Reinforcement Learning WS 2024
TAs: Pavel Kolev and Mikel Zhobro
Due-date: **10.12.2024, 14:00** (upload to ILIAS as one file)
Filename: homework8-NAME1-NAME2-...zip
Homework Sheet 8
December 2, 2024

1 Importance weighted policy gradient

The policy gradient with importance weighting used, e.g. in PPO is given by: (r is replaced by the advantage)

$$\nabla_\theta J(\theta) = \mathop{\mathrm{E}}_{\tau\sim\pi_{\theta^{\text{old}}}}\left[\frac{\nabla_\theta \pi_\theta\left(s_t, a_t\right)}{\pi_{\theta^{\text{old}}}\left(s_t, a_t\right)} r\right] \tag{1}$$

However, so far we have studied policy gradient formulations containing the score function:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(s, a)\, r\right]$$

Why is there no logarithm in Eq. (1)? Show that Eq. (1) is correct under the assumption that the difference in the state visitation distribution $\mu(s)$ between old and new policy can be ignored. Think of the expectation that needs to be computed, see slide 6 (*Recap: Policy Gradient*) in the lecture notes 8, however, with respect to the old parameters now.

2 Proximal Policy Gradient (PPO)

Preparation

Download the code from ILIAS: `8_gym-PPO.zip`. It is independent from the previous exercises, however, you can use the same pip/virtual environment from the last exercises.

Fill in some gaps in the PPO implementation

In this exercise, you will partially implement PPO with the clipping objective and test its parameters dependence in the LunarLander environment. The code contains the following files:

PPO.py The main code containing all that is needed. (Changes required!)

test.py Script to test saved policies and to render them. (no/little changes required)

Gym-PPO.ipynb notebook to plot results (Changes required!)

results folder where the result files will be stores

Tasks:

(a) complete the code in `PPO.py`. The comments in the code are guiding you the way. See slide 34 in the lecture nodes 8.

(b) Test the algorithm on the `LunarLander-v2` environment with clipping value $\epsilon = 0.2$ (run `PPO.py`). You should get good learning curve until episode 1000 to a reward ~ 100.

(c) Change the code to be able to scan for different values of ϵ (0.01,0.1,0.2,0.5,0.75) and plot the (smoothed) training reward *Hint:*[1]. Train for 2000 episodes each.

(d) which value of ϵ looks promising? Run 3-5 restarts with this value and plot the results (make sure you specify different seeds)

(e) Inspect the policies performance using the `test.py` script. Does the spaceship manage to land properly?

(f) Bonus: try to vary another parameter of your choice.

[1] you can use the command line interface and run a loop in your shell