Assignment 1: Multi-Armed Bandits and Dynamic Programming

Reinforcement Learning - A.Y. 2024/2025

Assigned: October 18th, 2024
Deadline: November 2nd, 2024

Rules

The assignment is due on November 2nd, 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 "assignment1.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. Consider a Multi-Armed Bandit with 3 arms. Suppose that, after $t = 10$ iterations of the UCB algorithm, we have the following statistics: $\hat{\mu}_1^t = 0.2$, $\hat{\mu}_2^t = 0.05$, $\hat{\mu}_3^t = 0.45$, $N_1^t = 5$, $N_2^t = 3$, $N_3^t = 2$. Assuming an 85% confidence bound and total number of iterations $T = 100$, Compute:
    (a) the next arm to be pulled following the UCB algorithm;
    (b) the updated statistics after the arm has been pulled, assuming a reward $r = 1$.
2. Given the following environment settings
    - States: $\{s_i : \ i \in \{1...6\}\}$
    - Reward:

$$r(s,a,s') = \begin{cases} 1 & \text{if } s = s_2 \\ -10 & \text{if } s = s_6 \\ 0 & \text{otherwise} \end{cases}$$

- Dynamics: $p(s_5|s_5, a_1) = 0.3$, $p(s_6|s_5, a_1) = 0.7$
- Policy: $\pi(s) = a_1 \;\; \forall s \in S$

and the value function at the iteration $k = 1$:

$$v_k = [0,\ 1,\ 0,\ 0,\ 0,\ -10]$$

with $\gamma = 0.8$.

Compute $V_{k+1}(s_5)$ following the *Value Iteration* algorithm.

Practice

1. Implement the transition probabilities and the reward function of a modified version the FrozenLake Gymnasium environment. See below for more details on the modified transition and reward semantics.

   In folder "policy_iteration" you find three files:

   - "policy_iteration.py" implements the policy iteration algorithm.
   - "main.py" contains the definition of the environment and its attributes (end_state, holes, etc.) and the code to run the tests
   - "student.py" contains the function "reward_function", and "transition_probabilities" that you have to fill in.

   The reward function must return 0 everywhere and 1 for the goal cell, which is always located at the bottom-right corner of the grid. The transition probabilities $p(s'|a, s)$ must be such that the agent moves in the correct direction with probability 1/2 and instead moves in the direction of $a - 1$ (or the last action in the action space if $a = 0$) with probability 1/2.
   Note: you can visualize the episodes by compiling with argument - -*render*.

2. Implement the *explore_and_commit* and the *epsilon_greedy* algorithms as an application of the **contextual bandits** on the recommendation systems domain that we have seen on the course' slides. In folder "cmabs" you find three files:

   - "env.py" implements the environment.
   - "main.py" contains the code to test your solution.
   - "student.py" contains the function "explore_and_commit" and "epsilon_greedy", that you have to fill in.

   The environment is a simple Contextual Multi-Armed Bandit with 3 states and 3 arms. The version of the epsilon-greedy algorithm contained in this assignment does not decay epsilon, but keeps it fixed for a specified number of steps, after which is it set to 0.