Journal of Machine Learning Research 22 (2021) 1-5 Submitted 1/18; Revised 6/21; Published 6/21

MushroomRL: Simplifying Reinforcement Learning Research

Carlo D'Eramo[1] CARLO@ROBOT-LEARNING.DE
Davide Tateo[1] DAVIDE@ROBOT-LEARNING.DE
Andrea Bonarini[2] ANDREA.BONARINI@POLIMI.IT
Marcello Restelli[2] MARCELLO.RESTELLI@POLIMI.IT
Jan Peters[1] MAIL@JAN-PETERS.NET
[1] *TU Darmstadt IAS, Hochschulstraße 10, 64289 Darmstadt, Germany*
[2] *Politecnico di Milano DEIB, Piazza Leonardo da Vinci 32, 20133 Milano, Italy*

Editor: Cheng Soon Ong

Abstract

MushroomRL is an open-source Python library developed to simplify the process of implementing and running Reinforcement Learning (RL) experiments. Compared to other available libraries, MushroomRL has been created with the purpose of providing a comprehensive and flexible framework to minimize the effort in implementing and testing novel RL methodologies. The architecture of MushroomRL is built in such a way that every component of a typical RL experiment is already provided, and most of the time users can only focus on the implementation of their own algorithms. MushroomRL is accompanied by a benchmarking suite collecting experimental results of state-of-the-art deep RL algorithms, and allowing to benchmark new ones. The result is a library from which RL researchers can significantly benefit in the critical phase of the empirical analysis of their works. MushroomRL stable code, tutorials, and documentation can be found at `https://github.com/MushroomRL/mushroom-rl`.

Keywords: reinforcement learning, python, open-source, benchmarking

1. Introduction

The advantages of Reinforcement Learning (RL) (Sutton and Barto, 1998) methodologies are mostly shown in terms of empirical performance, especially in the recent years after the emergence of deep RL (Mnih et al., 2015). In fact, in the vast majority of research papers, experimental evaluation makes a crucial difference between a successful and well-cited work, and a mostly unknown one. The need of evaluating algorithms comes together with the necessity of implementing and testing them in a quick and reliable way; thus, to address these problems, several RL libraries have been developed and are currently used by researchers. However, these libraries have various drawbacks, e.g. they only focus on benchmarking and are not easy to extend, or they do not have a sufficiently large number of already implemented algorithms. We introduce MushroomRL, an RL Python library developed to create a flexible, easy to understand, and comprehensive RL framework. MushroomRL comes with a strongly modular architecture that makes it easy to understand how each component is structured and how it interacts with other ones; moreover, it provides an exhaustive list of RL and deep RL methodologies that are ready to be used as baselines.

In this paper, we provide an overview of the most famous RL libraries and briefly compare them with MushroomRL; then, we present the main ideas of our library and how they make MushroomRL powerful and unique; finally, we describe the MushroomRL Benchmarking Suite, a framework based on MushroomRL for benchmarking RL algorithms, and providing results of most actor-critic methods on well-known problems, e.g. MuJoCo.

2. Related works

The number of open-source RL libraries has significantly increased with the success of deep RL. *OpenAI Baselines* (Dhariwal et al., 2017) is one of the most famous examples of deep RL libraries. It implements the majority of most recent techniques and allows to test them on well-known RL problems through an interface with the benchmarking framework OpenAI Gym (Brockman et al., 2016). Recognized drawbacks of this library are its complex architecture and the difficulty in understanding the code, thus researchers are discouraged to extend it with novel functionalities and only use it for running available baselines. The recent *Stable Baselines* (Hill et al., 2018) is a fork of OpenAI Baselines made with the purpose of simplifying its architecture. However, despite the simplified interface, extending it with novel methodologies is still not straightforward. A parallel project is *RL Baselines Zoo* (Raffin, 2018), a collection of agents trained with deep RL algorithms implemented in Stable Baselines. RLLib (Liang et al., 2018) is an RL library focused on distributed computation, based on the Ray library (Moritz et al., 2018). Another library is *KerasRL* (Plappert, 2016) which is built on the well-known deep learning library *Keras*. Unfortunately, this library is not maintained anymore. *ChainerRL* is a deep RL library based on the deep learning library *Chainer*. Considering its structure and ideas, ChainerRL can be compared to Keras RL, but it is still well maintained and documented. An example of a flexible RL library is *Tensorforce* (Kuhnle et al., 2017), which is strongly based on Tensorflow. This library implements several deep RL algorithms and its structure allows to easily test them on custom problems, besides the already available ones. Moreover, its modular architecture facilitate the process of extending the library with novel methodologies. Unfortunately, this library lacks of a complete documentation. Older RL libraries and interfaces have been proposed in the past, such as: *RL-Glue* (Tanner and White, 2009), *RLPy* (Geramifard et al., 2015), *RLLab* (Duan et al., 2016). However, these are not supporting most recent deep RL techniques and most of them are abandoned projects.

