DDA4230 Reinforcement learning MDPs and Optimality of MDPs

Lecture 3

Lecturer: Baoxiang Wang *Scribe: Baoxiang Wang*

1 Goal of this lecture

In this lecture we will introduce the optimality of Markov decision processes and discuss some examples of MDPs.
Suggested reading: Chapter 3 and 4 of *Reinforcement learning: An introduction*; *Human-level control through deep reinforcement learning* (Mnih, Volodymyr, et al. Nature 2015); Chapter 1 of *Reinforcement Learning: Theory and Algorithms*;

2 Recap: Markov decision processes

We consider the discrete-time Markov decision process (MDP) setting, denoted as the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{T}, r, \rho_0, \gamma)$.

- $\mathcal{S}$ the state space;
- $\mathcal{A}$ the action space. $\mathcal{A}$ can depend on the state s for $s \in \mathcal{S}$;
- $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ the environment transition probability function;
- $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \Delta(\mathbb{R})$ the reward function;
- $\rho_0 \in \Delta(\mathcal{S})$ the initial state distribution;
- $\gamma \in [0, 1]$ the unnormalized discount factor.

Note that $\Delta(\mathcal{X})$ denotes the set of all distributions over set $\mathcal{X}$.

A stationary MDP follows for $t = 0, 1, \ldots$ as below, starting with $s_0 \sim \rho_0$.

- The agent observes the current status s_t;
- The agent chooses an action $a_t \sim \pi(a_t \mid s_t)$;
- The agent receives the reward $r_t \sim \mathcal{R}(s_t, a_t)$;
- The environment transitions to a subsequent state according to the Markovian dynamics $s_{t+1} \sim \mathcal{T}(s_t, a_t)$.

This process generates the sequence $s_0, a_0, r_0, s_1, \ldots$ indefinitely. The sequence up to time t is defined as the trajectory indexed by t, as $\tau_t = (s_0, a_0, r_0, s_1, \ldots, r_t)$.

The goal is to optimize the expected return

$$\mathbb{E}_{s_t,a_t,r_t,t\geq 0}[R_0] = \mathbb{E}_{s_t,a_t,r_t,t\geq 0}[\sum_{t=0}^{\infty} \gamma^t r_t]$$

over the agent's policy π.

3 Optimality of MDPs

We now have all the background necessary to discuss the problem of reinforcement learning, where we seek to find a best policy that achieves the greatest value function among the set of all possible policies. In the context of reinforcement learning, this is precisely the objective of the agent.

To get started, we need to address the question

what do we exactly mean by finding an optimal policy?

Precisely we wanted to know whether a policy always exists, which we will denote by π^*, whose value function is at least as good as the value function of any other policy. In other words, we need to ensure that the supremum of the value function is actually attained for some policy.

To appreciate the subtlety of this point, consider the example of maximizing the function $f : \mathbb{R} \to \mathbb{R}$ on $(0, 1)$ defined as $f(x) = x$, and note that this problem does not have a solution. But $\sup f(x) = 1$, although there does not exist an $x \in (0, 1)$ for which this is attained. We first define precisely what it means for a policy, not necessarily stationary, to be an optimal policy. The existence of an optimal policy will be assumed throughout the course unless otherwise mentioned.

Definition 1 *A policy π^* is an optimal policy if for every policy π, for all states $s \in S$, $V^{\pi^*}(s) \geq V^{\pi}(s)$.*

When the MDP is non-stationary or is with a finite horizon, the definition of optimality will be $V_t^{\pi^*}(s) \geq V_t^{\pi}(s)$ for every π, s, and t.

In the setting of stationary, infinite-horizon MDPs (which is the MDP defined in this lecture notes), if some not-necessarily stationary policy is optimal, then at least one stationary policy is optimal. An intuitive proof sketch is constructive. Simply specify a not-necessarily stationary optimal policy and drop its dependency on t (for example, by setting $t = 0$). This will obtain a stationary optimal policy.

For stationary MDPs, if some not-necessarily deterministic policy is optimal, then at least one deterministic policy is optimal. The proof is also constructive. We can first specify a not-necessarily deterministic optimal policy and then define a new deterministic policy, whose action is an arbitrarily action of the optimal policy with probability greater than 0.

Taking discrete state space $\mathcal{S} = [n]$ and action space $\mathcal{A} = [m]$ as an example, the total number of policies is m^n. On the contrary, this number is m^{nT} if we allow non-stationarity for some horizon T. For stochastic policies, the size of the universe will be infinity. We see how significant stationary and deterministic policies reduce the universe of policies when searching for an optimal policy. Note that when the action space is depending on the state, this number of policies is $\prod_{s \in \mathcal{S}} |\mathcal{A}(s)|$.

We have thus established the existence of an optimal policy and moreover concluded that a deterministic stationary policy suffices. This then allows us to make the following definition.

Definition 2 *The optimal state value function for an infinite horizon MDP is defined as*

$$V^*(s) = \max_{\pi \in \Pi} V^{\pi}(s) \ , \tag{1}$$

and there exists a stationary deterministic policy $\pi^* \in \Pi$, which is an optimal policy, such that $V^*(s) = V^{\pi^*}(s)$ for all states $s \in \mathcal{S}$, where Π is the set of all stationary deterministic policies.

The optimal value function is unique for an MDP. Otherwise if two value functions do not agree on some state, the one possessing the smaller value is not the optimal value function. The uniqueness of optimal policies does not hold in general.

