Kalpa Publications in Computing
Volume 16, 2023, Pages 29–34
Proceedings of the 6th Workshop on Formal Methods for ML-Enabled Autonomous Systems

Prediction and Control of Stochastic Agents Using Formal Methods

Faculty of Engineering, Bar-Ilan University, Ramat Gan, Israel
{ravivav1, hillelk}@biu.ac.il

Abstract

In this paper, we propose an innovative approach that incorporates formal verification methods into the training process of stochastic Reinforcement Learning (RL) agents. Our method allows for the analysis and improvement of the learning process of these agents. Specifically, we demonstrate the capability to evaluate RL policies (prediction) and optimize them (control) using different model checkers. The integration of formal verification tools with stochastic RL agents strengthens the applicability and potential of our approach, paving the way for more robust and reliable learning systems.

1 Introduction

Reinforcement Learning (RL) [1] is a powerful paradigm in the field of artificial intelligence, enabling agents to learn optimal actions through trial and error while interacting with an environment. RL has demonstrated success in solving complex tasks, such as game playing [2], robotics [3], and autonomous systems [4]. Within the RL framework, prediction refers to the task of estimating the expected value or future outcomes given a particular state or action, whereas control involves determining the optimal actions or policy that maximize the expected cumulative reward. Despite recent successes, challenges persist, including instability in the training process, difficulty in achieving convergence, and the lack of formal guarantees for learned policies, particularly in the context of prediction and control tasks.

Imitation learning [5] is an approach within the realm of RL, where agents learn by observing and imitating the behavior of an expert. By leveraging the knowledge and experience of the expert, imitation learning can potentially lead to faster convergence and more reliable policies. However, obtaining a suitable expert can be difficult or expensive, and even with expert guidance, it is still challenging to prove convergence and correctness of the learned policies.

Previous work [6] presented a method to combine formal verification tools with deterministic RL agents to address these issues. By translating RL models into transition systems, one could apply formal verification techniques, and use the output of model checkers as expert trajectories to guide the learning process. This allows us to improve the learning process, demonstrate the convergence of RL models, and explore new trajectories that can aid the agent in overcoming challenges such as getting stuck in non-terminal states.

In this work, we tackle the challenge of adapting verification techniques, primarily tailored for deterministic systems, to address the distinct challenges associated with stochastic decision-making processes in RL. We illustrate how non-probabilistic model checking techniques can be effectively employed to enhance the learning process (control), while incorporating a probabilistic model checker for quantitative evaluation of the agent's behavior (prediction). This synergistic approach enables the comprehensive assessment and refinement of stochastic RL policies.

We acknowledge that there may be additional computational overhead in utilizing model checkers for verification and expert trajectory generation. However, we emphasize that our approach aims to strike a balance between computational complexity and the benefits gained from formal verification.

Our main contributions can be summarized as follows:

- Broadening the scope of the previous approach to encompass stochastic RL agents, showcasing the versatility of formal verification tools in handling both control and prediction tasks.
- Adeptly integrating probabilistic techniques with probabilistic model checkers, creating a holistic framework for the analysis and improvement of stochastic RL agents.
- Fortifying the bridge between Reinforcement Learning and Formal Verification, paving the way for the development of robust and dependable learning systems backed by formal guarantees.

2 Method

Verification Challenges with Stochastic RL Agents

One of the primary challenges in applying formal verification techniques to stochastic RL agents is that traditional nondeterministic model checkers, such as SMV-like verifiers [7, 8], cannot effectively represent the stochastic nature of these agents. These model checkers rely on nondeterministic state transition diagrams or automata to represent the system's behavior. However, in stochastic RL, the agent's behavior is inherently probabilistic, making it difficult to represent it as a non-stochastic finite state machine. Additionally, traditional model checkers do not have built-in support for probabilistic reasoning, further limiting their applicability in this context.

