# Review of Isolation and Robust Random Cut Forests

J. Marcus Hughes

*University of Colorado Boulder*

**Abstract**

A review of isolation forests and robust random cut trees.

*Keywords:* anomaly detection, novelty, online, algorithms, machine learning

The ideas and even some text are borrowed from the works specified. At times, the wording may not diverge from the original work. I do not claim it is an original synthesis of the ideas at this writing stage and acknowledge some copied proof text and intend no plagiarism. This a reproduction of earlier ideas with additional comparison of algorithms and proofs with some original figures and explanation. The text needs significant revision.

## 1. Introduction

Anomaly detection is the general problem of unsupervised rare class detection, i.e. in a data set we would like to find points or sets of points that are not typical, potentially because they are drawn from a different distribution. Detecting such points has different applications depending on the context. For example, when observing web traffic an anomaly could be a malicious attack since the behavior deviates from typical. In astronomy, an anomaly in a set of images could be a previously undiscovered kind of astronomical object. Or in general supervised classification, an anomaly in a training database could be a mislabeled sample. In a one dimensional setting, this problem may seem trivial, find points in low probability density regions. When we generalize to higher dimensions or online settings, we may not be able to rely on density estimation techniques. This paper summarizes the ideas of isolation forests and their related works, with special emphasis on robust random cut forests, and proposes some ideas for future work.

## 2. Background

Anomaly detection is the unsupervised (can sometimes be treated as supervised but I prefer to call that *novelty detection* to differentiate) task of detecting points, such as $o_1$, $o_2$, and $O_3$ in Figure 1, generated by an atypical distribution compared to the typical classes, $N_1$ and $N_2$ in Figure 1. Anomaly detection has mainly been an applied problem with a context specific definition without a more theoretical background, so there are many definitions of anomaly. We will adopt a definition by Hawkins, "An outlier [or anomaly] is

*Email address:* jahu5138@colorado.edu (J. Marcus Hughes)
*URL:* jmbhughes.com (J. Marcus Hughes)

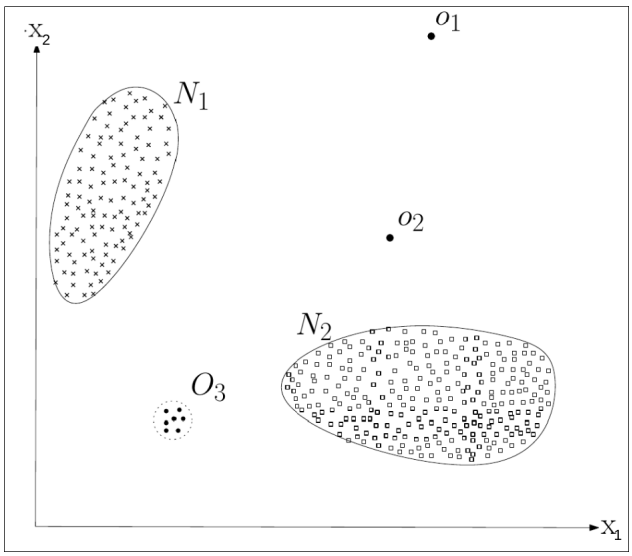


Figure 1: **Anomaly detection sketch**: This figure, modified from Chandola et al. [1], outlines the general task of anomaly detection in a two-dimensional mock-up. Regions $N_1$ and $N_2$ are typical while $o_1$, $o_2$, and $O_3$ indicate different examples of anomalous points.

an observation which deviates so much from the other observations as to arouse suspicions that it was generated by a different mechanism [2]. Some literature differentiates $o_1$ and $o_2$ as point anomalies and $O_3$ a clustered anomaly since it has so many neighbors. However, I discourage this distinction because it can be thought of as a chance occurrence, with more samples $o_1$ and $o_2$ could have nearby anomalous points. Some anomalous points will be harder to identify than others. For example, if one only observed $x_2$, $o_2$ and $O_3$ would no longer appear anomalous because they appear typical with respect to the $x_2$ marginal distribution. Emmott et al. [3] provide a more rigorous approach to the dimensions of the anomaly detection problem:

- **Point difficulty** describes how close the anomaly class is to the typical distribution. In some settings, such as computer security, an adversary tries to blend in with the typical distribution to avoid detection. In general, the farther an anomaly class is from the typical distribution the easier it is to identify. One might say $o_2$ is harder to find than $o_1$ in Figure 1, although an example right on the border of one of the typical classes $N_1$ and $N_2$ would be even harder.

- **Relative frequency** or anomaly rate is how often the anomaly class appears relative to the typical distribution. For example, 0.5% of the data points could be anomalous. Depending on the rate, some approaches may make more sense than other. For example, if the anomaly rate is very low attempting to model the anomaly classes' distribution shape is likely impossible.

- **Semantic variation** is how structured the anomaly class is. Finding points drawn from only one region would be much easier than if the anomaly class was a more complicated mixture of Gaussians. More structure means the examples are distributed over a wider area.

- **Feature irrelevance** is a common idea in supervised classification; you want to construct meaningful features and ignore spurious ones. However, it is impossible to know a priori what features will be beneficial in anomaly detection, especially if the distributions are non-stationary. Feature irrelevance describes how many features are not helpful in identifying anomalies.

There have been many approaches to the anomaly detection problem. Pimentel et al. [4] (and [1]) provide a concise review and categorization of many existing techniques. One set of approaches are based on ensembles of decision trees.

Decision trees are simple data structures that can partition the input space completely. Random forests combine many decision trees to get an ensemble decision, introducing randomness by bagging the training examples, so each tree sees a different chance structure and the ensemble is more robust to noise [5]. Random forests have been very successful in supervised classification problems [6, 7, see these papers for some examples]. There have been some attempts to directly use random forests for unsupervised tasks, but these require assigning randomized labels to the data or creating a synthetic dataset [8, 9]. Liu et al. [10] instead utilized trees directly on unlabeled data to measure anomonality via the depth of trees. We will discuss this work, briefly touch on some extensions of the work, and then focus on Robust Random Cut Trees by Guha et al. [11].

## 3. Original work

Isolation forests, an ensemble of isolation trees as introduces by Liu et al. [10], focus on using decision trees to isolate points with the observation that isolating an outlier will take fewer steps than isolating a typical point as illustrated in Figure 2. These algorithms do not require expensive distance calculations or assume models of the probability density. Consequently, they are very fast and require little space compared to some algorithms.

