EasyChair Preprint

№ 6379

An Efficient Anomaly Detection Approach using Cube Sampling with Streaming Data

Seemandhar Jain, Prarthi Jain and Abhishek Srivastava

August 26, 2021

An Efficient Anomaly Detection Approach using Cube Sampling with Streaming Data (paper ID:201)

No Author Given

No Institute Given

Abstract. Anomaly detection is critical in various fields, including intrusion detection, health monitoring, fault diagnosis, and sensor network event detection. The isolation forest (or *iForest*) approach is a well-known technique for detecting anomalies. It is, however, ineffective when dealing with dynamic streaming data, which is becoming increasingly prevalent in a wide variety of application areas these days. In this work, We proposed an efficient *iForest* based approach for anomaly detection using cube sampling that is effective on streaming data. Cube sampling is used in the initial stage to choose nearly balanced samples with equal or unequal inclusion probability, significantly reducing storage requirements while preserving efficiency. Following that, the data's streaming nature is addressed by a sliding window technique that generates consecutive chunks of data for systematic processing. The proposed approach is equally successful at detecting anomalies as existing state-of-the-art approaches while requiring significantly less storage and time complexity. We undertake empirical evaluations of the proposed approach using standard datasets and demonstrate that it outperforms traditional approaches in terms of Area Under the ROC Curve (AUC-ROC) and can handle high-dimensional streaming data.

Keywords: Anomaly Detection · Isolation Forest · Cube Sampling · Sliding window · Streaming data

1 Introduction

Anomalies are rare and distinct data patterns that vary from normal or anticipated behavior. According to Hawkins' definition [11], an anomaly, often referred to as an outlier, is significantly different from standard observable objects and is thought to have been created by a separate mechanism. Anomaly detection seeks to identify trends that deviate from anomalies or outliers. High dimensionality complicates anomaly identification because as the number of characteristics or features increases, the amount of data required to generalize properly increases as well, resulting in data sparsity or scattered and isolated data points. The data is primarily of streaming variety, which refers to information that flows in and out of a device similar to a stream. Continuously developing, sequential, and unlimited streaming data consists of a perpetual flow of data pieces

that evolve. This study is concerned with the identification of abnormalities in such streaming data. Due to high size, offline algorithms that attempt to store such flowing data ultimately can't handle high streams of data. The data must be handled sequentially or gradually on a data-by-data basis with the help of a sliding window technique [5, 9]. In the case of streaming data, where data is continuously generating, the model must be continually retrained or updated utilizing the incoming continuous data stream to minimize concept drift [7,8,21]. Concept drift is a typical occurrence in streaming data and must be considered while doing anomaly detection. Additionally, concept drift indicates that the trained model rapidly becomes antiquated and cannot produce accurate results since it is no longer compatible with the current data. The Isolation Forest (or *iForest*) algorithm is an excellent approach for anomaly identification that manages concept drift gracefully [13]. *iForest* is a well-known randomization-based method for detecting anomalies. Randomization is a strong technique that has been demonstrated to be beneficial in supervised learning [1]. The *iForest* algorithm's randomization mechanism is as follows: one attribute is picked randomly from the data's different features. Following that, a random value between the range of the characteristic is picked and the dataset is partitioned along this value. As a result of this partitioning, a binary tree is formed, with data points with values more significant than the randomly determined partitioning value constituting the right child and those with a smaller value becoming the left child. This technique is repeated repeatedly until all data points are separated. Due to the rarity and uniqueness of data points that may be deemed anomalies, the likelihood of such anomalies isolating sooner and closer to the root node is greater than for normal points. The first step in this work employs a balanced sampling approach (called cube sampling), which significantly reduces the size of the dataset and enables the iForest algorithm to be utilized effectively on streaming data with minimal computing cost. This includes the inclusion probability calculation detailed in subsequent parts, which is found to outperform previous methods for calculating the inclusion probability. Later, we present a method that makes efficient use of the *iForest* algorithm [13] with streaming data. *iForest* effectively accounts for the problem of concept drift that frequently occurs with streaming data and does not need costly distance computations, as other distances and density-based methods do. *iForest* is a randomization-based algorithm that may easily be expanded to handle streaming data. The proposed approach has several distinguishing characteristics, including the following: 1) significantly reduces the size of the data through cube sampling, allowing for detection of anomalies without the ample use of space and time; 2) effectively detects anomalies in streaming data; and 3) requires less time complexity to update the model, making it adaptable to the streaming data, thereby minimizing concept drift.

