Scalable Online Learning for Flink

SOLMA Library

W. Jamil, N-C. Duong, W. Wang, C. Mansouri, S. Mohamad, A. Bouchachia*
Department of Computing and Informatics, Machine Intelligence Group,
Bournemouth University, Poole, United Kingdom, BH12 5BB
{wjamil,cduong,wangw,cmansouri,smohamad,abouchachia}@bournemouth.ac.uk

ABSTRACT

Driven by the needs of Flink to expand the offline engine to a hybrid one, a new machine learning (ML) library, called SOLMA is proposed. This library aims to cover online learning algorithms for data streams. In this setting, data streams are processed sequentially example by example. SOLMA, which is under development, currently contains two classes of algorithms: (i) basic streaming routines such as online sampling, online PCA, online statistical moments and (ii) advanced online ML algorithms covering in particular classification, regression and drift/anomaly detection and handling. This paper briefly highlights the concepts underlying SOLMA.

KEYWORDS

Online learning, Scalability, Flink

ACM Reference Format:
W. Jamil, N-C. Duong, W. Wang, C. Mansouri, S. Mohamad, A. Bouchachia. 2018. Scalable Online Learning for Flink: SOLMA Library. In *Proceedings of BDVA Workshop on Software Architecture Challenges in Big Data (SACBD@ECSA'18).* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3241403.3241438

1 INTRODUCTION

Learning from high volume and high velocity continuous flow of data poses scientific challenges that need to be addressed. We aim at designing and developing a library of scalable streaming algorithms for predictive analytics and automatic knowledge discovery called SOLMA, standing for **S**calable **O**nline machine **L**earning and data **M**ining **A**lgorithms. It is very important to underline the fact that there exist few attempts to develop machine learning libraries for big data platforms that handle data streams. Hence, the primary goal of SOLMA is to contribute to the development of new online learning algorithms for high speed data streams.

The current state of the art of machine learning (ML) algorithms for Big Data is dominated by offline learning algorithms that process data often stored in the cloud. Offline algorithms use "full memory" which corresponds to iterative learning using the whole data [12]. In such setting, data can be revisited as many times as desired. In online learning, on the other hand, the algorithms see the data only once. To alleviate terminology ambiguities, by *streaming algorithms*, we refer to *sequential one-pass* algorithms that do not see the data more than once. Also we use streaming algorithms interchangeably with *online algorithms*. In this later, algorithms are fully online, operating with "no memory". Although there have been many developments related to online learning, the scalability remains still an issue [23]. There is currently an increasing interest from the machine learning and data mining communities to develop scalable online machine learning algorithms that are capable of efficiently handling continuous high-speed streams.

In streaming, ML algorithms should be adaptive and inherently open-ended [3, 4] to be amenable to refinement if data continues to arrive. When designing streaming algorithms, some criteria should be observed such as: the ability to generate an anytime model independently from the order of the training examples; use of a single scan of the data and efficient processing of data in a constant time; the ability to handle concept drift and novelty detection; and competitively performing against their offline counterpart.

The current state-of-the-art of algorithms shows that not all existing streaming algorithms meet the criteria mentioned earlier. Often there is a confusion between online learning in the traditional interpretation (data examples are processed sequentially, but the algorithm can still recycle over the data) and the online streaming algorithms where no recycling over the data is allowed. However, when restricted to one pass, traditional online learning may turn to be interesting if purposefully and adequately adapted. Nevertheless, it is not clear how good they can scale up in presence of continuous, high speed, and potentially distributed data streams.

From the perspective of Big Data, there exist many ML libraries dedicated to batch processing. Some are for big data such as GraphLab, Giraph, Pegasus, MLib and Mahout, while others are not. An almost exhaustive list of existing libraries is presented in a report for the H2020 PROTEUS project[1]. A comparative analysis of these libraries indicates that are almost all of them are developed for batch processing. There has been few attempts to develop libraries of streaming algorithms, mostly SOLMA and SAMOA. The idea of SAMOA was to distribute and parallelise the algorithms of the Massive Online Analytics (MOA) library targeting Storm and S4, but currently contains very few ML algorithms.

This research has been driven by two main motivations: (i) the need for scalable online learning algorithms to cope with high-velocity streams and (ii) lack of such algorithms for Flink as a novel big data platform. The advantageous feature of Flink from the perspective of Machine Learning is that it brings both batch and the stream processing together in the same environment as shown

*The corresponding author.

