24th European Conference on Artificial Intelligence - ECAI 2020
Santiago de Compostela, Spain

Anomaly Detection Forest

Jakob Sternby[1] and **Erik Thormarker**[2] and **Michael Liljenstam**[3]

Abstract. In many applications of anomaly detection, data with labeled anomalies may not be readily available. When designing a system for detection of malicious network activity, for instance, comprehensive malicious data can not be obtained since attacks constantly evolve and hence change their pattern. In such cases, an automatic anomaly detection system needs, at least partly, to rely on one-class learning, where training data exclusively contains normal instances. In this paper, we present a new anomaly detection algorithm, the Anomaly Detection Forest, optimized for the one-class learning problem. The algorithm is an ensemble of binary trees, where each tree is trained on a random subset and where the location of empty leaves define the anomaly score attributed to a data point. Our experimental results on a set with 14 public datasets show that the new algorithm outperforms the state-of-the-art algorithms Isolation Forest and One-Class Random Forest for the task of anomaly detection in the one-class learning setting.

1 Introduction

In machine learning, *anomalies* or *outliers* are data points that significantly differ from other data points in the same dataset. In many fields, such distinctly differing points often correspond to properties of particular interest such as cancer cells within otherwise healthy tissue, errors in automated systems, malicious network activity, or financial fraud, to give a few examples. The general task of automatically highlighting such unexpected data points with machine learning methods is usually called *anomaly detection*, *outlier detection*, or *novelty detection*. Methods are often *unsupervised*, as one common use case is the identification of errors within an unlabeled dataset. Another common use case is to train a model only on data known to be normal. This is often called *one-class learning*, and, as this utilizes only one label, it may be viewed as a special case of *semi-supervised* learning [1]. It is also common to use the term *novelty detection* specifically for the case of anomaly detection with *one-class learning* [20].

As the case of anomaly detection with exclusive normal instance learning is methodogically aligned with the problem of unsupervised anomaly detection, methods for the latter can also be applied to the first case [1]. It does not, however, imply that the same performance can be guaranteed. Most methods for these *anomaly detection* tasks employ a scoring function that aims at producing distinctly different values for anomalies than for normal points so that the decision function can be defined by a threshold value [20]. Common such scoring functions include distance functions [3], density measuring functions [2, 4, 15], reconstruction error [5, 6, 22], or expected path length in decision trees [13, 16]. Many of these are specific to the anomaly detection problem whereas others are adoptions of some well-known multi-class method to the one-class domain. Examples of the latter include one-class support vector machines [7] and one-class random forests [12].

The Isolation Forest (IF), is an ensemble of search trees that has been a consistent top performing algorithm for unsupervised anomaly detection [8, 10], but the authors also claim to have the same performance for one-class learning with normal instances [18]. Each isolation tree in the IF is a binary classification tree constructed by passing a subsample of the training set to the root node and successively splitting it into smaller portions by choosing a random pair of feature and split value at each node. In the IF, the split value is chosen from within the range of values handled in a particular node. The splitting continues at each node until there is only one sample left, in which case the node is called a leaf. Due to the randomness of feature and split-value selection, leaves can occur at various depths in the trees. An anomaly score is produced by passing a sample through each of the trees in the ensemble and calculating the normalized average length of the path before reaching a leaf. A shorter path implies that the leaf's creation, i.e. the separation of a single sample by a split-value threshold, was achieved on a large partition on the tree training subsample using only a small number of random cuts, suggesting it is an outlier.

In this paper we present a novel algorithm for creating a forest of search trees optimized for the *one-class learning* case of anomaly detection, motivated by the observation that the IF is not as well suited for this scenario. As described above, the leaves providing high anomaly scores in the IF all stem from observations in the training set. Since all observations in the training set are normal instances for the one-class learning case, this means that, for a given threshold, the IF algorithm has an intrinsic bias towards labelling similar uncommon normal instances as anomalies. The magnitude of this problem depends on the ratio of anomalies compared to normal instances similar to those that created the isolation leaves. The reverse problem for the IF in the one-class learning case may be even worse: for an anomaly to have a high probability of a high anomaly score it is required that there exist training instances with feature values, extreme in the same way as the anomaly, so that leaves based on these features are created close to the root during training. With our Anomaly Detection Forest (ADF) algorithm, we propose a new node structure to improve upon these deficiencies by introducing two new concepts: the first is the introduction of empty leaves or *anomaly leaves* with the objective of catching anomalies with feature values not contained in the range of the tree training subsample of observations. The other key concept is the *isolation level*, which is a lower size bound on the training samples in a node during training, determining when anomaly leaves should be created instead of subdividing the training samples further. In the experimental section we benchmark the ADF against state-of-the-art algorithms and show that ADF provides

[1] Ericsson AB, Sweden, email: jakob.sternby@ericsson.com
[2] Ericsson AB, Sweden
[3] Ericsson Inc, USA

the strongest results for the task of anomaly detection in the one-class learning setting. Furthermore, the ADF is flexible in the sense that it also supports binary and one-hot-encoding features.

2 Related work

An overview of methods for the one-class learning setting for anomaly detection (novelty detection in the authors terminology) can be found in [20]. Many of these are multi-class methods that have been adjusted to the one-class setting. They include one-class Support Vector Machines [7], Least Squares Anomaly Detection [21] and One-Class Random Forest (OCRF) [12]. Of these, OCRF recently showed the best performance and has been included in the experiments of this paper [12].

Despite the wide adoption of IF since its introduction in 2008, not much attention has been given to its performance in the one-class learning setting. A generalization that enables other internal distance functions can be found in [25]. In [14], the coarseness of the axis-parallel cuts of the IF is mitigated by using a randomly rotating axis, qualitatively showing more consistent anomaly scoring. In the SCi-Forest algorithm, the decision function in each node uses a hyperplane in randomly selected features to improve the splitting capacity [17]. The Random Cut Forest algorithm presented in [13] has a different focus and instead considers if there is a better choice than choosing the features at random. The authors show improvements on some datasets by weighing the probability of selecting a given feature by its relative variability in the training data.

