Formal Design of Cyber-Physical Systems with Learning-Enabled Components

Thao Dang[1]

[1]Univertity Grenoble Alpes, CNRS, Grenoble INP, VERIMAG, Grenoble, France

Abstract

We present two learning methods for designing Cyber-Physical Systems with Learning-Enabled Components: one is imitation learning of feedback controllers, and the other is learning of temporal specifications.

Keywords

AI, Formal Methods, Overlay

1. Introduction

Artificial intelligence (AI) and data sciences are revolutionizing information systems used for control and supervision of various devices (e.g., sensors, robots, IoT devices) with higher levels of autonomy in uncertain and dynamically changing environments. Among such information systems are Cyber-Physical Systems (CPS) from which emerges a new generation of AI-intensive Cyber-Physical Systems. There are a number of challenges in designing such systems. On one hand, the existing CPS design methodologies, relying on rather fixed models, face a fundamental problem because these systems containing learning-enabled components are supposed to learn from experience and interactions with the environment, and adapt their behaviors accordingly. It is thus imperative to ensure that their learning-enabled components work correctly. On the other hand, AI techniques are "unpredictable" due to a lack of formal framework to provide safety guarantees. In general, the outcomes of learning activities in AI components (e.g., deep neural nets) are not well-understood and interpretable. When coupling CPS with AI, the increased heterogeneity in dynamics and behaviors can aggravate the reliability and explainability issues, if the learning activities are not properly formalized.

In this work we present our recent results in this direction, namely making design of learning-enabled components in AI-intensive CPS more formal. More concretely, we first describe a method for imitation learning of feedback controllers satisfying temporal specifications. We then describe an approach for learning temporal specifications from labelled data. In both of these learning problems, temporal specifications are given in Signal Temporal Logic (STL) [1], a formal language that finds widespread use in formal methods and increasing adoption in industry [2].

OVERLAY 2023: 5th Workshop on Artificial Intelligence and Formal Verification, Logic, Automata, and Synthesis, November 7, 2023, Rome, Italy

thao.dang@univ-grenoble-alpes.fr (T. Dang)

2. Imitation learning for synthesizing state feedback controllers satisfying temporal specifications

Our goal is to integrate formal specification and validation techniques in the Imitation Learning (IL) methodology for synthesizing feedback controllers for complex dynamical systems. While formal methods have the advantage of rigorous formalization and reasoning, they are very limited in the complexity and scalability of the problems that can be practically solved. Imitation Learning, also known as learning from demonstrations, involves the process of learning how to mimic the behavior of an expert by observing their actions in a given task [3]. It has many successful applications in various fields such as robotics, natural language processing, image and speech recognition.

We focus on the problem of training a neural network (NN) (playing the role of a learner) to imitate a complex controller (playing the role of an expert). The ultimate goal is to replace this complex controller with a trained NN. NNs have long been used to control dynamical systems from inverted pendulums to quadcopters, learning from scratch to control the plant by maximizing an expected reward, e.g. [4, 5]. NNs can also be trained to replace an existing controller that is unsatisfactory for non-functional reasons, e.g., computationally expensive (consider model-predictive control [6]), slow, or energy intensive. A well-trained NN controller can provide similar control performance much faster and is readily implemented on cheap and energy-efficient embedded platforms [7].

To make such an imitation learning framework more formal and more efficient, we add the following novel features: (i) a formalization of performance evaluation for both the learner's and expert's policies using their abilities to satisfy requirements specified by temporal logic, (ii) a leverage of the power of existing temporal logic property falsification tools to create correct training data, (iii) a new method of data aggregation in order to guarantee a good performance of NN in terms of imitation and generalization.

To explain these features, we point out some major difficulties in this problem. Training a NN to imitate a feedback controller is more complex than the problem of approximating a function using pairs of input and output values, since feedback controllers can themselves be stateful dynamical systems. We identify the following difficulties in data generation by executing the nominal controller in closed loop:

- *Infinite behavior space.* The behavior space is not only large but can also be infinite. It is thus important to define a coverage measure to quantify how representative the generated training data is.
- *Non-uniform accuracy.* Depending on the control requirement, the NN may need to be very precise around some region of the state space while in other regions a rough approximation is acceptable.

Formal control requirements. The problem of non-uniform accuracy is particularly pronounced when the requirement depends on time or sequences of events. This is frequently the case in control applications, where properties such as *rise time*, *settling time*, and *overshoot* are typical. We consider complex properties that can include not only time but also causal relationships. They can be conveniently described in *Signal Temporal Logic* (STL) [2]. An STL formula φ

consists of atomic predicates along with logical and temporal operators. Atomic predicates are defined over signal values and have the form $g(y(t)) \sim 0$, where g is a scalar-valued function over the signal y evaluated at time t and $\sim \in \{<, \leq, >, \geq, =, \neq\}$. Temporal operators "always" ($\square$), "eventually" ($\lozenge$), and "until" ($\mathcal{U}$) have the usual meaning and are scoped using intervals of the form (a, b), $(a, b]$, $[a, b)$, $[a, b]$, or (a, ∞), where $a, b \in \mathbb{R}_0^+$ and $a < b$.