More formally, consider a set of points $S$, which may or may not contain anomalous points. The Isolation Tree algorithm (Algorithm 1) constructs an isolation tree from these points, call it $IT(S)$, by using random partitions iteratively to isolate every data point.

Then, for any point $x$, regardless of whether $x$ was used to train the Isolation Tree, we can traverse from the root of the tree to a leaf node by following the decision tree rules, writing a 1 for each left movement and a 0 for each right movement. This results in a string modeling modeling the decision path of $x$ in tree $T$, call it $m_T(x)$. For example $m_T(x) = 1001$ indicates it took 4 decision to get a leaf node, specifically a left movement followed by two right movements and finally a left movement. Then, $|m_T(x)|$ is the depth $x$ obtains in tree $T$, where $|m_T(s)| = 4$ in the example. For clarity, we will refer to this as $\textsc{Depth}_T(x)$ and $\textsc{Leaf}_T(x)$ to be the leaf node $x$ reaches. Since we may want to height limit the tree so it does not become unwieldy to store, the algorithm includes that option. If $H$=3, the resulting tree will not obtain a depth greater than 3. To avoid losing information, we store the number of points in a leaf node as *count*. We can use this to provide a better approximation of the expected depth had we completely grown the tree by setting $H = \infty$.

We can then score any point in a normalized fashion by comparing it to the average path length of an unsuccessful search in a binary search tree. The average path length of an

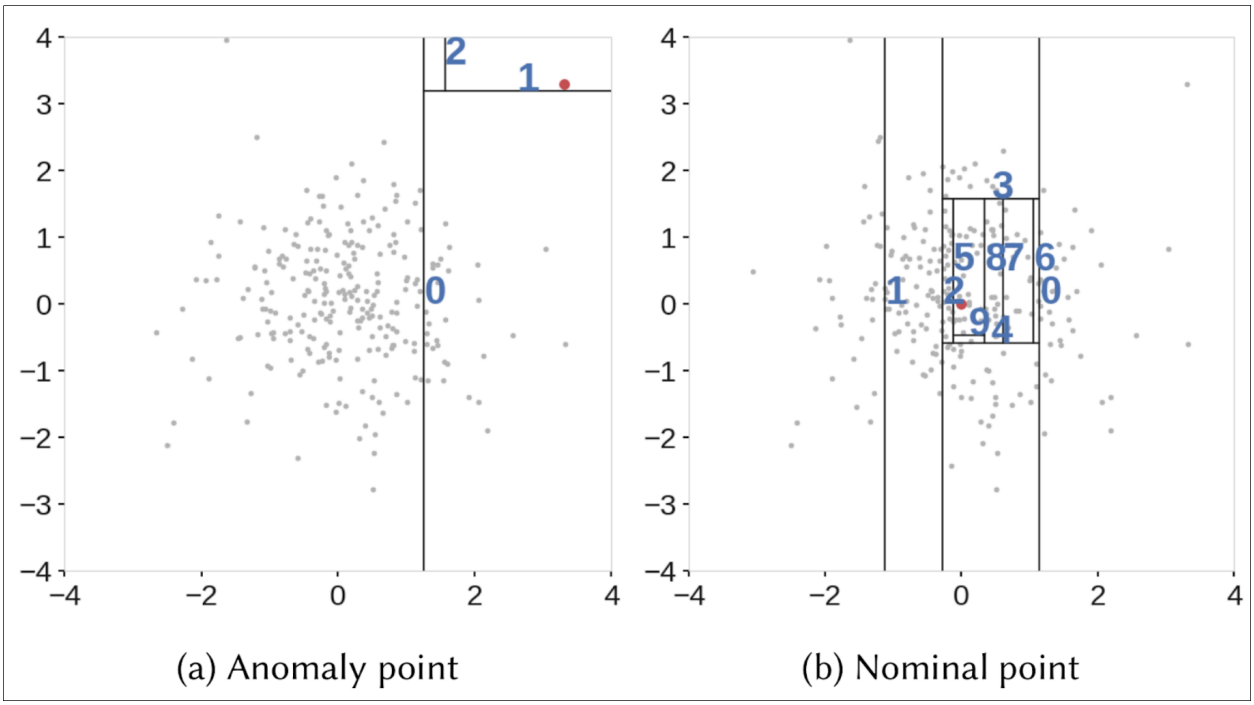


Figure 2: **Isolation forest paradigm**: This figure (reproduced from [12]) illustrates the central theme of the isolation forest algorithm. Each line indicates a decision rule, i.e. a step down the tree. For the red anomalous point in (a) only 3 cuts are needed while the red typical point immersed in the typical distribution requires at least 10 cuts.

unsuccessful search in a binary search tree is

$$p(n) = 2H(n-1) - (2(n-1)/n)$$

where $n$ is the *count* in a leaf node and $H(i)$ is the $i$-th harmonic number. We can define the anticipated depth of $x$ if $H = \infty$ as $\widehat{\text{DEPTH}}_T(x) = \text{DEPTH} + p(\text{LEAF}(x).count)$. Thus, the anomaly score for a point $x$ based on tree $T$ trained on set $S$ is

provide a proof of this

$$\log_2 \text{SCORE}_T(x) = -\frac{\widehat{\text{DEPTH}}_T(x)}{p(|S|)}$$

. When $H = \infty$, we can use $\text{DEPTH}_T(x)$ instead in the calculation since $\text{LEAF}_T(x).count = 1$.

Isolation forests are ensembles of isolation trees where each isolation tree is trained on a random sample of the full training dataset. For a forest $F$ composed of $n$ trees $\{T_1, T_2, \cdots, T_n\}$ when we assume that all the samples are the same size $|S|$, we can calculate the forest score as:

$$\log_2 \text{SCORE}_F(x) = \frac{-\mathbb{E}_T\left[\widehat{\text{DEPTH}}_T(x)\right]}{p(|S|)} \approx \frac{\frac{1}{n}\sum_{T\in F}\widehat{\text{DEPTH}}_T(x)}{p(|S|)}$$

.