3.1 The Bellman optimality equation

The Bellman optimality equation, named after Richard E. Bellman, is a necessary condition of a value function to be optimal. It describes the recursive property of the optimal value function

$$V^*(s_t) = \max_a \ \mathbb{E}[r_t + \gamma V^*(s_{t+1}) \mid a_t = a].$$

It is worth noting that the Bellman optimality equation is different from the Bellman equation. The Bellman equation describes an arbitrary policy's value function and it therefore averages over a_t instead of taking the maximum, which writes $V^\pi(s_t) = \mathbb{E}[r_t + \gamma V^\pi(s_{t+1})]$.

This property is critical and it leads to many algorithms to learn the optimal policy and value.

Value iteration LHS is no larger than RHS if we replace V^* by a not-necessarily optimal value function V. Therefore for discrete state and action space. we can assign RHS to V and repeat the iteration. This leads to improvements of the current value for each iteration and will converge to the optimal value function under some conditions.

Q-learning Despite that the property is necessary but not sufficient, a set of algorithms seek function who fulfill this property and are then more likely to be close to optimal. The approach to satisfy the Bellman optimality equation is to define a Bellman error (e.g. $\frac{1}{2}(LHS - RHS)^2$) and minimize it. It is commonly presented on an alternative form of $Q^*(s_t, a_t) = \max_a \mathbb{E}[r_t + \gamma Q^*(s_{t+1}, a)]$.

4 Examples of MDPs

4.1 Boyan chains (Markov process)

The Boyan chain for $N+1$ states is a Markov chain with the following transition probability. It has a reward of -1 for every step and the process terminates at state $N+1$.

The value function $V(s)$ is then the negative number of steps elapsed starting from state s until the terminal state. Then by the linearity of expectation, we have $V(s) = \frac{1}{2}V(s+1) + \frac{1}{2}V(s+2) - 1$ for $s \leq N-1$. With the boundary conditions $V(N) = -1$ and $V(N+1) = 0$ we find that

$$V(s) = \frac{4}{9} - \frac{2}{3}(n - s + 2) - \frac{4}{9}(-\frac{1}{2})^{n-s+2}.$$

Variants of Boyan chain are classical games to test the ability of the agents to explore and to not stuck at local optimums.

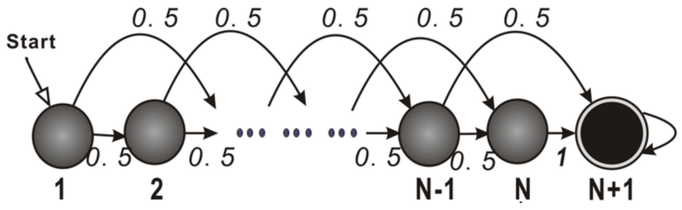


Figure 1: The Boyan Markov chain.

4.2 Mountain Car

The problem describes the decision-making of a car driver who, starts from a valley, aims to drive to the mountain peak (flag). It is however insufficient to drive directly from the valley rightwards.

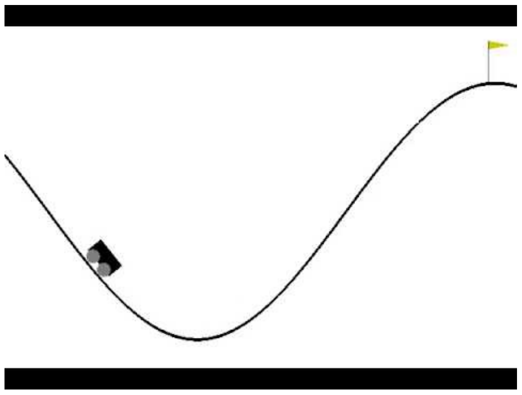


Figure 2: The Mountain Car problem.

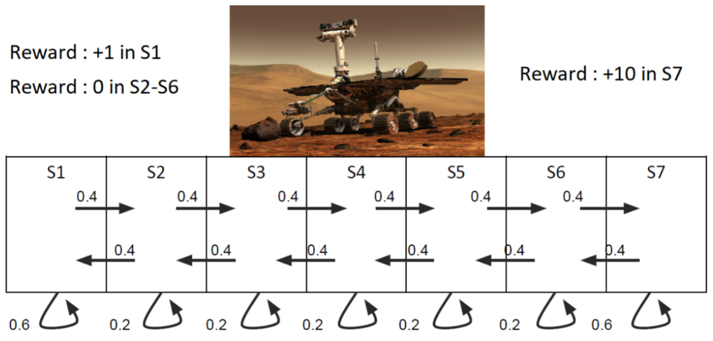
As it requires the agent to move left, which initially seems to be moving further away from the goal, it is a classical demonstration of delayed gratification and the problem of temporal credit assignment.

4.3 Mars Rover

A artificial example describing the movement of a Mars rover with some okay field site and fantastic field site to discover. The environment can be either deterministic or stochastic.

Acknowledgement

This lecture notes partially use material from *Reinforcement learning: An introduction* and *CS234: Reinforcement Learning* from Stanford. Figure 1 is from *Preconditioned temporal difference learning* by Hengshuai Yao and Zhi-Qiang Liu. Proofread by Shaokui Wei.



S1	S2	S3	S4	S5	S6	S7
Okay Field Site R=+1	R=0	R=0	R=0	R=0	R=0	Fantastic Field Site R=+10

$$P(s'|s,TL)=\begin{array}{ccccccc} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{array} \qquad P(s'|s,TR)=\begin{array}{ccccccc} 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{array}$$

Figure 3: The Mars Rover MDP.