Robust Random Cut Forest Based Anomaly Detection On Streams

Sudipto Guha SUDIPTO@CIS.UPENN.EDU
University of Pennsylvania, Philadelphia, PA 19104.

Nina Mishra NMISHRA@AMAZON.COM
Amazon, Palo Alto, CA 94303.

Gourav Roy GOURAVR@AMAZON.COM
Amazon, Bangalore, India 560055.

Okke Schrijvers OKKES@CS.STANFORD.EDU
Stanford University, Palo Alto, CA 94305.

Abstract

In this paper we focus on the anomaly detection problem for dynamic data streams through the lens of random cut forests. We investigate a robust random cut data structure that can be used as a sketch or synopsis of the input stream. We provide a plausible definition of non-parametric anomalies based on the influence of an unseen point on the remainder of the data, i.e., the externality imposed by that point. We show how the sketch can be efficiently updated in a dynamic data stream. We demonstrate the viability of the algorithm on publicly available real data.

1. Introduction

Anomaly detection is one of the cornerstone problems in data mining. Even though the problem has been well studied over the last few decades, the emerging explosion of data from the internet of things and sensors leads us to reconsider the problem. In most of these contexts the data is streaming and well-understood prior models do not exist. Furthermore the input streams need not be append only, there may be corrections, updates and a variety of other dynamic changes. Two central questions in this regard are (1) how do we define anomalies? and (2) what data structure do we use to efficiently detect anomalies over dynamic data streams? In this paper we initiate the formal study of both of these questions. For (1), we view the problem from the perspective of model complexity and say that a point is an anomaly if the complexity of the model increases substantially with the inclusion of the point. The labeling of a point is data dependent and corresponds to the externality imposed by the point in explaining the remainder of the data. We extend this notion of externality to handle "outlier masking" that often arises from duplicates and near duplicate records. Note that the notion of model complexity has to be amenable to efficient computation in dynamic data streams. This relates question (1) to question (2) which we discuss in greater detail next. However it is worth noting that anomaly detection is not well understood even in the simpler context of static batch processing and (2) remains relevant in the batch setting as well.

For question (2), we explore a randomized approach, akin to (Liu et al., 2012), due in part to the practical success reported in (Emmott et al., 2013). Randomization is a powerful tool and known to be valuable in supervised learning (Breiman, 2001). But its technical exploration in the context of anomaly detection is not well-understood and the same comment applies to the algorithm put forth in (Liu et al., 2012). Moreover that algorithm has several limitations as described in Section 4.1. In particular, we show that in the presence of irrelevant dimensions, crucial anomalies are missed. In addition, it is unclear how to extend this work to a stream. Prior work attempted solutions (Tan et al., 2011) that extend to streaming, however those were not found to be effective (Emmott et al., 2013). To address these limitations, we put forward a sketch or synopsis termed *robust random cut forest* (RRCF) formally defined as follows.

Definition 1 *A* **robust random cut tree** *(RRCT) on point set S is generated as follows:*

1. *Choose a random dimension proportional to $\frac{\ell_i}{\sum_j \ell_j}$, where $\ell_i = \max_{x \in S} x_i - min_{x \in S} x_i$.*
2. *Choose $X_i \sim$ Uniform$[\min_{x \in S} x_i, \max_{x \in S} x_i]$*
3. *Let $S_1 = \{x | x \in S, x_i \leq X_i\}$ and $S_2 = S \setminus S_1$ and recurse on S_1 and S_2.*

Proceedings of the 33^{rd} International Conference on Machine Learning, New York, NY, USA, 2016. JMLR: W&CP volume 48.

A robust random cut forest (RRCF) is a collection of independent RRCTs.

The approach in (Liu et al., 2012) differs from the above procedure in Step (1) and chooses the dimension to cut uniformly at random. We discuss this algorithm in more detail in Section 4.1 and provide extensive comparison.

Following question (2), we ask: Does the RRCF data structure contain sufficient information that is independent of the specifics of the tree construction algorithm? In this paper we prove that the RRCF data structure approximately preserves distances in the following sense:

Theorem 1 *Consider the algorithm in Definition 1. Let the weight of a node in a tree be the corresponding sum of dimensions $\sum_i \ell_i$. Given two points $u, v \in S$, define the tree distance between u and v to be the weight of the least common ancestor of u, v. Then the 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 $L_1(u, v)$.*

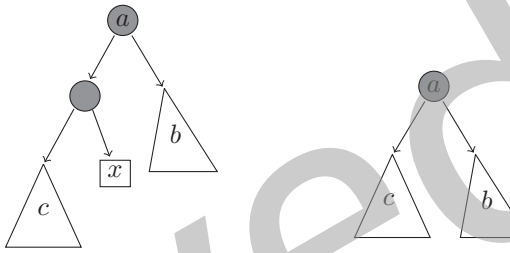
Theorem 1 provides a *low stretch distance preserving embedding*, reminiscent of the Johnson-Lindenstrauss Lemma (Johnson & Lindenstrauss, 1984) using random projections for $L_2()$ distances (which has much better dependence on d). The theorem is interesting because it implies that if a point is far from others (as is the case with anomalies) that it will continue to be at least as far in a random cut tree in expectation. The proof of Theorem 1 follows along the same lines of the proof of approximating finite metric spaces by a collection of trees (Charikar et al., 1998). Most of the proofs appear in the supplementary material.

