Efficient Generation of Hidden Outliers for Improved Outlier Detection

Jose Cribeiro-Ramallo[1], **Vadim Arzamasov**[1], **Klemens Böhm**[1]

[1]Karlsruhe Institute of Technology {jose.cribeiro, vadim.arzamasov, klemens.boehm}@kit.edu

Abstract

Outlier generation is a popular technique used for solving important outlier detection tasks. Generating outliers with realistic behavior is challenging. Popular existing methods tend to disregard the "multiple views" property of outliers in high-dimensional spaces. The only existing method accounting for this property falls short in efficiency and effectiveness. We propose BISECT, a new outlier generation method that creates realistic outliers mimicking said property. To do so, BISECT employs a novel proposition introduced in this article stating how to efficiently generate said realistic outliers. Our method has better guarantees and complexity than the current methodology for recreating "multiple-views". We use the synthetic outliers generated by BISECT to effectively enhance outlier detection in diverse datasets, for multiple use cases. For instance, oversampling with BISECT reduced the error by up to 3 times when compared with the baselines.

1 Introduction

Motivation. Outliers are observations in a dataset that stand out from the rest. They represent rare or surprising events, making outlier detection important in many applications (Pawar, Kalavadekar, and Tambe 2014; Aggarwal 2013). In a nutshell, outlier detection can be approached in two ways. The conventional approach is to use one outlier detection model fitted to all dimensions, the *full-space approach*. But according to the "multiple views" property (Müller et al. 2012), points often are outliers only in some sets of dimensions (subspaces). Together with the curse of dimensionality, this often renders the approach ineffective in high-dimensional data (Aggarwal, Hinneburg, and Keim 2001a; Keller, Muller, and Bohm 2012). A very common alternative approach uses an ensemble of outlier detection models trained on several smaller subspaces (Ho 1998; Aggarwal and Sathe 2017), dubbed *subspace approach*. We will build on this distinction in what follows.

Synthesizing outliers may improve outlier detection (El-Yaniv and Nisenson 2006). For instance, one may present synthetic outliers to a domain expert for labeling and later refine the outlier detection model using their feedback (Steinbuss and Böhm 2017). Next, utilizing synthetic outliers allows to reframe outlier detection as a classification problem (Liu et al. 2019). However, if objects generated are too far from the inliers, not representative of the outlier class, or even if there are too many artificial samples (Liu et al. 2019; Steinbuss and Böhm 2021), the classification boundary degrades (Hempstalk, Frank, and Witten 2008). This calls for a "careful" generation procedure, as opposed to a random one.

Most outlier generation approaches fall into one of two groups. *Original space generators* (El-Yaniv and Nisenson 2006; Abe, Zadrozny, and Langford 2006) generate outliers using the original domain of the data. *Embedded space generators* (Liu et al. 2019; Schlegl et al. 2017) create artificial outliers using a representation of the embedded latent space of the data. When the dimensionality is high, original space generators in particular tend to create outliers arbitrarily distant from the inliers (Steinbuss and Böhm 2021; Aggarwal, Hinneburg, and Keim 2001b). Embedded-space generators in turn produce samples in a lower-dimensional space to solve this problem. But coming up with an embedding model that accurately represents the domain is challenging because of the need for specific assumptions that are difficult to verify (van der Maaten, Postma, and Herik 2007), or because a lot of data is required (Sadr, Bassett, and Kunz 2019). So these models tend to be unsuitable if not enough is known about the data. In either case, synthetic outliers may not demonstrate the common "multiple view" characteristic of high-dimensional outliers (Müller et al. 2012), thus failing to replicate the outlier class.

Hidden outliers (Steinbuss and Böhm 2017) are outliers that represent the disagreement between full-space and subspace outlier detection approaches: By definition, hidden outliers are detectable by either a full-space approach or a subspace approach, but not both. Hence, they feature the "multiple views" property and at the same time tend to be close to the inliers. This is because distant points tend to be detectable by both approaches and thus are not hidden.

Example 1. *In Figure 1, the shaded areas represent regions where hidden outliers may occur. The solid ellipse line delimits the boundary of the two-dimensional full-space outlier detector, while the dashed lines constitute the boundary of the subspace detector, composed of detectors fitted in each one-dimensional subspace. Points in the shaded areas on the left can only be detected by the subspace detector, and the ones on the right by the full-space detector.*

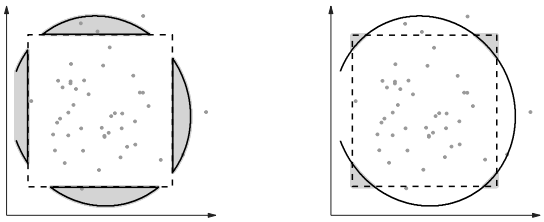


Figure 1: Example of regions of hidden outliers.

Challenges. Despite their desirable properties, generating hidden outliers is not trivial. First, there are no known guarantees that hidden outliers can be generated. Next, even if they can, it is generally impossible to describe the regions of the space from which they originate analytically and sample from there. It is necessary to generate candidate points and verify which ones qualify as hidden outliers. However, generating too many poor candidates renders computations intractable. Namely, the definition of hidden outliers builds on subspace outliers, while subspace outlier detection uses up to $2^N - 1$ models, one for each subspace, where N is the number of dimensions. Another issue is that an ill-designed candidate generation process could leave parts of the space unexplored. Finally, quantitative evaluation criteria for hidden outliers currently do not exist. All this calls for an efficient generation method and a procedure to assess its utility. The only existing generator for hidden outliers, HIDDEN (Steinbuss and Böhm 2017), heavily relies on its hyperparameter which one cannot readily tune, as we will explain. It is unclear whether a value of this hyperparameter exists that ensures both efficient synthesis and high quality of hidden outliers.

Contributions Our contributions are as follows. (1) We prove the existence of hidden outliers. In particular, we formulate and prove the so-called hidden outlier existence proposition which asserts that it is always possible to generate a hidden outlier between an inlier and an outlier of the full-space approach. (2) Leveraging this proposition, we propose a new method to efficiently generate hidden outliers, which we call BISECT. Importantly, BISECT gives certain guarantees in generating hidden outliers and does not require any external hyperparameters. (3) We propose a methodology to improve outlier detection using hidden outliers. We apply it to various datasets to showcase the value of hidden outliers synthesized by BISECT. There are certain obstacles that are in the way of very general claims regarding any potential superiority of BISECT for outlier detection, as we will explain in Sections 4 and 5. Having said this, we observed significantly reduced error rates in one-class classification and supervised outlier detection in our experiments compared to established methods. (4) We share the code of BISECT and of our experiments.[1]

[1]https://github.com/jcribeiro98/Bisect

2 Related Work

Outlier Generation. There exists an extensive body of work on synthetic outlier generation in tabular data (Steinbuss and Böhm 2021). Some of these methods do (Oh, Hong, and Baek 2019), while others do not (Liu et al. 2019), require genuine outliers. One approach is to sample points from a distribution distinct from that of inliers. Other representative approaches include adding noise to the inliers or permuting their attribute values. For a comprehensive review and taxonomy of outlier generation methods, we refer to (Steinbuss and Böhm 2021). However, the vast majority of these methods do not verify the "multiple-view" property, and may produce outliers not accurately representing the outlier class.

Generation of Hidden Outiers. To our knowledge, there is only one method, HIDDEN (Steinbuss and Böhm 2017), that specifically replicates the "multiple views" property by generating hidden outliers. At each iteration, HIDDEN randomly samples a candidate point from a hypercube centered at a random genuine point and checks whether this candidate is a hidden outlier. The lateral size of each hypercube is $\varepsilon \cdot \mathit{range}$, where *range* represents the maximum range of values among the genuine data points in any dimension. The hyperparameter $\varepsilon \in [0, 1]$ controls the size of each hypercube. However, choosing an appropriate ε value can be challenging. If values of ε are too small, the generation of hidden outliers may fail. Values close to 1 in turn often lead to inefficiencies due to the generation of many poor candidates. There is no recommended universal value for ε: An ε value that is efficient for certain datasets might not work well for different ones. Additionally, different ε values result in distinct and uncontrollable probability distributions of generated hidden outliers, potentially leaving out important regions in the data space. See Appendix for demonstrations. Our method, BISECT, does not rely on such a hyperparameter and exhibits more predictable behavior. We will compare BISECT with HIDDEN.

