18. Markov Decision Process

Dongwoo Kim

dongwoo.kim@postech.ac.kr

CSED515 - 2023 Spring

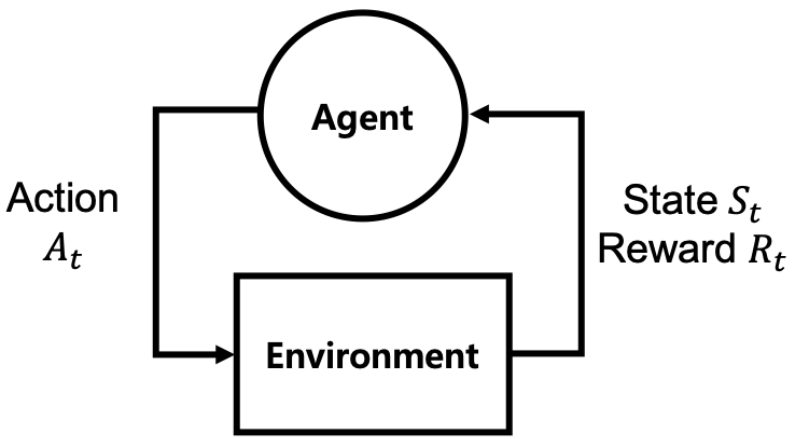
Outline

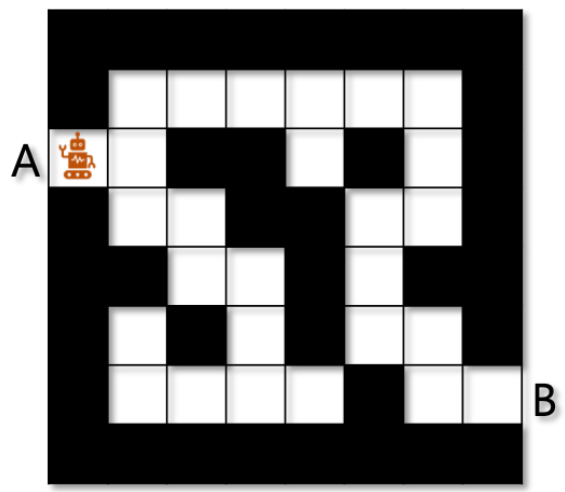
Markov Decision Process (MDP) and planning

Agent-Environment Interface

We consider discrete time, and at each step $t = 0, 1, 2, ...,$

- For given state S_t, the agent executes action A_t; receives reward R_{t+1}; and observes next state S_{t+1} of environment
- A trajectory $= (S_0, A_0, R_1, S_1, A_1, R_2, S_2, A_2, ...)$





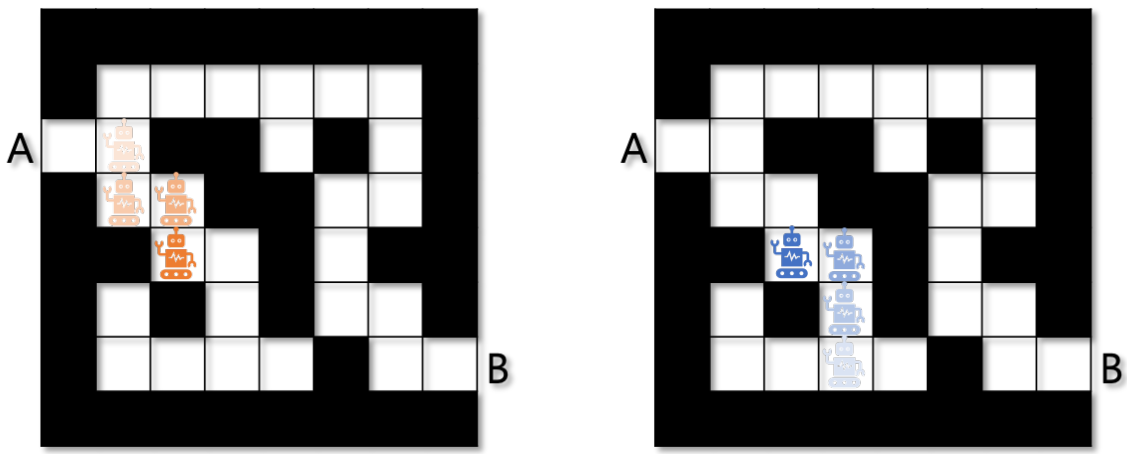
Markov Decision Process (MDP)

