# GA-iForest: An Efficient Isolated Forest Framework Based on Genetic Algorithm for Numerical Data Outlier Detection

*LI Kexin*[1], *LI Jing*[1*], *LIU Shuji*[2], *LI Zhao*[2], *BO Jue*[2], *LIU Biqi*[2]

1. College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 211106, P. R. China; 2. State Grid Liaoning Electric Power Supply Co., LTD, Shenyang 110004, P. R. China

(Received 25 January 2019; revised 23 April 2019; accepted 27 May 2019)

**Abstract:** With the development of data age, data quality has become one of the problems that people pay much attention to. As a field of data mining, outlier detection is related to the quality of data. The isolated forest algorithm is one of the more prominent numerical data outlier detection algorithms in recent years. In the process of constructing the isolation tree by the isolated forest algorithm, as the isolation tree is continuously generated, the difference of isolation trees will gradually decrease or even no difference, which will result in the waste of memory and reduced efficiency of outlier detection. And in the constructed isolation trees, some isolation trees cannot detect outlier. In this paper, an improved iForest-based method GA-iForest is proposed. This method optimizes the isolated forest by selecting some better isolation trees according to the detection accuracy and the difference of isolation trees, thereby reducing some duplicate, similar and poor detection isolation trees and improving the accuracy and stability of outlier detection. In the experiment, Ubuntu system and Spark platform are used to build the experiment environment. The outlier datasets provided by ODDS are used as test. According to indicators such as the accuracy, recall rate, ROC curves, AUC and execution time, the performance of the proposed method is evaluated. Experimental results show that the proposed method can not only improve the accuracy and stability of outlier detection, but also reduce the number of isolation trees by 20%—40% compared with the original iForest method.

**Key words:** outlier detection; isolation tree; isolated forest; genetic algorithm; feature selection

**CLC number**: TP301.6 **Document code**: A **Article ID**: 1005-1120(2019)06-1026-13

## 0 Introduction

As a significant subject, outlier detection has been widely researched recently. The general idea of outlier detection is to identify data objects that do not fit well in general data distribution. Outlier detection is related to many aspects of real life, such as track clustering[1], cyber malicious attacks, transaction fraud, grid data anomaly, and so on. For example, the governance of power grid data may be abnormal, missing and repeated to some extend for power data from different systems. These abnormal data often can reflect some potential problems faced by various business systems, power equipment and platforms in practical work. How to effectively govern the data from different sources will have a profound impact on the "management", "viewing", "using" and "checking" of data. How to detect and identify abnormal data (outlier) has become a hot research topic.

The definition of outliers given by Hawkins[2] is generally accepted, which is "An outlier is an observation that differs so much from other observation as to arouse suspicion that it was generated by a different mechanism". In the field of outlier detection, the commonly used methods mainly include distance based[3-5], density based[6-8], cluster based[9-11], tree based[12-13], etc.

In the process of detecting outliers, traditional methods based on distance, density and clustering

*Corresponding author, E-mail address: lijing@nuaa.edu.cn.

**How to cite this article**: LI Kexin, LI Jing, LIU Shuji, et al. GA-iForest: An Efficient Isolated Forest Framework Based on Genetic Algorithm for Numerical Data Outlier Detection[J]. Transactions of Nanjing University of Aeronautics and Astronautics, 2019,36(6):1026-1038.

http://dx.doi.org/10.16356/j.1005-1120.2019.06.015

usually need to define abnormal distance, neighborhood radius, abnormal density, etc., and then obtain outlier scores through a large number of calculations or comparisons on the whole dataset. The labels of data (outliers or inliers) are finally determined by different thresholds of different methods. These methods construct a profile of normal instances, then identify outliers that do not conform to the normal profile. In addition, some of these methods (such as clustering) don't intend to specifically detect outliers which are merely a by-product. Therefore, these methods have two major drawbacks: (i) they are optimized to profile normal instances, but not optimized to detect anomalies. And the results of outlier detection might not be as good as expected causing too many inliers identified as outliers or too few outliers identified as inliers. (ii) Many existing methods can only be used in the small data size because of their high computational complexity. Later, an outlier detection algorithm based on isolated forest (iForest for short) appears. This method uses the idea of random sampling to sample the original dataset without using all the whole data when performing the outlier detection. Therefore, it does not require a large amount of calculations and comparisons. And the amount of calculation required for this method is greatly reduced. The method of iForest obtains a number of samples by randomly sampling the original dataset, and then establishes an isolation tree (iTree for short) on each sub-sample. It is considered that the data closer to the root node is more likely to be abnormal data. Finally, the path lengths of the data on different iTrees are integrated to obtain the final abnormal score by which the abnormality of the data can be judged. Although iForest has logarithmic time complexity and can effectively process large data sets, it also has the following drawbacks:

(i) Since iTree is constructed based on subsamples from random sampling, the structure of iTree built by different samples may be similar or identical. As the number of iTrees increases, the difference between iTrees gradually decreases, which will cause memory waste and unnecessary computational overhead.

(ii) The purpose of building iTrees is to find out which data is closer to the root node. However, due to the random sampling of the original dataset and the next construction of iTrees, it is possible that some of the iTrees are complete binary trees and cannot realize the detection of outliers.

(iii) Although the iForest can use part of the data to detect outliers through random sampling of the original data, the iTree established on different sub-samples will appear similar or repeated, so the detection results of outliers may vary greatly each time, and the stability of detection cannot be guaranteed.

In response to the above questions, a method is proposed to optimize the construction of iForest using genetic algorithm on the basis of the original iForest algorithm. The proposed method selects iTrees with high detection accuracy and large difference to form iForest, thereby optimizing the detection of outliers. In this paper, the detection accuracy, difference index and fitness function of the iTrees are established first, and then the iTrees selected by the genetic algorithm are used to improve the accuracy and stability of outlier detection. The main contributions of this paper are as follows:

(i) It has reduced the generation of 20%—40% iTrees and saved computing resources and memory space.

(ii) Some indicators are built for selecting iTrees, such as accuracy of each iTrees, similarity of difference iTrees and fitness function.

(iii) It has improved the stability of outlier detection by adding a selection process of iTrees according to the accuracy and similarity of each iTree.

## 1 Related Work

Outlier detection is a very popular area in the field of data mining. In recent years, many scholars have done a lot of work in this field. Here, several advanced and representative outlier detection methods are introduced.

