AN AXIOMATIC PERSPECTIVE ON ANOMALY DETECTION

CHESTER WYKE

A THESIS PROPOSAL
SUBMITTED TO THE FACULTY OF GRADUATE STUDIES
IN PARTIAL FULFILMENT OF THE REQUIREMENTS
FOR THE DEGREE OF MASTER OF SCIENCE

GRADUATE PROGRAM IN
ELECTRICAL ENGINEERING AND COMPUTER SCIENCE
YORK UNIVERSITY
TORONTO, ONTARIO

AUGUST 2024

Abstract

A major challenge for both theoretical treatment and practical application of unsupervised learning tasks, such as clustering, anomaly detection or generative modeling, is the inherent lack of quantifiable objectives. Choosing methods and evaluating outcomes is then often a matter of ad-hoc heuristics or personal taste. Anomaly detection is often employed as a preprocessing step to other learning tasks, and unsound decisions for this task may thus have far-reaching consequences. In this work, we propose an axiomatic framework for analyzing behaviours of anomaly detection methods. We propose a basic set of desirable properties (or axioms) for distance-based anomaly detection methods and identify dependencies and (in-)consistencies between subsets of these. In addition, we include empirical results, which demonstrate the benefits of this axiomatic perspective on behaviours of anomaly detection methods. Our experiments illustrate how some commonly employed algorithms violate, perhaps unexpectedly, a basic desirable property. Namely, we highlight a material problem with a commonly used method called Isolation Forest, related to

infinite bands of space likely to be labelled as inliers that extend infinitely far away from the training data. Additionally, we experimentally demonstrate that another common method, Local Outlier Factor, is vulnerable to adversarial data poisoning. To conduct these experimental evaluations, a tool for dataset generation, experimentation and visualization was built, which is an additional contribution of this work.

Acknowledgements

I would like to thank God, through whom all things are possible. I would also like to thank my family and friends who supported me throughout. I would also like to thank the York community for their support, especially: Dr. Stephen Chen, who served on my examination committee and provided invaluable feedback on my Thesis; Dr. Jeff Edmonds for all his guidance, encouragement and constructive feedback; and, last but by no means least my supportive supervisor Dr. Ruth Urner for always making time to guide me along this journey.

This work was supported by the Vector Scholarship in Artificial Intelligence, provided through the Vector Institute.

Table of Contents

List of Tables

List of Figures

1 Introduction

1.1 Background

In contrast to supervised learning tasks, such as classification or regression, unsupervised learning tasks, such as clustering, generative modelling or anomaly detection, inherently lack objective evaluation measures. In supervised learning, the task itself typically dictates a loss function. For example, the classification loss assigns positive loss 1 to a predictor f on instance-label pair (x, y) if and only if $f(x) \neq y$. That is, positive loss is assigned if the predictor misclassifies the instance (say, an image classifier detects a zebra when really there is a tiger in the image). This loss function can then be used to evaluate any predictor on a labelled test dataset to get a sense of the predictor's classification accuracy. Since in supervised learning, training and test data come with observed outputs, a user has a straightforward way to estimate a quality measure of the predictor for the task at hand from data.

Unsupervised learning tasks, on the contrary, usually do not come with an inherently defined quality measure. Which type of qualities a clustering of a dataset

should have depends on the downstream task the clustering is used for (should there be a certain number of clusters? should clusters be dense? should clusters capture lower dimensional manifolds in the data?) [5]. Whether an image of a landscape produced by a generated model is photo-realistic or fits a certain artistic style may be assessed differently by different viewers. Whether a data entry should be considered anomalous based on, for example, one feature being outside the usual range for that feature, depends on the application the anomaly detection is used for (are we removing entries where a measurement error might have occurred to establish a clean dataset for a downstream task or to detect a faulty machine? or are we scanning for points that are at far distance from all other points, for example when we want to monitor a fleet of vehicles in a city and flag when one leaves the allowed area?).

Anomaly detection methods are used for a variety of applications. One may use anomaly detection as a preprocessing (or data-cleaning) step before a subsequent learning or statistical analysis step. Here, one may want to only remove data-points that were the results of measurements or transmission errors, and care needs to be taken so as to not skew the statistical patterns in the data when removing points. Thus, in such an application a user should be wary of being too generous with the concept of anomaly the method applies. On the other end of the spectrum, anomaly detection methods may be used for tasks that are sensitive in terms of

security, such as fraud detection for financial transactions or malware detection. For such applications it is crucial that the method does not miss any anomalous instances, while false alarms might naturally be tolerated for the sake of the overall safety of the system.

1.2 Thesis Organization

The focus of this thesis is to establish frameworks (both formal / theoretical and experimental) that allow users to better understand (and distinguish) the behaviours of various outlier detection methods.

This work is organized as follows:

- Chapter 1 Introduction - Gives an overview of the work done.
- Chapter 2 Related Work - Situates this work in the broader research landscape and covers other areas researched.
- Chapter 3 Theory - Covers the theoretical part of the axiomatic analysis. It proposes, motivates and analyzes the axioms and their interactions.
- Chapter 4 Tools - Covers the software tools developed in support of this work.
- Chapter 5 Experiments - Describes and discusses the experiments conducted.

- Chapter 6 Conclusion - Provides a short summary of the main ideas from this work as well as some pointers to follow-up research directions.

This research has been accepted for publication at ECAI 2024 [36].

1.3 Theory

Axiomatic or property-based analysis of tasks and algorithms offers a framework for clarifying behaviours of different methods. This allows users to make an informed choice when selecting a method for their unsupervised learning task. Such an axiomatic perspective on algorithms has been developed for clustering and community detection and led to property-based categorizations of common clustering methods (see discussion in related work, Section 2.1). We are aiming to initiate the development of such a framework for anomaly (or outlier) detection.

We focus on analyzing distance-based anomaly detection methods. We chose this approach since a feature-based representation implies distances, and thus is also included. Thus, working with pure distances simplifies the discussion without too much sacrifice of expressivity. Our intention is for this work to serve as a solid foundation to be built upon.

We start by phrasing five basic properties (or axioms) that a distance-based anomaly detection method should have: Scale Invariance (SI), Richness (R), Consistency (C), Permutation Invariance (PI) and Exile (E). The first three are inspired

by the axiomatic frameworks for clustering [19], but our Richness and in particular our Consistency axioms are adapted to reflect desirable behaviours of methods for outlier detection (rather than clustering) tasks. Richness requires that for any reasonably small subset of the points, there should be a set of distances that will result in this subset being assigned anomaly status. This property can be viewed as a way of preventing undesired biases, for example the method always detecting an even number of outliers, or the last point always being among the outliers. Consistency demands that when the distances are changed in such a way that inliers only move closer to each other and outliers only move further away from all other points, then the output of the anomaly detection should not change. Our last property, Exile, is specific to outlier detection: a point that is further away from all other points than the distance between any pair among the remaining points should be labeled as an anomaly.

We then show that the first three axioms, Scale Invariance, Richness and Consistency, while seemingly natural, cannot be satisfied simultaneously by any anomaly detection method. This implies that users will need to make a choice regarding their relative importance. We also show that each pair of the first three axioms is satisfiable, and also is satisfiable with any choice among Permutation Invariance and Exile, establishing that these two axioms are independent of the three pairs.

1.4 Tools

In support of the axiomatic analysis, we also developed a flexible testing framework that makes it simple to: specify synthetic datasets to generate, configure which experiments to be run, and choose the format for results to be reported. The framework supports several generation processes, such as gaussian, uniform and polynomial. In addition, the framework also supports 16 relevant performance measurements, most of which are standard. However we also included a few custom measurements such as the "open space detector". This open space detector looks for areas that are not near the training data but have a relatively high score (which makes it more likely for such a point to be misclassified as an inlier). In addition, some of the other performance measurements provide ease of access to functions available in the scikit-learn library [25]. All the models tested returned scores for each of the training points (scalars values). As a result, we needed to use a threshold to convert those scores into predictions for some of the metrics that we were interested in, such as the false positive rate. When we needed to use a threshold, we used the one that achieves the best F1 score, as this trades off both recall and precision (explained in more detail, with relevant formulas in Section 4.1). Along with the numeric performance measurements, the tool also provides easy access to the generation of visualizations that provide a graphical summary of the model's

performance.

We also developed a visual 2D dataset viewer and editor. It supports adding, editing and removing points directly on a plot view or via a table interface. It facilitates easy testing of ideas by integrating with the testing framework previously described. Furthermore, it provides one-click access to start experiments on the dataset currently being modified. The tool is able to be run in the browser with no installation required, or locally on the users machine with enhanced functionality.

1.5 Experiments

We also show experimentally that two well-established and commonly used outlier detection methods, Local Outlier Factor(LOF) and Isolation Forest, perhaps surprisingly, do not satisfy our Exile axiom. Our experiments also demonstrate that LOF might be vulnerable to data poisoning, in Section 5.3.

Our experiments demonstrate that Isolation Forest sometimes produces infinite bands of normal regions, which we explain in more detail in Section 5.2. While we initially observed the bands on our synthetic two-dimensional datasets (where we could see them), our testing showed that the bands are actually much more prevalent in the real world datasets than they are in the synthetic datasets.

2 Related Work

2.1 Axiomatic analysis

The line of research most relevant to our work here is the axiomatic treatment of clustering tasks. The initial study phrased the three axioms Scale Invariance, Richness and Consistency for clustering and proved their inconsistency as well as pairwise satisfiability [19]. This was followed up by a study suggesting a change of perspective from axioms for clustering methods to clustering quality measures and providing a consistent set of axioms for these [6]. This line of work was then further developed into a categorization of clustering methods by sets of properties that these satisfy [1, 2]. Axiomatic analysis with a similar program has also been developed for other tasks such as community detection [7]. However, we are not aware of prior work suggesting this type of analysis for the task of anomaly detection, which is the focus of our work.

On the other hand, axioms for outlier detection have been proposed before with a different focus [21]. These previously proposed axioms focused on how a

pair of datasets with one difference between them should affect the ranking of anomalies. This is in contrast to our theoretical analysis where we focused on an outlier detection paradigm in which a dataset and distance function is taken as input and a set of anomalies as output. This may include deciding on a threshold, as needed, depending on if the algorithm produces scores for the points. The main focus of the previous work was presenting a new algorithm; the axioms were proposed in support of it. The axioms proposed were:

1. *Distance Aware* - Given two datasets and all else being equal, if one dataset has a point that is further away from the normal data, then the farther away point from the normal observations should have a higher anomaly score.

2. *Density Aware* - Given two datasets and all else being equal, if one dataset is such that the normal points are more densely packed (more points in the same amount of space), then the outlier in the dataset that is more densely packed should have a higher anomaly score.

3. *Radius Aware* - Given two datasets and all else being equal, if one dataset is such that the same number of normal points occupy less space (same number of points in a smaller amount of space), then the outlier in the dataset with the smaller radius should have a higher anomaly score.

4. *Angle Aware* - In the study [21] in Section 3, on page 803, the axiom is stated

as, "All else being equal, smaller the angle of a point with respect to cluster of observation, more anomalous the outlier."

5. *Group-size Aware* - Given two datasets and all else being equal, if one dataset has fewer points in a group of anomalies, the points in the smaller group should be more anomalous.

The previous study tested if the algorithms satisfied the axioms statistically, by conducting two-sample t-tests. The tests showed that the new algorithm satisfied the axioms presented but no previous algorithm satisfied all of them.

The outlier detection task has also been phrased as a statistical learning problem with PAC type guarantees developed [31]. Non-parametric statistical guarantees have been provided for nearest-neighbour based methods under assumptions on the data-generating process [14]. An analysis of statistical requirements has also been employed to explore and analyze properties of fairness for anomaly detection [30].

Complementary to our theoretical work, there also is work [38] to select algorithms and their hyperparameters by matching new datasets to "training" datasets and then recommending the algorithm with associated hyperparameters that did the best on that dataset. Given the plethora of options available to practitioners, this provides useful guidance for model and hyperparameter selection. Unfortunately, this approach does require substantial investment upfront on the training

phase and is not very effective if there is not a good match on a new dataset to one of those included in the training phase.

2.2 Experiments

The relative lack of foundational treatment is in contrast to a large and well-established body of literature on practical outlier detection methods. A relatively recent survey provides an excellent overview on both classical and deep learning based methods, as well as the fundamental challenges with theoretical treatments [28]. In our experimental illustration of the Exile property (in Chapter 5), we employ two well-established classical methods: Local Outlier Factor (LOF) is a nearest neighbour based method that determines outlier status based on local densities [8], and Isolation Forest (iForest) is a tree-based ensemble method [22]. For both, we employ a standard implementation from the scikit-learn library [25]. We selected LOF [8] because of its reliance on only distances as input and iForest [22] was selected because of its continuing popularity for industry use [3, 12, 15, 23] and ongoing research applicability [35]. Deep learning methods have been growing in popularity in high-dimensional and complex settings [28] (such as images). However deep learning is inherently non-interpretable, which makes the task for developing a sound foundational basis for reasoning about algorithms more pressing.

In our other experiments we also tested OneClassSVM using the standard imple-

mentation from the scikit-learn library [25]. Using implementations by the authors we also included Extended Isolation Forest [17] and PIDForest [13] in our tests.

2.3 Datasets

In addition to synthetic datasets that we generated, we also used some real world datasets, from the UCI Machine Learning Repository [18]. We decided to use the specific real world datasets because of their use in previous work [13], where they were carefully selected for their suitability for use as an anomaly detection task and their variety. Some of the datasets are edited versions of the originals from the UCI Repository and the edited versions can be found in the Outlier Detection DataSets (ODDS) Library [27]. More details on the datasets can be found in Section 5.1.

