Shedding Light into the Black Box of Reinforcement Learning

Raphael Engelhardt*, Moritz Lange†, Laurenz Wiskott† and Wolfgang Konen*
*Cologne Institute of Computer Science, TH Köln
Email: {Raphael.Engelhardt,Wolfgang.Konen}@th-koeln.de
†Institute for Neural Computation, Ruhr-University Bochum
Email: {Moritz.Lange,Laurenz.Wiskott}@ini.rub.de

***Index Terms*—Rule learning, Reinforcement learning, Decision tree learning, Explainable AI**

I. Introduction

Reinforcement learning (RL) and deep learning (DL) have achieved remarkable, often super-human performance and dominate in many areas of machine learning. While machine learning algorithms show ever-improving results, even outperforming humans, they fall short when it comes to transparent decision making. Understanding the "reasoning" behind the decision of an RL agent might be considered optional in the environment of a game, but it becomes critical when RL agents act in the real world, e.g. on health- or safety-related tasks, where possibly drastic consequences and questions regarding liabilities might occur.

In this paper we present a new approach to shed some light into the black box of the decision making process of a trained RL agent by translating its behavior into intelligible rules.

Our approach is based on a rather simple idea:

1) First a black box RL agent is trained on a specific environment until satisfying returns are achieved with acceptable consistency.
2) In the next step the trained agent (which we call *oracle*) is evaluated for a number of episodes while at each timestep the observation and the action taken are logged. (The return of the respective episode is appended to each entry afterwards, allowing for the optional filtering of episodes of certain quality.)
3) In the third step a decision tree (DT) is induced from the samples formed in the previous step.

The resulting tree is evaluated by applying it as a decision maker in a number of evaluation episodes and comparing average and standard deviation of the episodes' returns with the ones achieved by the oracle.

Using this approach we aim to answer the question of whether a set of simple and intelligible rules can be deduced from an RL oracle that approximate the oracle's performance. In particular we investigate practical aspects such as the suitability of different types of decision trees or the amount of samples needed for satisfactory results.

II. Methods

We use different algorithms in our approach:

- As RL agents we use the DQN and PPO algorithms as implemented by Stable Baselines3 [1] since they present state-of-the-art algorithms in RL and DL
- Some simple problems can be solved by a fixed deterministic policy (handcrafted rules). These handcrafted rules (HC) may be used as a surrogate for an oracle and conveniently also serve as a benchmark for our core idea: If the extracted rules are close to the handcrafted ones, we know that our rule-extraction process works well.
- For the induction of decision trees we rely on CART [2], as implemented in Scikit-learn [3], and oblique decision trees (ODT), as described and implemented in [4].

We test our approach on three different environments implementing classic control problems with discrete action spaces:

- In `MountainCar-v0` (MC), provided by OpenAI Gym [5], a car initially positioned in a valley is supposed to reach a flag positioned on top of the mountain to the right as fast as possible. As the force of the car's motor is insufficient to simply drive up the slope, it needs to build up momentum by swinging back and forth in the valley.
- The `CartPole-v0` (CP) environment [5] consists of a pole balancing upright on top of a cart. By moving left or right, the cart should balance the pole in the upright position for as long as possible, while maintaining the limits of the one-dimensional track the cart moves on.
- CartPole Swing-Up (CPSU) [6] is conceptually similar to CartPole, with the additional difficulty that the pole starts in a downward position and has to be swung up by back-and-forth movements of the cart before being balanced upright.

III. Results

All three problems can be successfully solved with suitable RL oracles. On problems MC and CP, the trees from our approach reach rewards similar to the oracles' rewards (see Table I). This is an interesting result, because the trees translate the RL policy into a better understandable set of rules which are quite simple. We use a maximum tree depth of 4 for all MC and CP experiments to limit the complexity of the induced rules.

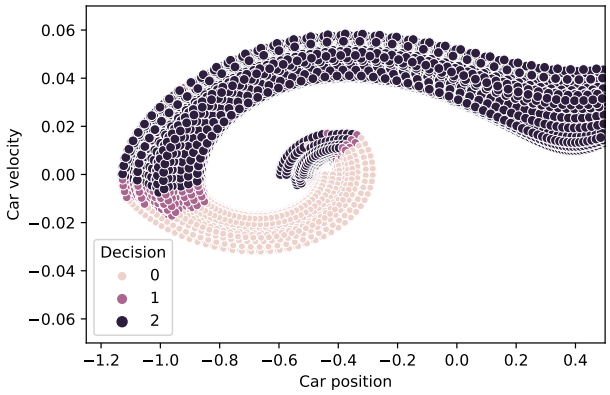

Fig. 1. Trajectories of the MountainCar DQN agent in the observation space tagged with the agent's actions (0: left, 1: none, 2: right)

