Universiteit Leiden

# Master Computer Science

Evolutionary Isolation Forest for Interactive Anomaly Detection in an Incremental Scenario

| | |
|---|---|
| Name: | Jichen Wu |
| Student ID: | 2074826 |
| Date: | 13/08/2020 |
| Specialisation: | Advanced data analytics |
| 1st supervisor: | Wojtek J. Kowalczyk |
| 2nd supervisor: | Bas van Stein |

Master's Thesis in Computer Science

Leiden Institute of Advanced Computer Science (LIACS)
Leiden University
Niels Bohrweg 1
2333 CA Leiden
The Netherlands

**Abstract**

Outliers in a dataset are points that statistically deviate from normal instances and anomalies are points that are abnormal based on some specific context or criteria. Unsupervised anomaly detection algorithms usually find statistical outliers. But in a realistic scenario, these outliers are not necessarily interesting or relevant anomalies. Limited availability of expert feedback is usually quite easy to obtain. Incorporating such feedback to adjust the detectors can significantly boost the performance. It discovers anomalies that are contextualized to the expert user's semantic understanding of the anomalies. The user then provides feedback on the more contextualized anomalies and the algorithm gives even more contextualized anomalies. This human-algorithm interaction loop is called Interactive Anomaly Detection. In this paper we make the first attempt to define Interactive Anomaly Detection in an incremental scenario. That is, data comes in streams and the more recent an anomaly is, the more relevant it is. We then propose a novel algorithm: the Evolutionary Isolation Forest. By using evolutionary algorithms to adapt the Isolation Forest to users' feedback, it outperforms the state-of-the-art method in the incremental scenario. We also show the realistic relevance of this algorithm by applying it to a real-world problem of detecting system disfunctions at energy infrastructures at the company WithTheGrid.

## 1 Introduction

The term "anomaly detection" usually refers to unsupervised detectors or classifiers that detect statistical outliers and predict that they are anomalies. The term is often used in the context of tabular data. In the context of streaming data, the problem is slightly different as both the data and the anomalies to be raised have a time aspect. Whenever new data comes in, the detector should make a prediction and the sequence (history data) may affect the prediction. Applications of anomaly detection in streaming data include electronic transactions, click streams, device monitoring, sales pattern, etc. However, as traditional anomaly detectors usually find statistical outliers in the data, they are not necessarily context-related. In real-world scenarios, we are often more interested in anomalies specified by the application, the users or the environment. For example, in sensor data, an extreme value could simply be noise and we are more interested in systematic deviations.

As such, a traditional method can produce many false alarms and a domain expert will have to manually pick out the "true anomalies" from the "false" ones. One way to get around this is to incorporate domain knowledge into the detector. However, besides the fact that the algorithm has been specially designed for the domain, it also requires knowledge from both data mining and that specific domain. This is not always feasible.

Fortunately, in a realistic scenario, a domain expert's knowledge is usually available to label a small part of the data. This leads to another approach: Incremental Learning [13, 4, 20]. When new data and labels become gradually available, the model is updated. However, Incremental Learning algorithms learn from new data passively, they do not ask the domain expert to label data and they do not efficiently use the labels to update the model. A more interactive and efficient structure is Interactive Anomaly Detection, which utilizes the feedback from the domain expert to improve the detector. The typical workflow is: (1) the detector shows $i$-top anomalies to the expert (2) the expert labels these anomalies (3) the detector learns from these anomalies and provides the next set of $i$-top anomalies to the expert. Such workflow is believed to significantly improve the detector's performance with minimal expert's effort. Among the state-of-the-art approaches, [7, 23, 15] all used ensemble detectors and use feedback to adjust the importance of each ensemble in the detectors. While [9] clustered data points and formed this as a k-armed bandit problem. However, these methods have three major drawbacks:

1. They are not suitable for a incremental scenario. Not only they do not take temporal information into account when dealing with time series data, but they are also not suitable for a incremental workflow, in which there is no concept of top-$i$ anomalies. The detector should query the expert for more recent data. Because the more recent the data is, the more interesting and relevant it is.

2. If the feedback does not come from a domain expert, but rather comes from multiple less-professional end-users, they have no means to incorporate different user's knowledge and have the detector adjusted to each user's preference.

3. The methods to adjust the detectors are relatively simple. If some information is missed by the detector in the beginning, the learning process cannot help the detector to retrieve such information.

As a real-life example, the company WithTheGrid(WTG) provide monitoring service for energy infrastructures with a variety of sensors. The purpose is to detect anomalies in these infrastructures, such as water pipe leakage, abnormal temperature and device misplacement. The sensors send data to the central processor several times a day and the central processor needs to react in real-time. In such a system, statistical outliers are very likely to be uninteresting as they can be caused by temporary sensor failures instead of real disfunction in the infrastructures. Also, the system is designed to face end-users and they have different preferences and sensitivities on anomalies. Since it is impossible to manually model each user's preference, it is crucial to incorporate user feedbacks automatically into the anomaly detection algorithm. Another difficulty is that there are many sensors of different types and we cannot assume that data from different sensors follow the same distribution. Thus, if we design a separate detector for each sensor, there will simply be too many detectors and the feedbacks will be too sparse for the algorithm to learn. If we design one detector for all sensors of the same type, then naturally the detector will be heavily biased as different sensors have a different distribution of data.

If we use existing methods in active anomaly detection[7, 23, 15] on the problem at WTG, there will be several problems. First, in streaming data, there is no concept of "top-$i$ anomalies", the algorithm has to react to new data in real-time. So it is more proper to raise anomalies with a certain threshold. Second, since there are so many sensors under different users, previous methods have no means to incorporate feedbacks of different sensors and users.

Inspired by the problem at Withthegrid and noticing the gap in the literature that no interactive anomaly detectors are designed to work with streaming data in a more realistic scenario, in this paper, we propose a novel algorithm: Evolutionary Isolation Forest. Similar to previous methods, we use an ensemble anomaly detector as the base detector. Specifically, it is Isolation Forest[16]. The novelty of the proposed method is that we use evolutionary operators to adapt the structure of each tree in such a forest to the feedbacks. Our hypothesis is that the use of evolutionary operators enables the forest to change to more complex structures, thus can handle more complex optimization problems. Such problems include working with streaming data and handling more complex user feedbacks. Also, as an evolutionary algorithm does not require the calculation of the gradient of the fitness/loss function, more sophisticated fitness functions can be used in a real-life environment. Additionally, we propose a location-based forest structure to incorporate feedbacks from different streams, environments and users.

The contribution of this paper is mainly in two aspects: First, it recognizes a gap in the field of anomaly detection, namely, interactive anomaly detection in streaming data. To the best of our knowledge, no previous literature has addressed this problem. We believe such a problem commonly exists in real-life applications, especially in the field of real-time surveillance systems. We use the problem at WTG as an example.

Second, beyond identifying the problem, we also make an attempt to solve the problem. To do so we propose an algorithm specifically designed for the problem: the Evolutionary Isolation Forest (EIF). Compared to previous approaches, we believe that EIF can adapt to more complex optimization problems with more complex fitness functions. This means EIF is more sophisticated in detecting contextualized anomalies. We show that in the classic top-$i$ scenario, EIF performs at the same level as existing algorithms. In the incremental scenario, however, EIF out-performs the state-of-art algorithms. Also, as a general anomaly detection method that is not specially designed for a specific domain, EIF, in theory, can be applied to any domain of anomaly detection.

The rest of the paper is organized as follows: in Section 2, related works are reviewed. In Section 3, we formally define the research problem. In Section 4, we introduce the EIF algorithm. In Section 5, we run experiments to compare EIF with other algorithms, as well as showing the implementation for WTG data. Finally, in Section 6, we conclude the paper with the contributions and possible future work.

## 2 Literature Review

Unsupervised anomaly detectors mainly operate by identifying statistical outliers in the dataset. Since anomalies are usually rare, data for anomaly detection tasks is usually unlabelled. Three main classes for anomaly detection techniques are classification-based, nearest neighbour based and clustering based [5]. Some examples for popular anomaly detection algorithms are: One Class SVM[22], which is a max-margin approach that separates novel points from high-density points, Isolation Forest[16], which is used in this paper and Autoencoder [21, 1], which reduces the dimensionality of the data so it is easier to find anomalies.

These above-mentioned methods are adapted to tabular data. However, in anomaly detection, streaming data is more common. That is, new data continuously comes in. For streaming data, temporal information is sometimes important, but there are not as many detectors for temporal data. Classic methods including STL[6], which decompose time series into trend, seasonality and noise, ARIMA[3], which predict the future based on the past data using auto regression and moving average, and Exponential Smoothing [10], which simply predict the future value by the sum of exponentially decreased weights on past data. These methods are closely related to time series forecasting. We refer to [2] for a review of newer and more sophisticated methods for time series anomaly detection. Classic anomaly detectors like Isolation Forest can also be used to handle time series if time-related features can be extracted from the data. However, most of these methods are completely unsupervised and, in many realistic scenarios, fail to achieve good accuracy [11].

