Lecture 2 - Optimality of MDPs

Guiliang Liu

The Chinese University of Hong Kong, Shenzhen

DDA4230: Reinforcement Learning
Course Page: [Click]

DDA 4230 Resources

Please join our Slack group.

https://join.slack.com/t/slack-us51977/shared_invite/zt-22g8b40v8-0qSs9o0G3~8hXHwWydlCpw

Please check our course page.

https://guiliang.github.io/courses/cuhk-dda-4230/dda_4230.html

香港中文大學(深圳)
The Chinese University of Hong Kong, Shenzhen

Optimality of MDPs

Reinforcement Learning seeks to find the best policy that achieves the greatest value function among the set of all possible policies.

What do we exactly mean by finding an optimal policy?

The existence of an optimal policy will be assumed throughout the course unless otherwise mentioned.

Optimality of MDPs

We first define precisely what it means for a policy, not necessarily stationary, to be an optimal policy.

Definition

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.

Optimality of MDPs

In the setting of stationary, infinite-horizon MDPs:

- If some not-necessarily stationary policy is optimal, then at least one stationary policy is optimal: $\exists \pi^*{}_t(a|s) \to \pi^*(a|s)$.
- if some not-necessarily deterministic policy is optimal, then at least one deterministic policy is optimal. $\exists \pi^*(a|s) \to \pi^*(s)$.

香港中文大學（深圳）
The Chinese University of Hong Kong, Shenzhen

Optimality of MDPs

Taking discrete state space $\mathcal{S} = [n]$ and action space $\mathcal{A} = [m]$ as an example,

- Stationary and deterministic MDP: the total number of policies is m^n.
- Non-Stationary and deterministic MDP: the total number of policies is m^{nT}.
- Stochastic MDP: the total number of policies will be infinite.

Stationary and deterministic policies can significantly reduce the size of the universe of policies when searching (especially the brute-force search that checks all policies) for an optimal policy.

香港中文大學（深圳）
The Chinese University of Hong Kong, Shenzhen

Optimality of MDPs

After establishing the existence of an optimal policy and moreover concluding that a deterministic stationary policy suffices, we define:

Definition

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.

Optimality of MDPs

The uniqueness of optimal value functions and policies.

- The optimal value function is unique for an MDP.
- The uniqueness of optimal policies does not hold.

Proof. Consider a counter-example to the uniqueness of optimal policy: we introduce a 'dummy' state that is never accessed, and which carries a reward value of zero. Under this hypothetical circumstance, any policy could execute arbitrary actions without influencing the overall value assessments.

Optimality of MDPs

Optimal policy (action) is likely to be unique when the decision matters and the task is natural and complicated enough.

Dynamic Programming

Sequential decision-making can be solved via dynamic programming for the finite horizon case. Denoting $V_t^*(s)$ to be the optimal value function at time t,

$$V_t^*(s) = \max_a \mathbb{E}[\mathcal{R}(s,a)] + \gamma \sum_{s' \in S} \mathbb{P}(s' \mid s,a) V_{t+1}^*(s') \,, \quad \forall\ t = 0, \ldots, T-1 \,,$$

$$V_T^*(s) = 0 \,.$$

For example,starting from T to $T-2$, we can compute:

$$V_T^*(s) = 0 \,, \quad V_{T-1}^*(s) = \max_a \mathbb{E}[\mathcal{R}(s,a)] \,,$$

$$V_{T-2}^*(s) = \max_a \mathbb{E}[\mathcal{R}(s,a)] + \gamma \sum_{s' \in S} \mathbb{P}(s' \mid s,a) V_{T-1}^*(s') \,.$$

Dynamic Programming

Sequential decision-making can be solved via dynamic programming for the finite horizon case. Denoting $V_t^*(s)$ to be the optimal value function at time t,

$$V_t^*(s) = \max_a \mathbb{E}[\mathcal{R}(s,a)] + \gamma \sum_{s' \in S} \mathbb{P}(s' \mid s,a) V_{t+1}^*(s'), \quad \forall\, t = 0, \dots, T-1,$$

$$V_T^*(s) = 0.$$

However, this requires the knowledge of the world model ($\mathcal{R}(s,a)$ and $\mathbb{P}(s' \mid s,a)$).

Can we estimate $\mathcal{R}(s,a)$ and $\mathbb{P}(s' \mid s,a)$ for dynamic programming?

