D.MCA: Outlier Detection with Explicit Micro-Cluster Assignments

Shuli Jiang
Carnegie Mellon University
shulij@andrew.cmu.edu

Robson L. F. Cordeiro
University of São Paulo
robson@icmc.usp.br

Leman Akoglu
Carnegie Mellon University
lakoglu@andrew.cmu.edu

***Abstract*—How can we detect outliers, both scattered and clustered, and also explicitly assign them to respective micro-clusters, without knowing apriori how many micro-clusters exist? How can we perform both tasks in-house, i.e., without any post-hoc processing, so that both detection and assignment can benefit simultaneously from each other? Presenting outliers in separate micro-clusters is informative to analysts in many real-world applications. However, a naïve solution based on post-hoc clustering of the outliers detected by any existing method suffers from two main drawbacks: (a) appropriate hyperparameter values are commonly unknown for clustering, and most algorithms struggle with clusters of varying shapes and densities; (b) detection and assignment cannot benefit from one another. In this paper, we propose D.MCA to Detect outliers with explicit Micro-Cluster Assignment. Our method performs both detection and assignment iteratively, and in-house, by using a novel strategy that prunes entire micro-clusters out of the training set to improve the performance of the detection. It also benefits from a novel strategy that avoids clustered outliers to mask each other, which is a well-known problem in the literature. Also, D.MCA is designed to be robust to a critical hyperparameter by employing a hyperensemble "warm up" phase. Experiments performed on 16 real-world and synthetic datasets demonstrate that D.MCA outperforms 8 state-of-the-art competitors, especially on the explicit outlier micro-cluster assignment task.**

***Index Terms*—outlier detection, outlier micro-cluster assignment, hyperparameter robustness**

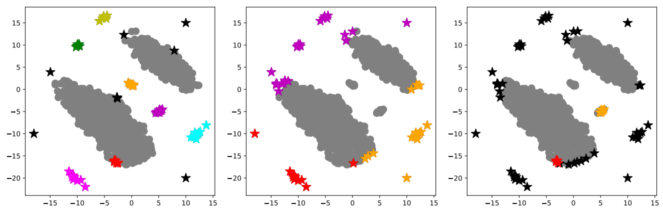
I. Introduction

Outlier detection aims to identify rare or unusual instances in the data that deviate from the majority. In many practical domains outliers can form groups or (micro-)clusters, which can be seen as anomalous patterns. Examples include bots or malware under the same command-control in network intrusion, coordinated attacks from multiple user accounts toward spreading fake news in social platforms, opportunistic fraud/scam/etc. that spread by word-of-mouth, adversarial schemes that exploit the same loophole or vulnerability in a system (e.g., medical insurance), to name a few.

Autonomous outlier detection systems are rarely used in the real world, where it is typical for the flagged outliers to go through a verification/vetting process by a domain analyst, especially when the cost of false positives is high, such as shutting down credit cards, blacklisting certain software or user accounts from access, charging a physician or hospital by fraud, etc. It is exactly in such scenarios that reporting micro-clusters of outliers, if any, is useful for the domain analyst whose job may be to go through hundreds of such alerts a day. Rather than inspecting each outlier one by one, a succinct presentation of the outliers by groups could speed up sense-making, characterization and ultimately decision making, where e.g. the analyst can take the same troubleshooting action for all outliers in the same cluster, or quickly ignore all in the same cluster provided a few have already been found to be false positives or semantically uninteresting outliers.

Note this is a short version of the paper. Please check out arXiv.org for the full version including Appendix.

Acknowledgement This work is partly sponsored by the PwC Risk and Regulatory Services Innovation Center at Carnegie Mellon University, and by the Brazilian foundations CAPES (finance code 001), FAPESP (grants 2021/05623-2, 2020/07200-9, 2018/05714-5 and 2016/17078-0), and CNPq.



(a) D.MCA (b) iForest+X-means (c) Gen^{2}Out

Fig. 1: Example dataset with two inlier modalities (gray dots), with scattered and clustered outliers. (a) Proposed D.MCA effectively flags the outliers (stars) and also explicitly assigns to micro-clusters (colored differently). (b) Two-stage solutions, detection+post hoc clustering, are not as effective. (c) SOTA baseline underperforms, often missing interior outliers.

The specific problem we consider in this work is then to not only identify the outlier instances (both clustered and scattered) but at the same time, output existing outlier micro-clusters explicitly. We refer to the latter task as outlier *micro-cluster assignment* (e.g. see Fig. 1(a)). Our goal is to design an algorithm that addresses both problems *in-house* as a result of its inherent detection mechanism, rather than *post-hoc* where one employs a clustering algorithm upon first detecting the outliers. Such straightforward approaches are arguably far from trivial to get right due to multi-pronged challenges. First, the detection algorithm should be very effective as those outliers are fed downstream to clustering. Second, clustering itself is a non-trivial task with small, i.e. micro, clusters of possibly varying size and density, where further the number of outlier clusters are not known apriori, all of which relate to the challenge of hyperparameter selection. Finally, a two-stage approach puts a barrier between the detection and assignment tasks that prevents them from potentially benefiting from each other. We compare to various two-stage baselines in experiments, using SOTA detectors such as Isolation Forest [1] and LOF [2], paired with parameter-free clustering algorithms such as X-means [3] and OPTICS [4] (e.g. see Fig. 1(b)).

The vast body of prior work is for detecting outliers only, without special focus on clustered outliers. In principle several of those detectors are capable of spotting outliers that are present in micro-clusters, provided suitable hyperparameter values, however they do not specify which outliers are scattered versus clustered. There exist work clustering detected outliers which is post-hoc and mainly focuses on explanation [5]. Some work have addressed detecting clustered or collective outliers specifically [6], [7], [8], however without explicitly assigning them to their respective clusters. The only work (to our best knowledge), Gen^{2}Out [9], that considers explicit outlier assignment is based on the aforementioned naïve two-stage pipeline: it detects outliers first and then applies the post-processing clustering algorithm DBSCAN [10] on the detected outliers. It relies heavily on the accuracy of the detector which falls short to spot outliers in the interior of the data manifold (e.g. see Fig. 1(c)). See Appendix V for detailed related work.

In this paper, we simultaneously address the outlier Detection and explicit outlier Micro-Cluster Assignment tasks. Our proposed D.MCA alternatingly refines the micro-clusters and the flagged outliers, aiming to synergize them to mutually benefit from one another. In a nutshell, D.MCA builds on the SOTA iNNE detector [11] which subsamples a set of ψ points, creates a hyperball centered at each one, and scores outlierness as relative distance to those representatives. The working assumption is that the subsample contains inliers with high probability, and hence the hyperballs are representative of the normal data distribution. Note that iNNE originally cannot produce explicit micro-cluster assignments.

