Machine Learning and Configurable Systems: A Gentle Introduction

Juliana Alves Pereira[1], Hugo Martin[2], Paul Temple[3], Mathieu Acher[2]
[1]PUC-Rio, Rio de Janeiro, Brazil, [2]Univ Rennes, IRISA, Inria, France; [3]University of Namur, Belgium
jpereira@inf.puc-rio.br,hugo.martin@irisa.fr,paul.temple@unamur.be,mathieu.acher@irisa.fr

ABSTRACT

The goal of this tutorial is to give a gentle introduction to how machine learning can be used to support software product line configuration. This is our second practical tutorial in this trending field. The tutorial is based on a systematic literature review and includes practical tasks (specialization, performance and bug prediction) on real-world systems (Linux, VaryLaTeX, x264). The material is designed for academics and practitioners with basic knowledge in software product lines and machine learning.

CCS CONCEPTS

• **Software and its engineering → Software product lines**;

KEYWORDS

Software Product Lines, Machine Learning, Configurable Systems

ACM Reference format:
Juliana Alves Pereira[1], Hugo Martin[2], Paul Temple[3], Mathieu Acher[2]. 2020. Machine Learning and Configurable Systems: A Gentle Introduction. In *Proceedings of 24th ACM International Systems and Software Product Line Conference, MONTREAL, QC, Canada, October 19–23, 2020 (SPLC '20),* 1 pages.
DOI: 10.1145/3382025.3414976

Motivation & Contents. Configurable software systems allow stakeholders to derive product variants that meet their specific functional and non-functional requirements. A straightforward way to find a suitable configuration is to measure the target (non-functional) property of each individual product variant, and then *e.g.* search for the configuration with the best performance. In practice, this process is usually unfeasible due to the combinatorial explosion of possible variants and the long measurement time needed to compute the non-functional property of a given configuration. Machine learning techniques play a central rule when it comes to predicting the behavior of configurable systems and mastering its complexity. In this tutorial, we rely on the pattern *"sampling, measuring, learning, validation"* emerged in the software engineering and machine learning literature [4]. The usual process is to sample some configurations, execute and measure them, and learn out of configuration measurements. The hope is that the learning phase generalizes well to the whole configuration space.

We will demonstrate the use of different sampling and learning techniques and which applications they are more likely to be applied from *pure prediction* to *automated specialization* and *program understanding*. The practical tasks will be conducted on three case studies: *Linux* [1], *VaryLaTeX* [2] and *x264* [3].

The tutorial is a half-day event structured as follows:

Part 1 (theoretical) – Motivation and a complete overview of the progress made in this field. We will start with a motivation session relying on the Linux kernel, a highly-configurable system with an enormous space. With 15K+ options, it is impossible to explore the whole configuration space, hence the need to rely on statistical machine learning approaches. Next, we will introduce a catalog of *"sampling, measurement, learning, validation"* techniques used in numerous recent works in the field [4]. We give concrete examples to illustrate the advantages of different techniques.

Part 2 (practical) – Specialization: the case of *VaryLaTeX.* In this practical session, we exercise how configurable systems can be *specialized* thanks to learning techniques used to automatically mine constraints among options. We use an intuitive example: a learning system, called *VaryLaTeX* [2], capable of generating LaTeX paper variants that respect constraints (*e.g.*, page limits). This part aims to explain how to frame a specialization problem as a classification problem and illustrate how to read and interpret decision trees which are used to specialize configurable systems.

Part 3 (practical) – Performance and bug prediction: the case of Linux and x264. We show how we can predict performance properties (e.g., execution time and size) and bugs of unlabeled configurations of x264 [3] and Linux [1]. With this part, attendees will understand how to frame performance and bug prediction problems as regression and classification problems, respectively.

Part 4 (theoretical) – Conclusion. In this section, we will recap all lessons learned, and point out limitations and open challenges that need attention in future work (e.g. resources cost vs error cost).

Link to the material (including slides, data, procedures[1]):
https://github.com/VaryVary/ML-configurable-SPLCTutorial

Acknowledgements. VaryVary project grant ANR-17-CE25-0010-01 and the CAPES grant 88887.473590/2020-00.

REFERENCES

[1] Mathieu Acher, Hugo Martin, Juliana Alves Pereira, Arnaud Blouin, Jean-Marc Jézéquel, Djamel Khelladi, Luc Lesoil, and Olivier Barais. 2019. Learning Very Large Configuration Spaces: What Matters for Linux Kernel Sizes. (2019).

[2] Mathieu Acher, Paul Temple, Jean-Marc Jézéquel, José A. Galindo, Jabier Martinez, and Tewfik Ziadi. 2018. VaryLATEX: Learning Paper Variants That Meet Constraints. In *VAMOS*. 83–88.

[3] Juliana Alves Pereira, Mathieu Acher, Hugo Martin, and Jean-Marc Jézéquel. 2020. Sampling Effect on Performance Prediction of Configurable Systems: A Case Study. In *ACM/SPEC ICPE*. 277–288.

[4] Juliana Alves Pereira, Hugo Martin, Mathieu Acher, Jean-Marc Jézéquel, Goetz Botterweck, and Anthony Ventresque. 2019. Learning Software Configuration Spaces: A Systematic Literature Review. (2019).

[1]Pre-requisite: Python, scikit-learn, and Jupyter notebooks must be installed.