Dynamic Programming and Reinforcement Learning

Lecturer: *Matteo Pirotta* *matteo.pirotta@inria.fr*

Report Deadline: November 10, midnight

Note: Your report should be *short* and based on the answers to questions Q1-Q6.
The code is not optional!

The solution (single archive with name *lastname_firstname_TP1.zip*) should be sent by e-mail to matteo.pirotta@inria.fr, with "[MVA 2017] TP1" as object.

We provide support for Matlab and Python but you can solve the homework using any language. You can choose between the following formats for the solution:

- Report and code: a short report (PDF) based on the answers to questions and the source code.
- Code with comments: integrate answers and comments in the code. For example, you can use a Matlab report or a Python notebook.

Note: if you want to receive notifications about the course you can add your email here `https://tinyurl.com/mva17rlcontacts`.

1 Dynamic Programming

The following simple example illustrates the basic components of a Markov Decision Process (MDP) with stationary reward and transition model. Fig. 1 illustrates a simple three-state MDP. The elements in braces denote the rewards, while the weights on the arcs are the transition probabilities. For example, action a_0 in s_0 is such that

$$p(s_0|s_0,a_0) = 0.45, \quad p(s_1|s_0,a_0) = 0.55, \quad p(s_2|s_0,a_0) = 0 \quad \text{and} \quad r(s_0,a_0) = -0.4$$
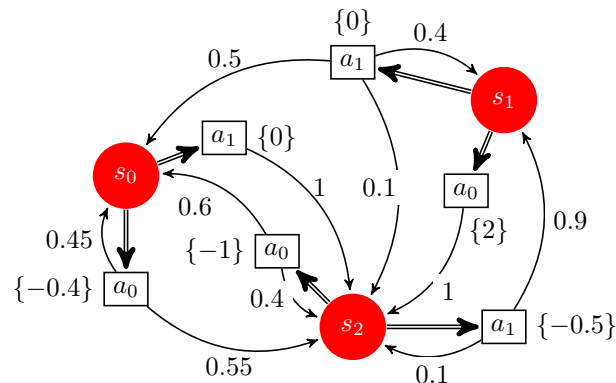

Figure 1: Symbolic representation of a three-state Markov decision process.

x_0	x_1	x_2	x_3
x_4	/////	x_5	x_6
x_7	x_8	x_9	x_{10}

Figure 2: Simple grid world with zero-based state enumeration. Since Matlab used one-based indices, in the provided code, states are numbered from 1 to 11.

Assume a discount factor $\gamma = 0.95$.

- Q1: Implement the discrete MDP model.

- Q2: Implement and run value iteration in order to identify a 0.01–optimal policy. Recall that the stopping criterion (theory slides) is to stop when

$$\|v^{k+1} - v^k\|_\infty < \epsilon.$$

  which implies that [Puterman, 1994, Th. 6.3.1]

$$\|v^{d_{k+1}^+} - v^*\| < \frac{2\epsilon\gamma}{1 - \gamma},$$

  where d_{k+1}^+ is the greedy policy w.r.t. v^{k+1}.
  Plot $\|v^k - v^*\|_\infty$ as a function of iteration k, knowing that $d^* = [a_1, a_0, a_1]$ (i.e., implement policy evaluation).

- Q3: implement policy iteration with arbitrary initial policy. Compare the speed of convergence w.r.t. VI and discuss the relative merits of the two approaches.

2   Reinforcement Learning

In the previous exercise, we have investigated methods that require the knowledge of the MDP, in particular the knowledge of the transition model. However, this information is often unavailable. In this exercise, we consider the case in which a simulator is available but the transition probabilities are unknown.

Consider the simple grid world reported in Fig. 2. The state space is $\mathcal{X} = \{x_0, \ldots, x_{10}\}$ and the action space is given by the four cardinal actions $\mathcal{A} = \{\text{right, down, left, up}\}$. The actions are stochastic, i.e., they may move the agent to an undesired state. In each state, the agent can execute only a subset of actions (i.e., actions leading against the wall are removed). For example,

$$\mathcal{A}(x_{10}) = \{\text{left, up}\} \quad \text{and} \quad \mathcal{A}(x_4) = \{\text{down, up}\}.$$

The cell filled with lines represents a wall, while target states are denoted in blue. The reward function is zero everywhere except in the target states ($r(x, a, x_3) = 1$ and $r(x, a, x_6) = -1$, for any (x, a)). The target state is absorbing and a single action is defined. Consider a discount factor of $\gamma = 0.95$.

A simulator of the MDP is available online (`gridworld.py` or `GridWorld.m`).

2.1 A Review of RL Agent/Environment Interaction

In this section, we review the process of interaction between agent and environment. We consider without loss of generality episodic settings[1] and that a simulator of the MDP is available.

The MDP simulator is in general composed by

1. `reset`: generates the initial states accordingly to the initial state distribution μ_0.

2. `step`: receives in input a state and an action and it returns the next state, reward and a terminal flag. The terminal flag is true if and only if the state that is reached is absorbing. For example, in the grid world it is true when x_3 and x_6 are reached.

Note that when an absorbing state is reached, the environment restarts from a state drawn accordingly to the initial state distribution. The overall structure of the interaction is reported in Alg. 1.

