Information Sciences 584 (2022) 433–449

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

A probabilistic generalization of isolation forest

Mikhail Tokovarov *, Paweł Karczmarek

Department of Computer Science, Lublin University of Technology, ul. Nadbystrzycka 36B, 20-618 Lublin, Poland

ARTICLE INFO

Article history:
Received 7 May 2021
Received in revised form 27 October 2021
Accepted 30 October 2021
Available online 7 November 2021

Keywords:
Anomaly detection
Isolation forest
Probabilistic generalization of isolation forest
Optimal division
Spatio-temporal datasets

ABSTRACT

The problem of finding anomalies and outliers in datasets is one of the most important challenges of modern data analysis. Among the commonly dedicated tools to solve this task one can find Isolation Forest (IF) that is an efficient, conceptually simple, and fast method. In this study, we propose the Probabilistic Generalization of Isolation Forest (PGIF) that is an intuitively appealing and efficient enhancement of the original approach. The proposed generalization is based on nonlinear dependence of segment-cumulated probability from the length of segment. Introduction of the generalization allows to achieve more effective splits that are rather performed between the clusters, i.e. regions where datapoints constitute dense formations and not through them. In a comprehensive series of experiments, we show that the proposed method allows us to detect anomalies hidden between clusters more effectively. Moreover, it is demonstrated that our approach favorably affects the quality of anomaly detection in both artificial and real datasets. In terms of time complexity our method is close to the original one since the generalization is related only to the building of the trees while the scoring procedure (which takes the main time) is kept unchanged.

1. Introduction

Detecting anomalies in data, often referred to as outliers, is a very important task that has been present in the scientific literature and everyday work of practitioners for several decades. Particularly, in recent years, it is extremely hard task due to increasing volume of data as well as their frequent low quality related to measurement errors, human factor, errors in the operation of IT systems, but also attempts to break the security procedures of electronic and physical security systems. Some anomalies in the data such as their absence or unrealistic measurement values, for instance, negative water temperature, are not difficult to find using simple scripts and can be easily excluded from consideration. However, there are also frequent attempts of phishing identity, access to accounts or passwords, etc. which are almost 100% similar to the actual data. An additional hindering factor is the frequent multidimensionality of data and even their heterogeneity. Data may take various forms such as typical datasets, images, videos, GPS positions, travel trajectories, time series, and many others.

The above-mentioned factors make the literature on the subject rich not only due to the variety of methods but also due to their applications. We will recall some important results here. There are many classes of algorithms used in practice such as support vector machines or core vector machine [1,2], deep neural networks [3–6] such as long-short term memory, self-organizing maps, autoencoders, clustering [7], various approaches to DBSCAN algorithm [8], and granular models or fuzzy set-based methods [9–13], in particular in an application to time series [14,15]. Another methods, particularly used to time series analysis, are random weight networks [16], immune systems [17], weighted graph representation [18], Bayesian net-

* Corresponding author.
E-mail address: p.karczmarek@pollub.pl (M. Tokovarov).

https://doi.org/10.1016/j.ins.2021.10.075
0020-0255/

works [19,20], scan statistics based on expectations [21], and many others. Finally, let us mention density-based methods that are very classic approaches such as *k*-nearest neighbor classifier-based results [22] and Isolation Forest [23,24]. The latter is based on training and traversing special forest of binary search trees. The trees are trained on a subset of original dataset. Anomaly scores depend on the length of tree paths traversed by each of the dataset records. It is worth noting that Isolation Forest is often one of a few methods recommended by practitioners.

Isolation Forest is still attracting a great interest of the researchers dealing with the issues related to outlier detection. One can easily find up-to-date examples of applications of Isolation Forest for credit card fraud detection [25], analysis of anomaly detection algorithms [26], and malware analysis [27]. There are also a few enhancements of the classic approach. Namely, in [28] the lines not necessarily parallel to the Cartesian axes and dividing the subsequent groups of data were proposed. In [29] a so-called CBiForest was described. It is based on a preprocessing of the data using *K*-Means [30] to find two input datasets considered separately. The smaller of them is considered as anomaly set. In [31] the authors present a method well suited for detecting both scattered and clustered anomalies. Its main idea is based upon generating split value with application of special technique of finding a hyperplane separating the data in the best way in the sense of specially constructed splitting criterion. What is interesting, the hyperplanes are not necessarily perpendicular to the axis, but can rather have arbitrary angle. The method exhibits better performance, compared to original iForest, especially in the case of clustered anomalies. However, the cost of training time is higher. Various novel anomaly scores were discussed in [32]. The authors marginalized randomness impact onto binary search trees. The binary trees were replaced by *n*-ary search trees with a new version of anomaly score in [33]. *K*-means and well-known elbow rule [34] was also utilized to find optimal number of tree nodes at each tree split in [35]. Such approach shortens the time of traversing the forest of trees and improves the accuracy of the method. Moreover, the number of divisions reflects the structure of data minimizing the randomness factor. Also fuzzy sets were applied to modify the Isolation Forest algorithm on a basis of membership of the records to the clusters obtained in the process of trees training [36]. A survey of anomaly detection methods can be found, among others in [37–40].

Despite many different attempts to improve the Isolation Forest algorithm, there is still no method that would unambiguously answer where to perform a random or pseudo-random division at the stage of building trees based on samples. In particular, the authors of this manuscript are not aware of any published method that would effectively minimize the possibility of creating such divisions (splits) within a group of closely spaced points that are known to be definitely not anomalies or points that are in such clusters, but close to their boundaries. At the same time, such a desired method should be able to find these intersection points with some influence of randomness. It seems that methods based on probability, in particular, inverse cumulative distribution function, may be a good solution of the problem.

The main goal of this study is to propose and verify a novel generalization of the Isolation Forest algorithm that is to enhance the effectiveness of outlier detection and preserve strong sides of the original approach, i.e., its stability and low computational complexity. We call our approach Probabilistic Generalization of Isolation Forest (PGIF). The proposed generalization is based upon an innovative algorithm that, to the best of the authors' knowledge, has not been applied to Isolation Forest before. We assign different probabilities to various regions of the explored space by building an empirical distribution of probability density on the basis of the training data. We construct the function in the form of piecewise defined probability density function. The function is defined on the separate segments between neighboring points of the training dataset. Due to useful analytical properties we use Kernel Density Estimation functions. The function is built thus that the probability cumulated on a segment is proportional to its length raised to *k*-th power. We describe the idea in Section 3 in more detail.

The proposed generalization is based upon the assumption that not every split of the dataset in course of building an isolation tree is equally profitable. Namely, a split through a gap, separating clusters or outliers, is more profitable as it allows to isolate outliers earlier, making the path in the tree shorter. On the other hand, the stability of Isolation Forest is based upon random building of the Isolation Trees. Therefore, randomness of the process of building an Isolation tree should be preserved but the cluster regions have to be assigned with lower probability density compared to the out-of-cluster space where the anomalies are supposed to be located. In such a manner we ensure more meaningful splitting of the data leading to earlier separation of outliers. The results of the numeric experiments clearly show that the proposed modifications allow to achieve higher performance in the majority of test cases.

The paper is structured as follows. In Section 2 we recall the Isolation Forest algorithm. The details of building an empirical probability density function as well as computational complexity analysis are presented in Section 3. Section 4 is devoted to the presentation of the results of numerical experiments and the discussion. Finally, conclusions and possible directions of future work are presented in Section 5.

2. Isolation Forest

In this section, the main principles of the Isolation Forest algorithm are recalled. In order to detect outliers in data, a forest of randomly built isolation trees is created. Each tree is grown on the basis of independently taken sample which is a subset of the analyzed dataset. The trees are called isolating as they recursively split the analyzed space until the separate points are isolated in its leaves or the depth limit is reached. Outlier detection is performed by passing the data to the grown forest and producing an anomaly measure for every data point. The measure is obtained with the formula (1) [24].

$$s(x,n) = 2^{\frac{E(h(x))}{C(n)}} \tag{1}$$

where

$$C(n) = 2H(n-1) - \frac{2(n-1)}{n} \tag{2}$$

$$H(n) = \ln(n) + 0.57772156649 \tag{3}$$

In the formulae (1), (2) and (3) the $E(x)$ term stands for the average lengths of all the paths that have to be gone through in the isolating trees in order to isolate the point x. $C(n)$ is theoretical average path length of unsuccessful search in the binary search tree.

The original algorithm of Isolation Forest building as well as the method of computing the anomaly score stays unchanged, so a particularly interested reader can refer to the original paper. However, it is the algorithm of building an isolation tree, what is the object of extension in the present paper. Therefore, we recall it in the form of pseudocode, see Algorithm 1 [24].

