Chapter 7
ITL: An Isolation-Tree-Based Learning of Features for Anomaly Detection in Networked Systems

Sara Kardani Moghaddam, Rajkumar Buyya, and Kotagiri Ramamohanarao

Contents

With the advances in monitoring techniques and storage capability in the cloud, a high volume of valuable monitoring data is available. The collected data can be used for profiling applications behavior and detecting anomalous events that identify unexpected problems in the normal functioning of the system. However, the fast-changing environment of the cloud brings a need for fast and efficient analytic solutions to monitor the cloud system for its correct operational behavior. The isolation-based method is an effective approach for detecting anomalies. This method randomly samples the data and builds several isolation-trees (iTrees) data structures to find anomalous records. However, a common challenge of iTrees as well as other anomaly detection algorithms is dealing with high-dimensional data that can impact the accuracy and execution time of the process. This is an important issue for cloud-hosted applications where a variety of problems are constantly changing the normal pattern of features from low-level network data to high-level

S. Kardani-Moghaddam · R. Buyya (✉) · K. Ramamohanarao
The University of Melbourne, Melbourne, VIC, Australia
e-mail: skardani@student.unimelb.edu.au; rbuyya@unimelb.edu.au

R. Buyya et al. (eds.), *New Frontiers in Cloud Computing and Internet of Things*, Internet of Things, https://doi.org/10.1007/978-3-031-05528-7_7

application performance. Therefore, refining the feature space for the removal of irrelevant attributes is a critical issue.

In this chapter, we introduce an iterative iTree based Learning (ITL) algorithm to handle high-dimensional data. ITL takes the advantage of iTree structure to learn relevant features for detecting anomalies. Initially, it builds iTrees making use of all features of the data. Then, in the iterative steps, it refines the set of the features to find the most relevant ones by selecting highly ranked anomalies discovered in the previous iteration. Experiments are conducted to validate the performance of our proposed ITL method on several benchmark datasets. The results show that ITL can achieve significant speedups with the appropriate choice of the number of iTrees while achieving or exceeding the area under the curve (AUC) values of other state-of-the-art isolation-based anomaly detection methods.

7.1 Introduction

Anomaly detection is an important field of knowledge discovery with rapid adoption in a variety of applications. The main goal is to find interesting patterns in the data that deviate from the expected behavior of the application. In the context of the cloud environment, this process is utilized for a variety of performance management applications. For example, intrusion detection systems provide frameworks that monitor the performance of the network to find misbehaving users, possible misconfiguration, or serious conditions from an attack on the system [1]. Similarly, SMART (Self-Monitoring, Analysis, and Reporting Technology)-based disk failure prediction applications perform regular monitoring and anomaly detection analysis to increase the reliability of storage systems [2].

With the advances in data collection techniques, storage capabilities, and high-performance computing, a huge volume of monitoring data is collected from continuous monitoring of the system attributes. Despite the appealing benefits of access to larger amounts of data for better diagnostics of anomalous events, the great challenge is how to deal with a high volume of information that should be processed effectively in real time. The increase in the volume of data is due to (1) recording of fine-grained measurements for long periods which increases the number of records to be processed and (2) high-dimensional data with many features that describe various aspects of the target system. The curse of dimensionality or having many features can make the problem of anomaly detection in high-dimensional data more complex in terms of the runtime efficiency and accuracy [3]. This is also becoming a critical issue in cloud systems which are exposed to several performance problems at different layers of computing. As a result, the collected performance data is heterogeneous and includes a variety of attributes from low-level operation logging data to hardware-specific features, application performance data, or network-related information. On the other hand, these performance data are exposed to a variety of problems such as different types of attacks and intrusion patterns in network-related performance data. Particularity, the general anomaly detection techniques cannot perform well for high-dimensional network data with a variety of data types and embedded meaningful subspaces from different sources [4]. Moreover, the collected data is dynamic and rapidly changing. All of these, together, highlight the need for highly adaptable and fast analytic solutions. Therefore, researchers are investigating more efficient techniques with the goal of better explorations of collected data and improving the quality of the extracted knowledge.

Traditional anomaly detection algorithms usually work based on the assumptions that highly deviated objects in terms of the common metrics such as distance or density measures have a higher probability of being anomalous (outliers in statistical methods). While these assumptions are applicable in general, their accuracy can be affected when the base assumptions do not hold, such as in the high-dimensional data [5, 6]. Moreover, in the traditional methods, anomalies are detected as a by-product of other goals such as classification and clustering. More recent approaches, such as isolation-based methods, directly target the problem of anomaly detection with the assumption that anomalies are few and different [7, 8]. However, the problem of having a high number of noisy features can also affect these methods. To improve the efficiency of detection algorithms in high-dimensional data, a variety of solutions such as random feature selection or subspace search methods are proposed [9, 10]. However, the proposed approaches are usually considered as the preprocessing steps which are performed as a separate process from the anomaly detection. Although this separation makes them applicable for a variety of algorithms, finding the relevant features in the datasets with many noisy features can be challenging when the mechanism of target detection algorithms in finding the anomalies is ignored. Therefore, a question arises that is there a way that one can improve the efficiency of anomaly detection by extracting knowledge from the assumptions and the process which leads to identifying potential anomalies in the data? Having this question in mind, in this chapter, we address the problem of anomaly detection in high-dimensional data by focusing on the information that can be extracted directly from the isolation-based mechanism for identifying anomalies. The reason for selecting this technique as the base process is that it is designed to directly target the most common characteristics of anomalous events such as rarity compared to other objects. We exploit the knowledge that comes from the detection mechanism to *identify* the features that have higher contribution in the separation of the anomaly instances from normal ones. This approach helps to identify and remove many irrelevant noisy features in high-dimensional data. The proposed method, Isolation-Tree (iTree)-Based Learning (ITL), addresses the problem of anomaly detection in high-dimensional data by refining the set of features to improve the efficiency of the detection algorithm. These are the features that appear in the short branches of iTrees. The refining procedure helps the algorithm to focus more on the subset of features where the chances of finding anomalies are higher while reducing the effect of noisy features. The process helps to obtain more informative anomaly scores and generates a reduced set of features that improves the detection capability with better runtime efficiency in comparison to the original method that uses all the features.

Accordingly, the major contributions of this work are as follows:

- Proposed an iterative mechanism for structural learning of data attributes and refining features to improve the detection efficiency of isolation-based methods.
- Exploited the inherent knowledge from iTree data structure to reduce the effect of noisy and irrelevant features.
- Efficiently found an essential subset of the features that can effectively detect anomalies. The simplified model is extremely fast to train so that the model can be periodically trained when the important features largely remain unchanged.
- Carried out experiments on network intrusion datasets and other high-dimensional benchmark datasets to demonstrate the effectiveness of ITL in improving the anomaly detection results as well as runtime efficiency.

ITL focuses on the data engineering part of data analytics which also helps to speed up the process of anomaly detection. We have compared ITL with the state-of-the-art feature learning-based framework [11] and show that not only ITL improves results as an ensemble learning method with the bagging of scores, but also it can discover a subset of the features that can detect anomalies with reduced complexity. Since anomalies can be different and dependent on the context of the datasets, we have considered the heterogeneity by analyzing different benchmark datasets with a variety of attributes and anomaly patterns.

