Ensemble of Local Decision Trees for Anomaly Detection in Mixed Data

Sunil Aryal[0000−0002−6639−6824] and Jonathan R. Wells[0000−0003−0550−1229]

School of Information Technology
Deakin University
Geelong, Victoria, Australia
{sunil.aryal,j.wells}@deakin.edu.au

Abstract. Anomaly Detection (AD) is used in many real-world applications such as cybersecurity, banking, and national intelligence. Though many AD algorithms have been proposed in the literature, their effectiveness in practical real-world problems are rather limited. It is mainly because most of them: (i) examine anomalies globally w.r.t. the entire data, but some anomalies exhibit suspicious characteristics w.r.t. their local neighbourhood (local context) only and they appear to be normal in the global context; and (ii) assume that data features are all numeric, but real-world data have numeric/quantitative and categorical/qualitative features. In this paper, we propose a simple robust solution to address the above-mentioned issues. The main idea is to partition the data space and build local models in different regions rather than building a global model for the entire data space. To cover sufficient local context around a test data instance, multiple local models from different partitions (an ensemble of local models) are used. We used classical decision trees that can handle numeric and categorical features well as local models. Our results show that an Ensemble of Local Decision Trees (ELDT) produces better and more consistent detection accuracies compared to popular state-of-the-art AD methods, particularly in datasets with mixed types of features.

Keywords: Anomaly Detection · Mixed Data · LOF · IForest · Ensemble Anomaly Detection · Decision Trees.

1 Introduction

Anomaly Detection (AD) is a machine learning task of identifying anomalous data instances automatically using algorithms. Anomalies (also refer to as outliers) are data instances that are significantly different from most of the other data causing suspicions that they are generated from a different mechanism from the one that is normal or expected. AD has many applications such as intrusion detection in computer networks, fraud detection in banking, detecting illegal activities (e.g., drug trafficking, money laundering) in national intelligence/security. In the literature, AD problems have been solved using three learning approaches [8, 12]: (i) **Supervised learning:** A classification model is learned using training instances from both normal and anomalous classes to make predictions for test data; (ii) **Unsupervised learning:** Given data instances (which may have anomalies) are ranked directly based on some outlier scores, i.e., no training involved; and (iii) **Semi-supervised learning:** A profile of normal/expected behaviour is learned from labelled training samples of normal data only, and test data are ranked based on how well they comply with the learned profile of normal data.

Regardless of the learning approaches used, existing AD methods have some limitations/issues that restrict their wide applicability in practice. Supervised methods have the following major issues [12]: (i) it might be very expensive or even impossible to obtain labelled training anomalous samples in many real-world applications; (ii) even if possible, they are infinitesimally rare resulting in the class imbalanced problem; and (iii) a few known anomalies are not enough to generalise characteristics of all possible anomalous patterns because anomalies can be anywhere in the feature space. Though techniques like minority class (anomalies) oversampling, majority class (normal) under-sampling, and algorithmic adjustments [18] are used to alleviate the above-mentioned issues, their effectiveness in practice are limited. It is because they assume that unseen/future anomalies are generated from the same distribution as previously seen/observed anomalies. Often, it is not the case in practice. New anomalies can be very different from previously seen anomalies.

Un/semi-supervised approaches do not require labelled training anomalous samples. Unsupervised approaches do not require training samples at all. Assuming anomalies are few and different, they use distance/density based scores to rank given data (which may have anomalies) directly. They may perform poorly when the assumption does not hold, i.e., when there are far too many anomalies [8, 12]. Semi-supervised approaches do not make such assumption. Because a vast majority of observed data are normal, normal training data can be obtained easily. Thus, we focus on semi-supervised AD approach in this paper.

Most existing un/semi-supervised AD methods assume that data have numeric features. However, in many real-world applications, data have both numeric (e.g., age, height) and categorical (e.g., gender, nationality) features. The common practice is to convert categorical features into numeric features using technique like one-hot encoding [15]. Each categorical label (e.g., Australian for nationality) is converted into a binary feature with the value of 1 (if the nationality is Australian) or 0 (otherwise) and treated as a numeric feature. A categorical feature with n possible values is converted into n binary numeric features, out of which only one has the value of 1 for each instance. Because of this, each original categorical feature and original numeric feature contribute differently to AD models, which can degrade the performances of AD methods. There are methods proposed for categorical data only [26]. Numeric features can be converted into categorical features through discretisation [15]. Most AD methods developed for categorical data have high computational complexities limiting their use in large real-world datasets.

Most existing AD methods examine data instances globally, i.e., w.r.t. the entire dataset. They can detect global anomalies that exhibit anomalous characteristics in the entire dataset. However, they cannot detect local anomalies that appear to be normal when examined globally but exhibit anomalous characteristics w.r.t. their local neighbourhood (i.e., in the local context). For example, in Figure 1(a), a_4 and a_5 have significantly lower density than normal cluster C_1 in their neighbourhood but have the same density as many instances in normal cluster C_2. Most existing methods fail to detect them as anomalies. Real-world data have complex structures and instances may exhibit characteristics that look normal in the global perspective but suspicious in their local contexts. There are some methods that examine anomalies w.r.t. their localities (e.g., [11, 5]), but they are limited to numeric data only.