3 Anomaly Detection Forest

Anomalies signified by discrepancy in a feature that shows great consistency for normal instances should be easy to detect. In high dimensional space, however, such distinct differences in a smaller subspace may be overshadowed by arbitrary variations in other features. Furthermore, as many anomaly detection algorithms have been designed for the *unsupervised* anomaly detection problem, it is common that the impact of a given feature is weighted by its variance in the set. This may not be effective for the one-class learning case, where an anomalous instance could be characterized exactly by the difference in a feature with very low variance in the training set. With the ADF we propose a novel forest method for detecting anomalies by focusing on feature values not previously observed in the subsamples used to construct each tree. With a random feature selection mechanism similar to the IF, this enables the ADF to produce high anomaly scores for data samples with distinct differences in any feature regardless of its variance in the training subsamples. In this section we will subsequently describe the training of the proposed new tree structure as well as the anomaly scoring adopted from [18].

3.1 Binary Decision Trees

We define a binary decision function as a function $f : \mathcal{F} \mapsto \{0, 1\}$, where $\mathcal{F}$ is the sample feature space. A binary decision tree $T = N_1, ..., N_m$ is a tree where each node N is associated with a binary decision function f_N. Each node in a binary tree has either 0 or 2 children which we call *left* and *right*. A node without children is called a *leaf*. We use operators $\mathcal{P}, \mathcal{C}$ to denote parent and children respectively s.t. $\mathcal{P}(N)$ is the parent of N and $\mathcal{C}_L(N)$ is the left child of N, where the subscript L, R denotes left or right child. Hence, we also have $\mathcal{P}(\mathcal{C}_R(N)) = N$. A set of training samples $X = x_1, ..., x_n$ of F features, s.t. $x_j = x_j(1), \ldots, x_j(F)$, is fit to a binary decision tree T by successively subdividing X from the root node N_1 (thus $X_{N_1} = X$) s.t. the subset X_{N_j} of a node $N_j = \mathcal{C}(N_k)$ is defined by:

$$X_{N_j} := \begin{cases} \{x | f_{N_k}(x) = 0, x \in X_{N_k}\} & \text{if } N_j = \mathcal{C}_L(N_k) \\ \{x | f_{N_k}(x) = 1, x \in X_{N_k}\} & \text{if } N_j = \mathcal{C}_R(N_k) \end{cases} \quad (1)$$

We use the notation $T(X)$ for a tree T fit to a dataset X.

A threshold t for a single feature r is one way of defining a simple binary decision function, often referred to as an axis parallel feature split as seen in (2).

$$f(x, r, t) = \begin{cases} 0 & \text{if } x(r) < t \\ 1 & \text{if } x(r) \geq t. \end{cases} \quad (2)$$

A node N in a binary decision tree with an axis-parallel feature split is defined by the triplet $N = (X_N, r_N, t_N)$, denoting node training samples, selected feature, and threshold split-value.

3.2 Anomaly Detection Trees

One of the distinguishing contributions of the ADF algorithm is the introduction of *anomaly catcher* nodes:

Definition 1. *An* anomaly catcher *is a node N with two children s.t. the corresponding training subset of one child N' is the empty set, while for the other child N'', we have $X_{N''} = X_N$.*

In other words, each *anomaly catcher* has a child which is an empty leaf that we call an *anomaly leaf*. In order for a node N to be an *anomaly catcher* according to Definition 1, we see from equations (1) and (2) that the following needs to hold for the node threshold t_N:

$$t_N \leq \min_{x \in X_N} x(r_N) \vee t_N > \max_{x \in X_N} x(r_N). \quad (3)$$

Whereas the *anomaly catcher* node aims to accentuate the anomaly scores of distinct anomalies by selecting split-values outside the range of the node training samples, the other key contribution of the ADF, the *isolation level* is introduced to reduce the anomaly scores of normal instances:

Definition 2. *The* isolation level $\eta \in (0, 0.25)$ *of a binary decision tree $T(X)$ is a size threshold such that any node N, where $\frac{|X_N|}{|X|} \leq \eta$, will be an* anomaly catcher.

In Definition 2 $|X|$ is used for the size of the set X, and this notation will be used throughout the rest of the paper.

In order to achieve *anomaly catchers* based on sample subsets of reasonable size, the nodes which are *not anomaly catchers* can be programmed to avoid isolation of training samples. With this objective, we propose Algorithm 2 to calculate the threshold t_N, for these nodes N, which we label as *subdivision nodes*. By Algorithm 2 the corresponding requirement to (3) for *subdivision nodes* is:

$$\lfloor (0.5 - 2\eta)|X_N| \rfloor \leq |\{x \in X_N : x(r) < t_N\}| \leq \lceil (0.5 + 2\eta)|X_N| \rceil. \quad (4)$$

Consequently, for most values of $\eta, |X|$, neither child to a *subdivision node* will be an *anomaly leaf*.

With Definition 1 and Definition 2, the novel Anomaly Detection Tree (ADT) structure can now be defined as:

Definition 3. *An ADT is a binary decision tree $T(\eta, X)$ with* isolation level *η, fit to a sample set X, consisting of* subdivision nodes *and* anomaly catchers *s.t. at least all nodes N with $|X_N| \leq \eta \cdot |X|$ are* anomaly catchers.

3.2.1 Threshold value space

One crucial aspect when using an *anomaly catcher threshold* as in (3) is how to choose reasonable values outside of the observed range. The feature value space $V = \{V_r\}_{r=1}^{F}$ in a root node is determined by the *a priori* value intervals $V_r = [f_r^{\min}, f_r^{\max}]$, of the theoretically possible values for feature r. Apart from the binary case, some outer boundary is necessary since even anomalies will have bounded values and the true value interval may be unknown. To get a reasonable starting point, the *standard deviation* (σ_r) of each feature $r \in (1, \ldots, F)$ in the complete training set $\mathcal{X}$ (not just the subset of the given tree) is used as boundary in the root node N_1:

$$V_r(N_1) = [f_r^{\min}, f_r^{\max}] \bigcap [\min_{x \in X_{N_1}} x(r) - a\sigma_r, \max_{x \in X_{N_1}} x(r) + a\sigma_r], \quad (5)$$

where a is the *anomaly margin*, which can typically be set to 1. Below the root node, the value space of a feature is further limited by any split made in that feature higher up in the tree. As an example, there can not be any values $x(r_{\mathcal{P}(N)}) < t_{\mathcal{P}(N)}, x \in X_N$ for any node $N = \mathcal{C}_R(\mathcal{P}(N))$ since any samples with such values arriving at $\mathcal{P}(N)$ would subsequently be sent to the left subtree according to (1). This can be formalized with the following inductive formula for the feature value space for feature r at the children of node N:

$$\begin{aligned} V_r(\mathcal{C}(N)) &= V_r(N), \text{ if } r \neq r_N \\ V_r(\mathcal{C}_R(N)) &= V_r(N) \bigcap [t_N, \infty), \text{if } r = r_N \\ V_r(\mathcal{C}_L(N)) &= V_r(N) \bigcap (-\infty, t_N], \text{if } r = r_N. \end{aligned} \quad (6)$$

