Machine Learning for Data Streams with CapyMOA

Yibin Sun[1,2], Heitor Murilo Gomes[2], Anton Lee[2], Nuwan Gunasekara[3], Guilherme Weigert Cassales[1], Jia Justin Liu[1], Marco Heyden[4], Vitor Cerqueira[5], Maroua Bahri[6], Yun Sing Koh[7], Bernhard Pfahringer[1], and Albert Bifet[1,8]

[1] AI Institute, University of Waikato, New Zealand
[2] Victoria University of Wellington, New Zealand
[3] Halmstad University, Sweden
[4] Karlsruhe Institute of Technology, Germany
[5] University of Porto, Portugal
[6] Sorbonne Université, CNRS, LIP6, France
[7] University of Auckland, New Zealand
[8] LTCI, Télécom Paris, IP Paris, France

Abstract. The exponential growth of data in recent decades has underscored the need for high-speed, real-time, and adaptive processing in machine learning. Data stream learning provides an effective framework to address this challenge. This article introduces CapyMOA, an open-source library designed specifically for data stream learning, offering powerful tools for building and deploying adaptive ML models. GitHub: https://github.com/adaptive-machine-learning/CapyMOA. Website: https://capymoa.org.

Keywords: Open-source · Data Streams · Machine Learning · Concept Drift · Online Continual Learning · Semi-supervised Learning.

1 Introduction

CapyMOA [2] is a cutting-edge open-source framework for machine learning on data streams, evolving beyond its origins as an extension of MOA [1] to offer a more comprehensive ecosystem for real-time analytics. It supports a diverse range of streaming algorithms while integrating modern machine learning libraries such as PyTorch [3] and Scikit-learn [4]. With optimized performance, scalable processing, and advanced evaluation strategies, CapyMOA enables seamless experimentation with high-velocity data streams. By continuously incorporating novel algorithms and state-of-the-art tools, it provides researchers and practitioners with a powerful platform for developing and benchmarking next-generation stream learning models. In this work, we provide code snippets and screenshots to demonstrate CapyMOA's abilities. A demonstration video is presented at: https://youtu.be/OEYUe6q04u4.

2 CapyMOA Key Features

Integration with Established Tools. CapyMOA provides a straightforward Python interface to the well-established algorithms and functionalities available in MOA by utilizing JPype as a bridging library. Additionally, CapyMOA integrates algorithms, datasets, and utilities from PyTorch and Scikit-learn, further expanding its applicability.
High Level Evaluation Functions. CapyMOA provides standard evaluation loops in stream learning as evaluation functions.

```
from capymoa.evaluation import prequential_evaluation
result = prequential_evaluation(stream, learner, window_size=500)
```

Concept Drift. CapyMOA simulates different types of concept drift using the `DriftStream` class and stores the drifting information in the stream. The following code defines a stream possessing an abrupt drift after 5,000 instances, and a gradual drift happening between 9,000 and 11,000 instances.

```
from capymoa.stream.generator import SEA
from capymoa.stream.drift import
    (DriftStream, AbruptDrift, GradualDrift)
stream = DriftStream(stream=[SEA(1), AbruptDrift(position=5000),
         SEA(3), GradualDrift(position=10000, width=2000), SEA(1)])
```

Dedicated Visualization Functions. CapyMOA offers a variety of visualization functions that are specifically designed for streaming scenarios. Fig. 1 exhibits a plotting example. The stream data used in this plot is the same as in the previous subsection, and the associated drifts are highlighted by a red vertical line (abrupt drift) a shaded area (gradual drift).

```
from capymoa.evaluation.visualization import plot_windowed_results
plot_windowed_results(knn_result, ht_result, arf_result,
                      metric='accuracy')
```