2.4 Isolation Forest bands of open space

During testing it was observed that iForest would sometimes produce infinite bands of axis aligned open space (areas where the scores iForest gave made any points there very likely to be labelled as an inlier). We discovered that, while substantial work has been done towards addressing the observed problems with modified versions of the original iForest, use of the original iForest is still common [3, 12, 15, 23, 35]. One study tried to address this problem by using random hyperplanes through the

points so that the splits do not favor the axes [16], instead of using axis aligned splits (i.e. splits based on one feature at a time), as done in iForest. This approach, called Extended Isolation Forest, did add some overhead in terms of runtime but does reduce the open space problem. A later study claimed that this did not adequately address the issue and created a new variant that used hyperspheres at each node instead of just a split though the infinite space [11]. While this approach does address the open space issue as the hyperspheres can only contain finite space, it comes at a substantial runtime cost compared to iForest.

2.5 Interpretable outlier detection

We view our work on property-based (or axiomatic) analysis of outlier detection methods as one way of facilitating explanations of algorithms' behaviour to users. Interpretability (or explainability) for outlier detection is an active area of research with many approaches being proposed. Our exploration of the interpretability space was one of the main motivations to take an axiomatic approach instead of depending on post-hoc approaches that are not always faithful to the underlying model or model-specific approaches that do not easily generalize to new models. The hope is that by creating an axiomatic basis that can be built upon, we will improve the quality of interpretability in the space overall.

One of the approaches to interpretability for outlier detection was based on

using supervised deep neural networks with explainability added by using gradient-based techniques to determine which input features were the most important for a prediction made [4]. However, most approaches used unsupervised learning, as labelled data for anomaly detection is not typically available. One of those approaches built on iForest by adding explanations for iForest predictions. It is called DIFFI (Depth-based Isolation Forest Feature Importance) and it used the structure of the trees to extract which features were important locally for a given prediction as well as globally across all the trees [9]. Other approaches focused on using the reconstruction error of generative neural networks as the basis for anomaly detection, with explanations extracted from the model's gradients [24, 29]. Instead of starting with uninterpretable models, as in the previous approaches, another study decided to start by building interpretable models [37]. In that study a two-stage process was used. In the first stage, decision trees were trained to predict each feature based on the remaining features. These trees were then fed random samples and those with high disagreement compared to the predictions from the trees were considered outliers and used to augment the training set so that an equal number of outliers to inliers were available. Using the augmented dataset, decision rules were extracted to identify outliers. The study claims that both stages were interpretable and thus the final output is also interpretable.

3 Theory

3.1 Setup

3.1.1 Notation and basic definitions

We use standard mathematical notation for sets and functions, such as $[n] = \{1, 2, \ldots, n\} \subset \mathbb{N}$ for the set of the first n natural numbers and $\mathbb{R}^+$ for non-negative real numbers. A function $\pi : [n] \to [n]$ is a *permutation* of this set if π is both one-to-one and onto.

Most practical anomaly detection methods take in a finite set of training data-points $S = (x_1, x_2, \ldots x_n)$, where each point x_i is a member of a larger ground set or *domain* X. Typically, the points are d-dimensional feature vectors, that is $X \subset \mathbb{R}^d$. Given such training data, method A outputs a real valued function $f = A(S)$, where $f : S \to \mathbb{R}$ or $f : X \to \mathbb{R}$, that assigns each point in the dataset or each point in the domain a score. Outlier status is then determined by thresholding this function f. That is, a point is declared an outlier if $f(x) \geq \theta$ for some threshold

$\theta \in \mathbb{R}$ (and there are various heuristics for determining this threshold). For our axiomatic framework, we for now focus on the former view, and analyze methods that determine outlier status only for points in training data. Additionally, we also consider the thresholding criteria to be the algorithm's responsibility, thus requiring the output of the algorithm to be a set of indices of the outliers, meaning we will require $f(S, d) \subseteq [n]$. This allows us to provide a clean abstraction, and is the relevant view for applications such as data cleaning. However, for the experiments we will focus on the latter, predicting on the entire domain and returning scores. Many methods support that capability, and their behaviour on points unseen during training is often quite important.

Distance-based anomaly detection The *data* $S = (x_1, x_2, \ldots x_n)$ of size n is a sequence of points indexed by $[n]$. Slightly abusing notation, we will use $|S| = n$ for the length of sequence $S = (x_1, x_2, \ldots x_n)$, and $[|S|] = [n]$. Throughout this work we will assume that $n \geq 4$. We use the notation S_{set} to denote the set of points from the sequence S. We assume the data is given together with a *distance function* $d : S_{\text{set}}^2 \to \mathbb{R}^+$ such that (S_{set}, d) is a metric space. That is, d satisfies symmetry, the triangle inequality and $d(x, z) = 0$ if and only if $x = z$. Note that for points x_i, x_j from a sequence $S = (x_1, x_2, \ldots x_n)$ we do allow $d(x_i, x_j) = 0$ for indices $i \neq j$ modeling that the given data may contain repetitions. If $d(x_i, x_j) = 0$ for $i \neq j$,

then we have $d(x_i, x_k) = d(x_j, x_k)$ for all $k \in [n]$. We will mostly refer to the data given as a pair (S, d) of the sequence of data-points S and distance function d.

Definition 1 (Anomaly detection method)**.** *An* anomaly (or outlier) detection method f *takes in data and distances* (S, d) *and outputs a subset* $f(S, d) \subseteq [n]$ *of the indices.*

Below we will formulate properties that, we would argue, an anomaly detection method may reasonably be expected to satisfy, such as being invariant under certain transformation (for example scaling or permuting) of the data. We now start with formally defining such transformations in preparation for stating our axioms. For a distance function d and positive scalar $\alpha \in \mathbb{R}^+$, we let αd denote a *scaled version* of d defined by $\alpha d(x, z) = \alpha \cdot d(x, z)$ for all pairs (x, z) in the domain of d.

Definition 2 (Permutation of data and distance function)**.** *Let* $S = (x_1, x_2, \ldots, x_n)$ *be a dataset and let* d *be a metric over* S_{set}*. We say that* (S', d') *for data* $S' = (y_1, y_2, \ldots, y_n)$ *and distance metric* d' *on* ${S'}_{\text{set}}$ is a permutation of (S, d) *if there exists a permutation* $\pi : [n] \to [n]$ *on the index set of* S *such that*

$$S' = (y_1, y_2, \ldots, y_n) = (x_{\pi(1)}, x_{\pi(2)}, \ldots, x_{\pi(n)})$$

and

$$d'(y_i, y_j) = d(x_{\pi(i)}, x_{\pi(j)})$$

for all $i, j \in [n]$.

Note that invariance to permutations is not a property that is specific to the outlier detection task. The next definition prepares the ground for our consistency property, which does capture an anomaly detection specific requirement.

Definition 3 (Consistent transformation)**.** *Let f be an outlier detection method and let (S, d) a pair of data and distance function. We call a distance metric d' on S_{set} a* consistent transformation *of d with respect to f if the following holds:*

- *for all $i, j \in [|S|] \setminus f(S, d)$ we have $d'(x_i, x_j) \leq d(x_i, x_j)$*
- *for all $k \in f(S, d)$ and $i \in [|S|]$ we have $d'(x_i, x_k) \geq d(x_i, x_k)$*

That is, a metric d' is a consistent transformation of metric d with respect to method f if inliers are pairwise closer to each other under d' in comparison to d and outliers are further from all other points (inliers as well as outlier) under d' in comparison to d. Intuitively, under the transformed distance function d' inliers are "more clearly inliers" and outliers are "more clearly outliers" in comparison to their distances under d.

3.1.2 Axioms for anomaly detection

We now introduce a basic set of desirable properties (or axioms) for anomaly detection. The first three axioms are analogous to axioms that have been proposed for clustering tasks in earlier work [19]. However, note that in particular the Consis-

tency axiom below is phrased very differently here to model properties of anomaly detection tasks (rather than clustering tasks). Likewise Exile, the fourth axiom in the list below, captures a property that is a natural requirement for an outlier detection method: if a point is further away from all other points, than any other pair of points from each other, then this point should be declared an outlier.

1. **Scale Invariance** (SI) - For all inputs (S, d) and any $\alpha > 0$, we have $f(S, d) = f(S, \alpha d)$.

2. **Richness** (R) - For all subsets $A \subseteq [n]$ of the set of indices, with $|A| \leq \frac{|S|}{2}$, there exists a distance metric d on S_{set} with $d(x_i, x_j) > 0$ for all $i \neq j$ such that $f(S, d) = A$.

3. **Consistency** (C) - For any input (S, d), if (S, d') is a consistent transformation of (S, d), then it is the case that $f(S, d) = f(S, d')$.

4. **Exile** (E) - If there is an index $e \in [|S|]$ such that the maximum pairwise distance in $S \setminus \{x_e\}$ is less than the distance from x_e to its nearest neighbour then $e \in f(S, d)$. That is

$$\min\{d(x_e, x_i) \mid i \in [|S|] \setminus e\} > \max\{d(x_i, x_j) \mid i, j \in [|S|] \setminus e\}$$
$$\Rightarrow \quad e \in f(S, d)$$

5. **Permutation Invariance** (PI) - For any input (S, d) if (S', d') is a permutation of (S, d) witnessed by $\pi : [n] \to [n]$, then it is the case that $\pi(f(S, d)) = f(S', d')$.

3.2 Relationships between the Axioms

In this section, we explore how our above-stated axioms relate to each other. In the first part (Subsection 3.2.1), we show that the first three axioms are inconsistent. This means that they cannot be satisfied simultaneously by any anomaly detection method, and thus any practitioner will need to make a choice between them. We then (in Subsection 3.2.2) show that any pair of the first three axioms is consistent, and moreover any pair is satisfiable independently of the status of the last two axioms (any pair is satisfiable simultaneously with any subset of (E) and (PI)).

Our proofs often require defining a distance function on some number of points. We mainly do so by "placing points in a Euclidean space". For example, to define distance function d on points x_1, x_2 and x_3, we will say we place points in a two-dimensional Euclidean space at $x_1 = (0, 0), x_2 = (1, 0)$ and $x_3 = (0, 1)$. This implicitly defines distance d with $d(x_1, x_2) = d(x_1, x_3) = 1$ and $d(x_2, x_3) = \sqrt{2}$. Defining the distance function implicitly this way relieves us from needing to verify that the stated distances indeed satisfy the properties of a metric (since these are now just inherited from the Euclidean metric).

3.2.1 Inconsistency of (SI), (R) and (C)

We start our explorations by proving that the three axioms (SI), (R) and (C) are not jointly satisfiable. The result is similar to the corresponding inconsistency result for clustering methods; however, our proof is original. Since our axioms, in particular our consistency property, are different and tailored to the anomaly detection task, we develop a novel line of argument for our anomaly detection axioms.

Theorem 1. *No anomaly detection method f can simultaneously satisfy the axioms Scale Invariance (SI), Richness (R) and Consistency (C).*

Proof. By way of contradiction, assume that f is an outlier detection method that satisfies the axioms (SI, R) and (C). Let S be a dataset consisting of four points $S = (x_1, x_2, x_3, x_4)$.

By the Richness axiom (R), there exists a distance metric d_1 on S, assigning strictly positive distances, such that

$$f(S, d_1) = \{1\},$$

and another distance metric d_2, assigning strictly positive distances, such that

$$f(S, d_2) = \{1, 2\}.$$

Let $a \in \mathbb{R}, a > 0$ be a positive number smaller than the minimum distance

among two points in S according to d_1, that is

$$0 \ < \ a \ < \ \min\{d_1(x_i, x_j) \mid i, j \in \{1, 2, 3, 4\}, i \neq j\}.$$

We further let $b \in \mathbb{R}$ be a number that is larger than the distance of the furthest point from outlier x_1 according to d_1, that is

$$b \ > \ \max\{d_1(x_i, x_1) \mid i \in \{2, 3, 4\}\}.$$

Then a distance metric d_1' obtained by placing the points on the real line with $x_2 = 0, x_3 = a/2, x_4 = a$ and $x_1 = a + b$ is a consistent transformation of d_1 with respect to f and the Consistency axiom (C) thus implies

$$f(S, d_1') = f(S, d_1) = \{1\}.$$

For a similar transformation of d_2, let $c \in \mathbb{R}, c > 0$ denote a number that is smaller than any pairwise distance according to d_2 and we let $g, h \in \mathbb{R}$ denote two numbers that are larger than any pairwise distance according to d_2 which additionally satisfy $h > g$. Then the distance metric d_2' obtained by placing the four points on the real line with $x_2 = 0, x_3 = g, x_4 = g + c$ and $x_1 = g + c + h$ is a consistent transformation of d_2 with respect to method f. We thus get

$$f(S, d_2') = f(S, d_2) = \{1, 2\}.$$

Now let $\alpha > 0$ be a scaling factor small enough, so that $\alpha \cdot (g + c) < a/2$. Then the distance metric d_1'' obtained by placing the points at $x_2 = 0, x_3 = \alpha \cdot g, x_4 = \alpha \cdot (g + c)$

and $x_1 = a + b$ is a consistent transformation of d'_1 (since the distances among the inliers only got smaller, and the outlier moved further away from all other points), and we thus have

$$f(S, d''_1) = f(S, d'_1) = \{1\}.$$

Now let d''_2 be a distance metric obtained by scaling d'_2 with factor α. Note that d''_2 can also be obtained by placing points $x_2 = 0, x_3 = \alpha \cdot g, x_4 = \alpha \cdot (g + c)$ and $x_1 = \alpha \cdot (g + c + h)$. By the Scale Invariance axiom (SI), we have

$$f(S, d''_2) = f(S, d'_2) = \{1, 2\}.$$

Now note that the distances among the points x_2, x_3 and x_4 are identical in both d''_1 and d''_2. Further, x_1 is an outlier according to f with regard to both metrics, which allows its pairwise distances to be increased in a consistent transformation. Thus, one of d''_1 and d''_2 is a consistent transformation of the other (depending on whether $a + b$ is smaller or larger than $\alpha \cdot (g + c + h)$). This implies that their output should be equal by (C), a contradiction to

$$f(S, d''_1) = \{1\} \neq \{1, 2\} = f(S, d''_2).$$

Thus, no outlier detection methods can satisfy all three axioms simultaneously. □

