Hindawi
Mathematical Problems in Engineering
Volume 2018, Article ID 2318763, 13 pages
https://doi.org/10.1155/2018/2318763

Research Article

An Optimized Computational Framework for Isolation Forest

Zhen Liu,[1,2] **Xin Liu**,[1] **Jin Ma**,[1] **and Hui Gao**[1,2]

[1] *Web Sciences Center, School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China*
[2] *Big Data Research Center, School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China*

Correspondence should be addressed to Zhen Liu; quake.liu0625@gmail.com

Received 28 February 2018; Revised 9 April 2018; Accepted 26 April 2018; Published 11 June 2018

Academic Editor: Sotiris B. Kotsiantis

Isolation Forest or iForest is one of the outstanding outlier detectors proposed in recent years. Yet, in the model setting, it is mainly based on the technique of randomization and, as a result, it is not clear how to select a proper attribute and how to locate an optimized split point on a given attribute while building the isolation tree. Aiming to the two issues, we propose an improved computational framework which allows us to seek the most separable attributes and spot corresponding optimized split points effectively. According to the experimental results, the proposed model is able to achieve overall better performance in the accuracy of outlier detection compared with the original model and its related variants.

1. Introduction

In the information society, tremendous data are generated to record the events happened in every second. Few of them are special for they are not yielded as our expectations which can be treated as anomalies. Anomalies exist nearly in every field. For example, events like a spam email, an incidental breakdown of the machine, or a hacking behavior on WWW can be regarded as anomalous in real lives. According to the definition given by Hawkins [1], an anomaly, sometimes called an outlier as well, is far different from the observed normal objects and suspected to be generated from a different mechanism. So, the anomalies should be few and distinct. How to detect them accurately is the main challenge in this field which is commonly called the study of anomaly detection or outlier detection.

In reality, the data annotated with normal or abnormal labels are commonly not available which, if existing, are known to be able to provide the prior knowledge to identify them. Meanwhile, the anomalies in the dataset are commonly rare. It suggests that the numbers of the normal instances and the abnormal ones tend to be heavily unbalanced between them. Due to the two reasons, researchers cannot treat the anomaly detection as a typical problem of classification by simply applying the traditional supervised machine learning methodologies. As a result, in the past decades, most proposed models are unsupervised, including statistical models, distance-based models, density-based models, and tree-based models.

If a dataset with multivariate attributes contains normal and anomalous points, in contrast to the traditional view angles, we consider that the essential distinction between the anomalies and normal instances lies in the discrepant values on each attribute. In other words, that the different value distributions towards normal and anomalous points on each attribute ensure them separable. iForest is one of the successful attempts to realize effective partition on the attribute values based on the method of randomization [2]. Yet, it is not clear how to select the proper split point in the problem settings.

Despite the advantages of the iForest, we consider it still has the space to be improved. For example, the selection of the attribute and determination of the split value regarding the chosen attribute are completely arbitrary while applying the model to build the iForest and some more smart methods would be possible to be studied. Since each tree in the iForest, i.e., iTree, is a binary tree with growth constraint by depth, the leaf node or external node of the tree should play a

role in identifying if an instance is normal or not. Yet, it is not considered in the previous studies. Aiming at the issues mentioned above, we consider proposing a new solution to optimize the tree building process from solving three key questions.

(1) How to select a suitable or distinguishable attribute to partition the instances?

(2) How to determine the appropriate split point on a given attribute? Split point refers to a chosen attribute value to partition the data into two sets on an attribute.

(3) How to mark the leaf node with proper category labels? As there are only two kinds of labels for anomaly detection, we can mark the leaf node with label 1 for normal instance and 0 for the anomaly. For a specified detection process in an iTree, when the step of the decision in the iTree reaches one of the leaf node, it depends on the label of the leaf with 1 or 0 to identify if the instance is normal or not.

The optimized iTree and corresponding improved iForest have four merits including the following:

(1) Via a heuristic searching method based on the gradient, the new model is able to locate the best split point on a given attribute efficiently.

(2) Compared to the extremely unbalanced isolation tree generated by the original model, i.e., iForest, the iTree generated by the proposed model is well balanced.

(3) Compared to iForest, the new model needs fewer isolation trees to constitute the forest.

(4) It has more favorable accuracy performance for outlier detection in terms of AUC results compared with the state-of-the-art methods.

The layout of the article has an organization as follows. In Section 2, related studies in the field of outlier detection will be reviewed. In Section 3, we firstly introduce some symbol definitions to be used in the rest of this paper. Then, the motivation of our study and some quantified analysis with regard to the proposed model will be given in Section 4. Some detailed algorithm descriptions are presented in Section 5. Analysis with extended experiments for comparisons between our model and existing methods is discussed in Section 6. Finally, we give a summary to conclude our work in Section 7.

2. Related Work

Despite extensive surveys have been made for the studies of the outlier detection in the review articles appeared in recent years [3–6], yet, as an active and fast updating community in the domain of data mining, some new insights and trends should be introduced here. In our viewpoint, the state-of-the-art approaches can be categorized as follows.

2.1. Distance-Based Approaches. The distance-based approaches [7] are commonly regarded as the basic methods in the study of outlier detection. The original idea of distance-based approaches is to calculate the distances between a given object and its kth-nearest neighbors and identify the objects as outliers with the largest kNN distance. Some variants are proposed as follow-ups such as *kNN-weight* [8], outlier detection using indegree number (*ODIN*) [9]. *kNN-weight* defines the weight of the object by summing up the distances within its kNN neighbors which is more favorable to tackle the high dimensional data. *ODIN* formalizes the correlations of k nearest neighbors into a kNN graph and detect the nodes as outliers in the graph with low degrees. There are also some other related models using pruning techniques for ranking top-N outliers based on distance [10, 11]. They collectively constitute the family of the distance-based approaches.

2.2. Density-Based Approaches. These approaches introduce a concept of Local Outlier Factor (LOF) [12]; i.e., each instance is assigned a score based on the neighbors' local density denoting a degree of outlierness. A potential outlier is identified by the relative high LOF value. Based on this main idea, some extended models are proposed. INFLO considers using NNs and reverse NNs to calculate the value of LOF [13]. LOCI further introduced the multigranularity deviation factor (MDEF) which uses ε-neighborhoods rather than kNNs [14]. To alleviate the sensitivity of the parameter tuning, an improved model LDOF was proposed which yet has similar performance to the traditional density-based approaches [15]. In the field of handling high dimensional data, some works tried to find the difference between normal instances and anomalous ones in the meaningful subspace of the data space [16, 17].

2.3. Tree-Based Approaches. In recent years, tree-based approaches have obtained comprehensive attentions in the field of outlier detection for its outstanding performance in detection accuracy and scalability. Some typical models include IHCCUDT [18], iForest [2], SCiForest [19], and RRCT [20]. iForest and RRCT both adopt the strategy of the randomization for tree building. RRCT considers the impact of the outliers to the entire data, while iForest merely focus on the cost of the instances isolation. IHCCUDT and SCiForest both considered how to reduce the inconsistency in the interior of the subset for each partition. As for IHCCUDT, it utilizes small number of instances with labels to mark other instances having no labels. SCiForest follows the idea of iForest to identify anomalies by the depth of the leaf nodes and gets an enhanced performance by introducing the random hyperplanes in subspaces.

2.4. Scenario-Aware Approaches. Applying available outlier detection models to a specific problem scenario is also a trend in this field for its visible practical interests. Schubert et al. proposed a unified framework by the means of abstracting the notion of locality from the classic distance-based notion and applied it to the applications of spatial, video, and network outlier detection [21]. Aiming at the communities existing in the network, Gao et al. use a generative model called CODA to detect outliers which are defined as individuals deviating significantly from the rest of the community members [22]. Pokrajac et al. devised an incremental LOF algorithm to perform the outlier detection of data streams [23]. Some special applications can also be found in the literature

Table 1: The notation definitions.

