JDSSV

Journal of Data Science, Statistics, and Visualisation

June 2025, Volume V, Issue VIII. *doi: 10.52933/jdssv.v5i8.152*

Kernel Outlier Detection

Can Hakan Dağıdır
KU Leuven

Mia Hubert
KU Leuven

Peter J. Rousseeuw
KU Leuven

Abstract

A new anomaly detection method called kernel outlier detection (KOD) is proposed. It is designed to address challenges of outlier detection in high-dimensional settings. The aim is to overcome limitations of existing methods, such as dependence on distributional assumptions or on hyperparameters that are hard to tune. KOD starts with a kernel transformation, followed by a projection pursuit approach. Its novelties include a new ensemble of directions to search over, and a new way to combine results of different direction types. This provides a flexible and lightweight approach for outlier detection. Our empirical evaluations illustrate the effectiveness of KOD on three small datasets with challenging structures, and on four large benchmark datasets.

Keywords: Anomaly detection, outlyingness, kernel transformation, projection depth.

1. Introduction

Most outlier detection methods for multivariate and possibly high-dimensional data assume that the non-outlying data are drawn from an elliptical distribution. However, many modern datasets such as images, sensor data or genetic data do not obey this assumption. In this work, we propose a kernel outlier detection method that does not rely on this assumption, by combining kernel transformations with the Stahel-Donoho outlyingness (Stahel 1981; Donoho 1982).

The Stahel-Donoho outlyingness (SDO) assigns an outlyingness value to each data point by considering the most extreme standardized deviation across all possible projection directions. In other words, it searches for the direction along which the observation deviates the most from the majority of the data. Formally, given a data matrix $(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)^T$ with p-variate data points $\boldsymbol{x}_i = (x_{i1}, \ldots, x_{ip})^T$, the SDO of each case is defined as

$$\mathrm{SDO}(\boldsymbol{x}_i) := \sup_{\boldsymbol{v} \in \mathcal{B}} \frac{\left| \boldsymbol{v}^T \boldsymbol{x}_i - \mathrm{med}_j(\boldsymbol{v}^T \boldsymbol{x}_j) \right|}{\mathrm{MAD}_j(\boldsymbol{v}^T \boldsymbol{x}_j)} \tag{1}$$

where $\mathcal{B} = \{\boldsymbol{v} \,;\, \|\boldsymbol{v}\| = 1\}$ is the set of all p-variate vectors of length 1, med denotes the median, and MAD is the median absolute deviation, defined as

$$\mathrm{MAD}(y_1, \ldots, y_n) = 1.483 \, \mathrm{med}_i |y_i - \mathrm{med}_j(y_j)| \tag{2}$$

for a univariate sample $y_1, \ldots, y_n$. The SDO is a projection pursuit method. It is based on the idea that if a point is a multivariate outlier, then there must be some one-dimensional projection of the data in which the point is a univariate outlier. It further relies on the empirical observation that one-dimensional projections of many high-dimensional data clouds often have a roughly normal distribution.

However, these assumptions are not always satisfied in real data. We illustrate the SDO on the bivariate dataset in the left panel of Figure 1. We call it the 'Inside-Outside data set', as it has 800 regular observations scattered around a circle (in green), 100 clustered outliers inside the circle, and 100 outliers near a circle with a larger radius. The outliers are shown in black. The middle panel shows the SDO values of these 1,000 points. We see that the inner cluster has much smaller outlyingness than the regular points.

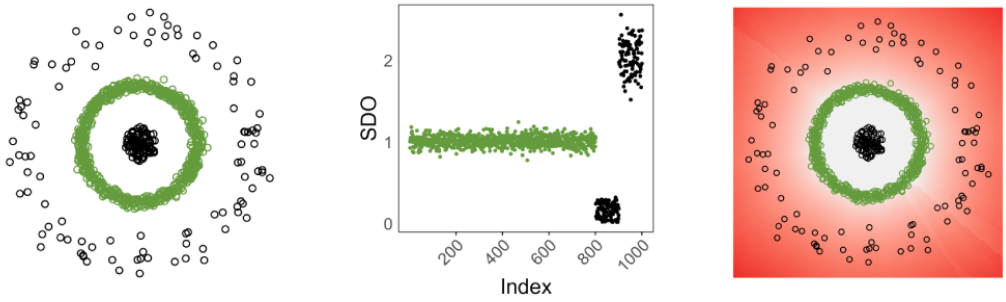

Figure 1: Inside-Outside dataset (left), the corresponding SDO values (middle) and their heatmap (right).

Note that the SDO can also be computed in an arbitrary point $\boldsymbol{x}$, which does not have to be a data point. All we need to do in (1) is replace $\boldsymbol{x}_i$ by $\boldsymbol{x}$, yielding $\mathrm{SDO}(\boldsymbol{x}) = \sup_{\boldsymbol{v} \in \mathcal{B}} \{ |\boldsymbol{v}^T \boldsymbol{x} - \mathrm{med}_j(\boldsymbol{v}^T \boldsymbol{x}_j)| / \mathrm{MAD}_j(\boldsymbol{v}^T \boldsymbol{x}_j) \}$. The rightmost panel of Figure 1 is a heatmap, obtained by computing the SDO on a fine grid of points $\boldsymbol{x}$ and coloring the results. The region with an SDO below the median SDO of the entire dataset is colored white, and the color gradually goes from white to red for increasing outlyingness. We see that the SDO does not flag the data points inside the regular circle as outliers.

The SDO values in Figure 1 were obtained from the function `depth.projection` in the R package **ddalpha** (Pokotylo et al. 2019) which computes the exact SDO values.

Whereas the computation of the exact SDO is feasible in small dimensions, it becomes too demanding in high dimensions due to the enormous set of unit vectors $\mathcal{B}$ in (1). Maronna and Yohai (1995) noted that even when replacing the median and the MAD by smooth estimators of location and scale, the function inside (1) has multiple local maxima, which makes gradient-based methods ineffective. Approximate algorithms to compute the SDO typically rely on using a specific subset of directions in $\mathcal{B}$. The oldest approach is to draw a random subsample of p data points, and to use a unit vector orthogonal to the affine hyperplane they span (Stahel 1981). This is then repeated many times. The advantage of generating these directions is that the resulting SDO values do not change when the data are subjected to a nonsingular linear transformation. This property is called affine invariance. However, the p-subset approach becomes very time consuming for high-dimensional data, and cannot be applied when there are more dimensions than data points.

A less computationally expensive approach is to consider directions through two randomly selected data points (Hubert et al. 2005; Segaert et al. 2024). The resulting SDO values are invariant to orthogonal transformations of the data, which is appropriate in for instance the context of principal component analysis (PCA). Another algorithm is based on randomly drawing directions from the uniform distribution on the hypersphere, as in Velasco-Forero and Angulo (2012). But according to Dyckerhoff et al. (2021) this yields poor performance that degrades with increasing dimension.

To relax the elliptical assumption underlying the SDO, Hubert and Van Der Veeken (2008) proposed a version of the SDO adjusted for skewed data, based on a robust measure of skewness. A different approach is proposed in Tamamori (2023). First the data are transformed to a kernel feature space, then their dimension is reduced by kernel PCA, and finally their SDO is computed using directions drawn randomly from the uniform distribution on the unit hypersphere. Kernel transformations can be beneficial in this context, because they make linear separations more feasible, which improves the performance of SDO. But as the resulting Kernel Random Projection Depth (KRPD) relies exclusively on these directions, its ability to identify outliers is limited, especially in high-dimensional scenarios. This will be illustrated empirically in Section 4.

Several other kernel-based outlier detection methods have been proposed, such as the One-Class Support Vector Machine (OCSVM) of Schölkopf et al. (1999) and the Kernel Minimum Regularized Covariance Determinant (KMRCD) method of Schreurs et al. (2021). The widely adopted OCSVM method suffers from sensitivity to hyperparameter choices, which is especially problematic in unsupervised contexts such as outlier detection, where cross validation is not possible. The KMRCD method combines robust covariance estimation and kernel transformations, but still relies on elliptical distribution assumptions in the kernel feature space, restricting its generality.

To address these issues we propose a novel Kernel Outlier Detection (KOD) approach. It contains two main novelties: (i) we introduce intuitive and computationally efficient directions that naturally arise from kernel transformations, and (ii) we propose a robust aggregation mechanism for combining multiple direction types, which enhances detection accuracy.

Our empirical validation uses synthetic toy datasets and high-dimensional image datasets. We compare KOD with existing unsupervised methods, including OCSVM, KMRCD, k-Nearest Neighbors, Local Outlier Factor (LOF) (Breunig et al. 2000), and Isolation Forest (IF) (Liu et al. 2008). Although no single method outperforms all others across all scenarios, KOD performed consistently among the best, especially on highly contaminated datasets.

The remainder of this paper is organized as follows. In Section 2, we describe the proposed methodology, and then illustrate it on challenging small datasets in Section 3. In Section 4, we describe a series of experiments on these small datasets and on large-scale real-world datasets, to investigate the performance against competing methods. Finally, Section 5 concludes with a discussion.

