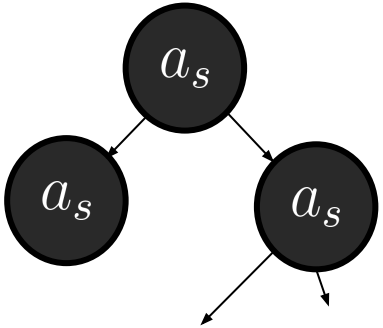


StreamRHF in CapyMOA

Giovanni BENEDETTI DA ROSA
Cristian Alejandro CHÁVEZ BECERRA

Master 2 Data Science
Institut Polytechnique de Paris-École Polytechnique
Data Streaming

Introduction

Introduction

- **Data streaming** applications: fraud detection, network monitoring, and real-time recommendation system

- **Anomaly Detections** algorithms identifying interesting or rare events that deviate from normality

Image extracted from [1]

Problematic

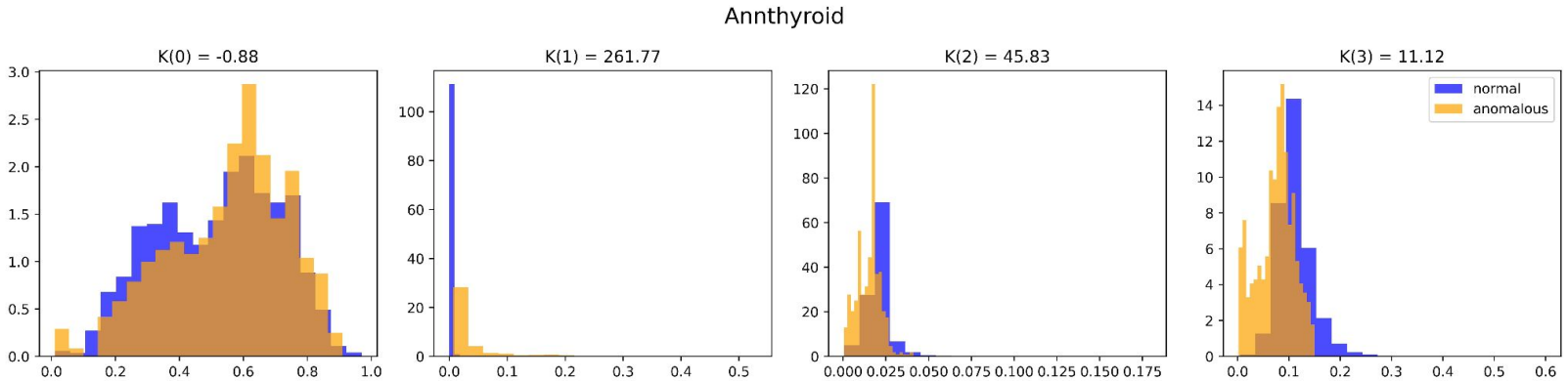
- Data streaming algorithms: memory requirements and faster processing times.
- Anomaly Detections algorithms generally not adapt to the requirements.
- **Implement StreamRHF[2] in the CapyMOA[1] library.**

Image extracted from [1]

Methods

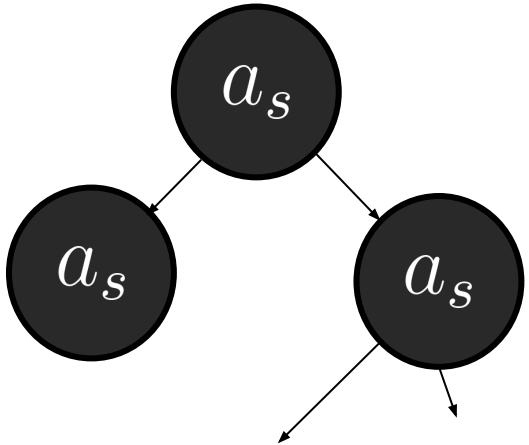
Methods: RHF

Moors et al[4] described kurtosis as a measure of dispersion and explained that high kurtosis values result from either: (i) occasional values far from the mean in a distribution with most of its probability mass concentrated around the mean, or (ii) significant probability mass located in the distribution's tails.



