DOI: 10.58895/ksp/1000162754-1 erschienen in:
Proceedings. 33. Workshop Computational Intelligence - Berlin, 23. – 24. November 2023
DOI: 10.5445/KSP/1000162754 | https://www.ksp.kit.edu/site/books/e/10.5445/KSP/1000162754/

Cognitive Architecture for Artificial Intelligence: Evaluating Realworld Applicability and the Significance of Online Machine Learning

Richard Schulz[1], Alexander Hinterleitner[1], Lukas Hans[1], Aleksandr Subbotin[1], Nils Barthel[1], Noah Pütz[1], Martin Rosellen[1], Thomas Bartz-Beielstein[1], Christoph Geng[2], Phillip Priss[2]

[1]Institute for Data Science, Engineering, and Analytics, TH Köln
Steinmüllerallee 6, 51643 Gummersbach
E-Mail: {richard.schulz, alexander.hinterleitner, lukas.hans, aleksandr.subbotin, nils.barthel, noah.pütz, martin.rosellen, thomas.bartz-beielstein}@th-koeln.de

[2] TH OWL
Campusallee 12, 32657 Lemgo
E-Mail: {christoph.geng, phillip.priss}@th-owl.de

1 Introduction

As the integration of hardware and software continues to evolve, production systems are becoming increasingly intricate, now often referred to as Cyber-Physical Production Systems (CPPS). Particularly, Artificial Intelligence (AI) can be instrumental in improving processes such as anomaly detection, optimization, or predictive maintenance. However, at the moment, incorporating AI algorithms into these systems is far from straightforward; it demands substantial time, financial resources, and expertise. Adopting a standardized architecture could facilitate the integration of AI technologies, especially for small and medium-sized enterprises, empowering them to remain competitive. For this reason the Cognitive Architecture for Artificial Intelligence (CAAI) was introduced in [1] as cognitive architecture for AI in CPPS. The goal of the system is to reduce the implementation effort by creating a standard architecture. The core of the CAAI is a cognitive module that processes the user's declarative goals, selects suitable models and algorithms, and creates a configuration for the execution of a processing pipeline on a big data platform. During the revision of the CAAI project, it became obvious that there are existing limitations in automating the algorithm pipeline development process for all environments and use cases. This is mainly due to rapidly changing software interfaces and transfer complexities. Additionally, it became apparent that the architecture in the use case for CPPS provides a good environment for the implementation of online machine learning (OML) algorithms. This can be explained by the continuous data streams produced by the system's machines. This abstract assesses the potential advantages of OML algorithms through an analysis of a real-world application in slitting machines.

Section 2 roughly describes the concept of OML. Furthermore, it includes a description of the experimental setup, including its real-world application. In Section 3 the results of the experiment are discussed. Finally, a short conclusion is presented at the end of the abstract.

2 Materials and Methods

2.1 Online Machine Learning

The amount of data generated from various sources has increased enormously in recent years ("Big Data"). Technological advances have enabled the continuous collection of data. Web, social media, share prices, search queries, but also sensors of modern machines produce continious streams of data. It becomes more and more challenging to store and process these infinite streams. Traditional batch machine learning is the common strategy to train machine learning models. It basically boils down to the following steps[2]:

1. Loading and pre-processing the train data;
2. Fitting a model to the data;
3. Calculating the performance of the model on the test data;

In modern OML approaches, we do not train the model with the entire dataset at once; instead, we update it incrementally with the newly arriving data. This way, we avoid storing large amounts of data by discarding it after the updating process. This procedure might be beneficial in terms of time and memory consumption. Additionally, this continuous updating allows OML methods to better address structural changes within the data, known as concept drift.The introduction of different methods of incremental learning has been quite slow over the years, but the situation is changing at the moment [3] [4] [5].