Our proposed D.MCA improves this base algorithm in two key aspects. Firstly, it employs an iterative pruning strategy where the top scoring points at each iteration are pruned. In effect, this improves the probability that the subsampled points in the next round are more likely to be inliers, which lead to better outlier detection. Second, D.MCA employs a hyperensemble "warm up" phase that makes it more robust to the choice of its hyperparameter, namely the subsample size ψ, which at the same time also helps alleviate the issue of "masking"; when two or more points from an outlier cluster are sampled, they mask/hide one another and cause all points in the micro-cluster to be deemed inliers (i.e. false negatives). We summarize our main contributions as follows.

- **Outlier detection and micro-cluster assignment:** We consider the two-pronged problem of (scattered and clustered) outlier detection and explicit outlier micro-cluster assignment, and propose a new algorithm called D.MCA to address both problems simultaneously.
- **Tackling various challenges:** D.MCA exhibits novel procedures toward (i) effective outlier pruning – which specifically aims to avoid pruning false positives (i.e. inliers), (ii) reduced "masking" effect – which carefully regulates the subsample size[1] and prunes true positive micro-clusters, and (iii) improved robustness to hyper-parameter settings – which employs a hyperensemble strategy as a "warm up" phase.
- **A synergistic solution:** D.MCA's detection and micro-cluster assignment steps work together in synergy under a unified algorithm, rather than two independent components: the pruning strategy reveals micro-clusters as a by-product, and true-positive micro-clusters are pruned to improve the subsample and hence the detection quality.
- **Effectiveness:** Through extensive experiments on synthetic datasets, real-world datasets injected with known clustered outliers, as well as benchmark real-world datasets, we show that D.MCA sets the state-of-the-art, significantly outperforming (to our knowledge) the only SOTA baseline Gen^{2}Out as well as a number of two-stage baselines with paired detector and post-hoc clustering, especially in the micro-cluster assignment task.

[1]Note that a large sample is more likely to induce "masking", leading to false negatives (i.e. missed outlier micro-clusters), whereas a small sample would contain insufficient representatives, leading to false positives.

Reproducibility. All our source code and the datasets used in the experiments are publicly shared at https://github.com/11hifish/D.MCA.

II. Preliminaries

Notation See Appendix I-A.

Problem Given a point-cloud dataset $\mathcal{D}$, our goal is to perform the following two tasks:

1) *Detection:* compute $\mathbf{x}_i \in \mathcal{D}, \forall i \in [n]$ an outlier score $s(\mathbf{x}_i)$; the larger, the more anomalous and

2) *Assignment:* explicitly output outlier micro-clusters $\mathcal{C}_1, \ldots, \mathcal{C}_m$, where m is apriori unknown to the algorithm.

iNNE Outlier Detector D.MCA uses individual models of Isolation using Nearest Neighbor Ensemble (iNNE) [11] as the base outlier detector. See Appendix I-B for more details.

III. Proposed Approach

The section is divided into two parts. We first present the D.MCA-0 algorithm (Algorithm 1), a sequential ensemble that alternates between detection and assignment at each iteration. However, D.MCA-0 is sensitive to an important hyperparameter ψ, the subsample size, inherited from the iNNE base detector. Next we present the proposed D.MCA (Algorithm 2), a two-phase algorithm that builds a *hyperensemble* based on varying values of ψ in the first phase prior to employing D.MCA-0 in the second phase, which is more robust to ψ compared to D.MCA-0.

A. Proposed D.MCA-0

The motivation of performing detection and assignment interactively across iterations in a sequential ensemble is two-fold: (i) to obtain a better assignment of the outliers based on improved detection, and (ii) to leverage the identified micro-clusters to improve the detection performance.

At a high level, D.MCA-0 builds upon iNNE to detect outliers, which outputs an outlier score s for each instance in $\mathcal{D}$ and a set of selected centers $\mathcal{R}_\psi$ as inlier representatives per iteration. D.MCA-0 keeps a running average of the outlier scores across iterations. We choose iNNE as the base model as it is fast and efficient, able to detect non-axis aligned outliers, adapts to data with varying density, and produces explicit subsamples $\mathcal{R}_\psi$ that our algorithm makes use of. D.MCA-0 keeps track of the assignment of outlier micro-clusters by a weighted neighbor graph $\mathcal{G} = (\mathcal{D}, \mathcal{E})$ where $\mathcal{E} = (\mathbf{x}, \mathbf{y}), \forall \mathbf{x}, \mathbf{y} \in \mathcal{D}$. The weight of an edge $(\mathbf{x}, \mathbf{y}) \in \mathcal{E}$ denotes the number of times both instances $\mathbf{x}, \mathbf{y} \in \mathcal{D}$ receive high outlier scores and are marked as "neighbors". D.MCA-0 updates the weight of $\mathcal{G}$ at each iteration, and outputs the final outlier micro-clusters $\mathcal{C} = \{\mathcal{C}_1, \mathcal{C}_2, \dots\}$ based on the connected components of $\mathcal{G}$ (Procedure FindClusters in Appendix II).

In what follows, we present four major steps of D.MCA-0 during a single iteration in the sequential ensemble: *1)* finding micro-cluster representatives via sampling, *2)* filtering out false positives from micro-cluster representatives toward pruning, *3)* updating the neighbor graph $\mathcal{G}$ and *4)* pruning true micro-cluster representatives and their neighbors. We remark that steps *1)* and *3)* are geared toward *(i)* to obtain outlier micro-clusters and steps *2)* and *4)* are geared toward *(ii)* to improve detection performance. Notice how these steps are interleaved, creating synergy between the two tasks.

1) Finding Micro-cluster Representatives via Maximin Sampling: To find outlier micro-clusters, the algorithm first applies Maximin Sampling [12] on outliers that receive p topmost outlier scores, i.e. points in $\mathcal{H}$-top (Line 7), to get an outlier representative set $\mathcal{H}$ which contains at least one representative per micro-cluster (Line 8). At each iteration, Maximin Sampling samples an unselected point in the dataset that has the maximum projection to the set of already sampled points.[2] When there are well-separated outlier micro-clusters, Maximin Sampling guarantees that we sample at least one point per micro-cluster. Furthermore, if all micro-clusters have one subsample already by Maximin Sampling, the projection of the next selected points is expected to have a large decrease, which gives us an estimation of the number of outlier micro-clusters. We stop sampling after such a large decrease so that $\mathcal{H}$ does not include an unnecessarily large number of points from the micro-clusters, to improve algorithm efficiency. We later use such outlier representatives as "anchor points" to identify different micro-clusters.

2) Filtering out False Positives from Micro-cluster Representatives toward Pruning: D.MCA-0 prunes outlier micro-clusters found at each iteration to improve the detection performance. Pruning can reduce one major drawback of iNNE at the presence of outlier micro-clusters, called "masking". Masking occurs when two or more points from the same outlier micro-cluster is subsampled into $\mathcal{R}_\psi$, which causes the members of the micro-cluster to be masked or hidden as inliers. Hence, training with a cleaner set with fewer outlier micro-clusters decreases the probability of "masking" and leads to better detection performance.

