International Journal of Machine Learning and Cybernetics (2021) 12:1137–1150
https://doi.org/10.1007/s13042-020-01225-0

ORIGINAL ARTICLE

usfAD: a robust anomaly detector based on unsupervised stochastic forest

Sunil Aryal[1] · **K.C. Santosh**[2] · **Richard Dazeley**[1]

Received: 11 April 2020 / Accepted: 16 October 2020 / Published online: 2 November 2020

Abstract
In real-world applications, data can be represented using different units/scales. For example, weight in kilograms or pounds and fuel-efficiency in km/l or l/100 km. One unit can be a linear or non-linear scaling of another. The variation in metrics due to the non-linear scaling makes Anomaly Detection (AD) challenging. Most existing AD algorithms rely on distance- or density-based functions, which makes them sensitive to how data is expressed. This means that they are representation dependent. To avoid such a problem, we introduce a new anomaly detection method, which we call 'usfAD: Unsupervised Stochastic Forest-based Anomaly Detector'. Our empirical evaluation in synthetic and real-world cybersecurity (spam detection, malicious URL detection and intrusion detection) datasets shows that our approach is more robust to the variation in units/scales used to express data. It produces more consistent and better results than five state-of-the-art AD methods namely: local outlier factor; one-class support vector machine; isolation forest; nearest neighbor in a random subsample of data; and, simple histogram-based probabilistic method.

Keywords Measurement scales and units · Anomaly detection · Outlier detection · Robust anomaly detection · Intrusion detection · Spam detection · And cyber security

1 Introduction

1.1 Background

Anomalies (also sometimes referred to as outliers) are data instances that are significantly different from most of the other data causing suspicions that they were generating from a different mechanism from the one that is normal or expected [23]. Anomaly Detection (AD) is the task of detecting anomalies in a given dataset automatically using computers and algorithms [16]. It has many applications such as [1]:

- *Intrusion detection* Detecting unauthorised access requests and malicious activities in computer networks.
- *Fraud detection* Detecting fraudulent and suspicious credit card and other financial transactions in banking.
- *Spam detection* Detecting malicious and phishing emails in electronic communications.

Most existing anomaly detection algorithms [3, 4, 15, 26] assume that anomalies have feature values that are significantly different from those of normal instances. In other words, anomalies are few and different and they lie in low density regions.

2 Motivation

In real-world applications, features of data objects can be measured in different units or recorded in different scales [5, 6, 19, 36]. For example: (1) patient's weight can be measured in kilograms or pounds and temperature in °C or °F; (2) price of vehicles can be recorded in integer scale as $x = 100,000$ or logarithmic scale of base 10 like $x' = 5$, fuel efficiency in km/l as $x = 9.0$ or l/100 km as $x' = 11.11$; and

✉ K.C. Santosh
santosh.kc@ieee.org

Sunil Aryal
sunil.aryal@deakin.edu.au

Richard Dazeley
richard.dazeley@deakin.edu.au

[1] School of Information Technology, Deakin University, 75 Pigdons Rd, Waurn Ponds, VIC 3216, Australia

[2] Department of Computer Science, University of South Dakota, 414 E Clark St, Vermillion, SD 57069, USA

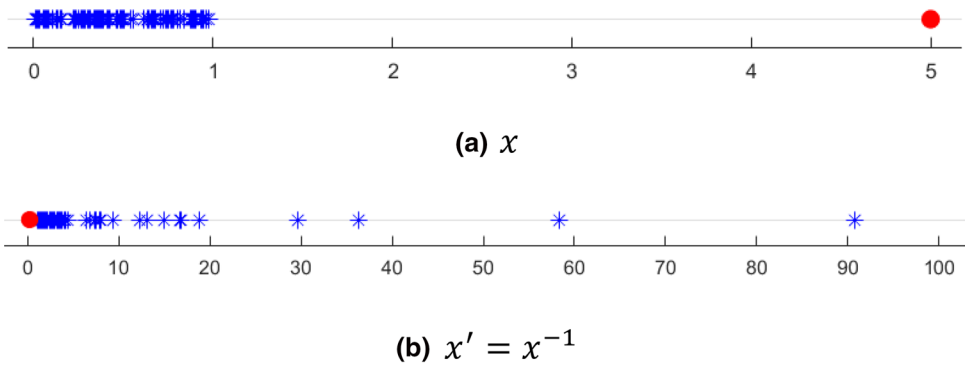


Fig. 1 An example of the same data represented in two scales. The red data point in Case (**a**) looks like an anomaly, whereas the corresponding point in Case (**b**) is more like a normal data

(3) sample variability can be measured in terms of standard deviation (*std*) or variance (*var*) ($std = \sqrt{var}$ or $var = std^2$). Note that the different representations in Case (1) are linear scaling of one another whereas those in Cases (2) and (3) are non-linear scaling of each other.

Data representation can vary because of various reasons: (1) settings of devices used for measurement; (2) domain and/or user requirements; and (3) data compression to store and transmit using less bits. Such variation can be common in many application domains, such as cyber-physical systems, networks and communications, internet-of-things and healthcare, where different features of data objects are measured/recorded by different sensors.

The variation in metrics due to non-linear scaling makes automatic detection of anomalies a challenging task. Existing anomaly detection algorithms [4, 11, 15] rank instances in a database based on their densities or distances to nearest neighbors (NNs). The instances with low densities or large distances to their NNs are flagged as anomalies. Because they use feature values to compute density or NN distances, they are sensitive to how features are measured. For example, the red data point in Fig. 1a clearly appears to be an anomaly, but when the same data is represented in the inverse scale as $x' = x^{-1}$, shown in Fig. 1b, the corresponding point looks like a normal data. Anomaly can be masked by the variation in data representation. Most existing anomaly detection algorithms cannot detect the anomaly in Fig. 1b.

When data is provided for anomaly detection only the magnitude (numbers) are available and the information about unit/scale used to measure/record them may not be available. Even if unit/scale used is known, we may not know whether the given form is appropriate for anomaly detection. Most existing algorithms do not consider such information and they can give misleading results if data is not given in the appropriate form. In critical applications, such as in cybersecurity and banking, the consequences of true negatives and/or false positives can be severe.

Data needs to be transformed into an appropriate form before using existing algorithms. The simplest way to identify the appropriate form is to try different representations (scaling/transformations) and test which one produces the best task specific result. Because there are infinite number of possible transformations to try for each feature, it is not feasible to find the best combination from trial and error even in low-dimensional problems with tens of features, let alone high-dimensional ones with hundreds and thousands of features.

As discussed in Baniya et al. [10], many psychologists argued that raw numbers can be misleading as they could have been measured/presented in different ways. One should not conclude anything from a given set of numbers without understanding where they come from and the underlying process of generating them [25, 28, 40, 41]. Velleman and Wilkinson [41] suggested that good data analysis does not make any assumption about data types/scales because data may not be what we see. Joiner [25] provided some examples of 'lurking variables'—data appear to have one pattern, but in fact hide other information. However, in data mining, numeric data is assumed to be in 'interval scale' [19]—the meaning of a unit difference is same everywhere in the space. This assumption may not be true when data is represented as non-linear scaling such as logarithm and inverse.

Recently, the impact of units/scales has been studied in the context of pairwise similarity measurement of data and robust (dis)similarity measures have been introduced [5, 6, 19]. These robust (dis)similarity measures have been shown to perform better than distance-based approaches (*e.g.*, Euclidean and Cosine distances) in content-based information retrieval and k-NN classification tasks. However, these robust (dis)similarity measures are not applicable in the anomaly detection task because of their implicit assumptions that are counter-intuitive for anomaly detection.

