Leave-one-out kernel density estimates for outlier detection

Sevvandi Kandanaarachchi *
RMIT University
and
Rob J Hyndman
Monash University

August 4, 2021

Abstract

This paper introduces *lookout*, a new approach to detect outliers using leave-one-out kernel density estimates and extreme value theory. Outlier detection methods that use kernel density estimates generally employ a user defined parameter to determine the bandwidth. Lookout uses persistent homology to construct a bandwidth suitable for outlier detection without any user input. We demonstrate the effectiveness of lookout on an extensive data repository by comparing its performance with other outlier detection methods based on extreme value theory. Furthermore, we introduce *outlier persistence*, a useful concept that explores the birth and the cessation of outliers with changing bandwidth and significance levels. The R package `lookout` implements this algorithm.

Keywords: anomaly detection, topological data analysis, persistent homology, extreme value theory, peak over thresholds, generalized Pareto distribution

*sevvandi.kandanaarachchi@rmit.edu.au

1 Introduction

Outliers are data points that are unusual compared to the rest. They are also known as anomalies or novelties. Outlier-detection methods are used in diverse applications ranging from detecting security breaches in the Internet of Things networks to identifying extreme weather events. Whatever the application, it is important to develop robust techniques to detect outliers, which minimize costly false positives and dangerous false negatives.

This diverse literature can be divided into approaches that use probability densities to define outliers, and those that use distances to define outliers. Outlier detection methods that use probability densities treat outliers as observations that are very unlikely given the other observations. Outlier detection methods that use distances treat outliers as observations that lie far from other observations.

We take a probability density approach and propose a new outlier detection algorithm that we call *lookout*, which uses leave-one-out kernel density estimates to identify the most unlikely observations. We address the challenge of bandwidth selection by using persistent homology — a concept in topological data analysis — and use extreme value theory (EVT) to identify outliers based on their leave-one-out density estimates.

The main challenge in using kernel density estimates for outlier detection is the selection of the bandwidth. Schubert et al. (2014) employ kernel density estimates to detect outliers using k-nearest neighbor distances where k is a user-specified parameter, which determines bandwidth. Qin et al. (2019) employ kernel density estimates to detect outliers in streaming data, using a user-specified radius parameter which is equivalent to the bandwidth. Tang & He (2017) use reverse and shared k-nearest neighbors to compute kernel density estimates and identify outliers. They also have a user-defined parameter that denotes the reverse k-nearest neighbors. The oddstream algorithm (Talagala et al. 2020) computes kernel density estimates on a 2-dimensional projection defined by the first two principal components, and so bandwidths need to be selected. We avoid subjective user-choice, and the inappropriate use of bandwidths optimized for some other purpose, by proposing the use of persistent homology as a new tool for bandwidth selection.

Extreme Value Theory has been gaining popularity in outlier detection because of its rich, theoretical foundations. Burridge & Taylor (2006) used EVT to detect outliers in time series data. Clifton et al. (2014) used Generalized Pareto Distributions to model the tails in high-dimensional data and detect outliers. Other recent advances in outlier detection that use EVT include the stray (Talagala et al. 2021), oddstream (Talagala et al. 2020) and HDoutliers (Wilkinson 2017) algorithms. Of these three methods, stray is an enhancement of HDoutliers and both use distances to detect outliers, while oddstream uses kernel density estimates to detect outliers in time series data. Our approach is closest to oddstream in that we also apply EVT to functions of kernel density estimates. However, we use a different functional, and we avoid the need for an outlier-free training set.

A brief introduction to persistent homology and EVT is given in Section 2. In Section 3 we introduce the algorithm *lookout* and the concept of outlier persistence, which explores the birth and death of outliers with changing bandwidth. We show examples illustrating the usefulness of outlier persistence and conduct experiments using synthetic data to evaluate the performance of lookout in Section 4. Using an extensive data repository of real datasets, we compare the performance of lookout to HDoutliers, stray, KDEOS (Schubert et al. 2014) and RDOS (Tang & He 2017) in Section 5.

We have produced an R package `lookout` (Kandanaarachchi & Hyndman 2021) containing this algorithm. In addition, all examples in this paper are available in the supplementary material at `https://github.com/sevvandi/supplementary_material/tree/master/lookout`.

We have used the R packages `TDAstats` (Wadhwa et al. 2018) and `ggtda` (Brunson et al. 2020) for all TDA and persistent homology computations and related graphs. We have used the R package `evd` (Stephenson 2002) to fit the Generalized Pareto Distribution.

2 Mathematical background

In this section we provide some brief background on three topics that we will use in our proposed `lookout` algorithm.

1. topological data analysis and persistent homology;
2. extreme value theory and the peaks-over-threshold approach; and
3. kernel density estimation.

2.1 Topological data analysis and persistent homology

Topological data analysis is the study of data using topological constructs, inferring high dimensional structure from low dimensional representations such as points and assembling discrete points to construct global structures (Ghrist 2008). Persistent homology is a method in algebraic topology that computes topological features of a space that persist across multiple scales or spatial resolutions. These features include connected components, topological circles and trapped volumes. Features that persist for a wider range of spatial resolutions represent robust, intrinsic features of the data while features that sporadically change are perturbations resulting from noise. Persistent homology has been used in a wide variety of applications including biology (Topaz et al. 2015), computer graphics (Carlsson et al. 2008) and engineering (Perea & Harer 2015). In this section we give a brief overview of persistent homology. Readers are referred to Ghrist (2008) and Carlsson (2009) for further details and Wasserman (2018) for a statistical viewpoint on the subject.

Simplicial complex

Consider a data cloud representing a collection of points. This set of points is used to construct a graph where the points are considered vertices and the edges are determined by the distance between the points. Given a proximity parameter ε, two vertices are connected by an edge if the distance between these two points is less than or equal to ε. Starting from this graph, a simplicial complex — a space built from simple pieces — is constructed. A simplicial complex is a finite set of k-simplices, where k denotes the dimension; for example, a point is a 0-simplex, an edge a 1-simplex, a triangle a 2-simplex, and a tetrahedron a 3-simplex. Suppose S denotes a simplicial complex that includes a k-simplex γ. Then all non-empty subsets of $\beta \subset \gamma$ are also included in S. For example, if S contains a triangle pqr, then the edges pq, qr and rs, and the vertices p, q and r, are also in S.

The *Vietoris-Rips* complex and the *Cech* complex are two types of k-simplicial complexes. We will construct a Vietoris-Rips complex from the data cloud as it is more computationally efficient than the Cech complex (Ghrist 2008). Given a set of points and a proximity parameter $\varepsilon > 0$, $k+1$ points within a distance of ε to each other form a k-simplex. For example, consider five points p, q, r, s and t and suppose the distance between any two points except t

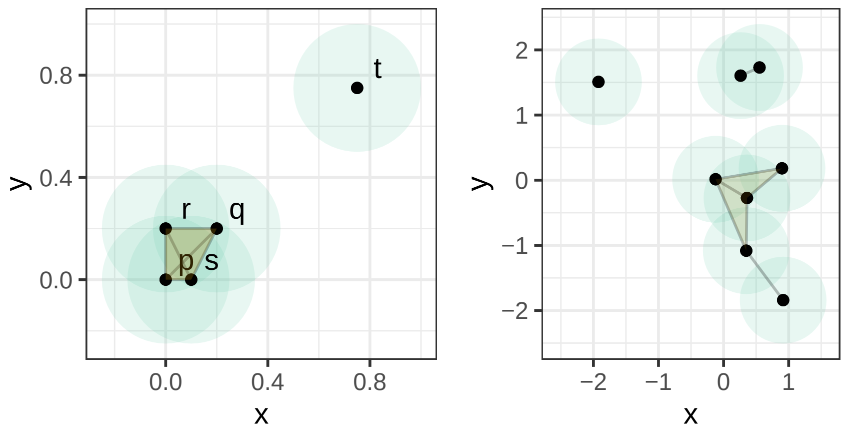

Figure 1: Left: points p, q, r, s and t with a proximity parameter $\varepsilon = 0.5$ and the resulting Rips complex consisting of the tetrahedron $pqrs$, triangles pqr, qrs, rsp, pqs, edges pq, qr, rs, sp, qs, pr and vertices p, q, r, s and t. Right: eight points and the resulting Rips complex with $\varepsilon = 4/3$.

is less than ε. Then we can construct the edges pq, pr, ps, qr, qs and rs. From the edges pq, qr and rp we can construct the triangle pqr, from pq, qs and sp the triangle pqs and so on, because the distance between any two points p, q, r and s is bounded by ε. By constructing the four triangles pqr, qrs, rsp and spq we can construct the tetrahedron $pqrs$. The vertex t is not connected to this 3-simplex because the distance between t and the other vertices is greater than ε. The simplicial complex resulting from these five points consists of the tetrahedron $pqrs$ and all the subset k-simplices and the vertex t. Figure 1 shows this simplicial complex on the left and another example on the right.