Remark 1. *The proof of Theorem 1 above requires that for the distance functions d_1 and d_2 of the two initial configurations (which are obtained by invoking the Richness*

axiom) all pairwise distances are strictly positive. We note here that Consistency and Scale Invariance are actually satisfiable jointly with a weakened Richness axiom that does not require pairwise distances to be strictly positive (as our suggested phrasing of (R) does). Consider a method duplicate *that outputs* $[|S|]$ *(that is, all points are outliers) if all distances are strictly positive, and, in case there are pairs of points with pairwise distance* 0*, the algorithm outputs this group of points as inliers and all remaining points as outliers. Note that distance* 0 *is transitive, thus such points form clusters within which all points have pairwise distance* 0*; scaling or consistent transformations could join such clusters, but not separate these points. Formally:*

$$\mathsf{duplicate}(S, d) = \{i \in [|S|] \mid d(x_i, x_j) > 0 \text{ for all } j \neq i, j \in [|S|]\}$$

Intuitively, this method recognizes a point as an inlier if and only if it has at least one duplicate point. The method duplicate *satisfies (SI) and (C), as well as (E) and (PI), and also a modified Richness axiom that is fulfilled by setting distance* 0 *between all required inliers and positive distance from the required outliers to all other points. However* duplicate *is arguably not a reasonable anomaly detection method, and we thus chose to focus on the here proposed more strict requirement of Richness.*

3.2.2 Any pair of (SI), (R) and (C) plus (E, PI) is consistent

Now, we show that each pair of the three axioms (SI) (R) and (C) is consistent and can be satisfied with or without any (subset) of (E) and (PI). More precisely, we will prove the following result:

Theorem 2. *Each of the pairs of axioms (SI, C), (R, SI) and (C, R) is consistent. That is, there exists anomaly detection methods that satisfy the pair. Moreover, the axioms (E) and (PI) are independent of each pair. That is, for each pair, there exists methods that satisfy the pair together with none, both or exactly either one of two additional axioms (E) and (PI).*

The remainder of the subsections in this chapter are devoted to prove this result, a subsection being devoted to each pair of (SI, R), (C, R) and (SI, C). For the proofs, we will define a variety of decision rules based on a single nearest neighbour (snn) scoring rule. We consider the following function that assigns each data-point a score equal to the distance to its nearest neighbour:

$$\mathsf{snn}(x, S) = \min\{d(x, z) \mid z \in S, z \neq x\}$$

If the data set S is clear from context, we may omit the second argument in the above notation. We will use the notation $\mathsf{nn}(x, S)$ (or $\mathsf{nn}(x)$ if S is clear from context) to denote a point in the dataset that minimizes the distance to x:

$$\mathsf{nn}(x, S) \in \operatorname{argmin}\{d(x, z) \mid z \in S, z \neq x\}$$

Observation 3. *Let (S, d) be a given dataset so that (S_{set}, d) is a metric space. Let $x \in S_{\text{set}}$ and $z = \mathsf{nn}(x, S)$ Then*

$$\mathsf{snn}(z) \leq \mathsf{snn}(x)$$

To see this, observe that as long as $z = \mathsf{nn}(x, S)$, then the nearest neighbour of z is not further away than $d(x, z) = d(z, x)$. We also note that the above inequality is strict if and only if x is not also a nearest neighbour of z.

3.2.3 Scale Invariance (SI) and Consistency (C)

To show that (SI) and (C) are jointly satisfiable, we define an anomaly detection method singlemax that always outputs a single index, namely the index of a point whose snn score is largest. In case there is a tie at the largest snn score, singlemax chooses the smallest index. That is

$$\mathsf{singlemax}(S, d) = \min\{\text{argmax}_{i \in [|S|]} \mathsf{snn}(x_i)\}$$

Note that $\mathsf{singlemax}(S, d)$ always returns a single index from $[|S|]$.

Observation 4. *The method* singlemax *defined above satisfies Scale Invariance (SI), Consistency (C) and Exile (E) but does not satisfy (R) nor (PI).*

Proof. Scale invariance (SI) and Exile (E) are obviously satisfied. For some (S, d), let i be so that $\mathsf{singlemax}(S, d) = \{i\}$. Let (S', d') be a singlemax-consistent transformation of (S, d). Note that then d' can only increase the distances of the single

outlier point from all other points and only decrease pairwise distances between inliers. Thus we have

$$d'(x_i, x_j) \geq d(x_i, x_j) \quad \text{for all } j \neq i$$

and

$$d'(x_j, x_k) \leq d(x_j, x_k) \quad \text{for all } j, k \neq i$$

and thus no point of index other than i can have a larger snn score according to d' than according to d. Therefore we have

$$\mathsf{singlemax}(S, d') = \{i\}.$$

Thus the function $\mathsf{singlemax}$ also satisfies Consistency (C). To see why the function does not satisfy Permutation Invariance (PI), consider two datasets of points in $\mathbb{R}$, each equipped with the standard distance $d(x, z) = |x - z|$ for two real numbers. $S_1 = (-3, -2, 10, 15)$ and $S_2 = (15, 10, -3, -2)$ obtained through the permutation $\pi : 1 \mapsto 3, 2 \mapsto 4, 3 \mapsto 2, 4 \mapsto 1$. These are permutations of each other, while we have $\mathsf{singlemax}(S_1, d) = \{3\}$ and $\mathsf{singlemax}(S_2, d) = \{1\} \neq \{\pi(3)\}$. Richness is not satisfied by $\mathsf{singlemax}$ since it never outputs a set of size other than 1. □

Remark 2. *Trivial methods that either always output $\emptyset$ or always output $[n]$ for sequences S of length n satisfy (SI), (C) and (PI), showing that the latter property is independent of the former two. To see that (E) is also independent of these two*

axioms, consider a method that always outputs index set $\{1\}$ *(for any* (S, d)*). This function satisfies (SI) and (C) but not (E) (and not (PI) either).*

We summarize the results of this section in Table 3.1 (where we use either an above-defined name or the constant output to denote an outlier detection method).

	(SI)	(C)	(E)	(PI)
singlemax	✔	✔	✔	✗
$\emptyset$	✔	✔	✗	✔
$[n]$	✔	✔	✔	✔
$\{1\}$	✔	✔	✗	✗

Table 3.1: Summary of methods satisfying (SI) and (C), but not (R)

3.2.4 Richness (R) and Scale Invariance (SI)

To see that this pair can be satisfied, we again consider the scoring function snn, but this time with a different decision rule. We define an outlier detection method scale$-\alpha$ as follows: for a value $\alpha \in [0, 1]$ the method scale$-\alpha$ determines a score threshold as $\alpha \cdot \max_{i,j \in [|S|]} d(x_i, x_j)$. All indices of points whose snn score is greater

than this threshold are determined as outliers; that is, the corresponding indices are in $f(S, d)$. Formally, for some $0 < \alpha < 1$, we define

$$\mathsf{scale}{-}\alpha(S, d) = \{k \in [|S|] \;\mid\; \mathsf{snn}(x_k) > \alpha \cdot \max_{i,j \in [|S|]} d(x_i, x_j)\}$$

Observation 5. *For $\alpha = 0.5$ the anomaly detection method $\mathsf{scale}{-}\alpha$ satisfies the four axioms (SI, R, E), and (PI), but does not satisfy (C).*

Proof. (SI, PI) are obviously satisfied by this anomaly detection rule. To see why it also satisfies (R), let S be a sequence of data-points and let $A \subseteq [|S|]$ be a subset of indices. Without loss of generality, we can assume $A = [m] = \{1, 2, \ldots, m\}$ for some $m \leq \frac{|S|}{2}$. We first also assume $m \geq 2$ (and will treat the cases where $|A| \in \{0, 1\}$ separately below). We now define distances by placing the points from S in an $m + 1$-dimensional Euclidean space and let d be the induced Euclidean distances. We place the first m points on the locations of the first m unit-vectors. Note that these then have pairwise distance $\sqrt{2}$. We place the remaining points $x_{m+1} \ldots x_n$ on different locations that have entry 0 in the first m coordinates and an entry in, say, the interval $[0, 0.5]$ with exactly one point sitting at the origin (the all-zero vector). Now, the largest pairwise distance in this dataset is $\sqrt{2}$, the distance between two distinct unit vectors, and the threshold for the snn scores is

$\sqrt{2}/2$. Note that we have

$$\mathsf{snn}(x_i) \begin{cases} = 1 > \sqrt{2}/2 & \text{if } i \leq m \\ \leq 0.5 < \sqrt{2}/2 & \text{if } i > m \end{cases}$$

and thus we get

$$\mathsf{scale}{-}\alpha(S, d) = [m] = A$$

as required. In case $|A| = 0$ (that is, we need to set distances such that $\mathsf{scale}{-}\alpha$ does not assign any point the outlier label), we can place all points consecutively on the real line, say point x_i at location $i \in \mathbb{N}$, then $\mathsf{snn}(x_i) = 1$ for all points while the maximal pairwise distance in the dataset is $n{-}1 > 2 = 2{\cdot}\mathsf{snn}(x_i)$ for $n \geq 4$. Thus all points are declared inliers for this set of distances. For the case $|A| = 1$, and without loss of generality $A = \{1\}$, we place x_1 at the origin $x_1 = 0$, the second point at distance 1 from x_1, say $x_2 = 1$, and the remaining points at distinct locations in the interval $[1, 1.5]$. Then the maximal pairwise distance in the dataset is larger than 1 but less than 1.5. Thus the point x_1 with $\mathsf{snn}(x_1) = 1 > 0.75 = 0.5 \cdot 1.5$ is declared an outlier while all other points satisfy $\mathsf{snn}(x_i) < 0.5 < 0.75$, and are therefore declared inliers. Thus, the Richness axiom is satisfied by $\mathsf{scale}{-}\alpha$ for $\alpha = 0.5$.

To see why the rule does not satisfy Consistency, consider the above configuration (for the case $m > 1$) with moving the first point out to have entry 10 in the first coordinate (that is, scale the first unit-vector with a factor of 10). Observe that this corresponds to a consistent transformation d' of the distances, since

only the distances involving outlier x_1 increased, and all other pairwise distances remained the same. But now the threshold for outliers is larger than 5 and thus the index set $\{1\}$ containing only the index of this first point will be the result of $\mathsf{scale}{-}\alpha(S, d')$ (rather than the first $[m]$ as before). Thus, the method $\mathsf{scale}{-}\alpha$ thus does not satisfy C.

Finally, we argue that (E) is also satisfied. Consider data (S, d) and a point x_e such that

$$\mathsf{snn}(x_e) > \max_{i,j \in [|S|] \setminus \{e\}} d(x_i, x_j). \tag{3.1}$$

It follows that x_e is one of the two points that maximize the pairwise distance over the whole dataset.

$$\max_{i,j \in [|S|]} d(x_i, x_j) = \max_{i \in [|S|] \setminus \{e\}} d(x_e, x_i).$$

Let k be the index of the other endpoint (or an index of one possible other endpoint in case of ties), that is

$$k \in \mathrm{argmax}_{i \in [|S|] \setminus \{e\}} d(x_e, x_i).$$

Let i denote the index of (one of) x_e's nearest neighbour(s) in S, that is $x_i = \mathsf{nn}(x_e)$. Since d satisfies the triangle inequality, we then get

$$d(x_e, x_k) \leq d(x_e, x_i) + d(x_i, x_k) < 2 \cdot d(x_e, x_i) = 2 \cdot \mathsf{snn}(x_e),$$

where the strict inequality follows from x_e being an exile point (see Equation 3.1).

This now implies

$$\mathsf{snn}(x_e) \geq 0.5 \cdot d(x_e, x_k) = \alpha \cdot \max_{i,j \in [|S|]} d(x_i, x_j),$$

which in turn implies $e \in \mathsf{scale}{-}\alpha(S, d)$. Thus this rule satisfies the Exile property.

□

Remark 3. *We again note that Exile and Permutation Invariance are not implied by the pair of axioms under consideration here (Richness and Scale Invariance). To see that the latter two can be satisfied without Exile, we can simply add an exception to the case where the data contains an exile point (note that at most one exile point can be present in any dataset). We let* $\mathsf{exiscale}{-}\alpha$ *denote the rule that acts exactly as* $\mathsf{scale}{-}\alpha$*, except that any exile point will be assigned inlier status. This rule does not satisfy (E), while still satisfying (R) and (SI).*

For a rule that is not permutation invariant, consider $\mathsf{scale}{-}\alpha_i$ *with a position-dependent threshold scaling parameter* α_i*. For a dataset* S *we define*

$$\alpha_i = \min\{\frac{i}{|S|}, 0.5\}$$

Thus α_i *ranges from* $1/|S|$ *to* $1/2$ *for the first half of the instances and then is constant* 0.5 *for the second half of the points. Since the threshold now varies with the positioning of points in the sequence* S*, this rule is not permutation invariant. The rule is still (obviously) scale invariant. That it satisfies Richness can be seen by employing the same construction as in the proof of Observation 5 above (by placing*

the dedicated inlier points sufficiently close to the origin, for example inside the interval $[0, \frac{1}{|S|})$*).*

Finally, to see that (SI, R) are satisfiable without satisfying either (E) or (PI), we consider the rule that adds an exception for exile points as above to scale$-\alpha_i$*, and denote it by* exiscale$-\alpha_i$*.*

	(R)	(SI)	(E)	(PI)
scale$-\alpha$	✔	✔	✔	✔
exiscale$-\alpha$	✔	✔	✗	✔
scale$-\alpha_i$	✔	✔	✔	✗
exiscale$-\alpha_i$	✔	✔	✗	✗

Table 3.2: Summary of methods satisfying (SI) and (R), but not (C); we set $\alpha = 0.5$.

