Self-supervised learning for outlier detection

Jan Diers | Christian Pigorsch

The identification of outliers is mainly based on unannotated data and therefore constitutes an unsupervised problem. The lack of a label leads to numerous challenges that do not occur or only occur to a lesser extent when using annotated data and supervised methods. In this paper, we focus on two of these challenges: the selection of hyperparameters and the selection of informative features. To this end, we propose a method to transform the unsupervised problem of outlier detection into a supervised problem. Benchmarking our approach against common outlier detection methods shows clear advantages of our method when many irrelevant features are present. Furthermore, the proposed approach also scores very well in the selection of hyperparameters, i.e., compared to methods with randomly selected hyperparameters.

KEYWORDS

hyperparameter, machine learning, noisy signal, outlier detection, self-supervised learning

1 INTRODUCTION

The task of outlier detection can be solved by both supervised and unsupervised methods. If annotated data are available, it is convenient to use a supervised classifier for the problem. However, in practical applications, for example, in auditing, the detection of intruders into an IT system or the detection of credit card fraud, such annotated data are often missing. In this case, unsupervised algorithms must be used, which have been proposed in many variants over recent years.

However, unsupervised methods lead to difficulties which can usually be easily solved in supervised learning. One of the difficulties is finding suitable hyperparameters for the concrete data set. In the supervised setting, labels are available to provide feedback on whether a particular hyperparameter is well-chosen or whether another choice should be preferred. An example is the k-nearest neighbour (kNN) classification. In the supervised context, the hyperparameter k of the nearest neighbours can be found, for example, by cross-validation. In the unsupervised context, in contrast, no optimal value for k can be determined without labels. In this case, a value for k can be guessed rather than determined. However, the correct choice of hyperparameters is important to ensure that unsupervised outlier detection can provide reliable results, as highlighted by Campos et al. (2015).

Another problem is the selection of features that are of interest to the targeted objective. In many data sets, some features correlate more or less strongly with the target. Additionally, some features are without any causality to the target, that is, they do not provide any information gain for the target class. In the supervised setting, these irrelevant features can be eliminated by suitable feature selection, again using the available labels. In the unsupervised context, it is more difficult to agree on a subset of features that are relevant for the target. For the area of outlier detection, too many irrelevant features can cause the outliers to be hidden in the noninformative features and cause outlier detection to fail (Zimek, Schubert, & Kriegel, 2012).

In this paper, we address these problems by transforming the task of unsupervised outlier detection into a supervised problem. Based on unlabelled data, we present an algorithm that generates data and labels which are suitable for the task of outlier detection. This corresponds to the idea of self-supervised learning. As such, our approach solves the previously mentioned problems in a similar way as supervised learning. We show that our method provides state-of-the-art results in the field of outlier detection and is extremely robust against features that have no information about the outlyingness of a point. To facilitate the application of our method in research and practice, we published our code as a ready-to-use package on Python's package index. Further information and the source code may be found on GitHub.¹

¹Implementation and installation guide is available under https://github.com/JanDiers/self-supervised-outlier

This is an open access article under the terms of the Creative Commons Attribution License, which permits use, distribution and reproduction in any medium, provided the original work is properly cited.

© 2020 The Authors. Stat published by John Wiley & Sons Ltd
2 | RELATED LITERATURE

Outlier detection can be turned into a classification problem and solved by any supervised classifier whenever labels are available for the data. However, for unlabelled data, this approach is infeasible and leads to the previously mentioned problems, which we mitigate in this paper. Thus, in this paper, we focus on the situation when annotated data are not available.

The literature in the field of outlier detection is extensive. A comprehensive overview is provided, for example, by Hodge and Austin (2004), Chandola, Banerjee, and Kumar (2009), or Aggarwal (2013a). In this section, we briefly discuss the methods that are particularly relevant to our work.

A variety of unsupervised algorithms are based on distances to the $k$-nearest neighbours (Angiulli & Pizzuti, 2002; Knorr & Ng, 1997; Ramaswamy, Rastogi, & Shim, 2000). A variation of $k$-nearest neighbour methods is proposed by Hautamäki, Karkkainen, and Franti (2004) by considering only the neighbourhood graph, not the absolute distances between points. The use of distances is a natural way to find outliers: large distances between points correspond to low density in the surrounding area. For this reason, the search for nearest neighbours can be interpreted as an estimate of the local density. Local variants of neighbourhood-based algorithms such as the local outlier factor (LOF) (Breunig, Kriegel, Ng, & Sander, 2000) set the distance to neighbours in relation to the usual distances of the surrounding regions. This helps to equally weight outliers in areas of different variance since the resulting outlier scores are normalized and not only dependent on the absolute distance between points.

