Robot Learning

Winter Semester 2022/2023, Homework 4

Prof. Dr. J. Peters, N. Funk, T. Schneider, J. Ritter, K. Hansel and D. Rother

TECHNISCHE UNIVERSITÄT DARMSTADT

Total points: 39 + 1 bonus
Due date: 23:59, Monday, 30 January 2023

Note: Inside the homework folder, there are:

- A Latex template for this exercise that you can use to fill in your answers and that you can also directly use in sharelatex to collectively write the answers!
- There is also a Conda environment prepared in the "code" folder. To install this, follow the instructions in the README file.

Important: For every answer, please describe precisely how you obtain the results and justify every answer. Otherwise, we cannot give you the points (as in the exam)!
Important: Please, don't forget to submit your code.

Problem 4.1 Questions on Model-Based Reinforcement Learning [17 Points + 1 Bonus]

In this exercise, we will re-visit several aspects of model-based reinforcement learning (MBRL).

a) MBRL vs. MFRL [2 Points]

Name two typical distinctions / differences between Model-based and Model-free Reinforcement learning and explain them.

b) Sample Efficiency [2 Points]

Under what circumstances is Model-based Reinforcement Learning (MBRL) more sample-efficient than Model-free Reinforcement Learning (MFRL)?

c) Different Strategies to Model Learning [6 Points]

As the name suggests, model-based reinforcement learning requires knowledge of the dynamics / kinematics of the overall system. Name the 3 different strategies for model learning and explain them. In particular, also mention how they differ from each other.

d) MBRL for unique, expensive Robot [1 Bonus Points]

Your boss has given you a unique robotic arm which is super expensive. He wants you to implement a MBRL controller for signing letters with his original signature. Given the high cost of the robot and that there only exists one of it, what kind of model learning strategy would you apply to mitigate the risk of destroying the robot?

e) Optimism in RL [3 Points]

First, describe in your own words what is meant by "optimism" in reinforcement learning. Second, give one concrete example of how a MBRL algorithm can be "optimistic" Third, please answer whether optimism in MBRL is beneficial or not.

f) Guided Policy Search [2 Points]

Please briefly summarize what Guided Policy Search is in your own words.

g) Sim-to-real Gap [2 Points]

First, describe in your own words what is meant by the sim to real gap. Second, name and explain a method for counteracting this gap.

Problem 4.2 Cross Entropy Method [22 Points]

In this exercise, you will make use of the Cross-Entropy Method (CEM) to control a cartpole system (also known as inverted pendulum) in the simulation environment MuJoCo. In the lecture, we have come up with 4 different classifications of how models can be used in reinforcement learning. If not stated differently, in this task, we will use the combination of offline model usage together with a sampling-based strategy for computing the optimal actions. Offline model useage refers to the strategy of only using the model a priori to compute the optimal actions. Subsequently, we will apply all of the precomputed actions without any further modifications.

Thus, to sum it up, applying CEM to an inverted pendulum task roughly works as follows. First, we define a distribution over possible control policies. Then we iteratively perform the following steps:

- Sample a set of control policies from the current distribution of policies.
- Evaluate the performance of each policy by running it on the inverted pendulum task.
- Select the top performing policies.
- Update the distribution of policies to be more focused on the top performers.

The process is repeated until the performance of the control policies reaches a satisfactory level or a maximum number of iterations is reached.

It is strongly recommended that you follow the instructions in the README file for installation. For running the experiment, use the provided Conda environment.

All following tasks will be located in the "inverted_pendulum.py" file, and you can find all helper methods in the "utils.py" file.

a) Initial Code understanding [2 Points]

The system we are dealing with has 2 moveable components. First of all, we have the cart which is described by position x and velocity $\dot{x}$, as well as the pole mounted on top of it which is described by angle θ and angular velocity $\dot{\theta}$. The actions a that can be applied to the system moves the cart. The position and velocity of the cart, as well as angle, and angular velocity of the pole are the observations that we get from the system (i.e., they define the observation space), while the action that moves the cart defines the action space. These action and observation spaces are implemented in the "InvertedPendulum" class inside (inverted_pendulum.py).

Inside inverted_pendulum.py, from line 42 onwards, we already implemented a reward function that takes as input state and action along other information that we do not need to consider for now. Given that the state is ordered as previously mentioned, i.e., state[0]$= x$, state[1]$= \dot{x}$,... please write down as a formula the reward function.

Given that the pendulum will always start from the upright position $x = \dot{x} = \theta = \dot{\theta} = 0$, please describe the behavior for which the reward will be maximized.

b) Control Law for the system [3 Points]

The code also already contains a class that implements the control law. It implementation can be found starting from lines 185 in utils.py. Upon object creation (init), it receives a set of parameters (theta) that defines the behavior of the controller together with the observation, as well as the action space. Important, these parameters theta are different from the θ of the previous task (which is part of the state of the system!). The class function "get_action" computes an action, given the current observation, which corresponds to the state of the system.

Please first, write down the formula of the control law. Second, descibe the type of control law we are dealing with. Third, how many parameters describe this control law?

c) Execute an Episode [2 Points]

Next, we want to be able to execute an episode, i.e., the "execute_episode" function (inside inverted_pendulum.py). This function takes in the following arguments: a policy, an MDP (environment), a number of steps, a discount factor, and a flag for enabling rendering. An episode in reinforcement learning refers to a single run of an MDP environment, and the "execute_episode" function is used to execute this run and collect the resulting rewards. The "execute_episode" function typically follows a loop that iteratively chooses actions based on the current state and control policy, executes those actions in the environment, and stores the resulting rewards. This process is repeated until the episode is over, at which point the function returns the collected discounted rewards.

For this task, you not only have to complete the "execute_episode" function, you should also complete the "test_execute_episode" function which creates a policy object and then calls the execute episode function.

Once you have implemented the missing lines in both of the functions, if you call the "test_execute_episode" function (first line of the main function), then a window should open and you should be able to see the cartpole system. For the future, please uncomment this call to the "test_execute_episode" function in the main function as it is not needed any longer.

d) Evaluation Function [2 Points]

Now complete the implementation of the function "evaluation" inside inverted_pendulum.py (using the previous functions!). This evaluation function takes in an MDP, a policy and the number of steps to be executed and it returns the total reward received by executing that policy for that number of steps.

e) CEM-Algorithm [6 Points]

Now, complete the function called "experiment" by applying the steps of the Cross-Entropy Method (CEM) algorithm (see the description above), and following the comments in the code.

Then, use and tune the parameters to achieve the best results (a reward close to zero in a few iterations).

f) Plot Results [1 Points]

Plot the collected rewards over 10 runs of the experiment. Use the "plot_mean_conf" function provided in the "utils.py" module to do this.

g) Assumptions [2 Points]

In this implementation of the CEM, we used the simulator to sort and evaluate the performance of the individual samples. If we apply the obtained distribution of policies to the same simulator we will therefore exactly get the expected behavior. However, if we would apply the policies to a real system, are we guaranteed to get the same performance as during policy optimization? Justify your answer.

h) Systems without prior knowledge [4 Points]

What can be added to a control system if there is no prior knowledge of the system's dynamics? Provide pseudocode for a control system pipeline that starts with no prior knowledge of the system's dynamics?