To compare OML with the classical batch learning method in our experiments, a Hoeffding Tree regressor (HTR) from the Python 'river' library [3] is used for online learning, while a Decision Tree regressor from the 'scikit-learn' [8] package is used for the classical approaches. In online machine learning, Hoeffding trees are preferred because they do not rely on previously used instances, instead they await the arrival of new instances [7]. Given their incremental learning capacity, Hoeffding trees are more adept at handling a data streaming context compared to traditional methods.

2.2 Experiment Setup: Slitting Machines

In the experiments discussed in this work, data was collected using a test setup for winding stations from "Kampf Schneid- und Wickeltechnik GmbH & Co. KG", a company that specializes in building machines for slitting and winding web-shaped materials. The idea of the experiment is to use the motor torque and revolution values of a slitting machine to predict the vibration level. All features are available in form of time series with with measuring intervals of 10 ms.

For our experiment, we divide the data into a training and a test set. The goal is to compare the prediction performance over an evaluation horizon that is subdivided into segments of 150 data points. Additionally, we analyze the calculation time and memory consumption during the evaluation. To compare OML with classical approaches, and to assess their respective strengths and weaknesses, we utilize four different approaches in our experiment. For all approaches, we train the initial model based on the training set before we start with the evaluation process. Three of the algorithms belong to the group of batch machine learning techniques. The first is the classical batch learning approach, where the model is trained only once on the training set and subsequently evaluated on the test set. Another method is the landmark approach, where we add the data from the current horizon to the training set after each evaluation step, and then train the model from scratch. The final batch learning method is the shifting window approach. Unlike the landmark approach, the algorithm is not trained on the entire set of observed data, but rather on a moving window of data. In our case, this window is the size of the initial training set, meaning we add 150 data points with each evaluation step and remove the earliest 150 data points. The last evaluation technique is the pure OML approach where we update the model incrementally after each prediction step with the new data. The implementations of all evaluation strategies can be found under the following link: `https://github.com/sequential-parameter-optimization/spotRiver`.

3 Results

In the following part, we compare the different approaches as introduced in Section 2.2.

The performance of the different approaches is visualized in the top graph of Figure 1. It shows how the MAE evolves over the evaluation horizon. All batch learning evaluation methods produce comparable results. Initially, performance degrades slightly and then improves continuously. The OML approach comparatively achieves constant results and outperforms the batch evaluations over the entire horizon.

The second diagram in Figure 1 shows a comparison of the computation times of the different methods. As assumed, the landmark and shifting-window methods show a continuous increase in computation time due to the models need to be retrained at each evaluation iteration. In contrast, the conventional batch learning approach exhibits a much lower processing time because of its singular model training phase. On the other hand, the OML algorithm

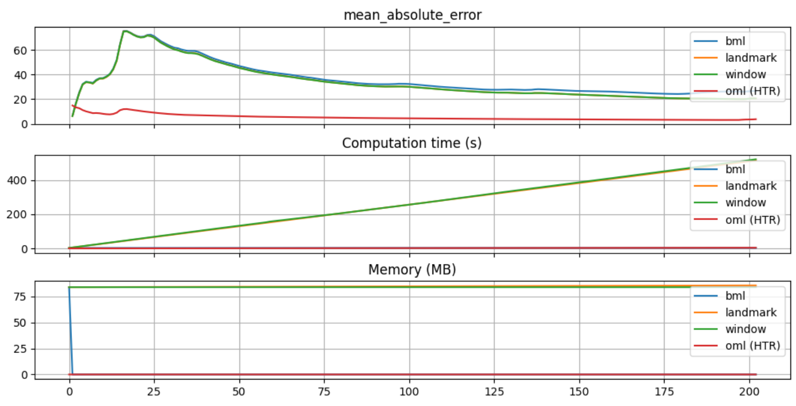

Figure 1: The MAE, computation time, and memory consumption of different approaches for each evaluation step. The MAE plot shows an overlap of the graphs of all bml methods. In the plots for computation time and memory consumption, the curves of landmark and shifting window, as well as the OML and the classical batch method overlap.

achieves time efficient results as well. This is because OML updates models incrementally, rather than training from scratch with each evaluation.