The theorem shows that if there is a lot of empty space around a point, i.e., $\gamma = \min_v L_1(u, v)$ is large, then we will isolate the point within $O(d \log |S|/\gamma)$ levels from the root. Moreover since for any $p \geq 1$, the p-normed distance satisfies $d^{1-1/p} L_p(u, v) \geq L_1(u, v) \geq L_p(u, v)$ and therefore the early isolation applies to all large $L_p()$ distances *simultaneously*. This provides us a pointer towards the success of the original isolation forest algorithm in low to moderate dimensional data, because d is small and the probability of choosing a dimension is not as important if they are small in number. Thus the RRCF ensemble contains sufficient information that allows us to determine distance based anomalies, without focusing on the specifics of the distance function. Moreover the distance scales are adjusted appropriately based on the empty spaces between the points since the two bounding boxes may shrink after the cut.

Suppose that we are interested in the sample maintenance problem of producing a tree at random (with the correct probability) from $\mathcal{T}(S - \{x\})$ or from $\mathcal{T}(S \cup \{x\})$. In this paper we prove that we can efficiently insert and delete points into a random cut tree.

Theorem 2 (Section 3) *Given a tree T drawn according to $\mathcal{T}(S)$; if we delete the node containing the isolated point x and its parent (adjusting the grandparent accordingly, see Figure 2), then the resulting tree T' has the same probability as if being drawn from $\mathcal{T}(S - \{x\})$. Likewise, we can produce a tree T'' as if drawn at random from $T(S \cup \{x\})$ is time which is $O(d)$ times the maximum depth of T, which is typically sublinear in $|T|$.*

Theorem 2 demonstrates an intuitively *natural* behavior when points are deleted — as shown in the schematic in Figure 1. In effect, if we insert x, perform a few more operations and then delete x, then not only do we preserve distributions but the trees remain very close to each other — as if the insertion never happened. This behavior is a classic desiderata of sketching algorithms.



(a) Before: T (b) After: T'

Figure 1. Decremental maintenance of trees.

The natural behavior of deletions is not true if we do not choose the dimensions as in Step (1) of RRCF construction. For example, if we choose the dimensions uniformly at random as in (Liu et al., 2012), suppose we build a tree for $(1, 0), (\epsilon, \epsilon), (0, 1)$ where $1 \gg \epsilon > 0$ and then delete $(1, 0)$. The probability of getting a tree over the two remaining points that uses a vertical separator is $3/4 - \epsilon/2$ and not $1/2$ as desired. The probability of getting that tree in the RRCF process (after applying Theorem 2) is $1 - \epsilon$, as desired. This natural behavior under deletions is also not true of most space partitioning methods –such as quadtrees (Finkel & Bentley, 1974), kd-trees (Bentley, 1975), and R-trees (Guttman, 1984). The dynamic maintenance of a distribution over trees in a streaming setting is a novel contribution to the best of our knowledge and as a consequence, we can efficiently maintain a tree over a sample of a stream:

Theorem 3 *We can maintain a random tree over a sample S even as the sample S is updated dynamically for streaming data using sublinear update time and $O(d|S|)$ space.*

We can now use reservoir sampling (Vitter, 1985) to maintain a uniform random sample of size $|S|$ or a recency-biased weighted random sample of size $|S|$ (Efraimidis & Spirakis, 2006), in space proportional to $|S|$ on the fly. In effect, the random sampling process is now orthogonal from the robust random cut forest construction. For example to produce a sample of size $\rho|S|$ for $\rho < 1$, in an uniform random sampling we can perform straightforward rejection sampling; in the recency biased sample

in (Efraimidis & Spirakis, 2006) we need to delete the $(1-\rho)|S|$ lowest priority points. This notion of downsampling via deletions is supported perfectly by Theorem 2 – even for downsampling rates that are determined after the trees have been constructed, during postprocessing. Thus,

Theorem 4 *Given a tree $T(S)$ for sample S, if there exists a procedure that downsamples via deletion, then we have an algorithm that simultaneously provides us a downsampled tree for every downsampling rate.*

Theorems 3 and 4 taken together separate the notion of sampling from the analysis task and therefore eliminates the need to fine tune the sample size as an initial parameter. Moreover the dynamic maintenance of trees in Theorem 3 provides a mechanism to answer counterfactual questions as given in Theorem 5.

Theorem 5 *Given a tree $T(S)$ for sample S, and a point p we can efficiently compute a random tree in $\mathcal{T}(S \cup \{p\})$, and therefore answer questions such as: what would have been the expected depth had p been included in the sample?*

The ability to answer these counterfactual questions are critical to determining anomalies. Intuitively, we label a point p as an anomaly when the joint distribution of including the point is significantly different from the distribution that excludes it. Theorem 5 allows us to efficiently (pretend) sketch the joint distribution including the point p. However instead of measuring the effect of the sampled data points on p to determine its label (as is measured by notions such as expected depth), it stands to reason that we should measure the effect of p on the sampled points. This leads us to the definition of anomalies used in this paper.

2. Defining Anomalies

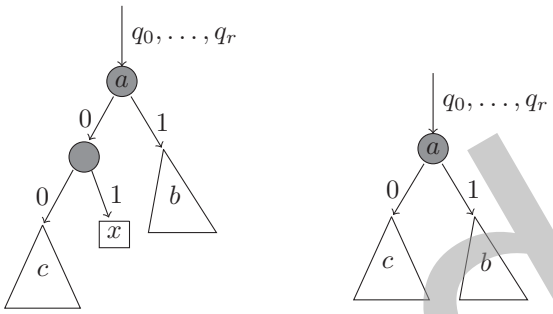
Consider the hypotheses:

(a) An anomaly is often easy to describe – consider Waldo wearing a red fedora in a sea of dark felt hats. While it may be difficult for us to find Waldo in a crowd, if we could forget the faces and see the color (as is the case when Waldo is revealed by someone else) then the recognition of the anomaly is fairly simple.