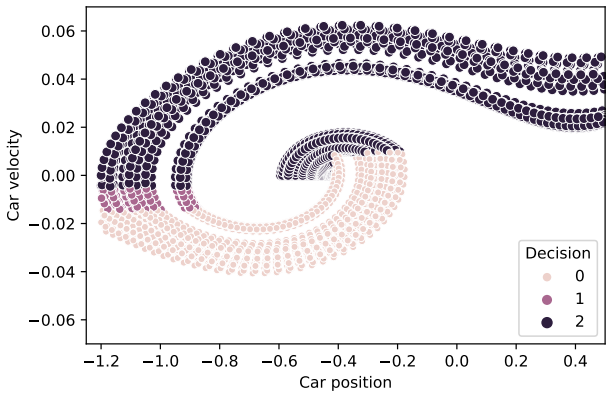

Fig. 2. Trajectories of the MountainCar CART tree agent induced from DQN

It should be noted that these problems, although not complex, are far from being trivial to learn: It was for example not possible to solve the MC problem with PPO. Fig. 1 and 2 show trajectory pictures for the MountainCar problem. Although CART can only induce vertical or horizontal boundaries in the state space, it can approximate the oracle's trajectories and episode returns very well.

The trees resulting from our approach have better average rewards than the rule sets of the programmatic approach proposed by Verma et al. [7] (lines *[Verma]* in Table I). Moreover, we could not reproduce their results: When we applied their precise rules given in [7] to the environments, we got the results in lines *[Verma]-rule* in Table I, which are lower than *[Verma]*.

The CPSU problem currently poses a challenge to our tree-based approach. Although we are able to learn a high-quality oracle with PPO, it is not possible to induce a *simple* tree from it: If we constrain the tree to maximum depth 4, the corresponding agent is not successful in bringing the pole in an upright position. Only with depth 10 can we reach an upright balanced pole and an average reward similar to the oracle (Table I). However, such a tree is too big to be interpretable.

TABLE I

RESULTS ON VARIOUS ENVIRONMENTS. *CART [DQN]* IS A DEPTH-4 TREE AGENT WITH ALGORITHM CART BASED ON SAMPLES FROM ORACLE DQN. $CART_{10}$ IS A DEPTH-10 TREE. $\langle R \rangle$ IS THE AVERAGE REWARD FROM 100 CONSECUTIVE EPISODES. SHOWN IS MEAN $\pm\sigma$(MEAN) FROM 10 RUNS. $\langle R \rangle^{(optimal)}$: OPTIMAL REWARD. AGENTS WITH REWARDS $\geq \langle R \rangle^{(solved)}$ ARE SAID TO *solve* AN ENVIRONMENT.

Environment	Agent	$\langle R \rangle$	$\langle R \rangle^{(optimal)}$ $[\langle R \rangle^{(solved)}]$
MountainCar	Oracle DQN	-101.9 ± 3.4	≈ -90
	Oracle HC	-108.3 ± 4.4	
	CART [DQN]	-103.5 ± 2.8	
	ODT [DQN]	-105.0 ± 4.2	
	CART [HC]	-107.8 ± 4.4	$[-110]$
	[Verma]	$-143.9,\ -108.1$	
	[Verma]-rule	-162.6 ± 3.8	
CartPole	Oracle PPO	200.0 ± 0.0	200
	Oracle HC	200.0 ± 0.0	
	CART [PPO]	200.0 ± 0.0	
	ODT [PPO]	199.4 ± 2.6	
	CART [HC]	200.0 ± 0.0	[195]
	[Verma]	$143.2,\ 183.2$	
	[Verma]-rule	106.0 ± 16.9	
CPSU	Oracle PPO	895.0 ± 16.0	1000
	CART [PPO]	118.1 ± 25.3	
	$CART_{10}$ [PPO]	655.2 ± 78.6	[800]

IV. CONCLUSION & OUTLOOK

With our current work, we propose a novel method of obtaining intelligible rules by training decision trees based on the actions made by well-performing oracles. Our results show how the system is able to reliably extract a given set of rules (over repeated runs with different seeds) and how it can induce new rules from the behavior of well-trained deep RL agents. In the test cases MC and CP the induced decision trees yield satisfactory results with very limited complexity, while for the CPSU problem a small and explaining set of rules from RL oracles remains to be found.

We note that the environments used so far exhibit discrete action spaces. The decision trees therefore solve a classification problem. The next logical step will be to extend our approach to operate on continuous action spaces by using regression trees.

REFERENCES

[1] A. Raffin, A. Hill, M. Ernestus *et al.*, "Stable Baselines3," 2019, https://github.com/DLR-RM/stable-baselines3.

[2] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification And Regression Trees*. Monterey, CA: Wadsworth & Brooks, 1984.

[3] F. Pedregosa, G. Varoquaux *et al.*, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[4] T. Stepišnik and D. Kocev, "Oblique Predictive Clustering Trees," *arXiv preprint arXiv:2007.13617*, 2020.

[5] G. Brockman, V. Cheung, L. Pettersson *et al.*, "OpenAI Gym," *arXiv preprint arXiv:1606.01540*, 2016.

[6] C. D. Freeman, L. Metz, and D. Ha, "Learning to predict without looking ahead: World models without forward prediction," *arXiv preprint arXiv:1910.13038*, 2019.

[7] A. Verma, V. Murali, R. Singh, P. Kohli, and S. Chaudhuri, "Programmatically interpretable reinforcement learning," *arXiv preprint arXiv:1804.02477*, 2018.