In most anomaly detection applications, it is a realistic assumption that expert knowledge is available to some extent. As such, Active Learning[11, 17, 18] incorporates supervision into unsupervised methods. In a general framework, such a method first clusters all data points. Points with the least confidence, e.g. through Expectation Maximization, are provided to the expert for labeling. The labels are then used to update the clusters. Finally, the anomalies are rare clusters or points that are far from cluster centers. A closely related field is Rare Category Detection [18, 12], which, instead of identifying anomalies from normal data, identifies multiple categories. It also puts more cognitive burden on the expert. A limitation of such methods is that data to be queried is not necessarily the most interesting anomalies. Also, the goal of these models is to achieve good performance after queries with test data (while raising queries with train data). However, in many realistic scenarios, we do not have data and human resources to train the model before-handed. Rather, we want the model to learn on-the-run and queries presented to the expert should be the most anomalous points, not necessarily the most informative points.

To handle these limitations, the framework Top-1 Feedback is proposed, in which the expert only checks the most anomalous point at a time. Active Anomaly Discovery (AAD) algorithm [7] incorporated feedback from the expert into a LODA detector [19] or into an Isolation Forest (IF) detector [8]. Since both detectors are ensemble algorithms, AAD updates the model by using feedback to adjust weights attached to each ensemble. Later, Siddiqui et al. [23] proposed a simpler update rule and Lamba et al. [15] introduced cognitive burden as a new constraint. They introduced a "show more like this" mechanism so that the expert sees similar anomalies successively. Although most of these papers [8, 23, 15] used IF as the anomaly detector, the Top-1 Feedback framework is theoretically suitable for any ensemble-based detector. On the application end, Vercruyssen et al. [25] described in detail how to apply such a method for water analytics, with temporal information taken into account. Alternatively, Ding et al. [9] formed interactive anomaly detection as a cluster-based k-armed bandit problem.

To the best of our knowledge, the Top-1 Feedback framework is the state-of-the-art approach for incorporating expert's feedback into active anomaly detectors. However, it has several disadvantages. First, the above-mentioned methods either cluster data and attach anomaly score / reward to each data point / cluster [25, 9] or adjust relative importance of detectors in an ensemble [7, 8, 23, 15]. However, if the abnormal points have a complex pattern so that the initial clusters or ensemble detectors are unable to separate them from normal points, feedback will not help the improvement. Alternatively, we proposed to constantly change the structure of the ensemble detectors, so that the detectors can learn more complex abnormal patterns. Second, they do not have a workflow that is suitable for streaming data. In the data stream, since the size of data keeps increasing and the most interesting anomalies are the most recent anomalies, it is inappropriate to query the top-1 anomaly in history. Rather the detector should query for incoming data when it passes a certain threshold, while the objective is still to maximise the amount of "interesting" anomalies provided to the human given a certain budget. Thus, this paper proposes a novel method to update an ensemble detector, which is designed to fit the incremental workflow.

# 3 Problem Definition

In this section, we define the two problems of interests. The first is the traditional problem to detect anomaly in static data, as in [7, 23, 15]. This is described in Sub-section 3.1. The second is the novel problem that is to detect anomaly in streaming data, which, to the best of our knowledge, is not formally defined in the literature yet. This is described in Sub-section 3.2. In Sub-section 3.3, the special case of WTG multi-stream data is described. We believe that this special case can be generalized to many other real-life applications.

## 3.1 Static Scenario

In this scenario, there exists a fixed dataset and we want to find anomalous instances in this dataset as efficient as possible with the help of an expert's labeling. In order to do that, we apply the workflow called interactive anomaly detection. That is, each time the algorithm queries the expert with potential anomalies and the expert labels them. The model then learns from these labels and then queries the expert with more relevant anomalies. The key problem in this workflow and in this scenario is that how do we update the algorithm according to the expert's labels. In this section we do not propose a solution to this key problem (instead it is provided in Section 4), but give a background and context of this problem using pseudo-code.

Given a dataset $\mathcal{D}_s = \{X_1, ...., X_q\}$ in $d$ dimensions and the evaluation budget of the expert $b$. At each round $t = 1, ..., b$, an anomaly detector $M$ calculates the anomaly scores $\{s_1, ..., s_q\}$ and raised the data point with the highest score as an anomaly to the expert (excluding previous raised ones). The expert then gives a label $l_t$ to update $M$. The evaluation metric is the recall when budget $b$ runs out. The scenario is summarized in Algorithm 1.

## 3.2 Incremental Scenario

In this scenario, the data comes in as a stream and we want to find anomalous instances in this stream as accurate and as early as possible with the help of an expert's labeling. Similar to the static scenario, we apply the interactive anomaly detection. The major difference in the incremental scenario is that the more recent an instance is, the more interesting it is. If we simply use

**Algorithm 1** Static scenario

```
1: for $t = 1 : b$ do
2:     $\{s_1, ...s_q\} = M(\mathcal{D}_s)$
3:     Pick $X_k$ with the highest score (among points that have not been picked yet)
4:     Get label $l_t$ of $X_k$ from the expert
5:     update $M$ with $l_t$ (how $M$ is updated depends on the specific algorithm)
```

an algorithm to query the most anomalous instance in the history, it could be too old in the history to be interesting. The key problem in this workflow is again how to update the model based on expert's feedback. Notice that here we use the word Incremental Scenario instead of Streaming Scenario due to that we focus on the fact that data keeps coming in but less on the sequence or order (time-relevance) of the data. Next, we give a background and context of this problem using pseudo-code.

Given a dataset of streaming data $\mathcal{D} = \{X_1, ...., X_t, ....\}$ in $d$ dimensions and with $t$ denoting the current time, an anomaly detector $M$ provides an anomaly score for each history point $\{s_1, ...., s_t\}$. If the rank of $s_t$ is bigger than some threshold $\zeta$, $X_t$ is raised as an anomaly. A domain expert then looks into $X_t$ and provides a label $l_t$. Furthermore, if $X_t$ is indeed an anomaly but the model fails to detect it, there is still a possibility $p \in [0, 1]$ that the expert will provide label on this point. This reflects the realistic event that the algorithm fails to detect an anomaly which leads to profound consequence and then the expert will label it. For simplicity, we denote the label attached to each point with $\{l_1, ..., l_t\}$ where $l_t = -1$ means the label is not provided, 0 means it is normal while 1 means it is an anomaly. There is also the possibility to define importance of each anomaly by assigning a weight to an anomaly. This feature is not implemented in this paper, but can be easily incorporated by the fitness function we described in Section 4.5.

In reality, the domain expert could also look into the history data and find anomalies that are missed by the model. Also, the modal may be asked to provide new anomalies on history data after later feedbacks. However, typically the more recent an anomaly, the more relevant it is. For simplicity of evaluation, we assume that all interactions are with the current data. While interaction with history data is similar and could be inferred.

The model $M$ is updated each time a label $l_t$ is provided, the evaluation metrics are the overall precision and recall at some terminal time $T$. The scenario is summarized in Algorithm 2.

**Algorithm 2** Incremental scenario

```
1: for $t = 1 : T$ do
2:     $s_t = M(X_t)$, $l_t = -1$
3:     if rank of $s_t > \zeta$ then get $l_t$ from the expert
4:     else
5:         if $X_t$ is truly an anomaly then
6:             set $l_t = 1$ with probability $p$
7:     update $M$ with $l_t$
```

## 3.3 WTG Scenario

In the WTG scenario the dataset consists of $n_0$ time series $\{X^1_{1:t}, ..., X^{n_0}_{1:t}\}$and we want to raise anomalies from each time series. All of them have the same dimensionality. Each data stream is assumed to be uncorrelated with one another. Each data stream can be treated as a case in the incremental scenario. However, the key problem is that $n_s$ is so large and feedbacks are relatively sparse. This means it would be very inefficient to build a separate model for each time series. To deal with this, we need to find a way to interconnect models so that feedbacks are shared among them. This scenario is summarized in Algorithm 3.