(b) An anomaly makes it harder to describe the remainder of the data – if Waldo were not wearing the red fedora, we may not have admitted the possibility that hats can be colored. In essence, an anomaly displaces our **attention** from the normal observation to this new one.

The fundamental task is therefore to quantify the shift in attention. Suppose that we assign left branches the bit 0 and right branches the bit 1 in a tree in a random cut forest. Now consider the bits that specify a point (excluding the bits that are required to store the attribute values of the point itself). This defines the complexity of a random model M_T which in our case corresponds to a tree T that fits the initial data. Therefore the number of bits required to express a point corresponds to its depth in the tree.

Given a set of points Z and a point $y \in Z$ let $f(y, Z, T)$ be the depth of y in tree T. Consider now the tree produced by deleting x as in Theorem 2 as $T(Z - \{x\})$. Note that given T and x the tree $T(Z-\{x\})$ is uniquely[1] determined. Let the depth of y in $T(Z - \{x\})$ be $f(y, Z - \{x\}, T)$ (we drop the qualification of the tree in this notation since it is uniquely defined).



(a) Tree $T(Z)$ (b) Tree $T(Z - \{x\})$

Figure 2. A correspondence of trees

Consider now a point y in the subtree c in Figure 2a. Its bit representation in T would be $q_0, \ldots, q_r, 0, 0, \ldots$. The model complexity, denoted as $|M(T)|$ the number of bits required to write down the description of all points y in tree T therefore will be $|M(T)| = \sum_{y \in Z} f(y, Z, T)$. If we were to remove x then the new model complexity is

$$|M(T')| = \sum_{y \in Z - \{x\}} f(y, Z - \{x\}, T')$$

where $T' = T(Z - \{x\})$ is a tree over $Z - \{x\}$. Now consider the expected **change** in model complexity under a random model. However since we have a many to one mapping from $T(Z)$ to $T(Z - \{x\})$ as a consequence of Theorem 2, we can express the second sum over $T(Z)$ instead of $T' = T(Z - \{x\})$ and we get

$$\begin{aligned} &\mathbb{E}_{T(Z)}\left[|M(T)|\right] - \mathbb{E}_{T(Z-\{x\})}\left[|M(T(Z-\{x\})|\right] \\ &= \sum_T \sum_{y \in Z - \{x\}} \mathbb{P}r\left[T\right] \left(f(y, Z, T) - f(y, Z - \{x\}, T') \right) \\ &+ \sum_T \mathbb{P}r\left[T\right] f(x, Z, T) \end{aligned} \tag{1}$$

Definition 2 *Define the bit-displacement or displacement of a point x to be the increase in the model complexity of all other points, i.e., for a set Z, to capture the externality introduced by x, define, where $T' = T(Z - \{x\})$,*

$$\textsc{Disp}(x, Z) = \sum_{T, y \in Z - \{x\}} \mathbb{P}r\left[T\right] \left(f(y, Z, T) - f(y, Z - \{x\}, T') \right)$$

[1]The converse is not true, this is a many-to-one mapping.

Note the total change in model complexity is $\text{DISP}(x, Z) + g(x, Z)$ where $g(x, Z) = \sum_T \mathbb{P}r[T] f(x, Z, T)$ is the expected depth of the point x in a random model. Instead of postulating that anomalies correspond to large $g()$, we focus on larger values of $\text{DISP}()$. The name displacement is clearer based on this lemma:

Lemma 1 *The expected displacement caused by a point x is the expected number of points in the sibling node of the leaf node containing x, when the partitioning is done according to the algorithm in Definition 1.*

Shortcomings. While Definition 2 points towards a possible definition of an anomaly, the definition as stated are not robust to duplicates or near-duplicates. Consider one dense cluster and a point p far from away from the cluster. The displacement of p will be large. But if there is a point q very close to p, then q's displacement in the presence of p is small. This phenomenon is known as outlier masking. Duplicates and near duplicates are natural and therefore the semantics of any anomaly detection algorithm has to accommodate them.

Duplicate Resilience. Consider the notion that Waldo has a few friends who help him hide – these friends are **colluders**; and if we were to get rid of all the colluders then the description changes significantly. Specifically, instead of just removing the point x we remove a set C with $x \in C$. Analogous to Equation (1) ,

$$\mathbb{E}_{T(Z)}[|M(T)|] - \mathbb{E}_{T(Z-C)}[|M(T(Z-C)|] = \text{DISP}(C, Z) + \sum_T \sum_{y \in C} \mathbb{P}r[T] f(y, Z, T) \quad (2)$$

where $\text{DISP}(C, Z)$ is the notion of displacement extended to subsets denoted as, where $T'' = T(Z - C)$,

$$\sum_{T, y \in Z-C} \mathbb{P}r[T] \Big(f(y, Z, T) - f(y, Z - C, T'') \Big) \quad (3)$$

Absent of any domain knowledge it appears that the displacement should be attributed equally to all the points in C. Therefore a natural choice of determining C seems to be $\max \text{DISP}(C, Z)/|C|$ subject to $x \in C \subseteq Z$. However two problems arise. First there are too many subsets C, and second, in a streaming setting it is likely we would be using a sample $S \subset Z$. Therefore the supposedly natural choice does not extend to samples. To avoid both issues, we allow the choice of C to be different for different samples S; in effect we are allowing Waldo to collude with different members in different tests! This motivates the following:

Definition 3 *The* **Collusive Displacement** *of x denoted by $\text{CODISP}(x, Z, |S|)$ of a point x is defined as*

$$\mathop{\mathbb{E}}_{S \subseteq Z, T} \left[\max_{x \in C \subseteq S} \frac{1}{|C|} \sum_{y \in S - C} \Big(f(y, S, T) - f(y, S - C, T'') \Big) \right]$$

Lemma 2 $\text{CODISP}(x, Z, |S|)$ *can be estimated efficiently.*

While $\text{CODISP}(x, Z, |S|)$ is dependent on $|S|$, the dependence is not severe. We envision using the largest sample size which is permitted under the resource constraints. We arrive at the central characterization we use in this paper:

Definition 4 *Outliers correspond to large* $\text{CODISP}()$.