The remainder of this chapter is structured as follows. Section 7.2 reviews some of the related works in the literature. Section 7.3 overviews the basic idea of isolation-based anomaly detection and the main assumptions in the problem formulation. Section 7.4 presents the ITL framework and details the steps of the algorithm. Section 7.5 presents experiments and results followed by time complexity and runtime analysis, and finally, Sect. 7.6 concludes the chapter with directions for future research.

7.2 Related Work

The general concept of anomaly detection indicates the exploration and analysis of data to find patterns that deviate from the expected behavior. The concept has been widely used and customized in a range of applications such as financial analysis, network analysis and intrusion detection, medical and public health, etc. [1, 12, 13]. The growing need for anomaly related analysis has led researchers to propose new ways of addressing the problem where they can target unique characteristics of anomalous objects in the context of the target applications. For example, distance-based algorithms address the problem of anomaly detection based on the distance of each instance from neighborhood objects. The greater the distance, the more likely it that the instance presents abnormal characteristics in terms of the values of the features [14, 15]. Alternatively, [16, 17] define the local density as a measure for abnormality of the instances. The objects with a low density in their local regions have a higher probability of being detected as an anomaly. Ensemble-based methods try to combine multiple instances of anomaly detection algorithms to improve the searching capability and robustness of the individual solutions [11, 18].

Performance anomaly detection has also widely been applied in the context of cloud resource management to identify and diagnose performance problems that affect the functionality of the system. These problems can happen at different levels of granularity from code-level bug problems to hardware faults and network intrusions. The fast detection of the problem is a critical issue due to the high rate of changes and volume of information from different sources. A variety of techniques from statistical analysis to machine learning solutions are used to process collected data. For example, Principal Component Analysis (PCA) is used in [19] to identify the most relevant components to various types of faults. Xiao et al. [2] applies random forests on various exported attributes of drive reliability to identify disk failures. Dean et al. [20] exploits the self-organizing map technique to proactively distinguish anomalous events in virtualized systems. Clustering techniques are utilized by [21] to split the network-related log data into distinctive categories. The generated clusters are then analyzed separately by anomaly detection systems to identify intrusion and attack events. Cao et al. [22] uses entropy concept on network and resource consumption data to identify denial of service attacks.

While the abovementioned approaches show promising results for a variety of problems, the exploding volume and speed of the data to be analyzed require complex computations which are not time efficient. A common problem that makes these difficulties even more challenging is the high-dimensional data. For example, the notion of distance among objects loses its usability as a discrimination measure as the dimension of data increases [3, 5]. Methods based on the subspace search or feature space projections are among approaches that are proposed as possible solutions for these problems [23]. The idea of dividing high-dimensional data into groups of smaller dimensions with related features is investigated in [24]. This approach requires a good knowledge of the domain to define meaningful groups. PCA-based methods try to overcome the problem by converting the original feature set to a smaller, uncorrelated set which also keeps as much of the variance information in data as possible [25]. PINN [26] is an outlier detection strategy based on the Local Outlier Factor (LOF) method which leverages random projections to reduce the dimensionality and improve the computational costs of LOF algorithm. Random selection of the features is used in [27] to produce different subspaces of the problem. The randomly generated sub-problems are fed into multiple anomaly detection algorithms for assigning the anomaly scores. While random selection can improve the speed of the feature selection process, as the selection is completely random there is no guarantee of having informative subspaces of data to improve the final scoring. A combination of correlation-based grouping and kernel analysis is applied in [28] for feature selection and anomaly detection in time-series data. Feature reduction is done by selecting representative features of final clusters. Keller et al. [9] and Pang et al. [11] propose two different variations of subspace searching. The former tries to find high contrast subspaces of the problem to improve the anomaly ranking of density-based anomaly detection algorithms. The subspace searching is based on the statistical features of the attributes and is performed as a preprocessing step separated from target anomaly detection algorithms. The latter, in contrast, integrates the subspace searching as a sequential refinement and learning in anomaly detection procedure where the calculated scores are used as a signal for the selection of the next subset of the features. Our proposed anomaly detection approach is inspired by such models and tries to refine the subset of the selected features at each iteration. However, we try to take advantage of the knowledge from the structure of constructed iTrees instead of building new models for the regression analysis.

7.3 Model Assumptions and an Overview on Isolation-Based Anomaly Detection

The iterative steps in the ITL process are based on the iTree structure for assigning the anomaly scores as well as identifying features. We choose the isolation-based approach and specifically IForest algorithm [7, 29] in this work due to its simplicity and the fact that they target the inherent characteristic of the anomalies as being rare and different without any prior assumption on their distributions. We note that the target types of the anomaly in this work are instances that are anomalous in comparison to the rest of the data and not as a result of being part of the larger groups [1, 30]. This is also consistent with the definition of anomaly in many cloud-related performance problems especially network and resource abnormalities.

The idea of Isolation-based methods is that for an anomaly object we can find a small subset of the features that their values are highly different compared to the normal instances, and therefore it can quickly be isolated in the feature space of the problem. IForest algorithm demonstrates the concept of the isolation and partitioning of the feature space through the structure of trees (iTrees), where each node represents a randomly selected feature with a random value and existing instances create two new child nodes based on their values for the selected feature. It is demonstrated that the anomaly instances usually create short branches of the tree, and therefore, the *length of the branch* is used as a criterion for the ranking of the objects [29]. Consequently, anomaly scores are calculated as a function of the path length of the branches that isolates the instance in the leaf nodes on all generated iTrees. This process can be formulated as follows [7]: let $h_t(x)$ be the path length of instance x on iTree t. Then, the average estimation of path length for a subset of N instances can be defined as Eq. 7.1:

$$C(N) = \begin{cases} 2H(N-1) - 2\frac{(N-1)}{N} & if N > 2, \\ 1 & if N = 2, \\ 0 & otherwise \end{cases} \tag{7.1}$$

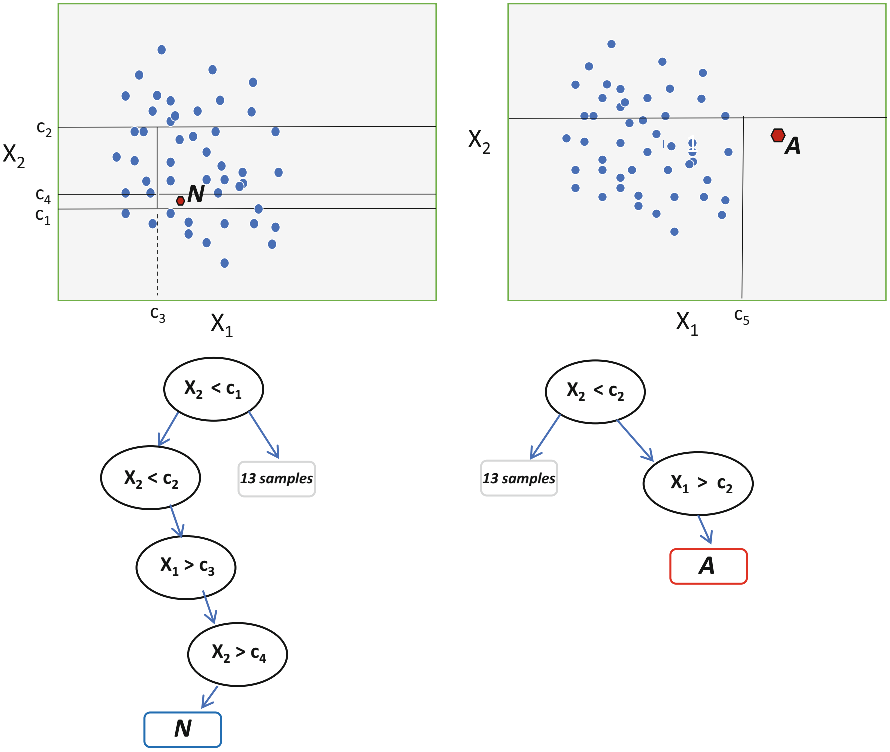

Fig. 7.1 Isolation-based anomaly detection. iTree structures are used to represent the partitioning and isolation process of instances in a dataset with two attributes. The left and right columns show example sequences of partitions to isolate normal and anomaly instances, respectively