2. The Kernel Outlier Detection Method

The proposed methodology consists of three main steps. First, we map the data into a high-dimensional feature space using kernel functions, and derive a finite-dimensional representation that preserves the essential structure of the data. Next, we adapt the SDO measure for high-dimensional data by constructing computationally feasible subsets of relevant directions. Finally, we combine the results from these subsets and identify outliers.

2.1. Constructing feature vectors

We begin by mapping the data into a feature space. Let $\mathcal{X}$ denote the input space, and consider a dataset $\mathbf{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\} \subset \mathcal{X}$. We define a feature mapping $\phi : \mathcal{X} \to \mathcal{F}$, where the feature space $\mathcal{F}$ is a vector space with an inner product $\langle \cdot, \cdot \rangle_{\mathcal{F}}$ defined on it. For each observation $\boldsymbol{x}_i$ we call $\phi(\boldsymbol{x}_i)$ its feature vector. A kernel function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is defined by

$$k(\boldsymbol{x}, \boldsymbol{x}') = \langle \phi(\boldsymbol{x}), \phi(\boldsymbol{x}') \rangle_{\mathcal{F}} \qquad \text{for } \boldsymbol{x}, \boldsymbol{x}' \in \mathcal{X}.$$

We consider only positive semidefinite (PSD) kernel functions to ensure that k defines a valid inner product in the feature space, as guaranteed by Mercer's theorem (Schölkopf and Smola 2001). A symmetric function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is called PSD if for any finite set $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\} \subset \mathcal{X}$ and real coefficients $\{c_1, \ldots, c_n\} \subset \mathbb{R}$ it holds that

$$\sum_{i=1}^{n} \sum_{j=1}^{n} c_i c_j k(\boldsymbol{x}_i, \boldsymbol{x}_j) \geqslant 0.$$

A commonly used kernel is the Radial Basis Function (RBF) defined as

$$k(\boldsymbol{x}, \boldsymbol{x}') = \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{x}'\|^2}{2\sigma^2} \right), \tag{3}$$

where σ is a tuning parameter. The RBF kernel is bounded, which turns out to yield favorable robustness properties (Debruyne et al. 2009). To select σ we employ the *median heuristic* (Gretton et al. 2012; Schreurs et al. 2021) given by

$$\sigma^2 = \operatorname{med}\left\{ \|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2 \; ; \; 1 \leqslant i < j \leqslant n \right\}. \tag{4}$$

This heuristic assumes that all the input variables are measured in the same units. If this assumption does not hold we first standardize the data, for instance by subtracting the columnwise median and dividing by the columnwise MAD. Note that in supervised learning tasks, there is a categorical or numerical response variable that one tries to fit according to a criterion, and then one can choose σ through cross-validation. But in our context of unsupervised outlier detection there is no response variable, so cross-validation is not applicable.

From a PSD kernel function one constructs an $n \times n$ kernel matrix $\mathbf{K}$ with entries $k_{ij} = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$. This kernel matrix contains all pairwise inner products of the feature vectors, so $\mathbf{K} = \boldsymbol{\Phi}\boldsymbol{\Phi}^T$ with $\boldsymbol{\Phi} = [\phi(\boldsymbol{x}_1), \ldots, \phi(\boldsymbol{x}_n)]^T$. To work efficiently within the feature space, we aim to obtain a finite-dimensional representation of it, even if the full feature space has infinitely many dimensions. First, we restrict ourselves to the subspace spanned by the feature vectors $\phi(\boldsymbol{x}_i)$, as it contains all the available information. To compute this subspace, we consider the centered feature vectors

$$\tilde{\phi}(\boldsymbol{x}_i) = \phi(\boldsymbol{x}_i) - \frac{1}{n}\sum_{i=1}^{n} \phi(\boldsymbol{x}_i)$$

that span a subspace $\widetilde{\mathcal{F}}$ of dimension at most $n-1$. The $n \times n$ centered kernel matrix $\widetilde{\mathbf{K}}$ has entries

$$\tilde{k}_{ij} = \langle \tilde{\phi}(\boldsymbol{x}_i), \tilde{\phi}(\boldsymbol{x}_j) \rangle_{\mathcal{F}} \,.$$

It can be computed directly from $\mathbf{K}$ as

$$\begin{aligned}
\tilde{k}_{ij} &= \left(\phi(\boldsymbol{x}_i) - \frac{1}{n}\sum_{\ell=1}^{n} \phi(\boldsymbol{x}_\ell)\right)^T \left(\phi(\boldsymbol{x}_j) - \frac{1}{n}\sum_{\ell'=1}^{n} \phi(\boldsymbol{x}'_\ell)\right) \\
&= k_{ij} - \frac{1}{n}\sum_{\ell=1}^{n} k_{\ell j} - \frac{1}{n}\sum_{\ell'=1}^{n} k_{i\ell'} + \frac{1}{n^2}\sum_{\ell=1}^{n}\sum_{\ell'=1}^{n} k_{\ell\ell'} \\
&= (\mathbf{K} - \mathbf{1}_{nn}\mathbf{K} - \mathbf{K}\mathbf{1}_{nn} + \mathbf{1}_{nn}\mathbf{K}\mathbf{1}_{nn})_{ij} \,, \qquad (5)
\end{aligned}$$

where $\mathbf{1}_{nn}$ is the $n \times n$ matrix with all entries equal to $1/n$.

Kernel methods typically work with $\mathbf{K}$ or $\widetilde{\mathbf{K}}$ without the need to explicitly know or compute the feature map. This approach is known as the *kernel trick*. In our proposed outlier detection method, we will however construct an $n \times r$ matrix $\mathbf{F}$ (with $r = \operatorname{rank}(\mathbf{K}) \leqslant n-1$) that satisfies

$$\widetilde{\mathbf{K}} = \mathbf{F}\mathbf{F}^T.$$

The matrix $\mathbf{F}$ can easily be derived from the spectral decomposition $\widetilde{\mathbf{K}} = \mathbf{V}\boldsymbol{\Lambda}\mathbf{V}^T$ of the centered kernel matrix, where $\mathbf{V}$ contains the eigenvectors of $\widetilde{\mathbf{K}}$ corresponding to the r strictly positive eigenvalues $\lambda_1 \geqslant \lambda_2 \geqslant \cdots \geqslant \lambda_r > 0$ that form the diagonal matrix $\boldsymbol{\Lambda}$. The matrix $\mathbf{F} = \mathbf{V}\boldsymbol{\Lambda}^{1/2}$ then satisfies $\mathbf{F}\mathbf{F}^T = \mathbf{V}\boldsymbol{\Lambda}^{1/2}(\mathbf{V}\boldsymbol{\Lambda}^{1/2})^T = \mathbf{V}\boldsymbol{\Lambda}\mathbf{V}^T = \widetilde{\mathbf{K}}$.

Note that this decomposition is not unique, as for any $r \times r$ orthogonal matrix $\mathbf{U}$ also $\mathbf{F}\mathbf{U}$ satisfies $(\mathbf{F}\mathbf{U})(\mathbf{F}\mathbf{U})^T = \mathbf{F}\mathbf{U}\mathbf{U}^T\mathbf{F}^T = \mathbf{F}\mathbf{F}^T = \widetilde{\mathbf{K}}$. However, we prefer to work with the matrix $\mathbf{F}$ because its columns are ranked by decreasing variance.

A different way to compute $\mathbf{F}$ is by means of a transformation matrix. Since all diagonal entries of $\boldsymbol{\Lambda}$ are strictly positive, we can compute $\boldsymbol{\Lambda}^{-1/2}$ as the $r \times r$ diagonal matrix

with diagonal entries $1/\sqrt{\lambda_1}, \ldots, 1/\sqrt{\lambda_r}$. We then construct the $n \times r$ transformation matrix $\mathbf{T}$ as $\mathbf{T} := \mathbf{V}\mathbf{\Lambda}^{-1/2}$. Then $\mathbf{F} = \widetilde{\mathbf{K}}\mathbf{T}$ since

$$\widetilde{\mathbf{K}}\mathbf{T} = (\mathbf{V}\mathbf{\Lambda}\mathbf{V}^T)(\mathbf{V}\mathbf{\Lambda}^{-1/2}) = \mathbf{V}\mathbf{\Lambda}(\mathbf{V}^T\mathbf{V})\mathbf{\Lambda}^{-1/2} = \mathbf{V}\mathbf{\Lambda}^{1/2} = \mathbf{F}.$$

The rows $\boldsymbol{f}_i$ of $\mathbf{F}$ are representations of the data points in the subspace $\widetilde{\mathcal{F}}$. They are equal to the feature vectors $\tilde{\phi}(\boldsymbol{x}_i)$ up to an orthogonal transformation. At large sample sizes, they can still be high dimensional, making all following computations intensive. Hence, in the decomposition of $\widetilde{\mathbf{K}}$, we retain only the q largest eigenvalues such that their cumulative sum accounts for at least 99% of the total sum, i.e., we set q as the smallest value for which

$$\frac{\sum_{j=1}^{q} \lambda_j}{\sum_{j=1}^{r} \lambda_j} \geqslant 0.99. \tag{6}$$

We define $\mathbf{\Lambda}_q$ as the $q \times q$ diagonal matrix of the retained eigenvalues, as well as the $n \times q$ matrix $\mathbf{V}_q$ containing the corresponding eigenvectors. Then we construct the matrix $\hat{\mathbf{F}} = \mathbf{V}_q\mathbf{\Lambda}_q^{1/2}$ which satisfies $\hat{\mathbf{F}}\hat{\mathbf{F}}^T \approx \widetilde{\mathbf{K}}$. We can also compute $\hat{\mathbf{F}}$ as

$$\hat{\mathbf{F}} = \widetilde{\mathbf{K}}\mathbf{T}_q \quad \text{with} \quad \mathbf{T}_q = \mathbf{V}_q\mathbf{\Lambda}_q^{-1/2}. \tag{7}$$

We call the rows $\hat{\boldsymbol{f}}_i$ of $\hat{\mathbf{F}}$ the *approximate feature vectors*. They can also be obtained as the scores from applying PCA to the feature vectors $\phi(\boldsymbol{x}_i)$, known as kernel PCA (Schölkopf et al. 1998). But whereas kernel PCA typically aims to reduce the dimension a lot (i.e., using a low q), our approach focuses on preserving enough structure for outlier detection. This motivates selecting the high threshold of 0.99 in Equation (6). The computation of $\hat{\mathbf{F}}$ can be carried out using the `makeFV` function in the R package **classmap** (Raymaekers et al. 2022; Raymaekers and Rousseeuw 2023).

