Anomaly Detection in Stream using RRCF

Introducing RRCF as a fast and stable learner in stream data and detecting anomalies using it.

Student:

Mohammad Doosti Lakhani

Instructor:

Dr. M. R. Kangavari

Course:

Machine Learning

Introduction

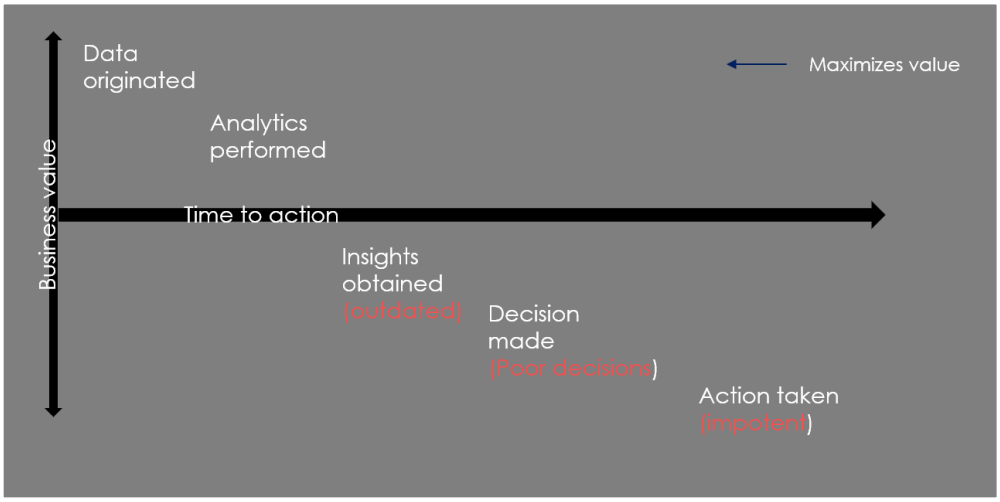
In this article we focus on the anomaly detection problem for dynamic data streams through the lens of random cut forests. Even though the problem has been well studied over the last few decades, the emerging explosion of data from the internet of things and sensors leads us to reconsider the problem. In most of these contexts the data is streaming and well-understood prior models do not exist.

But before going through concepts, let's discuss main reason that it gained attention. It can be argued in two sections:

1. The importance of stream data
2. Main challenge of stream data

These questions can be answered in many different ways but in this area but we can distinguish main reasons in this way:

1. Importance of stream data:
    a. All data originates in real time e.g. image segmentation, language models
    b. Emerging explosion of IoT
    c. It exists in everyday tasks
2. Main challenge of stream data:
    a. Batch operations take too long
    b. Insights are perishable



So, based on the aforementioned issues, a algorithm that can digest data as it is generated, process it on the fly and does real time machine learning is desirable. To address these issues, we define following questions:

Two central questions in this regard are:

1. how do we define anomalies?
2. what data structure do we use to efficiently detect anomalies over dynamic data streams?

In this report, this will be tackled.

Definitions

Let's first answer the two aforementioned questions in a precise manner to have some clue about the ideas.

For first question, the problem has been viewed 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.

For question two, randomized approaches have been considered as they are strong mainly in supervised fashion. But most of the algorithms based on random trees have been studied well for anomalies in a stream data situation. So, we deal with issue by proposing Robust Random Cut Tree.

