70s
10th	173272	9517	0.97	0.18	10.41s	232.07s

domain knowledge as well as data visualization during training, which is expected and consistent with application best practices.

Another interesting detail that can be observed from the figures is that, in spite of being a non-deterministic method, SDO analysis with different random seeds tend to provide equivalent performances (short box plot "arms"). The random selection of observers becomes irrelevant as soon as the number of dataset samples is high enough (e.g., *KDD*, *PenDigits*, *ALOI*), whereas in datasets with few samples (e.g., *Lymphography*, *HeartDis.*, *Parkinson*) performances show more variability. This fact is coherent with the foundations of statistical sampling. As commented before, SDO is designed to face large volumes of data, stream scenarios or autonomous systems, i.e., situations that justify the use of models. In such cases, SDO tends to provide more deterministic results. In scenarios that manage few samples, the benefits of SDO are not that evident since more complex and resource demanding solutions can be applied instead with no significant prejudice and better performances.

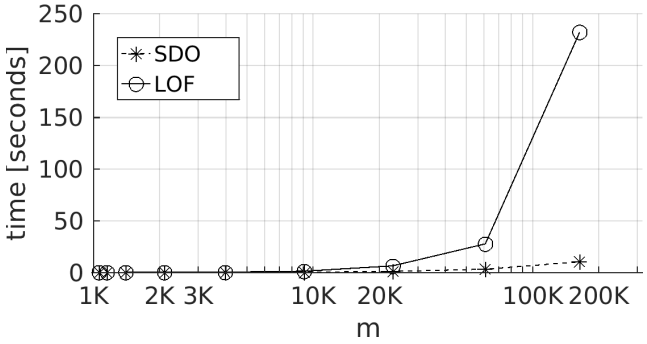


Fig. 6. SDO vs. LOF analysis on m: runtime performance.

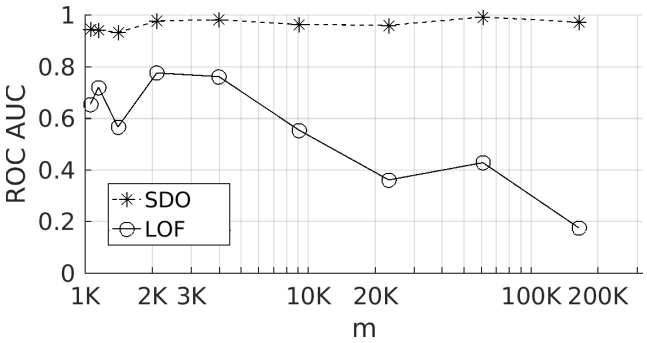


Fig. 7. SDO vs. LOF analysis on m: ROC AUC.

C. SDO vs. LOF runtime comparison

To consider the caveats in empirical evaluation of runtime behavior as discussed by Kriegel et al. [55], we compare an implementation of LOF [2] in the same environment as SDO (MATLAB), using the publicly available implementation from Yeh and Gui [56], and we draw conclusions from the scalability behavior of both methods rather than on absolute differences. However, the comparison proves positive for SDO in all aspects, confirming the theoretical expectations.

The script that runs this test can be downloaded from [51]. In the experiment, we used the synthetic dataset generator [52] to create 10 datasets. All datasets were 10-dimensional and contained 4 clusters and around 10% of outliers. The sensitive variable in the dataset generation process was the number of samples, which increased exponentially. LOF was set with *number of k-neighbours,* $k_{nn} = 15$, whereas SDO parameters where $k = 300$, $x = 5$, $q = Q_{0.3}(P)$. Figures 5(a), 5(b), 5(c) and 5(d) are shown to give an impression of the used datasets by displaying scatter plots of randomly paired features from the 4th dataset ($m = 1622$) and the 7th dataset ($m = 9814$).