where $H(N)$ is the harmonic number and can be calculated as $ln(N) + Euler\_Constant$. Using $C(N)$ for the normalization of expected h(x) of instance x on all trees, the anomaly scores can be calculated as follows:

$$s(x, N) = 2^{-\frac{E(h(x))}{C(N)}} \tag{7.2}$$

Considering this formula, it is clear that anomaly scores have an inverse relation with the expected path length. Therefore, when the average path length of an instance is close to zero, the anomaly score is close to 1, and vice versa.

Figure 7.1 shows a graphical representation of the isolation technique for a dataset with two attributes X_1 and X_2. The left and right columns show examples of random partitions on the attribute space and their corresponding tree structures to isolate a normal and anomaly instance, respectively. As it is shown, instance A (anomaly) can be isolated quickly considering the sparsity of values of X_1 around this instance. Though this example is a simple case with just two attributes, the general idea can be extended to the problems with many features and a variety of distributions.

Considering the above explanations, ITL process is based on the idea that iTrees can also give information on important features for detection purpose. Therefore, ITL analyzes the generated iTree structure to extract information about the features that have more contribution in creating short branches and detected anomalies. In order to better explain the problem, let us assume that the input D is a matrix of N instances, each instance explained with a row of M features such that

$$D = \{(X_i), 1 \leq i \leq N | X_i = (x_{ij}), 1 \leq j \leq M, \\ x_{ij} \in R\}. \tag{7.3}$$

We have excluded nominal data in our assumptions and definition of Eq. 7.3. However, the ITL process is general and can be combined with solutions that convert categorical data to numerical to cover both cases [11]. We formulate the problem as follows: given a matrix D as the input, we try to iteratively remove some irrelevant features from the feature space of D, keeping the more relevant features for the detected anomalies at each step in an unsupervised manner. The goal is to increase the quality of the scores in terms of assigning higher scores to the true anomaly points by reducing the effect of noisy features. The output at each step k is a set of the scores S_k on a set of the reduced features M_k. The idea is that the removal of noisy features makes it easier to focus on the relevant partitions of the data, where the values of the features show higher deviations for the anomalous objects in comparison to the normal ones. As a result, the ranking of the input objects would improve with regard to the true detected anomalies.

7.4 ITL Approach

Figure 7.2 shows the main steps in ITL framework. As we already discussed in Sect. 7.3, the iTree structure forms the base of the ITL learning phase following the assumption that short branches in the structure of iTree are generated by the attributes with higher isolation capability. In another word, a subset of the attributes that are creating the nodes in the short-length branches can from a vertical partition of the data that localize the process on anomaly related subset of the data. As we can see in Fig. 7.2, the process is completely unsupervised with the input matrix as the only input of each iteration (that is we have no information of anomalous instances). There are four main steps in the ITL process and these are:

1. Building iTrees Ensemble: IForest creates a set of the iTrees from input data. This is a completely unsupervised process with a random sampling of the instances/features to create the splitting nodes in each tree.

Algorithm 1: ITL process

input : $D = (X_1, X_2, ..., X_N)$, $X_i \in R^M$: D is a matrix of N records, each record including M features

Parameter: th: Anomaly score threshold value

output : Reduced Matrix, Scores

$D' \leftarrow D$

while *not (There are unseen features)* **do**

  Build $iTrees$ ensemble using iForest on D'

  `/* Calculate scores for all input instances using Eq. 7.2 */`

  $S = (S_k) = (s_{k1}, s_{k2}, ..., s_{kN}) \leftarrow Scores(iTrees, D')$

  `/* Filter a small part of the input matrix with higher anomaly scores */`

  $D\_subset \leftarrow \{x_i |\, x_i \in D' \;\&\&\; s_i \geq th\}$

  initialize $Frequency$ as an array with length equal to the number of features in D_subset all equal to zero

  for $tree \in iTrees$ **do**

    for $x \in D\_subset$ **do**

      update $Frequency$ of features by adding the occurrences of each attribute seen while traversing from the root node to the leaf node that isolates x

    end

  end

  $D' \leftarrow \{x_{ij} | x_{ij} \in D' \;\&\&\; frequency(j) \geq Average(frequency)\}$

end

return(D', S)

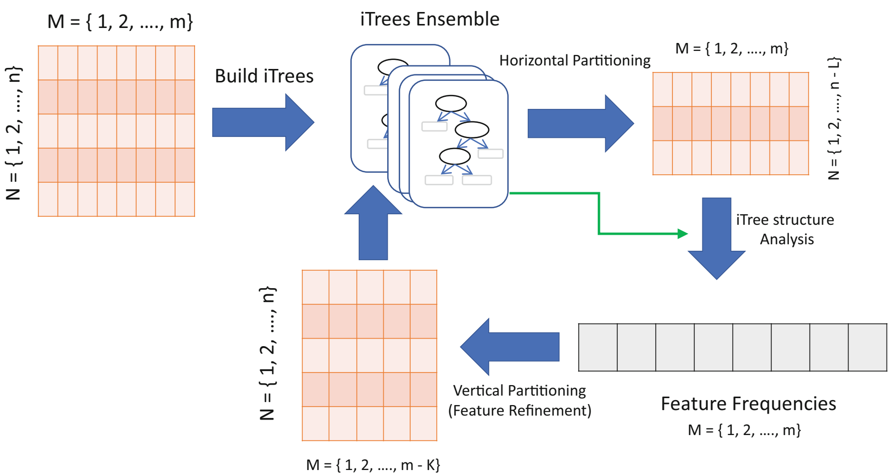

Fig. 7.2 ITL framework. The initial input is a matrix of N instances with M features. An ensemble of iTrees is created. Then, top ranked identified anomalies are filtered. The iTrees are analyzed for filtered instances to create a list of ranked features

2. Horizontal partitioning: The anomaly score for each instance is computed based on the length of the path traversed by the instance on the generated iTrees [7]. The final score shows the degree of outlierness of the instance. Our goal is to discover important features based on their contribution in the isolation of anomaly instances. The low score instances do not affect the determination of the important features for anomaly detection, and therefore, we can remove them to reduce the data size.
3. Extracting Feature Frequencies: We create a frequency profile of occurrences of different features observed during the traversal of short branches of iTrees. These features have a high probability of detecting anomalous instances.
4. Vertical Partitioning: Having a profile of the feature frequencies, a subset of the features with a higher contribution in the abnormality of data are selected and other features are removed. This process creates a vertically partitioned subset of data as the input for the next iteration of the ITL.

This process is repeated multiple times until the termination condition is met. As we continuously refine the features, we expect to see improvement in the anomaly detection process as the detection process becomes more focused on the interesting set of features. Therefore, the set of iTrees built during consecutive steps can be combined to create a sequence of the ensembles. Algorithm 1 shows the pseudo-code of ITL. A more detailed and formal description of the process is presented in the following section.