ECSA '18, September 24–28, 2018, Madrid, Spain

ACM ISBN 978-1-4503-6483-6/18/09…$15.00
https://doi.org/10.1145/3241403.3241438

[1] https://www.proteus-bigdata.com/app/download/9431544070/D4.3-V5$_final.pdf$?t = 1512389477

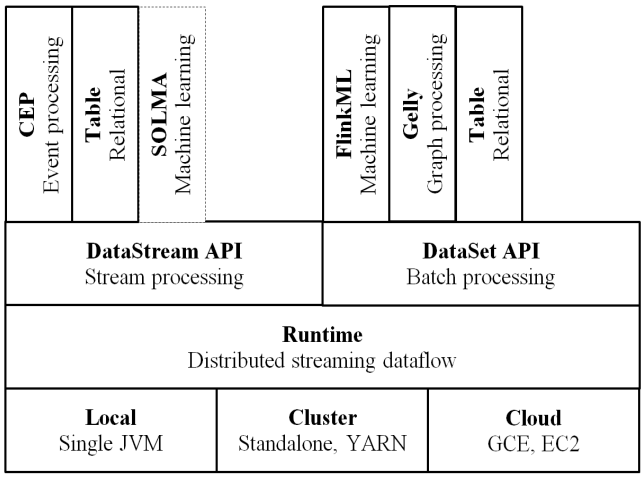

Figure 1: SOLMA as a module of Flink

in Figure 1 [2, 14], leading to low latency and high throughput. Therefore, SOLMA offers a response to the stream analytics needs for such big data platform.

The rest of this paper is organised as follows. In Section 2, a brief overview of SOLMA is given. Section 3 presents the technical and conceptual considerations of SOLMA algorithms before the paper is concluded in Section 4.

2 SOLMA LIBRARY

SOLMA intends to cover two classes of algorithms: basic streaming routines such as moments, sampling, heavy hitters feature extraction, and advanced machine learning algorithms such as classification, clustering, regression, drift handling and anomaly detection.

2.1 Basic algorithms

The list of basic algorithms to compute descriptive statistics consists of the following:

- *Online moments*: simple mean, simple variance, weighted mean, weighted variance, exponentially weighted mean and variance, moving average, aggregation algorithm.
- *Online sampling*: Simple reservoir sampling [26], weighted reservoir sampling [9] and adaptive reservoir sampling [1]
- *Online frequent directions* [13]
- *Incremental principal component analysis* [30]

2.2 Classification algorithms

- *Online support vector machines (OSVM)*: SVM has proven to be an efficient classification tool for batch learning. Recently, an online version of SVM was proposed to cope with large datasets. Examples of this approach are Pegasos [25] and Norma [20].
- *Online bi-level stochastic gradient for support vector machines (OBSG-SVM)* [28]: OBSG-SVM is proposed for adjusting SVM with an online selection of the hyperparameter C. It is based on the bi-level stochastic gradient algorithm for SVM (BSG-SVM) [6]. In OBSG-SVM algorithm, we use a different validation method on the outer level of the bi-level optimisation problem to enable the online selection of C. Convergence to a stationary point has been proven in [7] with stochastic moves.
- *Online passive-aggressive algorithms (PA)* [8]: PA is an example of adversarial online learning. In each round, PA receives a data point and predicts its label. After the prediction, the correct label is revealed and the algorithm suffers an instantaneous loss. PA updates the weight vector with a quantity that minimises the suffered loss. It aggressively forces the loss to be zero and passively makes sure that the updated weight vector will be not too far from the previous weight vector.

2.3 Regression algorithms

- *Online ridge regression (ORR)*: The algorithm performs the well studied ridge regression algorithm [16] in online mode. Ridge Regression adds a L_2 norm penalty term to ordinary least squares regression to deal with multicollinearity amongst regression predictor variables.
- *Online shrinkage via limit of Gibbs sampling (OSLOG)*: The algorithm is an online version of shrinkage via limit of Gibbs sampling (SLOG) [24]. OSLOG uses an L_1 norm penalty resulting in a difficult problem to bound because L_1 norm is non differentiable but is convex. An approximation has been used to obtain sparsity in the solution [18].
- *Aggregating algorithm for regression (AAR)*: This algorithm can be thought of as a game-theoretic version of ORR [27]. AAR algorithm is last-step min-max optimal [11], which allows AAR to shrink the predictions. Due to shrinkage in its prediction AAR is less likely to over-fit in comparison to ORR. Also, AAR has a better upper bound on the cumulative loss in comparison to ORR.
- *Competitive online iterated ridge regression (COIRR)*: COIRR can be thought of as a game-theoretic version of OSLOG. The algorithm has the best upper bound on cumulative loss under certain conditions [18].

