The Journal of Supercomputing (2021) 77:3062–3076
https://doi.org/10.1007/s11227-020-03384-x

An explainable outlier detection method using region-partition trees

Cheong Hee Park[1] · **Jiil Kim**[1]

Published online: 20 July 2020

Abstract

Most outlier detection methods output outlier score that measures the degree of deviation of a data sample from a normal data pattern. However, it is difficult to choose an optimal threshold on outlier scores by which outliers and normal data samples can be distinguished. In this paper, we propose a tree-based outlier detection method which computes normalized outlier scores for data samples. In particular, without the need to determine the threshold for outlier score it provides binary labels for outlier prediction. By using training data which consists of normal data samples, the proposed method builds a multi-way splitting tree, called region-partition tree (RP-tree), where normal data region is effectively described by the partition of data region into leaf nodes. By utilizing region-partition table (RP-table) which stores the information for splitting attributes and interval partition, RP-tree can be constructed so as to finely split the normal data region but keep the size of a tree be reasonably small. From the ensemble of RP-trees, the proposed method computes the normalized outlier scores ranging in [0, 1] and data samples with outlier score of 1 are predicted as outliers. Also it identifies the attributes responsible for outlier prediction. Experimental results demonstrate the outlier detection performance of the proposed method. The proposed method obtained an average F1-value of 0.72 and an AUC score of 0.96, while the second highest performance in the compared methods was an F1-value of 0.57 and an AUC score of 0.94, respectively.

Keywords Normalized outlier score · Outlier detection · Region-partition table · Region-partition tree · Ensemble method

✉ Cheong Hee Park
cheonghee@cnu.ac.kr

Jiil Kim
kimjiil2013@naver.com

[1] Department of Computer Science and Engineering, Chungnam National University, Daejeon, Korea

1 Introduction

An outlier is defined as an observation which deviates so much from other observations enough to arouse suspicions that it was generated by a different mechanism[1, 2]. Outlier detection can be applied to various problems such as fraud detection, intrusion detection in computer networks, system fault detection, and unexpected error detection in databases. Outlier score that represents the degree of deviation is a general form of output in most outlier detection algorithms. By using outlier scores, the top-k data samples with the highest scores can be identified as outliers or threshold are used to distinguish outliers. The mean and standard deviation in outlier scores of normal training data samples can be used to set the threshold, or validation set comprised of outliers can be utilized [3, 4]. However, it is difficult to choose an optimal threshold for outlier score in various data sets [5]. Another difficulty with learning an outlier detection model is that outliers occur at a very small rate and collecting outliers can be demanding in some application areas such as system fault detection.

Isolation forest [6] and random-split forest (RS-forest) [7] are well-known tree-based outlier detection methods. Tree structure was used to compactly describe the distribution of training data, and outlier scores were computed using root-leaf path lengths in Isolation forest or densities of leaf nodes in random-split forest. However, from some preliminary experiments on Isolation Forest and random-split forest using various data sets, we found that setting a threshold on outlier scores by which outliers and normal data can be distinguished is quite difficult depending on data characteristics. That motivated us to devise an outlier detection method which can give binary labeling for outlier prediction as well as normalized outlier scores.

In this paper, we propose a tree-based outlier detection method. Given a training data consisting of normal data samples, the proposed method builds a multi-way splitting tree, called region-partition tree (RP-tree), where normal data region is effectively described by the partition of data region into leaf nodes. In order to construct a tree which finely splits the normal data region but keep tree size small, we devise a table which guides the construction of RP-tree, called region-partition table (RP-table). RP-table stores the information for splitting attributes and interval partition. In the process of growing the RP-tree in a top-down manner, by referring to the RP-table, only nodes corresponding to the interval to which normal data samples belong are constructed. From the ensemble of multiple RP-trees, the normalized outlier scores ranging in $[0, 1]$ are computed. Without the need to determine the threshold on outlier scores, data samples with outlier score of 1 are predicted as outliers, therefore binary labeling for normality and abnormality is automatically obtained. Moreover, the proposed method can identify the attributes responsible for outlier designation, and the degree to which each attribute has played a role in outlier prediction is estimated. The performance of the proposed method is demonstrated experimentally using real and artificial data sets.

The contribution of this paper can be summarized as follows:

- We propose an outlier detection method based on region-partition trees (RP-trees). A RP-tree effectively represents a partition of the normal data region.
- Region-partition table (RP-table) is introduced in order to keep the size of RP-tree small by preventing the generation of redundant nodes.
- The proposed method gives the normalized outlier scores ranging in [0, 1], where a data sample with the outlier score of 1 is declared as an outlier.
- The proposed method can identify the attributes responsible for outlier prediction and visualize the outlier region where outliers arise.