Note that not all kernel matrices originate from numerical data. For instance, a string kernel can produce a kernel matrix from text or parts of DNA. Our methodology can handle such kernel matrices as well.

2.2. Constructing directions

The second step of the KOD method is to compute a measure of outlyingness applied to the approximate feature vectors $\hat{\boldsymbol{f}}_i$. In order to compute the SDO of (1) we need a set of directions. For this purpose, we construct four different types of directions, that we will then combine.

One Point type: a first set of directions $\mathcal{B}_1$ consists of the n directions passing through each point $\hat{\boldsymbol{f}}_i$ and a robust center. Similar to the choices made in Hubert et al. (2002) and Croux and Ruiz-Gazen (2005) in the context of robust PCA, we use the L_1-median as center. This estimator can withstand up to 50% of outliers. It is also orthogonally invariant, which is relevant in our context since rotations in feature space leave the kernel matrix unchanged, and the orthogonal equivariance of this center ensures that the SDO values remain unchanged as well. The L_1-median is computed using the NLM algorithm described in Fritz et al. (2012) and provided in the R package **pcaPP** (Filzmoser et al. 2006). At large sample sizes n, a random subset of $\mathcal{B}_1$ could be used, but in our examples we will consider all n directions.

Two Point type: the second set of directions $\mathcal{B}_2$ is given by lines passing through pairs of feature points $\hat{\boldsymbol{f}}_i$ and $\hat{\boldsymbol{f}}_j$ as in Hubert et al. (2005). We restrict $\mathcal{B}_2$ to at most 5,000 directions, selected randomly from the $\binom{n}{2}$ possible directions.

Basis Vector type: the third set $\mathcal{B}_b$ is new. It consists of the q basis vectors computed when constructing the $\hat{\boldsymbol{f}}_i$ and is natural in our context. Since they correspond to the first q principal components of the feature vectors $\phi(\boldsymbol{x}_i)$, it can be expected that some of these basis vectors point in the direction of outliers because the resulting projections yield large variances. We observed empirically that this often does happen. In Figure 2, we see a scatterplot of the $(\hat{f}_{i1}, \hat{f}_{i3})$ for the Inside-Outside dataset of Figure 1 after applying the RBF kernel. The kernel transformation has created a linear separation between the three groups in the data. Projecting the points onto the third basis vector then yields large SDO values for both the 'outside' and the 'inside' outliers.

Random type: we also randomly draw vectors from the unit hypersphere in q dimensions. This set of directions we denote as $\mathcal{B}_r$. Our default number of random directions is set to 1,000 as in Tamamori (2023).

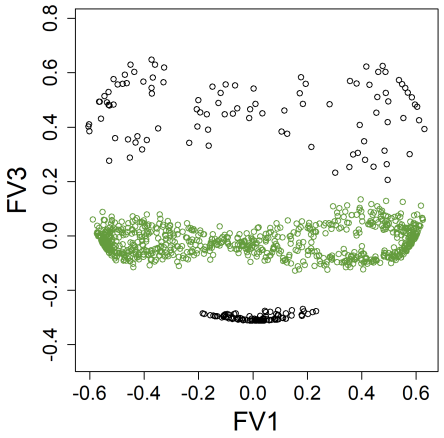

Figure 2: Scatterplot of the third versus the first coordinate in feature space, of the Inside-Outside dataset of Figure 1.

2.3. Flagging outliers

For each set of directions, we can now compute the SDO of the $\hat{\boldsymbol{f}}_i$ as in (1). In experiments, we noticed that for some directions the denominator could become extremely small, thereby unduly enlarging the result. To avoid this, we impose a lower bound c_d on the denominator. This c_d is obtained in a data-driven manner by

$$c_d = \frac{1}{5} \operatorname*{med}_{\boldsymbol{v} \in \mathcal{B}_r} \left(\mathrm{MAD}_j(\boldsymbol{v}^T \hat{\boldsymbol{f}}_j) \right) \ . \tag{8}$$

Next, we compute the outlyingness for each set of directions. The index `type` ranges over 1, 2, b and r, corresponding to the sets $\mathcal{B}_1, \mathcal{B}_2, \mathcal{B}_b$ and $\mathcal{B}_r$. For each `type` we get

$$\mathrm{outl}_{\mathrm{type}}(\hat{\boldsymbol{f}}_i) = \max_{\boldsymbol{v}\in\mathcal{B}_{\mathrm{type}}} \frac{\left|\boldsymbol{v}^T\hat{\boldsymbol{f}}_i - \mathrm{med}_j(\boldsymbol{v}^T\hat{\boldsymbol{f}}_j)\right|}{\max\left(\mathrm{MAD}_j(\boldsymbol{v}^T\hat{\boldsymbol{f}}_j), c_d\right)}. \tag{9}$$

Now we still have to combine these four measures of outlyingness into a single one. For this purpose, we first normalize each $\mathrm{outl}_{\mathrm{type}}$ by its median, and then take the largest normalized value. This yields the *Kernel Outlyingness* (KO) of each case i, given by

$$\mathrm{KO}_i = \max_{\mathrm{type}} \left(\frac{\mathrm{outl}_{\mathrm{type}}(\hat{\boldsymbol{f}}_i)}{\mathrm{med}_j(\mathrm{outl}_{\mathrm{type}}(\hat{\boldsymbol{f}}_j))} \right). \tag{10}$$

The normalization ensures that no single direction type dominates, as we observed empirically that $\mathrm{outl}_{\mathrm{type}}$ can have a rather different behavior for different types.

The last task is to flag outliers based on the KO values. Empirical evidence, such as provided by Rousseeuw et al. (2018), suggests that in high-dimensional settings the distribution of outlyingness values for non-outliers often approximates a log-normal distribution. Hence, we first apply a logarithmic transformation to the KO_i:

$$\mathrm{LO}_i = \log(0.1 + \mathrm{KO}_i), \tag{11}$$

and then estimate their center and scale with the Huber M-estimator of location $\hat{\mu}_M$ (Huber 1964) and the Q_n estimator of scale $\hat{\sigma}_{Qn}$ (Rousseeuw and Croux 1993). Next, we determine the cutoff value as

$$c = \exp\left(\hat{\mu}_M(\mathrm{LO}) + z_{0.99}\hat{\sigma}_{Qn}(\mathrm{LO})\right) - 0.1, \tag{12}$$

where $z_{0.99}$ is the 99th percentile of the standard normal distribution. Finally, the data points with $\mathrm{KO}_i \geqslant c$ are flagged as outliers.

The full procedure is summarized in Algorithm 1.

Algorithm 1 Kernel Outlier Detection

Input: Dataset $\mathbf{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\} \subset \mathcal{X}$, kernel function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$.

1: Compute the kernel matrix $\mathbf{K}$, and its centered version $\widetilde{\mathbf{K}}$ given by (5).
2: Perform the spectral decomposition of $\widetilde{\mathbf{K}} = \mathbf{V}\boldsymbol{\Lambda}\mathbf{V}^T$ and construct the approximate feature matrix $\hat{\mathbf{F}} = \mathbf{V}_q\boldsymbol{\Lambda}_q^{1/2}$.
3: Create the subsets of directions $\mathcal{B}_1, \mathcal{B}_2, \mathcal{B}_b$ and $\mathcal{B}_r$.
4: Compute a lower bound for the denominator according to (8).
5: Compute the outlyingness for each $\hat{\boldsymbol{f}}_i$ and each direction type, following (9).
6: Scale the outlyingness values by their medians and compute the kernel outlyingness as in (10).
7: Obtain a cutoff c for outlier detection using (11) and (12).
8: Flag data points as outliers if $\mathrm{KO}_i \geqslant c$.

Output: Kernel outlyingness KO_i of all cases, and the set of flagged outliers.