However, naïvely pruning points with the topmost outlier scores is not effective as this leads to also pruning false positives, i.e. inlier points that receive high outlier scores, which leads to even worse performance than iNNE (e.g. Fig. 5 in the Appendix). Therefore, we perform pruning by first deciding whether an outlier representative in $\mathcal{H}$ is a false positive, following a careful procedure that leverages what we call "clothes lines" (Lines 12-17), and prune only based on points in $\mathcal{H}$ that are estimated to be true outliers.

The idea is that false positive points are closer to many inlier points compared to those true outliers by definition, and thus the average score of the neighbors of false positive points is expected to be lower as compared to that of the true outliers. We describe the details of clothes-lines, i.e. distinguishing false positives from true positives next.

Computing Clothes-lines and Weighted Sums. To construct the clothes-line for a point $\mathbf{x}$, we first compute the distances from all points in $\mathcal{D}$ to $\mathbf{x}$ (Line 13), and sort the distances from the smallest to the largest to get $\mathcal{L}[\mathbf{x}] \in \mathbb{R}^n$ (Line 14). We then compute the average outlier score of the neighbor points of $\mathbf{x}$ with an increasing distance to $\mathbf{x}$, i.e. we compute

Algorithm 1 D.MCA-0

```
Input: data $\mathcal{D} \in \mathbb{R}^{n \times d}$, subsample size $\psi$, num. iterations $t$
    (default: 100), num. check-points $p$ (default: $0.1n$)
Output: outlier scores $\bar{\mathbf{s}} \in [0,1]^n$, explicit set of outlier
    micro-clusters $\mathcal{C} = \{\mathcal{C}_1, \mathcal{C}_2, \dots\}$, neighbor graph $\mathcal{G}$
 1: Outlier scores $\bar{\mathbf{s}} \leftarrow \mathbf{0} \in \mathbb{R}^n$ (init.)
 2: Neighbor graph $\mathcal{G} \leftarrow (\mathcal{V} = \mathcal{D}, \mathcal{E} = \emptyset)$ (init.)
 3: Clean set $\mathcal{R} \leftarrow \mathcal{D}$ (init.)
 4: for $i = 1, 2, \dots, t$ do
 5:     $\mathbf{s}^{(i)}$, centers $\mathcal{R}_\psi^{(i)} \leftarrow$ iNNE (train: $\mathcal{R}$, test: $\mathcal{D}$, $\psi$: $\psi$)
 6:     $\bar{\mathbf{s}} \leftarrow (\bar{\mathbf{s}} * (i-1) + \mathbf{s}^{(i)})/i$
 7:     High-score set $\mathcal{H}$-top $\leftarrow \{\mathbf{x} \in \mathcal{D} : \bar{\mathbf{s}}[\mathbf{x}]$ among top $p\}$
 8:     Representative set $\mathcal{H} \leftarrow$ MaximinSampling($\mathcal{H}$-top)
 9:     maximum radius $r_{\max} \leftarrow \max_{\mathbf{x} \in \mathcal{H}} \min_{\mathbf{c} \in \mathcal{R}_\psi^{(i)}} \|\mathbf{x} - \mathbf{c}\|_2$
10:     $\mathcal{A} \leftarrow \emptyset$   ▶ # areas under "clothes-lines"
11:     Sorted distances $\mathcal{L} \leftarrow \emptyset$
12:     for $\mathbf{x} \in \mathcal{H} \cup R_\psi^{(i)}$ do   ▶ #clothes-lines
13:         distance Dist$[\mathbf{x}, \mathbf{y}] \leftarrow \|\mathbf{y} - \mathbf{x}\|_2, \forall \mathbf{y} \in \mathcal{D}$
14:         $\mathcal{L}[\mathbf{x}] \leftarrow$ sorted Dist.s to $\mathbf{x}$ from smallest to largest
15:         $\mathbf{a} \leftarrow$ sorted average neighboring outlier scores
16:         $\mathcal{A}[\mathbf{x}] \leftarrow$ weighted sum of $\mathbf{a}$ based on $\mathcal{L}[\mathbf{x}]$ and
            $r_{\max}$ (See Eq. (1))
17:     end for
18:     # Decide whether $\mathbf{x}$ is a true outlier
19:     Candidate true set $\mathcal{H}_c \leftarrow \{\mathbf{x} \in \mathcal{H} : \mathcal{A}[\mathbf{x}] > \text{mean}(\mathcal{A})\}$
20:     Pruning set $\mathcal{P} \leftarrow \emptyset$
21:     for $\mathbf{x} \in \mathcal{H}_c$ do
22:         threshold $\tau_n \leftarrow$ FindFirstPeak($\mathcal{L}[\mathbf{x}]$)
23:         # Find neighbors of $\mathbf{x}$ to prune
24:         Neighbors $\mathcal{N} \leftarrow \{\mathbf{y} \in \mathcal{D} : \|\mathbf{x} - \mathbf{y}\|_2 < \tau_n\}$
25:         $\mathcal{G} \leftarrow$ increase edge weight $(\mathbf{x}, \mathbf{y})$ by 1, $\forall \mathbf{y} \in \mathcal{N}$
26:         $\mathcal{P} \leftarrow \mathcal{P} \cup \mathcal{N}$
27:     end for
28:     Clean set $\mathcal{R} \leftarrow \mathcal{D} \setminus \mathcal{P}$
29: end for
30: $\mathcal{C} \leftarrow$ FindClusters($\mathcal{G}$)   ▶ # Procedure 1
31: return $\bar{\mathbf{s}}, \mathcal{C}, \mathcal{G}$
```

[2]Projection is the minimum distance from a point to a set.

$\mathbf{a} \in \mathbb{R}^n$ (Line 15), such that $\mathbf{a}_i = \sum_{\mathbf{y}:\|\mathbf{y}-\mathbf{x}\| \le \mathcal{L}[\mathbf{x}]_i} \bar{\mathbf{s}}[\mathbf{y}]/i$, $\forall i \in [n]$. The clothes-line of $\mathbf{x}$ is then a curve on a 2D plot where the x-axis is $\mathcal{L}[\mathbf{x}]$ and the y-axis is $\mathbf{a}$. We illustrate in Fig. 2 example clothes-lines at iteration 10 for an outlier (in red) and an inlier (in green) in `synthetic10` (one of our datasets with ground-truth, see Appendix III-A).

Intuitively, for an outlier, the average outlier score of points by increasing distance is large and remains large[3], as such the curve looks flat like a clothes-line. In contrast, for an inlier (i.e. false positive), the average tends to drop fairly quickly (i.e. within a shorter distance away). Then, we use the area under the clothes-lines (shaded areas in Fig. 2) to distinguish between true and false positives, which is computed through a weighted sum of the outlier scores, where the weights are the distances to neighboring points (Line 16). Specifically, the weighted sum for $\mathbf{x}$ is given as

$$\mathcal{A}[\mathbf{x}] := \sum_{i:\mathcal{L}[\mathbf{x}]_i \le r_{\max}} \mathbf{w}_i \cdot (\mathcal{L}[\mathbf{x}]_{i+1} - \mathcal{L}[\mathbf{x}]_i) \cdot \mathbf{a}_i \tag{1}$$

where weight $\mathbf{w}_i = \frac{1}{2}(\mathcal{L}[\mathbf{x}]_{i+1} + \mathcal{L}[\mathbf{x}]_i)$. We weigh the sum by distance to reflect the fact that outliers are separated in distance from the inliers. Points with both large distances to inliers and large average scores of the neighbors receive the largest weighted sum, which makes true outliers well separated from false positives with a larger weighted sum.