Methods: RHF

- Unsupervised Anomaly Detection[3] method
- Data is split across different bins by means: each one of the t trees with height h is built by recursively splitting the whole dataset
- Each splitting decision is done selecting an attribute a to split according to its kurtosis score and a split value randomly



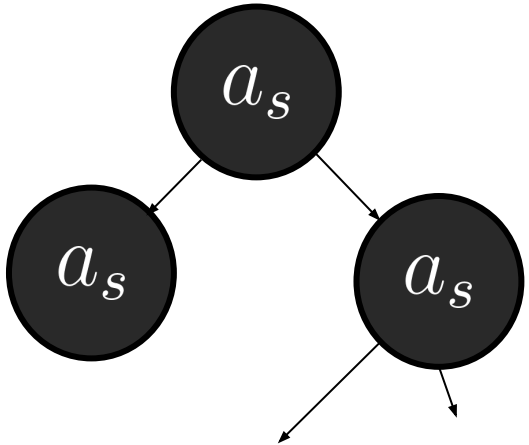
Methods: RHF

- As defined by Putina et al. Given a dataset $X \in \mathbb{R}^{n \times d}$ we can define kurtosis split as:

1) $K_s = \sum_{a=0}^{d} \log\left(K(X_a) + 1\right)$

2) $r \sim \text{Uniform}([0, K_s))$

3) $a_s = \arg\min\left(k \in [0, d] \,\middle|\, \sum_{a=0}^{k} \log\left(K(X_a) + 1\right) > r\right)$



Methods: RHF

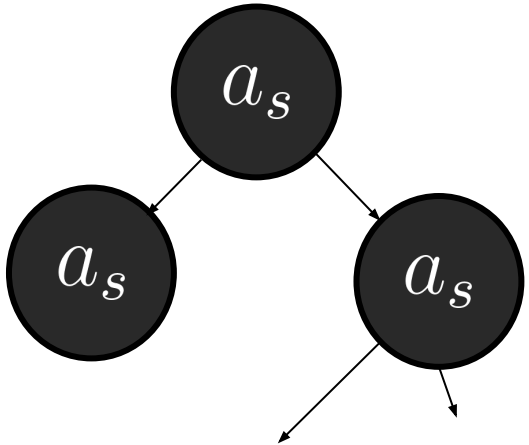
- **Anomaly Score**: Given that S_l represents the number of instances in each leaf, each instance i is stored in a tree T, it's possible to write the information content:

$$w_T(i) = \log\left(\frac{1}{P_{l_i}}\right)$$

  where P_l is the probability associated with leaf l.

The **anomaly score** is defined as:

$$w_i = \sum_T w_T(I).$$



Methods: StreamRHF

In streaming of datas, distribution of data is not time invariant, leading to concept drifts.

Nesic et. al[2] define **StreamRHF** uses a window(n) approach:

1) Wait for n initial points of the data, than build a tree using RHF and compute the score.
2) At each time a new instance arrive, check if the splitting attribute of each node will change; if it's the case: rebuild the subtree; Compute new score.
3) When new n data points arrive rebuild RHF, using these data.

Methods: StreamRHF

- **Time and Space Complexity**: Let w be the size of the sliding window, d the number of features, h the maximum height of every tree, and t be the total number of trees in our forest[2].

- For the space complexity: $O(wdt)$

- We rebuild only relatively small subtrees $\bar{h} < h$when doing an insertion with height , which gives a running time of for an update

$$O(wdt\bar{h})$$

Experimental Evaluation

Parameters settings

- To compare our implementation, we used the default values of he built-in implemented methods in CapyMOA, using the following parameters:
  1. **HalfSpaceTrees**[5]: window size = 100, number of trees = 25, max depth = 15, anomaly threshold= 0.5.
  2. **Online Isolation Forest**[6]: num trees = 32, max leaf samples = 32, growth criterion = ’adaptive’, subsample = 1.0, window size = 2048, branching factor = 2, split = ’axisparallel’.
  3. **Autoencoder** [7] :hidden layer = 2, learning rate = 0.5, threshold = 0.6.
  4. **StreamRHF**: number_of_trees = 100, max-height = 5, window_size: 1% of the dataset.

Metrics

- Imbalanced classes - intrinsic characteristic of anomaly detections.
- Streaming data the proportion of these anomalies is not constant. While is well known that AUCPR takes to account these for fix outlier proportion than in ROC AUC. Some others work shows the opposite for anomaly detections[8]. We then choose:

- **Average Precision** $AP = \sum_{N}^{n} (R_n - R_{n-1}) P_n,$
- **AUC** $AUC = \sum_{i=1} (FPR_i - FPR_{i-1}) \cdot TPR_i,$

Implementation in CapyMOA

- **Three classes implemented:** Node, RandomHistogramForest, StreamRHF

- **Normalization of the scores:** We defined a maximum and minimum value for the scores calculated by the original paper of StreamRHF. Initially, the score is given by the following equation:

$$w_T(i) = \log\left(\frac{1}{P_{l_i}}\right) = \log\left(\frac{n}{|S(l)|}\right)$$