3. Forest Maintenance on a Stream

In this section we discuss how Robust Random Cut Trees can be dynamically maintained. In the following, let $RRCF(S)$ be a the distribution over trees by running Definition 1 on S. Consider the following operations:

Insertion: Given T drawn from distribution $RRCF(S)$ and $p \notin S$ produce a T' drawn from $RRCF(S \cup \{p\})$.

Deletion: Given T drawn from distribution $RRCF(S)$ and $p \in S$ produce a T' drawn from $RRCF(S - \{p\})$. We need the following simple observation.

Observation 1 *Separating a point set S and p using an axis-parallel cut is possible if and only if it is possible to separate the minimal axis-aligned bounding box $B(S)$ and p using an axis-parallel cut.*

The next lemma provides a structural property about RRCF trees. We are interest in incremental updates with as few changes as possible to a set of trees. Note that given a specific tree we have two exhaustive cases, that (i) the new point which is to be deleted (respectively inserted) is not separated by the first cut and (ii) the new point is deleted (respective inserted) is separated by the first cut. Lemma 3 addresses these for collections of trees (not just a single tree) that satisfy (i) and (ii) respectively.

Lemma 3 *Given point p and 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 a dimension i that splits S using the weighted isolation forest 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 $RRCF(S \cup \{p\})$, conditioned on the fact the first cut isolates p from all points of S, the remainder of the tree is a random tree in $RRCF(S)$.

3.1. Deletion of Points

We begin with Algorithm 1 which is deceptively simple.

Algorithm 1 Algorithm ForgetPoint.

1: Find the node v in the tree where p is isolated in T.
2: Let u be the sibling of v. Delete the parent of v (and of u) and replace that parent with u (i.e., we short circuit the path from u to the root).
3: Update all bounding boxes starting from u's (new) parent upwards – this state is not necessary for deletions, but is useful for insertions.
4: Return the modified tree T'.

Lemma 4 *If T were drawn from the distribution $RRCF(S)$ then Algorithm 1 produces a tree T' which is drawn at random from the probability distribution $RRCF(S - \{p\})$.*

Lemma 5 *The deletion operation can be performed in time $O(d)$ times the depth of point p.*

Observe that if we delete a random point from the tree, then the running time of the deletion operation is $O(d)$ times the expected depth of any point. Likewise if we delete points whose depth is shallower than most points in the tree then we can improve the running time of Lemma 5.

3.2. Insertion of Points

Given a tree T from $RRCF(S)$ we produce a tree T' from the distribution $RRCF(S \cup \{p\})$. The algorithm is provided in Algorithm 2. Once again we will couple the decisions that is mirror the same split in T' as in T, as long as p is not outside a bounding box in T. Up to this point we are performing the same steps as in the construction of the forest on $S \cup \{p\}$, with the same probability.

Lemma 6 *If T were drawn from the distribution $RRCF(S)$ then Algorithm 1 produces a tree T' which is drawn at random from the probability distribution $RRCF(S \cup \{p\})$.*

4. Isolation Forest and Other Related Work

4.1. The Isolation Forest Algorithm

Recall that the isolation forest algorithm uses an ensemble of trees similar to those constructed in Definition 1, with the modification that the dimension to cut is chosen uniformly at random. Given a new point p, that algorithm follows the cuts and compute the average depth of the point across a collection of trees. The point is labeled an anomaly if the score exceeds a threshold; which corresponds to average depth being small compared to $\log |S|$ where S is suitably sized sample of the data.

The advantage of the isolation forest is that different dimensions are treated independently and the algorithm is invariant to scaling different dimensions differently. However consider the following example.

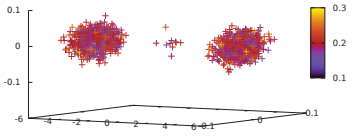
Algorithm 2 Algorithm InsertPoint.

1: We have a set of points S' and a tree $T(S')$. We want to insert p and produce tree $T'(S' \cup \{p\}$.
2: If $S' = \emptyset$ then we return a node containing the single node p.
3: Otherwise S' has a bounding box $B(S') = [x_1^\ell, x_1^h] \times [x_2^\ell, x_2^h] \times \cdots [x_d^\ell, x_d^h]$. Let $x_i^\ell \leq x_i^h$ for all i.
4: For all i let $\hat{x}_i^\ell = \min\{p_i, x_i^\ell\}$ and $\hat{x}_i^h = \max\{x_i^h, p_i\}$.
5: Choose a random number $r \in [0, \sum_i (\hat{x}_i^h - \hat{x}_i^\ell)]$.
6: This r corresponds to a specific choice of a cut in the construction of $RRCF(S' \cup \{p\})$. For instance we can compute $\arg\min\{j | \sum_{i=1}^{j} (\hat{x}_i^h - \hat{x}_i^\ell) \geq r\}$ and the cut corresponds to choosing $\hat{x}_j^\ell + \sum_{i=1}^{j} (\hat{x}_i^h - \hat{x}_i^\ell) - r$ in dimension j.
7: If this cut separates S' and p (i.e., is not in the interval $[x_j^\ell, x_j^h]$) then and we can use this as the first cut for $T'(S' \cup \{p\})$. We create a node – one side of the cut is p and the other side of the node is the tree $T(S')$.
8: If this cut does not separate S' and p then we throw away the cut! We choose the exact same dimension as $T(S')$ in $T'(S' \cup \{p\})$ and the exact same value of the cut chosen by $T(S')$ and perform the split. The point p goes to one of the sides, say with subset S''. We repeat this procedure with a smaller bounding box $B(S'')$ of S''. For the other side we use the same subtree as in $T(S')$.
9: In either case we update the bounding box of T'.

