Journal of Machine Learning Research 22 (2021) 1-8 Submitted 12/20; Revised 4/21; Published 4/21

River: machine learning for streaming data in Python

Jacob Montiel* jacob.montiel@waikato.ac.nz
AI Institute, University of Waikato, Hamilton, New Zealand
LTCI, Télécom Paris, Institut Polytechnique de Paris, Palaiseau, France

Max Halford* max.halford@alan.eu
Alan, Paris, France

Saulo Martiello Mastelini mastelini@usp.br
Institute of Mathematics and Computer Sciences, University of São Paulo, São Carlos, Brazil

Geoffrey Bolmier geoffrey.bolmier@volvocars.com
Volvo Car Corporation, Göteborg, Sweden

Raphael Sourty raphael.sourty@irit.fr
IRIT, Université Paul Sabatier, Toulouse, France
Renault, Paris, France

Robin Vaysse robin.vaysse@irit.fr
IRIT, Université Paul Sabatier, Toulouse, France
Octogone Lordat, Université Jean-Jaures, Toulouse, France

Adil Zouitine adil.zouitine@irt-saintexupery.com
IRT Saint Exupéry, Toulouse, France

Heitor Murilo Gomes heitor.gomes@waikato.ac.nz
AI Institute, University of Waikato, Hamilton, New Zealand

Jesse Read jesse.read@polytechnique.edu
LIX, École Polytechnique, Institut Polytechnique de Paris, Palaiseau, France

Talel Abdessalem talel.abdessalem@telecom-paris.fr
LTCI, Télécom Paris, Institut Polytechnique de Paris, Palaiseau, France

Albert Bifet abifet@waikato.ac.nz
AI Institute, University of Waikato, Hamilton, New Zealand
LTCI, Télécom Paris, Institut Polytechnique de Paris, Palaiseau, France

Editor: Andreas Mueller

Abstract

River is a machine learning library for dynamic data streams and continual learning. It provides multiple state-of-the-art learning methods, data generators/transformers, performance metrics and evaluators for different stream learning problems. It is the result from the merger of two popular packages for stream learning in Python: Creme and scikit-multiflow. River introduces a revamped architecture based on the lessons learnt from the seminal packages. River's ambition is to be the go-to library for doing machine learning on streaming data. Additionally, this open source package brings under the same um-

*. Co-first authors.

brella a large community of practitioners and researchers. The source code is available at https://github.com/online-ml/river.

Keywords: stream learning, online learning, data stream, concept drift, supervised learning, unsupervised learning, Python.

1. Introduction

In machine learning, the conventional approach is to process data in batches or chunks. Batch learning models assume that all the data is available at once. When a new batch of data is available, these models have to be retrained from scratch. The assumption of data availability is a hard constraint for the application of machine learning in multiple real-world applications where data is continuously generated. Additionally, keeping historical data requires dedicated storage and processing resources, which in some cases might be impractical, e.g. storing the network logs from a data center. A different approach is to treat data as a stream, in other words, as an infinite sequence of items; data is not stored and models continuously learn one data sample at a time (Bifet et al., 2018).

Creme (Halford et al., 2019) and scikit-multiflow (Montiel et al., 2018) are two open-source libraries to perform machine learning in the stream setting. River is the merger of these projects, combining their strengths while leveraging the lessons learnt during their development. More than a simple merge of code, River includes a revamped architecture and expands functionality, e.g. support for mini-batches, processing time improvements, more metrics for classification, regression and clustering, more clustering methods, etc. River supersedes its parent packages and unifies continuous development under a single project. River is mainly written in Python, with some core elements written in Cython (Behnel et al., 2011) for performance. Supported applications are generally as diverse as those found in traditional batch settings, including: classification, regression, clustering, representation learning, multi-label and multi-output learning, forecasting, and anomaly detection.

2. Architecture

River's architecture is the result from the lessons learned during the development of its parent packages Creme and scikit-multiflow. Machine learning models in River are extended classes of specialized `mixins` that mirror the different type of learning tasks, e.g. classification, regression, clustering, etc. This ensures compatibility across the library and eases the extension/modification of existing models, as well as the creation of new models compatible with the rest of the API.

All predictive models perform two core functions: learn (also referred to as training or fitting) and predict. Learning takes place via the `learn_one` method (updates the internal state of the model). Depending on the learning task, models provide predictions via the `predict_one` (classification, regression, and clustering), `predict_proba_one` (classification), and `score_one` (anomaly detection) methods. Note that River also contains transformers, which are stateful objects that transform an input via the `transform_one` method. The suffix `*_one` indicates that the input is a single data sample.