**Algorithm 1** Setting up an Isolation Tree

**Precondition:** $S \subset \mathbb{R}^d$, unlabeled set of point in which to find anomalies; $H$ the height limit specified; $h$, the current depth, initially 0

**function** InitializeIT($S$, $h$)
  **if** $|S| = 1$ or $h = H$ **then** ▷ a point has been isolated
    $count \leftarrow |S|$
    $c_l \leftarrow$ None ▷ left child is empty
    $c_r \leftarrow$ None ▷ right child is empty
    **return** External node of only $S$
  **else**
    Choose a dimension, $k \sim \text{Uniform}[1, d]$.
    $x^l \leftarrow \min_{x \in S} x_k$
    $x^h \leftarrow \max_{x \in S} x_k$
    Choose a cut $C \sim \text{Uniform}[x^l, x^h]$.
    $S_l \leftarrow \{x | x \in S, x_k \leq C\}$ ▷ left child tree
    $S_r \leftarrow S \setminus S_1$ ▷ right child tree
    $c_l \leftarrow$ InitializeIT($S_l$, h+1) ▷ recurse on left
    $c_r \leftarrow$ InitializeIT($S_r$, h+1) ▷ recurse on right
    **return** Internal node
  **end if**
**end function**

### *3.1. Hyperplanes instead of axis aligned cuts*

Isolation forests use axis-aligned cuts. However, this introduces an artifact into the anomaly score map as shown in Figure 3. Furthermore, limiting cuts to being axis-aligned is a strong simplification. Relaxing that constraint leads to the extended isolation forest [12]. (We could relax even further by not requiring hyper-planes but more complex boundaries.) This change resolves the artifacts without significant modification of the algorithm. Instead of picking a random dimension and cut, the extended isolation forest algorithm picks a random direction from $\mathbb{R}^d$ at uniform and a random intercept in the data sample (like the isolation forest).

The SciForest algorithm [13] similarly uses hyperplanes instead of axis-aligned cuts. It was designed to specifically deal with clustered anomalies by introducing a new "goodness" metric to select the randomized cuts, so that they separated clumps better.

### *3.2. Online Approaches*

I would like to extend this algorithm to an online version. Before we get to robust random cut trees, there are two more direct online versions to note: the sliding window version and the half-space trees. The sliding window version of Ding and Fei [14] is a very direct extension of isolation forests to the online setting. Assume the data is coming in on a stream instead of being available all at once. Then, partition this stream into $k$ data point sections. At any given point in the algorithms execution, we will retain two forests: the training forest and the evaluation forest. The evaluation forest has seen $k$ data points in the previous data partition and been completely grown. Upon seeing a new data point we

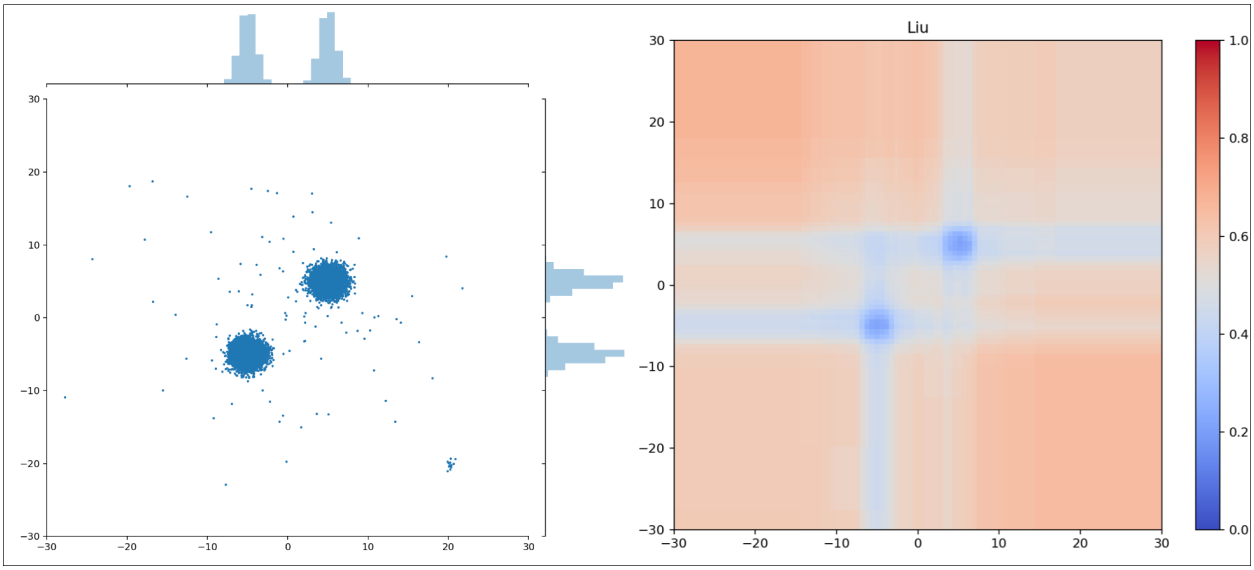


Figure 3: **Isolation forest artifact**: At left is an input probability distribution on which an Isolation Forest was trained (similar to the one shown in Hariri et al. [12]). The right figure is the resulting anomaly score map. Regions that are strongly red are considered anomalous while blue regions are typical. The blue vertical and horizontal regions of typical designation are an artifact of picking axis aligned cuts.

use this to evaluate the anomaly score. The training forest is collecting data points in the current data partition. After collecting $k$ data points, we can grow a new forest. During the algorithm's execution, they occasionally evaluate whether the training forest should become the evaluation forest by detecting concept drift if the anomaly rate has exceeded some user input threshold. Half-space trees use a similar windowing procedure but modify the tree growing algorithm considerably (to the point we might not consider it an isolation based approach because they're estimating the density in regions) [15].

Both of these ideas add the complication of selecting the correct window size and assuming the window switch is done at the appropriate time. They are mini-batch approaches so the forest is not giving an up-to-date score but a "reasonably" up-to-date score if the hyper-parameters of window size and window start time were selected appropriately. This can go horribly wrong as illustrated in Figure 4.

## 4. Robust random cut forest

Robust random cut forests depart from a couple of ideas of isolation forests in order to allow better performance but also permit proofs regarding their performance [11]. They shift the focus from saying anomalies have the shallowest depths in a tree to points that are the most "surprising," i.e. points that cause the most displacement in the depth of other points to include. Together, these terms define the complexity change of including a point in the model. This will be rigorously stated shortly.