The following are the primary contributions of the paper.

1. The paper uses the cube sampling technique to minimize the amount of the dataset significantly.

2. The effectiveness of using a sliding window to manage flowing data is illustrated.
3. The proposed approach is evaluated on standard datasets and shown to be successful. AUC-ROC values are generally superior to and, in a few cases, equivalent to those of existing anomaly detection algorithms.
4. The algorithm's operation in a simulated environment is illustrated.

2 Related Work

Anomaly detection, also known as outlier identification, is the study of data patterns that do not comply with typical or expected behaviour. A comprehensive examination and survey of anomaly detection is accessible in [2,3,11]. A substantial amount of research has been conducted on the use of tree-based algorithms for outlier identification; notable examples are $iForest$ [13] and Extended $iForest$ [14,20]. While these efforts are beneficial for static data, they are ineffective for streaming data. To deal with constant data generation, model updates are required to avoid concept drift. We selected the balanced sample and updated the model using cube sampling. Sampling is the process of choosing a subset of the population to reflect the entire population accurately. Sampling techniques are classified as Probability Sampling techniques [4] and Non-Probability Sampling techniques [19]. [17] for a thorough study of sampling approaches. Prior to using unequal probability sampling, an inclusion probability must be determined. There have been efforts to assess the probability of inclusion for large datasets, such as Shastri et al. [16] and Nigam et al. [15]. This is not a feasible strategy for dealing with multi-dimensional data. This paper presents an improvised use of the $iForest$ Algorithm based on cube sampling that is both effective and memory efficient for detecting anomalies in streaming data.

3 Proposed Approach

In this paper, we have used the well-known *iForest* anomaly detection technique on streaming data, followed by cube sampling approach to select the balanced sample to update the model, that make the approach effective for high dimensional data and mitigate concept drift.

3.1 Proposed Approach

Due to the endless nature of streaming data, any offline approach would run out of memory if it tries to save the whole data. Because streaming data changes over time, the model must be updated on a frequent basis to minimize concept drift (as previously stated) and to maintain a high degree of accuracy.

The rank of each element in the forest is directly proportional to the number of anomalous points detected. So lower the number of anomalies, the better the $iTree$ performance. Hence, when the model is updated, the high rank trees are

Algorithm 1 *AnomlayDetection*($ntrees$, $\omega = 256$, X,$ktrees = 50$)

Input: $X(StreamingData) = \{X_1, X_2, \ldots, X_i, \ldots, \}$, $\omega-$ Sliding Window Size (=256), *ntrres* - No. of iTree in forest, *ktrees* - No. of iTree to be inserted and deleted, to update the model

Output: Anomalies

Initialize Forest $F \leftarrow []$
Sliding Window $Y \leftarrow []$
Initialize c=0*, which counts the number of data-points*
while c != ω *do*
    $c \leftarrow c + 1$
    Add X_i *in* Y
end while
Initialize the height of iTree $h \leftarrow ceil\,(log_2(\omega))$
for $i \leftarrow 0$ *to ntrees do*
    $F[i] \leftarrow iTree(Y,0,h)$
end for
Re-initialize $c \leftarrow \omega$ *and* $Y \leftarrow []$
while $c > 0$ *do*
    $c \leftarrow c - 1$
    Add X_i *in* Y
end while
report anomaly detector $G(Y)$
$S \leftarrow$ *Cube Sampling(*Y*)*
for $i \leftarrow$ *top rank trees do*
    delete $F[i]$
    Assign ranks to newly created iTrees
    $F[i] \leftarrow iTree(S,0,h)$
end for
goto 10 for upcoming Streaming data points