Note that we only compute the weighted sum for $\mathbf{x}$ based on all neighbors $\mathbf{y}$ such that $\|\mathbf{x} - \mathbf{y}\|_2 \le r_{\max}$, instead of computing the sum under the entire clothes line. In other words, we only consider the average outlier scores of close neighbors of $\mathbf{x}$ instead of all points in $\mathcal{D}$. Here, $r_{\max}$ represents the distance from $\mathbf{x}$ to the nearest inlier point in R_ψ (i.e. the projection of $\mathbf{x}$ to R_ψ) and is an estimation of the neighboring regions around $\mathbf{x}$ that should be focused. Setting $r_{\max}$ to be the maximum over all projections of $\mathbf{x} \in \mathcal{H}$ to the inlier points gives us a sufficiently large radius that includes neighbors necessary to separate false positive points from true outliers. In fact, the sum under the entire clothes line makes the difference of the weighted sums between false positives and true outliers less apparent (see Fig. 2). Computing the sum up to $r_{\max}$ also makes the algorithm more efficient.

Deciding True Outlier Candidates. We compute the weighted sum for both $\mathbf{x} \in \mathcal{H}_\mathbf{c}$ and $\mathbf{x} \in \mathcal{R}_\psi$, and build a candidate true outliers set as the points with a weighted sum that is greater than the average weighted sum mean($\mathcal{A}$) (Line 19). The weighted sums of $\mathbf{x} \in \mathcal{R}_\psi$, representing such sums of the inlier points, are used as comparison against that of the true outliers to make mean($\mathcal{A}$) a meaningful threshold.

3) Update the Neighbor Graph $\mathcal{G}$: The points in the candidate set $\mathcal{H}_c$ can be seen as refined outlier micro-cluster representatives. Then, the neighbors of $\mathbf{x} \in \mathcal{H}_c$ can be marked as points in the same outlier micro-cluster. To this end, we first estimate the diameter of the micro-cluster that includes $\mathbf{x}$, denoted τ_n, based on $\mathcal{L}[\mathbf{x}]$ (Line 22), and then compute the neighbors of $\mathbf{x}$, denoted $\mathcal{N}$, as points $\mathbf{y} \in \mathcal{D}$ within the diameter, i.e. $\|\mathbf{y} - \mathbf{x}\|_2 \le \tau_n$ (Line 24). The diameter τ_n is found by a peak finding algorithm, detecta[4], on the sorted distances $\mathcal{L}[\mathbf{x}]$. Since outlier micro-clusters are separated in distance from the other points by definition, the first peak in the sorted distance $\mathcal{L}[\mathbf{x}]$ indicates such a gap between the micro-cluster and the rest of the points in $\mathcal{D}$. We store the information of the neighbors $\mathcal{N}$ in graph $\mathcal{G}$ by increasing the edge weight $(\mathbf{x}, \mathbf{y})$ by one, $\forall \mathbf{y} \in \mathcal{N}$ (Line 25).

4) Pruning True Micro-cluster Representatives and Their Neighbors: Finally, we prune the candidate true positive points $\mathbf{x} \in \mathcal{H}_c$ along with their neighbors (Line 28). This effectively prunes the entire micro-cluster that contains $\mathbf{x}$, towards the goal of reducing "masking".

We remark that the synergy between micro-cluster assignment based on detection and leveraging micro-clusters to improve detection via pruning during a single iteration can be seen through the following perspectives: (1) We find outlier representatives $\mathcal{H}$ based on outlier scores (i.e. current detection results); (2) The candidate set $\mathcal{H}_c$ is computed based on both outlier representatives $\mathcal{H}$ and neighboring outlier scores; and (3) The candidate set $\mathcal{H}_c$ is used for both pruning toward improving detection as well as outlier micro-cluster assignment – where we find micro-clusters as neighbors of points $\mathbf{x} \in \mathcal{H}_c$ and prune both $\mathbf{x} \in \mathcal{H}_c$ and their neighbors.

Outputting Outlier Micro-clusters Based on $\mathcal{G}$. At the end of our sequential ensemble, we find outlier micro-clusters based on the neighbor graph $\mathcal{G}$. We first find a threshold τ_e to be the first peak of sorted edge weights (from the largest to the smallest) using detecta[4]. We keep only those strong edges with weights larger than τ_e to get a pruned graph $\mathcal{G}'$. The outlier micro-clusters $\mathcal{C} = \{\mathcal{C}_1, \mathcal{C}_2, \dots\}$ are output as the connected components of $\mathcal{G}'$. See FindClusters in Appendix II.

B. Proposed D.MCA

Thus far, we have proposed D.MCA-0, a sequential ensemble that alternates between outlier detection and micro-cluster assignment. However, there exists a remaining key challenge: the choice of the hyperparameter ψ, i.e. the subsample size, for

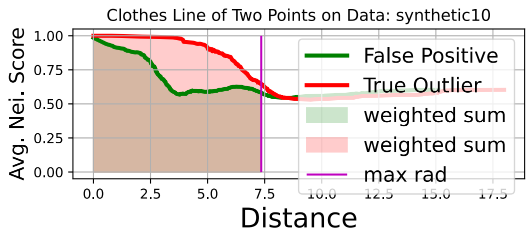


Fig. 2: Example clothes-line of an outlier (red curve) and an inlier (green curve). For an outlier (i.e. true positive), average neighboring scores (y-axis) by distance (x-axis) remains high and flat, and hence looks like a "clothes-line" while the curve for a false positive inlier drops relatively quickly, even if they both receive a high score initially (as shown at distance 0).

[3]This is the case for both scattered and clustered outliers, where for the former the average includes only the point itself for a large radius around it.

[4] detecta: https://github.com/demotu/detecta

the base detector. Although D.MCA-0 improves the robustness to ψ relatively thanks to "cleaning" the dataset via outlier pruning, it remains sensitive to the choice (see e.g. Fig. 4 in Appendix IV-B). In prior works (e.g. [11], [13]), only the best detection performance of iNNE with a chosen hyperparameter ψ via a grid search is reported. However, ψ is far from trivial to set in practice for fully *unsupervised* settings.

