Master project
Summer term 2018

HPI Hasso Plattner Institut
IT Systems Engineering | Universität Potsdam

Software 2.0 – Online Machine Learning Models

Introduction

Mark Andreesen wrote in the year 2011 [1] that software is eating the world. Now many are saying that machine learning is eating software [2]. This latter idea was recently labeled “software 2.0” [3] and it basically consists of creating software by training models instead of manually coding them.

Proponents of software 2.0 claim that technology such as deep learning will bring a set of benefits to develop software. Software will be more homogenous, faster to create, higher performance, more portable, more configurable, and adaptable, less fault-prone, etc. [3]. Although there is controversy about these benefits [4], there is also agreement that these models have inherent advantages and came to stay [5].

From a software engineering standpoint, similar claims were raised before and could be subsumed under the idea of a software that creates software. Take for instance componentization [5] and a set of technologies cited by Brooks, among them object-oriented programming, artificial intelligence, and automatic programming [6]. In the face of historically missed silver-bullets [6], it might be more realistic to assume that the technology will, when the hype subsides, provide a new plateau of productivity [7].

Innovation and challenges

Therefore, the more interesting question is not about the speed of adoption of machine learning models (MLM), instead it is about the challenges we face to engineer these models. From an architectural standpoint, an MLM consists of at least two components: (1) the data features necessary to make accurate predictions and (2) the mathematical function to compute the predictions from these features. Other components can be (3) the hyperparameter configurations and the (4) data preprocessing and sampling strategies. These are at least four components that demand specification, design, coding (reuse), and testing.

While in the current practice MLMs is usually employed once to produce predictions based on a fixed data set, an online MLM (OMLM) is hooked to a data pipeline [8] that continuously delivers new data. This new data is consumed by the online OMLM to produce predictions in real time.

Note however that as new data arrives, the patterns in the data might differ from the ones originally used to train the model. Depending on the type of differences, the quality of predictions might decay or be biased [9]. In this case, the MLM will fail in the longer run while a OMLM can be retrained with the new data.

The decisions on when and how to retrain the OMLM are difficult for many reasons. We might not know how much of the changes in the new data are permanent. In the case that changes in the data affects only part of the predictions, we should be able to make risk/benefit decisions between reusing the old model or redesigning (training) a new one. Note that training a new model can be very expensive and not all models can be incrementally trained.

Ultimately, an OMLM as a software component faces at least the following challenges: modularization, refactoring, versioning (model and data), and testing (validation and verification). These are the topics we will explore in the project.

Project goals

The goal of the project is to characterize the innovative aspects of OMLMs and discuss challenges and risks involved. To accomplish that we will have two deliverables:

- survey of the state of art of engineering OMLMs
- predictive model running on one or two cloud platforms (Table-1)

The survey will comprise of three parts: report on the literature (see suggested material), perform interviews (online survey) with software engineers and data engineers, and write a comparative analysis discussing current pros, cons, opportunities, and risks.

We will provide the predictive model to the students. We currently have predictive models in the domains of Questions and Answers (e.g., StackOverflow) and Utility-Based Decision Making (e.g., Self-adaptive systems). We are also open to new case studies for predictive models.

The goal is to deploy, run, and retrain the predictive model on one or two platforms (Table-1). Although these platforms already provide basic support for application deployment and access to machine learning libraries and databases, we do not know the challenges to evolve an OMLM on these platforms. In this sense we will mostly focus on studying the activities of training (refactoring), versioning (data and model), and testing (validate and verify) the predictive model.

Table-1 Candidate Cloud Platforms for Studying Online Machine Learning Models

PLATFORM	REFERENCE	APIS
ÜBER MICHELANGELO	https://eng.uber.com/michelangelo/	HDFS, Spark, Samza, Cassandra, MLLib, XGBoost, TensorFlow
SELDON	https://www.seldon.io/	Kubernets
MICROSOFT AZURE	https://docs.microsoft.com/en-us/azure/machine-learning/preview/model-management-service-deploy	Spark, Keras, Tensorflow, and Python
GOOGLE TFX	http://www.kdd.org/kdd2017/papers/view/tfx-a-tensorflow-based-production-scale-machine-learning-platform	Tensorflow
OTHERS	Amazon AWS and SAP Hanna	-

The project results will also be divided in two parts. Part-1 will cover the analysis of the survey data and recommendations for a generalized process for engineering OMLMs. Part-2 will contain the discussion of the challenges and lessons learned while deploying and running the predictive model on one or two cloud platforms. The conclusion of the report will point to claims that were confirmed and/or contradicted in parts one and two.