As a result, we cannot rely on them for our stochastic RL agents' formal verification needs. Instead, we turn to probabilistic model checkers such as PRISM [9], which are specifically designed to handle the stochastic behavior of systems. To address the challenges posed by the stochastic nature of RL agents, we propose a hybrid approach that combines both non-probabilistic and probabilistic verification techniques. Specifically, we utilize non-probabilistic verification techniques for control tasks, and probabilistic verifiers for prediction tasks, allowing us to leverage their support for temporal logic model checking and quantitative rewards. This combination enables us to perform fine-grained analysis of the agent's behavior while maintaining scalability and efficiency.

Hybrid Approach for Enhanced Control and Prediction of Stochastic RL Agents

At a high level, our approach consists of the following steps—Training, Evaluation and Improvement: First, we train a stochastic RL agent using standard RL methods. Next, we evaluate the probability of the agent reaching the goal using PRISM, a probabilistic model checker. To improve the agent's performance, after x epochs, we create a deterministic version of the policy by taking the greedy action at each state, using the learned Q-table. We then use non-probabilistic model checking to verify the policy's correctness. If the policy hasn't converged to the optimal policy, we take the counter example generated by the model checker, representing a trajectory violating the desired specification, and use it as an expert trajectory to guide the agent's learning process. This counterexample serves as invaluable expert data, providing specific insights into the agent's erroneous behavior and guiding the learning process towards improvement. While it is true that providing counterexamples from a model checker is a common capability found in many mature model checking tools, the significance lies in the integration of these counterexamples as expert trajectories to enhance the training of our stochastic RL agent. This process is repeated for a fixed number of epochs, or until the probability of reaching the goal exceeds a predefined threshold, allowing us to iteratively refine the agent's policy until we achieve satisfactory performance.

The main advantage of our hybrid approach is that it allows us to leverage the strengths of both non-probabilistic and probabilistic verification techniques. By using a non-probabilistic model checker, we can generate valuable expert trajectories to guide the agent's learning process, even in the presence of stochastic behavior. Meanwhile, by using a probabilistic model checker, we can evaluate the agent's expected behavior and make informed decisions about when to continue training and when to refine the agent's policy.

Formulation of the Hybrid Process

To formalize the entire process, let us consider an RL problem defined by a Markov Decision Process (MDP), which consists of a tuple $\{\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma\}$. Here, $\mathcal{S}$ represents the state space, $\mathcal{A}$ denotes the action space, $\mathcal{P}$ defines the transition probabilities, $\mathcal{R}$ represents the reward function, and γ is the discount factor.

During the training phase, we obtain a tabular solution or policy, which can be expressed as a transition system:

$$T = \langle \mathcal{V}, \theta, \rho \rangle \tag{1}$$

Here, $\mathcal{V}$ represents the system variables, θ represents the initial conditions, and ρ represents the transition relation. The variables $\mathcal{V}$ encode the state space $\mathcal{S}$, while ρ captures the actions taken from each state based on the current policy $\pi^{\mathcal{Q}}$. Additionally, θ includes the initial position(s) of the system.

Once we have learned a model in the form of a transition system, we can utilize it as a finite state machine, enabling us to leverage model checking techniques for verification. The query typically involves LTL/CTL properties, with the primary question being, "Does the current model reach the goal from all optional initial states?" In terms of LTL, the query is "Is Finally the goal state always reachable?"

Once we obtain the output from the model checker, we utilize it to enhance the model, incorporating the insights gained from the analysis. Subsequently, we evaluate the quality of the enhanced model using a probabilistic model checker. Specifically, we pose the question, "Given the learned model formulated as a finite state machine (FSM), what is the probability of reaching the goal, starting from initial states?". We express this probability using PCTL as "$P = ?$ [F state=Goal]". This quantitative assessment allows us to measure the likelihood of successfully achieving the specified goal, providing valuable insights into the effectiveness of the learned model and guiding further refinements in the reinforcement learning process.