Another family of methods builds on dimensionality reduction techniques to detect outliers (Candès, Li, Ma, & Wright, 2011; Hoffmann, 2007). The underlying assumption is that outliers do not conform with the general structure of the data and therefore are not well mapped when the dimension of the data is reduced. The retransformation of the data from the dimension reduced-form into the full dimension is then less successful for outliers. The distance between the original point and its restored variant provides a measure of outlyingness. This approach is also used in the deep learning community by using an autoencoder for dimensionality reduction (Sakurada & Yairi, 2014; Zhou & Paffenroth, 2017; Zong et al. 2018). Variants of the autoencoder such as the variational autoencoder have also proven to be successful in detecting outliers (An & Cho, 2015; Vasilev et al. 2018).

In addition to the approaches based on distances or dimensionality reduction, there also exist methods based on models for one-class classification. Schölkopf, Platt, Shawe-Taylor, Smola, and Williamson (2001) and Tax and Duin (2004) suggest variants of the support vector machine (SVM) for outlier detection. The deep learning community has taken this up. Erfani, Rajasegarar, Karunasekera, and Leckie (2016) use deep learning models for feature extraction and a one-class classification for anomaly detection. Recent studies directly transfer the core idea of one-class classification into a model class of neural networks that is suitable for one-class classification (Chalapathy, Menon, & Chawla, 2018; Perera & Patel, 2019; Ruff et al. 2018).

The presence of irrelevant features can quickly lead to the curse of dimensionality such that feature selection methods are also relevant for the area of unsupervised outlier detection. Especially methods which are based on distances to other points suffer from the curse of dimensionality. Beyer, Goldstein, Ramakrishnan, and Shaft (1999) show that the distances converge to the maximum distance in the data set with increasing dimensionality; that is, the variance of the different distances decreases. This can be problematic for outlier detection methods that define outliers through the distance to other points. The authors emphasize that in data sets with $10$–$15$ features, a concentration effect of distances can already occur; that is, the meaningfulness of distances is lost. The implications of these results for the field of outlier detection was examined in more detail by Zimek et al. (2012). They also observe a concentration effect of distances but find that the problem of high dimensionality is particularly challenging when there are features that have no information about the outlyingness of a point.

One possibility to eliminate the irrelevant features is to utilize ensembles of methods that each operate on a subset of the features. Besides mitigating the curse of dimensionality, the use of ensemble methods can also increase the accuracy of outlier detection. As a result, the success of ensemble learning for supervised applications has also influenced the unsupervised area of outlier detection (Aggarwal, 2013b; Aggarwal & Sathe, 2015; Zimek, Campello, & Sander, 2014). An autoencoder that implements an ensemble of classifiers was proposed by Chen, Sathe, Aggarwal, and Turaga (2017). The isolation forest (Liu, Ting, & Zhou, 2008) can be seen as an ensemble since it is based on a random forest. In addition to the ensemble, the isolation forest also provides options for eliminating irrelevant features: By selecting new features for each split (feature bagging), a feature selection takes place without additional costs. Our method can also be considered as an ensemble method since we base the necessary algorithms on random forests. For the same reason, we also have a built-in feature selection mechanism.

The bagging of features can also be conducted without random or isolation forest as suggested by Lazarevic and Kumar (2005). They draw random subsets of features in different iterations and perform outlier detection on that subset. Afterwards, the outlier scores from the different iterations are aggregated for each point. This way, the authors obtain an ensemble of methods that improves the results in the experiments compared to a simple model. Instead of selecting the features randomly, different weights can be assigned to them. For this purpose, assumptions are made about which features may be relevant for the detection of outliers and which may not. Kriegel, Kröger, Schubert, and Zimek (2009b) work with a weighted distance function that assigns each feature a different weight. The weights are determined for each point from a set of reference points, measured by the variance of the features. The assumption is that features with high variance have little meaning about outliers.
The observation that certain feature distributions are less meaningful about outliers also motivates the work of Müller, Schiffer, and Seidl (2011). They use statistical tests to check whether the feature is uniformly distributed. If so, the feature is removed from the set of relevant features for outlier detection.

While these methods directly address the problem of irrelevant features, it remains open how the hyperparameters can be optimally chosen. Our work is motivated by the fact that the problems of feature selection and the choice of hyperparameters are easier to solve in supervised learning. For this reason, we aim to transform unsupervised outlier detection into a supervised problem. To our knowledge, self-supervised learning (Doersch, Gupta, & Efros, 2015), previously also self-taught learning (Raina, Battle, Lee, Packer, & Ng, 2007), has so far rarely been used in the field of outlier detection. Self-supervised learning addresses the problem that for many machine learning applications a great amount of unlabelled data is available, which, however, cannot be used for supervised learning. Self-supervision aims at exploiting the unlabelled data for the training of the classifier. For this purpose, labels are automatically created by the computer and guide the training. How the label is created depends on the underlying data and the task to be solved.

