International Journal of Data Science and Analytics
https://doi.org/10.1007/s41060-020-00222-4

APPLICATIONS

Automatic optimization of outlier detection ensembles using a limited number of outlier examples

Niko Reunanen[1] · **Tomi Räty**[2] · **Timo Lintonen**[2]

Received: 7 January 2020 / Accepted: 2 May 2020
© The Author(s) 2020

Abstract
In data analysis, outliers are deviating and unexpected observations. Outlier detection is important, because outliers can contain critical and interesting information. We propose an approach for optimizing outlier detection ensembles using a limited number of outlier examples. In our work, a limited number of outlier examples are defined as from 1 to 10% of the available outliers. The optimized outlier detection ensembles consist of outlier detection algorithms, which provide an outlier score and utilize adjustable parameters. The automatic optimization determines the parameter values, which enhance the discrimination of inliers and outliers. This increases the efficiency of the outlier detection. Outliers are rare by definition, which makes the optimization with a few examples beneficial. Obtaining examples of outliers can be prohibitively challenging, and the outlier examples should be used efficiently.

Keywords Bagging · Outlier detection · Outlier detection ensemble · Semi-supervised outlier detection

1 Introduction

Outlier detection is an important form of data analysis [16]. An outlier is an unexpected data observation that does not match the existing data or assumptions of how the observations are generated [31]. Outliers deviate significantly from the expectations [29]. Normal and expected data observations are called inliers. An outlier can entail interesting information. It consists of unusual, unexpected and new information in comparison with inliers [14]. Other names for outliers include fault [22], intrusion [25,85] and anomaly [48]. Outlier detection has been successfully applied in different fields [8,15,24–26,28,36,38,62,68,80,81].

We propose an approach for optimizing outlier detection ensembles by automatically adjusting the parameters of the combined outlier detection algorithms using a limited number of outlier examples. The outlier detection algorithms are called detectors [43]. An outlier detection ensemble is a combination of detectors; see Sect. 2.1 and [2,3,89] for more information. In the context of our work, a limited number of outlier examples range from a single example to 10% of the available outliers for experiments. The optimization improves the efficiency of the outlier detection, which is empirically validated in Sect. 4.3. The optimization method is introduced in detail in Sect. 3. Section 2 defines the outlier detection algorithms and outlier detection ensembles in detail. Section 5 surveys the related work. Section 6 discusses about the acquired results and concludes this article. The optimization is suitable for a combination of detectors, which (1) provide scores as the magnitude of an observation being an outlier and (2) utilize adjustable parameters. See Sect. 2.1 and [69] for more information on the outlier scores. Additionally, the detectors are required to detect outliers in the given data. If the combination of detectors does not detect outliers in the given data, even with suitable parameter values, then the optimization will not benefit the outlier detection ensemble.

To further improve the efficiency of the outlier detection, we sample a subset of the available features in the analyzed data [6]. A feature is a single dimension of the analyzed data; see the beginning of Sect. 2 for details. The existing work in [43,49,51] samples a random subset of the available features for outlier detection. This approach is called feature

✉ Timo Lintonen
timo.lintonen@vtt.fi

Niko Reunanen
niko.reunanen@hellon.com

Tomi Räty
tomi.raty@vtt.fi

[1] Hellon Oy, Pursimiehenkatu 26 C, 00150 Helsinki, Finland

[2] VTT Technical Research Centre of Finland, Kaitoväylä 1, 90571 Oulu, Finland

Published online: 08 June 2020

bagging; see [43] for more details. Every feature has an equal probability of being selected.

Unlike the previous work in [72,86,87], our proposed approach optimizes the parameters of the detectors using outlier examples. The works in [72,86,87] do not modify the parameters of the detectors. Instead, they propose new algorithms, which utilize the individual outlier examples. The utilization of a limited number of outlier examples has been studied in [87] (10% of the available outliers). Our experiments in Sect. 4 use from a single example to 10% outlier examples. Therefore, our approach uses a smaller number of outlier examples than the existing work.

We combine the following two outlier detection algorithms: k-nearest neighbor (KNN) and local outlier factor (LOF). These outlier detection algorithms are well established and commonly utilized in outlier detection ensembles [12,42,43,49,69]. KNN and LOF are presented in Sects. 2.4 and 2.5. The selection of KNN and LOF is motivated by the results in [69], which show that KNN and LOF can be combined successfully.

2 Outlier detection ensembles and algorithms

Let $\mathbf{X} \in \mathbb{R}^{n \times d}$ be a matrix with n rows and d columns of real numbers ($\mathbf{X}_{ij} \in \mathbb{R}$). The matrix $\mathbf{X}$ represents a static dataset that contains the data for the outlier analysis. The d columns are called features. The n rows are called data points or data observations. Vector $\mathbf{x}_i \in \mathbb{R}^d$ is a data point, which is a row in $\mathbf{X}$. The matrix $\mathbf{X}$ consists of n data points $\mathbf{X} = \{\mathbf{x}_1^{\mathrm{T}}, \mathbf{x}_2^{\mathrm{T}}, \ldots, \mathbf{x}_n^{\mathrm{T}}\}$. Outlier detection algorithms attempt to detect outliers in dataset $\mathbf{X}$. The feature space is a vector space defined by the given features, which measure the properties of the inspected phenomenon. Inliers are located in subsets of the feature space. These subsets are known as normal regions [16,70]. Therefore, inliers are data points (vectors) in the normal regions. An outlier is a data observation $\mathbf{x}_i \in \mathbf{X}$ that does not belong in the normal region. The following subsection defines an outlier detection ensemble in detail.

2.1 Outlier detection ensemble

An outlier detection ensemble combines multiple detectors for accurate outlier detection. The combination of algorithms can reduce bias and variance of the ensemble [10,73]. Bias is the prediction error resulting from the training data of a model, and variance is the prediction error related to unobserved data. Small bias indicates that the model has learned the training data well, because it can predict the training data with small error. Small variance indicates that the model can generalize to different data because it can predict unobserved data with small error. Unfortunately, a low bias causes a high variance and vice versa. This problem is called the bias–variance dilemma [73] or bias–variance trade-off [10]. See [3] for a detailed study of the bias–variance dilemma in the context of outlier detection ensembles.

The scope of our work is in outlier detection ensembles of detectors, which measure an outlier score for the data points. The scoring of outliers is utilized by many of the existing outlier detection algorithms [14,34,57,82]. Outlier score is a quantified measure, which indicates the likeliness of a data point $\mathbf{x}_i$ being an outlier. Without loss of generality, a higher score implies a more likely outlier detection. An outlier is a data point $\mathbf{x}_i$ that has the corresponding score above a threshold value T.

Let K denote the number of detectors, which are the outlier detection algorithms in an outlier detection ensemble. The K detectors operate independently of each other. The detectors calculate an outlier score s_{ij}, in which $i \in 1, 2, \ldots, n$ and $j \in 1, 2, \ldots, K$, for each n of data points $\mathbf{x}_i$ in a dataset $\mathbf{X}$. The subscript j in s_{ij} denotes the score of the jth detector of the outlier detection ensemble, and the subscript i refers to the data point $\mathbf{x}_i$. A detector is mathematically defined as a function $g_j(\mathbf{x}_i) = s_{ij}$, which returns nonnegative real value (outlier score). The function is formally defined as $g_j : \mathbb{R}^d \rightarrow \mathbb{R}^+$ and $g_j(\mathbf{x}_i) = s_{ij}$.

To form a single outlier score for a data point $\mathbf{x}_i$, the outlier scores of the K individual detectors $g_j(\mathbf{x}_i)$ are weighted and summed as follows:

$$g(\mathbf{x}_i) = \sum_{j=1}^{K} w_j g_j(\mathbf{x}_i) = \sum_{j=1}^{K} w_j s_{ij}. \tag{1}$$

The values $w_1, \ldots w_K$ in Eq. (1) are weights assigned to each detector. The weights are discussed further in Sect. 3.2 and Eq. (13). The effect of the weights on the performance is discussed in Sect. 4.4. There exist other options in the literature for the ensemble combination function such as maximum value and average value [3]. However, no consensus exists on the choice of the best method to combine the scores [89]. Therefore, we utilize the summation as the ensemble combination function for the outlier scores [43]. We also normalize the analyzed data to the range [0, 1] on each feature. This alleviates the outlier detection, because the features have similar ranges of values and no feature dominates the rest in scale. The normalization of data is also presented in [43] to be a common step in the process of combining results of detectors in an outlier detection ensemble.

The outlier scores of KNN and LOF do not have a maximum value. This means that it is hard to define the magnitude of a high score and a low score [33]. The scores vary in their scale and range [42]. Even subsets of same data can result in different scores [41]. The outlier scores of a detector may

vary for the same data point $\mathbf{x}_i$ when different subsets of data are used. Therefore, it is very hard to compare the scores between different algorithms and datasets [42]. See [42] for a detailed discussion on how to normalize the outlier scores. To alleviate the scale problem, we utilize linear scaling (similarly to [88]) to normalize the outlier score of a data point between zero and one as follows:

$$g_j^{\text{norm}}(\mathbf{x}_i) = \frac{g_j(\mathbf{x}_i) - g_j(\mathbf{x}_{\min})}{g_j(\mathbf{x}_{\max}) - g_j(\mathbf{x}_{\min})} \quad , \tag{2}$$

where $\mathbf{x}_i$ is a data point in the analyzed dataset ($\mathbf{x}_i \in \mathbf{X}$), $g_j(\mathbf{x}_i)$ is the score of the jth detector for a data point $\mathbf{x}_i$, $g_j(\mathbf{x}_{\max})$ is the maximum score of the detector in the analyzed dataset $\mathbf{X}$ and $g_j(\mathbf{x}_{\min})$ is the minimum score of the detector in the analyzed dataset $\mathbf{X}$.

The detectors have to be accurate and provide diverse results to benefit from their combination [69,88]. First, the detectors have to provide results that are more accurate than random classification. If the detectors assign outliers randomly, then the outlier detection ensemble also provides random results. Second, the detectors have to provide results that are not identical. It is not meaningful to combine multiple instances of identical results, because no new information is gained. Therefore, the correlation should be low between the results. The following subsection introduces a method to create diversity in an outlier detection ensemble.

2.2 Bagging

Bagging (bootstrap aggregating) is an ensemble method to reduce the variance of the results by inducing diversity [13]. In bagging, multiple subsets of the data are sampled randomly with replacements and a model is trained using the sampled subsets. The results of the models are combined (e.g., average or majority voting, see [3,88]) to provide more accurate results. For clarity, we will use the term data bagging when the sample is drawn from the available observations. Data bagging is used in [49].

Another way to utilize bagging in outlier detection is to draw a sample from the available features. This approach is called feature bagging. Feature bagging has been successfully utilized to build outlier detection ensembles in [43]. However, in feature bagging, the d features are randomly sampled without replacements, because multiple copies of the same features do not provide new information. The feature bagging in [43] samples from $d/2$ to $d-1$ features for every detector. Our work applies both data bagging (in Algorithm 2) and feature bagging (in Algorithm 1) to create diversity between the detectors. The following subsection introduces outlier detection algorithms in detail.

2.3 Outlier detection algorithms

Algorithms for outlier detection classify the data points in a dataset $\mathbf{X}$ as inliers and outliers. The algorithms are typically categorized by how they detect outliers as listed below. Many outlier publications [14,57,69,82] define the following categories:

Distribution-based outlier detection [17,47,58,60] models the normal region in feature space as a region of high probability. Data observations with a low probability in the probability distribution are assumed to be outliers.

Distance-based outlier detection [5,82] determines the outlier status of a data observation using distances. Data observations that have a high distance to other data observations are outliers.

Density-based outlier detection [14,57] defines outliers as data observations, which are located in regions with low density in feature space.

Clustering-based outlier detection [11,34,45] commences outlier detection by clustering dataset $\mathbf{X}$. Outliers are data points within deviating clusters or the data points, which deviate relative to the formed clusters.

