MVA: Reinforcement Learning (2020/2021) **Homework 1**

Dynamic Programming

Lecturers: *A. Lazaric, M. Pirotta* *(October 21, 2020)*

Solution by FILL fullname command at the beginning of latex document

Instructions

- The deadline is **November 8, 2020. 23h00**
- By doing this homework you agree to the *late day policy, collaboration and misconduct rules* reported on Piazza.
- **Mysterious or unsupported answers will not receive full credit**. A correct answer, unsupported by calculations, explanation, or algebraic work will receive no credit; an incorrect answer supported by substantially correct calculations and explanations might still receive partial credit.
- Answers should be provided in **English**.

1 Question

Consider the following grid environment. The agent can move up, down, left and right. Transitions are deterministic. Attempts to move in the direction of the wall will result in staying in the same position. There are two absorbing states: 1 and 14. Taking any action in 1 (resp 14) leads to a reward r_r (resp. r_g) ($r(1,a) = r_r, r(14,a) = r_g, \forall a$) and *ends the episode*. Everywhere else the reward is r_s. Assume discount factor $\gamma = 1$, $r_g = 10$ and $r_r = -10$, unless otherwise specified.

0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

1. Define r_s such that the optimal policy is the shortest path to state 14. Using the chosen r_s, report the value function of the optimal policy for each state.
   There is a simple solution that doesn't require complex computation. You can copy the image an replace the id of the state with the value function.

2. Consider a general MDP with rewards, and transitions. Consider a discount factor of $\gamma < 1$. For this case assume that the horizon is infinite (so there is no termination). A policy π in this MDP induces a value function V^π. Suppose an affine transformation is applied to the reward, what is the new value function? Is the optimal policy preserved?

3. Consider the same setting as in question 1. Assume we modify the reward function with an additive term $c = 5$ (i.e., $r_s = r_s + c$). How does the optimal policy change (just give a one or two sentence description)? What is the new value function?

2 Question

Consider infinite-horizon γ-discounted Markov Decision Processes with S states and A actions. Denote by $Q^\star$ the Q-function of the optimal policy $\pi^\star$. Prove that, for any function $Q(s,a)$, the following inequality holds for any s

$$V^{\pi_Q}(s) \geq V^\star(s) - \frac{2\|Q^\star - Q\|_\infty}{1-\gamma}$$

where $e = (1,\dots,1)$, $\|Q^\star - Q\|_\infty = \max_{s,a} |Q^\star(s,a) - Q(s,a)|$ and $\pi_Q(s) = \arg\max_a Q(s,a)$. Thus $\pi^\star(s) = \arg\max_a Q^\star(s,a)$.

3 Question

Consider the average reward setting ($\gamma = 1$) and a Markov Decision Process with S states and A actions. Prove that

$$g^{\pi'} - g^\pi = \sum_s \mu^{\pi'}(s) \sum_a (\pi'(a|s) - \pi(a|s)) Q^\pi(s,a) \tag{1}$$

using the fact that in average reward the Bellman equation is

$$Q^\pi(s,a) = r(s,a) - g^\pi + \sum_{s'} p(s'|s,a) \sum_{a'} \pi(a'|s') Q^\pi(s',a'), \quad \forall s,a,\pi$$

and μ^π is the **stationary distribution** of policy π. Note also that $g^\pi = \sum_x \mu^\pi(s) \sum_a \pi(a|s) r(s,a)$.

Note: All the information provided to prove Eq. 1 are mentioned in the question. Start from the definition of Q and use the property of stationary distribution.

4 Question

Provide an MDP modeling, specifying all its defining elements, of the following process:

- Elevator dispatching. The elevator controllers assign elevators to service passenger requests in real-time while optimizing the overall service quality, e.g. by minimizing the waiting time and/or the energy consumption. The agent can simultaneously control all the elevators. In order to model this problem, consider a 6-story building with 2 elevators. Explain the choices made for modeling this problem.

5 Question

Implement value iteration and policy iteration. We have provided a custom environment in the starter code.

1. (coding) Consider the provided code `vipi.py` and implement `policy_iteration`. Use γ as provided by `env.gamma` and the terminal condition seen in class. Return the optimal value function and the optimal policy.
2. (coding) Implement `value_iteration` in `vipi.py`. The terminal condition is based on $\|V_{new} - V_{old}\|_\infty$ and the tollerance is 10^{-5}. Return the optimal value function and the optimal policy.
3. (written) Run the methods on the deterministic and stochastic version of the environment. How does stochasticity affect the number of iterations required, and the resulting policy?
   You can render the policy using `env.render_policy(policy)`
4. (written) Compare value iteration and policy iteration. Highlight pros and cons of each method.