Previous applications of self-supervised learning mostly use image data. Isola, Zoran, Krishnan, and Adelson (2015) classify tuples of images according to whether a tuple appears next to each other on the image. Other works model a neural network to predict the relative position of two image fragments (Doersch et al. 2015; Noroozi & Favaro, 2016). Gidaris, Singh, and Komodakis (2018) rotate the entire image and train the classifier to determine the type of rotation. Liu, van de Weijer, and Bagdanov (2019) evaluate the quality of pictures. For this purpose, they add Gaussian noise of varying intensities to the image to learn an order in image quality.

Recently, outliers have also been associated with adversarial attacks on existing classifiers. In the deep learning community, this is called out-of-distribution detection. It aims at the identification of data points that do not stem from the distribution of the training data. For this purpose, Hendrycks, Mazeika, and Dietterich (2019) and Hendrycks, Mazeika, Kadavath, and Song (2019) use self-supervised learning for outlier detection. These approaches are enhancements of supervised learning to make the classifier more robust to changes in the input. It further requires that a classifier is already available, which should be made more robust against outliers at the time of inference. These approaches belong to the field of supervised learning, and the goal is therefore different from our method.

Our work builds upon the work of Abe, Zadrozny, and Langford (2006), who also use synthetic points to transform outlier detection into a supervised classification problem. Our work provides essential extensions in at least three points. First, we propose a refined algorithm to generate the artificial outliers. Second, as we find that the choice of hyperparameters can have a strong influence on the performance of outlier detection methods, we propose an algorithm for optimizing the choice of hyperparameters, such that a poor ad hoc selection of hyperparameters is avoided. Third, we show that the proposed method, in contrast to all the other outlier detection methods considered, does not suffer from the curse of dimensionality when noninformative features are present.

### 3 | METHOD

#### 3.1 | Overview

A key element of our method is the transformation of the unsupervised problem of outlier detection into a supervised one. For this purpose, we assume that a data set \( \mathbf{X} \) exists, which may already be contaminated by outliers. As we focus on the common situation that labels are unavailable, a supervised learning method cannot be used to train the classifier, and, thus, it is unknown which point \( x_i \in \mathbf{X} \) represents an outlier. The raw data set \( \mathbf{X} \) does therefore not allow to solve a classification problem already.

To transform the problem into a supervised learning problem, points \( \tilde{\mathbf{N}} := \{\tilde{n}_i, i = 1, \ldots, k\} \) which do not follow the distribution in \( \mathbf{X} \) are generated. For this, we draw random points from a uniform distribution in different iterations and remove those points that are too similar to the actual points in \( \mathbf{X} \). By knowing which points belong to \( \mathbf{X} \) and which points belong to \( \tilde{\mathbf{N}} \), we can generate labels \( y \) that allow a classifier to learn the difference between \( \mathbf{X} \) and \( \tilde{\mathbf{N}} \). The details for generating the points \( \tilde{\mathbf{N}} \) are described in Algorithm 1.

If the classifier has learned to detect differences between \( \mathbf{X} \) and \( \tilde{\mathbf{N}} \), this implies that the same classifier can also be used to determine whether a certain point fits into the structure of \( \mathbf{X} \). This transfer of the unsupervised to the supervised problem is therefore suitable for identifying outliers in \( \mathbf{X} \). If a classifier has difficulties deciding for a point \( x_i \) whether it belongs to \( \mathbf{X} \) or \( \tilde{\mathbf{N}} \), then this point does not follow the natural distribution in \( \mathbf{X} \) and can, therefore, be identified as an outlier.

To ensure that the classification problem is suitable for outlier detection, the classifier must be chosen appropriately. In our case, we use a random forest because it has hyperparameters that are beneficial for outlier detection and learning decision boundaries, especially on tabular data. The choice of the hyperparameters is based on an adjusted objective function for a random search. The details are given in Sections 3.3 and 3.4 and Algorithm 2.

