







SDOclust: Clustering with Sparse Data Observers

Félix Iglesias¹(✉) , Tanja Zseby¹ , Alexander Hartl¹ ,
and Arthur Zimek²(✉) 

¹ TU Wien, Gusshausstraße 25/E389, 1040 Vienna, Austria

{felix.iglesias,tanja.zseby,alexander.hartl}@tuwien.ac.at

² SDU, Campusvej 55, 5230 Odense, DK, Denmark

zimek@imada.sdu.dk

Abstract. Sparse Data Observers (SDO) is an unsupervised learning approach developed to cover the need for fast, highly interpretable and intuitively parameterizable anomaly detection. We present SDOclust, an extension that performs clustering while preserving the simplicity and applicability of the original approach. In a nutshell, SDOclust considers *observers* as graph nodes and applies local thresholding to divide the obtained graph into clusters; later on, observers' labels are propagated to data points following the *observation* principle. We tested SDOclust with multiple datasets for clustering evaluation by using no input parameters (default or self-tuned) and nevertheless obtaining outstanding performances. SDOclust is a powerful option when statistical estimates are representative and feature spaces conform distance-based analysis. Its main characteristics are: lightweight, intuitive, self-adjusted, noise-resistant, able to extract non-convex clusters, and built on robust parameters and interpretable models. Feasibility and rapid integration into real-world applications are the core goals behind the design of SDOclust.

Keywords: clustering · graphs · unsupervised learning · anomalies

1 Introduction

Sparse Data Observers (SDO) is a recent algorithm for anomaly detection [15]. Although it is general purpose, it was originally conceived for network traffic analysis, a field that demands the fast processing of high data volumes. Beyond complexity requirements, core goals in SDO are: (a) robust and intuitive parameterization, (b) the use of explainable models, and (c) the capability to effectively identify “novelties” as anomalies, even when they are dense and collective. SDOstream [12] is an extension of SDO for data streams. Since its publication in 2018, SDO has been used in diverse applications; e.g., advanced multi-fault diagnosis of batteries [25], anomaly detection in sensor networks [5].

In this work we propose SDOclust, which is the extension of SDO for clustering, therefore covering the two main branches of unsupervised learning. In the related literature we find an extensive collection of methods for clustering, each of them showing pros and cons, and being suitable for specific environments.

Suffice it to mention the survey by Xu and Tian, in which up to 71 algorithms are recalled and compared [29]. It is well accepted in the field that there is no “best” algorithm, that they must be evaluated based on application and goals, and that finding better evaluation procedures is even more pressing [18]. However, regardless of the algorithm used, experts still run into concerns related to whether the resulting clustering is reliable and reflects the real structure of the data, or whether the parameters used in the configuration are adequate [26]. Further discussing practical common disadvantages, Böhm et al. state that “they all [clustering algorithms] suffer from one or more of the following drawbacks: they focus on spherical or Gaussian clusters, and/or they are sensitive to outliers, and/or they need user-defined thresholds and parameters” [2].

To a large extent, SDOclust overcomes these common issues, fact that underlines its relevance among well-established methods. In another popular survey about clustering [30], Xu and Wunsch emphasize nine characteristics that are desirable in new generation algorithms. We analyze SDOclust in this light:

1. *Arbitrary shapes.* SDOclust is not confined to particular shapes and is able to capture non-convex and even nested clusters.
2. *Large volumes and high-dimensionality.* Based on statistical sampling, SDOclust benefits the larger the data volume both in terms of accuracy and complexity. On the other hand, SDOclust is a distance-based method that operates directly on the input feature space, so it is affected by the curse of dimensionality similarly to equivalent approaches. That said, empirical tests show excellent results up to 1024 dimensions (Sect. 4).
3. *Outlier/noise detection and removal.* SDOclust nests SDO, therefore inherently generating outlierness scores. On the other hand, clustering formation is rarely affected by outliers/noise in SDOclust. Finally, SDOclust does not remove or set outliers in a binary way, leaving the “oulier thresholding” task to be externally tackled according to application requirements.
4. *Low dependency on parameters.* SDOclust solves most scenarios with default parameters, which are also robust, intuitive, and self-adjustable. Challenging cases might require fine parameterization.
5. *Upgradeable models.* SDOclust can process data in chunks, meaning that it updates models with new data without retraining from scratch.
6. *Immune to data-order.* For SDOclust it makes no difference whether patterns are entered sequentially or jumbled.
7. *Guessing the number of clusters.* SDOclust does not require the number of expected clusters as a parameter externally imputed.
8. *Enriched outputs.* In addition to cluster labels, SDOclust outputs outlierness scores and cluster memberships per point, which can be easily converted into purity estimations¹. Additionally, SDOclust generates low-density models of the data shape by means of *observers*, which are representatives that preserve data geometry and relative density. Overall, outputs in SDOclust provide comprehensive information for visualization, description and post-analysis.

¹ We term a data point as *impure* when it lies in unclear zones between clusters.

9. *Mixed data types.* SDOclust is a numerical method and does not natively work on categorical data. In the current implementation, mixed data types require being adapted during preprocessing.

