Prof. Joschka Bödecker, Gabriel Kalweit

# Reinforcement Learning
Exercise 7

## 0 Lecture

Watch *Lecture 09: Exploration and Exploitation*[1] before the upcoming session on Friday, January 11.

## 1 PPO

**Algorithm 1** Proximal Policy Optimization

```
procedure PPO
    initialize parameters $\theta^{\pi}$ of policy $\pi$ arbitrarily
    initialize parameters $\theta^{V}$ of value function $V$ arbitrarily
    update parameters $\theta^{\pi_{\text{old}}}$ of policy $\pi_{\text{old}}$
    for iteration=1,2,... do
        Run policy $\pi_{\text{old}}$ for $e$ episodes
        Compute advantage estimates $\hat{A}_i$
        for some epochs do
            train $V$
            Optimize surrogate $L^{\text{CLIP}}(\theta^{\pi}) = \hat{\mathbf{E}}_t[\min(\frac{\pi(a_t|s_t)}{\pi_{\text{old}}(a_t|s_t)}\hat{A}_i, \text{clip}(\frac{\pi(a_t|s_t)}{\pi_{\text{old}}(a_t|s_t)}, 1-\epsilon, 1+\epsilon)\hat{A}_i)]$
        update parameters $\theta^{\pi_{\text{old}}}$ of policy $\pi_{\text{old}}$
```

Implement the `ppo` function in `ppo.py` using the *Clipped Surrogate Objective*, where `update_frequency` is the amount of episodes between updates and `epochs` is the number of full iterations over the collected data, which can be implemented using the `next_batch(index, batch_size)` method of the `ReplayBuffer` class. Evaluate again on the modified `MountainCar` environment.

## 2 Experiences

Make a post in thread *Week 08: Advanced Policy Gradient Algorithms* in the forum[2], where you provide a brief summary of your experience with this exercise, the corresponding lecture and the last meeting. We wish you a nice Winter break and a happy new year.

[1] `https://ilias.uni-freiburg.de/goto.php?target=xvid_1121354&client_id=unifreiburg`

[2] `https://ilias.uni-freiburg.de/goto.php?target=frm_1121060&client_id=unifreiburg`