# 4 Method

As a solution to the problems given in the previous section: how to interactively update an anomaly detector based on expert's labels, in this section, we propose the Evolutionary Isolation Forest (EIF). We use Isolation Forest as the base detector to raise anomalies. Because such a tree-based algorithm is easy to update and change. When labels are provided, evolutionary operators update the trees accordingly so that the trees are adapted to provide

**Algorithm 3** WTG scenario

```
1: for $t = 1 : T$ do
2:     for $i = 1 : n_s$ do
3:         $s_t = M_i(X_t^i)$, $l_t = 0$
4:         if rank of $s_t > \zeta$ then get $l_t$ from the expert
5:         else
6:             if $X_t$ is truly an anomaly then
7:                 set $l_t = 1$ with probability $p$
8:         update $M_i$ with $l_t$
9:         for $j = 1 : n_s \;\; j \neq i$ do
10:            update $M_j$ with $l_t$, based on the similarity between $X_{1:t}^i$ and $X_{1:t}^j$
```

more relevant anomalies.

In this algorithm, each tree is treated as an individual and various mutation and crossover operators are used on the population. We know from experiments that in such a way the structure of the trees constantly adapts to the feedbacks and is optimized to more complex behaviours than changing weights of algorithm ensembles. The core difference between EIF and Isolation Forest is that, Isolation Forest works with unlabeled data. However in our case the number of cases will be growing as well as some labels of "true anomalies" will be incrementally added. Consequently, EIF will be modifying the initial population of random trees in order to maximize the fitness function that measures the similarity between the "computed anomaly scores" and "true labels".

In the rest of this section, we first give a brief introduction to Isolation Forest and Evolutionary Algorithm and then describe the model using evolutionary algorithm terminologies: population initialization, mutation, crossover, selection and fitness function. Finally, we describe how to adapt this algorithm to a multi-stream scenario using a location-based structure.

## 4.1 Background

As the base detector for EIF, Isolation Forest is a tree-based unsupervised algorithm with random splits (branches). The idea is that anomalies are more easily separated from other instances. To construct an isolation tree, first a training dataset $\{X_i\}$ needs to be sub-sampled from the original dataset as experiments show that having the full dataset as the training set degrades the performance. Each tree starts with a root node $n_0$. Next, An attribute (dimension) $a_0$ is randomly chosen from the dimensionality of the data $\{1, ..., d\}$ and a split value $v_0$ is randomly chosen between the lower and upper bound of dimension $a_0$. Then, all points $\{X_i\}$ in the sub-set are fed into $n_0$, if $X_i^{a_0} < v_0$ , the point is fed into a newly created node $n_0.left = n_1$, fed into $n_0.right = n_2$ otherwise. Here $X_i^{a_0}$ denotes the $a_0$th entry of $X_i$. This is repeated until the number of data points fed to a node $n_k$ is smaller or equal to one or the path length between $n_k$ and $n_0$ reaches a height limit. Suppose the size of $\{X_i\}$ is $\psi$ ,this height limit is normally set as $log_2\psi$ since we are not interested in instances that have path lengths higher than average. Such a node $n_k$ is called a terminal node and it has a size $\beta_k$ which is the number of data points fed to the node. A non-terminal node is called a branch node. An isolation forest is then an ensemble of isolation trees. For a data instance, each isolation tree will produce an anomaly score by how deep this instance terminates in the tree and the size of the terminal node. The final anomaly score is the mean of scores from all trees. Please refer to the original paper [16] for more details of this algorithm.

Next, we give a brief introduction to the basic process of an evolutionary algorithm. Specifically, the evolutionary algorithms used in this paper can be classified to the sub-category: Genetic Programming as it is less of an optimization problem but more of a program-update problem. However, we also use operators that are commonly used in another sub-category, Evolution Strategy. To avoid complexity and possible confusion, we only refer to Evolutionary Algorithm throughout the paper.

Evolutionary Algorithm is a population-based algorithm and a population needs to be initialized. A population consists of one or many individuals. Each represents a solution to the problem. In our case, each individual is an Isolation Tree. Each individual also has an mutation rate $\sigma$, controls how rapid an individual changes. At each iteration, offsprings are created through crossover and mutation. In crossover, two or more individuals are selected from the population. An offspring is the combination of the parents. In mutation, the attributes of an individual is randomly changed to create an offspring. Typically, through crossover and then mutation (sometimes only mutation), a number of offsprings is generated. A fitness value needs to be calculated for all individuals from the old population and the offsprings. This fitness value represents how good an individual is at solving the given problem. The new population is generated from a selection of the old population and the offsprings (sometimes from only the offsprings). Such selection is based on the fitness value. Typcially, $\mu$ is used to represent the size of the old population and $\lambda$ is used to represent the size of the offsprings. $(\mu + \lambda)$ indicates a selection strategy that involves both the old population and the offsprings. While $(\mu, \lambda)$ indicates a selection strategy that only involves the offsprings. For those who are interested in more details about Evolutionary Algorithm, we recommend the book by J.R. Koza [14].

## 4.2 Population Initialization and Selection Strategy

The initialization of the population is very similar to a standard IF algorithm. We create a forest of $m$ trees: $\{N_1, ....N_m\}$ with the given dataset. We define a base mutation rate $\sigma$ and a learning rate $\gamma$. Then, a tree $N_i$ also has an initial mutation rate $\sigma_i = \sigma * e^{rand()}$ where $rand()$ is drawn from $\mathcal{N}(0, 1)$.

For selection, we apply the greedy strategy, that is, rank all individuals by fitness and pick the top $m$ individuals as the next generation.

## 4.3 Mutation Operators

To mutate a tree $N_i$ to create a new tree $N_i'$, the mutation operator first changes the mutation rate: $\sigma_i' = \sigma_i * e^{\gamma * rand()}$. Next, for each branch node $n_k$, its attribute $a_k$ changes to a random dimension of the data with probability $\sigma_i'$ and when this happens, $v_k$ is drawn between the lower bound and the upper bound of the new dimension. If this does not happen, $v_k$ changes according to: $v_k' = v_k + \sigma_i' * rand() * (ub_{a_k} - lb_{a_k})$ where $ub_{a_k}$ and $lb_{a_k}$ represent the upper bound and lower bound of dimension $a_k$ respectively.

After the mutation of each branch node, for each terminal node $n_k$, either (1) the whole tree is re-trained through the standard IF training process and $\beta_k$ is reassigned or (2) $\beta_k$ is mutated according to $\beta_k' = \beta_k + \sigma_i' * rand()$. The mutation operators are summarized in Algorithm 4.

**Algorithm 4** Mutation operators

```
    Input $N_i, \sigma_i$
    Output $N_i, \sigma'_i$
 1: $\sigma'_i = \sigma_i * e^{\gamma * rand()}$
 2: for each node $n_k$ in $N_i$ do
 3:     if $n_k$ is a branch node then
 4:         if $rand() < \sigma'_i$ then
 5:             randomly select $a_k$ from $1, ...., d$
 6:             randomly draw $v_k$ from $(lb_{a_k}, ub_{a_k})$
 7:         else
 8:             $v_k = v_k + \sigma'_i * rand() * (ub_{a_k} - lb_{a_k})$
 9:     else
10:         do one of the two followings:
11:         (1) $\beta_k = \beta_k + \sigma'_i * rand()$
12: (2) Feed a sub-dataset to $N_i$ and obtain $\beta_k$ for all terminal node $n_k$
```

## 4.4 Crossover Operator

For crossover, two random parents are selected from the population, one branch node is randomly selected from each parent and the selected branch node from the second parent and all following nodes are used to replace the selected branch node and all following nodes of the first parent to create a child. An illustration is shown in Figure 1. The possibility is set so that the crossover point is more likely to be close to the branch node. There is also a chance that the crossover is not performed. The crossover operator is explained in detail in Algorithm 5.

## 4.5 Fitness Function

For the two problems defined in Section 3, we designed two fitness function. For static data, inspired by [15], we used the cross entropy loss. Each time when a new feedback $l_t$ is received on point $X_i$ with score $s_i$, we pair $X_i$ with a number of points which have scores $S_t = \{s_k\}$. Suppose the desired probability of $s_i > s_k$ is $p_{ik}$, the fitness function is then:

$$f_s = \sum_{s_k \in S_t} p_{ik} log(\hat{p}_{ik}) + (1 - p_{ik}) log(1 - \hat{p}_{ik}) \tag{1}$$

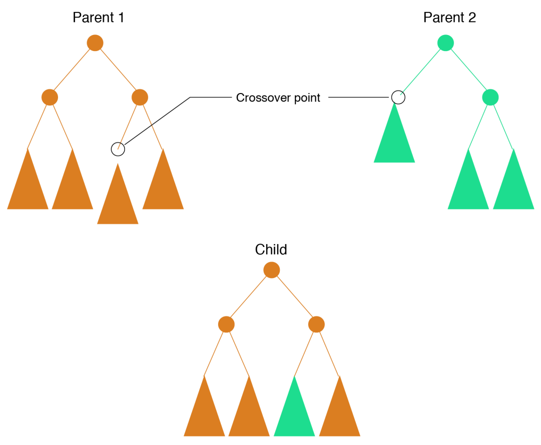

Figure 1: The crossover operator. Each filled circle represents a node and each triangle represents a tree structure.

where:

$$\hat{p}_{ik} = \frac{e^{s_i - s_k}}{1 + e^{s_i - s_k}}$$

We retrieve $S_t$ with two methods. The prioritized method is the set of all history points with an opposite label as $l_t$. If $k$ is retrieved from history, then $p_{ik} = 1$ if $l_t = 1$, $p_{ik} = 0$ otherwise. At the early stage, when there are not enough points in the history, $k$ is sampled from the whole dataset with probability proportional to $1/s_k$ if $l_t = 1$, or with probability proportional to $(-0.99 * s_k + 1)^{1/-0.99}$ if $l_t = 0$. If $k$ is retrieved by sampling, then $p_{ik} = \hat{p}_{ik} + 0.1$ if $l_t = 1$, $p_{ik} = \hat{p}_{ik} - 0.1$ otherwise.

