Reinforcement Learning: From Foundations to Deep Approaches

TECHNISCHE UNIVERSITÄT DARMSTADT

Summer Semester 2024, Homework 1

Prof. Georgia Chalvatzaki, Dr. Davide Tateo

Total points: 103

Deadline: 23:59, Thursday, 7. June 2024

Submission: Use your PC with the MushroomRL Python library to solve the problems. Answers to questions should be added via comments at the end of the completed python files.

Each submission is individual. Submit the completed code by the deadline!

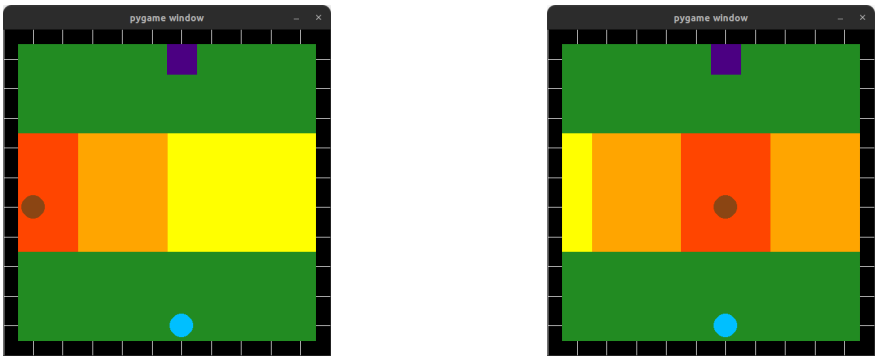
Late submissions: We will accept late submissions, but you will lose 10% of the total reachable points for every day of delay. Note that even 15-minute delays will be counted as being one day late! Submissions three days later than the designated deadline will not be considered. If you are not able to 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.

1.1 Hiker and Bear Environment [30 Points]

A hiker (agent) starts in the environment, aiming to reach the hut. To achieve its goal, the agent has to cross an area occupied by a bear; whenever the agent gets caught by the bear, it gets a penalty and the **episode is terminated**. The probability of getting caught depends on how close the agent is to the bear. Figure 1 depicts the environment.

The agent receives a reward of **1.0** for reaching the goal, a penalty of **-0.1** for getting caught by the bear, and a small penalty of **-0.01** for all other transitions.

The agent can move left, right, up, or down. The bear spawns on the **fifth row** either in the **left-most grid cell** or in the **sixth grid-cell from the left** and does not move during an episode. There is a 5% probability of action failure (the agent stays in its position). The probability of the agent getting caught depends on the **horizontal distance to the bear**: **0.9** if the agent is within **0-1 squares left or right** of the bear, **0.5** if the agent is within **2-4 squares left or right** of the bear, and **0.001** if the agent is elsewhere within the bear's territory.



(a) Bear in position 0 (b) Bear in position 1

Fig. 1: Hiker and Bear environment: The agent is the blue circle, the bear is the brown circle, and the hut is the purple square. The middle 4 rows represent the bear's territory, and the colours presents the areas of increasing risk of getting caught by the bear (red = high risk, orange = medium risk, yellow = low risk). The agent is spawned randomly in the bottom row of the grid.

1. **Define the HikerAndBear environment in `Ex1_env_HB.py`. This class extends the HikerAndBearBase class from `hiker_and_bear_HB.py`, which extends the MushroomRL class FiniteMDP. The class HikerAndBearBase provides some basic functions such as rendering. Define functions to acquire the probability transition matrix, the reward, and the initial state probability distribution. Remember: the state space gets doubled because of the two resting positions of the bear. To test your environment, you can run `Ex1_test_HB.py`. Follow the same steps shown in Practical session 2. [30 Points]**

1.2 Dynamic programming on Hiker and Bear [30 Points]

1. **Solve the problem with Policy Iteration (PI) and Value Iteration (VI) using `Ex2_DP_HB.py`. Visualize the value function and the optimal policy for PI and VI and save them as images in the folder "fig". Adapt the visualization functions from Practical session 2 to use the grid map of the MDP (you will need the visualization functions for all of the following questions).**
   Compare the value maps for PI and VI. What do you observe? [15 Points]
2. **Modify the Policy Iteration method in Mushroom (see Practical session 3) so that it can accept a policy initialization and stopping criterion (in terms of a number of iterations). Initialize with a policy that always goes to the LEFT (action = 2) and plot the obtained Value Function and Policy for 3 and 10 iterations. Save the final policy after 10 iterations of PI. What do you observe compared to the PI that runs until convergence? [15 Points]**

1.3 Model-free RL on Hiker and Bear [40 Points]

1. **In `Ex3_MF_Eval_HB.py`, implement Every-Visit Monte Carlo evaluation and n-step TD prediction (TD(n)) for value prediction. Using the policy of iteration 10 of the modified policy iteration algorithm from Q1.2.2, visualize the predicted value function for MC and TD(5) for at least 1000 rollouts. [10 Points]**
2. **In `Ex3_MFRL_HB.py`, run Q-Learning, SARSA and SARSA(λ) with an ϵ-Greedy policy on the MDP. Run 10 random seeds and plot the performance curves of the behaviour policy and greedy for all algorithms when you evaluate the policy every n steps. Plot the evaluation curves average returns J w.r.t. epochs. (see Practical session 3). Save the final agent of each algorithm to use it later.**
   Which algorithm learns the fastest? Can you think of a reason why?
   Compare the final performance of the greedy policies of Q Learning and SARSA. Which one performs better, and why?
   Compare the behaviour policies of Q Learning and SARSA. How does the final performance compare? Why is this different to the final performance of the greedy policies? [20 Points]

   Hint: Using the provided hyperparameter settings, all of the agents should converge after 100 epochs of 1000 steps/epoch.
3. **Visualize the predicted value matrices for the final seed for each agent from q.3.2. using MC and TD(5) policy evaluation function developed in Q1.3.1. Discuss how the value maps explain the final results in Q1.3.2 using the code given in `Ex3_MF_Eval_RL_HB.py` [10 Points]**

1.4 BONUS [3 Points]

1. **For an algorithm of your choice from the previous question, visualise and label 3 trajectories from different starting points. [3 Points]**