In summary, our method consists of the following three steps:

1. creation of points \( \tilde{\mathbf{N}} \), which are suitable for the discrimination between \( \mathbf{X} \) and \( \tilde{\mathbf{N}} \),
2. search for suitable hyperparameters to make the classification problem between \( \mathbf{X} \) and \( \tilde{\mathbf{N}} \) useful for outlier detection and
3. learning of the decision boundaries for the classification problem and inference of outlier scores for each point.
3.2 Generation of data for self-supervised task and inference of outlier scores

The basic idea of our method is to learn a classifier that can distinguish between real points \( X \) of the data set and artificially created points \( \tilde{N} \). The points \( \tilde{N} \) should fill the feature space evenly and ideally should also be similar to outliers in the data set. This works especially well if the outliers occur in isolation from other observations, which is by definition a property of outliers.

In the following, it is assumed that the data \( X \) is normalized within an interval \([0, 1]\). The artificial data \( \tilde{N} \) is generated in iterations by repeatedly sampling and removing points from a uniform distribution. The first step is to find hyperparameters for a classifier that can separate the data \( X \) from random noise \( \tilde{n} \). The classifier is necessary to remove points in later iterations if they are too similar to the actual data. Each feature \( \tilde{n}_i \) of \( \tilde{n} \) is filled with values from a uniform distribution over the interval \([0, 1]\).

After an initial classifier is obtained to separate points \( X \) from \( \tilde{n} \), further noise \( \tilde{n} \) is drawn in iterations. After each iteration, those points are removed from \( \tilde{n} \), which the classifier has classified as points in \( X \). This prevents that existing clustered structures in \( X \) are destroyed by the artificially created points. The remaining points are added to \( \tilde{N} \). For our experiments, we repeated the sampling loop 20 times. However, since undesired points are removed after every iteration, one can also repeat the sampling more often. This does not hurt the outlier detection but will take more time to estimate outlier scores. Therefore, we recommend not setting this value too high. The points in \( \tilde{N} \) are suitable to convert the outlier detection into a classification. Algorithm 1 describes the details for generating the data.

Figure 1 illustrates the results of Algorithm 1. The data were taken from the publication of Kriegel, Kröger, Schubert, and Zimek (2009a). The blue dots belong to the original data \( X \), and the orange dots represent the artificial points \( \tilde{N} \). The points in \( \tilde{N} \) are sampled around the clusters on \( X \), while some regions, where both points from \( X \) and points from \( \tilde{N} \) fall close together, mark the border of clusters for inlier in \( X \). The outliers are isolated from the rest of the points in \( X \) and well surrounded by points \( \tilde{N} \). This makes the points of \( \tilde{N} \) suitable for the classification and the inference of outlier scores.

**Algorithm 1** Generate data \( \tilde{N} \) distributed around \( X \)

**Require:** \( X \): Data normalized to be in range \([0, 1]\)

**Require:** \( h_i, i \in \{1, \ldots, H\} \) different choices of hyperparameters

**Require:** \( f_i \): Random Forest Classifier with hyperparameters \( h_i \)

**procedure** search hyperparameters for creation of data

\[ \tilde{n} \leftarrow \text{Uniform}[0, 1] \text{ with same dimension as } X \]
\[ \tilde{X} \leftarrow \left( X \ \tilde{n} \right)^T, \tilde{y} \leftarrow \left( \begin{array}{c} 0 \ 1 \end{array} \right)^T \text{ matrix concatenation} \]
\[ f^* \leftarrow f_i \text{ with highest accuracy to estimate } P(\tilde{y} | \tilde{X}) \]

**end procedure**

\[ \tilde{N} \leftarrow \{ \} \]

**procedure** create data for self-supervised task

repeat

\[ \tilde{n} \leftarrow \text{Uniform}[0, 1] \text{ with same dimension as } X \]
\[ \tilde{N} \leftarrow \tilde{N} \cup \{ \tilde{n}_i : f^*(\tilde{n}_i) > 0.5, \ \forall \tilde{n}_i \in \tilde{N} \} \]

until 20 iterations are reached

**end procedure**

The data generated according to Algorithm 1 is suitable to solve a normal classification problem. The classification between \( X \) and \( \tilde{N} \) is suitable for the inference of outlier scores. For this purpose, the classifier is asked for the probability that a point \( x_i \) is a point from \( \tilde{N} \). Thus, it is classified according to whether the point \( x_i \) was artificially created or whether it actually corresponds to the distribution in \( X \). The greater the classifier estimates the probability \( P(\tilde{N}|x_i) \) that \( x_i \in \tilde{N} \), the higher is the outlier score of that point.

3.3 Hyperparameter optimization

The optimization of the hyperparameters for the inference of the outlier scores is based on a random search. A random search tests different combinations of hyperparameters, whereby the specific value for a hyperparameter is chosen randomly. The user specifies a probability distribution from which random values for hyperparameters are drawn. The best combination of hyperparameters is found using a scoring function. In many cases, this can be the accuracy or the area under the curve (AUC) value. The final choice is the combination of hyperparameters that has achieved the highest value in the scoring function.

