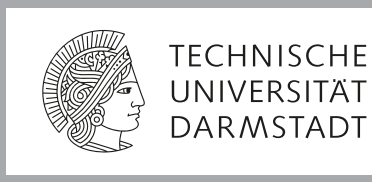
Reinforcement Learning: From Foundations to Deep Approaches



Summer Semester 2024, Homework 3

Prof. Georgia Chalvatzaki, Dr. Davide Tateo

Total points: 103

Deadline: 23:59, August 9th 2024

Submission: Use the provided Python scripts to solve the problems. We provided you with a zip file for the homework project that includes the Python scripts (to complete) and bash scripts (to run experiments). The structure of the project is similar to MushroomRL. Provide a report with all the requested plots and answers mentioning the respective question number. In addition, submit your solution code in the provided Python scripts. For the submission, upload a zip file that contains the Python scripts and the report.

To load the virtual environment needed for this assignment on the cluster, run the following command in a bash terminal in your cluster workspace:

```
conda activate /home/kurse/kurs00077/rl_assign3
```

Each submission is individual!

Late submissions: We will accept late submissions, but you will lose 10% of the total reachable points for every day of delay. Note that even 15-minute delays will be counted as being one day late! Submissions three days later than the designated deadline will not be considered. If you are not able to make the deadline, e.g.due to medical reasons, you need to contact us before the deadline. We might waive the late penalty in such a case.

3.1 Implementation [30 Points]

For this project, you will be part of the team responsible for giving feedback to the athletes of your country who are participating in the Paris Olympic Games this summer. In order to improve the athlete's gait for the 100-meter hurdles race, you have the idea of leveraging the knowledge you learned during the Reinforcement Learning (RL) lecture. You will train an RL agent, Hopper, which learns how to run while having to jump over two hurdles (see Figure 1). To solve this reinforcement learning challenge, we will use three algorithms: Proximal Policy Optimization (PPO) [1], Twin Delayed Deep Deterministic Policy Gradient (TD3) [2], and Soft Actor-Critic (SAC) [3].

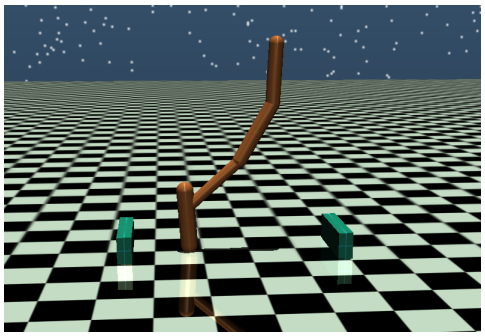


Fig. 1: Hopper agent in the simulated Olympic hurdles environment with Hopper (brown) and hurdles (green).

For this question, you will implement the environment interface and set up the learning agents and networks. The code in the Python files requiring your attention will be annotated with '# TODO'. Complete these to implement the training process using the specified reinforcement learning algorithms. You ask you to only insert lines of code between the comments '# [START YOUR CODE HERE]' and '# [END YOUR CODE HERE]'.

1. **The mdp can be found in *mushroom_rl_extensions/environments/mujoco_envs/data/hopper_fixed_hurdles.xml*. You must complete the gaps in *mushroom_rl_extensions/environments/mujoco_envs/hopper_hurdles.py* to complete the mdp interface. Details on the implementation are provided in the python file. [5 Points]**

2. **The algorithms you will train are constructed in *setup_algorithms.py*. Fill in the gaps to construct the networks in each algorithm appropriately. [10 Points]**

3. **The networks you will use for the agents are described in *networks.py*. Fill in the gaps in this file to construct the networks according to the architectures described. [10 Points]**

4. **The scripts for on-policy and off-policy training are detailed in the *run_single_experiment.py* Fill in the gaps in this file to enable the on-policy and off-policy training scripts to function correctly. [5 Points]**

3.2 Tuning [45 Points]

Hyperparameter tuning is a critical aspect of solving high-dimensional problems in deep reinforcement learning systems. For this exercise, you will adjust specific algorithmic hyperparameters. Ensure that the hyperparameters stay within the specified ranges during each experiment. To speed up the hyperparameter tuning, we allow you to tune the hyperparameters for only one seed. However, note that in practice, multiple seeds are required to have significant results.