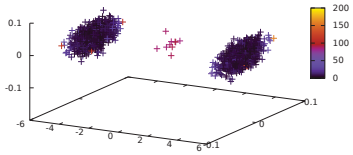
Example 1 (IRRELEVANT DIMENSIONS.) *Suppose we have two clusters of* 1000 *points each corresponding to $x_1 = \pm 5$ in the first dimension, and $x_i = 0$ in all remaining dimensions i. In all coordinates (including x_1) we add a random Gaussian noise with mean* 0 *and standard deviation* 0.01 *simulating white noise. Now consider* 10 *points with $x_1 = 0$ and the same behavior in all the other coordinates. When $d = 2$ the small cluster of points in the center is easily separated by the isolation forest algorithm which treats the dimensions independently. When $d = 30$ the vast majority of cuts are in irrelevant dimensions, and the algorithm fails (when run on entire data) as shown in Figure 1a for a single trial over* 100 *trees. For* 10 *trials (for the same data set), the algorithm determined that* 430*,* 270*,* 147*,* 220*,* 48*,* 244*,* 193*,*158*,* 250 *and* 103 *points had the same of higher anomaly score than the point with the highest anomaly score among the* 10 *points (the identity of this point varied across the trials).*

In essence, the algorithm either produces too many false alarms or does not have good recall. Note that AUC is not a relevant measure here since the class sizes between anomalous and non-anomalous are skewed, $1 : 200$. The results were consistent across multiple data sets generated according to the example. Figure 3b shows a corresponding single trial using CODISP(). The CODISP() measure places the 10 points in the largest 20 values most of the

time. Example 1 shows that scale independence therefore can be negative feature if distance is a meaningful concept in the dataset. However in many tasks that depend on detecting anomalies, the relevance of different dimensions is often unknown. The question of determining the appropriate scale of measurement often has far reaching consequences in data analysis.



(a) Performance of Isolation Forest (Liu et al., 2012). Note that the score never exceeds 0.3 whereas a score of 0.5 corresponds to an outlier. Note also that the two clusters are not distinguishable from the 10 points near origin outliers in depth values (color).



(b) Performance of CoDisp$(x, Z, |Z|)$. Observe that the clusters and outliers are separated; some of the extremal points in the clusters have the same (collusive) displacement as the 10 points near the origin, which is expected.

Figure 3. The result of running isolation forest and CoDisp() on the input in Example 1 for $d = 30$.

A modified version of the above example also is helpful in arguing why depth of a point is a not always helpful in characterizing anomalies, even in low dimensions. Consider,

Example 2 (Held Out Data.) *Consider the same dataset as in Example 1 in $d = 2$ dimensions. Suppose that we have only sampled 100 points and all the samples correspond to $x_1 = \pm 5$. Suppose we now want to evaluate: is the point $(0, 0)$ an anomaly? Based on the samples the natural answer is yes. The scoring mechanism of isolation forest algorithm fails because once the two clusters are separated, this new point $(0, 0)$ behaves as a point in one of the two other clusters! The situation however changes completely if we include $(0, 0)$ to build the trees.*

The example explains why the isolation forest algorithm is sensitive to sample size. However most anomalies are not usually seen in samples – anomaly detection algorithms should be measured on held out data. Note that Theorem 5 can efficiently solve the issue raised in Example 2 by answering the contrafactual question of what is the expected height has we observed $(0, 0)$ in the sample (without rebuilding the trees). However expected depth seems to generate more false alarms, as we investigate this issue further in the supplementary material.

4.2. Other Related Work

The problem of (unsupervised) outlier detection has a rich literature. We survey some of the work here; for an extensive survey see (Aggarwal, 2013; Chandola et al., 2009) and references therein. We discuss some of techniques which are unrelated to the concepts already discussed.

Perhaps the most obvious definition of an anomaly is density based outlier detection, which posits that a low-probability events are likely anomalous. This has led to different approaches based on estimating the density of data sets. For points in $\mathcal{R}^n$, Knorr & Ng (1997; 1998; 1999); Knorr et al. (2000) estimate the density by looking at the number of points that are within a ball of radius d of a given data point. The lower this number, the more anomalous the data point is. This approach may break down when different parts of the domain have different scales. To remedy this, there a methods (Breunig et al., 1999; 2000) that look at the density around a data point compared to its neighborhood. A variation of the previous approach is to consider a fixed k number of nearest neighbors and base the anomaly score on this (Eskin et al., 2002; Zhang & Wang, 2006). Here the anomaly score is monotonically increasing in the distances to the k nearest-neighbors. Taking the idea of density one step further, some authors have looked at finding structure in the data through clustering. The intuition here is that for points that cannot easily be assigned to a cluster, there is no good explanation for their existence. There are several clustering algorithms that work well to cluster part of the data, such as DBSCAN (Ester et al., 1996) and STREAM (Guha et al., 2003). Additionally, FindOut (Yu et al., 2002) removes points it cannot cluster, and then recurses. Finally the notion of sketching used in this paper is orthogonal to the notion used in (Huang & Kasiviswanathan, 2015) which uses streaming low rank approximation of the data.