The distance-based outlier detection method is a basic method for outlier detecting, which tries to find the global outliers far from the rest of the data

based on $k$ nearest neighbor distances of the data points. This method calculates the distance of the $k$ nearest neighbors of each data as the abnormal distance, and then sorts the abnormal distance. The data with a large abnormal distance is more likely to be considered as outliers. Then, based on this point of view, some distance-based outlier detection methods are proposed, for example, KNN-weight[14].

The density-based approach and its variants try to find the local outliers located in a lower density region compared to their $k$ nearest neighbors. This method defines the anomaly factor[15] concept for the first time and counts the number of neighbors within the specified radius of the data point as outlier score. It is generally considered that the data with a small outlier score is outlier. Based on the idea of density, some improved density-based outlier detection methods have been proposed, for instance, K-LOF[16]. This method uses a clustering algorithm to find the center of the original dataset, and then determines whether the data is outlier according to the distance from the data to the center.

The cluster-based method divides the data into a series of clusters by basic cluster algorithm, and judges whether the data is abnormal according to the size of the cluster. The data in clusters with less data is generally considered to be outlier. Later, some scholars proposed an anomaly data recognition algorithm based on sample boundaries[10]. Firstly, clustering was used to obtain the boundary sample set of normal clusters, and then the relationship between the data to be tested and the boundary sample set was analyzed. If the data to be tested belonged to the boundary sample set, it was judged as normal data; otherwise, it was outlier.

The above abnormal value detection methods are usually based on a large number of calculations or comparisons, which will result in a large computational overhead when the number of data is larger. In addition, these methods often under-perform resulting in too many false alarms (having normal instances identified as anomalies) or too few anomalies be detected. Although the emergence of iForest-based outlier detection methods have improved the efficiency of outlier detection, as mentioned above, it still has some shortcomings. Aryal at al.[17] pointed out that the underlying similarity or distance measures in iForest had not be well understood. Contrary to the claims that these methods never rely on any distance measures, they found that iForest had close relationships with certain distance measures. This implies that the current use of this fast isolation mechanism is only limited to these distance measures and fails to generalize to other commonly used measures. And then they proposed a generic framework named LSHiForest for fast tree isolation based ensemble anomaly analysis with the use of a locality-sensitive hashing (LSH) forest. This framework can be instantiated with a diverse range of LSH families, and the fast isolation mechanism can be extended to any distance measures, data types and data spaces where an LSH family is defined. Zhang et al.[18] indicated that while iForest-based methods were effective in detecting global anomalies, they failed to detect local anomalies in datasets having multiple clusters of normal instances because the local anomalies were masked by normal clusters of similar density and they became less susceptible to the isolation. And they proposed a very simple but effective solution to overcome this limitation by replacing the global ranking measure based on the path length with a local ranking measure based on relative mass that took local data distribution into consideration. Although the shortcomings of the iForest-based outlier detection methods were pointed out, they could not be fundamentally improved. In this paper, indicators such as detection accuracy and difference are established for the iTrees, and then the genetic algorithm is used to select the iTrees to filter those iTrees with low detection accuracy and duplicates or similar ones, so as to reduce the generation of iTrees and improve the effectiveness and stability of detection results.

## 2 The Proposed Solution

### 2.1 Fundamental iForest algorithm

The idea of the iForest algorithm comes from two characteristics of outliers[19]: (i) the exception data instance occupies a small part of the entire data-

set; (ii) their attribute values are greatly different from the normal data attribute values. That is to say, the outliers are usually "less and different", which makes them easily to be recognized by the outlier detection algorithm, i.e. "isolated".

The iForest uses the structure of the binary tree to define the iTree. Since the outliers are usually a small part of the entire dataset, iForest can isolate the outliers to the leaf nodes that are close to the root node, thereby identifying the outliers. The key to iForest algorithm is to construct iTrees so as to form the forest. For the convenience of description, the notion of iTree and calculation method of path length and outlier score are defined as follows.

**iTree** Assuming that $T$ is a node of an iTree, $T$ is either an external node (leaf node) with no child or an internal node with one test and exactly two child nodes ($T_l$, $T_r$). The test at node $T$ consists of an attribute $q$ (segmentation attribute) and a split value $p$ (segmentation value) so that the test $q < p$ determines the traversal of a data point to either $T_l$ or $T_r$. The data records less than the segmentation value are assigned to the left child node, while the data records larger than the segmentation value are assigned to the right child node. Repeat this process until the child node has only one data or has reached the maximum height of the iTree.

Let $X = \{x_1, x_2, \cdots, x_n\}$ be the given data set of a $d$-variate distribution. A sample of $\varphi$ instances $X' \subset X$ is used to build an iTree. $X'$ is recursively divided by randomly selecting an attribute $q$ and a split value $p$, until either: (i) the node has only one instance or (ii) all the data at the node have the same values. An iTree is a proper binary tree, where each node in the tree has exactly zero or two daughter nodes.

**Path length** The path length $h(x)$ of the data record $x$ refers to the number of edges $x$ traverses an iTree from the root node until the traversal is terminated at an external node.

**Outlier score** The path length of the data record in iForest algorithm is taken as an outlier score. Firstly, the length of the data record $x$ in different iTrees is solved. Then the average path length of iTrees is calculated as a normalization factor. Finally, the outlier score of data record $x$ is obtained by the normalization of path length. According to the binary search tree, for a sample instance with a given sample size of $\varphi$, the average length of the corresponding binary search tree is

$$c(\varphi) = \begin{cases} 2H(\varphi - 1) - \dfrac{2(\varphi - 1)}{\varphi} & \varphi > 2 \\ 1 & \varphi = 2 \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

where $H(\varphi)$ is the harmonic function and can be estimated by $\ln(\varphi) + 0.577\,215\,664\,9$ (Euler constant). As $c(\varphi)$ is the average of $h(x)$ given $\varphi$, it can be used to normalize $h(x)$. The outlier score $s$ of an instance $x$ is defined as

$$E(h(x)) = \sum_{i=1}^{n} h_i(x) \Big/ n \tag{2}$$

$$s(x, \varphi) = 2^{-\frac{E(h(x))}{c(\varphi)}} \tag{3}$$

where $h_i(x)$ represents the path length of data $x$ in the $i$th iTree and $E(h(x))$ is the average of $h(x)$ from a collection of iTrees. The following conditions provide three special values of the anomaly score.