2.1 Contributions

In this research, we introduce an anomaly detection technique that is robust to data representation. Our work is motivated by ideas from 'Unsupervised Stochastic Forest' (USF) [19] and Isolation Forest (iforest) [26]. USF

Table 1 Notations

$\boldsymbol{x} = \langle x_1, x_2, \cdots, x_d \rangle$	A vector representing a data instance in an d-dimensional continuous space ($\mathbb{R}^d$)
$x_i \in \mathbb{R}$	The value of $\boldsymbol{x}$ in the i^{th} dimension (feature)
$D(\|D\| = n)$	A training set of n normal instances
$Q(\|Q\| = q)$	A test set of q instances that is a mixture of normal and anomalous data
$\mathcal{D} \subset D(\|\mathcal{D}\| = \psi \ll n)$	A small subset of training data D
$kNN(\cdot, D)$	The set of k nearest neighbours of a test instance in D using Euclidean distance
$T_j(j = 1, 2, \ldots, t)$	A tree in an ensemble of t trees
h	The parameter that defines the height of a tree
$\ell_j(\cdot)$	The pathlength of a test instance in tree T_j
$L_j(\cdot)$	The leaf node in tree T_j where a test instance reached
$m(\cdot)$	The data mass in a region (e.g., a node of a tree)

is a variant of Unsupervised Random Forest (URF) [14, 34] used by Fernando and Webb [19] to develop a robust similarity measure. Iforest is a fast anomaly detection method based on a variant of random forest. It does not use distance/density and hence runs significantly faster than distance/density-based methods. However, its anomaly detection result is sensitive to the units/scales of data. This paper makes the following two main contributions:

1. Propose a new robust anomaly detection method by combining the ideas of USF and iforest. We extend USF to make it applicable in the anomaly detection task, by incorporating the iforest approach to rank data instances based on their anomaly scores. We call the proposed method as 'usfAD' (Unsupervised Stochastic Forest based Anomaly Detector).
2. Compare the anomaly detection performance of usfAD against five state-of-the-art methods using synthetic and real-world cybersecurity (spam detection, phishing URL detection and intrusion detection) datasets. To evaluate the robustness of methods to different representations of the same data (x), we use the non-linear scaling functions: square (x^2); square root ($\sqrt{x}$); logarithm ($\log x$); and, inverse (x^{-1}).

2.2 Paper organization and notations

The rest of the paper is structured as follows. Prior works related to this paper are discussed in Sect. 2 and the proposed usfAD is discussed in Sect. 3. Experimental setup and results are presented in Sect. 4 followed by related discussion in Sect. 5. The last section concludes the paper with potential directions for future research.

The list of key notations and symbols used in this paper are provided in Table 1.

3 Related works

The anomaly detection problem can be solved using three approaches: supervised, unsupervised or semi-supervised learning [16]. In the supervised approach, a classification model is learned from labelled training instances from both normal and anomalous classes, which is then used to predict class labels for test data (a mixture of normal data and anomalies). Labelled training samples from anomalous class may not be available in many real-world applications [16]. In the unsupervised approach, instances in a database are ranked directly based on some outlier score. In the training process, a scoring model is learned (without using labels), which is used to compute anomaly score of test data to rank them in the testing phase. Anomalies are assumed to be few and different. This approach may not work well when the assumption does not hold, i.e., when there are far too many anomalies [13, 16]. In the semi-supervised approach, a profile of normal data is learned using training data from the normal class only, but the label information is not used in the learning process. In this regard, it is different from the semi-supervised learning approach discussed in other machine learning problems, where a model is learned in the training phase using partially labelled training set (only a subset of training data is labeled). In the testing phase, instances are ranked based on how well they comply with the learned profile of normal data. It makes no assumptions about anomalies nor does it require any training samples from the anomalous class [13, 16]. Thus, the semi-supervised approach is more realistic in real-world applications.

In this paper, we focus on the semi-supervised approach where a model is learned from a training set D of n instances belonging to the normal class only and evaluated on a test set Q of q instances from a mixture of normal and

anomalous data. In this section, we review some widely used existing anomaly detection methods that are applicable for the semi-supervised approach.

3.1 Nearest neighbor based methods

In Nearest Neighbor (NN)-based methods, the anomaly score of a test instance $\boldsymbol{x} \in Q$ is estimated based on the distances to its k NNs in D. Local Outlier Factor (LOF) [15] and kth NN distance [11] are the most widely used semi-supervised and unsupervised methods. Being different from normal instances, anomalies are expected to have larger distances to their kNNs than normal instances. To compute the anomaly score of $\boldsymbol{x} \in Q$, NN-based methods require to compute its distances with all instances in D. It is computationally expensive if the size of training set D is large. Sugiyama and Borgwardt [37] showed that the nearest neighbor search in a small subset $\mathcal{D} \subset D, |\mathcal{D}| = \psi \ll n$ is enough for anomaly detection. They proposed a simple, but very fast, anomaly detector called Sp. The anomaly score of $\boldsymbol{x} \in Q$ is its distances to the nearest neighbor (1NN) in $\mathcal{D}$. It has been shown that Sp with ψ as small as 25 produces competitive results to LOF but runs several orders of magnitude faster [37]. Ting et al. [39] used an ensemble approach using multiple subsamples $\mathcal{D}_j \subset D, \left|\mathcal{D}_j\right| = \psi \ll n (j = 1, 2, \ldots, t)$ and computed the anomaly score of $\boldsymbol{x} \in Q$ as the average distance to the nearest neighbor (1NN) in t subsamples [39].

3.2 Support vector based methods

These methods define the boundary around normal (expected) data and identify a set of data instances lying in the boundary called Support Vectors (SVs) using the kernel trick. They compute the pairwise similarities of instances in the training set D using a kernel function. Gaussian kernel that uses Euclidean distance is a popular choice. In the testing phase, the anomaly score of $\boldsymbol{x} \in Q$ is estimated based on its kernel similarities with the SVs. One-Class Support Vector Machine (SVM) [33] and Support Vector Data Description (SVDD) [38] are widely used methods in this class. The training process is computationally expensive in the case of large D because of the pairwise similarity calculations. In the testing phase, the runtime is linear to the size of SVs. Different approaches have been suggested to speed up the training and testing process [24, 32]

3.3 Isolation based methods

This class of methods are based on the assumption that anomalies are more susceptible to isolation. Isolation Forest (iforest) [26] uses an ensemble of t random trees. Each tree T_j $(j = 1, 2, \ldots, t)$ is constructed from a small subsample of training data ($\mathcal{D}_j \subset D, \left|\mathcal{D}_j\right| = \psi \ll n$). The idea is to isolate every instance in $\mathcal{D}_j$ by random partitioning of data space into two non-empty sets in each node. Anomalies are expected to isolate early and have shorter average path-lengths than normal data. Aryal et al. [7] used the average relative data mass in the leaf nodes and their immediate parents as the anomaly score of a test instance [7]. These methods are very efficient as they do not require any distance/similarity calculations. Isolation using nearest neighbor ensemble (iNNE) [9] uses a different mechanism of isolation in $\mathcal{D}_j$. It creates a hypersphere centered at each instance in $\mathcal{D}_j$ with the radius equal to the distance to its nearest neighbor (NN). Anomalies are expected to fall in hyperspheres with a large radius.

3.4 Histogram based methods

Methods based on one-dimensional histograms [3, 4, 21] are another type of efficient methods. In each dimension, a fixed number of equal-width bins are created and the data mass (frequency) in each bin is recorded. Because anomalies are few and different, they are expected to fall in bins with lower frequencies (*i.e.,* low density bins) in many dimensions. Simple Probabilistic Anomaly Detector (SPAD) [4] is a histogram-based anomaly detector, where bin width in each dimension depends on the variance of data in that dimension.

4 usfAD: a robust anomaly detector based on unsupervised stochastic forest

All methods discussed in Sect. 2 are sensitive to how data is represented because they assume that anomalies have feature values significantly different from those of normal data. They often produce poor results in datasets where this assumption does not hold, such as when data is not given in the appropriate form (Fig. 1b). However, we may not know the appropriate form of data in many real-world applications. In this section, we propose a new anomaly detection method that is robust to the units/scales of data used. Our proposed method is motivated by the ideas of Unsupervised Stochastic Forest (USF) and Isolation Forest (iforest).