For streaming data, we use a slightly different approach. In the fitness function for static data, the detector is encouraged to detect the most anomalous point that is similar to the last anomaly or opposite to the last nominal. However, in streaming data, the detector would not know if the next anomaly is similar to the last anomaly or to other anomalies deep in history. Thus the fitness function for streaming data should encourage the detector to assign high scores to all history anomalies. Suppose $S_p = \{s_j\}$ denotes the set of all anomalies in the history. While $S_n = \{s_k\}$ denotes the set of all nominals in the history. The fitness function is then"

$$f = \sum_{s_j \in S_p} \sum_{s_k \in S_t} p_{jk} log(\hat{p}_{jk}) + (1 - p_{jk}) log(1 - \hat{p}_{jk}) \qquad (2)$$

**Algorithm 5** Crossover operator

```
    **Input** parents $N_i, N_j$, crossover rate $\eta$
    **Output** children $N_o$
 1: $N_o = N_i$
 2: $n = N_j.n_0$
 3: crossover possibility $p_c = \eta$
 4: crossover point $N_j.cp = null$
 5: while $n.left$ exist & $N_j.cp$ does not exist do
 6:     if $rand() < p_c$ then
 7:         $N_j.cp = n$
 8:     else
 9:         $n = n.left$ or $n.right$ with equal possibility
10:         $p_c = p_c * (1 - \eta)$
11: $n = N_o.n_0$
12: crossover possibility $p_c = \eta$
13: crossover point $N_o.cp = null$
14: while $n.left$ exist  $N_o.cp$ does not exist do
15:     if $rand() < p_c$ then
16:         $N_o.cp = n$
17:     else
18:         $n = n.left$ or $n.right$ with equal possibility
19:         $p_c = p_c * (1 - \eta)$
20: $N_o.cp = N_j.cp$
```

The value of $p_{jk}$ and how to sample $s_j, s_k$ when there are not enough points in history is the same as in the static method.

## 4.6 Location Based EIF

To deal with the problem mentioned in Section 3.3, that is, there are multiple streams in which anomalies need to be detected independently and the feedbacks are sparse, we propose the Location-based EIF.

Given the $n_0$ streams in the environment, we put each streams in a feature space as a point. This space is defined relatively arbitrarily. For example, we can define a three dimensional space and the location of each stream is the mean value, standard deviation and sensor type. However, the distance between streams should indicate their similarity as much as possible. A distance matrix $D$ is then calculated that contains distances between all pairs of streams. The rows and columns of $D$ are both $n_0$. Next, isolation trees are assigned to each stream. We use a procedure so that similar streams share more trees together. The initialization algorithm is described in Algorithm 6 and the anomaly detection algorithm is described in Algorithm 7.

**Algorithm 6** Location-based Initialization

```
    Input Streams $X^1, ..., X^{n_0}$, distance threshold $\zeta$, number of trees $m$
    Output Model $M$ and indexing $I$, tree count $c$
 1: Initialize $I$ as a $n_0 * 0$ matrix ($I_i$ is the $i$-th row while $I_{:,i}$ is the $i$-th column)
 2: construct $D$ based on the distance between streams
 3: tree count $c = 0$
 4: while $sum(I_i) < m \ \ \forall i \in \{1, ..., n_0\}$ do
 5:     for $i = 1 : n_0$ do
 6:         if $sum(I_i) < m$ then
 7:             initialize tree $N_c$ with $X^i$
 8:             add $N_c$ to $M$
 9:             initialize zero vector $v$ of length $n_0$
10:             for $j = 1 : n_0$ do
11:                 $v_j = 1$ if $D_{j,i} < \zeta$
12:             add $v$ as a column of $I$
13:             $c = c + 1$
```

**Algorithm 7** Location-based Anomaly Detection

```
    Input point in Streams $X^i_t$, model $M$, indexing $I$
    Output anomaly score $s$
 1: tree indices = $I_i$
 2: $s = M(X\hat{\imath}_t, I_i)$ where $I_i$ is used to select a sub-group of the ensembled forest.
```

Due to this special ensemble structure, the population-based evolutionary operators mentioned in the preceding sections cannot be directly used. Because each stream and thus each feedback is attached to a different sub-group of the entire population. To handle this, instead of updating the population globally with a $(\mu + \lambda)$ selection algorithm, we update each tree locally with a $(1 + \lambda)$ algorithm. Suppose $S_p^i$ denotes all labeled anomalies in the $i$-th stream and $S_n^i$ denotes all labeled nominals in the $i$-th stream, the evolutionary operators are summarized in Algorithm 8. Additionally, we give a possibility $p_r$ that a random new tree is added into the local population to avoid staying in local optimization.

**Algorithm 8** Location-based Updating

```
    Input model $M$, indexing $I$, labels $S_p$, $S_n$, tree count $c$, population $\lambda$
    Output $M$
 1: for tree $N_j$ in $\{N_1, ..., N_c\}$ do
 2:     indexing $v = I_{:,j}$
 3:     local labels $S'_p = \{\}$, $S'_n = \{\}$
 4:     for non zero entry $k$ in v do
 5:         add $S_p^k$ to$S'_p = \{\}$, add $S_n^k$ to$S'_n = \{\}$
 6:     local population = $\{N_j\}$
 7:     for count = 1:$\lambda$ do
 8:         random selected $N_k$ from global population that has at least one
    stream in common with $N_j$
 9:         if rand() > $p_r$ then
10:             $N'$ = crossover($N_j$, $N_k$)
11:             $N'$ = mutate($N'$)
12:         else
13:             $N'$ is random initialized
14:         add $N'$ to local population
15:     calculate fitness of local population with Equation 2 and labels $S'_p$,
    $S'_n$
16:     relpace $N_j$ with the best fitted individual in the local population
```

## 5 Experiments

In this section, we first describe some details of experimental setup in Section 5.1. We tested our proposed algorithm under two different problems, as specified in Section 3. In Section 5.2, it is tested with static data. In Section

5.3, it is tested with streaming data. In both Section 5.2 and 5.3, we use banchmark datasets from ODDS[1] which is commonly used in anomaly detection researches. Each dataset in ODDS is multi-dimensional point dataset (in opposite to time-series). More details of the datasets will be given later. In Section 5.4, we describe the dataset from the company WTG and performs some preliminary experiments on this dataset.

## 5.1 Experimental Setup

For each tabular dataset or stream, we use a population of 100 trees and create 300 offsprings in each iteration. One and only one iteration is performed after each feedback is provided. However, for Location-based EIF, where trees are evolved locally, 1 offspring is created for each tree in one iteration and three iterations are performed with one feedback. For the mutation operators, we mutate $\beta_k$ by $\beta'_k = \beta_k + \sigma'_i * rand()$ in Section 5.2 and 5.3 as it seems to have better performance. In Section 5.3, however, we retrain each tree after mutation. This is to guarantee that anomaly scores given by different groups of trees can be compared. The initial mutation rate is $\sigma = 0.25/\sqrt{m} * e^{rand()}$ and the learning rate is $\gamma = 1/\sqrt{m}$.

The program is implemented in *python* with common libraries, e.g. *numpy*. We provide our code online[2] for reproducibility.

## 5.2 Static Scenario

We use 16 benchmark datasets from ODDS and compare the performance with AAD[7] and OJRank[15]. A statistic summary of the datasets is listed in Table 1. On each dataset, we use a budget of 200 and calculate the average results from 10 runs. We compared the recall over all 200 feedback rounds and the results are shown in Figure 2. Notice that we omitted the result of Breastw because the algorithms have almost the same performance.