(a) When $E(h(x)) \to 0$, $s \to 1$;

(b) When $E(h(x)) \to \varphi - 1$, $s \to 0$;

(c) When $E(h(x)) \to c(\varphi)$, $s \to 0.5$.

The process of detecting outliers by the iForest algorithm can be divided into two steps: (I) The training process. The original dataset is randomly sampled, and multiple iTrees are constructed according to the sub-dataset. Then the iForest is composed by the constructed iTrees. (II) The evaluation process. The outlier score is calculated based on the path length of the data record in the constructed iForest. For the calculation method of the outlier score established above, the following assessments can be made: (i) the closer the average path length $E(h(x))$ of data $x$ is to the average length $c(\varphi)$ of the corresponding binary search tree, the closer the outlier score $s$ is to 0.5. If all the instances return $s \approx 0.5$, the entire sample does not really have any distinct outliers. (ii) The closer $E(h(x))$ is to 0, the closer $s$ is to 1. If instances return $s$ very close

to 1, they are definitely outliers. (iii) The closer $E(h(x))$ is to the sampling size $\varphi$, the closer $s$ is to 0 for the outlier score. If instances have $s$ much smaller than 0.5, they are quite safe to be regarded as normal instances.

### 2.2 GA-iForest: iForest based on genetic algorithm

Although the iForest algorithm integrates the iTree through the idea of selective integration and realizes the accurate detection of abnormal data. However, it does not consider the difference between the iTrees (individual classifier), which will affect the final effect and the stability of the algorithm. In this paper, based on the original iForest algorithm, the idea of genetic algorithm is used to select the iTrees. After the selection of genetic algorithm, the iTrees with similarity, repetition and poor detection effect are removed, so as to improve the difference between individual classifiers and finally form a classifier with stronger generalization ability.

The iForest based on genetic algorithm (GA-iForest) consists of three algorithms, whose specific descriptions are shown in Algorithms 1, 2 and 3. Firstly, the iTree is constructed by sampling the original dataset. Secondly, the detection accuracy and the difference of the constructed iTrees are calculated according to the test data (randomly sampling from the original dataset and has no label). Thirdly, the iTrees with high detection accuracy and great difference are selected by the genetic algorithm according to the corresponding indexes. Finally, the selected iTrees are used to construct the iForest to calculate the outlier score, and the outlier is judged according to the outlier score. The general process of GA-iForest is shown in Fig. 1.

The process of detecting outliers for GA-iForest can be roughly divided into five steps: sampling, building iTrees, selecting iTrees, building iForest, and scoring outliers. The specific steps for building iTrees, selecting iTrees, and scoring outliers are described in Algorithms 1, 2 and 3.

In algorithm 1, there is an input parameter, which is the sampled data. Different sub-datasets

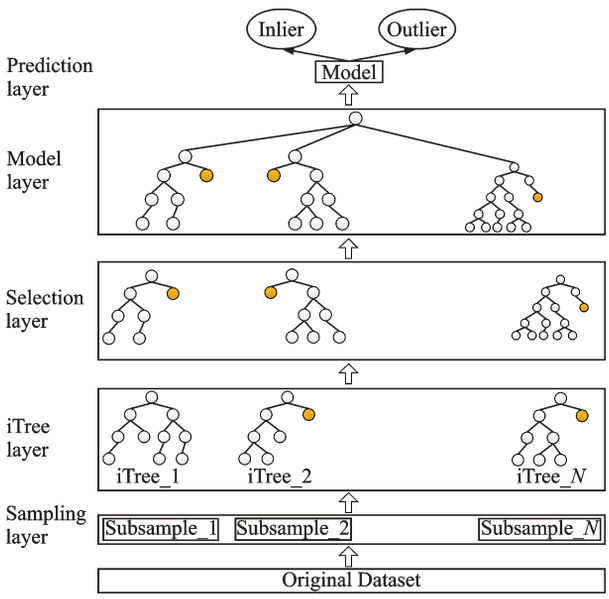


Fig.1 Framework of GA-iForest

Algorithm 1: iTree ($X'$)

Inputs: $X'$- input data
Output: an iTree
1. if $X'$ cannot be divided then
2. return external Node{size ← $|X'|$}
3. else
4. let $Q$ be a list of attributes in $X'$
5. randomly select an attribute $q \in Q$
6. randomly select an split value p between the max and min values of the $q$
7. $X_l \leftarrow \text{select}(X', q < p)$
8. $X_r \leftarrow \text{select}(X', q \geqslant p)$
9. return inNode{Left ← iTree($X_l$),
10. Right ← iTree($X_r$),
11. SplitAttribute ← $q$,
12. SplitValue ← $p$}
13. End if

are obtained by subsampling approach[20] randomly sampling the original dataset, and then an iTree is established on each dataset. The process of building an iTree is the same as the iForest mentioned above. And then the iTrees are selected according to the detection accuracy and difference between the iTrees. The specific process of selecting is described in algorithm 2. The fitness function and related concepts in algorithm 2 are defined as Eqs. (4—7).

In this algorithm, a part of the data (randomly sampling from the original dataset and has no label)

Algorithm 2: Select_iTree (iTrees)

Inputs: $M$ iTrees
Output: the $n$ optimal iTrees
1. encoding the iTrees according the accuracy and dissimilarity of each iTrees
2. initial population
3. calculate the fitness
4. for fitness value<expected and iterations <maximum do
5. nature select by the higher fitness with high probability of selection
6. cross with the probability $P_c$
7. calculate the fitness
8. if fitness value>expected and iterations >maximum then
9. End
10. else
11. variation with probability $P_m$
12. End if
13. End for
14. return $n$ optimal iTrees

is used to calculate the detection accuracy and difference of the iTree, and then the genetic algorithm is used to select the iTrees. The iTrees with high detection accuracy and large difference are selected and then the iForest can be generated.

Algorithm 3: Outlier ($D$,$N$,$M$)

Inputs: $D$ - input dataset, $N$ - the sample size, $M$ - the number of trees
Output: outliers
1. set the parameters of the iTree
2. for $i$=1 to $M$ do
3. $X' \leftarrow$ sample($D$,$N$)
4. iTree($X'$)
5. End for
6. Select_iTree(iTrees)
7. build the iForest according to the selected $n$ optimal iTrees
8. calculate path length to get the outlier scores
9. rank the outlier scores
10. return outliers

Algorithm 3 is based on algorithms 1 and 2. In this algorithm, the iTrees are generated by algorithm 1, and then the iTrees are selected by algorithm 2. Finally, the iForest is constructed to calculate the path length of the data point and determine which data is abnormal according to the path length. The GA-iForest consists of the above three algorithms. The process of selecting the iTrees by genetic algorithm is divided into two steps.