We combine two algorithms: KNN and LOF. We do not combine distribution-based detectors, because distribution-based detectors assume a distribution [34]. In addition, distance-based and density-based detectors are known to perform better than distribution-based and clustering-based detectors on high-dimensional datasets [57]. This combination of KNN and LOF is utilized also in [3] in outlier detection ensembles.

KNN and LOF are chosen, because they are well-established algorithms for outlier detection in the literature [12,42,43,49,69]. They also represent a different category of outlier detection algorithms in which KNN is a distance-based algorithm and LOF is a density-based algorithm. This demonstrates how a combination of different outlier detection algorithms can be optimized to detect outliers. The following subsections introduce KNN and LOF in detail.

2.4 k-nearest neighbors for outlier detection

k-nearest neighbors (KNN) is a distance-based algorithm for outlier detection [61,76]. Outliers are defined as data points, which are distant to the neighboring data points. The outliers are data points in isolated, or sparsely populated, regions in the feature space. The degree of a data point being an outlier is measured by its location in a local neighborhood. The local neighborhood of a data point $\mathbf{x}_i$ is a k-neighborhood, which is defined as the k nearest data points for a data point $\mathbf{x}_i$. The resulting set of k nearest data points for $\mathbf{x}_i$ is denoted as $\mathcal{N}(\mathbf{x}_i, k)$ by Zhang et al. [82]. The members of $\mathcal{N}(\mathbf{x}_i, k)$ are called local neighbors of $\mathbf{x}_i$. The distances of $\mathbf{x}_i$ to its neighbors ($\mathcal{N}(\mathbf{x}_i, k)$) are combined by using a combination

function (KNN combination function). Typical choices are the maximum value and average value. Let T denote a distance threshold that discriminates outliers from inliers. An outlier is an observation that has a combined distance greater than the distance threshold.

The distance between two data points $\mathbf{x}_i$ and $\mathbf{x}_j$ is measured using a distance metric $D(\mathbf{x}_i, \mathbf{x}_j)$. The distance metric quantifies the difference of the values of two data points (vectors). See [39] for a detailed description of distance metrics and vector spaces. We utilize the following distance metrics (see Sect. 3):

$$\text{Manhattan distance:} \quad D(\mathbf{x}_i, \mathbf{x}_j) = \sum_{k=1}^{d} |x_{ik} - x_{jk}| \ , \tag{3}$$

$$\text{Euclidean distance:} \quad D(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{\sum_{k=1}^{d} (x_{ik} - x_{jk})^2} \ , \tag{4}$$

$$\text{Chebyshev distance:} \quad D(\mathbf{x}_i, \mathbf{x}_j) = \max(|x_{ik} - x_{jk}|) \forall_k \ , \tag{5}$$

$$\text{Cosine distance:} \quad D(\mathbf{x}_i, \mathbf{x}_j) = 1 - \frac{\mathbf{x}_i \cdot \mathbf{x}_j}{||\mathbf{x}_i|| \ ||\mathbf{x}_j||} \ , \tag{6}$$

$$\text{Correlation distance:} \quad D(\mathbf{x}_i, \mathbf{x}_j) = 1 - \frac{(\mathbf{x}_i - \bar{\mathbf{x}_i})^{\mathrm{T}} (\mathbf{x}_j - \bar{\mathbf{x}_j})}{||\mathbf{x}_i|| \ ||\mathbf{x}_j||} \ , \tag{7}$$

$$\text{Canberra distance:} \quad D(\mathbf{x}_i, \mathbf{x}_j) = \frac{1}{d} \sum_{k=1}^{d} \frac{|x_{ik} - x_{jk}|}{x_{ik} + x_{ik}}. \tag{8}$$

In Eqs. (3)–(8), the symbol $||.||$ denotes the norm of a vector, $\bar{\mathbf{x}}$ denotes the mean value of a vector $\mathbf{x}$ and $\mathbf{x}_{ik}$ denotes the kth feature of a data point $\mathbf{x}_i$.

2.5 Local outlier factor

Local outlier factor (LOF) is a well-established outlier detection algorithm [14]. LOF estimates the density $p(\mathbf{x})$ of each observation and classifies the observations in low the density neighborhoods as outliers. Let $D_k^{\mathbf{x}_j}$ be the distance of $\mathbf{x}_j$ to its kth nearest neighbor in $\mathcal{N}(\mathbf{x}_j, k)$ and $D(\mathbf{x}_i,\mathbf{x}_j)$ the distance from $\mathbf{x}_i$ to $\mathbf{x}_j$. The distance $D_k^{\mathbf{x}_j}$ is used to calculate reachability distance $reachdist_k(\mathbf{x}_i,\mathbf{x}_j)$ between points $\mathbf{x}_i$ and $\mathbf{x}_j$. The reachability distance is defined as the maximum between $D_k^{\mathbf{x}_j}$ and $D(\mathbf{x}_i,\mathbf{x}_j)$. The reachability distance is at least $D_k^{\mathbf{x}_j}$, and it is greater than $D_k^{\mathbf{x}_j}$ if $\mathbf{x}_i$ is not a local neighbor of $\mathbf{x}_j$. Formally, the reachability distance is defined as

$$reachdist_k(\mathbf{x}_i, \mathbf{x}_j) = \max(D_k^{\mathbf{x}_j}, D(\mathbf{x}_i, \mathbf{x}_j)) \ . \tag{9}$$

Let $avgreach(\mathbf{x}_i)$ be the average reachability distance between $\mathbf{x}_i$ and all the data points $\mathbf{x}_j$ in the k-neighborhood $\mathbf{x}_j \in \mathcal{N}(\mathbf{x}_i, k)$ of $\mathbf{x}_i$. The $avgreach(\mathbf{x}_i)$ is defined as

$$avgreach(\mathbf{x}_i) = \frac{\sum_{\mathbf{x}_j \in \mathcal{N}(\mathbf{x}_i,k)} reachdist_k(\mathbf{x}_i, \mathbf{x}_j)}{k} \ . \tag{10}$$

The $avgreach(\mathbf{x}_i)$ is used to calculate local reachability density $lrd_k(\mathbf{x}_i)$ for data point $\mathbf{x}_i$. The local reachability density is the inverse of $avgreach(\mathbf{x}_i)$. Local reachability density is defined as

$$lrd_k(\mathbf{x}_i) = \frac{1}{avgreach(\mathbf{x}_i)} = \frac{k}{\sum_{\mathbf{x}_j \in \mathcal{N}(\mathbf{x}_i,k)} reachdist_k(\mathbf{x}_i, \mathbf{x}_j)} \ . \tag{11}$$

Finally, the LOF score, $\mathrm{LOF}_k(\mathbf{x}_i)$, is computed using the local reachability densities $lrd_k(\mathbf{x}_i)$. The LOF score of a data point $\mathbf{x}_i$ is defined as the ratio between the average lrd of the neighborhood $\mathcal{N}(\mathbf{x}_i, k)$ and the $lrd_k(\mathbf{x}_i)$ as follows:

$$\mathrm{LOF}_k(\mathbf{x}_i) = \frac{\sum_{\mathbf{x}_j \in \mathcal{N}(\mathbf{x}_i,k)} lrd_k(\mathbf{x}_j)}{lrd_k(\mathbf{x}_i) * k} \ . \tag{12}$$

The LOF_k can be interpreted as a degree of measure on how packed a given data observation is in a locally reachable neighborhood [82] without assumptions of the distribution of all the data. A high $\mathrm{LOF}_k(\mathbf{x}_i)$ score means that $\mathbf{x}_i$ has a deviating density compared to its neighborhood, which indicates that $\mathbf{x}_i$ is an outlier.

3 The proposed optimization approach

In this section, we propose an approach for optimizing the parameters of outlier detection ensembles using outlier examples. Our proposed optimization increases the accuracy of the outlier detection, which is justified in Sect. 4.3. The process of adjusting the parameters is based on a set of previously known and available examples of outliers.

3.1 Optimization of detectors

In Sect. 1, we assumed that each detector in the ensemble utilizes one or more tunable parameters (e.g., the number of neighbors k). For optimal performance, these parameters must be tuned optimally. We propose a surrogate cost function for the selection of the parameter values in Sect. 3.3. LOF and KNN utilize the following parameters:

- The number of neighbors in a local neighborhood: k
- The selected distance metric: Eqs. (3)–(8)

In addition to these parameters, KNN uses also a combination function. In our setting, the KNN combination function

is chosen from the following combination functions: sum, average, median and maximum function.

Let $\mathbf{X} \in \mathbb{R}^{n \times d}$ be the data to be analyzed and $\mathbf{X}_o \in \mathbb{R}^{h \times d}$ be a set of h known outlier examples. We assume that the number of analyzed data points is greater than the number of outlier examples ($n > h$), because outliers are rare. The data points $\mathbf{x}_i$ in $\mathbf{X}$ do not have labels, and they consist of both inliers and outliers. The outliers in the dataset $\mathbf{X}$ are not known in advance. Therefore, the outliers in the dataset $\mathbf{X}$ are called hidden outliers and we say that the dataset $\mathbf{X}$ is contaminated with the hidden outliers. All of the data points in $\mathbf{X}_o$ are outliers. Let $\mathbf{X}_+ \in \mathbb{R}^{(n+h) \times d}$ be a matrix that results in concatenating the rows of the matrices $\mathbf{X}$ and $\mathbf{X}_o$ as $\mathbf{X} \cup \mathbf{X}_o = \mathbf{X}_+ \in \mathbb{R}^{(n+h) \times d}$. The matrix $\mathbf{X}_+$ consists of the n unlabeled data points and the h labeled outlier examples.

As established in Sect. 2, KNN and LOF provide outlier scores for the data points in a dataset. The outlier scores discriminate the outliers and inliers so that outliers have high scores and inliers have low scores. Therefore, a contrast between the inlier scores and outlier scores helps the separation of the inliers and outliers [42]. However, the outlier scores do not typically provide a good separation of the scores between the inliers and outliers [42]. Therefore, the goal of our proposed optimization is to build a contrast between the inlier and outlier scores by minimizing the inlier scores and maximizing the outlier scores. The underlying assumption in our work is that a contrast between the inlier and outlier scores makes the outlier detection more efficient. It is easier to use a score threshold (T) to classify data points into inliers and outliers if the scores of inliers and outliers do not resemble each other. One motivation for our approach comes from the field of classification in machine learning. Support vector machine (SVM) is a classifier that defines a maximum margin between the target classes in the utilized feature space [10]. The SVM classifier is accurate and capable to generalize, by utilizing the maximum margin as the target classes are separated with a clear contrast. We call the contrast between the inlier and outlier scores a score margin, which is utilized in the surrogate cost function in Eq. (15).

3.2 Maximization of score margin

The score margin is quantified as a real value. Let $g_j^{\text{norm}}(\mathbf{X})$ denote the normalized scores of the analyzed data of the jth detector. Let $g_j^{\text{norm}}(\mathbf{X}_o)$ denote the normalized scores of the outlier examples by the jth detector. Then, the margin is the difference $g_j^{\text{norm}}(\mathbf{X}_o) - g_j^{\text{norm}}(\mathbf{X})$. To create a contrast between normal data and outliers, the outlier score distributions of normal data and outliers have to have a positive margin. However, the analyzed dataset is contaminated by the hidden outliers, which prevents the exact computation of the margin. Therefore, a robust measurement is required for determining the magnitude of the score margin. The measurement must not be affected by the hidden outliers, because they are not known in advance in the contaminated dataset.

