Finding the Relevant Samples for Decision Trees in Reinforcement Learning

Raphael C. 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 trees, Explainable AI**

I. Introduction

The demand for interpretability and transparency in reinforcement learning (RL) grows with its continued increase in performance. We found that decision trees (DT) induced from samples of deep reinforcement learning (DRL) agents are able to solve control problems with remarkably small depths. The method's success is, however, sensitive to the sampling method used for tree induction. In this short abstract we briefly describe two previous methods and one new approach to collect the samples leading to a well-performing DT.

II. Methods

The main idea of our investigations follows a simple structure: Starting from an opaque DRL agent (henceforth called "oracle") trained on a specific environment, samples are collected by querying the oracle for actions a in given points of the observation space o. These samples, consisting of features o and actions a, are fed into a DT algorithm. The induced DT is then used as agent in the same environment for a number of evaluation episodes to assess whether the classification or regression made by the DT based on the training samples translates well into a high return in the RL environment.

A. Episodes

In a previous publication [1] we described how we induced DTs from samples collected during (100) evaluation episodes of the oracle. This lead to remarkably shallow DTs solving a variety of control problems. This approach, however, has downsides: (i) The trajectories in episodes of well-performing oracles generally lie in a narrow corridor. Therefore, the collected samples do not sufficiently cover regions of the observation space, so the DT cannot generalize well. (ii) The nature of some environments leads to an oversampling in certain regions. This in turn incentivizes the DTs to accurately represent those regions which are not necessarily relevant to solve the problem. The inverted pendulum is a simple example for this; a good oracle swings the pendulum up quickly where it stands still in the unstable equilibrium, continuously generating very similar samples in the same region of the observation space until the time limit is reached.

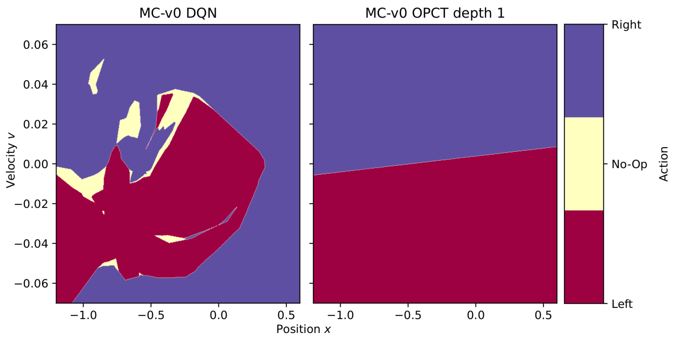

Fig. 1. Decision surfaces for MountainCar of the DQN model (left) and the DT created with the *Iterative Exploration* method (right)

B. Bounding Box

To overcome these downsides and achieve a more uniform coverage of the observation space, another method is tested. Points of the observation space visited in a set number of evaluation episodes of the oracle are logged. The borders of a hyperrectangle (bounding box) are fixed to $[L_i - 1.5r_i, U_i + 1.5r_i]$ with L_i and U_i being the minimum and maximum in each dimension i of the observation space and r_i the interquartile range. The oracle is then asked to predict actions for a set number (3×10^4) of randomly chosen points in the bounding box. Subsequently DTs are trained on this set of observations and actions. While this mitigates problems related to the previous method, it introduces new challenges: Randomly chosen points in the observation space are not guaranteed to be consistent or reachable, given the dynamics of the RL environment. In addition, samples may be added to the training set from regions which the oracle itself has not explored during its training. The training set is therefore polluted with predictions which are not necessarily sensible.

C. Iterative Exploration

To generate samples which on the one hand are possible within the environment's dynamic and on the other hand offer enough coverage beyond the trajectories of well-performing oracles, a third method was investigated.

1) Initially a DT $T^{(0)}$ is trained on 2×10^4 samples produced during evaluation episodes of the oracle. (The initial step corresponds to the *Episodes* approach.)
2) The often quite imperfect $T^{(0)}$ is then evaluated, while logging the visited points $o_{T^{(0)}}$ of the observation space.
3) Subsequently the oracle is queried for actions a corresponding to states $o_{T^{(0)}}$ visited during $T^{(0)}$'s evaluation episodes.
4) From data consisting of $(o_{T^0}|a)$-pairs 1×10^3 randomly drawn samples are added to the training dataset.
5) The next DT $T^{(1)}$ is trained on this enriched dataset and we continue with step 2).

This process is repeated for 10 iterations.

D. Experimental Setup

We evaluate our methods using control problems from OpenAI Gym [2] and CartPole-SwingUp as implemented in [3]. For the training of oracles we rely on DQN [4], PPO [5], and TD3 [6] in their implementation of stable-baselines3 [7]. Although we also tested the widely-used CART [8] (as implemented in [9]) with axis-parallel splits, for the induction of DTs we mostly rely on oblique predictive clustering trees (OPCT) described and implemented in [10]. As the OPCTs exhibit quite large fluctuations with respect to random initialization, wherever we speak of training an OPCT, we actually train 10 OPCTs and pick the best-performing one. The experiments are done for DT-depths from 1 to 10 and repeated 10 times to account for statistical fluctuations.