On one hand, ψ cannot be smaller than the number of inlier clusters; otherwise, a small $\mathcal{R}_\psi$ underrepresents the inlier distribution, which leads to many false positives as the hyperspheres constructed by a small $\mathcal{R}_\psi$ are unable to cover the inliers effectively. On the other hand, ψ cannot be too large. A large ψ increases the chance of "masking" which leads to false negatives (micro-clusters) and hence overall poor detection. Therefore we ask the question: *Can we make D.MCA-0 even more robust to ψ?*

To this end, we propose D.MCA (Algorithm 2), where the main idea is to take advantage of *hyper*-ensembling to increase hyperparameter robustness to specific settings [14]. As such, D.MCA consists of two phases: Phase 1 is a "warm up" phase (Lines 2-8), consisting of a hyper-ensemble of D.MCA-0 models with varying values of $\psi \in [2, \psi_{\max}]$, and Phase 2 simply employs the D.MCA-0 algorithm one last time on the cleaned dataset using $\psi_{\max}$ (Line 12).

As discussed earlier, a small ψ is more likely to yield false positive errors, whereas a large ψ is to yield false negatives due to "masking". We do not have a mechanism to recover from missed outliers, while the clothes-lines strategy can be applied to filter out some false positives. That is why in Phase 1 we vary $\psi \in [2, \psi_{\max}]$, starting small and gradually increasing it.

The outcome of Phase 1 is a set of outlier micro-clusters by the hyper-ensemble (Line 9), which are pruned (Line 10) before D.MCA-0 is employed in Phase 2, based on which the outlier micro-clusters are updated and the outlier scores are returned. Note that Phase 2 uses a fixed subsample size of $\psi_{\max}$, i.e. the input and also the largest value that Phase 1 has used. Since Phase 2 trains D.MCA-0 on the cleaned dataset, in the absence of most outlier micro-clusters, a larger subsample size poses a smaller risk for "masking".

Algorithm 2 D.MCA

```
Input: data $\mathcal{D}\in\mathbb{R}^{n\times d}$, max subsample size $\psi_{\max}$, num. iterations $t$ (default: 100), num. check-points $p$ (default: $0.1n$)
Output: outlier scores $\bar{\mathbf{s}} \in [0,1]^n$, explicit set of outlier micro-clusters $\mathcal{C} = \{\mathcal{C}_1, \mathcal{C}_2, \dots\}$
1:  Neighbor graph $\mathcal{G} \leftarrow (\mathcal{V} = \mathcal{D}, \mathcal{E} = \emptyset)$ (init.)
2:  # Phase 1: hyperensemble as "warm up"
3:  $t' \leftarrow \lfloor t/2 \rfloor$
4:  $\Psi \leftarrow$ Pick $t'$ values in $\in [2, \psi_{\max}]$ with equal gap
5:  for $i = 1, 2, \dots, t'$ do
6:      $\_, \_, \mathcal{G}^{(i)} \leftarrow$ D.MCA-0 ($\mathcal{D} : \mathcal{D}$, $\psi : \Psi[i]$, $t : i$, $p : p$)
7:      $\mathcal{G} \leftarrow$ increase weight of $(\mathbf{x}, \mathbf{y})$ by edge weight in $\mathcal{G}^{(i)}$
8:  end for
9:  $\mathcal{C}_{\text{warmup}} \leftarrow$FindClusters($\mathcal{G}$)        ▶ # Procedure 1
10: Clean set $\mathcal{R} \leftarrow \mathcal{D} \setminus \mathcal{C}_{\text{warmup}}$
11: # Phase 2: sequential ensemble
12: $\bar{\mathbf{s}}, \_, \mathcal{G}_0 \leftarrow$ D.MCA-0 ($\mathcal{D} : \mathcal{R}$, $\psi : \psi_{\max}$, $t : (t - t')$, $p : p$)
13: $\mathcal{G} \leftarrow \mathcal{G} \cup \mathcal{G}_0$
14: $\mathcal{C} \leftarrow$ FindClusters($\mathcal{G}$)
15: return $\bar{\mathbf{s}}, \mathcal{C}$
```

IV. Experiments

We evaluate D.MCA against 16 datasets and 8 SOTA detectors and 2 hyperparameter-free clustering algorithms w.r.t. both detection and micro-cluster assignment. We also provide an in-depth analysis of its various elements in the Appendix.

Datasets. Our datasets fall into three main categories: 1) *2D synthetic datasets*; 2) *Semi-synthetic datasets*: Datasets `letter`, `musk`, `thyroid`, `optdigits`, and `satimage-2` 3) *Real-world benchmark datasets*: Datasets `lympho`, `ecoli`, `musk_real`, `satellite`, `shuttle`, `smtp`, and `http`. Both *Semi-synthetic datasets* and *Real-world benchmark datasets* come from the ODDS repository[5] with preprocessing that ensures the datasets contain small, compact outlier micro-clusters. See Appendix III-A for more details.

[5]http://odds.cs.stonybrook.edu

Baselines and Configurations. We evaluate both tasks, detection and micro-cluster assignment, with separate experiments; accordingly, we consider two different sets of baselines: *Detection baselines* and *Assignment baselines*. See Appendix III-B and III-C for details on baselines and configurations used in the experiments.

Performance Metrics. 1) *Detection*: We report both the area under the ROC curve (ROC AUC) and the Average Precision (AP). 2) *Assignment*: We report the average F1 score against the true set of outlier micro-clusters. For methods with hyperparameters, we report the *averaged* performance and one stdev over the list of hyperparameter settings and 5 random runs each. For hyperparameter-free algorithms, we report the average performance and one stdev only over 5 random runs.

A. Assignment Evaluation

Here we evaluate the methods w.r.t. outlier micro-cluster assignment. D.MCA, and D.MCA-0 do assignment in-house. Gen^{2}Out has a post-hoc assignment procedure. For the other methods, we post-process the outliers they detect using two off-the-shelf clustering algorithms: X-means, and OPTICS. They are both parameter-free, and well-regarded in the literature. Table I reports the results of this experiment. We use "+O" to indicate post-hoc clustering with OPTICS; "+X" stands for X-means. Note that our proposed methods are notably effective in assignment. D.MCA, and D.MCA-0 obtain respectively the best and the third best average rankings among all datasets. Also, D.MCA is the best or the second best method in 10 out of our 16 datasets; D.MCA-0 does the same for 9 datasets. Finally, note that Gen^{2}Out has the worst average ranking among all 17 methods tested; interestingly, (to the best of our knowledge) it is the only method in literature that is originally designed to perform micro-cluster assignment.

TABLE I: **D.MCA excels in micro-cluster assignment**: we report F1 scores of micro-cluster assignment for our methods D.MCA, and D.MCA-0, and also for 8 state-of-the-art competitors. D.MCA outperforms everyone with the *very best* average ranking; it is the winner or the runner up in 10 out of our 16 datasets. The **first** place is in bold, and second place is underlined.