To summarise, most existing AD methods do not work well in practical applications due to the following three main issues:

- **Lack of sufficient examples of known anomalies**: It is not possible to have a good representative sample of known anomalies to generalise characteristics of all possible anomalies.
- **Global view of anomalies**: Data often exhibit suspicious characteristics w.r.t. their neighbourhood (in local context) that can appear to be normal in the global context.
- **Limitations to handle mixed types of data features**: Most real-world applications have numeric and categorical features, but most existing methods can not handle mixed types of features well.

In this paper, we present a simple idea to address the above-mentioned issues and introduce a new semi-supervised AD method. Instead of using one global model, we propose to partition the data space into many regions and build an AD model in each region using data falling in the region only, i.e., many local AD models are built. To make prediction for a test instance, the AD model learned on the region where it falls is used. Instead of just relying on a local region from one partitioning of the space, we propose to create multiple partitions of the data space and use ensemble of multiple local models learned on local regions from each partition. It exploits the benefits of ensemble learning to consider sufficient locality around the test instance. Though there are not many AD models that can work well with mixed data, there are classifiers such as traditional Decision Tree (DT) [22] that can handle numeric and categorical features directly. We used DTs in local regions for AD without using labelled anomalies by adding synthetic data. Our results show that an Ensemble of Local Decision Trees (ELDT) produces better and more consistent detection results compared to popular state-of-the-art AD methods, particularly in datasets with mixed types of features.

2 Related work

In the semi-supervised approach, 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, which is a mixture of normal and anomalous data. Let $\mathbf{x}$ be a data instance represented as an M-dimensional vector $\langle x_1, x_2, \cdots, x_M \rangle$, where each component represents its value of a feature that can be either numeric $x_i \in \mathbb{R}$ ($\mathbb{R}$ is a real domain) or categorical $x_i \in \{v_{i_1}, v_{i_2}, \cdots, v_{i_w}\}$ (where v_{i_j} is a label out of w possible labels for feature i). Let $F = \{A_1, A_2, \cdots, A_M\}$ be a set of data features, also called as attributes of data.

In this section, we review prior work related to this paper that includes AD methods for numeric and categorical data, and ensemble approaches for AD.

2.1 Methods for numeric data

Because anomalies are few and different, they are expected to have feature values that are significantly different from most data and lie in low density regions. Most of them use distance/density-based anomaly scores to rank data according to their degrees of outlying behaviour, e.g., Nearest Neighbours (NNs) or Support Vectors (SVs) based methods.

In the NN-based methods, the anomaly score of $\mathbf{x} \in Q$ is estimated based on the distances to its kNNs in D, where k is a user defined neighbourhood parameter. Local Outlier Factor (LOF) [11] and k^{th} NN distance [6] are the most widely used NNs-based methods. Being different from normal instances, anomalies are expected to have larger distances to their kNNs than normal instances. They require to compute distances of $\mathbf{x}$ with all instances in D, which can be computationally expensive when D is large. Though the nearest neighbour search can be speed up by using indexing schemes such as k-d tree [7], their effectiveness reduces as the number of dimension increases and become useless in high dimensional problems [19]. Sugiyama and Borgwardt (2013) [25] showed that the nearest neighbour search in a small subset $\mathcal{D} \subset D$ ($|\mathcal{D}| = \psi \ll N$) is enough. They proposed a simple, but very fast, anomaly detector called Sp where the anomaly score of $\mathbf{x}$ is its distance 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 [25].

The SV-based methods define the boundary around normal (expected) data and identify a set of data instances lying in the boundary called Support Vectors (SVs). They compute the pairwise similarities of data using a kernel function. Gaussian kernel that uses Euclidean distance is a popular choice. In the testing phase, the anomaly score of $\mathbf{x} \in Q$ is estimated based on its kernel similarities with the SVs. One-Class Support Vector Machine (OCSVM) [23] and Support Vector Data Description (SVDD) [27] 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.

2.2 Methods for categorical data

Despite the widespread prevalence of categorical/qualitative data in real-world applications, AD in categorical data has not received much attention in the research community [26]. The common practice is to convert categorical features into numeric features and use methods designed for numeric data. There are some methods proposed in the literature specifically for categorical data based on frequencies of categorical labels, information theory and data compression/encoding [26]. They are computationally expensive to run in large datasets and do not perform better than using methods for numeric data by converting categorical data into numeric data [4].