Evaluation of Artificial Outliers. We are not aware of any established methodology to evaluate hidden outliers. The paper (Steinbuss and Böhm 2017) primarily focuses on using hidden outliers for exploratory purposes. To demonstrate the utility of BISECT and its advantages over HIDDEN, we leverage evaluation techniques from outlier generation in general. Synthetic outliers have several applications within outlier detection (Steinbuss and Böhm 2021). Common use cases in the literature are self-supervised and supervised outlier detection (Liu et al. 2019; Dlamini and Fahim 2021).

Self-supervised outlier detection uses synthetic outliers as the positive class to construct a two-class classifier. Subsequently, this classifier is applicable in various tasks (Chandola, Banerjee, and Kumar 2009), particularly in one-class classification. The effectiveness of the generated outliers determines the performance of the resulting classifier.

The supervised case for outlier detection is akin to an unbalanced binary classification problem with an extremely scarce minority class (Aggarwal 2013). To address imbalanced data, a common technique is oversampling the minority class (Chawla et al. 2002; Oh, Hong, and Baek 2019;

Zheng et al. 2020). This has also been applied in the supervised outlier detection case (Dlamini and Fahim 2021; Sharma et al. 2018). The quality of the generated data depends on the performance gain achieved compared to the unbalanced classifier.

Oversampling and self-supervised learning are among the most frequently used applications of synthetic outliers in the literature (Khan and Madden 2014; Chandola, Banerjee, and Kumar 2009). Furthermore, theoretical studies have demonstrated that self-supervised classifiers are particularly well-suited for one-class classification (El-Yaniv and Nisenson 2006). Hence, we will evaluate synthetic hidden outliers with these tasks.

3 Our Method: BISECT

This section presents BISECT, a novel method for generating hidden outliers. We start with the necessary definitions and derive new theoretical results regarding the existence of regions with hidden outliers. Subsequently, we explore how these findings can be used to generate hidden outliers effectively. Finally, we comprehensively describe BISECT and analyze its properties. All propositions and theorems have their corresponding proofs in the appendix.

3.1 Hidden Outliers and the Hidden Region

Let $X = \mathbb{R}^d$ be a metric space, and $D = \{x^i\} \subset X$ be a finite dataset. Let $\mathcal{M}(\cdot) : X \longrightarrow \{0, 1\}$ denote an outlier detector model, and let $\mathcal{M} = \mathcal{M}(D)$ be its fitted version with D. Next, let the closed set $R(\mathcal{M}) = \{x \in X | \mathcal{M}(x) = 0\}$ be the acceptance region (i.e., the one containing inliers) of $\mathcal{M}$ with the boundary $\partial R(\mathcal{M})$. Further, let $\mathcal{M}_S : S \subset X \longrightarrow \{0, 1\}$ be a detector fitted with the projected dataset over the subspace $S \subset X$, $D|_S = \{x^i|_S\}$. For $\mathcal{M}(\cdot)$ and the set of subspaces $\{S_i\}$, we define the subspace ensemble of $\mathcal{M}(\cdot)$ as the mapping $\mathcal{E}_\mathcal{M} : X \longrightarrow \{0, 1\}$, such as $\mathcal{E}_\mathcal{M}(x) = a\left(\mathcal{M}_{S_1}(x|_{S_1}), \ldots, \mathcal{M}_{S_m}(x|_{S_m})\right)$, where a is an aggregation function. We now formally define hidden outliers and hidden region:

Definition 1. *(Hidden region, hidden outlier, adversary) Given a fitted outlier detector $\mathcal{M}$, a subspace ensemble $\mathcal{E}_\mathcal{M}$, and aggregation function $a = max$, define the sets:*

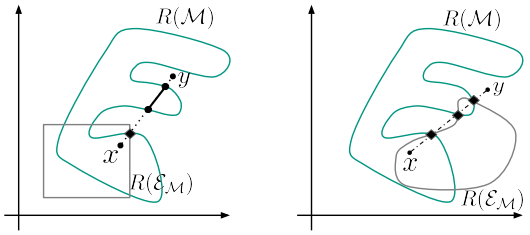
$$H_1(\mathcal{M}) = R(\mathcal{M}) \setminus R(\mathcal{E}_\mathcal{M}),$$
$$H_2(\mathcal{M}) = R(\mathcal{E}_\mathcal{M}) \setminus R(\mathcal{M}).$$

The union $H_1(\mathcal{M}) \cup H_2(\mathcal{M})$ is the hidden region of $\mathcal{M}$. A hidden outlier is a point $h \in X$ in the hidden region of $\mathcal{M}$. The outlier detector $\mathcal{M}(\cdot)$ used is the adversary of h.

For instance, in Figure 1, the shaded area on the left plot is H_1, and H_2 is the shaded area on the right.

By this definition, hidden outliers are points that are outliers in certain subspaces of X but inliers in the original space, or vice-versa. This implies that hidden outliers necessarily exhibit the "multiple views" property when $\{S_i\}$ are generated with the canonical basis of X.

Without a guarantee that a hidden region exists, effectively searching for good candidates is hard. The following proposition addresses this concern and provides guidance on where one could generate hidden outliers. The conditions presented are sufficiently versatile to occur in real scenarios.



(a) (b)

Figure 2: Examples of different hidden regions. $R(\mathcal{M})$ is marked in green, $R(\mathcal{E}_\mathcal{M})$ in grey, and the convex combination of x and y is represented with a dashed line.

Proposition 1. *("Hidden outlier existence"): Let x and y be points in the previously defined metric space such that $x \in R(\mathcal{M})$ and $y \notin R(\mathcal{M})$. Assume that there exists a point z in the convex combination of x and y such as $z \in \partial R(\mathcal{M}) \Rightarrow z \notin \partial R(\mathcal{E}_\mathcal{M})$. Then there exists z' in the convex combination of x and y such as $z' \in H_1(\mathcal{M}) \cup H_2(\mathcal{M})$.*

The existence of z condition in the proposition guarantees that the convex combination crosses the hidden region.

Example 2. *In Figure 2 the dashed line marks the convex combination of x and y. We marked with a straight line the intersection of the convex combination with the hidden region. The condition in Proposition 1 rules out cases like the one in Figure 2b, where there is no intersection with the hidden region. Cases like in Figure 2a that cross the intersection of boundaries but contain hidden outliers are accepted.*

We can limit the search for candidate hidden outliers to the convex combination of two points. This shifts the problem from searching in X to searching in the image of $\alpha(t) = ty + (1 - t)x$, i.e., in the set of points $ty + (1 - t)x$ for $t \in [0, 1]$. The following section elaborates on this.

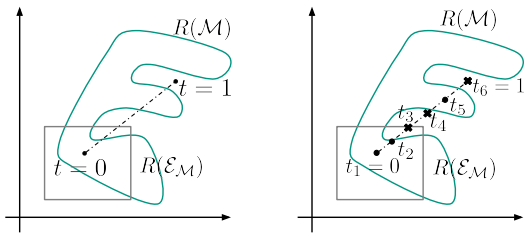
3.2 Generating Hidden Outliers by Finding Roots

We reformulate $\alpha(t)$ so that the points of the image that are hidden outliers are now its zeroes. This will allow us to use the bisection method to generate hidden outliers. Let F be an indication function, such as:

$$F(x) = \begin{cases} 1, & \text{if } \mathcal{M}(x) = \mathcal{E}_\mathcal{M}(x) = 1, \\ 0, & \text{if } \mathcal{M}(x) \neq \mathcal{E}_\mathcal{M}(x), \\ -1, & \text{if } \mathcal{M}(x) = \mathcal{E}_\mathcal{M}(x) = 0. \end{cases}$$

The roots of the function $f = (F \circ \alpha)(t)$ are $t \in [0, 1]$ such that $\alpha(t)$ are hidden outliers. To ensure convergence of the bisection method to a root of f, f must be monotonic. The following theorem states that f is monotonic when the image of α crosses the hidden region once.

Theorem 1. *("Convergence into a hidden outlier") Let f be defined as before. Assume that at most exist, and are unique, z and z' in the convex combination of $x \in R(\mathcal{M})$ and $y \notin R(\mathcal{M})$ such as: $z \in \partial R(\mathcal{M})$, $z' \in \partial R(\mathcal{E}_\mathcal{M})$,*



(a) Before cuts (b) After cuts

Figure 3: An example of the "cut trick" .

$z \neq z'$, and both verify the last condition of proposition 1. Then the bisection method will converge to a root of f.