The first step is to define the accuracy index of the iTree and the similarity index between different iTrees. To reduce the similar and repeated iTrees, the degree of similarity between different iTrees is measured by cosine similarity. The $M$ datasets are obtained by randomly sampling the original data, and the $M$ iTree ($T_1, T_2, \cdots, T_M$) are established according to algorithm 1. Then a part of the data $D=\{d_1, d_2, \cdots, d_n\}$ (randomly sampling of the original data and has no label) is used to selecting iTrees. For each data sample $d_j (1 \leqslant j \leqslant n)$, if the score on $T_i$ is greater than 0.6 (this value is the abnormal value threshold), $r_{ij}=1$, otherwise 0. That is

$$r_{ij}=\begin{cases}1 & \text{score}(d_j)>0.6\\ 0 & \text{otherwise}\end{cases} \tag{4}$$

The result vector $V_i=\{r_{i1}, r_{i2}, \cdots, r_{in}\} (1 \leqslant i \leqslant M)$ is constructed for each iTree. The accuracy of each tree is defined as

$$a_i=\sum_{j=1}^{n} r_{ij} \qquad 1 \leqslant i \leqslant M \tag{5}$$

The similarity between the two trees $T_i$ and $T_j$ is

$$\cos\theta_{ij}=\frac{\boldsymbol{V}_i \cdot \boldsymbol{V}_j}{\|\boldsymbol{V}_i\| \times \|\boldsymbol{V}_j\|} \tag{6}$$

where "$\cdot$" represents the inner product of the vector, "$\times$" the normal multiplication, and $\|\cdot\|$ the length of the vector. Thus, the similarity coefficient matrix between $M$ iTrees can be constructed as

$$\mathbf{Diff}=\begin{bmatrix}\cos\theta_{11} & \cos\theta_{12} & \cdots & \cos\theta_{1M}\\ \cos\theta_{21} & \cos\theta_{22} & \cdots & \cos\theta_{2M}\\ \vdots & \vdots & \ddots & \vdots\\ \cos\theta_{M1} & \cos\theta_{M2} & \cdots & \cos\theta_{MM}\end{bmatrix} \tag{7}$$

It can be known from the linear algebra, if the similarity between two iTrees is lower, the value of cosine is close to $-1$. The higher of the similarity, the closer the cosine value is to 1. And the value of cosine on $[-1, 1]$ increases with the increase of the similarity.

In the second step, according to the difference

index and the accuracy index, different test data (random sampling of the original dataset has no label) are used to compute the detection accuracy and similarity of different iTrees. Then, according to the detection accuracy of the iTrees and the dissimilarity between different iTrees, the fitness function is constructed. Finally the genetic algorithm is used to select the iTrees with high detection accuracy and small similarity. The fitness function is

$$f(T_i) = \frac{w_1}{\cos\theta_{ij}} + w_2 \times a_j \qquad 1 \leqslant i,j \leqslant M \quad (8)$$

where $f(T_i)$ represents the fitness function of $T_i$, $\cos\theta_{ij}$ the phase difference between $T_i$ and $T_j$, $a_j$ the accuracy of $T_j$, and $w_1$ and $w_2$ are the weights corresponding to the phase difference and accuracy, respectively.

The constructed $M$ iTrees are selected according to the detection accuracy and the difference indexes defined above, and the specific steps are as follows.

(1) Individual coding: encoding the accuracy and difference of each iTrees. The accuracy uses the binary coding with the length $\lceil \log_2^n \rceil$, and the difference uses the floating-point coding.

(2) Generation of initial population: a population of size $n$ is generated, and the encoding format of each individual is (01110…1, 0.54). The first dimension is the accuracy coding, and the second dimension is the difference coding with a floating point number between $-1$ and 1.

(3) Fitness calculation: the individual fitness is calculated according to the fitness function defined by Eq. (8). Fitness value determines the merits of each iTrees and the chance of inheritance to the next generation.

(4) Selecting the operation: the specific process includes: (i) Calculating the total fitness $\sum_{i=1}^{n} f(T_i)$ of all individuals in the group; (ii) Calculating the relative fitness of the each individual $P(T_i) = f(T_i) / \sum_{k=1}^{n} f(T_k)$ and it is used as the group probability of each individual to inherit to the next generation; (iii) Calculating the cumulative probability $q_i = \sum_{k=1}^{i} P(T_k)$ of each individual; (iv) Generating a random number $r$ in the interval of [0, 1]; (v) If $r < q_1$, select individual 1, otherwise select individual $k$ such that $P_{k-1} < r \leqslant P_k$ holds; (vi) repeat (iv) and (v) $n$ times;

(5) Crossing: according to the probability $P_c$ to exchange some codes between two individuals, a single point crossing method is adopted, such as the crossing of individual $T_1 = (011010\cdots0, 0.42)$ and individual $T_2 = (101010\cdots1, 0.35)$ at position 1. $T_t$ turns into $T_1 = (111010\cdots0, 0.42)$ and $T_2 = (001010\cdots1, 0.35)$ after crossing or the exchange of the second-dimensional decimal.

(6) Variation: the coding of a certain position of the individual was changed according to the probability $P_m$, for example, the fourth position of the first dimension of the individual $T_i = (101010\cdots0, 0.42)$ is mutated. It turns to $T_i = (101110\cdots0, 0.42)$ after the variation.