This schema is the building block of almost all the RL approaches. It can be used to collect trajectories ($\tau = \{(x_i, a_i, r_i, x_{i+1})\}_{i \geq 0}$) or to compute any estimate required by the RL algorithm (e.g., temporal difference error, MC estimate of value function).

Note on $T_{\max}$. In several episodic problems, it is easy to select the value for $T_{\max}$ but, in general, it is not clear when to stop the simulation. In infinite-horizon discounted problems, we can use the notion of effective horizon: $\mathcal{O}(1/(1-\gamma))$. In particular, if the reward is bounded $r_t \in [0, R_{\max}]$, by setting $T_{\max} = \mathcal{O}(-\log(\delta/R_{\max})/(1-\gamma))$, the discounted truncated sum or rewards is δ-close to the infinite sum [Kakade et al., 2003, Sec. 2.3.3].

2.2 Work to do

1. Policy evaluation. Consider the deterministic policy that is selecting the action *right* when available, otherwise *up*.
   <u>Q4</u>: denote with $Q_n(x, a)$ the value function estimated using *first-visit* Monte-Carlo:

$$Q_n(x, a) = \frac{1}{N(x,a)} \sum_{k=1}^{N(x,a)} \left[\sum_{t=1}^{T_{\max}} \gamma^{t-1} r_t^{(k)} \right], \text{ with } x_1 = x, a_1 = a, a_{t>1} \sim \pi \tag{1}$$

where $\sum_{x,a} N(x, a) = n$ and $(r_t^{(k)})$ is the sequence of rewards obtained when simulating the k-th trajectory (using the simulator).

---

[1]Non-episodic problems can be viewed as episodic problems with one episode.

Algorithm 1: Agent/Environment interaction

```
1 for e = 1, ..., E do
2 │   x_0 = reset()
3 │   t = 0
4 │   repeat
5 │   │   a_t ~ π(·|x_t)
6 │   │   x_{t+1}, r_t, term = step(x_t, a_t)
7 │   until (t < T_max ∧ not term)
8 end
```

Build such estimator and plot $J_n - J^\pi$ as a function of n, where

$$V^\pi = [0.877, 0.928, 0.988, 0, 0.671, -0.994, 0, -0.828, -0.877, -0.934, -0.994]^\mathsf{T}$$

is the value function computed with DP, μ_0 is the uniform distribution and

$$J_n = \sum_{x \in \mathcal{X}} \mu_0(x) V_n(x), \quad \text{and} \quad J^\pi = \sum_{x \in \mathcal{X}} \mu_0(x) V^\pi(x).$$

If you want to directly compare the Q-function, Q^π is provided in the code (together with V^π).

Note: you should adapt the structure provided in Alg. 1. The number of episodes to generate is $E = n$. Pay attention to the fact that the problem is episodic while defining $T_{\max}$. Functions for the visualization of the Q-function and policy are provided.

2. Policy optimization: *the Q-learning algorithm.*
   Q-Learning estimates the quantities $Q^*(x, a)$ for all $(s, a) \in \mathcal{X} \times \mathcal{A}$. The algorithm simulates trajectories $\{\tau_i\}$ and updates its estimates along the way using

   $$Q_{t+1}(x_t, a_t) = \left(1 - \alpha_{N(x_t, a_t)}(x_t, a_t)\right) Q_t(x_t, a_t) + \alpha_{N(x_t, a_t)}(x_t, a_t) \left[r_t + \gamma \max_{b \in \mathcal{A}} Q_t(x_{t+1}, b)\right],$$

   where

   - $N(x, a)$ is the number of visits of the state-action (x, a)
   - for each (x, a), $\alpha_i(x, a)$ is a sequence of *stepsizes*
   - in state x_t, a_t is chosen based on some *exploration policy*

   For the algorithm to converge, the stepsizes and exploration policy should be chosen such that all state-action pairs are visited infinitely often and $\alpha_i(x, a)$ satisfies the usual stochastic approximation requirements (Robbins-Monro conditions [Robbins and Monro, 1951]):

   $$\sum_i \alpha_i(x, a) = +\infty \quad \text{and} \quad \sum_i \alpha_i^2(x, a) < +\infty.$$

   You will implement Q-Learning in *episodes*[2] of length $T_{\max}$, with an ϵ-greedy exploration policy within each episode, i.e., select an action as $a_t = \arg\max Q(x_t, a)$ with probability $1 - \epsilon$ and randomize with probability ϵ.

   Q5: Describe the (parameters of the) exploration policy and learning rate chosen, and illustrate the convergence of Q-Learning using some of the following performance metrics:

   - Performance over all the other state $||v^* - v^{\pi_n}||_\infty$, where π_n is the greedy policy w.r.t. Q_n at the end of the n-th episode
   - Reward cumulated over the episode

   Note that
   $$v^* = [0.877, 0.928, 0.988, 0, 0.824, 0.928, 0, 0.778, 0.824, 0.877, 0.828]^\mathsf{T}.$$

3. Q6: Is the optimal policy of an MDP affected by the change of the initial distribution μ_0?

---

[2] *Note:* pay attention to how terminal states are treated.

References

Sham Machandranath Kakade et al. *On the sample complexity of reinforcement learning.* PhD thesis, University of London London, England, 2003.

Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming.* John Wiley & Sons, Inc., New York, NY, USA, 1994. ISBN 0471619779.

Herbert Robbins and Sutton Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951.