The lowest graph of Figure 1 shows the memory consumption. Here, the OML approach also delivers comparable results to the basic batch approach. However, it should be emphasized again that the batch approach's memory consumption only takes place during the training step, and the remaining consumption is negligible. This fact is also visualized by the graph. In the first evaluation step, the memory consumption for the classic batch method drops towards zero. The shifting window and the landmark approach perform comparably poorly. This is mainly due to the generated model, which must be built again in each iteration.

Furthermore, the assertions regarding the superior performance of the OML algorithms have been statistically substantiated by a one-sided t-test. This test demonstrates that the average deviation of predictions made by the OML algorithm from the actual values is significantly smaller than that observed in predictions from all batch learning approaches.

4 Conclusion and Discussion

The OML algorithms outperformed the classical approaches not only in terms of memory consumption and computation time, but they also achieved significantly better results in terms of prediction accuracy. This can be attributed to the algorithms' enhanced responsiveness to concept drift, a decisive advantage especially in the domain of production machinery. Further improvements to the results presented here could be realized through additional experiments, particularly with surrogate model based optimization of the hyperparameters. This evidence suggests that OML algorithms should undoubtedly be considered in the development of CAAI for CPPS.

Acknowledgement

This research work is funded by the German Federal Ministry for Economic Affairs and Climate Action as part of the project "*IMProvT_II* – Intelligente Messverfahren zur energetischen Prozessoptimierung von Trinkwasserbereitstellung und -verteilung II", funding code 03EN2086A.

References

[1] A. Fischbach, J. Strohschein, A. Bunte, J. Stork, H. Faeskorn-Woyke, N. Moriz, and T. Bartz-Beielstein. “CAAI—a cognitive architecture to introduce artificial intelligence in cyber-physical production systems”. The International Journal of Advanced Manufacturing Technology, 111:609–626. Springer. 2020.

[2] E. Bartz, T. Bartz-Beielstein, M. Zaefferer, and O. Mersmann. “Hyperparameter Tuning for Machine and Deep Learning with R: A Practical Guide”. Springer Nature. 2023.

[3] J. Montiel, M. Halford, S.M. Mastelini, G. Bolmier, R. Sourty, R. Vaysse, A. Zouitine, H.M. Gomes, J. Read, T. Abdessalem, et al. “River: machine learning for streaming data in Python”. 2021.

[4] A. Bifet, R. Gavalda, G. Holmes, B. Pfahringer. “Machine Learning for Data Streams with Practical Examples in MOA”. MIT Press. 2018. `https://moa.cms.waikato.ac.nz/book`.

[5] V. Losing, B. Hammer, H. Wersing. “Incremental on-line learning: A review and comparison of state of the art algorithms”. Neurocomputing, 275, 1261–1274. 2018. `https://doi.org/10.1016/j.neucom.2017.06.084`.

[6] A. Bifet, R. Gavalda. “Adaptive learning from evolving data streams”. In Proceedings of the Advances in Intelligent Data Analysis VIII: 8th International Symposium on Intelligent Data Analysis, IDA 2009, Lyon, France, August 31-September 2, 2009. Proceedings 8. Springer. 2009, pp. 249–260.

[7] P.M. Domingos, G. Hulten. “Mining high-speed data streams”. In Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining, Boston, MA, USA, August 20-23, 2000; R. Ramakrishnan, S.J. Stolfo, R.J. Bayardo, I. Parsa, Eds. ACM. 2000, pp. 71–80. `https://doi.org/10.1145/347090.347107`.

[8] 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, E. Duchesnay. “Scikit-learn: Machine Learning in Python”. Journal of Machine Learning Research. 2011, 12, pp. 2825–2830.