## 3 Analysis of Algorithm Complexity

### 3.1 Analysis of time complexity

In this paper, the genetic algorithm is used to realize the improvement of iForest. The algorithm complexity of iForest is analyzed firstly. In the iForest algorithm, the storage structure of the binary tree is used to construct the iTrees, and then the corresponding path length is calculated. For a binary tree, in the worst case where all nodes have only left children or only right children, the time of each search operation is $T(n) = cn$ and the traversal can only be assigned into the left or the right subsequence each time. So the time complexity satisfies

$$\begin{aligned} T(n) &= T(n-1) + cn = \\ &T(n-2) + c(n-1) + cn = \\ &T(n-3) + c(n-2) + c(n-1) + cn = \\ &T(1) + 2c + 3c + \cdots + cn = O(n^2) \end{aligned} \quad (9)$$

In the average case, the binary tree is a full binary tree. Each search only needs to traverse half of the sequence, and then each traversal sequence is divided into two halves each time, so the time com-

plexity satisfies

$$T(n)=2T(n/2)+cn/2$$

$$T(n)=2\left[2T\left(n/2^2\right)+cn/2^2\right]+cn/2$$

$$\cdots$$

$$T(n)=2^k\,T\left(n/2^K\right)+kcn/2 \tag{10}$$

Supposing that $2^k=n$, then

$$T(n)=O(n\log n) \tag{11}$$

For the iForest, the average time complexity is $O(n\log n)$.

For the genetic algorithm used in this paper, it can be seen as a ONE-MAX problem, which is a GA-easy problem. The problem can be described as

$$\max\{f(x);x\in X\} \tag{12}$$

where $x=(x_1,x_2,\cdots,x_n)$ is the individual code, $x_i\in\{0,1\}$; $n$ the code length; the fitness function $f(x)=x_1+x_2+\cdots+x_n$. The optimal solution is $x^*=(11\cdots1)$, and the optimal function value is $f(x)=n$. The following genetic algorithm is constructed to solve the time complexity of the above problems.

$N$ samples are selected as the initial population, denoted as $\xi_0$. Let $f(\xi_0)=0$ and $k=0$.

Reorganization operation: For any reorganization operation, generating a new population, denoted as $\xi_i$; Mutation operation: For each individual $x_i$ (binary string) in $\xi_i$, one bit is randomly selected and converted into its complement to generation a new individual $y_i$ and a new population $\xi_N=\{y_1,y_2,\cdots,y_n\}$; Selection operation: Select $N$ individuals from the population $\xi_N$ as the next generation population, denoted as $\xi_{N+1}$, and concatenate $k=k+1$. If $f(\xi_{N+1})=n$, the process is ended, otherwise the above process is repeated.

Let time $T=\min\limits_k\{f(\xi_k)=n\}$, the expectation of this time is considered as the average calculation time. Let $f(\xi_0)=0$, then

$$E[T|f(\xi_0)=0]\leqslant O(n^2)$$

**Prove** Let $T_0=0$, $T_1=T_0+\min\limits_k\{f(\xi_{k+T_0})\geqslant 1\}$, $\cdots$ and $T_n=T_{n-1}+\min\limits_k\{f(\xi_{k+T_{n-1}})\geqslant n\}$, then

$$T=T_1-T_0+T_2-T_1+\cdots+T_n-T_{n-1}$$

For a given $m(1\leqslant m\leqslant n)$, let $T'_m=T_{m-1}+\min\limits_k\{f(\xi_{k+T_{m-1}})=m\}$, on the one hand

$$T_m-T_{m-1}=\min_k\{f(\xi_{k+T_{m-1}})\geqslant m\}\geqslant \min_k\{f(\xi_{k+T_{m-1}})=m\}=T'_m-T_{m-1}$$

On the other hand,

$$T_m-T_{m-1}=\min_k\{f(\xi_{k+T_{m-1}})\geqslant m\}\leqslant M\min_k\{f(\xi_{k+T_{m-1}})=m\}$$

So

$$E(T'_m-T_{m-1})\leqslant E(T_m-T_{m-1})\leqslant ME(T'_m-T_{m-1})$$

Let the population $\xi'$ be the progeny population after a cycle of population $\xi$, since $P(f(\xi')<m-1|f(\xi)=m-1)=0$, therefore

$$P(f(\xi')=m|f(\xi)=m-1)\geqslant 1/n$$

$$\text{So } E(T'_m-T_{m-1})=1/P(f(\xi')=m|f(\xi)=m-1)\leqslant n$$

$$n^2\leqslant E(T|f(\xi_0)=0)\leqslant Mn^2$$

Therefore, the time complexity of genetic algorithm application in this paper is $O(n^2)$.

### 3.2 Analysis of spatial complexity

In this paper, the iTrees need to be constructed in the process of detecting outliers, and the construction process of iTrees is similar to the construction process of the binary tree. Therefore, the spatial complexity of the method is the same as that of the binary tree, namely $O(N)$.

## 4 Experiment

In this paper, the "Spark on Yarn" cluster mode is used to construct the experimental environment of outlier detection. The experiment uses three virtual machines as the outlier detection nodes, one of which is the master and the other two are the slaves. Each node uses the Ubuntu-14.04 operating system, Hadoop-2.7.6, Spark-2.1.1-bin-hadoop2.7, Scala-2.11.7, and JDK-1.8.0_16 to build a software environment. The three virtual machines are all 4 G memory. In this paper, the parameter setting of the iForest and the GA-iForest is set by the default, that is, the number of iTree is 100 and the sampling size is 256. In the LOF algorithm, the parameter $K$ is settled to 6. The experimental datasets are all from ODDS, and the specific information of the dataset is shown in Table 1.

**Table 1 Information of experiment dataset**

| Dataset | No. of points | No. of dimension | Outliers (Percentage/%) |
|---|---|---|---|
| Annthyroid | 7 200 | 6 | 534(7.42) |
| Arrhythmia | 452 | 274 | 66(15) |
| Breastw | 683 | 9 | 239(35) |
| Forestcover | 286 048 | 10 | 274 7(0.9) |
| Pendigits | 6 870 | 16 | 156(2.27) |
| Mammography | 11 183 | 6 | 260(2.32) |
| Mulcross | 262 144 | 4 | 262 14(10) |
| Cardio | 1 831 | 21 | 176(9.6) |

To verify that as the number of iTrees increases, the similarity between them gradually increases, the final detection accuracy is compared by changing the number of iTrees. In the experiment, two datasets of Annthyroid and Arrhythmia are selected as the representatives and AUC is used as the comparison index. The results are shown in Figs. 2 and 3.

As can be seen from Figs.2 and 3, as the number of iTrees increases gradually, the value of AUC gradually tends to be stable. In Fig. 2, when the number of iTrees is around 80, the change of detection accuracy AUC tends to be flat. While in Fig.3, when the number of iTrees is around 60, the detection accuracy AUC starts to be flat. This indicates that as the number of iTrees increases, the similarity between them increases gradually, leading to no significant improvement in detection accuracy.