The theorem straightforwardly extends to the case when the image of α crosses the hidden region a finite number of times. This is because segmenting the interval $[0, 1]$ sufficiently many times yields subintervals that fulfill the conditions in Theorem 1.

Example 3. *Figure 3 illustrates the process, which we will term the "cut trick." In the right plot, six equally spaced points denoted as $t_1, \ldots, t_6 \in [0, 1]$ divide the original line segment (on the left plot) into five equal-length parts. Dots represent inliers (in $R(\mathcal{M})$), while crosses represent outliers (not in $R(\mathcal{M})$). Although the original image for $t \in [0, 1]$ does not satisfy the conditions of the theorem, the segments corresponding to $[t_2, t_3]$, $[t_4, t_5]$, and $[t_5, t_6]$ do meet these conditions. Consequently, the bisection method can generate hidden outliers within each of these segments.*

3.3 The BISECT Algorithm

Combining all the above results, we propose BISECT (Algorithm 1). It works with the following steps:

1. Randomly select an origin, $\tilde{o} \in R(\mathcal{M}) \cap D$. If $\mathcal{M}$ is a composition of two functions, $m : X \to \mathbb{R}$, a scoring function, and $\mathcal{T} : \mathbb{R} \to \{0, 1\}$, a threshold function, BISECT selects $\tilde{o}$ using the weighted distribution obtained by the scoring of m. We do this as we found it to be superior to random selection during preliminary experiments.

2. Select a random direction v in $\mathbb{S}^{d-1}(\mathbb{R})$, the d-dimensional unitary sphere. Define a sufficiently large $L \in \mathbb{R}$ so that the point $y = \tilde{o} + Lv$ is not in $R(\mathcal{M})$ (Lines 1–3). If $y \in R(\mathcal{M})$, restart BISECT with a new origin (Lines 5–7).
3. Obtain a subinterval of the convex combination $\alpha(t) = ty + (1 - t)\tilde{o}$ using the "cut trick" (Line 4).
4. For this subinterval, perform the bisection method for $f = F \circ \alpha$ to obtain one hidden outlier (Lines 9–14).

We now discuss the complexity of BISECT.

Proposition 2. *Let n_{subs} be the number of selected subspaces for the ensemble. The worst-case complexity of* BISECT *is $\mathcal{O}\left(\log(\frac{L}{n_{cuts}}) \cdot (n_{subs} + n_{cuts}) \cdot \aleph\right)$ where $\aleph$ is the inference complexity of the adversary $\mathcal{M}(\cdot)$.*

Algorithm 1: BISECT

Require: $\tilde{o}$, n_{cuts}, and $\mathcal{M}$ & $\mathcal{E}_\mathcal{M}$ fitted with D.
1: $v \leftarrow Unif(1, \mathbb{S}^{d-1}(\mathbb{R}))$
2: $l \leftarrow \max_{x \in D} \|x\|$
3: $L = l + \text{Unif}\left(1, (-\frac{l}{2}, l)\right)$
4: $\{a, b\} \leftarrow \text{INTERVAL}(\tilde{o}, v, 0, L, n_{\text{cuts}})$
5: **if** $\{a, b\} = \emptyset$ **then**
6: Select new origin $\tilde{o}'$
7: BISECT$(\tilde{o}', n_{\text{cuts}}, \mathcal{M}, \mathcal{E}_\mathcal{M})$
8: $c \leftarrow \frac{a+b}{2}$
9: **while** $F(c) \neq 0$ **do**
10: **if** $F(a) = F(c)$ **then**
11: $a \leftarrow c$
12: **else**
13: $b \leftarrow c$
14: $c \leftarrow \frac{a+b}{2}$
15: **return** c

Algorithm 2: INTERVAL (cut trick)

Require: $\tilde{o}, v, a, b, n_{\text{cuts}}$
1: Initialize $I = \emptyset$
2: $\{t_i\}_{i=1}^{n_{\text{cuts}}+1} \leftarrow \text{sequence}(a, b, n_{\text{cuts}})$
3: **for** i in 1 to n_{cuts} **do**
4: check$_i \leftarrow \mathcal{M}(\tilde{o} + t_i v)$
5: **if** check$_i \neq$check$_{i-1}$ **then**
6: Add $\{t_{i-1}, t_i\}$ to I
7: $\{l, r\} \leftarrow Unif(1, I)$
8: **return** l, r

According to this proposition, the parameter n_{cuts} used for the cut trick affects the complexity of BISECT. In practice, low values of n_{cuts} are sufficient, as we now explain.

Selecting the Number of Cuts. The number of iterations needed for the bisection method to generate a hidden outlier in the worst case can be calculated as $n_{\text{iter}} = Int(\frac{\log(L/n_{\text{cuts}}) - \log(Err)}{\log(2)} - 1)$. Where Err is the error rate and $Int(\cdot)$ is the mapping to the nearest integer. For $Err = 0.05$ and a unitary interval, five cuts on the interval are the smallest number of cuts that results in only a single iteration needed to converge. Since we have found through experimentation that five cuts are usually enough to satisfy the conditions of Proposition 1, we recommend setting $n_{\text{cuts}} = 5$.

Complexity Comparison to HIDDEN. For a fixed n_{cuts}, $\mathcal{O}\left(n_{\text{samp}} \cdot \log(L) \cdot n_{\text{subs}} \cdot \aleph\right)$ is the worst case complexity of BISECT to generate n_{samp} hidden outliers. The corresponding worst-case complexity of HIDDEN is $\mathcal{O}\left(n_{\text{samp}} \cdot \mathcal{P}(D, \varepsilon) \cdot n_{\text{subs}} \cdot \aleph\right)$, where the parameter $\mathcal{P}(D, \varepsilon)$ inversely correlates with the probability of generating a hidden outlier for a given data at a particular ε (Steinbuss and Böhm 2017). It is impossible to estimate this parameter beforehand. We expect that in practice it will be greater and more irregular than $\log(L)$.

# Clusters	# Features	# Observations
1, 2, 5	$7, 15, 30, 50, 100, 150$	1000

Table 1: Characteristics of synthetic datasets.

Dataset	# Features	$\lvert D^{full} \rvert$	# Outliers
Wilt	5	4819	261
Pima	8	768	268
Stamps	9	340	31
PageBlocks	10	5473	560
Heart Disease	13	270	120
Annthyroid	21	7129	534
Cardiotocography	21	2114	471
Parkinson	22	195	147
Ionosphere	32	351	126
WPBC	33	198	47
SpamBase	57	4207	1813
Arrhythmia	259	450	206

Table 2: Characteristics of real datasets.

4 Experiments

In this section, we empirically demonstrate the utility of hidden outliers generated with BISECT. First, we describe the synthetic and real datasets and configurations of hidden outlier generation methods for our experiments. Next, we compare the runtime performance of BISECT and HIDDEN on datasets with varying complexity. After that, we study how various outlier detection tasks can benefit from generating hidden outliers. To finalize, we briefly comment on the limitations of the experimental study at the end of the section. We implemented the experiments in R and Python[2] and run them on a ThinkPad P14s-gen2 with 16GB of RAM using Ubuntu v22.04.1. For all calculations, used the CPU, a Ryzen PRO 7 5850u.

4.1 Datasets

Synthetic Data. To comprehensively evaluate hidden outlier generation methods under controlled conditions, we produce several synthetic datasets using a multidimensional clustered Gaussian distribution. We systematically varied the number of clusters, features (columns), and observations (rows). Table 1 provides an overview of the dataset characteristics; for each combination of parameters, we randomly generated five synthetic datasets.

Real Data. We use real datasets[3] collected by Campos et al. for evaluating unsupervised outlier detection. To ensure a reasonable runtime, we only consider datasets containing fewer than 10,000 observations and fewer than 1,000 features. To ensure reliable performance estimates, we only retain datasets with more than 30 outliers. Table 2 summarizes the resulting 12 datasets, referred to as D^{full}. We also use modified versions of these datasets, denoted as D, where the outlier class is downsampled to 2% of the total number of observations. Whenever possible, we obtain D from (Campos et al. 2016). For the Ionosphere and WPBC datasets, we created the downsampled versions ourselves.

[2]Using `reticulate` (Ushey, Allaire, and Tang 2023).

[3]https://www.dbs.ifi.lmu.de/research/outlier-evaluation/

4.2 Method Configurations

We use BISECT and HIDDEN with LOF (Breunig et al. 2000) as an adversary. In some experiments, we also tried the KNN outlier detection method (Angiulli and Pizzuti 2002) as an alternative adversary for BISECT (BISECT_K). We configure BISECT to use the weighted origin selection and fix the number of cuts to 5. For HIDDEN, we set $\varepsilon = 0.1$, as we found larger values to be inefficient when generating hidden outliers for certain datasets, see Appendix.

