The Journal of Open Source Software

SCALAR - A Platform for Real-time Machine Learning Competitions on Data Streams

Nedeljko Radulovic[1], **Dihia Boulegane**[1, 2], **and Albert Bifet**[1, 3]

1 LTCI, Télécom Paris, IP-Paris, Paris, France **2** Orange Labs, Grenoble, France **3** University of Waikato, New Zealand

DOI: 10.21105/joss.02676

Software

- Review
- Repository
- Archive

Editor: Gabriela Alessio Robles

Reviewers:

- @GregaVrbancic
- @atanikan
- @xiaohk

Submitted: 31 August 2020
Published: 05 December 2020

Summary

SCALAR is a new platform for running real-time machine learning competitions on data streams. Following the intent of Kaggle, which serves as a platform for organizing machine learning competitions adapted for batch learning, we propose SCALAR as a novel platform explicitly designed for stream learning in real-time. SCALAR supports both classification and regression problems in the data streaming setting. It has been developed in Python, using state of the art open-source solutions: Apache Kafka, Apache Spark, gRPC, Protobuf, and Docker.

Statement of need

Existing machine learning platforms for research and competitions, such as Kaggle and Alibaba Tianchi provide static data sets that users leverage to do batch training and scoring. These popular platforms, especially Kaggle, enable users to propose better solutions for a wide range of machine learning problems and applications, pushing forward the whole research community.

While these platforms enable collaborative model building, they do not provide real-time settings and data streaming instead of static data sets. Inspired by the great reception and adoption that these platforms have had, SCALAR was built for real-time environments by supporting data streaming to fill this gap, helping users test and improve their methodologies in a real-time machine learning scenario.

On SCALAR, data are continuously released in batches during defined time intervals. Predictions for each current batch sent before designated deadline are evaluated in real-time, and the results are shown on a live leaderboard.

SCALAR can be applied to provide real-time machine learning support in multiple scenarios, one of the more notable and recent ones was its role as the core organizing and coordination platform for the first Real-time Machine Learning Competition on Data Streams (Boulegane et al., 2019) as part of the IEEE Big Data 2019 Cup Challenges.

Streaming learning setting

Most of the existing platforms for data science competitions are tailored to offline learning where a static dataset is made available to the participants before the competition starts. This dataset is divided into training and test sets. The training set is used to build and train the model, which is then scored on the test set.

In online learning, data arrive in a stream of records (instances) and the model needs to be trained incrementally as new instances are released. Since the data arrive at high speed, predictions have to be issued within a short time. Considering this specific setup of the data stream mining scenario (Figure 1), every model should use a limited amount of time and memory, process one instance at a time and inspect it only once. The model must be able to predict at any time (Bifet et al., 2010).

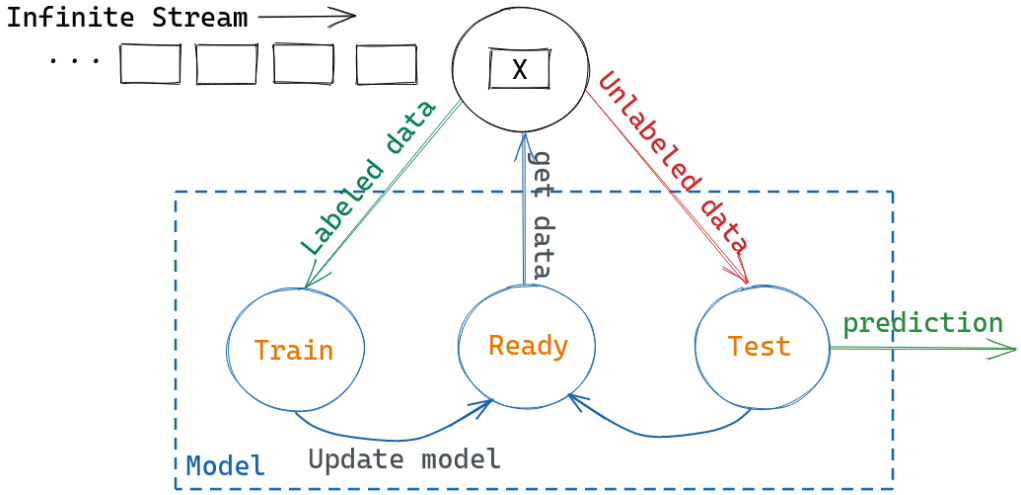

Figure 1: Stream data mining scenario

The model is continuosly updated using the labeled instances and then evaluated on new unlabeled instances. This scenario represents the prequential evaluation scenario or "test-then-train" setting, to make it possible we first assume that the records in the data stream arrive in batches and that each batch can be of size 1 or more. Then, we define two different kinds of batches:

- Initial batch: This batch is used to build and train the initial model. It is aimed to avoid the cold start problem and as such, contains both features and targets. The initial batch usually has a large number of instances.
- Regular batch: The regular batch contains new test instances while providing training instances of previous batches that are used to evaluate the quality of the model up to the current time.

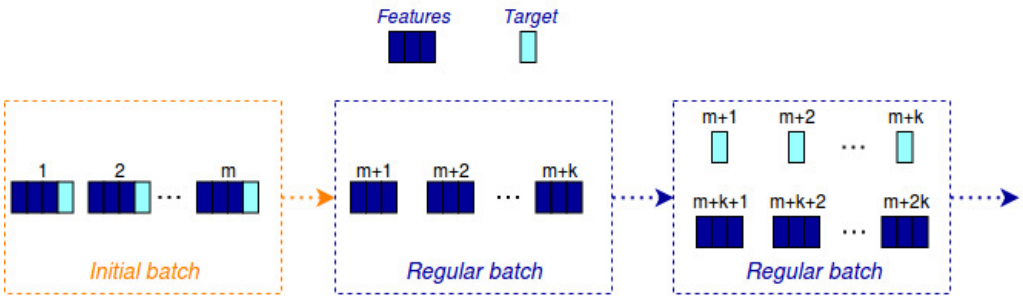

Figure 2: Initial and regular batches in the data stream

Architecture

To support all the aforementioned requirements for stream data mining and to be able to organize competitions in such a scenario, a specifically dedicated platform was needed. To the best of our knowledge, there doesn't exist such a platform. Thus we propose SCALAR. Figure highlights the architecture of the platform that includes a web application, and a streaming engine following the fundamentals of Information Flow Processing (IFP)(Cugola & Margara, 2012).