The remainder of the paper is organized as follows. In Sect. 2, related work for outlier detection is reviewed. In Sect. 3, we propose an outlier detection method using the ensemble of region-partition trees. Experimental results are given in Sect. 4 and discussion follows in Sect. 5.

2 Related work

Outlier detection can be performed in a supervised mode, where data samples that are labeled as "normal" or "abnormal" are given and a classifier trained from the data is used for detecting abnormal instances in test data [8, 9]. Although various classifiers can be applied, considering that obtaining a sufficient number of outliers is difficult, it may not be a practical approach in real application areas. On the other hand, under the assumption that most data instances are normal, the instances that seem to fit least to the rest of the data can be detected. It is referred as unsupervised outlier detection. Many outlier detection methods belong to this category.

In the distance-based outlier detection method in [10], for given parameters k and R, a data sample is considered an outlier if less than k data samples are within distance R from it. When there exist clusters of different densities, the distance-based outlier detection method has difficulty with choosing appropriate k and R values. Overcoming the difficulty in such circumstance, the method in [11] gives an outlier score, local outlier factor (LOF), based on local density around a data instance. LOF provides an indication of whether a data sample is in a denser or sparser region of the neighborhood compared with its neighbors.

The clustering-based outlier detection method in [12] identified small-sized clusters as outliers among the clusters generated by clustering analysis on a given data set. In [13], cluster-based local outlier factor was proposed based on the size of the cluster to which the object belongs or the distance between the object and its closest cluster.

When a training set consists of normal data, a model representing normal behavior can be built and the likelihood of a test instance to be generated from the model is computed. Outlier detection models based on reconstruction errors usually belong to this category. In [3, 4, 14], a deep-learning model to reconstruct normal data is trained, and samples with high reconstruction error are identified as outliers. Recently, outlier detection methods using Autoencoders or Generative Adversarial Network(GAN) have been proposed [15, 16].

Isolation Forest [6] and random-split forest (RS-Forest) [7] are tree-based outlier detection methods. Isolation Forest constructs a tree ensemble from unlabeled data samples. Isolation tree is built by repeating the random selection of an attribute and a splitting position on each node until all instances are isolated in leaf nodes or designated tree height is reached. Isolation trees constructed from subsampling of training data compose Isolation forest. Based on the idea that outliers are more susceptible to isolation than normal data, the length of a path where a data sample traverses from the root node to an external node is used to compute an outlier score.

While Isolation Forest builds trees using unlabeled data samples, RS-Forest constructs complete binary trees based on the data range derived from a training set composed of normal data. After building tree structure by repeated random selection of an attribute and a splitting value within the data range, normal training data samples are inserted to the tree, and a node profile which is the number of instances falling into the node is computed. Outlier score of an incoming instance is computed by the density of a node which is obtained from node profiles and node volume estimation.

Outlier detection methods have been used in various application areas. The paper [17] addressed how outlier detection techniques can be used for practical use cases and real-life problems in the business landscape. In [18], Isolation Forest was applied on a real industrial data set related to etching which is one of semiconductor manufacturing processes. The monitoring was performed exploiting Optical Spectroscopy Data. In [19], the performance of unsupervised techniques such as LOF, one class SVM, K-means and Isolation Forest was compared for credit card fraud detection.

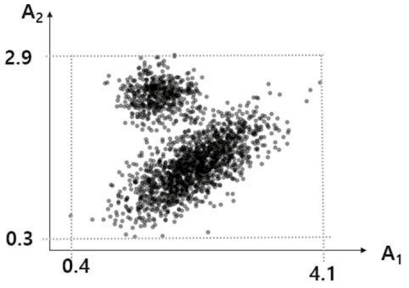
3 The proposed outlier detection method

The proposed method is based on a K-way tree that can have a maximum of K child nodes. When the height of a tree is H, it has a maximum of $\frac{K^{H+1}-1}{K-1}$ nodes, which means that it is difficult to cope with the size of a tree even if H increases slightly. However, in order to represent a normal data region accurately, the data region must be finely divided and a sufficiently large value of H and K should be set accordingly.

For a tree structure with a reasonable size while finely dividing a data area, we first construct a table that stores splitting attributes and partition information. To construct a tree of height H and degree K, the information of H attributes and K subintervals of each attribute is stored in the table, and it is used for the node configuration in the level from 0 to $H-1$. In the proposed outlier detection method, only nodes on the path through which the normal data passes are generated by using the information in the table. Hence, the leaf nodes in the depth H correspond to a partition of the normal data region. The procedure of the proposed method including the construction of region-partition table (RP-table), the construction of region-partition tree (RP-tree), and outlier prediction by the ensemble approach of region-partition forest (RP-forest) is described in detail in the following subsections.