2.4 Drift handling and anomaly detection

- *Online weighted averaging passive-aggressive algorithm* (WAPA) [29]: This algorithm employs the weighted average with passive-aggressive (PA) updates (See Sec 2.2). It passively retains the weighted average (WA) of the previous weight vectors when the hinge loss is zero; otherwise, it reduces the hinge loss suffered on the current data point less aggressively than PA. WAPA enhances the robustness and the learning ability when dealing with fluctuations (e.g. label noise).
- *Online normalised least mean square regression (ONLMSR)* [17]: This algorithm is a competitive online regression algorithm. It has the ability to handle drift based on first order information and is computationally the most efficient regression algorithm in SOLMA.
- *Anomaly detection using incremental PCA*: Candid covariance-free incremental PCA (CCIPCA) [30] is used to compute

PCs data incrementally without re-estimating the covariance matrix. Anomaly detection uses Hotelling T^2 and squared prediction error (SPE or Q). The T^2 statistic detects variation within the PC subspace. The Q statistic measures the lack of fit of the data to the PCs.

3 CONCEPTUAL CONSIDERATIONS

Apache Flink is a stream processing framework that offers an open-source software stack for implementing data processing applications at large scale. SOLMA users can make use of Flink which generalises the concepts of the MapReduce programming model to offer not only Map and Reduce functions but also high-level transformations such as Join, Filter, Aggregations, Iterations, etc. Moreover, Flink extends the (Key, Value) pair data model of MapReduce by enabling the use of any Java or Scala data types making data pre-processing tasks very convenient and concise for the user.

On the other hand, SOLMA developers can exploit the ability of Flink to chain together different transformers and predictors, resulting in a pipelined processing model which makes it easy for developers to implement ML algorithms. SOLMA developers can implement iterative algorithms by defining a step function and embedding it into an iteration or a delta iteration [10] which is then executed in Flink's pipelined processing engine, where intermediate results are forwarded ahead to next operator in the pipelined execution.

In our experience we found various event controls and the windowing mechanism in Flink very useful for implementing our online learning algorithms since:

- windows over time allow sampling the data stream.
- windowing with Event Time semantics helps perform computations over the streams when the arrival of events is delayed or out of order.
- control events (checkpoint barriers, watermarks and iteration barriers) are very useful for developers to control the streaming rates and avoid back pressure [5].

The technical challenge in the design of distributed algorithms is how to achieve lossless parallelism for the online learning algorithms. In existing literature, parallelism is achieved by either data parallelism [15, 22] or model parallelism [19, 21]. Data parallelism requires replicating the model over different machines and each model/machine receives chunks of data. The replicas of the model on each machine synchronise the model parameters after a fixed number of presentations . In contrast, the model parallelism is partitioned in sub-models that correspond to different tasks.

Furthermore, although exposing a shared data is fairly straightforward, efficiently providing model parallelism in existing applications is non-trivial. It requires modifying machine learning algorithms to ensure that the model is split such that the communication costs are limited within each data presentation (this is zero for data-parallelism). SOLMA's goal is to provide a simple, general purpose API that integrates easily with the Apache Flink framework, with a wide variety of algorithms and reasonable development effort. Hence, SOLMA limits itself to data parallelism.

In data-parallelism learning, model replicas are trained over multiple machines and each replica trains over a subset of the data. All the distributed algorithms in SOLMA are implemented as Flink programs, which are inherently parallel and distributed. During execution, a stream has one or more partitions, and each operator has one or more operator subtasks. The operator subtasks are independent of one another and execute in different threads and possibly on different machines. SOLMA achieves parameter synchronization between machines by following the Master-Slave architecture. The stream is distributed to workers and the master receives parameters from each worker, master updates the parameter and sends the updated parameters to each worker. Updates of parameters are asynchronous when the communication between the master and the workers is not synchronised, possibly due to the delay in the arrival of the stream and the converse refers to the synchronous case. The implementation of the parameter server [22, 31] allows synchronisation of the model parameters between the workers and the master.

4 CONCLUSION

In this short paper, we introduced SOLMA as a library for streaming data developed for Flink. It is being populated by basic streaming routines and advanced machine learning algorithms. Currently it contains a number of scalable online algorithms. In the near future, we aim to cover other algorithms, especially in the area of clustering, semi-supervised and active learning.

ACKNOWLEDGMENTS

The European Commission supports the authors under the Horizon 2020 Grant 687691 related to the project *PROTEUS: Scalable Online Machine Learning for Predictive Analytic and Real-Time Interactive Visualisation*

REFERENCES

[1] M. Al-Kateb, B. Lee, and X. Wang. 2007. Adaptive-Size Reservoir Sampling over Data Streams. In *Proceedings of the 19th International Conference on Scientific and Statistical Database Management (SSDBM '07)*. IEEE Computer Society, Washington, DC, USA, 22–. https://doi.org/10.1109/SSDBM.2007.29

[2] A. Alexandrov et al. 2014. The stratosphere platform for big data analytics. *The VLDB Journal* 23, 6 (2014), 939–964.

[3] A. Bouchachia and R. Mittermeir. 2007. Towards incremental fuzzy classifiers. *Soft Computing* 11, 2 (2007), 193–207.