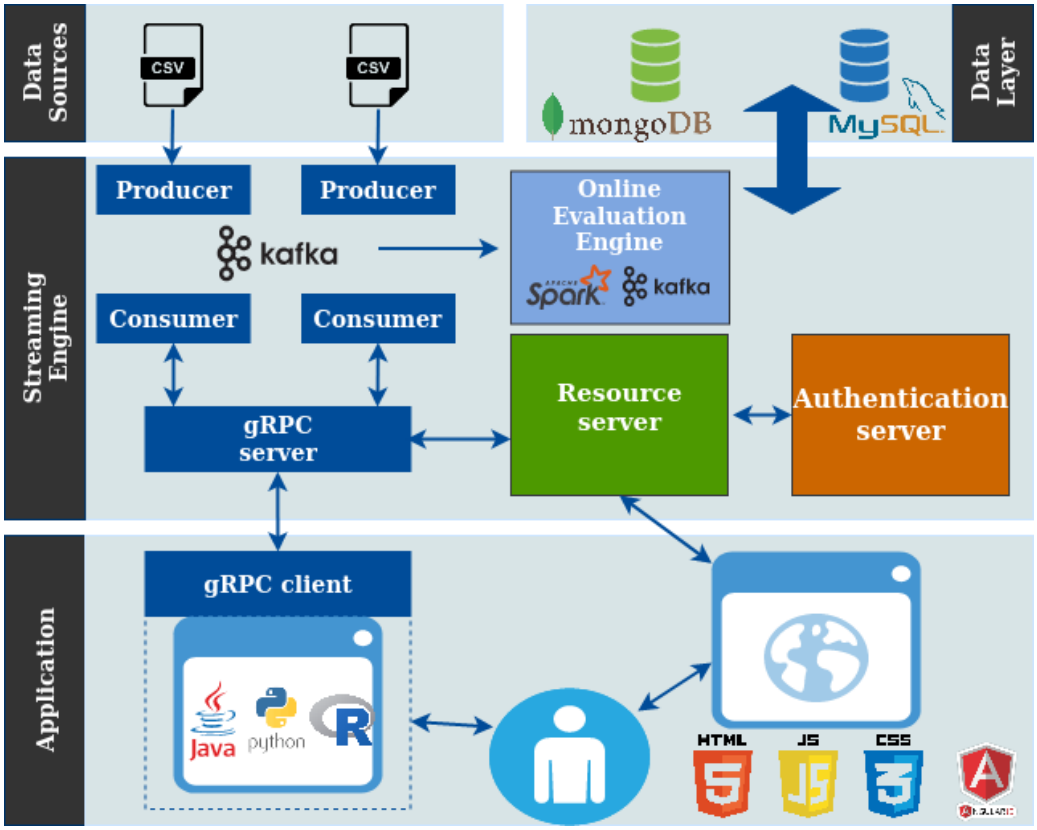

Figure 3: Architecture of the platform

Layers and components' descriptions:

- Data sources: SCALAR enables competitions by providing data in the form of a CSV file, which is then used to recreate a continuous stream following predefined competition settings such as the time interval between two releases, and the number of instances at a time.
- Data layer: represents the persistence node in the system where different kinds of information are stored. MongoDB is used for unstructured data (e.g., user's predictions, records from the stream, evaluation metrics, etc.) and MySQL for structured data (competition information, user information).
- Streaming Engine: this layer is responsible for handling the streams. From the CSV files, Kafka recreates multiple streams to be sent to multiple users. SCALAR provides a secure and independent bi-directional streaming communication between the user and the streaming engine. This allows each user to consume training and test instances and submit the respective predictions according to the competition predefined data format.

The resource server is the `API` that handles all authenticated requests from the client application, whereas the authorization server is in charge of users' identification. The Online evaluation engine handles both the stream of instances, and the streams of participants' predictions in order to compute and update the evaluation metrics online before storing them into the dedicated database.

- Application layer: consists of two parts the web application, and client application. The web application represents a user-friendly interface that allows participants to register, subscribe to competitions, and follow leaderboard and model performance online. The client application is provided to accommodate participants' code to solve the machine learning problem at hand. It delivers a secure communication channel to receive the stream of training and test instances and send their respective predictions to the server.

Acknowledgements

We would like to thank Nenad Stojanovic for his contribution to the project and to acknowledge support for this project from the EDF (Electricité de France) and OpenML.

References

Armbrust, M., Das, T., Torres, J., Yavuz, B., Zhu, S., Xin, R., Ghodsi, A., Stoica, I., & Zaharia, M. (2018). Structured streaming: A declarative API for real-time applications in apache spark. *Proceedings of the 2018 International Conference on Management of Data*, 601–613.

Bifet, A., Holmes, G., Kirkby, R., & Pfahringer, B. (2010). Moa: Massive online analysis. *Journal of Machine Learning Research*, *11*(May), 1601–1604.

Boulegane, D., Radulovic, N., Bifet, A., Fievet, G., Sohn, J., Nam, Y., Yu, S., & Choi, D.-W. (2019). Real-time machine learning competition on data streams at the IEEE big data 2019. *2019 IEEE International Conference on Big Data (big Data)*, 3493–3497. https://doi.org/10.1109/BigData47090.2019.9006357

Cugola, G., & Margara, A. (2012). Processing flows of information: From data stream to complex event processing. *ACM Comput. Surv.*, *44*(3), 15:1–15:62. https://doi.org/10.1145/2187671.2187677

Garg, N. (2013). *Apache kafka*. Packt Publishing Ltd.

Kreps, J., Narkhede, N., Rao, J., & others. (2011). Kafka: A distributed messaging system for log processing. *Proceedings of the NetDB*, *11*, 1–7.

Merkel, D. (2014). Docker: Lightweight linux containers for consistent development and deployment. *Linux Journal*, *2014*(239), 2.

Montiel, J., Read, J., Bifet, A., & Abdessalem, T. (2018). Scikit-multiflow: A multi-output streaming framework. *The Journal of Machine Learning Research*, *19*(1), 2915–2914.

Team, R. C., & others. (2013). *R: A language and environment for statistical computing*. Vienna, Austria.

Zaharia, M., Chowdhury, M., Franklin, M. J., Shenker, S., Stoica, I., & others. (2010). Spark: Cluster computing with working sets. *HotCloud*, *10*(10-10), 95.