We can see that EIF performs around the same level as AAD and OJRank. However, notice that for a few datasets (e.g. Optdigits and Glass), OJRank clearly outperforms EIF and AAD and we think OJRank is still in general a

[1]`http://odds.cs.stonybrook.edu/`

[2]https://github.com/z841860053/Evolutionary-Isolation-Forest/tree/master

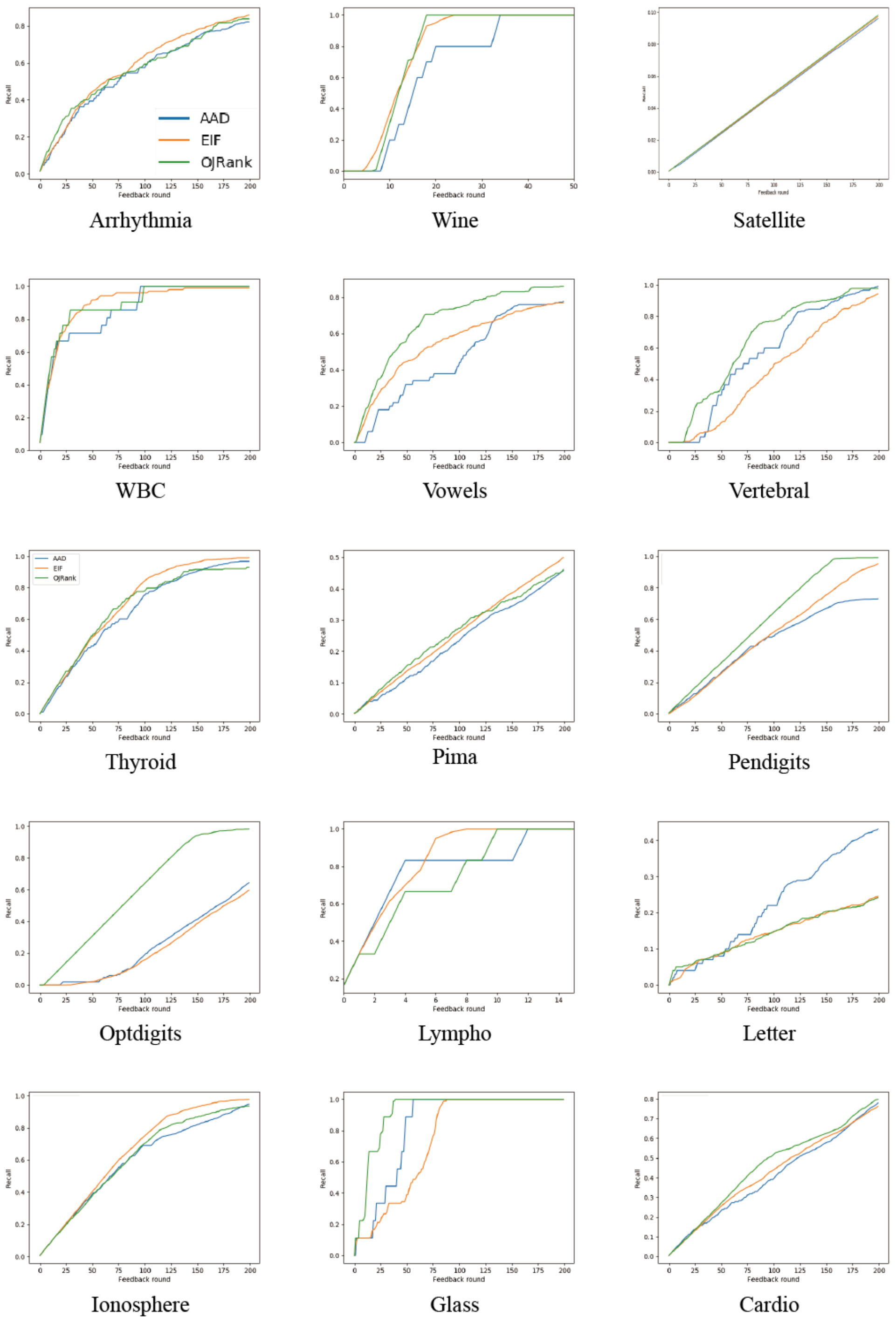

Figure 2: Results on 15 benchmark datasets. X axis is the feedback round. Y axis is recall.

| Name | Points | Anom. % | Name | Points | Anom. % |
|---|---|---|---|---|---|
| Arrhythmia | 452 | 15 | Pendigits | 6870 | 2.27 |
| Wine | 129 | 7. | Optdigits | 5216 | 37 |
| Satellite | 6435 | 32 | Lympho | 148 | 4.1 |
| WBC | 278 | 5.6 | Letter | 1600 | 6.25 |
| Vowels | 1456 | 3.4 | Breastw | 683 | 35 |
| Vertebral | 240 | 12.5 | Ionosphere | 351 | 36 |
| Thyroid | 3772 | 2.5 | Glass | 214 | 4.2 |
| Pima | 768 | 35 | Cardio | 1831 | 9.6 |

Table 1: Statistic summary of the ODDS benchmark datasets used for static data experiments

better choice for problems with static data. The major reason for the slightly worse behaviour of EIF is, we think, the fact that in the problem of static data, the algorithm only needs to provide the current most anomalous point. It would not matter if the history anomalies have low scores. This means the optimization problem is relatively simple and it is likely that simply changing the weights of the nodes with gradient descent can already find a good optimal. At the same time, adding evolutionary operators that change the structures of the trees might not add extra value to this optimization problem.

## 5.3 Incremental Scenario

For the incremental scenario, we use 15 point datasets from ODDS to simulate streaming. To do this, a dataset is randomly shuffled and the data points are fed to the algorithm one by one. These datasets are slightly different than that in Table 1 and they are listed in Table 2. We compare EIF with OJRank that uses loss function of Equation 2. Even though the original OJRank does not support the incremental scenario, changing the original loss function to Equation2 let it adapt to the incremental scenario. Both algorithms do not detect anomalies in the first 256 data points and only use them to train the Isolation Forest. For convenience, we set the detection threshold $\zeta$ to the percentage of anomalies in the dataset, but no bigger than 5% as we have limited computational power. Also the probability $p$ that the expert looks at an undetected anomaly is 0. We run 10 times on each dataset and calculate the average recall and precision. Furthermore, at the end of each run, the algorithms calculate scores for all points in the dataset and we count the percentage of true anomalies in the top $n$ and $2n$ points, with $n$ being the total number of anomalies in the dataset. This extra information can help us understand how well the algorithms can find anomalies in history data and how well they fit the overall dataset. The results are shown in Table 3.

| Name | Points | Anom. % | Name | Points | Anom. % |
|---|---|---|---|---|---|
| Annthyroid | 7200 | 7.42 | Arrhythmia | 452 | 15 |
| Cardio | 1831 | 9.6 | Ionosphere | 351 | 36 |
| Letter | 1600 | 6.25 | mammography | 11183 | 2.32 |
| Mnist | 7603 | 9.2 | Musk | 3062 | 3.2 |
| Pendigits | 6870 | 2.27 | Pima | 768 | 35 |
| Satellite | 6435 | 32 | Satimage-2 | 5803 | 1.2 |
| Speech | 3686 | 1.65 | Thyroid | 3772 | 2.5 |
| Vowels | 1456 | 3.4 | | | |

Table 2: Statistic summary of the ODDS benchmark datasets used for streaming data experiments

We can see that compared to OJRank, EIF generally has superior recall while having inferior precision. This means that, even though the threshold $\zeta$ is set to be the same, EIF queries more instances than OJRank. This could simply imply that at each query, the update of EIF changes much tree structure, resulting in new data points be ranked as top anomalies. For the percentage of true anomalies in top $n$ and $2n$ points, EIF performs better on more datasets. This confirms our hypothesis, that since EIF constantly changes the structure and values of the trees instead of simply adjusting weights of each node, it fits better to the terrain of the whole dataset.