As the value space of each node depends on that of its parent, the span of the value space will decrease with depth. This implies that anomaly leaves deeper in the tree may require less deviation from the seen data, thereby providing some quantification of abnormality for unseen values.

3.3 Training

Algorithm 1 describes a method to fit (train) an ADT T to a sample set X based on Definition 3. The function rand(a, b) corresponds to the generation of random real values from the uniform distribution in the interval $[a, b]$. The function randi(a, b) is the analogue for random integer values. For a positive integer F, the function randperm(F) returns a permutation of the list $(1, 2, ..., F)$ chosen uniformly at random. The training of ADF with trees $\mathcal{T} = \{T_1, \ldots, T_{|\mathcal{T}|}\}$, is initialized by selecting corresponding subsets $X^i, i = 1, \ldots, |\mathcal{T}|$ randomly drawn from the complete set of training samples $\mathcal{X}$ and running the *ADTree*() function in Algorithm 1. Depending on the size $|X|$ of the supplied subset (it will be less or equal than that of its parent at each recursive call), thresholds will be selected according to one of the two functions *SDThreshold*() or *ACThreshold*().

In the first levels of the tree, until the size of the supplied subset $|X_N|$ is below the isolation level $\eta \cdot |X|$, the *SDThreshold* function is used to determine the threshold value t_N for node N. These nodes are called *subdivision nodes*. Algorithm 2 describes the process for selecting such a subdivision threshold t in a Tree $T(\eta, X)$.

Note that each time that Algorithm 2 returns a value for node N, the sizes of both subsets $X_{\mathcal{C}_L(N)}, X_{\mathcal{C}_R(N)}$ will be smaller than $|X_N|$. Hence, after a sufficient number of recursions in Algorithm 1, the size of the supplied subset X_N to the node N falls below $\eta \cdot |X_{N_1}|$ and the *ACThreshold*() function is used to determine the node threshold t_N. The *ACTreshold*() function described in Algorithm 3, is slightly more involved due to the inductive nature of (6), but it should

Algorithm 1: ADTree(X,η, d, D)

Data: Sample subset X with F features, isolation level η, depth d, max depth D, feature value space V
Result: Anomaly Detection Tree T
begin
  $T = \{L = \text{NULL}, R = \text{NULL}\}$
  if $d > D$ *or* $|X| \leq 1$ **then**
    return T
  for r **in** randperm(F) **do**
    Update V_r for current node according to (6)
    if $|X| \leq \eta \cdot |X_{N_1}|$ **then**
      `/* Anomaly catcher */`
      $t = \text{ACThreshold}(X, r, V_r)$
      if $t \neq$ *NULL* **then**
        $T.t = t, T.r = r$
        break
    else
      `/* Subdivision node */`
      $t = \text{SDThreshold}(X, \eta, r)$
      if $t \neq$ *NULL* **then**
        $T.t = t, T.r = r$
        break
  $T.L \leftarrow \text{ADTree}(\{x(r) < t | x \in X\}, \eta, d + 1, D, V)$
  $T.R \leftarrow \text{ADTree}(\{x(r) \geq t | x \in X\}, \eta, d + 1, D, V)$
  return T

Algorithm 2: SDThreshold(X, η, r)

Data: Sample subset X, Isolation level η, feature r
Result: A randomly chosen Subvidision Threshold t satisfying (4), if such a threshold exists.
begin
  $X' = \{x'_1, \ldots, x'_{|X|}\} = \text{Sort}(X(r))$ ascending
  $a = x'_{\text{floor}((0.5-2\eta)\cdot|X|)}(r)$
  $b = x'_{\text{ceil}((0.5+2\eta)\cdot|X|)}(r)$
  if $a = b$ **then**
    return NULL
  return rand(a, b)

still be clear that the obtained threshold t satisfies both the requirement in (3) and that it is reachable according to the discussion in the section about threshold value space. As these *anomaly catcher* nodes never divide the subset into smaller parts, the creation of a leaf ($|X| \leq 1$) is not a sufficient condition for Algorithm 1 to converge. Consequently, each node keeps track of its depth d, and the training also terminates when the supplied maximum depth D is reached.

Algorithm 3: ACThreshold(X, r, V_r)

Data: Samples X, feature r, value space V_r for feature r
Result: A randomly chosen Anomaly Cather Threshold $t \in V_r$ satisfying (3), if such a threshold exists.

begin
  /* check existence outside feature span */
  if $V_r \setminus [\min_{x \in X} x(r), \max_{x \in X} x(r)] = \emptyset$ **then**
    return NULL
  /* check existence below feature span */
  if $\min_{v \in V_r} v = \min_{x \in X} x(r)$ **then**
    return rand$(\max_{x \in X} x(r), \max_{v \in V_r} v)$
  if *rand*$(0, 1) > 0.5$ **then**
    return rand$(\max_{x \in X} x(r), \max_{v \in V_r} v)$
  return rand$(\min_{v \in V_r} v, \min_{x \in X} x(r))$