MDP = agent-environment interface with Markov property

- History $H_t := (S_0, A_0, R_1, S_1, A_1, ..., R_t, S_t, A_t)$
- Markov property

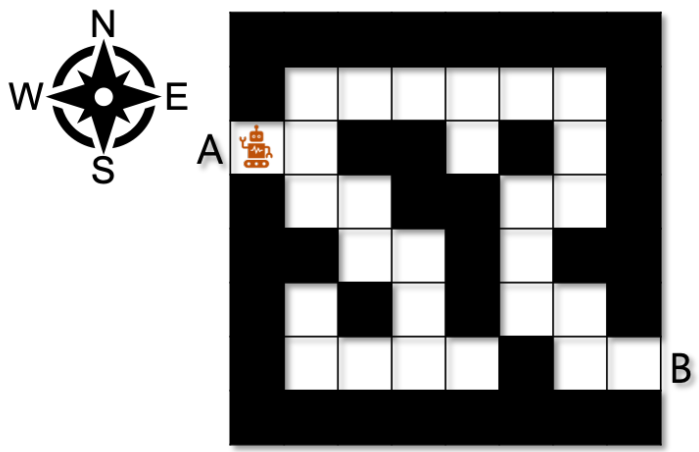
$$\mathbb{P}[S_{t+1}, R_{t+1} | H_t] = \mathbb{P}[S_{t+1}, R_{t+1} | S_t, A_t] \ .$$

- i.e., current state and action fully determine the distribution of next state and reward

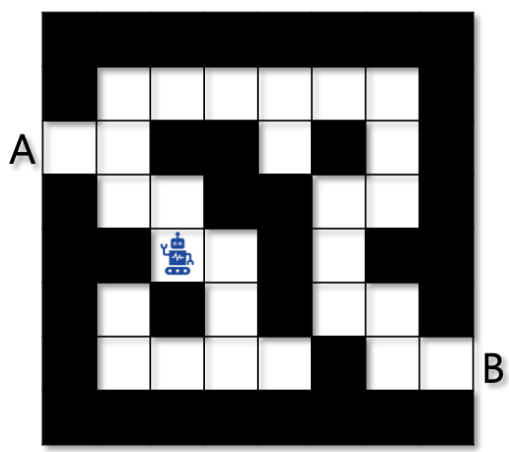


Markov Decision Process: $\langle \mathcal{S}, \mathcal{A}, \mathcal{R}, p \rangle$

- State space $\mathcal{S}$, action space $\mathcal{A}$, and reward space $\mathcal{R}$ are the set of all possible states, actions, and rewards, respectively
  - $\mathcal{A}(s) \subset \mathcal{A}$ is the set of actions feasible at state $s \in \mathcal{S}$
  - Each reward $R_t \in \mathcal{R} \subset \mathbb{R}$ is a real number
  - Each $S_t \in \mathcal{S}$ contains the information deciding what will happen at next $t+1$ given action $A_t \in \mathcal{A}(S_t)$



Feasible action = $\{E\}$



Feasible action = $\{N, E\}$

Markov Decision Process: $\langle \mathcal{S}, \mathcal{A}, \mathcal{R}, p \rangle$

Markov property

$$\begin{aligned}\mathbb{P}[S_{t+1}, R_{t+1} | H_t] &= \mathbb{P}[S_{t+1}, R_{t+1} | S_t, A_t] \\ &= p(S_{t+1}, R_{t+1} | S_t, A_t)\end{aligned}$$

where $p : \mathcal{S} \times \mathcal{R} \times \mathcal{S} \times \mathcal{A} \to [0, 1]$ is often called kernel of MDP[1]

Notations (with slight abuse)

