Reinforcement Learning WS 2024
TAs: Pierre Schumacher
Due-date: **03.12.2024, 14:00** (upload to ILIAS as one file)
Filename: homework7-NAME1-NAME2-...zip
Homework Sheet 7
November 25, 2024

1 Score Function for Gaussian Policy

For the policy gradient, the likelihood ratios trick is useful to compute the right expectations. The resulting function we need to compute is the score function: $\nabla_\theta \log \pi_\theta(s,a)$. In the lecture we did this for the soft-max policy which is useful for a discrete action space. For continuous action spaces we can use a gaussian policy (here for 1-dimensional a):

$$\pi_\theta(s,a) = p(a \mid s,\, \theta) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(a-\mu(s,\theta))^2}{2\sigma^2}\right), \tag{1}$$

where the mean depends on s using linear combination of features:

$$\mu(s,\theta) = \phi(s)^\top \theta$$

(a) Compute the score function ($\nabla_\theta \log \pi_\theta(s,a)$) for the policy parametrization in Eq. (1).

2 Deep Q-Learning (Q-Networks)

Preparation

Download the code from ILIAS: `7_gym-DQN.zip`. It contains the same files as last time, but adds the main code: `Gym-DQN.ipynb`. Please check the instructions from last time. You can use the same virtual environment from the last exercise (you can also just copy the notebook file into the other folder)
Then you can fire up `jupyter notebook` and you should be ready to go.

Implement DQN

In this exercise, you will implement DQN and and test it on two environments. The code contains the following files:

policy.py (not needed this time)(no changes required)

memory.py Implements a replay buffer with a fixed size that provides a method *add_transition* to add new transitions to the buffer and *sample* that, in our case, returns a tuple of the form $(s_t, a_t, r_{t+1}, s_{t+1}, d_{t+1})$ where s_t is the current and s_{t+1} the next state, a_t is the action and r_{t+1} the reward and d_{t+1} is the done signal (always false for this environment). (no changes required)

feedforward.py Implentation of a feed forward neural network. (No changes required)

Gym-DQN.ipynb starter code for this exercise (Changes required!)

Tasks:

(a) complete the code in the notebook (i.e. implement DQN). I suggest you start with one Q-network

(b) Test the algorithm in the `Pendulum-v1` environment (with the DiscreteActionWrapper).

(c) add the target network and create the following analysis: Show the training curves and (final) test-performance after 600 episodes

- without target network
- with target network updated every batch-training call
- with target network updated every 20 batch-training calls

(d) Visualize the value function, i.e. the maximal q function

(e) Do the same procedure for the `CartPole-v1` environment