III. Results

Our experiments show that all problems could be solved by DTs of relatively small depths (≤ 7). The *Iterative Exploration* method of generating samples solves all environments with lower (or at worst equal) depth compared to the *Bounding Box* or *Episodes* methods. Remarkably, in almost all cases (CartPole-SwingUp being the only exception) DTs even surpass the performance of the oracles that contributed the actions to their training data (Table I). This is especially interesting as the DTs are not only simpler (linear inequalities) than the DRL agents, but also need orders of magnitude fewer parameters (Table II). This challenges the often reported trade-off between performance and transparency of AI models. Where possible (low dimensional observation space) visual inspection of the decision surfaces (Figure 1), in fact, suggests that DRL agents perform a needlessly complicated partitioning.

TABLE I
NUMBERS SHOW THE LOWEST DEPTHS AT WHICH THE OPCT OBTAINED BY THE METHOD OF THE RESPECTIVE COLUMN **SURPASSES THE ORACLE'S** PERFORMANCE. 10^+ MEANS THE MAXIMUM INVESTIGATED DEPTH IS NOT SUFFICIENT.

Environment	DRL	Episodes	B. Box	Iterative
`Acrobot-v1`	DQN	3	10^+	**1**
`CartPole-v1`	PPO	6	**1**	**1**
`CartPoleSwingUp-v1`	DQN	10^+	10^+	10^+
`LunarLander-v2`	PPO	10^+	**2**	3
`MountainCar-v0`	DQN	5	6	**4**
`MountainCarCont.-v0`	TD3	**1**	4	**1**
`Pendulum-v1`	TD3	10^+	8	**6**
	Sum	44^+	41^+	$\mathbf{26^+}$

TABLE II
ORACLE AND TREE COMPLEXITY. n DENOTES THE DIMENSION OF THE OBSERVATION SPACE, d THE OPCT'S DEPTH.

			Number of parameters		
Environment	n	DRL	Oracle	Iterative	d
`Acrobot-v1`	6	DQN	136,710	9	1
`CartPole-v1`	4	PPO	9,155	7	1
`LunarLander-v2`	8	PPO	9,797	71	3
`MountainCar-v0`	2	DQN	134,662	61	4
`MountainCarCont.-v0`	2	TD3	732,406	5	1
`Pendulum-v1`	3	TD3	734,806	316	6

IV. Conclusion and Outlook

Our work has shown that for the investigated control problems simple DTs of small depth can provide models as good as and even better than DRL models while being fully transparent and requiring only a fraction of parameters. The success of DT induction relies heavily on the training samples. We have shown that the capability of *Iterative Exploration* to generate samples from the relevant regions of the observation space in the right amount is superior to the other two investigated methods. It should be noted, however, that the DRL oracle is still required to generate successful DTs.

Besides tuning of the parameters of our experiments, an interesting topic is the induction of DTs based on samples generated by the oracle during its training. Initial experiments did not show promising results. This needs to be investigated further to establish whether those samples generally lead to less successful DTs and if so, why oracles apparently explore less fruitful regions during training than the DTs during the iterative process.

References

[1] R. C. Engelhardt, M. Lange, L. Wiskott, and W. Konen, "Sample-Based Rule Extraction for Explainable Reinforcement Learning," in *Machine Learning, Optimization, and Data Science*, G. Nicosia, V. Ojha, E. La Malfa *et al.*, Eds. Springer Nature, 2023, pp. 330–345.

[2] G. Brockman, V. Cheung, L. Pettersson *et al.*, "OpenAI Gym," 2016, https://doi.org/10.48550/arXiv.1606.01540.

[3] A. G. Lovatto, "CartPole Swingup - A simple, continuous-control environment for OpenAI Gym," 2021, https://github.com/0xangelo/gym-cartpole-swingup.

[4] V. Mnih, K. Kavukcuoglu *et al.*, "Playing Atari with deep reinforcement learning," 2013, https://doi.org/10.48550/arXiv.1312.5602.

[5] J. Schulman, F. Wolski, P. Dhariwal *et al.*, "Proximal policy optimization algorithms," 2017, https://doi.org/10.48550/arXiv.1707.06347.

[6] S. Fujimoto, H. van Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *Proc. 35th ICML*, J. Dy and A. Krause, Eds. PMLR, 2018, pp. 1587–1596.

[7] A. Raffin, A. Hill *et al.*, "Stable-baselines3: Reliable reinforcement learning implementations," *JMLR*, vol. 22, no. 268, pp. 1–8, 2021.

[8] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification And Regression Trees*. Routledge, 1984.

[9] F. Pedregosa, G. Varoquaux *et al.*, "Scikit-learn: Machine learning in Python," *JMLR*, vol. 12, pp. 2825–2830, 2011.

[10] T. Stepišnik and D. Kocev, "Oblique predictive clustering trees," *Knowledge-Based Systems*, vol. 227, p. 107228, 2021.