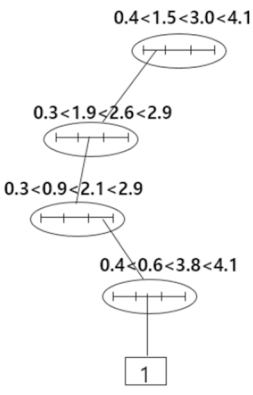
3.1 The construction of region-partition table

When the height of a tree is set to H, the H attributes are selected randomly with uniformly distributed probability, each of which will be used as a splitting attribute in each level from 0 to $H-1$ of a tree. When attribute A_i is selected, we can compute the range of values of attribute A_i from training data containing only normal data. By selecting $K-1$ partition values within the range of A_i values, we can divide the range of A_i into K subintervals. The table stores the selected attributes and partition information so that they are used to guide the node construction process at each level of the tree. We call the table region-partition table (RP-table). Suppose we have training data consisting of normal data samples in Fig. 1a. Figure 1b shows an example of RP-table of height $H = 4$ and degree $K = 3$ which can be constructed from training data in Fig. 1a.



level	selected attributes	partition points
0	A_1	0.4 1.5 3.0 4.1
1	A_2	0.3 1.9 2.6 2.9
2	A_2	0.3 0.9 2.1 2.9
3	A_1	0.4 0.6 3.8 4.1

(b)



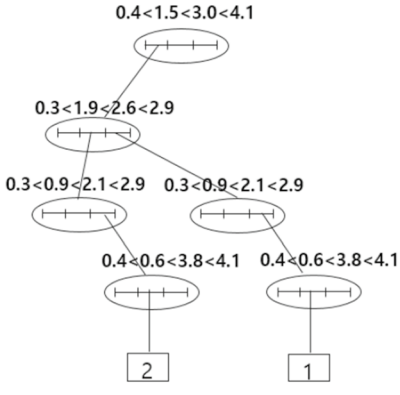


Fig. 1 **a** An example of normal training data. **b** An example of RP-table of height $H = 4$ and degree $K = 3$. **c** The tree grown after inserting the first data sample (1.4, 2.3). **d** The tree grown after inserting the second data sample (1.2, 2.4) and the third data sample (1.0, 2.7)

3.2 The construction of region-partition tree

Region-partition tree (RP-tree) is constructed by starting from the empty tree, inserting a training data sample into the tree one by one, and creating a new child node if the child node of the interval to which the data sample belongs was not generated before.

When the initial training data sample comes in as input, it constructs a root node with the attribute and interval information corresponding to level 0 stored in the RP-table. A child node of the interval including the training data sample is generated using the partition information and it is repeatedly until reaching the level $H-1$. A leaf node is finally created in the level H where the number of data samples which have arrived in the leaf node is accumulated. If a data sample belongs to an interval in which a child node has already been created, it descends to the child node and repeats the process until it reaches the leaf in the depth H. Figure 1c illustrates the tree grown after inserting the first data sample (1.4, 2.3). After inserting the second data sample (1.2, 2.4) and the third data sample (1.0, 2.7), the tree has grown as shown in Fig. 1d. Since the second data sample (1.2, 2.4) goes to the same leaf node as the first data sample (1.4, 2.3), the count of the leaf node was increased to 2. The third data sample (1.0, 2.7) causes a new child node to be created on the node at level 1.

Since the training data samples consist of only normal data, the constructed RP-tree has the nodes corresponding to the normal data region. However, some nodes could be created due to presence of noise. In order to minimize the effect of noise, the nodes having only a very small number of data samples can be removed in the constructed RP-tree.

3.3 Outlier prediction using region-partition forest

When a test data sample starts from the root and arrives at a leaf node at the level H, it can be declared as being in a normal data region. However, if it is stuck in an interval in which a child node is not created, it can be predicted as an outlier. Suppose that after inserting all the training normal data samples, we have the RP-tree in Fig. 2a. In Fig. 2b, the prediction of two data samples is shown. The data sample (0.9, 2.8) goes to a leaf node which indicates the prediction as a normal data sample. On the other hand, the data sample (3.5, 2.8) is predicted as an outlier since it belongs to an interval in which a child node was not created.

Outlier detection performance of RP-tree can be improved by an ensemble approach. To construct RP-forest which is composed of m RP-trees, we first create m RP-tables. Since splitting attributes and partition values are randomly selected, each RP-tree represents a partition of normal data region in a different shape. For a binary decision in outlier prediction by RP-Forest, we use the agreement-by-all strategy where a data sample is predicted as an outlier when it is considered to be an outlier by all members of the ensemble.