7.4.1 Feature Refinement Process

We assume that the input D is a matrix of objects labeled as one of the classes of normal or anomaly. These labels are not part of the ITL process as it is an unsupervised mechanism. They are used for evaluating the output results of proposed algorithms and other benchmarks for validation purpose. The goal is to find a ranking of the objects so that the higher values imply higher degrees of abnormality. Considering this objective, the first step of ITL process is to build the initial batch of the iTrees from the input matrix. IForest is used to create t iTrees. To create each iTree, ψ random instances are selected from D, and each node of the tree is created by randomly selecting a feature and a value and splitting the instances based on this selection to form two branches. The output is iTrees ensemble and anomaly scores $S = (s_1, s_2, ..., s_N)$ computed for all instances based on Eq. 7.2 (Lines 3–4, Algorithm 1).

After creating new iTrees, the next step is to reduce the target instances for the learning procedure (Line 5). A threshold value (th) is defined and all instances with an anomaly score lower than this value are discarded. The idea behind this selection is to focus better on parts of the data which have a higher degree of abnormality based on the iTrees structure as well as reducing the complexity of the problem. As the learning phase is the most time-consuming part of the ITL process, this reduction dramatically decreases the runtime of the algorithm. The selection can also take advantage of the expert knowledge on the characteristics such as the contamination ratio of the dataset for defining a proper cut-off value of anomaly scores. The output of this step is a subset of the input matrix D (D') with p instances such that $p << |D|$. We emphasize that the process is unsupervised as we do not have the knowledge of anomalous instances. However, based on the assumption that anomalies are few and different, we expect to see many of the anomaly instances in D'. It should also be noted that the generation of each iTree is completely random in terms of the splitting features and value selection. Therefore, one tree may not be informative per se. However, when the random process is repeated to generate many trees, the overall observed patterns confirm the idea of short branch isolation of anomaly instances [7]. This can be observed in Fig. 7.1 as well. Generating iTree structure on high-density regions requires many nodes and splitting conditions to isolate one instance, while for an anomaly instance there is one feature or more that can quickly differentiate that from the rest of the data.

The instances that passed the filtering procedure from the previous step (highly ranked anomalies) are processed by each iTree from the ensemble model to record the frequency of occurrences of features when traversing the trees. The frequency profile of features allows determining important features relevant to detecting target anomalies. According to the formulas in Sect. 7.3 and their interpretation as an iTree structure, we expect to see a subset of more important features for anomalous instances in the short branches of trees. It should be noted again that these are the expected observations from an ensemble of many random trees and are not attributed to any specific iTree. Consequently, we keep the features whose frequencies are higher than the average of the frequency profile (Lines 6–12).

The above steps are repeated multiple times. The output is a set of the anomaly scores for each subset of the data, starting from full data with all features. Therefore, the iteration k of ITL process creates a set S_k of anomaly scores for all instances on the reduced feature set M_k (M_0 is the full set of the features for the first iteration). We note that each iteration would produce a potentially different set of anomalous points and hence a different frequency profile of the features. The termination condition we choose is when the frequency of occurrences for all features is greater than one, meaning that every feature has seen at least one anomalous point in the short branches of iTrees. The idea behind this condition is that as the noisy features are removed during the iterative process, ITL produces better iTrees for detecting the true set of anomalies. Therefore, the observed features become more important in the detection process. When ITL reaches a state that all the features are present in the short branches, it indicates that all current features are contributing to the detection of anomalous instances. Therefore, the termination condition T_k at iteration k is evaluated as follows:

$$T_k = \begin{cases} True & \text{if } Size(M_k) \leq 1 \text{ or} \\ & Frequency_k > 0 \\ False & \text{otherwise,} \end{cases} \tag{7.4}$$

where $Size(M_k)$ evaluates the number of remaining features at the iteration k. $Frequency_k$ is the corresponding frequency profile which is an array of length M initialized by zero (Line 6). The term $Frequency_k > 0$ evaluates the condition that the frequencies of all attributes in M_k are greater than zero. When T_k evaluates to true, ITL process terminates and the final outputs are evaluated as follows:

- *Bagging of the Scores*: Each iterative step of the ITL process produces a score for each data point in D, which represents the degree of anomalousness based on the corresponding set of the reduced features. As we try to improve the detection capability of the ensemble by reducing the noisy features, we expect to get better anomaly scores in terms of the ranking of instances. Therefore, in this approach, the goal is to take advantage of detection results from all iterations by averaging the scores and defining a new score for each instance. Accordingly, the final score of each instance is calculated as follows:

$$S_f(x) = \frac{1}{K} \sum_{k=1}^{k=K} S_k(x), \tag{7.5}$$

  where S_f is the final score and S_k is the score at iteration k from K iterations of ITL process.
- *Reduced Level Scores*: ITL produces an ensemble of iTrees on the important features for anomaly detection. The generated iTrees on the reduced features can be used for detecting anomalies in new data. Therefore, the anomaly scores are calculated directly based on the extracted reduced feature set from the process.

7.5 Performance Evaluation

In this section, an empirical evaluation of ITL process on two network intrusion datasets and three other benchmark datasets is presented. The two sets of experiments are designed to demonstrate the behavior of ITL in bagging and reduced modes on the target datasets. First, Sect. 7.5.1 presents the datasets and parameter settings of the experiments. Then, Sect. 7.5.2 shows the comparison results of ITL in the bagging mode with a recently proposed state-of-the-art sequential ensemble learning method and then investigates the improvements made by reduced level features in terms of both AUC and runtime analysis in a set of the cross-validated experiments. Sections 7.5.2.3 and 7.5.3 discuss runtime complexity and weakness/strength points of ITL approach.

Table 7.1 Properties of data used for experiments. N and M are the number of instances and features in each dataset, respectively

	N	M	Anomaly ratio (%)
DOS	69363	37	3
U2R	69363	37	3
AD	3279	1558	13
Seizure	11500	178	20
SECOM	1567	590	6

7.5.1 Experimental Settings

Table 7.1 shows a summary of statistics for the benchmark datasets. All datasets are publicly available in UCI machine learning repository [31].[1] For U2R and DOS datasets which are network intrusion dataset from Kddcup99, a downsampling of attack classes is performed to create the anomaly class. In other datasets, the instances in the minority class are considered as the anomaly.

In order to evaluate the results, we select the Receiver Operating Characteristics (ROCs) technique and present Area Under the Curve (AUC) as a measure of the accuracy of the system. AUC value summarizes the trade-off between true positive and false positive detection rates as shown in Eqs. 7.6 and 7.7 under different threshold values. Higher AUC indicates better performance with regard to the detected anomalous instances.

$$TPR = \frac{TP}{TP + FN} \tag{7.6}$$

$$FPR = \frac{FP}{FP + TN}. \tag{7.7}$$

ITL process is implemented based on the publicly available Python library, scikit-learn [32]. Unless otherwise specified, the values of the parameters for iTree generation step of ITL process are according to the recommended settings as explained in [7]. The value of other parameters is set based on the experimental tunings. The threshold value for the horizontal partitioning (*th* in Algorithm 1) is determined by assuming a contamination ratio equal to 0.05% for all datasets. This means that the cut-off threshold is identified so that 0.05% of the objects have a score higher than the *th*, which is good enough considering the number of instances and the contamination ratio in our target datasets. The frequency profiling is done on the branches with a maximum length of 4. This threshold has been selected based on the average length of the trees which is dependent on the sample size and therefore constant in all experiments. To ensure comparativity, the number of trees for the IForest algorithm in all methods is the same and is between 600 and 900 trees.