To maintain tractability, we limited the number of subspaces used to generate hidden outliers to a maximum of 2048; when necessary, we used feature bagging without repetition (Lazarevic and Kumar 2005) for subspace selection. When a generational method failed to produce a successful candidate within a 30-minute timeframe, we regarded the respective experiment as unsuccessful (marked as ot). For all other methods used in our experiments, we adopted the default parameter settings suggested by the authors or provided by their respective implementations.

4.3 Hidden Outlier Generation Efficiency

In this section, we compare the time required by BISECT and HIDDEN to generate 500 hidden outliers (in seconds).

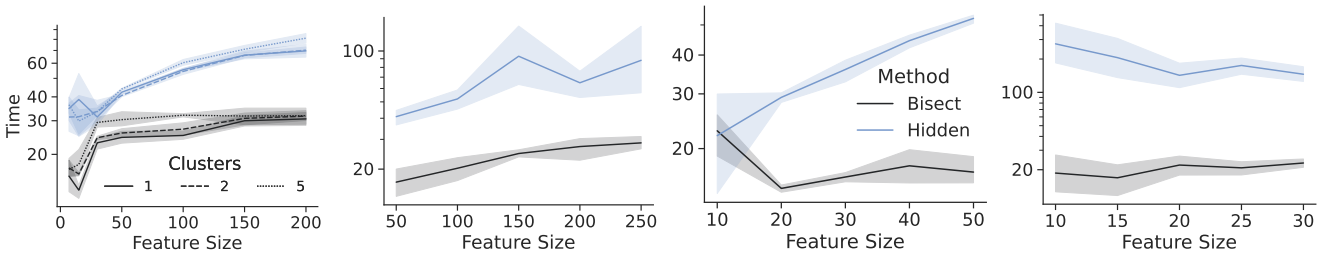
Synthetic Data. Figure 4a shows the time needed to generate 500 hidden outliers with BISECT and HIDDEN, contingent on the number of features in a dataset. HIDDEN requires 2–3 times more runtime than BISECT, and this difference increases with dimensionality. Dimensionality has a limited impact on the generation time when the number of subspaces is fixed, as expected by Proposition 2. Additionally, the number of clusters in the data has little effect on the performance of HIDDEN or BISECT.

Real Data. We used four real datasets with the highest feature count. To achieve the desired dimensionality, we created five lower-dimensional datasets by selecting five distinct subsets of features with minimal overlap. Figure 4 plots the time required to generate 500 hidden outliers using BISECT and HIDDEN for each dataset, depending on dimensionality. As with synthetic data, we observe that BISECT is several times faster than HIDDEN.

Table 3 summarizes the generation times from our experiments, both with synthetic and real data, for BISECT and HIDDEN. It is evident that the performance of BISECT is significantly more predictable than that of HIDDEN – the interquartile range for the generation time is an order of magnitude lower for BISECT.

4.4 Outlier Detection Experiments

One-class Classification. We leverage hidden outliers for self-supervised one-class classification. By augmenting the training data with these outliers, we train a binary classifier for one-class classification, following these steps:



(a) Synthetic datasets (b) Arrythmia (c) SpamBase (d) WPBC

Figure 4: Time to generate 500 hidden outliers (in seconds) in synthetic and real data contingent on feature count.

	Min	Q_1	Q_2	Q_3	Max	IQR
Bisect	9.9	15.3	19.6	25.2	44	9.9
Hidden	9.2	46.2	72.6	124.7	566	78.4

Table 3: Time to generate 500 hidden outliers (in seconds).

1. Divide the dataset D^{full} into D_{train} (80% inliers and no outliers) and D_{test} (20% inliers and all outliers).
2. Using D_{train}, generate $|D_{train}|$ outliers with an outlier generation method and add them to D_{train}.
3. Train a classifier on D_{train} to distinguish inliers and outliers, and evaluate its performance on D_{test}.

We repeat the experiment seven times for each dataset with different random splits into D_{train} and D_{test}. In Step 2, we use BISECT, HIDDEN, and Hyperbox (HB). HB is a naïve baseline that samples points uniformly from the minimal bounding box surrounding D_{train} and retains only those marked as outliers by LOF. In Step 3, we use random forest[4] since it performs well in binary classification tasks (Wainberg, Alipanahi, and Frey 2016).

We compare this self-supervised approach with LOF and KNN adjusted for one class classification as they serve as adversaries for BISECT. We also compare it to DeepSVDD (Ruff et al. 2018), a deep outlier detection method, and OCSVM, a one-class SVM with a radial kernel (Schölkopf et al. 1999). From generative baselines, we included AnoGAN, a popular self-supervised baseline, and MO-GAAL,[5] a recent deep-learning-based self-supervised method (Schlegl et al. 2017; Liu et al. 2019).

Table 4 presents the test ROC AUC scores. We group LOF-related and KNN-related methods together and identify the best method within each group using bold font. If a self-supervised method significantly outperforms its adversary (p-value of the Wilcoxon signed rank test ≤ 0.05), we mark the respective AUC value with an asterisk. Conversely, the dagger symbol indicates when a self-supervised method performs worse than its adversary. Additionally, the overall best-performing method for each dataset is in italics.

We observe that Bisect-based hidden outlier methods often outperform their adversary counterparts by much. This suggests that we can recommend replacing conventional full-space outlier detection methods with Bisect-based self-supervised methods with respective adversaries. In addition, Bisect-based methods offer a runtime improvement for inference when dealing with high-dimensional datasets. For instance, using the SpamBase dataset, RF trained after BISECT with LOF as an adversary processes each test point nearly 10 times (31ms vs 3.4ms) faster in average than LOF.

Supervised Outlier Detection. In this section, we perceive outlier detection as an imbalanced classification problem, with the train set having very few instances of the minority class (outliers). We use artificial hidden outliers to help balance the data, which enables us to train a binary classification model for outlier detection with the augmented set. The experiment has the following steps:

1. Take a dataset D with 2% outliers and split it randomly into 20% train set (D_{train}) and 80% test set (D_{test}).
2. Add to D_{test} outliers from D^{full}, excluding those in D, to get a more reliable outlier detection quality estimate.
3. Add extra outliers to D_{train} with outlier generation to balance outlier and inlier numbers. Do the same with oversampling, in order to have reference points.
4. Train a classifer on D_{train} and evaluate it on D_{test}.

We repeat the experiment seven times for each dataset with random splits into D_{train} and D_{test}. In Step 4, we employ random forest. In Step 3, we use BISECT, HIDDEN, and Hyperbox, as before. The oversampling methods[6] used in Step 3 are the neighbors-based approach SMOTE, the density-based technique ADASYN, and the clustering-based method DB-SMOTE (Chawla et al. 2002; He et al. 2008; Bunkhumpornpat, Sinapiromsaran, and Lursinsap 2012). We also employed an outlier-based oversampling method using a cWGAN [7], like multiple authors have proposed (Zheng et al. 2020; Mottini, Lheritier, and Acuna-Agost 2018; Engelmann and Lessmann 2021). Finally, we compare to random forest trained using only the train data.

Table 5 shows test ROC AUC scores. The two best methods for each dataset are in bold. A method significantly outperforming random forest on non-augmented data (p-value of Wilcoxon signed rank test < 0.05) has an asterisk, while

[4]Implementation from `caret` (Kuhn and Max 2008)

[5]Implementation from `pyod` (Zhao, Nasrullah, and Li 2019)

[6]Implementations from `smotefamily` (Siriseriwan 2019)

[7]Code by https://github.com/johaupt/GANbalanced/

Dataset	BISECT	HIDDEN	HB	LOF	BISECT$_K$	KNN	IForest	DSVDD	OCSVM	GAAL
Wilt	***0.975****	0.942*	0.520†	0.556	**0.970***	0.537	0,720	0.529	0.643	0.653
Pima	0.579†	0.499†	0.523†	**0.695**	0.563†	***0.747***	0,718	0.511	0.664	0.498
Stamps	**0.954***	0.699†	0.952*	0.895	***0.961***	0.957	0,935	0.766	0.877	0.913
PageBlocks	**0.952**	0.911	0.895†	0.931	**0.925***	0.661	*0,957*	0.676	0.619	0.415
Heart Disease	0.806	0.825	***0.829***	0.839	0.839	**0.813**	0,799	0.737	0.801	0.818
Annthyroid	**0.924***	0.877	0.514†	0.767	***0.944****	0.747	0,827	0.764	0.596	0.617
Cardio...	**0.826***	0.812	0.730†	0.798	***0.845****	0.776	0,757	0.626	0.840	0.540
Parkinson	**0.822***	0.777	0.779	0.745	**0.842**	0.838	*0,913*	0.839	0.738	*ot*
Ionosphere	0.927	0.535†	**0.950**	0.946	0.922†	***0.970***	0,965	0.951	0.801	0.757
WPBC	**0.613**	0.591	0.588	0.574	0.534	***0.635***	0,523	0.527	0.499	0.632
SpamBase	0.793*	***0.848****	0.793*	0.731	**0.800***	0.700	0,791	0.651	0.629	0.700
Arrhythmia	**0.756***	0.745	0.719	0.727	***0.769****	0.734	0,755	0.740	0.745	0.724

Table 4: Median performance of the different one-class classification methods. DSVDD stands for DeepSVDD.