The other change is in the construction of a robust random cut tree; it is very similar to construction an isolation tree except that the selection of the random cut is not done at uniform random but instead weighted by the dimension size. For example in a two

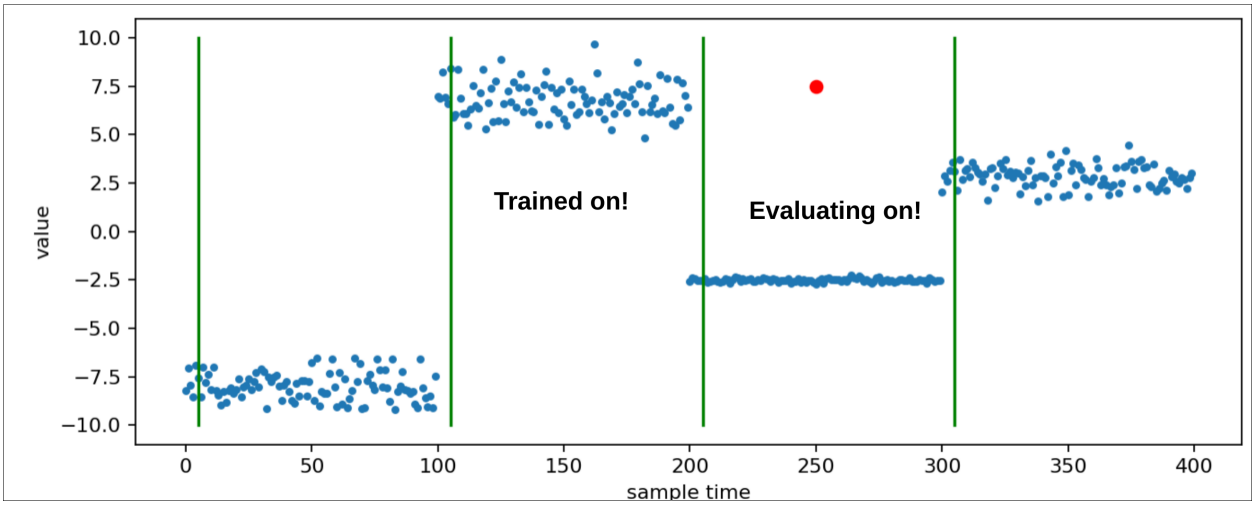


Figure 4: **Problem with windowing**: The green vertical lines indicate the window boundaries. Thus, when we evaluate the red data point's score the forest will have been trained using the previous window with the label "trained on." With respect to this "trained on" window, the red point is not anomalous. However, it should be considered anomalous with respect to the current data.

dimensional setting when the set $S$ had a minimal bounding box $B(S) = [-1, 1] \times [-10, 10]$, we see that the first dimension has length 2 while the second dimension has length 20. In the isolation tree, each dimension is equally likely to be selected when making a cut, but in the RRCT the second dimension is 10 times more likely to be chosen for a cut. This is reflected in Algorithm 2.

This creates a binary tree in which each leaf node corresponds to a single data point, the same as the isolation forest. The authors do not include a notion of forcing depth limitations other than subsampling. The largest differences arise in how the anomaly score is calculated, discussed in Section 4.2.

### *4.1. RRCTs preserve distance*

Before we embark on using RRCTs to detect anomalies, we would like to know they actually preserve information relevant to anomalies. The isolation forest approach lacks this proof, so the user is left *hoping* that the algorithm works in their case. Since anomalies can be considered as points that are distant from their neighbors, we will show that the RRCT approximately preserves distance.

**Theorem 1.** ***RRCTs approximate distance.*** *Using Algorithm 2 let the weight of a node in a tree be the corresponding sum of dimensions of its bounding box, $\sum_i l_i$. Given two points $u, v \in S$ define the tree distance between $u$ and $v$ to the be the weight of the least common ancestor of $u$ and $v$. This tree distance is always at least the Manhattan distance $L_1(u, v)$ and in expectation at most $O\left(d \log \frac{|S|}{L_1(u,v)}\right)$ times $L_1(u, v)$.*

*Proof.* When $B(S)$ is the bounding box in the construction of $RRCT(S)$, we define $P(S)$ to be the sum of the lengths of the sides, $\sum_i l_i = \sum_i \max_{x \in S} x_i - \min_{x \in S} x_i$. Consider points

**Algorithm 2** Setting up a Robust Random Cut Tree

**Precondition:** $S \subset \mathbb{R}^d$, unlabeled set of point in which to find anomalies

**function** InitializeRRCT($S$)
  $x_i^l \leftarrow \min_{x \in S} x_i$
  $x_i^h \leftarrow \max_{x \in S} x_i$
  $B(S) \leftarrow [x_1^l, x_1^h] \times [x_2^l, x_2^h] \times \cdots \times [x_d^l, x_d^h]$ ▷ minimal bounding box
  **if** $|S| = 1$ **then**
    $c_l \leftarrow$ None ▷ left child is empty
    $c_r \leftarrow$ None ▷ right child is empty
    **return** External node of only $S$
  **else**
    Choose a dimension, $k$, proportional to $\frac{l_k}{\sum_i^d l_i}$.
    Choose a cut $C \sim \text{Uniform}[x_k^l, x_k^h]$.
    $S_l \leftarrow \{x | x \in S, x_k \leq C\}$ ▷ left child tree
    $S_r \leftarrow S \setminus S_1$ ▷ right child tree
    $c_l \leftarrow$ InitializeRRCT($S_l$) ▷ recurse on left
    $c_r \leftarrow$ InitializeRRCT($S_r$) ▷ recurse on right
    **return** Internal node
  **end if**
**end function**

$a$ and $b$. The probability that we separated them in the first cut is proportional to the $L_1$ measure:

$$\frac{1}{P(S)} \sum_i |a_i - b_i|$$

This is because in dimension $i$ we must make a cut in $\frac{|a_i - b_i|}{l_i}$. When we can pick any dimension $i$ we get the aforementioned expression. Further, $P(S)$ decreases in expectation by at least a factor of $\left(1 - \frac{1}{2d}\right)$ as we move down the tree. We can see this because if we start with a bounding box where side $i$ has length $l_i$ then the new expected perimeter decreases by $\sum_i \frac{l_i}{P(s)} \frac{l_i}{2}$. The first term corresponds to the probability of picking that dimension and the second is the expected decrease. So:

$$\sum_i \frac{l_i}{P(S)} \frac{l_i}{2} \geq \frac{\sum_i l_i^2}{2 \sum_i l_i} \geq \frac{\sum_i l_i}{2d}$$

Then, we prove the theorem by observing that the distance assigned to a level corresponding to $S'$ is $P(S')$ and the probability of that distance assignment is $L_1(a, b)/P(S')$. The expected distance therefore is $L_1(a, b)$ times the expected number of steps that separate the two points. The expected number of steps can be bounded by $O(d \log P(S)/L_1(a, b))$ since $P(S)$ decreases by a factor of $\left(1 - \frac{1}{2d}\right)$ in expectation. □

*4.2. Anomaly score calculations*

Since RRCTs preserve distance, let's consider how they can be used to score anomalies. Instead of strictly considering depth, let's build some intuition for what it means to be an

anomaly. Imagine you had observed a crowd of dogs and cats. They're a variety of colors, weights, and heights. Since you have only observed cats and dogs of these colors your model of an animal would consist of three traits: color, weight, and height. (Maybe if you knew a taxonomy your model of the world would include species.) Now, you are shown a pink flamingo. It may weight about the same as a cat but has a very different color and height. To incorporate it into your model you must amend your understanding of the world; it forces you to pay attention to a different set of features. The robust random cut tree conceptualizes of degree of anomaly as how much it forces you to change your model.