3. Ideas and content

MushroomRL is developed to address the issues of other RL libraries. The following are the qualities that make MushroomRL a unique, yet powerful tool to carry out RL research.

General purpose The research on RL is not only about deep RL. Shallow RL techniques (e.g. Q-Learning) are still important algorithms to consider, but most of RL libraries ignore them. Since there are no fundamental differences between shallow RL and deep RL, MushroomRL adapts to heterogeneous learning tasks just focusing in modeling the interaction of an *agent* with an *environment*. This is achieved by a common interface which unifies a broad variety of RL techniques, such as: batch and online algorithms, episodic and infinite horizon tasks, on-policy and off-policy learning, shallow and deep RL (see Table 1).

Value-based	Q-Learning, Double Q-Learning, Weighted Q-Learning, Speedy Q-Learning, R-Learning, SARSA, Expected SARSA, True Online SARSA-λ, FQI, LSPI, DQN, Prioritized DQN, Double DQN, Averaged DQN, Categorical DQN, Dueling DQN, Maxmin DQN, Noisy DQN, Rainbow
Policy-search	REINFORCE, GPOMDP, eNAC, RWR, PGPE, REPS
Actor-critic	COPDAC-Q, Stochastic Actor-Critic, A2C, DDPG, TD3, SAC, TRPO, PPO

Table 1: Several algorithms already implemented in MushroomRL.

Lightweight MushroomRL is both user-friendly and flexible: only a high-level interface is exposed to the user, hiding low-level aspects. For instance, the user should not care about the implementation details to use a function regressor for different tasks, since they are hidden by a simple common interface. However, we leave the check of consistency constraints to the user, e.g. avoiding the use of a tabular algorithm for an environment with continuous state space. Minimal interfaces simplify the implementation of new algorithms, as there are no hard constraints in the prototypes.

Compatible Standard Python libraries useful for RL tasks have been adopted:

- *Scientific calculus*: Numpy, Scipy;
- *Basic ML*: Numpy-ml, Scikit-Learn;
- *RL benchmark*: OpenAI Gym, DeepMind Control Suite (Tassa et al., 2018), Pybullet, MuJoCo (Todorov et al., 2012), ROS;
- *Neural networks and GPU computation*: PyTorch, Tensorflow.

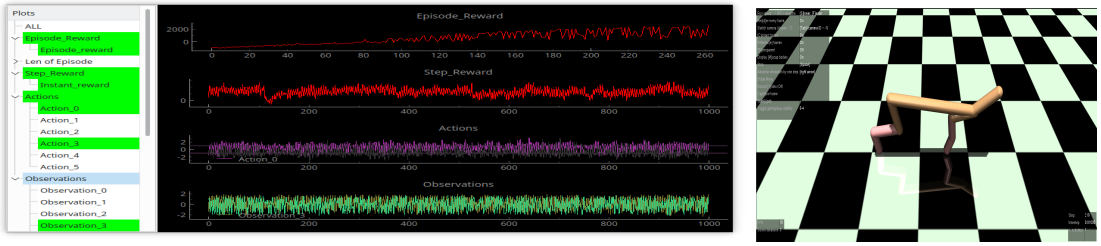
MushroomRL provides an interface to these libraries, in order to integrate their functionalities in the framework, e.g. an interface for Gym environments, support for regression with scikit-learn models and Pytorch neural networks.

Easy to use MushroomRL enables to develop and run experiments writing a minimal amount of code. In most cases, an experiment can be written in a few Python lines without the need of complex configuration files. The majority of the RL problems can be solved with experiments written following the structure of the provided examples.