When we need to compute the KO of out-of-sample cases $\{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_m\}$, we first compute all the kernel values $k^{yx}(\boldsymbol{y}_i, \boldsymbol{x}_j)$ for $i = 1, \ldots, m$ and $j = 1, \ldots, n$ and store them in the $m \times n$ matrix $\mathbf{K}^{yx}$. Then we compute the centered kernel matrix

$$\widetilde{\mathbf{K}}^{yx} = \mathbf{K}^{yx} - \mathbf{K}^{yx}\mathbf{1}_{nn} - \mathbf{1}_{mn}\mathbf{K} + \mathbf{1}_{mn}\mathbf{K}\mathbf{1}_{nn}$$

with $\mathbf{1}_{mn}$ the $m \times n$ matrix with all elements equal to $1/n$. Finally we compute the approximate feature vectors $\hat{\mathbf{F}}^y = \widetilde{\mathbf{K}}^{yx}\mathbf{T}_q$ as in (7). For each set of directions $\mathcal{B}_{\text{type}}$, the outlyingness of each $\hat{\boldsymbol{f}}^y$ is computed as in (9) where the center $\text{med}_j(\boldsymbol{v}^T\hat{\boldsymbol{f}}_j)$ and scale $\text{MAD}_j(\boldsymbol{v}^T\hat{\boldsymbol{f}}_j)$ in each direction are those from the training data.

3. Toy Datasets

To support the rationale behind the proposed KOD method and provide an intuitive understanding of its strengths, we first show detailed results on three synthetic but challenging bivariate datasets. These 'toy' datasets illustrate how different types of projection directions influence outlier detection performance.

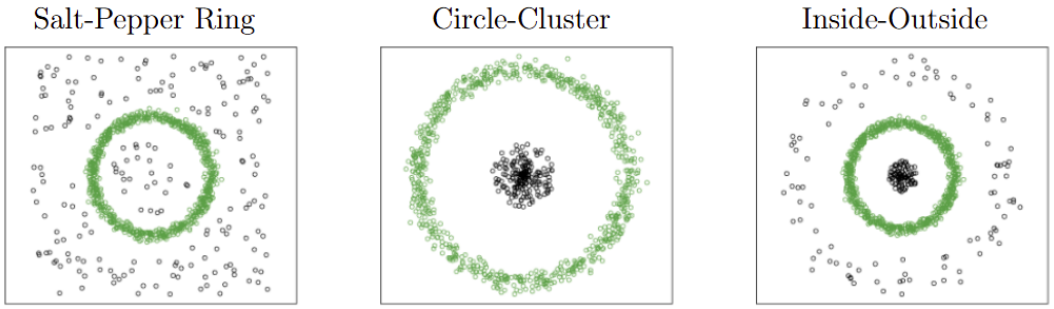

Figure 3: Scatter plots of three toy datasets. Green dots represent regular data points, and black dots represent outliers.

The three datasets are displayed in Figure 3 and referred to as *Salt-Pepper Ring*, *Circle-Cluster*, and *Inside-Outside*.

- **Salt-Pepper Ring:** In this dataset, the regular data points are arranged in a roughly circular pattern, while the outliers are uniformly sprinkled both inside the circle and in the background, creating sparse noise with no coherent structure.
- **Circle-Cluster:** The regular data points again follow a circular pattern, but the outliers now form a cluster near the center. This is harder than the Salt-Pepper Ring, since some methods may interpret the centrally located outliers as a dense cluster of regular points.
- **Inside-Outside:** This dataset was already shown in Section 1. It contains two concentric circular patterns and a cluster in the center. This is the most challenging setup because it combines clustered outliers at the center with rather structured outliers outside the main circle. We put the same number of outliers outside as inside.

Each dataset contains $n = 1000$ observations, with contamination levels set at 20%. In Section 4, we will compare KOD with competing methods on these datasets, and then we will use contamination levels of 5%, 10%, and 20%.

We apply the RBF kernel to each of these datasets. They all have 1,000 cases so the kernel matrix is always 1000×1000, but the centered kernel matrices are of ranks 98,

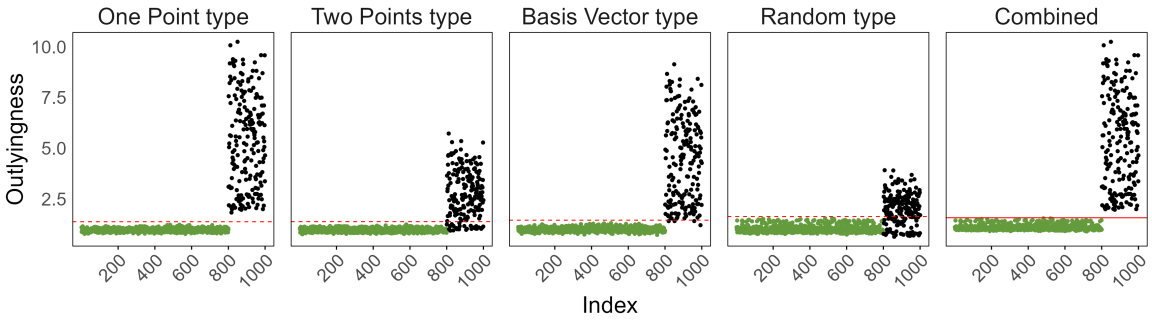

Figure 4: Outlyingness values obtained from each direction type and the combined KO outlyingness on the *Salt-Pepper Ring* dataset with 20% contamination.

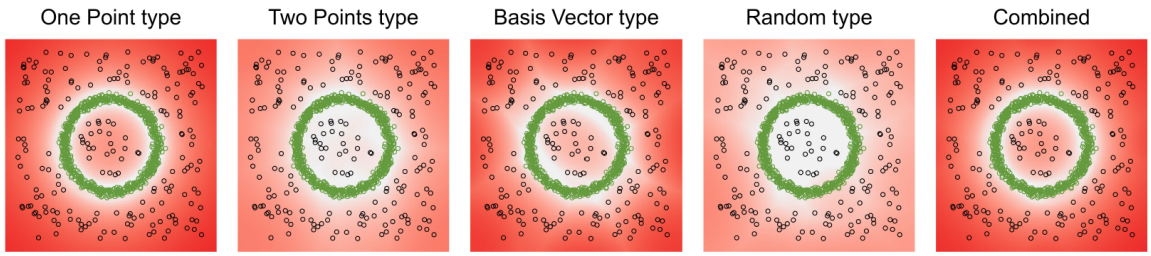

Figure 5: Heatmaps of the outlyingness values obtained from each direction type and the combined KO on the *Salt-Pepper Ring* dataset with 20% contamination.

61 and 79 (we discard eigenvalues below 10^{-12}). Following (6) the retained number of dimensions q is 9, 6, and 8. So even though we have applied the RBF kernel, our computations are performed in a space of dimension much lower than $n - 1 = 999$.

Figures 4 and 5 show some results on the *Salt-Pepper Ring* dataset. The first four panels display the normalized outlyingness values $\mathrm{outl}_{\mathrm{type}} / \mathrm{med}_j(\mathrm{outl}_{\mathrm{type}}(\hat{\boldsymbol{f}}_j))$ for each type of direction. The last panel shows their maximum, which is the final KO. To each panel we have added a horizontal line. The solid red line in the last panel has height c, the cutoff given by (12) for flagging the outliers. The dashed red lines in the other panels represent the hypothetical thresholds that would be obtained if only that single direction type was used for deriving an outlier cutoff.

In Figure 4, the outliers were put at the end for visual support, but in real data they can be anywhere. The Two Points type and the Random type directions detected most of the outliers outside the circle, but were unable to discover the centrally located anomalies. They consider the entire region inside the circle as a regular region. The Basis Vector type roughly captures the shape of the ring, but does not detect all outliers. The One Point type did provide a clear separation between regular data points and outliers, which is also reflected in the combined KO values in the rightmost panel.

The differences between the direction types can be seen more clearly in Figure 5, which displays heatmaps of the SDO computed on a grid as in Figure 1. Again points whose $\mathrm{outl}_{\mathrm{type}}$ is below $\mathrm{med}_j(\mathrm{outl}_{\mathrm{type}}(\hat{\boldsymbol{f}}_j))$ are colored white, and darkening shades of red indicate increasing outlyingness. The Two Points and Random types leave the region inside the circle almost white, so they failed to capture the structure of the regular

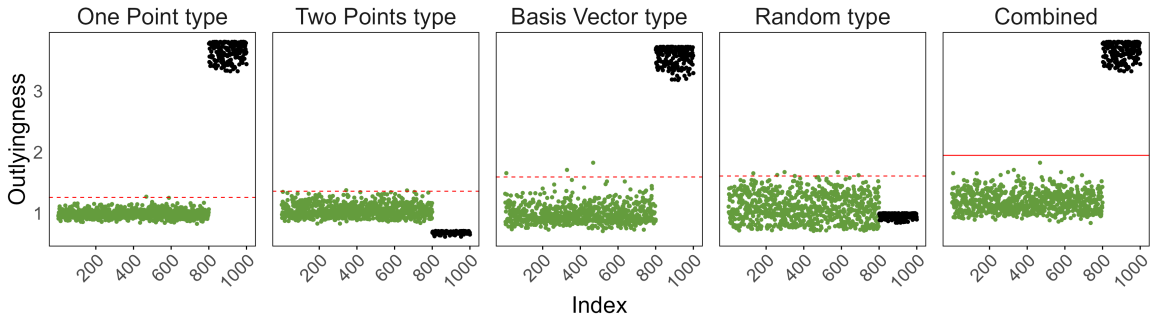

Figure 6: Outlyingness values obtained from each direction type and the combined KO outlyingness on the *Circle-Cluster* dataset with 20% contamination.