Skill and knowledge to be acquired

We expect that the students will develop a broad knowledge about OMLMs (process and architecture) and be able to speak with authority about the challenges of engineering this type of software.

Concerning the organization, the project will have an introductory set of lectures to equalize the knowledge about the following topics and tools:

- Introduction to large scale models
  - Reinforcement Learning [10]
  - Neural Networks and Deep learning [11]
  - Random Forests and Boosted Trees [12]
- Fundamentals
  - Model Selection and Hyper-Parameter Tuning [13]
  - Markov-Chain Monte Carlo [14]
  - Bayesian Networks and Decision graphs [15]
  - Decision theory - Utility and Voting [16]
- Tools
  - Data analysis and model building: R, Weka or Scikit-learn
  - Neural networks and Reinforcement learning: Tensorflow and OpenAI tools
  - Integration and batch processing: Java or Python

Information

For questions and more information, we will be happy to chat about it. Please send us an email.

Professor. Holger Giese (holger.giese@hpi.de)

Christian Adriano (christian.adriano@hpi.de)

References

[1] Andreessen, M., "Why is software eating the world," available at: https://a16z.com/2016/08/20/why-software-is-eating-the-world/

[2] Warden, P., "Deep learning is eating software," available at: https://petewarden.com/2017/11/13/deep-learning-is-eating-software/

[3] Karpathy, A., "Software 2.0“, available at:

https://medium.com/@karpathy/software-2-0-a64152b37c35

[4] Perez, C., "Is deep learning software 2.0?", available at: https://medium.com/intuitionmachine/is-deep-learning-software-2-0-cc7ad46b138f

[5] McIlroy, M. Douglas, et al., (1968), "Mass-produced software components," in *proceedings of the 1st ICSE.*

[6] Brooks, F. P., (1987), "No silver bullet," *IEEE Computer* 20 (4), pp. 10-19.

[7] Gartner, “Hype Cycle Methodology”, available at: https://www.gartner.com/technology/research/methodologies/hype-cycle.jsp

[8] Wetzler, M., “Architecture of giants: data stacks at Facebook, Netflix, Airbnb, and Pinterest”, available at: https://blog.keen.io/architecture-of-giants-data-stacks-at-facebook-netflix-airbnb-and-pinterest-9b7cd881af54

[9] Mhamdi, E., Guerraoui, R., Hendrikx, H., & Maurer, A., (2017), "Dynamic Safe Interruptibility for Decentralized Multi-Agent Reinforcement Learning," *in proceedings of NIPS*, available at: https://arxiv.org/pdf/1704.02882.pdf

[10] Kochenderfer, M. J., (2015), *Decision making under uncertainty: theory and application*. MIT press.

[11] Goodfellow, I., Bengio, Y., & Courville, A., (2016), *Deep learning*, MIT press.

[12] Chen, T., & Guestrin, C., (2016), “Xgboost: A scalable tree boosting system,” *in proceedings of the 22nd ACM SIGKDD,* pp. 785-794, available at: http://www.kdd.org/kdd2016/papers/files/rfp0697-chenAemb.pdf

[13] Kuhn, M., & Johnson, K. (2013), *Applied predictive modeling* (Vol. 810), Springer.

[14] Andrieu, C., De Freitas, N., Doucet, A., & Jordan, M. I., (2003), “An introduction to MCMC for machine learning,” *Machine learning*, *50* (1-2), pp.5-43, available at: http://www.cs.ubc.ca/~arnaud/andrieu_defreitas_doucet_jordan_intromontecarlomachinelearning.pdf

[15] Nielsen, T. D., & Jensen, F. V., (2009), *Bayesian networks and decision graphs*, Springer Science & Business Media.

[16] Peterson, M., (2017), *An introduction to decision theory*, Cambridge University Press.