Reinforcement Learning WS 2024
TAs: Pavel Kolev and Mikel Zobhro
Due-date: **17.12.2024, 14:00** (upload to ILIAS as one file)
Filename: homework9-NAME1-NAME2-...zip
Homework Sheet 9
December 10, 2024

1 Deep Deterministic Policy Gradient (DDPG)

(a) The critic network is updated using the temporal difference (TD) error between the estimated state-action value and the target value. However, the target value is computed by applying the actor's target network to the next state. Explain the intuition behind this approach and how it contributes to the learning process.

(b) DDPG utilizes target networks to stabilize the learning process by providing consistent target values. However, the target networks are updated using a "soft update" mechanism. Explain how this soft update is achieved, and discuss its significance in mitigating the problems associated with hard updates. (Note that in the implementation we use hard updates for simplicity)

(c) One of the challenges in DDPG is the exploration-exploitation trade-off. How can the incorporation of noise in the actor's actions during training help address this trade-off and improve the exploration capabilities of the algorithm?

(d) DDPG can suffer from overestimating the state-action values, especially in the presence of function approximation. Provide an intuitive explanation of how this overestimation occurs.

2 DDPG – Hands On

Preparation

Download the code from ILIAS: `9_gym-DDPG.zip`. It is independent from the previous exercises, however, you can use the same virtual environment from the last exercises. For the second environment you need mujoco. Install it with `pip3 install "gymnasium[mujoco]"`.

Fill in some gaps in the DDPG implementation

In this exercise, you will implement DDPG and test its parameters dependence in the Pendulum and Walker environment. The code contains the following files:

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

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

feedward.py Feedforward Neural Network (MLP). Updated version version earlier exercises. (no changes required)

memory.py Simple replay buffer implementation. (no changes required)

results folder where the result files will be stores

Tasks:

(a) complete the code in `DDPG.py`. The places you have to edit are marked with *TODO*. See slide 19 in the lecture nodes 9.

(b) Test the algorithm on the `Pendulum-v1` environment with default parameters (run `DDPG.py`). The reward should reach around -500 after episode 1000. Train for 2000 episodes and plot the losses and rewards over time (see `Gym-DDPG-plots.ipynb`)

(c) Check the value function (Q-function with the actions of the policy) for the Pendulum by running 100 episodes with noise strength 0.2

(d) Investigate the dependency on the actor learning for update frequency 20 and 100. Try $[0.001, 0.0005, 0.0001, 0.00005]$.

(e) Test the algorithm on the `Halfcheetah-v4` environment. Here you need to train for 6000 episodes (this takes some time, so let it run in the background)

(f) Inspect the policies in the notebook. Does the simulated creature run properly?

(g) Bonus 1: Try `LunarLander-v2`: See that it does not learn to land the space ship. What do you think is the reason? (Or can you make it work?, Disclaimer: we have not found a solution quickly.)

(h) Bonus 2: Implement soft target updates.