In the following example, we show a complete machine learning task (learning, prediction and performance measurement) easily implemented in a couple lines of code:

```
from river import evaluate, metrics, synth, tree

stream = synth.Waveform(seed=42).take(1000)
model = tree.HoeffdingTreeClassifier()
metric = metrics.Accuracy()
evaluate.progressive_val_score(stream, model, metric)
# >>> Accuracy: 77.58%
```

2.1 Data structure

The de facto container for *multidimensional*, homogeneous arrays of fixed-size items in Python is the `numpy.ndarray` (van der Walt et al., 2011). However, in the stream setting, data is available one sample at a time. Accordingly, dictionaries are the default data structure in River as they efficiently store *one-dimensional* data with $O(1)$ lookup and insertion (Gorelick and Ozsvaldl, 2020)[1]. Additional advantages of dictionaries include: 1. Accessing data by name rather than by position is convenient from a user perspective. 2. The ability to store different data types. For instance, the categories of a nominal feature can be encoded as strings alongside numeric features. 3. The flexibility to handle new features that might appear in the stream (feature evolution) and sparse data.

River provides an efficient Cython-based extension of dictionary structures that supports operations commonly applied to unidimensional arrays. These operations include, for instance, the four basic algebraic operations, exponentiation, and the dot product.

2.2 Pipelines

Pipelines are an integral part of River. They are a convenient and elegant way to "chain" a sequence of operations and warrant reproducibility. A pipeline is essentially a list of estimators that are applied in sequence. The only requirement is that the first $n-1$ steps are transformers. The last step can be a regressor, a classifier, a clusterer, a transformer, etc. For example, some models such as logistic regression are sensitive to the scale of the data. A best practice is to scale the data before feeding it to a linear model. We can chain the scaler transformer with a logistic regression model via a `|` (pipe) operator as follows:

```
from river import linear_model, preprocessing

model = (preprocessing.StandardScaler() |
         linear_model.LogisticRegression())
```

2.3 Instance-incremental and batch-incremental

Instance-incremental methods update their internal state one sample at a time. Another approach is to use mini-batches of data, known as batch-incremental learning. River offers some limited support for batch-incremental learning. Some models have dedicated meth-

1. The actual performance of this operations can be affected by the size of the data to store. We assume that samples from a data stream are relatively small.

Table 1: Benchmark accuracy (%) for the Elec2 data set.

model	scikit-learn	Creme	scikit-multiflow	River
GNB	73.22	72.87	**73.30**	72.87
LR	**68.01**	67.97	NA	67.97
HT	NA	74.48	**75.82**	75.55

Table 2: Benchmark processing time (seconds) for the Elec2 data set.

	scikit-learn		Creme		scikit-multiflow		River	
model	learn	predict	learn	predict	learn	predict	learn	predict
GNB	10.94 ± 0.26	5.43 ± 0.10	$\mathbf{0.32 \pm 0.01}$	3.22 ± 0.09	1.39 ± 0.02	$\mathbf{2.91 \pm 0.03}$	$\mathbf{0.32 \pm 0.01}$	3.27 ± 0.13
LR	8.72 ± 0.14	3.15 ± 0.06	2.03 ± 0.04	0.42 ± 0.01	NA	NA	$\mathbf{0.95 \pm 0.06}$	$\mathbf{0.18 \pm 0.01}$
HT	NA	NA	2.66 ± 0.06	$\mathbf{0.48 \pm 0.02}$	2.95 ± 0.06	2.21 ± 0.03	$\mathbf{0.99 \pm 0.04}$	0.65 ± 0.03

ods to process data in mini-batches, designated by the suffix `_many` instead of `_one`, e.g. `learn_one()` — `learn_many()`. These methods expect `pandas.DataFrame` (pandas development team, 2020) as input, a flexible data structure with labeled axes. This in turn allows a uniform interface for instance-incremental and batch-incremental learning.

3. Benchmark

We benchmark the implementation of 3 algorithms[2] available in scikit-learn (Pedregosa et al., 2011), Creme and scikit-multiflow: Gaussian Naive Bayes (GNB), Logistic Regression (LR) (Hastie et al., 2009), and Hoeffding Tree (HT) (Hulten et al., 2001). Table 1 shows similar accuracy between implementations (as expected) for all models. Table 2 shows the processing time (learn and predict). River models perform at least as fast but overall faster than the rest. Tests are performed on the Elec2 data set (Harries and Wales, 1999) which has 45312 samples with 8 numerical features. Reported processing time is the average of running the experiment 7 times on a system with a 2.4 GHz Quad-Core Intel Core i5 processor and 16GB of RAM. Additional benchmarks for other data sets, machine learning tasks and packages are available in the project's repository.