Definition 1: RRCT

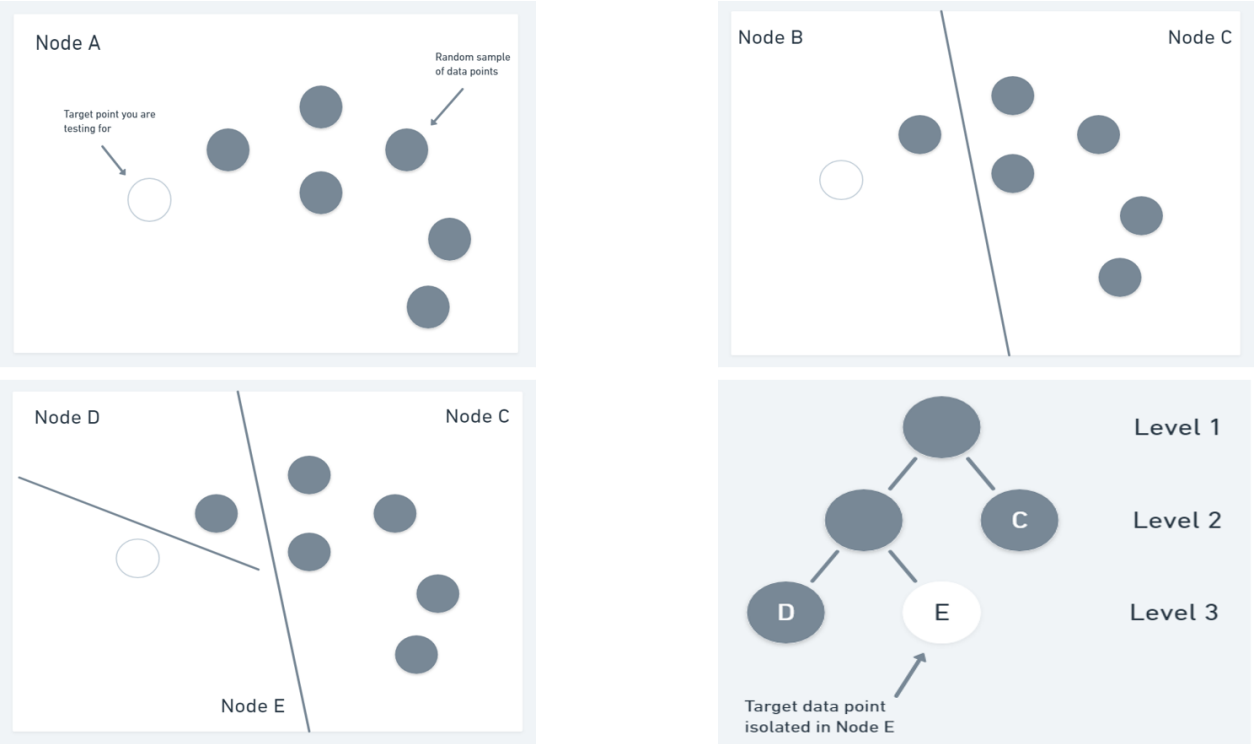
RRCT is an unsupervised but very fast type of decision tree. In below, we can see the main idea:

- Choose a random dimension proportional to:

  $$\frac{l_i}{\sum_j l_j}$$

  where $l_i = \max(x_i) - \min(x_i)$
- Choose $X_i \sim uniform(\min(x_i), \max(x_i))$
- Let $S_1 = \{x | x \in S, x_i < X_i\}$ and $S_2 = \frac{S}{S_1}$ and recurse on S_1 and S_2

To understand how it works, an example can be expressed which as follows:



To explain this image, on top left, we start by mentioning that we want to isolate that particular white node. To do so, we use random cuts on dimensions of the data so get top right image, and we continue doing until we isolate white note which we can see it as a node in the corresponding constructed tree in bottom right image.

This is the core of this article as an anomaly point can be expressed by its distance from root. Nodes that in the original data are far from the data points (outliers/anomalies) are still far but from root of tree. So, based on this idea, we may need to define anomaly once more.

Definition 2: Anomaly in RCF

Here are the major attributes of anomaly point that can be expressed:

- Anomaly points are isolated much faster and they will be on top of the tree
- The points near the root will get higher score
- The score is distance based, so the points far from normal clusters get higher score (nearer to root)
    - So, a point will be anomaly if it increases the size of tree profoundly
- Same idea has been used for test
    - If test node is near to root, then it is probably an anomaly

- If a point is far in from data in N-dim data, it will be as far relatively in RCF

Theorems

Theorem 1:

Consider the algorithm in Definition 1. Let the weight of a node in a tree be the corresponding sum of dimensions $\sum_i \boldsymbol{l_i}$ *Given two points* $\boldsymbol{u, v} \in \boldsymbol{S}$ *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* $\boldsymbol{L_1(u, v)}$ *and in expectation, at most* $\boldsymbol{O(d \,.\, log \frac{|S|}{L_1(u,v)})}$ *times* $\boldsymbol{L_1(u, v)}$.

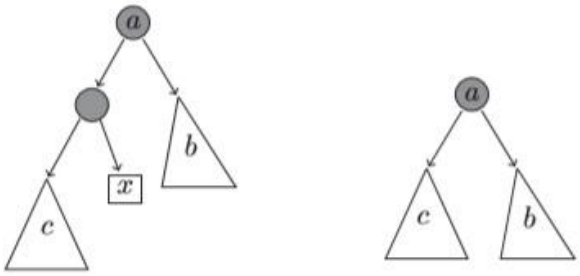
The theorem is interesting because it implies that if a point is far from others (as is the case with anomalies) that it will continue to be at least as far in a random cut tree in expectation. The proof of Theorem 1 follows along the same lines of the proof of approximating finite metric spaces by a collection of trees (Charikaretal.,1998).

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.

Theorem 2:

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

Theorem 2 demonstrates an intuitively natural behavior when points are deleted. 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 desideratum of sketching algorithms. In the below figure, we can see an intuitive demonstration of this theorem:



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

Theorem 3:

We can maintain a random tree over a sample $\boldsymbol{S}$ even as the sample $\boldsymbol{S}$ is updated dynamically for streaming data using sublinear update time and $\boldsymbol{O(d|S|)}$ 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

Theorem 4:

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

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.

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 distribution over trees by running Definition 1 on S. Consider the following operations.

Deletion

Given T drawn from distribution $RRCF(S)$ and $p \in S$ produce a T' drawn from $RRCF(S - \{p\})$.