For the optimization of the classifier’s hyperparameters in Algorithm 2, we use an adjusted scoring function. The reason is that the classifier should be chosen so that it is not able to distinguish perfectly between \( X \) and \( \tilde{N} \). A perfect classification is useless for outlier detection since the outliers in \( X \) are also correctly assigned to \( X \). However, our goal is to classify the outliers in \( X \) as points in \( \tilde{N} \) in order to achieve outlier detection. This is illustrated in Figure 2.

The scoring function for optimizing the hyperparameters follows these principles and is shown in Algorithm 2. For effective optimization, an assumption is necessary about the percentage of points in \( X \) representing outliers. We note this value as \( c \). For the illustration here, we assume the value of \( c = 5\% \). First, we draw a new combination of hyperparameters and classify the points according to their belonging to \( X \) or \( \tilde{N} \). Then,
we consider the outlier score for the points in $X$: $\text{predicted\_scores} := f_i(X)$. For the $\text{predicted\_scores}$, we look at the 95% ($= (1 - c)$) quantile and call it the p-score. In the same way, we also consider the minimum outlier score that was predicted (m-score). The score for the present combination of hyperparameters is the difference between the p-score and the m-score.

**Algorithm 2** Hyperparameter optimization for classification problem

Require: $c$: percentage of suspected outliers in $X$

Require: $X$: Data normalized to be in range $[0, 1]$,

Require: $h_i; i \in \{1, \ldots, H\}$ different choices of hyperparameters,

Require: $f_i$: Random Forest Classifier as trained in algorithm 2 with hyperparameters $h_i$

procedure test all given hyperparameters

repeat

$f_i \leftarrow$ fit Random Forest on $X$ according to algorithm 1

predicted\_scores $\leftarrow f_i(X)$

p-score $\leftarrow (1 - c)$-percentile(predicted\_scores)$

m-score $\leftarrow \min(\text{predicted\_scores})$

model\_score$_i$ $\leftarrow$ p-score $-$ m-score

until all hyperparameters are tested

end procedure

$f^* \leftarrow \arg\max_h(\text{model\_score}_i)$

$h^* \leftarrow \arg\max_h(\text{model\_score}_i)$

This scoring is motivated by the objective to achieve heterogeneous outlier scores. While a minimum outlier score of 0 is desirable for many points, some points should also have large outlier scores. This is represented by the minimum and quantile of outlier scores. The bigger the difference between minimum and quantile, the more heterogeneous are the outlier scores, and we may assume that some points are isolated and can, therefore, be considered as outliers (Figure 3).

### 3.4 Choice of hyperparameters to optimize

A random forest has hyperparameters that describe on the one hand the growth of the trees and on the other hand the composition of the individual trees to form the random forest. In our experiments, we found some hyperparameters to be important for performance, while the
A perfect classifier would classify $x_j$ as a point in $X$, whereas from the outlier perspective, the classifier should predict $x_j$ to be part of $\tilde{N}$. Therefore, we need a custom scoring function for the random search.

Predicted outlier scores according to the discrimination between $X$ and $\tilde{N}$. The classifier is asked to predict the probability $P(\tilde{N}|x_i)$ that $x_i \in \tilde{N}$. The higher the classifier estimates the probability that $x_i$ is a point in $\tilde{N}$, the more likely this point actually is an outlier.

Choice of other hyperparameters seems to have little impact on the results. Therefore, we optimize only those hyperparameters that have a larger impact and fix the other parameters. For our experiments, we split using the Gini impurity, and consider the full data set for each tree and $\sqrt{n_{\text{features}}}$ for each split. Other parameters are optimized according to Algorithm 2. These include the number of estimators in the random forest, the minimum number of samples at the end of each tree, and the pruning parameter. We look for the number of estimators in a range of [50; 150], the minimum number of samples at the end in the range [5; 10], and the parameter for the pruning between [0.01; 0.3]. With these values, we test a wide range of settings for different hyperparameters.