To begin defining our models from the trees, consider the path from the root node to a leaf, the path that a single point follows as it propagates through the binary rules. At the root, it either goes left or right based on a rule. While following this path, we can write each decision down in the same . Let 0 be a left decision and 1 be a right decision. Thus, for $x \in S$, we can write its path as a binary string in the same manner of the isolation forest, $m(x)$. We can repeat this for all $x \in S$ and arrive at a complete model description: $M(S) = \oplus_{x \in S} m(x)$, the appending of all our $m(x)$ together. We will consider the anomaly score of a point $x$ to be the degree to which including it causes our model description to change if we include it or not, i.e.:

$$\mathbb{E}_T \left[|M(T)|\right] - \mathbb{E}_T \left[|M(\textsc{Delete}(x, T)|\right]$$

We have to consider expectations over the trees because our trees are constructed with randomness included. This definition also requires that we have a DELETE and INSERT procedure that returns a tree after adding a point. Furthre, this algorithm must be "memoryless," i.e. if we perform a series of insertions and corresponding deletions of points the initial tree and final tree, constructed from identical point sets $S$, should be drawn at random from the same probability distribution $RRCT(S)$ and not have some long trailing structures or other indicators that make them special to the process or insertion and deletion. Given a tree $T$ built from a point set $Z$ which appears to be drawn at random from a distribution $RRCT(Z)$, we wish to have an INSERT procedure that inserts a point $p$ to $T$ and results in a tree that appears to be drawn at random from a distribution $RRCT(Z \cup \{p\})$. Similarly, we wish that a DELETE procedure removes a point $p$ from $T$ and results in a tree that appears to be drawn at random from a distribution $RRCT(Z \setminus \{p\})$. Such algorithms are stated in Algorithms 3 and 4.

With these provided, we can expound on the anomaly score on a full point set $Z$ (assume

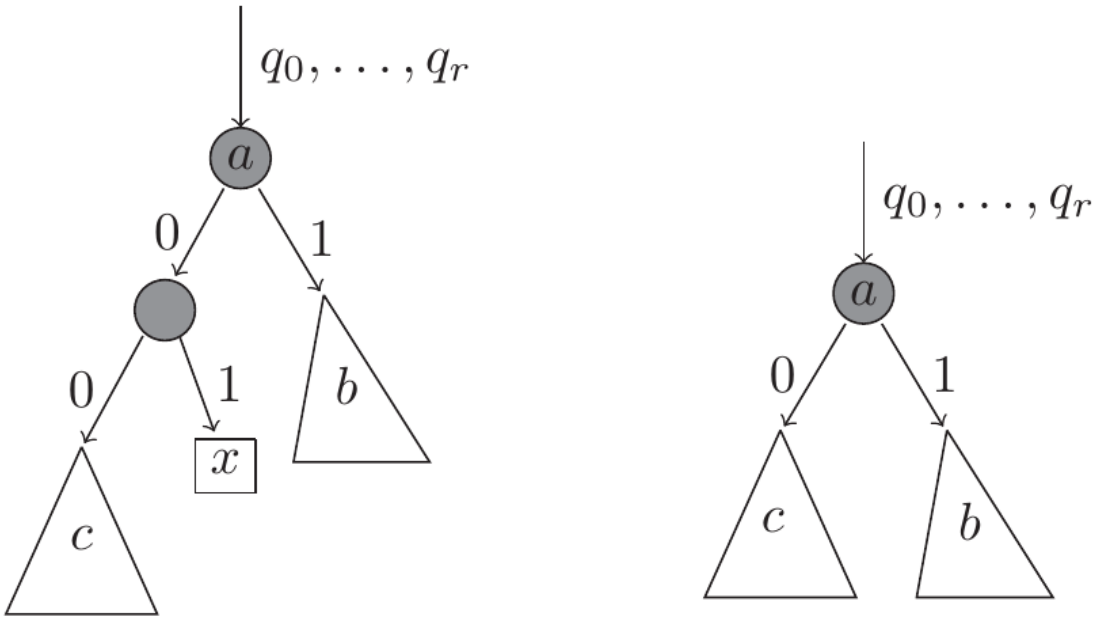


Figure 5: **Model, Insert and Delete**: On the left we see a tree with $x$ and on the right is a tree with $x$ deleted using the DELETE algorithm. Similarly, we can move from the right to the left using the INSERT algorithm. However, the INSERT algorithm could result in other trees instead. If we write the model of $x$, we find $m(x) = q_0...q_r01$.

we are not subsampling when constructing a tree):

$$\begin{aligned}
&\mathbb{E}_T\left[|M(T)|\right] - \mathbb{E}_T\left[|M(\textsc{Delete}(x,T)|\right] \\
&= \left(\sum_T \mathbb{P}\left[T\right]\left(\textsc{Depth}(x,T) + \sum_{y\in Z-\{x\}} \textsc{Depth}(y,T)\right)\right) \\
&\quad - \left(\sum_T \sum_{y\in Z-\{x\}} \mathbb{P}\left[T\right]\textsc{Depth}\left(y,\textsc{Delete}(x,T)\right)\right) \\
&= \sum_T \mathbb{P}\left[T\right]\textsc{Depth}(x,T) + \sum_T \sum_{y\in Z-\{x\}} \mathbb{P}\left[T\right]\left(\textsc{Depth}(y,T) - \textsc{Depth}(y,\textsc{Delete}(x,T))\right) \\
&= \mathbb{E}_T\left[\textsc{Depth}(x,T)\right] + \textsc{Disp}(x,Z)
\end{aligned}$$