Pipelines. Building a pipeline is challenging in data stream scenarios because it requires continuous updates and synchronization, especially when concept drift occurs. CapyMOA tackles this by introducing the `PipelineElement` class — a modular component that supports feature selection, normalization, missing-value imputation, parameter searching and tuning, and more.

```
from capymoa.stream.preprocessing import
    (ClassifierPipeline, MOATransformer)
from moa.streams.filters import NormalisationFilter
from capymoa.drift.detectors import ADWIN
normalisation = MOATransformer(schema=stream.get_schema(),
                moa_filter=NormalisationFilter())
pipeline = ClassifierPipeline().add_transformer(normalisation)
           .add_classifier(learner).add_drift_detector(ADWIN())
```

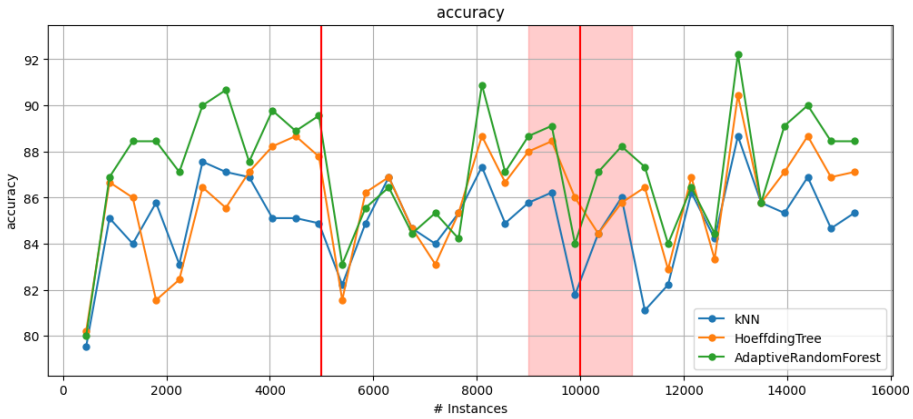

Fig. 1: Prequential Accuracy Over Time Highlighting Two Concept Drifts

3 Learning Tasks

Since learning on data streams is the main focus of CapyMOA, plenty of streaming tasks and functionalities are provided.

Supervised Learning. The supervised learning procedures are wrapped into the prequential evaluation function (aforementioned in Section 2), including classification, regression, and prediction interval.

Semi-supervised Learning. CapyMOA also supports the under-explored Semi-supervised learning for data streams, including algorithms and evaluation functions.

```
from capymoa.ssl.classifier import OSNN
from capymoa.evaluation import prequential_ssl_evaluation
osnn = OSNN(schema=stream.get_schema(), optim_steps=10)
results_osnn = prequential_ssl_evaluation(stream=stream,
               learner=osnn, label_probability=0.01)
```

Unsupervised Learning.

– **Data Stream Clustering.** CapyMOA supports most clustering algorithms from MOA while introduces a redesigned evaluation framework for a more streamlined process. It visualizes the evolution of micro- and macro-clusters over time for 2D datasets (as illustrated in Fig. 2) and supports the extraction of clustering metrics based on established methods from the literature.

```
from capymoa.cluster import Clustream_with_kmeans as WithKmeans
from capymoa.cluster.visualization import plot_clustering_evolution
from capymoa.stream.generator import RandomRBFGeneratorDrift
plot_clustering_evolution(RandomRBFGeneratorDrift(), WithKmeans(),
                          frame_duration=1000)
```