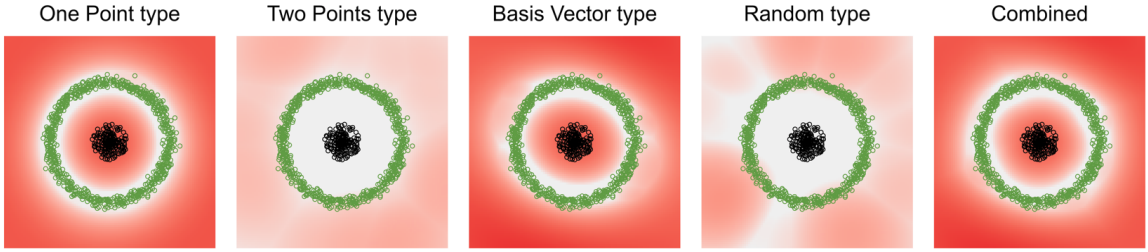

Figure 7: Heatmaps of the outlyingness values obtained from each direction type and the combined KO on the *Circle-Cluster* dataset with 20% contamination.

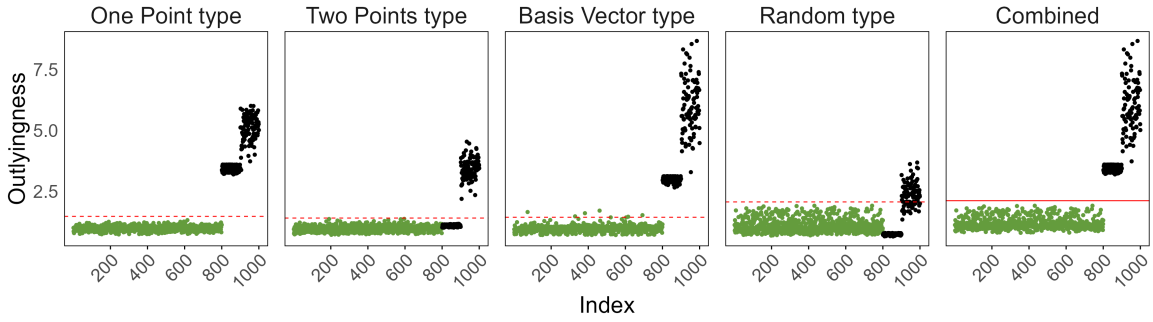

Figure 8: Outlyingness values obtained from each direction type and the combined KO outlyingness on the *Inside-Outside* dataset with 20% contamination.

region. The heatmap of the combined directions in the rightmost panel looks a lot like that of the One Point type, which in this dataset gave the best separation.

Figures 6 and 7 show the results on the *Circle-Cluster* dataset. The Two Points type directions and the Random type directions were unable to discover the centrally located anomalies. They consider the entire region inside the circle as a regular region. The One Point and Basis Vector types did provide a clear separation between regular data points and outliers.

We now revisit the *Inside-Outside* dataset of Figure 1. Both the 'inside' and 'outside' outliers need to be identified, which SDO cannot do in the input space as illustrated in Section 1. In Figures 8 and 9, we see that Two Points and Random struggle to

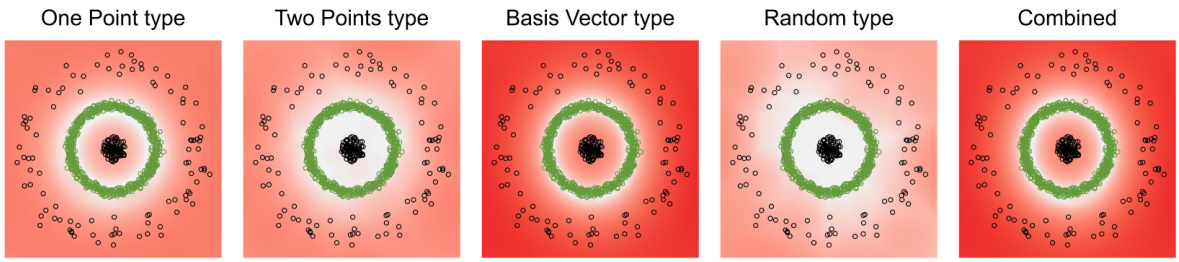

Figure 9: Heatmaps of the outlyingness values obtained from each direction type and the combined KO on the *Inside-Outside* dataset with 20% contamination.

separate the 'inside' outliers from the regular observations, whereas One Point and especially Basis Vector correctly identify the region of the regular points. The cutoff on the combined KO provides a good separation.

In the Appendix, an additional toy dataset is studied.

4. Numerical Experiments

In this section, we compare KOD with several competitors, both on the toy datasets of Section 3 and on four large benchmark datasets. The choice of datasets was guided by the principle that in order to evaluate outlier detection methods we need to know which cases are the outliers, which does not happen very often in real data. We will first describe the competing methods and the evaluation metrics.

4.1. Competing methods

We compare KOD with three other kernel-based methods, as well as three well-known anomaly detection methods that do not use kernels.

Kernel random projection depth (KRPD). Tamamori (2023) first applies the kernel, and then switches to the kernel PCA scores of the data points. Next, 1,000 directions are generated randomly from the uniform distribution on the hypersphere, from which the SDO of (1) is computed. Also a monotone decreasing function of the SDO is considered, given by $D(\boldsymbol{x}_i) := 1/(1+\mathrm{SDO}(\boldsymbol{x}_i))$ and called projection depth. The paper does not provide an unsupervised selection method for the σ of the RBF kernel, or for the number of principal components. To make a fair comparison we use the same number of principal components as in KOD, and consider 8,000 random directions, which is roughly the total number of directions in KOD when $n = 1000$.

Kernel Minimum Regularized Covariance Matrix (KMRCD). The KMRCD method of Schreurs et al. (2021) computes a robust regularized covariance estimator in the feature space. The method uses a fast algorithm that exploits the kernel trick to speed up computations. The hyperparameter α that determines the proportion of data points used to estimate the covariance matrix is set to 0.5 for maximal robustness. The resulting outlier diagnostic is the robust Mahalanobis distance computed in the feature space. We have implemented this method in R.

One-Class Support Vector Machine (OCSVM). OCSVM maps the data into the feature space and searches for a hyperplane that separates them from the origin with maximum margin (Schölkopf et al. 1999). It needs an additional hyperparameter ν that controls the maximum allowed fraction of outliers. We set it to 0.5 for maximal robustness. Moreover, OCSVM provides a decision function whose values indicate the position of each point relative to the separating boundary. These decision values can thus be interpreted as an outlyingness values. Computations are performed with the `svm` function from the R package **e1071** (Meyer et al. 2024).

kNN for outlier detection (KNN). This method and the following ones do not use kernels. KNN ranks the observations based on the distance to their k-th nearest neighbor (Ramaswamy et al. 2000). We use the L_1 distance which is more informative than the Euclidean distance on high dimensional data, and we consider both $\sqrt{n}$ and $\log(n)$ for the hyperparameter k. Computations are performed with the `kNNdist` function from the R package **dbscan** (Hahsler et al. 2025).

Local Outlier Factor (LOF). LOF is a density-based method that assigns to each point an outlyingness value based on how much it deviates from its local neighborhood in terms of density (Breunig et al. 2000). We again used the L_1 distance for finding the nearest neighbors, and considered two alternatives for the size of the neighborhoods. First, we set $k = 20$. Then, as a hyperparameter-free alternative and as advocated in Breunig et al. (2000), we calculated outlyingness values for $k = \{10, 20, 30, 40, 50\}$ and set for each data point its final outlyingness as the maximum among the five values (denoted as LOF Pmax). We use the `lof` function from the R package **dbscan** (Hahsler et al. 2025).

Isolation Forest (IForest). IForest is a tree-based method that recursively partitions the data to isolate outliers (Liu et al. 2008). The anomaly score of a point is related to the number of splittings required to isolate it. The results depend highly on hyperparameter choices, and there are numerous alternatives to choose from. We used the default settings from **scikit-learn** (Pedregosa et al. 2011) and also considered a density-based alternative. The function `isolation.forest` from the R package **isotree** (Cortes 2019) was used.

4.2. Evaluation metrics

In the literature, one often evaluates outlier detection performance by the Area Under the Curve (AUC) of the Receiver Operating Characteristic (ROC). However, Campos et al. (2016) show that it is not optimal for this purpose. Another measure is the Matthews Correlation Coefficient (MCC), which considers all aspects of the confusion matrix: the number of true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN). It is given by

$$\mathrm{MCC} = \frac{\mathrm{TP} \times \mathrm{TN} - \mathrm{FP} \times \mathrm{FN}}{\sqrt{(\mathrm{TP} + \mathrm{FP})(\mathrm{TP} + \mathrm{FN})(\mathrm{TN} + \mathrm{FP})(\mathrm{TN} + \mathrm{FN})}}. \tag{13}$$

However, when comparing a broad range of methods, a practical difficulty arises. Many outlier detection methods in the literature output a ranking or score indicating the degree of outlyingness, but do not provide a cutoff threshold beyond which a data

Table 1: Comparison of outlier detection methods applied to the three toy datasets, with varying contamination percentage. The entries are the Precision at N (P@N) of (14), averaged over 10 replications of each dataset.