Symbol	Description
X^k	The value set for attribute k
X_a^k	The anomalous value set for attribute k
X_n^k	The normal value set for attribute k
$E(x)$	Mean function
$D(x)$	Variance function
$dist(x, y)$	The distance between x and y
$disp(x, y)$	The dispersity between x and y
$sep(X^k, X_a^k, X_n^k)$	An index to measure the degree of the separability for attribute k
$grad$	Variable of gradient
$step$	A variable to quantify how many points can be skipped
$h(x)$	A function to measure the depth of a leaf node x in the iTree

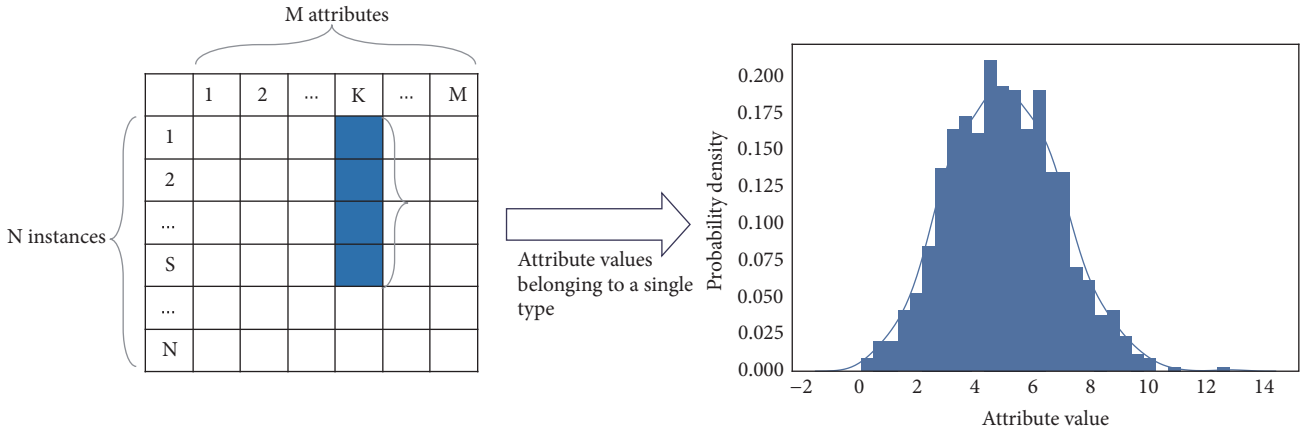


Figure 1: An example of the distribution of probability density on a given attribute's values.

like trajectory outlier detection [24], outlier detection with uncertain data [25], etc.

3. Notation Definitions

Some used symbols in this article are defined in Table 1.

4. The Proposed Solution

4.1. Two Fundamental Assumptions for the Attribute Values. It is wildly accepted that the anomalous instances should be different to the normal ones. Yet how to quantify the difference is an open issue in this field. In this study, we consider that the distributions of the attribute values could be utilized as a basis to investigate the difference between the two kinds of instances. The idea is that the cumulated discrepancy of values distributions on various attributes will be able to tell us how different the anomalous instances and normal ones are. Two assumptions for the data representation are presented as follows:

(1) From a perspective of probability theory, we can use the probability density functions (pdf) to depict the distribution of attribute's values. No matter for normal or anomalous instances, we suppose that the corresponding attribute values for each of them tend to cluster together and have some centralized probability density values. Ideally, the values distribution of probability density with respect to the attribute values for a single class of instances (normal or anomalous) would have only one peak as shown in Figure 1. As a result, the values of probability density deviating from the center are relatively small.

(2) We suppose that the normal and anomalous instances have different clusters of values on a given attribute which implies separability between them. There are commonly two kinds of mixed distributions of probability density as shown in Figure 2. In the left diagram, the distributions for normal instances versus anomalous ones are heavily overlapped so that the integrated distribution has just one peak in probability density. On the contrary, in the right diagram, the two distributions are less overlapped and, as a result, the integrated distribution appears with two peaks. With respect to the degree of separability on the attribute values, the right diagram is apparently better than the left one. Inspired by this observation, we conclude that the separability of an attribute actually is affected by two factors, i.e., the distances between peaks and the dispersity of the values for each category of instances.

4.2. The Study of the Separability on the Attribute's Values. In this section, we try to quantify the separability between the

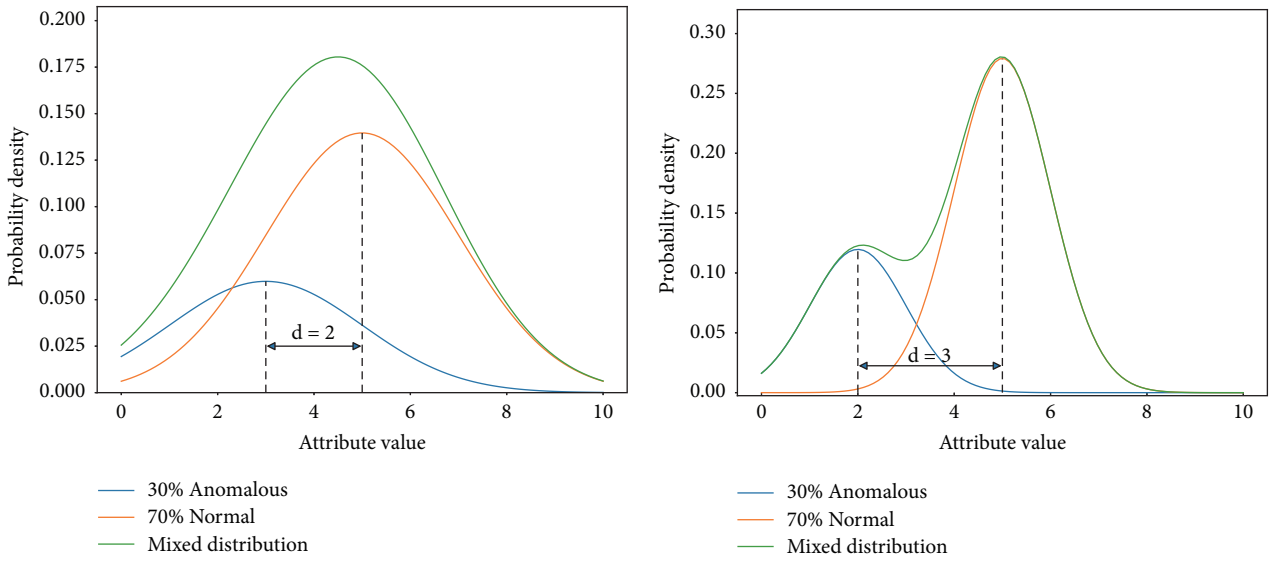


FIGURE 2: Two examples of the distributions of probability density with different clusters of attribute's values.

normal and anomalous instances. Based on the assumptions raised in the last section, we have summarized two factors which play some significant roles to distinguish the two types of objects by attribute values. Here, we average over the values on each type of instances to get a mean value, denoting a center of values. The distance between the centers of the two types of data is defined by Euclidean distance as

$$dist\left(X_a^k, X_n^k\right) = \sqrt{\left(E\left(x \mid x \in X_a^k\right) - E\left(x \mid x \in X_n^k\right)\right)^2}, \quad (1)$$

where X_a^k and X_n^k represent the value sets for anomalous instances and normal instances on attribute k and $E(x)$ is the mean function. The dispersity of the attribute values for each type of instances can be measured by the variance over the values. So we have

$$disp\left(X_a^k, X_n^k\right) = D\left(x \mid x \in X_a^k\right) + D\left(x \mid x \in X_n^k\right), \quad (2)$$

where $D(x)$ is the function of variance. Besides the two factors mentioned above, another influential factor is accounted for the information contained in an attribute. From a perspective of information theory, if the values belonging to an attribute are more scattered, it indicates that the attribute carrying more information can be regarded more important in feature selection. We again use the variance function to measure the degree of dispersivity of the values. Hereafter combining the three aspects together, we propose a separability index shown in (3) to measure how separable an attribute is in identifying the normal instances and anomalous ones.

$$sep\left(X^k, X_a^k, X_n^k\right) = \frac{dist\left(X_a^k, X_n^k\right) D\left(X^k\right)}{disp\left(X_a^k, X_n^k\right)}. \quad (3)$$

Equation (3) demonstrates that the separability of an attribute should be positively correlated with the $dist(X_a^k, X_n^k)$ and $D(X^k)$, whereas negatively correlated with the $disp(X_a^k, X_n^k)$. Thereby, as a criterion, the separability index allows us to select an attribute with a high value of (3) to partition the instances. Consequently, the separability index answers the first question regarding how to choose a suitable attribute for iTree building.