Next, we study the effect of the probability $p$ that the expert founds an undetected anomaly. Our hypothesis is that the higher $p$ is, the better the model will perform, especially at early runs. To test it, on three datasets Thyroid, Mammography and Pendigits, we run the algorithm once on each $p$ value between 0 and 1 with a step of 0.1. Thus 11 runs in total. For each run, we calculate the recall of the entire run. We then calculate the correlation

|  | EIF |  |  |  | OJRank |  |  |  |
|---|---|---|---|---|---|---|---|---|
| Dataset | Precision | Recall | Top $n$ | Top $2n$ | Precision | Recall | Top $n$ | Top $2n$ |
| Annthyroid | 0.72 | 0.49 | 0.64 | 0.73 | 0.52 | 0.34 | 0.38 | 0.50 |
| Arrhythmia | 0.54 | 0.20 | 0.40 | 0.56 | 0.61 | 0.10 | 0.35 | 0.52 |
| Cardio | 0.74 | 0.39 | 0.57 | 0.69 | 0.81 | 0.38 | 0.53 | 0.61 |
| Ionosphere | 1.00 | 0.13 | 0.75 | 0.97 | 0.92 | 0.06 | 0.57 | 0.90 |
| Letter | 0.13 | 0.11 | 0.16 | 0.19 | 0.10 | 0.07 | 0.13 | 0.18 |
| Mammography | 0.45 | 0.39 | 0.39 | 0.42 | 0.38 | 0.38 | 0.39 | 0.49 |
| Mnist | 0.71 | 0.34 | 0.51 | 0.63 | 0.51 | 0.27 | 0.37 | 0.50 |
| Musk | 0.88 | 0.99 | 1.00 | 1.00 | 0.93 | 0.92 | 1.00 | 1.00 |
| Pendigits | 0.67 | 0.78 | 0.83 | 0.86 | 0.77 | 0.74 | 0.81 | 0.85 |
| Pima | 0.67 | 0.12 | 0.49 | 0.73 | 0.55 | 0.05 | 0.47 | 0.78 |
| Satellite | 0.99 | 0.12 | 0.53 | 0.76 | 1.00 | 0.16 | 0.46 | 0.70 |
| Satimage-2 | 0.74 | 0.89 | 0.88 | 0.90 | 0.82 | 0.84 | 0.88 | 0.94 |
| Speech | 0.02 | 0.03 | 0.04 | 0.05 | 0.04 | 0.04 | 0.06 | 0.07 |
| Thyroid | 0.72 | 0.76 | 0.79 | 0.88 | 0.75 | 0.74 | 0.73 | 0.83 |
| Vowels | 0.25 | 0.34 | 0.38 | 0.44 | 0.46 | 0.41 | 0.49 | 0.61 |
| Count. Better | 6 | **12** | **10** | **9** | **9** | 3 | 2 | 5 |
| Count. Dominate | **4** |  |  |  | 2 |  |  |  |

Table 3: Results of EIF and OJRank on 15 benchmark datasets. Count. Better is the number of times one algorithm is better in one attribute. Count. Dominate counts the number of times one algorithm is better than or equal to another in all attributes

between $p$ and recall using Kendall's Tau. Finally, we plot the change of the correlation values over time (percent of data seen). The result is shown in Figure 3. We can see that the results depend on the datasets. For Pendigits, the correlation is relatively high at all times, while for Mammography, the correlation is relatively low. Even though both Pendigits and Mammography's correlation reduces slightly with time, only Thyroid's correlation drops significantly. However, all three datasets show a dropping trend. This matches our original hypothesis, that a high $p$ will lead to good performance in the beginning, but would not matter as much in the long run. However, notice that due to limitation on computational power, on each $p$ we run only

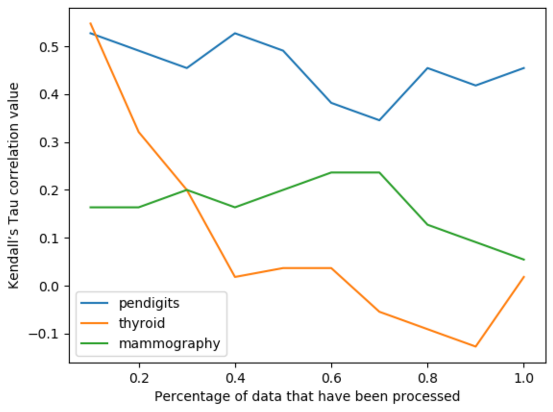

Figure 3: Kendall's Tau value between recall and $p$ over time.

once, this might leads to quite noticeable instability on the results.

Then, we study how the choice of threshold $\zeta$ affects the performance. Only one dataset: Thyroid with an anomaly percentage of 2.5% is used and 5 different $\zeta$ : $0.02, 0.04, 0.06, 0.08, 0.1$ are tested. On each $\zeta$ we run 5 runs and calculate the average. The recall and precision are shown in Figure 4. As one would expect, the higher threshold leads to higher recall and lower precision. However, interestingly, as shown in Table 4, after the whole dataset is processed, the percentage of top anomalies does not seem to correlate with the threshold. Even though one would assume that a higher threshold means more feedback to the algorithm and the algorithm should be improved better. This, together with our experiments on $p$, might imply that after a certain amount of feedbacks are received, extra feedbacks will have little impact on the further improvement of the algorithm. This means two things: (1) EIF does not need very dense feedbacks (2) For EIF there is space for improvement so that it can incorporate more feedbacks.

## 5.4 WTG Scenario

In this section, we first describe the data and preprocessing steps in Section 5.4.1 and then the implementation and preliminary results in Section 5.4.2.

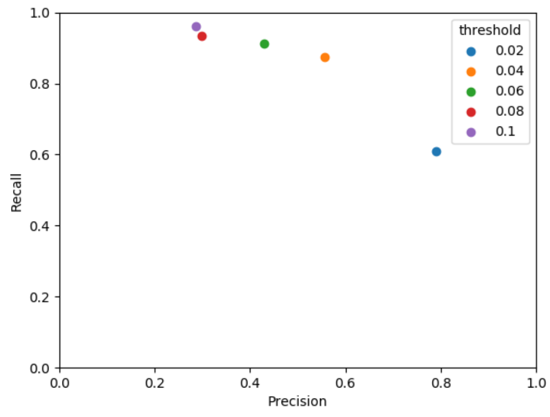

Figure 4: Recall and precision of different choices of $\zeta$.

| Threshold | Top $n$ | Top $2n$ |
|---|---|---|
| 0.02 | 0.76 | 0.88 |
| 0.04 | 0.82 | 0.96 |
| 0.06 | 0.78 | 0.92 |
| 0.08 | 0.77 | 0.98 |
| 0.1 | 0.77 | 0.95 |

Table 4: Percentage of true anomalies in top $n$ and $2n$ ($n$ is the number of total anomalies in the dataset) after the whole dataset is processed

### 5.4.1 Data Description and Preprocessing

The data comes from monitoring energy infrastructure assets. New streaming data coming around every hour. For the purpose of experiments, throughout this paper, we use data span from 2018-05-31 to 2020-06-25. In the dataset, there are 1822 measurement groups, which are called ”pins”. Each pin contains one or several sensors that monitoring activities at the same physical location. There are 4276 sensors measuring 52 different quantities, including voltage, insulation resistance, loop resistance, device temperature, etc. Successive data sent by one sensor is called a signal. Signals in the dataset have various periods even though the most common period is around 6 hours and coming after it is 1 hour.

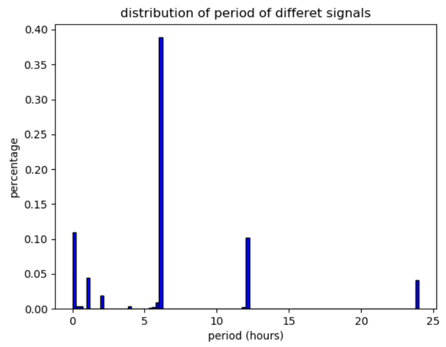

Figure 5: Distribution of dominant periods over all signals in the dataset.

The period for a signal is also not stable. For example, for all signals with a 6-hour period, around 78% of all intervals is actually 6 hours. Around 6% intervals are less than 10 minutes and a few points have negative intervals. These may be caused by the sensor sending a reading twice. Less than 1% intervals double the major interval. To handle these, we discard all readings that have intervals less than 1% of the dominant interval. We do not interpolate longer intervals as this will create artificial data. Instead, we take the time difference into account when calculating features from the time series, as described later in this section. For each signal, we calculate a dominant period and we plot the distribution of the dominant periods over all signals in the dataset in Figure 5. Nearly 40% of signals has a 6-hour period. The second and third common periods are less than 10 minutes and 12 hours respectively. 1-hour and 24-hour period is also common in the dataset.

Next, we inspect the patterns of different signals. There are two distinct patterns. In the first the signal jumps back and forth from two or three values and is in general more stable, for example, in loop resistance(lusweerstand). In the second the signal has higher accuracy and is less predictable, for example, in insulation resistance(isolatieweerstand). In Figure 6, we show signals from 12 most common quantities in the dataset. one signal is chosen randomly from each quantity.