Implementation in CapyMOA: Normalization of the scores

$$\text{min\_score} = \log\left(\frac{n}{n}\right) = \log(1) = 0$$

$$\text{max\_score} = \log\left(\frac{n}{1}\right) = \log(n)$$

$$\text{normalized\_score} = \frac{w_T(i) - \text{min\_score}}{\text{max\_score} - \text{min\_score}}$$

Results: Comparison using the AP score

Dataset	StreamRHF	Autoencoder	HalfSpaceTrees	OnlineIsolationForest
abalone	0.237 ± 0.024	0.116	**0.531**	0.033
annthyroid	0.396 ± 0.019	0.086	**0.577**	0.079
kdd_ftp	0.280 ± 0.008	0.157	**0.439**	0.196
magicgamma	**0.623 ± 0.008**	0.352	0.272	0.287
mammography	**0.219 ± 0.021**	0.157	0.184	0.049
mnist	**0.303 ± 0.006**	0.092	0.117	0.085
musk	**0.771 ± 0.061**	0.032	0.021	0.495
satellite	**0.593 ± 0.012**	0.015	0.317	0.518
satimages	**0.856 ± 0.020**	0.012	0.256	0.012
shuttle_odds	**0.858 ± 0.044**	0.072	0.548	0.040
spambase	0.478 ± 0.038	0.507	0.282	**0.568**
thyroid	**0.544 ± 0.031**	0.103	0.559	0.014
Mean	**0.513 ± 0.024**	0.142	0.342	0.198
Median	**0.511**	0.098	0.300	0.082

Results: Comparison using the AP score

1. We achieved very similar results compared to the ones shown in the original paper in the datasets we used using this score (the score used in the original paper)

$$AP = \sum_{n} (R_n - R_{n-1}) P_n,$$

2. StreamRHF generally outperforms the default configurations of the built-in CapyMOA models.

3. In the case of very small datasets, such as the abalone dataset, HalfSpaceTrees surpasses StreamRHF.

Results: Comparison using the AUC score

Dataset	StreamRHF	Autoencoder	HalfSpaceTrees	OnlineIsolationForest
abalone	0.862 ± 0.005	0.855	**0.963**	0.318
annthyroid	0.865 ± 0.006	0.534	**0.974**	0.516
kdd_ftp	0.423 ± 0.022	0.124	**0.662**	0.350
magicgamma	**0.701 ± 0.010**	0.500	0.382	0.396
mammography	0.858 ± 0.004	0.766	**0.912**	0.749
mnist	**0.809 ± 0.005**	0.500	0.558	0.499
musk	**0.980 ± 0.007**	0.500	0.026	0.612
satellite	0.939 ± 0.009	0.500	**0.975**	0.541
satimages	**0.989 ± 0.003**	0.500	0.965	0.494
shuttle_odds	**0.994 ± 0.003**	0.500	0.899	0.101
spambase	0.636 ± 0.034	**0.678**	0.271	0.496
thyroid	0.977 ± 0.001	0.823	**0.979**	0.168
Mean	**0.839 ± 0.011**	0.556	0.725	0.423
Median	**0.862**	0.500	0.662	0.396

Results: Comparison using the AUC score

1. We decided to use the AUC metric since is a standard metric for Anomaly Detection models that was not used in the original paper.

$$AUC = \sum_{i=1}^{N} (FPR_i - FPR_{i-1}) \cdot TPR_i,$$

2. StreamRHF doesn't always outperform other CapyMOA models in terms of AUC, but it delivers competitive results, often close to the best model, HalfSpaceTrees

3. Its average and median AUC values are still higher, showing its overall reliability.

Results: Insertion time

Dataset	StreamRHF	Autoencoder	HalfSpaceTrees	OnlineIsolationForest
abalone	117.338 ± 1.076	0.310 ± 0.015	**0.098 ± 0.026**	5.642 ± 0.044
annthyroid	122.116 ± 1.671	0.303 ± 0.001	**0.032 ± 0.005**	10.633 ± 0.023
kdd_ftp	112.115 ± 1.050	0.298 ± 0.000	**0.032 ± 0.001**	10.603 ± 0.020
magicgamma	202.003 ± 7.166	0.305 ± 0.001	**0.033 ± 0.010**	13.816 ± 0.032
mammography	134.445 ± 2.414	0.304 ± 0.003	**0.031 ± 0.003**	13.813 ± 0.018
mnist	398.449 ± 5.247	0.370 ± 0.001	**0.032 ± 0.000**	13.300 ± 0.023
musk	547.362 ± 12.781	0.422 ± 0.002	**0.045 ± 0.024**	8.237 ± 0.040
satellite	206.419 ± 4.093	0.324 ± 0.000	**0.032 ± 0.000**	9.457 ± 0.026
satimages	205.343 ± 6.415	0.323 ± 0.001	**0.032 ± 0.000**	11.490 ± 0.033
shuttle_odds	189.577 ± 6.058	0.304 ± 0.002	**0.029 ± 0.000**	12.556 ± 0.275
spambase	177.934 ± 2.605	0.338 ± 0.001	**0.032 ± 0.000**	7.627 ± 0.031
thyroid	112.401 ± 0.947	0.303 ± 0.001	**0.037 ± 0.007**	8.647 ± 0.021