Experiment outcomes are detailed in Table IV and visualized in Figures 6 and 7. In the conducted test SDO outperformed LOF from a classification perspective in addition to keeping considerably better runtime performances for larger datasets. The classification degradation suffered by LOF when the number of samples tend to increase has to do with the fact that the k_{nn} parameter in LOF depends on the total number of samples, i.e., the higher m, the higher k_{nn} is required for

achieving optimal classifications; however, the higher k_{nn}, the higher computational costs become too. This issue does not happen in the SDO case, whose parameters are not affected by the total number of samples, provided the number of observers is able to capture a representative low density model.

In addition, results in tables and figures clearly illustrate why classic algorithms not based on models become bottlenecks when the number of samples is growing large. The analysis of big data and stream data is unfeasible for lazy methods or methods with quadratic (or higher) complexity. SDO is based on principles of sampling theory, i.e., it assumes that the larger the dataset the smaller the quotient $\frac{model\_samples}{total\_samples}$ is. It is worth remarking here that, in stream data applications, and equivalently to other stream data solutions, SDO models must periodically be retrained to avoid model degradation.

V. Conclusions

This work has introduced and deeply described the SDO algorithm for outlier scoring and detection. SDO operates by creating a low-density model of the data during a training phase; therefore, it is suitable for integration into autonomous systems that must perform fast analysis and decision making in big data and stream data applications.

We have also presented rules of thumb for the parameterization of SDO in absence of pre-knowledge. Tests with synthetic cluster-like datasets as well as popular datasets classically used for testing outlier detection methods have validated SDO. In summary, SDO is a novel and highly competent outlier detection alternative specifically addressing big data challenges.

Acknowledgement

This research has been funded by the Vienna Science and Technology Fund (WWTF) through project ICT15-129, "BigDAMA".

References

[1] S. Ramaswamy, R. Rastogi, and K. Shim, "Efficient algorithms for mining outliers from large data sets," 2000, pp. 427–438.

[2] M. M. Breunig, H.-P. Kriegel, R. Ng, and J. Sander, "LOF: Identifying density-based local outliers," 2000, pp. 93–104.

[3] E. Schubert, A. Zimek, and H.-P. Kriegel, "Local outlier detection reconsidered: a generalized view on locality with applications to spatial, video, and network outlier detection," vol. 28, no. 1, pp. 190–237, 2014.

[4] G. O. Campos, A. Zimek, J. Sander, R. J. G. B. Campello, B. Micenková, E. Schubert, I. Assent, and M. E. Houle, "On the evaluation of unsupervised outlier detection: Measures, datasets, and an empirical study," vol. 30, pp. 891–927, 2016.

[5] G. H. Orair, C. Teixeira, Y. Wang, W. Meira Jr., and S. Parthasarathy, "Distance-based outlier detection: Consolidation and renewed bearing," vol. 3, no. 2, pp. 1469–1480, 2010.

[6] E. Kirner, E. Schubert, and A. Zimek, "Good and bad neighborhood approximations for outlier detection ensembles," 2017, pp. 173–187.

[7] H.-P. Kriegel, P. Kröger, E. Schubert, and A. Zimek, "Interpreting and unifying outlier scores," 2011, pp. 13–24.

[8] V. J. Hodge and J. Austin, "A survey of outlier detection methodologies," *Artificial Intelligence Review*, vol. 22, no. 2, pp. 85–126, 2004.

[9] A. S. Hadi, A. H. M. Rahmatullah Imon, and M. Werner, "Detection of outliers," vol. 1, no. 1, pp. 57–70, 2009.

[10] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Comput. Surv.*, vol. 41, no. 3, pp. 15:1–15:58, Jul. 2009.

[11] P. J. Rousseeuw and M. Hubert, "Robust statistics for outlier detection," vol. 1, no. 1, pp. 73–79, 2011.

[12] D. M. Hawkins, *Identification of outliers*. Chapman and Hall London ; New York, 1980.

[13] Z. He, X. Xu, and S. Deng, "Discovering cluster-based local outliers," *Pattern Recogn. Lett.*, vol. 24, no. 9-10, pp. 1641–1650, Jun. 2003.