For the values of a given attribute, some of them correspond to the normal instances and the others relate to the anomalous ones. However, we do not know in advance which part of the values belongs to the normal instance or the anomalous. So, we need to find a split value to divide the attribute values into two parts. Based on the separability of an attribute discussed above, if we randomly choose a v from the attribute values as the split value, we can calculate the separability index by

$$sep\left(X^k, v\right) = \frac{\sqrt{\left(E\left(x \mid x \in X^k, x < v\right) - E\left(x \mid x \in X^k, x > v\right)\right)^2} D\left(X^k\right)}{D\left(x \mid x \in X^k, x < v\right) + D\left(x \mid x \in X^k, x > v\right)}. \quad (4)$$

It can be deduced that the v in the attribute values having the highest value of the index should be the best split point. Note that, when performing some parameter tunings, we found that the roles of the distance between the centers and the variance $D(x^k)$ should be enhanced in the formula. Thereby, we slightly change the formula as

$$sep\left(X^k, v\right) = \frac{e^{\sqrt{\left(E\left(x \mid x \in X^k, x < v\right) - E\left(x \mid x \in X^k, x > v\right)\right)^2}} D\left(X^k\right)^2}{D\left(x \mid x \in X^k, x < v\right) + D\left(x \mid x \in X^k, x > v\right)}. \quad (5)$$

4.3. A Gradient Method for Fast Searching the Split Point. For a given attribute, if we sort all the values within the attribute in ascending order, the simplest way to locate the best split value is trying to calculate the values of separability index

Input:
X^k: sorted values of the kth attribute;
Output:
bestX: the attribute value having the largest value of the separability index;
bestSep: the largest value of the separability index;
(1) Initiate *step* as $ceil(|X^k| * 0.01)$
(2) Let *bestSep* be $sep(X^k, x_1)$ and *bestX* be $(x_1 + x_2)/2$ and $i = 0$
(3) **while** $i < |X^k|$ **do**
(4) Set $i = i + step$
(5) Set $currSep = sep(X^k, x_i)$
(6) **if** $currSep > bestSep$ **then**
(7) Set $bestSep = currSep$
(8) Set $bestX = (x_i + x_{i+1})/2$
(9) **end if**
(10) Update *step* by following the formulas (6) and (7)
(11) **end while**
(12) Return *bestX* and *bestSep*

Algorithm 1: GradFindSplit.

one by one based on the sorted attribute values and choose one of the attribute values with the largest value of the index, i.e., split point. However, if the number of values for the attribute is tremendous, an exhaustive search may not be a good solution. Hence, we propose an approximately optimum method for searching the splitting point towards a given attribute. Calculating the gradients of the separability index values between two adjacent attribute values will be able to guide us to skip some attribute values which are not possible to be determined as split point and thus speed up the search process. The definition of the gradient for two neighbored attribute values x_i and x_{i+1} is

$$grad = \frac{sep\left(X^k, x_{i+1}\right) - sep\left(X^k, x_i\right)}{x_{i+1} - x_i} \tag{6}$$

While searching the attribute values from small to large, we observe that the attribute values should be inspected one after another when the gradient of the two adjacent candidate split values is close to zero which means that it is possibly approaching the best split value. On the other hand, the more the value of gradient deviates the zero, the more attribute values can be jumped over as the current two attribute values and their neighbors have a little possibility of being the best split values. So we defined a variable *step* based on the gradient to measure how many points can be skipped while searching.

$$step = \begin{cases} \dfrac{3}{1 + e^{grad * \log_{10}|X^k|}} * \dfrac{\left|X^k\right|}{100} & grad < 0 \\ 0.7 - \dfrac{1.3}{1 + e^{grad * \log_{10}|X^k|}} * \dfrac{\left|X^k\right|}{100} & grad \geqslant 0 \end{cases} \tag{7}$$

where $|X^k|$ denotes the number of values in the attribute X^k. The detailed description of the searching process is summarized in Algorithm 1. Please note that, in the implementation, the derived split point is a mean value of two neighbored attribute values.

4.4. Parameter Setting of the Separability Index. In some cases, we notice that (5) will yield a very biased split point beyond our expectation. Here, we give an instance to address this issue. We generate a one-dimensional binary data with two different Gaussian distributions, marked as $G(3, 1.5)$ and $G(7, 1.5)$, where the first parameter is the mean and the second one is the variance. Of the synthetic data, 800 instances having the first statistical distribution are treated as the normal and 200 instances having the second statistical distribution are anomalous. Before calculating the separability index, all the values are normalized by using

$$v = \frac{v - v_{min}}{v_{max} - v_{min}} \tag{8}$$

where v_{max} and v_{min} are the maximum and the minimum of the generated points, respectively. We derive the best split point by searching the largest value of the separability index over the attribute values shown in Figure 3. As we can see, the obtained split point is apparently not the expected one. After investigating the values of the separability index, it turns out that the values are biased, influenced by the variances $D(x \mid x \in X^k, x < v)$ and $D(x \mid x \in X^k, x > v)$. The reasons behind the results are analyzed as follows. In our opinion, two aspects for $D(x \mid x \in X^k, x < v)$ and $D(x \mid x \in X^k, x > v)$ should be considered while calculating the value of the separability index:

(1) When the number of attribute values belonging to one class of the data occupies a major part, it means that the calculation of the variance over these values would be somewhat reliable since the value of the variance is more statistical credible with more instances involved. Thereby, in this case, to keep the reliability of the obtained variance, it is necessary to assign the obtained variance with a heavy weight.

(2) On the other hand, when the number of attribute values belonging to one class of the data is a minor part, it suggests that the derived variance would be not quite reliable since a small number of samples would be lack of statistical

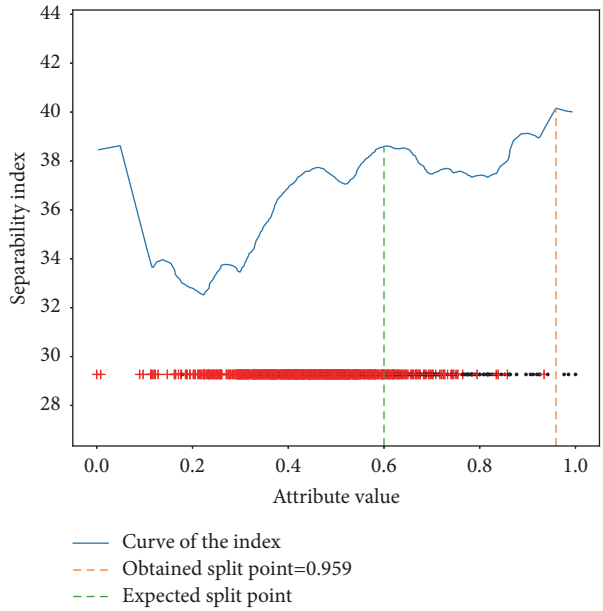


Figure 3: An example of the biased split point obtained by (5).

significance and fewer instances will lead to the outcome of the variance more distorted. In this case, it is reasonable to impose on a light weight to the obtained variance to minimize its biased influence to the separability index.