He et al. (2005) [17] proposed a method for categorical data based on frequent patterns. The intuition is that an instance is more likely to be an anomaly if it has a few or none of the frequent patterns. Akoglu et al. (2012) [2] proposed a pattern-based compression technique called COMPREX. The intuition is that the higher the cost of encoding $\mathbf{x}$, the more likely it is to be an anomaly. Aryal et al. (2016) [4] revisited the Simple Probabilistic AD (SPAD) where multi-dimensional probability is estimated as the product of one-dimensional probability and show that it works quite well compared to more complex state-of-the-art methods, such as LOF, One-Class SVM, in datasets with categorical only and mixed types of features. It uses the frequencies of categorical label in each feature individually assuming features are independent to each other. Most of these methods for categorical datasets except SPAD have high time and/or space complexities limiting their use in small and low-dimensional datasets only. SPAD is simple and arguably the fastest AD method.

2.3 Ensemble approaches

To solve a given task, the ensemble methods build multiple models by using an algorithm on different subsets of given data (data sampling or feature sampling) or using different parameter settings of the algorithm [13]. The final decision of the ensemble is an aggregation of decisions by its individual models. The main idea is that models are different and they make different errors so that they compensate each other's weaknesses and results in better overall performance than any individual model. Ensemble learning is widely studied for classification problems and various frameworks have been proposed that can be used with different base classifiers [9, 10, 28]. However, the use of ensemble learning to solve the AD problem is rather limited [1]. Ensemble based AD methods build multiple models using subsamples of data and/or subsets of features, e.g., Lazarevic and Kumar (2005) [20] and Zimek et al. (2013) [29] used LOF using random subsets of features (i.e., subspaces) and data (i.e., subsamples), respectively. AD techniques such as iForest [21] and usfAD [3] used a collection of random trees to partition the data space using small subsamples of data until the instances are isolated. Each tree is using a small subset of features. The main idea of these methods is that anomalies are expected to isolate early in the trees and lie in leaves with low heights. They run very fast as they do not require pairwise distance calculations. All these ensemble-based methods assume that data have numeric features only. Most of them are not applicable to data with categorical only or mixed features. Also, many of them build multiple global models, they can not detect local anomalies.

3 Our proposal: An ensemble of Local Decision Trees

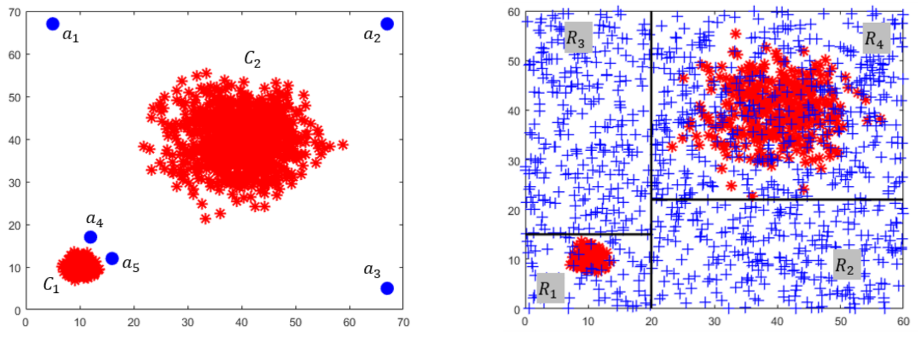
To address the three limitations of existing AD methods in practical real-world applications discussed in Section 1, we develop a new ensemble learning framework for anomaly detection based on the of idea of Feating [28]. First, we explain Feating for classification as used by Ting et al. (2011) [28] (Subsection 3.1) and then discuss how we can adapt it for anomaly detection (Subsecion 3.2).

3.1 Feating for classification

Feature-Subspace Aggregat**ing** (Feating) [28] is an ensemble framework developed for classification that uses an ensemble of local models. It is a feature bagging approach, ensemble learning using subsets of features of fixed size $m < M$ (i.e., using m-dimensional subspaces). In each subspace $S \subset F$ with $|S| = m$, rather than building a global model trained on the entire training set, it first partitions the subspace using a tree structure called "`Level Tree`" (LT). At each node of the tree, the space is partitioned using one of the m features in S. Each feature in S is used only once in the tree, resulting in the maximum tree height of m. LTs can handle both numeric and categorical features. For numeric feature, the space is divided into two regions by the cut-point selected in the same manner as in ordinary decision tree [22] based on information gain. For categorical feature with w possible values, the space is partition into w regions, one for each categorical label. Further partitioning of a node stops when the node is either pure (i.e., has instances belonging to the same class), there are less than $minPts$ data instances or reaches the maximum height of m. In each impure leaf node with more than $minPts$ samples, a classifier is learned from the training samples falling in the node only, i.e., a Local Model (LM) is built. For rest of the other leaves (with less than $minPts$ instances or pure), class probabilities are recorded based on the training samples they have. In the testing phase, a test instance is traversed from the root to a leaf in each LT. If a LM was built in the leaf node, the class probabilities are the predicted probabilities of the LM. Otherwise, the recorded class probabilities are used. The final prediction is based on the aggregated class probabilities from multiple LTs. The enumerated version of Feating builds $\binom{M}{m}$ LTs, which has a large space complexity making it infeasible in problems with large M (high-dimensional applications). To overcome this issue, Ting et al. (2011) introduced a randomised version, where only $t \ll \binom{M}{m}$ random subspaces of size m are used. It significantly improves the time and space requirements without any significant compromise in accuracy.