Algorithm 1 :$iTree(X, e, l)$

Inputs: X – input data, e – current tree height, l – height limit
Output: iTree

1. **if** $e \geq l$ or $|X| \leq 1$ **then**
2. return $exNode\{Size \leftarrow |X|\}$
3. **else**
4. let Q be a list of attributes in X
5. randomly select an attribute $q \in Q$
6. $p \leftarrow uniformRandomReal(from \leftarrow \min(X_q), to \leftarrow \max(X_q)$
7. $X_l \leftarrow filter(X_q \leq p)$
8. $X_r \leftarrow filter(X_q > p)$
9. return $inNode\{Left \leftarrow iTree(X_l, e+1, l),$
10. $Right \leftarrow (X_r, e+1, l),$
11. $SplitAtt \leftarrow q, SplitValue \leftarrow p\}$
12. **end if**

3. Proposed generalization

3.1. General idea of probabilistic generalization

The algorithm (see, Algorithm 1) of building the isolation tree employs uniform distribution for generating a split point p in every node as shown in the line 6. We are interested in ensuring the data splitting in more meaningful way by assigning lower probability density to the densely populated regions, i.e., clusters understood as the groups of records or points in a multidimensional space. Out-of-cluster regions, on the contrary, have to be assigned with higher probability density. Such approach is justified by an intuitively appealing statement that an outlier is a point that is separated from the majority of the data by some empty region. Obviously, one may assume that the same mechanism is also applicable on very lower levels of splits when only a few records are considered.

Fig. 1 illustrates the approach, the part (A) presents a 256-point artificial training set sampled for building the isolation tree. Parts (B) and (C) present original approach (with uniform distribution), and the idea of the modified algorithm, respectively. The black dots at figures (B) and (C) show the distribution of the sample data along the Feature 1 axis. Note that the Feature 1 corresponds to the x-axis while Feature 2 represents the y-axis values, i.e., the first and the second dimension of the data. Casting the data onto the dimension of the Feature 1, we can identify a gap between the clusters at approximately 0.5 and a clearly distinguishable outlier at 0.8. In case of uniform distribution, shown at the left figure, it is equally probable that a split is performed at any region.

Fig. 1 (C) shows the probability density distribution built as piecewise defined function. The value of the function on segments is proportional to their lengths. Here, we can see that the probability density at the clusters being densely populated regions is notably lower compared to the gap in-between the clusters at 0.5 and between a cluster and the outlier at 0.8. Integrated value of the density probability function for top-5 longest segments is equal to 0.5. Even the longest segment alone, i.e. the gap separating the outlier at 0.8 from the rest of the data obtains 0.25 of the total probability weight.

Thanks to the higher probability density it is more likely that an outlier is isolated at the earlier stages of the process of a tree building. Therefore, its anomaly score would be higher.

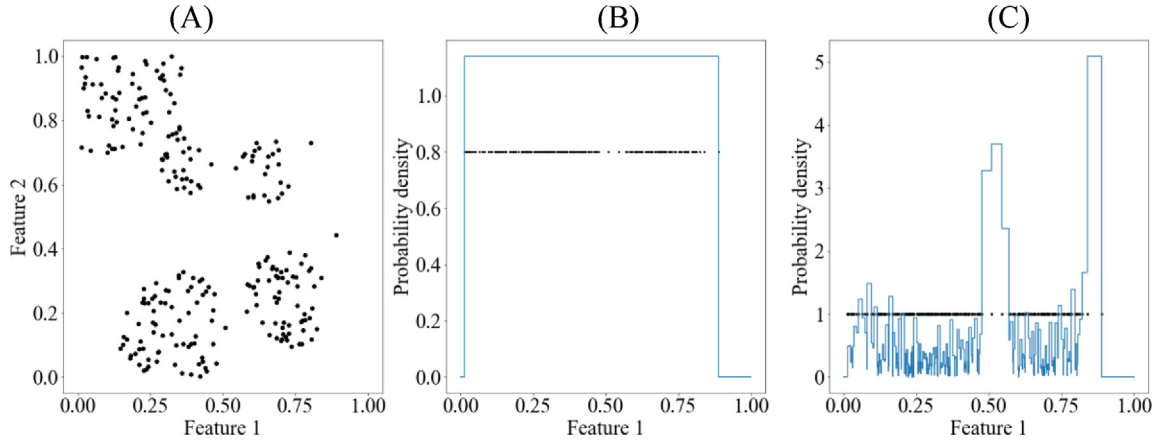

Fig. 1. Illustration of the proposed method. (A) 256-point sample, (B) original approach (uniform distribution of split value generation), and (C) modified approach.

3.2. Piecewise defined cumulated probability function

Here, let us describe the enhancement of Isolation Forest at the step of the choice of splitting parameter. We start from defining a piecewise cumulated probability function for the original Isolation Forest algorithm. Let $f(x)$ be a nonnegative function such that $f(x) = 0, \forall x \notin [a, b), \{a, b\} \subset \mathbb{R}$. Due to useful analytical properties, let it be expressed by scaled and shifted Kernel Density Function $K(x)$ with limited support:

$$f(x) = K\left(\frac{x - \frac{b+a}{2}}{\frac{b-a}{2}}\right) \tag{4}$$

It is worth noting that an unshifted and unscaled Kernel Density Function with limited support fulfills the requirements (5)-(8).

$$K(u) \geq 0 \forall u \in \mathbb{R} \tag{5}$$

$$K(u) = K(-u) \tag{6}$$

$$\int_{-\infty}^{\infty} K(u)du = 1 \tag{7}$$

$$K(u) = 0, \forall u \notin (-1, 1) \tag{8}$$

Since the support is limited then (7) can be expressed by the formula (9).

$$\int_{-\infty}^{\infty} K(u)du = \int_{-1}^{1} K(u)du = 1 \tag{9}$$

It is easy to see that by changing the integration interval we can move from $\{-1, 1\}$ to any limits $\{a, b\}$. Rewriting (9) we obtain the integral of $f(x)$ – $F(x)$ in the form of the formula (10).

$$F_{(a,b)}(x) = \int_a^b K\left(\frac{x - \frac{b+a}{2}}{\frac{b-a}{2}}\right)dx = \frac{b-a}{2}\int_{-1}^{1} K(u)du = \frac{b-a}{2} \tag{10}$$

Let X be increasing sequence of real numbers containing n elements. For any pair of its elements x_i and x_{i+1}, $i \in [1, n-1]$, we can write the formulae (9) and (10) in the form shown in (11) and (12), respectively.

$$f_i(x) = K\left(\frac{x - \frac{x_{i+1}+x_i}{2}}{\frac{x_{i+1}-x_i}{2}}\right) \tag{11}$$

$$F_{x_i,x_{i+1}}(x) = \frac{x_{i+1} - x_i}{2} \tag{12}$$

On the basis of (12), the probability of randomly choosing a split value x_g belonging to the i-th segment reads as the formula (13) shows.

$$P_i = P(x_g \in [x_i, x_{i+1})) = \frac{F_{x_i,x_{i+1}}(x)}{s} \tag{13}$$

Where s is the sum of values of F for all the segments, essential for ensuring that P is bounded above by 1 as shows (14).

$$s = \sum_{i=1}^{n-1} F_{x_i,x_{i+1}}(x) \tag{14}$$

Thus, the probability density function at the i-th segment can be found by the formula (15).

$$p_i(x) = \frac{1}{s} f_i(x) \tag{15}$$

Due to piecewise definition of the probability density function and because of the fact that we use Kernel functions for defining it, the cumulative probability function for generation of x_g can be written with the formula (16).

$$P(x \leq x_g) = \sum_{i=1}^{m-1} P_i + \int_{x_m}^{x_g} p_m(x)dx \tag{16}$$

where $m = \max(j), j \in N, x_j \leq x_g$

3.3. Mathematical formulation of Probabilistic generalization of Isolation Forest

Because of continuity of the segments, in case of uniform $K(\cdot)$, the probability of the x_g belonging to any segment $\{a, b\}$ within $[x_1, x_n]$ depends solely on the distance between a and b as shown in the formula (17).

$$P_i = P(x_g \in [a, b)) = \frac{b-a}{x_n - x_1} \tag{17}$$

It means that in the original Isolation Forest the probability of generating x_g such that the split is performed across a cluster is much higher in comparison to splitting across an inter-cluster gap that is relatively narrower (See Fig. 1).

The basic idea of the proposed method is the assumption that the normal datapoints in clusters are located close to each other. The outliers, on the contrary, are located outside of the clusters and are separated by relatively wide gaps. Therefore, we introduce the generalization expressed by the formula (18), by adding to the formula (11) the dependency yielded from k-th power of a segment's length.

$$f_i(x) = (x_{i+1} - x_i)^k K\left(\frac{x - \frac{x_{i+1}+x_i}{2}}{\frac{x_{i+1}-x_i}{2}}\right) \tag{18}$$

Note that within the light of the proposed generalization the original Isolation Forest method becomes a special case with $k = 0$ and uniform$K(\cdot)$.