Yes, but good decisions do not mean good estimations.

For example, by empirical estimation: $\hat{\mathbb{P}}(s' \mid s,a) = \frac{n_\#(s')}{n_\#(s,a)}$.

香港中文大學（深圳）
The Chinese University of Hong Kong, Shenzhen

The Bellman Optimality Equation

The Bellman optimality equation, named after Richard E. Bellman, is a necessary condition for a value function to be optimal:

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

The Bellman optimality equation $\neq$ The Bellman equation.

- The Bellman equation describes an arbitrary policy's value function $V(s_t) = \mathbb{E}[r_t + \gamma V(s_{t+1})]$ (expected w.r.t. $\pi(a_t|s_t)$).
- The Bellman optimality equation takes the maximum overall actions (no policy in the expectation).

The Bellman Optimality Equation

The Bellman optimality equation, named after Richard E. Bellman, is a necessary condition for a value function to be optimal:

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

Value Iteration (VI). If we replace V^* by a not-necessarily optimal value function V, VI assigns RHS to V and repeats the iteration:

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

This leads to improvements of the current value for each iteration and V will converge to the optimal value function under some conditions.

香港中文大學(深圳)
The Chinese University of Hong Kong, Shenzhen

The Bellman Optimality Equation

The Bellman optimality equation, named after Richard E. Bellman, is a necessary condition for a value function to be optimal:

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

Q-learning. In order to satisfy the Bellman optimality equation in an alternative form (Q^* is the action-value function for an optimal policy):

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

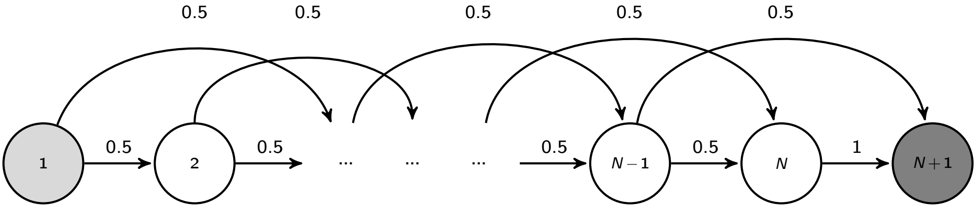
Q-learning defines a Bellman error (e.g. $\frac{1}{2}(LHS - RHS)^2$) and minimizes it,

$$\frac{1}{2}\left[Q(s_t, a_t) - \max_a \mathbb{E}[r_t + \gamma Q(s_{t+1}, a)]\right]^2$$

香港中文大學(深圳)
The Chinese University of Hong Kong, Shenzhen

Examples of MDPs: Boyan chain

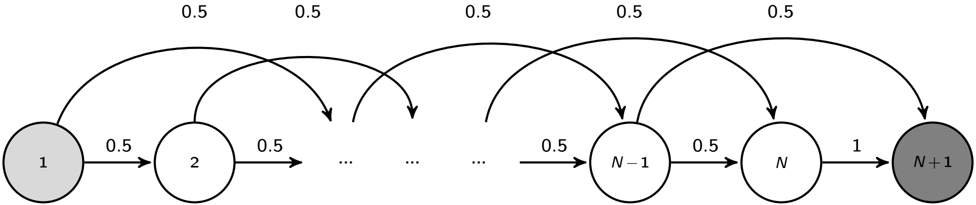
For the **Boyan chain**, the starting state is 1. It has a reward of -1 for every step except for the terminal state and the process terminates at state $N+1$. The terminal state has no reward. The transition probability shown in:



Examples of MDPs: Boyan chain

- By the linearity of expectation: $V(s) = \frac{1}{2}V(s+1) + \frac{1}{2}V(s+2) - 1$ for $s \leq N-1$.
- With the boundary conditions $V(N) = -1$ and $V(N+1) = 0$.