3.2 Feating for anomaly detection

We propose the following adjustments to use Feating in semi-supervised AD, where there are no labelled anomalies. LT building process uses class information but in this case we do not have labelled anomalies. We are given D which is a set of normal data only. To build each LT, we propose to consider the given data D as "`+ve`" class and the same number of synthetic points are added as "`-ve`" class



(a) Example dataset (b) Local Regions

Fig. 1: An example of dataset and definition of local regions. (a) C_1 and C_2 are clusters of normal data, whereas $a_1, \cdots, a_5$ are anomalies. (b) Note only half of the normal data (red points) are used in the training process as "+ve" class samples. Blue points, which are uniformly generated synthetic points, are "-ve" class samples. R_1, R_2, R_3, and R_4 are local regions created by a Level Tree. Note that local classifiers are built in regions R_1 and R_4 only.

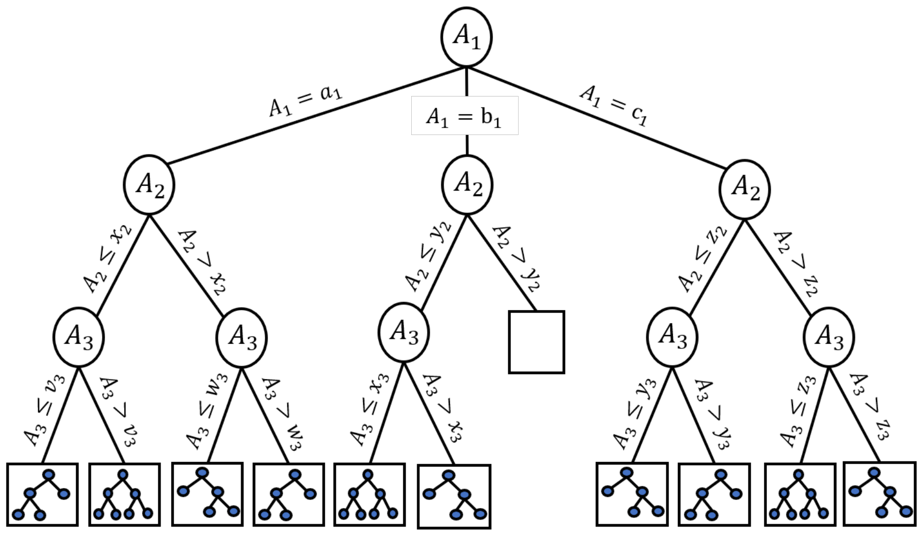


Fig. 2: An example of a Level Tree in 3-dimensional subspace $S = \{A_1, A_2, A_3\}$. Note that A_1 is a categorical feature with three possible values (a_1, b_1, c_1), and A_2 and A_3 are numeric features.

as done by Shi and Horvath (2006) [24] to use Random Forest in unsupervised problems. The values of synthetic points in each feature are selected uniformly at random from the range of possible values, i.e., for a numeric feature, values are selected uniformly randomly between the possible range defined by instances in

D, and for a categorical feature, values are selected randomly from the possible values. With the samples from "`+ve`" and "`-ve`" classes, a LT can be built exactly in the same way as for the classification task. As example of space partition into local regions in shown in Figure 1(b). Note that we add new "`-ve`" class samples for each LT. It is possible to have two LTs using the same subset of m features if $t > \binom{M}{m}$, but the two LTs will be different because of the new set of "`-ve`" class samples.

Algorithm 1: Feating(D, A, m, t) - Build a set of Level Trees

```
Input  : $D$ - Given data, $A$ - a set of given features, $m$ - the maximum level
         of the level tree. $t$ - number of trees to build
Output: $E$ - a collection of Level Trees
1  $N = |D|$ (#training samples);
2  $M \leftarrow |A|$ (#features);
3  $minPts = \lfloor \log_2(N) \rfloor + 1$;
4  $m = \lfloor \log_2(M) \rfloor + 1$;
5  $E \leftarrow \emptyset$;
6  for $i = 1$ to $t$ do
7      // get a set of $m$ attributes from $A$
8      $L \leftarrow randomSetOfAttributes(A, m)$;
9      $D_s \leftarrow addSyntheticPoints(D)$;
10     $E \leftarrow E \bigcup$ BuildLevelTree($D_s$, $L$, 0);
11 end
12 return $E$;
```