The introduced generalization requires the modification of procedure of generating the split value. The first step includes generation of $c \in [0, 1)$, drawn from uniform distribution. x_g has to be generated on the basis of c with the use of inverse cumulative probability function shown by (19).

$$x_g = P^{-1}(c), x_g : P(x \leq x_g) = c \tag{19}$$

In the proposed method the value of x_g can be obtained on the basis of the following formula:

$$c = \sum_{i=1}^{m} P_i + \int_{x_{m+1}}^{x_g} p_{m+1}(x)dx \tag{20}$$

where $m = min(j) - 1,\ j \in N,\ c \leq \sum_{i=1}^{j} P_i$

Using (10) we can express (20) by the formula (21).

$$\frac{(x_{m+2} - x_{m+1})^k}{s} \cdot \int_{x_{m+1}}^{x_g} K\left(\frac{x - \frac{x_{m+2}+x_{m+1}}{2}}{\frac{x_{m+2}-x_{m+1}}{2}}\right)dx = c - \sum_{i=1}^{m} P_i \tag{21}$$

On the basis of the variable change $u = \frac{x - \frac{x_{m+2}+x_{m+1}}{2}}{\frac{x_{m+2}-x_{m+1}}{2}}$ and the formula (10) one can transform (21) to the form of (22).

$$\int_{-1}^{u_g} K(u)du = \frac{\left(c - \sum_{i=1}^{m} P_i\right)}{P_{m+1}} \tag{22}$$

Next, u_g can be obtained as the solution of the equation (23):

$$\kappa(u_g) + \frac{1}{2} - \frac{\left(c - \sum_{i=1}^{m} P_i\right)}{P_{m+1}} = 0 \tag{23}$$

where $\kappa(u)$ is the antiderivative of the applied $K(\cdot)$. In general, some numerical methods, e.g. Newton-Raphson or Bisection, can be used for obtaining u_g. Uniform $K(\cdot)$ is in particular convenient as for this kind of kernel u_g can be found directly as (24) shows.

$$u_g = \frac{2 \cdot (c - \sum_{i=1}^{m} P_i)}{P_{m+1}} - 1 \tag{24}$$

Finally, x_g is obtained with the use of formula (25).

$$x_g = u_g \cdot \frac{x_{m+2} - x_{m+1}}{2} + \frac{x_{m+2} + x_{m+1}}{2} \tag{25}$$

3.4. Algorithm of Probabilistic generalization of Isolation Forest

Our approach extends the Algorithm 1 by modifying the operation described in the line 6 in the way presented in the Algorithm 2.

Algorithm 2: *generateSplitNonUniformly*(X_q, K, k)
Inputs:X_q – input data cast to randomly selected axis q
K – selected kernel
k – power to which the length of segments is raised
Output: x_g – split value $x \in [\min(X_q), \max(X_q)]$

1. $X_{unq} \leftarrow sortedUniqueValues(X_q)$
2. $D \leftarrow neighborDifferences(X_{unq})$
3. $P \leftarrow \left\{D_i^{k+1} \text{ for each } D_i \text{ in } D\right\}$
4. $s \leftarrow sum(P)$
5. $P \leftarrow \{P_i/s \text{ for each } P_i \text{ in } P\}$
6. $i \leftarrow 1$
7. $c \leftarrow uniformRandomReal(from \leftarrow 0, to \leftarrow 1)$
8. **while** $P_i < c$
9. $c \leftarrow c - P_i$
10. $i \leftarrow i + 1$
11. **end while**
12. $u_g \leftarrow invertedCumulativeProbabilityFucntion(K, c/P_i)$
13. **return** $u_g \cdot (x_{m+2} - x_{m+1})/2 + (x_{m+2} + x_{m+1})/2$

The Algorithm 2 allows to obtain a split value generated with probability density function built on the basis of the training sample in the following way:

1. The input data X_q are sorted and unique values are taken, the result is stored in the variable X_{unq}.
2. The lengths of sequent sections are found as the differences between the neighboring datapoints, the result is stored in the variable D.
3. Each distance is raised to (k + 1)-th power, the resulting array is stored in the variable P.
4. The sum of the elements of the array P is computed, the result is stored in the variable s.
5. Each element of P is divided by s in order to ensure the sum of P's elements equal to 1.
6. A random number is drawn from uniform distribution from the interval [0, 1) and stored in the variable c.
7. The counter variable i, indicating the order number of currently considered segment and the cumulated probability assigned to it (P_i).
8. Condition of the loop: the loop body is performed while the value of cumulated probability of the currently considered segment (P_i) is lower than the current value of the variable c containing the random number generated in the line 7.

The loop body:

9. The value of c is decreased by the value of cumulated probability of currently considered segment (P_i).
10. The counter i is decreased.

End of the loop body

11. The value from the interval [−1, 1] corresponding to the value of c is calculated with the use of inverted cumulative probability function found on the basis of applied kernel K, the result is stored in u_g.

12. Since $u_g \in [-1, 1]$, it has to be scaled (multiplied by $(x_{i+1} - x_i)/2$) and shifted (increased by $(x_{i+1} + x_i)/2$), thanks to that we obtain a random value that belongs to the interval $[x_i + x_{i+1}]$ and is generated in accordance with piecewise defined probability density function built on the basis of the training sample.

3.5. Yet another generalization: U-shaped probability density

Another possibly profitable generalization can be the introduction of U-shaped probability density function for generating c instead of uniform distribution as described in the line 7 of the Algorithm 2. The intuition behind the proposed approach is based upon the assumption that anomalies tend to be located on the periphery of the dataset. Hence, slightly higher probability density at the sides can lead to improvement of anomaly detection quality.

Such function can be built by means of any Kernel Density Function as (26) shows.

$$\theta(x) = \begin{cases} \dfrac{1 - c \cdot K(x)}{2 - c}, x \in [-1, 1] \\ 0, \qquad \text{otherwise} \end{cases} \tag{26}$$

If we use a kernel function, we may be sure that $\theta(\cdot)$ is symmetric, nonnegative, if $c \in \left[0, \frac{1}{\max(K(x))}\right]$, and $\int_{-1}^{1} f(x)dx = 1$, thanks to multiplying by $\frac{1}{2-c}$.

Setting $c = 0$ we obtain uniform probability density that is used in the original IF. In case of nonuniform Kernel and $c > 0$ the Algorithm 2 is modified by adding generation of random value on the basis of U-shaped function (26). The procedure consists of two steps. In the first of them a random number $c_u \in [0, 1)$ is drawn from uniform distribution. In the second one, on the basis of c_u the value of c is obtained with the use of inverse cumulative probability function (23), (25).Fig. 2 shows the example of scaled and shifted U-shaped probability density function. It was built with c = 0.5 and trisquare Kernel, expressed by (27).

$$K(x) = \frac{35}{32}\left(1 - u^2\right)^3 \tag{27}$$

3.6. Computational complexity analysis

The time complexity of building an isolation tree in an original Isolation Forest is $O(n\log_2 n)$. Here, we present the estimation of average time complexity of the proposed algorithm. The most costly operation introduced by our modification is sorting, needed to obtain the distances between sequent data points. Assuming that we apply some log-linear sorting, e.g. Mergesort, it is easy to see that additional operations would have the cost equal to $O(n\log_2 n)$ per a level of the built binary tree. The number of levels is equal to $\log_2 n$, so, theoretically, in the worst case the total complexity would increase from $O(nlog_2 n)$ to $O\left(n \cdot \log_2^2 n\right)$ per a tree. The numeric experiments conducted on artificial and real data show that the modification does not cause significant increase in the time cost of training and even lead to decrease of training time in case of extensive datasets.

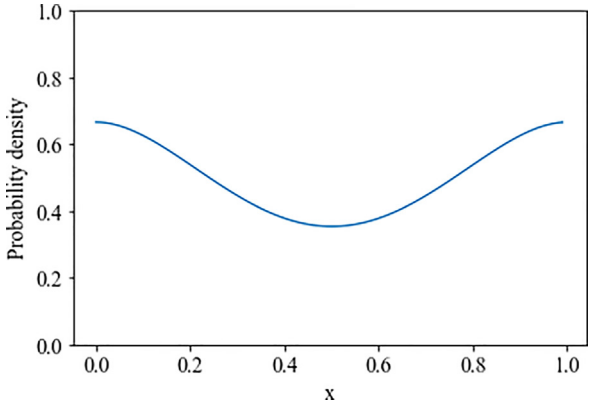

Fig. 2. Example of U-shaped probability density function. c = 0.5.

4. Numerical experiments

4.1. Experiments on artificial data

The implementation of our algorithm utilized selected elements of the publicly available implementation of Isolation Forest published by [41]. The program is written in high-level Python 3 language. In our realization the code underwent major refactoring in order to ensure the module structure convenient for experimenting with various extensions. Namely, specific class TreeGrower, responsible for building an Isolation Tree was defined. All the modifications were implemented as the children of this class, overriding respective methods. The tests were conducted on a laptop personal computer with the following parameters: RAM: 8 Gb, CPU: 4-core 1.8 Hz. The complete implementation together with artificially generated and real datasets utilized in the experiments are available at https://github.com/mtokovarov/probabilistic_generalization_isolation_forest.

In order to compare the performance of the generalized method the following parameters were used for testing on the artificial datasets: Uniform Kernel and $k = 2$.

A series of experiments on artificially generated datasets was conducted. The main goal of the experiments was to ensure testing of the examined models on broad range of datasets of specified configuration. The artificial datasets were generated randomly, but in accordance with the following criteria:

1. Every dataset contained 5 non-intersecting clusters of random radius and outliers located in-between them.
2. Total number of data points in a dataset was equal to 5000 and the outlier percentage was 1% that gave 50 outliers.
3. All the points of a dataset were located in a unite hypercube with the main diagonal $\left((0,0,\cdots,0)^k,(1,1,\cdots,1)^k\right)$, where k is the number of dimensions.
4. Outliers are located not closer than 0.05 (in sense of Euclidian metric) to any cluster.