Algorithm:

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 o u) and replace that parent with u (i.e., we short circui the path from u to the root).
3: Update all bounding boxes starting from u's (new) par ent upwards – this state is not necessary for deletions but is useful for insertions.
4: Return the modified tree T'.

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 $RRCF(S - \{p\})$.
We can derive inverse concept in the same manner for insertion.

Insertion

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

Algorithm:

Algorithm 2 Algorithm InsertPoint.

1: We have a set of points S' and a tree $T(S')$. We wan to insert p and produce tree $T'(S' \cup \{p\})$.
2: If $S' = \emptyset$ then we return a node containing the singl 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 th construction of $RRCF(S' \cup \{p\})$. For instance we ca compute $\arg\min\{j | \sum_{i=1}^{j} (\hat{x}_i^h - \hat{x}_i^\ell) \geq r\}$ and the cu corresponds to choosing $\hat{x}_j^\ell + \sum_{i=1}^{j} (\hat{x}_i^h - \hat{x}_i^\ell) - r$ i dimension j.
7: If this cut separates S' and p (i.e., is not in the interva $[x_j^\ell, x_j^h]$) then and we can use this as the first cut fo $T'(S' \cup \{p\})$. We create a node – one side of the cut i 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 throv away the cut! We choose the exact same dimension a $T(S')$ in $T'(S' \cup \{p\})$ and the exact same value of th cut chosen by $T(S')$ and perform the split. The point goes to one of the sides, say with subset S''. We repea this procedure with a smaller bounding box $B(S'')$ o S''. For the other side we use the same subtree as i $T(S')$.
9: In either case we update the bounding box of T'.

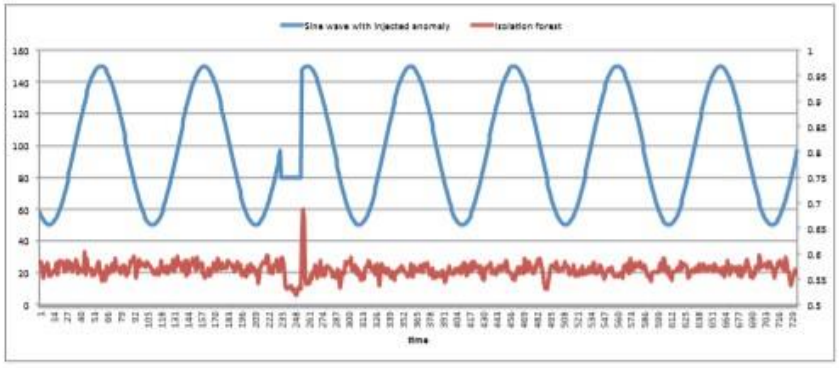
Applications

Real Life Data: NYC Taxi cabs

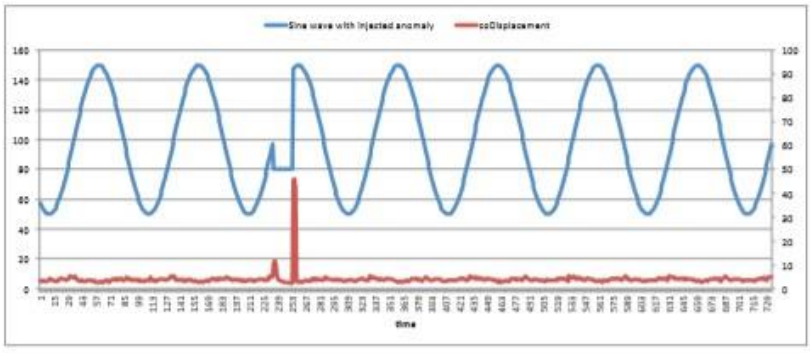
Authors considered data as a stream of the total number of passengers aggregated over a 30 minute time window. Data is collected over a 7month time period. Note while this is a 1-dimensional datasets, authors treat it as a 48-dimensional data set where each point in the stream is represented by a sliding window.
The famous holidays have been manually labeled as anomaly. For simplicity, authors label a 30 minute window an anomaly if it overlaps one of these days.

Authors 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. They use IF as the baseline.

In below images, we can see the behavior of these two algorithms regarding a anomaly window:



(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.

Also, in below image, we can see the differences between IF and $RRCF$ model w.r.t. multiple metrics for a better comparison. 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). Authors focus their attention on positive precision and positive recall.

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.

Robust Random Cut Forest Based Anomaly Detection On Streams

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

References

1. Guha, Sudipto, et al. "Robust random cut forest based anomaly detection on streams." International conference on machine learning. 2016
2. Wagner, Tal, et al. "Semi-supervised learning on data streams via temporal label propagation." International Conference on Machine Learning. 2018
3. Eswaran, Dhivya, et al. "Spotlight: Detecting anomalies in streaming graphs." Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. 2018
4. Some of images are barrowed from Manning Publications