[1] http://archive.ics.uci.edu/ml.

For the comparison, we have selected a recently proposed sequential learning method, CINFO, designed for outlier detection in high-dimensional data [11]. CINFO works based on lasso-based sparse regression modeling to iteratively refine the feature space. As their method is general, we select the IForest-based implementation which considers the scores generated by IForest algorithm as the dependent variable of the regression model. Due to the randomness feature of iTree generation, each experiment is repeated for minimum of 5 times, and the average of results is reported. For CINFO, the number of repeated experiments is based on their recommended values to have stable results [11].

7.5.2 Experiment Results

7.5.2.1 ITL with Bagging of the Scores

Table 7.2 shows the AUC results for the base IForest algorithm as well as both ITL and CINFO learning methods. The best results are highlighted in bold. As the results show, the ITL process improves the performance of IForest by combining the scores from various subsets of the feature space. The best AUC results are achieved for *AD* dataset for which the results of ITL show a dramatic improvement (around 22%) compared to the base method. This is a result of the higher ratio of noisy features in this dataset. In comparison to CINFO method, the same or better performance is observed for 4 of the 5 datasets. The only exception is *Secom* where ITL shows improvements compared to the base, but not as much as the CINFO. This could be attributed to the greedy removal of features in vertical partitioning of ITL as we explained in Sect. 7.4.1. Since the results for *DOS* and *Seizure* are very high, even with the base IForest (higher than 95%), we do not expect to see too much improvement. However, ITL still shows comparable or improved AUC while achieving a reduction of about 8 and 43% in the size of the feature set. In general, ITL shows improved results as well as a reduction of the features between 9 and 97% compared to the original set. These results are especially important when the quality of reduced features is investigated for the detection of unseen anomalies.

Table 7.2 AUC results for the base IForest, ITL, and CINFO. M and M' show the size of the original and reduced features for ITL. The best AUC for each dataset is highlighted in bold

				ITL Feature Reduction		
	IForest	CINFO	ITL	M	M'	Reduction
DOS	**0.981**	0.971	**0.981**	37	21	43%
U2R	0.874	0.894	**0.901**	37	18	51%
SECOM	0.551	**0.655**	0.594	590	80	86%
AD	0.704	0.850	**0.856**	1558	54	97%
Seizure	0.989	0.987	**0.990**	178	163	8%

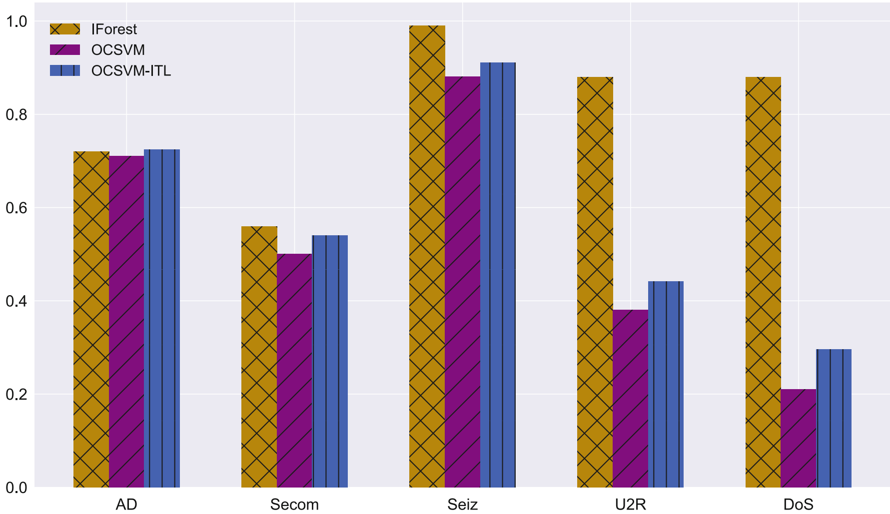

Fig. 7.3 AUC comparison for IForest and OCSVM. OCSVM results are shown when applied on input data with all features and with ITL reduced set of the features. The results are average AUC over cross-validation folds

Therefore, in the following, we further study the effectiveness of the reduced subset of features produced by ITL in anomaly detection results.

7.5.2.2 ITL with Reduced Features

To validate the efficacy of the reduced subset of features on the detection capability of IForest algorithm, a series of experiments are conducted based on the k-fold cross-validation. Figure 7.3 compares the performance of isolation-based technique (IForest) and a kernel-based anomaly detection method (OCSVM). OCSVM is a novelty detection method that can also be used in unsupervised anomaly detection by selecting soft boundaries. The figure also shows the results of OCSVM when applied on ITL reduced set of features. As the results show while applying OCSVM with ITL approach can improve the results of OCSVM, the isolation-based technique (IForest) shows higher performance. Therefore, in the next experiments, the performance of ITL on IForest is studied. The five-fold validation is used to train the IForest model on 4 parts of the data when all features are included in comparison to the data with the reduced features from the ITL process and AUC values of validation parts are reported. Figure 7.4 demonstrates ITL results for different numbers of trees from 1 to 100. As we can see, reduced features can achieve or improve AUC value compared to the full set of the features for a range of number of trees in all datasets. The interesting observation is that the reduction in the number of trees has less impact on the performance, especially for the reduced set as shown in Fig. 7.4. For example, even with 10 trees, the results are very close to the

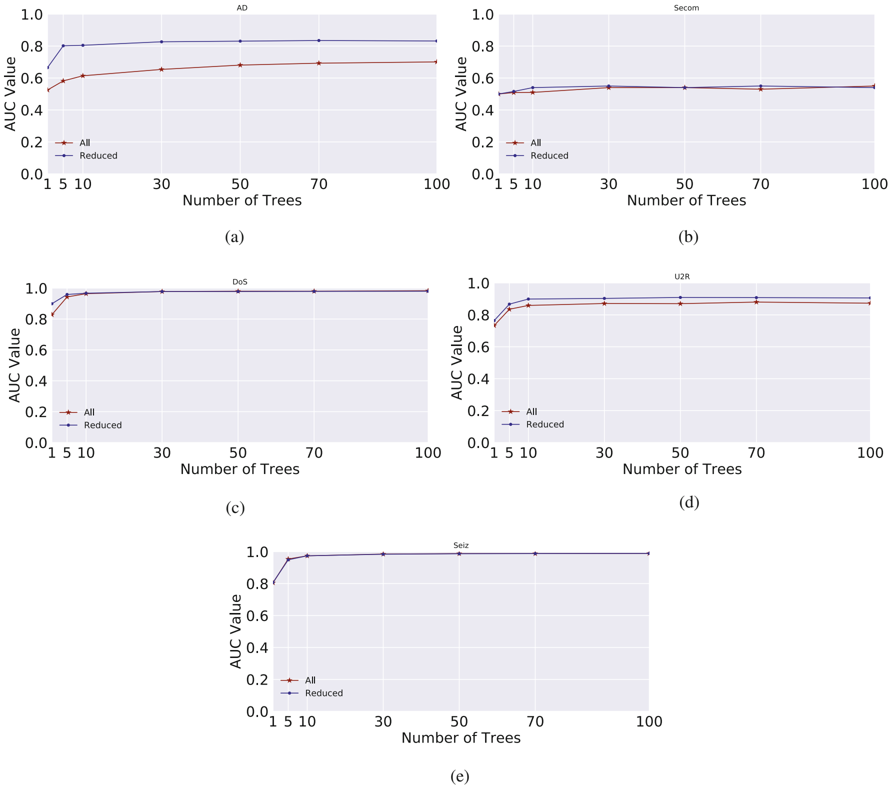

Fig. 7.4 AUC comparison for IForest when applied on input data with all features and with ITL reduced set of the features. The results are average AUC over cross-validation folds. (**a**) AD. (**b**) Secom. (**c**) DoS. (**d**) U2R. (**e**) Seiz