3.2.5 Consistency (C) and Richness (R)

To prove that (C) and (R) are jointly satisfiable , we define yet another decision rule based on the snn scoring function. The rule thres$-\tau$ declares as anomaly all points whose nearest neighbour is further away than some fixed threshold $\tau \in \mathbb{R}$.

Formally for some $\tau > 0$, we define

$$\mathsf{thres}{-}\tau(S, d) = \{i \in [|S|] \mid \mathsf{snn}(x_i) \geq \tau\}.$$

Observation 6. *For any $\tau > 0$ the anomaly detection rule $\mathsf{thres}{-}\tau$ defined above satisfies (C, R, PI) but not (E) nor (SI).*

Proof. (PI) is obviously satisfied by this rule. To see that (C) is satisfied, let (S, d) be a given dataset. Note that Observation 3 implies that, under the $\mathsf{thres}{-}\tau$ rule, the nearest neighbour of any inlier is also an inlier. This implies that under consistent transformations (S, d'), the snn scores under d' of any inliers are smaller or equal to the scores under d, and thus inliers will remain inliers. The nearest neighbours of outliers might be inliers or outliers. However, a consistent transformation must move any outlier further away from all other points. Thus, snn scores of outliers can only increase, and they will remain outliers under consistent transformations.

To see why Richness is satisfied, we can again consider a similar construction as in the argument for (R) in the proof of Observation 6. In the constructions for the case $|A| \geq 2$, we scale the entries of the outlier points to be τ rather than 1 (that is we use scaled unit-vectors), and place the to-be-inliers at distance strictly less than τ from each other. For $|A| = 0$ we place all points in a ball of diameter less than τ and for $|A| = 1$ we place a single point at distance greater than τ from

this ball that contains the remaining points.

Lastly, we argue that (E) and (SI) are not satisfied: to see this we simply consider a scaling of a given dataset such that the maximal distance between points in the scaled version is smaller than τ. This results in all points being inliers, even if there is an exile point among them and even if before scaling only a subset of the data S were inliers. □

Remark 4. *Again, we argue that the axioms (E) and (PI) are independent of the two axioms under consideration here. To show that (C) and (R) can be satisfied without Permutation Invariance, we consider a variation of* thres$-\tau$ *that we will name* specthres$-\tau$*. This rule acts like* thres$-\tau$*, except assigning a special treatment to the first point x_1 in the data in case $n \geq 6$ (in case $n \leq 5$, we let* specthres$-\tau$ *be identical to* thres$-\tau$ *to maintain (R)). For large enough datasets,* specthres$-\tau$ *uses threshold $\tau/2$ for the first point x_1 (only) and threshold τ for the remaining points $x_2, \ldots, x_n$. Additionally, x_1 is assigned to be an inlier only if there are at least two other points within $\tau/2$ distance to it, and x_1 is disregarded for the nearest neighbour calculations for all other points. This does not satisfy (PI), while (C) and (R) are still satisfied. (R) can be seen with the same construction as for the original* thres$-\tau$ *rule. For Consistency, note that the status (inlier versus outlier) of the first point cannot change under consistent transformations. In the case that the first point is an inlier, the two other points within distance $\tau/2$ are both also inliers*

(since they each have a point other than x_1 within distance τ). Thus, these three points will only get closer under consistent transformations and the assignment, for all three of them, will remain inlier. On the rest of the dataset S without the first point, the new rule acts exactly as thres$-\tau$*. Thus, (C) is satisfied (but (E) is not).*

Now, to see that (C, R, PI) are satisfiable together with (E) we introduce a modification of thres$-\tau$*, that we will name* exthres$-\tau$*. This rule will act exactly like* thres$-\tau$ *except that it will always declare exile points (note that there can be at most one exile point in any dataset) as outliers. Thus, in the case that all* snn *scores in data (S, d) are below τ, but one of the points is an exile point, this point will then be the single outlier. In the case where some* snn *scores are larger than τ, (E) is already satisfied. Note that, for the former case, any consistent transformation of such a configuration will decrease distances between inlier points and can only increase the distance of the exile point from other points, and thus this point will remain the single outlier under any consistent transformation. The latter case is already covered by the consistency of* thres$-\tau$*.*

Finally, we can combine the two rules introduced here to obtain an outlier detection rule specexthres$-\tau$ *that does satisfy Exile but not Permutation Invariance.*

Again, we summarize the behaviours established in this section in Table 3.3.

	(C)	(R)	(E)	(PI)
thres$-\tau$	✔	✔	✗	✔
specthres$-\tau$	✔	✔	✗	✗
exthres$-\tau$	✔	✔	✔	✔
specexthres$-\tau$	✔	✔	✔	✗

Table 3.3: Summary of methods satisfying (C) and (R), but not (SI)

4 Tools

As part of this work we also built two tools that facilitated experimentation. The first is a framework to setup, run and report on experiments. The second is a Graphical User Interface (GUI) tool to create and edit datasets.

4.1 Experiment Framework

The experiment framework is logically divided into two parts (but both are part of the same code base)[1]. The first handles outlier dataset generation and the second manages experiments. All parts of this tool are written in python and configuration for the tool is done via a dedicated configuration file in the code. However, even though the configuration is done in code, employing the tool is still very approachable for someone without experience with python because configuration is very similar to just key-value pairs. The configuration has default values provided for all options so only values of interest need to be specified. Unless indicated oth-

[1]Code available at `https://github.com/uruth-lab/experiment_framework`

erwise below, all generation processes support any number of dimensions greater than 0.

4.1.1 Dataset Generation

Four main choices are required to generate a dataset. The first is how many data-points should be generated in total. The second is what percentage of those points should be outliers. The third and fourth are what processes should be used to generate the inliers and outliers respectively.

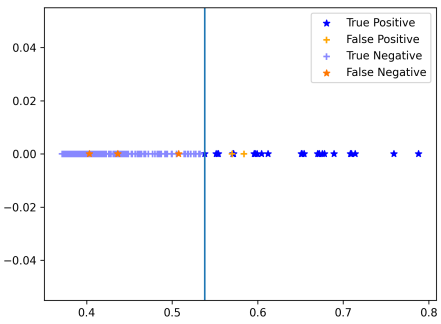
The following generation processes have been implemented and can be used for either inliers or outliers:

- gaussian - generates points from a Gaussian distribution with the option to choose the variance for each dimension.

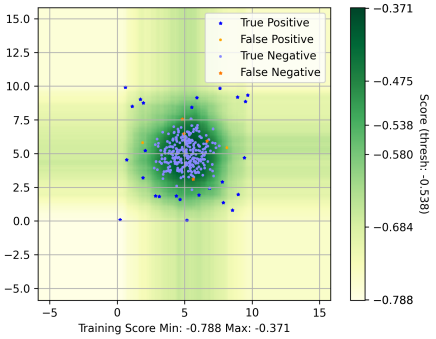
- polynomial - generates points along the graph of a polynomial function parametrized by the coefficients of the polynomial. (Only supports 2 or more dimensions, with 0's set for all dimensions above 2).

- sine - generates points along the graph of a sine wave parametrized by the desired start and end x-axis values. (Only supports 2 or more dimensions, with 0's set for all dimensions above 2).

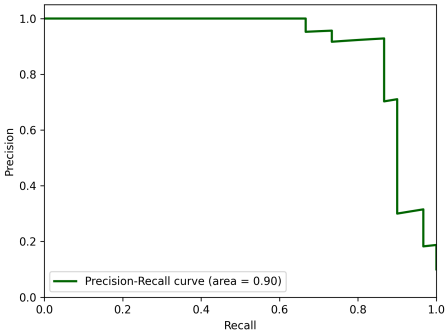
- sphere - not only generates a sphere, but any user specified number of dimen-



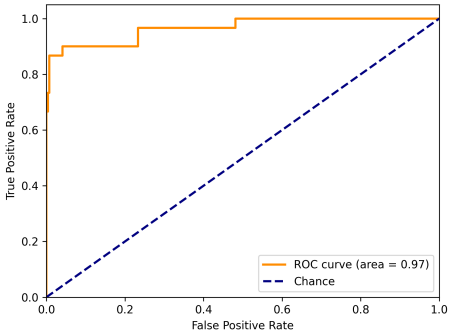
(a) Model Scores



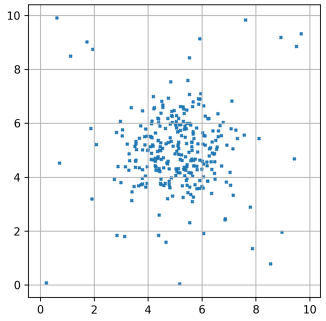
(b) Scatter Plot (Classified Points)



(c) Precision recall curve



(d) ROC curve



(e) Scatter Plot (Raw Points)

Figure 4.1: Visualizations Example

sions (not only three). For example, if two dimensions is selected, it would generate points from a circle. It is worth mentioning that only points on the outer part of the shape are generated. Thus, it produces a circle and not a disk.

- uniform - generates points from a uniform distribution in a hypercube with the option to select the minimum and maximum value for each feature.

The observant reader may have noticed that no options were given for the center of the Gaussian nor the circle. This is because it would have been redundant as all generation processes support specifying a linear transformation to be applied after being generated. This provides rotation, translation and all other linear operations.

All generation processes also support the ability to add (or not add) Gaussian noise to the generated points, with the option for specifying the variance of the noise. The added noise provides a way to have, for instance, a sine-like shape but not exactly a sine wave. There is also the option to add a limiter on the noise added to ensure that the tails of the Gaussian do not lead to points arbitrarily far from the originally generated point.

It is sometimes the case that we wish to ensure that the generated outliers, are sufficiently far from the generated inliers. This was implemented using rejection sampling so that it works with all the generation processes. An example of why

rejection sampling is necessary can be seen in Figure 4.1b where some of the generated outliers are very near the inliers. Since, depending on the settings chosen, the area selected for outliers may be a subset of the area selected for inliers, a maximum rejection threshold prevents infinite loops.

The final setting common to all generation processes is the number of uninformative features to add. Uninformative features are features that provide no signal and test the algorithm's ability to determine useful features. This is implemented using a uniform distribution for those features for all points, that is, both inliers and outliers.

4.1.2 Experiments Management

Experiment management, handles the generation of the actual experiments to run based on the configuration, monitoring the execution of the experiments, running the performance measurements and ensuring the desired output is generated. Each of the management components is explained in the following subsections.

Experiments Configuration

The configuration for which experiments to run is an ordered list of groups of experiment settings. Each of the experiment settings groups in the list have 4 main options. The first is the list of algorithms and their parameters. Each algorithm-

parameters pair can be named so that it is easier to identify in the results. The second is the datasets and the third is a list of performance measurements to use to evaluate the outcomes. Finally, the fourth is how many trials to use for each experiment. For non-deterministic algorithms, many trials helps with assessing average behaviour for randomized algorithms.

When the program is started, it takes each experiment settings group and creates a list of experiments to be run by taking the cross product of the algorithms with the datasets. Each experiment generated from that process is given a sequential ID for traceability throughout the process to make it easy to associate logs, numerical results and generated images to a particular experiment.

Experiments Execution

Once all the experiments are prepared it transitions to training models for each of the configured experiments. Exceptions for each experiment are isolated so that one experiment does not cause all the others to also fail. The tool provides a notification upon completion. The type of notification is operating system dependent. It beeps on Windows and speaks on Linux and Mac using the built-in text to speech capabilities. If any errors occur, they are logged separately and an error count is included in the notification at the end. Each stage (and substage) of the experiments is timed to make it easy to identify which parts take the longest. This is

useful for when iterating on something new, letting the user know how long each stage takes so that it can be skipped when not critical. It also makes it possible to gauge how much longer an experiment is likely to run based on past records.

Performance measurements

Terminology Below we add to the notation defined in Section 3.1.1:

- *True Positive* (TP) - a point predicted as an outlier that is actually an outlier.
- *False Positive* (FP) - a point predicted as an outlier that is actually an inlier.
- *True Negative* (TN) - a point predicted as an inlier that is actually an inlier.
- *False Negative* (FN) - a point predicted as an inlier that is actually an outlier.
- trial will be used as short for "trial for an experiment". We clarify trial here to reduce repetition, since most performance measurements are per trial.
- Instead of f outputting a set of indices as previously used, we will instead work with f outputting scores on the domain X, that is $f : X \to \mathbb{R}$
- *Threshold* refers to the scalar used to convert scores from f into predictions.
- RAW_SCORES - is the set of scores for a model on the training data. That is RAW_SCORES $= \{f(x) \mid x \in S\}$

- AUGMENTED_SCORES - is the same as RAW_SCORES with an additional score added that is larger than all the training scores. That is AUGMENTED_SCORES = RAW_SCORES $\cup$ {max(RAW_SCORES) + 1}. Note the use of 1 here is not significant and any positive scaler value would do.

- SS - is a sorted sequence of scores from the set AUGMENTED_SCORES, sorted from low to high.

- p - is a function $p : \theta \to \mathbb{R}$ that maps from a threshold to the *precision score* for model f on dataset S at threshold θ.

- q - is a function $q : \theta \to \mathbb{R}$ that maps from a threshold to the *false positive rate* for model f on dataset S at threshold θ.

- r - is a function $r : \theta \to \mathbb{R}$ that maps from a threshold to the *recall score* for model f on dataset S at threshold θ.