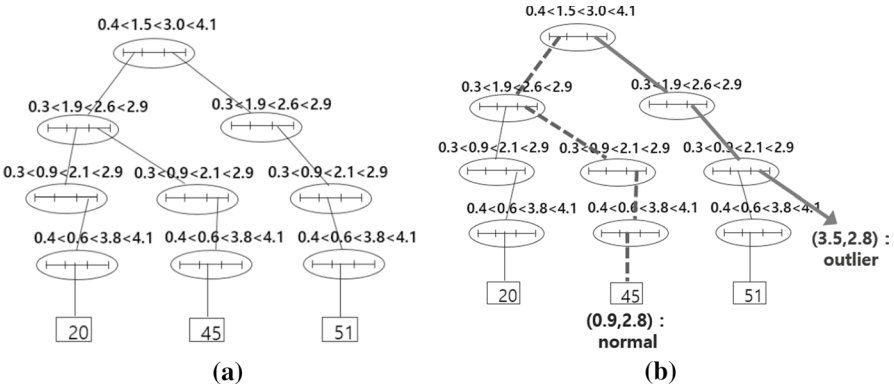


Fig. 2 **a** An example of RP-tree obtained after processing all training normal data samples. **b** Prediction for two data samples (0.9, 2.8) and (3.5, 2.8)

While the proposed method focuses on binary prediction for outliers, it also gives outlier scores which are ranged in [0, 1]. The outlier score of a data sample can be computed by the ratio of trees predicting it as an outlier among all RP-trees in the ensemble. When all the trees predict a data sample as normal, the outlier score of the data sample is zero, and when all the trees predict a data sample as an outlier, the outlier score becomes one.

3.4 Explainable outlier detection

When a test data sample is predicted as an outlier, it would be useful to be able to explain the reason for outlier prediction. In the proposed method, a test data sample stops moving downward when the child node of the interval to which it belongs was not created. Then a hypercube representing the outlier region of the data sample can be constructed by combining intervals on the path from the root to the last node, and the intersection of outlier regions obtained by each RP-tree can explain why the data sample was declared as an outlier. In particular, for each attribute, we can count the number of times that it was the splitting attribute at the level of the last node. Through the frequency ratios for attributes, it can be inferred how much each attribute has played a role in outlier prediction.

For example, we have the data described by three attributes $\{A_1, A_2, A_3\}$ and the ensemble of four RP-trees was built. Suppose that a data sample was declared as an outlier in all four trees and the hypercube to which the data sample belongs was obtained in each RP-tree. The outlier region of the data sample can be found by the intersection of four hypercubes. If the attribute A_1 is the last splitting attribute in three among four trees, the attribute A_1 is responsible at the weight of 0.75 for outlier prediction for the data sample. The algorithm in Table 1 summarizes the proposed outlier detection method by RP-Forest.

Table 1 Algorithm of the proposed outlier detection method

Input H is the tree height, K is the tree degree, m is the number of RP-trees

t is the count threshold for the deletion of leaf nodes

X is the training data composed of normal data samples

for $i = 1 : m$
  % the construction of RP-table
  for $j = 0 : H - 1$
    Select an attribute Aj randomly and $K - 1$ partition values between the minimum value and maximum of values of attribute Aj
  end for
  % the construction of RP-tree
  for each normal data sample x in the training data X
    Construct a root node corresponding to the level 0 in the RP-table
    for $j = 0 : H - 1$
      if the value of attribute Aj of x belongs to the subinterval in which a child node has not been created, then generate a child node of the subinterval
      Descend to the child node of the subinterval where the value of attribute Aj of x belongs
    end for
    At the leaf node of level H, increase the member count of the leaf node by one
  end for
  Delete leaf nodes whose member count is equal to or less than t
end for
% Prediction phase of outliers
for each test data sample
  Set $s = 0$ where s represents the number of RP-trees giving outlier prediction
  Set $w_i = 0$ which is an initial weight for each attribute Ai
  for each RP-tree
    From the root node, descend down to a child node of the corresponding subinterval
    if it stops at a node of level k before reaching a leaf node of level H, then
      Increase s by one
      Increase w_k by one for the splitting attribute Ak of level k
      Construct a hypercube by combining subintervals on the path from the root to the level k
    end if
  end for
  Compute outlier score for the test data sample by $\frac{s}{m}$
  if the outlier score equals to 1
    Predict the test data sample as an outlier
    Construct the outlier region for the test data sample by intersecting the hypercubes from RP-trees
    Compute the responsibility weight for each attribute Ai as $\frac{w_i}{m}$
  end if
end for