Results

Frozen Lake is a widely used benchmark in reinforcement learning, consisting of a grid world environment with multiple obstacles and a frozen lake, where an agent must navigate to reach a goal state while avoiding holes and frozen squares. We ran vanilla Q-learning on a stochastic agent, and found that the resulting policy had a low probability of reaching the goal. However, when we incorporated trajectories from the model checker, the final probability increased significantly. We tested the algorithm using different hyperparameter configurations, and found that using the expert's trajectories consistently led to higher probabilities of success. In Figure 1, we show results for the Frozen Lake game with two different grid sizes (10×10 and 20×20) and various stochastic rates, upon which we ran Q-learning using our approach. Without the verification mechanism, all training sessions resulted in an almost zero probability of reaching the goal state (vanilla agents without the verification mechanism achieved very low probabilities of success in both grid sizes, and hence were not included in the graph). However, when incorporating the verification mechanism, we observed significantly improved success rates, with the agent reaching high probabilities of success where the stochastic rates are lower. Notably, the agent's probability of success remained sufficient even with aggressive stochastic rates, compared to the vanilla agent.

Particularly, we were able to quantitatively compare the performance of the agent both with and without the expert trajectories, thanks to the use of the probabilistic model checker, PRISM. This allows us to demonstrate the efficacy of our approach and the improvement gained from incorporating expert guidance.

3 Related Work

The field of artificial intelligence has a long history of research in safe reinforcement learning, with many different approaches proposed. While most methods do not use formal verification, recent works have started to apply formal methods to reinforcement learning. One example is the Justified Speculative Control (JSC) [10] technique, which combines formal verification and run-time monitoring to ensure system safety. Another approach, called Verily [11], verifies deep reinforcement learning systems by converting them into satisfiability problems and solving them using a SAT solver. Meanwhile, other works [12,13] focus on synthesizing deep neural network controllers for nonlinear systems subject to safety constraints, or on training and verifying DNNs with finite input states [14]. Finally, counter-example driven techniques like the one proposed by Zhu et al. [15] have also been used to improve the safety of reinforcement learning algorithms. Our approach differs from these works in two key aspects. First, we generate counterexamples using formal methods in a different way. Second, we combine different model checkers for both prediction and control, allowing us to ensure safety during both the training and deployment phases.

4 Summary

In this paper we discuss an extension of previous work [6] on incorporating formal verification methods into deterministic RL agents to now include stochastic RL agents. We propose a hybrid approach that combines both non-probabilistic and probabilistic verification techniques to create a comprehensive framework for the analysis and improvement of stochastic RL policies. We demonstrate our approach on the Frozen Lake benchmark in RL and show that incorporating

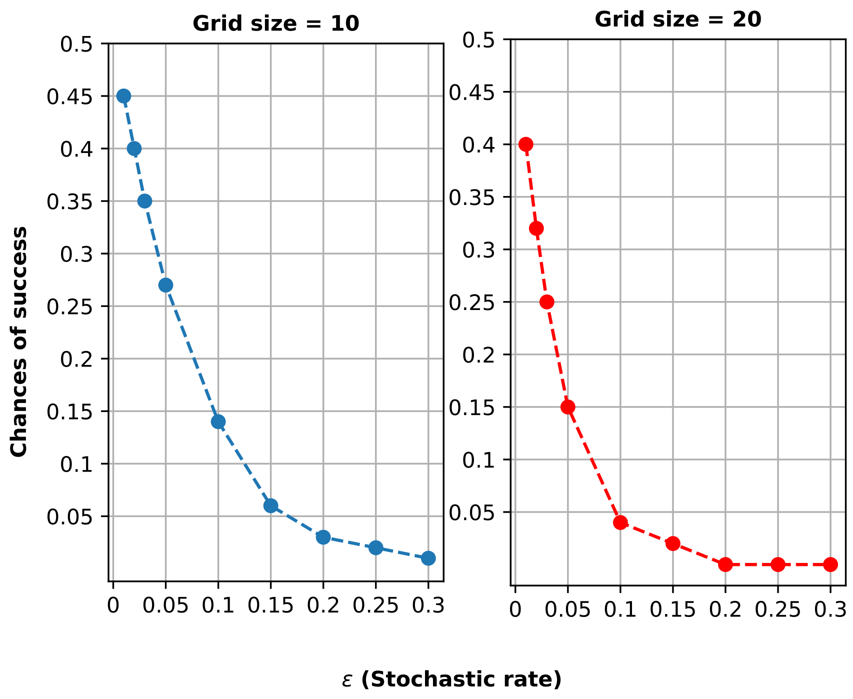

Figure 1: **Performance of Frozen Lake agent with verification mechanism.** For two different grid sizes, we evaluated the performance of our mechanism on the Frozen Lake game, showing the probability of success achieved by the agent under various stochastic rates.

expert trajectories from the model checker leads to higher probabilities of success for the agent. This work is still in its initial stages and further evaluation and progress is needed.