[4] A. Bouchachia and C. Vanaret. [n. d.]. *GT2FC*.

[5] P. Carbone, A. Katsifodimos, S. Ewen, V. Markl, S. Haridi, and K. Tzoumas. 2015. Apache Flink: Stream and batch processing in a single engine. *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering* 36, 4 (2015).

[6] N. Couellan and W. Wang. 2015. Bi-level Stochastic Gradient for Large Scale Support Vector Machine. *Neurocomputing* 153 (2015), 300–308.

[7] N. Couellan and W. Wang. 2017. On the Convergence of Stochastic Bi-level Gradient Methods. *e-print Optimisation online* (2017).

[8] K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer. 2006. Online Passive-Aggressive Algorithms. *J. Mach. Learn. Res.* 7 (Dec. 2006), 551–585.

[9] P. Efraimidis and P. Spirakis. 2006. Weighted random sampling with a reservoir. *Inform. Process. Lett.* 97, 5 (2006), 181 – 185. https://doi.org/10.1016/j.ipl.2005.11.003

[10] S. Ewen, K. Tzoumas, M. Kaufmann, and V. Markl. 2012. Spinning fast iterative data flows. *Proceedings of the VLDB Endowment* 5, 11 (2012), 1268–1279.

[11] J. Forster. 1999. On relative loss bounds in generalized linear regression. In *International Symposium on Fundamentals of Computation Theory*. Springer, 269–280.

[12] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia. 2014. A survey on concept drift adaptation. *ACM computing surveys (CSUR)* 46, 4 (2014), 44.

[13] M. Ghashami, E. Liberty, J. Phillips, and D. Woodruff. 2015. Frequent Directions : Simple and Deterministic Matrix Sketching. *CoRR* abs/1501.01711 (2015). arXiv:1501.01711

[14] Apache Hadoop. 2015. The Apache software foundation.

[15] Q. Ho, J. Cipar, H. Cui, S. Lee, J. Kim, P. Gibbons, G. Gibson, G. Ganger, and E. Xing. 2013. More effective distributed ml via a stale synchronous parallel parameter server. In *Advances in neural information processing systems*. 1223–1231.

[16] A. Horel. 1962. Applications of Ridge Analysis to regression Problems. *Chem. Eng. Progress.* 58 (1962), 54–59.
[17] W. Jamil and A. Bouchachia. 2018. Competitive Normalised Least Squares Regression. *In preparation* (2018).
[18] W. Jamil and A. Bouchachia. 2018. Competitive Online Regularised Regression. *Machine Learning (submitted)* (2018).
[19] J. Kim et al. 2016. STRADS: a distributed framework for scheduled model parallel machine learning. In *Proceedings of the Eleventh European Conference on Computer Systems*. ACM, 5.
[20] J. Kivinen, A. Smola, and R. Williamson. 2003. Online Learning with Kernels.
[21] S. Lee et al. 2014. On model parallelization and scheduling strategies for distributed machine learning. In *Advances in neural information processing systems*. 2834–2842.
[22] M. Li et al. 2014. Scaling Distributed Machine Learning with the Parameter Server.. In *OSDI*, Vol. 14. 583–598.
[23] J. Qiu, Q. Wu, G. Ding, Y. Xu, and S. Feng. 2016. A survey of machine learning for big data processing. *EURASIP Journal on Advances in Signal Processing* 2016, 1 (2016), 67.
[24] B. Rajaratnam, S. Roberts, D. Sparks, and O. Dalal. 2016. Lasso regression: estimation and shrinkage via the limit of Gibbs sampling. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 78, 1 (2016), 153–174.
[25] S. Shalev-Shwartz, Y. Singer, N. Srebro, and A. Cotter. 2011. Pegasos: primal estimated sub-gradient solver for SVM. *Mathematical Programming* 127, 1 (01 Mar 2011), 3–30.
[26] J. Vitter. 1985. Random Sampling with a Reservoir. *ACM Trans. Math. Softw.* 11, 1 (March 1985), 37–57.
[27] V. Vovk. 2001. Competitive On-line Statistics. *International Statistical Review* 69, 2 (2001), 213–248.
[28] W. Wang and A. Bouchachia. 2018. Online Bilevel Stochastic Gradient for SVM. *In preparation* (2018).
[29] W. Wang, W. Jamil, and A. Bouchachia. 2018. Online weighted averaging passive-aggressive algorithms. *In preparation* (2018).
[30] W. Weng, Y. Zhang, and W. Hwang. 2003. Candid covariance-free incremental principal component analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 25, 8 (2003), 1034–1040.
[31] E. Xing et al. 2015. Petuum: A new platform for distributed machine learning on big data. *IEEE Transactions on Big Data* 1, 2 (2015), 49–67.