Assignment 2

Reinforcement Learning - A.Y. 2024/2025

Assigned: November 8th, 2024
Deadline: November 23rd, 2024

Rules

The assignment is due on November 23rd, 2024. Students may discuss assignments, but **each student must code up and write up their solutions independently**. **Students must also indicate on each homework the names of the colleagues they collaborated with** and what online resources they used.

The theory solutions must be submitted in a pdf file named "XXXXXXX.pdf", where XXXXXXX is your matricula. We encourage you to type the equations on an editor rather than uploading a scanned written solution. **In the pdf** you have to hand over **the answers to the theory questions (not just the numerical results, but also the derivations)** and a **small report of the practice exercises**.

The practice exercises must be uploaded in a zip file named "XXXXXXX.zip", where XXXXXXX is your matricula. **The zip file must have the same structure of the assignment.zip** that you find in the attachments, but with the correct solution. You are only allowed to type your code in the files named "student.py". Any modification to the other files will result in penalization. You are not allowed to use any other python library that is not present in python or in the "requirements.txt" file. You can use as many functions you need inside the "student.py" file. The zip file must have the same structure of the "assignment2.zip" (see below).

All the questions must be asked in the Classroom platform but it is forbidden to share the solutions on every forum or on Classroom.

Theory

1. Given the following Q table:

$$Q(s,a) = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} = \begin{pmatrix} Q(1,1) & Q(1,2) \\ Q(2,1) & Q(2,2) \end{pmatrix}$$

   Assume that $\alpha = 0.2$, $\gamma = 0.7$, after an experience $(s, a, r, s') = (2, 2, 5, 1)$ compute the update of both Q-learning and SARSA. For the latter consider $a' = \pi_\epsilon(s') = 1$.

2. Derive the complete Linear Quadratic approximation of a generic non-linear system in a Linear Time-Varying LQR for trajectory following. In particular, given:

   - a non-linear transition function $f(x, u)$ such that $x_{t+1} = f(x_t, u_t)$;
   - a non-quadratic cost function $c(x, u)$, assumed for simplicity to satisfy $c(x, u) = c(x) + c(u)$;
   - a nominal trajectory $\{(x_t^*, u_t^*)\}$, with $t = 0, \ldots, H$;

   derive the general form of the matrices A, B, Q and R that linearize the system around a generic trajectory point (x_t^*, u_t^*), in the appropriate variable substitutions for x_t and u_t, using first-order Taylor expansion for the transition function and second-order Taylor expansion for the cost function.

Practice

1. Implement the Sarsa-λ algorithm on the Taxi environment (https://gymnasium.farama.org/environments/toy_text/taxi/).“” In the folder “sarsa_lambda” you find three files:
   - “main.py” that contains the main script to evaluate your solution. Don’t modify this file!
   - “student.py’ is the file you have to modify, by implementing the function “sarsa_lambda”.
   - “requirements.txt” contains the name of the libraries needed for this part of the assignment.
2. Implement the Q-Learning TD(λ) with linear approximation on the Mountain Car environment using the RBF representation for states and actions. (You can either implement the forward or backward view): https://gymnasium.farama.org/environments/classic_control/mountain_car/.

   In folder “rbf” you find three files:
   - “main.py”, that contains the python script to train the agent and run the tests:
     - Running “python main.py –train model.pkl” you run the training and save the agent in the file “model.pkl”.
     - Running “python main.py –evaluate model.pkl” you evaluate the agent saved in “model.pkl”.
     - Running “python main.py –evaluate model.pkl –render” you evaluate and render the agent saved in “model.pkl”.
     - Running “python main.py –train model.pkl –evaluate model.pkl –render” you train, evaluate and render the agent using the file “model.pkl” to store it.
   - “student.py” contains the classes “TDLambda_LVFA” and “RBFFeatureEncoder” that you have to fill in with the needed code to implement the exercise:
     - For the encoder you need to fill in the functions “__init__” “encode” and “size”.
     - For the agent you need to fill in the function “update_transition”. For the eligibility traces you can use the parameter traces, that is already initialized in the “__init__” function.
   - “requirements.txt” contains the name of the libraries needed for this part of the assignment.

   You must submit **1 model file** named “model.pkl” inside the “rbf” folder of the zip file.

   HINT: You can choose to code the RBF by yourselves or use some implementations online. You cannot copy-paste code, but you can reimplement it getting inspired from some other code (remember to cite the source whenever you use something). There is an implementation of RBF on sklearn that you can import and use (you have sklearn in the requirements.txt): https://scikit-learn.org/stable/modules/generated/sklearn.kernel_approximation.RBFSampler.html