Based on the analysis mentioned above, we introduce two parameters into (5) to regulate the value of the $D(x \mid x \in X^k, x < v)$ and $D(x \mid x \in X^k, x > v)$ as

$$sep\left(X^k, v\right) = \frac{e^{\sqrt{(E(x|x\in X^k, x<v) - E(x|x\in X^k, x>v))^2}} D\left(X^k\right)^2}{\alpha D\left(x \mid x \in X^k, x < v\right) + \beta D\left(x \mid x \in X^k, x > v\right)}, \tag{9}$$

where parameters α and β are set as

$$\alpha = \frac{\left|\left\{x \mid x \in X^k, x < v\right\}\right|}{\left|X^k\right|}, \qquad \beta = \frac{\left|\left\{x \mid x \in X^k, x > v\right\}\right|}{\left|X^k\right|}, \tag{10}$$

which have no fixed values but varied along with the change of v. By doing so, we again calculate the best split point and will be able to obtain a more reasonable result as shown in Figure 4 where the value curve of the separability index will form a desired single peak.

4.5. The Algorithms for iTree and iForest Building. iTree is a height-limited proper binary tree. In other words, it is a truncated tree by the depth. In general, the height of the tree does not exceed $\log_2 n$ given n training instances. The depth of a leaf node in the iTree would not be its actual depth if it could be fully grown. If we use a function $h(x)$ denoting the depth of the leaf node x in the iTree, its actual depth $\widehat{h}(x)$ in the fully grown tree can be estimated as

$$\widehat{h}(x) = h(x) + 2\ln(n-1) + 2E_{const} - \frac{2(n-1)}{n}, \tag{11}$$

where Euler's constant $E_{const} \approx 0.5772$. So, we can mark the leaf nodes in the iTree by

$$label(x) = \begin{cases} 0 & \widehat{h}(x) \leqslant \log_2 n \\ 1 & \widehat{h}(x) > \log_2 n \end{cases} \tag{12}$$

The detailed algorithm description is shown in Algorithm 2.

To identify whether an instance is normal or not in the tree, one can simply make comparison processes throughout the tree with a start from the root of the tree and if the reached leaf node's label is 0 then the instance is determined as the anomalous and vice versa. Thus, it is no longer necessary to calculate the anomaly score as shown in the literature [2]. The proposed iTree algorithm has a similar time complexity to that of its original version. We suppose the training set contains N samples and choose k features (attributes) to construct an OIT with a depth $\log_2(N)$. To build the tree, the required calculations level by level are $kN * (\log_2(N) + \log_2(N/2) + \cdots + 1)$. Thereby, the complexity of the algorithm is $O(kN[\log_2(N)]^2)$.

The algorithm of building iForest is largely similar to the one in the literature [2]. The main difference between the new iForest and the original one is that it does not require the calculation of the anomaly score and use the following criterion instead to detect if an instance v is anomalous.

$$anomaly(x) = \begin{cases} 0 & E\left(\widehat{h}(v)\right) \leqslant \log_2 n \\ 1 & E\left(\widehat{h}(v)\right) > \log_2 n \end{cases} \tag{13}$$

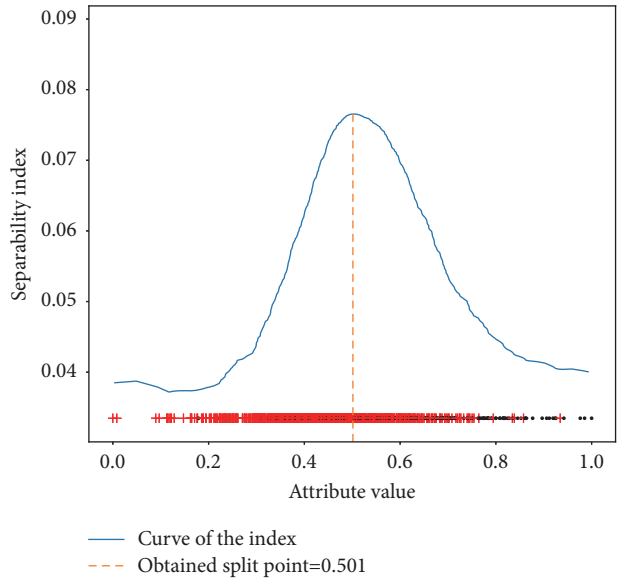


Figure 4: An example of the expected split point obtained by the improved equation (9).

Input:
D: a dataset with m attributes and n records;
$depth$: current depth of the tree;
lim: limit of the tree height;
Output:
OIT: an optimized isolation tree;
(1) **if** $depth > lim$ or $|D| < 2$ **then**
(2) return the label of the node
(3) **end if**
(4) randomly select $k(k \leqslant m)$ distinct attributes from D
(5) pick out the attribute i with the highest value of the separability index
(6) $Dleft = \{x \mid x < bestX, x \in D\}$
(7) $Dright = \{x \mid x \geqslant bestX, x \in D\}$
(8) Return $T = \{bestX, bestSep, leftchild = iTreeBuilding(Dleft, depth + 1, lim), rightchild = iTreeBuilding(Dright, depth + 1, lim)\}$

Algorithm 2: iTreeBuilding.

where $E(\widehat{h}(v))$ is the average of all the path lengths of the node v over the trees in the iForest.

5. Experiments

5.1. Description of the Datasets and Evaluation Metric. The datasets used for evaluation in this paper have two sources. The data of *Unionpay* is provided by the China Unionpay company, which contains the statistical expenditure information regarding the merchants (the data is available upon request). Of the 3074 members in the dataset, there are two types of merchants including the anomalous ones with the behavior of tax evasion and the normal ones. The rest of the data come from the open accessed UCI data repository (http://archive.ics.uci.edu/ml/index.php). If there are just two kinds of instances in the data, the minor ones are regarded as anomaly data. Of course, some datasets have more than two kinds of instances, for convenience, we incorporate all minor types into one as the anomaly data. Some detailed information regarding the datasets used in this article is summarized in Table 2.

5.2. Detection Accuracy Comparisons with the State-of-the-Art Models. The evaluation metric used in this article is AUC value. AUC means the area under the ROC curve. The bigger the area, the better the model's performance in accuracy. Here, we make accuracy comparisons for anomaly detection between OIT, OIF, IForest, SCiForest, and LOF. To avoid any biased outcomes, 10-fold cross validation is adopted to generate the averaged AUC values and corresponding standard deviations in the tests. According to the statistical tests, the experimental results are significant with respect to a significance level of 0.05. Based on the results shown in Table 3, apart from the datasets of *ionosphere* and *Kddcup*99,

Table 2: Statistics of eight datasets.

Date sets	# of records	# of attributes	The ratio of anomaly data
Breast	198	33	Malignant (23.7%)
Ionosphere	351	33	Bad (35.8%)
Pima-diabetes	768	8	Pos (34.9%)
Breast-diagnostic	569	30	Malignant (37.3%)
CreditCardDefault	30000	23	yes (22.1%)
PenDigits	9868	16	yes (2%)
UnionPay	3074	95	Fraud merchants (11.5%)
Kddcup99	60839	80	Abnormal (0.4%)

Table 3: The AUC results of the five methods on eight datasets. Values marked in bold are the best results.