Additionally, in order to allow some quantitative behavioral flexibility, we use *parametric* STL (PSTL) to specify the expected properties of a nominal controller. Then, we want to train a neural network-based controller achieving performance comparable to that of the nominal controller, as measured by *valid* parameters of the PSTL requirement. Parametric STL (PSTL) is a variant of STL which makes it possible to replace numeric constants in an STL formula with symbolic variables or parameters. For instance, the formula $\varphi = \square_{[0,\tau]}(\|y(t)\| < s)$ with two parameters τ and s expresses the requirement that during τ seconds, the norm of signal y should be less than s. For example, (2) defines a formula requiring that if the system is not stabilizing (that is, the formula μ_{st} is not satisfied), then it should eventually stabilize, *i.e.*, after at most τ_{tr} seconds, μ_{st} should remain true for at least τ_{st} seconds.

$$\mu_{\text{st}} := \|y(t)\| < s_{\text{st}} \tag{1}$$

$$\varphi_{\text{st}} := \neg\mu_{\text{st}} \Rightarrow \lozenge_{[0,\tau_{\text{tr}}]}\square_{[0,\tau_{\text{st}}]}\mu_{\text{st}} \tag{2}$$

Control policy performance measure and imitation learning quality. In imitation learning, it is essential to have an appropriate measure of performance of control policies, especially when it is unclear what reward function is being optimized [8]. In our framework, we use the relation between the parameters in the PSTL requirements to compare the performance of different controllers. E.g., for the stabilization requirement (2), for a given size s_{st} of the neighborhood around the equilibrium, the smaller the stabilization time τ_{st} is, the faster the controller is. Given a compact set $\mathcal{P}(\Phi)$ parameter valuations for a formula Φ is compact, any controller $\mathcal{C}$ defines a partition of this set into falsified and valid formulas, which are separated by the *Pareto front* [9]; that means no parameter can be improved without compromising the others. We use the relative volumes of the validity and invalidity sets to measure and compare performance of controllers. This notion of policy performance to quantify the difference between the policy of the learner and the expert is necessary to assess the imitation quality.

Finally our state feedback controller imitation learning can be formally stated as follows. Given a plant S, a nominal controller $\mathcal{C}$ such that the closed-loop system $\mathcal{C}||S$ satisfies a PSTL specification Φ, our problem is to learn a neural network controller $\mathcal{N}$ to imitate $\mathcal{C}$ such that the closed-loop system $\mathcal{N}||S$ satisfies Φ, and the performance similarity $\sigma_{S,\Phi}(\mathcal{C}, \mathcal{N})$ is as small as possible.

Learning guidance. The learning guidance here is provided using positive examples, i.e. good behaviors, generated by the nominal controller which already satisfies the desired requirement.

While observing closed-loop behaviors may reduce the number of behaviors to be sampled, we still need to find good training samples that are relevant to an STL property. To do this, we find counter-examples that are closed-loop behaviors violating this property by leveraging the existing falsification tools [10]. A falsification process can also be useful in providing correctness guarantees for the resulting NN. Indeed, if no counter-example is found after a sufficiently large number of scenarios, we consider the NN controller satisfactory and stop. If a counter-example is found, we replay the nominal controller from the counter-example situation in order to obtain new training data, and retrain the neural network. This new data creation is crucial for the efficiency of the process of correcting counter-examples as well as assuring good generalization of the NN. To this end, we propose a dataset aggregation-based learning methodology. This methodology also tailored to provide data representing diverse settings that the NN should learn to cope with, which is captured by a coverage measure using ε-net [11] to quantify how well a finite set of sampled states covers the reachable set. We propose a simple grid-based method to construct ε-nets satisfying a separation requirement.

Finally, we demonstrate our approach on a robotic case study where a NN controller is designed to imitate a model-predictive controller.

3. Temporal Logic Specification Learning

The success of learning in AI has also impacted the field of formal modelling and specification. We focus here on supervised learning, and use parametric temporal logics (PSTL) [12] to represent the hypothesis class, and observations given as time-series labelled by experts. Note that this work has also been extended to Parametric Timed Regular Expressions [13] as specification formalism [14]. From the labelled data we compute the hypothesis that is most suitable in expressing the relation between the observations and the labelling. Given a PSTL formula, the structure of the formula is known but the parameters are not. The process of finding the parameter values for which the resulting STL formula is satisfied over all the observations is called parametric identification. Instead of strict parametric identification, we compute the parameter values for which the formula approximately matches the observations within the user defined bounds on the false positive and false negative error rates. To define such quantities, we show that one can use neither counts of time points or of intervals nor the Lebesgue measure since, and hence we adapt the notion of ϵ-separated set from information theory [11] to propose a new measure with suitable properties, called ϵ-count, to reflect how much a Boolean signal is true.