Dataset	BISECT	HIDDEN	HB	Plain RF	cWGAN	SMOTE	DB SMOTE	ADASYN	# out.
Wilt	**0.957**	0.945	0.500†	0.951	0.953	0.947	**0.956**	0.945	19
Pima	0.625*	**0.700***	**0.642***	0.589	0.584	*na*	*na*	*na*	2
Stamps	**0.967***	0.898†	0.879†	0.948	**0.969***	*na*	*na*	*na*	2
PageBlocks	0.982†	0.910†	0.980†	**0.993**	**0.996**	0.966†	0.980†	0.961†	20
Heart Disease	0.814*	**0.820**	**0.831**	0.731	0.723	*na*	*na*	*na*	1
Annthyroid	**0.981**	**0.990***	0.471†	0.969	0.980	0.979	0.980	0.978	27
Cardio...	**0.921***	0.838	0.641†	0.895	0.916	0.913	0.904	**0.918**	7
Parkinson	0.549	*ot*	*ot*	**0.590**	**0.563**	*na*	*na*	*na*	1
Ionosphere	**0.937***	0.735	0.905	0.882	**0.931**	*na*	*na*	*na*	1
WPBC	**0.615**	**0.640**	0.566	0.602	0.547†	*na*	*na*	*na*	1
SpamBase	0.917*	0.868†	0.738†	0.908	0.903	0.920*	**0.921***	**0.921***	11
Arrhythmia	**0.975***	**0.973***	0.798*	0.689	0.924*	*na*	*na*	*na*	1

Table 5: Median performance of a random forest coupled with a generational method.

an inverse case is marked with the dagger symbol. The last column contains the outlier count in D_{train}.

The BISECT method is a clear winner overall, while conventional oversampling techniques are not applicable for datasets with sparse outliers as they have less than 3 minority class objects (marked as na). The only other method thought for scarce examples of the minority class, cWGAN, failed to significantly improve the classifier as much as our approach. Hence, we also recommend the BISECT method for oversampling in the case with few recorded outliers.

Limitations and Future Work. To cover recent and popular methods, our experiments include various competitors for supervised outlier detection and one-class classification. However, we faced difficulties including certain methods due to either outdated or unavailable implementations (Désir et al. 2013; Abe, Zadrozny, and Langford 2006; Zheng et al. 2020), or implementations designed solely to replicate outcomes on specific benchmark datasets (Oh, Hong, and Baek 2019; Dlamini and Fahim 2021).

In addition, there are other approaches, largely or completely unexplored for these tasks but adaptable like our use of BISECT. For example, beyond the Hyperbox and HIDDEN methods we have included in our comparison, 17 outlier generation methods reviewed in (Steinbuss and Böhm 2021) fall into this category. The experimental design space could also extend to other conventional outlier detection methods from (Campos et al. 2016) — beyond LOF, KNN, and OCSVM which we have covered. These could serve as further adversaries of BISECT or HIDDEN, or as competitors in one-class classification. However, such a broad comparison exceeds the scope of one conference publication, and we see it as future work.

5 Conclusions

Generating outliers is useful. Nevertheless, most outlier generation methods disregard the "multiple-views" property of outliers in high-dimensional spaces. Synthetic hidden outliers are the sole exception. However, the existing method for generating hidden outliers is inefficient and sensitive to hyperparameter selection, as we have shown. Furthermore, the utility of hidden outliers remains to be shown.

In this paper, we have established that synthetic hidden outliers exist under versatile conditions and, based on this theory, propose a way to search for respective candidates efficiently, which we call BISECT. BISECT is notably faster than the current alternative.

Next, we developed a methodology that makes use of outliers generated with BISECT to enhance outlier detection in the context of one-class classification and highly unbalanced

supervised outlier detection tasks. In both scenarios, the use of BISECT yielded significant improvements over conventional methods, surpassing widely adopted alternatives tailored for these tasks. These results confirm the potential of hidden outliers to advance various outlier detection tasks.

6 Aknowledgments

This work was supported by the Ministry of Science, Research and the Arts Baden Württemberg, project Algorithm Engineering for the Scalability Challenge (AESC).

References

Abe, N.; Zadrozny, B.; and Langford, J. 2006. Outlier detection by active learning. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, 504–509.

Aggarwal, C. C. 2013. *Outlier Analysis*. Springer. ISBN 978-1-4614-6396-2.

Aggarwal, C. C.; Hinneburg, A.; and Keim, D. A. 2001a. On the surprising behavior of distance metrics in high dimensional space. In *Database Theory—ICDT 2001: 8th International Conference London, UK, January 4–6, 2001 Proceedings 8*, 420–434. Springer.

Aggarwal, C. C.; Hinneburg, A.; and Keim, D. A. 2001b. On the Surprising Behavior of Distance Metrics in High Dimensional Space. In Van den Bussche, J.; and Vianu, V., eds., *Database Theory — ICDT 2001*, 420–434. Berlin, Heidelberg: Springer Berlin Heidelberg. ISBN 978-3-540-44503-6.

Aggarwal, C. C.; and Sathe, S. 2017. *Outlier ensembles: An introduction*. Springer.

Angiulli, F.; and Pizzuti, C. 2002. Fast Outlier Detection in High Dimensional Spaces. In Elomaa, T.; Mannila, H.; and Toivonen, H., eds., *Principles of Data Mining and Knowledge Discovery*, 15–27. Berlin, Heidelberg: Springer Berlin Heidelberg. ISBN 978-3-540-45681-0.

Breunig, M. M.; Kriegel, H.-P.; Ng, R. T.; and Sander, J. 2000. LOF: Identifying Density-Based Local Outliers. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, SIGMOD '00, 93–104. New York, NY, USA: Association for Computing Machinery. ISBN 1581132174.

Bunkhumpornpat, C.; Sinapiromsaran, K.; and Lursinsap, C. 2012. DBSMOTE: Density-Based Synthetic Minority Oversampling TEchnique. *Applied Intelligence*, 36[3]: 664–684.

Campos, G. O.; Zimek, A.; Sander, J.; Campello, R. J. G. B.; Micenková, B.; Schubert, E.; Assent, I.; and Houle, M. E. 2016. On the evaluation of unsupervised outlier detection: measures, datasets, and an empirical study. *Data Mining and Knowledge Discovery*, 30[4]: 891–927.

Chandola, V.; Banerjee, A.; and Kumar, V. 2009. Anomaly Detection: A Survey. *ACM Comput. Surv.*, 41[3].

Chawla, N. V.; Bowyer, K. W.; Hall, L. O.; and Kegelmeyer, W. P. 2002. SMOTE: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16: 321–357.

Dlamini, G.; and Fahim, M. 2021. DGM: a data generative model to improve minority class presence in anomaly detection domain. *Neural Computing and Applications*, 33[20]: 13635–13646.

Désir, C.; Bernard, S.; Petitjean, C.; and Heutte, L. 2013. One class random forests. *Pattern Recognition*, 46[12]: 3490–3506.

El-Yaniv, R.; and Nisenson, M. 2006. Optimal Single-Class Classification Strategies. In Schölkopf, B.; Platt, J.; and Hoffman, T., eds., *Advances in Neural Information Processing Systems*, volume 19. MIT Press.

Engelmann, J.; and Lessmann, S. 2021. Conditional Wasserstein GAN-based oversampling of tabular data for imbalanced learning. *Expert Systems with Applications*, 174: 114582.