5. Experiments

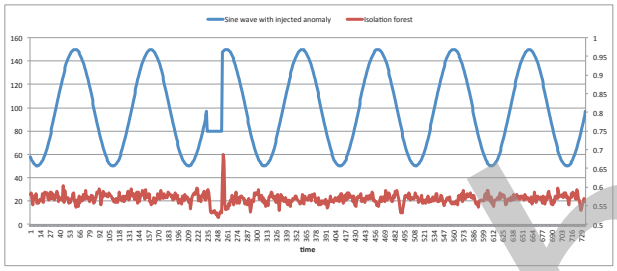
In the experiments, we focus on datasets where anomalies are visual, verifiable and interpretable. We begin with a synthetic dataset that captures the classic diurnal rhythm of human activity. We then move to a real dataset reflecting taxi ridership in New York City. In both cases, we compare the performance of RRCF with IF.

A technique that turns out to be useful for detecting anomalies in streams is shingling. If a shingle of size 4 is passed over a stream, the first 4 values of the stream received at time t_1, t_2, t_3, t_4 are treated as a 4-dimensional point. Then, at time t_5, the values at time t_2, t_3, t_4, t_5 are treated as as the next four-dimensional point. The window slides over one unit at each time step. A shingle encapsulates a typical shape of a curve – a departure from a typical shape could be an anomaly.

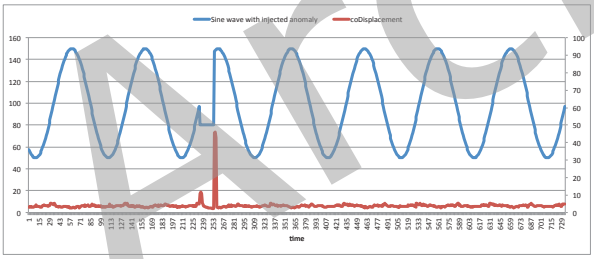
5.1. Synthetic Data

Many real datasets implicitly reflect human circadian rhythms. For example, an eCommerce site may monitor the number of orders it receives per hour. Search engines may monitor search queries or ad clicks per minute. Content delivery networks may monitor requests per minute. In these cases, there is a natural tendency to expect higher values during the day and lower values at night. An anomaly may reflect an unexpected dip or spike in activity.

In order to test our algorithm, we synthetically generated a sine wave where a dip is artificially injected around timestamp 500 that lasts for 20 time units. The goal is to determine if our anomaly detection algorithm can spot the beginning and end of the injected anomaly. The experiments were run with a shingle of length four, and one hundred trees in the forest, where each tree is constructed with a uniform random reservoir sample of 256 points. We treat the dataset as a stream, scoring a new point at time $t+1$ with the data structure built up until time t.



(a) The bottom red curve reflects the anomaly score produced by IF. Note that the start of the anomaly is missed.



(b) The bottom red curve represents the anomaly score produced by RRCF. Both the beginning and end of the anomaly are caught.

Figure 4. The top blue curve represents a sine wave with an artificially injected anomaly. The bottom red curve shows the anomaly score over time.

In Figure 4a, we show the result of running IF on the sine wave. For anomalies, detecting the onset is critical – and even more important than detecting the end of an anomaly. Note that IF misses the start of the anomaly at time 500. The end of the anomaly is detected, however, by then the system has come back to its normal state – it is not useful to fire an alarm once the anomaly has ended. Next, consider Figure 4b which shows the result of running RRCF on the same sine wave. Observe that the two highest scoring moments in the stream are the end and the beginning of the anomaly. The anomaly is successfully detected by RRCF. While the result of only a single run is shown, the experiment was repeated many times and the picture shown in Figure 4 is consistent across all runs.

5.2. Real Life Data: NYC Taxicabs

Next we conduct a streaming experiment using taxi ridership data from the NYC Taxi Commission[2]. We consider a stream of the total number of passengers aggregated over a 30 minute time window. Data is collected over a 7-month time period from 7/14 – 1/15. Note while this is a 1-dimensional datasets, we treat it as a 48-dimensional data set where each point in the stream is represented by a sliding window or shingle of the last day of data, ignoring the first day of data. The intuition is that the last day of activity captures a typical shape of passenger ridership.

The following dates were manually labeled as anomalies based on knowledge of holidays and events in NYC (Lavin & Ahmad, 2015): Independence Day (7/4/14-7/6/14), Labor Day (9/1/14), Labor Day Parade (9/6/14), NYC Marathon (11/02/14), Thanksgiving (11/27/14), Christmas (12/25/14), New Years Day (1/1/15), North American Blizzard (1/26/15-1/27/15). For simplicity, we label a 30-minute window an anomaly if it overlaps one of these days.

Stream We treat the data as a stream – after observing points $1, \ldots, i$, our goal is to score the $(i+1)$st point. The score that we produce for $(i+1)$ is based only on the previous data points $1, \ldots, i$, but not their labels. We use IF as the baseline. While a streaming version was subsequently published (Tan et al., 2011), since it was not found to improve over IF (Emmott et al., 2013), we consider a more straightforward adaptation. Since each tree in the forest is created based on a random sample of data, we simply build each tree based on a random sample of the stream, e.g., uniform or time-decayed as previously referenced. Our aim here is to compare to the baseline with respect to accuracy, not running time. Each tree can be updated in an embarrassingly parallel manner for a faster implementation.

Metrics To quantitatively evaluate our approach, we report on a number of precision/recall-related metrics. We learn a threshold for a good score on a training set and report the effectiveness on a held out test set. The training set contains all points before time t and the test set all points after time t. The threshold is chosen to optimize the F1-measure (harmonic mean of precision and recall). We focus our attention on positive precision and positive recall to avoid "boy who cried wolf" effects (Tsien & Fackler, 1997; Lawless, 1994).