We will use TP, FP, TN and FN to refer to the number of points in that category, with respect to f. For instance $n = \text{TP} + \text{FP} + \text{TN} + \text{FN}$, because the sum of the categories should equal the size of the dataset.

Performance measurements implemented The tool supports using the following performance measurements to help evaluate experiments:

- scores - returns the raw scores on the training data for the trial (does not support providing a mean and standard deviation across trials like the other measurements do). This is similar to RAW_SCORES except that RAW_SCORES is a set and scores is a sequence and thus may have duplicates. scores is ordered such that each index corresponds to the score for that point in S. That is scores $= (f(x_1), f(x_2), \ldots f(x_n))$

- $\mathsf{min} - \mathsf{score}$ - returns the minimum score on the training data for the trial. That is $\mathsf{min} - \mathsf{score} = \min(\text{RAW\_SCORES})$.

- $\mathsf{max} - \mathsf{score}$ - returns the maximum score on the training data for the trial. That is $\mathsf{min} - \mathsf{score} = \max(\text{RAW\_SCORES})$.

- $\mathsf{f1} - \mathsf{score}$ - returns the best F1 score achieved (across different options for threshold) for a trial. The F1 score is the harmonic mean of the model's precision and recall (both of which are defined below). It can be simplified and expressed in terms of just TP, FP and FN in the following formula:

$$\mathsf{f1} - \mathsf{score} = \frac{2 \cdot \text{TP}}{2 \cdot \text{TP} + \text{FP} + \text{FN}}$$

- $\mathsf{f1} - \mathsf{threshold}$ - returns the threshold value that produces the best $\mathsf{f1} - \mathsf{score}$ for the trial.

- fpr[2] (False Positive Rate) - returns the False Positive Rate at $\mathsf{f1-threshold}$. It is calculated using the following formula:

$$\mathsf{fpr} = \frac{\mathrm{FP}}{\mathrm{FP} + \mathrm{TN}}$$

- tpr[2] (True Positive Rate / Recall) - returns the True Positive Rate (aka Recall) at $\mathsf{f1-threshold}$. It is calculated using the following formula:

$$\mathsf{tpr} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}$$

- $\mathsf{precision}$[2]- returns the precision at $\mathsf{f1-threshold}$. It is calculated using the following formula:

$$\mathsf{precision} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}$$

- $\mathsf{average-precision}$[2] (AP) - the scikit-learn library [25] documentation for *average_precision_score* states, "AP summarizes a precision-recall curve as the weighted mean of precisions achieved at each threshold, with the increase in recall from the previous threshold used as the weight". It is calculated using the following formula:

$$\mathsf{average-precision} = -\Sigma_{k\in[n]} p(\mathsf{SS}_k) \cdot (r(\mathsf{SS}_{k+1}) - r(\mathsf{SS}_k))$$

- $\mathsf{precision-recall-auc}$ (area under the curve) - is an alternative to $\mathsf{average-precision}$ for summarizing the precision recall curve. It uses a func-

[2] Wraps a call to the scikit-learn library [25]

tion from the scikit-learn library [25] to compute AUC for the precision recall curve using the trapezoidal rule. It is calculated using the following formula:

$$\mathsf{precision} - \mathsf{recall} - \mathsf{auc} = -\Sigma_{k\in[n]}\frac{1}{2}\cdot(p(\mathsf{SS}_{k+1}) + p(\mathsf{SS}_k))\cdot(r(\mathsf{SS}_{k+1}) - r(\mathsf{SS}_k))$$

- $\mathsf{roc} - \mathsf{auc}$ (Receiver Operating Characteristic AUC) - uses the same function as $\mathsf{precision} - \mathsf{recall} - \mathsf{auc}$ to computer the AUC of the ROC curve. It is calculated using the following formula:

$$\mathsf{roc} - \mathsf{auc} = -\Sigma_{k\in[n]}\frac{1}{2}\cdot(r(\mathsf{SS}_{k+1}) + r(\mathsf{SS}_k))\cdot(q(\mathsf{SS}_{k+1}) - q(\mathsf{SS}_k))$$

- $\mathsf{open} - \mathsf{space} - \mathsf{band} - \mathsf{detector}$ - is a custom heuristic we added to try to detect infinite bands produced by isolation forest [22]. More details on why infinite bands are problematic and example images can be seen in Section 5.2. It works by iterating over the training data for each feature. For each iteration, it copies the training data, finds the minimum value for that feature, then sets all values for that feature in the copy to a value below the minimum. It then scores modified version of the training data, after which it finds the highest score that extends along that dimension to 2, 10 and 100 times the range of that feature to confirm that the band extends. It then copies the training data again and repeats the test, modifying to above the maximum and takes the higher of the two high scores found. It then converts the highest score found into a percentage and returns that as the output.

More formally, let $S = (x_1, x_2, \ldots x_n)$, drawn from domain $X \in \mathbb{R}^d$. Let f be an anomaly detection method, where $f : X \to \mathbb{R}$. Let $x_i[j] \in \mathbb{R}$ be the value of the j^{th} feature of $x_i \in S$. Let $\alpha_j = \min_{i \in [n]}(\{x_i[j] \mid x_i \in S\})$, that is α_j is the minimum value for feature j for any point in S. Let $\beta_j = \max_{i \in [n]}(\{x_i[j] \mid x_i \in S\})$, that is β_j is the maximum value for feature j for any point in S. Let $y_{\text{train\_min\_score}} = \min_{x \in S}(f(x))$. Let $y_{\text{train\_max\_score}} = \max_{x \in S}(f(x))$. Let $g(x, j, w) : x \mapsto x'$ where the following holds:

1. for all $i \in [d]$ if $i \neq j$ then $x'[i] = x[i]$
2. and $x'[j] = w$

Let $t_j = \beta_j - \alpha_j$.

The max value in a band detected is calculated as follow:

$$
\begin{aligned}
y_{\text{below}} = \max_{j \in [d]} \{ f(z) \mid \quad & z \in \{ g(x, j, \alpha_j - t_j) \mid x \in S \}, \\
& f(z) = f(g(z, j, \alpha_j - 2t_j)), \\
& f(z) = f(g(z, j, \alpha_j - 10t_j)), \\
& f(z) = f(g(z, j, \alpha_j - 100t_j)) \\
\}& \\
y_{\text{above}} = \max_{j \in [d]} \{ f(z) \mid \quad & z \in \{ g(x, j, \beta_j + t_j) \mid x \in S \}, \\
& f(z) = f(g(z, j, \beta_j + 2t_j)), \\
& f(z) = f(g(z, j, \beta_j + 10t_j)), \\
& f(z) = f(g(z, j, \beta_j + 100t_j)) \\
\}& \\
y_{\text{band\_score}} = \max(y_{\text{below}}, y_{\text{above}})&
\end{aligned}
$$

This score, $y_{\text{band\_score}}$ is converted to a percentage (the actual output) using:

$$\left(\frac{y_{\text{band\_score}} - y_{\text{train\_min\_score}}}{y_{\text{train\_max\_score}} - y_{\text{train\_min\_score}}} \right) \% \tag{4.1}$$

This percentage allows the bands observed in the plots to be quantified. The percentage also works when the data is not able to be visualized such as on the real world datasets.

- $\mathsf{percent-predicted-anomalies}$ - returns the percentage of anomalies of the total dataset predicted by the model for a trial. It is calculated using the following formula:

$$\mathsf{percent-actual-anomalies} = \frac{|\{x \in S \mid f(x) \geq \mathsf{f1-threshold}\}|}{n}$$

- $\mathsf{percent-actual-anomalies}$ - uses the ground truth in dataset to calculate the percentage of anomalies for the dataset; not affected by algorithm and only depends on the dataset but provided as a convenience for comparison to $\mathsf{percent-predicted-anomalies}$. Let g be an indicator function $g : x \mapsto \{0, 1\}$ such that $g(x) = 1 \iff$ the ground truth label for x is $\mathsf{outlier}$. It is calculated using the following formula:

$$\mathsf{percent-actual-anomalies} = \frac{\Sigma_{x \in S} g(x)}{n}$$

- $\mathsf{time-trial}$ - returns the wall clock time for each trial; includes the time for training and scoring of the training data.

- $\mathsf{time-experiment}$ - returns the wall clock time that the experiment took to run from start to finish for all trials. May be larger than the sum of $\mathsf{time-trial}$ as it also includes the orchestration time (time taken to run the loop over the trials), which should be negligible.

In addition to the above performance measurements, the tool also supports the following visualizations:

- model − scores - shows scores for the training data along a number line to facilitate understanding how the scores are distributed in relation to misclassifications. See example in Figure 4.1a.

- precision − recall − curve^2- shows the standard precision recall curve for a trial. See example in Figure 4.1c.

- roc − curve^2 (Receiver Operating Characteristic curve) - shows the standard ROC for a trial. See example in Figure 4.1d.

- scatter − plot − classified - shows a scatter plot with the ambient space scored and colored to show a model's behaviour on points other than just the training data. Only works on datasets in $\mathbb{R}^1$ or $\mathbb{R}^2$. See example in Figure 4.1b.

- scatter − plot − raw - shows only the raw training points without any coloring or special shapes. Only works on datasets in $\mathbb{R}^1$ or $\mathbb{R}^2$. See example in Figure 4.1e.

Reporting Results

The tool provides the results of each trial along with the mean and standard deviation where applicable. It also supports exporting results in three formats, a latex table, comma separated values (CSV) and YAML. Each serves a different purpose. The latex table facilities inclusion into latex documents. The main goal with the

CSV version is to facilitate conversion into a spreadsheet for further analysis. And finally the YAML format captures all the data not just the mean and serves as a way to store results in an easily machine readable format.

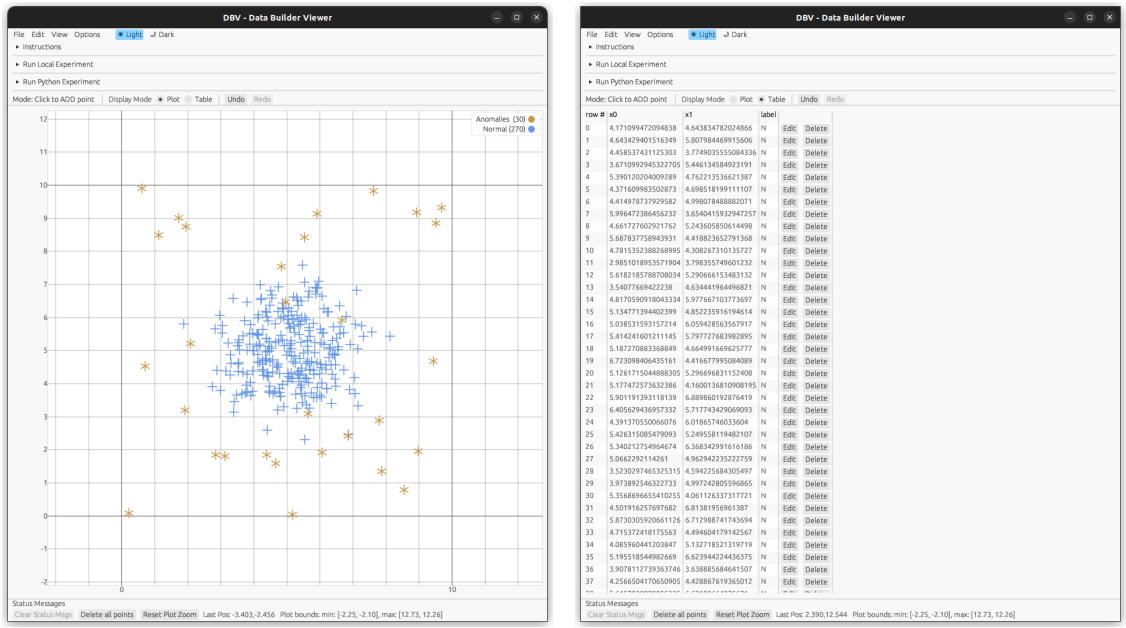
4.2 Data Builder Visualizer (DBV)

We also developed a 2D dataset viewer and editor[3][4]. The main motivation for its development was to be able to easily generate toy datasets to test ideas. The tool has two modes of interaction, plot and table.

plot mode (see Figure 4.2a) allows the user to interact with the dataset by visualizing the points on a plot. In plot mode the tool allows the user to easily add inlier and outlier points using left and right click respectively. The exact value of added points can be made more precise by enabling the rounding feature that allows choosing how many decimal positions a new point should be rounded to. delete is also accomplished by using left and right click after toggling the delete option. Toggling the delete option is done via the options menu or using middle click. For deleting, because clicking exactly on a point is non-trivial, it will delete the point closest to where the click is detected. The tool makes use of both color and shapes to differentiate between inlier and outlier points.

[3]Code available at `https://github.com/uruth-lab/dbv/`

[4]Online version of tool available at `https://uruth-lab.github.io/dbv/`



(a) Plot Mode (b) Table Mode

Figure 4.2: Data Builder Visualizer (DBV)

Table mode (see Figure 4.2b) works similar to a spreadsheet where it shows the list of points and allows the user to edit or delete them as desired. All operations related to the dataset are supported in the undo / redo system so any changes can easily be reversed.

When running natively (i.e., not in the browser), the system is able to save to and load from both MATLAB and CSV files. MATLAB files support is needed because that is the format that is used by the experimentation framework. CSV was added to make it easy to interoperate with spreadsheet software. The tool is also able to be run in the browser without any install required but when run this

way it does not support MATLAB files and can only save to / load from CSV.

In addition to manipulation of datasets, when running in native mode, it can also initiate running the experiment on the dataset currently loaded. This provides an easy way to make changes and with one click start the experiments to see updated results of how the algorithms selected performed.