Dataset \ Method	D.MCA-0	D.MCA	iNNE+O	iNNE+X	LOF+O	LOF+X	kNN+O	kNN+X	iForest+O	iForest+X	SCiForest+O	SCiForest+X	COPOD+O	COPOD+X	HBOS+O	HBOS+X	Gen^{2}Out
`synthetic10`	**0.68 ± 0.47 (1)**	0.67 ± 0.47 (2)	0.41 ± 0.21 (9)	0.35 ± 0.30 (12)	0.32 ± 0.25 (13)	0.31 ± 0.34 (14)	0.49 ± 0.15 (5)	0.29 ± 0.18 (15)	0.43 ± 0.20 (8)	0.61 ± 0.35 (3)	0.53 ± 0.13 (4)	0.46 ± 0.23 (7)	0.25 ± 0.26 (17)	0.41 ± 0.43 (10)	0.39 ± 0.20 (11)	0.47 ± 0.40 (6)	0.29 ± 0.44 (16)
`spiral`	0.68 ± 0.47 (2)	**0.70 ± 0.46 (1)**	0.28 ± 0.27 (8)	0.31 ± 0.38 (5)	0.30 ± 0.27 (7)	0.30 ± 0.35 (6)	0.39 ± 0.29 (4)	0.40 ± 0.38 (3)	0.08 ± 0.17 (14)	0.16 ± 0.36 (12)	0.10 ± 0.21 (13)	0.17 ± 0.37 (10)	0.08 ± 0.17 (14)	0.17 ± 0.37 (10)	0.00 ± 0.00 (16)	0.00 ± 0.00 (16)	0.17 ± 0.37 (10)
`sandwich`	**0.97 ± 0.18 (1)**	0.96 ± 0.20 (2)	0.40 ± 0.26 (10)	0.37 ± 0.30 (11)	0.26 ± 0.28 (14)	0.23 ± 0.30 (15)	0.54 ± 0.13 (5)	0.56 ± 0.32 (4)	0.46 ± 0.26 (7)	0.53 ± 0.39 (6)	0.56 ± 0.14 (3)	0.43 ± 0.16 (8)	0.29 ± 0.31 (13)	0.41 ± 0.42 (9)	0.19 ± 0.27 (16)	0.10 ± 0.16 (17)	0.35 ± 0.35 (12)
`vdensity`	**0.94 ± 0.22 (1)**	0.94 ± 0.23 (2)	0.46 ± 0.22 (11)	0.71 ± 0.39 (3)	0.30 ± 0.27 (16)	0.41 ± 0.42 (12)	0.47 ± 0.23 (10)	0.67 ± 0.41 (5)	0.39 ± 0.24 (14)	0.64 ± 0.42 (7)	0.55 ± 0.09 (8)	0.65 ± 0.27 (6)	0.34 ± 0.25 (15)	0.67 ± 0.47 (4)	0.27 ± 0.27 (17)	0.40 ± 0.44 (13)	0.50 ± 0.50 (9)
`letter`	0.88 ± 0.33 (2)	**0.98 ± 0.15 (1)**	0.36 ± 0.34 (14)	0.46 ± 0.44 (11)	0.26 ± 0.34 (16)	0.34 ± 0.42 (15)	0.42 ± 0.37 (12)	0.46 ± 0.42 (10)	0.63 ± 0.26 (6)	0.52 ± 0.30 (9)	0.63 ± 0.26 (6)	0.79 ± 0.32 (3)	0.63 ± 0.26 (6)	0.76 ± 0.32 (4)	0.63 ± 0.26 (6)	0.42 ± 0.25 (13)	0.00 ± 0.00 (17)
`musk`	0.96 ± 0.19 (2)	**0.99 ± 0.10 (1)**	0.52 ± 0.39 (11)	0.46 ± 0.41 (14)	0.45 ± 0.42 (15)	0.42 ± 0.44 (16)	0.48 ± 0.42 (12)	0.46 ± 0.44 (13)	0.71 ± 0.31 (4)	0.60 ± 0.36 (9)	0.71 ± 0.31 (4)	0.66 ± 0.38 (8)	0.71 ± 0.31 (4)	0.67 ± 0.37 (7)	0.71 ± 0.31 (4)	0.53 ± 0.33 (10)	0.00 ± 0.00 (17)
`thyroid`	0.92 ± 0.27 (3)	0.96 ± 0.20 (2)	0.27 ± 0.32 (15)	0.35 ± 0.43 (14)	0.10 ± 0.21 (17)	0.16 ± 0.34 (16)	0.41 ± 0.32 (13)	0.48 ± 0.43 (12)	0.62 ± 0.15 (9)	0.70 ± 0.32 (7)	0.62 ± 0.15 (9)	0.77 ± 0.28 (6)	0.62 ± 0.15 (9)	0.83 ± 0.26 (5)	0.62 ± 0.15 (9)	0.87 ± 0.21 (4)	**1.00 ± 0.00 (1)**
`optdigits`	0.88 ± 0.33 (2)	**0.96 ± 0.19 (1)**	0.31 ± 0.31 (13)	0.30 ± 0.39 (14)	0.24 ± 0.30 (15)	0.22 ± 0.35 (16)	0.35 ± 0.31 (11)	0.34 ± 0.40 (12)	0.54 ± 0.23 (6)	0.56 ± 0.40 (3)	0.54 ± 0.23 (4)	0.54 ± 0.39 (5)	0.53 ± 0.22 (7)	0.39 ± 0.36 (10)	0.52 ± 0.23 (8)	0.48 ± 0.39 (9)	0.00 ± 0.00 (17)