[2] http://www.nyc.gov/html/tlc/html/about/trip_record_data.shtml

Table 1. Comparison of Baseline Isolation Forest to proposed Robust Random Cut Forest

Method	Sample Size	Positive Precision	Positive Recall	Negative Precision	Negative Recall	Accuracy	AUC
IF	256	0.42 (0.05)	0.37 (0.02)	0.96 (0.00)	0.97 (0.01)	0.93 (0.01)	0.83 (0.01)
RRCF	256	0.87 (0.02)	0.44 (0.04)	0.97 (0.00)	1.00 (0.00)	0.96 (0.00)	0.86 (0.00)
IF	512	0.48 (0.05)	0.37 (0.01)	0.97 (0.01)	0.96 (0.00)	0.94 (0.00)	0.86 (0.00)
RRCF	512	0.84 (0.04)	0.50 (0.03)	0.99 (0.00)	0.97 (0.00)	0.96 (0.00)	0.89 (0.00)
IF	1024	0.51 (0.03)	0.37 (0.01)	0.96 (0.00)	0.98 (0.00)	0.94 (0.00)	0.87 (0.00)
RRCF	1024	0.77 (0.03)	0.57 (0.02)	0.97 (0.00)	0.99 (0.00)	0.96 (0.00)	0.90 (0.00)

Method	Segment Precision	Segment Recall	Time to Detect Onset	Time to Detect End	Prec@5	Prec@10	Prec@15	Prec@20
IF	0.40 (0.09)	0.80 (0.09)	22.68 (3.05)	23.30 (1.54)	0.52 (0.10)	0.50 (0.00)	0.34 (0.02)	0.28 (0.03)
RRCF	0.65 (0.14)	0.80 (0.00)	13.53 (2.05)	10.85 (3.89)	0.58 (0.06)	0.49 (0.03)	0.39 (0.02)	0.30 (0.00)

Table 2. Segment-Level Metrics and Precision@K

For the finer granularity data in the taxi cab data set, we view the ground truth as segments of time when the data is in an anomalous state. Our goal is to quickly and reliably identify these segments. We say that a segment is identified in the test set if the algorithm produces a score over the learned threshold anytime during the segment (including the sliding window, if applicable).

Results In the experiments, there were 200 trees in the forest, each computed based on a random sample of 1K points. Note that varying the sample size does not alter the nature of our conclusions. Since ridership today is likely similar to ridership tomorrow, we set our time-decayed sampling parameter to the last two months of ridership. All results are averaged over multiple runs (10). Standard deviation is also reported. Figure 5 shows the result of the anomaly scores returned by CoDisp().

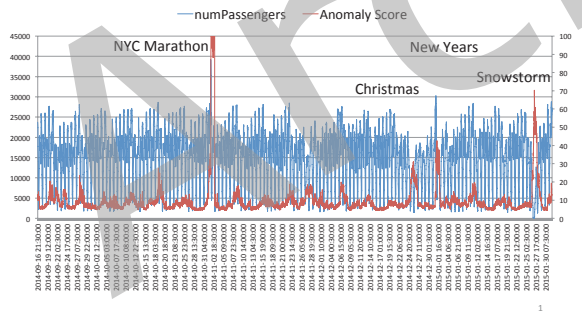


Figure 5. NYC taxi data and CoDisp(). Note that Thanksgiving is not captured.

In a more detailed evaluation, the first set of results (Table 1) show that the proposed RRCF method is more accurate than the baseline. Particularly noteworthy is RRCF's higher positive precision, which implies a lower false alarm rate. In Table 2, we show the segment-based results. Whereas Table 1 may give more credit for catching a long anomaly over a short one, the segment metric weighs each alarm equally. The proposed RRF method not only catches more alarms, but also catches them more quickly. The units are measured in 30 minute increments – so 11 hours on average to catch an alarm on the baseline and 7 hours for the RRCF method. These actual numbers are not as important here, since anomaly start/end times are labeled somewhat loosely. The difference in time to catch does matter. Precision@K is also reported in Table 2.

Discussion: Shingle size, if used, matters in the sense that shingles that are too small may catch naturally varying noise in the signal and trigger false alarms. On the other hand, shingles that are too large may increase the time it takes to find an alarm, or miss the alarm altogether. Time decay requires knowledge of the domain. Sample size choice had less effect – with varying sample sizes of 256, 512 and 1K the conclusions are unchanged on this dataset.

6. Conclusions and Future Work

We introduced the robust random cut forest sketch and proved that it approximately preserves pairwise distances. If the data is recorded in the correct scale, distance is crucially important to preserve for computations, and not just anomaly detection. We adopted a model-based definition of an anomaly that captures the differential effect of adding/removing a point on the size of the sketch. Experiments suggest that the algorithm holds great promise for fighting alarm fatigue as well as catching more missed alarms.

We believe that the random cut forest sketch is more beneficial than what we have established. For example, it may also be helpful for clustering since pairwise distances are approximately preserved. In addition, it may help detect changepoints in a stream. A *changepoint* is a moment in time t where before time t the data is drawn from a distribution D_1 and after time t the data is drawn from a distribution D_2, and D_1 is sufficiently different from D_2 (Kifer et al., 2004; Dasu et al., 2006). By maintaining a sequence of sketches over time, one may be able to compare two sketches to determine if the distribution has changed.

Acknowledgments

We thank Roger Barga, Charles Elkan, and Rajeev Rastogi for many insightful discussions. We also thank Dan Blick, Praveen Gattu, Gaurav Ghare and Ryan Nienhuis for their help and support.

References

Aggarwal, Charu C. *Outlier Analysis*. Springer New York, 2013.