The ensembles were built with the following parameters:

a) Tree number (t): 100
b) Sample number(n): 256
c) Tree depth limit(l): $\log_2 n = \log_2 256 = 8$

The tests were conducted for the following dimension numbers: {2, 3, 4, 5, 6, 7, 8, 9}. The experiments for these dimension numbers revealed the greatest differences between performance of the basic and modified models. 100 datasets were generated for every dimension, providing in total 800 datasets. ROC AUC was selected to be the quality metric. This choice was made due to the fact that confusion matrix-based metrics require the anomaly score threshold. In most cases it has to be defined by an expert and may be different for various datasets. Moreover, original paper on Isolation Forest [24], that we were basing our research on, also utilized solely AUC metric.

The experiments provided us with extensive results. The average values of AUC for the respective models are presented on the Fig. 3. As it may be observed, the proposed model ensures superior performance compared to the original model for all the numbers of dimensions, particularly for two-, three-, four-, and five-dimensional data.

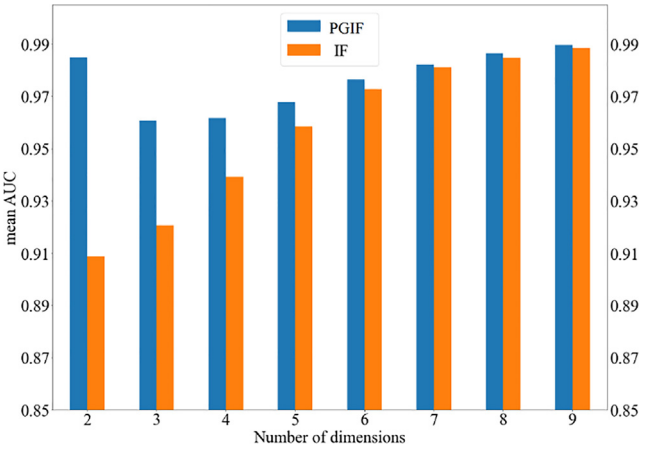

Fig. 3. Mean AUC values for separate models for specified dimension numbers.

Fig. 4 shows the comparison between mean time required for training an original IF and PGIF. It is clearly seen that even in case of the greatest difference the proposed model is less than two times slower in training compared to the original IF on the artificial datasets.

Very interesting insights can be observed from visualization of the anomaly scores of the analyzed datasets. Fig. 5 presents a series of results obtained with example datasets. Each example contains the illustrations of the dataset with the outliers marked red (1), anomaly score assigned to particular points (2), the normalized order numbers obtained after sorting the points according to their anomaly score (3) in ascending order, and heatmap of the anomaly score in the analyzed space (4).

The results presented on the Fig. 5 show that PGIF performs favorably especially in the cases of datasets with extensive clusters. Note how the original IF treats the whole dataset as one large cluster. It can be seen from the fact that the majority of the heatmap for every example of IF is colored with various shades of blue. It means that the low anomaly score was assigned even to the intra-cluster gaps. The generalized model, on the contrary, assigns higher anomaly score to the intra-cluster regions. It can be seen from orange and red taints that can be observed in between clusters.

Original IF tends to assign higher anomaly scores to the points close to the borders of the dataset and lower – to the data at the center, so that the evident outliers surrounded by clusters obtain low anomaly scores that can be noted clearly in the case of the datasets (A) and (B). Although being sparsely populated, this region at the center obtains low anomaly score (it is blue) in the case of IF. Our model (PGIF) finds all the outliers hidden between clusters, assigning high anomaly score to them (they are marked with red). The PGIF algorithm also assigns high anomaly score to the small cluster at the center of the dataset (C). This result is perfectly reasonable as in comparison to the other clusters, the small cluster can be treated as an anomaly. It has lower number of points and occupies notably less space. As for the original approach, the small cluster of the dataset (C) obtains low anomaly score. Together with the two greater clusters at the right side of the dataset, it is treated as one stretched cluster as the heatmap shows. As for quantitative difference, the basic IF algorithm allowed achieving the following AUCs for the respective datasets: 0,857, 0.887, 0.805. PGIF demonstrated notably better performance: 0.978, 0.99, 0.992.

4.2. Experiments on real datasets: Piecewise probability density function

In this section, we present the result of exhaustive experiments including 30 repetitions for every dataset. Such number of repetitions was set in order to obtain stable results. Table 1 presents the basic information regarding the applied benchmark datasets.

Table 2 contains the results obtained with uniform function used for random value generation and $k = 1, 2$. The presented results show that the modified method allows achieving superior results in the majority of cases (8/14), it provided better or not worse results for 12 out of 14 datasets. The original model provided better results only for two datasets (Mammography, Annthyroid). In case of 4 datasets the values of AUC were equal for the both approaches, however, in 3 cases AUCs were fairly close to 1, leaving little room for improvement (Http – 0.994, Shuttle – 0.997, Breastw – 0.988).

4.3. Analysis of the obtained results

An interesting point, is the fact that the original IF method shows superior performance in case of two datasets: Annthyroid and Mammography, see Table 2. It is worth special consideration as it is important not only to have better performance

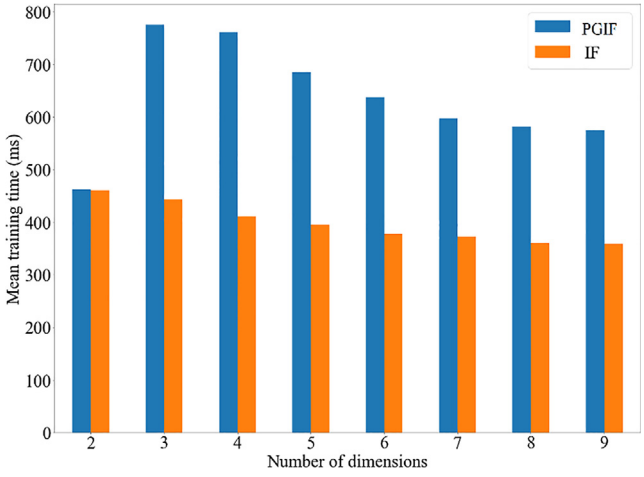

Fig. 4. Comparison of mean training time for various dimension numbers.

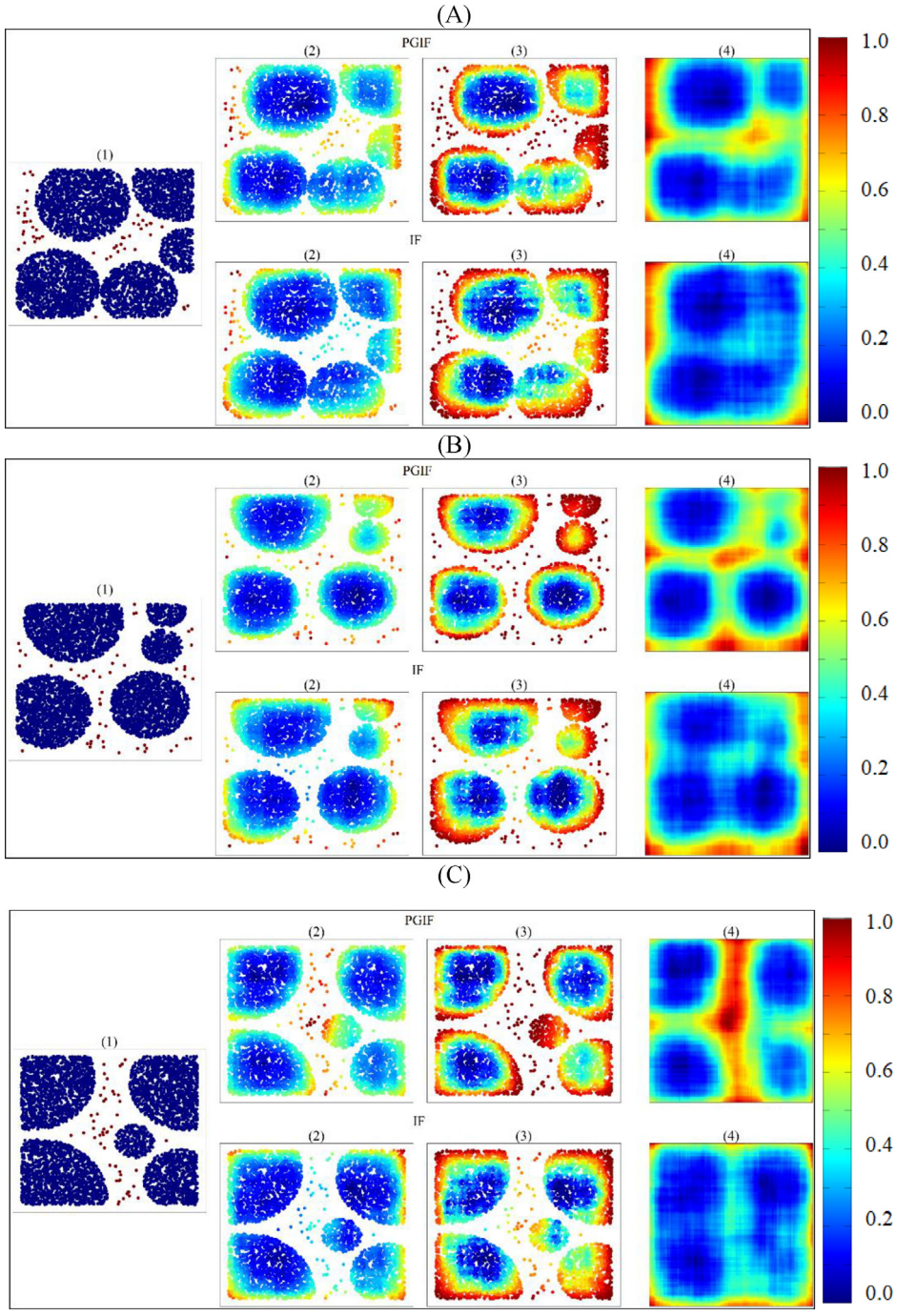

Fig. 5. Comparison of the anomaly scores produced by original IF and PGIF. Every group of 7 figures rounded with a black frame shows the results obtained for a separate artificial dataset. For every row: (1) the analyzed dataset with the outliers marked with red, (2) point anomaly scores, (3) order numbers of the points sorted ascendingly in accordance with their anomaly score, (4) heatmap of the anomaly score values in the analyzed space. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

with specific methods but also to understand why they allow to achieve it. As the authors of the original method state, the main idea behind the Isolation Forest algorithm is the assumption that anomalies are "few and different" [24]. When it comes to space intuition one may formulate this assumption in the following way: "The anomalies are the points that are located far from the inliers in the analyzed feature space". The proposed generalization has its goal in maximizing of this effect. Introducing generalized algorithm we want to ensure that sparsely populated regions of the analyzed feature space are found. The outliers are supposed to populate these regions. Generally speaking, they have to be far from inliers. On the

Table 1
The parameters of the applied benchmark datasets.