USF **is** a variant of random forest used by Fernando and Webb [19] to define a similarity measure that is robust to units/scales of data. The similarity of two instances is defined as the number of shared leaves in the forest of t trees. With a user defined tree height parameter (h), each tree T_j in the ensemble is constructed from a small subsample of data, $\mathcal{D}_j \subset D$ $(j = 1, 2, \ldots, t)$, where $\left|\mathcal{D}_j\right| = 2^h$. The subsample of data is divided into two equal subsets at each internal node of a tree by splitting at the median of the sample values of an attribute

selected at random. The process is repeated until nodes have one instance each. It creates balanced binary tress with all leaf nodes at the same height h. The tree structure is robust to how data is measured because of the median splits.

As discussed in Sect. 2.3, iforest is a variant of random forest used for anomaly detection. Each tree T_j is constructed from a small subsample of data, $\mathcal{D}_j \subset D$ $(j = 1, 2, \ldots, t)$, where $\left|\mathcal{D}_j\right| = \psi \ll n$. Instances in $\mathcal{D}_j$ are isolated using random partition at each internal node. Both attribute and split point are selected at random. iforest is sensitive to how data is measured because of the random split. The probability of having a split between two consecutive data points is proportional to their distance. The random split results in unbalanced binary trees which are the core of iforest for anomaly detection. The average pathlengths of a test data instance in trees is used as its anomaly scores.

Because of the balanced binary trees, pathlength cannot be used as the anomaly score in USF. We propose the following extensions to USF so that pathlength can be used. When a tree T_j is constructed from $\mathcal{D}_j \subset D$, normal and anomalous regions are defined in each node using the entire training data D. Therefore, tree construction is a two-step process.

Algorithm 1: Build a tree from a given subsample of data

Input: $\mathcal{D}$ – subsamples of training data ($|\mathcal{D}| = 2^h$)

Output: a USF tree

`usfTree`($\mathcal{D}$)

1. `IF` $|\mathcal{D}| = 1$ `THEN` /* check if leaf node is reached */
2. `RETURN` /* return if leaf */
3. $self.a \leftarrow$ `select`$(1, 2, \cdots, d)$ /* randomly select an attribute */
4. $S \leftarrow$ `sort`$(\mathcal{D}_{self.a})$ /* sort sample values for the selected attribute */
5. $self.s \leftarrow \left(S\left[\frac{|\mathcal{D}|}{2}\right] + S\left[1 + \frac{|\mathcal{D}|}{2}\right]\right)/2$ /* median split */
6. $\mathcal{D}_L \leftarrow$ `filter`$\left(\mathcal{D}_{self.a} \leq self.s\right)$; $\mathcal{D}_R \leftarrow$ `filter`$\left(\mathcal{D}_{self.a} > self.s\right)$ /* filter samples */
7. $self.lNode \leftarrow$ `usfTree`$(\mathcal{D}_L)$; $self.rNode \leftarrow$ `usfTree`$(\mathcal{D}_R)$ /* build two subtrees */

(a) **Building T_j from $\mathcal{D}_j$** This is the same as in USF where instances in the subsample $\mathcal{D}_j$ are isolated using median splits in each node. The detailed procedure is provided in Algorithm 1.

Algorithm 2: Update tree to define normal regions in each node

Input: D – the entire normal training data

`updateTree`(D)

1. `IF` $self.lNode =$ `NULL THEN` /* check if it is a leaf */
2. $self.m \leftarrow |D|$ /* record training data mass */
3. $self.range \leftarrow$ `rangeAll`(D) /* range of training data in all dimensions */
4. `RETURN` /* return */
5. $self.range \leftarrow$ `range`$(D_{self.a})$ /* range of values in dimension $self.a$ */
6. $D_L \leftarrow$ `filter`$\left(D_{self.a} \leq self.s\right)$; $D_R \leftarrow$ `filter`$\left(D_{self.a} > self.s\right)$ /* filter data */
7. $self.lNode$.`updataTree`(D_L); $self.rNode$.`updataTree`(D_R) /* do it on subtrees */

Algorithm 3: Compute anomaly score of $\boldsymbol{x}$ in a tree

Input: $\boldsymbol{x}$ – A test data instance; ℓ – pathlength so far ($\ell = 0$ for root)

Output: Anomaly score of $\boldsymbol{x}$

```
score(x, p)
1.  IF self.lNode = NULL THEN                                   /* check if it is a leaf */
2.    IF inNormalRegion(x) THEN                                 /* check if x in the normal region */
3.      RETURN ℓ + log2(self.m)                                 /* return augmented pathlength */
4.    RETURN ℓ                                                  /* return pathlength (in anomaly region) */
5.  IF inNormalRange(x_self.a) THEN                             /* check if it is in normal range */
6.    p ← ℓ + 1                                                 /* increase pathlength */
7.    IF x_self.a ≤ self.s THEN                                 /* check if x falls to left subtree */
8.      RETURN self.lNode.score(x, ℓ)                           /* get score from the left subtree */
9.    RETURN self.rNode.score(x, ℓ)                             /* else get score from the right subtree */
10. RETURN ℓ                                                    /* out of normal range in internal node, return pathlength */
```

b. **Defining normal and anomalous regions in nodes of T_j using D** Once T_j is built, the entire training data D is passed to T_j. The region defined by the bounding hyper-rectangle that covers the training data within a leaf node is considered as the "normal region" and the rest is considered as the "anomalous region". An example of space partitioning and the definition of "normal regions" in leaf nodes is shown in Fig. 2. The training data mass in each leaf node is stored. In internal nodes, only the attribute selected for partitioning the space is used to define the normal data range. The algorithm to update T_j to define normal and anomalous regions, and store data mass in leaf nodes is provided in Algorithm 2.

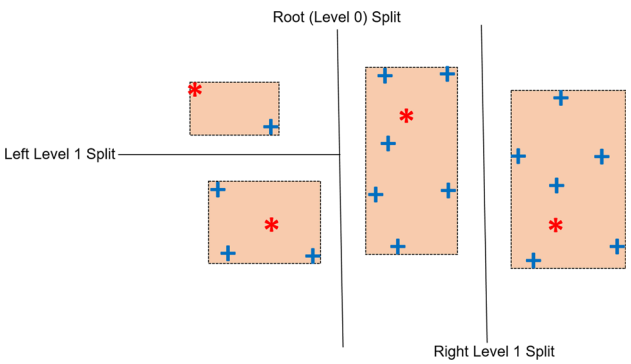


Fig. 2 An example of partitioning and definition of normal regions in leaf nodes with the training data D of 20 normal instances and h = 2. Data point represented by red asterisks are subsamples $\mathcal{D} \subset$ D selected to build the tree ($|\mathcal{D}| = 2^2 = 4$). Orange rectangles represent normal regions in leaf nodes

In the testing phase, the anomaly score of a test instance $\boldsymbol{x}$ in each tree T_j is computed as the pathlength of the first node where it falls outside of the normal region. As shown in Algorithm 3, $\boldsymbol{x}$ is traversed down the T_j in each internal node only if its value of the attribute selected to split the node is within the normal data range. Otherwise the traversal is terminated, and the current height is returned as the score of $\boldsymbol{x}$ in T_j,$\ell_j(\boldsymbol{x})$. If $\boldsymbol{x}$ reaches a leaf node, $\ell_j(\boldsymbol{x})$ is estimated based on whether it lies in the normal or anomalous region: (1) if it lies in the anomalous region, $\ell_j(\boldsymbol{x}) = h$; and, (2) if it lies in the normal region, the score is the height augmented by the training data mass in the leaf ($L_j(\boldsymbol{x})$), $\ell_j(\boldsymbol{x}) = h + \log_2\big(m\big(L_j(\boldsymbol{x})\big)\big)$, where $m\big(L_j(\boldsymbol{x})\big)$ is the data mass in the leaf where $\boldsymbol{x}$ falls in tree T_j. The second term is to ensure that leaf nodes with higher densities have a larger score than those with low densities. Similar adjustment was done in iforest [26]. The final anomaly score of $\boldsymbol{x}$ is the average score over the ensemble of t trees like in iforest [26]:

$$s(\boldsymbol{x}) = \frac{1}{t}\sum_{j=1}^{t} \ell_j(\boldsymbol{x}) \tag{1}$$

We call the proposed method 'usfAD' as it is based on the Unsupervised Stochastic Forest (USF). It uses the same idea of isolating anomaly regions from normal regions as in iforest [26] but using different mechanism of isolation. Anomalies will have smaller scores than normal instances. Because of the median splits, usfAD is robust to the change in units/scales of data measurement. It is based on the orderings or ranks of data which is either preserved or reversed if