Persistent homology

Given a point cloud of data, the resulting Rips complex depends on the value of the proximity parameter ε. As we increase ε, topological features such as connected components and holes appear and disappear. This is the focus of persistent homology. For example, in Figure 2, we start with a large number of connected components (top-left) and as ε increases to 0.8 the number of connected components merge and decrease to 1 (bottom-left). Around this value of ε, a hole appears and as ε increases to 1.5, it disappears (bottom-right). The appearances and disappearances of these topological features are referred to as births and deaths and are illustrated using a *barcode* or a *persistence diagram*.

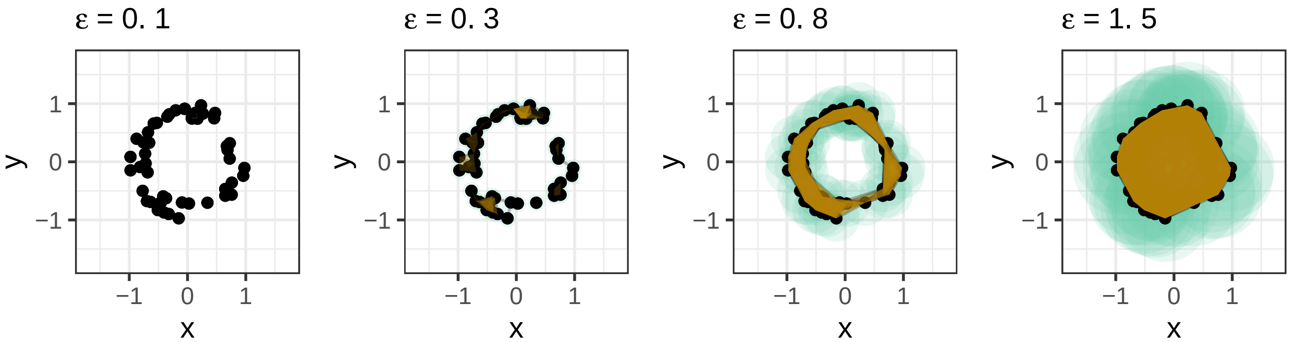

Figure 2: Rips complexes resulting from different ε values.

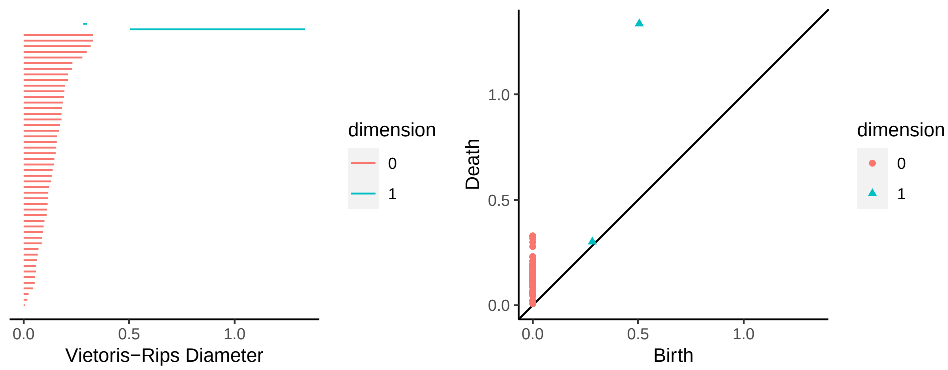

Figure 3: Left: the barcode of the point cloud in Figure 2. Right: the persistence diagram.

Figure 3 shows the barcode and the persistence diagram of the point cloud shown in Figure 2. The barcode comprises a set of horizontal line segments, each denoting a feature that starts at its birth diameter and ends at its death diameter. These line segments are grouped by their dimension. The orange lines in Figure 3 denote the 0-dimensional holes (connected components) and blue lines denote 1-dimensional holes. The longer blue line which is born at 0.51 and dies at 1.33, corresponds to the hole at the center of the point cloud in Figure 2. Features that continue for a large range of ε represent structural properties of the data that are of interest to us. The same information is shown in the persistence diagram, where the birth and the death of each feature is denoted by a point. Points away from the diagonal inform about the structure of the data while points closer to the diagonal are perturbations related to noise. In this plot, the triangle near the top represents the same feature as the long blue line in the left plot.

These considerations lead to a natural question: which ε is most representative of the structure of the data cloud? We will return to this question later.

2.2 Extreme value theory

Extreme Value Theory is used to model rare, extremal events. It is used in many industries including hydrology (to study 100-year floods), finance (to explore extreme risks) and insurance (to mitigate against losses due to disasters) (Reiss & M.Thomas 2001). EVT has also been used in outlier detection (Wilkinson 2017, Talagala et al. 2020). In this section we will give a brief introduction to EVT using the notation in Coles (2001).

Consider n independent and identically distributed random variables $X_1, \ldots, X_n$ with a distribution function $F(x) = P\{X \leq x\}$. Then the maximum of these n random variables is $M_n = \max\{X_1, \ldots, X_n\}$. If F is known, the distribution of M_n is given by (Coles 2001, p45) $P\{M_n \leq z\} = (F(z))^n$. However, F is usually not known in practice. This gap is filled by Extreme Value Theory, which studies approximate families of models for F^n so that extremes can be modeled and uncertainty quantified. The Fisher-Tippet-Gnedenko Theorem states that under certain conditions, a scaled maximum $\frac{M_n - a_n}{b_n}$ can have certain limit distributions.

Theorem 2.1 (Fisher-Tippett-Gnedenko)**.** *If there exist sequences $\{a_n\}$ and $\{b_n\}$ such that*

$$P\left\{\frac{(M_n - a_n)}{b_n} \leq z\right\} \to G(z) \quad as \quad n \to \infty,$$

where G is a non-degenerate distribution function, then G belongs to one of the following families:

$$\begin{aligned}
&\textit{Gumbel}: && G(z) = \exp\left(-\exp\left[-\left(\frac{z-b}{a}\right)\right]\right), \quad -\infty < z < \infty, \\
&\textit{Fréchet}: && G(z) = \begin{cases} 0, & z \leq b, \\ \exp\left(-\left(\frac{z-b}{a}\right)^{-\alpha}\right), & z > b, \end{cases} \\
&\textit{Weibull}: && G(z) = \begin{cases} \exp\left(-\left(-\left[\frac{z-b}{a}\right]\right)^{\alpha}\right), & z < b, \\ 1, & z \geq b, \end{cases}
\end{aligned}$$

for parameters a, b and α where $a, \alpha > 0$.

These three families of distributions can be further combined into a single family by using the following distribution function known as the Generalized Extreme Value (GEV) distribution,

$$G(z) = \exp\left\{-\left[1 + \xi\left(\frac{z-\mu}{\sigma}\right)\right]^{-1/\xi}\right\},$$

where the domain of the function is $\{z : 1 + \xi(z-\mu)/\sigma > 0\}$. The location parameter is $\mu \in \mathbb{R}$, $\sigma > 0$ is the scale parameter, while $\xi \in \mathbb{R}$ is the shape parameter. When $\xi = 0$ we obtain a Gumbel distribution with exponentially decaying tails. When $\xi < 0$ we get a Weibull distribution with a finite upper end and when $\xi > 0$ we get a Fréchet family of distributions with polynomially decaying tails.

The Generalized Pareto Distribution and the POT approach

The Peaks Over Threshold (POT) approach regards extremes as observations greater than a threshold u. We can write the conditional probability of extreme events as

$$P\{X > u + y \mid X > u\} = \frac{1 - F(u+y)}{1 - F(u)}, \quad y > 0,$$

giving us

$$P\{X \leq u + y \mid X > u\} = \frac{F(u+y) - F(u)}{1 - F(u)}, \quad y > 0.$$

The distribution function

$$F_u(y) = P\{X \leq u + y \mid X > u\},$$

describes the *exceedances* above the threshold u. If F is known we could compute this probability. However, as F is not known in practice we use approximations based on the Generalized Pareto Distribution (Pickands 1975).

Theorem 2.2 (Pickands)**.** *Let $X_1, X_2, \ldots, X_n$ be a sequence of independent random variables with a common distribution function F, and let $M_n = \max\{X_1, \ldots, X_n\}$. Suppose F satisfies Theorem 2.1 so that for large n, $P\{M_n \leq z\} \approx G(z)$, where*

$$G(z) = \exp\left\{ -\left[1 + \xi\Big(\frac{z-\mu}{\sigma}\Big)\right]^{-1/\xi} \right\},$$

for some $\mu, \xi \in \mathbb{R}$ and $\sigma > 0$. Then for large enough u, the distribution function of $(X-u)$ conditional on $X > u$, is approximately

$$H(y) = 1 - \Big(1 + \frac{\xi y}{\sigma_u}\Big)^{-1/\xi}, \tag{1}$$

where the domain of H is $\{y : y > 0$ and $(1 + \xi y)/\sigma_u > 0\}$, and $\sigma_u = \sigma + \xi(u - \mu)$.

The family of distributions defined by equation (1) is called the **Generalized Pareto Distribution** (GPD). We note that the GPD parameters are determined from the associated GEV parameters. In particular, the shape parameter ξ is the same in both distributions.

For a chosen threshold u, the parameters of the GPD can be estimated by standard maximum likelihood techniques. As an analytical solution that maximizes the likelihood does not exist, numerical techniques are used to arrive at an approximate solution. We use the R package `evd` to fit a GPD using the POT approach.

