Multi-target Compiler for the Deployment of Machine Learning Models

Oscar Castro-Lopez
Facultad de Informatica
Universidad Autonoma de Sinaloa
Culiacan, Mexico
oscarcastro@uas.edu.mx

Ines F. Vega-Lopez
Parque de Innovacion Tecnologica
Universidad Autonoma de Sinaloa
Culiacan, Mexico
ifvega@uas.edu.mx

***Abstract*—The deployment of machine learning models into production environments is a crucial task. Its seamless integration with the operational system can be quite challenging as it must adhere to the same software requirements such as memory management, latency, and scalability as the rest of the system. Unfortunately, none of these requirements are taken into consideration when inferring new models from data. A straightforward approach for deployment consists of building a pipeline connecting the modeling tools to the operational system. This approach follows a client-server architecture and it may not address the design requirements of the software in production, especially the ones related to efficiency. An alternative is to manually generate the source code implementing the model in the programming language that was originally used to develop the software in production. However, this approach is usually avoided because it is a time-consuming and error-prone task. To circumvent the aforementioned problems, we propose to automate the process of machine learning model deployment. For this, we have developed a special-purpose compiler. Machine learning models can be formally defined using a standard language. We use this formal description as an input for our compiler, which translates it into the source code that implements the model. Our proposed compiler generates code for different programming languages. Furthermore, with this compiler we can generate source code that exploits specific characteristics of the systems hardware architecture such as multi-core CPUs and graphic processing cards. We have conducted experiments that indicate that automated code generation for deploying machine learning models is, not only feasible but also efficient.**

***Index Terms*—Machine Learning Deployment, ML Compiler, ML Engineering, PMML**

I. Introduction

Data availability and advances in computing power have enabled an incredible growth in machine learning research and practice. Nowadays machine learning modeling tools abound and are commonly used for rapid model building and prototyping. However, the actual value of machine learning models can only be harnessed when they are deployed to production environments for either automating or aiding in the decision making process. Therefore, a repeatable, fast, and reliable deployment process is crucial for successful machine learning workflows.

Moving machine learning models from the prototype, or proof-of-concept, stage to production, where we expect both, their seamless integration and efficient execution, is a challenging task. We observe that, while a clear option for deployment, the manual translation of machine learning models to production software is a time-consuming and error-prone task that requires a team of highly specialized professionals. As an alternative to manual translation, it is possible to use a client-server architecture and create a pipeline between the production software and the modeling tools. Although this approach effectively reduces the time-to-deploy, it creates a technological dependency [1]. Additionally, a client-server approach introduces a potential bottleneck because modeling tools are not designed to be high performance servers.

II. Multi-target compiler

Our research aims to automate the process of machine learning model deployment and its efficient execution in production environments. Our proposal is based on the fact that every machine learning model type has a well defined set of features that defines it, thus it can be formally expressed. In fact, currently at least two standards exist for describing machine learning models, namely Predictive Model Markup Language (PMML) and Portable Format for Analytics (PFA). Hence, to facilitate deployment, we have designed and developed a special-purpose compiler that translates machine learning models from their formal description into source code. A high-level design is shown in Figure 1.

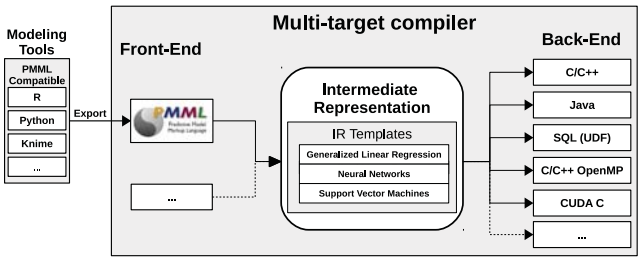

Fig. 1. Overall design of the multi-target compiler, centered on our intermediate representation to allow standard formats as inputs and source code as outputs for an efficient execution in production environments.