Date sets	LOF	iForest	SCiForest	OIT	OIF
Breast	0.494±0.132	0.496±0.149	0.513±0.128	0.558±0.152	**0.560±0.126**
Ionosphere	0.893±0.093	0.855±0.083	**0.916±0.071**	0.843±0.074	0.874±0.069
Pima-diabetes	0.587±0.051	0.679±0.053	0.631±0.036	0.649±0.058	**0.734±0.055**
Breast-diagnostic	0.525±0.086	0.794±0.069	0.853±0.053	0.671±0.109	**0.883±0.050**
CreditCardDefault	0.499±0.014	0.592±0.017	0.587±0.014	0.619±0.017	**0.653±0.013**
PenDigits	0.650±0.193	0.796±0.088	0.802±0.100	0.865±0.128	**0.938±0.035**
UnionPay	0.468±0.056	0.645±0.052	0.472±0.066	0.601±0.060	**0.716±0.057**
Kddcup99	0.616±0.110	**0.989±0.006**	0.987±0.009	0.962±0.039	0.986±0.010

Table 4: The training timings of the OIT and OIF on eight datasets.

Date sets	Training time of OIT (s)	Training time of OIF (s)
Breast	0.047	0.523
Ionosphere	0.077	0.818
Pima-diabetes	0.105	1.096
Breast-diagnostic	0.124	1.429
CreditCardDefault	7.65	95.16
PenDigits	2.23	24.18
UnionPay	1.08	12.06
Kddcup99	11.85	138.63

OIF outperforms the rest of methods on the six other datasets. The encouraging results suggest that the new proposed solution for iForest indeed can overall give an improvement compared to its previous versions, i.e., iForest and SCiForest.

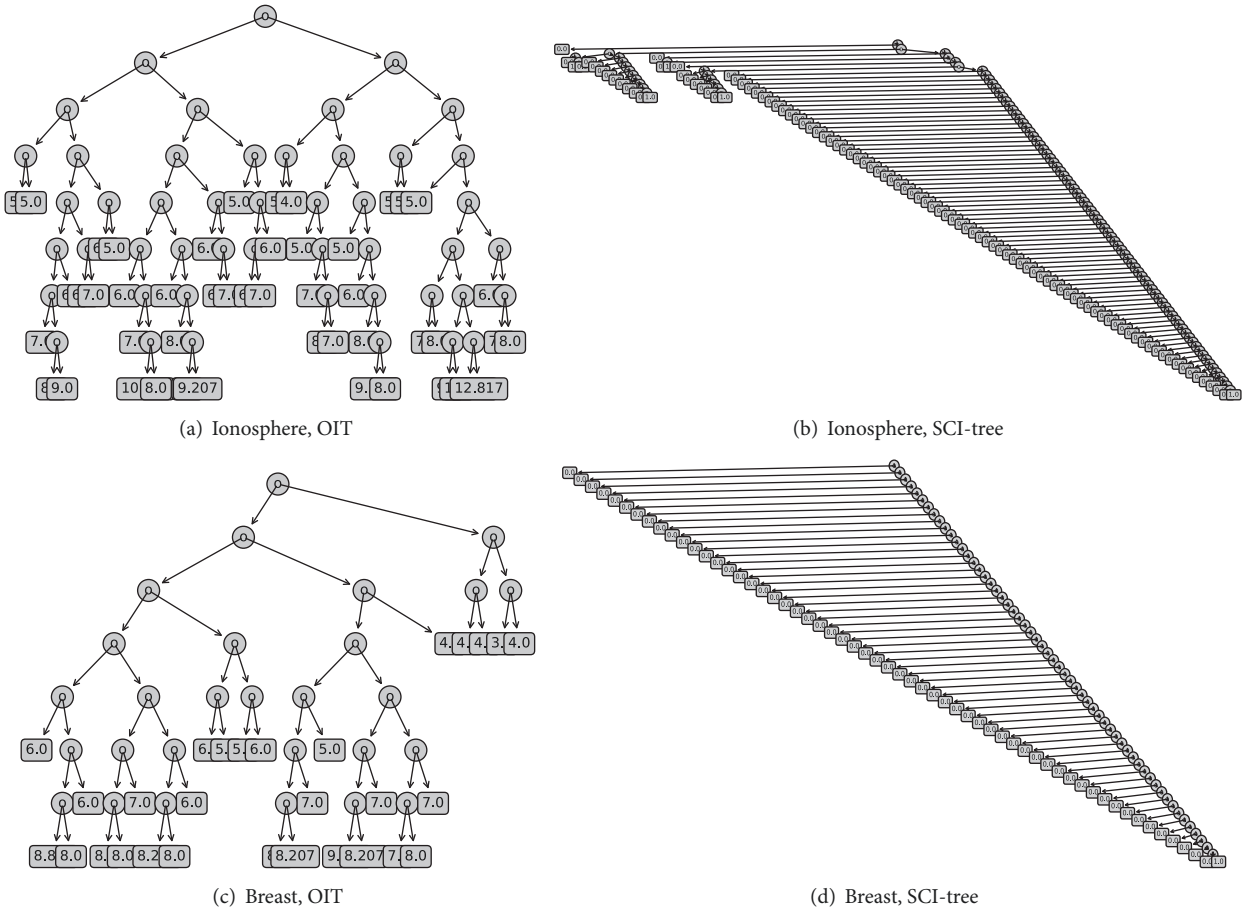
The information about training time efficiency of the proposed method on the eight datasets is presented in Table 4. The testing machine is a desktop with 8 Gigabyte memory and 3.4 GHz Intel core-i3 processor.

6. Analysis

6.1. Tree Structure Comparisons between the Proposed Model and SCiForest. Both iForest and our model need to build a proper binary tree to isolate the outliers. But, when applying the two models to build the trees on datasets, we notice that the generated trees by the two models are looking very different in structure. Here, we give two examples to demonstrate how different they are. In Figure 5, it shows the trees generated by the two models on datasets *Ionosphere* and *Breast*. It turns out that the trees built by the OIT are well balanced in structure, whereas the ones build by the iForest are extremely unbalanced, which would lead to relatively low efficiency in computations.

6.2. Convergence Comparisons between the Proposed Model and iForest. In this section, we try to evaluate the speed of convergence of the proposed model. Since both OIF and iForest have a procedure of convergence when building the forest, we would like to find out which one is quicker to be convergent. In Figure 6, we randomly generate 1000 points. Among them, 800 are normal instances and 200 are anomalous. An anomalous point and a normal one are marked with x_1 and x_0 to be detected by applying the models. To compare the speed of convergence between the OIF and iForest, we produce a number of trees to constitute a forest for each model and calculate the average path length of x_0 and x_1 over the generated trees. The standard deviation of average path length over the trees can be treated as a criterion to determine if the average path length converges or not along with the increasing size of the forest.

The experiments employ the rates of downsampling varying by 0.25, 0.5, and 0.75, respectively. The threshold is set as 0.15 which means that, when the deviation of average path lengths over 50 generated forests is smaller than the threshold, the model reaches the state of convergence. Figure 7 shows that far fewer numbers of trees are required by the OIF to ensure the deviation of average path lengths converged compared with those required by the iForest. Figure 8 also validates that our model converges faster than iForest on the real dataset *Pima-diabetes*.



(a) Ionosphere, OIT (b) Ionosphere, SCI-tree

(c) Breast, OIT (d) Breast, SCI-tree

Figure 5: The structure comparisons of the trees generated by two models on datasets: *Ionosphere* and *Breast*.