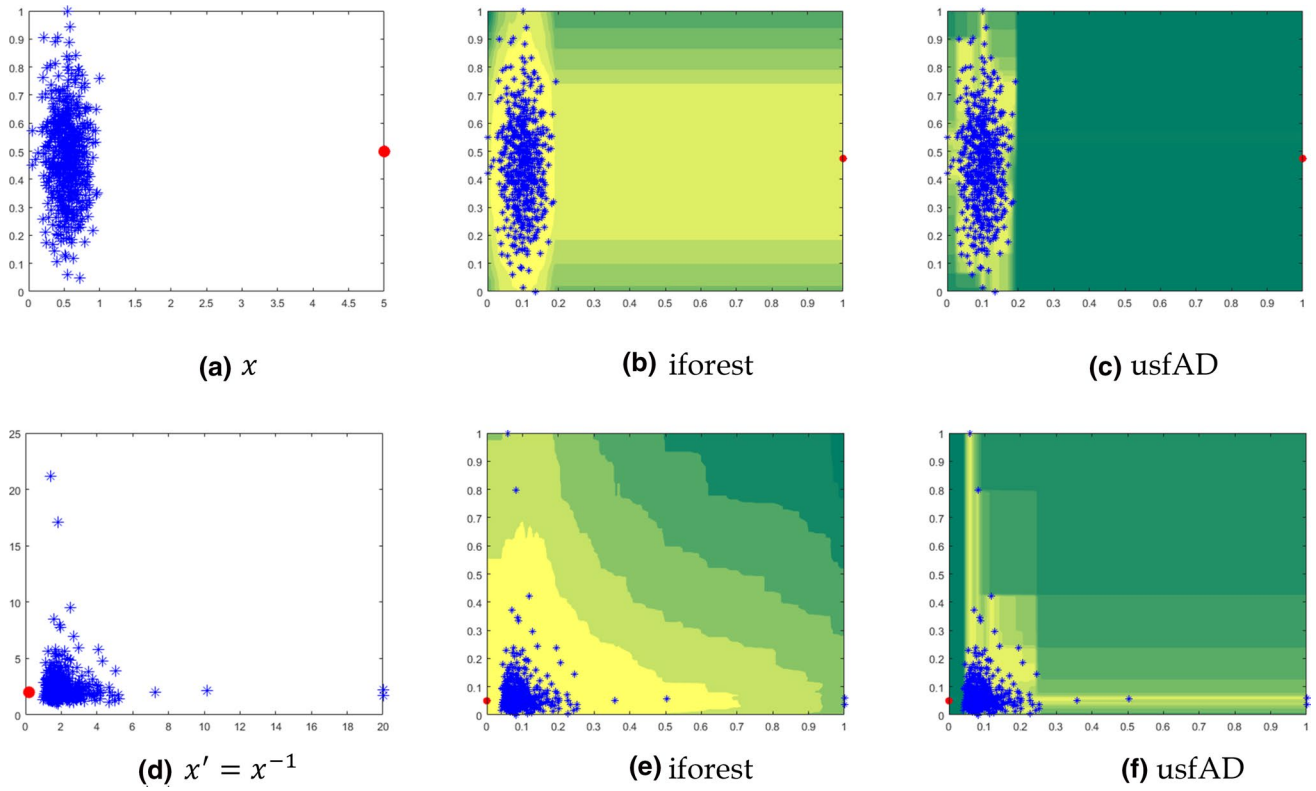


Fig. 3 An example dataset in two scales: x and $x' = x^{-1}$ and anomaly contours of iforest ($t = 100, \psi = 256$) and usfAD ($t = 100, h = 5$). Note that data is normalized to the unit range in both dimensions. The darker the color, the lower the anomaly score, i.e., higher the chances of being anomaly. Note that training data D or D' does not include the red dot, only normal data points represented by blue asterisks are included

the data is measured in different scales. If a data point u lies between $[x, y]$ in one scale, the corresponding point u' lies in between $[x', y']$ in another scale. As the median of even data samples is the mid-point of the two data in the middle, the definition of normal regions in leaf nodes can vary slightly if data is measured differently. It may cause small differences in the anomaly detection results.

Figure 3 shows the contour plots of anomaly scores of points in a two-dimensional space using iforest ($t = 100, \psi = 256$) and usfAD ($t = 100, h = 5$) in a dataset in two scales: x and $x' = x^{-1}$. It is clear that the anomaly represented by the red dot is not identified as a strong anomaly by iforest in both representations. It appears to be an anomaly in the original scale (Fig. 3b) but the actual score is not significantly different from those of some blue points. It clearly appears to be a normal point in the inverse scale (Fig. 3e). The anomaly can be detected as a strong anomaly by usfAD in both scales (Fig. 3c and f).

In terms of runtime, usfAD runs slightly slower than iforest. In the training phase, it requires to define normal regions in nodes passing the entire training data in each tree. During testing, for each test data, it requires an overhead to check if it falls in the normal or anomaly region in each node. Its training runtime complexity is $O(n\text{th} + t\psi d)$, where $\psi = 2^h$. The testing runtime complexity to rank a test instance is $(t(h + d))$. It needs $O(t\psi d)$ space to store the ensemble of trees. Like iforest, its testing time is independent of training data size n.

5 Empirical evaluation

This section presents the setup and results of experiments conducted to evaluate the performance of usfAD against existing anomaly detection methods. Five widely used state-of-the-art methods of LOF, one-class SVM, iforest, Sp and SPAD were used as contenders for performance evaluation. Six two-dimensional synthetic datasets and four benchmark cybersecurity datasets were used. All experiments were conducted in the semi-supervised setting. In each dataset, half of the normal instances were used as training data D and the remaining other half of normal data and all anomalies are considered as test data Q as done in [13]. Anomaly detection model was learned from D and tested on Q.