We utilize the median value (MED) of the score distributions, because the median measures the 50-percentile (middle value) of a distribution. The median is robust to the scores of the hidden outliers, because the median has a breakdown point of 50% (see [66] for details). If at least 50% of the data points in a sample are inliers, then the median score is not arbitrary large. Therefore, the impact of the hidden outliers is negligible, because the inlier scores are likely to induce the median value. The median can acquire arbitrary large values if at least 50% of the data points in the contaminated analysis data result in high outlier scores. However, 50% of the data points should not form the set of outliers as this is in stark contrast with the inherent stipulation of outliers being rare by definition. Therefore, the median is a robust measurement for the outlier scores, because it is not significantly affected by the hidden outliers in the contaminated analysis data.

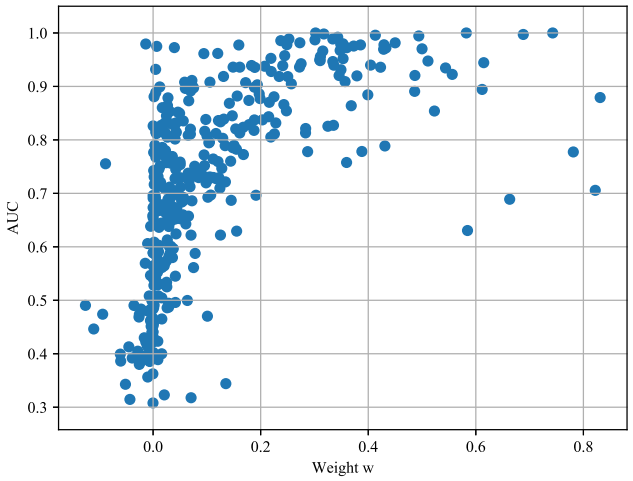
For the jth detector, the optimization finds the parameter values that maximize the distance $\text{MED}(g_j^{\text{norm}}(\mathbf{X}_o)) - \text{MED}(g_j^{\text{norm}}(\mathbf{X}))$ between the medians of the normalized score distributions. By maximizing the score margin, the detectors are more likely to discriminate the inliers from the outliers. However, we formulate the score margin maximization through minimization by minimizing the negative score margin. The following equation defines an objective function to be minimized for the jth detector:

$$-w_j = \text{MED}(g_j^{\text{norm}}(\mathbf{X})) - \text{MED}(g_j^{\text{norm}}(\mathbf{X}_o)) \quad . \tag{13}$$

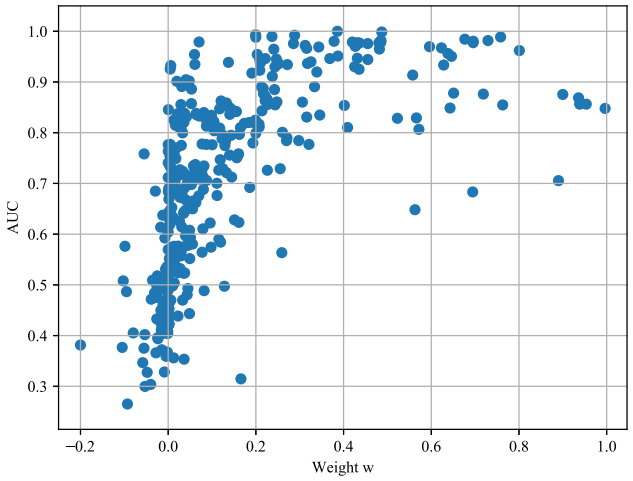
The objective function in Eq. (13) is a fine candidate for a surrogate objective function for the outlier ensemble optimization. This claim is supported by the correlation of the value w_j (which is the score margin) in Eq. (13) and the detector performance in Fig. 1.

Figure 1 illustrates the correlation between the AUC score (see Sect. 4.1) and the weight in Eq. (13). The AUC score and the weight are computed on different datasets (summarized in Table 1) with a set of known outliers sampled from all the outliers. The sample size of the known outliers is ten percent of the total amount of the outliers in the dataset. In order to introduce more variability in both AUC scores and the weight values, we utilize a sample of the features of the original data. In addition, the parameters of the detectors (KNN and LOF) are also selected randomly. Figure 1 shows that the larger values of the weight (the score margin) tend to predict larger values of the AUC scores.

The detectors are optimized individually one detector at a time. The optimization finds the parameters for the individual detectors. The optimization is a greedy algorithm from the point of view of the outlier detection ensemble, because the detectors are optimized separately. Greedy algorithms make locally optimal choices in optimization



(a)



(b)

Fig. 1 The relation between AUC score and the weight in Eq. (13) on multiple datasets. To induce variability, detectors are executed on a random sample of features and parameters. **a** AUC scores and weights on KNN. **b** AUC scores and weights on LOF

tasks [18]. The selected parameters maximize the distance between the distributions of the inlier and outlier scores. In the existing literature, outlier detection ensembles have been constructed using a greedy algorithm in [69]. However, instead of selecting the algorithms, our greedy algorithm optimizes the parameters of a fixed set of detectors. Our work is the first to optimize the parameters of detectors directly in an outlier detection ensemble.

As noted in [88], the detectors should make a small number of different errors. This allows the detectors as a combined model to mitigate the weaknesses of individual detectors. To impose diversity between the detectors, we update Eq. (13) in the following subsection to also consider the correlation of the scores between the detectors.

3.3 Minimization of correlation between results

The optimization in the previous subsection increases the contrast between the inlier scores and outlier scores. Our greedy optimization attempts to improve the accuracy of the outlier detection of the individual detectors. However, in addition to being accurate, the results have to be diverse. The contrast maximization in Eq. (13) does not guarantee that the results of the detectors are diverse. Therefore, as in [69], we adjust the optimization to minimize the correlation between the results of the detectors. We utilize Pearson correlation (corr) to measure the amount of dependency between two distributions of normalized scores $(g_j^{\text{norm}}(\mathbf{X}), g_l^{\text{norm}}(\mathbf{X}))$ of jth and lth detectors. For two vectors $(\mathbf{x}, \mathbf{y})$, Pearson correlation is defined as:

$$\text{corr}(\mathbf{x}, \mathbf{y}) = \frac{\text{cov}(\mathbf{x}, \mathbf{y})}{\text{std}(\mathbf{x})\text{std}(\mathbf{y})} \quad , \tag{14}$$

where cov is the covariance and std is the standard deviation. See [50] for more details. The amount of correlation between the detector scores is averaged, and the average correlation is utilized to weight the margin of the current solution: A high correlation penalizes the margin. Therefore, the final form of the objective function for the optimization of the jth detector is as follows:

$$f_j = (\text{MED}(g_j^{\text{norm}}(\mathbf{X})) - \text{MED}(g_j^{\text{norm}}(\mathbf{X}_o))) * \left(1 - \frac{1}{K}\sum_{l=0}^{j-1} |\text{corr}(g_j^{\text{norm}}(\mathbf{X}), g_l^{\text{norm}}(\mathbf{X}))|\right), \tag{15}$$

where $|.|$ is the absolute value, corr is the correlation function, K is the number of detectors, MED is the median function, g_j^{norm} is the normalized score function of jth detector, $\mathbf{X}$ is the contaminated dataset and $\mathbf{X}_o$ are the outlier examples. The objective function in Eq. (15) utilizes outlier examples $\mathbf{X}_o$ to (1) create a contrast between inliers and outliers using accurate detectors and to (2) acquire diverse results from the detectors.

3.4 Combining outlier scores with logistic regression

After the optimizations in Eq. (15), we have a set of diverse detectors. The results of these detectors can be combined using the weighted summation in Eq. (1). We will call this approach optimized outlier ensemble (OOE). However, Micenková et al. [49] propose to use supervised methods to learn the outliers from the outputs of the detectors. Micenková et al. [49] utilize a set of unoptimized detectors.

We will show in Sect. 4 that the performance of outlier learning can be further enhanced with our diverse set of optimized detectors.

In this subsection, we summarize the algorithm dubbed *proposed*+ in [49]. The supervised method that *proposed*+ uses is logistic regression with l_1-penalty. Micenková et al. [49] propose l_1-regularization to overcome the curse of dimensionality. In the training phase, *proposed*+ uses the outlier examples $\mathbf{X}_o$ as the positive class ($C = 1$), while the unlabeled data $\mathbf{X}$ constitute the negative class ($C = 0$). The hidden outliers in the unlabeled data are labeled incorrectly, but since outliers are rare by definition [16], they are considered to be noise during the training.

Proposed+ utilizes the base detector outputs as a set of additional features to the original data. Let the vector $\mathbf{z}_i = (\mathbf{x}_i, g_1(\mathbf{x}_i), \ldots, g_K(\mathbf{x}_i))$ be a data point that is extended with its outlier scores for all $i = 1, \ldots, n$. Let the variables C_i be indicators whether a given data point $\mathbf{x}_i$ is a known outlier. Logistic regression assumes that the probability of a data point $\mathbf{x}_i$ being a known outlier is

$$p(C_i = 1|\mathbf{z_i}; \beta_0, \boldsymbol{\beta}) = \frac{1}{1 + \exp(-\beta_0 - \mathbf{z}^{\mathrm{T}}\boldsymbol{\beta})}. \tag{16}$$

The model parameters β_0 and $\boldsymbol{\beta}$ in Eq. (16) are found by minimizing a loss function. In the case of l_1-regularization, this loss function is

$$J(\beta_0, \boldsymbol{\beta}) = -\sum_{i=1}^{n}[C_i(\beta_0 + \mathbf{z}_i{}^{\mathrm{T}}\boldsymbol{\beta}) - \log(1 + e^{\beta_0 + \mathbf{z}_i{}^{\mathrm{T}}\boldsymbol{\beta}})] + \lambda \sum_{j=0}^{d+K} |\beta_j|. \tag{17}$$

The parameter λ in Eq. (17) controls the effect of regularization. We will select the parameter λ by minimizing Akaike information criterion (AIC) [75]. In the case of logistic regression, both AIC and cross-validation are similar in terms of performance, but AIC is less computationally demanding [52].

As a supervised method, imbalanced classes may deteriorate the accuracy of logistic regression [7,35]. For this reason, Micenková et al. [49] propose data bagging. The unlabeled class $\mathbf{X}$ is down-sampled to include as many data points as there are known outlier examples ($|\mathbf{X}_o| = h$). Both classes are sampled with replacement. In the experiments, we fixed the minimum sample size for the bagged datasets to be 20. Smaller sample sizes conflict with the well-known rule-of-thumb that the number of observations in the minority class (in this case, the known outliers $|\mathbf{X}_o|$) should be at least ten per variable [59]. In addition, smaller sample sizes resulted in each β_j to equal zero and the model giving each data point a probability of 0.5 of being an outlier. Such a model is incapable of differentiating between the data points and does not add any value to the ensemble. Finally, *proposed*+ combines the outputs of the logistic regression models by averaging the outputs.

In Sect. 4.4, we use *proposed*+ with the two modifications (finding the parameter λ by minimizing AIC and setting a minimum sample size of 20) as discussed earlier in this section. We call this modified *proposed*+ algorithm logistic regression with transformed features (LOG+).

3.5 Algorithm for optimizing an outlier detection ensemble

We present two strategies for combining the normalized outlier scores $g_j^{\mathrm{norm}}(\mathbf{X})$ for each of the detectors, $j = 1, \ldots, K$. The first strategy (OOE) is the weighted sum of the normalized outlier scores using Eq. (1). The normalized outlier scores and the weights for the detectors are computed according to Algorithm 1. The K detectors are greedily optimized to be accurate [as defined in Eq. (13)] and diverse [as defined in Eq. (15)].

In Algorithm 1, Steps 2 and 3 represent feature bagging. Integer l is picked randomly from the set $\{\lceil d/2 \rceil, \ldots, d-1\}$ with equal probabilities. Then, l features are picked from the original data $\mathbf{X}_+$ without replacement. The optimization loop in Steps 4–7 iterates until a stopping condition is reached. In this article, we use a predefined number of iterations as the stopping condition. The parameters, which are utilized in Algorithm 1, are found by using random search [9].