3.4 Anomaly scoring

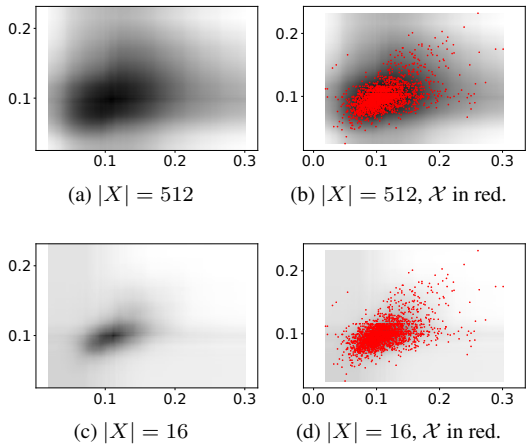
The scoring mechanism of the ADF is almost the same as that of the IF, however, there is a slight difference in how normalization is conducted. In IF, the average path length of the trees in the forest is divided by a constant that relates to the average path length of unsuccessful searches in a binary search tree [18]. In the case of the ADF, however, the average path length for normal samples is more dependent on the data, as well as the isolation level, and, maximum depth that control the topology of the constituent trees. For this reason we change the normalization of the scores to use the observed average path length over the training set instead. The anomaly score for a sample x' over a set of trees $\mathcal{T} = \{T_1, \ldots, T_{|\mathcal{T}|}\}$ trained on set $\mathcal{X}$ is then obtained by

$$s(x', \mathcal{X}) = 2^{-\frac{\sum_j (l(x', T_j))/|\mathcal{T}|}{l^*(\mathcal{X}, \mathcal{T})}}, \tag{7}$$

where $l(x', T)$ is the path length for sample x' in tree T and $l^*(\mathcal{X}, \mathcal{T}) = \frac{1}{|\mathcal{X}||\mathcal{T}|} \sum_{x \in \mathcal{X}} \sum_j l(x, T_j)$. Recall that $\mathcal{X}$ is the complete training set and not only the subsample X^i used to train tree T_i. For normal instances x, $l(x, T_i)$ is expected to be close to the max depth D_{T_i} of tree T_i, whereas distinct anomalies are expected to end up in at least some anomaly leaves during scoring.

3.5 Properties of ADF

Each path to the first anomaly catcher nodes in an Anomaly Detection tree can be seen as a random subspace containing a subsample of the training set. The following anomaly catcher nodes continue to put more borders in random dimensions further enclosing this part.



(a) $|X| = 512$ (b) $|X| = 512$, $\mathcal{X}$ in red. (c) $|X| = 16$ (d) $|X| = 16$, $\mathcal{X}$ in red.

Figure 1. Anomaly catcher subspace density plot for ADF model (*thyroid-ann* dim. 3,4).

Adding many such tree models creates overlapping subspaces such that dense parts of training space will be present in more such enclosures. In this respect, the ADF can be seen as a density model of the training sample space[4], where samples corresponding to dense areas in training space have a larger probability of being covered (i.e. *not* isolated) by more trees. Figure 1 shows plots visualizing this by creating a histogram of the covered areas of the paths to the anomaly catcher nodes at the tree max depth, i.e. the anomaly catcher node without children, in all trees of two ADF models (trained with differing subsample size). As seen in the figure, a larger subsample size corresponds to more smoothing of the density estimation of the normal space.

3.5.1 *Isolation*

By searching for values outside of the observed value span for different features, the ADF is very capable of providing lower path lengths for anomalies with feature values different from those observed during training. It is quite obvious that the IF may miss such anomalies unless there is also some training sample with similar properties since in the IF, leaves are only created by isolating values in the training set. This problem has been presented in [13] as the *Held Out Data* problem. Consider the probabilities for creating an isolating leaf in the two cases. In the IF, given a distinct anomaly z s.t. for some feature r we have $z(r) > \max_{x \in \mathcal{X}} x(r)$, the probability for creating an isolating leaf from a node N with feature r is the same as setting the threshold t larger than the next to largest feature value $x'_{|X'|-1}(r)$ in the sorted node subsample $X' = \{x'_1, \ldots, x'_{|X'|}\}, x'_i \leq x'_j, \forall i < j$ i.e. $\frac{|x'_{|X'|} - x'_{|X'|-1}|}{|x'_{|X'|} - x'_1|}$ due to the uniform selection of t. In the ADF the corresponding probability for choosing a threshold that isolates z does not depend on the density of the node subsample, and instead increases linearly with the separation of $z(r)$ from the interval $[\min_{x \in X'} x(r), \max_{x \in X'} x(r)]$.

There is also the special case of when training samples show zero variance in a feature for a node subsample X_N. The standard IF ignores such features since the node subsample cannot be split. For the ADF however, such features can also be used to create anomaly

[4] Note that the use of the word density here is not meant to indicate any direct relation to the density measuring functions of Section 1.

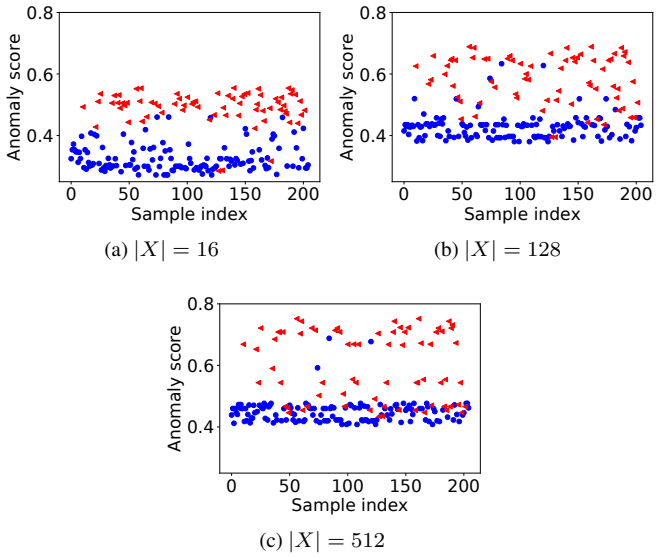
leaves catching data points that show deviation in this feature. This will give a high anomaly score and indicate abnormality as one should expect for feature values that are distinctly different from what is encountered in the training set. Note that this may be particularly useful for binary features (such as one-hot encodings) where duplicate values are common.

The inverse problem may also cause noise from normal samples when trying to detect anomalies. Since each leaf close to the root in the IF corresponds to a normal instance having an isolated value in the training set, any new normal instance with similar feature values may end up in the same leaf during scoring, and thus, get a short path length. Even though an anomaly with even more extreme value for that feature ends up in the same leaf during scoring, the IF algorithm has no way of discriminating the anomaly from the normal instance.

