Robot Learning

Winter Semester 2022/2023, Homework 2

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

TECHNISCHE UNIVERSITÄT DARMSTADT

Total points: 44 + 10 bonus
Due date: 23:59, Monday, 28 November 2022

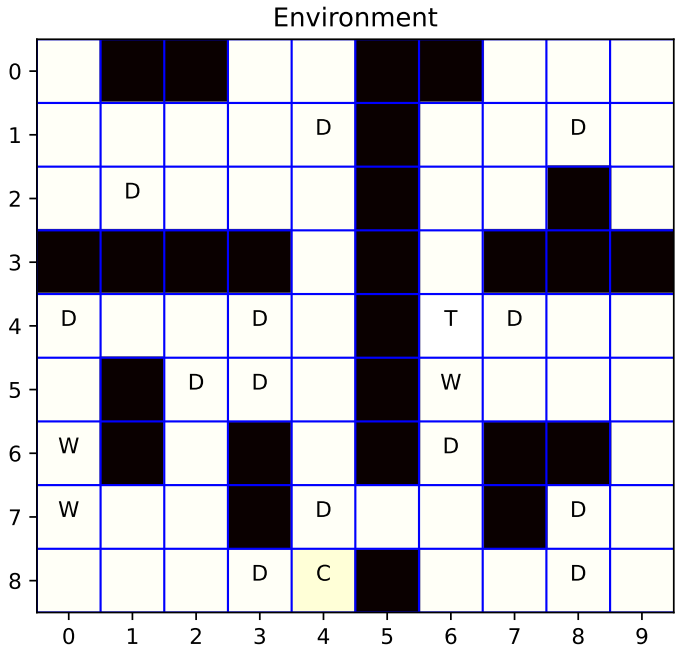
Note: Inside the homework folder, there are:

- 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.
- Attach your plots to the PDF file and submit your code as a python file for every task beside the PDF 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)!

Problem 2.1 Dynamic Programming [24 Points + 10 Bonus]

You recently acquired a robot for cleaning you apartment but you are not happy with its performance and you decide to reprogram it using the latest AI algorithms. As a consequence the robot became self-aware and, whenever you are away, it prefers to play with toys rather than cleaning the apartment. Only the cat has noticed the strange behavior and attacks the robot. The robot is about to start its day and its current perception of the environment is as following



The black squares denote extremely dangerous states that the robot must avoid to protect its valuable sensors. The reward of such states is set to $r_{\text{danger}} = -10^5$ (NB: the robot can still go through these states!). Moreover, despite being waterproof, the robot developed a phobia of water (W), imitating the cat. The reward of states with water is $r_{\text{water}} = -100$. The robot is also afraid of the cat (C) and tries to avoid it at any cost. Note that the cat sleeps and does not move. The reward when encountering the cat is $r_{\text{cat}} = -3000$. The state containing the toy (T) has a reward of $r_{\text{toy}} = 1000$, as the robot enjoys playing with them. Some of the initial specification still remain, therefore the robot receives $r_{\text{dirt}} = 35$ in states with dirt (D).

Also note that the robot does not remove the dirt, as this would impact the solution. If the robot could remove the dirt, the presence of the dirt would be an additional state variable that would make the task much more complex.

The reward collected at an instant of time "t" depends only on the state where the robot is at that instant. Assume that the robot collects the reward at exactly the same instant it starts executing an action and that each action takes one time step to be executed. The robot can perform the following actions: down, right, up, left and stay.

In our system we represent the actions with the an ID (0:down, 1:right, 2:up, 3:left, 4:stay), while the grid is indexed as $\{\texttt{row}, \texttt{column}\}$. The robot can't leave the grid as it is surrounded with walls. A skeleton of the gridworld code and some plotting functions are available at the webpage for all the following questions.

a) Finite Horizon Problem [14 Points]

In the first exercise we consider the finite horizon problem, with horizon $T = 15$ steps. The goal of the robot is to maximize the expected return

$$J_\pi = \mathbb{E}_\pi \left[\sum_{t=1}^{T-1} r_t(s_t, a_t) + r_T(s_T) \right], \tag{1}$$

according to policy π, state s, action a, reward r, and horizon T. Since rewards in our case are independent of the action and the actions are deterministic, Equation (1) becomes

$$J_\pi = \sum_{t=1}^{T} r(s_t). \tag{2}$$

Using the Value Iteration algorithm, determine the optimal action for each state when the robot has 15 steps left. Attach the plot of the policy to your answer and a mesh plot for the value function. Describe and comment the policy: is the robot avoiding the cat and the water? Is it collecting dirt and playing with the toy? With what time horizon would the robot act differently in state $(9, 4)$?

