Shedding Light into the Black Box of Reinforcement Learning

Raphael C. Engelhardt [1] Moritz Lange [2] Laurenz Wiskott [2] Wolfgang Konen [1]

[1]Cologne Institute of Computer Science, TH Koln [2]Institute for Neural Computation, Ruhr-University Bochum

Gefördert von
Ministerium für
Kultur und Wissenschaft
des Landes Nordrhein-Westfalen

Motivation

Reinforcement learning (RL) and deep learning (DL) have achieved remarkable, often super-human performance and dominate in many areas of machine learning. While algorithms show ever-improving results, they fall short when it comes to transparent decision making. Our goal is 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.

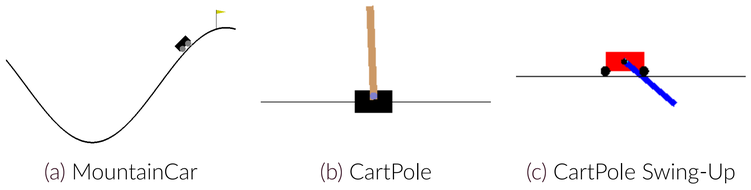
Approach

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**.
3. In the third step **a decision tree (DT) is induced** from the samples collected in the previous step.

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 approximates the oracle's performance.

Methods

- Oracles are obtained by
  - training black box RL agents using DQN, PPO and TD3 algorithms, as implemented by Stable Baselines3 [1]
  - developing simple handcrafted policies. These are used as a surrogate for a black box agent and they serve as a benchmark for the approach: we can compare extracted rules to the handcrafted ones (see Figure 2)
- For the induction of decision trees: CART [2], as implemented in Scikit-learn [3] and oblique decision trees (ODT), as described and implemented in [4]
- Approach is tested on four different environments (Figure 1) implementing classic control problems with discrete and continuous action spaces: `MountainCar-v0` (MC), `MountainCarContinuous-v0` (MCCont), `CartPole-v0` (CP), provided by OpenAI Gym [5] and CartPole Swing-Up (CPSU), adapted from [6] to offer a discrete action space.



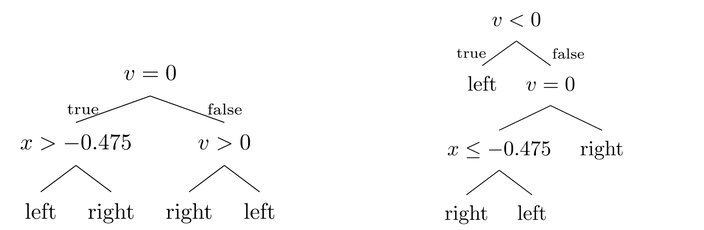
(a) MountainCar (b) CartPole (c) CartPole Swing-Up

Figure 1. Renderings of the environments the approach has been tested on

(a) In MC and MCCont a car initially positioned in a valley is supposed to reach a flag positioned on top of the mountain to the right. 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.
(b) The CP environment 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 it moves on.
(c) CPSU is conceptually similar to CP, 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.

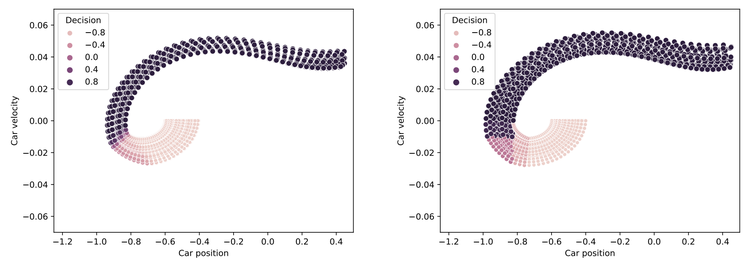
Results

- All four problems can be solved by suitable RL oracles.
- On problems MC, MCCont and CP, the induced trees reach returns similar to the oracles (see Table 1).
- CPSU currently poses a challenge:
  - When constraining the tree to a maximum depth 4, the corresponding agent is not successful in bringing the pole in an upright position.
  - Only with depth 10 the average return is somewhat similar to oracle's (Table 1). However, such a tree is too big to be interpretable.