Note that SDOclust adjusts itself under the assumption that data statistical estimates are representative. Also clusters that are overlapping or show very strong differences in density might be difficult to solve and require fine parameterization. For the evaluation, we test SDOclust on a large number of datasets with a wide variety of shapes and challenges. Experiments are divided into: *two-dimensional data*, to provide a visual assessment, and *multi-dimensional data*, to cover more practical clustering application environments. In both cases internal and external validation metrics are shown. As competitors we use HDBSCAN [3] and *k*-means- [4]. HDBSCAN is one of the most notable general-purpose clustering algorithm, as it also meets the three key requirements: (a) operating with default parameters in a wide assortment of cases, (b) detecting outliers, and (c) capturing non-convex patterns. *k*-means- is selected as it is a most traditional clustering approach in its implementation with outlier detection.

The rest of the paper is organized as follows: Sect. 2 delves into SDO/SDOclust algorithms and parameters. In Sect. 3 we present evaluation data and metrics. Results are shown and discussed in Sect. 4, and two real application examples are explored in Sect. 5. We conclude with main remarks in Sect. 6.

2 Clustering Based on Sparse Data Observers

SDOclust is based on SDO. In this section we briefly explain the basic principles of SDO and then expand in more detail on SDOclust. Configurable parameters are highlighted in **bold** to be later explained in Sect. 2.3, where we discuss default values, automatic adjustment and the implications of their variation.

2.1 SDO

SDO [15] consists of two phases, here referred to as LEARNING and PREDICTION. During the LEARNING phase, the algorithm performs the following steps:

- L1 *Sampling.* S being a set of m input data points, O is a subset of k_0 random data points from S ($k_0 \ll m$). Elements in O are called *observers*.
- L2 *Observation.* With D as the distance² matrix between each pair of data points of S and O , an observation matrix I is derived by storing the \mathbf{x} -closest observers to each data point in S . Intuitively, each data point is “observed” only by its \mathbf{x} -closest observers.
- L3 *Removal of idle observers.* From the observation matrix I , we can compute P , an array that contains the occurrences of each observer in I , or, in other words, the number of data points that each observer “observes”. If an

² With *distance*—or the $d(\cdot)$ function—we refer to *Euclidean* distance, but the method is not restricted to it.

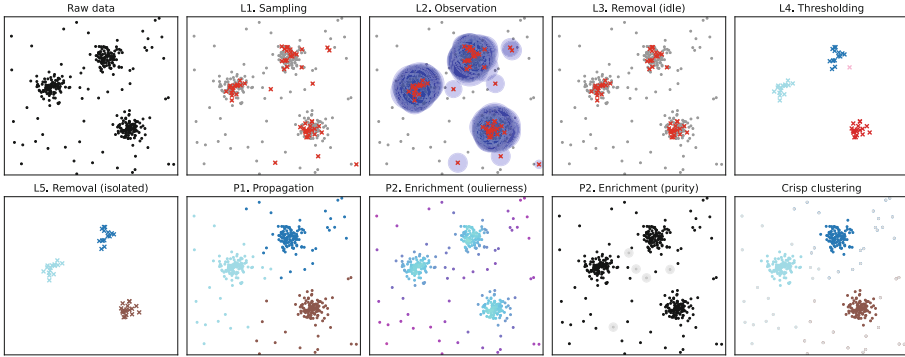


Fig. 1. Description of LEARNING and PREDICTION. L1 samples input data to create the observers set (red ‘x’s). Observers “observe” input samples in L2 (shadow size represents observations). *Idle* observers are removed in L3. Remaining observers are connected in clusters in L4 by considering cut-off thresholds. In L5 *isolated* observers are removed. Observers’ labels are propagated to data points in P1. Outlierness, membership and purity estimations are bound to data points in P2 (*impure* data points cast a shadow). The last plot (right-bottom) shows the derived crisp clustering with outlier detection (outliers in gray color). (Color figure online)

observer does not observe at least q data points, it is considered *idle* and removed from O . This minimizes the chance of selecting outliers as observers. Thus, the final O is a low-density *model* of S formed by k *active* observers ($k < k_0$).

During the PREDICTION phase:

P1 *Observation*. Data points in S , or new objects from a consistent dataset (we keep calling them S for consistency), are evaluated by O . This generates new D and I matrices (but note that now O only contains active observers).

P2 *Outlierness estimation*. For each data point s_i in S , an outlier score y_i is calculated as the average distance from s_i to its x -closest observers.

2.2 SDOclust

SDOclust also implements LEARNING and PREDICTION (Fig. 1). Additionally, it includes an UPDATE phase that is only called in batch mode, i.e., SDOclust processes new data while keeping and updating an already trained model.

The LEARNING phase of SDOclust follows these steps:

L1, L2, L3 *Sampling, Observation and Removal of idle observers* are exactly the same as in SDO (Sect. 2.1).

L4 *Thresholding for graph-edge cutting*. Here, observers in O are seen as nodes of an undirected graph to be clustered. We need to create and adjacency matrix A that will ultimately group or separate observers.

For this, instead of using a unique external cutoff threshold, each observer estimates its own locally. Therefore, the cutoff threshold h_i of a given observer o_i is:

$$h_i = d(o_i, o_{i \leftarrow \chi}) \quad (1)$$

i.e., the distance to its χ th-closest observer. Thus, the element $a_{i,j}$ of A is:

$$a_{i,j} = \begin{cases} 1 & \text{if } d(o_i, o_j) < h_i \text{ and } d(o_i, o_j) < h_j \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

meaning that observers o_i and o_j will be connected if the distance between them is below h_i and h_j . The purpose of using per-node (or local) cutoff thresholds is to allow solutions with clusters of different densities. On the other hand, this is prone to connect clusters lying very close to each other and create clusters in noise. To compensate this, a ζ coefficient weights the local contribution of h_i with a global contribution \bar{h} obtained as the average of $\{h_1, h_2, \dots, h_k\}$. Therefore, the final h'_i remains:

$$h'_i = \zeta h_i + (1 - \zeta) \bar{h} \quad (3)$$

and A is constructed with $\{h'_1, h'_2, \dots, h'_k\}$ instead. From each isolated subgraph of connected observers in A we extract a unique label c_i (any given observer is at least connected to itself), generating the set of unique labels $C = \{c_1, c_2, \dots, c_z\}$. Since connected observers share the same label of C , this results in the array of labels $L = \{l_1, l_2, \dots, l_k\}$, where $l_i \in C$.