Once the space is partitioned into regions, the idea is to build a classifier to separate "`+ve`" class (given training data, which are normal) from "`-ve`" class (synthetic data). Anomalies are expected to have low probabilities of belonging to "`+ve`" class (normal data). Local classifier is built in each leaf with more than $minPts$ samples from the "`+ve`" class and current class probabilities are recorded in other leaves. To ensure balanced class distribution to build a classifier in a local region, we first remove all "`-ve`" class samples (synthetic points) and then add the same amount of new synthetic points ("`-ve`" class) as the "`+ve`" class samples in the region. The synthetic points are sampled uniformly at random from the range of possible values in the region. With the same amount of "`+ve`" class and "`-ve`" class (newly added) samples, local classifier is built. We use Decision Tree (DT) [22] that can handle numeric and categorical features directly as local classifier. An example of a Level Tree in 3-dimensional subspace is provided in Figure 2 and the procedures to build LTs and local DTs are provided in Algorithms 1-3. In the testing phase the anomaly score of a test instance $\mathbf{x}$ is estimated as the aggregated $P(+ve|\mathbf{x})$ over t local models. Anomalies are expected to have low aggregated score than normal data. We call the proposed method as "**Ensemble of Local Decision Trees (ELDT)**".

Algorithm 2: BuildLevelTree(D_s, L, j) - Build a single Level Tree recursively

```
Input  : $D_s$ - Data with synthetic points to build a tree, $L$ - Attribute list, $j$
         - Current tree level
Output: $node$ - Level Tree node

// Check if we have enough positive samples
if $|D_s^+| < minPts$ then
    return As a leaf with $P^+ = \frac{|D_s^+|}{N}$ and $P^- = 1.0 - P^+$;
end
// Check if we have a pure node with all -ve class samples
if $|D_s^-| = |D_s|$ then
    return As a leaf with $P^+ = 0.0$ and $P^- = 1.0$;
end
// Check if we have a pure node with all +ve class samples
if $|D_s^+| = |D_s|$ then
    // Build a local DT in the node.
    return BuildLocalDecisionTree($D_s$);
end
if $j = m$ then // $m$ is the maximum level of the Level Tree
    // Build a local DT in the node.
    return BuildLocalDecisionTree($D_s$);
end
// retrieve the next attribute from $L$ based on current level, $j$
$a \leftarrow nextAttribute(L, j)$;
// Construct a node with attribute $a$
if $a$ is a numeric attribute then
    // cut-point selection based on information gain
    $node.splitpoint \leftarrow$ findSplitPoint($a$, $D_t$);
    $D_1 \leftarrow$ filter($D_t$, $a > node.splitpoint$);
    $D_2 \leftarrow$ filter($D_t$, $a \leq node.splitpoint$);
    $node.branch(1) \leftarrow$ BuildLevelTree($D_1$, $L$, $j + 1$);
    $node.branch(2) \leftarrow$ BuildLevelTree($D_2$, $L$, $j + 1$);
else
    // split according to categorical values
    let $\{v_1, \ldots, v_w\}$ be possible values of $a$;
    for $i = 1$ to $w$ do
        $D_i \leftarrow$ filter($D_t$, $a == v_i$);
        $node.branch(i) \leftarrow$ BuildLevelTree($D_i$, $L$, $j + 1$);
    end
end
return $node$;
```

Algorithm 3: BuildLocalDecisionTree(D_s) - Build a local Decision Tree

```
Input  : $D_s$ - Training set
Output: $node$ - Level Tree node
1 RemoveOldSyntheticPoints($D_s$);
2 $D_S \leftarrow addSyntheticPoints(D_s)$;
3 // Learn a Decision Tree
4 $node.localModel \leftarrow$ BuildDecisionTree($D_S$);
5 return $node$;
```

The ensemble of local DT based on the idea of Feating addresses the three limitations of existing AD approaches in practical problems discussed in Section 1. It does not require labelled anomalies. It examines anomalies with respect to their local context or locality defined by multiple local regions. This is useful to detect local anomalies. Using DT that can handle categorical and/or mixed features directly at the local regions, it works well with categorical and mixed data.

4 Experimental results

In this section, we present the results of our experiments conducted to evaluate the performance of ELDT. The three parameters of ELDT were set as default to: $minPts = \lfloor \log_2(|D|) \rfloor + 1$, (note that D is the training normal data); $m = \lfloor \log_2(|F|) \rfloor + 1$ (note that F is the set of features of D), and $t = 100$. We compared the performance of ELDT with Bagging using DT (Bag.DT) [9] and Random Forest (RF) [10], where each model in the ensemble is a global DT for the entire data space. They also used D as "`+ve`" class and the same amount of synthetic points as "`-ve`" class as did in building level trees in ELDT. Each tree in the ensemble has different -ve class samples to ensure diversity between trees. We considered the following three state-of-the-art AD methods as main baselines:

1. iForest [21]: It is an ensemble-based AD method. It uses a collection of t random trees, where each tree T_i is constructed from a small random subsample of data $\mathcal{D}_i \subset D$, $|\mathcal{D}_i| = \psi$ (=256 by default). The idea is to isolate each instance in $\mathcal{D}_i$. Anomalies are expected to have shorter average path lengths over the collection of random trees. It produces good results and runs significantly fast. It works only with numeric features, so categorical features are converted into numeric features using one-hot encoding. It is unable to detect local anomalies [5].
2. LOF [11]: It is the most widely used AD method based on kNN ($k = \lfloor \sqrt{N} \rfloor$ by default) search. It compares the density of a test instance with the average densities of its kNNs. It examines anomalies w.r.t. to their locality defined by the kNNs. It is a local model-based existing AD method. It is also mainly for numeric data, categorical features have to be first converted into numeric features. It is computationally very expensive when D is large.