4 Experimental results

4.1 Data description

In order to test the performance of the proposed method, we used eight data sets including real and artificial data. Table 2 describes the details of the data. Creditcard data is a summary of credit card usage by some card holders in Europe in September 2013 [20]. The total number of data samples was 284,807, and the number of outliers was 492. We used 28 attributes except the attribute indicating usage amount. KDD-Http data was obtained from kddcup.data-10-percent-corrected data by the DARPA Intrusion Detection Evaluation Program [21]. We extracted 567,498 samples whose value of attribute *service* was http, and three attributes of *duration*, *src-bytes*, and *dst-bytes* were used as in [22]. The number of outliers indicating network intrusion was 2211. Gaussian data is synthetic 1-dimensional data where 100,000 normal samples were generated using 5 Gaussian mixture distributions with mean values 0, 1, 2, 3 and 4, and 2,500 abnormal data samples were generated from Gaussian distribution with mean value 6. RBFevents data was constructed using the artificial streaming data generator RandomRBFevents of MOA [23] with normal data from five normal distributions and outliers from random uniform distribution. Annthyroid, Shuttle, and Covtype data were downloaded from UCI machine learning repository [24], and Mammography data from OpenML site [25]. Experimental settings for normal and outliers followed those in [6] as shown in Table 2.

4.2 Performance comparison

We compared the performance of the proposed method RP-Forest with that of one-class SVM (OSVM) [26], LOF [11], and two tree-based outlier detection method, Isolation Forest and RS-Forest. Except RS-Forest which was coded in JAVA by ourselves, for the other compared methods, the implementation in Scikit-learn [27] was used.

The outlier detection performance for binary prediction was evaluated with precision, recall, and F1 value.

Table 2 Data description

Data	Attributes	Samples	Outliers	Outlier ratio (%)	Outlier(O) versus Normal(N)
Creditcard	28	284,807	492	0.17	
Kdd-Http	3	567,498	2211	0.39	
Gaussian	1	102,500	2500	2.44	
RBFevents	5	100,000	10,201	10.2	
Annthyroid	6	7200	534	7.42	O: class 1, 2, N: class 3
Shuttle	9	49,097	3511	7.15	O: class 2, 3, 5, 6, 7, N: class 1
Covtype	10	286,048	2747	0.9	O: class 4, N: class 2
Mammography	6	11,183	260	2.3	O: class 1, N: others

- True positive (TP): The number of outliers that are predicted as an outlier.
- False positive (FP): The number of normal data samples that are predicted as an outlier.
- False negative (FN): The number of outliers that are predicted as a normal data sample.
- Precision = TP / (TP + FP), Recall = TP / (TP + FN)
- F1 = 2 * Precision * Recall / (Precision + Recall)

Outlier detection models were constructed using a training set of normal data samples corresponding to 30% of the total data samples, and the outlier detection was performed for a test set of 70% data. By randomly splitting the data into training and test sets, the experiments were repeated 10 times and the average F1 value was used for performance comparison.

OSVM builds the hypersphere containing the normal data by SVM, and data samples deviating from it are judged as an outlier. In OSVM, the variable ν for the upper limit of training error rate was set as 0.0001, 0.0005, 0.001, and 0.005, and the highest F1 value among four cases in each data set was chosen. The other three methods, LOF, Isolation Forest, and RS-Forest, compute outlier scores for data samples. For performance comparison with the proposed method which gives binary labels of an outlier or normal, 70% test data samples are ranked according to outlier scores, and top ranked data samples of the number corresponding to various outlier rates are predicted as outliers. We tried outlier rates of 1%, 2%, 5%, 10%, and 15%, and the highest F1 value was chosen. However, in real situation it would be impossible to know the actual outlier ratio and therefore it is difficult to set the threshold for binary labeling.

In LOF, the number of neighbors was set to 20 which is a default value in Scikit-learn [27]. Default parameter values in Isolation Forest [6] are sub-sampling size=256 and ensemble size=100. The tree height limit H is automatically set by the sub-sampling size ψ as $H = ceiling(log_2 \psi)$. Ensemble size in RS-Forest was also set as 30 or 100, and tree height was 15. All training samples were used for the construction of RS-trees.

The parameters to be determined in the proposed method, RP-Forest, are the number of RP-trees m, height H, and degree K of a tree. While tree height was set as 15 as in RS-Forest, we tried the values of the following range in pre-testing stage using KDD-Http and Creditcard data. After constructing RP-tree, the nodes containing only one data sample were removed from the tree.

- The number of RP-trees m: 10, 20, 30, 40
- Degree K: 2, 4, 9, 14, 19

Table 3 shows the average of F1 values of KDD-http and Creditcard at height $H = 15$. Overall, when $K = 9$, the ensemble of 20 trees gave the good performance. Hence, in the performance comparison with other outlier detection methods, the parameters for RP-Forest were set as $H = 15$, $K = 9$, $m = 20$ for all data sets.