5 Experiments

To illustrate the benefits of our property-based approach, we examined two well-established outlier detection methods, namely Local Outlier Factor (LOF) [8] and Isolation Forest (iForest) [22]. Specifically we were interested how these methods behave in terms of our Exile property: Would points that are further away from the training data than any pairwise distance between training data-points always be labeled as anomalies? We found that this is not the case for either of these methods.

We start this chapter by first describing the datasets used in our experiments. Then we demonstrate that iForest sometimes produces infinite bands using one of our synthetic datasets, followed by a discussion on its behaviour on the real world datasets. And finally we demonstrate that LOF might be vulnerable to data poisoning.

5.1 Datasets

The datasets used throughout our experiments are listed in Table 5.1 with the exception of those specifically created for demonstrating vulnerability to data poisoning that are described in Section 5.3. The synthetic datasets (generated by us) are:

- diamond - this is a simple hand-crafted dataset with 9 inliers in the shape of a diamond (has no outliers). A visualization of the dataset is available in Figure A.1.
- square - this is a simple hand-crafted dataset with 9 inliers in the shape of a square (has no outliers). A visualization of the dataset is available in Figure A.2.
- normal − uniform - in this dataset the inliers are drawn from a Gaussian distribution. And the outliers are drawn from a uniform distribution. A visualization of the dataset is available in Figure A.3.
- sine − uniform - in this dataset for points $(x_1, x_2) \in \mathbb{R}^2$ the inliers are generated by uniformly sampling x_1 and letting $x_2 = \sin(x_1)$. The outliers are generated the same way but Gaussian noise is added to the final x_2 value. A visualization of the dataset is available in Figure A.4.

- polynomial - in this dataset for points $(x_1, x_2) \in \mathbb{R}^2$ the inliers are generated by uniformly sampling x_1 and letting $x_2 = x_1$. The outliers are generated the same way but Gaussian noise is added to the final x_2 value. A visualization of the dataset is available in Figure A.5.

- circle - in this dataset the inliers are drawn from a uniform distribution over the points on the circumference of a circle and the outliers are drawn from the same distribution but with Gaussian noised added to the x_2 value. A visualization of the dataset is available in Figure A.6.

- sphere - in this dataset the inliers are drawn from a uniform distribution over the points on the circumference of a circle and the outliers are drawn from a uniform distribution. The sphere dataset differs from the circle in how the outliers are generated. In sphere the outliers are uniformly distributed while in circle the outliers are generally closer to the inliers because they are from the same distribution with noise added. A visualization of the dataset is available in Figure A.7.

- single − dimension - this dataset only has a single dimension. It is comprised of 38 inliers sampled from a uniform distribution over the range $[0, 2.5]$, 30 inliers sampled from a uniform distribution over the range $[7.5, 10]$ and 7 outliers sampled from a uniform distribution over the range $[2.5, 7.5]$. A visualization

of the dataset is available in Figure A.8.

- all − but − one - all the inliers have the same x_2 value of zero and the outliers have a non zero value for x_2. A visualization of the dataset is available in Figure A.9.

For the real world datasets we used the versions, including any preprocessing as done by a previous study [17], as explained in Section 2.3. The real world datasets we used are:

- http and smtp - are derived from the KDD Cup 1999 network intrusion task [34] along with the preprocessing as was done in the previous study [17].
- mammography - is a binary classification dataset with imbalanced classes [27].
- musk - is a multi-class classification dataset where classes are combined and divided into inliers and outliers as done in the previous study [17]. The original dataset is the UCI Repository "Musk (Version 2)" [10] but the version used here is the edited version from ODDS Library [27].
- satimage − 2 - is a multi-class classification dataset where classes are combined and divided into inliers and outliers as done in the previous study [17]. The original dataset is the UCI Repository "Statlog (Landsat Satellite)" [33] but the version used here is the edited version from ODDS Library [27].

- siesmic - is a binary classification dataset with imbalanced classes [32].
- thyroid - is binary classification dataset with imbalanced classes [26].
- vowels - is a multi-class classification dataset where classes are combined and divided into inliers and outliers as done in the previous study [17]. The original dataset is the UCI Repository "Japanese Vowels" [20] but the version used here is the edited version from ODDS Library [27].

Dataset	Total Points	Anomaly Percentage	Feature Count
diamond	9	0.00%	2
square	9	0.00%	2
normal-uniform	300	10.00%	2
sine-uniform	400	10.00%	2
polynomial	200	10.00%	2
circle	400	10.00%	2
sphere	1,000	10.00%	2
single-dimension	75	9.33%	1
all-but-one	50	10.00%	2
http	567,498	0.39%	3
mammography	11,183	2.32%	6
musk	3,062	3.17%	166
satimage-2	5,803	1.22%	36
siesmic	2,584	6.58%	15
smtp	95,156	0.03%	3
thyroid	7,200	7.42%	6
vowels	1,456	3.43%	12

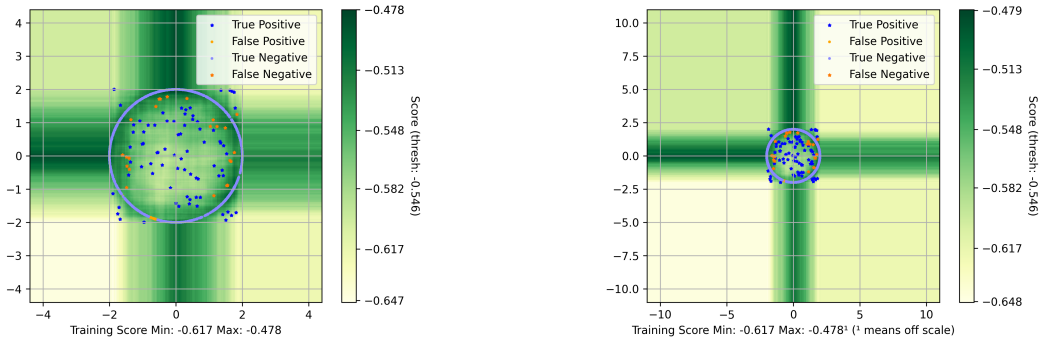
Table 5.1: Datasets Info

5.2 Isolation Forest Infinite bands of normal regions

Isolation Forest is still in current use, as established in Section 2.4. However, we demonstrate that it often produces areas of high likelihood to be inliers that extend infinitely far away from the training data. Examples of this can be found in Figures 5.1, 5.3c and 5.3d. This is problematic because it is desirable that there exists a "reasonable" threshold that does well on the training data and does not cause points infinitely far away to be labelled as inliers. An example of such a threshold might be the one that produces the best F1 score. However, because of the high scores in the bands (likely to be inlier) it means that a point anywhere in the band is likely to be labelled as an inlier.

Sphere Dataset

We first demonstrate the bands on one of our synthetic datasets where the behaviour can be visualized. We employed the standard implementation of iForest from the scikit-learn library [25] and ran it on the Sphere dataset. Figure 5.1a shows the visualization of the data along with the model labelling and Figure 5.1b shows the same plot but zoomed out more to help provide more perspective. The figures show the bands extending both vertically and horizontally. Darker colors in the image correspond to higher scores from the model.



(a) Normal Zoom

(b) Zoomed Out

Figure 5.1: Isolation forest behaviour on Sphere dataset. High band 98.5%

Real world datasets

Using $\mathsf{open} - \mathsf{space} - \mathsf{band} - \mathsf{detector}$, which looks for bands extending away from the training data and reports the highest percentage of the range of training scores found (more detailed description available in Section 4.1), we were able to explore Isolation Forest's behaviour with regard to infinite bands on the real world datasets. Our experiments revealed that the infinite bands produced are much more prevalent on the real world datasets. The results showed that very high-percentage bands were detected in all of the real world datasets as shown in Table 5.2. While bands were also detected in the synthetic datasets (and this is where we first observed them) the magnitude (and thus "inlierness") of the scores found in the synthetic datasets is not as bad as in the real world datsets. This phenomena is very concerning as

it means that in real use cases with real data this problem is more likely to occur. Given that it is so likely to occur it makes it more likely that an attacker with the ability to run enough black box queries against an iForest model may be able to find these bands and utilize them to pass points that are arbitrarily far away from the data the model was trained on to pass as inliers.

In Table 5.2 a higher percentage corresponds to a larger percentage of the range of training scores. Thus, the higher the percentage the more likely points in the band are to be labelled as inliers. For a more detailed explanation of the percentages see Section 4.1 where $\mathsf{open} - \mathsf{space} - \mathsf{band} - \mathsf{detector}$ was described. In all cases for the real world datasets the bands detected exceeded the threshold of the best F1 Score.

Dataset	Highest Percentage Detected in a Band
diamond	21%
square	61%
normal-uniform	32%
sine-uniform	60%
polynomial	46%
circle	84%
sphere	96%
single-dimension	37%
all-but-one	68%
http	100%
mammography	100%
musk	100%
satimage-2	99%
siesmic	100%
smtp	100%
thyroid	98%
vowels	90%

Table 5.2: Isolation forest highest bands detected

5.3 Data poisoning

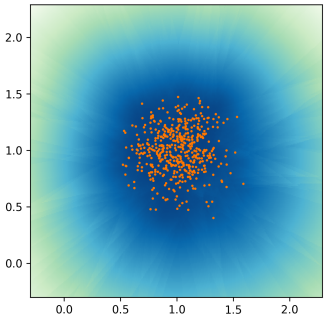
In this section we demonstrate how LOF might be vulnerable to data poisoning: while on our original datasets the method did not show a violation of Exile, a strategic addition of a few datapoints can lead to large areas outside the range of the training data that receive scores in the normal range. We include the behaviour of iForest on these datasets for comparison.

We employed standard implementations from the scikit-learn library [25] for both methods. We generated two types of datasets in a two-dimensional Euclidean space for the sake of visualization: (1) data drawn from a standard Gaussian distribution and (2) data from a lower-dimensional manifold, in our case a sine-shape one-dimensional curve in the two-dimensional space. For both datasets 500 points were generated. We ran both methods on these datasets, and then evaluated the resulting scoring function in the ambient space in order to visualize which areas received high scores, corresponding to inlier or normal assignment, and which areas received lower scores, corresponding to outlier assignment, see Figures 5.2 and 5.3. In these visualization darker color (dark blue) corresponds to a higher score, thus more likely normal assignment, while a lighter color corresponds to a lower score, thus more likely anomalous assignment. For both datasets, we evaluate the method both on the clean data (generated as described above) and on a poisoned

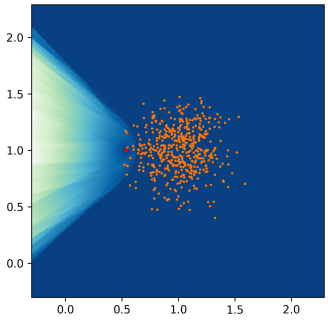
dataset where we added 21 extra points at a tiny distance to each other (in comparison to other distances in the original data). The clean datasets correspond to Figures 5.2a, 5.2c, 5.3a and 5.3c, while the poisoned datasets correspond Figures 5.2b, 5.2d, 5.3b and 5.3d (the locations of the added points are indicated in red).

Results For Local Outlier Factor, we observe the desired behaviour on the "clean" datasets Figures 5.2a and 5.2c. The areas of and immediately around the datapoints are shaded dark (indicating likely normal assignment) while the scores decrease with increasing distance from the training data (the coloring of the ambient space gets lighter towards the edges of the images). However, we show that a strategic addition of 21 densely located "poisoning" points can disturb this behaviour (we needed to add 21 points for the default setting of 20-nearest neighbour based LOF). It is noteworthy that the number of points required for poisoning is only based on the number of nearest neighbours and is independent of the size of the dataset. In Figures 5.2b and 5.2d we observe large areas around the data that still receive high (that is, likely normal) scores. We note that these normal areas do not extend indefinitely for this method, however they do extend to distances substantially larger than the diameter of the training data.

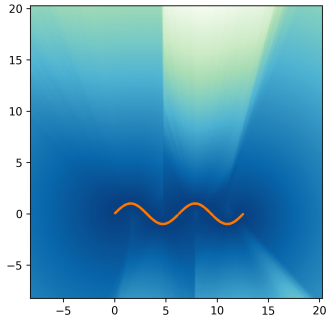
It is also possible that a dataset can be accidentally poisoned by duplicates in the



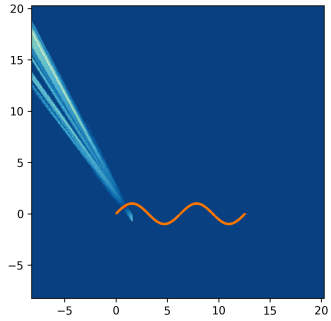
(a) Clean Gaussian



(b) Poisoned Gaussian



(c) Clean Sine

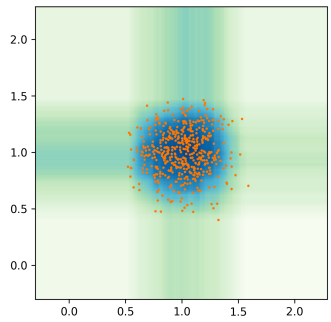


(d) Poisoned Sine

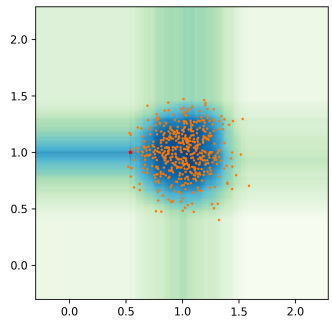
Figure 5.2: Local Outlier Factor behaviour on clean and poisoned datasets. Darker colors are more likely to be inliers. Note the large areas in dark blue that are likely to produce false negatives in (b) and (d).

dataset. This is a consequence of duplicates having zero distance from each other. This is easy to mitigate provided the user is aware of the problem. However, more care than just looking for duplicates is required, as in our example no duplicate points were used, just points that are very near to each other. Thus, care must be taken with regard to sourcing the training data to protect against malicious agents. Counter measures to guard against poisoning are also required if the training data is augmented by runtime queries.