	Salt-Pepper Ring			**Circle-Cluster**			**Inside-Outside**		
	Contamination			Contamination			Contamination		
	5%	10%	20%	5%	10%	20%	5%	10%	20%
KOD	1.00	1.00	0.94	1.00	1.00	1.00	1.00	1.00	1.00
KRPD	0.78	0.77	0.81	0.00	0.00	0.00	0.47	0.50	0.54
KMRCD	0.39	0.46	0.60	0.00	0.00	0.00	0.22	0.34	0.41
OCSVM	0.87	0.86	0.88	0.00	0.00	0.00	0.50	0.50	0.50
KNN $k = \sqrt{n}$	1.00	1.00	1.00	0.83	0.41	0.20	1.00	0.92	0.69
KNN $k = \log(n)$	1.00	1.00	1.00	0.50	0.39	0.23	0.80	0.77	0.70
LOF $k = 20$	1.00	0.97	0.75	0.30	0.52	0.61	0.50	0.66	0.78
LOF Pmax	1.00	1.00	0.98	0.27	0.46	0.66	0.99	0.62	0.77
IForest Default	0.90	0.94	0.96	0.12	0.03	0.00	0.62	0.58	0.50
IForest Density	0.98	0.98	0.99	0.98	0.60	0.14	1.00	0.92	0.58

point is flagged as an outlier. Introducing an arbitrary cutoff for such methods could introduce bias, and lead to an inaccurate assessment of their performance. We will therefore use a different evaluation metric, called *Precision at N* and denoted P@N. If N is the known number of true outliers, then P@N is defined as

$$\text{P@N} = \frac{\text{number of true outliers among the N highest values of the criterion}}{\text{N}}. \quad (14)$$

The P@N measure is particularly well-suited in our situation, as it circumvents the need for a threshold, by focusing on the proportion of true outliers among the N top ranked cases. For a comprehensive discussion on the appropriateness of P@N in outlier detection see Campos et al. (2016).

4.3. Performance comparison on the toy datasets

Applying KOD and the outlier detection methods listed in Section 4.1 to the three toy datasets in Section 3 yielded the P@N results in Table 1. In all three datasets we varied the contamination percentage from 5% to 20%, and the reported P@N is the average over 10 replications of each setting.

We observe that KOD worked well on all three datasets, even for 20% of contamination. The performance of the competing methods was quite variable, ranging from reasonable to poor. On some datasets, certain methods failed entirely. The kNN method with $k = \sqrt{n}$ and the density version of IForest had good performance on *Circle-Cluster* for 5% of contamination, but degraded substantially as the contamination level increased.

The fact that KOD outperformed KRPD illustrates that only using directions generated from the uniform distribution on the hypersphere is not enough. The combination of direction types in KOD makes it more versatile. We know from partial results that

sometimes one direction type does better, and sometimes another. We also observed that some of the competing methods are highly sensitive to hyperparameter choices, and in this unsupervised setting those cannot be tuned by cross validation. Even when hyperparameters are set using prior knowledge about the data, no method consistently produces satisfactory results. This is because the clean data can have many different distributions, and the outliers can take many forms.

4.4. Performance comparison on real datasets

Now we move on to more complex real-world scenarios. To evaluate the effectiveness of the proposed KOD method, we conducted a series of experiments on three well-known benchmark datasets of images. As these datasets contain nonlinear relationships and are high-dimensional, they provide a suitable testing ground. Also, it is easy for the human eye to verify outliers in images.

The MNIST, MNIST-C, and Fashion MNIST datasets contain grayscale images with a resolution of 28×28 pixels, each categorized into 10 distinct classes. The MNIST data (Lecun et al. 1998) consist of handwritten digits from 0 to 9. MNIST-C extends the MNIST dataset by introducing several types of corruption applied to the handwritten digit images, some of which are shown in Figure 10. The Fashion MNIST grayscale images depict various types of clothing, distributed across 10 categories.

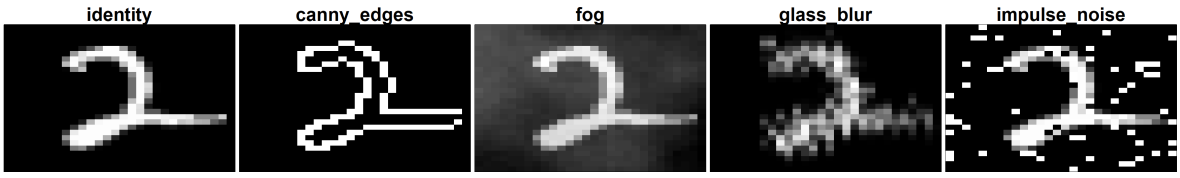

Figure 10: An original image and some corruption types in the MNIST-C dataset.

For MNIST and Fashion MNIST we consider ten different setups. In each setup, one of the ten classes was designated as the regular images. Next, outliers were generated by randomly sampling images from the remaining classes. The MNIST-C data already contained outliers. For each digit and for each corruption type, the original digit images were treated as non-outliers, while the corresponding corrupted versions were considered as outliers. This resulted in a total of 140 different experimental setups (10 digits by 14 corruption types).

In addition to these image datasets we also consider a dataset from Campos et al. (2016), the PageBlocks data. It is about blocks in document pages, and has continuous variables. The regular data correspond to blocks of text, and the blocks with images are considered as outliers.

To investigate the performance under varying outlier percentages, we considered three contamination rates: 5%, 10% and 20%. In all experiments we set the sample size n to 1000, and the results were averaged over 5 replications.

An issue with experiments of this type is that for real data there is no absolute 'ground truth'. In the image datasets, we observed that some of the supposedly clean images looked very strange, and some of the methods indeed flagged them as outlying. On the other hand, some of the supposedly contaminated images looked as if they could

belong to the main class, explaining why they were not always flagged. This raises the important issue of mislabeling in classification, which is an active research topic, see e.g. Raymaekers et al. (2022) and the references cited therein.

For some other datasets, such as the PageBlocks data, there is a semantic justification why some cases are outlying, but is it really certain that the variables that have been measured give enough information about the difference between clean and outlying cases? For instance, medical datasets that contain many healthy people and some sick persons may have recorded many variables, but do they really discriminate between healthy and sick? And might not the healthy set contain people that are outlying for other reasons? Moreover, the diagnosis of the study participants may be subject to mislabeling also.

The choice of kernel significantly impacts the performance of kernel-based methods, as it determines the feature space in which outlyingness is computed. We used the RBF kernel for all image datasets, with the parameter σ selected by the median heuristic given in (4). The RBF kernel is a reasonable choice due to its demonstrated success on image datasets. For the PageBlocks dataset, we used the linear kernel, but using the RBF kernel did not substantially change the performance of KOD.

Table 2: Comparison of outlier detection methods applied to four real datasets, with varying contamination percentage. The entries are the averaged P@N, normalized by its highest value per column.

	MNIST			**MNIST-C**			**Fashion MNIST**			**PageBlocks**		
	Contamination			Contamination			Contamination			Contamination		
	5%	10%	20%	5%	10%	20%	5%	10%	20%	5%	10%	20%
KOD	0.80	0.92	0.95	0.80	0.80	0.83	0.80	0.90	0.97	0.60	0.66	0.88
KRPD	0.64	0.80	0.87	0.66	0.66	0.71	0.89	0.92	0.94	0.68	0.73	0.92
KMRCD	0.68	0.92	1.00	0.66	0.66	0.72	0.89	0.95	0.99	0.41	0.55	0.83
OCSVM	0.68	0.83	0.84	0.69	0.69	0.68	1.00	0.98	0.96	0.40	0.43	0.55
KNN $k = \sqrt{n}$	0.78	0.92	0.94	0.78	0.76	0.77	0.96	0.97	0.94	0.73	0.79	0.84
KNN $k = \log n$	0.85	0.95	0.94	0.69	0.73	0.76	0.77	0.73	0.75	0.76	0.80	0.81
LOF $k = 20$	0.97	0.93	0.87	0.59	0.62	0.65	0.48	0.35	0.38	0.82	0.91	0.91
LOF Pmax	1.00	1.00	0.95	0.66	0.68	0.72	0.68	0.54	0.46	1.00	1.00	1.00
IForest Default	0.64	0.78	0.83	0.94	0.94	0.96	0.98	1.00	1.00	0.68	0.73	0.85
IForest Density	0.61	0.75	0.76	1.00	1.00	1.00	0.75	0.78	0.79	0.51	0.60	0.65

Table 2 shows the performance of KOD and the competing methods in terms of P@N, which has been normalized by dividing each P@N entry by the highest P@N in its column. The performance of KOD was fairly reliable. The performance of some other methods was more variable across settings. Note that KOD does not require tuning of hyperparameters, making it easy to apply.

The Appendix contains an additional table in which the performance is measured by the Matthews correlation of (13) instead of P@N. There KOD is only compared to the three other methods that provide cutoff values. The results are qualitatively similar. Also a figure with computation times is provided there.