b) Infinite Horizon Problem - Part 1 [4 Points]

We now consider the infinite horizon problem, where $T = \infty$. Rewrite Equation (1) for the infinite horizon case adding a discount factor γ. Explain briefly why the discount factor is needed.

c) Infinite Horizon Problem - Part 2 [6 Points]

Calculate the optimal actions with the infinite horizon formulation. Use a discount factor of $\gamma = 0.8$ and attach the new policy and value function plots. What can we say about the new policy? Is it different from the finite horizon scenario? Why?

d) Finite Horizon Problem with Probabilistic Transition Function [10 Bonus Points]

After a fight with the cat, the robot experiences control problems. For each of the actions up, left, down, right, the robot has now a probability 0.7 of correctly performing it and a probability of 0.3 of performing another action according to the following rule: if the action is left or right, the robot could perform up or down with 0.1 probability each. If the action is up or down, the robot could perform left or right. Additionally, the action can fail causing the robot to remain on the same state with probability 0.1. Using the finite horizon formulation, calculate the optimal policy and the value function. Use a time horizon of $T = 15$ steps as before. Attach your plots and comment them: what is the most common action and why does the learned policy select it?

Problem 2.2 Optimal Control [20 Points]

In this exercise, we consider a finite-horizon discrete time-varying Stochastic Linear Quadratic Regulator with Gaussian noise and time-varying quadratic reward function. Such system is defined as

$$\boldsymbol{s}_{t+1} = \boldsymbol{A}_t \boldsymbol{s}_t + \boldsymbol{B}_t a_t + \boldsymbol{w}_t\,, \tag{4}$$

where $\boldsymbol{s}_t$ is the state, a_t is the control signal, $\boldsymbol{w}_t \sim \mathcal{N}\left(\boldsymbol{b}_t, \boldsymbol{\Sigma}_t\right)$ is Gaussian additive noise with mean $\boldsymbol{b}_t$ and covariance $\boldsymbol{\Sigma}_t$ and $t = 0, 1, \ldots, T$ is the time horizon. The control signal a_t is computed as

$$a_t = -\boldsymbol{K}_t \boldsymbol{s}_t + k_t \tag{5}$$

and the reward function is

$$reward_t = \begin{cases} -(\boldsymbol{s}_t - \boldsymbol{r}_t)^\mathsf{T} \boldsymbol{R}_t (\boldsymbol{s}_t - \boldsymbol{r}_t) - a_t^\mathsf{T} \boldsymbol{H}_t a_t & \text{when} \quad t = 0, 1, \ldots, T-1 \\ -(\boldsymbol{s}_t - \boldsymbol{r}_t)^\mathsf{T} \boldsymbol{R}_t (\boldsymbol{s}_t - \boldsymbol{r}_t) & \text{when} \quad t = T \end{cases} \tag{6}$$

a) Implementation [8 Points]

Implement the LQR with the following properties

$$\boldsymbol{s}_0 \sim \mathcal{N}\left(\boldsymbol{0}, \boldsymbol{I}\right) \qquad T = 50$$

$$\boldsymbol{A}_t = \begin{bmatrix} 1 & 0.1 \\ 0 & 1 \end{bmatrix} \qquad \boldsymbol{B}_t = \begin{bmatrix} 0 \\ 0.1 \end{bmatrix}$$

$$\boldsymbol{b}_t = \begin{bmatrix} 10 \\ 0 \end{bmatrix} \qquad \boldsymbol{\Sigma}_t = \begin{bmatrix} 0.001 & 0 \\ 0 & 0.001 \end{bmatrix}$$

$$\boldsymbol{K}_t = \begin{bmatrix} 5 & 0.3 \end{bmatrix} \qquad k_t = 0.3$$

$$\boldsymbol{H}_t = 1 \qquad \boldsymbol{R}_t = \begin{cases} \begin{bmatrix} 10000 & 0 \\ 0 & 0.01 \end{bmatrix} & \text{if} \quad t = 14 \text{ or } 40 \\ \begin{bmatrix} 0.01 & 0 \\ 0 & 0.1 \end{bmatrix} & \text{otherwise} \end{cases} \qquad \boldsymbol{r}_t = \begin{cases} \begin{bmatrix} 15 \\ 0 \end{bmatrix} & \text{if} \quad t = 0, 1, \ldots, 14 \\ \begin{bmatrix} 25 \\ 0 \end{bmatrix} & \text{if} \quad t = 15, 16, \ldots, T \end{cases}$$