Parametric Pattern Predictors (PPP) can be defined as specifications, for a given value of parameters, take an observation and produce a Boolean signal which is true where a pattern is predicted an false elsewhere [15]. Increasing Parametric Pattern Predictors (IPPP) are the class where increasing the parameter values augments the set of time points where the pattern is predicted. We show how finding pattern predictors is linked to the problem of exploring Pareto optimal sets using queries, and propose an algorithm that approximately computes the intersection set contained between two Pareto optimal sets of opposite polarities. The crucial idea behind the algorithm is binary search adapted to continuous intervals and multiple dimensions. We demonstrate the approach with analysis of labelled electrocardiograms (ECG).

Acknowledgments. The results discussed in this extended abstract were obtained within the collaboration with Eugene Asarin, Nicolas Basset, Alexandre Donzé, Inzemamul Haque, Nikolaos Kekatos, Akshay Mambakam, José Ignacio Requeno Jarabo, and Indranil Saha. This work is partially supported by the Indo-French Collaborative research project FOVERAS funded by IFCPAR/CEFIPRA, joint French-Japanese ANR-JST project CyPhAI, the ANR project Maveriq, and the Auvergne-Rhône-Alpes Region Project DetAI.

References

[1] O. Maler, D. Nickovic, Monitoring temporal properties of continuous signals, in: FORMATS/FTRTFT, volume 3253 of *Lecture Notes in Computer Science*, Springer, 2004, pp. 152–166.

[2] E. Bartocci, J. Deshmukh, A. Donzé, G. Fainekos, O. Maler, D. Ničković, S. Sankaranarayanan, Specification-based monitoring of cyber-physical systems: a survey on theory, tools and applications, in: Lectures on Runtime Verification, 2018, pp. 135–175.

[3] S. Schaal, Learning from demonstration, in: Advances in Neural Information Processing Systems, 1996.

[4] M. T. Hagan, H. B. Demuth, O. D. Jesús, An introduction to the use of neural networks in control systems, International Journal of Robust and Nonlinear Control: IFAC-Affiliated Journal 12 (2002) 959–985.

[5] C. Nicol, C. J. B. Macnab, A. Ramirez-Serrano, Robust neural network control of a quadrotor helicopter, in: Canadian Conference on Electrical and Computer Engineering, 2008, pp. 1233–1238.

[6] C. E. Garcia, D. M. Prett, M. Morari, Model predictive control: Theory and practice—a survey, Automatica 25 (1989) 335–348.

[7] P. Varshney, G. Nagar, I. Saha, Deepcontrol: Energy-efficient control of a quadrotor using a deep neural network, in: IEEE/RSJ International Conference on Intelligent Robots and Systems, 2019, pp. 43–50.

[8] T. Osa, J. Pajarinen, G. Neumann, J. A. Bagnell, P. Abbeel, J. Peters, An algorithmic perspective on imitation learning, CoRR abs/1811.06711 (2018).

[9] V. Pareto, Manuel d'é conomie politique, Bull. Amer. Math. Soc 18 (1912) 462–474.

[10] A. Donzé, Breach, A toolbox for verification and parameter synthesis of hybrid systems, in: Computer Aided Verification, 22nd International Conference, Springer, 2010, pp. 167–170.

[11] A. N. Kolmogorov, V. M. Tikhomirov, ε-entropy and ε-capacity of sets in function spaces, Uspekhi Matematicheskikh Nauk 14 (1959) 3–86.

[12] E. Asarin, A. Donzé, O. Maler, D. Nickovic, Parametric identification of temporal properties, in: RV, volume 7186 of *Lecture Notes in Computer Science*, Springer, 2011, pp. 147–160.

[13] E. Asarin, P. Caspi, O. Maler, Timed regular expressions, J. ACM 49 (2002) 172–206.

[14] A. Mambakam, E. Asarin, N. Basset, T. Dang, Pattern matching and parameter identification for parametric timed regular expressions, in: Proceedings of the 26th ACM International Conference on Hybrid Systems: Computation and Control, HSCC 2023, San Antonio, TX, USA, May 9-12, 2023, ACM, 2023, pp. 14:1–14:13.

[15] N. Basset, T. Dang, A. Mambakam, J. R. Jarabo, Learning specifications for labelled patterns, in: N. Bertrand, N. Jansen (Eds.), Formal Modeling and Analysis of Timed Systems - 18th International Conference, FORMATS 2020, Vienna, Austria, September 1-3, 2020, Proceedings, volume 12288 of *Lecture Notes in Computer Science*, Springer, 2020, pp. 76–93.