(a) Handcrafted policy for MC (b) Decision tree induced by CART

Figure 2. Comparison between the oracle's policy (a) and the decision tree as extracted from samples by the CART algorithm (b). Even if the trees are structured differently, the underlying rules are the same under the given precision.



(a) Trajectory of the TD3 oracle ($\overline{R} = 93.85 \pm 0.25$) (b) Trajectory of the induced DT of depth 3 ($\overline{R} = 94.01 \pm 0.27$)

Figure 3. Trajectories of a TD3 black box oracle (a) and of the DT extracted from samples generated by the oracle (b) for MCCont. Both agents exhibit a very similar return during the 100 evaluation episodes.

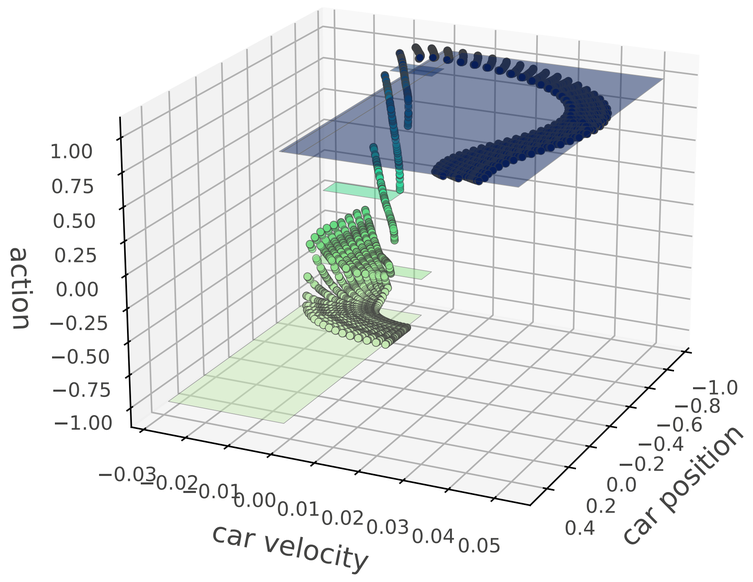


Figure 4. Visualization of samples collected during the evaluation of a TD3 agent on MCCont and the decision surface of the depth 3 DT induced by the CART algorithm

Table 1. 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	
	[Verma]	$-143.9,\ -108.1$	$[-110]$
	[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	
	[Verma]	$143.2,\ 183.2$	[195]
	[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]
MountainCarCont.	Oracle TD3	93.9 ± 0.1	100
	Oracle HC	97.1 ± 0.2	
	CART [TD3]	94.0 ± 0.1	
	CART [HC]	97.2 ± 0.2	[90]

Summary

- All considered environments can be solved by RL agents
- For MC, MCCont and CP the approach works
- The approach produces better results than the ones reported by [7] (cf. table 1 rows *[Verma]*)
- CPSU is a challenge: better returns only with deeper trees

Environment	Black box RL agent	Interpretable DT
MountainCar (disc.)	✓	✓
CartPole	✓	✓
CartPole Swing-Up	✓	✗
MountainCar (cont.)	✓	✓

References

[1] A. Raffin, A. Hill, M. Ernestus, A. Gleave, A. Kanervisto, and N. Dormann. Stable Baselines3, 2019. `https://github.com/DLR-RM/stable-baselines3`.

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

[3] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, et al. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

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

[5] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. OpenAI Gym. *arXiv:1606.01540*, 2016.

[6] D. Freeman, L. Metz, and D. Ha. Learning to predict without looking ahead: World models without forward prediction. 2019. `https://learningtopredict.github.io`.

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