The front-end translates formal descriptions of models to an internal representation. The back-end translates the internal representation to a specific target programming language. The internal representation is a set of templates that supports a dynamic architecture allowing different machine learning

models as inputs and many programming languages as outputs. We define several building blocks such as linear algebra operations and machine learning functions to create a lean internal representation.

III. Experimental results

Currently, our compiler consumes models described in PMML and targets the C, Java, and SQL (via user defined functions) languages. Additionally, we generate source code that uses multiple cores (C-OpenMP) or GPUs (C-CUDA). A reduced version of our compiler is currently available as an R package [2]. We have conducted experiments to evaluate the running time of models deployed using our compiler. Figures 2 and 3 show preliminary performance results that illustrate the efficiency of the generated source code.

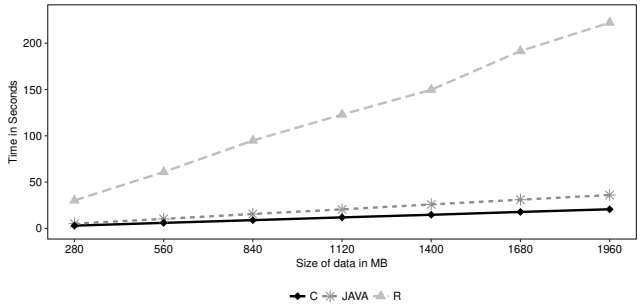

Fig. 2. Experimental results of single-threaded execution time of a generalized linear model using R and generated source code in C and Java. The model was built using the dataset described in [3] and the source code was generated using the glm.deploy R package [2].

IV. Conclusions

Our experimental evaluation shows not only that we can effectively use the compiler to deploy machine learning models but also that the source code generated runs efficiently in production environments. Furthermore, the modularity in our compiler allows for an easy extension of both new types of models and new target languages.

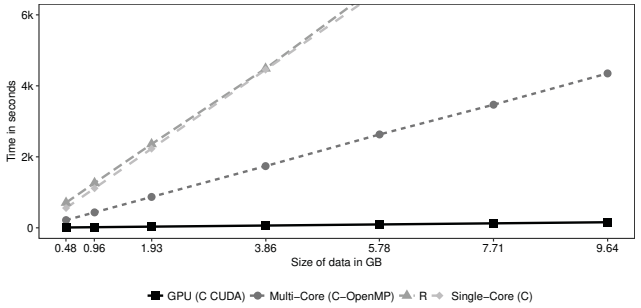

Fig. 3. Experimental results of single-threaded, multi-threaded, and GPU execution time of a support vector machine model using R and generated source code in C, C with OpenMP and C CUDA. The model was built using the dataset described in [4].

References

[1] D. Sculley, G. Holt, D. Golovin, E. Davydov, T. Phillips, D. Ebner, V. Chaudhary, M. Young, J.-F. Crespo, and D. Dennison, "Hidden Technical Debt in Machine Learning Systems," in *Proceedings of the International Conference on Neural Information Processing Systems - Volume 2*, Cambridge, MA, USA, 2015, pp. 2503–2511. [Online]. Available: http://dl.acm.org/citation.cfm?id=2969442.2969519

[2] O. Castro-Lopez and I. Vega-Lopez, *glm.deploy: 'C' and 'Java' Source Code Generator for Fitted Glm Objects*, 2018, r package version 1.0.4. [Online]. Available: https://CRAN.R-project.org/package=glm.deploy

[3] I.-C. Yeh and C. hui Lien, "The comparisons of data mining techniques for the predictive accuracy of probability of default of credit card clients," *Expert Systems with Applications*, vol. 36, no. 2, Part 1, pp. 2473 – 2480, 2009. [Online]. Available: https://doi.org/10.1016/j.eswa.2007.12.020

[4] E. Lopez-Rojas, A. Elmir, and S. Axelsson, "PaySim: A financial mobile money simulator for fraud detection," in *Proceedings of the European Modeling and Simulation Symposium*. Larnaca, Cyprus: Dime University of Genoa, 2016, pp. 249–255.