[14] Z. He, X. Xu, Z. Huang, and S. Deng, "FP-outlier: Frequent pattern based outlier detection," *Computer Science and Information Systems/ComSIS*, vol. 2, no. 1, pp. 103–118, 2005.

[15] L. Duan, L. Xu, Y. Liu, and J. Lee, "Cluster-based outlier detection," *Annals of Operations Research*, vol. 168, no. 1, pp. 151–168, 2009.

[16] V. Barnett and T. Lewis, *Outliers in statistical data*, 2nd ed. John Wiley & Sons Ltd., 1978.

[17] C. Leys, C. Ley, O. Klein, P. Bernard, and L. Licata, "Detecting outliers: Do not use standard deviation around the mean, use absolute deviation around the median," *Journal of Experimental Social Psychology*, vol. 49, no. 4, pp. 764–766, Jul. 2013.

[18] G. Danuser and M. Stricker, "Parametric model fitting: from inlier characterization to outlier detection," vol. 20, no. 3, pp. 263–280, 1998.

[19] P. J. Rousseeuw and A. M. Leroy, *Robust Regression and Outlier Detection*. New York, NY, USA: John Wiley & Sons, Inc., 1987.

[20] M. Goldstein and A. Dengel, "Histogram-based outlier score (HBOS): A fast unsupervised anomaly detection algorithm," in *KI-2012: Poster and Demo Track*, 2012, pp. 59–63.

[21] D. M. J. Tax and R. P. W. Duin, "Outlier detection using classifier instability," in *Advances in Pattern Recognition: Joint IAPR International Workshops SSPR'98 and SPR'98 Sydney, Australia, August 11–13, 1998 Proceedings*, 1998, pp. 593–601.

[22] A. Zimek, E. Schubert, and H.-P. Kriegel, "A survey on unsupervised outlier detection in high-dimensional numerical data," vol. 5, no. 5, pp. 363–387, 2012.

[23] E. M. Knorr and R. T. Ng, "Algorithms for mining distance-based outliers in large datasets," 1998, pp. 392–403.