- Transition probability

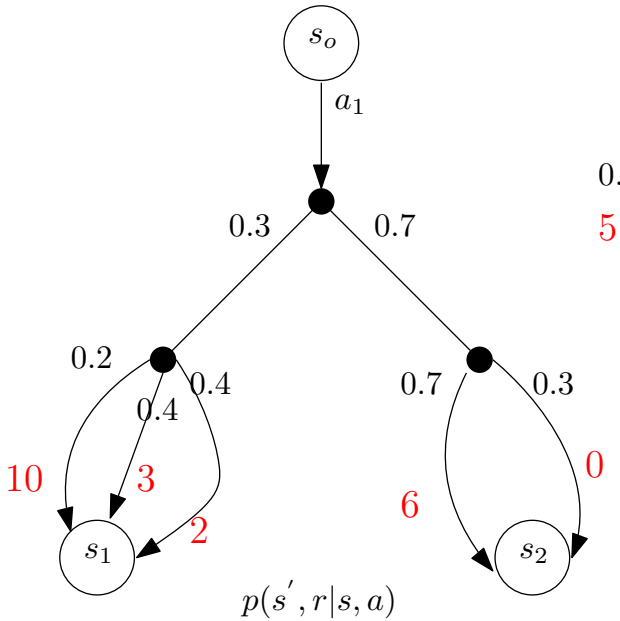
$$p(s' \mid s, a) := \mathbb{P}[S_{t+1} = s' \mid S_t = s, A_t = a] = \int_{\mathcal{R}} p(s', r \mid s, a) dr$$

- Mean reward

$$r(s, a) := \mathbb{E}[R_{t+1} \mid S_t = s, A_0 = a] = \int_{\mathcal{R}} \left(\sum_{s' \in \mathcal{S}} p(s', r \mid s, a) \right) r dr$$

[1] Throughout this lecture, we assume that we know p

Example of Kernel



0.4 : probability

5 : reward

s	a	$s^{'}$	r	$p(s^{'}, r\|s, a)$	
s_0	a_1	s_1	10	$p(s_1, 10\|s_0, a_1)$	0.06
s_0	a_1	s_1	3	$p(s_1, 3\|s_0, a_1)$	0.12
s_0	a_1	s_1	2	$p(s_1, 2\|s_0, a_1)$	0.12
s_0	a_1	s_2	0	$p(s_2, 0\|s_0, a_1)$	0.21
s_0	a_1	s_2	6	$p(s_2, 6\|s_0, a_1)$	0.49

$$\sum_r p(s^{'}, r|s, a) = p(s^{'}|s, a)$$

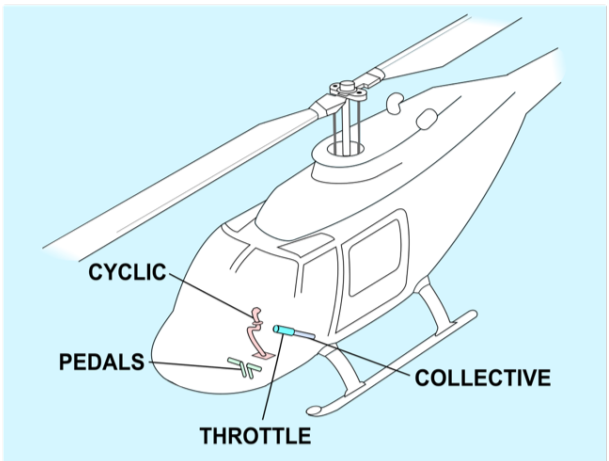
$$p(s_1|s_0, a_1) = \sum_r p(s_1, r|s_0, a_1)$$

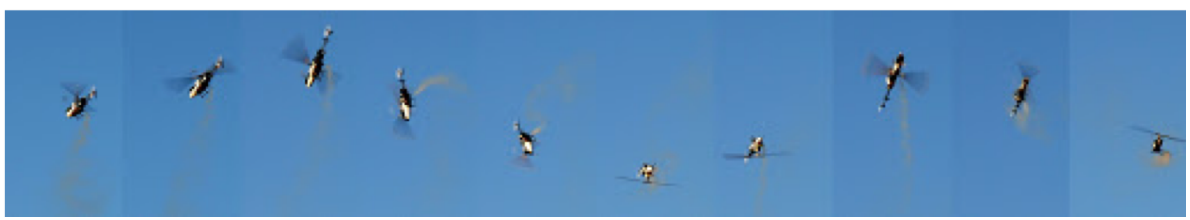
Examples of MDP

- Vehicle routing
- Helicopter control
- Game playing
- Self-driving car
- Computational finance
- Elevator scheduling
- Data center energy optimization
- ...