`satimage-2`	0.45 ± 0.50 (8)	**0.82 ± 0.39 (1)**	0.31 ± 0.35 (13)	0.44 ± 0.47 (9)	0.34 ± 0.37 (12)	0.47 ± 0.47 (7)	0.41 ± 0.36 (10)	0.58 ± 0.45 (6)	0.61 ± 0.27 (5)	0.68 ± 0.32 (3)	0.61 ± 0.27 (4)	0.70 ± 0.29 (2)	0.37 ± 0.35 (11)	0.28 ± 0.34 (14)	0.11 ± 0.18 (15)	0.11 ± 0.23 (16)	0.00 ± 0.00 (17)
`lympho`	0.00 ± 0.00 (16)	0.00 ± 0.00 (16)	0.48 ± 0.27 (8)	0.50 ± 0.27 (6)	0.55 ± 0.30 (4)	0.57 ± 0.23 (3)	0.41 ± 0.32 (11)	0.43 ± 0.33 (10)	0.40 ± 0.35 (12)	0.44 ± 0.36 (9)	0.49 ± 0.37 (7)	0.60 ± 0.35 (2)	0.53 ± 0.28 (5)	**0.64 ± 0.25 (1)**	0.38 ± 0.38 (13)	0.36 ± 0.36 (14)	0.00 ± 0.00 (16)
`ecoli`	0.42 ± 0.42 (2)	0.38 ± 0.41 (4)	0.26 ± 0.29 (11)	0.25 ± 0.29 (13)	0.31 ± 0.27 (9)	0.26 ± 0.23 (12)	0.38 ± 0.35 (6)	0.36 ± 0.33 (7)	0.40 ± 0.31 (3)	0.33 ± 0.24 (8)	**0.46 ± 0.36 (1)**	0.38 ± 0.30 (5)	0.22 ± 0.17 (15)	0.20 ± 0.16 (16)	0.30 ± 0.24 (10)	0.24 ± 0.20 (14)	0.00 ± 0.00 (17)
`musk_real`	0.47 ± 0.17 (9)	0.49 ± 0.20 (8)	0.37 ± 0.24 (11)	0.55 ± 0.39 (4)	0.21 ± 0.23 (14)	0.34 ± 0.41 (12)	0.32 ± 0.25 (13)	0.49 ± 0.41 (7)	0.53 ± 0.05 (5)	**0.87 ± 0.13 (1)**	0.52 ± 0.05 (6)	**0.87 ± 0.13 (1)**	0.07 ± 0.07 (15)	0.07 ± 0.08 (16)	0.40 ± 0.06 (10)	0.65 ± 0.08 (3)	0.00 ± 0.00 (17)
`satellite`	**0.59 ± 0.49 (1)**	0.57 ± 0.49 (2)	0.13 ± 0.27 (15)	0.12 ± 0.30 (16)	0.28 ± 0.43 (8)	0.16 ± 0.27 (14)	0.24 ± 0.38 (12)	0.20 ± 0.37 (13)	0.29 ± 0.37 (7)	0.33 ± 0.47 (4)	0.27 ± 0.39 (10)	0.33 ± 0.47 (4)	0.27 ± 0.39 (10)	0.33 ± 0.47 (4)	0.27 ± 0.39 (10)	0.33 ± 0.47 (4)	0.00 ± 0.00 (17)
`shuttle`	0.14 ± 0.35 (16)	0.21 ± 0.40 (15)	0.40 ± 0.45 (8)	0.25 ± 0.32 (12)	0.37 ± 0.44 (9)	0.24 ± 0.31 (13)	0.55 ± 0.44 (5)	0.34 ± 0.34 (10)	0.60 ± 0.23 (3)	0.64 ± 0.31 (2)	**0.65 ± 0.40 (1)**	0.59 ± 0.37 (4)	0.45 ± 0.32 (7)	0.53 ± 0.33 (6)	0.22 ± 0.22 (14)	0.32 ± 0.37 (11)	0.00 ± 0.00 (17)
`smtp`	0.05 ± 0.13 (8)	0.06 ± 0.16 (6)	0.05 ± 0.09 (7)	0.04 ± 0.07 (10)	0.09 ± 0.15 (5)	0.09 ± 0.19 (4)	0.03 ± 0.05 (16)	0.03 ± 0.05 (13)	0.04 ± 0.07 (9)	0.03 ± 0.05 (13)	0.03 ± 0.05 (17)	0.03 ± 0.05 (13)	0.04 ± 0.06 (10)	0.03 ± 0.05 (13)	**0.25 ± 0.21 (1)**	0.23 ± 0.24 (2)	0.23 ± 0.23 (3)
`http`	0.00 ± 0.06 (17)	0.08 ± 0.27 (16)	0.22 ± 0.33 (5)	0.15 ± 0.30 (12)	0.22 ± 0.31 (4)	0.16 ± 0.29 (10)	0.12 ± 0.30 (14)	0.12 ± 0.30 (13)	0.19 ± 0.36 (7)	0.15 ± 0.34 (11)	0.25 ± 0.35 (2)	0.23 ± 0.33 (3)	**0.34 ± 0.37 (1)**	0.21 ± 0.32 (6)	0.18 ± 0.37 (8)	0.10 ± 0.26 (15)	0.17 ± 0.37 (9)
Avg. Rank	5.69	**5.00**	10.56	10.38	11.12	11.56	9.94	9.56	7.44	6.69	6.19	5.44	9.94	8.44	10.50	10.44	13.25

