Data Stream Processing

Sophie Normand
Alexandra Amani
Eva Clergue

M2 Data Science
January 2023

Implementation of xStream algorithm for anomaly detection in River 🌊

Outline

- PySAD – Streaming anomaly detection framework in Python
- xStream algorithm
- Implementation of xStream in River
- Model performance
- Conclusion – Next steps

Our strategy :

1. Familiarize with the library and models → ***Testing with PySAD***
2. Transfer in "River mode" : re-write the model as new .py file that could be integrated to River → ***Testing with a "local" River***
3. Adapt the code so that it fits the River guidelines (dictionaries, descriptions,…)
4. Testing on different datasets

PySAD – Streaming anomaly detection framework in Python

- Stream simulators, evaluators, preprocessors, statistic trackers, postprocessors, probability calibrators...
- Integrations for batch anomaly detectors of PyOD (Python Library for outlier detection)

Models : **xStream**, **LODA** (Lightweight on-line detector of anomalies), **RS-Hash** (Randomized Subspace Hashing algorithm)

Performances obtained with PySAD : **ROC-AUC metric**, area under the Receiver Operating Characteristic Curve

DATA		xStream		LODA		RS-Hash	
Name	N, % anomaly	ROC-AUC (%)	Time (s)	ROC-AUC (%)	Time (s)	ROC-AUC (%)	Time (s)
Arrhythmia	452, 14.6%	71,3	146	49,9	12	73,6	5
Wine	129, 7.8%	50,3	26	70	5	82,9	2
Breast	683, 34,9%	94,3	135	49,4	19	97,1	9
Optdigits	5216, 2,8%	66,3	1296	49,9	137	51,3	61

Mini-batch learning ? LODA and RS-Hash remarks

LODA

Collection of k histograms, approximating the probability density of input data projected onto a single projection vector

Score : return probabilities from histograms (average)

Algorithm 1: Loda's training (update) routine.

input: data samples $\{x_i \in \mathbb{R}^d\}_{i=1}^n$;
output: histograms $\{h_1, \ldots, h_n\}$, projection vectors $\{w_i\}_{i=1}^k$. ;
initialize projection vectors with $\left\lceil d^{-\frac{1}{2}} \right\rceil$ non zero elements $\{w_i\}_{i=1}^k$;
initialize histograms $\{h_i\}_{i=1}^k$;
for $j \leftarrow 1$ **to** n **do**
  for $i \leftarrow 1$ **to** k **do**
    $z_i = x_j^{\mathrm{T}} w_i$;
    update histogram h_i by z_i;
  end
end
return $\{h_i\}_{i=1}^k$ and $\{w_i\}_{i=1}^k$.

Score one sample
Training on one sample

Pevný, T. (2016)

RS-Hash

Method based on Sub-Hashing of the space : select some dimensions, and apply hash functions on transformed samples label

Outlier score : based on search the minimum of hash functions values

Remarks :
Learning: « Randomly sample the dataset D for a training sample S of s points. » and "s = min(1000, n)"
Normalization min-max for the training sample
→ *Online learning with one sample ?*

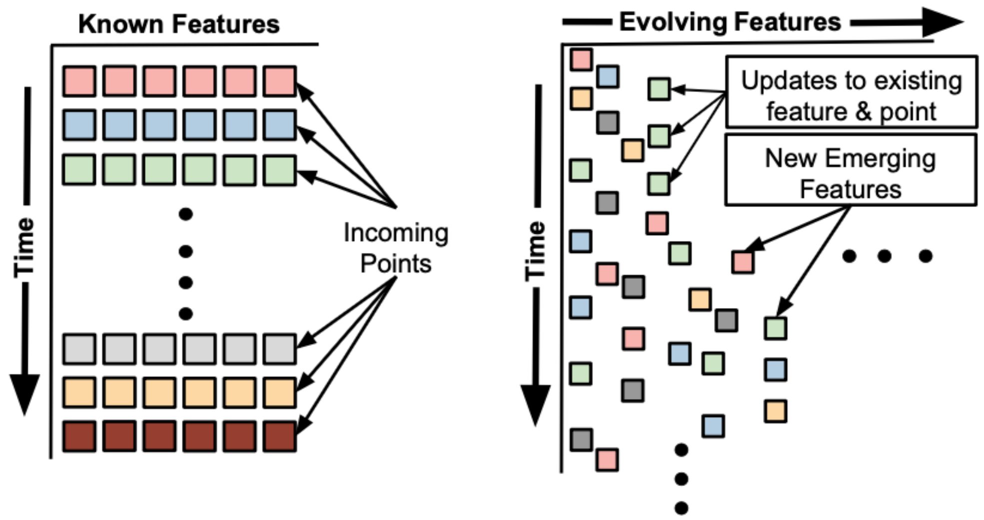
Scoring : use min-max values for normalization *(NB: small score = anomaly, how to reverse ?)*

Score one sample
Training on one sample

Sathe, S. and Aggarwal, C.C. (2016)

xStream model

- Outlier detection problem for feature-evolving streams:
  - Data points may evolve, with feature values changing
  - Feature space may evolve with newly-emerging features over time
- As competitive as state-of-the-art detectors and particularly effective in high-dimensions with noise



(a) row-stream (b) *feature-evolving* stream

Properties:

- Constant memory: processing each stream element in constant time
- Tackles high-dimensionality
- Measures outlierness at multiple scales
- Handles non-stationarity

xStream algorithm

xStream is an ensemble of Half-Space Chains that approximates density efficiently, without needing to know the underlying feature space a-priori.