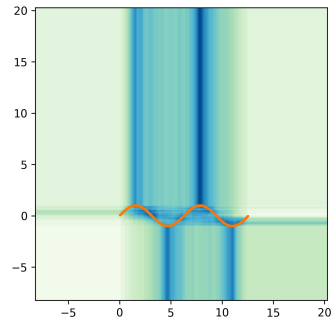
For Isolation Forest, we did not observe such a clear vulnerability to data poisoning. However, as Figures 5.3c and 5.3d illustrate, the method also produces "bands" of normal scores that extend indefinitely on this dataset.



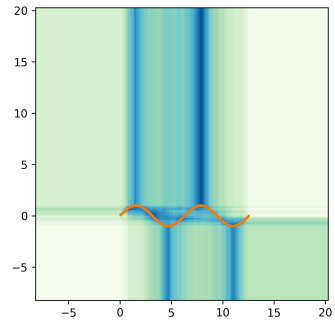
(a) Clean Gaussian. High band 35.3%



(b) Poisoned Gaussian. High band 62%



(c) Clean Sine. High band 99.1%



(d) Poisoned Sine. High band 85.5%

Figure 5.3: Isolation Forest behaviour on clean and poisoned datasets. Darker colors are more likely to be inliers. Figures (c) and (d) show the infinite bands of normal regions on the sine datasets. Also note that there is not much difference in (a) vs (b) and in (c) vs (d) meaning the poisoning did not materially affect the algorithm's output.

6 Conclusion

Anomaly detection is an important task, with often security related applications. Research has however mostly focused on developing methods, rather than systematic, theoretical understanding and performance guarantees. Our work paves the way for a so far unexplored avenue of theoretical treatment: axiomatic analysis of anomaly detection methods. We have proposed an initial set of axioms and provided a full analysis of their dependencies and compatibilities. Moreover, we have demonstrated through our experiments how such an axiomatic (or property-based) approach can yield better understanding of algorithmic behaviour: in our case, we uncovered a way in which LOF is vulnerable to strategic data poisoning. We hope our work will inspire follow up studies to develop comprehensive property-based guidelines for choosing anomaly detection methods based on the needs of various applications. Another important direction for follow-up work would be a more thorough understanding of vulnerabilities to data poisoning or other types of adversarial attacks.

Bibliography

[1] Margareta Ackerman, Shai Ben-David, and David Loker. "Characterization of Linkage-based Clustering". In: *COLT 2010 - The 23rd Conference on Learning Theory*. Ed. by Adam Tauman Kalai and Mehryar Mohri. Omnipress, 2010, pp. 270–281.

[2] Margareta Ackerman, Shai Ben-David, and David Loker. "Towards Property-Based Classification of Clustering Paradigms". In: *Advances in Neural Information Processing Systems 23: 24th Annual Conference on Neural Information Processing Systems 2010. NIPS*. Ed. by John D. Lafferty et al. Curran Associates, Inc., 2010, pp. 10–18.

[3] Chris Aldrich and Xiu Liu. "Monitoring of Mineral Processing Operations with Isolation Forests". In: *Minerals* 14.1 (2024). ISSN: 2075-163X. DOI: 10.3390/min14010076. URL: https://www.mdpi.com/2075-163X/14/1/76.

[4] Kasun Amarasinghe, Kevin Kenney, and Milos Manic. "Toward Explainable Deep Neural Network Based Anomaly Detection". In: *2018 11th International Conference on Human System Interaction (HSI)*. 2018, pp. 311–317. DOI: 10.1109/HSI.2018.8430788.

[5] Shai Ben-David. "Clustering - What Both Theoreticians and Practitioners Are Doing Wrong". In: *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18)*. Ed. by Sheila A. McIlraith and Kilian Q. Weinberger. AAAI Press, 2018, pp. 7962–7964.

[6] Shai Ben-David and Margareta Ackerman. "Measures of Clustering Quality: A Working Set of Axioms for Clustering". In: *Advances in Neural Information Processing Systems 21, Proceedings of the Twenty-Second Annual Conference on Neural Information Processing Systems. NIPS*. Ed. by Daphne Koller et al. Curran Associates, Inc., 2008, pp. 121–128.

[7] Christian Borgs et al. “An Axiomatic Approach to Community Detection”. In: *Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science. ITCS*. Ed. by Madhu Sudan. ACM, 2016, pp. 135–146.

[8] Markus M. Breunig et al. “LOF: Identifying Density-Based Local Outliers”. In: *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*. Ed. by Weidong Chen, Jeffrey F. Naughton, and Philip A. Bernstein. ACM, 2000, pp. 93–104.

[9] Mattia Carletti, Matteo Terzi, and Gian Antonio Susto. “Interpretable Anomaly Detection with DIFFI: Depth-based feature importance of Isolation Forest”. In: *Engineering Applications of Artificial Intelligence* 119 (2023), p. 105730. ISSN: 0952-1976. DOI: https://doi.org/10.1016/j.engappai.2022.105730. URL: https://www.sciencedirect.com/science/article/pii/S0952197622007205.

[10] David Chapman and Ajay Jain. *Musk (Version 2)*. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C51608. 1994.

[11] Jayanta Choudhury et al. “Hypersphere for Branching Node for the Family of Isolation Forest Algorithms”. In: *2021 IEEE International Conference on Smart Computing (SMARTCOMP)*. 2021, pp. 418–423. DOI: 10.1109/SMARTCOMP52413.2021.00090.

[12] Emmanouil Christoforou et al. “MRI Condition Monitoring with Explainable AI and Feature Selection”. In: *2022 30th Signal Processing and Communications Applications Conference (SIU)*. 2022, pp. 1–4. DOI: 10.1109/SIU55565.2022.9864924.

[13] Parikshit Gopalan, Vatsal Sharan, and Udi Wieder. “PIDForest: anomaly detection via partial identification”. In: *Proceedings of the 33rd International Conference on Neural Information Processing Systems*. Red Hook, NY, USA: Curran Associates Inc., 2019.

[14] Xiaoyi Gu, Leman Akoglu, and Alessandro Rinaldo. “Statistical Analysis of Nearest Neighbor Methods for Anomaly Detection”. In: *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS*. Ed. by Hanna M. Wallach et al. 2019, pp. 10921–10931.

[15] Lu Han, Yuan Gao, and Xuejian Zheng. “Anomaly detection and identification of power consumption data based on LOF and isolation forest”. In: *International Conference on Mechatronic Engineering and Artificial Intelligence (MEAI 2023)*. Ed. by Yonghe Wei and Fengli Liu. Vol. 13071. In-

ternational Society for Optics and Photonics. SPIE, 2024, p. 1307134. DOI: 10.1117/12.3025688. URL: https://doi.org/10.1117/12.3025688.

[16] S. Hariri, M. Carrasco Kind, and R. J. Brunner. "Extended Isolation Forest". In: *IEEE Transactions on Knowledge and Data Engineering* (2019), pp. 1–1. DOI: 10.1109/TKDE.2019.2947676.

[17] Sahand Hariri, Matias Carrasco Kind, and Robert J. Brunner. "Extended Isolation Forest". In: *IEEE Trans. Knowl. Data Eng.* 33.4 (2021), pp. 1479–1489.

[18] Markelle Kelly, Longjohn Rachel, and Kolby Nottingham. *The UCI Machine Learning Repository*. URL: https://archive.ics.uci.edu.

[19] Jon M. Kleinberg. "An Impossibility Theorem for Clustering". In: *Advances in Neural Information Processing Systems 15, NIPS*. Ed. by Suzanna Becker, Sebastian Thrun, and Klaus Obermayer. MIT Press, 2002, pp. 446–453.

[20] Mineichi Kudo, Jun Toyama, and Masaru Shimbo. *Japanese Vowels*. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5NS47.

[21] Meng-Chieh Lee et al. "Gen2Out: Detecting and Ranking Generalized Anomalies". In: *2021 IEEE International Conference on Big Data (Big Data)*. 2021, pp. 801–811. DOI: 10.1109/BigData52589.2021.9671550.

[22] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. "Isolation Forest". In: *Proceedings of the 8th IEEE International Conference on Data Mining ICDM*. IEEE Computer Society, 2008, pp. 413–422.

[23] Alex Mourer et al. "Automatic Detection of Rare Observations During Production Tests Using Statistical Models". In: *PHM 2020 - Annual Conference of the PHM Society*. Nashville, United States, Nov. 2020. URL: https://hal.science/hal-03130682.

[24] Quoc Phong Nguyen et al. "GEE: A Gradient-based Explainable Variational Autoencoder for Network Anomaly Detection". In: *2019 IEEE Conference on Communications and Network Security (CNS)*. 2019, pp. 91–99. DOI: 10.1109/CNS.2019.8802833.

[25] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[26] Ross Quinlan. *Thyroid Disease*. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5D010. 1987.

[27] Shebuti Rayana. *ODDS Library*. 2016. URL: https://odds.cs.stonybrook.edu.

[28] Lukas Ruff et al. “A Unifying Review of Deep and Shallow Anomaly Detection”. In: *Proc. IEEE* 109.5 (2021), pp. 756–795.

[29] Alain Ryser et al. “Anomaly Detection in Echocardiograms with Dynamic Variational Trajectory Models”. In: *Proceedings of the 7th Machine Learning for Healthcare Conference*. Ed. by Zachary Lipton et al. Vol. 182. Proceedings of Machine Learning Research. PMLR, 2022, pp. 425–458. URL: https://proceedings.mlr.press/v182/ryser22a.html.

[30] Shubhranshu Shekhar, Neil Shah, and Leman Akoglu. “FairOD: Fairness-aware Outlier Detection”. In: *AIES ’21: AAAI/ACM Conference on AI, Ethics, and Society*. Ed. by Marion Fourcade et al. ACM, 2021, pp. 210–220.

[31] Md Amran Siddiqui et al. “Finite Sample Complexity of Rare Pattern Anomaly Detection”. In: *Proceedings of the Thirty-Second Conference on Uncertainty in Artificial Intelligence. UAI*. Ed. by Alexander Ihler and Dominik Janzing. AUAI Press, 2016.

[32] Marek Sikora and Lukasz Wrobel. *seismic-bumps*. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5W902. 2013.

[33] Ashwin Srinivasan. *Statlog (Landsat Satellite)*. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C55887. 1993.

[34] Salvatore Stolfo et al. *KDD Cup 1999 Data*. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C51C7N. 1999.

[35] Jun Wang, Junxing Cao, and Zhege Liu. “Unsupervised machine learning-based multi-attributes fusion dim spot subtle sandstone reservoirs identification utilizing isolation forest”. In: *Geoenergy Science and Engineering* 234 (2024), p. 212626. ISSN: 2949-8910. DOI: https://doi.org/10.1016/j.geoen.2023.212626. URL: https://www.sciencedirect.com/science/article/pii/S2949891023012137.

[36] Chester Wyke and Ruth Urner. *An Axiomatic Perspective on Anomaly Detection*. Accepted at the 27TH European Conference on Artificial Intelligence (ECAI) 2024.

[37] Michael Alexander Zenkl-Galaz, Octavio Loyola-González, and Miguel Angel Medina-Pérez. “IOGOD: An interpretable outlier generation-based outlier detector for categorical databases”. In: *Expert Systems with Applications* 195 (2022), p. 116570. ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2022.116570. URL: https://www.sciencedirect.com/science/article/pii/S0957417422000689.

[38] Yue Zhao, Ryan Rossi, and Leman Akoglu. "Automatic Unsupervised Outlier Model Selection". In: *Advances in Neural Information Processing Systems*. Ed. by M. Ranzato et al. Vol. 34. Curran Associates, Inc., 2021, pp. 4489–4502. URL: https://proceedings.neurips.cc/paper_files/paper/2021/file/23c894276a2c5a16470e6a31f4618d73-Paper.pdf.