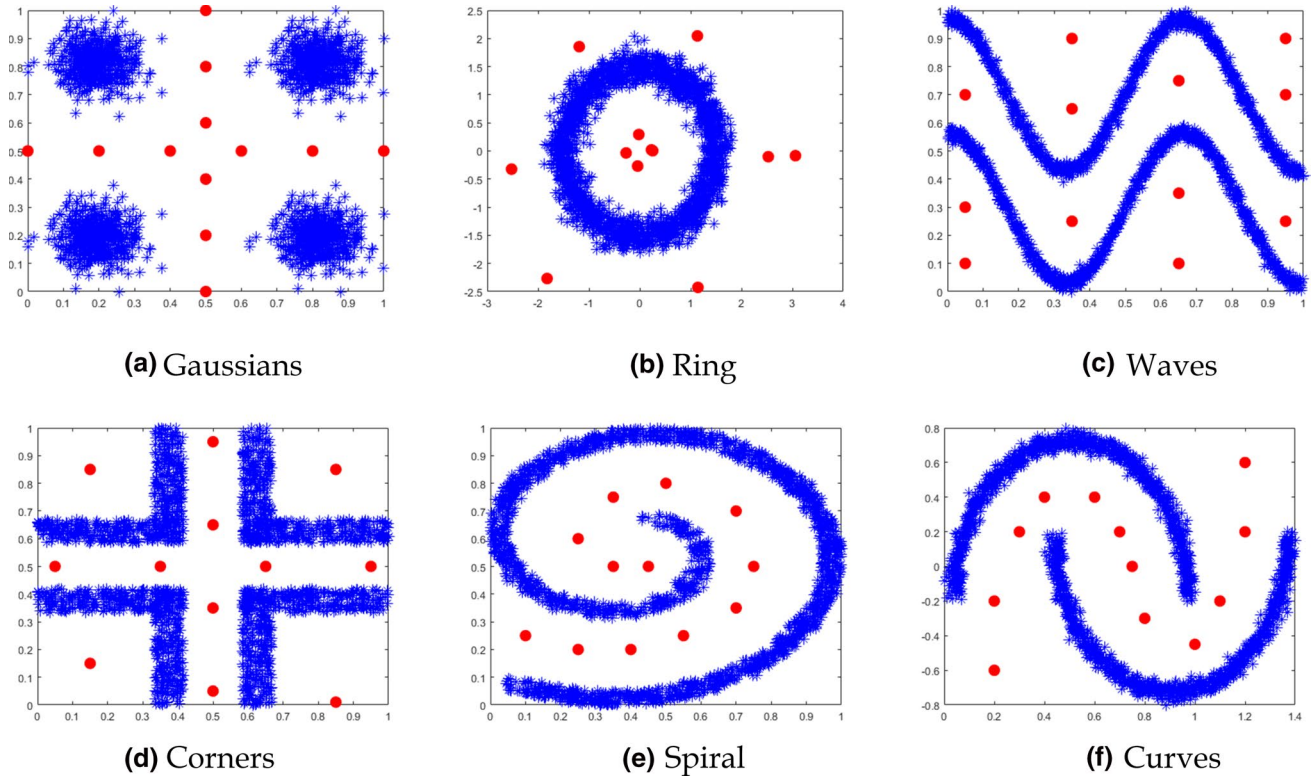


Fig. 4 Six two-dimensional synthetic datasets. Each has 2000 normal instances represented by blue asterisks and 12 anomalies represented by red dots

The performance was evaluated in terms of the Area Under the ROC Curve (AUC). AUC is estimated using the rankings of test instances in Q based on their anomaly scores (anomalies are expected to have higher ranks than normal instances) and ground truth labels [22, 39]. It is equivalent to the probability that a randomly chosen anomaly will be ranked below a randomly chosen normal instance [22]. For the random methods of iforest, Sp and usfAD, each experiment was repeated 10 times and reported the average AUC. The same training and test sets of a dataset were used for all experiments with the dataset.

In terms of implementation, the python implementations of LOF and SVM included in the Scikit-learn Machine Learning Library [31] were used. Other methods and experimental setups were also implemented in Python. All the experiments were conducted in a Linux machine with 2.27 GHz processor and 8 GB memory. Parameters in algorithms were set to suggested default values by respective authors as:

- LOF: Nearest Neighbour parameter $k = \lfloor \sqrt{n} \rfloor$;
- Sp: Subsample size $\psi = 25$;
- SPAD: Number of bins $b = \lfloor \log_2 n \rfloor + 1$;
- iforest: Ensemble size $t = 100$, and Subsample size $\psi = 256$;
- SVM: Default settings of all parameters ($kernel =$ 'rbf', $\gamma = 1/d$, $\nu = 0.5$); and
- usfAD: Tree height $h = 5$, and Ensemble size $t = 100$.

To evaluate the robustness of algorithms with different representations of the same data, non-linear scaling using square (x^2), square root ($\sqrt{x}$), logarithm ($\log x$) and inverse (x^{-1}) were used. To cater for $\log x$ and x^{-1} at $x = 0$, all transformations were applied on $\hat{x} = c(x + \delta)$, with $\delta = 0.0001$ and $c = 100$. Note that data was normalized to the unit range in each feature before rescaling to ensure the same effect of δ and c in all features. Once data was rescaled, it was renormalized to be in the unit range again. This study used

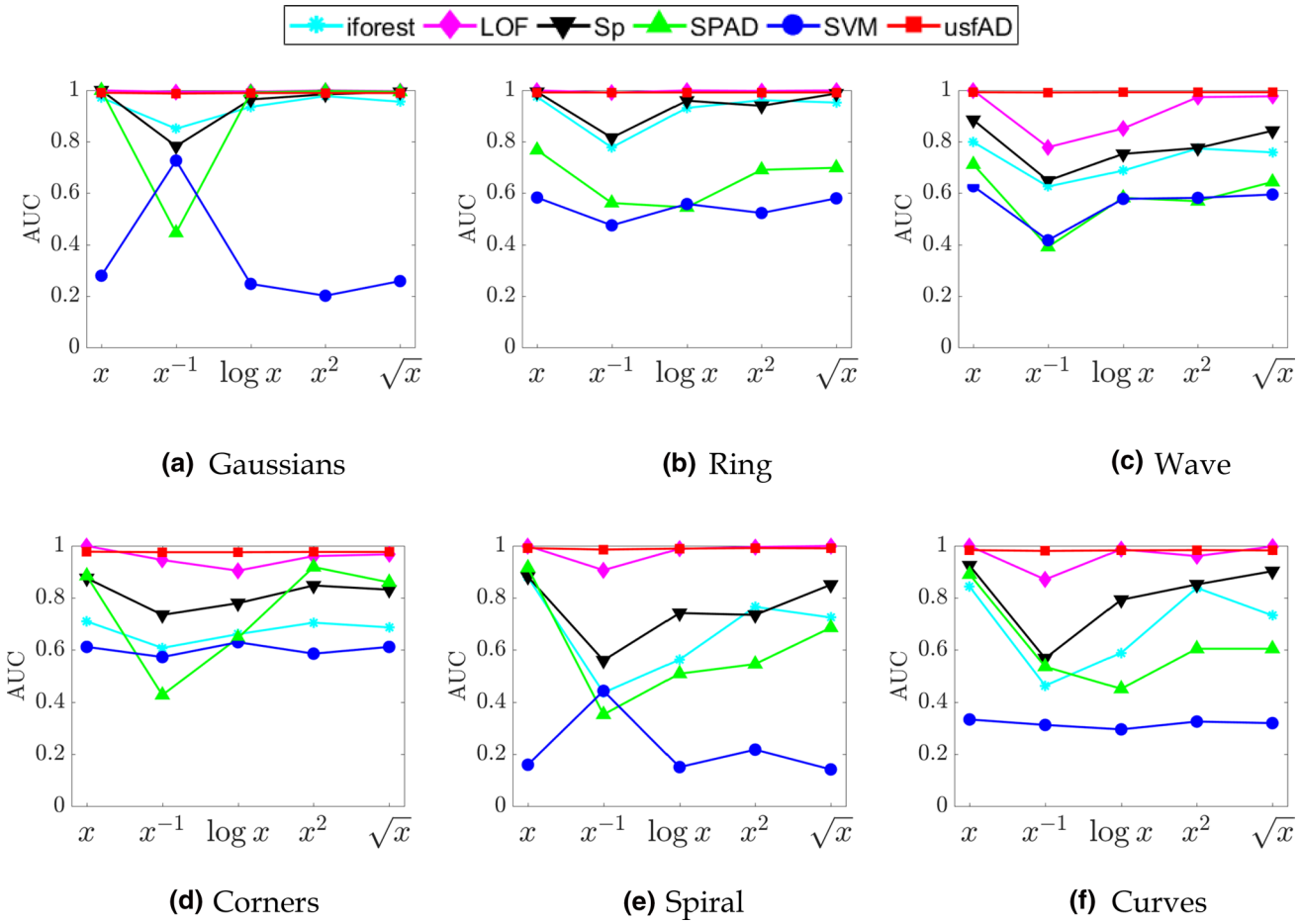


Fig. 5 AUC in the six synthetic datasets with and without non-linear scaling of data

Table 2 Characteristics of cybersecurity datasets used

Dataset	#Dimensions (d)	#Training normal data (n)	Test data	
			#Normal data	#Anomalies
Spambase	57	1394	1394	176
ISCX-URL	75	3889	3890	7570
NSL-KDD	38	38,527	38,527	71,463
UNSW-NB15	39	82,336	82,337	93,000

the same procedure of rescaling as employed by Fernando and Webb [19].