2.3 Kernel density estimation

For a sample $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n \in \mathbb{R}^p$, the kernel density estimate is given by

$$\hat{f}(\boldsymbol{x}; \boldsymbol{H}) = \frac{1}{n} \sum_{i=1}^{n} K_{\boldsymbol{H}} \left(\boldsymbol{x} - \boldsymbol{x}_i \right), \tag{2}$$

where $\boldsymbol{H}$ denotes a $p \times p$ positive definite bandwidth matrix, $K_{\boldsymbol{H}}(\boldsymbol{z}) = \boldsymbol{H}^{-1/2} K(\boldsymbol{H}^{-1/2}\boldsymbol{z})$, and K is a kernel function. For consistency, we scale the kernels so that $\int \|\boldsymbol{z}^2\| K(\boldsymbol{z}) d\boldsymbol{z} = 1$. The multivariate Gaussian kernel is given by

$$K(\boldsymbol{x}) = \frac{1}{(2\pi)^{p/2}} \exp(\|\boldsymbol{x}\|^2/2),$$

and the multivariate scaled Epanechnikov kernel by

$$K(\boldsymbol{x}) = \frac{p+2}{c_p}(1 - \|\boldsymbol{x}\|^2/5)_+, \tag{3}$$

where c_p is the volume of the unit sphere in $\mathbb{R}^p$ and $u_+ = \max(0, u)$.

We will use the leave-one-out kernel density estimator given by

$$\hat{f}_{-j}(\boldsymbol{x}; \boldsymbol{H}) = \frac{1}{n-1} \sum_{i \neq j} K_{\boldsymbol{H}}(\boldsymbol{x} - \boldsymbol{x}_i),$$

which can be simplified to

$$\hat{f}_{-j}(\boldsymbol{x}_j; \boldsymbol{H}) = \left[n\hat{f}(\boldsymbol{x}_j; \boldsymbol{H}) - \boldsymbol{H}^{-1/2} K(\boldsymbol{0}) \right] / (n-1)$$

when evaluated at the observation omitted.

A major challenge in using kernel density estimates for outlier detection is selecting the appropriate bandwidth. There is a large body of literature on kernel density estimation and bandwidth selection (Scott 2015, Wang & Scott 2019) that focuses on computing density estimates that represent the data as accurately as possible, where measures of accuracy have certain asymptotic properties. However, our goal is somewhat different as we are interested in finding outliers in the data, rather than finding a good representation for the rest of the data. Often the usual bandwidth selection methods result in bandwidths that are too small and can cause the kernel density estimates of the boundary and near-boundary points to be confused with outliers. In high dimensions this problem is exacerbated due to the sparsity of the data. Thus, we need a bandwidth that assists outlier detection. A too small bandwidth causes everything to be outliers, while too large a bandwidth will lead to outliers being hidden.

3 Methodology

3.1 Bandwidth selection using TDA

To select a bandwidth for a kernel density estimate designed for outlier detection, we propose to use the barcode discussed in Section 2.1. First we construct the barcode of the data cloud for dimension zero using the Vietoris-Rips diameter. From the barcode we obtain the sequence of death diameters $\{d_i\}_{i=1}^N$ for the connected components. By construction this is an increasing sequence as seen in Figure 3.

Consider the example shown in Figure 4. At the top left, the data are shown with most points lying on an annulus, and a few points near the centre. The barcodes for dimension 0 are shown at the top right, with lengths equal to the death diameters $\{d_i\}_{i=1}^N$. The bottom left panel shows violin plots comparing the death diameters (denoted by TDA) with k nearest neighbor distances for $k \in \{1, 5, 10\}$. The TDA Rips diameters fall within the range of the combined KNN distances. Consequently, we can use TDA distances without having to select the parameter k for KNN distances.

The plot on the bottom right in Figure 4 shows the largest 20 Rips diameters (out of the 999 diameters shown in the top right plot). A vertical dashed line is drawn at diameter 0.468, the second largest Rips diameter. As the diameter increases from 0.468 till 1.041, which is the maximum diameter, the number of connected components stay the same. For this point cloud, $(0.468, 1.041)$ is the largest diameter range for which the number of components

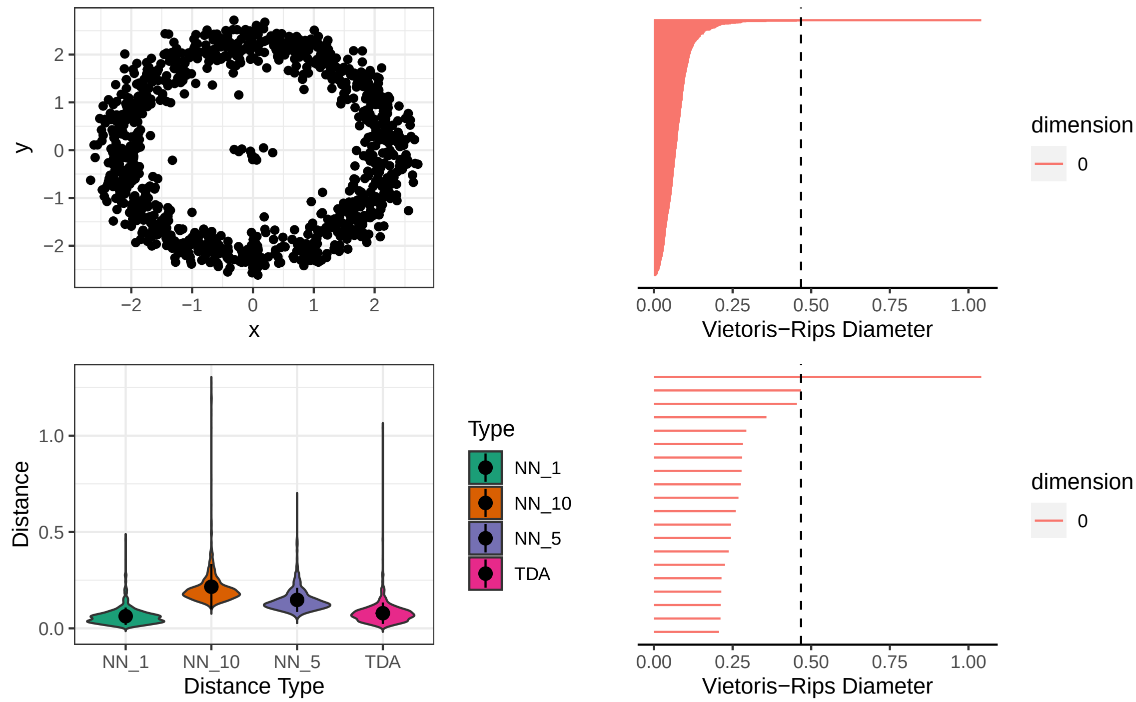

Figure 4: Top left: A scatterplot of 1000 observations with most points falling on an annulus and some points near the center. Its TDA barcode on the top right and the violin plot of TDA death diameters and KNN distances at the bottom.

stay the same. Thus, it signifies a global structural property of the point cloud. We want to take this structure into account when selecting the bandwidth. Therefore, we choose a diameter that gives rise to persisting features, which in our case are connected components. We consider the Rips diameter intervals (d_i, d_{i+1}) for all i, and find the largest interval by computing successive differences $\Delta d_i = d_{i+1} - d_i$, for $i \in \{1, \cdots, N-1\}$. We choose the Rips diameter d_i corresponding to the maximum Δd_i:

$$d_* = d_{i_*} \qquad \text{where} \quad i_* = \arg\max\{\Delta d_i\}_{i=1}^{N-1}. \tag{4}$$

Then the bandwidth matrix is given by

$$\boldsymbol{H} = d_*^{2/p} \boldsymbol{I}, \tag{5}$$

so that

$$\|\boldsymbol{H}^{-1/2}(\boldsymbol{x} - \boldsymbol{x}_i)\|^2 = \frac{1}{d_*^2}\|\boldsymbol{x} - \boldsymbol{x}_i\|^2. \tag{6}$$

This ensures that points within a distance of d_* contribute to the kernel density estimate of $\boldsymbol{x}$, resulting in the following leave-one-out kernel density estimate

$$\hat{f}_{-j}(\boldsymbol{x}_j; \boldsymbol{H}) = \left[n\hat{f}(\boldsymbol{x}_j; \boldsymbol{H}) - d_*^{-1}K(\boldsymbol{0})\right]/(n-1). \tag{7}$$

3.2 Algorithm *lookout*

Now we have the building blocks necessary to describe the algorithm *lookout*. Consider an $N \times p$ data matrix $\boldsymbol{X}$ with N observations in $\mathbb{R}^p$. It is customary in outlier detection to scale the data so that all variables contribute equally. We normalize the data using Min-Max normalization, which scales each attribute to $[0, 1]$ and has been shown to be effective compared to other normalization techniques (Kandanaarachchi et al. 2020). To accommodate datasets that do not need to be normalized, we make normalization optional.