The first term is analogous to what the isolation forest focuses on, the average depth of a point indicates its anomaly score. The second term is what RRCT focuses on, displacement, the surprise induced by including a point or not in a set $Z$:

$$\textsc{Disp}(x,Z) = \sum_T \sum_{y\in Z-\{x\}} \mathbb{P}\left[T\right]\left(\textsc{Depth}(y,T) - \textsc{Depth}(y,\textsc{Delete}(x,T))\right)$$

This measure is almost, but not exactly, what we want to consider when evaluating the anomaly score. The problem arises that our anomaly may have nearby neighboring points,

such as the orange $x'$ in Figure 6, an example of a clustered anomaly member of $O_3$ in Figure 1. In this case, the anomalies form a sub-tree and removing a single one does not impact the model complexity that much because only one or a few points' models are changed, those in the anomaly sub-tree ($x$ and $x'$). Instead, we wish to remove the entire anomaly sub-tree (both $x$ and $x'$ to result in the right panel of Figure 5) to change the model of a much greater number of points. We can observe this is the case by considering the ramifications of Theorem 2.

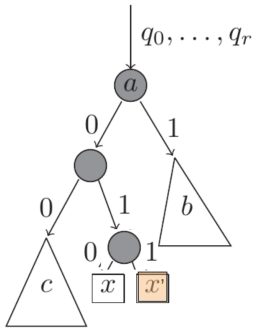


Figure 6: **Colluders:** $x$ and $x'$ are very similar anomalous points. Removing only one causes a change of 1 in the model complexity. However, removing both would cause $|c|$ displacement, a much larger number, and result in the right panel of Figure 5.

**Theorem 2.** *The expected displacement caused by a point $x$is the expected number of points in the sibling node of the node containing $x$, when the partitioning is done according to the RRCT algorithm.*

*Proof.* In the absence of $x$ (the right panel in Figure 5), the representation of a point in $c$ would be $q_0, ..., q_r, 0, ...$, and would require precisely 1 fewer bits to represent than if $x$ were included. Thus,

$$\textsc{Depth}(y, T) - \textsc{Depth}(y, \textsc{Delete}(x, T) = \begin{cases} 1 & y \in c, \text{the sibling of } x \\ 0 & \text{otherwise} \end{cases}$$

The total displacement is summed over all $y$ and thus in expectation is the number of siblings of $x$. □

Thus if the anomaly had a neighboring point, it the model change from removing only one of them is 1 and would not trigger our anomaly detection algorithm. Thus, we consider clusters of anomalies as "colluders." Together they mask the presence of anomalies. This colluding should be an expected natural and common phenomena because the process that generates one anomaly can, although potentially very rarely, generate another anomaly. We do not expect only one flamingo in the world. Even if flamingos are rare compared to cats and dogs we expect to have a few. Thus, we require a Co-Displacement measure that removes a point set $C$, all the colluding anomalies at one time. However, the number of point sets to consider when looking for colluding sets is exponential, $\propto 2^{|Z|}$. Further, we

have not incorporated the ability for each tree is trained on a subset $S$ of $Z$. Thus, we define the collusive displacement as follows:

$$\textsc{CoDisp}(x, S, Z) = \mathbb{E}_{S \subset S, T}\left[\max_{C \subset S, x \in C} \frac{1}{|C|} \sum_{y \in S - C} (\textsc{Depth}(y, T) - \textsc{Depth}(y, \textsc{Delete}(C, T))\right]$$

Here, $\textsc{Delete}(C, T)$ means we delete all elements of $C$ from the $T$ iteratively at random. Further, we equally weight the displacement of each element of $C$ by multiplying by $\frac{1}{|C|}$. If we knew something about the power of each point, we might consider a different approach.

While this is a wonderful notion, we need to insure that we can actually calculate it in reasonable time so that we can using this algorithm in a streaming setting. This is proven in Theorem 3.

**Theorem 3.** CoDisp*(x, S, Z) can be estimated efficiently.*

*Proof.* From Theorem 2, we know that $\textsc{Depth}(y, T) - \textsc{Depth}(y, \textsc{Delete}(C, T))$ is nonzero for $y \in Z - C$ if and only if we delete all elements in a sibling tree containing $y$. For example in the left panel of Figure 5 when $|b|$ is large, the collusive displacement will be large only if we delete all the nodes in $|c|$ and $x$. We must simultaneously delete all copies and colluders of $x$. For a given tree $T$, we can compute

$$\max_{C \subset S, x \in C} \frac{1}{|C|} \sum_{y \in S - C} (\textsc{Depth}(y, T) - \textsc{Depth}(y, \textsc{Delete}(C, T)))$$

by just considering the sub-trees containing $x$ along the leaf to root path of $x$. This will be small compared to the number of points. Further, we could limit the number of steps up from the leaf we go because deleting more than a certain fraction of points means that $x$ is not an anomaly by definition. □

### *4.3. Tree maintenance: insertion and deletion*

The functionality of RRCT hinges upon an efficient Insert and Delete that is nice with respect to probability distributions; it's memoryless. Finally, we define them and prove that Delete conserves the property. The proof of Insert is very similar and provided in the supplementary materials of Guha et al. [11].

**Algorithm 3** Delete

**Precondition:** $T$, an RRCT tree, and a point $p$ to delete

1: **function** Delete($p$, $T$)
2: $v \leftarrow$ node where $p$ is isolated in $T$, i.e. leaf node found by $p$ traversing from the root
3: $u \leftarrow$ sibling of $v$
4: Replace the parent of $v$ with $u$.
5: Update all bounding boxes starting from $u$'s parent upwards
6: **return** modified tree
7: **end function**

The Delete procedure is fairly simple; we just short circuit the parent of the leaf by replacing it with the leaf node's sibling.