Table 3 Comparison of outlier detection performance on Creditcard and KDD-http data when tree degree K is varied to 2, 4, 9, 14, 19 and ensemble size m is changed to 10, 20, 30, 40

K	m	Creditcard			KDD-http		
		Precision	Recall	F1	Precision	Recall	F1
2	10	0.87	0.16	0.27	0.7	0.7	0.7
	20	0.88	0.12	0.21	0.7	0.7	0.7
	30	0.88	0.11	0.19	0.7	0.7	0.7
	40	0.88	0.1	0.17	0.7	0.7	0.7
4	10	0.67	0.21	0.32	0.99	0.99	0.99
	20	0.75	0.18	0.3	0.95	0.8	0.8
	30	0.77	0.18	0.29	0.95	0.8	0.8
	40	0.8	0.18	0.29	0.95	0.79	0.8
9	10	0.36	0.6	0.45	0.98	0.99	0.99
	20	0.44	0.54	0.48	0.98	0.99	0.99
	30	0.47	0.47	0.47	0.98	0.99	0.99
	40	0.49	0.45	0.47	0.99	0.99	0.99
14	10	0.11	0.83	0.19	0.93	0.99	0.96
	20	0.19	0.75	0.3	0.96	0.99	0.98
	30	0.22	0.74	0.34	0.97	0.99	0.98
	40	0.23	0.72	0.35	0.97	0.99	0.98
19	10	0.04	0.84	0.08	0.88	0.99	0.93
	20	0.08	0.82	0.14	0.91	0.99	0.95
	30	0.09	0.81	0.16	0.93	0.99	0.96
	40	0.1	0.81	0.17	0.94	0.99	0.97

Table 4 Comparison of F1 values in outlier detection methods

Data	OSVM	LOF	Isolation forest	RS-forest	RP-forest
Creditcard	0.21	0.11 (2)	0.23 (1)	0.11 (30, 1)	**0.48**
Kdd-Http	0.94	0.2 (5)	0.43 (2)	0.71 (100, 1)	**0.99**
Gaussian	0.77	0.76 (5)	0.78 (5)	0.77 (30, 5)	**0.94**
RBFevents	0.89	0.98 (15)	0.98 (15)	0.40 (100, 15)	**0.99**
Annthyroid	0.08	0.39 (10)	**0.51** (15)	0.47 (30, 10)	0.44
Shuttle	0.29	0.96 (10)	0.97 (10)	0.96 (100, 10)	**0.98**
Covtype	0.03	**0.83** (1)	0.12 (5)	0.11 (100, 5)	0.57
Mammography	0.1	0.31 (2)	0.32 (5)	0.30 (30, 10)	**0.4**
average	0.41	0.57	0.54	0.48	**0.72**

Table 4 compares the average F1 values obtained from 10 repeated random splitting to training and test sets. For LOF and Isolation Forest, the outlier ratio when the highest F1-value was obtained is shown in the parenthesis, and for RS-Forest, ensemble size and outlier ratio are shown. For each dataset, the highest F1-value was denoted as boldface. The paired t test was performed to determine whether the F1-values by the other methods were statistically equal to the highest F1-value at the

Table 5 Comparison of AUC scores in outlier detection methods

Data	OSVM	LOF	Isolation forest	RS-forest	RP-forest
Creditcard	0.942	0.926	0.949	0.641	**0.956**
Kdd-Http	**0.999**	0.949	0.991	**0.999**	**0.999**
Gaussian	0.959	0.957	**0.985**	0.979	0.959
RBFevents	0.882	**0.999**	**0.999**	0.659	**0.999**
Annthyroid	0.648	0.747	**0.905**	0.760	0.864
Shuttle	0.998	0.998	0.995	0.971	**0.999**
Covtype	0.508	**0.997**	0.828	0.799	0.986
Mammography	0.856	0.794	0.878	0.706	**0.887**
average	0.898	0.921	0.941	0.814	**0.956**

Table 6 The total number of nodes in trees constructed in three tree-based outlier detection methods

	Credit.	Kdd-Http	Gaussian	RBFevents	Annthyroid	Shuttle	Covtype	Mammography
Iso-Forest	9152	11,548	19,093	14,873	9663	11,636	13,821	10,279
RP-Forest	292,220	34,440	14,800	154,660	51,560	22,940	1,826,720	50,920
RS-Forest	1,966,050	6,553,500	1,966,050	6,553,500	1,966,050	6,553,500	6,553,500	1,966,050

significance level 0.01, but no such cases were found. In all data sets except Annthyroid and Covtype, the proposed method obtained the highest F1 values.

In addition to performance comparison by F1-measure in binary prediction, comparison by AUC (Area under the ROC Curve) score is also shown in Table 5. The same experimental setting as in the previous experiments was used. Isolation Forest and RP-Forest obtained the highest AUC scores in most data sets.