Results in synthetic datasets and real-world cybersecurity datasets are discussed separately in the following two subsections.

5.1 Synthetic datasets

To evaluate the effectiveness and robustness of usfAD and other contenders in detecting anomalies in various data distributions, six two-dimensional datasets as shown in Fig. 4 were used. These datasets represent good examples of scenarios where normal or expected data (represented by blue asterisks) form complex structures and anomalies (represented by red dots) are added at various parts in the data space. In all cases, points represented by red dots clearly look like anomalies visually. We expect existing anomaly detection methods to detect them successfully.

AUCs of the contending methods in the six synthetic datasets in the scales of $x, x^{-1}, \log x, x^2$ and $\sqrt{x}$ are presented in Fig. 5. It clearly shows that usfAD produced the best or equivalent to the best results in all datasets. It produced similar results with and without non-linear scaling of data. It shows that usfAD is robust to how data was measured. It is

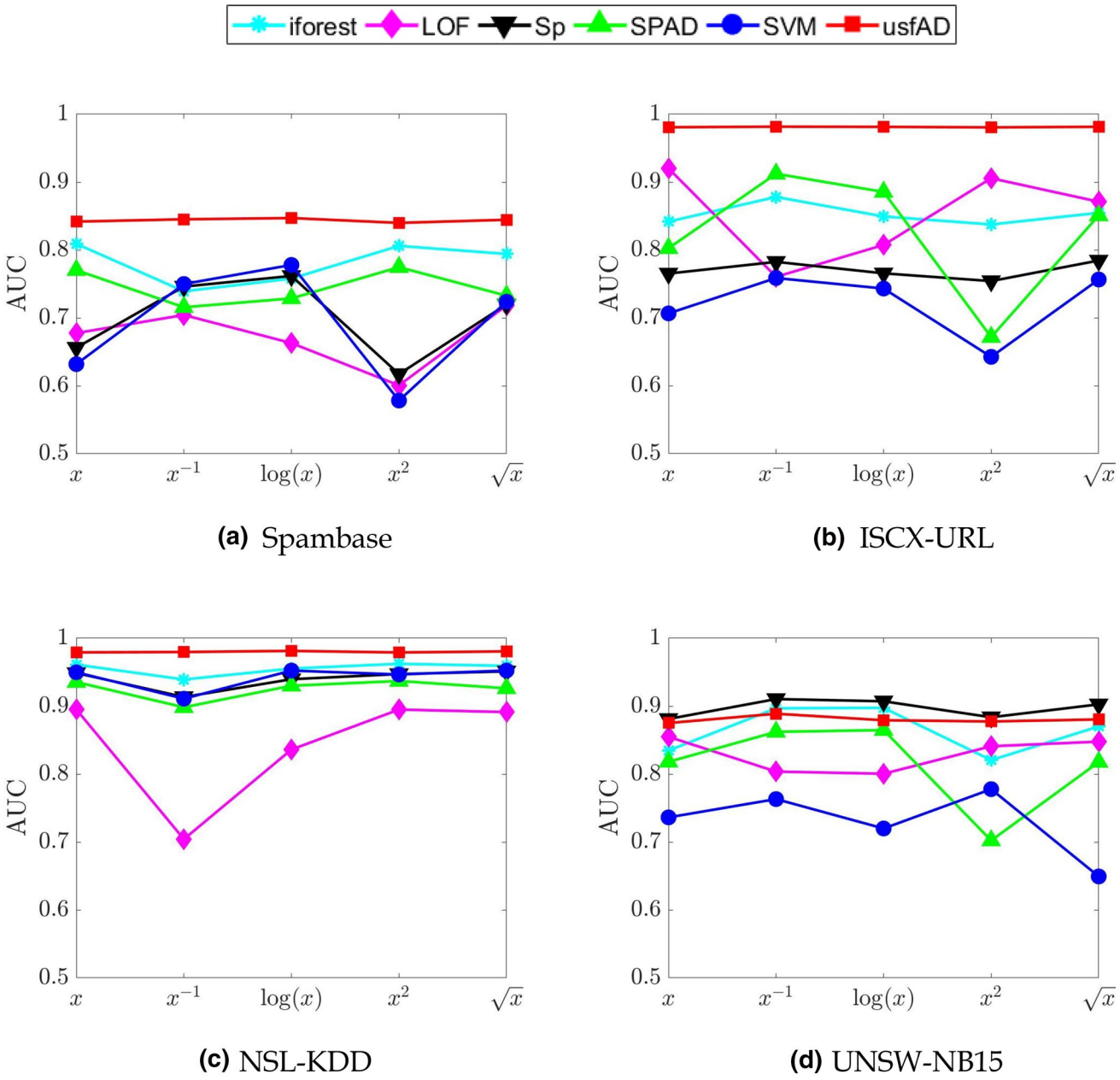
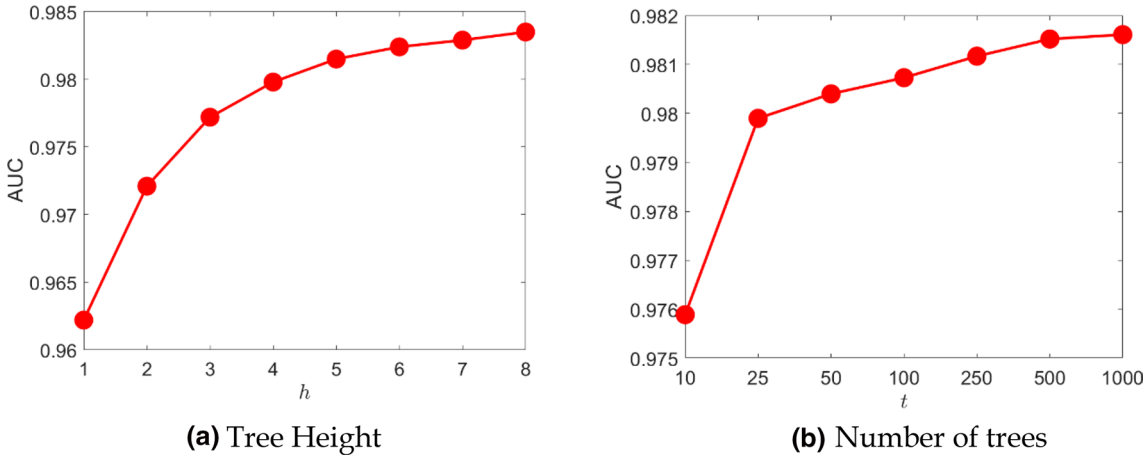


Fig. 6 AUC of contending methods in four widely used benchmark cybersecurity datasets with order preserving and order reversing scaling of data

clear that all five existing anomaly detection methods were sensitive to how data is expressed. It shows LOF is the least sensitive existing method, but its performance dropped with some scaling in some datasets, e.g., inverse and logarithmic scaling in Waves and Corners.

It is interesting to note that four existing methods (SVM, iforest, Sp and SPAD) did not rank all anomalies before normal instances even in the original scale (x) in most datasets. These results demonstrate their limitations. In many of these examples, it is not possible to learn the perfect boundary covering the normal data only without any error. Thus, SVM ranked some normal instances before anomalies resulting in poor AUC. Because some anomalies in these examples lie between normal data in both axes, they cannot be isolated easily in trees. They have longer pathlengths than some instances at the fringe of data distribution in any axis that



(a) Tree Height **(b)** Number of trees

Fig. 7 Sensitivity of two parameters in usfAD: tree height (h) and number of trees (t)

can be isolated early. Thus, iforest failed to detect these types of anomalies. Sp has high variance because of the use of a very small subsample of data resulting in poor performance in datasets with complex structures. Because SPAD computes anomaly scores in each axis separately as it assumes the dimensions are independent. This results in it being unable to detect anomalies that look normal when examined in any individual axis but would have appeared as anomalous when examined from both axes together.

Another interesting result to note is that some existing methods produced better results after rescaling than in the original space, e.g., SVM produced best results with the inverse scaling in Gaussians and Spiral.