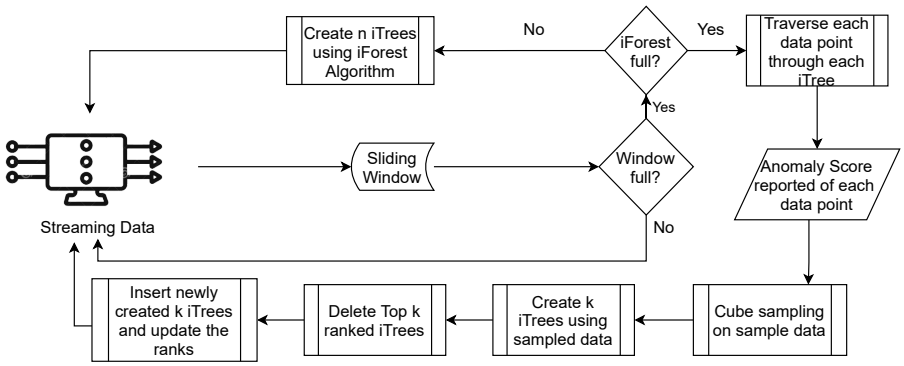

Fig. 1: Framework of the proposed approach for Streaming Data

deleted first. The anomaly detection in streaming data is a process to identify points that are out of the typical or abnormal. The figure 1 illustrates the basic framework for detecting anomalies in streams. The streaming data is initially sent through a sliding window. The complete process of anomaly detection in streaming data is explained in detail in Algorithm 1. A sliding window is a helpful technique for doing calculations on streaming data in such a way that only blocks of data containing ω items of the stream are evaluated at a time.

4 Experimental Results

The proposed methodology to detect anomaliesis evaluated in the following manner: 1) The proposed approach is compared with other state-of-art anomaly detection approaches in terms of *AUC-ROC*; 2) Finally, the proposed approach is evaluated on a variety of anomaly patterns artificially introduced into the data stream.

In our experiments, we utilize the formula in Equation 1 (from [13]) to compute the value of *AUC-ROC*.

$$AUC\text{-}ROC = \left(\Sigma r_i - \left(n_\alpha^2 + n_a\right)/2\right) / \left(n_\alpha \times n_n\right) \tag{1}$$

Where n_a represents the number of actual anomalies, n_n denotes the number of actual normal points, and r_i denotes the rank of the i^{th} anomaly in the descending ranked list of anomalies.

Table 1: Dataset information

Dataset	no. of records	no. of attributes	Anomaly threshold
Mulcross	262144	4	10.00%
Forest Cover	286048	10	0.90%
Breastw	683	9	35.00%
Http(KDDcup99)	567497	3	0.39%
Satellite	6435	36	32.00%
Shuttle	49097	9	7.15%

Five real-world datasets and one artificial dataset were used in the experiments [6]. Table 1 contains a brief summary of the datasets utilized. The table's anomaly threshold column indicates the proportion of abnormalities in the dataset.

The tests were conducted on these datasets because they contain pre-defined anomalies that serve as class labels, making them suitable for assessment. This is precisely why these datasets are often regarded as industry standards for anomaly detection.

The experiments were conducted on a personal computer equipped with an Intel(R) Core(TM) *i7-7500U* processor running at @2.70GHz, 2.90GHz, and

16.0GB of memory (RAM). Windows 10 Pro was used as the operating system. Python 3.6 was used to program the techniques mentioned in Section 3.

4.1 Comparison of the Proposed Approach with state-of-art approaches

The proposed approach is compared with a few state-of-art anomaly detection approaches for streaming data: 1) iForest [13], 2) RRCF [10], 3) HSTa [18], and 4) PiForest [12]. Table 2 shows the comparison of the proposed approach with other anomaly detection techniques in terms of space complexities. The comparison in terms of AUC is shown in Table 3. The proposed approach works well on streaming data and the results shows that the proposed approach out-performs state-of-art algorithms for streaming data.