The second strategy, which we call Hybrid, utilizes logistic regression from Sect. 3.4. This strategy is summarized in Algorithm 2. In step 5, the parameter λ for Logistic regression in Eq. (17) is chosen by minimizing AIC over the parameter λ. Hybrid utilizes four sources of detector variability: different feature sets, different training sets, different classifiers and different parameter choices [20]. These sources of variability ensure that the set of detectors is diverse.

In Step 2 of Algorithm 2, the unlabeled augmented data $\mathbf{Z}$ consist of both inliers and hidden outliers. In the training phase in Step 5, both inliers and hidden outliers are used as the negative class in which the hidden outliers are considered to be noise as discussed in Sect. 3.4. In Steps 3–8, the index j runs from $K + 1$ to $2K$ to differentiate the outputs of Algorithm 1 from the probability estimates in Step 6. In Step 7, the weights for the probability estimates are computed according to Eq. (13). It is not necessary to normalize the probability estimates, because they are within the range [0, 1] by definition. Finally, in Step 9, the outlier scores of Hybrid approach are computed as the weighted sum of the outlier scores from the detectors (outputs of Algorithm 1) and the probability estimates of the logistic regression models (Step 6.)

Earlier in this section, we discussed the optimization loop in Steps 4–7 of Algorithm 1. We recognize that there are more advanced hyperparameter optimization strategies than a simple random search [9]. For example, tree of Parzen estimators and genetic algorithm typically find better solutions than random search when optimizing hyperparameters of convolutional neural networks [30]. In Sect. 4.3, we show that the random search is sufficient for achieving better performance compared to unoptimized approaches. Experimenting with other optimization strategies is left for future work.

4 Experiments

In this section, we examine the performance of our proposed model (OOE and Hybrid) using benchmark datasets from two outlier data repositories (which are presented in Sect. 4.2). To apply our approach in practice, the final composition of the ensembles must be chosen. The computation time increases as the number of detectors (K) increases. To compromise between the computation time and the completeness of the evaluation experiments, the utilized amount of the detectors in both LOF and KNN, respectively, in the experiments is $\{1, 2, 4, 8, 16\}$ (respectively, powers of two). This provides a broad view of the efficiency of the outlier detection ensemble with different numbers of detectors. To reduce the number of possible combinations and computation time, the number of LOF and KNN are kept identical. Therefore, the total number of detectors is $K \in \{2, 4, 8, 16, 32\}$ in the experiments. In the existing work, 50 detectors are utilized in [49], $K \in \{5, 10, 20, 50\}$ in [88], $K \in \{1, 2, \ldots, 25\}$ in [90] and $K \in \{3, 4, 5, 10\}$ in [42]. There are no established number of detectors for evaluation in outlier detection ensembles. Our experiments utilize the powers of two to determine the number of detectors deterministically in the experiments.

As listed in Sect. 3.1, the following parameters are optimized for the individual detectors in an outlier detection ensemble:

- *LOF and KNN* the number of neighbors in a local neighborhood: k [in $\mathcal{N}(\mathbf{x}_i, k)$ as in Sect. 2.4 and Eq. (9)–(12)]
- *LOF and KNN* the selected distance metric [Eq. (3)–(8)]
- *KNN* the KNN combination function of the distances in the local neighborhood: sum, average, median and maximum (Sect. 2.4)

For the optimization, we define the maximum number of neighbors (k) as the square root of the number of data points in an analyzed dataset ($\sqrt{n}$). This heuristic for k is suggested in [19]. The value is rounded to its nearest integer. The minimum number of neighbors is set to ten. Models with less than ten neighbors are susceptible to noise [14]. Let us consider the optimization of the KNN detector for a dataset with 1000 data points. The maximum (minimum) number of neighbors is 32 (10). There are six distance metrics [Eq. (3)–(8)] and four KNN combination functions available. Therefore, in case of $n = 1000$, the optimization is implemented as random sampling from a uniform distribution in which each of the $(32 - 10 + 1) * 6 * 4 = 552$ configurations has an equal probability of being evaluated (since both ends of the range $10, \ldots, 32$ for parameter k are included in the random search). Our experiments utilize 20 iterations of random sampling per detector to provide a trade-off between the quality of the optimization and the computation time. In our experiments, increasing the number of iterations (> 20) did not significantly improve the quality of the optimization. The optimization evaluates a diverse set of detector configurations while restricting the total number of optimization iterations.

4.1 Metrics for evaluating the outlier detection

The following metrics are utilized in the evaluation of the outlier detection: recall (REC), false positive rate (FPR), receiver operating characteristics (ROC) curve and the area under ROC (AUROC). Let TP be the number of detected true positives, FP the number of detected false positives, TN the number of detected true negatives and FN the number of detected false negatives. Recall is defined as $\text{REC} = \text{TP}/(\text{TP} + \text{FN})$, and the false positive rate is defined as $\text{FPR} = \text{FP}/(\text{FP} + \text{TN})$. See [23] for a detailed definition of the evaluation metrics.

The outlier scores (see Sect. 2.1) discriminate inliers from outliers using a threshold (T). Outliers (inliers) are the data points with a score greater (less) than T. A low value of the threshold results in a high value of recall and a high value of false positive rate, and vice versa. Let us consider, for example, using a low value of the threshold. Many of the outliers are detected with various magnitudes of the score. However, the more unusual inliers are incorrectly detected as outliers. Therefore, the threshold T defines a trade-off between the recall and false positive rate.

To study the trade-off and the efficiency of the outlier detection ensembles with different threshold values, we utilize ROC to summarize the resulting pairs of recall and false positive rate. ROC is a graph of the resulting REC (y-axis) and FPR (x-axis) values in which the threshold is varied. The ROC presents a complete view of the efficiency of the outlier detection ensemble. ROC is used to evaluate outlier detection in [1,41,43,44,49,69,72]. See [23] for a detailed tutorial on ROC.

The ROC graphs are quantified as real values between zero and one by computing the area under the ROC curve (AUC). The value of AUC shows the efficiency of an outlier detection ensemble over a range of values of the score threshold. A perfect algorithm acquires $\text{AUC} = 1$ by detecting all of

Algorithm 1 Generate optimized ensemble.

Input: Dataset $\mathbf{X}$, positive example(s) $\mathbf{X}_o$, ensemble size K.
Output: Normalized outlier scores $g_j^{norm}(\mathbf{X}_+^{(j)})$ and weights w_j for $j = 1, \ldots, K$.
1: **for** $j \in \{1, 2, ..., K\}$ **do** ▷ Add K detectors in the outlier detection ensemble
2: Sample a random integer l from the interval $[d/2, d-1]$
3: Sample l features from all features without replacement to construct a dataset $\mathbf{X}_+^{(j)} = \mathbf{X}^{(j)} \cup \mathbf{X}_o^{(j)}$
4: **while** Optimization is not finished **do** ▷ e.g iterations remaining > 0
5: Sample a set of parameters for the j:th detector
6: Compute the loss function value in Eq. (15) with the sampled parameters
7: **end while**
8: Compute and normalize the outlier scores $g_j^{norm}(\mathbf{X}_+^{(j)})$ using j:th detector with parameters that resulted in the lowest loss function value in Step 6
9: Compute the weight w_j in Eq. (13)
10: **end for**

Algorithm 2 Hybrid approach.

Input: Dataset $\mathbf{X}$, positive example(s) $\mathbf{X}_o$, ensemble size K.
Output: Outlier scores $g(\mathbf{X})$ for Hybrid approach.
1: Compute the normalized outlier scores $g_j^{norm}(\mathbf{X}_+^{(j)})$ and the weights w_j for $j = 1, \ldots, K$ using Algorithm 1.
2: Augment the data such that $\mathbf{Z} = \left(\mathbf{X}, g_1^{norm}(\mathbf{X}^{(1)}), \ldots, g_K^{norm}(\mathbf{X}^{(K)})\right)$ and $\mathbf{Z}_o = \left(\mathbf{X}_o, g_1^{norm}(\mathbf{X}_o^{(1)}), \ldots, g_K^{norm}(\mathbf{X}_o^{(K)})\right)$
3: **for** $j \in \{K+1, K+2, ..., 2K\}$ **do**
4: Sample $\min\{10, |\mathbf{Z}_o|\}$ data points from both unlabelled data $\mathbf{Z}$ and known outliers $\mathbf{Z}_o$ with replacement to form a sample $\mathbf{Z}_{(j)+}$.
5: Train logistic regression using l_1-regularization according to Eq. (17) with the sample $\mathbf{Z}_{(j)+}$
6: Compute the probability estimates $p(C = 1|\mathbf{Z}_{(j)+}; \beta_0^{(j)}, \beta^{(j)})$ ▷ Note that the index j is defined $j = K+1, \ldots, 2K$ to make a distinction to the outputs of Algorithm 1
7: Compute the weight w_j in Eq. (13) using probability estimates in the place of the normalized scores
8: **end for**
9: Compute final outlier scores $g(\mathbf{X}) = \sum_{j=1}^{K} w_j g_j^{norm}(\mathbf{X}^{(j)}) + \sum_{j=K+1}^{2K} w_j p(C = 1|\mathbf{Z}; \beta_0^{(j)}, \beta^{(j)})$

the outliers without false positives [23]. The worst possible algorithm acquires AUC $= 0$ by intentionally misclassifying the data points [23]. Notice that AUC $= 0.5$ is acquired by randomly guessing if a data point is an outlier or an inlier [23]. The general efficiency of the algorithms is determined by comparing the resulting values of AUC. The outlier detection ensemble with the highest value of AUC is declared the best performing ensemble.

We use statistical tests to critically examine the results of the experiments. We utilize the paired t test [65], which tests the average difference on paired data. Let a random variable D denote the difference in AUC scores between two different algorithms on the same data. The null hypothesis is that the difference in AUC scores between the two different algorithms is zero ($H_0 : D = 0$). The paired t test assumes that the differences follow the normal distribution ($D \sim N(\mu, \sigma^2)$). Then, the test variable $t := D/\mathrm{SE}(D)$, in which SE denotes the standard error, follows Student's t distribution. If the p value of the test variable t is less than 0.05, then the difference D differs from zero on a statistically significant level. In addition to the paired t test, we utilize two nonparametric tests, bootstrapping [74] and Wilcoxon signed-rank test [65,77], to the difference D. If all the three tests result in a p value less than 0.05, then we conclude that the difference in the performance of the two algorithms is statistically significant.

4.2 Data

The outlier detection ensembles are evaluated using public real-world data, which are commonly utilized in outlier publications [4,12,33,34,41,42,46,51,71,78,82]. Real-world datasets are recorded and aggregated from real-world environments. It is challenging to evaluate outlier detection, because only a few datasets exist with specifically distinguished outliers [40,82]. In the literature, two approaches to acquire annotated outlier data are utilized: either generate data with outliers [4,33,78] or sample imbalanced data from existing datasets [51,82]. We utilize the second option, because many outlier publications sample imbalanced data [33,34,41,42,69,78,79] to validate outlier detection.

For reproducibility, we utilize publicly available outlier dataset repositories. The datasets we use as benchmark datasets are gathered from two repositories, outlier detection datasets (ODDS) [63] and anomaly detection meta-analysis datasets (Oregon) [21]. All the methods (OOE, LOG+ and Hybrid) are ensemble methods. For this reason, we have chosen to limit the size of the datasets to maximum of $n = 20{,}000$ [86] to keep the computation times manageable. Both ODDS and Oregon are collections of outlier datasets with annotations (each data point is classified either inlier or outlier) from multiple domains [21,63].