**Algorithm 4** Insert

**Precondition:** $T$, an RRCT tree, and a point $p$ to delete

1: **function** INSERT($p$, $T$)
2: $S \leftarrow$ the set of points $T$ corresponds to
3: **if** $|S| = 0$ **then**
4: **return** tree containing only $p$
5: **else**
6: $x_i^l \leftarrow \min_{x \in S} x_i$
7: $x_i^h \leftarrow \max_{x \in S} x_i$
8: $B(S) \leftarrow \prod_i [x_i^l, x_i^h]$ ▷ bounding box of S
9: $\hat{x}_i^l \leftarrow \min\{p_i, x_i^l\}$
10: $\hat{x}_i^h \leftarrow \max\{x_i^h, p_i\}$
11: Select $r \sim \text{Uniform}[0, \sum_i \left(\hat{x}_i^h - \hat{x}_i^l\right)]$
12: $k \leftarrow \text{argmin}\{j | \sum_{i=1}^{j} \left(\hat{x}_i^h - \hat{x}_i^l\right) \geq r$ ▷ dimension of random cut
13: $C \leftarrow \hat{x}_k^l + \sum_{i=1}^{k} \left(\hat{x}_i^h - \hat{x}_i^l\right) - r$ ▷ a cut in dimension $k$
14: **if** $C \notin [x_k^l, x_k^h]$ **then** ▷ $C$ separates $S$ and $p$
15: $u \leftarrow$ a node ▷ the new root node of the tree
16: **if** $p > C$ **then**
17: $u_l \leftarrow T$ ▷ copy the existing tree
18: $u_r \leftarrow$ INITIALIZERRCT($\{p\}$) ▷ make a node for $p$
19: **else** ▷ $p \leq C$
20: $u_l \leftarrow$ INITIALIZERRCT($\{p\}$) ▷ make a node for $p$
21: $u_r \leftarrow T$ ▷ copy the existing tree
22: **end if**
23: **else** ▷ $C$ does not separate $S$ and $p$
24: $u \leftarrow$ root node of $T$
25: $C \leftarrow$ cut at the root of $T$ ▷ throw away generated $C$ from Line 13
26: **if** $p > C$ **then**
27: $u_r \leftarrow$ INSERT($x, u_r$) ▷ $u_l$ unchanged
28: **else** ▷ $p \leq C$
29: $u_l \leftarrow$ INSERT($x, u_l$) ▷ $u_r$ unchanged
30: **end if**
31: **end if**
32: **end if**
33: **return** updated tree
34: **end function**

INSERT is comparatively more difficult. We first have to randomly attempt to insert the node at the root of the tree. If this fails, we propagate it down a level and recursively tree to insert on whichever side it went to. The other side remains the same.

*4.3.1. Online nature of maintenance procedures*

Before proving any properties of the procedures, we will introduce a Theorem 4 we must use in the property proofs.

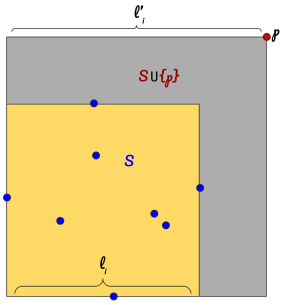


Figure 7: **Theorem 4 diagram**: This figure clarifies some of the theorem probabilities. Let dimension $i$ the be horizontal axis.

**Theorem 4.** *Given a point $p$ and a set of points $S$ with an axis parallel minimal bounding box $B(S)$ such that $p \notin B$:*

(i) *For any dimension $i$, the probability of choosing an axis parallel cut in dimension $i$ that splits $S$ using the RRCT algorithm is exactly the same as the conditional probability of choosing an axis parallel cut that splits $S \cup \{p\}$ in dimension $i$ conditioned on not isolating $p$ from all points of $S$.*

(ii) *Given a random tree of $RRCT(S \cup \{p\})$, conditioned on the first cut isolating $p$ from all points of $S$, the remainder of the tree is a random tree in $RRCT(S)$.*

*Proof.* The second part follows from the construction of the tree. Thus, we focus on the first part, referring to Figure 7 for a picture motivation. . Let the length of the minimum bounding box of $S$ in dimension $i$ b $l_i$, i.e. $l_i = \max_{x \in S} x_i - \min_{x \in S} x_i$. Similarly define the length of the minimum bounding box of $S' = S \cup \{p\}$ in dimension $i$ be $l'_i$. The probability density of choosing a cut $C$ in dimension $i$ that splits $S$ is $\frac{1}{l_i}\frac{l_i}{\sum_i l_i}$ because $\frac{l_i}{\sum_i l_i}$ is the probability of choosing dimension $i$ given the algorithm and $\frac{1}{l_i}$ is the probability density of choosing a specific cut.

The probability density of achieving this same cut in $S \cup \{p\}$ conditioned on not isolating $p$ and $S$ is

$$\frac{1}{l_i}\mathbb{P}\left[\text{choosing dimension } i | \text{not isolating } p \text{ and } S\right]$$

Observe that

$$\mathbb{P}\left[\text{choosing dimension } i \textbf{ and } \text{not isolating } p \text{ and } S\right] = l_i / \sum_i l'_i$$

. Further,

$$\mathbb{P}\left[\text{not isolating } p \text{ and } S\right] = \sum_i l_i / \sum_i l'_i$$

. Finally,

$$\begin{aligned}\mathbb{P}\left[\text{choosing dimension } i | \text{not isolating } p \text{ and } S\right] &= \frac{\mathbb{P}\left[\text{choosing dimension } i \textbf{ and } \text{not isolating } p \text{ and } S\right]}{\mathbb{P}\left[\text{not isolating } p \text{ and } S\right]} \\ &= \frac{l_i / \sum_i l'_i}{\sum_i l_i / \sum_i l'_i} \\ &= \frac{l_i}{\sum_i l_i}\end{aligned}$$

Thus, we find that the probability of choosing an axis parallel cut in dimension $i$ that splits $S$ is that same as choosing an axis parallel cut in that splits $S \cup \{p\}$ in dimension $i$, conditioned on not isolating $p$ from all points of $S$. □

Now we can prove the niceness of DELETE.

**Theorem 5.** *If $T$ were drawn from a distribution $RRCT(S)$ then Algorithm 2 produces a tree $T'$ which is drawn at random from the probability distribution $RRCT(S - \{p\})$ via Algorithm 3.*

*Proof.* In this theorem we will consider three cases illustrated in Figure 8.