4.3 Comparison of a tree size

Table 6 compares the tree size in three tree-based outlier detection methods, Isolation Forest, RS-Forest, and RP-Forest. Total numbers of nodes in the tree ensemble were counted in the experimental setting where F1-values in Table 4 were obtained. Since RS-Forest builds a complete binary tree, the number of nodes in a tree of height H is computed by $2^{H+1} - 1$ regardless of the size of training data. Isolation Forest grows a tree until all data samples are isolated or the specified tree height is reached. When the sub-sampling size 256 is set in Isolation Forest, the constructed tree becomes very small as shown in Table 6. In the proposed method, the number of nodes was larger than that of Isolation Forest, but much smaller than that of RS-Forest in all data sets. Since RP-table stores the splitting attributes and partition information used for node construction in each level of a tree, only nodes

corresponding to a normal region are generated, and therefore a tree of small size can describe normal data region finely.

4.4 An example of explainable outlier detection

Figure 3 shows the hypercubes representing outlier regions which were obtained in the proposed method using KDD-Http data. For each test data sample predicted as an outlier, we can find the hypercube representing the outlier region to which it belongs and compute the responsibility weight of each attribute for outlier prediction. In Fig. 3, the dark circles represent normal training data samples, and some hypercubes among total 51 outlier regions are shown along with responsibility weights for attributes. For many predicted outliers, the attribute "src-byte" which represents source bytes sent from the source to the destination has been described as a main cause.

4.5 Sensitivity on parameters of the proposed method

Through the pre-test using Creditcard and KDD-http data in Table 3, the parameters for RP-Forest were set as tree height $H = 15$, tree degree $K = 9$, the number of RP-trees $m = 20$ for all data sets. By varying parameter values, we tested the sensitivity on parameters of the proposed method. Figure 4 compares the average F1-value in all the data sets for the parameter values of the following range:

- Height H: 10, 15, 20
- Degree K: 4, 9, 14
- The number of RP-trees m: 10, 20, 30

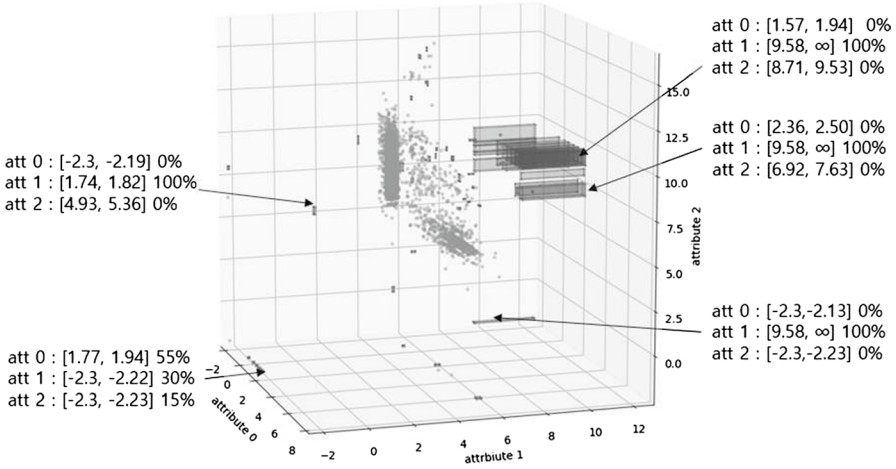


Fig. 3 The illustration of the hypercubes representing outlier regions of data samples predicted as outlier in KDD-http data. (att0 for duration, att1 for src-bytes, and att2 for dst-bytes)

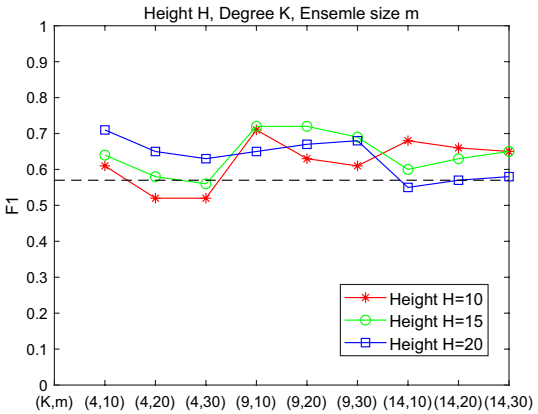


Fig. 4 The performance comparison when using various parameter values

The horizontal line was drawn for F1 value 0.57, which was the second highest F1 value as shown in Table 4. Among 27 combinations of parameter values, for only four cases, F1 value was less than 0.57. The minimum F1 value was 0.52 when $H = 10$, $K = 4$, and $m = 30$, and the maximum F1 value was 0.72 when $H = 15$, $K = 9$, and $m = 10$ or 20. The average F1 value in 27 cases was 0.63 and standard deviation was 0.058.