Execute the system 20 times. Plot the mean and 95% confidence (see "68–95–99.7 rule" and matplotlib.pyplot.fill_between function) over the different experiments of the state $\boldsymbol{s}_t$ and of the control signal $\boldsymbol{a}_t$ over time. How does the system behave? Compute and write down the mean and the standard deviation of the cumulative reward over the experiments.

b) LQR as a P controller [4 Points]

The LQR can also be seen as a simple P controller of the form

$$a_t = \boldsymbol{K}_t \left(\boldsymbol{s}_t^{\text{des}} - \boldsymbol{s}_t \right) + k_t \,, \tag{7}$$

which corresponds to the controller used in the canonical LQR system with the introduction of the target $\boldsymbol{s}_t^{\text{des}}$. Assume as target

$$\boldsymbol{s}_t^{\text{des}} = \boldsymbol{r}_t = \begin{cases} \begin{bmatrix} 10 \\ 0 \end{bmatrix} & \text{if} \quad t = 0, 1, \ldots, 14 \\ \begin{bmatrix} 20 \\ 0 \end{bmatrix} & \text{if} \quad t = 15, 16, \ldots, T \end{cases} \tag{8}$$

Use the same LQR system as in the previous exercise and run 20 experiments. Plot in one figure the mean and 95% confidence (see "68–95–99.7 rule" and matplotlib.pyplot.fill_between function) of the first dimension of the state, for both $\boldsymbol{s}_t^{\text{des}} = \boldsymbol{r}_t$ and $\boldsymbol{s}_t^{\text{des}} = \boldsymbol{0}$.

c) Optimal LQR [8 Points]

To compute the optimal gains $\boldsymbol{K}_t$ and $\boldsymbol{k}_t$, which maximize the cumulative reward, we can use an analytic optimal solution. This controller recursively computes the optimal action by

$$a_t^* = -\left(\boldsymbol{H}_t + \boldsymbol{B}_t^T \boldsymbol{V}_{t+1} \boldsymbol{B}_t\right)^{-1} \boldsymbol{B}_t^T \left(\boldsymbol{V}_{t+1} \left(\boldsymbol{A}_t \boldsymbol{s}_t + \boldsymbol{b}_t\right) - v_{t+1}\right), \tag{9}$$

which can be decomposed into

$$\boldsymbol{K}_t = -\left(\boldsymbol{H}_t + \boldsymbol{B}_t^T \boldsymbol{V}_{t+1} \boldsymbol{B}_t\right)^{-1} \boldsymbol{B}_t^T \boldsymbol{V}_{t+1} \boldsymbol{A}_t, \tag{10}$$

$$\boldsymbol{k}_t = -\left(\boldsymbol{H}_t + \boldsymbol{B}_t^T \boldsymbol{V}_{t+1} \boldsymbol{B}_t\right)^{-1} \boldsymbol{B}_t^T \left(\boldsymbol{V}_{t+1} \boldsymbol{b}_t - v_{t+1}\right). \tag{11}$$

where

$$\boldsymbol{M}_t = \boldsymbol{B}_t \left(\boldsymbol{H}_t + \boldsymbol{B}_t^T \boldsymbol{V}_{t+1} \boldsymbol{B}_t\right)^{-1} \boldsymbol{B}_t^T \boldsymbol{V}_{t+1} \boldsymbol{A}_t \tag{12}$$

$$\boldsymbol{V}_t = \begin{cases} \boldsymbol{R}_t + \left(\boldsymbol{A}_t - \boldsymbol{M}_t\right)^T \boldsymbol{V}_{t+1} \boldsymbol{A}_t & \text{when} \quad t = 1 \ldots T-1 \\ \boldsymbol{R}_t & \text{when} \quad t = T \end{cases} \tag{13}$$

$$v_t = \begin{cases} \boldsymbol{R}_t \boldsymbol{r}_t + \left(\boldsymbol{A}_t - \boldsymbol{M}_t\right)^T \left(v_{t+1} - \boldsymbol{V}_{t+1} \boldsymbol{b}_t\right) & \text{when} \quad t = 1 \ldots T-1 \\ \boldsymbol{R}_t \boldsymbol{r}_t & \text{when} \quad t = T \end{cases} \tag{14}$$

Run 20 experiments with: the controller from a); the P controller from b) with $\boldsymbol{s}_t^{\text{des}} = \boldsymbol{r}_t$ (with $\boldsymbol{r}_t$ as defined in a)); and with the controller c) resulting from computing the optimal gains $\boldsymbol{K}_t$ and $\boldsymbol{k}_t$. Plot the mean and 95% confidence (see "68–95–99.7 rule" and matplotlib.pyplot.fill_between function) of both states for all three different controllers used so far. Use one figure per state. Report the mean and std of the cumulative reward for each controller and comment the results.