SIDRA - Summer School 2025

An Introduction to Stochastic Control and Reinforcement Learning - Homework assignments

Please address the two assignments outlined below and provide a brief report (indicatively 2–3 pages). Assignment 1 also requires a MATLAB implementation. Please provide also the corresponding code and, if needed, a brief explanation of how to run it.

Assignment 1

Consider a mobile agent that moves on an *n×m* planar grid (think of a chessboard with *(1,1)* denoting the bottom- left- most cell). At each time step, the robot is located in a cell and decides whether to move upward, downward, to the right, or to the left in the next time step. The goal is to design a learning scheme such that the agent repeatedly moves from cell *(1,1)* to cell *(n,m)* and vice versa (i.e., once *(n,m)* is reached, the goal becomes reaching *(1,1)*, and so forth and so on), avoiding collisions with the "walls" of the grid and refraining from traversing certain cells. These cells which are unknown beforehand, represent immovable obstacles that are revealed during exploration.

Introduce a proper Markov Decision Process (MDP) that describes the problem at hand (defines clearly the states, the control actions, and the transitions) and introduce appropriate losses associated to the transitions. Then, implement in MATLAB a program that receives as input the grid size (n,m) and the cells that are forbidden and then run a simulation of the MDP along with a RL algorithm to learn the optimal policy from experience. You may try one or two RL algorithm (e.g. SARSA+(policy iteration), Q-learning, or Policy Gradient). Provide a brief description of the MDP formulation and the code as MATLAB files.

Assignment 2

Given a finite-state finite-control action Markov Decision Process, consider an infinite horizon optimal control problem where the total cost is defined as

$$\mathrm{J}(x,\{u_t\}) = \mathrm{E}\left[\sum_{t=0}^{+\infty} \alpha(t)\cdot g(x_t,u_t,w_t)\right],$$

and: 1. $x_t \in X = \{1,2,\dots,n\}$ is the state and $x_0 = x \in X$ is the initialization; 2. $u_t \in U$ is the control action; 3. $g(x_t,u_t,w_t)$ is the (bounded) cost generated at every transition (w_t is the exogenous stochastic input driving the state evolution); 4. and

$$\alpha(t) = \begin{cases} \gamma^t & \text{if } t \text{ is even} \\ \gamma^{t-1} & \text{if } t \text{ is odd} \end{cases}$$

(assume that $0 < \gamma < 1$). Moreover, x_t is only observed when t is even and this information is used to decide both u_t and u_{t+1}. In other words,

$$u_t = \begin{cases} \mu_t(x_t) & \text{if } t \text{ is even} \\ \mu_t(x_{t-1}) & \text{if } t \text{ is odd} \end{cases}$$

where $\pi = \{\mu_0, \mu_1, \mu_2 \dots\}$ is a policy. For the present problem, letting

$$V^*(x) = \min_\pi J\big(x,\{\mu_t(x_t)\}\big)$$

be the value function, the Bellman functional equation becomes ($p(i|j,u)$, $i,j \in X, u \in U$, denotes a transition probability):

$$V^*(x) = \min_{u,v \in U} \left[E[g(x,u,w_0)] + \sum_{i=1}^{n} p(i|x,u) \cdot E[g(i,v,w_1)] + \gamma^2 \sum_{i,j=1}^{n} p(i|x,u) \cdot p(j|i,v) \cdot V^*(j) \right].$$

i. Explain why the Bellman functional equation takes the form above (note: a complete formal proof is not required, give ideas instead).

ii. Introduce a modified Q-learning algorithm and the corresponding learning scheme that accommodates the present setup.