5 Discussions

We proposed an outlier detection method based on region-partition tree. The proposed method builds a tree which describes a partition of the normal data region. The region-partition table stores the splitting attribute and partition information which is used in each level of RP-tree. Utilizing RP-table can make a partition of the normal region finely with a tree of a small size. In particular, the proposed method gives both binary labeling and normalized outlier scores. Considering that the ratio of generated outliers is usually very small and it is difficult to determine whether it is an outlier or not based on a threshold on outlier scores, giving binary labeling for outlier prediction is an attractive feature of the proposed method. Moreover, the proposed method identifies the attributes responsible for outlier prediction. In application areas where an outlier detection model is installed, upon the occurrence of abnormality in the system, information about which attributes or components need to be inspected can be very informative to the system manager.

Experimental results demonstrated the outlier detection performance of the proposed method. The proposed method obtained an average F1-value of 0.72, while the second highest F1 value in the compared methods was 0.57. On the other hand, there was no significant difference in the comparison by AUC score where the proposed method showed AUC of 0.96 and Isolation Forest obtained AUC of 0.94. This shows that the proposed method is effective for binary classification of outliers.

Acknowledgements This research was supported in part by Korea Electric Power Corporation (Grant Number: R18XA05) and in part by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. NRF-2019R1F1A1062341).

References

1. Hawkins D (1980) Identification of outliers. Springer, Netherlands
2. Aggarwal C (2017) Outlier analysis. Springer, Netherlands
3. Chauhan S, Vig L (2015) Anomaly detection in ECG time series via deep long short-term memory networks. In: Proceedings of DSAA
4. March E, Vesperini F, Eyben F, Squartini S, Schuller B (2015) A novel approach for automatic acoustic novelty detection using a denoising autoencoder with bidirectional ISTM neural networks. In Proceedings of ICASSP
5. Yang J, Rahardja S, Franti P (2019) Outlier detection: how to threshold outlier scores?. In: Proceedings of the international conference on artificial intelligence, information processing and cloud computing
6. Liu F, Ting K, Zhou Z (2008) Isolation forest. In: Proceedings of the 8th international conference on data mining
7. Wu K, Zhang K, Fan W, Edwards A, Yu P (2014) RS-forest: a rapid density estimator for streaming anomaly detection. In: Proceedings of the 14th international conference on data mining
8. Remy P (2016) Anomaly detection in time setries using auto encoders. Bolg positng from http://philipperemy.github.io/anomaly-detection. Accessed 2 Oct 2016
9. Park C (2019) Outlier and anomaly pattern detection on data streams. J Supercomput 75:6118–6128
10. Knorr E, Ng R(1999) Finding intensional knowledge of distance-based outliers. In: Proceedings of 25th international conference on very large databases
11. Breunig M, Kriegel H, Ng J, Sander R (2000) LOF: Identifying density-based local outliers. In: Proceedings of the 2000 ACM sigmod international conference on management of data
12. Jiang M, Tseng S, Su C (2001) Two-phase clustering process for outliers detection. Pattern recognition letters 22:691–700
13. He Z, Xu X, Deng S (2003) Discovering cluster-based local outliers. Pattern Recognit Lett 24:1641–1650
14. Zhai S, Cheng Y, Lu W, Zhang Z (2016) Deep structured energy based models for anomaly detection. In: Proceedings of the ICML
15. Wang H, Li X, Zhang T (2018) Generative adversarial network based novelty detection using minimized reconstruction error. Frontiers Inf Technol Electron Eng 19:116–125
16. Zenati H, Romain M, Foo C, Lecouat B, Chandrasekhar V (2018) Adversarially learned anomaly detection. In: Proceedings of the ICDM
17. Alla S, Adari S (2019) Practical use cases of anomaly detection beginning anomaly detection using python-based deep learning. Apress, Berkeley
18. Susto G, Beghi A, McLoone S (2017) Anomaly detection through on-line isolation forest: an application to plasma etching. In: Proceedings of the 28th annual semi advanced semiconductor manufacturing conference (ASMC)
19. Ounacer S, Bour H, Oubrahim Y, Ghoumari M, Azzouazi M (2018) Using Isolation Forest in anomaly detection: the case of credit card transactions. Period Eng Nat Sci 6(2):394–400
20. https://www.kaggle.com/dalpozz/creditcardfraud
21. http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html
22. Hawkins S, Hongxing H, Williams G, Baxter R (2002) Outlier detection using replicator neural networks. In: Proceedings of DaWaK
23. Bife A, Holmes G, Kirkby R, Pfahringer B (2010) Moa: massive online analysis. J Mach Learn Res 11:1601–1604
24. https://archive.ics.uci.edu/ml/index.php
25. https://www.openml.org
26. Scholkopf B, Platt J, Shawe-Taylor J, Smola A, Williamson R (2001) Estimating the support of a high-dimensional distribution. Neural Comput 13(7):1443–1471
27. Pedregosa F et al (2011) Scikit-learn: machine learning in python. J Mach Learn Res 12:2825–2830