We compute the kernel density estimates defined by (2) and (7), with the bandwidth matrix (5) and the scaled Epanechnikov kernel (3). Denote the kernel density estimate of $\boldsymbol{x}_i$ by y_i and the leave-one-out kde of $\boldsymbol{x}_i$ (by leaving out $\boldsymbol{x}_i$) by y_{-i}. Then we fit a Generalized Pareto Distribution to $-\log(y_i)$ using the POT approach discussed in Section 2.2. We use the 90^{th} percentile as the threshold for the POT approach as recommended by Bommier (2014). Using the fitted GPD parameters, μ, σ and ξ, we declare points with $P\left(-\log(y_{-i})|\mu, \sigma, \xi\right) < \alpha$ to be outliers. We summarize these steps in Algorithm 1.

The output probability of lookout is the GPD probability of the points, so that low probabilities indicate likely outliers and high probabilities indicate normal points. Note that the scaling factor of the kernel $K(\boldsymbol{x})$ does not affect the GPD parameters as it is just an offset after taking logs of y_i or y_{-i}.

The algorithm lookout has only two inputs: α (the threshold for outlier detection) and *unitize* (allowing optional normalization). We set $\alpha = 0.05$ and *unitize* = `TRUE` as default parameter values. We use the Epanechnikov kernel in lookout due to ease of computation. However, any kernel can be incorporated as long as the variances are matched.

3.3 Outlier persistence

Lookout identifies outliers after selecting an appropriate bandwidth using TDA. The points identified as outliers may depend on this bandwidth. We explore the relationship between

Algorithm 1: *lookout.*

input : The data matrix $\boldsymbol{X}$, parameters α and *unitize.*

output : The outliers, the GPD probabilities of all points, GPD parameters and bandwidth

1. If *unitize* = *TRUE*, then normalize the data so that each column is scaled to $[0, 1]$.
2. Construct the persistence homology barcode of the data.
3. Find d_* as in equation (4).
4. Using $\boldsymbol{H} = (d_*)^{2/p}\boldsymbol{I}$, compute kernel density estimates (2) and leave-one-out kernel density estimates (7) using the scaled Epanechnikov kernel (3).
5. Denote the kde of $\boldsymbol{x}_i$ by y_i and leave-one-out kde by y_{-i}.
6. Using the POT approach, fit a GPD to $\{-\log(y_i)\}_{i=1}^N$ and estimate μ, σ and ξ.
7. Using the GPD estimates $\hat{\mu}$, $\hat{\sigma}$ and $\hat{\xi}$, find the probability of the leave-one-out kde values $\{-\log(y_{-i})\}_{i=1}^N$, i.e., $P\left(-\log(y_{-i})|\mu, \sigma, \xi\right)$ for all i.
8. If $P\left(-\log(y_{-i})|\mu, \sigma, \xi\right) < \alpha$, then declare $\boldsymbol{x}_i$ as an outlier.

outliers and bandwidth by varying bandwidth values in lookout. Similar work is discussed by Minnotte & Scott (1993), who introduce the Mode Tree, which tracks the modes of the kernel density estimates with changing bandwidth values. Another related idea is the SiZer map (Chaudhuri & Marron 1999), a graphical device that studies features of curves for varying bandwidth values.

If a point is identified as an outlier by the lookout algorithm for a range of bandwidth values, then it increases the validity of that point as an outlier. Consider an annulus with some points in the middle as shown in the left plot of Figure 5. The plot on the right, which is similar to a barcode, shows the outliers identified by lookout for different bandwidth values. Each horizontal line segment shows the range of Rips diameter values that has identified each point as an outlier. With some abuse of notation, we label the x axis “bandwidth”, even though it actually represents the Rips diameter $d_* = h^{p/2}$, where the bandwidth matrix $\boldsymbol{H} = h\boldsymbol{I}$. This is motivated from equation (6), as we only consider points $\boldsymbol{x}_i$ within a distance d_* of every point $\boldsymbol{x}$ when computing kernel density estimates using the Epanechnikov kernel.

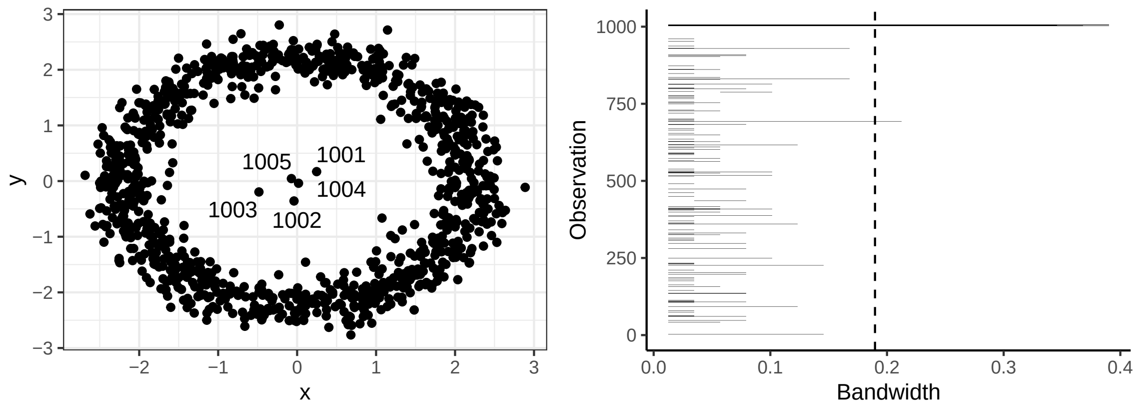

Figure 5: Outliers at the center of the annulus in the left plot showing the outlier labels 1001–1005. The outlier persistence diagram on the right with the y-axis denoting the labels. The dashed line shows the lookout bandwidth d_*.

In this plot, the y-axis corresponds to the point index. We call this plot *the outlier persistence diagram* signifying the link to topological data analysis.

In the example in Figure 5, we see that the points 1001–1005 are identified as outliers in the outlier persistence diagram for a large range of bandwidth values. The Rips diameter d_* selected by lookout is shown by a vertical dashed line. Many points are identified as outliers for small bandwidth values but do not continue to be outliers for long; these outliers are born at small bandwidth values and die out after a relatively short increase in bandwidth. Some points are never identified as outliers, even at small bandwidths.

The outlier persistence diagram is a tool to observe the persistence of outliers with changing bandwidth values. We vary the bandwidth values while keeping the GPD parameters fixed at the values obtained using d_* as in equation (4). The death diameter sequence d_i is used to construct the set of bandwidth values used in this plot. We use ℓ bandwidth values starting from the β^{th} percentile of sequence $\{d_i\}$ ending at $\sqrt{5} \times \max_i d_i$. The parameters ℓ and β are user-defined with default values $\ell = 20$ and $\beta = 90$. Increasing ℓ gives better granularity but increases the computational burden. As the death diameters are tightly packed, the default value of 90^{th} percentile gives a small enough starting bandwidth and $\sqrt{5} \max_i d_i$ gives a large ending bandwidth. We summarize these steps in Algorithm 2.

Next, we extend this notion of persistence to include significance levels $\alpha \in \{0.01, 0.02, \ldots, 0.1\}$.

Algorithm 2: *outlier persistence for fixed α.*

input : The data matrix $\boldsymbol{X}$, parameters α, *unitize* and bandwidth range parameters ℓ and β

output : An $N \times \ell$ binary matrix $\boldsymbol{Y}$ where N denotes the number of observations and ℓ denotes the number of bandwidth values with $y_{ik} = 1$ if the i^{th} observation is identified as an outlier for bandwidth index k.

Initialize matrix $\boldsymbol{Y}$ to zero.

Run Algorithm 1 to determine the death diameter sequence $\{d_i\}$ and the GPD parameters μ_0, σ_0 and ξ_0.

Construct an equidistant bandwidth sequence of length ℓ starting from the β^{th} percentile of $\{d_i\}$ to $\sqrt{5}\max_i d_i$. Call the bandwidth sequence $\{b_k\}_{k=1}^{\ell}$.

for *k from 1 to ℓ* **do**

- Using $h = (b_k)^{2/p}$ and $\boldsymbol{H} = h\boldsymbol{I}$ compute kernel density estimates and leave-one-out kernel density estimates using the scaled Epanechnikov kernel.
- Denote the kde of $\boldsymbol{x}_i$ by y_i and leave-one-out kde by y_{-i}.
- Using the GPD parameters μ_0, σ_0 and ξ_0 find the GPD probability of the leave-one-out kde values $\{-\log(y_{-i})\}_{i=1}^{N}$, i.e., $P\left(-\log(y_{-i})|\mu_0, \sigma_0, \xi_0\right)$ for all i.
- If $P\left(-\log(y_{-i})|\mu_0, \sigma_0, \xi_0\right) < \alpha$, then declare $\boldsymbol{x}_i$ as an outlier and let $y_{ik} = 1$.

We define the *strength* of an outlier based on its significance level. Let $\boldsymbol{x}_j$ be an outlier identified at significance level α, where α is the smallest significance level for which $\boldsymbol{x}_j$ is identified as an outlier. Then

$$\text{strength}(\boldsymbol{x}_j) = \frac{(0.11 - \alpha)_+}{0.01} \tag{8}$$

Thus, if a point is identified as an outlier with a significance level $\alpha = 0.01$, then it has strength 10, and an outlier with $\alpha = 0.1$ has strength 1. To compute persistence over significance levels, the only modification that needs to be done to Algorithm 2 is to fix $\alpha = 0.1$ and to record the minimum significance level if $P\left(-\log(y_{-i})|\mu_0, \sigma_0, \xi_0\right) < 0.1$. Then we can use equation (8) to compute its strength.