5.2 Real-world datasets: cybersecurity

In real-world datasets, four widely used benchmark cybersecurity datasets were used: Spambase[1] (a publicly available emails collection to detect spams), NSL-KDD[2] (a new version of the traditional KDD99 intrusion detection dataset created by resolving some inherent issues and limitations), ISCX-URL[3] (a dataset of benign and malicious URLs) and UNSW-NB15[4] (a widely used network intrusion detection dataset). The characteristics of the four datasets used are provided in Table 2.

Cybersecurity datasets were used because their feature values can be measured or expressed in different forms. For example, in the ISCX-URL dataset, URLs are represented by Lexical features [29]. Several features are based on the ratios of the lengths of various components of URLs such as domain, path and arguments. Some of them are represented in such a way that the ratio is less than 1, whereas others are not. Length of a component (such as domain) is used in numerator in one ratio and as denominator in others, e.g., domain to URL length ratio, path to domain length ratio, argument to path length ratio, etc. Depending on how such features are measured, existing anomaly detectors produce different results.

Anomaly detection results in terms of AUC of existing methods and the proposed usfAD in the four cybersecurity datasets using the given form of data (x) and their nonlinear scalings using $x^{-1}, \log x, x^2$ and $\sqrt{x}$ are presented in Fig. 6. As expected, all five existing methods were sensitive to the scales of data used, their performances varied significantly. Some existing methods produced better results with other scales. For example, in ISCX-URL, AUCs of all existing methods except LOF increased with the inverse scale. LOF produced better result with the inverse scaling than x in Spambase. The proposed method of usfAD produced the same results regardless of the scaling used in all four datasets. It produced the best results in three datasets except in UNSW-NB15, where Sp produced slightly better results than usfAD. However, Sp produced significantly worse results than usfAD in the other three datasets. The results of usfAD were more consistent than any other contender across the four datasets.

In terms of runtime, as expected from the time complexities discussed in Sect. 3, we observed that usfAD was faster than LOF and SVM, and slower than iforest and Sp. For example, in the largest dataset of UNSW-NB15, the total runtime (including training and testing) of usfAD was 437 s compared to 36 s (Sp), 60 s (SPAD), 84 s (iforest), 606 s (LOF), and 1562 s (SVM). Note that we used the

[1] https://archive.ics.uci.edu/ml/datasets/spambase

[2] https://www.unb.ca/cic/datasets/nsl.html

[3] https://www.unb.ca/cic/datasets/url-2016.html

[4] https://www.unsw.adfa.edu.au/unsw-canberra-cyber/cybersecurity/ADFA-NB15-Datasets/

implementations of LOF and SVM available in the Scikit-learn Library which are optimized to run fast using efficient data structures. For other methods (iforest, Sp, SPAD and usfAD), we implemented them without ensuring code was optimized.

5.3 Sensitivity of parameters

There are two parameters in usfAD: tree height (h) and number of trees (t). Note that h determines the subsample size to build each tree, i.e., $\psi = 2^h$. The sensitivity analysis of the two parameters were conducted in the ISCX-URL dataset. To analyze the effect of one parameter, the other parameter was set to the default value. When h was varied from 1 to 8, t was set to the default value of 100; and when t was varied from 10 to 1000, h was set to the default value of 5.

The AUCs of usfAD with different settings of h and t in the ISCX-URL dataset are provided in Fig. 7. In both cases, with the increase in the parameter value, the AUC of usfAD improved significantly for the first few lower values and then gradually flatten out towards the higher values. These results indicate that the parameters are not very sensitive as long as they are set to sufficiently large values. Note that higher values of h and t will increase the runtime linearly. We need to trade-off performance and efficiency. For a right balance, h and t were set to 5 and 100, respectively, as default values. The default settings were used in all experiments conducted in this study. The performance of usfAD can be further improved by proper tuning of the two parameters.

6 Discussion

Results in this research show that the given representation of data may not be appropriate for anomaly detection using existing distance or density-based methods. In the literature, representation learning techniques [12, 30, 43] were used to map data from the given input space into a latent space that maximizes the task specific performance of a given data mining algorithm. Representation learning can be viewed as learning appropriate transformations that best suits the dataset and algorithm at hand. It is primarily used for unstructured data such as image, text, audio and video. Learning an appropriate representation for anomaly detection is very challenging because anomalies can be of different types and can appear anywhere in the data space, i.e., anomalies do not have a particular set of characteristics. Furthermore, representation learning has some other issues too: (1) requires extensive learning which can be computationally expensive in large and/or high-dimensional data sets; (2) learns representation appropriate for the given algorithm, representation learned for one algorithm may not be appropriate for others in the same data set; and (3) one cannot interpret the meaning of new features and what type of information they capture.

Another alternative suggested in the literature to address the issue of units and scales of measurement of data is rank transformation [18]. It is robust to how data is measured or represented because ranks/orders are either preserved or reversed when data is expressed differently. It has been shown that using ranks instead of actual values, distance or similarity based algorithms produces better results for tasks such as classification and clustering [5, 6, 19]. Rank transformation, however does not work for the task of anomaly detection. Because the rank differences of consecutive values are always one irrespective of the magnitude difference, the transformed data is uniformly distributed making anomalies difficult to detect.

To address the above-mentioned issue of rank transformation, Baniya et al. [10] proposed a robust alternative to rank transformation that preserves differences between data instances in the transformed space to some extent. They use average ranks over multiple subsamples of data. Data transformation based on the Average Rank over an Ensemble of Subsamples (ARES) has been shown to improve classification accuracies of algorithms such as k NN, Artificial Neural Networks and Logistic Regression [10]. However, the similar results could not be achieved in the anomaly detection task.

At the first glance, this work may appear to be related to multiscale data analysis [8, 42]. However, they are fundamentally different concepts and they are addressing different issues. In the literature, the term 'multiscale' is used in the context of applications/problems where data can be viewed at different hierarchical levels (scales). For example, (1) temporal data (*e.g.*, daily, monthly, yearly, etc.), images (pixels, shapes, objects, etc.), spatial data (various geographic resolutions). Multiscale data analysis examines data at multiple levels (scales in time, frequency or resolution) and integrate the information in the learning process. There are several works using multiscale data for anomaly detection in spatial and temporal domains [17, 20, 27, 35]. Multiscale data analysis does not address the issue related to the form/format when data is given for analysis. The issue is also applicable while recording data at different levels. Learning algorithms may not perform well if the data at each level are not presented in an appropriate form.

7 Conclusions and future work

In today's world, data is recorded by sensors everywhere around us. The sensors' readings are transferred to a server, and stored in a database before they are analyzed. In this process, the representation of data may have been changed

for various reasons such as sensors' settings, compression to decrease communication and storage costs, and encoding for security reasons. When data is given for analysis, we may not know how they are measured and stored. This makes anomaly detection very challenging. Anomalies can be masked and look like normal data in the given representation and existing anomaly detection algorithms that are primarily based on distance or density may give true negative or false positive results.

This is the first work studying the issue of data representation in the context of anomaly detection. It demonstrates that the fundamental assumption made by almost all existing methods that anomalies are few and different can be counterproductive if data is represented inappropriately. Often in practice, we simply have a bunch of numbers and we may not know how they are represented. Therefore, we need anomaly detection algorithms that do not make such assumptions and are robust to data representation. We believe this work will change the way automatic anomaly detection problem has been studied in the past and lead to a potential paradigm shift in future anomaly detection research.

This paper introduces a simple robust anomaly detection algorithm based on unsupervised stochastic forest called usfAD. Experimental results in synthetic and real-world cyber security datasets with different scaling of data show that it is robust to how data is measured. Because of the median split used to construct trees in the forest, its results are consistent and stable across different scaling in all datasets. The implementation is based on the ordering of data and it is robust to how data is presented as the variation in units/scales either preserves or reverses the orderings. It outperforms most existing state-of-the-art anomaly detection methods even when using the given form of the data. This result shows that some features of the given datasets may not be recorded in the appropriate form for anomaly detection using existing methods, which are sensitive to units/scales of data measurement.