3.5.2 *Impact of subsample size*

An interesting parameter is the subsample size $|X|$, i.e. the size of the partition used to train each tree. There is a simple probabilistic relation between $|X|$ and the probability that a feature value of a normal instance is outside the range $[\min_{x\in X} x(r_N), \max_{x\in X} x(r_N)]$ of a node N. Consider a normal instance x' from a i.i.d set $\mathcal{X}$ and a node subsample $X = \{x_1, \ldots, x_{|X|}\} \subset \mathcal{X}$. Let's now consider a fixed threshold $t \in (\min_{x\in\mathcal{X}} x(r), \max_{x\in\mathcal{X}} x(r))$ for a feature r in ADF. This gives:

$$P(x'(r) \geq t, \max_{x\in X} x(r) < t) =$$
$$P(x'(r) \geq t) \prod^{|X|} P(x(r) < t) =$$
$$(1 - P(x'(r) < t)) \prod^{|X|} P(x(r) < t) \to 0, |X| \to \infty. \quad (8)$$



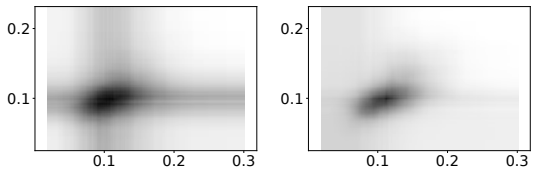
(a) $|X| = 16$ (b) $|X| = 128$

(c) $|X| = 512$

Figure 2. Anomaly scores for ADF for different subsample sizes ($|X|$) on the *breast-w* dataset. Normal instances are depicted by dots and anomalies by triangles.

This means that a forest trained with smaller subsample sizes have a higher probability of treating instances from the normal class as anomalies. It will also imply a higher probability of catching anomalies with values overlapping those of normal instances. Depending on how separated anomalies are from normal instances, this will imply that some anomalies will get lower anomaly scores for a larger subsample size.

When viewing the ADF as a density estimation model of normal space, the subsample size can be seen as a smoothing parameter. The effect of this is shown in Figure 1. In terms of anomaly scoring, the smoothing effect reduces the variance of the scores of samples classified as normal and makes them more similar, while at the same time, it increases the anomaly score of samples classified as anomalous as seen in Figure 2.



(a) $D = 16$ (b) $D = 48$

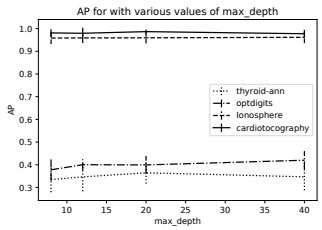
Figure 3. Density plots for ADF of *thyroid-ann* dimensions 3,4 with subsample size 16 at different depths.

3.5.3 *Impact of depth*

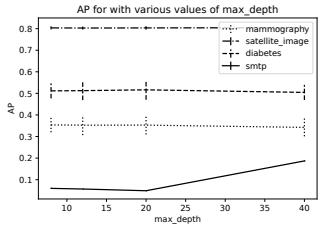
The depth of the anomaly catchers after the subdivision nodes provides boundaries for the normal models. With insufficient depth with respect to the number of features, the boundaries may fail to enclose the normal model, causing the phenomenon of axis-parallel bias, where the normal density spreads to unobserved areas parallel to the modeled area as seen in Figure 3a. This phenomenon for the anomaly scores of IF is the main motivation behind [14]. By increasing the depth as seen in Figure 3b, the arbitrary feature choices for the anomaly catchers succeed in enclosing the training space to be modeled and the phenomenon is reduced. Although the phenomenon is clearly visible in Figure 3a, it does not seem to have a major effect on other performance metrics, and both metrics described in the experiments are fairly constant for most datasets for $|X| = 256$ as seen in Figure 4.

4 Experiments

Since our work can be seen as an adaptation of the IF algorithm for the case of one-class learning we have focused our comparison to this case. We also run experiments for the one-class random forest of [12] using the implementation found in [11] as their results are among the strongest reported in the literature for the one-class anomaly detection problem. Thus, we believe using IF and OCRF as comparison basis is supported by the literature, with IF found to be a top performing anomaly detection algorithm [8, 10] and OCRF among top for one-class anomaly detection. For some use cases, slightly better outlier detection results than IF have been reported recently using Deep or Recurrent Neural Network algorithms [23]. But given our space constraints here, we reserve such comparisons for future work. Performance for anomaly detection is usually presented with the properties of *Receiving Operating Characteristic* (ROC) curves and/or *Precision-Recall* (PR) curves [1]. We provide Area Under



(a)



(b)

Figure 4. AP for varying depths, D

Curve (AUC) for ROC and *average precision* (AP) which is approximately the AUC of the PR curve. In the Results section we argue that the AP is a more suitable performance marker for the common case when the budget for analysis of detections is limited. Since there are random elements to the algorithms benchmarked here, we present all results as average and standard deviation over several runs.

4.1 Datasets

To enable comparison with state-of-the-art algorithms, the datasets in the Open ML [24] and UCI [9] repositories have been prepared in accordance with previous work, as listed in Table 1. Since OCRF only supports variables with a certain level of continuity, some datasets have been tested also with binary features removed, listed as (cont) in Table 2.

Dataset	Source	Anomaly definition	Size	Anomalies	R	R(cont)
thyroid-ann	OML	[15]	3772	7.5%	21	6
cardiotocography	OML	[6]	1831	9.6%	35	24
ecoli	OML	[6]	336	3.0%	7	7
arrhythmia	OML	[12]	451	14.7%	275	192
diabetes	OML	[15]	768	35.1%	8	8
Ionosphere	OML	[18]	351	35.8%	33	32
ForestCover	UCI	[12]	286048	1.0%	10	10
breast-w	OML	[18]	683	35.1%	9	9
mammography	OML	[18]	11183	2.3%	6	6
satellite_image	OML	[18]	6435	31.6	36	36
optdigits	OML	[6]	5620	9.8%	64	64
smtp	UCI	[18]	95380	0.01%	3	3
spambase	UCI	[12]	3902	39.4%	57	57
pendigits	UCI	[12]	10992	10.4%	16	16

Table 1. Table of tested Open-ML (OML) and UCI datasets. R/R(cont) are the complete and continuous number of features respectively.