Dataset	Number of samples	Anomalies (number/%)	Dimension number
Http KDDCUP99 [42]	58,725	2209 / 3.8	3
ForestCover [43]	286,048	2747 / 1	10
Mulcross [44]	262,144	26,214 / 10	4
Shuttle [43]	49,097	3511 / 7.2	9
Mammography [45]	11,183	260 / 2.3	6
Annthyroid [43]	7200	534 / 7.4	6
Satellite [43]	6435	2036 / 31.6	36
Pima [43]	768	268 / 35	8
Breastw [43]	682	239 / 35	9
Arrhythmia [43]	452	66 / 14.6	274
Ionosphere [43]	351	126 / 36	33
Wine [45]	129	10 / 7.7	13
Credit card [25,42–50]	284,807	492 / 0.17	30
Cardio [42]	1831	176 / 9.6	21

Table 2
Results obtained with the original method as well as with PGIF applying uniform function for random value generation and k = 1, 2. Mean and standard deviation of the AUC values are provided in the following format: Mean AUC (std AUC). The best results in the rows are bold. If two or more models show similarly best results no value is marked.

Dataset	IF	PGIF, k=1	PGIF, k=2
Http KDDCUP99	0.994 (0.001)	0.994 (0.001)	0.992 (0.002)
ForestCover	0.873 (0.024)	0.929 (0.017)	**0.940 (0.014)**
Mulcross	0.958 (0.008)	0.993 (0.002)	**0.995 (0.001)**
Shuttle	0.997 (0.001)	0.997 (0.001)	0.997 (0.001)
Mammography	**0.866 (0.009)**	0.731 (0.019)	0.689 (0.018)
Annthyroid	**0.817 (0.016)**	0.786 (0.018)	0.759 (0.029)
Satellite	0.704 (0.014)	**0.711 (0.014)**	0.700 (0.016)
Pima	0.679 (0.013)	**0.686 (0.009)**	0.672 (0.009)
Breastw	0.988 (0.002)	0.989 (0.001)	0.989 (0.001)
Arrhythmia	0.792 (0.015)	**0.793 (0.014)**	0.791 (0.018)
Ionosphere	0.855 (0.006)	0.873 (0.005)	**0.879 (0.005)**
Wine	0.791 (0.041)	0.838 (0.020)	**0.843 (0.020)**
Credit card	0.948 (0.003)	**0.951 (0.002)**	0.950 (0.003)
Cardio	0.926 (0.011)	0.939 (0.007)	**0.943 (0.008)**

other hand, the inliers, have to be similar to each other. It means that they should tend to be located closer, forming a dense cluster or several clusters.

Now, we can take a look at the real datasets considered. All of them contain highly-dimensional data, so straight graphic examination is not possible. Instead, we will introduce simple intuitive metric. Let the average Euclidean distance between:

- An outlier and the closest respective inlier be the metric, showing how "abnormal" the outliers of the considered dataset be ($\widehat{d}_{out-in}$),
- An inlier and the closest respective inlier to be the metric, showing how "normal" the inliers of the considered dataset be ($\widehat{d}_{in-in}$).

We also have to take into account the fact that with the growth of dimensionality Euclidean distance also grows, so the Euclidean metric has to be modified as the formula (28) shows.

$$d^{*}_{Eucl}(a,b) = \frac{d_{Eucl}(a,b)}{\sqrt{N}} \tag{28}$$

where a and b are the points between which the metric is computed and N is the dimensionality of the space.

Fig. 6 shows the 2D scatter plot on which every point corresponds to a dataset and the axes are the metrics defined above. One can easily notice that exactly the datasets Annthyroid and Mammography on which PGIF did not improve the quality of anomaly detection, show the lowest average value of both metrics. Interpreting the Fig. 6, one may state that the anomalies in the mentioned datasets are "less abnormal", compared to the other datasets in the sense of the introduced metrics. Furthermore, in the case of these datasets the difference between the values of these metrics is notably lower, compared to the other datasets. This phenomenon can be explained in the following way: Probably, the anomalies were marked as such by some expert, who knew that they were abnormal. However, the applied features do not show the difference explicitly (many

abnormal datapoints are located close to inliers). Such datasets as breast, Shuttle, Http, Mulcross, Creditcard, Cover, demonstrate extremely low values of $\widehat{d}_{in-in}$ and comparatively high values of $\widehat{d}_{out-in}$. Because of that fact all the results achieved with PGIF on these datasets are above 0.94 AUC. In some cases they are equally high as for IF (Breast: 0.986, Shuttle:0.997, Http: 0.994). In other datasets they are even better, namely: Cover: 0.873 (IF) – 0.943 (PGIF), Creditcard: 0.948 (IF) – 0.951 (PGIF), Mulcross: 0.958 (IF) – 0.995 (PGIF). In case of such datasets as Pima, Arrhytmia, Wine, Satellite, Cardio, Ionosphere the value of $\widehat{d}_{in-in}$ is not that low. It means that the inliers do not form such dense clusters. Therefore, the task of anomaly detection becomes harder. However, in most cases PGIF allows to improve the performance notably, e.g.: Wine: 0.791 (IF) – 0.843 (PGIF), Satellite: 0.703 (IF) – 0.71 (PGIF), Cardio: 0.926 (IF) – 0.943 (PGIF), Pima: 0.679 (IF) – 0.686 (PGIF), Ionosphere: 0.855 (IF) – 0.878 (PGIF) or ensure similar performance: Arrythmia: 0.792 (IF) – 0.793 (PGIF).

This point is also supported by the Fig. 7, presenting the scatter plots of the first two principal components of the analyzed datasets. One can clearly see that many outliers (marked with orange) are located close to the inlier cluster and are even included in it. Moreover, many inliers (marked with blue) are located notably far from clusters. In this light it is completely reasonable that PGIF demonstrated worse performance – it is designed to effectively find sparcely populated regions and separately located points that are not always anomalies in these datasets. However, the generalized approach still can lead to better results as it is shown in the next section where we use U-shaped probability density function for generating random values.

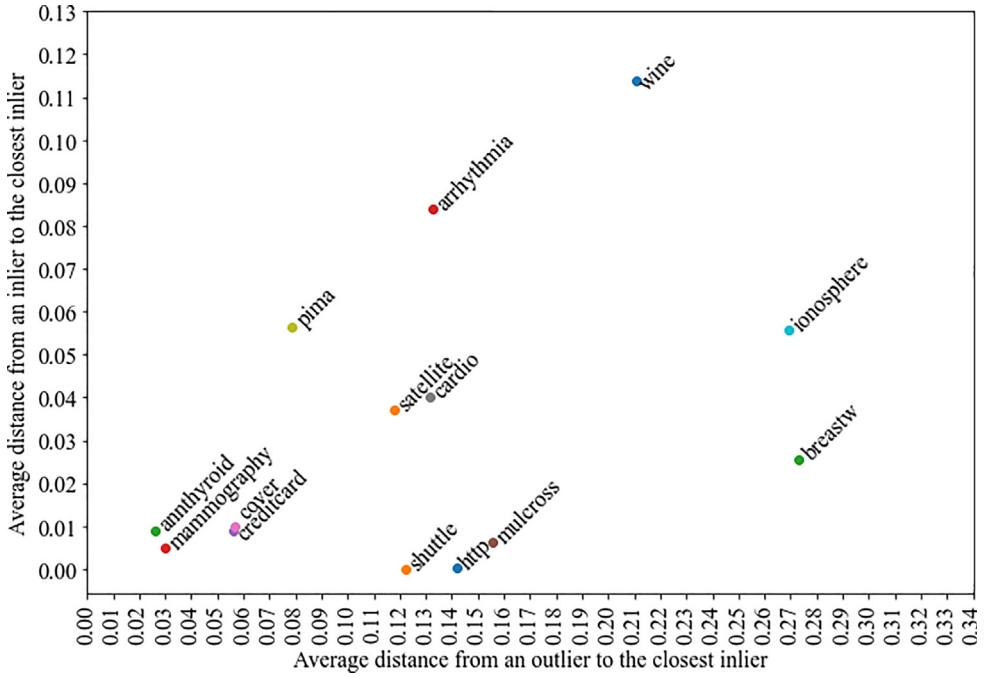

Fig. 6. Average distance between an outlier and the closest inlier for the analyzed datasets.