Table 3: Insertion times (in ms) for different datasets and models, with the smallest values highlighted in bold.

Results: Insertion time

1. Taking into account the importance of time in a data streaming framework we decided to compare the insertion time of the models.

2. StreamRHF takes significantly longer than other models, which can be attributed to several factors: default number of trees, maximum height of the trees and language implementation (cython in the original paper)

3. We may reduce the insertion time by reducing the number of trees and/or maximum height of the trees but it is a trade-off in performance and speed

Results: CapyMOA check

Dataset	Original Paper Procedure (AP)	CapyMOA Procedure (AP)
abalone	0.233 ± 0.035	0.123 ± 0.020
annthyroid	0.401 ± 0.035	0.354 ± 0.028
kdd_ftp	0.280 ± 0.011	0.169 ± 0.036
thyroid	0.548 ± 0.044	0.444 ± 0.017
Mean	0.365 ± 0.031	0.272 ± 0.012
Median	0.341	0.259

Results: CapyMOA check

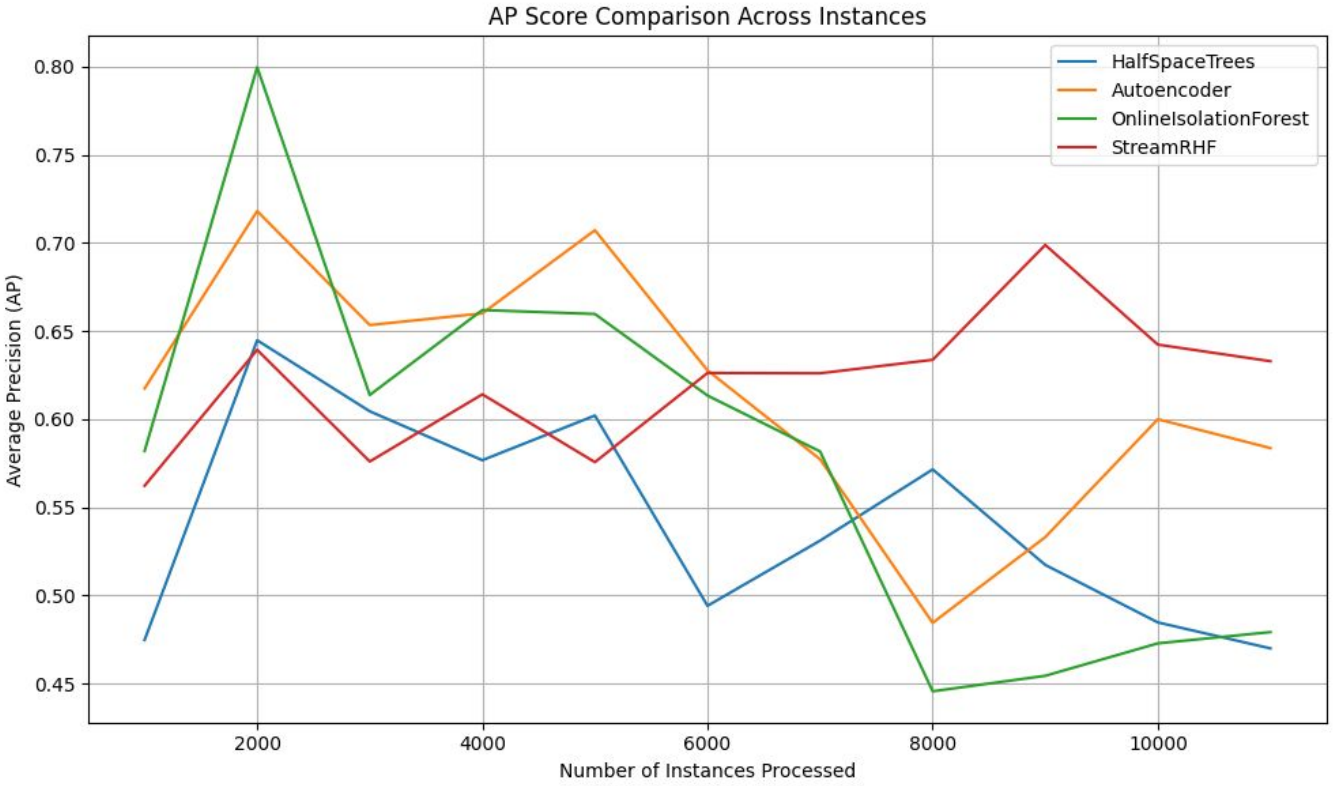
1. In the original paper, the procedure to evaluate the model the authors did for every instance was to first insert the instance and then score the instance. The other way around according to CapyMOA tutorials.