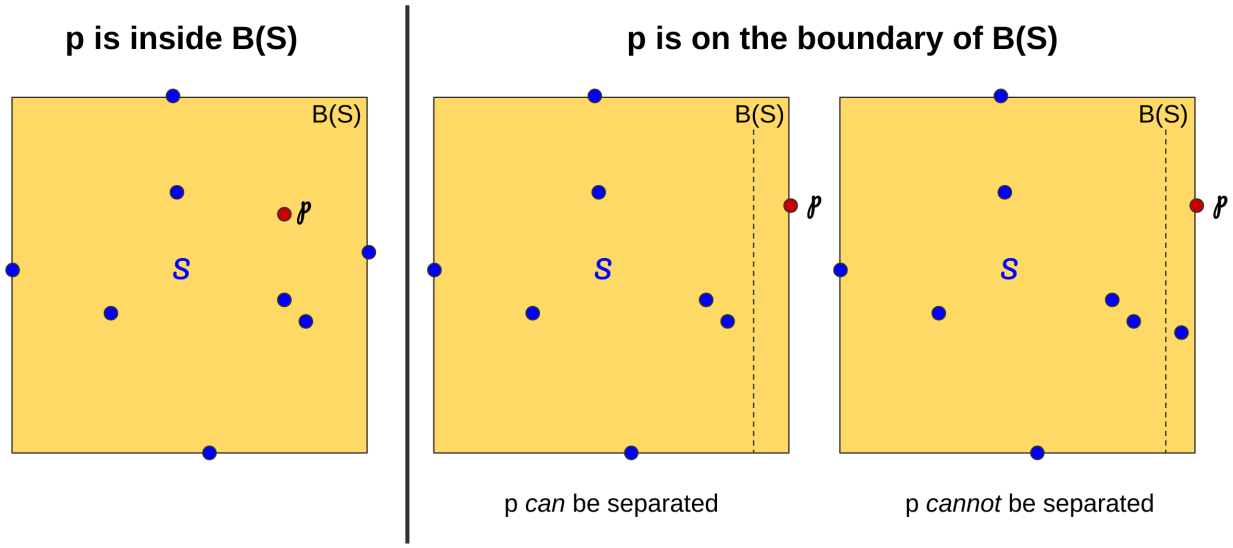


Figure 8: **Cases for Theorem 5**: The three cases for the proof.

This proof uses the idea of stochastically coupling decisions of the split operation – mirroring the same split in $T'$ as in $T$ [16]. Thus, if we only consider one tree, despite the cuts being correlated, it would appear that they are drawn from the correct distribution.

(a) $p$ **inside** $B(S)$ **in dimension** $i$. Suppose that we choose the dimension $i$ in splitting $T$ and the point $p$ does not lie on the bounding box of $S$ in dimension $i$. In this case, the distribution of cuts with or without $p$ are the same, so the construction of $T'$ can choose the same dimension $i$ and cut as in $T$. After this cut $p$ belongs to one side of the cut, and we recurse on that side. The other side will produce the same trees regardless of $p$.

(b) $p$ **on the boundary of** $B(S)$ **in dimension** $i$. We have two cases:

(i) $p$ **is separated from the rest of the points.** In this case $T$ produces a sibling tree that does not contain $p$ since it was isolated. This sibling tree is randomly drawn from $RRCT(S - \{p\})$ by the second part of Theorem 4.

(ii) $p$ **is not separated from the rest of the points.** By Theorem 4 conditioned on the fact that $p$ is not separated from $S$ we are choosing a random cut which separates $S - \{p\}$ along dimension $i$. This is an appropriate choice for $T'$ and we choose the same cut. We recurse on the side that $p$ belongs to for a complete proof.

Since these cases are exhaustive, the DELETE procedure is nice with respect to our distributions. □

A similar proof can be made for INSERT.

Finally, the DELETE operation can be performed in $O(d)$ times the depth of point $p$ because we go down to the point, updating the bounding box of $d$ dimensions. Similarly, the INSERT operation can be performed in $O(d)$ times the maximum depth of $T$, which is typically sub-linear in $|S|$ because consider $d$ dimensional bounding boxes and iteratively move down the tree with only two simple cases. Since these operations are sub-linear in $|S|$ we only need to do some form of sampling, such as reservoir sampling [17] or recency biased weighted random sampling [18], of a fixed size, $|S|$, to maintain a forest in sub-linear update time and $O(d|S|)$ space.

## 5. Bibliography


[1] V. Chandola, A. Banerjee, V. Kumar, Anomaly detetion: A survey, ACM Computing Surveys 9 (2009) 1–72.

[2] D. Hawkins, Identification of Outliers, Monographs on applied probability and statistics, Chapman and Hall, 1980.

[3] A. Emmott, S. Das, T. Dietterich, A. Fern, W.-K. Wong, A Meta-Analysis of the Anomaly Detection Problem, ArXiv e-prints (2015).

[4] M. A. F. Pimentel, D. A. Clifton, L. Clifton, L. Tarassenko, A review of novelty detection, Signal Processing 99 (2014) 215–249.

[5] L. Breiman, Random forests, Machine learning 45 (2001) 5–32.

[6] K. Fawagreh, M. M. Gaber, E. Elyan, Random forests: from early developments to recent advancements, Systems Science & Control Engineering 2 (2014) 602–609.

[7] M. Robnik-Šikonja, Improving random forests, in: J.-F. Boulicaut, F. Esposito, F. Giannotti, D. Pedreschi (Eds.), Machine Learning: ECML 2004, Springer Berlin Heidelberg, Berlin, Heidelberg, 2004, pp. 359–370.

[8] T. Shi, S. Horvath, Unsupervised learning with random forest predictors, Journal of Comptational and Graphical Statistics 15 (2006) 118–138.

[9] D. Baron, D. Poznanski, The weirdest sdss galaxies: results from an outlier detection algorithm, Monthly Notices of the Royal Astronomical Society 465 (2017) 4530–4555.

[10] F. T. Liu, K. M. Ting, Z. Zhou, Isolation forest, in: 2008 Eighth IEEE International Conference on Data Mining, pp. 413–422.

[11] S. Guha, N. Mishra, G. Roy, O. Schrijvers, Robust random cut forest based anomaly detection on streams, in: International conference on machine learning, pp. 2712–2721.

[12] S. Hariri, M. Kind, R. Brunner, Extended isolation forest, ? ? (2018) ?

[13] F. T. Liu, K. M. Ting, Z.-H. Zhou, On detecting clustered anomalies using sciforest, in: Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Springer, pp. 274–290.

[14] Z. Ding, M. Fei, An anomaly detection approach based on isolation forest algorithm for streaming data using sliding window, IFAC Proceedings Volumes 46 (2013) 12–17.

[15] S. C. Tan, K. M. Ting, T. F. Liu, Fast anomaly detection for streaming data, in: Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence.

[16] T. Lindvall, Lectures on the coupling method, Courier Corporation, 2002.

[17] J. S. Vitter, Random sampling with a reservoir, ACM Transactions on Mathematical Software (TOMS) 11 (1985) 37–57.

[18] P. S. Efraimidis, P. G. Spirakis, Weighted random sampling with a reservoir, Information Processing Letters 97 (2006) 181–185.