Example of MDP: Helicopter Control

- **Agent**: controller
- **Environment**: helicopter
- **State**: position, orientation, velocity, angular velocity, ...
- **Action**: collective pitch, cyclic pitch, pedals, throttle, ...
- **Reward**: negative of deviation from desired trajectory



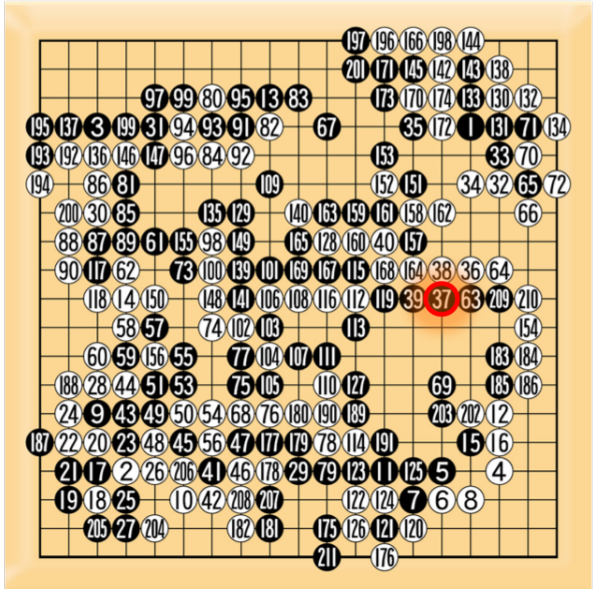


Helicopter demo [Andrew Ng, 08]

Example of MDP: Game Playing

- **Agent**: player
- **Environment**: opponent
- **State**: configuration of stones on board
- **Action**: next location of stone
- **Reward**: +1 for win and -1 for lose

AlphaGo (black) vs. Lee, Sedol (white)



AlphaGo defeats the legendary professional Go player, Sedol Lee (2016)

- AlphaGo's 37-th move in second round was critical but unexpected (a pro play wouldn't play this: odds 0.01%)
- This shows advantage of RL over supervised learning as such creative move might not be possible under supervision of pro-players

Example of MDP: Inventory Management

Consider a store selling bacon. For each week t, the store has $S_t \in \{0, 1, ..., M\}$ boxes of bacon at the end of last week, and orders A_t boxes at the beginning of this week. Let D_t denote the demand on bacon in boxes for week t. Furthermore, we know that

- The cost of maintaining an inventory of s is $h(s)$
- The cost of order a boxes is $C(a)$
- The income for selling q items is $f(q)$
- We order once in a week
- We can't store bacons more than M boxes

Example: Inventory Management

- State space: $\{0, 1, ..., M\}$
- Action space: $\mathcal{A}(s) = \{0, 1, ..., M - s\}$
- Transition: $S_{t+1} = [S_t + A_t - D_t]_+$
- Reward: $R_t = -C(A_t) - h(S_t + A_t) + f([S_t + A_t - S_{t+1}]_+)$
- For the Markov property in MDP, we may assume a time-independent conditional distribution $\mathcal{D}$ of demand D_t given current state and action

Policy

A trajectory $= (S_0, A_0, R_1, S_1, A_1, R_2, S_2, A_2, ...)$

- Kernel p describes the state transition and reward distribution (blue parts)
- A policy $\pi : \mathcal{S} \to [0, 1]^{\mathcal{A}}$ is a mapping from states to probabilities of selecting each possible action, i.e., policy π describes the selection of actions (red parts)
  - It is deterministic if $\pi : \mathcal{S} \to \{0, 1\}^{\mathcal{A}}$ (or, simply, $\pi : \mathcal{S} \to \mathcal{A}$)
  - It is stationary if it is constants for all t, c.f., non-stationary $\pi = (\pi_0, \pi_1, ...)$

Example:

	s_1	s_2	s_3
$\pi_1(a_i\|s_i)$	a_1 a_2 a_3 1.0 0 0	a_1 a_2 a_3 0.5 0.5 0	a_1 a_2 a_3 0.2 0.3 0.4
$\pi_2(a_i\|s_i)$	a_1 a_2 a_3 0 1.0 0	a_1 a_2 a_3 1/3 1/3 1/3	a_1 a_2 a_3 0 0.5 0.5

Goal in MDP

Goal: to find optimal policy maximizing cumulated reward in long run,

- Expectation of cumulated reward until reaching terminal state
- Expectation of reward per unit time, or averaged reward
- **Discounted cumulated reward**
- ...