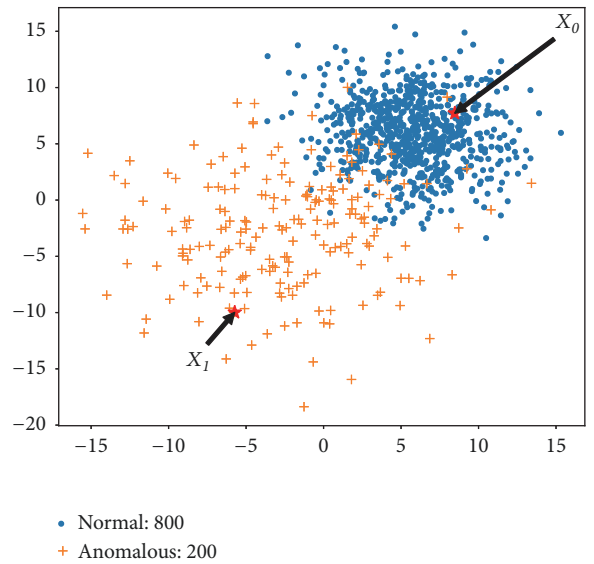


Figure 6: Generated points distribution with 800 normal instances versus 200 anomalous ones.

6.3. The Impact of Point Density on the Proposed Model. To further study the performance of the proposed models on datasets with different point density, we generate two synthetic data shown in Figure 9 to carry out the tests. Here, 800 normal points and 200 anomalous points with two dimensional features are generated which have Gaussian distributions with varied standard deviation to represent different point densities. The experimental results shown

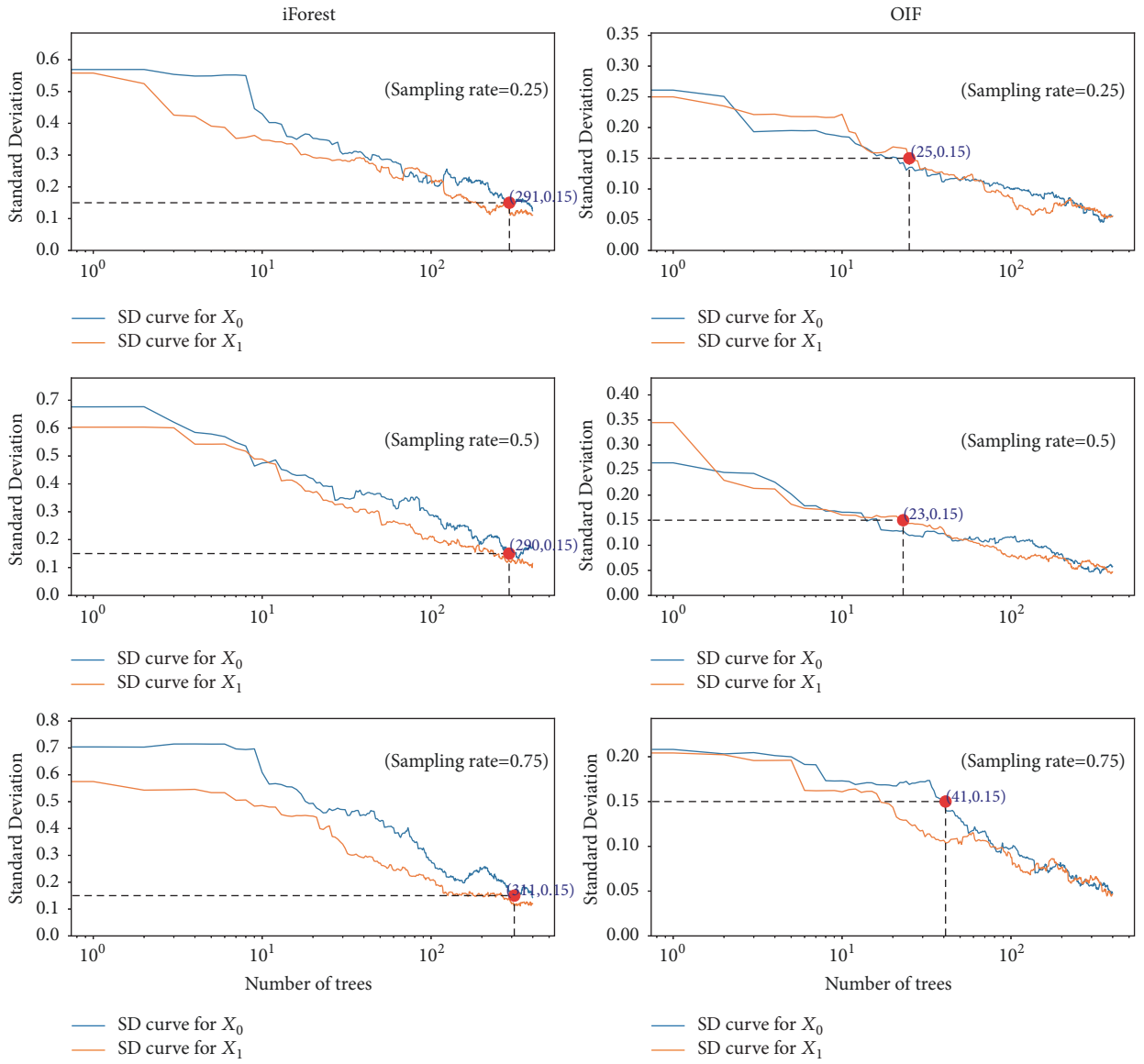


Figure 7: Convergence curves for models of iForest and OIF on a synthetic data with varied downsampling rates.

Table 5: The AUC results on two synthetic datasets with different point density.

Algorithms	AUC (SD_a=1, SD_n=3)	AUC (SD_n=3, SD_a=5)
OIT	0.406	0.899
OIF	0.899	0.992

Table 6: The AUC results on four synthetic datasets with different point distributions.

Algorithms	Gaussian	Moons	Circles	Classification
LOF	0.981	0.593	**0.658**	0.613
iForest	**0.989**	0.864	0.008	0.728
SCiForest	0.854	**0.921**	0.055	0.666
OIF	0.979	0.911	0.014	**0.775**

in Table 5 indicate that OIT and OIF tend to have good performance when the anomalous points are more scattered than the normal ones.

6.4. The Impact of Point Distribution on the Proposed Model. We select four available functions in scikit-learn (an open-sourced Python package) to generate the synthetic datasets with specified distributions including Gaussian, Moons, Circles, and Classification. As shown in Figure 10, each distribution has 200 instances and the ratio between normal instances and anomalies is 4:1. The AUC results obtained from the experiments by iForest and OIF are shown in Table 6. The results indicate that the four models have their own merits on distinct data distributions and each of them has its specified application prospect.

