Prof. Joschka Bödecker, Gabriel Kalweit, Maria Huegle
Submit until **Friday, February 14 at 11:00am**

REINFORCEMENT LEARNING
Project

0 Preliminaries

Find one or two peers and form a study group. If you do not find a group directly, use the forum. Send your names and group's email addresses at `kalweitg@cs.uni-freiburg.de` (**subject: [RL1920] Project**) until **Friday, January 31 at 11:00am** latest. We will provide git repositories to each group. If you do not know git, go through a tutorial[1]. Please push your slides as a pdf and your code for the project to your assigned repository until **Friday, February 14 at 11:00am**. **Submissions via email will not be accepted**.

1 Problem

The project is based on the *Cart Pole* problem in `continuous_cartpole.py`. This classic control task has a four-dimensional continuous observation space – position of the cart, velocity of the cart, angle of the pole, rotation rate of the pole – as well as a one-dimensional continuous action space in $[-1, 1]$ representing the force which is applied to the cart. The goal is to push and hold the pole in an upright position. We modified the environment, such that it yields sparse rewards (1 if the pole is upright, -1 if the cart is off the track and 0 otherwise). A visualization of the environment can be seen in Figure 1.

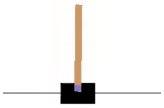

Figure 1: Visualization of the Cart Pole environment.

You can pass your own reward function as a parameter to overwrite the sparse reward. It has to have the form `def reward(cart_pole)`, where `cart_pole` is a `ContinuousCartPoleEnv`-instance, and has to return a scalar. You get the current state of the agent via `cart_pole.state` and its last action via `cart_pole.last_u`. In the control loop, `ContinuousCartPoleEnv` can be used like any Gym environment. You can set the episode length to 500.

[1] `https://rogerdudler.github.io/git-guide/index.html`

2 Implementation

You can choose to implement and apply any reinforcement learning algorithm (from the lecture or beyond) to solve this problem. Explain all your choices and steps. Aim to optimize your results considering the return, learning stability and speed (in terms of episodes/samples).

Since the Cart Pole is a continuous control problem, you have to discretize the actions in order to apply your *DQN* implementation, whereas policy gradient methods, such as *REINFORCE* or *PPO*, naturally fit continuous action spaces. You can think about subtracting a baseline or even implement an actor-critic approach.

You can also develop a shaped reward function, your own exploration strategy or make use of a heuristic (such as hint-to-goal). We also discussed employing a model of the environment or mixing MC and TD.

3 Evaluation

The evaluation should at least include learning curves (i.e. the return over time) of your chosen approach and settings. You can additionally think of your own metric and evaluate that as well.

The initialization of function approximators and exploration lead to randomness, so take into account the performance of multiple runs (e.g. mean and standard deviation or median and interquartile ranges). It is important that your evaluation builds the basis for discussion and scientifically analyzes which are the important aspects and characteristics of your approach – your slides have to highlight your findings in a convincing manner. Please also make sure to carefully keep track of intermediate results.

4 Slides

Submit two slides as a PDF. They should provide answers to the following questions:

- Why did you choose your algorithm? (Slide 1),
- How does your algorithm work? (Slide 1)
- How well did your approach perform and how fast? What was important? (Slide 2)

Please push your slides as a PDF and your code to your assigned git-repository **until Friday, February 14 at 11:00am**. Submissions after that will not be accepted.