performance of the algorithm with default parameters (100 trees). This improvement can be attributed to having fewer features to be explored during the random selection of the features. In other words, having a subset of the features learned through ITL process, one can achieve improved results with less number of trees. The reduction of features, as well as the number of trees, can help to reduce the complexity in terms of the memory and runtime requirements. Figure 7.5 shows the running time taken for a variety of tree numbers. As we can see, the number of trees can hugely impact the testing time. This is highly important for dynamic environments such as the cloud where the testing should be performed regularly. These results indicate ITL approach as a potential choice to be employed by real-time applications where the new incoming stream of data requires quick online tests for identifying possible problems.

During the ITL learning phase, the number of iTrees in each ensemble is a parameter that should be decided for each iteration. In order to have a better

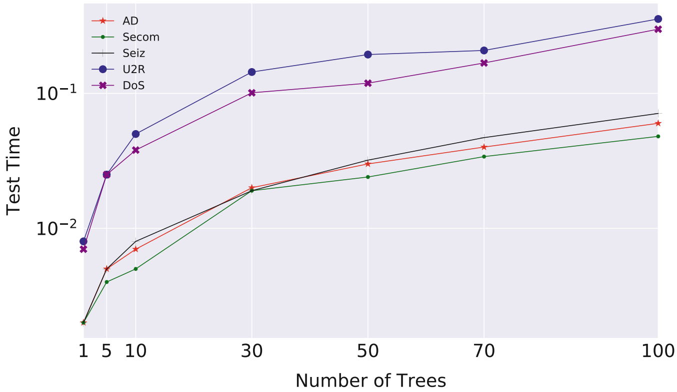

Fig. 7.5 Runtime for the Testing of cross-validated results on the reduced features. Logarithmic scale is used on y axis

understanding of the sensitivity of ITL to this parameter, we run ITL several times for a range of values for the number of trees. Figure 7.6 shows the AUC distribution of each set of the experiments for all datasets. As the results show, ITL is sensitive to this parameter. However, AUC values show improvements with the increased number of trees and are stable for numbers larger than 600. Practically, we found that a value between 600 and 900 trees is sufficient in most cases to have a good trade-off between accuracy and training complexities in terms of memory and runtime.

7.5.2.3 Time Complexity and Runtime Analysis

Algorithm 1 presents the main steps of the ITL process. The main while loop (Line 2) continues until the termination condition of having zero unseen attributes is met. The termination condition is guaranteed to converge as during the vertical partition phase, features with zero-seen or low frequency are removed which reduces the feature space. As a result, the remaining features have more chance to be explored with regard to anomalous instances (and possibly showing in short branches of the tree which increases their potential to be included in high-frequency profile). Since we always have potential anomalies seen in short branches, there is at least one feature with a frequency higher than one which will create the final reduced subspace. Therefore, we finally get to a level where all features are seen at least one time or the reduced subspace has just one feature left. The loop typically converges in less than 5 iterations. Lines 3–11 build IForest models and filter high-rank instances based on the predefined threshold. Considering the IForest trees as the base structure for these steps, it takes $O(t\psi log(\psi))$ for constructing where ψ is the number of selected subsamples and t is the number of constructed trees. If there are N testing points, it requires $O(Ntlog\psi)$ for determining anomalous points and

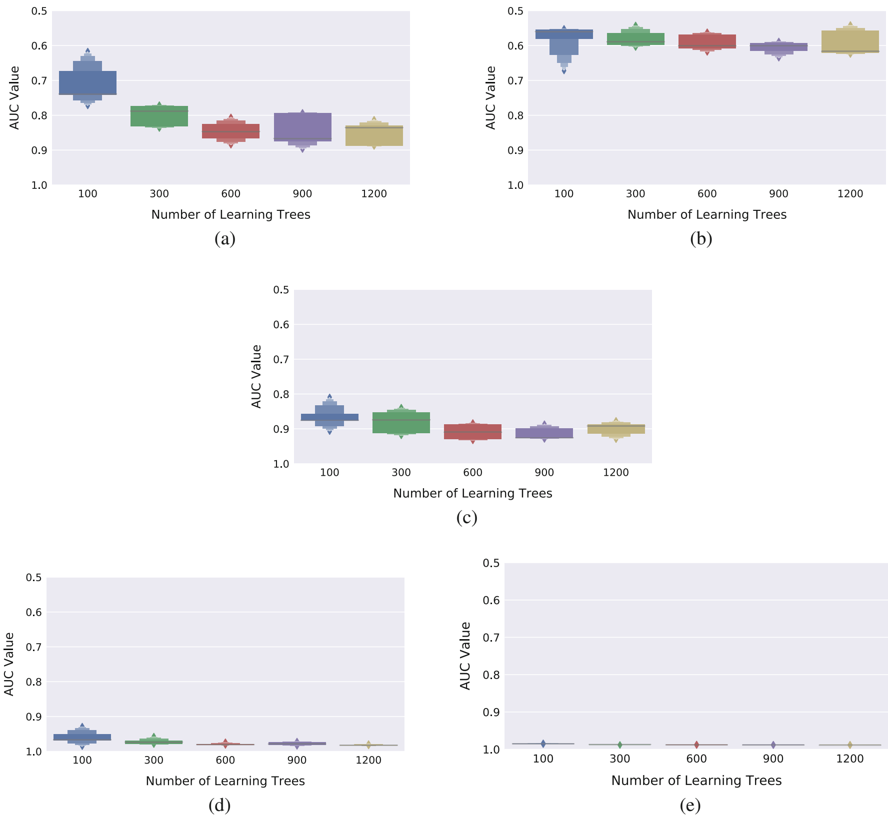

Fig. 7.6 AUC value distribution for ITL reduced features in training. This plot shows the sensitivity of ITL process to different numbers of the learning trees. (**a**) AD. (**b**) Secom. (**c**) U2R. (**d**) DoS. (**e**) Seiz

$O(Ktlog\psi)$ for updating frequency profile of the features, where $K << N$ (Line 5) is filtered anomalies (worst-case complexity is order $O(t\psi(\psi + N))$). Therefore, we expect a linear complexity with regard to data size.