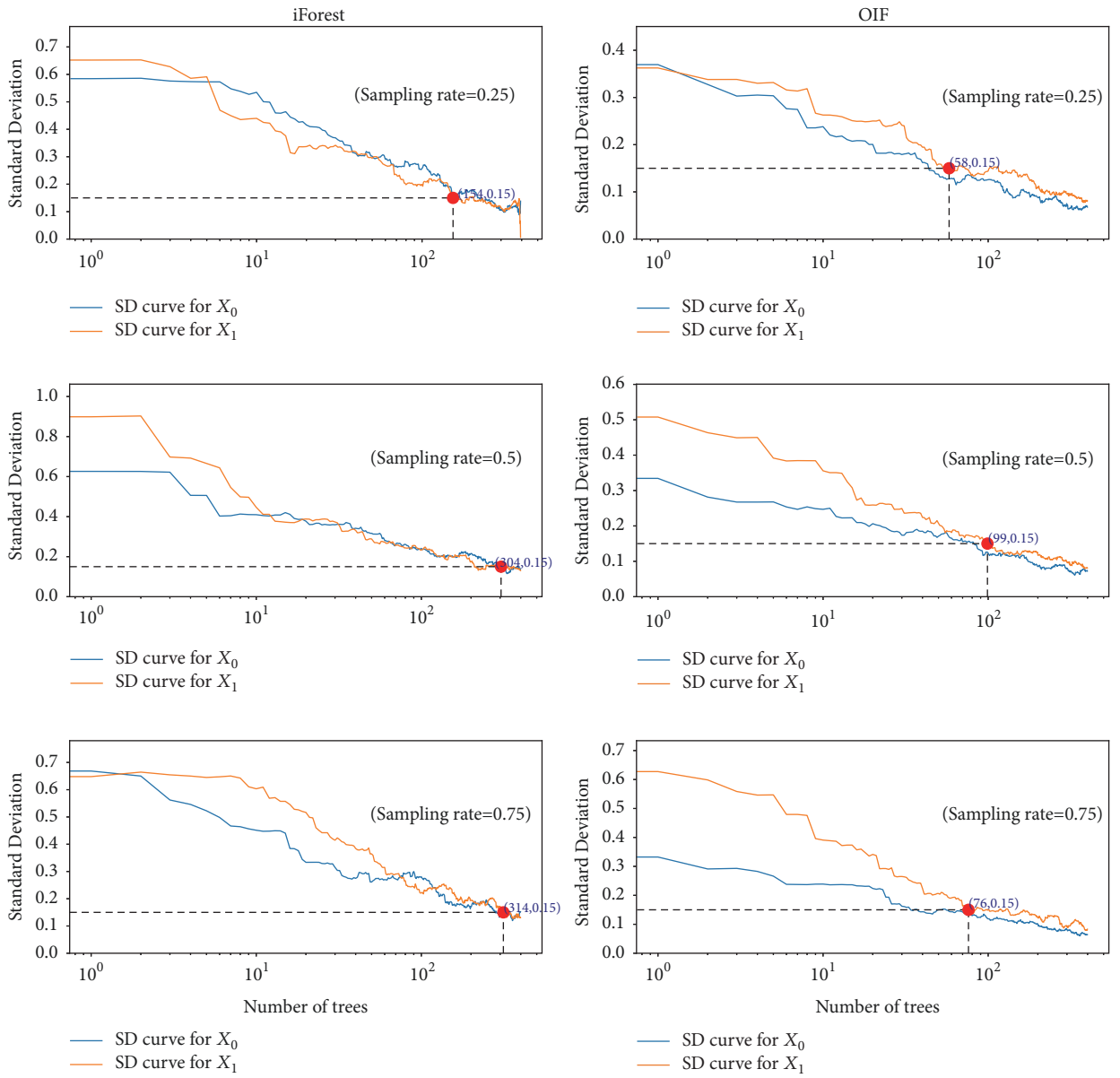


Figure 8: Convergence curves for models of iForest and OIF on a real dataset *Pima-diabetes* with varied downsampling rates.

7. Conclusion

In this article, we proposed a novel model for outlier detection which is an optimized version of iForest. The advancements of the new model, compared with the iForest, can be summarized as follows. (1) A separability index is devised which allows us to select proper attribute and locate the splitting point on it accurately. (2) A fast searching algorithm based on gradient is proposed which is able to speed up the process in retrieving the best splitting point. (3) Compared with the iForest, it requires fewer trees to reach the status of convergence in our model while generating the forest. Besides, the detection accuracy results shown on 8 real datasets also validate that our model performs overall better than the iForest and its improved version SCiForest. Of course, although it is called an optimized framework for iForest, it does not mean that the proposed model has been perfect and there is no space to further improve. In fact, we consider that two directions could be studied in the future. (1) Compared to the AUC result of LOF on data with circle distribution, iForest and its improved versions all performed very poor. It would be of theoretical interest to investigate the possible causes behind the results. (2) All the tree-based models for outlier detection are applied with the strategy of randomization. Despite it is a powerful tool in applications, the strategy of randomization is not interpretable and some further understandings towards it are required to study imminently.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

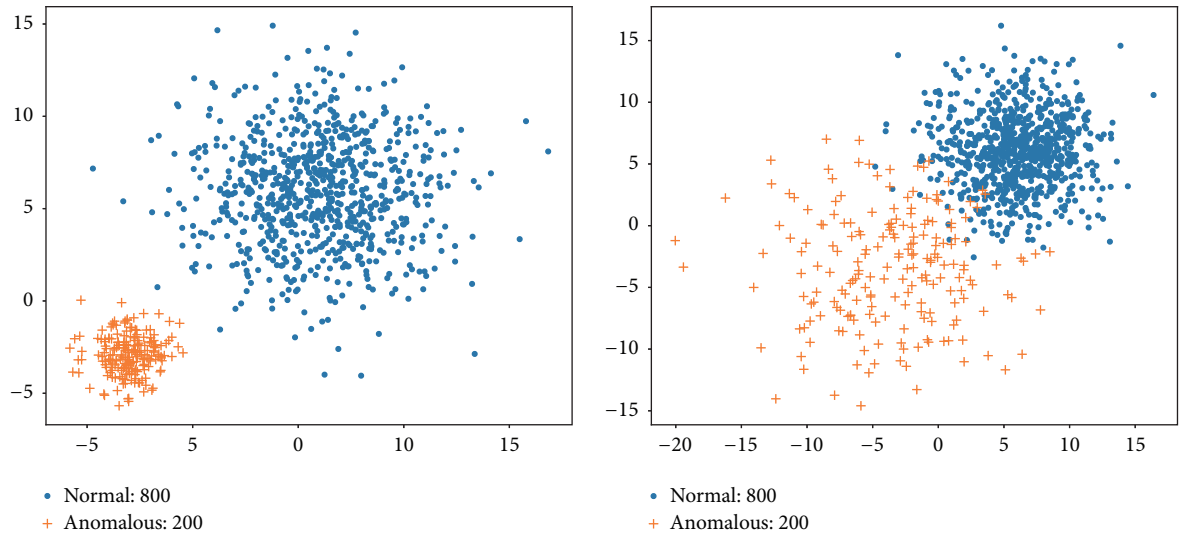


Figure 9: Two examples of the normal and anomalous points distributions with different densities.