5. Conclusion

In this paper, we introduced a new anomaly detection method called kernel outlier detection (KOD), designed to address the challenges of outlier detection in high-dimensional settings. Our research was motivated by limitations of existing methods, some of which assume underlying distribution types, or rely on hyperparameters that are hard to tune, or struggle when data exhibit nonlinear structures, or whose computation does not scale well for larger datasets. By combining kernel transformations with a new ensemble of projection directions, KOD provides a flexible and lightweight framework for outlier detection.

From a methodological viewpoint, our work contributes two main enhancements. First, we introduced the Basis Vector type of direction, that is handy in kernel-induced feature spaces. And second, we introduced a new way to combine the information from different direction types.

Our empirical evaluations on both synthetic and real datasets illustrated the effectiveness of KOD. On three small datasets with challenging structures we saw how different projection direction types capture various aspects of the data, with their combination improving outlier detection under diverse geometric structures. We also studied outlier detection on four large benchmark datasets. On the high-dimensional image datasets MNIST, MNIST-C, and Fashion MNIST we saw that KOD performed among the best methods, illustrating its reliability in practical applications.

We end with a cautionary note. In spite of the good performance of KOD in these examples, there are no guarantees. Indeed, no method can produce good results in all situations. This is because the clean data can have many different distributions, and the outliers can take many forms. What constitutes an outlier also depends on the kind of analysis that is being carried out. For instance, a case may be an outlier in a linear fit but not in a quadratic fit or a regression tree. In that sense anomaly detection methods are only a first step, that should be followed by interpreting its results.

Supplementary Material

The R code of the proposed method and an example script are in https://wis.kuleuven.be/statdatascience/code/kod_r_code_script.zip .

Acknowledgments

The comments of two reviewers have improved the presentation.

References

Breunig, M. M., Kriegel, H.-P., Ng, R. T., and Sander, J. (2000). LOF: Identifying density-based local outliers. *ACM SIGMOD Record*, 29(2):93–104, DOI: 10.1145/335191.335388.

Campos, G. O., Zimek, A., Sander, J., Campello, R. J. G. B., Micenková, B., Schubert, E., Assent, I., and Houle, M. E. (2016). On the evaluation of unsupervised outlier detection: Measures, datasets, and an empirical study. *Data Mining and Knowledge Discovery*, 30(4):891–927, DOI: 10.1007/s10618-015-0444-8.

Cortes, D. (2019). **isotree**: Isolation-Based Outlier Detection, DOI: 10.32614/CRAN.package.isotree. R package version 0.6.1-4.

Croux, C. and Ruiz-Gazen, A. (2005). High breakdown estimators for principal components: The projection-pursuit approach revisited. *Journal of Multivariate Analysis*, 95(1):206–226, DOI: 10.1016/j.jmva.2004.08.002.

Debruyne, M., Hubert, M., and Van Horebeek, J. (2009). Detecting influential observations in Kernel PCA. *Computational Statistics & Data Analysis*, 54:3007–3019, DOI: 10.1016/j.csda.2009.08.018.

Donoho, D. L. (1982). *Breakdown Properties of Multivariate Location Estimators*. Ph.D. qualifying paper, Harvard University.

Dyckerhoff, R., Mozharovskyi, P., and Nagy, S. (2021). Approximate computation of projection depths. *Computational Statistics & Data Analysis*, 157:107166, DOI: 10.1016/j.csda.2020.107166.

Filzmoser, P., Fritz, H., and Kalcher, K. (2006). **pcaPP**: Robust PCA by Projection Pursuit, DOI: 10.32614/CRAN.package.pcaPP. R package version 2.0-5.

Fritz, H., Filzmoser, P., and Croux, C. (2012). A comparison of algorithms for the multivariate L_1-median. *Computational Statistics*, 27:393–410, DOI: 10.1007/s00180-011-0262-4.

Gretton, A., Borgwardt, K. M., Rasch, M. J., Schölkopf, B., and Smola, A. (2012). A kernel two-sample test. *Journal of Machine Learning Research*, 13(25):723–773, https://jmlr.csail.mit.edu/papers/volume13/gretton12a/gretton12a.pdf.

Hahsler, M., Piekenbrock, M., Arya, S., and Mount, D. (2025). **dbscan**: Density-Based Spatial Clustering of Applications with Noise (DBSCAN) and Related Algorithms, DOI: 10.32614/CRAN.package.dbscan. R package version 1.2.2.

Huber, P. (1964). Robust estimation of a location parameter. *The Annals of Mathematical Statistics*, 35:73–101, DOI: 10.1214/aoms/1177703732.

Hubert, M., Rousseeuw, P. J., and Verboven, S. (2002). A fast robust method for principal components with applications to chemometrics. *Chemometrics and Intelligent Laboratory Systems*, 60:101–111, DOI: 10.1016/S0169-7439(01)00188-5.

Hubert, M., Rousseeuw, P. J., and Vanden Branden, K. (2005). ROBPCA: A new approach to robust principal component analysis. *Technometrics*, 47(1):64–79, DOI: 10.1198/004017004000000563.

Hubert, M. and Van Der Veeken, S. (2008). Outlier detection for skewed data. *Journal of Chemometrics*, 22(3-4):235–246, DOI: 10.1002/cem.1123.

Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86:2278–2324, DOI: 10.1109/5.726791.

Liu, F. T., Ting, K. M., and Zhou, Z.-H. (2008). Isolation Forest. In *2008 Eighth IEEE International Conference on Data Mining*, pages 413–422. DOI: 10.1109/ICDM.2008.17.

Maronna, R. A. and Yohai, V. J. (1995). The behavior of the Stahel-Donoho robust multivariate estimator. *Journal of the American Statistical Association*, 90(429):330–341, DOI: 10.1080/01621459.1995.10476517.

Meyer, D., Dimitriadou, E., Hornik, K., Weingessel, A., and Leisch, F. (2024). **e1071**: Misc Functions of the Department of Statistics, Probability Theory Group, TU Wien. DOI: 10.32614/CRAN.package.e1071. R package version 1.7-16.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). **Scikit-learn**: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830. https://www.jmlr.org/papers/v12/pedregosa11a.html

Pokotylo, O., Mozharovskyi, P., and Dyckerhoff, R. (2019). Depth and depth-based classification with R package **ddalpha**. *Journal of Statistical Software*, 91(5):1–46, DOI: 10.18637/jss.v091.i05.

Ramaswamy, S., Rastogi, R., and Shim, K. (2000). Efficient algorithms for mining outliers from large data sets. *ACM SIGMOD Record*, 29(2):427–438, DOI: 10.1145/335191.335437.

Raymaekers, J. and Rousseeuw, P. (2023). **classmap**: Visualizing Classification Results, DOI: 10.32614/CRAN.package.classmap. R package version 1.2.5.

Raymaekers, J., Rousseeuw, P. J., and Hubert, M. (2022). Class maps for visualizing classification results. *Technometrics*, 64(2):151–165, DOI: 10.1080/00401706.2021.1927849.

Rousseeuw, P. J. and Croux, C. (1993). Alternatives to the median absolute deviation. *Journal of the American Statistical Association*, 88:1273–1283, DOI: 10.1080/01621459.1993.10476408.

Rousseeuw, P. J., Raymaekers, J., and Hubert, M. (2018). A measure of directional outlyingness with applications to image data and video. *Journal of Computational and Graphical Statistics*, 27(2):345–359, DOI: 10.1080/10618600.2017.1366912.

Schölkopf, B., Smola, A., and Müller, K.-R. (1998). Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10(5):1299–1319, DOI: 10.1162/089976698300017467.

Schölkopf, B. and Smola, A. J. (2001). *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. The MIT Press, ISBN: 978-0-262-25693-3.

Schölkopf, B., Williamson, R., Smola, A., Shawe-Taylor, J., and Platt, J. (1999). Support vector method for novelty detection. *Advances in Neural Information Processing Systems*, 12:582–588, https://dl.acm.org/doi/10.5555/3009657.3009740.

Schreurs, J., Vranckx, I., Hubert, M., Suykens, J. A. K., and Rousseeuw, P. J. (2021). Outlier detection in non-elliptical data by kernel MRCD. *Statistics and Computing*, 31(5):66, DOI: 10.1007/s11222-021-10041-7.

Segaert, P., Hubert, M., Rousseeuw, P. J., and Raymaekers, J. (2024). **mrfDepth**: Depth Measures in Multivariate, Regression and Functional Settings. DOI: 10.32614/ CRAN.package.mrfDepth. R package version 1.0.17.

Stahel, W. A. (1981). *Robust Estimation: Infinitesimal Optimality and Covariance Matrix Estimators*. PhD Thesis, ETH, Zürich, Switzerland.

Tamamori, A. (2023). Kernel random projection depth for outlier detection. *2023 Asia Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, pages 317–324, DOI: 10.1109/APSIPAASC58517.2023.10317422.

Velasco-Forero, S. and Angulo, J. (2012). Random projection depth for multivariate mathematical morphology. *IEEE Journal of Selected Topics in Signal Processing*, 6(7):753–763, DOI: 10.1109/JSTSP.2012.2211336.

A. An Additional Dataset

In addition to the 'toy' datasets *Salt-Pepper Ring*, *Circle-Cluster* and *Inside-Outside* we also considered another, the *Moons* data. This dataset consists of two interleaving half moon shapes, as seen in Figure 11. It is widely used to demonstrate the limitations of linear separation and to motivate the use of kernel-based approaches. Here the upper half circle (in green) contains most of the cases, while the lower one (in black) has 10% of the cases. The latter are therefore outlying relative to the majority.