- L5 *Removal of isolated observers*: L4 might generate clusters with isolated observers (commonly located in noisy areas). To obtain clean clusters, observers forming subgraphs of less than e nodes are removed (from O , P and L).

During the PREDICTION phase:

- P1 *Propagation*. This step runs an *Observation* phase and additionally propagates observers' labels to data points. Therefore, each data point in S , or new data points from a consistent dataset (we continue calling them S), inherits the labels of its \mathbf{x} -closests observers in addition to the outlieriness score. Unlike with the outlieriness score, which is computed as an average distance, observers' labels are categories, making that each data point in S can be multi-labeled. Therefore, if we imagine a space where discovered observers' clusters are orthogonal coordinates $\hat{c}_1, \hat{c}_2, \dots, \hat{c}_z$, a data point s_i will obtain a vector label $\vec{l}_i = (l_{i,1}, l_{i,2}, \dots, l_{i,z})$, where $l_{i,j}$ is the sum of the \mathbf{x} -closest observers to s_i that belong to cluster c_j .

P2 *Enrichment*. This phase acknowledges that any point s_i obtains an outlier-ness score y_i and a label vector \vec{l}_i . The label vector can be turned into a crisp label: $l_i|_{\text{crisp}}$ marking the *dominant* coordinate³ of \vec{l}_i , into a *membership* vector: $l_i|_{\text{memb}} = (\frac{l_{i,1}}{\|\vec{l}_i\|_1}, \frac{l_{i,2}}{\|\vec{l}_i\|_1}, \dots, \frac{l_{i,z}}{\|\vec{l}_i\|_1})$, or into a *purity* estimation: $l_i|_{\text{pur}} = \|\vec{l}_i|_{\text{memb}}\|_\infty$.

Finally, in the UPDATING phase both LEARNING and PREDICTION are repeated with minor variations. The new subset of data points is sampled for observers while keeping the proportion of one observer per m/k data points in the batch (minimum 1). New observers are added to O , new observations are added to P and, finally, the less-active observers are removed to keep k invariant.

2.3 Parameters: Interpretation, Default Values and Self-tuning

\mathbf{k}_0 , \mathbf{x} and \mathbf{q} (or $\boldsymbol{\rho}$)⁴ are intrinsic to SDO and widely discussed in the original paper [15]. We comment on them briefly. If not given externally, the initial number of observers—here termed \mathbf{k}_0 —is set based on statistical sampling of finite populations. For this estimation, the most variant dimension of the input data after PCA (Principal Component Analysis) is used (Eq. 4):

$$\mathbf{k}_0 = \frac{mZ^2\sigma^2}{(m-1)\epsilon^2 + Z^2\sigma^2} \quad (4)$$

with σ as the variance and $\text{CI} = 95\%$ ($Z = 1.96$), error $\epsilon = 0.1\sigma$. This ensures good scalability regardless of the number of data points and dimensions of the input data. For instance, in a n -dimensional dataset S with one thousand data points generated at random with normal distributions ($\mu = \sigma = 1$), SDO/SDOclust ($\mathbf{x} = 5$, $\boldsymbol{\rho} = 0.3$) estimates $k \simeq 191$; whereas for the same dataset with one million data points $k \simeq 266$. On the other hand, \mathbf{x} expresses neighborhood similarly to k in the k -nearest neighbor algorithm, but without being sensitive to density [31]. Empirical tests confirm the robustness of \mathbf{x} and $\boldsymbol{\rho}$, with $\mathbf{x} = 5$ and $\boldsymbol{\rho} = 0.3$ as suitable default values.

Parameters exclusive of SDOclust are:

- χ defines the χ th-closest observer of any given observer. It is used to establish the local threshold for cutting-off graph edges (Eq. 1). Small χ is appropriate when solutions with many clusters are expected, whereas high χ should be used when only few clusters are foreseen.
- ζ sets a trade-off between locality and globality in thresholds for cutting-off graph edges, $\zeta = 1$ for purely local and $\zeta = 0$ for purely global (Eq. 3). Local thresholding allows clusters of considerable density difference; however, it tends to merge nearby clusters and form clusters in noise. Global thresholding avoids such merger, but might divide legit clusters.

³ i.e., the coordinate with the highest value. In the absence of a dominant coordinate, the algorithm forces it randomly among the highest candidates.

⁴ \mathbf{q} is commonly obtained as $\mathbf{q} = Q(\boldsymbol{\rho}, P)$, where $Q(\cdot)$ is the *quantile* function.

- e sets the minimum number of observers that a cluster can have. This parameter commonly takes small values and is used to avoid clustering noise.