4. Summary

River is a machine learning package for data streams in Python. It is the merger of Creme and scikit-multiflow and supersedes said packages. The architecture is designed for both flexibility and ease of use, with the goal of facilitating the deployment of stream learning in diverse domains, both in industrial applications and in academic research. One of our next steps is to propose a canonical way to deploy online models in production. This will most likely result in another open source library, which we plan to work on in parallel of River's development.

2. These methods are selected for illustrative purposes only; scikit-learn has many other batch learning methods. On the other hand, River has a substantial set of streaming learning methods including those available in Creme and scikit-multiflow.

References

Charu C Aggarwal, Jiawei Han, Jianyong Wang, and Philip S Yu. A Framework for Clustering Evolving Data Streams - Proceedings of the 29th international conference.pdf. *Vldb*, pages 81–92, 2003. URL `http://www.vldb.org/conf/2003/papers/S04P02.pdf`.

Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47(2):235–256, 2002.

Manuel Baena-Garcıa, José del Campo-Ávila, Raúl Fidalgo, Albert Bifet, R Gavalda, and R Morales-Bueno. Early drift detection method. In *Fourth international workshop on knowledge discovery from data streams*, volume 6, pages 77–86, 2006.

S. Behnel, R. Bradshaw, C. Citro, L. Dalcin, D.S. Seljebotn, and K. Smith. Cython: The best of both worlds. *Computing in Science Engineering*, 13(2):31 –39, 2011. doi: 10.1109/MCSE.2010.118.

Albert Bifet and Ricard Gavalda. Learning from time-changing data with adaptive windowing. In *Proceedings of the 2007 SIAM international conference on data mining*, pages 443–448. SIAM, 2007.

Albert Bifet and Ricard Gavalda. Adaptive learning from evolving data streams. In *International Symposium on Intelligent Data Analysis*, pages 249–260. Springer, 2009.

Albert Bifet, Ricard Gavalda, Geoff Holmes, and Bernhard Pfahringer. *Machine Learning for Data Streams with Practical Examples in MOA*. MIT Press, 2018. `https://moa.cms.waikato.ac.nz/book/`.

Feng Cao, Martin Estert, Weining Qian, and Aoying Zhou. Density-based clustering over an evolving data stream with noise. In *Proceedings of the 2006 SIAM international conference on data mining*, pages 328–339. SIAM, 2006.

Tony F Chan, Gene H Golub, and Randall J LeVeque. Algorithms for computing the sample variance: Analysis and recommendations. *The American Statistician*, 37(3):242–247, 1983.

Krzysztof Dembczynski, Weiwei Cheng, and Eyke Hüllermeier. Bayes optimal multilabel classification via probabilistic classifier chains. In *ICML*, 2010.

John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7), 2011.

Tony Finch. Incremental calculation of weighted mean and variance. *University of Cambridge*, 4(11-5):41–42, 2009.

Philippe Flajolet, Éric Fusy, Olivier Gandouet, and Frédéric Meunier. Hyperloglog: the analysis of a near-optimal cardinality estimation algorithm. In *Discrete Mathematics and Theoretical Computer Science*, pages 137–156. Discrete Mathematics and Theoretical Computer Science, 2007.

Isvani Frias-Blanco, José del Campo-Ávila, Gonzalo Ramos-Jimenez, Rafael Morales-Bueno, Agustin Ortiz-Diaz, and Yailé Caballero-Mota. Online and non-parametric drift detection methods based on hoeffding's bounds. *IEEE Transactions on Knowledge and Data Engineering*, 27(3):810–823, 2014.

Joao Gama, Pedro Medas, Gladys Castillo, and Pedro Rodrigues. Learning with drift detection. In *Brazilian symposium on artificial intelligence*, pages 286–295. Springer, 2004.

João Gama, Indrė Žliobaitė, Albert Bifet, Mykola Pechenizkiy, and Abdelhamid Bouchachia. A survey on concept drift adaptation. *ACM Computing Surveys*, 46(4): 1–37, mar 2014. ISSN 03600300. doi: 10.1145/2523813. URL `http://dl.acm.org/citation.cfm?doid=2597757.2523813`.

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256. JMLR Workshop and Conference Proceedings, 2010.

Heitor M Gomes, Albert Bifet, Jesse Read, Jean Paul Barddal, Fabrício Enembreck, Bernhard Pfharinger, Geoff Holmes, and Talel Abdessalem. Adaptive random forests for evolving data stream classification. *Machine Learning*, 106(9):1469–1495, 2017.

Heitor Murilo Gomes, Jesse Read, and Albert Bifet. Streaming random patches for evolving data stream classification. In *2019 IEEE International Conference on Data Mining (ICDM)*, pages 240–249. IEEE, 2019a.