**Experiment 1** Comparison of detection accuracy and actual number of iTrees

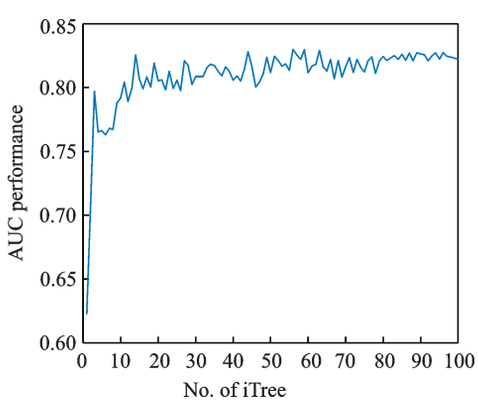


Fig.2 Detection performance AUC varying with the number of iTrees on Annthyroid dataset

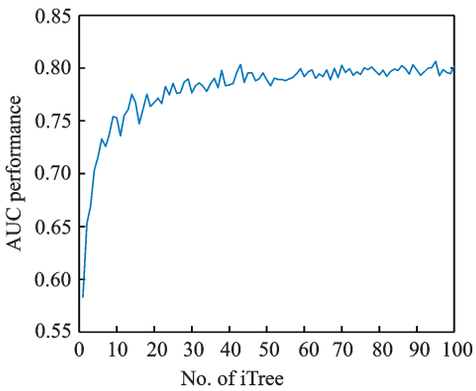


Fig.3 Detection performance AUC varying with the number of iTrees on Arrhythmia dataset

To verify the detection accuracy of GA-iForest, the datasets in Table 1 are used as the test data. For the sake of ensuring the credibility of the experiment, the experiment is performed 10 times and the average value is taken as the final result on each dataset. For the verification of the detection accuracy, ROC curve and the value of AUC are selected as metrics, whose results are shown in Figs.4, 5 and 6.

From the above experiment, it can be seen that the ROC curve of GA-iForest can completely cover the ROC curves of iForest and LOF. It indicates the detection accuracy of the GA-iForest is better than iForest and LOF in Breastw and Forestcover dataset. The GA-iForest selects high-performance iTrees and optimizes the construction process of iForest. Therefore, GA-iForest algorithm can improve the detection accuracy of outliers. To fully verify the superiority of the detection accuracy of the GA-iForest to iForest and LOF, experiments on several other datasets are carried out, whose results are shown in Fig.6.

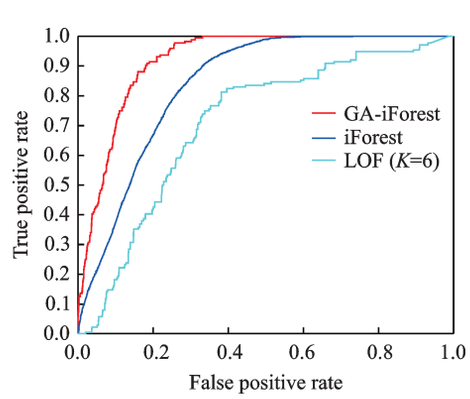


Fig.4 ROC curve on Annthyroid dataset

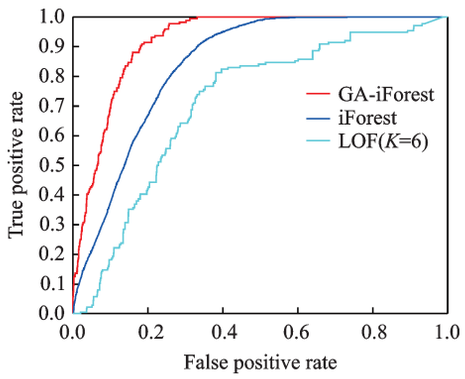


Fig.5 ROC curve on Forestcover dataset

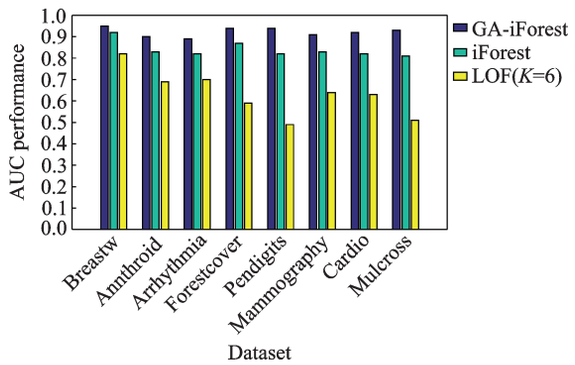


Fig.6 AUC performance on the other datasets

It can be seen from the results of ROC curve and histogram of AUC performance that the proposed method GA-iForest is superior to the iForest and LOF in terms of detection accuracy. In this paper, efficient detection of abnormal data is achieved by filtering similar or repeated iTrees. Due to different datasets, the number of iTree reductions on each dataset is not equal, the specific results of iTrees reduction are shown in Table 2.

**Experiment 2** Comparison of detection time

To compare the execution time of each algorithm, the above datasets are also used in this experiment. For GA-iForest and iForest algorithms, the training time and prediction time are selected as the comparison basis, and their total time is calculated. For the LOF algorithm, there is no training and prediction process, so the total time is directly used as the comparison basis. The specific running time of each algorithm is shown in Table 3.

**Table 2 Reduction of iTrees on different datasets**

| Dataset | Original iTrees numbers | Selected iTrees | Reduction ratio /% |
|---|---|---|---|
| Annthyroid | 100 | 80 | 20 |
| Arrhythmia | 100 | 63 | 37 |
| Breastw | 100 | 78 | 22 |
| Forestcover | 100 | 90 | 10 |
| Pendigits | 100 | 76 | 24 |
| Mammography | 100 | 65 | 35 |
| Mulcross | 100 | 84 | 16 |
| Cardio | 100 | 70 | 30 |

From Table 3, it can be seen that in terms of total time, the GA-iForest has no significant improvement compared with the iForest algorithm. The reason is that GA-iForest needs a selection process when training. For the sake of comparison, the predicted time and training time in Table 3 are made into a histogram, whose results are shown in Figs.7 and 8.