Empirical tests show that the previous parameters tolerate variations around default values: $\chi = \min(8, \frac{k}{20})$, $\zeta = 0.6$ and $e = 3$. The reason why such parameters work properly in most data scenarios is because, regardless of the number of samples and dimensions, SDOclust summarizes data in a model composed by a number of observers that falls always in a similar order of magnitude.

Finally, since the selection of observers is natively performed at random, SDOclust is not free from stochastic problems (more noticeable in small datasets).

2.4 Complexity

The increase in complexity of SDOclust with respect to SDO is not significant, since the additional operations carried out by SDOclust happen on the observer set which, as seen in Sect. 2.3, if not imposed as an external parameter, converges asymptotically as a function of m . Therefore, the added algorithmic load of SDOclust happens in the thresholding and clustering of the graph during the LEARNING phase. SDOclust works at all times with matrices of size $m \times k$, $k \times k$, $k \times \mathbf{x}$, and does not involve iterative processes. Therefore, as in [15], in Big-O notation the complexity of SDOclust is described as $O(mk)|_{m \rightarrow \infty} = O(m)$.

3 Evaluation

Implementations of SDOclust, experimental tests and sensitivity analysis for main parameters are freely available in our repository⁵, and through a stable DOI-citable version for Reproducible Research in [16]. Datasets can be downloaded from their original sources or—in case that they are further processed or generated by tools—they are also included in our repository. In all experiments, HDBSCAN (as provided in [20]) and SDOclust run with no parameters (default or self-tuned)⁶, whereas k -means- is always imputed with the right number of clusters and outliers to discover⁷, extracted from the ground truth (GT).

⁵ <https://github.com/CN-TU/pysdoclust>.

⁶ HDBSCAN parameters: `min_cluster_size = 5`, `cluster_selection_epsilon = 0.0`, `approx_min_span_tree = True`, `allow_single_cluster = False`, `min_samples = None`, `algorithm = 'best'`, `p = None`, `alpha = 1.0`, `metric = 'euclidean'`, `leaf_size = 40`, `memory = Memory(location = None)`, `cluster_selection_method = 'eom'`, `gen_min_span_tree = False`, `core_dist_n_jobs = 4`, `prediction_data = False`, `match_reference_implementation = False`;
SDOclust parameters: `x = 5`, `qv = 0.3`, `zeta = 0.6`, `chi_min = 8`, `chi_prop = 0.05`, `e = 3`, `chi = None`, `xc = None`, `k = None`, `q = None`.

⁷ k -means- is tuned with `maximum_iterations = 1000` and `tol = 0.0001`, where `tol` is the convergence criterion for centroid displacement.

3.1 Datasets

Evaluation experiments are in two sets: (a) 15 *two-dimensional datasets* to visually assess performances; (b) 138 *multi-dimensional datasets*. Dataset collections are taken from different sources: (i) The Clustering Basic Benchmark of the University of Eastern Finland [10], which is one of the most known data collections for clustering evaluation⁸. This includes two-dimensional datasets: *s1* [8], *r15* [27], *aggregation* [11], *skewed* [22]; also the low-to-high dimensional set termed *d*, which combines two collections [9,17] and shows datasets from 3 to 1024 dimensions. (ii) Datasets generated or processed with the `sklearn.datasets` package⁹, due to its popularity and widespread use [21], including the two-dimensional datasets named: *rings* and *moons*. Later on, *iris* (Iris Flower), *mallcust* (Mall Customers Segmentation) and *pima* (Pima Indians Diabetes) are popular real datasets here projected into two-dimensional spaces by using t-distributed stochastic neighbor embedding (tSNE) [19]. (iii) Datasets generated with MDCgen [14], a tool to create multi-dimensional data for testing, evaluating, and benchmarking unsupervised classification algorithms. It includes the two-dimensional datasets: *close*, *separated*, *complex*, *high-noise* and *low-noise*; and 6 groups of multi-dimensional datasets, each group addressing a different data challenge, namely: *c* (close clusters, reduced space), *p* (separated clusters, large space), *n* (between 5% and 15% noise aprox.), *h* (between 15% and 30% noise aprox.), *f* (clusters with high density differences), and *x* (complex setups by combining previous challenges).

3.2 Metrics

Experiments are validated externally and internally. For external validation we use the Adjusted Rand Index (ARI) [13], which approaches ‘1’ the better the discovered partition matches the GT, and gives ‘-0.5’ for completely discordant partitions. For internal validation we use the Silhouette index (Sil) [23], which scores ‘1’ when intra-cluster cohesion and inter-cluster separation are maximized. To make Sil consistent, we first remove the top ‘n’ outliers, where ‘n’ is given by the GT. We also show the difference between the number of clusters discovered by the algorithm by excess (+) or deficiency (−) compared with the GT.

4 Results and Discussion

Table 1 shows two-dimensional experiments results. SDOclust guesses the number of clusters considerably better than HDBSCAN, which tends to cluster noise and divide bigger structures into micro-clusters. SDOclust average performances are 0.59 ± 0.22 (Sil) and 0.86 ± 0.10 (ARI), vs 0.53 ± 0.27 (Sil) and 0.81 ± 0.19 (ARI) in HDBSCAN and 0.57 ± 0.10 (Sil) and 0.67 ± 0.23 (ARI) in *k*-means--.

⁸ <https://cs.joensuu.fi/sipu/datasets/>.

⁹ <https://scikit-learn.org/stable/datasets.html>.