[24] W. Wang, J. Zhang, and H. Wang, "Grid-odf: Detecting outliers effectively and efficiently in large multi-dimensional databases," in *Proc. CIS*, 2005, pp. 765–770.
[25] V. Hautamäki, I. Kärkkäinen, and P. Fränti, "Outlier detection using k-nearest neighbor graph," 2004, pp. 430–433.
[26] M. Radovanović, A. Nanopoulos, and M. Ivanović, "Reverse nearest neighbors in unsupervised distance-based outlier detection," 2014.
[27] J. Tang, Z. Chen, A. W.-c. Fu, and D. W. Cheung, "Enhancing effectiveness of outlier detections for low density patterns," 2002, pp. 535–548.
[28] S. Papadimitriou, H. Kitagawa, P. B. Gibbons, and C. Faloutsos, "LOCI: fast outlier detection using the local correlation integral," 2003, pp. 315–326.
[29] W. Jin, A. K. H. Tung, J. Han, and W. Wang, "Ranking outliers using symmetric neighborhood relationship," 2006, pp. 577–593.
[30] L. J. Latecki, A. Lazarevic, and D. Pokrajac, "Outlier detection with kernel density functions," in *Proc. MLDM*, 2007, pp. 61–75.
[31] H.-P. Kriegel, P. Kröger, E. Schubert, and A. Zimek, "LoOP: local outlier probabilities," 2009, pp. 1649–1652.
[32] H.-P. Kriegel, M. Schubert, and A. Zimek, "Angle-based outlier detection in high-dimensional data," 2008, pp. 444–452.
[33] Y. Pei, O. Zaïane, and Y. Gao, "An efficient reference-based approach to outlier detection in large datasets," 2006, pp. 478–487.
[34] R. T. Ng and J. Han, "Clarans: A method for clustering objects for spatial data mining," *IEEE Trans. on Knowl. and Data Eng.*, vol. 14, no. 5, pp. 1003–1016, Sep. 2002.
[35] S. Guha, R. Rastogi, and K. Shim, "CURE: An efficient clustering algorithm for large databases," 1998, pp. 73–84.
[36] T. Zhang, R. Ramakrishnan, and M. Livny, "BIRCH: An efficient data clustering method for very large databases," 1996, pp. 103–114.
[37] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," 1996, pp. 226–231.
[38] H. Fan, O. R. Zaïane, A. Foss, and J. Wu, "A nonparametric outlier detection for effectively discovering top-n outliers from engineering data," 2006, pp. 557–566.
[39] M. F. Jaing, S. S. Tseng, and C. M. Su, "Two-phase clustering process for outliers detection," *Pattern Recogn. Lett.*, vol. 22, no. 6-7, pp. 691–700, May 2001.
[40] S. Jiang and Q. An, "Clustering-based outlier detection method," in *5th Int. Conf. on Fuzzy Systems and Knowledge Discovery*, 2008, pp. 429–433.
[41] S. Chawla and A. Gionis, "k-means--: A unified approach to clustering and outlier detection," 2013, pp. 189–197.
[42] R. J. G. B. Campello, D. Moulavi, A. Zimek, and J. Sander, "Hierarchical density estimates for data clustering, visualization, and outlier detection," vol. 10, no. 1, pp. 5:1–51, 2015.
[43] E. Müller, I. Assent, P. Iglesias, Y. Mülle, and K. Böhm, "Outlier ranking via subspace analysis in multiple views of the data," 2012, pp. 529–538.
[44] Y. Wang, S. Parthasarathy, and S. Tatikonda, "Locality sensitive outlier detection: A ranking driven approach," 2011, pp. 410–421.
[45] T. de Vries, S. Chawla, and M. E. Houle, "Density-preserving projections for large-scale local anomaly detection," vol. 32, no. 1, pp. 25–52, 2012.
[46] E. Schubert, A. Zimek, and H.-P. Kriegel, "Fast and scalable outlier detection with approximate nearest neighbor ensembles," 2015, pp. 19–36.
[47] S. Sathe and C. C. Aggarwal, "Subspace histograms for outlier detection in linear time," *Knowledge and Information Systems*, vol. 56, no. 3, pp. 691–715, Sep 2018.
[48] D. M. J. Tax and R. P. W. Duin, "Support vector data description," *Machine Learning*, vol. 54, no. 1, pp. 45–66, 2004.
[49] L. Swersky, H. O. Marques, J. Sander, R. J. G. B. Campello, and A. Zimek, "On the evaluation of outlier detection and one-class classification methods," 2016, pp. 1–10.
[50] H. O. Marques, R. J. G. B. Campello, A. Zimek, and J. Sander, "On the internal evaluation of unsupervised outlier detection," 2015, pp. 7:1–12.
[51] TU Wien CN Group, "Sparse Data Observers (SDO): Outlier Detection Based on Low Density Models," https://www.cn.tuwien.ac.at/data-analysis/outliers-sdo/, 2017, gitHub: https://github.com/CN-TU/SDO-MATLAB.
[52] ——, "MDCGen: Generator of Multidimensional Datasets for Clustering," https://www.cn.tuwien.ac.at/data-analysis/mdcgen/, 2017, gitHub: https://github.com/CN-TU/mdcgen-matlab.
[53] D. M. W. Powers, "Evaluation: From precision, recall and f-measure to roc., informedness, markedness & correlation," *Journal of Machine Learning Technologies*, vol. 2, no. 1, pp. 37–63, 2011.
[54] L. Hubert and P. Arabie, "Comparing partitions," *Journal of Classification*, vol. 2, no. 1, pp. 193–218, Dec 1985.
[55] H.-P. Kriegel, E. Schubert, and A. Zimek, "The (black) art of runtime evaluation: Are we comparing algorithms or implementations?" vol. 52, no. 2, pp. 341–378, 2017.
[56] Y.-R. Yeh and Z.-W. Gui, "Anomaly Detection Toolbox: Local Outlier Factor (MATLAB)," 2014, gitHub: https://github.com/dsmi-lab-ntust/AnomalyDetectionToolbox.