4. Benchmarking Reinforcement Learning algorithms

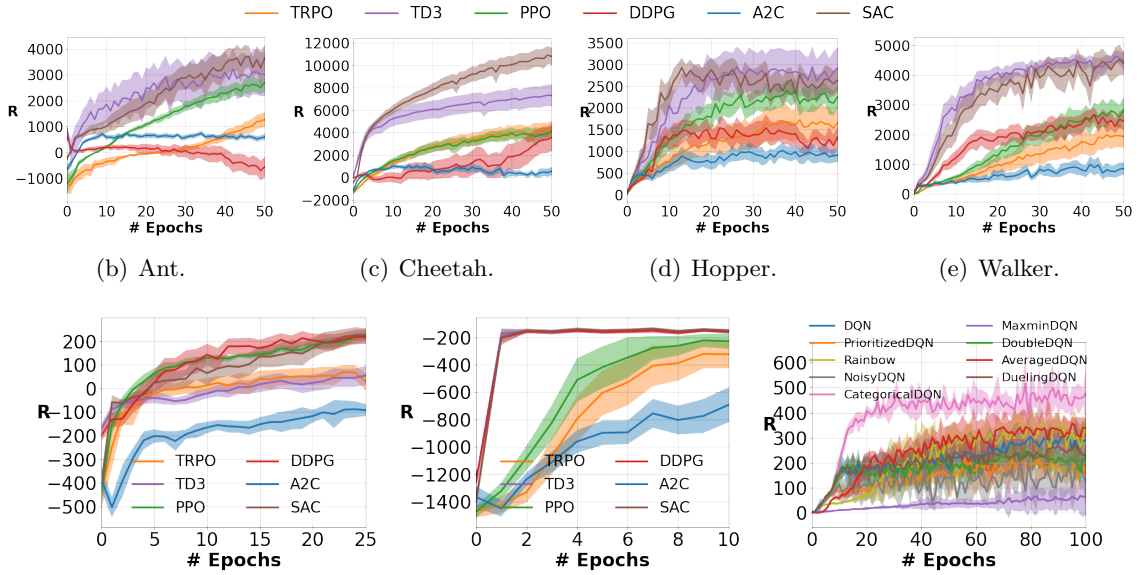
Monitoring the execution of RL experiments, especially in deep RL, is crucial to properly assess the performance of the algorithms and the quality of the learned policies. MushroomRL provides a text logger to print experiment statistics, such as cumulative reward and progress of the experiment, at run time; moreover, MushroomRL allows to show live plots of experiment statistics (Figure 1(a)) and rendering of the environment (Figure 1(b)).

MushroomRL Benchmarking Suite We developed the MushroomRL Benchmarking Suite, a framework based on MushroomRL for running large-scale benchmarking experiments on the already provided algorithms, or new ones implemented by users. Figure 2



(a) Live plots. (b) Rendering.

Figure 1: Monitoring experiments with live plots of statistics and environment rendering.



(b) Ant. (c) Cheetah. (d) Hopper. (e) Walker.

(f) Lunar lander continuous. (g) Pendulum. (h) Breakout.

Figure 2: Empirical performance of deep RL algorithms in continuous control problems and the Breakout Atari game (Bellemare et al., 2013).

shows several empirical results obtained using the MushroomRL Benchmarking Suite. Our results are comparable with the ones in literature assert the quality of the implementation of the algorithms in MushroomRL. Further results and details on the MushroomRL Benchmarking Suite, e.g. the hyper-parameters used in the experiments, can be found at `https://mushroom-rl-benchmark.readthedocs.io/en/latest/index.html`, while the code repository is available at `https://github.com/MushroomRL/mushroom-rl-benchmark`.

5. Conclusion

We presented MushroomRL, an RL library to help researchers to easily develop their works and compare the results w.r.t. classical and deep RL algorithms. Full documentation and tutorials are available at `http://mushroomrl.readthedocs.io/en/latest/`.

References

Marc G Bellemare, Yavar Naddaf, et al. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, 2013.

Greg Brockman, Vicki Cheung, et al. Openai gym, 2016.

Prafulla Dhariwal, Christopher Hesse, et al. Openai baselines. `https://github.com/openai/baselines`, 2017.

Yan Duan, Xi Chen, et al. Benchmarking deep reinforcement learning for continuous control. In *International Conference on Machine Learning*, pages 1329–1338, 2016.

Alborz Geramifard, Christoph Dann, et al. Rlpy: a value-function-based reinforcement learning framework for education and research. *Journal of Machine Learning Research*, 16(1):1573–1578, 2015.

Ashley Hill, Antonin Raffin, et al. Stable baselines. `https://github.com/hill-a/stable-baselines`, 2018.

Alexander Kuhnle, Michael Schaarschmidt, et al. Tensorforce: a tensorflow library for applied reinforcement learning. Web page, 2017. URL `https://github.com/tensorforce/tensorforce`.

Eric Liang, Richard Liaw, et al. Rllib: Abstractions for distributed reinforcement learning. In *International Conference on Machine Learning*, pages 3053–3062. PMLR, 2018.

Volodymyr Mnih, Koray Kavukcuoglu, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

Philipp Moritz, Robert Nishihara, et al. Ray: A distributed framework for emerging ai applications, 2018.

Matthias Plappert. keras-rl. `https://github.com/keras-rl/keras-rl`, 2016.

Antonin Raffin. Rl baselines zoo. `https://github.com/araffin/rl-baselines-zoo`, 2018.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

Brian Tanner and Adam White. Rl-glue: Language-independent software for reinforcement-learning experiments. *Journal of Machine Learning Research*, 10(Sep):2133–2136, 2009.

Yuval Tassa, Yotam Doron, et al. Deepmind control suite. *arXiv preprint arXiv:1801.00690*, 2018.

Emanuel Todorov, Tom Erez, et al. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE, 2012.