IForest is shown to have a very fast and memory-efficient runtime for both modeling and testing purposes. In order to have a clear understanding of the ITL contribution to make this process even faster, a series of execution times with respect to the number of learning trees are presented. Figure 7.7 shows the learning time in ITL, where the main feature refinements are done by constructing iTrees and creating a new subset of features. The diagram shows the learning time for a variety of tree numbers. As it is mentioned before, 600–900 usually is enough to have a sufficient exploration of feature space for target datasets. When the learning phase of ITL is completed, the anomaly detection is done by modeling iTrees with extracted features. To have a better comparison of execution times,

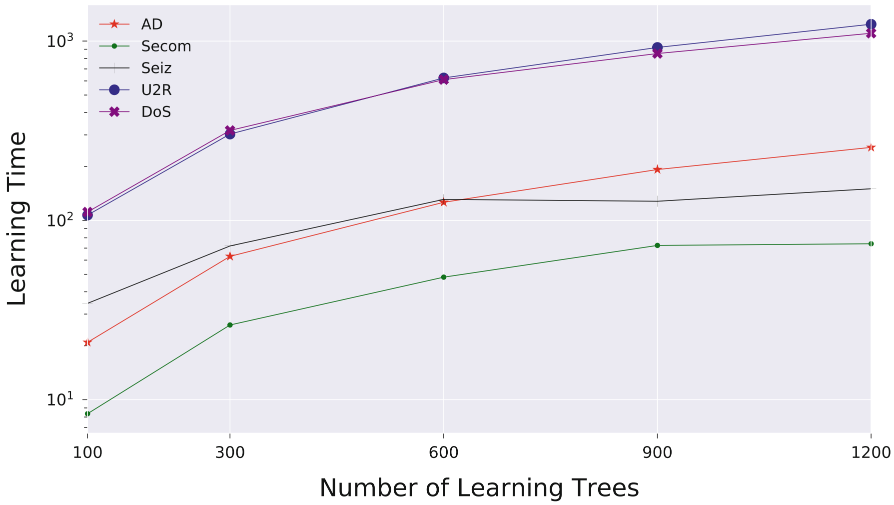

Fig. 7.7 Total runtime of learning phase of ITL. Logarithmic scale is used on y axis

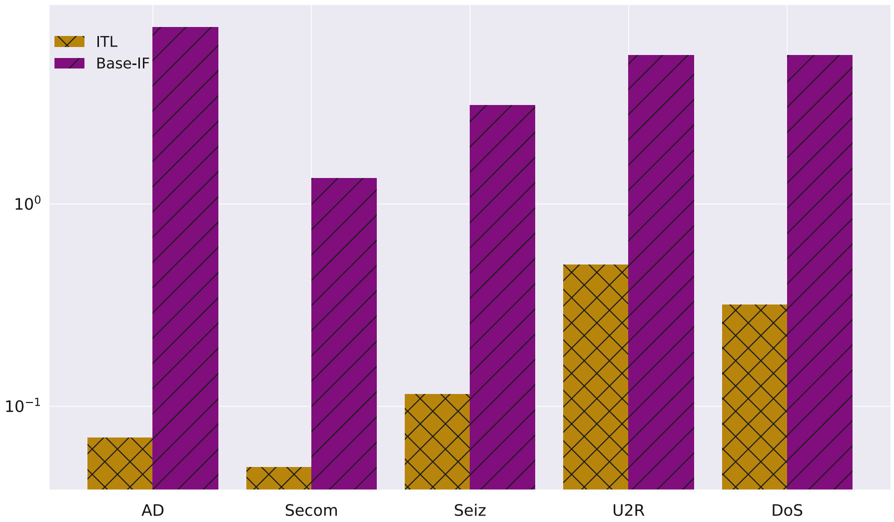

Fig. 7.8 Comparison of modelling times for ITL-produced features with reduced number of iTrees (yellow) and base IForest algorithm (Purple) with default parameters. Logarithmic scale is used on y axis

Fig. 7.8 compares modeling time of ITL-learned features with the reduced number of trees with the base IForest without feature refinements and with the recommended number of trees in the literature. As we can see, ITL process makes a dramatic decrease in modeling times by helping to decrease the number of features/trees which makes the construction of iTrees and training step much faster. It should be highlighted that this reduction is achieved by keeping or improving the detection accuracy as it is shown in Fig. 7.4. However, the feature refinement process of ITL as shown in Fig. 7.7 is the cost of achieving these results. But the learning phase is a one-time process which is performed off-line, and the final subset is used for subsequent anomaly detection task which is significantly improved in terms of both modeling and testing times as shown in Figs. 7.5 and 7.8, respectively. Considering the context of one application, the learning phase can be done with a low frequency and as a background process. Therefore, systems that require regular updating of their performance models can highly benefit from time/memory reductions of this process.

In conclusion, ITL shows that by targeting the main contributing features which isolate the instances in iTrees, we can reach a refined set of the features that can be used by fewer trees to create a model with better results.

7.5.3 *Strength and Limitations of ITL Approach*

IForest algorithm, as described in Sect. 7.3, is designed to detect anomalous objects by the ensemble of binary trees from input data. ITL tries to take the advantage of this mechanism to extract information about relevant features that better isolate instances. Since the core of the ITL is iTree data structures from IForest, the same advantages of random-based sampling and feature selection are equally applicable to ITL. Moreover, it can be used as a preprocessing step to learn a reduced set of features for any other anomaly detection algorithm. ITL is a promising method for real-time applications as high detection accuracy can be achieved with small memory and time complexity, and it can perform well without prior knowledge of the specific distribution. We have tested the applicability of our proposed techniques on a variety of datasets with different characteristics and application domains. The benefit of our method is that it can be trained rapidly as trees are a very fast construct. Finding optimal features can be computed in the background without sacrificing response time for anomaly detection. Moreover, ITL is an unsupervised method and does not require training data containing anomaly annotations.

Similarly, ITL inherits the same drawbacks as the base algorithm in detecting local clustered anomalies [8]. This can affect the filtering of instances when the assumption is made that there are a majority of anomaly instances at the top of the ranking list. Adaptive, data-dependent configuration for parameters such as the maximum height of trees or customized split point selection for node constructions may help to reduce this effect but requires more preprocessing and knowledge on statistical characteristics of anomalous data.

7.6 Conclusions and Future Work

In this part, we introduced an iterative learning framework (ITL) for the refinement of features and improvements of the anomaly detection process. Advances in monitoring and storage capabilities provide a high volume of information on the performance of applications and systems to be used for anomaly and fault analysis. This requires real-time analysis of data to quickly identify problems and take appropriate corrective actions. However, high-dimensional data can adversely affect the traditional measures of anomaly detection such as distance between instances in terms of efficacy and time complexity. More recent approaches such as the isolation-based technique try to directly target the main features of anomalies as being different and rare. ITL is designed based on the idea that isolation-based generated trees can give some insights on the importance of the features. Therefore, the learning phase of ITL is based on the knowledge from iTree structures which are binary trees constructed by random selection of the features from domain problems. The assumption is made that the features on the short branches of iTree can be used as a reference to identify relevant features to the detection of anomaly instances. The learning is based on the iterative removal of the noisy and irrelevant features in terms of their importance for isolating anomalies to generate a final subset of the features to be used for anomaly detection. The experiments show that the anomaly scores from IForest algorithm on generated subsets of the data at each iteration can be combined to create a more informative set of the scores in terms of the detection capability of anomaly instances. Moreover, the experiments on five benchmark datasets demonstrate that with the reduced set of the features and choosing a proper number of trees IForest can achieve better results in terms of the detection accuracy while reducing the complexity of the algorithm.

For future work, we plan to enhance ITL framework to identify groups of anomalous metrics that isolate individual faults. The isolation can be achieved for environments that different groups of features are impacted by different types of faults. This helps to distinguish among different anomalies such as various types of attacks. We also would like to extend the ITL idea to more flexible tree structures (for example, trees with more than two branches) to investigate the possibility of further improvements for clustered anomalies. We also highlight that anomaly detection, in general context, considers any significant deviation in the values of the attributes from the past data (training part) as an anomaly which will be reflected in anomaly scores. However, with regard to the required actions after detecting anomalies, some level of knowledge expert may be required. For example, after detecting abnormality in packet-level information that can be a sign of the attacks (as shown by datasets of U2R and DoS in the experiments), application owners may prefer to shut down targeted resources (VM or physical machine) to reduce the cost of wasted resources. The integration of anomaly detection part and resource management modules may bring new challenges in the design of scaling solutions that require further investigations.