$$w_T(i) = \log\left(\frac{1}{P_{l_i}}\right) = \log\left(\frac{n}{|S(l)|}\right)$$

2. This procedure reduces the anomaly score for the all the instances but in different proportions.

3. As can be seen in table, the original paper procedure leads to better results for the AP score.

Demo


AP Score Comparison Across Instances
HalfSpaceTrees
Autoencoder
OnlineIsolationForest
StreamRHF
Average Precision (AP)
0.80
0.75
0.70
0.65
0.60
0.55
0.50
0.45
2000
4000
6000
8000
10000
Number of Instances Processed

Conclusion

Conclusion

- StreamRHF is a novel method that demonstrates promising results and competes with state-of-the-art data streaming models.
- We were able to reproduce results similar to those in the original paper.
- We successfully integrated our implementation with CapyMOA.
- Additionally, we established maximum and minimum scores to facilitate its integration into CapyMOA.
- Our implementation in CapyMOA provides results comparable to the original implementation in the .py file.
- However, the main drawback of our implementation is the computational time, which is impacted by the high number of trees and the use of a less efficient programming language.

Code Availability

Our GitHub repository for the project can be found at:

https://github.com/AlejandroUN/Stream-Random-Histogram-Forest

The from-scratch implementation Python file of the algorithm StreamRHF can be found in the following link in our GitHub repository:

https://github.com/AlejandroUN/Stream-Random-Histogram-Forest/blob/main/src/capymoa/anomaly/_stream_rhf.py

The demo showing the implementation properly integrated with CapyMOA using CapyMOA models and plots to compare can be found in the following notebook:

https://github.com/AlejandroUN/Stream-Random-Histogram-Forest/blob/main/src/capymoa/anomaly/streamrhf/notebook_presentation.ipynb

References

1. Gomes, H. M., Bifet, A. (2024). Practical Machine Learning for Streaming Data. ACM Digital Library, 6418-6419. Available: https://doi.org/10.1145/3637528.3671442
2. Stefan Nesic, Andrian Putina, Maroua Bahri, Alexis Huet, Jose Manuel, et al.. STREAMRHF: Tree-Based Unsupervised Anomaly Detection for Data Streams. AICCSA 2022 - 19th ACS/IEEE International Conference on Computer Systems and Applications, Dec 2022, Abu Dhabi, United Arab Emirates. ffhal-03948938f
3. A. Putina, M. Sozio, D. Rossi, and J. M. Navarro, “Random histogram forest for unsupervised anomalydetection,” in International Conference on Data Mining (ICDM). IEEE, 2020, pp. 1226–1231.
4. J. J. A. Moors, “The meaning of kurtosis: Darlington reexamined”, The American Statistician, vol. 40, no. 4, pp. 283–284, 1986.
5. Swee Chuan Tan, Kai Ming Ting, and Tony Fei Liu. Fast anomaly detection for streaming data. In International Joint Conference on Artificial Intelligence (IJCAI), pages 1469–1495, 2017.
6. Filippo Leveni, Guilherme Weigert Cassales, Bernhard Pfahringer, Albert Bifet, and Giacomo Boracchi. OnlineIsolation Forest. In International Conference on Machine Learning (ICML), Proceedings of Machine Learning Research (PMLR), 2024.
7. Richard Hugh Moulton, Herna L. Viktor, Nathalie Japkowicz, and Jo˜ao Gama. Contextual one-class classifi- cation in data streams. arXiv preprint arXiv:1907.04233, 2019.
8. Minjae Ok, Simon Kl¨uttermann, and Emmanuel M¨uller, ”Exploring the Impact of Outlier Variability on Anomaly Detection Evaluation Metrics,” arXiv preprint, arXiv:2409.15986v1 [cs.LG], 24 Sep 2024, License: CC BY 4.0.
9. Bifet, Albert, et al. Machine Learning for Data Streams: with Practical Examples in MOA. The MIT Press, 2018. DOI: https://doi.org/10.7551/mitpress/10654.001.0001.