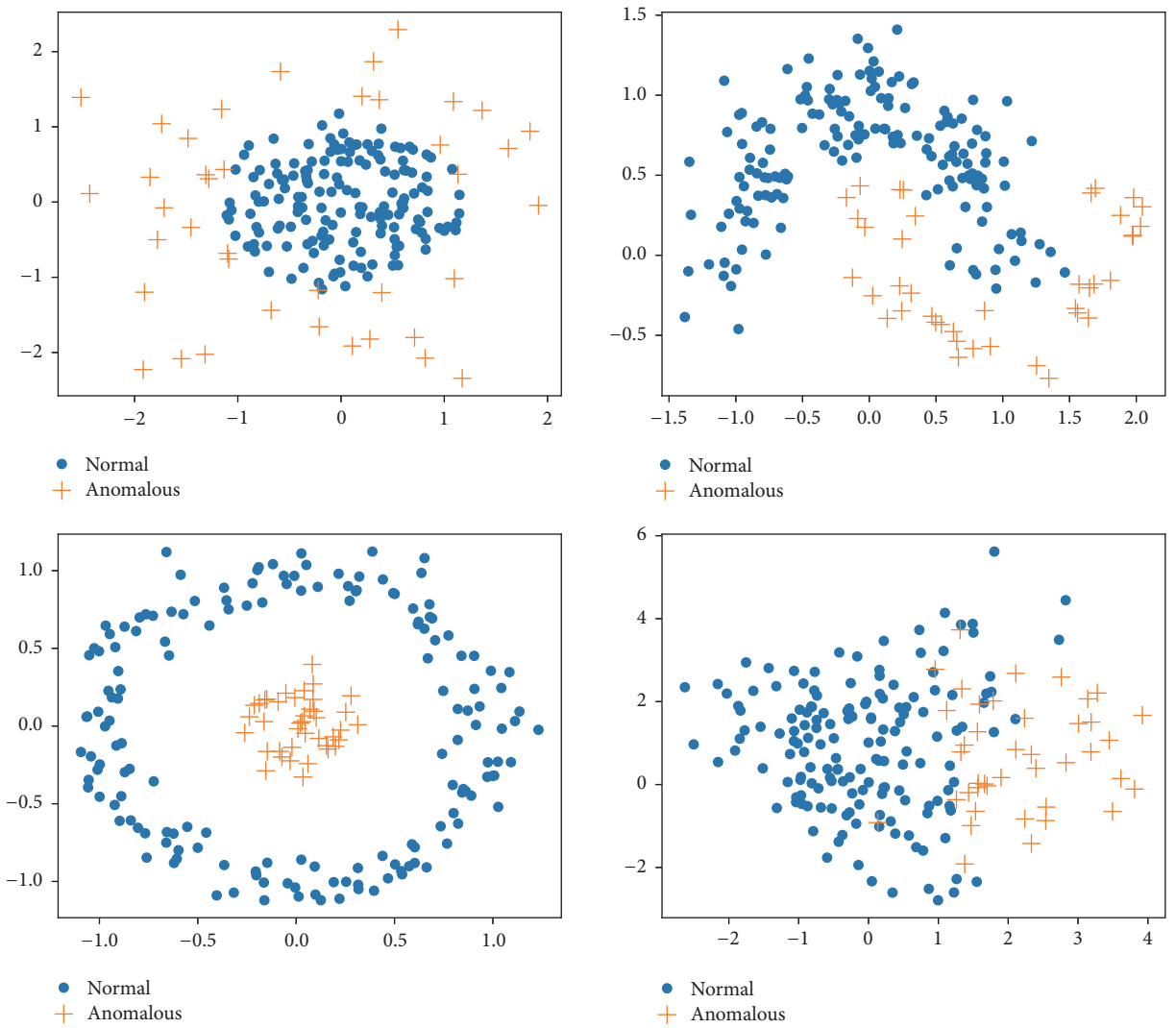


Figure 10: Four specified points distributions including Gaussian, Moons, Circles, and Classification.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Authors' Contributions

Zhen Liu and Xin Liu are co-first authors for their equal contributions to this work.

Acknowledgments

Z. Liu acknowledges the fund of National Natural Science Foundation of China under Grant no. 60903073 and Special Project of Sichuan Youth Science and Technology Innovation Research Team (2013TD0006).

References

[1] D. M. Hawkins, *Identification of Outliers*, vol. 11, Springer, Berlin, Germany, 1980.

[2] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation-based anomaly detection," *ACM Transactions on Knowledge Discovery from Data*, vol. 6, no. 1, p. 3, 2012.

[3] G. O. Campos, A. Zimek, J. Sander et al., "On the evaluation of unsupervised outlier detection: measures, datasets, and an empirical study," *Data Mining and Knowledge Discovery*, vol. 30, no. 4, pp. 891–927, 2016.

[4] A. Zimek, E. Schubert, and H.-P. Kriegel, "A survey on unsupervised outlier detection in high-dimensional numerical data," *Statistical Analysis and Data Mining*, vol. 5, no. 5, pp. 363–387, 2012.

[5] X. Su and C.-L. Tsai, "Outlier detection," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 1, no. 3, pp. 261–268, 2011.

[6] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection for discrete sequences: a survey," *IEEE Transactions on Knowledge and Data Engineering*, vol. 24, no. 5, pp. 823–839, 2012.

[7] E. M. Knorr and R. T. Ng, "A unified notion of outliers: Properties and computation," in *KDD*, pp. 219–222, 1997.

[8] F. Angiulli and C. Pizzuti, "Fast outlier detection in high dimensional spaces," in *European Conference on Principles of Data Mining and Knowledge Discovery*, pp. 15–27, Springer, Berlin, Germany, 2002.

[9] V. Hautamaki, I. Karkkainen, and P. Franti, "Outlier detection using k-nearest neighbour graph," in *Proceedings of the 17th International Conference on Pattern Recognition, ICPR '04*, vol. 3, pp. 430–433, IEEE, 2004.

[10] N. H. Vu and V. Gopalkrishnan, "Efficient pruning schemes for distance-based outlier detection," in *Proceedings of the Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, vol. 5782, pp. 160–175, Springer.

[11] G. Kollios, D. Gunopulos, N. Koudas, and S. Berchtold, "Efficient biased sampling for approximate clustering and outlier detection in large data sets," *IEEE Transactions on Knowledge and Data Engineering*, vol. 15, no. 5, pp. 1170–1187, 2003.

[12] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "LOF: identifying density-based local outliers," *ACM SIGMOD Record*, vol. 29, no. 2, pp. 93–104, 2000.

[13] W. Jin, A. K. H. Tung, J. Han, and W. Wang, "Ranking outliers using symmetric neighborhood relationship," in *PAKDD*, vol. 6, pp. 577–593, Springer, Berlin, Germany, 2006.

[14] S. Papadimitriou, H. Kitagawa, P. B. Gibbons, and C. Faloutsos, "Loci: Fast outlier detection using the local correlation integral," in *Proceedings of the 19th International Conference on Data Engineering*, U. Dayal, K. Ramamritham, and T. M. Vijayaraman, Eds., pp. 315–326, IEEE, Bangalore, India, March 2003.

[15] K. Zhang, M. Hutter, and H. Jin, "A new local distance-based outlier detection approach for scattered real-world data," *Advances in Knowledge Discovery and Data Mining*, pp. 813–822, 2009.

[16] F. Keller, E. Muller, and K. Bohm, "HiCS: High Contrast Subspaces for Density-Based Outlier Ranking," in *Proceedings of the IEEE International Conference on Data Engineering (ICDE '12)*, pp. 1037–1048, IEEE, April 2012.

[17] H. Kriegel, P. Kröger, E. Schubert, and A. Zimek, "Outlier detection in axis-parallel subspaces of high dimensional data," *Advances in Knowledge Discovery and Data Mining*, pp. 831–838, 2009.

[18] J. Basak and R. Krishnapuram, "Interpretable hierarchical clustering by constructing an unsupervised decision tree," *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 1, pp. 121–132, 2005.

[19] F. T. Liu, K. M. Ting, and Z. Zhou, "On detecting clustered anomalies using sciforest," in *Proceedings of the Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 274–290, Springer, Berlin, Germany, 2010.

[20] S. Guha, N. Mishra, G. Roy, and O. Schrijvers, "Robust random cut forest based anomaly detection on streams," in *Proceedings of the International Conference on Machine Learning, ICML '16*, pp. 2712–2721, 2016.

[21] E. Schubert, A. Zimek, and H.-P. Kriegel, "Local outlier detection reconsidered: a generalized view on locality with applications to spatial, video, and network outlier detection," *Data Mining and Knowledge Discovery*, vol. 28, no. 1, pp. 190–237, 2014.

[22] J. Gao, F. Liang, W. Fan, C. Wang, Y. Sun, and J. Han, "On community outliers and their efficient detection in information networks," in *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM '10*, pp. 813–822, 2010.

[23] D. Pokrajac, A. Lazarevic, and L. J. Latecki, "Incremental local outlier detection for data streams," in *Proceedings of the 1st IEEE Symposium on Computational Intelligence and Data Mining (CIDM '07)*, pp. 504–515, IEEE Press, Honolulu, Hawaii, USA, 2007.

[24] J. Lee, J. Han, and X. Li, "Trajectory outlier detection: A partition-and-detect framework," in *Proceedings of the IEEE 24th International Conference on Data Engineering (ICDE '08)*, pp. 140–149, IEEE, Cancun, Mexico, 2008.

[25] C. C. Aggarwal and P. S. Yu, "Outlier detection with uncertain data," in *Proceedings of the SIAM International Conference on Data Mining, SIAM '08*, pp. 483–493, 2008.