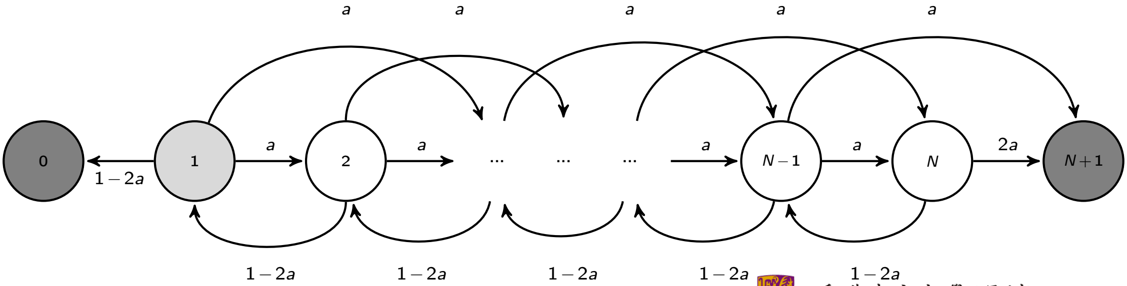
We find that $V(s) = \frac{4}{9} - \frac{2}{3}(n-s+2) - \frac{4}{9}(-\frac{1}{2})^{n-s+2}$.



香港中文大學(深圳)
The Chinese University of Hong Kong, Shenzhen

Examples of MDPs: The Boyan chain variant

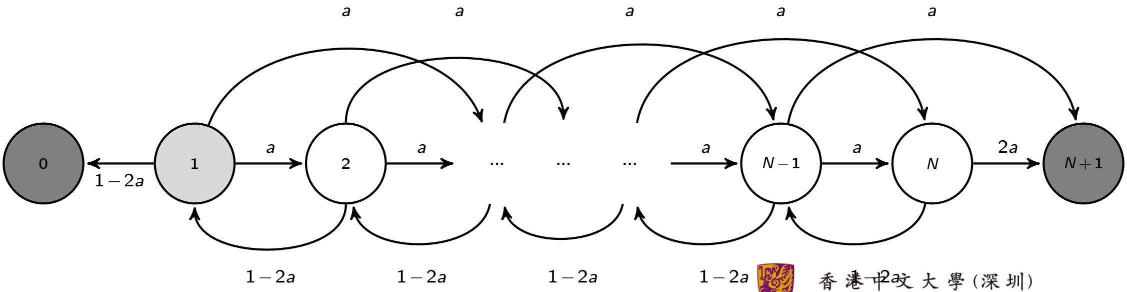
An action $a \in [0, 0.5]$ needs to be decided by the agent at every state. The starting state is 1 and the set of terminal states is $\{0, N+1\}$. The reward is -1 for every transition. and the reward at state 0 and state $N+1$ are 1 and N^2.



香港中文大學(深圳)
The Chinese University of Hong Kong, Shenzhen

Examples of MDPs: The Boyan chain variant

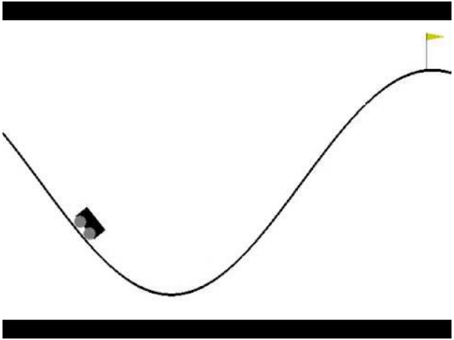
- Finding the optimal policy is hard: set $a = 0.5$ at every state.
- Finding a sub-optimal policy is relatively easy: set $a = 0$ in the starting state, the algorithm can generate a return of $-1+\gamma$.



Examples of MDPs: Mountain Car

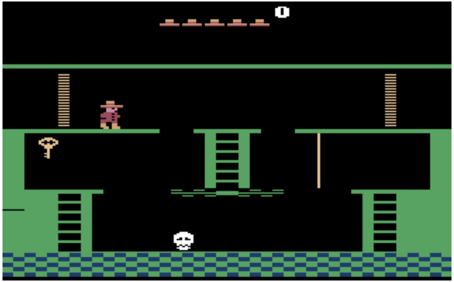
The problem describes the decision-making of a car driver who, starts from a valley and aims to drive to the mountain peak (the flag).

- The optimal policy is state-dependent: move left and then move right.



Examples of MDPs: Mountain Car

Montezuma's Revenge is one of the Atari 2600 games, which compose the Atari learning environment (ALE) commonly used in reinforcement learning tests.



- Sparse Rewards and Long Horizon: the agent has to ever touch some positive reward to receive a reinforcement signal.
- Problems in Exploration and Credit Assignment: Properly crediting the reinforce to its long sequence of actions is difficult.

香港中文大學（深圳）
The Chinese University of Hong Kong, Shenzhen

Question and Answering (Q&A)