Every dataset has been randomly split into training and testing set ($70/30\%$). To enable a more robust statistical analysis, the function *StratifiedShuffleSplit* of `scikit-learn` has been used to test over 10 different train/test splits. By using the stratified version of the shuffle split each test set was constructed to contain the same ratio of anomalies as the whole dataset [19]. For the same reason, the anomalies were removed from the training set after the splits were made. Since each algorithm was trained in two different instances on each split, a total of 20 results was obtained for each dataset/algorithm combination. Statistical analysis on these were performed by comparing the algorithm with the highest average against the others with a pairwise Wilcoxon signed-rank test. Recall that this is a non-parametric test that, in our case, can reject a null hypothesis that the difference of the performance of two methods is symmetrically distributed around 0 in favour of an alternative hypothesis that it is not.

4.1.1 Complexity

As the tree resulting from Algorithm 1 has the same structure as the *iTree* of IF in [18], including the node operations, the time complexity during inference is also comparable to that of IF. The number of comparison operations is bounded by the maximum depth of the trees. Training complexity is also similar although somewhat higher due to the additional sorting in Algorithm 2.

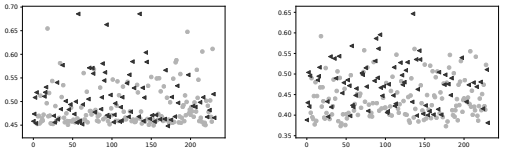
4.2 Standard settings

As default settings for ADF for the experimental results given in Table 2 we have used a subsample size $|X| = 256$, *anomaly margin* $a = 1$, *isolation level* $\eta = 0.1$ and *max depth* $D = 13$. The settings for the IF and OCRF were also chosen to be the default as specified in the respective papers [12, 18].

4.3 Results

As seen in Table 2 ADF shows the best results on a majority of the datasets both for ROC AUC and AP metrics. One of the few datasets with a significant decrease in performance compared to IF is the *pima/diabetes* dataset but as seen in Figure 5, it is questionable whether it is meaningful to grade performance between two methods for novelty detection at such a low level of utility. The other dataset that has better performance for both IF and OCRF (both AUC, AP) is the `breast-w` dataset. For this dataset the ADF lacks the power to distinguish some of the anomalies and fails to give high anomaly scores for these as seen in Figure 2. A failure to provide a higher anomaly score for these anomalies with ADF implies that the feature values of the anomalies are within the density distributions modelled. Analysis show that the `breast-w` training set contains a few samples with extreme values, and with such a small set (683) and large subsample size (256), this may reduce the possibility for ADF to give high anomaly scores to *actual anomalies* since anomaly catcher nodes look for anomalies outside the feature span for the node subsample. IF, on the other hand, assigns high anomaly scores to actual anomalies that have corresponding isolations in the training set and this is therefore a case where the previously mentioned weaknesses of IF is not a problem. There are some possible mitigations to improve ADF under such conditions, like reducing the subsample size, as seen in Figure 2 or normalizing training data in some fashion, but this is left for future work. OCRF shows the best results for some datasets, in particular *ForestCover* and *ann-thyroid*, but not so great results for the majority of datasets, possibly due to its strong requirements on continuity. ADF on the other hand, has the capability to handle all feature types and for the *cardiotocography* and *Ionosphere* datasets the addition of binary features give a boost in performance.

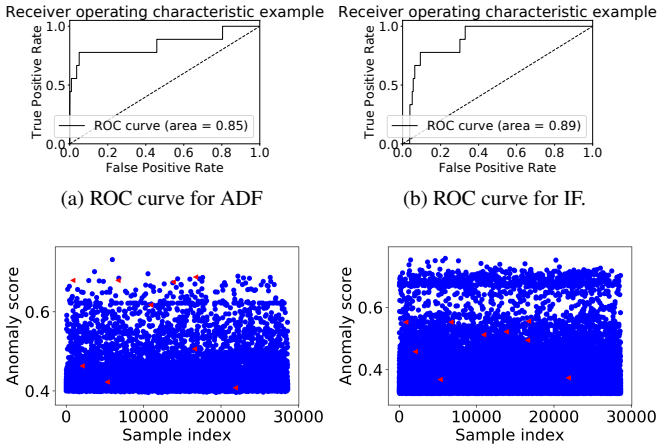
Compared to the IF, the anomaly scores coming from the ADF seems to be higher for anomalies with fairly high scores from IF



(a) ADF (b) IF

Figure 5. Anomaly scores for IF and ADF on the *pima/diabetes* dataset. Normal test instances in dots and anomalies with triangles.

while they are lower for anomalies with low scores as seen in Figures 6c and 6d. Given the use of thresholds outside the range of observed node samples, it is probable that the anomalies that got higher scores with ADF have more distinctly differing features from the training data whereas the anomalies that got lower scores are less separable from the training set. This increased separation of distinct anomalies with ADF seems to be useful as in Figure 6c, where at least four of the anomalies could be detectable without an overwhelming amount of false positives. However, it also highlights the general difficulty in defining relevant performance measures for novelty detection algorithms. The insufficiency in using ROC AUC as a performance indicator for anomaly detection is particularly striking in the results for the `smtp` set seen in Figure 6, where ADF ranks the top scoring anomalies higher (corresponding to the first jump to 0.4 TPR (*True Positive Rate = #Correctly identified anomalies / #Anomalies*) in Figure 6a) whereas the IF ranks the lowest scoring anomalies higher (corresponding to the jump to 1.0 in TPR in Figure 6b).



(a) ROC curve for ADF (b) ROC curve for IF.

(c) ADF scores. (d) IF scores.

Figure 6. Anomaly scores for *smtp* testset with $|X| = 256$, anomalies are red triangles.

The ROC AUC is shown to be significantly higher for the IF but as seen in the figure it does not seem to indicate that the method is better to use to detect the smtp anomalies. The problem is that ROC AUC does not differentiate between absolute anomaly ranking differences since it only measures the area under curve [1]. In many applications of anomaly detection with one-class learning, anomalies are rare, and, only the fraction of samples with the highest anomaly scores can be handled for further actions. In such cases, as illustrated above, the AP gives a more reliable indication of utility with its focus on the performance of the anomaly class. ADF shows particularly good performance in terms of AP as seen in Table 2.