Table 2: Comparison of space complexity of state-of-art algorithms

	Proposed Approach	PiForest	iForest	RRCF	HSTa
Space complexity	O(Ψtb)	O(Ψtb)	O(Ψtb)	O(Ψtb)	O(t2^h)
Parameters	Ψ-Sub-sampling size t-No. of trees b-Size of a node	Ψ-Sub-sampling size t-No. of trees b-Size of a node	Ψ-Sub-sampling size t-No. of trees b-Size of a node	Ψ- Sample size t- No. of trees b- Size of a node	t- No. of trees h- Depth of tree
Parameter values(Default in the experiment)	Ψ=256 t=50 b depends on data	Ψ=511 t=10 b=4.125 Bytes	Ψ=511 t=100 b depends on data	Ψ=1000 t=200 b depends on data	t=25 h=15

Table 3: Comparison of proposed approach with state-of-art algorithms in terms of AUC

Dataset	Proposed Approach	iForest	PiForest	HSTa	RRCF
Http(KDDcup99)	0.99	0.99	0.99	0.96	0.97
Satellite	0.72	0.70	0.67	0.66	0.64
Mulcross	0.98	0.97	0.87	0.86	0.87
Forest Cover	0.93	0.91	0.70	0.74	0.65
Breastw	0.96	0.95	0.88	0.81	0.89
Shuttle	0.99	0.99	0.99	0.98	0.99

4.2 The proposed approach's performance on a variety of anomaly types

We evaluate the proposed approach's effectiveness against a variety of patterns of anomalies [2] that occur often in streams. We produced a synthetic dataset by injecting anomalies into a sine wave. There are three types of anomalies: point anomaly, contextual anomaly , and collective anomaly . These anomalies are injected as shown in Figure 2. We manage the streaming data using a method

termed shingling, similar to Guha *et al.* [10]. A shingle size of n placed across a stream collects the stream's initial n data points from time $T = 0, T = 1, ..., T = N$ to generate a N-dimensional data point. Following that, at time $T = (N+1)$, the data from $T = 2, ..., t = (N + 1)$ are recorded to construct a second N-dimensional data point. A shingle gives the curve a characteristic form, and any variation from that shape signals an abnormality. In Figure 2, the curve below

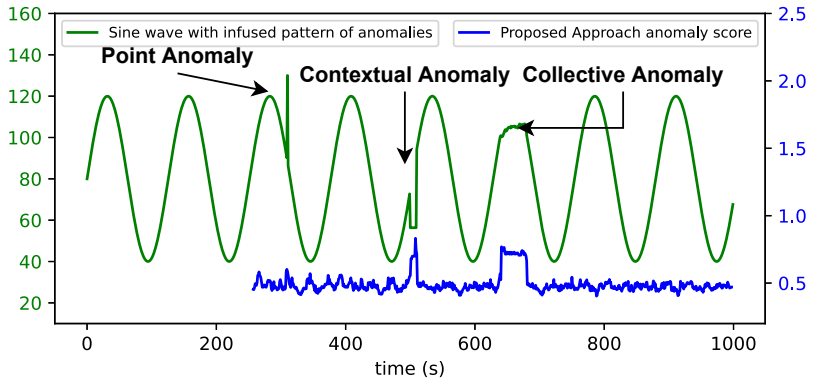

Fig. 2: Anomaly Score with generated data

the sine wave shows the anomaly score obtained using the proposed method. The first 256 samples are utilized to train the anomaly detector, and the succeeding data stream is used to compute the anomaly scores. The following classification accuracy metrics are collected for specific anomalies: Contextual Anomaly (0.99); Point Anomaly (1.00); and Collective Anomaly (0.944). Moreover, a *AUC-ROC* score of 0.981 is obtained, indicating that the suggested method is effective with a variety of anomaly patterns.