Table 1 Summary of the datasets

Data	n	d	Outliers	Frequency
Abalone	4177	7	2081	0.498
Annthyroid	7200	6	534	0.074
Arrhythmia	452	274	66	0.146
Breastw	683	9	239	0.350
Cardio	1831	21	176	0.096
Comm and crime	1994	101	993	0.498
Concrete	1030	8	515	0.500
Ecoli	336	7	9	0.027
Fault	1941	27	673	0.347
Gas	13,790	128	3517	0.255
Glass	214	9	9	0.042
Imgseg	2310	18	990	0.429
Ionosphere	351	33	126	0.359
Letter	1600	32	100	0.062
Lympho	148	18	6	0.041
Magic gamma	19,020	10	6688	0.352
Mammography	11,183	6	260	0.023
Mnist	7603	100	700	0.092
Musk	3062	166	97	0.032
Optdigits	5216	64	150	0.029
Pageb	5473	10	560	0.102
Pendigits	6870	16	156	0.023
Pima	768	8	268	0.349
Satellite	6435	36	2036	0.316
Satimage-2	5803	36	71	0.012
Spambase	4601	57	1813	0.394
Speech	3686	400	61	0.017
Thyroid	3772	6	93	0.025
Vertebral	240	6	30	0.125
Vowels	1456	12	50	0.034
Wave	5000	21	1657	0.331
Wbc	378	30	21	0.056
Wine	129	13	10	0.078
Yeast	1484	8	507	0.342

The datasets in Table 1 [21,63] do not separate the outliers into known and unknown outliers, in which the known outliers are exploited in our approach to correctly detect the unknown outliers. To acquire the known outliers, we sample the outlier class randomly. The amount of known outliers affects the performance of the models. For this reason, we generate four sample sizes. The hardest setting has the sample size of only one known outlier. The other sample sizes for the known outliers are 1%, 10% [87] and 50% [49] of the total number of outliers. The assumption that 50% of the outliers are known outliers is a rather strong assumption since acquiring a representative set of outlier examples is difficult and often expensive [16]. However, we include the sample size of 50% known outliers in our experiments, as in [49], for completeness.

The selection of the known outliers affects the model performance. According to chance, the set of the known outliers may represent all the outliers either poorly or quite well. In addition, it is difficult to measure how representative the set of the known outliers is to all the outliers [16]. To mitigate this effect, we repeat the experiments for sample sizes 1, 1% and 10% ten times on each algorithm (OOE, LOG+ and Hybrid) on each dataset with a different set of known outliers. The results reported on each algorithm on each dataset are the average AUC scores over these ten repetitions. The tests for the sample size of 50% are not repeated, because using such a large sample size is not our primary goal, as such a well-sampled set of known outliers is hard to acquire [16]. In addition, such a large sample size diminishes the effect of random chance in itself.

4.3 The effect of the parameter optimization

Now, we present our examinations on the effectiveness of our proposed optimization procedure. We test the hypothesis that the proposed optimization in Eq. (15) improves the outlier detection. We compare our proposed method (OOE) against three ensemble configurations (presented in the ensuing paragraph) that do not utilize known outliers. We attempt to show that our optimization procedure has a positive effect on the results. The comparison against LOG+ [49] (which is another method that utilizes known outliers) is presented in Sect. 4.4.

The first two unoptimized ensemble configurations are dubbed Random [69] and Default [49]. Both of these configurations utilize feature bagging and the detectors KNN and LOF, as in OOE. The difference to OOE is that both configurations (Random [69] and Default [49]) use a heuristic instead of optimization procedure in the selection of the parameters. In other words, both Random and Default implement Algorithm 1 with $\mathbf{X}_o = \emptyset$, without Steps 4–7 and with equal weights $w_j = 1$ for all $j = 1, \ldots, K$. The outlier scores of the detectors are aggregated according to Eq. 1. Random configuration samples the parameters for KNN and LOF from the same pool as OOE ($k \in \{10, \ldots, \text{Round}(\sqrt{n})\}$, distance metric as in Eq. (3)–(8) and a combination function as each of the following: sum, average, median or maximum. Default configuration uses $k = 20$, Euclidean distance as a metric and sum as the combination function [49]. The Default configuration is utilized by LOG+ [49].

The third unoptimized ensemble is SELECT [64]. From all the presented SELECT configurations, we choose Horizontal SELECT with robust rank aggregation since that combination seems to have the highest total performance in [64]. Horizontal SELECT models the outlier scores as a mixture of

Table 2 OOE compared to Random and Default configurations

Versus	Random	Default	SELECT	OOE
Random	0.741	− 0.011	**− 0.043***	**0.028***
Default		0.730	**− 0.033***	**0.038***
SELECT			0.698	**0.071***
OOE				0.768

The diagonal indicates the average AUC scores of each configuration. The off-diagonal indicates the average difference in AUC scores between the two configurations

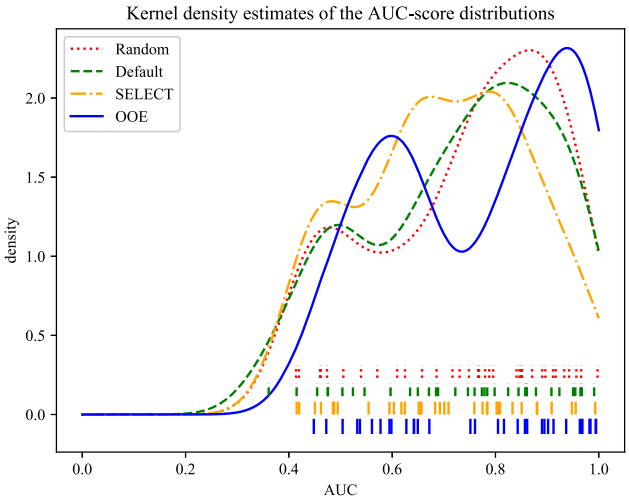


Fig. 2 The kernel density estimates of the AUC score distributions with only one known example. The vertical markers at the bottom of the figure indicate the AUC scores of OOE, Random, Default and SELECT configurations on the individual datasets in Table 1

an exponential and a Gaussian distribution [27]. The pseudo-target outliers are chosen under a hypothesis that their outlier scores are generated from the Gaussian distribution. A set of relevant base detectors is chosen based on how strongly a given base detector agrees with the pseudo-target. Finally, the outlier scores of the relevant base detectors are combined using robust rank aggregation [37]. We use the same set of base detectors for SELECT as for the Random configuration.

The data used in our experimentation are summarized in Table 1. We optimize OOE with only one known outlier example, as it is the most difficult setting. This means that OOE utilizes the minimal amount of external knowledge. If OOE performs better than Random, Default and SELECT with just a single known outlier example, then the optimization routine in Algorithm 1 has a positive impact on the performance. In Table 4 of Sect. 4.4, we show that the more the known outlier examples available, the higher the AUC score of OOE is. To mitigate the randomness that is caused by the sampling of the known outlier example, the experiments are repeated ten times on each dataset [54]. In addition, we reduce the random effect of the feature bagging so that each method uses exactly the same feature bags throughout the experiments.

The results of the comparison of the methods (OOE, Random, Default and SELECT) are presented in Table 2. The diagonal identifies the average AUC scores of each method. The off-diagonal indicates the average difference in AUC scores between the two algorithms over all the datasets in Table 1. The average differences in AUC scores between OOE versus Random, OOE versus Default, and OOE versus SELECT are statistically significant [less than 5% (0.05)] at p values 0.014, 0.021 and 0.000 (see Table 3), respectively, according the paired t test (presented in Sect. 4.1). These statistically significant differences at confidence level 95% (0.95) are bolded and indicated with asterisks ($*$) in Table 2.

Table 3 presents the corresponding p values on all the three tests discussed in Sect. 4.1 (the paired t test, bootstrap and Wilcoxon signed-rank test). The statistically significant p values [less than 5% (0.05)] are bolded. All the three tests agree that OOE performs better than Random, Default and SELECT on a statistically significant level. The three tests cannot differentiate between Random and Default configurations. SELECT has the lowest performance on a statistically significant level.

The AUC scores of each configuration on each dataset in Table 1 are presented in Fig. 2. The markers at the bottom are the AUC scores for each configuration on each individual dataset. The depicted curves represent the kernel density estimate. The two modes in the kernel density estimates allude that it is rather easy to classify the observations into inliers and outliers on half of the datasets, while this outlier detection is considerably harder on the rest of the datasets. The kernel density estimates of the AUC scores of Random and Default configurations are very similar. Indeed, the paired t test does not indicate that there would be any significant difference in performance between Random and Default configurations in Table 2. The mass of the kernel density estimate of the AUC scores of OOE clearly locates more to the right-hand side of Fig. 2 compared to Random and Default configurations. This indicates that OOE achieves higher AUC scores than Random and Default configurations in general. The paired t tests in Table 2 confirm this observation.

Figure 3 presents the difference in AUC scores between OOE and Random configuration with one known outlier example on the datasets in Table 1. Figure 3 also includes the confidence interval of the mean at 95% certainty level. The markers at the density level 0 are the differences in the AUC scores on individual datasets. Values larger than zero indicate datasets on which OOE performs better than Random configuration. The extreme differences are highlighted with the red markers and the corresponding dataset names.

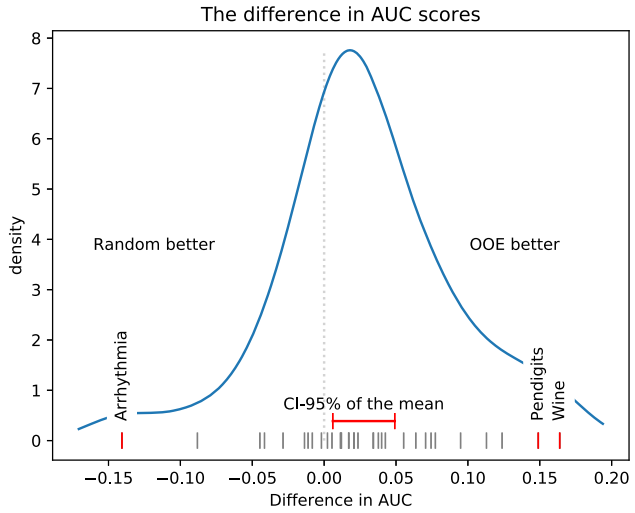


Fig. 3 The differences in AUC scores between OOE and Random configuration on the datasets in Table 1. Each vertical marker at the bottom of the figure indicates the difference on each dataset. The continuous line is the kernel density estimate

Table 4 The average AUC scores of OOE, LOG+ and Hybrid with different amounts of known outlier examples on ODDS and Oregon datasets

Known outliers	OOE	LOG+	Hybrid	H–L
1	0.768	0.769	0.792	**0.023***
1%	0.788	0.829	0.844	0.015
10%	0.809	0.895	0.897	0.002
50%	0.816	0.926	0.929	0.003

The column dubbed H–L indicates the average difference in AUC scores between Hybrid and LOG+

Table 5 The p values on the average difference on AUC scores between Hybrid and LOG+ in Table 4

Known outliers	p value	p-Bootstrap	p-Wilcoxon
1	**0.011**	**0.002**	**0.043**
1%	0.060	**0.023**	0.064
10%	0.512	0.491	0.623
50%	0.359	0.358	0.964

4.4 Performance comparison between OOE, LOG+ and Hybrid on benchmark datasets

Here, we present our comparisons of OOE and Hybrid against LOG+ [49]. The experiments are performed on the data summarized in Table 1. We repeat the test ten times [54] for outlier example sample sizes of one, 1% and 10%. The tests for outlier example sample size of 50% are performed only once, because the larger sample size lessens the effect of random chance and our model is designed specifically for smaller sets of known outliers. As in Sect. 4.3, we use the same features in feature bagging between the methods to reduce the effect of random chance. In addition, LOG+ and Hybrid use the same data samples in data bagging [3] to reduce the impact of random selection.

The average AUC scores of all the algorithms are presented in Table 4. The column dubbed H–L is the average difference in AUC scores between Hybrid and LOG+. Table 4 summarizes the performance of each algorithm on a different sample size of the known outliers. Each of the algorithms performs better when more known outlier examples are available. Hybrid achieves better performance than OOE and LOG+ on all the sample sizes of known outliers. The difference in performance between Hybrid and LOG+ is the largest when only one of the outliers is known and the rest are hidden. With only one known outlier, this difference in performance is statistically significant (p value less than 0.05; bolded and indicated with an asterisk). This is remarkable since gathering an excessive amount of known outliers is difficult and often expensive in practice [16]. The difference between the performances of Hybrid and LOG+ lessens when more external knowledge (known outliers) is available.