In future, we would like to extend this idea to work in the clustering problem, where most existing state-of-the-art methods use distance or density. Also, we would like to explore a non-tree-based implementation of usfAD. Another potential avenue for future research would be to investigate on scale-invariant representation learning for anomaly detection..

Acknowledgements This paper is an extension of a conference paper published in Proceedings of the 22nd Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD) 2018 [2]. Authors would like to thank Mr Arbind Agrahari Baniya for his help to run some experiments in this extended version of the paper. This material is based upon work supported by the Air Force Office of Scientific Research under award number FA2386-20-1-4005.

Compliance with ethical standards

Conflicts of interest Authors declare no conflict of interest.

References

1. Aggarwal CC (2017) Outlier analysis. Springer, Berlin
2. Aryal S (2018) Anomaly detection technique robust to units and scales of measurement. In: Proceedings of the 22nd Pacific-Asia Conference on Knowledge Discovery and Data Mining, pp 589–601
3. Aryal S, Baniya AA, Santosh K (2019) Improved histogram-based anomaly detector with the extended principal component features. arxiv. https://arxiv.org/abs/1909.12702
4. Aryal S, Ting KM, Haffari G (2016) Revisiting attribute independence assumption in probabilistic unsupervised anomaly detection. In: Proceedings of the 11th Pacific Asia Workshop on Intelligence and Security Informatics, pp 73–86
5. Aryal S, Ting KM, Washio T, Haffari G (2017) Data-dependent dissimilarity measure: an effective alternative to geometric distance measures. Knowl Inf Syst 53(2):479–506
6. Aryal S, Ting KM, Washio T, Haffari G (2020) A comparative study of data-dependent approaches without learning in measuring similarities of data objects. Data Min Knowl Disc 34(1):124–162. https://doi.org/10.1007/s10618-019-00660-0
7. Aryal S, Ting KM, Wells JR, Washio T (2014) Improving iForest with Relative Mass. In: Proceedings of the 18th Pacific Asia Conference on Knowledge Discovery and Data Mining (PAKDD), pp 510–521
8. Bakshi BR (1999) Multiscale analysis and modelling using wavelets. J Chemom 13(1):415–434
9. Bandaragoda T, Ting KM, Albrecht D, Liu F, Wells J (2014) Efficient anomaly detection by isolation using nearest neighbour ensemble. In: Proceedings of the IEEE international conference on data mining workshops, pp 698–705
10. Baniya AA, Aryal S, Santosh KC (2019) A novel data pre-processing technique: making data mining robust to different units and scales of measurement. In: Proceedings of the 26th international conference on neural information processing (ICONIP) of the Asia-Pacific Neural Network Society, (p. Accepted)
11. Bay SD, Schwabacher M (2003) Mining distance-based outliers in near linear time with randomization and a simple pruning rule. In: Proceedings of the ninth ACM SIGKDD conference on knowledge discovery and data mining, pp 29–38
12. Bengio Y, Courville A, Vincent P (2013) Representation learning: a review and new perspectives. IEEE Trans Pattern Anal Mach Intell 35(8):1798–1828
13. Boriah S, Chandola V, Kumar V (2008) Similarity measures for categorical data: a comparative evaluation. In: Proceedings of the eighth SIAM international conference on data mining, pp 243–254
14. Breiman L (2001) Random forests. Mach Learn 45(1):5–32
15. Breunig MM, Kriegel H-P, Ng RT, Sander J (2000) LOF: identifying density-based local outliers. In; Proceedings of ACM SIGMOD conference on management of data, pp 93–104
16. Chandola V, Banerjee A, Kumar V (2009) Anomaly detection: a survey. ACM Comput Surv 41(3):15-1-15–58
17. Cheng T, Li Z (2006) A multiscale approach for spatio-temporal outlier detection. Trans GIS 10(2):253–263
18. Conover WJ, Iman RL (1981) Rank transformations as a bridge between parametric and nonparametric statistics. Am Statist 35(3):124–129

19. Fernando TL, Webb GI (2017) SimUSF: An efficient and effective similarity measure that is invariant to violations of the interval scale assumption. Data Min Knowl Disc 31(1):264–286
20. Gao Z, Guo L, Ma C, Ma X, Sun K, Xiang H, Liu X et al (2019) AMAD: adversarial multiscale anomaly detection on high-dimensional and time-evolving categorical data. In: Proceedings of the 1st international workshop on deep learning practice for high-dimensional sparse data (DLP-KDD '19), pp 1–8
21. Goldstein M, Dengel A (2012) Histogram-based outlier score (HBOS): a fast unsupervised anomaly detection algorithm. In: Proceedings of the 35th German Conference on Artificial Intelligence, pp 59–63
22. Hand DJ, Till RJ (2001) A simple generalisation of the area under the roc curve for multiple class. Mach Learn 45(2):171–186
23. Hawkins DM (1980) Identification of outliers. Chapman and Hall, London
24. Jiang H, Wang H, Hu W, Kakde D, Chaudhuri A (2017) Fast incremental SVDD learning algorithm with the Gaussian Kernel. In: Proceedings of the Thirty-Third AAAI conference on artificial intelligence (AAAI), pp 3991–3998
25. Joiner BL (1981) Lurking variables: some examples. Am Statist 35(4):227–233
26. Liu F, Ting KM, Zhou Z-H (2008) Isolation forest. In: Proceedings of the Eighth IEEE international conference on data mining, pp 413–422
27. Liu Q, Klucik R, Chen C, Grant G, Gallaher D, Lv Q, Shang L (2017) Unsupervised detection of contextual anomaly in remotely sensed data. Remote Sens Environ 202(1):75–87
28. Lord FM (1953) On the statistical treatment of football numbers. Am Psychol 8(12):750–751
29. Mamun MS, Rathore MA, Lashkari AH, Stakhanova N (2016) Detecting malicious URLs using lexical analysis. In: Proceedings of the international conference on network and system security (NSS 2016), pp 467–482
30. Pang G, Cao L, Chen L, Liu H (2018) Learning representations of ultrahigh-dimensional data for random distance-based outlier detection. In: Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining, pp 2041–2050
31. Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Duchesnay E et al (2011) Scikit-learn: machine learning in python. J Mach Learn Res 12:2825–2830
32. Rekha AG (2015) A fast support vector data description system for anomaly detection using big data. In: Proceedings of the 30th Annual ACM symposium on applied computing (SAC), pp 931–932
33. Scholkopf B, Platt JC, Shawe-Taylor J, Smola AJ, Williamson RC (2001) Estimating the support of a high-dimensional distribution. Neural Comput 13(7):1443–1471
34. Shi T, Horvath S (2006) Unsupervised learning with random forest predictors. J Comput Graph Stat 15(1):118–138
35. Siddiqui S, Khan MS, Ferens K (2017) Multiscale Hebbian neural network for cyber threat detection. In: Proceedings of the international joint conference on neural networks (IJCNN), pp 1427–1434
36. Stevens SS (1946) On the theory of scales of measurement. Science 103(2684):677–680
37. Sugiyama M, Borgwardt KM (2013) Rapid distance-based outlier detection via sampling. In: Proceedings of the 27th annual conference on neural information processing systems, pp 467–475
38. Tax D, Duin R (2004) Support vector data description. Mach Learn 54(1):45–66
39. Ting KM, Washio T, Wells JR, Aryal S (2017) Defying the gravity of learning curve: a characteristic of nearest neighbour anomaly detectors. Mach Learn 106(1):55–91
40. Townsend JT, Ashby FG (1984) Measurement scales and statistics: the misconception misconceived. Psychol Bull 96(2):394–401
41. Velleman PF, Wilkinson L (1993) Nominal, ordinal, interval, and ratio typologies are misleading. Am Stat 47(1):65–72
42. Weinan E (2011) Principles of multiscale modeling (Vol 6). Cambridge University Press, Cambridge
43. Zhong G, Wang L-N, Ling X, Dong J (2016) An overview on data representation learning: from traditional feature learning to recent deep learning. J Financ Data Sci 2(4):265–278