3. SPAD [4]: It is a simple probabilistic AD method, where multi-dimensional probability is estimated as the product of one-dimensional probabilities assuming features are independent. It works with discrete or categorical data. Numerical features are converted into categorical features through equal-width discretisation [15] with the number of bins b $(=\lfloor \log_2(N) \rfloor + 1$ by default). Despite its simplicity, it has been shown to perform better than more complex methods such as LOF, One-Class SVM and iForest [4].

We used 10 benchmark datasets with categorical only, mixed (categorical and numeric) and numeric only features. The characteristics of datasets used in terms of data size, dimensionality (numeric and categorical) and proportion of anomalies are provided in Table 1. Most of these datasets are from the UCI Machine Learning Repository [14][1]. All methods are implemented in JAVA using the WEKA platform [16]. We used **A**rea **U**nder the Receiver Operating Characteristic (ROC) **C**urve (AUC) as the performance evaluation metric. We conducted 10 trials of different train (D) and test (Q) sets and presented the average AUC over 10 runs.

Table 1: Characteristics of data sets. #Inst: data size, #Feat: num. of features, #NFeat: num. of numeric features, #CFeat: num. of categorical feaures, anomaly%: percentage of anomalies

Name	#Inst	#Feat	#NFeat	#CFeat	anomaly%
Census	299285	40	7	33	6.0
Covertype	287128	12	10	2	1.0
Kddcup99	64759	41	34	7	6.5
U2r	60821	41	34	7	0.5
Mnist	20444	96	96	0	3.5
Annthyroid	7200	21	6	15	7.5
Chess	4580	6	0	6	0.5
Mushroom	4429	22	0	22	5.0
Hypothyroid	3772	29	7	22	7.5
Spambase	2964	57	57	0	6.0

The average AUC results of contending methods are provided in Table 2. The results show that ELDT produced best results overall with the average AUC of 0.918 and average rank of 2.0 over 10 datasets used. It had the best AUC in four out of 10 datasets followed by Bag.DT in three datasets, RF and SPAD in two datasets each, and iFoest and LOF in only one dataset each. The closest contender in terms of consistent performance across datasets is SPAD the average AUC of 0.864 and the average rank of 3.3. Though Bag.DT produced the best results in three datasets, it performed worst in the other four datasets, whereas ELDT was ranked second in three datasets, third in two datasets and forth in the remaining one dataset. This results show that ELDT produced more consistent

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

results across different datasets with numeric only, categorical only and mixed attributes and those with local and global anomalies. Among the three baselines, SPAD has the best overall performance. These results are consistent with those claimed by the authors in [4].

The runtime results in the five largest datasets with more than 10,000 instances are presented in Table 3. These results show that ELDT ran slower than all contender except LOF, but it had the runtimes in the same order of magnitudes with them. It was at last one order of magnitude faster than LOF, two orders of magnitude faster in the largest dataset.

Table 2: Average AUC over 10 runs. The best performance in each dataset is highlighted on bold.

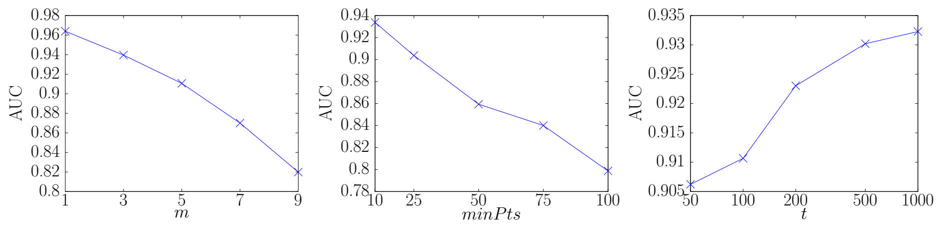
Dataset	ELDT	Bag.DT	RF	iForest	LOF	SPAD
Census	**0.713**	0.561	0.57	0.589	0.491	0.684
Covertype	**0.995**	0.974	0.984	0.945	0.992	0.966
Kddcup99	0.993	0.504	0.636	**0.998**	0.896	**0.998**
U2r	**0.988**	0.515	0.576	0.978	0.931	0.988
Mnist	0.815	0.691	0.771	0.841	**0.880**	0.824
Annthyroid	0.921	**0.975**	0.802	0.771	0.612	0.705
Chess	0.998	0.997	**1.000**	0.889	0.968	0.995
Mushroom	0.999	**1.000**	**1.000**	0.791	0.996	0.977
Hypothyroid	0.953	**0.977**	0.894	0.694	0.607	0.723
Spambase	**0.808**	0.549	0.718	0.805	0.659	0.781
Avg. AUC	**0.918**	0.774	0.795	0.830	0.803	0.864
Avg. Rank	**2.0**	3.9	3.4	3.8	4.3	3.3

Table 3: Average runtime (in seconds) over 10 runs in the five largest datasets with more than 10,000 instances.