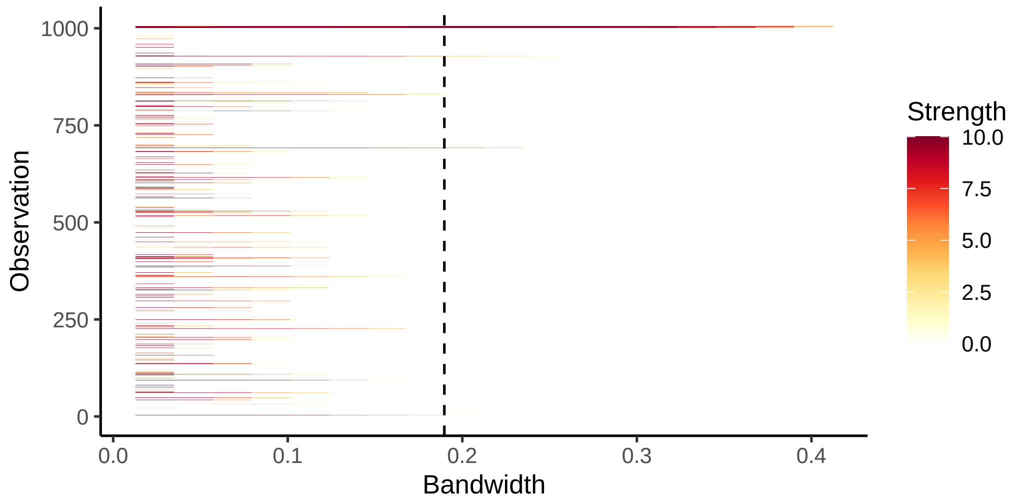

Figure 6: Outlier persistence over different bandwidth values and their strengths. The dashed line corresponds to the lookout bandwidth d_*.

Figure 6 shows the persistence of outliers over different bandwidth values and significance levels for the dataset in Figure 5. We see that points 1001–1005 are identified as outliers with high strength even for large bandwidths. This gives a comprehensive representation of outliers as it encapsulates the bandwidth as well as the strength (which corresponds to significance).

3.4 Time series outliers

Following Burridge & Taylor (2006) we extend lookout to a time series setting. They consider a time series with k additive outliers defined as

$$z_t = y_t + \sum_{j=1}^{k} \mathcal{I}_{\tau_j} x_{\tau_j},$$

$$\text{where} \qquad \mathcal{I}_{\tau_j} = \begin{cases} 1\,, & t = \tau_j \\ 0\,, & t \neq \tau_j \end{cases},$$

$$\text{and} \qquad y_t = \phi y_{t-1} + u_t,$$

with $\{u_t\}$ an autoregressive moving average process. They consider the case when $\phi = 1$ and estimate x_{τ_j} by

$$\hat{x}_{\tau_j} = \frac{1}{2}\left(\Delta z_{\tau_j} - \Delta z_{\tau_j+1}\right) = \frac{1}{2}\left(\Delta y_{\tau_j} - \Delta y_{\tau_j+1} + 2x_{\tau_j}\right)\,.$$

From the spacings of $\hat{x}_t$ they find outliers using Algorithm 1 in their paper. Due to successive differences each outlier in $\{z_t\}$ can give rise to three consecutive outliers in $\{\hat{x}_t\}$. We extend lookout to a time series setting by using it on $\{\hat{x}_t\}$. To mitigate for successive outliers, we select the observation with the highest strength or the lowest probability from each set of consecutive outliers.

4 Experiments with synthetic data

4.1 Outlier persistence examples

Figure 7 shows five simple examples in $\mathbb{R}^2$, each showing the data, outliers identified by lookout, their strengths and the corresponding outlier persistence diagram. In each example, the outliers are placed at the end of the synthetic dataset. The dashed lines in the outlier persistence diagram indicate the bandwidth chosen by lookout. The top left plot shows normally distributed data with five outliers in the top right corner, which are identified by lookout with high strength. Upon close inspection we see two further points, both approximately at $(-2.6, 1.8)$ and slightly detached from the main group of points, also being identified by lookout as outliers with low strength. The lookout strength of each of these points is less than 2, and so they would only be identified as outliers when $\alpha > 0.09$. The corresponding persistence diagram shows high strength outlier persistence for the points at the top right hand corner. Similarly, from other graphs in column 1, we see that outliers far from high density regions are identified by lookout with high strength and points close to but outside high density regions are identified with low strength. From the persistence diagrams we see that many points are identified as outliers for low bandwidth values, but only a small number of outliers persist until the bandwidth is large.

From the examples in Figure 7 we see that lookout selects a bandwidth appropriate for outlier detection that is neither too small nor too large. In addition, lookout identifies outliers correctly. The outlier persistence diagram gives a snapshot of the outliers for varying bandwidths and significance levels increasing our understanding of the dataset and its outliers.

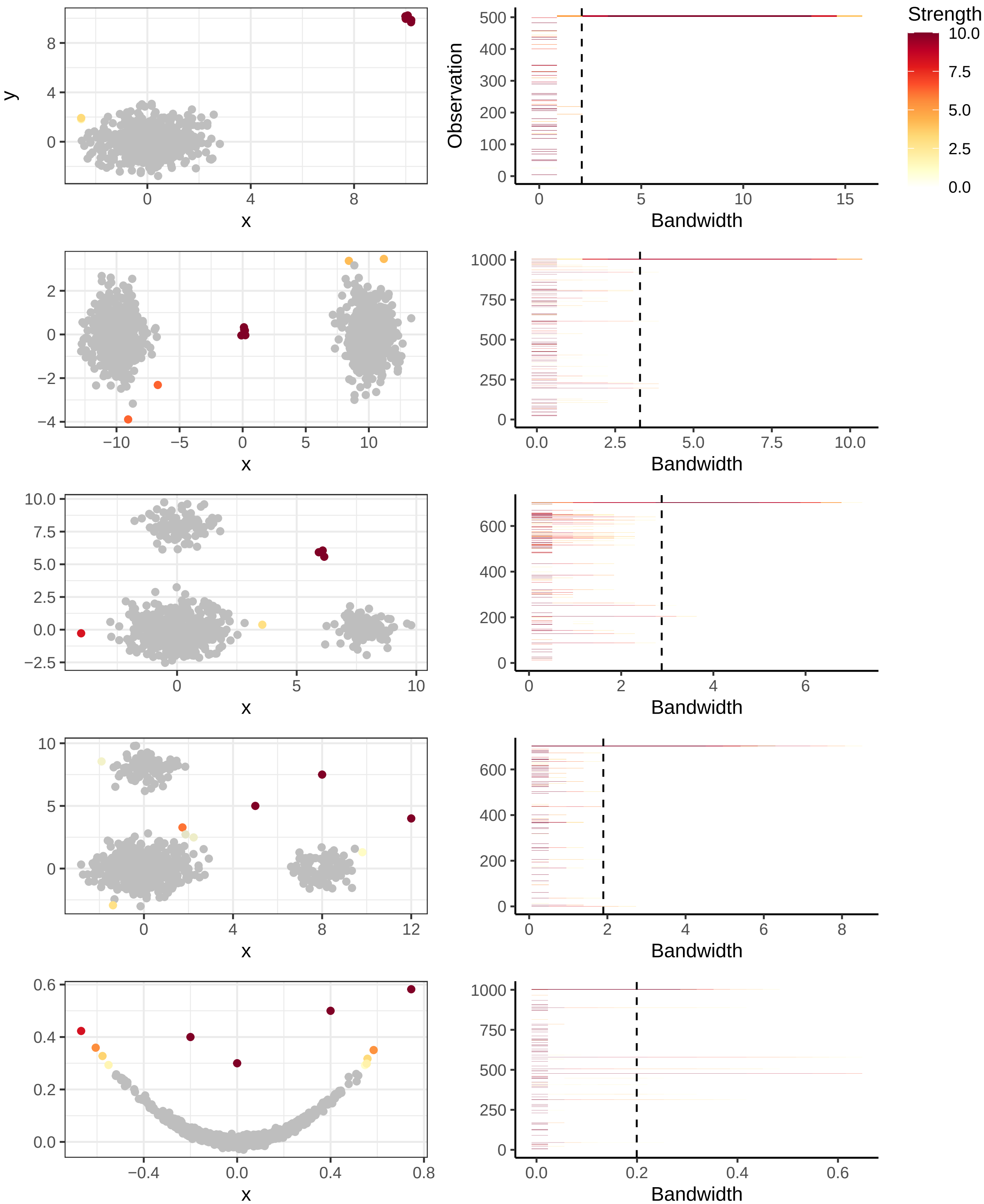

Figure 7: Examples of outlier persistence and strength. Left: data with the outliers identified by lookout with $\alpha = 0.1$ colored by its strength. Right: outlier persistence diagrams for these examples. The dashed line corresponds to the lookout bandwidth.