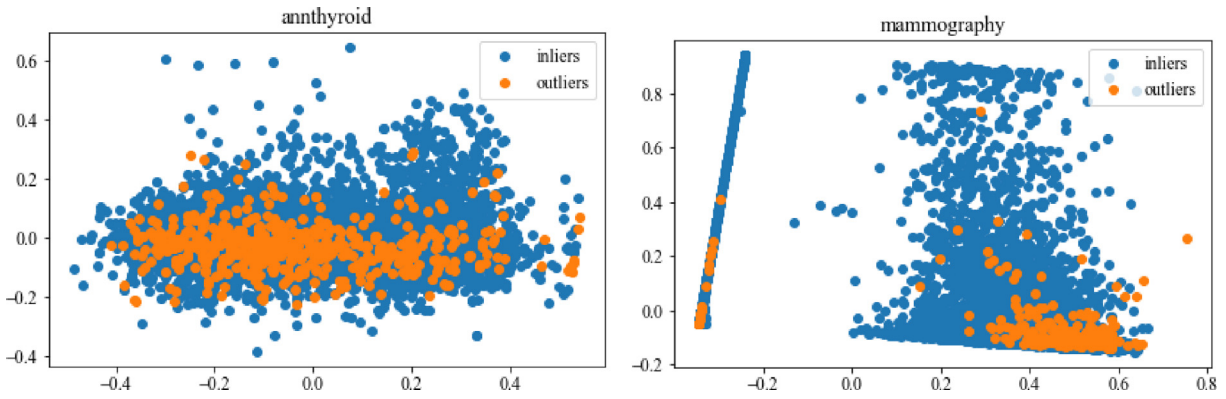

Fig. 7. Average distance between an outlier and the closest inlier for the analyzed datasets.

4.4. Experiments on real datasets: U-shaped probability density

Table 3 shows the results of applying U-shaped probability density function. Interestingly, this solution allowed achieving higher result in case of Annthyroid and Ionosphere datasets.

4.5. Comparison with selected methods of outlier detection

The comparison of results (values of AUC) obtained with our method (PGIF), original Isolation Forest, and selected outlier detection methods are presented in Table 4. Some datasets have N/A in the 1-class SVM column as time complexity is too high. Mean and standard deviation of AUC values are shown for Isolation Forest and its generalization due to the fact that random sample of dataset is used for fitting the forests. Therefore the AUC values can differ throughout runs of the experiment. LOF and 1-class SVM, on the other hand, are deterministic algorithms, analyzing the complete dataset. Due to that fact standard deviation cannot be obtained for these methods.

The comparison between Isolation Forests shows that the generalized model demonstrated superior performance in the majority of cases. It allows to improve the outlier detection performance in the cases, where the other methods were better, ensuring at least not worse or even better results (Ionosphere and Cardio). Taking into account the fact that original Isolation Forest approach can be treated as an individual case of the generalized method, it may be noticed that the family of Isolation Forests ensured better results in 11 out of 14 cases.

4.6. Comparison with existing modifications of Isolation Forest

A series of experiments was run in accordance with previously adopted procedure. The proposed generalization was compared to one of the newest modifications of Isolation Forest, namely Extended Isolation Forest (EIF) [28]. A publicly available implementation of EIF (https://github.com/sahandha/eif) was used in the experiments. The hyperparameters of the Forest, i.e. tree number and sample size were the same (100 and 256). Table 5 presents the comparison of the results obtained with various methods: Original IF, PGIF, and EIF. The columns containing p-values are located between the pairs of columns to which difference they correspond. In order to test statistical significance of the obtained differences non-parametric Mann-Whitney U test was applied. This choice was due to lack of normality in the majority of the obtained results.

The results show that the generalization allows to achieve better results in the majority of cases (8 out of 14). However, in case of only 4 datasets the EIF approach demonstrated better performance.

4.7. Comparison of computation complexity

It is worth stressing that the presented generalization deals only with training process, while the stage of anomaly score computation stays not modified. Therefore, as for the time complexity, theoretically only the training process execution time was affected. As Table 6 shows, in all the cases the average time of training a modified model was less than two times longer. Interestingly, in the case of the large datasets (Mulcross, Credit card, Forest cover) training time of the modified model was even notably shorter (up to 7.3 faster in case of Credit card dataset). As for the total elapsed times, they do not differ notably for the original and modified approaches: The highest difference was registered in case of Ionosphere dataset. However it increased only by 15% (from 767.45 to 911.33).

We suppose that such a drop of training time can be potentially related to the fact that in these datasets the outliers are well separated by gaps from the main part of data, so an isolation tree in PGIF has fairly unbalanced structure. The outliers

Table 3
Results obtained with the original method as well as with PGIF applying U-shaped function for random value generation and k = 0, 1, 2. Mean and standard deviation of the AUC values are provided in the following format: mean AUC (std AUC). The best results in the rows are bold. If two or more models show similarly best results, no value is marked.

Dataset	IF	PGIF, U-shaped, k=0	PGIF, U-shaped, k=1	PGIF, U-shaped, k=2
Http KDDCUP99	0.994 (0.001)	0.993 (0.001)	0.994 (0.001)	0.994 (0.001)
ForestCover	0.873 (0.024)	0.922 (0.020)	0.939 (0.019)	**0.943 (0.023)**
Mulcross	0.958 (0.008)	0.975 (0.005)	0.993 (0.002)	**0.995 (0.001)**
Shuttle	0.997 (0.001)	0.997 (0.001)	0.998 (0.000)	0.998 (0.001)
Mammography	0.866 (0.009)	0.866 (0.009)	0.728 (0.016)	0.694 (0.018)
Annthyroid	0.817 (0.016)	**0.831 (0.019)**	0.796 (0.019)	0.776 (0.028)
Satellite	**0.704 (0.014)**	0.642 (0.014)	0.657 (0.014)	0.678 (0.014)
Pima	**0.679 (0.013)**	0.644 (0.015)	0.661 (0.011)	0.655 (0.011)
Breastw	0.988 (0.002)	0.985 (0.003)	0.986 (0.002)	0.986 (0.002)
Arrhythmia	0.792 (0.015)	0.794 (0.012)	**0.796 (0.017)**	0.788 (0.020)
Ionosphere	0.855 (0.006)	0.841 (0.008)	**0.881 (0.004)**	0.888 (0.005)
Wine	0.791 (0.041)	0.713 (0.043)	0.826 (0.027)	**0.828 (0.024)**
Credit card	0.948 (0.003)	0.948 (0.003)	0.949 (0.003)	0.949 (0.002)
Cardio	0.926 (0.011)	0.932 (0.008)	0.937 (0.009)	**0.939 (0.011)**

Table 4
Comparison of results obtained by selected outlier detection methods, original Isolation Forest, and the proposed modification. The last column contains the notes showing which setup of the proposed method allowed achieving the highest result. The best results in the rows are bold. If two or more models show similarly best results, no value is marked.

Dataset	1-class SVM	LOF (Local Outlier Factor)	IF	The best result of PGIF	Notes
Http KDDCUP99	N/A	0.337	0.994 (0.001)	0.994 (0.001)	–
ForestCover	N/A	0.526	0.873 (0.024)	**0.943 (0.023)**	Uniform, $k = 2$
Mulcross	N/A	0.603	0.958 (0.008)	**0.995 (0.001)**	$k = 2$
Shuttle	0.986	0.524	0.997 (0.001)	0.998 (0.000)	k = 2
Mammography	0.84	0.74	0.866 (0.009)	0.866 (0.009)	U-shaped, $k = 0$
Annthyroid	0.61	0.708	0.817 (0.016)	**0.831 (0.019)**	U-shaped, $k = 0$
Satellite	0.653	0.544	0.704 (0.014)	**0.711 (0.014)**	Uniform, $k = 1$
Pima	0.61	0.598	0.679 (0.013)	**0.686 (0.009)**	Uniform, $k = 1$
Breastw	0.95	0.366	0.988 (0.002)	0.989 (0.001)	all PGIFs
Arrhythmia	0.795	0.789	0.792 (0.015)	**0.796 (0.017)**	U-shaped, $k = 1$
Ionosphere	0.828	0.89	0.855 (0.006)	**0.888 (0.005)**	U-shaped, $k = 2$
Wine	0.678	**0.876**	0.791 (0.041)	0.843 (0.020)	Uniform, $k = 2$
Credit card	N/A	0.626	0.948 (0.003)	**0.951 (0.002)**	Uniform, $k = 1$
Cardio	0.933	0.637	0.926 (0.011)	**0.994 (0.001)**	Uniform, $k = 2$

Table 5
Comparison of results obtained by various modifications of Isolation Forest (original IF, PGIF and Extended IF). p-values characterizing the statistical significance in the pairs (IF-PGIF and PGIF-EIF) are shown between the respective columns. Significantly best results in the rows are marked as bold. If two or more models show similarly best results, no value is marked.