Dataset	Ft.DT	Bag.DT	RF	iForest	LOF	SPAD
Census	420	346	119	180	58,140	180
Covertype	1,621	1,572	187	39	10,429	150
Kddcup99	39	88	20	6	1,424	7
U2r	32	79	20	4	1,431	8
Mnist	84	282	19	2	160	6

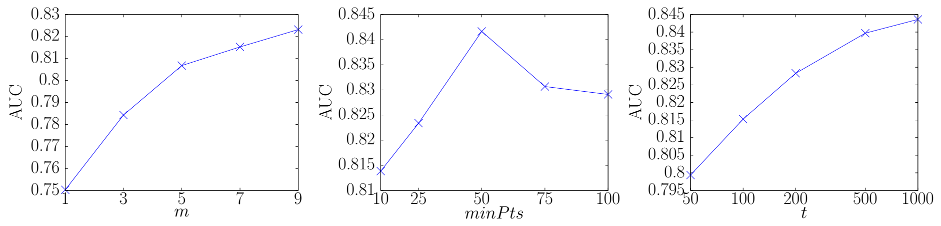
4.1 Sensitivity of parameters

In this section, we present the results of experiments conducted to assess the sensitivity of the three parameters, m (the size of subspaces that determines the maximum height of Level Trees), $minPts$ (minimum points required at leaf nodes to build local AD models) and t (ensemble size), in the performance of ELDT. We varied one parameter at a time setting the other two parameters to default values. For this experiments, we used two datasets - Annthyroid and Mnist. The results are presented in Figures 3 and 4. The results show that the performance of ELDT can be improved by setting m and $minPts$ properly. In terms of t, higher the better. In both cases, performance improved when t was increased and started to flatten. Increasing t also increases time and space complexities linearly. Therefore, there has to be a trade-off between performance and complexities.



(a) m (b) $minPts$ (c) t

Fig. 3: Annthyroid: Effect of parameter in ELDT



(a) m (b) $minPts$ (c) t

Fig. 4: Mnist: Effect of parameter in ELDT

5 Conclusions and future work

In this paper, we presented a simple idea to address the three main limitations of existing Anomaly Detection (AD) methods in practical applications: (i) lack of sufficient examples of known anomalies; (ii) unable to detect local anomalies; and (iii) inability to handle mixed attributes well. Instead of using one global model, we propose to partition the data space into many regions and build an AD model in each region using data falling in the region only, i.e., many local AD models are built. To make prediction for a test instance, the AD model learned on the region where it falls is used. Instead of just relying on a local region from one partitioning of the space, we proposed to create multiple partitions of the data space and use ensemble of multiple local models learned on local regions from each partition. It exploits the benefits of ensemble learning to consider sufficient locality around the test instance. We used the idea of Feating to partition the data space. Though there are not many AD models that can work well with mixed data, there are classifiers such as traditional Decision Tree (DT) that can handle numeric and categorical features directly. We used DTs in local regions for AD without using labelled anomalies by adding synthetic data. We presented a new AD method called Ensemble of Local Decision Trees (ELDT). Our results show that ELDT produces better and more consistent detection results compared to popular state-of-the-art AD methods, particularly in datasets with mixed types of features.

Our results suggest that ensemble of local AD models produces better results than using a single global model. AD algorithms that can handle categorical and numeric features directly without any conversion produce better results than using methods designed for only type of features, which require all features to be converted into the supported type. AD problems can be converted into classification problems by adding uniformly distributed synthetic points and classification algorithms can be used. Our results indicate that it is a very promising line of research to investigate further to develop a flexible and robust AD framework for practical use. It can lead to a general ensemble learning framework for AD, where different space partitioning techniques can be used to define local regions and any classifier or AD algorithm can be used in local regions. In this paper, we presented one simple variant of it. In future, we would like to focus on: (i) using other classifiers (e.g., Naive Bayes, KNN, SVM, Neural Networks, etc.) and AD methods (e.g., LOF, SPAD, One-Class SVM, etc.) as local models; and (ii) investigating different implementations of space partitioning: using trees (e.g., Feating), grids, nearest neighbours, etc.

Acknowledgement

This research was funded by the Department of Defence and the Office of National Intelligence under the AI for Decision Making Program, delivered in partnership with the Defence Science Institute in Victoria, Australia.