From the perspective of training time and prediction time, the GA-iForest requires longer training time and shorter prediction time. The reason lies in that the method in this paper will select the iTree when constructing the iForest, so the training time will be longer than the original method. Through the selection of genetic algorithm, the iTrees with partial similarity, repetition and poor detection ef-

**Table 3 Comparison of execution time** s

| Dataset | GA-iForest | | | iForest | | | LOF |
|---|---|---|---|---|---|---|---|
| | Train | Prediction | Total | Train | Prediction | Total | |
| Breastw | 0.34 | 0.12 | 0.46 | 0.15 | 0.21 | 0.36 | 1.80 |
| Annthyroid | 1.35 | 1.37 | 2.72 | 0.56 | 2.14 | 2.70 | 73.54 |
| Arrhythmia | 0.86 | 0.32 | 1.18 | 0.24 | 0.71 | 0.95 | 7.24 |
| Forestcover | 3.26 | 10.31 | 13.57 | 1.20 | 17.31 | 18.51 | 234 180.14 |
| Pendigits | 0.51 | 0.32 | 0.83 | 0.23 | 0.48 | 0.71 | 2.78 |
| Mammography | 4.15 | 0.76 | 4.91 | 3.42 | 1.25 | 4.67 | 14 847.23 |
| Cardio | 1.53 | 0.38 | 1.91 | 1.24 | 0.56 | 1.80 | 5.48 |
| Mulcross | 1.84 | 6.45 | 8.29 | 1.57 | 10.34 | 11.91 | 26 381.4 |

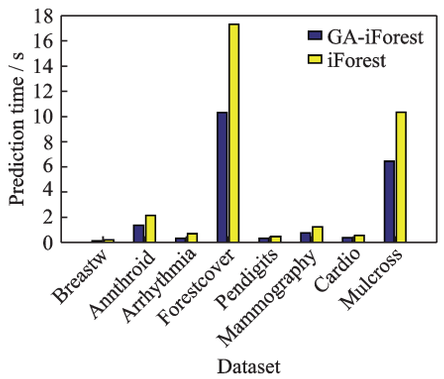


Fig.7 Comparison of prediction time

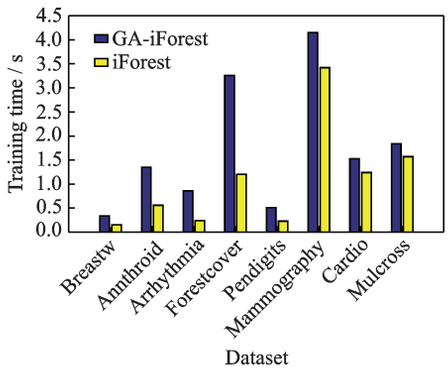


Fig.8 Comparison of train time

fect are reduced, so the prediction time will be improved to a certain extent.

**Experiment 3** Comparison of algorithm stability

To compare the stability, the amount of change in detection accuracy AUC is chosen as the measurement index. The iTree established by iForest through random sampling will have similar structure or repeated iTrees. Due to the randomness of sampling, the results of each detection may be different. GA-iForest selects the iTrees with high detection accuracy and large difference for outlier detection by filtering the established iTrees. Although sampling is random, there is no similar or repeated iTrees after selection, which improves the stability of detection results.

The construction of iTrees is based on randomly sampled dataset. To compare the stability of the detection results, 10 times experiments are used on a dataset (Mammography), and the mean detection accuracy AUC and the standard deviation of AUC for each experiment are compared. Fig.9 shows the variation of the detection accuracy of the three algorithms on Mammography dataset. Table 4 shows the average detection accuracy AUC and standard deviation of AUC of the three algorithms on different datasets.

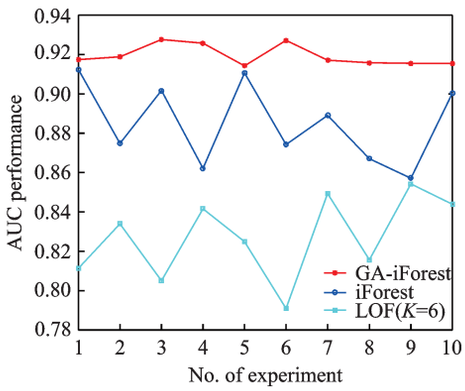


Fig.9 The variation of detection accuracy

From Fig.9, it can be seen that GA-iForest has a smaller fluctuation range compared with iForest and LOF. It indicates that GA-iForest has the better stability on the Mammography. According to the calculation, the standard deviation of GA-iForest is 0.005 while iForest and LOF are 0.058 and 0.041, respectively. The variation of detection accuracy AUC is also compared on the other datasets, whose results are shown in Table 4. From Table 4, it can be known that the mean detection accuracy AUC of GA-iForest is higher than that of LOF and iForest, and the standard deviation of AUC is lower than that of LOF and iForest. It can be concluded that the stability of GA-iForest is better than LOF and iForest. The reason why GA-iForest is more stable in detecting accuracy is that the selection oper-

**Table 4 The mean and standard deviation of AUC performance**

| Dataset | AUC performance | | | | | |
|---|---|---|---|---|---|---|
| | Mean | | | Standard deviation | | |
| | GA-iForest | iForest | LOF ($K$=6) | GA-iForest | iForest | LOF ($K$=6) |
| Breastw | 1.000 | 0.988 | 0.836 | 0.002 | 0.013 | 0.010 |
| Annthyroid | 0.896 | 0.801 | 0.780 | 0.004 | 0.025 | 0.037 |
| Arrhythmia | 0.889 | 0.836 | 0.818 | 0.007 | 0.094 | 0.054 |
| Forestcover | 0.893 | 0.875 | 0.642 | 0.003 | 0.031 | 0.026 |
| Pendigits | 0.984 | 0.951 | 0.863 | 0.007 | 0.091 | 0.058 |
| Mammography | 0.921 | 0.873 | 0.894 | 0.005 | 0.058 | 0.041 |
| Cardio | 0.948 | 0.920 | 0.886 | 0.001 | 0.016 | 0.034 |
| Mulcross | 0.947 | 0.886 | 0.753 | 0.002 | 0.029 | 0.013 |

ation of iTrees is added on the basis of the original iForest algorithm. Therefore, the uncertainty and similar iTree caused by random sampling is reduced.