The trees in the forest are fully grown until the required minimum number of samples is undercut. The pruning in the following describes a cost-complexity pruning and weighs the depth of the tree in relation to the generated accuracy. Details can be found in Breiman, Friedman, Olshen, and Stone (1984). Defining $T$ to be the set of all leaves of the tree, $\alpha$ is the value of the pruning parameter and $R(T)$ is the accuracy, a subtree of the tree is searched during pruning that minimizes the following criterion:

$$R_\alpha(T) = R(T) + \alpha \cdot |T|.$$ 

The hyperparameter $\alpha$ is the one that is determined by hyperparameter optimization. In other words, pruning leads to decision trees with the highest possible accuracy, while model complexity is low measured by the number of leaves.
TABLE 1  Benchmarking data in our study

| Features   | Inlier | Outlier | Pct. outlier |
|------------|--------|---------|--------------|
| Lymphography | 18     | 142     | 6            | 4.05%        |
| HeartDisease | 13     | 150     | 17           | 4.46%        |
| Stamps      | 9      | 325     | 16           | 4.92%        |
| WPBC        | 30     | 367     | 10           | 2.72%        |

FIGURE 4  Results on different benchmarking data sets. We report the AUC-value for different settings of hyperparameters. For every method, 10 different choices of hyperparameters are randomly drawn, whereas for our method, we search for optimal hyperparameters based on Algorithm 2. One can clearly see that the choice of hyperparameters highly affects the detection rate of outliers.

4  RESULTS

4.1  Data and benchmark algorithms

In our experiments, we use different data sets that are publicly available. The data sets are taken from the collection of benchmarks published by Campos et al. (2015). The details of the data sets are listed in Table 1. We test different scenarios. The first scenario is the pure detection of outliers without modifying the data set. The second scenario simulates the detection of outliers when features are present that have no relevance for outlier detection. The modelling of irrelevant features is based on the work of Zimek et al. (2012). They keep some meaningful features for outlier detection and fill the remaining features with values of a uniform distribution. We follow this setup in our work. We take the data set with its full features and then add more features. We fill these features with values that follow a uniform or normal distribution in equal parts. Each time we extend the dimension of the data set with irrelevant features, we measure the detection rate of outliers.

4.2  Results without artificial features

First, we look at the results without additional features. In Figure 4, 10 different runs are shown, with different hyperparameters selected randomly for each run. For KNN, the number of nearest neighbours $k$ was varied, as well as for the LOF. For the isolation forest, the number of trees available to isolate the points is varied. The one-class SVM (OC-SVM) takes different values for the kernel bandwidth. The kernel used is the radial basis function. The autoencoder varies the number of hidden layers and, thus, the dimension in the bottleneck. Our method does not vary the hyperparameters but searches for the hyperparameters as described in Algorithm 2 initialized with different seeds.

It can be seen that the detection rate of outliers for many methods depends on how the hyperparameters are chosen. Especially when outliers are detected using the $k$-nearest neighbours, it is noticeable that the variance of the results is relatively small. This can be explained by the fact that larger $k$ values lead to more robust estimates of the distance. The distance to the nearest neighbour is exposed to a larger variance than the distance when choosing larger $k$. Although the LOF is likewise based on distances, the additional normalization of densities that the method

2 Link to download: https://www.dbs.ifi.lmu.de/research/outlier-evaluation/DAMI/
The additional features reduce the ability to detect outliers in most of the cases. Our method, due to the self-supervised strategy to detect outliers, remains robust in the ability to detect outliers. The experiments were repeated 10 times, and the median of the runs is shown here.

The ability of the different methods to find outliers in this setup decreases as the number of noise features increases. It can be observed that the different methods suffer differently in their ability to identify outliers. In the setup without artificially added features, the different methods have approximately similar levels of outlier detection performance. This is independent of the fact that the methods are based on different assumptions: the autoencoder works with dimensionality reduction, while KNN and LOF detect outliers using nearest neighbours.

The ability of the different methods to find outliers in this setup decreases as the number of noise features increases. It can be observed that the different methods suffer differently in their ability to identify outliers. In the setup without artificially added features, the different methods have approximately similar levels of outlier detection performance. This is independent of the fact that the methods are based on different assumptions: the autoencoder works with dimensionality reduction, while KNN and LOF detect outliers using nearest neighbours.

On the one hand, the accuracy of the outlier detection in this noisy setup is dependent on which method is used. On the other hand, it also has an influence on how the data is formed: the behaviour of the methods is also different for different data sets. This can be seen from the two data sets, HeartDisease and Lymphography. While the detection rate on HeartDisease gets worse for all methods, it is especially the OC-SVM on Lymphography that rapidly shows an undesired behaviour in the detection of outliers. The isolation forest also suffers from the additional dimensions, but less severely. Other methods show robust behaviour towards the additional noise in the data. Our method benefits from the design to detect outliers: due to the self-supervised learning, the additional noisy features do not reduce the ability to detect outliers. This is
because the noisy features do not contain information about the clusters in the data set. Consequently, our method clearly outperforms the competitors in this setup, even on the challenging data set WPBC.

5 | CONCLUSIONS

In this article, we present a method to transform unsupervised outlier detection into a supervised problem. The classifier from the supervised problem is suitable to make predictions about the outlier score of points. Furthermore, we present an algorithm to determine the hyperparameters for the classifier appropriately.