The corresponding p values on the average differences in AUC scores between Hybrid and LOG+ are presented in Table 5. All three tests agree that Hybrid performs better than LOG+ when only one known outlier is available. With one percent of all the outliers known beforehand, the difference in performance is not statistically significant (p value is not less than 0.05) for paired t test and Wilcoxon signed-rank test. However, all the tests indicate that the difference in performance still exists on the confidence level of 90% (0.9). If ten percent or more of all the outliers are known beforehand, both Hybrid and LOG+ are similar in performance.

Table 3 The p values on the average differences presented in Table 2

Test	p value	p-Bootstrap	p-Wilcoxon
OOE versus Random	**0.014**	**0.009**	**0.006**
OOE versus Default	**0.021**	**0.009**	**0.023**
OOE versus SELECT	$<$ **0.001**	$<$ **0.001**	$<$ **0.001**
Random versus Default	0.309	0.292	0.478
Random versus SELECT	$<$ **0.001**	$<$ **0.001**	$<$ **0.001**
Default versus SELECT	**0.006**	**0.001**	**0.006**

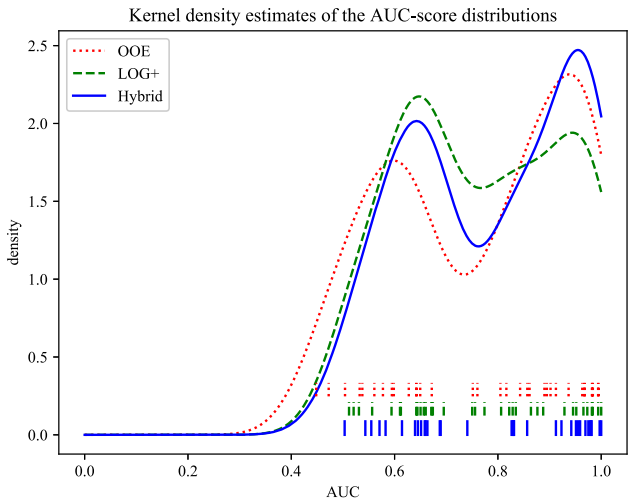


Fig. 4 The kernel density estimates of the AUC score distributions with only one known example. The vertical markers at the bottom of the figure indicate the AUC scores of OOE, LOG+ and Hybrid on the individual datasets in Table 1

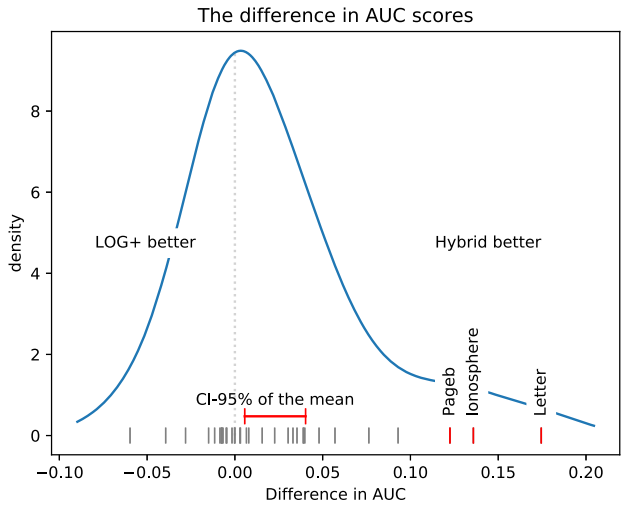


Fig. 5 The differences in AUC scores between Hybrid and LOG+ with one known outlier example of the datasets in Table 1. Each vertical marker at the bottom of the figure indicates the difference on each dataset. The continuous line is the kernel density estimate

OOE and Hybrid approaches utilize the combination function in Eq. (1) with the weights defined in Eq. (13). The weights appear to have little effect on the performance of OOE and Hybrid compared to the flat weights ($w_j = 1$ for all $j = 1, \ldots K$). The difference in the average AUC scores between the combination function in Eq. (1) and the combination function with flat weights ranges from -0.019 to 0.020. These differences fail to be statistically significant (p value is not less than 0.05) when there are only few outlier examples available. With 10% of known outlier examples, the difference in the average AUC scores between OOE and flat-weight OOE is 0.015 ($p < 0.001$). With 50% of known outlier examples, the difference in the average AUC scores between OOE and flat-weight OOE is 0.020 ($p < 0.001$) and between Hybrid and flat-weight Hybrid 0.007 ($p = 0.020$). The consensus seems to be that the weights in Eq. (13) are more relevant when there are more known outlier examples available. This result is reasonable, because when there is more external knowledge available, evaluating the performance of the base detectors during optimization becomes more accurate. Further experimentation with different weighting schemes is left for future work.

Figure 4 shows the kernel density estimates of the AUC score distributions of all the algorithms with one known outlier example. The markers at the bottom in Fig. 4 indicate the AUC scores on the individual datasets in Table 1, and the depicted curves are the kernel density estimates over the AUC scores on respective datasets. The kernel density estimates for Hybrid and LOG+ are rather similar in shape, but the kernel density estimate of Hybrid has more mass around the second peak at AUC score 0.950. This means that Hybrid produces better results than LOG+ in general. The average difference in AUC scores between Hybrid and LOG+ in Table 4 and the p values in Table 5 confirm this observation in the situations in which there is one known outlier.

The differences in AUC scores between Hybrid and LOG+ with one known outlier example on individual datasets are presented in Fig. 5. The continuous line is the kernel density estimate of the distribution of the differences. The confidence interval at the 95% (0.95) confidence level is indicated with the red horizontal bar. The markers at the density 0 represent the differences in AUC scores on each individual dataset in Table 1. The datasets with extreme differences are highlighted with the red vertical markers and dataset names. Values larger than zero indicate datasets on which Hybrid performs better than LOG+.

4.5 Generalizability of the outlier detection

Outliers are often dissimilar to each other [55,67] and contain unusual, unexpected and new information [14]. This means that it is unreasonable to assume that the known outliers cover every type of the possible outliers [16]. For this reason, we test the generalizability of the proposed methods to detect previously unseen outliers.

We select five datasets (Letter, Optdigits, Pageb, Satellite and Yeast) for generalizability tests. These datasets consist of multiple classes that are on a nominal scale. First, in each of the datasets, we construct the normal data (C_N) by selecting the three most frequent classes (except in the dataset Pageb, in which we select only the most frequent class as that class constitutes 89.8% of all the data). Second, we select three classes (two in the dataset Pageb) randomly from all the remaining classes to represent the outlier classes (C_O). All the remain-

Table 6 The results of the generalizability tests

	Known outliers: 1			Known outliers: 10%		
	OOE	LOG+	Hybrid	OOE	LOG+	Hybrid
Letter	**0.981** (0.952)	0.856 (0.952)	0.975 (*0.961*)	**0.986** (0.963)	0.831 (0.970)	0.983 (*0.982*)
Optdigits	**0.982** (0.953)	0.903 (*0.985*)	0.976 (0.984)	**0.982** (0.944)	0.922 (*0.997*)	0.979 (0.992)
Pageb	**0.986** (0.974)	0.602 (0.832)	0.984 (*0.976*)	**0.986** (*0.979*)	0.460 (0.766)	**0.986** (*0.979*)
Satellite	0.959 (0.886)	0.751 (*0.990*)	**0.968** (0.984)	0.973 (0.937)	0.937 (*0.995*)	**0.977** (0.994)
Yeast	0.622 (0.677)	0.599 (0.747)	**0.661** (*0.774*)	**0.705** (0.782)	0.646 (0.855)	0.669 (*0.873*)
Average	0.906 (0.888)	0.742 (0.901)	**0.913** (*0.936*)	**0.926** (0.921)	0.759 (0.917)	0.919 (*0.964*)

The values in the table represent average AUC scores on test data which consist of normal data and previously unseen outliers. The best performance between the methods are bolded. The values in parenthesis are the average AUC scores on test data which consist of only normal data and outliers similar to training data. The best performances on these test data are italicized

ing classes represent previously unseen outlier classes (C_U). Third, we construct the training data by sampling 60% of the normal data C_N and down-sampling the outlier classes C_O until 10% [45] of the training data are from these outlier classes (or until 40% of the data in the outlier classes remain). Finally, we construct two test sets: one test set with outliers (O) and one test set with previously unseen outliers (U). Both test sets, O and U, share the remaining 40% of the normal data C_N (using the train/test split of 60:40 [49]), but O contains a sample from the remaining data from outlier classes C_O, while U contains a sample from the previously unseen outlier classes C_U. We down-sample both test sets until half of the data in the test sets are normal data.

We execute the generalizability test by first training the models (OOE, LOG+ and Hybrid) on the training set with a small sample of known example outliers and then testing the models on both test sets O and U. If a model generalizes well to detect previously unseen outliers, then the model should achieve similar, high AUC scores on both test sets O and U. We repeat the generalizability test ten times with one randomly chosen known example and ten times with a known example sample size of 10% and average the repetitions [54] to receive two AUC scores for each model on each dataset: one AUC score for the test set O and one for the test set U. The known outlier examples are sampled from the training set similarly to Sect. 4.4.

The results of the generalizability tests are presented in Table 6. The values in Table 6 are the AUC scores on the test set U, and the values in the parenthesis are the AUC scores on the test set O. The best AUC scores are bolded on the test set U and italicized on the test set O. Both OOE and Hybrid achieve similar, high AUC scores on the test set U implying high generalizability toward new outliers. The previously unseen outliers seem to degenerate the performance of LOG+ quite drastically. In general, having more examples of labeled outliers available in the training data improves performance in both test sets U and O.

5 Related work

There is plenty of previous work available on outlier detection algorithms [14,33,34,41,51,53,57,82]. There also exists previous work on outlier detection ensembles, such as in [2,42,43].

Bouguessa [12] proposed a probabilistic approach for combining multiple detectors. This approach assumes that the outlier scores follow a multivariate beta mixture model. Rayana and Akoglu [64] utilize a mixture of exponential and Gaussian distributions to model the outlier scores and to generate a pseudo-ground truth. Nguyen et al. [51] proposed an ensemble framework (HeDES) for finding outliers in the random subspaces of a dataset. HeDES creates a synthetic dataset based on the unlabeled dataset and injects artificial outliers in the data. The artificial outliers are sampled from a uniform distribution. Our work does not assume how the outliers or the scores are distributed.

Several studies propose that only a subset of well-performing base detectors should be utilized in the ensemble. Schubert et al. [69] studied combining different detectors and stated that resulting errors from detectors should be uncorrelated. Schubert et al. devised a greedy algorithm for selecting a subset of the base detectors that maximize the diversity of results. Another approach for selecting base detectors is to maximize their correlation with the pseudo-ground truth [64,84]. SELECT [64] selects a fixed set of well-performing base detectors, while LSCP [84] optimizes the set of base detectors for each observation independently. In our work, the optimization is responsible for creating the diverse detectors by adjusting their parameters. Our optimization, and the model of an outlier detection ensemble, allows the use of a detector that (1) provides a score, and (2) utilizes adjustable parameters which greatly improve the flexibility and performance of the detectors. In addition, we utilize few labeled outliers efficiently instead of a pseudo-ground truth. The experiment results show that even a limited number of outlier examples is sufficient.