4.2 Comparison Study

In this section we conduct three experiments with synthetic data. Each experiment considers two data distributions; the main distribution and a handful of outliers which are distributed differently. There are several iterations to each experiment. The iterations serve as a measure of the degree of outlyingness of the small sample. The outliers start off with the main distribution and slowly move out of the main distribution with each iteration. Consequently, in the initial iterations the points in the outlying distribution are not actual outliers as they are similar to the points in the main distribution, while in the later iterations they are quite different from the main distribution. We repeat each iteration ten times to account for the randomness of the data generation process.

We compare the results of lookout with four other algorithms; HDoutliers (Wilkinson 2017), stray (Talagala et al. 2021), KDEOS (Schubert et al. 2014) and RDOS (Tang & He 2017). HDoutliers and stray both use extreme value theory for outlier detection, while KDEOS and RDOS both use kernel density estimates. For KDEOS and RDOS we use the default parameters. For lookout, HDoutliers and stray we use $\alpha = 0.05$.

HDoutliers and stray identify outliers explicitly. KDEOS and RDOS do not identify outliers directly; instead, they give an anomaly score which can be used to rank points according to how outlying they are. As the algorithms give different outputs, we employ different methods to compare their performance. To compare the performance of lookout with HDoutliers and stray, we assess the identified outliers based on the F-measure and the geometric mean of sensitivity and specificity denoted by Gmean, discussed below. To compare lookout with KDEOS and RDOS, we assess the outlier scores using the area under the Receiver Operator Characteristic curve denoted by AUC. Note that the evaluation metrics F-measure, Gmean and AUC are suited for unbalanced datasets.

The F-measure is defined as

$$\text{F-measure} = 2\frac{\text{precision} \times \text{recall}}{(\text{precision} + \text{recall})},$$

where

$$\text{precision} = \frac{tp}{tp + fp}, \qquad \text{and} \qquad \text{recall} = \frac{tp}{tp + fn},$$

where tp, fp and fn denote true positives (predicted = true, actual = true), false positives (predicted = true, actual = false) and false negatives (predicted = false, actual =true) respectively. The F-measure is undefined when both precision and recall are zero, which occurs when the true positives tp are zero. This happens when the outlier detection algorithm does not identify any correct outliers. We assign zero to the F-measure in such instances.

Sensitivity and specificity are similar evaluation metrics more frequently used in a medical diagnosis context:

$$\text{sensitivity} = \frac{tp}{tp + fn}, \qquad \text{and} \qquad \text{specificity} = \frac{tn}{tn + fp},$$

and

$$\text{Gmean} = \sqrt{\text{sensitivity} \times \text{specificity}},$$

where tn denotes the true negatives (predicted = false, actual = false). In fact, sensitivity and recall are two different terms denoting the same quantity of interest.

For all three experiments we compute the time taken for each algorithm on a Microsoft Surface Pro 3 laptop, with Intel(R) Core(TM) i5-4300U, 2.50GHz processor. We report the time taken as additional metrics.

Experiment 1

For this experiment we consider two normally distributed samples in $\mathbb{R}^6$; one large and one small starting at the same location with the small sample slowly moving out in each iteration. The set of points belonging to the small sample are considered outliers. We consider 400 points in the bigger sample and 5 in the outlying sample, placed at indices 401–405. The points in the larger sample are distributed in each dimension as $\mathcal{N}(0, 1)$. The outliers differ from the standard normal points in only the first dimension; i.e. they are distributed as $\mathcal{N}\,(2 + (i - 1)/2, 0.2)$, where i denotes the iteration. In the first iteration the outliers are distributed as $\mathcal{N}\,(2, 0.2)$ in the first dimension and in the tenth iteration they are distributed as $\mathcal{N}\,(6.5, 0.2)$. Each iteration is repeated ten times to account for randomness.

The top left graph of Figure 8 shows the first two dimensions of this experimental dataset in its last iteration and repetition, with outliers identified by lookout shown in different colors. The top right graph shows the performance comparison of lookout, HDoutliers, stray, KDEOS and RDOS. We see that for each iteration, lookout surpasses HDoutliers and stray significantly. Lookout gives much better performance than KDEOS, and from the fourth iteration onwards lookout surpasses RDOS. The graph on the bottom left shows the time

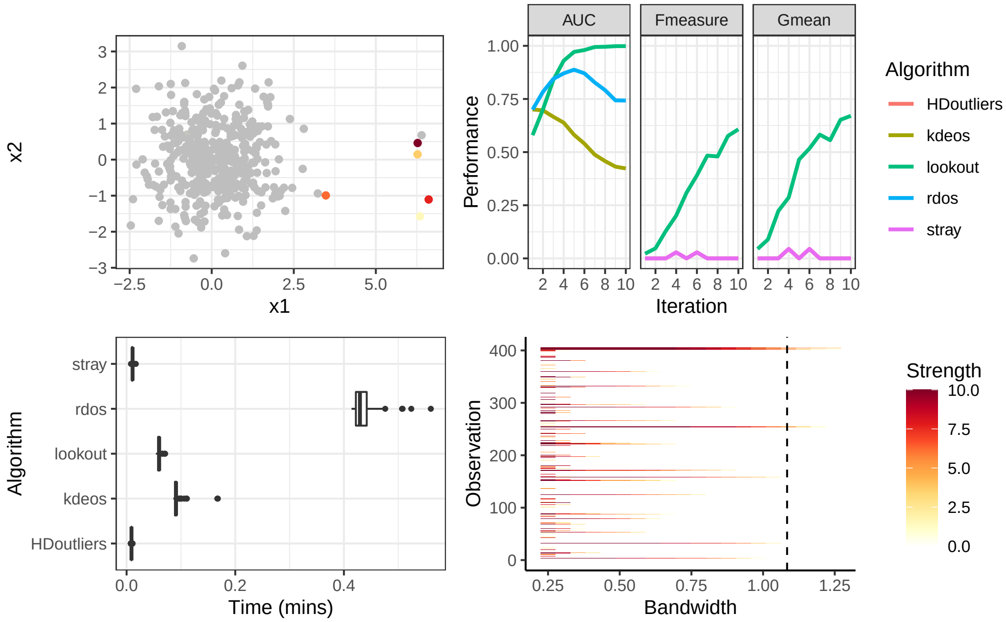

Figure 8: Experiment 1 with outliers moving out from the normal distribution in $\mathbb{R}^6$. Top left: first two dimensions in the last iteration and last repetition. Top right: performance comparison between lookout, HDoutliers, stray, kdeos and rdos using AUC, Fmeasure and Gmean over ten repetitions. In this experiment, stray and HDoutliers gave identical results. Bottom left: time taken for these five algorithms. Bottom right: corresponding outlier persistence plot.

taken for each algorithm. We see that RDOS takes a much longer time than the other algorithms. HDoutliers and stray are the fastest, followed by lookout. The graph on the bottom right shows the outlier persistence plot for this data with the dashed line denoting the lookout bandwidth.

Experiment 2

For this experiment we consider an annulus in $\mathbb{R}^2$ with outlying points moving into the center with each iteration. We consider 800 points in the annulus with five outlying points. The outliers are normally distributed and have a smaller standard deviation compared to the other points. The mean of the outliers in the i^{th} iteration is $(5 - (i-1)/2, 0)$, so that the outliers start at the right of the annulus with mean $(5, 0)$ and move in with each iteration, ending with mean $(0, 0)$. We repeat each iteration ten times.

The graph at the top left of Figure 9 shows the points in the final iteration and repetition. The graph at the top right shows the performance comparison using AUC, F-measure and Gmean. We see that lookout performs better than the other algorithms across different metrics. The graph on the bottom left shows the time taken for each algorithm. RDOS takes much longer compared to others, while HDoutliers and stray are the fastest followed by lookout. The graph on the bottom right shows the outlier persistence for the final iteration and repetition. The outliers are placed at indices 801–805 and we see that they are not identified as outliers for small bandwidth values because the outliers are clustered together. This shows the importance of bandwidth selection for outlier detection when using kernel density estimates. The bandwidth selected by lookout is shown as a dashed line.