One would assume that time series from the same quantity are very likely to follow the same distribution. However, we find out this is not the case. For example, for the most common quantity: insulation resistance, there are in total 897 time series. We run the Kolmogorov–Smirnov test [24] on random

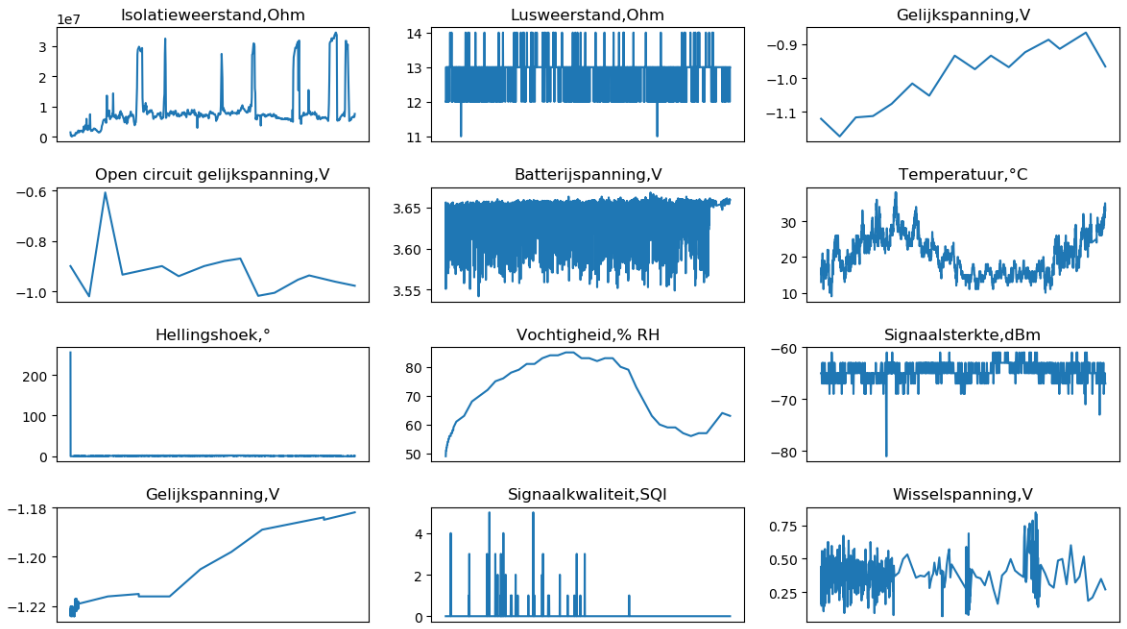

Figure 6: Example signals of 12 most common quantities. Y-axis is the value from the sensors. It varies depends on the sensor type.

100 pairs of these time series and the average p-value is only 0.052. This means for most pairs with $p < 0.1$ we reject the null hypothesis that the two data samples come from the same distribution. This means we cannot simply use one model for each quantity and thus Location-based algorithms come handy.

To be able to evaluate the algorithm, we run the original Isolation Forest model on the data and ask an expert to label top anomalies raised by the original Isolation Forest. We collect 40 labels from the domain expert during the period 2020-4-16 to 2020-6-26. Among them, 26 are true anomalies. The label density is 0.303% for this period.

In the final step of preprocessing, we extract 4 features from each time point in a stream. These will be the features fed to the EIF model. The features are: 1. first order deviation: the reading from the current time subtracts the reading from the previous time point, divided by time difference 2. second order deviation 3. mean value of the past 5 periods 4. standard deviation of the past 5 periods.

### 5.4.2 Results

We compare the proposed algorithm: Location-based EIF to two baselines: Location-based Isolation Forest (IF) and naive IF (that each stream has a separate forest). We expect Location-based IF to have similar performance as naive IF but with a lower computational cost. While the location-based forest should make it easier for EIF to learn so Location-based EIF should have better performance than both baselines.

For each above-mentioned algorithm, all data before date 2020-4-16 is fed as training data for IF. Then, batches of streams are fed to the algorithm on a daily basis. Then the algorithm ranks each instance in the streams by its anomaly score. We want labeled anomalies to be ranked as high as possible while labeled normal instances to be ranked as low as possible. For EIF, after anomalies are raised each day, feedbacks on the labeled instances are provided. Notice that this experimental workflow is slightly different than in a realistic workflow. As in a realistic workflow, feedback is usually only provided for the top anomalies. While in this experimental workflow, since we pre-collected the labels, labels can be attached to instances that rank low. Nevertheless, such an experiment should indicate the performance of EIF.

We run each algorithm for 10 times. The results are shown in Figure 7 and Table 5. From Figure 7, we can see that with naive IF, the distributions of anomalies and normal instances are hardly separable. With Location-based IF, the distribution of normal instances moves slightly backward. With Location-based EIF, anomalies form a peak at rank 0-5 while normal instances form a peak around rank 30. Additionally, after all labels are provided to Location-based EIF, it is asked to re-rank all anomalies in history. This indicates the performance of the model when encountering similar instances in the future. The results show that Location-based EIF adapts pretty well to the labels. Figure 7 shows the distribution over all ranks. However, in a realistic scenario, we are usually more interested in top ranks. Table 5 shows the false alarm rates for top 1, top 5 and top 20 instances over 3 different algorithms. Location-based EIF is clearly superior to the baselines, with one exception: higher top 1 FAR compare to naive IF. This is very likely due to the fact that the Location-based structure mixed trees for different streams that shouldn't be mixed. For example, two streams that have similar instance distributions but anomalies expected from them are quite different. However, for lower computational cost and more efficient updates, such a trade-off has to be made. For this particular dataset, anomaly detection for a day's data costs roughly a matter of seconds and a complete update of all trees costs a matter of minutes on one core of a standard personal computer.

| Algorithm | Top 1 FAR | Top 5 FAR | Top 20 FAR |
|---|---|---|---|
| Naive IF | 0.00 | 0.21 | 0.29 |
| Location-based IF | 0.17 | 0.18 | 0.23 |
| Location-based EIF | 0.08 | 0.17 | 0.15 |
| Location-based EIF re-rank | 0.00 | 0.00 | 0.03 |

Table 5: False alarm rate (FAR) of different algorithms on different ranks.

This experimental evaluation of WTG scenario is constrained by many factors and cannot fully reflect the realistic scenario. For example, compare to the amount of data in the streams, the labels are extremely sparse and might not be a very accurate indicator of the false alarm rate. Also, the labels should not be pre-collected, but collected on the run. Nevertheless, we believe this evaluation reflects the realistic performance to some extent. At the finishing time of this paper, the proposed Location-based EIF is being implemented in the environment of the company WTG and being used and evaluated.

## 6 Conclusion

Motivated by the problem at WTG, in this paper, we formally recognise the problem: interactive anomaly detection in streaming data. We then propose a novel algorithm: Evolutionary Isolation Forest to solve this problem. EIF

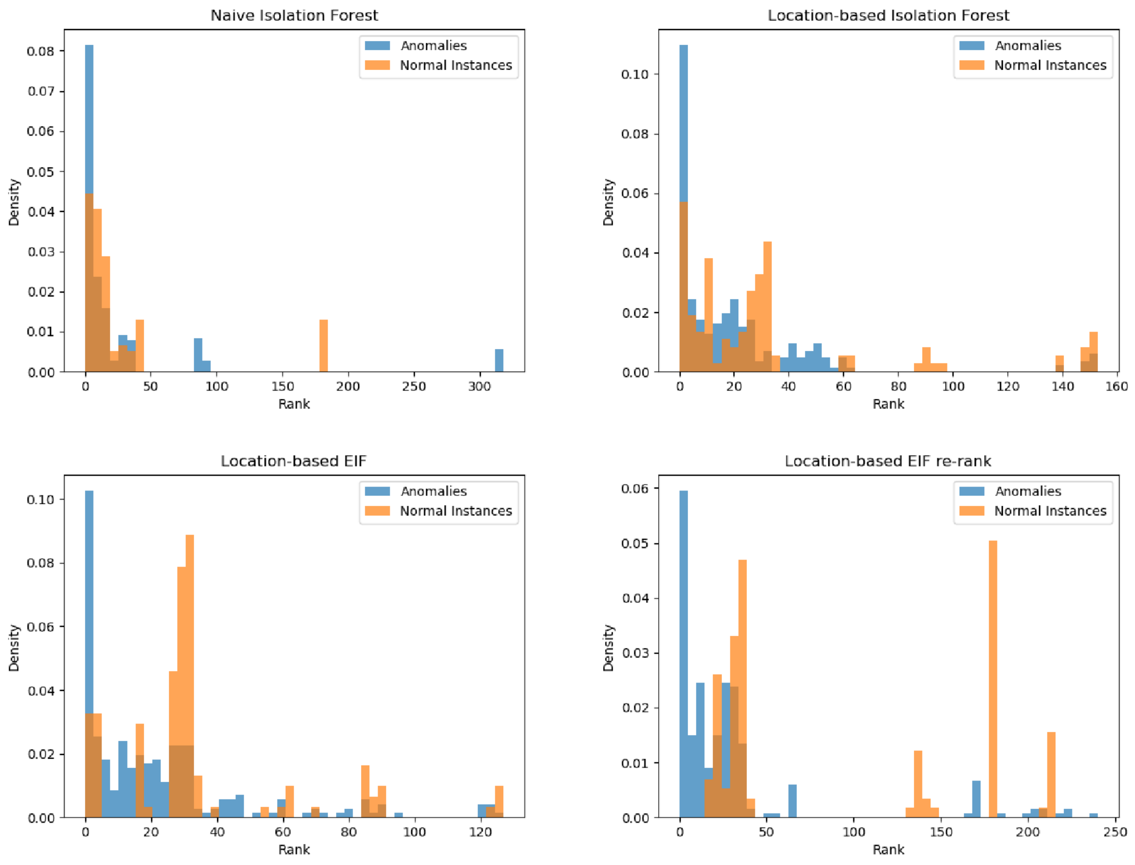

Figure 7: Density histogram of ranks of labeled instances. Top row: two baseline algorithms. Bottom left: Location-based EIF. Bottom right: Re-rank all instances after EIF received all labels.

uses evolutionary operators to adapt trees in an Isolation Forest ensemble detector to expert feedbacks. For a tree, the operators change the attribute and value at each node as well as replace branches with branches from other trees. By doing so the detector constantly evolves to provide more accurate, contextualised anomalies based on feedbacks.