Recently, the utilization of few examples of labeled outliers has gained attention. An approach called example-based outlier detection by Zhu et al. [87] uses a linear classifier to distinguish outliers from normal data. The outliers are provided by the user, while the normal data are determined as data that receive low outlier scores with LOCI [57]. Then, the linear classifier is iteratively retrained using the data points, which are classified outliers and normal data by the linear classifier. Additional examples of outliers are created by modifying the outlier examples, similarly to the work of Nguyen et al. [51]. Another example-based outlier detection algorithm by Zhu et al. [86] utilizes an evolutionary algorithm to find a lower-dimensional data representation, in which most of the provided outlier examples are significantly outstanding. Then, the outliers reside in the regions, in which the density is lower than in the regions of the provided outlier examples. Unlike the work in [86,87], we do not propose a single algorithm for outlier detection. Our main contribution is an example-based approach for optimizing outlier detection ensembles with detectors that provide outlier scores. Therefore, the algorithms by Zhu et al. [86,87] could be optimized as detectors in an outlier detection ensemble using our proposed approach.

The work in [49,83] constructs a data representation that augments the original data with outlier scores from the base detectors. Outliers are learned in this data representation using logistic regression [49] and XGBoost [83]. The reported results show that the proposed method is efficient for outlier detection. The work in [49] uses 50% of the available outliers as outlier examples for training the classifiers. It is not realistic to have a sample of 50% annotated outliers because outliers are rare by definition [16]. As established previously, our work uses from 1 to 10% outlier examples to optimize the parameters of outlier detection ensembles. Therefore, compared to work in [49], our work utilizes a significantly smaller number of outlier examples to optimize the parameters of outlier detection ensembles. Additionally, our optimization procedure could be used to optimize the base detectors in [49,83].

There are also approaches that utilize deep learning to learn a lower-dimensional data representation in a semi-supervised manner in conjunction with outlier detection [54–56,67]. By integrating representation learning into semi-supervised outlier detection, the learner is able to learn a more relevant data representation compared to the traditional two-step outlier detection, which first learns an unsupervised data representation and then executes outlier detection [55]. REPEN [54] learns a data representation, in which normal data resemble other normal data, while known outliers and probable outlier candidates clearly differ from the normal data. Deep SAD [67] initializes itself by learning a low-dimensional data representation using autoencoder and then enhances that representation by minimizing the volume of a hypersphere surrounding normal data and pushing outliers away from that hypersphere. DevNet [55] learns a data representation that yields high, positive values (around five standard deviations from the mean) in a reference distribution for outlying data. PReNet [56] augments the data by pairing the observations, and then it learns the relation between the paired observations that is either both normal, both outliers or a normal and an outlying observation.

The following list recapitulates the novelty and the benefits of our work compared to the existing work:

- Our proposed example-based optimization is applicable when a limited number of outlier examples are available (1–10%). Our experiments use a smaller number of outlier examples than the work in [87] (7–33) and [49] (50%).
- Our model of an outlier detection ensemble requires the detectors to provide outlier scores. However, the selected set of the outlier detection algorithms is not fixed.
- Our work is the first effort to directly optimize the parameters of detectors of outlier detection ensembles. The results of our experiments show that the optimization is capable of utilizing outlier examples to improve the efficiency of the outlier detection.
- Our work does not require the utilization of artificially created outliers. Therefore, our work does not impose assumptions on how the artificial outliers are distributed.

6 Conclusions

We present an optimization approach for outlier detection ensembles in Sect. 3. The experiments show that individual examples of outliers are sufficient for optimizing outlier detection ensembles. Unlike the previous work in example-based outlier detection [49,86,87], our optimization encompassed with only a few examples can be used for outlier detection algorithms, which provide an outlier score. The experiments in [86,87] use 10% of the available outliers (7–33) as examples (50% in [49]), while our experiments use from 1 to 10% of the available outliers. Therefore, our proposed optimization is suitable when a limited number of outlier examples is available.

Instead of only providing an algorithm for outlier detection (as in [14,33,34,41,51,53,57,82]), our work focuses on optimizing the detector parameters. The previous work in outlier detection ensembles [12,32,51,53,69,82] defines a specific set of outlier detection algorithms. Our work is applicable for a wide variety of outlier detection algorithms. Our proposed optimization is an approach for optimizing the parameters of detectors, which define an outlier score for data points. Our work is the first effort to directly adjust the parameters of the detectors to provide diverse and accurate

results, which improve the efficiency of the outlier detection ensemble.

Our proposed method for optimization opens possibilities for future research. Our method could be extended to use a semi-supervised data representation instead of feature bagging [54]. The use of Bayesian and evolutionary methods could speed up the optimization of the base detectors [30]. Hybrid method can be extended to weight the outlier scores and logistic regression in Step 9 in Algorithm 2 more appropriately than equal weights according to a meta-analysis. Also, it would be beneficial to experiment Hybrid approach with other classifiers in addition to logistic regression such as XGBoost [83]. Another idea is to construct outlier detection ensembles sequentially (see [3]). For example, the detectors could be added dynamically in an outlier detection ensemble using our proposed optimization.

Acknowledgements Open access funding provided by Technical Research Centre of Finland (VTT).

Compliance with ethical standards

Conflict of interest On behalf of all authors, the corresponding author states that there is no conflict of interest.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

References

1. Abe, N., Zadrozny, B., Langford, J.: Outlier detection by active learning. In: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '06, pp. 504–509. ACM, New York, NY, USA (2006)
2. Aggarwal, C.: Outlier ensembles: position paper. SIGKDD Explor. Newsl. **14**(2), 49–58 (2013)
3. Aggarwal, C., Sathe, S.: Theoretical foundations and algorithms for outlier ensembles. SIGKDD Explor. Newsl. **17**(1), 24–47 (2015)
4. Alcala, C., Qin, S.: Reconstruction-based contribution for process monitoring. Automatica **45**(7), 1593–1600 (2009)
5. Angiulli, F., Fassetti, F.: Dolphin: an efficient algorithm for mining distance-based outliers in very large datasets. ACM Trans. Knowl. Discov. Data **3**(1), 1–57 (2009)
6. Azmandian, F., Yilmazer, A., Dy, J.G., Aslam, J.A., Kaeli, D.R.: Harnessing the power of gpus to speed up feature selection for outlier detection. J. Comput. Sci. Technol. **29**(3), 408–422 (2014). https://doi.org/10.1007/s11390-014-1439-4
7. Barandela, R., Sánchez, J., García, V., Rangel, E.: Strategies for learning in class imbalance problems. Pattern Recognit. **36**(3), 849–851 (2003)
8. Bellaachia, A., Bari, A.: A flocking based data mining algorithm for detecting outliers in cancer gene expression microarray data. In: Proceedings of the International Conference on Information Retrieval Knowledge Management, CAMP, pp. 305–311 (2012)
9. Bergstra, J., Bengio, Y.: Random search for hyper-parameter optimization. J. Mach. Learn. Res. **13**, 281–305 (2012)
10. Bishop, C.: Pattern Recognition and Machine Learning. Springer, New York (2006)
11. Böhm, C., Faloutsos, C., Plant, C.: Outlier-robust clustering using independent components. In: Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD '08, pp. 185–198 (2008)
12. Bouguessa, M.: A probabilistic combination approach to improve outlier detection. Int. Conf. Tools Artif. Intell. (ICTAI) **1**, 666–673 (2012)
13. Breiman, L.: Bagging predictors. Mach. Learn. **24**(2), 123–140 (1996)
14. Breunig, M., Kriegel, H., Ng, R., Sander, J.: Lof: identifying density-based local outliers. In: Proceedings of the International Conference on Management of data, SIGMOD '00, pp. 93–104. ACM, New York, NY, USA (2000)
15. Budalakoti, S., Srivastava, A., Otey, M.: Anomaly detection and diagnosis algorithms for discrete symbol sequences with applications to airline safety. IEEE Trans. Syst. Man Cybern. Part C (Appl. Rev.) **39**(1), 101–113 (2009)
16. Chandola, V., Banerjee, A., Kumar, V.: Anomaly detection: a survey. ACM Comput. Surv. **41**(3), 15:1–15:58 (2009)
17. Cheng, H., Ooi, M., Kuang, Y.C., Demidenko, S., Cheah, B.: Outlier distribution detection approach to semiconductor wafer fabrication process monitoring. In: Proceedings of the 3rd Asia Symposium on Quality Electronic Design, ASQED, pp. 62–67 (2011)
18. Cormen, T., Stein, C., Rivest, R., Leiserson, C.: Introduction to Algorithms, 2nd edn. McGraw-Hill Higher Education, New York (2001)
19. Duda, R., Hart, P., Stork, D.: Pattern Classification, 2nd edn. Wiley, Hoboken (2000)
20. Duin, R.P.W.: The combining classifier: to train or not to train? In: Object recognition Supported by User Interaction for Service Robots, vol. 2, pp. 765–770 (2002). https://doi.org/10.1109/ICPR.2002.1048415
21. Emmott, A., Das, S., Dietterich, T., Fern, A., Wong, W.K.: Anomaly detection meta-analysis benchmarks (2016). https://doi.org/10.7267/N97H1GGX. https://ir.library.oregonstate.edu/concern/datasets/47429f155
22. Fang, L., Zhi-zhong, M.: An online outlier detection method for process control time series. In: Proceedings of the Chinese Control and Decision Conference, CCDC, pp. 3263–3267 (2011)
23. Fawcett, T.: An introduction to ROC analysis. Pattern Recognit. Lett. **27**(8), 861–874 (2006)
24. Ganapathy, S., Jaisankar, N., Yogesh, P., Kannan, A.: An intelligent system for intrusion detection using outlier detection. In: Proceedings of the International Conference on Recent Trends in Information Technology (ICRTIT), pp. 119–123 (2011)
25. Ganapathyand, S., Jaisankarand, N., Yogesh, P., Kannan, A.: An intelligent system for intrusion detection using outlier detection. In: International Conference on Recent Trends in Information Technology (ICRTIT), pp. 119–123 (2011)
26. Ganeriwal, S., Balzano, L., Srivastava, M.: Reputation-based framework for high integrity sensor networks. ACM Trans. Sens. Netw. **4**(3), 1–37 (2008)
27. Gao, J., Tan, P.: Converting output scores from outlier detection algorithms into probability estimates. In: Proceedings of the 6th