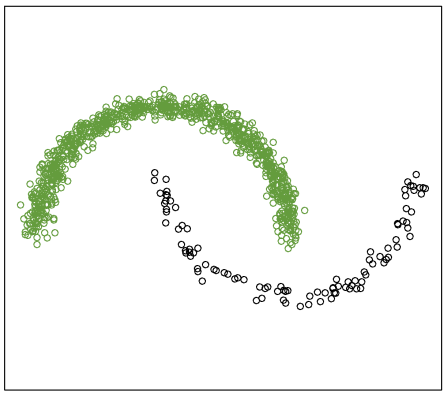

Figure 11: The Moons dataset.

Let us now apply the KOD method, starting with the RBF kernel. Figure 12 shows some results. The first four panels display the normalized outlyingness values $\text{outl}_{\text{type}} / \text{med}_j(\text{outl}_{\text{type}}(\hat{\boldsymbol{f}}_j))$ for each type of direction. The last panel shows their maximum, which is the final KO.

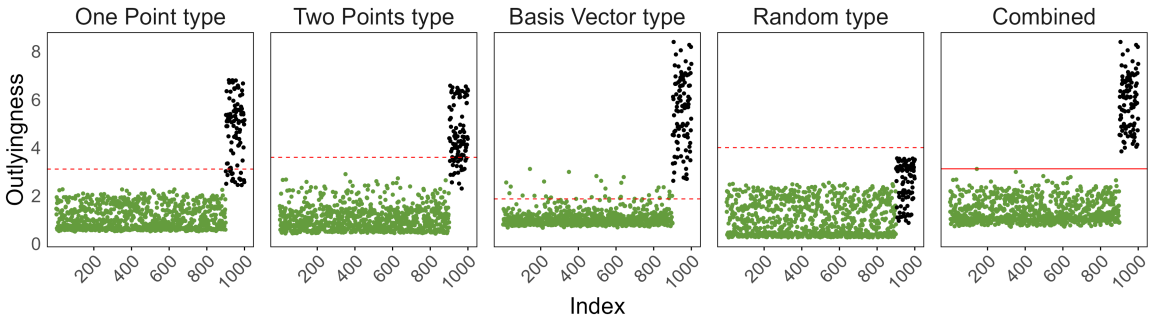

Figure 12: Outlyingness values obtained from each direction type and the combined KO outlyingness on the *Moons* dataset with 10% contamination.

In Figure 12 we see that the Random directions failed to separate the outliers from the regular points. The other three types did better, but none of them in a perfect way. The combined KO values in the last panel do separate the outliers from the inliers, with the cutoff line nicely in between.

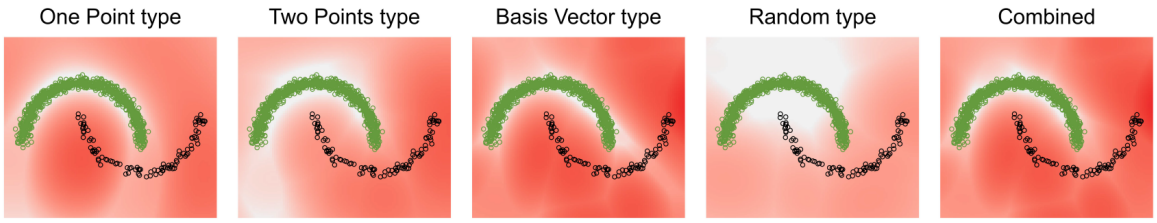

Figure 13: Heatmaps of the outlyingness values obtained from each direction type and the combined KO, on the *Moons* dataset with 10% contamination.

The differences between the direction types can also be seen in the heatmaps of the SDO in Figure 13. The white region in each shows where points would be considered regular, and it varies greatly across direction types. For instance, Random considers part of the lower half circle as regular. Fortunately the heatmap of the combination, in the rightmost panel, is quite accurate.

B. Additional Table with Matthews Correlation

In the main text the comparisons between methods were reported using the P@N measure given by (14). Here is an additional table in which the performance is instead measured by the Matthews correlation of (13). But now we can only make comparisons between KOD and the three other methods that provide cutoff values.

Table 3: Comparison of outlier detection methods applied to four real datasets, with varying contamination percentage. The entries are the averaged MCC.

	MNIST			**MNIST-C**			**Fashion MNIST**			**PageBlocks**		
	Contamination			Contamination			Contamination			Contamination		
	5%	10%	20%	5%	10%	20%	5%	10%	20%	5%	10%	20%
KOD	0.44	0.45	0.33	0.46	0.47	0.41	0.42	0.44	0.41	0.26	0.34	0.37
KRPD	0.31	0.33	0.20	0.36	0.35	0.24	0.43	0.41	0.21	0.27	0.36	0.44
KMRCD	0.31	0.34	0.28	0.35	0.35	0.34	0.35	0.37	0.30	0.20	0.33	0.27
OCSVM	0.21	0.28	0.34	0.10	0.14	0.19	0.20	0.28	0.38	-0.02	-0.04	-0.07

Table 3 shows the MCC results for the four real datasets. The first thing we notice is that the MCC values are quite low overall, because here we did not standardize each column by its maximal value. The datasets are indeed quite challenging due to their complexity, and as mentioned before there is no 'absolute ground truth' for real data. The performance of KOD relative to the other three methods with a built-in decision rule for flagging outliers is qualitatively similar to the situation for the P@N measure. We see that OCSVM struggled a bit with MNIST-C and PageBlocks.

C. Computation Time

We could not really compare the computation times of the methods in the study because some were coded in Python, some in R, and some used R with certain components in compiled C++. Instead we investigated the computation time of the proposed method on the three image datasets. These runs were part of the simulation leading to Table 2 in the main text and Table 3 here. The data dimension was the number of pixels per image, yielding $p = 196$, $p = 784$ and $p = 3136$. We varied the sample size over $n = 250, 500, 1000, 2000$.

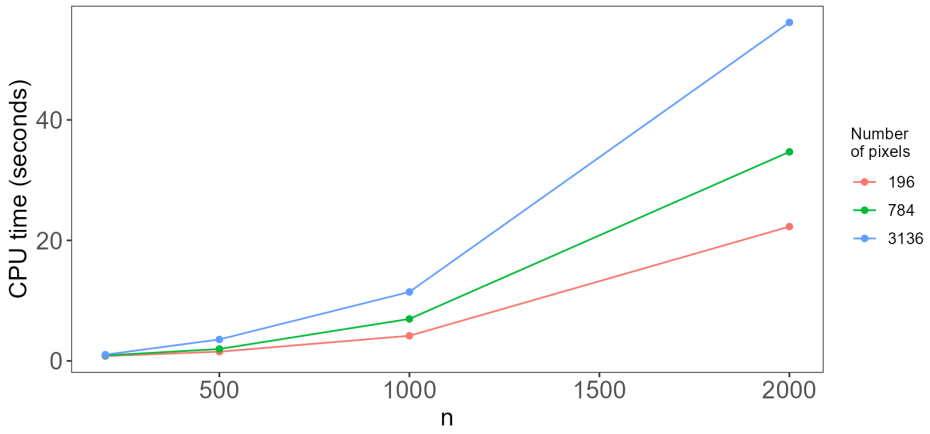

Figure 14: Computation time of KOD. The wall-clock time is shown in function of the sample size n. The three curves correspond to different dimensions. Experiments were run under Windows 11 Enterprise on an HP EliteBook 865 G10 notebook equipped with an AMD Ryzen 7 PRO 7840U CPU (8 cores and 16 threads, 3.3 GHz base) and 32 GB RAM.

Figure 14 shows the wall-clock time in function of the sample size, with separate curves for the dimensions $p = 196$ (red), 784 (green) and 3136 (blue). As expected we see that the computation time goes up with n and p, but it remains feasible. Speedups are possible by rewriting parts of the R code in C++.

Affiliation:

Can Hakan Dağıdır
Section of Statistics and Data Science
Department of Mathematics, KU Leuven
Celestijnenlaan 200B
BE-3001 Leuven, Belgium
E-mail: canhakan.dagidir@kuleuven.be
URL: https://wis.kuleuven.be/statdatascience/robust

Mia Hubert
Section of Statistics and Data Science
Department of Mathematics, KU Leuven
Celestijnenlaan 200B
BE-3001 Leuven, Belgium
E-mail: mia.hubert@kuleuven.be
URL: https://wis.kuleuven.be/statdatascience/robust

Peter J. Rousseeuw
Section of Statistics and Data Science
Department of Mathematics, KU Leuven
Celestijnenlaan 200B
BE-3001 Leuven, Belgium
E-mail: peter.rousseeuw@kuleuven.be
URL: https://wis.kuleuven.be/statdatascience/robust

Journal of Data Science, Statistics, and Visualisation https://jdssv.org/
published by the International Association for Statistical Computing
http://iasc-isi.org/

ISSN 2773-0689
June 2025, Volume V, Issue VIII
doi:10.52933/jdssv.v5i8.152

Submitted: 2025-04-14
Accepted: 2025-06-08