Dataset	IF	p-value (IF-PGIF)	PGIF (best result)	p- value (EIF-PGIF)	EIF
Http KDDCUP99	0.994 (0.001)	0.4	0.994 (0.001)	$1.51 \cdot 10^{-11}$	0.875 (0.012)
ForestCover	0.873 (0.024)	$2.3 \cdot 10^{-10}$	**0.943 (0.023)**	$1.44 \cdot 10^{-10}$	0.867 (0.028)
Mulcross	0.958 (0.008)	$1.5 \cdot 10^{-11}$	**0.995 (0.001)**	$1.51 \cdot 10^{-11}$	0.969 (0.010)
Shuttle	0.997 (0.001)	$3 \cdot 10^{-8}$	**0.998 (0.000)**	$1.51 \cdot 10^{-11}$	0.986 (0.002)
Mammography	0.866 (0.009)	0.4	0.866 (0.009)	$1.84 \cdot 10^{-11}$	0.829 (0.008)
Annthyroid	0.817 (0.016)	$4.5 \cdot 10^{-4}$	**0.831 (0.019)**	$1.51 \cdot 10^{-11}$	0.646 (0.010)
Satellite	0.704 (0.014)	0.05	0.711 (0.014)	0.015	**0.717 (0.009)**
Pima	0.679 (0.013)	0.03	0.686 (0.009)	$1.19 \cdot 10^{-7}$	**0.701 (0.008)**
Breastw	0.988 (0.002)	0.01	**0.989 (0.001)**	$8.46 \cdot 10^{-11}$	0.985 (0.002)
Arrhythmia	0.792 (0.015)	0.09	0.796 (0.017)	0.04	**0.804 (0.010)**
Ionosphere	0.855 (0.006)	$1.51 \cdot 10^{-11}$	0.888 (0.005)	$1.51 \cdot 10^{-11}$	**0.913 (0.006)**
Wine	0.791 (0.041)	$1 \cdot 10^{-7}$	**0.843 (0.020)**	0.03	0.826 (0.034)
Credit card	0.948 (0.003)	$1 \cdot 10^{-4}$	**0.951 (0.002)**	$5.5 \cdot 10^{-11}$	0.943 (0.003)
Cardio	0.926 (0.011)	$7.4 \cdot 10^{-8}$	**0.943 (0.008)**	$2.8 \cdot 10^{-4}$	0.935 (0.008)

Table 6
Comparison of training times required by the original IF and PGIF.

Dataset	Mean elapsed time: overall (training) (ms)	
	IF	PGIF
Http KDDCUP99	41310.59 (215.26)	41881.81 (279.33)
ForestCover	880048.41 (337.22)	893662.79 (185.71)
Mulcross	283355.18 (869.35)	276837.14 (149.8)
Shuttle	50234.12 (371.37)	50752.26 (460.97)
Mammography	10256.92 (241.86)	9933.60 (298.05)
Annthyroid	20240.37 (276.86)	20144.52 (339)
Satellite	6621.77 (424.5)	6786.98 (633)
Pima	977.14 (299.68)	966.52 (350.84)
Breastw	1974.66 (327.07)	2117.25 (514.61)
Arrhythmia	1218.10 (300.63)	1266.45 (351.77)
Ionosphere	767.45 (391.83)	911.33 (523.58)
Wine	326.99 (189.59)	340.99 (290.08)
Credit card	257353.19 (926.62)	260472.13 (127.61)
Cardio	4490.13 (239.26)	4548.80 (310.1)

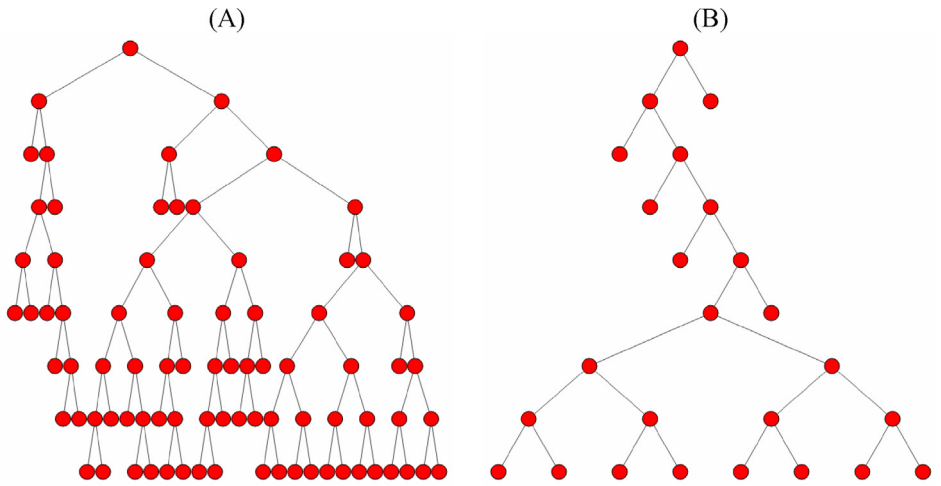

Fig. 8. Example Isolation trees built for the Credit card dataset by the IF (A) and PGIF (B). Red circles represent the nodes of an isolation tree. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

are isolated at high levels of the tree, so that the tree is formed in rather linear structure, producing lower number of nodes. Lower number of nodes leads to less number of time-consuming operations of split value generation. This situation is shown in Fig. 8. The figure can be interpreted in such a way that the peripheral less densely populated regions of the dataset are "stripped off" by early splits resulting in the situation when the tree achieves maximal depth faster thus lowering the computational cost.

5. Conclusions and future work

In this study, we have presented a novel generalization of the Isolation Forest algorithm. As for the original Isolation Forest, it is the conceptual simplicity and low time complexity together with high efficiency that are the main advantages of our approach in comparison to other outlier detection methods. To high extent, the generalization that we have introduced, preserves the strong sides of Isolation Forest related to intuitiveness and time complexity along with further improvement of anomaly detection efficiency. We also propose the idea of the U-shaped probability density function used for generating random values specifically for applications in outlier detection. In the series of numerical experiments our model called Probabilistic Generalization of Isolation Forest (PGIF) has proven its usefulness in detection of anomalies located not only far from other groups of records but also in intra-cluster areas. Also PGIF easily finds distinct clusters when they are located fairly close to each other.

Another notable achievement is the fact that PGIF performs favorably in the case of extensive datasets (Mulcross, Creditcard, Forest cover) both in terms of outlier detection and time complexity of training. In comparison with other algorithms our model demonstrated superior performance for 10 of 14 benchmark datasets. If we consider the original IF as a special case of our generalization, the obtained results are superior for the Isolation Forest family in 11 out of 14 datasets compared to the other methods. Only for one small dataset (Wine) a competing model (LOF) showed better results.

We conclude that the proposed generalization allows to achieve higher performance in many cases also in comparison with other modifications of Isolation Forest such as Extended Isolation Forest (EIF). The fact that EIF in some cases may be the preferable solution shows the path of further research. As PGIF, EIF and IF are based on slightly different principles and provide different results, one can use them in an aggregated ensemble where various techniques of fuzzy logic can be applied in order to obtain yet more promising results.

Future work would also include analysis of the proposed model in various applications and in the framework of Granular Computing and fuzzy techniques related to modifications of relatively non-intuitive measure of anomaly proposed in original version of Isolation Forest. Moreover, it is possible to utilize the aggregation methods and replace the process of summing the anomaly scores resulting from each of the search trees by other operators and to apply weights related to measures of quality of particular search trees.

CRediT authorship contribution statement

Mikhail Tokovarov: Conceptualization, Methodology, Validation, Software, Formal analysis, Writing – original draft, Visualization, Supervision, Project administration. **Paweł Karczmarek:** Conceptualization, Formal analysis, Resources, Writing – original draft, Writing – review & editing, Supervision, Project administration, Funding acquisition.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

Funded by the National Science Centre, Poland under CHIST-ERA programme (Grant no. 2018/28/Z/ST6/00563).