TABLE II: **D.MCA is competitive in detection**: we report Average Precision (AP) scores for 10 detectors. Our D.MCA obtains the second best average ranking. Note that the best detector Gen^{2}Out is actually the *worst* method in assignment; therefore, we have a much better balance w.r.t. these two tasks. The **first** place is in bold, and second place is underlined.

Dataset \ Method	D.MCA-0	D.MCA	iNNE	iForest	SCiForest	LOF	kNN	COPOD	HBOS	Gen^{2}Out
`synthetic10`	0.86 ± 0.17 (4)	0.88 ± 0.14 (2)	0.73 ± 0.29 (7)	0.71 ± 0.27 (8)	0.87 ± 0.29 (3)	0.57 ± 0.39 (9)	0.79 ± 0.30 (5)	0.53 ± 0.00 (10)	0.76 ± 0.00 (6)	**0.95 ± 0.00 (1)**
`spiral`	0.58 ± 0.37 (2)	0.57 ± 0.38 (3)	0.46 ± 0.38 (5)	0.16 ± 0.08 (8)	0.17 ± 0.06 (7)	0.57 ± 0.44 (3)	**0.68 ± 0.45 (1)**	0.03 ± 0.00 (10)	0.04 ± 0.00 (9)	0.22 ± 0.01 (6)
`sandwich`	0.98 ± 0.04 (2)	**1.00 ± 0.01 (1)**	0.67 ± 0.36 (6)	0.51 ± 0.31 (7)	0.81 ± 0.34 (5)	0.43 ± 0.36 (8)	0.84 ± 0.24 (4)	0.36 ± 0.00 (9)	0.27 ± 0.00 (10)	0.90 ± 0.01 (3)
`vdensity`	0.98 ± 0.07 (2)	0.98 ± 0.07 (2)	0.79 ± 0.30 (5)	0.59 ± 0.29 (8)	0.83 ± 0.28 (4)	0.47 ± 0.39 (10)	0.77 ± 0.33 (6)	0.69 ± 0.00 (7)	0.52 ± 0.00 (9)	**1.00 ± 0.00 (1)**
`letter`	0.89 ± 0.20 (5)	0.93 ± 0.11 (4)	0.64 ± 0.36 (9)	0.88 ± 0.31 (6)	0.88 ± 0.31 (6)	0.51 ± 0.34 (10)	0.71 ± 0.41 (8)	**1.00 ± 0.00 (1)**	**1.00 ± 0.00 (1)**	**1.00 ± 0.00 (1)**
`musk`	0.93 ± 0.14 (4)	0.93 ± 0.12 (4)	0.75 ± 0.34 (8)	0.88 ± 0.31 (7)	0.89 ± 0.30 (6)	0.66 ± 0.40 (10)	0.70 ± 0.42 (9)	**1.00 ± 0.00 (1)**	**1.00 ± 0.00 (1)**	**1.00 ± 0.00 (1)**
`thyroid`	0.92 ± 0.11 (5)	0.94 ± 0.09 (4)	0.48 ± 0.38 (9)	0.90 ± 0.29 (6)	0.90 ± 0.29 (6)	0.26 ± 0.26 (10)	0.73 ± 0.38 (8)	**1.00 ± 0.00 (1)**	**1.00 ± 0.00 (1)**	**1.00 ± 0.00 (1)**
`optdigits`	0.88 ± 0.19 (7)	0.94 ± 0.10 (4)	0.60 ± 0.39 (9)	0.90 ± 0.29 (5)	0.90 ± 0.29 (5)	0.48 ± 0.34 (10)	0.70 ± 0.42 (8)	**1.00 ± 0.00 (1)**	**1.00 ± 0.00 (1)**	**1.00 ± 0.00 (1)**
`satimage-2`	0.95 ± 0.06 (2)	0.93 ± 0.09 (3)	0.57 ± 0.36 (8)	0.58 ± 0.38 (6)	0.88 ± 0.29 (4)	0.58 ± 0.42 (6)	0.70 ± 0.42 (5)	0.53 ± 0.00 (9)	0.33 ± 0.00 (10)	**0.97 ± 0.01 (1)**
`lympho`	0.67 ± 0.14 (6)	0.67 ± 0.15 (6)	0.63 ± 0.16 (8)	0.54 ± 0.31 (9)	0.72 ± 0.35 (3)	0.68 ± 0.25 (5)	0.53 ± 0.10 (10)	**0.88 ± 0.00 (1)**	0.86 ± 0.00 (2)	0.72 ± 0.08 (3)
`ecoli`	0.54 ± 0.20 (3)	0.55 ± 0.21 (2)	0.40 ± 0.26 (5)	0.18 ± 0.19 (10)	0.39 ± 0.31 (6)	0.39 ± 0.18 (6)	**0.67 ± 0.17 (1)**	0.25 ± 0.00 (9)	0.44 ± 0.00 (4)	0.29 ± 0.05 (8)
`musk_real`	0.78 ± 0.37 (3)	0.76 ± 0.39 (4)	0.71 ± 0.42 (5)	0.33 ± 0.42 (9)	0.65 ± 0.37 (6)	0.37 ± 0.45 (8)	0.60 ± 0.44 (7)	0.06 ± 0.00 (10)	0.88 ± 0.00 (2)	**1.00 ± 0.00 (1)**
`satellite`	0.66 ± 0.28 (6)	0.69 ± 0.23 (5)	0.35 ± 0.34 (9)	0.61 ± 0.32 (7)	0.73 ± 0.24 (4)	0.22 ± 0.29 (10)	0.42 ± 0.40 (8)	0.77 ± 0.00 (3)	**0.82 ± 0.00 (1)**	0.81 ± 0.01 (2)
`shuttle`	0.36 ± 0.16 (6)	0.36 ± 0.16 (6)	0.34 ± 0.20 (8)	0.62 ± 0.32 (4)	0.76 ± 0.27 (2)	0.20 ± 0.15 (10)	0.31 ± 0.18 (9)	0.53 ± 0.00 (5)	0.72 ± 0.00 (3)	**0.79 ± 0.04 (1)**
`smtp`	0.12 ± 0.10 (4)	0.12 ± 0.09 (4)	0.13 ± 0.09 (3)	0.08 ± 0.06 (8)	0.12 ± 0.04 (4)	0.12 ± 0.14 (4)	0.04 ± 0.02 (10)	0.07 ± 0.00 (9)	**0.33 ± 0.00 (1)**	0.17 ± 0.00 (2)
`http`	0.14 ± 0.05 (7)	0.15 ± 0.05 (6)	0.16 ± 0.05 (4)	0.10 ± 0.07 (9)	**0.20 ± 0.09 (1)**	**0.20 ± 0.11 (1)**	0.08 ± 0.05 (10)	0.11 ± 0.00 (8)	0.19 ± 0.00 (3)	0.16 ± 0.02 (4)
Avg. Rank	4.25	3.75	6.75	7.31	4.50	7.50	6.81	5.87	4.00	**2.31**

B. Detection Evaluation

The results of detection in AP scores are in Table II. See Appendix IV for results in AUC scores. As it can be seen, our methods are quite competitive in terms of detection. Note that D.MCA and D.MCA-0 are respectively the second and the fourth best methods in the average ranking of AP, while a similar pattern is observed in the AUC scores. The best performing method is Gen^{2}Out. It is quite surprising as Gen^{2}Out is the worst performing method w.r.t. assignment (Table I). A similar scenario is also seen for HBOS, which is both one of the top performing detectors, yet one of the worst performing in assignment. These results make it clear that our proposal is undoubtedly the best option among all the 10 methods studied; D.MCA and D.MCA-0 are the only methods that excel both in detection *as well as* in assignment.

References

[1] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in *ICDM*, 2008, pp. 413–422.

[2] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "LOF: Identifying density-based local outliers," *SIGMOD*, vol. 29, no. 2, p. 93–104, 2000.

[3] D. Pelleg and A. W. Moore, "X-means: Extending k-means with efficient estimation of the number of clusters." in *ICML*, 2000, pp. 727–734.

[4] M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. Sander, "OPTICS: Ordering points to identify the clustering structure," *SIGMOD*, vol. 28, no. 2, pp. 49–60, 1999.

[5] M. Macha and L. Akoglu, "Explaining anomalies in groups with characterizing subspace rules." *ACM DAMI*, vol. 32, no. 5, 2018.

[6] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "On detecting clustered anomalies using sciforest," in *Mach. Learning and Knowledge Disc. in Databases*. Berlin, Heidelberg: Springer, 2010, pp. 274–290.

[7] G. D. Silva, L. Akoglu, and R. L. Cordeiro, "C-allout: Catching & calling outliers by type," *arXiv preprint arXiv:2110.08257*, 2021.

[8] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM computing surveys (CSUR)*, vol. 41, no. 3, pp. 1–58, 2009.

[9] M.-C. Lee, S. Shekhar, C. Faloutsos, T. N. Hutson, and L. Iasemidis, "gen2Out: detecting and ranking generalized anomalies," in *2021 IEEE International Conference on Big Data (Big Data)*. IEEE, 2021.

[10] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *KDD*, ser. KDD'96. AAAI Press, 1996, p. 226–231.

[11] T. Bandaragoda, K. Ting, D. Albrecht, F. T. Liu, Y. Zhu, and J. Wells, "Isolation-based anomaly detection using nearest-neighbor ensembles: inne," *Computational Intelligence*, vol. 34, 01 2018.

[12] O. A. Ibrahim, J. Keller, J. C. Bezdek, and M. Popescu, "Experiments with maximin sampling," in *2020 IEEE Inter. Conf. on Fuzzy Sys.*, 2020.

[13] K. M. Ting, B.-C. Xu, T. Washio, and Z.-H. Zhou, "Isolation distributional kernel: A new tool for kernel based anomaly detection," in *KDD*. New York, NY, USA: ACM, 2020, p. 198–206.

[14] X. Ding, L. Zhao, and L. Akoglu, "Hyperparameter sensitivity in deep outlier detection: Analysis and a scalable hyper-ensemble solution," *NeurIPS*, 2022.