Heitor Murilo Gomes, Jesse Read, Albert Bifet, Jean Paul Barddal, and João Gama. Machine learning for streaming data. *ACM SIGKDD Explorations Newsletter*, 21(2):6–22, 2019b. doi: 10.1145/3373464.3373470. URL `http://dl.acm.org/citation.cfm?doid=3373464.3373470`.

Micha Gorelick and Ian Ozsvaldl. *High Performance Python*. O'Reilly Media, Inc., 2020.

Max Halford, Geoffrey Bolmier, Raphael Sourty, Robin Vaysse, and Adil Zouitine. creme, a Python library for online machine learning, 2019. URL `https://github.com/MaxHalford/creme`.

Michael Harries and New South Wales. Splice-2 comparative evaluation: Electricity pricing, 1999.

Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning: data mining, inference, and prediction*. Springer Science & Business Media, 2009.

Geoff Hulten, Laurie Spencer, and Pedro Domingos. Mining time-changing data streams. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '01*, volume 18, pages 97–106, New York, New York, USA, 2001. ACM Press. ISBN 158113391X. doi: 10.1145/502512.502529. URL `http://portal.acm.org/citation.cfm?doid=502512.502529`.

Eric Jacobsen and Richard Lyons. The sliding dft. *IEEE Signal Processing Magazine*, 20 (2):74–80, 2003.

Yuchin Juan, Yong Zhuang, Wei-Sheng Chin, and Chih-Jen Lin. Field-aware factorization machines for ctr prediction. In *Proceedings of the 10th ACM conference on recommender systems*, pages 43–50, 2016.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Tor Lattimore and Csaba Szepesvári. *Bandit algorithms*. Cambridge University Press, 2020.

Viktor Losing, Barbara Hammer, and Heiko Wersing. Knn classifier with self adjusting memory for heterogeneous concept drift. In *2016 IEEE 16th international conference on data mining (ICDM)*, pages 291–300. IEEE, 2016.

Liangchen Luo, Yuanhao Xiong, Yan Liu, and Xu Sun. Adaptive gradient methods with dynamic bound of learning rate. *arXiv preprint arXiv:1902.09843*, 2019.

Jacob Montiel, Jesse Read, Albert Bifet, and Talel Abdessalem. Scikit-multiflow: A multi-output streaming framework. *Journal of Machine Learning Research*, 19(72):1–5, 2018. URL `http://jmlr.org/papers/v19/18-251.html`.

Nikunj C Oza and Stuart J Russell. Online bagging and boosting. In *International Workshop on Artificial Intelligence and Statistics*, pages 229–236. PMLR, 2001.

Ewan S Page. Continuous inspection schemes. *Biometrika*, 41(1/2):100–115, 1954.

The pandas development team. pandas-dev/pandas: Pandas, February 2020.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Christoph Raab, Moritz Heusinger, and Frank-Michael Schleif. Reactive soft prototype computing for concept drift streams. *Neurocomputing*, 416:340–351, 2020.

Jesse Read, Luca Martino, and David Luengo. Efficient monte carlo methods for multi-dimensional learning with classifier chains. *Pattern Recognition*, 47(3):1535–1546, 2014.

Sashank J Reddi, Satyen Kale, and Sanjiv Kumar. On the convergence of adam and beyond. *arXiv preprint arXiv:1904.09237*, 2019.

Steffen Rendle. Factorization machines. In *2010 IEEE International Conference on Data Mining*, pages 995–1000. IEEE, 2010.

Herbert Robbins and Sutton Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951.

Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

Erich Schubert and Michael Gertz. Numerically stable parallel computation of (co-) variance. In *Proceedings of the 30th International Conference on Scientific and Statistical Database Management*, pages 1–12, 2018.

Jonathan A Silva, Elaine R Faria, Rodrigo C Barros, Eduardo R Hruschka, André CPLF de Carvalho, and João Gama. Data stream clustering: A survey. *ACM Computing Surveys (CSUR)*, 46(1):1–31, 2013.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Yufei Tao and Dimitris Papadias. Maintaining sliding window skylines on data streams. *IEEE Transactions on Knowledge and Data Engineering*, 18(3):377–391, 2006.

S. van der Walt, S. C. Colbert, and G. Varoquaux. The numpy array: A structure for efficient numerical computation. *Computing in Science Engineering*, 13(2):22–30, 2011. doi: 10.1109/MCSE.2011.37.

BP Welford. Note on a method for calculating corrected sums of squares and products. *Technometrics*, 4(3):419–420, 1962.

DHD West. Updating mean and variance estimates: An improved method. *Communications of the ACM*, 22(9):532–535, 1979.

Matthew D Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.