In benchmarking with other state-of-the-art methods for tabular data, our method provides at least comparable results. In the relevant case, when noninformative features are available, for example, big data applications, the strengths of the proposed method become apparent. Due to the self-supervised learning, the classifier is able to identify common structures in the data. This makes it robust against features that do not provide information about clusters in the data. In the benchmark with other methods under the presence of noninformative features, our method is clearly superior. We have made the implementation of our code publicly available and published it as an installable package on Python’s package index.

The data that support the findings of this study are openly available on GitHub at https://github.com/JanDiers/self-supervised-outlier. It is taken from benchmarks used by Campos et al. (2015) and may also been accessed here: https://www.dbs.ifi.lmu.de/research/outlier-evaluation/DAMI/. Campos et al. (2015) preprocessed the data according to the steps described in their paper. Originally, the data are taken from the UCI Machine Learning Repository at http://doi.org/10.17616/R3T91Q.

ACKNOWLEDGEMENT

Open access funding enabled and organized by Projekt DEAL.

ORCID

Jan Diers  https://orcid.org/0000-0002-0900-3801

REFERENCES

Abe, N., Zadrozny, B., & Langford, J. (2006). Outlier detection by active learning. KDD ’06.
Aggarwal, C. (2013a). Outlier analysis. New York: Springer.
Aggarwal, C. (2013b). Outlier ensembles: Position paper. SIGKDD Explorations, 14, 49–58.
Aggarwal, C., & Sathe, S. (2015). Theoretical foundations and algorithms for outlier ensembles. SIGKDD Explorations, 17, 24–47.
An, J., & Cho, S. (2015). Variational autoencoder based anomaly detection using reconstruction probability. Special Lecture on IE, 2(1), 1–18.
Angiulli, F., & Pizzuti, C. (2002). Fast outlier detection in high dimensional spaces. In European conference on principles of data mining and knowledge discovery, Springer, pp. 15–27.
Beyer, K., Goldstein, J., Ramakrishnan, R., & Shaft, U. (1999). When is “nearest neighbor” meaningful? In International Conference on Database Theory, Springer, pp. 217–235.
Breiman, L., Friedman, J. H., Olshen, R. A., & Stone, C. J. (1984). Classification and regression trees. Biometrics, 40(3), 874. https://doi.org/10.2307/2530946
Breunig, M., Kriegel, H., Ng, R., & Sander, J. (2000). LOF: Identifying density-based local outliers. SIGMOD ’00.
Campos, G. O., Zimek, A., Sander, J., Campello, R., Micenková, B., Schubert, E., ..., & Houle, M. E. (2015). On the evaluation of unsupervised outlier detection: Measures, datasets, and an empirical study. Data Mining and Knowledge Discovery, 30, 891–927.
Candès, E., Li, X., Ma, Y., & Wright, J. (2011). Robust principal component analysis? Journal of the ACM (JACM), 58, 11:1–11:37.
Chalapathy, R., Menon, A., & Chawla, S. (2018). Anomaly detection using one-class neural networks. ArXiv, abs/1802.06360.
Chandola, V., Banerjee, A., & Kumar, V. (2009). Anomaly detection: A survey. ACM Computing Surveys, 41, 15:1–15:58.
Chen, J., Sathe, S., Aggarwal, C., & Turaga, D. (2017). Outlier detection with autoencoder ensembles. In Proceedings of the 2017 SIAM International Conference on Data Mining, SIAM, pp. 90–98.
Doersch, C., Gupta, A., & Efros, A. A. (2015). Unsupervised visual representation learning by context prediction. In Proceedings of the IEEE International Conference on Computer Vision, pp. 1422–1430.
Erfani, S., Rajasegarar, S., Karunasekera, S., & Leckie, C. (2016). High-dimensional and large-scale anomaly detection using a linear one-class SVM with deep learning. Pattern Recognition, 58, 121–134.
Gidaris, S., Singh, P., & Komodakis, N. (2018). Unsupervised representation learning by predicting image rotations. arXiv preprint arXiv:1803.07728, abs/1803.07728.
Hautamaki, V., Karkkainen, I., & Franti, P. (2004). Outlier detection using k-nearest neighbour graph. In Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004., 3, IEEE, pp. 430–433.
Hendrycks, D., Mazeika, M., & Dietterich, T. G. (2019). Deep anomaly detection with outlier exposure. ArXiv, abs/1812.04606.