Dataset	OCRF		ADF		SciKit_iForest	
	ROC	AP	ROC	AP	ROC	AP
thyroid-ann	N/A		$\mathbf{84.2 \pm 3}$	$\mathbf{32.2 \pm 5}$	78.2 ± 3	20.0 ± 2
thyroid-ann(cont)	$\mathbf{97.1 \pm 0}$	$\mathbf{63.9 \pm 5}$	94.2 ± 1	56.6 ± 5	93.0 ± 1	53.6 ± 5
mammography	81.7 ± 3	24.9 ± 5	$\mathbf{88.6 \pm 2}$	$\mathbf{35.0 \pm 5}$	87.9 ± 2	22.5 ± 5
optdigits	N/A		$\mathbf{89.8 \pm 2}$	$\mathbf{42.2 \pm 6}$	79.1 ± 4	22.2 ± 4
optdigits(cont)	49.0 ± 3	10.6 ± 1	$\mathbf{90.2 \pm 2}$	$\mathbf{43.2 \pm 6}$	81.3 ± 4	24.6 ± 5
cardiotocography	N/A		$\mathbf{99.8 \pm 0}$	$\mathbf{97.8 \pm 1}$	88.8 ± 3	51.1 ± 9
cardiotocography(cont)	90.9 ± 2	68.9 ± 3	$\mathbf{97.2 \pm 1}$	$\mathbf{81.5 \pm 5}$	93.7 ± 2	63.0 ± 7
diabetes	68.1 ± 3	51.3 ± 5	64.5 ± 3	50.4 ± 5	$\mathbf{72.2 \pm 3}$	$\mathbf{55.7 \pm 3}$
ForestCover	$\mathbf{98.5 \pm 0}$	$\mathbf{50.2 \pm 2}$	97.7 ± 0	20.6 ± 2	85.1 ± 4	4.3 ± 1
satellite_image	78.3 ± 1	75.3 ± 1	$\mathbf{82.3 \pm 1}$	$\mathbf{80.2 \pm 1}$	80.8 ± 2	77.8 ± 1
Ionosphere	N/A		$\mathbf{97.2 \pm 2}$	$\mathbf{95.5 \pm 3}$	92.4 ± 2	88.4 ± 3
Ionosphere(cont)	93.4 ± 3	83.4 ± 8	$\mathbf{97.0 \pm 2}$	$\mathbf{95.2 \pm 3}$	91.9 ± 2	87.8 ± 3
pendigits	88.2 ± 1	30.0 ± 2	$\mathbf{93.1 \pm 1}$	$\mathbf{47.7 \pm 5}$	78.9 ± 3	20.5 ± 2
smtp	$\mathbf{90.0 \pm 5}$	1.4 ± 1	84.9 ± 10	$\mathbf{12.0 \pm 8}$	89.1 ± 4	$.2 \pm 0$
breast-w	98.7 ± 1	96.3 ± 2	95.3 ± 4	93.0 ± 4	$\mathbf{99.5 \pm 0}$	$\mathbf{99.0 \pm 0}$
spambase	$\mathbf{85.8 \pm 1}$	76.1 ± 2	85.6 ± 1	$\mathbf{81.8 \pm 1}$	83.6 ± 2	75.9 ± 3
ecoli	N/A		82.1 ± 15	49.4 ± 23	$\mathbf{82.4 \pm 15}$	$\mathbf{54.0 \pm 26}$
ecoli(cont)	44.3 ± 8	3.8 ± 1	51.5 ± 14	$\mathbf{11.0 \pm 10}$	$\mathbf{51.7 \pm 11}$	8.4 ± 8
arrhythmia	N/A		$\mathbf{0.836 \pm 0.04}$	$\mathbf{0.568 \pm 0.06}$	0.835 ± 0.04	0.544 ± 0.08
arrhythmia(cont)	0.521 ± 0.07	0.158 ± 0.04	$\mathbf{0.845 \pm 0.04}$	$\mathbf{0.547 \pm 0.06}$	0.834 ± 0.04	0.518 ± 0.06

Table 2. ROC AUC and AP in percent for 20 runs (10 train/test splits and 2 training instances) on each dataset. Wilcoxon signed-rank test (significance level $\alpha = 0.05$) has been run pairwise for the method with the highest average score (boldface) versus the other methods and significance is indicated by gray background.

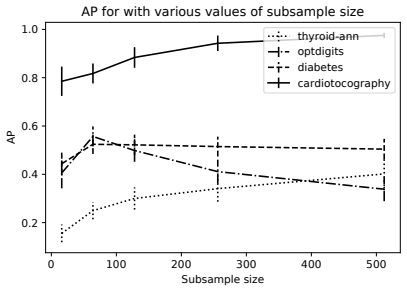
4.3.1 Subsample size

Figure 7 shows how the AP values vary with subsample size for standard settings as described above. For most of the datasets, increasing the subsample size from 16 provides a higher AP but levels out for a subsample size of 128-256. As a larger subsample size has a smoothing effect on the normal modeling as seen in Figure 1, anomalies that are surrounded by normal instances will not be likely to produce high anomaly scores for this case. In the case of the `breast-w` and `optdigits` datasets, AP values are best for small subsample sizes. As discussed in the properties of ADF, this is likely because the smoothing of a larger subsample size makes the border between anomalies and normal instances too indistinct. As seen in Figure 2, a smaller subsample size produces better performance for the `breast-w` dataset by more distributed scoring of anomalous and normal instances.

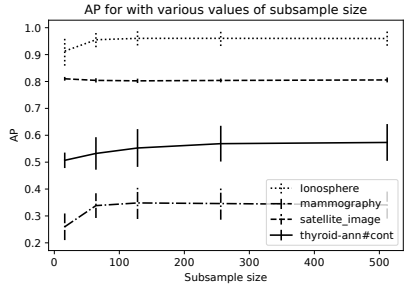
5 Conclusions

In this paper we have presented a new ensemble method for the one-class learning case of anomaly detection, based on random binary trees, called *Anomaly Detection Forest* (ADF). It is an adaptation of the isolation paradigm of the IF algorithm to work better for the one-class learning case. By introducing elements that reduce the risk for classifying uncommon normal instances as anomalies as well as adding elements that have a chance of detecting anomalies in features that are dense and stable in training data, we show considerable improvements both in ROC AUC and AP compared to IF as well as to OCRF. Another merit of ADF is that it can make use of binary features, such as one-hot-encodings.