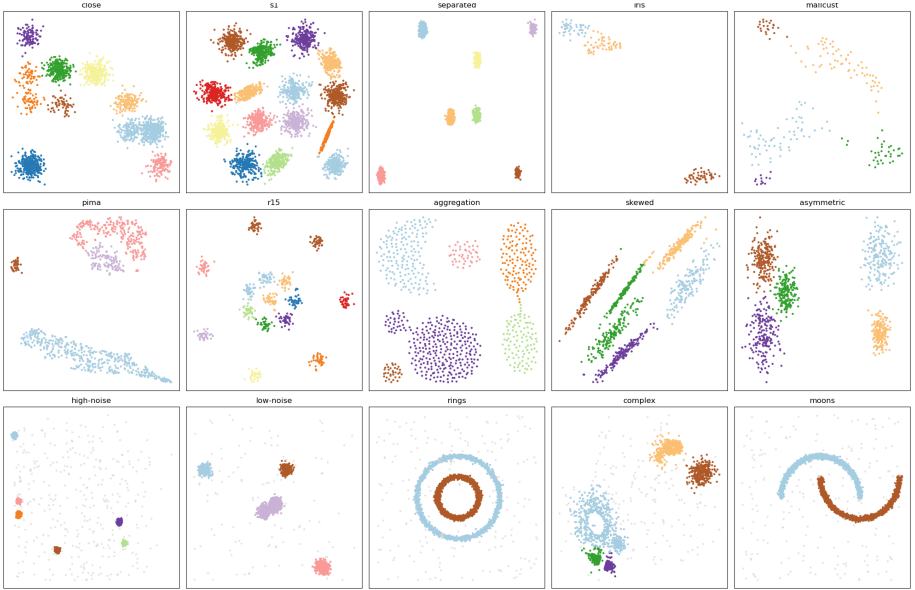


Fig. 2. Clustering of SDOclust on two-dimensional data. Clustered data points take colors based on their cluster, whereas outliers are in gray color (Color figure online)

Table 1. Results of tests with two-dimensional datasets

dataset	inlier samples	dimensions	outliers	clusters GT	clusters HDBSCAN	clusters k -means	clusters SDOclust	Sil HDBSCAN	Sil k -means	Sil SDOclust	ARI HDBSCAN	ARI k -means	ARI SDOclust
chase	2000	2	0	11	-1	=	-2	0.63	0.58	0.65	0.87	0.81	0.87
s1	5000	2	0	15	+38	=	=	0.24	0.58	0.71	0.54	0.83	1.00
separated	2000	2	0	7	=	=	=	0.91	0.58	0.91	1.00	0.72	1.00
iris	150	2	0	3	-1	=	=	=	0.89	0.72	0.57	0.90	0.90
mallcust	200	2	0	4	=	=	+1	0.57	0.62	0.52	0.96	0.89	0.85
pima	768	2	0	3	=	=	+1	0.60	0.61	0.40	1.00	0.72	0.82
r15	600	2	0	15	=	=	-1	0.72	0.67	0.70	0.94	0.90	0.92
aggregation	788	2	0	7	-2	=	-1	0.37	0.44	0.46	0.81	0.63	0.91
skewed	1000	2	0	6	+5	=	-1	0.25	0.44	0.37	0.89	0.60	0.81
asymmetric	1000	2	0	5	-2	=	=	0.68	0.53	0.64	0.47	0.70	0.98
high-noise	2000	2	345	6	+10	=	+1	0.71	0.72	0.90	0.92	0.83	0.83
low-noise	2000	2	111	5	-1	=	-1	0.82	0.68	0.85	0.70	0.61	0.67
rings	3000	2	200	2	+4	=	=	0.08	0.38	0.12	0.97	0.07	0.88
complex	2000	2	155	6	+17	=	-1	0.38	0.51	0.56	0.61	0.62	0.65
moons	3000	2	200	2	+8	=	=	0.13	0.49	0.34	0.96	0.29	0.88

Both internal and external validation highlights SDOclust over its competitors. Figure 2 shows SDOclust clustering. Most noticeable inaccuracies are the tendency of SDOclust to merge very close clusters, more so if they show some overlap or are bridged by intermediate data points. There are different strategies to alleviate this, e.g., reducing ζ and χ while increasing ρ , but such modifications might cause arbitrary divisions in some cases.

Table 2 shows main properties of multi-dimensional dataset collections, while Table 3 provides clustering performances. Again, HDBSCAN and SDOclust run with default parameters, whereas k -means-- is imputed with the expected number of clusters and outliers. Summarizing results, we observe that discrepancies in the number of clusters between GT and SDOclust partitions is minimal, while it is larger for HDBSCAN, prone to overclustering in the presence of outliers. Even

Table 2. Description of multi-dimensional data groups

group	abbrev.	datasets	inlier samples	dimensions	outliers	clusters in GT
close	c	20	5000	3 to 23	0	10 to 14
separated	p	20	5000	5 to 22	0	3 to 7
low-to-high dim.	d	18	1024 to 10126	3 to 1024	0	9 to 16
density diff.	f	20	5000	3 to 22	0	3 to 7
medium noise	n	20	5000	3 to 22	252 to 748	3 to 7
high noise	h	20	5000	3 to 23	858 to 1981	4 to 7
complex	x	20	5000	3 to 22	272 to 747	3 to 7