He, H.; Bai, Y.; Garcia, E. A.; and Li, S. 2008. ADASYN: Adaptive synthetic sampling approach for imbalanced learning. In *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, 1322–1328.

Hempstalk, K.; Frank, E.; and Witten, I. H. 2008. One-class classification by combining density and class probability estimation. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2008, Antwerp, Belgium, September 15-19, 2008, Proceedings, Part I 19*, 505–519. Springer.

Ho, T. K. 1998. The random subspace method for constructing decision forests. *IEEE transactions on pattern analysis and machine intelligence*, 20[8]: 832–844.

Keller, F.; Muller, E.; and Bohm, K. 2012. HiCS: High contrast subspaces for density-based outlier ranking. In *2012 IEEE 28th international conference on data engineering*, 1037–1048. IEEE.

Khan, S. S.; and Madden, M. G. 2014. One-class classification: taxonomy of study and review of techniques. *The Knowledge Engineering Review*, 29[3]: 345–374.

Kuhn; and Max. 2008. Building Predictive Models in R Using the caret Package. *Journal of Statistical Software*, 28[5]: 1–26.

Lazarevic, A.; and Kumar, V. 2005. Feature Bagging for Outlier Detection. In *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, KDD '05, 157–166. New York, NY, USA: Association for Computing Machinery. ISBN 159593135X.

Liu, Y.; Li, Z.; Zhou, C.; Jiang, Y.; Sun, J.; Wang, M.; and He, X. 2019. Generative adversarial active learning for unsupervised outlier detection. *IEEE Transactions on Knowledge and Data Engineering*, 32[8]: 1517–1528.

Mottini, A.; Lheritier, A.; and Acuna-Agost, R. 2018. Airline Passenger Name Record Generation using Generative Adversarial Networks. *arXiv e-prints*, arXiv:1807.06657.

Müller, E.; Assent, I.; Iglesias, P.; Mülle, Y.; and Böhm, K. 2012. Outlier Ranking via Subspace Analysis in Multiple Views of the Data. In *2012 IEEE 12th International Conference on Data Mining*, 529–538.

Oh, J.-H.; Hong, J. Y.; and Baek, J.-G. 2019. Oversampling method using outlier detectable generative adversarial network. *Expert Systems with Applications*, 133: 1–8.

Pawar, A. D.; Kalavadekar, P. N.; and Tambe, S. N. 2014. A survey on outlier detection techniques for credit card fraud detection. *IOSR Journal of Computer Engineering*, 16[2]: 44–48.

Ruff, L.; Vandermeulen, R.; Goernitz, N.; Deecke, L.; Siddiqui, S. A.; Binder, A.; Müller, E.; and Kloft, M. 2018. Deep One-Class Classification. In Dy, J.; and Krause, A., eds., *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, 4393–4402. PMLR.

Sadr, A. V.; Bassett, B. A.; and Kunz, M. 2019. A Flexible Framework for Anomaly Detection via Dimensionality Reduction. In *2019 6th International Conference on Soft Computing & Machine Intelligence (ISCMI)*, 106–110.

Schlegl, T.; Seeböck, P.; Waldstein, S. M.; Schmidt-Erfurth, U.; and Langs, G. 2017. Unsupervised Anomaly Detection with Generative Adversarial Networks to Guide Marker Discovery. In Niethammer, M.; Styner, M.; Aylward, S.; Zhu, H.; Oguz, I.; Yap, P.-T.; and Shen, D., eds., *Information Processing in Medical Imaging*, 146–157. Cham: Springer International Publishing. ISBN 978-3-319-59050-9.

Schölkopf, B.; Williamson, R. C.; Smola, A.; Shawe-Taylor, J.; and Platt, J. 1999. Support Vector Method for Novelty Detection. In Solla, S.; Leen, T.; and Müller, K., eds., *Advances in Neural Information Processing Systems*, volume 12. MIT Press.

Sharma, S.; Bellinger, C.; Krawczyk, B.; Zaiane, O.; and Japkowicz, N. 2018. Synthetic Oversampling with the Majority Class: A New Perspective on Handling Extreme Imbalance. In *2018 IEEE International Conference on Data Mining (ICDM)*, 447–456.

Siriseriwan, W. 2019. smotefamily: a collection of oversampling techniques for class imbalance problem based on SMOTE. *R package version*, 1[1].

Steinbuss, G.; and Böhm, K. 2017. Hiding outliers in high-dimensional data spaces. *International Journal of Data Science and Analytics*, 4: 173–189.

Steinbuss, G.; and Böhm, K. 2021. Generating artificial outliers in the absence of genuine ones—A survey. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 15[2]: 1–37.

Ushey, K.; Allaire, J.; and Tang, Y. 2023. *reticulate: Interface to 'Python'*. Https://rstudio.github.io/reticulate/, https://github.com/rstudio/reticulate.

van der Maaten, L.; Postma, E.; and Herik, H. 2007. Dimensionality Reduction: A Comparative Review. *Journal of Machine Learning Research - JMLR*, 10.

Wainberg, M.; Alipanahi, B.; and Frey, B. J. 2016. Are Random Forests Truly the Best Classifiers? *Journal of Machine Learning Research*, 17[110]: 1–5.

Zhao, Y.; Nasrullah, Z.; and Li, Z. 2019. PyOD: A Python Toolbox for Scalable Outlier Detection. *Journal of Machine Learning Research*, 20[96]: 1–7.

Zheng, M.; Li, T.; Zhu, R.; Tang, Y.; Tang, M.; Lin, L.; and Ma, Z. 2020. Conditional Wasserstein generative adversarial network-gradient penalty-based approach to alleviating imbalanced data classification. *Information Sciences*, 512: 1009–1023.

A Theoretical Appendix

This appendix includes all the proofs for the theoretical results presented in Section 3, as well as other supplementary results needed to prove the said results. We include all the proofs in the order of inclusion in the text. For completeness, the statements are repeated before the proof.

Additionally, we are going to introduce a couple of notations needed for the proofs. First, let us consider everything from Section 3. Now, let $C(a, b)$ be the image of the convex combination of two points of X, a, and b. Unless is stated otherwise, $\alpha(t) = ty + (1-t)x$. Additionally, $C(a, b)_-$ is the image of the convex combination with $t \in (0, 1]$ and $C(a, b)^+$ with $t \in [0, 1)$. The set $C(a, b)^+_-$ is the convex combination with $t \in (0, 1)$.

Whenever we write $B_X(z, \varepsilon)$ we refer to an open ball of X centered in z with radius $\varepsilon > 0$. $B_A(z, \varepsilon)$, with $A \subset X$, stands for the open ball in the induced topology, i.e. $B_A(z, \varepsilon) = B_X(z, \varepsilon) \cap A$.

Proposition 3. *("Hidden outlier existence"): Let x and y be points in the previously defined metric space such that $x \in R(\mathcal{M})$ and $y \notin R(\mathcal{M})$. Assume that there exists a point z in the convex combination of x and y such as $z \in \partial R(\mathcal{M}) \Rightarrow z \notin \partial R(\mathcal{E}_\mathcal{M})$. Then there exists z' in the convex combination of x and y such as $z' \in H_1(\mathcal{M}) \cup H_2(\mathcal{M})$.*

Proof. Assume $z \in \partial R(\mathcal{M})$, otherwise $z \in R(\mathcal{M}) \setminus R(\mathcal{E}_\mathcal{M})$ trivially. Also, since $z \notin \partial R(\mathcal{E}_\mathcal{M})$ by hypothesis, z has to be in the interior, $R(\mathcal{E}_\mathcal{M})^\circ$. Now, since $z \in \partial R(\mathcal{M})$, by definition of boundary point, $\forall \varepsilon > 0,\ B_X(z, \varepsilon) \cap (X \setminus R(\mathcal{M})) \neq \emptyset$. Recall that, by the notion of induced topology,

$$C(x, y) \subset X \Longrightarrow B_{C(x,y)}(z, \varepsilon) = B_X(z, \varepsilon) \cap C(x, y).$$

Then, since $z \in \partial R(\mathcal{M}) \cap C(x, y)$, we have

$$\forall \varepsilon > 0,\ B_{C(x,y)}(z, \varepsilon) \cap X \setminus R(\mathcal{M}) \neq \emptyset. \tag{1}$$

Since,

$$z \in R(\mathcal{E}_\mathcal{M})^\circ \Longrightarrow \exists \varepsilon' > 0,\ B_{C(x,y)}(z, \varepsilon') \subset R(\mathcal{E}_\mathcal{M}),$$

we have that

$$B_{C(x,y)}(z, \varepsilon') = B_{C(x,y)}(z, \varepsilon') \cap R(\mathcal{E}_\mathcal{M}). \tag{2}$$

By (1),

$$B_{C(x,y)}(z, \varepsilon') \cap X \setminus R(\mathcal{M}) \neq \emptyset.$$

By (2),

$$B(z, \varepsilon') \cap X \setminus R(\mathcal{M}) = B(z, \varepsilon') \cap X \setminus R(\mathcal{M}) \cap R(\mathcal{E}_\mathcal{M}) \neq \emptyset.$$

By Zornn's Lemma, we can find $z' \in X$ such that

$$z \in B_{C(x,y)}(z, \varepsilon') \cap R(\mathcal{E}_\mathcal{M}) \setminus R(\mathcal{M}) = C(x, y) \cap H_2.$$

Changing $z \in \partial R(\mathcal{M})$ to $z \in \partial R(\mathcal{E}_\mathcal{M})$ leads to the inclusion to H_1. □

Before proving Theorem 1 we need to introduce a Lemma that is going to help us through the proof. It will help us assess when one should expect a boundary point on the convex combination of two points, and under what conditions.