For some applications, such as those that will send alarms for manual human inspection, anomaly detection may only be useful if it can keep the number of false positives very low. The ADF is particularly suitable when the inspection budget is limited, as it increases the relative scores of distinct anomalies at the expense of difficult cases. We argue that this is demonstrated more clearly by the AP performance metric, rather then ROC AUC that is commonly used for anomaly detection, in general.



(a)



(b)

Figure 7. AP for varying subsample size of ADF over the different datasets.

ACKNOWLEDGEMENTS

We would like to thank Luis Barriga, Julen Kahles, and Amine Boukhtouta for support and valuable comments during preparation of this manuscript.

REFERENCES

[1] Charu C. Aggarwal, *Outlier Analysis*, Springer Publishing Company, Incorporated, 2nd edn., 2016.
[2] Marcelo Bacher, Irad Ben-Gal, and Erez Shmueli, 'An information theory subspace analysis approach with application to anomaly detection ensembles.', in *KDIR*, volume 1, pp. 27–39, (2017).
[3] Stephen D Bay and Mark Schwabacher, 'Mining distance-based outliers in near linear time with randomization and a simple pruning rule', in *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 29–38. ACM, (2003).
[4] Markus M. Breunig, Hans-Peter Kriegel, Raymond T. Ng, and Jörg Sander, 'Lof: Identifying density-based local outliers', *SIGMOD Rec.*, **29**(2), 93–104, (May 2000).
[5] Raghavendra Chalapathy, Aditya Krishna Menon, and Sanjay Chawla, 'Robust, deep and inductive anomaly detection', in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 36–51. Springer, (2017).
[6] Jinghui Chen, Saket Sathe, Charu Aggarwal, and Deepak Turaga, 'Outlier detection with autoencoder ensembles', in *Proceedings of the 2017 SIAM International Conference on Data Mining*, pp. 90–98. SIAM, (2017).
[7] Yuting Chen, Jing Qian, and Venkatesh Saligrama, 'A new one-class svm for anomaly detection', in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 3567–3571. IEEE, (2013).
[8] Rémi Domingues, Maurizio Filippone, Pietro Michiardi, and Jihane Zouaoui, 'A comparative evaluation of outlier detection algorithms: Experiments and analyses', *Pattern Recognition*, **74**, 406–421, (2018).
[9] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017.
[10] Andrew F Emmott, Shubhomoy Das, Thomas Dietterich, Alan Fern, and Weng-Keen Wong, 'Systematic construction of anomaly detection benchmarks from real data', in *Proceedings of the ACM SIGKDD workshop on outlier detection and description*, pp. 16–21. ACM, (2013).
[11] Nicolas Goix. https://github.com/ngoix/ocrf, 2016.
[12] Nicolas Goix, Nicolas Drougard, Romain Brault, and Mael Chiapino, 'One class splitting criteria for random forests', in *Proceedings of the Ninth Asian Conference on Machine Learning*, eds., Min-Ling Zhang and Yung-Kyun Noh, volume 77 of *Proceedings of Machine Learning Research*, pp. 343–358. PMLR, (15–17 Nov 2017).
[13] Sudipto Guha, Nina Mishra, Gourav Roy, and Okke Schrijvers, 'Robust random cut forest based anomaly detection on streams', in *International conference on machine learning*, pp. 2712–2721, (2016).
[14] Sahand Hariri, Matias Carrasco Kind, and Robert J Brunner, 'Extended isolation forest', *arXiv preprint arXiv:1811.02141*, (2018).
[15] Fabian Keller, Emmanuel Muller, and Klemens Bohm, 'Hics: High contrast subspaces for density-based outlier ranking', *Proceedings - International Conference on Data Engineering*, 1037–1048, (04 2012).
[16] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou, 'Isolation forest', in *2008 Eighth IEEE International Conference on Data Mining*, pp. 413–422. IEEE, (2008).
[17] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou, 'On detecting clustered anomalies using sciforest', in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 274–290. Springer, (2010).
[18] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou, 'Isolation-based anomaly detection', *ACM Transactions on Knowledge Discovery from Data (TKDD)*, **6**(1), 3, (2012).
[19] 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, and E. Duchesnay, 'Scikit-learn: Machine learning in Python', *Journal of Machine Learning Research*, **12**, 2825–2830, (2011).
[20] Marco AF Pimentel, David A Clifton, Lei Clifton, and Lionel Tarassenko, 'A review of novelty detection', *Signal Processing*, **99**, 215–249, (2014).
[21] John A Quinn and Masashi Sugiyama, 'A least-squares approach to anomaly detection in static and sequential data', *Pattern Recognition Letters*, **40**, 36–40, (2014).
[22] Mei-Ling Shyu, Shu-Ching Chen, Kanoksri Sarinnapakorn, and LiWu Chang, 'A novel anomaly detection scheme based on principal component classifier', Technical report, MIAMI UNIV CORAL GABLES FL DEPT OF ELECTRICAL AND COMPUTER ENGINEERING, (2003).
[23] Aaron Tuor, Samuel Kaplan, Brian Hutchinson, Nicole Nichols, and Sean Robinson, 'Deep learning for unsupervised insider threat detection in structured cybersecurity data streams', in *Artificial Intelligence for Cyber Security Workshop (AAAI-2017)*. AAAI, (2017).
[24] Joaquin Vanschoren, Jan N. van Rijn, Bernd Bischl, and Luis Torgo, 'Openml: Networked science in machine learning', *SIGKDD Explorations*, **15**(2), 49–60, (2013).
[25] Xuyun Zhang, Wanchun Dou, Qiang He, Rui Zhou, Christopher Leckie, Ramamohanarao Kotagiri, and Zoran Salcic, 'Lshiforest: A generic framework for fast tree isolation based ensemble anomaly analysis', in *2017 IEEE 33rd International Conference on Data Engineering (ICDE)*, pp. 983–994. IEEE, (2017).