References

1. Aggarwal, C.C., Sathe, S.: Outlier Ensembles: An Introduction. Springer, Cham (2017)
2. Akoglu, L., Tong, H., Vreeken, J., Faloutsos, C.: Fast and reliable anomaly detection in categorical data. In: Proceedings of the 21st ACM Conference on Information and Knowledge Management (CIKM). pp. 415–424 (2012)

3. Aryal, S.: 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 (2018)
4. Aryal, S., Ting, K.M., Haffari, G.: 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 (2016)
5. Aryal, S., Ting, K.M., Wells, J.R., Washio, T.: Improving iForest with Relative Mass. In: Proceedings of the 18th Pacific-Asia Conference on Knowledge Discovery and Data Mining. pp. 510–521 (2014)
6. Bay, S.D., Schwabacher, M.: Mining distance-based outliers in near linear time with randomization and a simple pruning rule. In: Proceedings of the ninth ACM International Conference on Knowledge Discovery and Data Mining. pp. 29–38 (2003)
7. Bentley, J.L., Friedman, J.H.: Data structures for range searching. ACM Computing Surveys **11**(4), 397–409 (1979)
8. Boriah, S., Chandola, V., Kumar, V.: Similarity measures for categorical data: A comparative evaluation. In: Proceedings of the eighth SIAM International Conference on Data Mining. pp. 243–254 (2008)
9. Breiman, L.: Bagging predictors. Machine Learning **24**(2), 123–140 (1996)
10. Breiman, L.: Random forests. Machine Learning **45**(1), 5–32 (2001)
11. Breunig, M.M., Kriegel, H.P., Ng, R.T., Sander, J.: LOF: Identifying Density-Based Local Outliers. In: In Proceedings of ACM SIGMOD International Conference on Management of Data. pp. 93–104 (2000)
12. Chandola, V., Banerjee, A., Kumar, V.: Anomaly detection: A survey. ACM Computing Surveys **41**(3), 15:1–15:58 (2009)
13. Dietterich, T.G.: Ensemble methods in machine learning. In: Proceedings of the First International Workshop on Multiple Classifier Systems. pp. 1–15 (2000)
14. Dua, D., Graff, C.: UCI machine learning repository. http://archive.ics.uci.edu/ml (2019)
15. Duda, R.O., Hart, P.E., Stork, D.G.: Pattern Classification (2nd Edition). Wiley-Interscience (2000)
16. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H.: The WEKA Data Mining Software: An Update. SIGKDD Exploration Newsletter **11**(1), 10–18 (2009)
17. He, Z., Xu, X., Huang, J.Z., Deng, S.: FP-Outlier: Frequent Pattern Based Outlier Detection. Computer Science and Information Systems **2**(1), 103–118 (2005)
18. Hilario, A.F., López, S.C., Galar, M., Prati, R., Krawczyk, B., Herrera, F.: earning from Imbalanced Data Sets. Springer (2018)
19. Hinneburg, A., Aggarwal, C.C., Keim, D.A.: What is the nearest neighbor in high dimensional spaces? In: Proceedings of the 26th International Conference on Very Large Data Bases. pp. 506–515. VLDB '00 (2000)
20. Lazarevic, A., Kumar, V.: Feature bagging for outlier detection. In: Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining (KDD). pp. 157–166 (2005)
21. Liu, F., Ting, K.M., Zhou, Z.H.: Isolation forest. In: In Proceedings of the Eighth IEEE International Conference on Data Mining. pp. 413–422 (2008)
22. Quinlan, J.R.: Induction of decision trees. Machine Learning **1**(1), 81–106 (1986)
23. Schölkopf, B., Platt, J.C., Shawe-Taylor, J.C., Smola, A.J., Williamson, R.C.: Estimating the support of a high-dimensional distribution. Neural Computation **13**(7), 1443–1471 (2001)

24. Shi, T., Horvath, S.: Unsupervised learning with random forest predictors. Journal of Computational and Graphical Statistics **15**(1), 118–138 (2006)
25. Sugiyama, M., Borgwardt, K.M.: Rapid distance-based outlier detection via sampling. In: Proceedings of the 27th Annual Conference on Neural Information Processing Systems. pp. 467–475 (2013)
26. Taha, A., Hadi, A.S.: Anomaly detection methods for categorical data: A review. ACM Computing Surveys **52**(2), 38:1–38:35 (2019)
27. Tax, D.M., Duin, R.P.: Support vector data description. Machine Learning **54**, 45–66 (2004)
28. Ting, K.M., Wells, J.R., Tan, S.C., Teng, S.W., Webb, G.I.: Feature-subspace aggregating: Ensembles for stable and unstable learners. Machine Learning **82**(3), 375–397 (2011)
29. Zimek, A., Gaudet, M., Campello, R.J., Sander, J.: Subsampling for efficient and effective unsupervised outlier detection ensembles. In: Proceedings of KDD. pp. 428–436 (2013)