Bentley, Jon Louis. Multidimensional binary search trees used for associative searching. *Commun. ACM*, 18(9): 509–517, September 1975. ISSN 0001-0782.

Breiman, Leo. Random forests. *Machine Learning*, pp. 5–32, 2001.

Breunig, Markus M, Kriegel, Hans-Peter, Ng, Raymond T, and Sander, Jörg. Optics-of: Identifying local outliers. In *PKDD*, pp. 262–270, 1999.

Breunig, Markus M, Kriegel, Hans-Peter, Ng, Raymond T, and Sander, Jörg. Lof: identifying density-based local outliers. In *ACM sigmod record*, volume 29, pp. 93–104, 2000.

Chandola, Varun, Banerjee, Arindam, and Kumar, Vipin. Anomaly detection: A survey. *ACM computing surveys (CSUR)*, 41(3):15, 2009.

Charikar, Moses, Chekuri, Chandra, Goel, Ashish, Guha, Sudipto, and Plotkin, Serge. Approximating a finite metric by a small number of tree metrics. *Proceedings of Foundations of Computer Science*, pp. 379–388, 1998.

Dasu, Tamraparni, Krishnan, Shankar, Venkatasubramanian, Suresh, and Yi, Ke. An information-theoretic approach to detecting changes in multi-dimensional data streams. In *In Proc. Symp. on the Interface of Statistics, Computing Science, and Applications*. Citeseer, 2006.

Efraimidis, Pavlos S. and Spirakis, Paul G. Weighted random sampling with a reservoir. *Information Processing Letters*, 97(5):181–185, 2006.

Emmott, Andrew F., Das, Shubhomoy, Dietterich, Thomas, Fern, Alan, and Wong, Weng-Keen. Systematic construction of anomaly detection benchmarks from real data. In *ACM SIGKDD Workshop on Outlier Detection and Description*, pp. 16–21, 2013.

Eskin, Eleazar, Arnold, Andrew, Prerau, Michael, Portnoy, Leonid, and Stolfo, Sal. A geometric framework for unsupervised anomaly detection. In Barbará, Daniel and Jajodia, Sushil (eds.), *Applications of Data Mining in Computer Security*, pp. 77–101, Boston, MA, 2002.

Ester, Martin, Kriegel, Hans-Peter, Sander, Jörg, and Xu, Xiaowei. A density-based algorithm for discovering clusters in large spatial databases with noise. In *KDD*, volume 96, pp. 226–231, 1996.

Finkel, R. A. and Bentley, J. L. Quad trees a data structure for retrieval on composite keys. *Acta Informatica*, 4(1): 1–9, 1974.

Guha, Sudipto, Meyerson, Adam, Mishra, Nina, Motwani, Rajeev, and O'Callaghan, Liadan. Clustering data streams: Theory and practice. *IEEE Trans. Knowl. Data Eng.*, 15(3):515–528, 2003.

Guttman, Antonin. R-trees: A dynamic index structure for spatial searching. In *SIGMOD*, pp. 47–57, 1984.

Huang, Hao and Kasiviswanathan, Shiva Prasad. Streaming anomaly detection using randomized matrix sketching. *Proceedings of the VLDB Endowment*, 9(3):192–203, 2015.

Johnson, William B. and Lindenstrauss, Joram. Extensions of lipschitz mappings into a hilbert space. *Contemporary Mathematics 26. Providence, RI: American Mathematical Society*, 1984.

Kifer, Daniel, Ben-David, Shai, and Gehrke, Johannes. Detecting change in data streams. In *VLDB*, pp. 180–191, 2004.

Knorr, Edwin M and Ng, Raymond T. A unified notion of outliers: Properties and computation. In *KDD*, pp. 219–222, 1997.

Knorr, Edwin M and Ng, Raymond T. Algorithms for mining distancebased outliers in large datasets. In *VLDB*, pp. 392–403, 1998.

Knorr, Edwin M and Ng, Raymond T. Finding intensional knowledge of distance-based outliers. In *VLDB*, volume 99, pp. 211–222, 1999.

Knorr, Edwin M, Ng, Raymond T, and Tucakov, Vladimir. Distance-based outliers: algorithms and applications. *VLDB Journal*, 8(3-4):237–253, 2000.

Lavin, Alexander and Ahmad, Subutai. Evaluating real-time anomaly detection algorithms-the numenta anomaly benchmark. *arXiv:1510.03336*, 2015.

Lawless, Stephen T. Crying wolf: false alarms in a pediatric intensive care unit. *Critical care medicine*, 22(6): 981–985, 1994.

Lindvall, T. *Lectures on the coupling method*. Wiley, New York, 1992.

Liu, Fei Tony, Ting, Kai Ming, and Zhou, Zhi-Hua. Isolation-based anomaly detection. *ACM Trans. Knowl. Discov. Data*, 6(1):3:1–3:39, March 2012.

Tan, Swee Chuan, Ting, Kai Ming, and Liu, Fei Tony. Fast anomaly detection for streaming data. In *IJCAI*, pp. 1511–1516, 2011.

Tsien, Christine L and Fackler, James C. Poor prognosis for existing monitors in the intensive care unit. *Critical care medicine*, 25(4):614–619, 1997.

Vitter, Jeffrey S. Random sampling with a reservoir. *ACM Transactions on Mathematical Software*, 11(1): 3757, 1985.

Yu, Dantong, Sheikholeslami, Gholamhosein, and Zhang, Aidong. Findout: finding outliers in very large datasets. *Knowledge and Information Systems*, 4(4):387–412, 2002.

Zhang, Ji and Wang, Hai. Detecting outlying subspaces for high-dimensional data: the new task, algorithms, and performance. *Knowledge and information systems*, 10 (3):333–355, 2006.