IEEE International Conference on Data Mining (ICDM 2006), 18–22 December 2006, Hong Kong, China, pp. 212–221. IEEE Computer Society (2006). https://doi.org/10.1109/ICDM.2006.43
28. Gaspar, J., Lopes, F., Freitas, A.: An analysis of hospital coding in Portugal: Detection of patterns, errors and outliers in female breast cancer episodes. In: Proceedings of the 6th Iberian Conference on Information Systems and Technologies (CISTI), pp. 1–6 (2011)
29. Hawkings, D.: Identification of Outliers. Chapman and Hall, London (1980)
30. Hinz, T., Navarro-Guerrero, N., Magg, S., Wermter, S.: Speeding up the hyperparameter optimization of deep convolutional neural networks. Int. J. Comput. Intell. Appl. **17**(02), 1850008 (2018). https://doi.org/10.1142/S1469026818500086
31. Hodge, V., Austin, J.: A survey of outlier detection methdologies. Artif. Intell. Rev. **22**, 85–126 (2004)
32. Huang, B., Li, W., Chen, D., Shi, L.: An intrusion detection method based on outlier ensemble detection. In: Proceedings of the International Conference on Networks Security, Wireless Communications and Trusted Computing, NSWCTC '09, vol. 2, pp. 600–603 (2009)
33. Janssen, J., Huszar, F., Postma, E., van den Herik, E.: Stochastic outlier selection (2012)
34. Jiang, S., An, Q.: Clustering-based outlier detection method. In: Proceedings of the Fifth International Conference on Fuzzy Systems and Knowledge Discovery, FSKD '08, vol. 2, pp. 429–433 (2008)
35. Jiang, Y., Li, M., Zhou, Z.H.: Software defect detection with rocus. J. Comput. Sci. Technol. **26**(2), 328–342 (2011). https://doi.org/10.1007/s11390-011-9439-0
36. Ju, C., Wang, N.: Research on credit card fraud detection model based on similar coefficient sum. In: DBTA, pp. 295–298. IEEE Computer Society (2009)
37. Kolde, R., Laur, S., Adler, P., Vilo, J.: Robust rank aggregation for gene list integration and meta-analysis. Bioinformatics **28**(4), 573–580 (2012). https://doi.org/10.1093/bioinformatics/btr709
38. Konijn, R., Kowalczyk, W.: Finding fraud in health insurance data with two-layer outlier detection approach. In: Proceedings of the 13th International Conference on Data Warehousing and Knowledge Discovery, DaWaK'11, pp. 394–405. Springer, Berlin, Heidelberg (2011)
39. Kreyszig, E.: Introductory Functional Analysis with Application. Wiley, Hoboken (1978)
40. Kriegel, H., Hubert, M., Zimek, A.: Angle-based outlier detection in high-dimensional data. In: Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '08, pp. 444–452. ACM, New York, NY, USA (2008)
41. Kriegel, H., Kröger, P., Schubert, E., Zimek, A.: Loop: Local outlier probabilities. In: Proceedings of the 18th ACM Conference on Information and Knowledge Management, CIKM '09, pp. 1649–1652. ACM, New York, NY, USA (2009)
42. Kriegel, H., Kröger, P., Schubert, E., Zimek, A.: Interpreting and unifying outlier scores. In: Proceedings of the SIAM International Conference on Data Mining, SDM, pp. 13–24. SIAM/Omnipress (2011)
43. Lazarevic, A., Kumar, V.: Feature bagging for outlier detection. In: Proceedings of the Eleventh International Conference on Knowledge Discovery in Data Mining, KDD '05, pp. 157–166. ACM, New York, NY, USA (2005)
44. Lee, Y., Yeh, Y., Wang, Y.: Anomaly detection via online oversampling principal component analysis. IEEE Trans. Knowl. Data Eng. **25**(7), 1460–1470 (2013)
45. Lei, D., Zhu, Q., Chen, J., Lin, H., Yang, P.: Automatic k-means clustering algorithm for outlier detection. In: Zhu, R., Ma, Y. (eds.) Information Engineering and Applications, pp. 363–372. Springer, London (2012)
46. Li, C., Georgiopoulos, M., Anagnostopoulos, G.: Kernel principal subspace mahalanobis distances for outlier detection. In: Proceedings of the 2011 International Joint Conference on Neural Networks (IJCNN), pp. 2528–2535 (2011)
47. Li, Y., Nitinawarat, S., Veeravalli, V.: Universal outlier detection. Computer Research Repository (2013). arXiv:1302.4776
48. Maya, S., Ueno, K., Nishikawa, T.: dLSTM: a new approach for anomaly detection using deep learning with delayed prediction. Int. J. Data Sci. Anal. **8**(2), 137–164 (2019). https://doi.org/10.1007/s41060-019-00186-0
49. Micenkova, B., McWilliams, B., Assent, I.: Learning outlier ensembles: the best of both worlds—supervised and unsupervised. In: Proceedings of the ACM SIGKDD Workshop on Outlier Detection and Description, ODD '14 (2014)
50. Murphy, K.: Machine Learning: A Probabilistic Perspective. The MIT Press, Cambridge (2012)
51. Nguyen, H., Ang, H., Gopalkrishnan, V.: Mining outliers with ensemble of heterogeneous detectors on random subspaces. In: Proceedings of the 15th International Conference on Database Systems for Advanced Applications, DASFAA'10, pp. 368–383. Springer, Berlin, Heidelberg (2010)
52. Ninomiya, Y., Kawano, S.: AIC for the LASSO in generalized linear models. Electron. J. Stat. **10**(2), 2537–2560 (2016). https://doi.org/10.1214/16-EJS1179
53. Pamula, R., Deka, J., Nandi, S.: Distance-based fast outlier detection method. In: Proceedings of the 2010 Annual IEEE India Conference., INDICON, pp. 1–4 (2010)
54. Pang, G., Cao, L., Chen, L., Liu, H.: Learning representations of ultrahigh-dimensional data for random distance-based outlier detection. In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD '18, p. 2041–2050. Association for Computing Machinery, New York, NY, USA (2018). https://doi.org/10.1145/3219819.3220042
55. Pang, G., Shen, C., van den Hengel, A.: Deep anomaly detection with deviation networks. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '19, pp. 353–362. Association for Computing Machinery, New York, NY, USA (2019). https://doi.org/10.1145/3292500.3330871
56. Pang, G., Shen, C., Jin, H., van den Hengel, A.: Deep weakly-supervised anomaly detection (2019)
57. Papadimitriou, S., Kitagawa, H., Gibbons, P., Faloutsos, C.: Loci: Fast outlier detection using the local correlation integral. In: Proceedings of the International Conference on Data Engineering, ICDE, pp. 315–326. IEEE Computer Society (2003)
58. Paschalidis, I., Chen, Y.: Statistical anomaly detection with sensor networks. ACM Trans. Sens. Netw. **7**(2), 1–23 (2010)
59. Peduzzi, P., Concato, J., Kemper, E., Holford, T.R., Feinstein, A.R.: A simulation study of the number of events per variable in logistic regression analysis. Journal of Clinical Epidemiology **49**(12), 1373–1379 (1996). https://doi.org/10.1016/S0895-4356(96)00236-3
60. Rajasegarar, S., Bezdek, J., Leckie, C., Palaniswami, M.: Elliptical anomalies in wireless sensor networks. ACM Trans. Sens. Netw. **6**(1), 1–28 (2010)
61. Ramaswamy, S., Rastogi, R., Shim, K.: Efficient algorithms for mining outliers from large data sets. In: Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, SIGMOD '00, pp. 427–438. ACM, New York, NY, USA (2000)
62. Raniga, P., Schmitt, P., Bourgeat, P., J. Fripp, V.V., Rowe, C., Salvado, O.: Local intensity model: an outlier detection framework with applications to white matter hyperintensity segmentation. In: IEEE International Symposium on Biomedical Imaging, pp. 2057–2060 (2011)
63. Rayana, S.: ODDS library (2016). http://odds.cs.stonybrook.edu

64. Rayana, S., Akoglu, L.: Less is more: building selective anomaly ensembles. ACM Trans. Knowl. Discov. Data (2016). https://doi.org/10.1145/2890508
65. Rietveld, T., van Hout, R.: The paired t test and beyond: recommendations for testing the central tendencies of two paired samples in research on speech, language and hearing pathology. J. Commun. Disord. **69**, 44–57 (2017). https://doi.org/10.1016/j.jcomdis.2017.07.002
66. Rousseeuw, P., Leroy, A.: Robust Regression and Outlier Detection. Wiley, New York (1987)
67. Ruff, L., Vandermeulen, R.A., Görnitz, N., Binder, A., Müller, E., Müller, K.R., Kloft, M.: Deep semi-supervised anomaly detection (2019)
68. Sarasamma, S., Zhu, Q., Huff, J.: Hierarchical Kohonenen net for anomaly detection in network security. IEEE Trans. Syst. Man Cybern. Part B **35**(2), 302–312 (2005)
69. Schubert, E., Wojdanowski, R., Zimek, A., Kriegel, H.P.: On evaluation of outlier rankings and outlier scores. In: Proceedings of the SIAM International Conference on Data Mining, SDM, pp. 1047–1058 (2012)
70. Singh, K., Upadhyaya, S.: Outlier detection: applications and techniques. IJCSI Int. J. Comput. Sci. Issues **9**(3), 307–323 (2012)
71. Tao, W., Wenbo, Z., Jun, W., Hua, Z.: Workload-aware online anomaly detection in enterprise applications with local outlier factor. In: Proceedings of the IEEE 36th Annual Computer Software and Applications Conference, pp. 25–34 (2012)
72. Tax, D., Duin, R.: Support vector data description. Mach. Learn. **54**(1), 45–66 (2004)
73. Theodoridis, S., Koutroumbas, K.: Pattern Recognition, 4th edn. Academic Press, Cambridge (2008)
74. Trede, M.: Bootstrapping inequality measures under the null hypothesis: is it worth the effort? J. Econ. **77**(1), 261–282 (2002). https://doi.org/10.1007/BF03052507
75. Vidaurre, D., Bielza, C., Larrañaga, P.: A survey of l1 regression. Int. Stat. Rev. **81**(3), 361–387 (2013). https://doi.org/10.1111/insr.12023
76. Wang, X.T., Shen, D.R., Bai, M., Nie, T.Z., Kou, Y., Yu, G.: An efficient algorithm for distributed outlier detection in large multi-dimensional datasets. J. Comput. Sci. Technol. **30**(6), 1233–1248 (2015). https://doi.org/10.1007/s11390-015-1596-0
77. Wilcoxon, F.: Individual comparisons by ranking methods. Biom. Bull. **1**(6), 80–83 (1945)
78. Yang, P., Huang, B.: KNN based outlier detection algorithm in large dataset. Proc. Int. Workshop Geosci. Remote Sens. **1**, 611–613 (2008)
79. Yu, B., Song, M., Wang, L.: Local isolation coefficient-based outlier mining algorithm. In: International Conference on Information Technology and Computer Science, ITCS 2009, vol. 2, pp. 448–451 (2009)
80. Zengan, G.: Application of cluster-based local outlier factor algorithm in anti-money laundering. In: Proceedings of the International Conference on Management and Service Science, pp. 1–4 (2009)
81. Zhang, J., Zulkernine, M.: Anomaly based network intrusion detection with unsupervised outlier detection. In: Proceedings of the IEEE International Conference on Communications (ICC '06), vol. 5, pp. 2388–2393 (2006)
82. Zhang, K., Hutter, M., Jin, H.: A new local distance-based outlier detection approach for scattered real-world data. In: Computing Research Repository (2009)
83. Zhao, Y., Hryniewicki, M.K.: XGBOD: improving supervised outlier detection with unsupervised representation learning. In: 2018 International Joint Conference on Neural Networks, IJCNN 2018, Rio de Janeiro, Brazil, July 8–13, 2018, pp. 1–8. IEEE (2018). https://doi.org/10.1109/IJCNN.2018.8489605
84. Zhao, Y., Nasrullah, Z., Hryniewicki, M.K., Li, Z.: LSCP: locally selective combination in parallel outlier ensembles. In: Berger-Wolf, T.Y., Chawla, N.V. (eds.) Proceedings of the 2019 SIAM International Conference on Data Mining, SDM 2019, Calgary, Alberta, Canada, May 2–4, 2019, pp. 585–593. SIAM (2019). https://doi.org/10.1137/1.9781611975673.66
85. Zhou, C., Huang, S., Xiong, N., Yang, S., Li, H., Qin, Y., Li, X.: Design and analysis of multimodel-based anomaly intrusion detection systems in industrial process automation. IEEE Trans. Syst. Man Cybern. Syst. **45**(10), 1345–1360 (2015)
86. Zhu, C., Kitagawa, H., Faloutsos, C.: Example-based robust outlier detection in high dimensional datasets. In: Proceedings of the Fifth IEEE International Conference on Data Mining (2005)
87. Zhu, C., Kitagawa, H., Papadimitriou, S., Faloutsos, C.: Obe: outlier by example. In: Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining, pp. 222–234 (2004)
88. Zimek, A., Campello, R., Sander, J.: Data perturbation for outlier detection ensembles. In: Proceedings of the 26th International Conference on Scientific and Statistical Database Management, SSDBM '14, pp. 13:1–13:12. ACM, New York, NY, USA (2014)
89. Zimek, A., Campello, R., Sander, J.: Ensembles for unsupervised outlier detection: challenges and research questions a position paper. SIGKDD Explor. Newsl. **15**(1), 11–22 (2014)
90. Zimek, A., Gaudet, M., Campello, R., Sander, J.: Subsampling for efficient and effective unsupervised outlier detection ensembles. In: Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '13, pp. 428–436. ACM, New York, NY, USA (2013)

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.