A Appendix

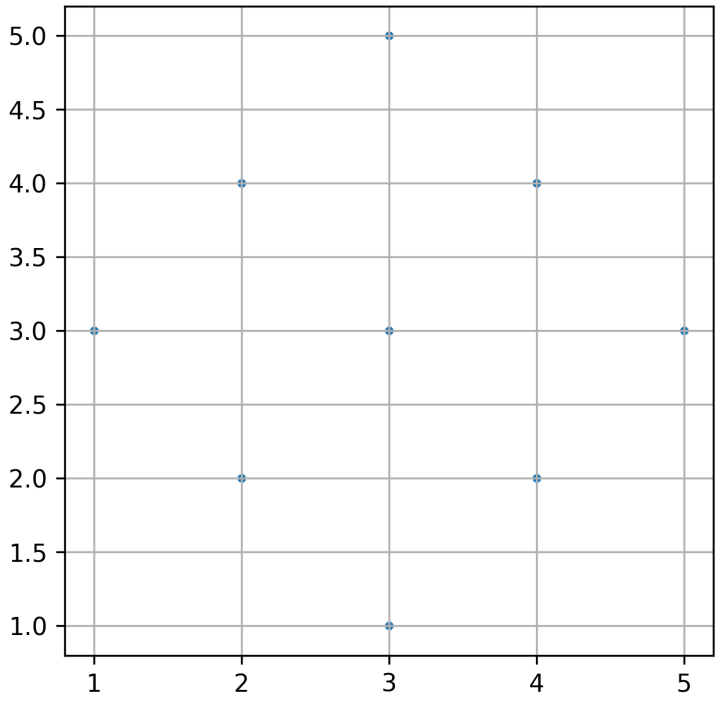


Figure A.1: diamond

A.1 Synthetic Dataset Images

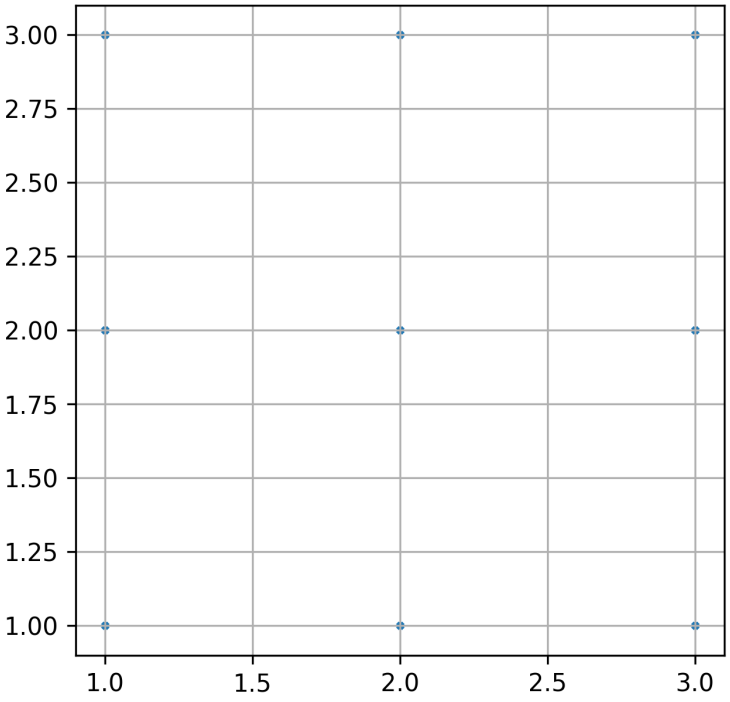


Figure A.2: square

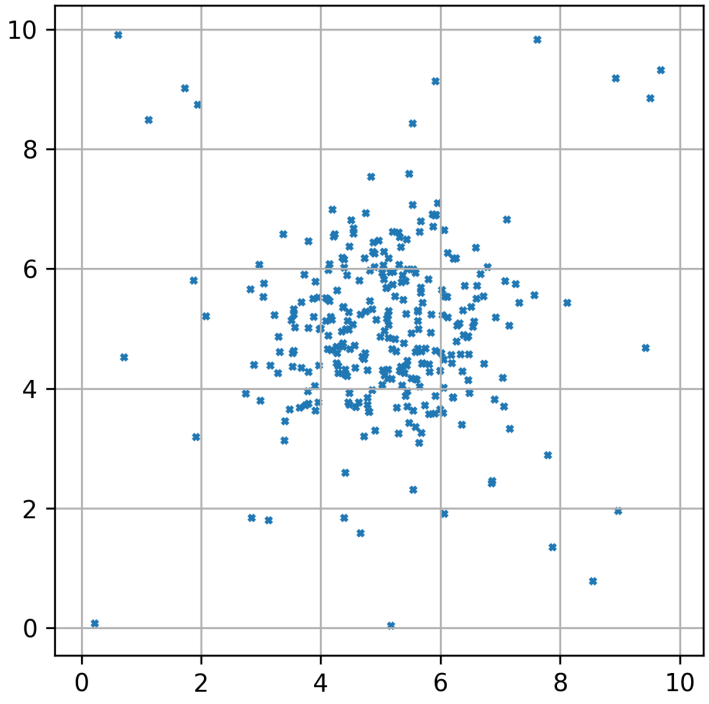


Figure A.3: normal-uniform

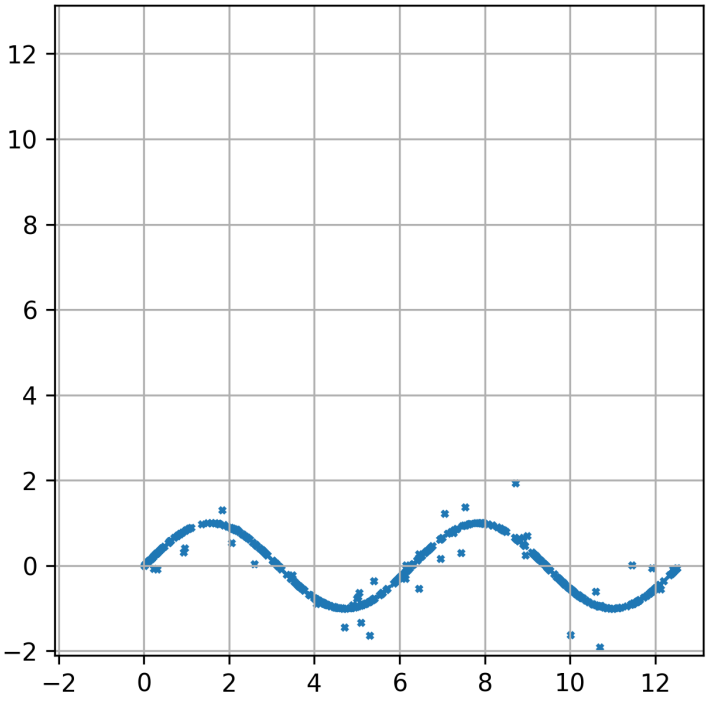


Figure A.4: sine-uniform

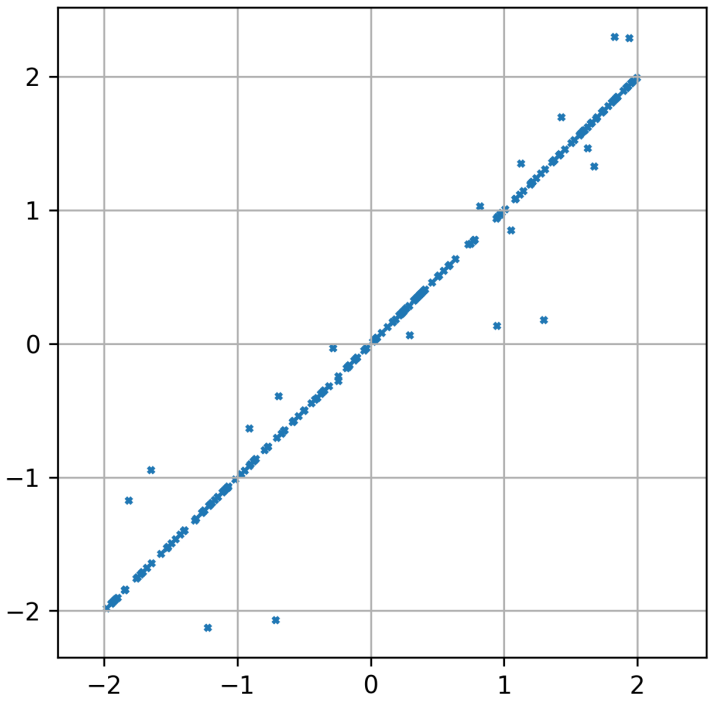


Figure A.5: polynomial

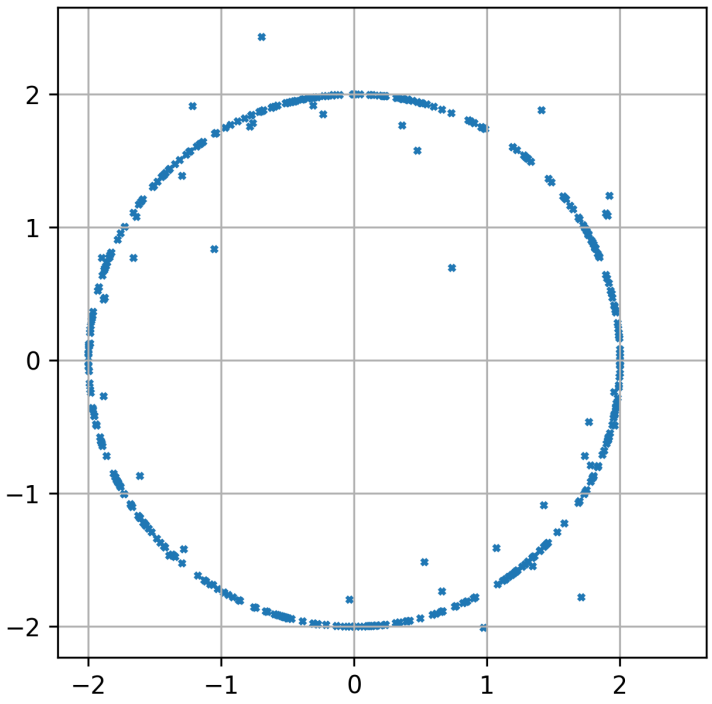


Figure A.6: circle

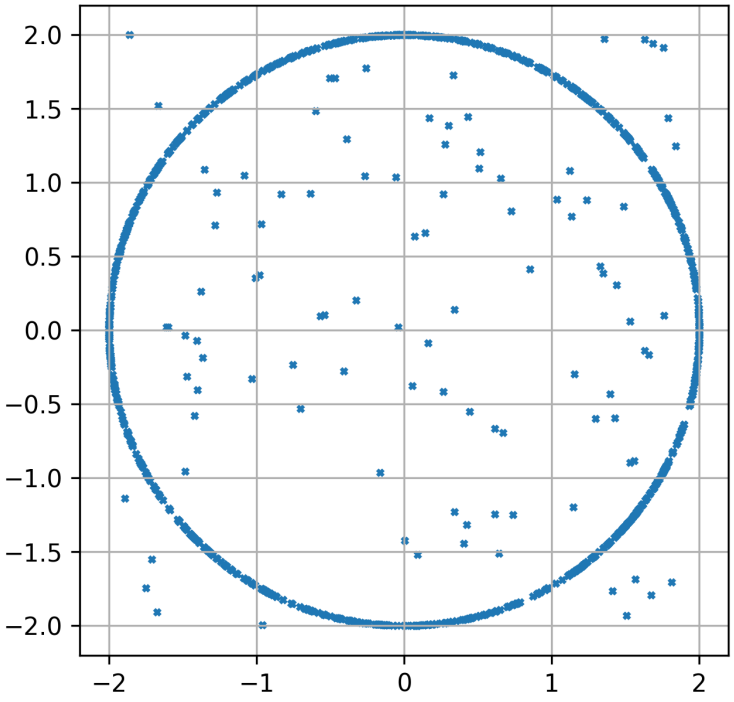


Figure A.7: sphere

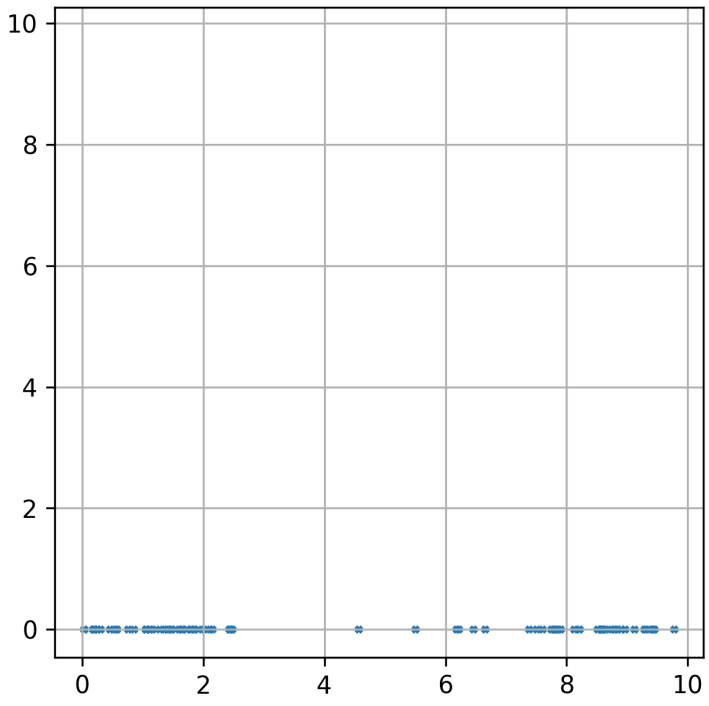


Figure A.8: single-dimension

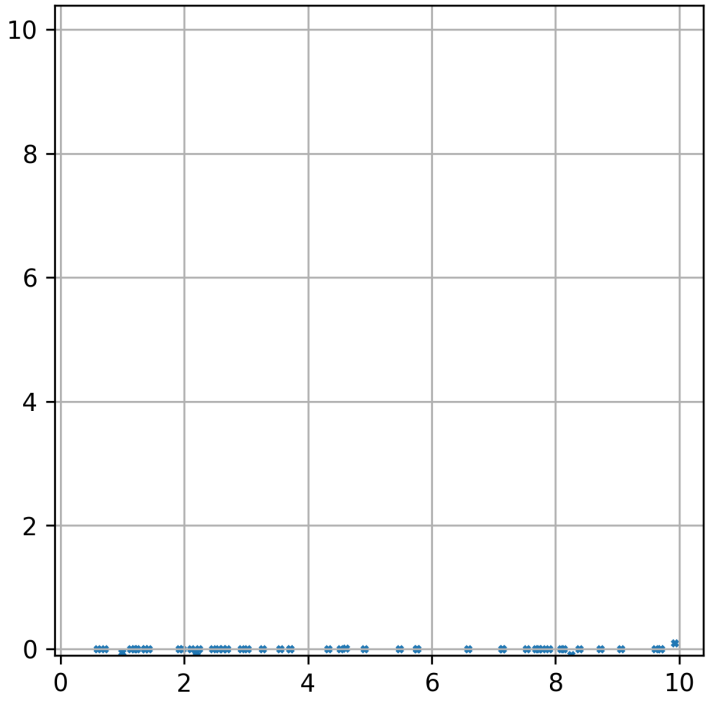


Figure A.9: all-but-one