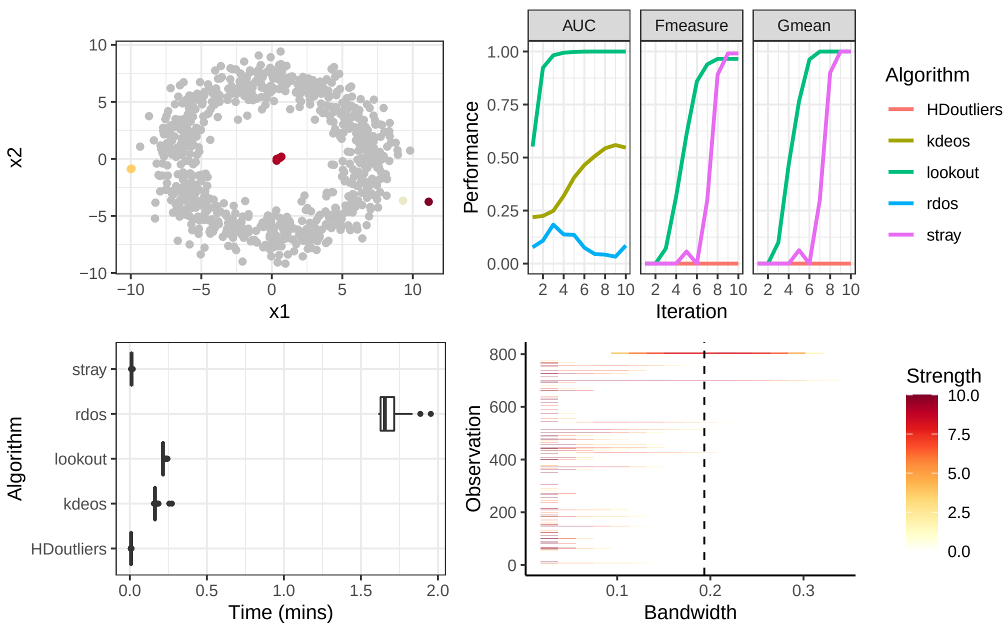

Figure 9: Experiment 2 with outliers moving into the center of the annulus in $\mathbb{R}^2$. Top left: points from the last iteration and repetition. Top right: performance comparison between lookout, HDoutliers, stray, kdeos and rdos using AUC, Fmeasure and Gmean over 10 repetitions. Bottom left: time taken for these five algorithms. Bottom right: the corresponding outlier persistence plot.

Experiment 3

For this experiment we consider a unit cube in $\mathbb{R}^{20}$ with 500 points, of which 499 points are uniformly distributed in each direction and the remaining point is an outlier. The outlier moves towards the point $(0.9, 0.9, \ldots, 0.9)$ with each iteration. For the i^{th} iteration the first i coordinates of the outlier are each equal to 0.9 and the remaining coordinates are uniformly distributed in $(0, 1)$. The index of the outlier is 500. Each iteration is repeated 10 times with different randomizations.

The top graph in Figure 10 shows the performance comparison between the different algorithms. We see that RDOS performs better than lookout in the initial eight iterations of this experiment. After the 8^{th} iteration, lookout performs better than RDOS. We also see that lookout performs significantly better than stray and HDoutliers. The graph on the bottom left shows the time taken for each algorithm: stray is the fastest followed by HDoutliers; the slowest algorithm is RDOS. The graph in the bottom right shows the outlier persistence for the last iteration and repetition. The dashed line shows the bandwidth chosen by lookout.

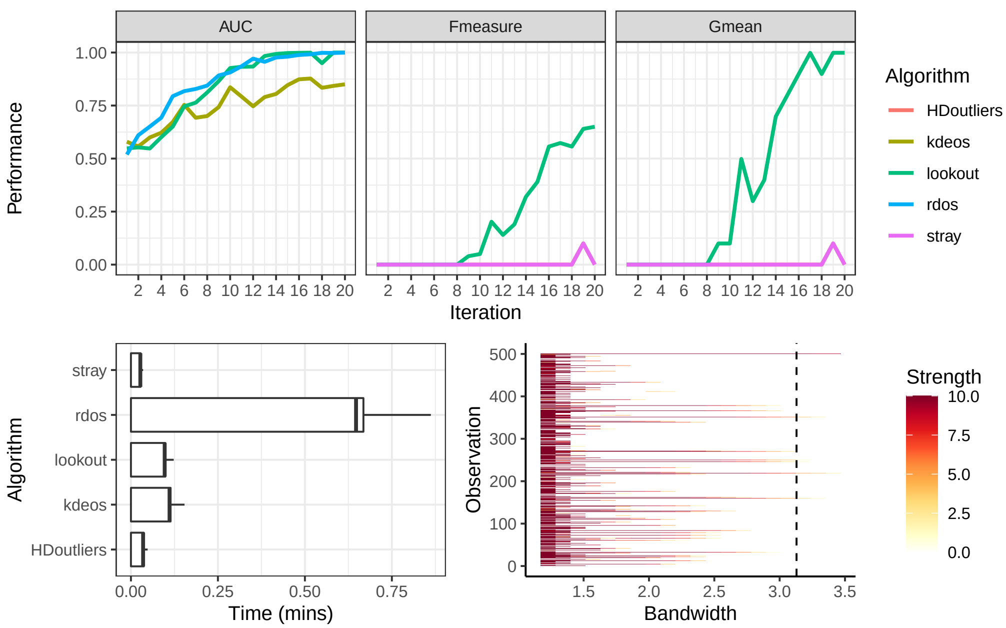

Figure 10: Experiment 3 with an outlier moving to $(0.9, 0.9, \ldots)$. Top: performance comparison between lookout, HDoutliers, stray, KDEOS and RDOS using AUC, Fmeasure and Gmean over ten repetitions. Bottom left: time taken for these five algorithms. Bottom right: corresponding outlier persistence plot.

Table 1: Specificity for lookout, HDoutliers and stray algorithms when no outliers are present.

	lookout		**stray**		**HDoutliers**	
Experiment	mean	sd	mean	sd	mean	sd
1	0.9933	0.0017	1.0000	0e+00	1.0000	0e+00
2	0.9934	0.0016	0.9999	4e-04	0.9999	4e-04
3	0.9936	0.0025	1.0000	0e+00	1.0000	0e+00

Comparison of false positives

A low false positive rate is an attractive feature for an outlier detection method. High false positives diminish user confidence with the risk of abandonment of such systems; consider a home security system with constant alarms. To get an understanding of the false positives, we redo experiments 1–3 without any outliers and compare the specificity $(tn/(tn + fp))$ of HDoutliers, stray and lookout. We repeat each experiment 10 times. As KDEOS and RDOS do not identify outliers, we do not include them in this comparison. High values of specificity is preferred as low values indicate more false positives. Table 1 shows the specificity comparison results. Stray has the highest, average specificity of 1 with zero standard deviation; HDoutliers has a similar specificity for all three experiments. We see that the average specificity of lookout is greater than 0.99 with low standard deviation values. Thus lookout's specificity is comparable to HDoutliers and stray, demonstrating that its false positive rate is low.

Evaluating time series outliers

In this section we compare the time series implementation of lookout with the outlier detection method proposed by Burridge & Taylor (2006), which we denote by BT. For this example we simulate an ARIMA$(1, 1, 0)$ time series with the autoregressive parameter taking values $0.5, 0.6, 0.7, 0.8$ and 0.9. We add an outlier to each simulated time series and repeat the simulation 10 times for each parameter value to account for randomness. Table 2 gives the results of lookout and BT using Gmean and Fmeasure. We see that lookout gives better performance than BT for all parameter values.

Table 2: Time series outliers performance comparison

	Gmean				FMeasure			
	lookout		BT		lookout		BT	
AR	mean	sd	mean	sd	mean	sd	mean	sd
0.5	0.9	0.316	0.399	0.515	0.9	0.316	0.317	0.434
0.6	1.0	0.000	0.898	0.316	1.0	0.000	0.683	0.328
0.7	1.0	0.000	0.398	0.514	1.0	0.000	0.217	0.284
0.8	0.8	0.422	0.899	0.316	0.8	0.422	0.767	0.344
0.9	1.0	0.000	0.699	0.483	1.0	0.000	0.633	0.457

5 Results on two data repositories

For this section we use two data repositories: the ODDS data repository (Rayana 2016) and the KMASH repository at Kandanaarachchi et al. (2019). Using the ODDS repository, we will look at the individual performance of these five outlier detection methods applied to 12 well-known datasets. As KMASH has more than 12000 outlier detection datasets, we will evaluate the performance of these methods on the repository and report summary statistics because the number of datasets is large.

5.1 ODDS data repository

Table 3 shows the results of lookout, HDoutliers, stray, KDEOS and RDOS on the twelve ODDS datasets. As previously, we have used F-measure and Gmean to compare the performance of lookout, HDoutliers and stray, and AUC to compare lookout, KDEOS and RDOS. We see that for datasets glass, optdigits and wine, none of the lookout, HDoutliers and stray algorithms identify any outliers. Therefore, the Gmean and F-measure are zero for these three datasets. For the other nine datasets, lookout outperforms stray and HDoutliers.

The AUC values in Table 3 show that lookout gives the best performance for 8 of the 12 datasets. For the speech dataset lookout is tied with RDOS. RDOS gives the best performance for three datasets. Two datasets give errors for KDEOS.

Table 3: Performance evaluation of 12 datasets in the ODDS repository.