## 5 Conclusions

In this paper, an iForest algorithm based on genetic algorithm GA-iForest is designed on the basis of the iForest algorithm. The method uses genetic algorithm to select the high-performance iTrees to construct the iForest and optimize the construction of the iForest. The optimized iForest does not have similar, duplicate and low detection accuracy iTrees, which saves memory space and improves the detection accuracy compared with the original iForest algorithm. However, GA-iForest needs a more stage of selection when constructing the iForest, which will lead to an increase in the training time. In the future work, how to improve the training process of the proposed method will be studied.

**References**


[1] WANG Lili, PENG Bo. Track clustering based on LOFC time window segmentation algorithm[J]. Journal of Nanjing University of Aeronautics and Astronautics, 2018, 50(5): 661-665. (in Chinese)

[2] HAWKINS D M. Identification of outliers [M]. 1st Edition. Berlin: Springer Netherlands, 1980: 11-12.

[3] ZHANG K, HUTTER M, JIN H. A new local distance-based outlier detection approach for scattered real-world data[C]// Pacific-Asia Conference on Knowledge Discovery and Data Mining. Berlin, Heidelberg: Springer, 2009: 813-822.

[4] XU H, MAO R, LIAO H, et al. Closest neighbors excluded outlier detection[C]// Online Analysis & Computing Science. Chongqing, China: IEEE, 2016: 105-110.

[5] LEYS C, KLEIN O, DOMINICY Y, et al. Detecting multivariate outliers: Use a robust variant of the Mahalanobis distance [J]. Journal of Experimental Social Psychology, 2018, 74: 150-156.

[6] LIU J, WANG G. Outlier detection based on local minima density[C]// Information Technology, Networking, Electronic & Automation Control Conference. Chongqing, China: IEEE, 2016: 718-723.

[7] TANG B, HE H. A local density-based approach for outlier detection [J]. Neurocomputing, 2017, 241: 171-180.

[8] KELLER F, MULLER E, BOHM K. HiCS: High contrast subspaces for density-based outlier ranking[C]// IEEE International Conference on Data Engineering. Washing DC, USA: IEEE, 2012: 1037-1048.

[9] MAHESHWARI K, SINGH M. Outlier detection using divide-and-conquer strategy in density based clustering[C]// International Conference on Recent Advances & Innovations in Engineering. Jaipur, India: IEEE, 2017: 1-5.

[10] VWEMA P, YADAVA R D S. Fuzzy c-means clustering based outlier detection for SAW electronic nose[C]//2017 2nd International Conference for Convergence in Technology. Mumbai, India: IEEE, 2017: 513-519.

[11] CHRISTY A, GANDHI G M, VAITHYASUBRAMNIAN S. Cluster based outlier detection algorithm for healthcare data [J]. Procardia Computer Science, 2015, 50: 209-215

[12] LIU F T, TING K M, ZHOU Z H. Isolation-based anomaly detection [J]. ACM Transactions on Knowledge Discovery from Data, 2012, 6(1): 1-39.

[13] STRPLING E, BAESENS B, CHIZI B, et al. Isolation-based conditional anomaly detection on mixed-attribute data to uncover workers′ compensation fraud[J]. Decision Support Systems, 2018, 111: 13-26.

[14] HAUTAMAKI V, KARKKAINEN I, FRANTI P. Outlier detection using *k*-nearest neighbour graph[C]// International Conference on Pattern Recognition. Cambridge, UK: IEEE, 2004: 430-433.

[15] BREUNING M M, KRIEGEL H P, NG R T. LOF: Identifying density-based local outliers[C]// ACM Sigmoid International Conference on Management of Data. New York, USA: ACM, 2000: 93-104.

[16] ZHANG S M, LUO X Y, WANG B Y. An improved outlier detection algorithm K-LOF based on density [J]. Computing, Performance and Communication System, 2017, 2: 1-7.

[17] ARYAL S, TING K.M, WELLS J.R, at al. Improving iForest with relative mass[C]// Advances in Knowledge Discovery and Data Mining. Cham, Switzerland: Springer, 2014: 510-522.

[18] ZHANG X, DOU W, HE Q, et al. LSHiForest: A generic framework for fast tree isolation based ensemble anomaly analysis[C]// 2017 IEEE 33rd International Conference on Data Engineering. San Diego, CA, USA: IEEE Computer Society, 2017: 983-994.

[19] HOU Y X, DUAN L, QIN J L, et al. Parallel detection design based on iforest [J]. Journal of Computer Engineering and Science, 2017, 39(2): 236-244.

[20] AGGARWAL C C. Outlier ensembles: Position paper[C]//ACM SIGKDD Explorations Newsletter. New York, USA: ACM, 2013, 13(2):49-58.

**Acknowledgements** This work was supported by the State Grid Liaoning Electric Power Supply CO, LTD. We are grateful to the reviewers who have given their support and valuable comments and the financial support for the "Key Technology and Application Research of the Self-Service Grid Big Data Governance (No. SGLNXT00YJJS1800110)".

**Authors** Mr. **LI Kexin** is currently a M.S. candidate at College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics (NUAA). His major research interests include data mining and software verification.

Dr. **LI Jing** received her Ph.D. degree in Computer Science and Technology from Nanjing University in 2004. She is currently an associate professor at College of Computer Science and Technology, NUAA. Her main research interests include computer vision, software verification, data mining and so on.

Mr. **LIU Shuji** is currently a senior engineer of State Grid Liaoning Electric Power Supply CO, LTD. His research interests include power data governance and data visualization.

Mr. **LI Zhao** is currently a senior engineer of State Grid Liaoning Electric Power Supply CO, LTD.

Ms. **BO Jue** received her double B.S. degree in electrical engineering & automation and computer science & technology from Changchun Institute of Technology in 2009. She received her M.S. degree in electrical engineering from Northeast Electric Power University in 2015. Her main research interests include the interaction between systems and data, data governance remote, database disaster recovery and data transmission.

Ms. **LIU Biqi** received her M.S. degree in faculty of engineering from Imperial College London in 2017. Her research interests are focused on data mining.

**Author contributions** Mr. **LI Kexin** proposed the method of this paper, conducted the analysis and wrote the manuscript. Dr. **LI Jing** gave detailed guidance on the preparation of the paper and the operation of the experiment. Mr. **LIU Shuji** and Mr. **LI Zhao** contributed to the discussion and background of the study. Ms. **BO Jue** and Ms. **LIU Biqi** provided the data needed for the implementation, and analyzed the problems encountered during the experiment.

**Competing interests** The authors declare no competing interests.

(Production Editor: Wang Jing)