We have prepared a simple tool for tuning the hyperparameters. All hyperparameters are stored in YAML files under the *configs* directory. Adjust the hyperparameters to optimize the agent's performance. The results will be recorded in the folder under the *out/{hash}*, where *hash* is an 8-character hash code generated from the corresponding parameters to ensure uniqueness. The purpose of this assignment is to provide a more intuitive understanding of the various impacts of different hyperparameters in the algorithms.

For debugging purposes, you can execute the command `python watch_agent.py –ckpt {path to best-agent.msh}` on your local machine to visualize and assess the agent's performance.

For visualization, you can execute the command `python logger_visualizer.py -p {logger directory}` to generate the plots. This will allow you to monitor the changes in R, J, V, and H.

To download any file from the cluster to your local machine, you can execute the following command line in you local machine `scp -r remote_username@IP-address:/remote/directory /local/directory`.

Store the best agents of SAC, TD3, and PPO algorithms in a folder named "best agents". This folder should include the *logger.log* file, the *config.yml* file and the *agent-best.msh* file, with each subfolder named according to the algorithm. For example, the files for the SAC algorithm should be stored in *best_agents/sac*.

Generate and add the plots for R, J, V, and H for the best-performing agent of each algorithm in the report. You can achieve the optimal solution by using different seeds and combinations of parameters. It is best not to alter other hyperparameters; however, you are entirely capable of conducting a comprehensive analysis of each hyperparameter, as the config file allows for this. Nonetheless, we recommend starting with the range we suggest. We provide a logger visualizer to export the desired plots in *logger_visualizer.py*.

1. **Tune the following algorithm-specific hyperparameters: (a) the learning rate η of SAC within the range [0.0005, 0.001], (b) the policy sigma σ of TD3 within the range [0.0001, 0.01], and (c) the clipping threshold ϵ for PPO within the range [0.1, 0.3]. Modify the respective YAML files in the configs directory for the above adjustments. Submit the jobs to the cluster using: `sbatch tune_{ppo/sac/td3}_params.sh`; results are stored in the directory structure: *out/logs/hopper_fixed_hurdles/{ppo/sac/td3}/{seed}* [45 Points]**

3.3 Analysis of SAC's agent [25 Points]

In this exercise, we will investigate different metrics related to the trained SAC agent. Execute the command `python analyze_sac.py -p {logger directory}` to generate two plots that we will analyze. We ask you to add them to the report.

1. **We start by analyzing the plot named *Analyze_SAC_physical_quantities_{hash}_plot.pdf*. (a) Using the Gymnasium documentation, describe what 'actions[:, 2]', 'state[:, 0]' and 'states[:, 11]' stand for. (b) Comment on the shape of the Q function prediction along the x-axis. For that, we advise looking at the reward function of the environment. (c) Explain why 'actions[:, 2]' is negative before the first vertical black line. [15 Points]**

2. **We now analyze the plot named *Analyze_SAC_Q_and_pi_{hash}_plot.pdf*. (a) Explain the values of the Q-function according to the values of the actions. (b) Has the policy been well-trained for the considered state? Two reasons supporting your answer are expected. Be careful. In some cases, the left plot might seem empty. This can happen if the policy focuses on one edge of the plot. In this case, make sure to identify the edge before making the analysis. [10 Points]**

3.4 Bonus - TD3 without Tanh [3 Points]

1. **Train a TD3 agent that does not use a Tanh activation function after the last layer for the actor. For that, you can use the class *ActorNetwork* instead of *BoundedActorNetwork*. What is the main issue affecting the performance? Report the plot showing R, J, V, and H. [3 Points]**

Notes: If the code is implemented correctly, training an agent on the cluster should not take more than 2-4 hours to complete. You should only require close to the full 6 allowed hours if you are running heavy training (e.g. large neural networks).

To test your code without the cluster (e.g. on your local machine) you may simply run the provided bash scripts commands with bash instead of sbatch.

In the bash scripts provided, do not increase the number of cores used beyond 1. This is to save on computation.

If you have a powerful PC, you may find it useful to runs some jobs locally instead of on the cluster in order to save cluster resources for your fellow students.

Reference

[1] John Schulman et al. "Proximal policy optimization algorithms". In: *arXiv preprint arXiv:1707.06347* (2017).

[2] Scott Fujimoto, Herke Hoof, and David Meger. "Addressing function approximation error in actor-critic methods". In: *International conference on machine learning*. PMLR. 2018, pp. 1587–1596.

[3] Tuomas Haarnoja et al. "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor". In: *International conference on machine learning*. PMLR. 2018, pp. 1861–1870.