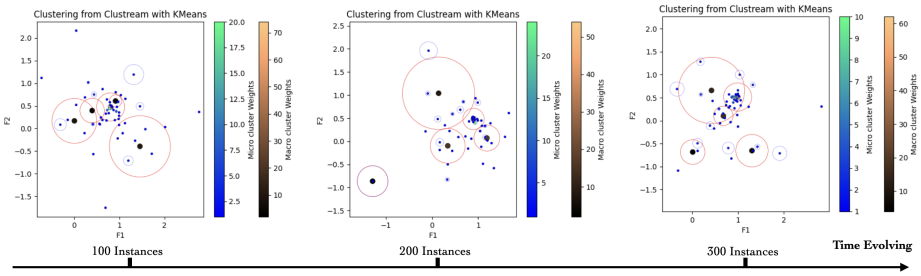

Fig. 2: Clusters Evolving Over Time

– **Anomaly Detection.** CapyMOA includes a wide range of anomaly detection algorithms from MOA. In addition, it features cutting-edge algorithms and is constantly updated with the latest ones.

```
from capymoa.evaluation import prequential_evaluation_anomaly
from capymoa.anomaly import HalfSpaceTrees
from capymoa.stream import NumpyStream
stream_ad = NumpyStream(X, y, "AD_Dataset", feature_names,
            target_name, "categorical")
hst = HalfSpaceTrees(schema=stream_ad.get_schema())
results_hst = prequential_evaluation_anomaly(stream=stream_ad,
              learner=hst, window_size=1000)
```

AutoML for Data Streams. CapyMOA provides AutoML capabilities for streaming data by the introduction of the `AutoClass` class, which reads a json file containing algorithm configuration options and automatically selects the best-performing one for prediction.

```
from capymoa.automl import AutoClass
autoclass = AutoClass(schema=schema,
            configuration_json="./settings_autoclass.json")
results_autoclass = prequential_evaluation(stream=stream,
                    learner=autoclass)
```

AutoML in CapyMOA can also be applied using Random search to find the best configuration for a combination of a preprocessor and a learner in a **streaming scenario**. This is achieved via the use of pipelines, enabled by the `RandomSearchClassifierPE` class, which facilitates the composition of preprocessing steps, learning algorithms, and automated hyperparameter optimization.

```
from capymoa.stream.preprocessing import RandomSearchClassifierPE
randomsearch_pe = RandomSearchClassifierPE(HoeffdingTree,
                  hparams_ranges, n_combinations, rng)
```

Online Continual Learning. In a recent release, CapyMOA introduced support for Online Continual Learning (OCL), an advanced research area that integrates continual learning with stream learning. Similar to the previously introduced stream learning interface, OCL in CapyMOA also offers high-level evaluation and additional functionalities. An example code snippet is shown below.

```
from capymoa.classifier import HoeffdingTree
from capymoa.datasets.ocl import TinySplitMNIST
from capymoa.evaluation.ocl import ocl_train_eval_loop
scenario = TinySplitMNIST()
model = HoeffdingTree(scenario.schema)
metrics = ocl_train_eval_loop(model, scenario.train_streams,
          scenario.test_streams)
```

4 Conclusions

CapyMOA is an open-source platform for machine learning and continual learning on streaming data, supporting both Java and Python. It offers essential tools for building, training, and evaluating models in real-time environments.
Education. CapyMOA helps students learn stream learning concepts through hands-on experience.
Research. Its transparency and flexibility support reproducible and extensible experimentation.
Development. Developers benefit from easy prototyping and integration into real-world applications.

Overall, CapyMOA serves as a valuable resource across education, research, and development, lowering the barrier to effective streaming data analysis. Please refer to the CapyMOA website for more information and tutorials, and [2] for an empirical comparison of CapyMOA against other frameworks.

References

1. Bifet, A., Holmes, G., Pfahringer, B., Kranen, P., Kremer, H., Jansen, T., Seidl, T.: MOA: Massive online analysis, a framework for stream classification and clustering. In: Proceedings of the first workshop on applications of pattern analysis. pp. 44–50. PMLR (2010)
2. Gomes, H.M., Lee, A., Gunasekara, N., Sun, Y., Cassales, G.W., Liu, J.J., Heyden, M., Cerqueira, V., Bahri, M., Koh, Y.S., Pfahringer, B., Bifet, A.: CapyMOA: Efficient machine learning for data streams in python (2025), https://arxiv.org/abs/2502.07432
3. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al.: Pytorch: An imperative style, high-performance deep learning library. Advances in neural information processing systems **32** (2019)
4. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al.: Scikit-learn: Machine learning in python. the Journal of machine Learning research **12**, 2825–2830 (2011)