Lemma 1. *Consider X as our metric space as before, and $A \in X$ a subspace. Then,*

$$\begin{cases} x \in A, \\ y \notin A. \end{cases} \implies \exists z \in \partial A \cap C(x,y). \tag{3}$$

$$\begin{cases} x \in A, \\ y \in A, \\ \nexists z \in \partial A \cap C(x,y). \end{cases} \implies C(x,y) \subset A. \tag{4}$$

Proof. First, let us prove statement 3. Let $\{t_i\}_{i\in\mathbb{N}}$ be a monotonone sequence in $[0,1]$ such that $\alpha(t_i) \in A$ for all points in the sequence. As $y \notin A$, very clearly such sequence has to be bounded. As $\{t_i\}$ is bounded, then, it has a supremum T. By the monotonic convergence theorem,

$$t_i \longrightarrow T.$$

Assume that $\alpha(T) \notin \partial A$. Then, you can fit an open ball with radius $\varepsilon > 0$ in $A \cap C(x,y)$, and get a point $z' = \alpha(T')$ in the ball such that $T' = T + \varepsilon > T$. As T is the supremum, this cannot happen. Therefore, $\alpha(T) = z \in \partial A$.

Let us now prove statement 4. Assume that it exists a t in $[0,1]$ such that $\alpha(t) \notin A$. Then, by the statement 3 of this very same lemma, $\exists z \in \partial A$, which is untrue by hypothesis. □

Now, we can finally tackle the proof for Theorem 1.

Theorem 2. *("Convergence into a hidden outlier") Let f be defined as before. Assume that at most exist, and are unique, other z and z' in the convex combination of $x \in R(\mathcal{M})$ and $y \notin R(\mathcal{M})$ such as $z \in \partial R(\mathcal{M})$, $z' \in \partial R(\mathcal{E}_\mathcal{M})$, $z \neq z'$, and both verify the last condition of proposition 1. Then the bisection method will converge to a root of f.*

Proof. The way of proving this statement will be to consider all inclusion possibilities and check when we can converge to the function f zeroes using the bisection method. Recall that the bisection method always converges as long as there is a sequence of nested intervals $[a_1,b_1] \supset \cdots \supset [a_n,b_n] \supset \cdots$ such that the root is always in the sequence, and are such that $sign(f(a_i)) \neq sign(f(b_i))$. We will first break down all possible inclusions from the different acceptance regions. After that, we will study when the function f converges into a hidden outlier by looking at all its possible values.

First, let us assume that $x \in R(\mathcal{E}_\mathcal{M})$. The case $x \notin R(\mathcal{E}_\mathcal{M})$ will be proven afterward.

Part I $(x \in R(\mathcal{E}_\mathcal{M}))$: Consider that $z \in R(\mathcal{M})$ since $R(\mathcal{M})$ is closed and $x \in R(\mathcal{M})$ by hypothesis. Additionally, $\nexists z'' \in \partial R(\mathcal{M})$ in $C(x,z)$ as z is the only boundary point in $C(x,z) \subset C(x,y)$ by hypothesis. Then, we have that $C(x,z) \subset R(\mathcal{M})$ by Lemma 1. Now, we have two possibilities:

1. $z \in R(\mathcal{E}_\mathcal{M})$: As z hast to be an interior point by hypothesis, $C(x,z)$ is completely contained in $R(\mathcal{M}) \cap R(\mathcal{E}_\mathcal{M})$. Assuming that:
   a. $y \in R(\mathcal{E}_\mathcal{M})$: Assume that there are no $z' \in \partial R(\mathcal{E}_\mathcal{M})$ in $C(z,y)$. Therefore by Lemma 1, we have that $\forall c \in C(x,y), C(c,y) \subset R(\mathcal{E}_\mathcal{M})$, as long as $c \neq z$.
   b. $y \notin R(\mathcal{E}_\mathcal{M})$: As $z \in R(\mathcal{E}_\mathcal{M})$ and $y \notin R(\mathcal{E}_\mathcal{M})$, by Lemma 1 we can find $z' \in \partial R(\mathcal{E}_\mathcal{M})$. Additionally, by hypothesis, $z' \notin \partial R(\mathcal{M})$, and both z and z' are the unique boundary points in $C(x,y)$. That also means that they are the only boundary points in $C(z,z')$. Now, as there are no boundary points in $C(z,z')$, we have that $\forall c \in C(z,z'), c \neq z$ are not points of $R(\mathcal{M})$ thanks to statement 4 from Lemma 1. Lastly, is instant by the same argument that $C(c,y) \subset X \setminus (R(\mathcal{M} \cup R(\mathcal{E}_\mathcal{M})$.

   Then, in case 1.a, $C(x,y) = C(x,z) \cup C(z,y)$ and $f(C(x,z)) = -1,\ f(C(z,y)) = 0$. For case 1.b, $C(x,y) = C(x,z) \cup C(z,z') \cup C(z',y)$ and $f(C(x,z)) = -1,\ f(C(z,z')_-) = 0,\ f(C(z',y)_-) = 1$.
2. $z \notin R(\mathcal{E}_\mathcal{M})$: As $z \notin R(\mathcal{E}_\mathcal{M})$ and $x \in R(\mathcal{E}_\mathcal{M})$, by Lemma 1 and hypothesis, there exists, and is unique in $C(x,y)$, a point $z' \in \partial R(\mathcal{E}_\mathcal{M})$ in $C(x,z)$.
   a. $y \in R(\mathcal{E}_\mathcal{M})$: This cannot occur since it leads to a contradiction by using Lemma 1 and obtaining a different z'' in the boundary of $R(\mathcal{E}_\mathcal{M})$.
   b. $y \notin R(\mathcal{E}_\mathcal{M})$: By the statement 4 from Lemma 1 we have that $C(z,y) \subset X \setminus R(\mathcal{E}_\mathcal{M})$. Additionally, using the same argument as in 1.b, $C(c,y) \subset X \setminus R(\mathcal{M})$ for all $c \in C(z,y)$ such that $c \neq z$. By utilizing statement 3 again with the same argument as in 1.b, but changing $z \in \partial R(\mathcal{M})$ to $z' \in \partial R(\mathcal{E}_\mathcal{M})$, get that $C(x,z') \subset R(\mathcal{M}) \cap R(\mathcal{E}_\mathcal{M})$.

   Thus, we have that for case 2.b $f(C(x,z')) = -1$, $f(C(z',z)_-) = 0$, $f(C(z,y)_-) = 1$.

Part II $(x \notin R(\mathcal{E}_\mathcal{M}))$: By Lemma 1 and hypothesis, $C(x,z) \subset R(\mathcal{M})$. Consider again:

1. $z \in R(\mathcal{E}_\mathcal{M})$: By Lemma1, there exists $z' \in \partial R(\mathcal{E}_\mathcal{M})$ in the convex combination between x and z. As there are no more boundary points left, we have that $C(x,z')^+ \subset R(\mathcal{M}) \setminus R(\mathcal{M})$ and $C(z',z) \subset R(\mathcal{M}) \cap R(\mathcal{E}_\mathcal{M})$. Again, let us divide the hypothesis space into two cases:
   a. $y \in R(\mathcal{E}_\mathcal{M})$: By pretty similar arguments as in I.1.b and before, $C(z,y)_- \subset R(\mathcal{E}_\mathcal{M}) \setminus R(\mathcal{M})$.
   b. $y \in R(\mathcal{E}_\mathcal{M})$: As in 2.b, this case is impossible, otherwise it will lead to a contradiction of the hypothesis, just like in I.2.b.