References

[1] B. Schölkopf, J.C. Platt, J. Shawe-Taylor, A.J. Smola, R.C. Williamson, Estimating the support of a high-dimensional distribution, Neural Computation 13 (7) (2001) 1443–1471.
[2] X.-S. Gan, J.-S. Duanmu, J.-f. Wang, W. Cong, Anomaly intrusion detection based on PLS feature extraction and core vector machine, Knowledge-Based Systems 40 (2013) 1–6.
[3] E. de la Hoz, E. de la Hoz, A. Ortiz, J. Ortega, A. Martínez-Álvarez, Feature selection by multi-objective optimisation: Application to network anomaly detection by hierarchical self-organising maps, Knowledge-Based Systems 71 (2014) 322–338.
[4] S.M. Erfani, S. Rajasegarar, S. Karunasekera, C. Leckie, High-dimensional and large-scale anomaly detection using a linear one-class SVM with deep learning, Pattern Recognition. 58 (2016) 121–134.
[5] P. Malhotra, L. Vig, G. Shroff, G., P. Agarwal, Long short term memory networks for anomaly detection in time series, in: European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, 2015, pp. 89-94.
[6] C. Zhou, R.C. Paffenroth, Anomaly detection with robust deep autoencoders, KDD '17 Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Halifax, 2017, pp. 665-674.
[7] R.J.G.B. Campello, D. Moulavi, A. Zimek, J. Sander, Hierarchical density estimates for data clustering, visualization, and outlier detection, ACM Transactions on Knowledge Discovery from Data 10 (1) (2015) article no. 5.
[8] Z. Wu, J. Huang, Application of DBSCAN cluster algorithm in anormaly detection, Journal of Network Security, Computer Networks 8 (2007) 43–46.
[9] W. Chimphlee, A.H. Abdullah, M.N.M. Sap, S. Srinoy, S. Chimphlee, in: Anomaly-based intrusion detection using fuzzy rough clustering, Information Technology, Cheju Island, 2006, pp. 329–334.
[10] J. Gomez, F. Gonzalez, D. Dasgupta, An immuno-fuzzy approach to anomaly detection, in: The 12th IEEE International Conference on Fuzzy Systems, FUZZ '03.,vol. 2. St Louis, 2003, pp. 1219-1224.
[11] X.D. Hoang, J. Hu, P. Bertok, A program-based anomaly intrusion detection scheme using multiple detection engines and fuzzy inference, Journal of Network and Computer Applications 32 (6) (2009) 1219–1228.
[12] C.-H. Tsang, S. Kwong, H. Wang, Genetic-fuzzy rule mining approach and evaluation of feature selection techniques for anomaly intrusion detection, Pattern Recognition 40 (9) (2007) 2373–2391.
[13] A. Kiersztyn, P. Karczmarek, K. Kiersztyn, W. Pedrycz, Detection and Classification of Anomalies in Large Data Sets on the Basis of Information Granules, IEEE Transactions on Fuzzy Systems (2021).
[14] H. Izakian, W. Pedrycz, Anomaly detection in time series data using a fuzzy c-means clustering, in: 2013 Joint IFSA World Congress and NAFIPS Annual Meeting (IFSA/NAFIPS), Edmonton, AB, 2013, pp. 1513-1518.
[15] H. Izakian, W. Pedrycz, Anomaly detection and characterization in spatial time series data: A cluster-centric approach, IEEE Transactions on Fuzzy Systems 22 (6) (2014) 1612–1624.
[16] H. Faris, A.M. Al-Zoubi, A.A. Heidari, I. Aljarah, M. Mafarja, M.A. Hassonah, H. Fujita, An intelligent system for spam detection and identification of the most relevant features based on evolutionary Random Weight Networks, Information Fusion 48 (2019) 67–83.
[17] D. Dasgupta, S. Forrest, Novelty detection in time series data using ideas from immunology, in: 5th Int. Conf. on Intelligent Syst., 1996.
[18] H. Cheng, P. Tan, C. Potter, S. Klooster, A robust graph-based algorithm for detection and characterization of anomalies in noisy multivariate time series, Proc. IEEE Int. Conf. Data Mining Workshops, Pisa, Italy, 2008, pp. 349-358.
[19] E.W. Dereszynski, T.G. Dietterich, Spatio-temporal models for data anomaly detection in dynamic environmental monitoring campaigns, ACM Transactions on Sensor Networks 8 (1) (2011) article no. 3.
[20] D.J. Hill, B.S. Minsker, E. Amir, Real-time Bayesian anomaly detection for environmental sensor data, in: Proc. 32nd Congress of the Int. Assoc. of Hydraulic Eng. and Research, 2007.
[21] D.B. Neill, Expectation-based scan statistics for monitoring spatial time series data, International Journal of Forecasting 25 (3) (2009) 498–517.
[22] F. Angiulli, C. Pizzuti, Fast outlier detection in high dimensional spaces, in: Principles of Data Mining and Knowledge Discovery. Lecture Notes in Computer Science 2431, 2002, pp. 15-26.
[23] F.T. Liu, K.M. Ting, Z.-H. Zhou, Isolation-based anomaly detection, ACM Transactions on Knowledge Discovery from Data 6 (2012) 3.
[24] F.T. Liu, K.M. Ting, Z.-H. Zhou, Isolation forest, Eighth IEEE International Conference on Data Mining 2008 (2008) 413–422.
[25] F. Carcillo, Y. Le Borgne, O. Caelen, Y. Kessaci, F. Oblé, G. Bontempi, Combining unsupervised and supervised learning in credit card fraud detection, Information Sciences 557 (2021) 317–331.
[26] P. Kulczycki, K. Franus, Methodically unified procedures for a conditional approach to outlier detection, clustering, and classification, Information Sciences 560 (2020) 504–527.
[27] N. Shang, A. Wang, Y. Ding, K. Gai, L. Zhu, G. Zhang, A machine learning based golden-free detection method for command-activated hardware Trojan, Information Sciences 540 (2020) 292–307.
[28] S. Hariri, M.C. Kind, R.J. Brunner, Extended isolation forest, IEEE Transactions on Knowledge and Data Engineering 33 (4) (2021) 1479–1489, https://doi.org/10.1109/TKDE.6910.1109/TKDE.2019.2947676.
[29] J. Liu, J. Tian, Z. Cai, Y. Zhou, R. Luo, R. Wang, A hybrid semi-supervised approach for financial fraud detection, 2017 International Conference on Machine Learning and Cybernetics (ICMLC), Ningbo, 2017, pp. 217-222.
[30] J.A. Hartigan, M.A. Wong, A K-Means clustering algorithm, Journal of Applied Statistics 28 (1) (1979) 100–108.
[31] F. T. Liu, K. M. Ting, Z. H. Zhou (2010, September). On detecting clustered anomalies using SCiForest. In Joint European Conference on Machine Learning and Knowledge Discovery in Databases. Springer, Berlin, Heidelberg, 2010, pp. 274-290.
[32] A. Mensi, M. Bicego, A novel anomaly score for isolation forests, in: Image Analysis and Processing – ICIAP 2019. Lecture Notes in Computer Science 11751. Springer, Cham, (2019), pp. 152-163.
[33] P. Karczmarek, A. Kiersztyn, W. Pedrycz, N-ary isolation forest: An experimental comparative analysis," in: L. Rutkowski et al. (Eds.), ICAISC 2020 Proceedings, 2020.
[34] R.L. Thorndike, Who belongs in the family?, Psychometrika 18 (4) (1953) 267–276
[35] P. Karczmarek, A. Kiersztyn, W. Pedrycz, E. Al, K-means-based isolation forest, Knowledge-Based Systems 195 (2020) 105659, https://doi.org/10.1016/j.knosys.2020.105659.
[36] P. Karczmarek, A. Kiersztyn, W. Pedrycz, Fuzzy set-based isolation forest, in: 2020 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), 2020, pp. 1–6.

[37] L. Akoglu, H. Tong, D. Koutra, Graph based anomaly detection and description: a survey, Data Mining and Knowledge Discovery 29 (3) (2015) 626–688.
[38] V. Chandola, A. Banerjee, V. Kumar, Anomaly Detection: A Survey, ACM Computing Surveys (CSUR) 41 (3) (2009) 1–72.
[39] H. Fanaee-T, J. Gama, Tensor-based anomaly detection: An interdisciplinary survey, Knowledge-Based Systems 98 (2016) 130–147.
[40] R.A.A. Habeeb, F. Nasaruddin, A. Gani, I.A.T. Hashem, E. Ahmed, M. Imran, Real-time big data processing for anomaly detection: A Survey, International Journal of Information Management 45 (2019) 289–307.
[41] Matias Carrasco Kind, & Mahdi Sadeghzadeh Ghamsary. (2019, January 31). mgckind/iso_forest: iso_forest 1.0.3 (Version v1.0.3). Zenodo. http://doi.org/10.5281/zenodo.2553679
[42] K. Yamanishi, J.I. Takeuchi, G. Williams, P. Milne, On-line unsupervised outlier detection using finite mixtures with discounting learning algorithms, Data Mining and Knowledge Discovery 8 (3) (2004) 275–300.
[43] A. Asuncion, D. Newman. "UCI machine learning repository." (2007).
[44] D.M. Rocke, D.L. Woodruff, Identification of outliers in multivariate data, Journal of the American Statistical Association 91 (435) (1996) 1047–1061.
[45] A. Dal Pozzolo, O. Caelen, R.A. Johnson, G. Bontempi, Calibrating Probability with Undersampling for Unbalanced Classification, Symposium on Computational Intelligence and Data Mining (CIDM), 2015.
[46] A. Dal Pozzolo, O. Caelen, Y.-A. Le Borgne, S. Waterschoot, G. Bontempi, Learned lessons in credit card fraud detection from a practitioner perspective, Expert Systems with Applications 41 (10) (2014) 4915–4928.
[47] A. Dal Pozzolo, G. Boracchi, O. Caelen, C. Alippi, G. Bontempi, Credit card fraud detection: a realistic modeling and a novel learning strategy, IEEE Transactions on Neural Networks and Learning Systems 29 (8) (2018) 3784–3797.
[48] A. Dal Pozzolo, Adaptive Machine learning for credit card fraud detection ULB MLG PhD thesis (supervised by G. Bontempi)
[49] F. Carcillo, A. Dal Pozzolo, Y.-A. Le Borgne, O. Caelen, Y. Mazzer, G. Bontempi, Scarff: a scalable framework for streaming credit card fraud detection with Spark, Information Fusion 41 (2018) 182–194.
[50] F. Carcillo, Y.-A. Le Borgne, O. Caelen, G. Bontempi, Streaming active learning strategies for real-life credit card fraud detection: assessment and visualization, International Journal of Data Science and Analytics 5 (4) (2018) 285–300.