Using fitness functions similar to OJRank [15], on the classic interactive anomaly detection problem, we show that EIF performs on the same level as AAD [7] and OJRank. On the interactive anomaly detection problem with streaming data, we show that EIF out-performs OJRank. Additionally, by applying EIF to a real-world problem: monitoring energy infrastructures at the company WTG, we show the performance of EIF at application and its realistic relevance. The advantages and disadvantages EIF has over previous methods are summarized as follow:

(1) EIF directly manipulates the base anomaly detector, while previous methods add another layer to adjust above the base detector and leave the base detector untouched. This brings one limitation, EIF cannot be applied directly to other ensemble anomaly detection algorithms than tree-based detectors. However, this also means EIF is not limited by the initial base detector. It can be adapted to a more complex context with more complex feedbacks. This is particularly important in streaming data anomaly detection.

(2) EIF offers much freedom in the designing of the algorithm. For example, covariate shift, which is very common in streaming data, can be handled by randomly adding new trees trained on new data at each evolutionary step. As another example, the problem of sparse feedbacks and various types of time series at WTG can be solved by location-based forests.

(3) EIF is simple to implement. It can be used with any fitness function with no requirement to calculate the gradient.

(4) EIF requires more computational power while updating the detector compare to previous methods. However, in realistic scenarios, this is part of a human-computer interaction loop and the speed of this loop is mostly dependant on the human's speed of data processing. A slightly slower algorithm will not affect the efficiency of this loop. While processing data to raise anomalies, EIF is as fast as other tree-based anomaly detectors.

The research field of interactive anomaly detection has received increasing attention in the recent years. As a realistic assumption, the presence of online expert feedback can greatly boost the accuracy of anomaly detectors. In this paper we make the first attempt to pin down the problem of interactive anomaly detection in a incremental scenario. We also propose a novel algorithm that adds to the variety of interactive anomaly detectors. We believe our work contributes to the research field by recognising the new direction and providing more (and in some cases better) varieties of algorithms.

Future works are suggested in two directions. First, we defined the incremental scenario fairly simple. Although such simplification does not affect the methodology and evaluation of our algorithm, a more carefully defined scenario may be needed for a more dedicated algorithm. Second, as our experiments show, feedbacks provided to EIF can easily saturate. A very likely reason is that we use the most common evolution strategy operators. These operators are not specially designed for tree-based classifiers. For example, having a crossover operator that can increase the depth of the tree when needed may provide more adaptability to feedbacks.

## References

[1] Jinwon An and Sungzoon Cho. Variational autoencoder based anomaly detection using reconstruction probability. *Special Lecture on IE*, 2(1):1–18, 2015.

[2] Ane Blázquez-García, Angel Conde, Usue Mori, and Jose A Lozano. A review on outlier/anomaly detection in time series data. *arXiv preprint arXiv:2002.04236*, 2020.

[3] George EP Box and David A Pierce. Distribution of residual autocorrelations in autoregressive-integrated moving average time series models. *Journal of the American statistical Association*, 65(332):1509–1526, 1970.

[4] Kalle Burbeck and Simin Nadjm-Tehrani. Adaptive real-time anomaly detection with incremental clustering. *information security technical report*, 12(1):56–67, 2007.

[5] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM computing surveys (CSUR)*, 41(3):1–58, 2009.

[6] Robert B Cleveland, William S Cleveland, Jean E McRae, and Irma Terpenning. Stl: A seasonal-trend decomposition. *Journal of official statistics*, 6(1):3–73, 1990.

[7] Shubhomoy Das, Weng-Keen Wong, Thomas Dietterich, Alan Fern, and Andrew Emmott. Incorporating expert feedback into active anomaly discovery. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*, pages 853–858. IEEE, 2016.

[8] Shubhomoy Das, Weng-Keen Wong, Alan Fern, Thomas G Dietterich, and Md Amran Siddiqui. Incorporating feedback into tree-based anomaly detection. *arXiv preprint arXiv:1708.09441*, 2017.

[9] Kaize Ding, Jundong Li, and Huan Liu. Interactive anomaly detection on attributed networks. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, pages 357–365, 2019.

[10] Everette S Gardner Jr. Exponential smoothing: The state of the art. *Journal of forecasting*, 4(1):1–28, 1985.

[11] Nico Görnitz, Marius Kloft, Konrad Rieck, and Ulf Brefeld. Toward supervised anomaly detection. *Journal of Artificial Intelligence Research*, 46:235–262, 2013.

[12] Jingrui He and Jaime G Carbonell. Nearest-neighbor-based active learning for rare category detection. In *Advances in neural information processing systems*, pages 633–640, 2008.

[13] W. Khreich, E. Granger, A. Miri, and R. Sabourin. A comparison of techniques for on-line incremental learning of hmm parameters in anomaly detection. In *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, pages 1–8, 2009.

[14] John R Koza, David Andre, Martin A Keane, and Forrest H Bennett III. *Genetic programming III: Darwinian invention and problem solving*, volume 3. Morgan Kaufmann, 1999.

[15] Hemank Lamba and Leman Akoglu. Learning on-the-job to re-rank anomalies from top-1 feedback. In *Proceedings of the 2019 SIAM International Conference on Data Mining*, pages 612–620. SIAM, 2019.

[16] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation forest. In *2008 Eighth IEEE International Conference on Data Mining*, pages 413–422. IEEE, 2008.

[17] Nir Nissim, Aviad Cohen, Robert Moskovitch, Assaf Shabtai, Mattan Edry, Oren Bar-Ad, and Yuval Elovici. Alpd: Active learning framework for enhancing the detection of malicious pdf files. In *2014 IEEE Joint*

*Intelligence and Security Informatics Conference*, pages 91–98. IEEE, 2014.

[18] Dan Pelleg and Andrew W Moore. Active learning for anomaly and rare-category detection. In *Advances in neural information processing systems*, pages 1073–1080, 2005.

[19] Tomáš Pevnỳ. Loda: Lightweight on-line detector of anomalies. *Machine Learning*, 102(2):275–304, 2016.

[20] Kirthanaa Raghuraman, Monisha Senthurpandian, Monisha Shanmugasundaram, V Vaidehi, et al. Online incremental learning algorithm for anomaly detection and prediction in health care. In *2014 International Conference on Recent Trends in Information Technology*, pages 1–6. IEEE, 2014.

[21] Mayu Sakurada and Takehisa Yairi. Anomaly detection using autoencoders with nonlinear dimensionality reduction. In *Proceedings of the MLSDA 2014 2nd Workshop on Machine Learning for Sensory Data Analysis*, pages 4–11, 2014.

[22] Bernhard Schölkopf, Robert C Williamson, Alex J Smola, John Shawe-Taylor, and John C Platt. Support vector method for novelty detection. In *Advances in neural information processing systems*, pages 582–588, 2000.

[23] Md Amran Siddiqui, Alan Fern, Thomas G Dietterich, Ryan Wright, Alec Theriault, and David W Archer. Feedback-guided anomaly discovery via online optimization. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2200–2209, 2018.

[24] Nikolai V Smirnov. On the estimation of the discrepancy between empirical curves of distribution for two independent samples. *Bull. Math. Univ. Moscou*, 2(2):3–14, 1939.

[25] Vincent Vercruyssen, Meert Wannes, Verbruggen Gust, Maes Koen, Bäumer Ruben, and Davis Jesse. Semi-supervised anomaly detection with an application to water analytics. In *Proceedings/IEEE International Conference on Data Mining.* IEEE, 2018.