Table 3. Results of tests with multi-dimensional data. “NCA-error” stands for the “accumulated error in the number of clusters due to over- or underclustering”

group	NCA-error HDBSCAN	NCA-error k -means-	NCA-error SDOclust	Sil HDBSCAN	Sil k -means-	Sil SDOclust	ARI HDBSCAN	ARI k -means-	ARI SDOclust
c	1	35	2	0.83 ± 0.05	0.53 ± 0.10	0.83 ± 0.04	1.00 ± 0.00	0.66 ± 0.39	1.00 ± 0.00
p	0	0	0	0.95 ± 0.01	0.70 ± 0.19	0.95 ± 0.01	1.00 ± 0.00	0.82 ± 0.16	1.00 ± 0.00
d	0	0	0	0.94 ± 0.03	0.64 ± 0.17	0.94 ± 0.03	1.00 ± 0.00	0.77 ± 0.14	1.00 ± 0.00
f	8	0	9	0.70 ± 0.18	0.64 ± 0.14	0.71 ± 0.12	0.95 ± 0.09	0.89 ± 0.11	0.97 ± 0.03
n	67	11	0	0.85 ± 0.04	0.81 ± 0.30	0.97 ± 0.01	0.83 ± 0.06	0.85 ± 0.29	1.00 ± 0.00
h	203	5	0	0.64 ± 0.09	0.81 ± 0.22	0.99 ± 0.00	0.48 ± 0.11	0.80 ± 0.18	1.00 ± 0.00
x	43	0	2	0.74 ± 0.06	0.73 ± 0.16	0.84 ± 0.09	0.82 ± 0.07	0.84 ± 0.17	0.98 ± 0.02

k -means-- shows higher drifts due to global failures when analyzing some scenarios. As for validation indices, SDOclust scores 0.89 ± 0.11 (Sil) and 0.99 ± 0.02 (ARI) in average, vs 0.80 ± 0.14 (Sil) and 0.87 ± 0.19 (ARI) in HDBSCAN and 0.70 ± 0.22 (Sil) and 0.81 ± 0.21 (ARI) in k -means--.

In Fig. 3 we show *critical difference diagrams* from Wilcoxon signed-rank tests over all 153 experiments [7]. The diagrams confirm the statistical differences when comparing results, SDOclust standing out with the best performance.



Fig. 3. Critical difference diagrams comparing algorithm results with Wilcoxon signed-rank tests [7]. Best methods are placed on the right side

The reason behind SDOclust ability to self-tune and obtain suitable clustering lies in leveraging statistical estimations to create simplified models—always with a number of elements (i.e., observers) in a similar order of magnitude—that capture shapes and relative densities independently of the number of input data points and dimensions. This makes possible to calculate and establish neighborhoods, coefficients and thresholds that satisfy a large number of cases.

5 Examples with Real Data

We show two examples of SDOclust for discovering clusters and patterns in real applications. We have selected cases with recent public data and where clustering makes practical sense beyond algorithm testing. We use HDBSCAN

as benchmark. As before, both algorithms apply default parameters and are not imputed with the number of clusters to discover. SDOclust establishes outliers per cluster as any data point with an outlieriness score beyond the mean plus the Median Absolute Deviation (MAD) over the standard deviation [1].

5.1 Clustering Network Traffic

Communication network traffic is hard to analyze: data spaces are noisy, classes heterogeneous and multiform, and features commonly show non-normal distributions and collinearity. We explore a traffic capture (pcap) from the MAWI samplepoint-F for July 31, 2022¹⁰. The MAWI Working Group daily publishes 15 min of backbone traffic publicly for research purposes [6]. We extract bi-directional flows between IP addresses (`srcIP`, `dstIP`) that exchange information for a maximal duration of 60 s. Since modern network traffic nowadays is mostly encrypted, we select features that are available regardless of encryption¹¹: the flow duration (`dur`) and statistics related the number of packets sent (`pkt`), the length of packets (`len`) and the inter-arrival time between packets (`iat`), both forward (F) and backward (B). The final vector that expresses a sample is:

$T:srcIP:dstIP \rightarrow dur, pktF, Md(lenF), Mn(iatF), pktB, Md(lenB), Mn(iatB)$

where T is the *timestamp* and, together with the `srcIP` and `dstIP`, forms a unique ID to identify flows. $Md()$ and $Mn()$ stand for the statistical *Mode* and *Mean*. The capture contains about 9 million flows in the given format, of which we randomly sample a thousandth part (8630 flows) for our exploration and normalize them with quantile normalization (the most consistent based on feature distributions). Domain knowledge suggests that higher layer protocols, yet heterogeneously, may influence traffic shapes. We use this information as a tentative benchmark for external validation (i.e., ARI). Analyzed flows account for: 82.5% TCP, 15.1% UDP, 2.1% ICMP, and 0.3% of others protocols.

Results reveal very different performances in terms of the number of clusters between HDBSCAN and SDOclust, although similar in validation scores:

HDBSCAN: 117 clusters, 322 outliers, Sil = 0.89, Sil(inliers) = 0.96, ARI = 0.32
 SDOclust: 8 clusters, 53 outliers, Sil = 0.89, Sil(inliers) = 0.89, ARI = 0.31

In Fig. 4 we use tSNE to visualize solutions given by HDBSCAN, SDOclust and protocol distribution. tSNE has proven excellent at capturing the structure of high dimensional data, it performs projections based on local similarity that are consistent with cluster shapes. From this perspective, SDOclust appears to align more naturally with tSNE and the protocol labeling and number of clusters than HDBSCAN. Nevertheless, by removing outliers and due to the high number of clusters found, HDBSCAN shows a remarkable improvement in the internal validation, thus indicating its ability to find micro-clusters of high purity.