   In this case, we will have for 1.a that $f(C(x,z')^+ = 0,\ f(C(z',z) = -1,\ f(C(z,y)) = 0$.
2. $z \notin R(\mathcal{E}_\mathcal{M})$
   a. $y \in R(\mathcal{E}_\mathcal{M})$: There has to exist $z' \in \partial R(\mathcal{E}_\mathcal{M})$ in $C(z,y)$ by the statement 3 of Lemma 1. Then, as there are no more boundary points, we can use statement 4 to obtain that $C(x,z) \subset R(\mathcal{M}) \setminus R(\mathcal{E}_\mathcal{M})$, $C(z,z')^+_- \subset X \setminus (R(\mathcal{M}) \cup R(\mathcal{E}_\mathcal{M}))$, and that $C(z',y) \subset R(\mathcal{E}_\mathcal{M}) \setminus R(\mathcal{M})$.

b. $y \notin R(\mathcal{E}_\mathcal{M})$: Lastly, let us assume that $\nexists z' \in C(x,y)$ such that $z' \in \partial R(\mathcal{E}_\mathcal{M})$, as we did in I.1.a. Then, $C(z,y) \subset X \setminus (R(\mathcal{E}_\mathcal{M}) \cup R(\mathcal{M}))$.

Therefore, for 2.a: $f(C(x,z)) = 0,\ f(C(z,z')_-) = 1,\ f(C(z',y)) = 0$. And, finally, for 2.b: $f(C(x,z)) = 0,\ f(C(z,y)) = 1$.

That way, in cases I.1.b, I.2.b, II.1.a, and II.2.a, it is fairly obvious how to construct a sequence such that we can encapsulate all roots. For cases I.1.b and I.2.b it suffices with selecting the hole interval. For cases II.1.b and II.2.a it suffices to restrict the function f to the convex combination of a point $c \in C(z,z')$ and x. For cases I.1.a and II.2.b, if one assumes that there exists another boundary point $z' \partial R(\mathcal{E}_\mathcal{M})$ also in $C(z,y)$ it can be proven similarly as for cases I.1.a and II.2.b. In both cases z' is the only point from $R(\mathcal{E}_\mathcal{M})$ in the convex combination, otherwise by Lemma 1 we could get another boundary point and get to a contradiction. Then it suffices to take again a point $c \in C(z,z')^+$ and restrict the f to the convex combination of x and c, as before. Utilizing this idea of segmenting the intervals by a closer outlier of $\mathcal{M}$ could be extended to tackle the case where there are more than one z and z'.

This proves the convergence of the bisection method. However, consider that $length([a_n, b_n]) \longrightarrow 0$ with the bisection method. Consider as well that for every h root of f there exists a neighborhood $N(h)$ of length greater than 0. Then, $N(h)$ is not the supremum of the sequence $[a_n, b_n]$, and therefore, the bisection method will converge in finite time into any point h' from $N(h)$. I.e., it will find hidden outliers in finite time. □

Proposition 4. *Let n_{subs} be the number of selected subspaces for the ensemble. The worst-case complexity of* BISECT *is $\mathcal{O}\left(\log(\frac{L}{n_{cuts}}) \cdot (n_{subs} + n_{cuts}) \cdot \aleph\right)$ where $\aleph$ is the inference complexity of the adversary $\mathcal{M}(\cdot)$.*

Proof. Consider that we obtained a L sufficiently large as stated in the steps for BISECT. Then, complexity can be trivially bounded by $\mathcal{O}\left(n_{\text{iter}} \cdot (n_{\text{subs}} + n_{\text{cuts}}) \cdot \aleph\right)$, with n_{iter} being the number of iterations for the bisection method. As the bisection method is also bounded by $\mathcal{O}\left(\log(L')\right)$ being L' the length of the interval (in our case $L' = \frac{L}{n_{\text{cuts}}}$), we have that:

$$\mathcal{O}\left(\log(L') \cdot (n_{\text{subs}} + n_{\text{cuts}}) \cdot \aleph\right) \leq \mathcal{O}\left(n_{\text{iter}} \cdot (n_{\text{subs}} + n_{\text{cuts}}) \cdot \aleph\right).$$

Let us derive the bounding for the bisection method for completion. Assume that we want to converge to the right side of the interval w.l.g, as the total length traveled inside the interval will be equivalent. Consider,

$$\sum_{i=1}^{n_I} \frac{x_i - x_{i-1}}{2} = L' - Err,$$

where n_I is a finite number of iterations, and Err the error of the algorithm. Then, as we want to convert to the right side, L':

$$\sum_{i=1}^{n_I} \frac{L'}{2^i} = L' - Err.$$

Dataset	$\varepsilon = 0.1$	ε_{opt}
Stamps	**0.942***	0.853
Annthyroid	**0.944***	0.793
Cardio...	0.812	**0.815**
Parkinson	0.777	**0.867***
Ionosphere	0.535	**0.969***
WPBC	0.591	**0.644**
SpamBase	0.848	**0.864**

Table 6: Median AUC for HIDDEN with $\varepsilon = 0.1$ and optimal ε in time, in Supervised Outlier Detection.

Dataset	$\varepsilon = 0.1$	ε_{opt}
Stamps	**0.898***	0.699
Annthyroid	**0.990***	0.944
Cardio...	**0.838**	0.815
Parkinson	*ot*	*ot*
Ionosphere	0.735	**0.97***
WPBC	0.566	**0.644**
SpamBase	0.738	**0.864***

Table 7: Median AUC for HIDDEN with $\varepsilon = 0.1$ and optimal ε in time, in One-class Classification.

We can rewrite our series as a finite geometric series by $\frac{1}{2}\sum_{i=0}^{n_I} \frac{L'}{2^i} = \sum_{i=1}^{n_I} \frac{L'}{2^i}$. Then, by considering its sum:

$$\frac{1}{2}\left(\frac{1 - \frac{1}{2^{n_I+1}}}{1 - \frac{1}{2}}\right) = L' - Err. \tag{5}$$

By (5), and after doing some algebra,

$$Err = \frac{L'}{2^{n_I+1}},$$

which leads to the desired bounding. Additionally, one could obtain the number of iterations to converge with an error for a desired interval length with (5). □

B Experimental Appendix

In this Appendix, we include the experimental results supporting the selection of ε. In Table 8 we collected the time for generating 100 hidden outliers on each task-specific training set. We selected the ε with the smaller maximum execution time between a small, medium, and large value of epsilon ($0.1,\ 0.5$ and 0.75, respectively) in this case it was $\varepsilon = 0.1$. Additionally, we also performed each experiment with the corresponding fastest epsilon between the tested. Results were gathered in Table 6 for Supervised Outlier Detection and in Table 7 for One-class Classification. We bolded those results with higher median AUC in each row. We marked with an asterisk those that were significantly different by the Wilcoxon-signed rank test, as in Section 4. As we can see from both tables, there are no significant differences between both values. Sometimes ε_{opt} degrades the performance of the classifier and sometimes increases it compared to the smaller value of ε.

Datasets	One-class classification			Supervised outlier detection		
	.1	.5	.75	.1	.5	.75
Wilt	**4.28**	10.80	26.89	**3.978**	10.646	24.776
Pima	**0.852**	1.512	4.214	**0.596**	1.039	2.539
Stamps	8.285	**5.428**	12.719	13.18	**5.641**	10.018
PageBlocks	-	-	-	**4.731**	131.461	216.037
Heart Disease	**1.151**	1.195	1.529	**1.1**	**1.1**	1.267
Annthyroid	5.698	**4.217**	4.577	4.837	4.226	**2.267**
Cardiotocography	20.661	3.847	**3.705**	15.638	4.588	**4.559**
Parkinson	76.473	**20.685**	27.01	-	-	-
Ionosphere	38.38	8.524	**7.793**	53.485	9.82	**8.784**
WPBC	95.828	**7.264**	25.046	90.687	**7.609**	21.963
SpamBase	14.444	12.406	**11.004**	12.606	13.127	**11.878**
Arrhytmia	**10.277**	23.696	15.58	**11.819**	38.832	25.815

Table 8: Time to generate 100 hidden outliers for each dataset using the training set for each use case.