Each chain approximates the density of a point by counting its nearby neighbors at multiple scales.

3 key steps:

- **StreamHash:** subspace-selection and dimensionality reduction via sparse random projections for evolving feature spaces
- **Half-Space Chains:** an efficient ensemble to estimate density at multiple scales
- extensions to handle non-stationarity and evolving data points in the stream

Table 1: Comparing xSTREAM with state-of-the-art outlier detection techniques in terms of various properties.

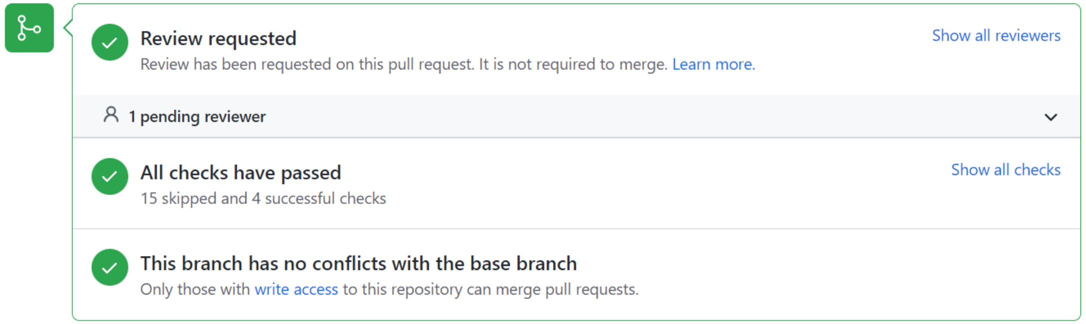
Methods/ Properties	LOF [10]	Feat.Bag. [22]	LOCI [25]	HiCS [21]	iForest [23]	HS-Stream [31]	STORM [6]	LODA [26]	RS-Hash [29]	RS-Forest [32]	xSTREAM
Static	✔	✔	✔	✔	✔	✔		✔	✔	✔	✔
Streaming						✔	✔	✔	✔	✔	✔
Multi-scale			✔								✔
Subspaces		✔		✔	✔	✔		✔	✔	✔	✔
Projections								✔			✔
Evolving feature space											✔
Evolving points/ feature values											✔

Manzoor et al. (2018)

xStream in River

- Objective : create a class implementing the xStream model compatible with River API
  - Anomaly detection module : adaptation to the « base » class for this type of model
  - *learn_one* and *predict_one* methods : adapt the *fit_partial* and *score_partial* from PySAD (and all particularities of this algorithm…)
    - Local tests modifying steps by steps : the algorithm needs to work with our type of iteration on data, with numpy arrays
    - Once it is running, re-writing of the code using only dictionaries, code not yet optimized and clean
    - Adaptation to **River** : keep dictionaries, create functions, "clean" code
    - Follow the **final guidelines River** : comments, code style, indentations

- Pull request ? → no conflicts with the base branch + 4 successful checks



DEMO :

Run xStream adapted for River

Model performance – xStream in River 🌊

DATA		xStream	
Name	N, % anomaly	ROC-AUC (%)	Time (s)
Arrhythmia	452, 14.6%	70,9	530
Wine	129, 7.8%	73,6	13
Breast	683, 34,9%	77,9	65
Optdigits	5216, 2,8%	73,8	877

- Still nice results
- ROC-AUC metric different from PySAD : fact/way that we controlled randomness ?, shuffling for iterations on samples ?, order *score_one/learn_one* ?
- Time of execution : clearly slower than PySAD : use of dictionaries

Conclusion – Next steps

- Approach for anomaly detection : xStream is adaptable to River, but how « batchy » is it ?
  - Real adaptation in the construction of the algorithm for learning on ONE sample ?
- Benchmarking : xStream seems to give good results on the datasets we used
  - Real applications ? Medicine, cybersecurity, predictive maintenance, supply chain management…
  - Challenge the model in real situations
- LODA and RS-Hash ? Could we re-adapt this models, modifying the construction / type of implementation ?
- Comments in our pull request, benchmark and challenge the method

References

Emaad Manzoor, Hemank Lamba, and Leman Akoglu. 2018. “XStream: Outlier Detection in Feature-Evolving Data Streams”. *In* Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD '18). Association for Computing Machinery, New York, NY, USA, 1963–1972.

Pevný, T. 2016. “Loda: Lightweight on-line detector of anomalies.” *Mach Learn 102*, 275–304

Sathe, S. and Aggarwal, C.C. (2016). "Subspace Outlier Detection in Linear Time with Randomized Hashing," 2016 IEEE 16th International Conference on Data Mining (ICDM), 459-468

Yilmaz, Selim F & Kozat, Suleyman, S. 2020. “PySAD : A Streaming Anomaly Detection Framework in Python”, arXiv preprint arXiv:2009.02572