References

1. V. Chandola, A. Banerjee, V. Kumar, Anomaly detection: a survey. ACM Comput. Surv. **41**(3), 15 (2009)
2. J. Xiao, Z. Xiong, S. Wu, Y. Yi, H. Jin, K. Hu, Disk failure prediction in data centers via online learning, in *Proceedings of the 47th International Conference on Parallel Processing, ICPP 2018* (ACM, New York, 2018), pp. 35:1–35:10
3. C.C. Aggarwal, P.S. Yu, Outlier detection for high dimensional data, in *Proceedings of the 2001 ACM SIGMOD International Conference on Management of Data* (ACM, New York, 2001), pp. 37–46
4. M.H. Bhuyan, D. Bhattacharyya, J. Kalita, A multi-step outlier-based anomaly detection approach to network-wide traffic. Inf. Sci. **348**, 243–271 (2016)
5. A. Zimek, E. Schubert, H.-P. Kriegel, A survey on unsupervised outlier detection in high-dimensional numerical data. Stat. Anal. Data Min. ASA Data Sci. J. **5**(5), 363–387 (2012)
6. H. Liu, X. Li, J. Li, S. Zhang, Efficient outlier detection for high-dimensional data. IEEE Trans. Syst. Man Cybern. Syst. **48**, 2451–2461 (2017)
7. F.T. Liu, K.M. Ting, Z.-H. Zhou, Isolation-based anomaly detection. ACM Trans. Knowl. Discover. Data **6**, 3:1–3:39 (2012)
8. F.-T. Liu, K.-M. Ting, Z.-H. Zhou, On detecting clustered anomalies using sciforest, in *Proceedings of the 2010 European Conference on Machine Learning and Knowledge Discovery in Databases: Part II* (2010), pp. 274–290
9. F. Keller, E. Muller, K. Bohm, HiCS: high contrast subspaces for density-based outlier ranking, in *Proceedings of the 2012 IEEE 28th International Conference on Data Engineering, ICDE'12* (IEEE Computer Society, Washington, 2012), pp. 1037–1048
10. H. Shi, H. Li, D. Zhang, C. Cheng, X. Cao, An efficient feature generation approach based on deep learning and feature selection techniques for traffic classification. Comput. Netw. **132**, 81–98 (2018)
11. G. Pang, L. Cao, L. Chen, D. Lian, H. Liu, Sparse modeling-based sequential ensemble learning for effective outlier detection in high-dimensional numeric data, in *Thirty-Second AAAI Conference on Artificial Intelligence* (2018), pp. 3892–3899
12. S. Agrawal, J. Agrawal, Survey on anomaly detection using data mining techniques. Proc. Comput. Sci. **60**, 708–713 (2015)
13. C. Pascoal, M.R. de Oliveira, R. Valadas, P. Filzmoser, P. Salvador, A. Pacheco, Robust feature selection and robust PCA for internet traffic anomaly detection, in *2012 Proceedings IEEE INFOCOM* (2012), pp. 1755–1763
14. F. Angiulli, C. Pizzuti, Fast outlier detection in high dimensional spaces, in *Proceedings of the 6th European Conference on Principles of Data Mining and Knowledge Discovery* (Springer, London, 2002), pp. 15–26
15. S. Ramaswamy, R. Rastogi, K. Shim, Efficient algorithms for mining outliers from large data sets, in *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data* (2000), pp. 427–438
16. M. Ester, H.-P. Kriegel, J. Sander, X. Xu, A density-based algorithm for discovering clusters a density-based algorithm for discovering clusters in large spatial databases with noise, in *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining* (AAAI Press, 1996), pp. 226–231
17. Y. Zhu, K.M. Ting, M.J. Carman, Density-ratio based clustering for discovering clusters with varying densities. Pattern Recogn. **60**, 983–997 (2016)
18. J. Chen, S. Sathe, C. Aggarwal, D. Turaga, Outlier detection with autoencoder ensembles, in *Proceedings of the 2017 SIAM International Conference on Data Mining* (SIAM, Philadelphia, 2017), pp. 90–98
19. Q. Guan, S. Fu, Adaptive anomaly identification by exploring metric subspace in cloud computing infrastructures, in *Proceedings of the 32nd IEEE International Symposium on Reliable Distributed Systems, SRDS'13* (IEEE Computer Society, Braga, 2013), pp. 205–214

20. D.J. Dean, H. Nguyen, X. Gu, UBL: unsupervised behavior learning for predicting performance anomalies in virtualized cloud systems, in *Proceedings of the 9th International Conference on Autonomic Computing* (ACM, New York, 2012), pp. 191–200
21. A.B. Ashfaq, S. Rizvi, M. Javed, S.A. Khayam, M.Q. Ali, E. Al-Shaer, Information theoretic feature space slicing for statistical anomaly detection. J. Netw. Comput. Appl. **41**, 473–487 (2014)
22. J. Cao, B. Yu, F. Dong, X. Zhu, S. Xu, Entropy-based denial of service attack detection in cloud data center, in *Proceedings of the Second International Conference on Advanced Cloud and Big Data* (2014), pp. 201–207
23. J. Li, K. Cheng, S. Wang, F. Morstatter, R.P. Trevino, J. Tang, H. Liu, Feature selection: a data perspective. ACM Comput. Surv. **50**(6), 94:1–94:45 (2017). http://doi.acm.org/10.1145/3136625
24. D. Fesehaye, L. Singaraveluy, C. Chen, X. Huang, A. Banerjee, R. Zhou, R. Somasundaran, Group clustering using inter-group dissimilarities, in *Proceedings of the 2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)* (2017), pp. 1011–1021
25. P. Filzmoser, R. Maronna, M. Werner, Outlier identification in high dimensions. Comput. Stat. Data Anal. **52**(3), 1694–1711 (2008)
26. T. de Vries, S. Chawla, M.E. Houle, Finding local anomalies in very high dimensional space, in *Proceedings of the 2010 IEEE International Conference on Data Mining* (2010), pp. 128–137
27. A. Lazarevic, V. Kumar, Feature bagging for outlier detection, in *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining* (ACM, New York, 2005), pp. 157–166
28. S. Jin, Z. Zhang, K. Chakrabarty, X. Gu, Toward predictive fault tolerance in a core-router system: anomaly detection using correlation-based time-series analysis. IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. **37**(10), 2111–2124 (2018)
29. F.T. Liu, K.M. Ting, Z.-H. Zhou, Isolation forest, in *Proceedings of the 8th IEEE International Conference on Data Mining* (IEEE, Piscataway, 2008), pp. 413–422
30. N. Moustafa, J. Hu, J. Slay, A holistic review of network anomaly detection systems: a comprehensive survey. J. Netw. Comput. Appl. **128**, 33–55 (2019)
31. D. Dua, C. Graff, UCI machine learning repository (2017). http://archive.ics.uci.edu/ml
32. F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, Scikit-learn: machine learning in Python. J. Mach. Learn. Res. **12**, 2825–2830 (2011)