5 Conclusion

In this paper, we present a method for detecting anomalies utilizing the *iForest* algorithm and cube sampling. Additionally, the method permits the processing of streaming data, which is data that flows constantly and may be regarded as limitless in all practical senses. We handle such data well by utilizing a sliding window. The approach's efficacy in terms of anomaly identification is demonstrated by comparison to the operation of various well-known anomaly detection algorithms. In each example, despite dealing with limited data, the strategy produces findings that are equal to or superior to those obtained using existing anomaly detection approaches. Additionally, we show the proposed approach works well in a simulated environment.

References

1. Breiman, L.: Random forests. Machine learning **45**(1), 5–32 (2001)
2. Chandola, V., Banerjee, A., Kumar, V.: Anomaly detection: A survey. ACM computing surveys (CSUR) **41**(3), 1–58 (2009)
3. Chandola, V., Mithal, V., Kumar, V.: Comparative evaluation of anomaly detection techniques for sequence data. In: 2008 Eighth IEEE international conference on data mining. pp. 743–748. IEEE (2008)
4. Cochran, W.G.: Sampling techniques. John Wiley & Sons (2007)
5. Datar, M., Gionis, A., Indyk, P., Motwani, R.: Maintaining stream statistics over sliding windows. SIAM journal on computing **31**(6), 1794–1813 (2002)
6. Frank, A.: Uci machine learning repository. http://archive. ics. uci. edu/ml (2010)
7. Gaber, M.M., Zaslavsky, A., Krishnaswamy, S.: Mining data streams: a review. ACM Sigmod Record **34**(2), 18–26 (2005)
8. Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M., Bouchachia, A.: A survey on concept drift adaptation. ACM computing surveys (CSUR) **46**(4), 1–37 (2014)
9. Gibbons, P.B., Tirthapura, S.: Distributed streams algorithms for sliding windows. In: Proceedings of the fourteenth annual ACM symposium on Parallel algorithms and architectures. pp. 63–72. ACM (2002)
10. Guha, S., Mishra, N., Roy, G., Schrijvers, O.: Robust random cut forest based anomaly detection on streams. In: International conference on machine learning. pp. 2712–2721 (2016)
11. Hawkins, D.M.: Identification of outliers, vol. 11. Springer (1980)
12. Jain, P., Jain, S., Zaïane, O.R., Srivastava, A.: Anomaly detection in resource constrained environments with streaming data. IEEE Transactions on Emerging Topics in Computational Intelligence (2021)
13. Liu, F.T., Ting, K.M., Zhou, Z.H.: Isolation forest. In: 2008 Eighth IEEE International Conference on Data Mining. pp. 413–422. IEEE (2008)
14. Liu, Z., Liu, X., Ma, J., Gao, H.: An optimized computational framework for isolation forest. Mathematical Problems in Engineering **2018**, 1–13 (2018)
15. Nigam, A., Kumar, P., Gupta, V.: Some methods of inclusion probability proportional to size sampling. Journal of the Royal Statistical Society: Series B (Methodological) **46**(3), 564–571 (1984)
16. Shastri, A.A., Ahuja, K., Ratnaparkhe, M.B., Busnel, Y.: Probabilistically sampled and spectrally clustered plant genotypes using phenotypic characteristics. arXiv preprint arXiv:2009.09028 (2020)
17. Taherdoost, H.: Sampling methods in research methodology; how to choose a sampling technique for research. How to Choose a Sampling Technique for Research (April 10, 2016) (2016)
18. Tan, S.C., Ting, K.M., Liu, T.F.: Fast anomaly detection for streaming data. In: Twenty-Second International Joint Conference on Artificial Intelligence. pp. 1511–1516 (2011)
19. Vehovar, V., Toepoel, V., Steinmetz, S.: Non-probability sampling. The Sage handbook of survey methods pp. 329–345 (2016)
20. Xu, D., Wang, Y., Meng, Y., Zhang, Z.: An improved data anomaly detection method based on isolation forest. In: 2017 10th International Symposium on Computational Intelligence and Design (ISCID). vol. 2, pp. 287–291. IEEE (2017)
21. Zhou, J., Fu, Y., Wu, Y., Xia, H., Fang, Y., Lu, H.: Anomaly detection over concept drifting data streams. Journal of Computational Information Systems **5**(6) (2009)