Hendrycks, D., Mazeika, M., Kadavath, S., & Song, D. (2019). Using self-supervised learning can improve model robustness and uncertainty. In Advances in Neural Information Processing Systems, pp. 15663–15674.
Hodge, V., & Austin, J. (2004). A survey of outlier detection methodologies. Artificial Intelligence Review, 22, 85–126.
Hoffmann, H. (2007). Kernel PCA for novelty detection. Pattern Recognition, 40, 863–874.
Isola, P., Zoran, D., Krishnan, D., & Adelson, E. H. (2015). Learning visual groups from co-occurrences in space and time. ArXiv, abs/1511.06811.
Knorr, E., & Ng, R. (1997). A unified notion of outliers: Properties and computation. In KDD, 97, pp. 219–222.
Kriegel, H., Kröger, P., Schubert, E., & Zimek, A. (2009a). LoOP: Local outlier probabilities. In Proceedings of the 18th ACM Conference on Information and Knowledge Management, pp. 1649–1652.
Kriegel, H., Kröger, P., Schubert, E., & Zimek, A. (2009b). Outlier detection in axis-parallel subspaces of high dimensional data. In Pacific-Asia Conference on Knowledge Discovery and Data Mining, Springer, pp. 831–838.
Lazarevic, A., & Kumar, V. (2005). Feature bagging for outlier detection. In Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining, pp. 157–166.
Liu, F., Ting, K., & Zhou, Z. (2008). Isolation forest. In 2008 Eighth IEEE International Conference on Data Mining, IEEE, pp. 413–422.
Liu, X., van de Weijer, J., & Bagdanov, A. D. (2019). Exploiting unlabeled data in CNNs by self-supervised learning to rank. IEEE Transactions on Pattern Analysis and Machine Intelligence, 41(8), 1862–1878.
Müller, E., Schiffer, M., & Seidl, T. (2011). Statistical selection of relevant subspace projections for outlier ranking. In 2011 IEEE 27th international conference on data engineering, IEEE, pp. 434–445.
Noroozi, M., & Favaro, P. (2016). Unsupervised learning of visual representations by solving jigsaw puzzles. In European Conference on Computer Vision, Springer, pp. 69–84.
Perera, P., & Patel, V. (2019). Learning deep features for one-class classification. IEEE Transactions on Image Processing, 28, S450–S463.
Raina, R., Battle, A., Lee, H., Packer, B., & Ng, A. (2007). Self-taught learning: Transfer learning from unlabeled data. In Proceedings of the 24th International Conference on Machine Learning, pp. 759–766.
Ramasesh, R., Rastogi, R., & Shim, K. (2000). Efficient algorithms for mining outliers from large data sets. In Proceedings of the 24th International Conference on Management of Data, pp. 427–438.
Ruff, L., Göritz, N., Deecke, L., Siddiqui, S., Vandermeulen, R. A., Binder, A., ... & Kloft, M. (2018). Deep one-class classification. In International Conference on Machine Learning, pp. 4393–4402.
Zimek, A., & Yairi, T. (2014). Anomaly detection using autoencoders with nonlinear dimensionality reduction. In Proceedings of the MLSDA 2014 2nd Workshop on Machine Learning for Sensory Data Analysis, pp. 4–11.
Schölkopf, B., Platt, J. C., Shawe-Taylor, J., Smola, A., & Williamson, R. (2001). Estimating the support of a high-dimensional distribution. Neural Computation, 13(7), 1443–1471.
Tax, D., & Duin, R. P. (2004). Support vector data description. Machine Learning, 54(1), 45–66.
Vasilev, A., Golkov, V., Lipp, I., Sgarlata, E., Tomassini, V., Jones, D., & Cremers, D. (2018). q-space novelty detection with variational autoencoders. arXiv preprint arXiv:1806.02997.
Zhou, C., & Paffenroth, R. C. (2017). Anomaly detection with robust deep autoencoders. In Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 665–674.
Zimek, A., Campello, R., & Sander, J. (2014). Ensembles for unsupervised outlier detection: challenges and research questions a position paper. ACM SIGKDD Explorations Newsletter, 15(1), 11–22.
Zimek, A., Schubert, E., & Kriegel, H. (2012). A survey on unsupervised outlier detection in high-dimensional numerical data. Statistical Analysis and Data Mining: The ASA Data Science Journal, 5(5), 363–387.
Zong, B., Song, Q., Min, M. R., Cheng, W., Lu, X., Cho, D., & Chen, H. (2018). Deep autoencoding Gaussian mixture model for unsupervised anomaly detection. In International Conference on Learning Representations.

How to cite this article: Diers J, Pigorsch C. Self-supervised learning for outlier detection. Stat. 2021;10:e322. https://doi.org/10.1002/sta4.322