Acknowledgment

This work was supported by the Horizon 2020 research and innovation programme for the Bio4Comp project under grant agreement number 732482 and by the ISRAEL SCIENCE FOUNDATION (Grant No. 190/19) and by Bar-Ilan University's DATA SCIENCE INSTITUTE.

References

[1] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction.* MIT press, 2018.

[2] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

[3] Jens Kober, J Andrew Bagnell, and Jan Peters. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32(11):1238–1274, 2013.

[4] B Ravi Kiran, Ibrahim Sobh, Victor Talpaert, Patrick Mannion, Ahmad A Al Sallab, Senthil Yogamani, and Patrick Pérez. Deep reinforcement learning for autonomous driving: A survey. *IEEE Transactions on Intelligent Transportation Systems*, 2021.

[5] Ahmed Hussein, Mohamed Medhat Gaber, Eyad Elyan, and Chrisina Jayne. Imitation learning: A survey of learning methods. *ACM Computing Surveys (CSUR)*, 50(2):1–35, 2017.

[6] Avraham Raviv, Eliya Bronshtein, Or Reginiano, Michelle Aluf-Medina, and Hillel Kugler. Learning through imitation by using formal verification. In *International Conference on Current Trends in Theory and Practice of Computer Science*, pages 342–355. Springer, 2023.

[7] Alessandro Cimatti, Edmund Clarke, Fausto Giunchiglia, and Marco Roveri. Nusmv: a new symbolic model checker. *International Journal on Software Tools for Technology Transfer*, 2(4):410–425, 2000.

[8] Roberto Cavada, Alessandro Cimatti, Michele Dorigatti, Alberto Griggio, Alessandro Mariotti, Andrea Micheli, Sergio Mover, Marco Roveri, and Stefano Tonetta. The nuxmv symbolic model checker. In *International Conference on Computer Aided Verification*, pages 334–342. Springer, 2014.

[9] M. Kwiatkowska, G. Norman, and D. Parker. PRISM 4.0: Verification of probabilistic real-time systems. In G. Gopalakrishnan and S. Qadeer, editors, *Proc. 23rd International Conference on Computer Aided Verification (CAV'11)*, volume 6806 of *LNCS*, pages 585–591. Springer, 2011.

[10] Nathan Fulton and André Platzer. Safe reinforcement learning via formal methods: Toward safe control through proof and learning. *AAAI*, 2018.

[11] Yafim Kazak, Clark Barrett, Guy Katz, and Michael Schapira. Verifying deep-rl-driven systems. In *Proceedings of the 2019 Workshop on Network Meets AI & ML*, NetAI'19, page 83–89, New York, NY, USA, 2019. Association for Computing Machinery.

[12] Zhengfeng Yang, Yidan Zhang, Wang Lin, Xia Zeng, Xiaochao Tang, Zhenbing Zeng, and Zhiming Liu. An iterative scheme of safe reinforcement learning for nonlinear systems via barrier certificate generation. In Alexandra Silva and K. Rustan M. Leino, editors, *Computer Aided Verification*, pages 467–490, Cham, 2021. Springer International Publishing.

[13] Peng Jin, Min Zhang, Jianwen Li, Li Han, and Xuejun Wen. Learning on abstract domains: A new approach for verifiable guarantee in reinforcement learning. *CoRR*, abs/2106.06931, 2021.

[14] Guy Katz, Clark W. Barrett, David L. Dill, Kyle D. Julian, and Mykel J. Kochenderfer. Reluplex: a calculus for reasoning about deep neural networks. *Formal Methods in System Design*, 60:87 – 116, 2021.

[15] He Zhu and Stephen Magill. Systems support for hardware anti-rop. Technical report, Galois, Inc., 2017. Available at `https://galois.com/reports/formal-methods-for-reinforcement-learning/`.