¹⁰ <https://mawi.wide.ad.jp/mawi/samplepoint-F/2022/202207311400.html>.

¹¹ Extracted with Go-flows [28].

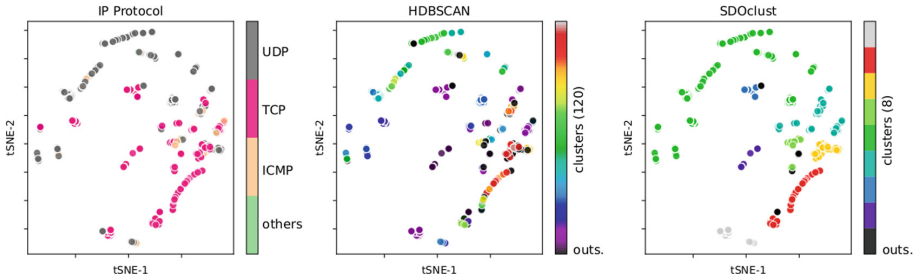


Fig. 4. tSNE proj. (colors are protocol), HDBSCAN and SDOclust clustering. (Color figure online)

5.2 Energy Building Profiles

This second example applies clustering to reveal characteristic profiles in building energy performance. We use real electricity consumption data of the “CN Rectorat” of the Polytechnic University of Catalonia (UPC) during 2022. This data is openly published by the Sirena Project [24]¹². We rearrange hourly consumption data to process 365 vectors (days) of 24 features (hours). Note that each vector is a time series, i.e., a daily energy consumption curve. Since raw data is magnitude consistent and in order not to break dependencies between features, we do not apply any normalization. Euclidean distances estimate similarity as usual since we want to group samples considering both curve shape and volume of electricity. SDOclust and HDBSCAN obtain the following results:

HDBSCAN: 8 clusters, 77 outliers, Sil = 0.28, Sil(inliers) = 0.48

SDOclust: 7 clusters, 16 outliers, Sil = 0.43, Sil(inliers) = 0.48

Again, differences in validation between SDOclust and HDBSCAN are small, this time also with regard to discovered clusters (both in number and shapes, Fig. 5). The similarity between both clusterings is $ARI = 0.76$. The deterioration of HDBSCAN in the complete internal assessment (inliers and outliers) is due to its tendency to find many outliers (21% in this case). As for the application, it is tempting to think that discovered profiles correspond to the days of the week; however, the close analysis reveals that they rather show a seasonal character. Taking SDOclust clusters as a reference: (a) P2 is dominant throughout the year; (b) P3 (winter-spring) and P6 (summer-autumn) are almost exclusively for Mondays; (c) P4 occurs mainly in winter and early spring; (d) P1 mostly in April; (e) P5 mainly from late spring to autumn, but excluding the summer holiday period; (f) and P0 mainly during summer.

¹² <https://upsirena.app.dexma.com/>.

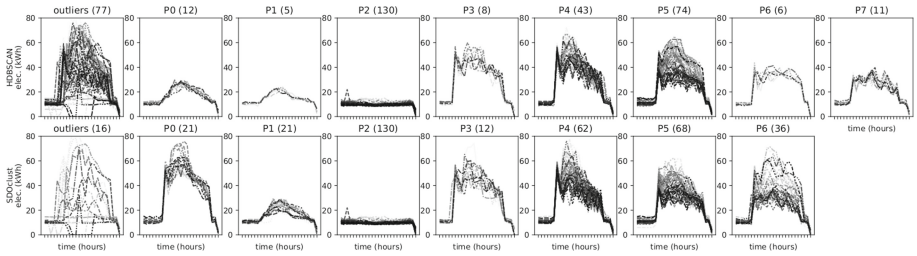


Fig. 5. Clustered profiles disclosed by HDBSCAN (top) and SDOclust (bottom)

These two examples with real data show how HDBSCAN is a method that prioritizes cluster purity (then with a marked risk of overclustering), while SDOclust prioritizes generalization (therefore with a risk of underclustering).

6 Conclusions

In this paper we have presented SDOclust, a clustering algorithm based on the unsupervised analysis approach pioneered by its predecessor SDO. We have tested SDOclust with default parameters on 155 datasets comprising a wide variety of sizes, dimensions and specific data challenges, yet obtaining excellent results and outperforming the popular HDBSCAN and k -means- algorithms. Considering default settings, compared to HDBSCAN, which is arguably the most efficient and autonomous alternative, SDOclust tends to find the main top-level spatial partitions and, therefore, is significantly less prone to generate micro-clusters. Key features of SDOclust are: linear complexity, scalability, parameters that are robust, intuitive and self-tuned, resistance to outliers and noise, ability to discover non-convex clusters, use of simple and updateable models, and the generation of rich outputs for detailed post-analysis. Next planned steps mainly involve an incremental version of SDOclust for streaming data.