	Gmean			FMeasure			AUC		
Filename	lookout	stray	HDoutliers	lookout	stray	HDoutliers	lookout	KDEOS	RDOS
cardio	0.32	0.00	0.08	0.19	0.00	0.01	0.80	0.48	0.55
glass	0.00	0.00	0.00	0.00	0.00	0.00	0.51	0.44	0.43
letter	0.10	0.00	0.00	0.02	0.00	0.00	0.50	0.58	0.91
lympho	0.58	0.41	0.00	0.50	0.29	0.00	0.99	0.77	0.98
musk	0.69	0.00	0.00	0.64	0.00	0.00	1.00		0.22
optdigits	0.00	0.00	0.00	0.00	0.00	0.00	0.46	0.59	0.75
satimage-2	0.96	0.00	0.00	0.94	0.00	0.00	0.98	0.37	0.76
speech	0.13	0.00	0.00	0.02	0.00	0.00	0.52		0.52
thyroid	0.31	0.00	0.00	0.13	0.00	0.00	0.70	0.60	0.57
vowels	0.24	0.00	0.00	0.09	0.00	0.00	0.62	0.64	0.83
wbc	0.44	0.00	0.00	0.31	0.00	0.00	0.84	0.48	0.72
wine	0.00	0.00	0.00	0.00	0.00	0.00	0.65	0.50	0.61

5.2 KMASH data repository

The KMASH repository at Kandanaarachchi et al. (2019) has more than 12000 outlier detection datasets that were prepared from classification datasets. Dataset preparation involved downsampling the minority class in the classification datasets, converting the categorical variables to numerical and accounting for missing values, all of which is detailed in Kandanaarachchi et al. (2020).

We evaluate the performance of these five outlier detection methods on this data repository. While lookout, HDoutliers and stray gave valid output for all 12000+ datasets, RDOS and KDEOS gave errors for 5745 datasets. Consequently, we only compare lookout, stray and HDoutliers on this data repository.

In Table 4, we examine the performance differences between (a) lookout and HDoutliers and (b) lookout and stray, using Gmean and Fmeasure after removing the entries that have zero values for all three algorithms. We see that for both Gmean and F-measure, the median, mean and the 95% confidence interval from Student's t-tests are away from zero. In fact, the Gmean has median values 0.1307 and 0.1385 for lookout - HDoutliers and lookout - stray

Table 4: Summary statistics for comparing lookout with the HDoutliers and stray algorithms.

Statistic	lookout - HDoutliers	lookout - stray
Fmeasure		
Median	0.1307	0.1385
Mean	0.0487	0.0838
95% Confidence Interval	(0.0385,0.0589)	(0.0743,0.0932)
Gmean		
Median	0.0405	0.0406
Mean	0.0711	0.0768
95% Confidence Interval	(0.0662,0.0760)	(0.0721,0.0815)

respectively. Similarly, the corresponding F-measure values are 0.0405 and 0.0406. Given that both Gmean and F-measure are bounded by 1, this shows that lookout gives better performance than HDoutliers or stray.

6 Conclusions

Lookout uses leave-one-out kernel density estimates and EVT to detect outliers. Outlier detection methods that use kernel density estimates generally employ a user-defined parameter to construct the bandwidth. Selecting a bandwidth for outlier detection is different from selecting a bandwidth for general data representation, because the goal is to make outliers have lower density estimates compared to the non-outliers. In addition, it is a challenge to select an appropriate bandwidth in high dimensions. To make outliers have lower density estimates compared to the rest, a reasonably large bandwidth needs to be chosen. We introduced an algorithm called *lookout* that uses persistent homology to select the bandwidth.

We compared the performance of lookout with four outlier detection algorithms, two of which use EVT to detect outliers and the others use KDE. These algorithms are HDoutliers, stray, KDEOS and RDOS. Our results on experimental data and on two data repositories showed that lookout achieves better performance.

We also introduced the concept of *outlier persistence*, exploring the birth and death of outliers with changing bandwidth and significance values. Outlier persistence gives a bigger picture, taking a step back from fixed parameter values. It explores the bandwidth and significance parameters and highlights the outliers that persist over a range of bandwidth values and their significance levels. We suggest that it is a useful measure that increases our understanding of outliers.

7 Supplementary materials

The R package `lookout` is available at `https://github.com/sevvandi/lookout`. The KMASH outlier detection data repository used in Section 5 is available at Kandanaarachchi et al. (2019). The programming scripts used in Sections 4 and 5 are available at `https://github.com/sevvandi/supplementary_material/tree/master/lookout`.

References

Bommier, E. (2014), Peaks-over-threshold modelling of environmental data, Master's thesis, Department of Mathematics, Uppsala University.

Brunson, J. C., Wadhwa, R. & Scott, J. (2020), *ggtda: ggplot2 Extension to Visualize Topological Persistence*. R package version 0.1.0.
URL: *https://github.com/rrrlw/ggtda*

Burridge, P. & Taylor, A. M. R. (2006), 'Additive outlier detection via extreme-value theory', *Journal of Time Series Analysis* **27**(5), 685–701.

Carlsson, G. (2009), 'Topology and data', *Bulletin of the American Mathematical Society* **46**(2), 255–308.

Carlsson, G., Ishkhanov, T., De Silva, V. & Zomorodian, A. (2008), 'On the local behavior of spaces of natural images', *International Journal of Computer Vision* **76**(1), 1–12.

Chaudhuri, P. & Marron, J. S. (1999), 'SiZer for Exploration of Structures in Curves', *Journal of the American Statistical Association* **94**(447), 807–823.

Clifton, D. A., Clifton, L., Hugueny, S. & Tarassenko, L. (2014), 'Extending the generalised Pareto distribution for novelty detection in high-dimensional spaces', *Journal of Signal Processing Systems* **74**(3), 323–339.

Coles, S. (2001), *An introduction to statistical modeling of extreme values*, Vol. 208 of *Springer Series in Statistics*, Springer, London UK.

Ghrist, R. (2008), 'Barcodes: the persistent topology of data', *Bulletin of the American Mathematical Society* **45**(1), 61–75.

Kandanaarachchi, S. & Hyndman, R. J. (2021), *lookout: Leave One Out Kernel Density Estimates for Outlier Detection.* R package version 0.1.0.
URL: *https://cran.r-project.org/web/packages/lookout/*

Kandanaarachchi, S., Muñoz, M. A., Hyndman, R. J. & Smith-Miles, K. (2020), 'On normalization and algorithm selection for unsupervised outlier detection', *Data Mining and Knowledge Discovery* **34**(2), 309–354.

Kandanaarachchi, S., Muñoz, M. A., Smith-Miles, K. & Hyndman, R. J. (2019), 'Datasets for outlier detection'.
URL: *https://doi.org/10.26180/5c6253c0b3323*

Minnotte, M. C. & Scott, D. W. (1993), 'The mode tree: A tool for visualization of nonparametric density features', *Journal of Computational and Graphical Statistics* **2**(1), 51–68.

Perea, J. A. & Harer, J. (2015), 'Sliding windows and persistence: An application of topological methods to signal analysis', *Foundations of Computational Mathematics* **15**(3), 799–838.

Pickands, J. (1975), 'Statistical Inference Using Extreme Order Statistics', *The Annals of Statistics* **3**(1), 119–131.

Qin, X., Cao, L., Rundensteiner, E. A. & Madden, S. (2019), Scalable kernel density estimation-based local outlier detection over large data streams, *in* 'Proceedings of the 22nd International Conference on Extending Database Technology (EDBT)', pp. 421–432.

Rayana, S. (2016), 'ODDS library'. Stony Brook University, Department of Computer Sciences.
URL: *http://odds.cs.stonybrook.edu*

Reiss, R.-D. & M.Thomas (2001), *Statistical Analysis of Extreme Values: with Applications to Insurance, Finance, Hydrology and Other Fields*, 3rd edn, Birkhäuser, Basel, Switzerland.

Schubert, E., Zimek, A. & Kriegel, H. P. (2014), 'Generalized outlier detection with flexible kernel density estimates', *SIAM International Conference on Data Mining 2014* pp. 542–550.

Scott, D. W. (2015), *Multivariate Density Estimation: Theory, Practice, and Visualization.*, 2nd edn, John Wiley & Sons, Hoboken, NJ.

Stephenson, A. G. (2002), 'evd: Extreme value distributions', *R News* **2**(2), 31–32.

Talagala, P. D., Hyndman, R. J. & Smith-Miles, K. (2021), 'Anomaly detection in high-dimensional data', *Journal of Computational and Graphical Statistics* **30**(2), 360–374.

Talagala, P. D., Hyndman, R. J., Smith-Miles, K., Kandanaarachchi, S. & Muñoz, M. A. (2020), 'Anomaly detection in streaming nonstationary temporal data', *Journal of Computational and Graphical Statistics* **29**(1), 13–27.

Tang, B. & He, H. (2017), 'A local density-based approach for outlier detection', *Neurocomputing* **241**, 171–180.

Topaz, C. M., Ziegelmeier, L. & Halverson, T. (2015), 'Topological data analysis of biological aggregation models', *PloS ONE* **10**(5), e0126383.

Wadhwa, R. R., Williamson, D. F. K., Dhawan, A. & Scott, J. G. (2018), 'TDAstats: R pipeline for computing persistent homology in topological data analysis', *Journal of Open Source Software* **3**(28), 860.
URL: *https://doi.org/10.21105/joss.00860*

Wang, Z. & Scott, D. W. (2019), 'Nonparametric density estimation for high-dimensional data: Algorithms and applications', *Wiley Interdisciplinary Reviews: Computational Statistics* **11**(4), e1461.

Wasserman, L. (2018), 'Topological data analysis', *Annual Review of Statistics and Its Application* **5**, 501–532.

Wilkinson, L. (2017), 'Visualizing big data outliers through distributed aggregation', *IEEE Transactions on Visualization and Computer Graphics* **24**(1), 256–266.