References

1. Archana, N., Pawar, S.: Periodicity detection of outlier sequences using constraint based pattern tree with mad. *Int. J. Adv. Stud. Comput. Sci. Eng.* **4**(6), 34 (2015)
2. Böhm, C., Faloutsos, C., Pan, J.Y., Plant, C.: Robust information-theoretic clustering. In: *12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2006*, pp. 65–75. Association or Computer Machine, New York (2006)
3. Campello, R.J.G.B., Moulavi, D., Zimek, A., Sander, J.: Hierarchical density estimates for data clustering, visualization, and outlier detection. *ACM Trans. Knowl. Discov. Data* **10**(1), 5:1–5:51 (2015)
4. Chawla, S., Gionis, A.: k -means-: a unified approach to clustering and outlier detection. In: *2013 SIAM International Conference on Data Mining*, pp. 189–197. SIAM (2013)

5. Chen, L., Xu, L., Li, G.: Anomaly detection using spatio-temporal correlation and information entropy in wireless sensor networks. In: IEEE Congress on Cybermatics: iThings, GreenCom, CPSCom, SmartData, pp. 121–128 (2020)
6. Cho, K., Mitsuya, K., Kato, A.: Traffic data repository at the wide project. In: Proceedings of the Annual Conference on USENIX Annual Technical Conference, ATEC 2000, p. 51. USENIX Association, USA (2000)
7. Demšar, J.: Statistical comparisons of classifiers over multiple data sets. *J. Mach. Learn. Res.* **7**, 1–30 (2006)
8. Fränti, P., Virtajoki, O.: Iterative shrinking method for clustering problems. *Pattern Recogn.* **39**(5), 761–765 (2006)
9. Fränti, P., Virtajoki, O., Hautamäki, V.: Fast agglomerative clustering using a k-nearest neighbor graph. *IEEE Trans. Pattern Anal. Mach. Intell.* **28**(11), 1875–1881 (2006)
10. Fränti, P., Sieranoja, S.: K-means properties on six clustering benchmark datasets. *Appl. Intell.* **48**(12), 4743–4759 (2018)
11. Gionis, A., Mannila, H., Tsaparas, P.: Clustering aggregation. *ACM Trans. Knowl. Disc. Data (TKDD)* **1**(1), 4-es (2007)
12. Hartl, A., Iglesias, F., Zseby, T.: Sdostream: low-density models for streaming outlier detection. In: 28th ESANN, Bruges, Belgium, 2–4 October 2020, pp. 661–666 (2020)
13. Hubert, L., Arabie, P.: Comparing partitions. *J. Classif.* **2**(1), 193–218 (1985)
14. Iglesias, F., Zseby, T., Ferreira, D., Zimek, A.: Mdcgen: multidimensional dataset generator for clustering. *J. Classif.* **36**(3), 599–618 (2019)
15. Iglesias, F., Zseby, T., Zimek, A.: Outlier detection based on low density models. In: IEEE International Conference on Data Mining Workshops (ICDMW), pp. 970–979 (2018)
16. Iglesias Vázquez, F.: SDOclust Evaluation Tests (2023). <https://doi.org/10.48436/3q7jp-mg161>
17. Kärkkäinen, I., Fränti, P.: Gradual model generator for single-pass clustering. *Pattern Recogn.* **40**(3), 784–795 (2007)
18. von Luxburg, U., Williamson, R.C., Guyon, I.: Clustering: science or art? In: Proceedings of ICML Workshop on Unsupervised and Transfer Learning, vol. 27, pp. 65–79. PMLR (2012)
19. Van der Maaten, L., Hinton, G.: Visualizing high-dimensional data using t-sne. *J. Mach. Learn. Res.* **9**(2579–2605), 9 (2008)
20. McInnes, L., Healy, J., Astels, S.: hdbscan: Hierarchical density based clustering. *J. Open Source Softw.* **2** (2017). <https://hdbscan.readthedocs.io/en/latest/>
21. Pedregosa, F., et al.: Scikit-learn: machine learning in Python. *J. Mach. Learn. Res.* **12**, 2825–2830 (2011)
22. Rezaei, M., Fränti, P.: Can the number of clusters be determined by external indices? *IEEE Access* **8**, 89239–89257 (2020)
23. Rousseeuw, P.J.: Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *J. Comput. Appl. Math.* **20**, 53–65 (1987)
24. Ruiz Martorell, G., López Plazas, F., Cuchí Burgos, A.: Sistema d’informació del consum d’energia i d’aigua de la UPC (Sirena). 1r Congrés UPC Sostenible (2007)
25. Sun, J., Qiu, Y., Shang, Y., Lu, G.: A multi-fault advanced diagnosis method based on sparse data observers for lithium-ion batteries. *J. Energy Storage* **50**, 104694 (2022)

26. Tsai, C.-Y., Chiu, C.-C.: A clustering-oriented star coordinate translation method for reliable clustering parameterization. In: Washio, T., Suzuki, E., Ting, K.M., Inokuchi, A. (eds.) PAKDD 2008. LNCS (LNAI), vol. 5012, pp. 749–758. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-68125-0_72
27. Veenman, C.J., Reinders, M.J.T., Backer, E.: A maximum variance cluster algorithm. *IEEE Trans. Pattern Analy. Mach. Intell.* **24**(9), 1273–1280 (2002)
28. Vormayr, G., Fabini, J., Zseby, T.: Why are my flows different? a tutorial on flow exporters. *IEEE Commun. Surv. Tutor.* **22**(3), 2064–2103 (2020)
29. Xu, D., Tian, Y.: A comprehensive survey of clustering algorithms. *Ann. Data Sci.* **2** (2015)
30. Xu, R., Wunsch, D.: Survey of clustering algorithms. *IEEE Tran. Neural Netw.* **16**(3), 645–678 (2005)
31. Zimek, A., Gaudet, M., Campello, R.J., Sander, J.: Subsampling for efficient and effective unsupervised outlier detection ensembles. In: Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 428–436 (2013)