Value Function

For discount factor $\gamma \in [0, 1]$, value of policy π starting at state s is formally defined as:

$$v_\pi(s) := \mathbb{E}_\pi \left[\sum_{k=1}^{T-t} \gamma^{k-1} R_{t+k} \mid S_t = s \right] = \mathbb{E}_\pi \left[G_t \mid S_t = s \right]$$

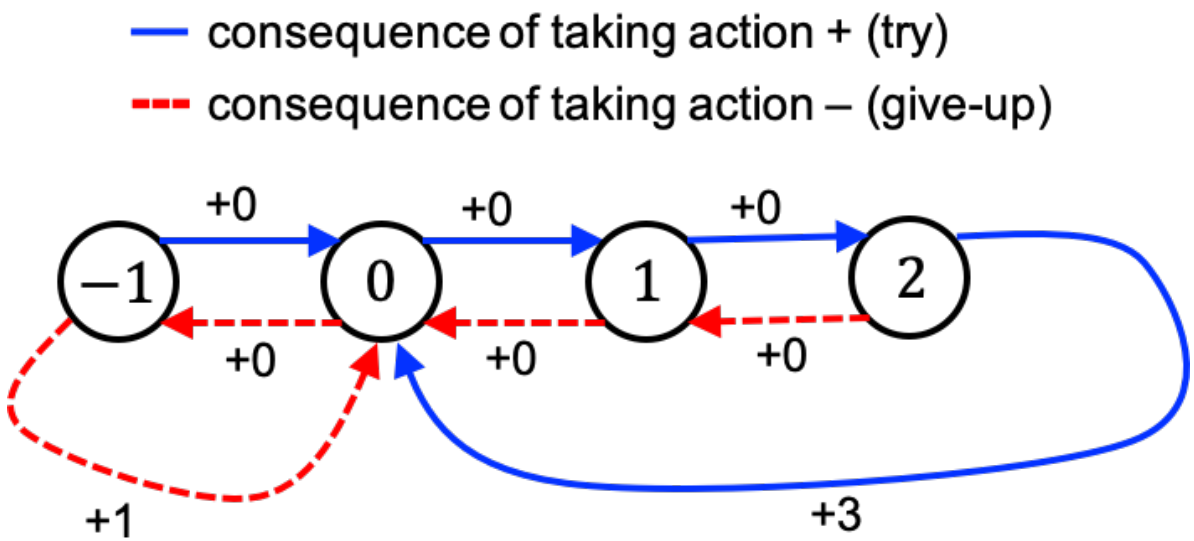
where T is time to reaching terminal state and $G_t := \sum_{k=1}^{T-t} \gamma^{k-1} R_{t+k}$ is often called return

- The value of policy depends on state s, but not t (thanks to Markov property)
- Discount factor γ quantifying the appreciation on future rewards
- Optimal policy, denoted by π_*, maximizes value at every state, i.e.,

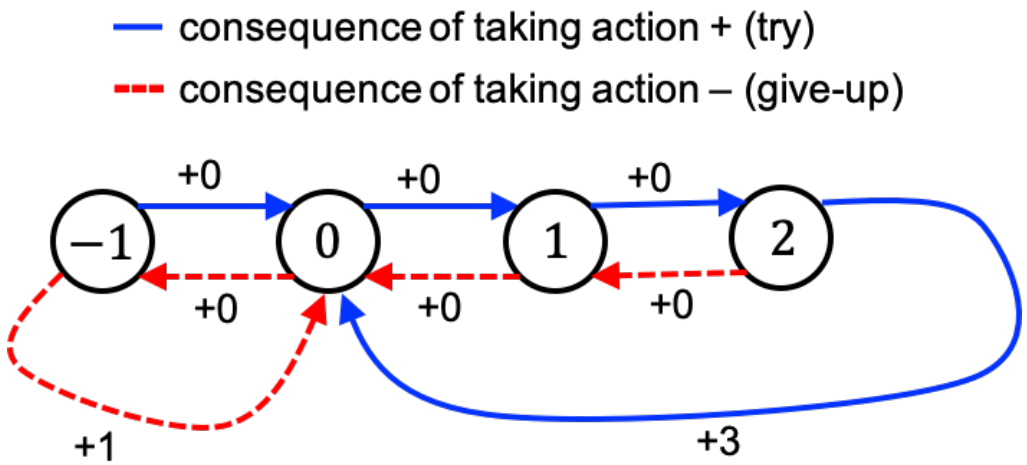
$$v_{\pi_*}(s) = \max_{\pi:\text{all policies}} v_\pi(s) \quad \forall s \in \mathcal{S}$$

Example: Life MDP

The reward $r(s, a)$ of taking action a at state s is non-zero only for $(s, a) \in \{(-1, -), (2, +)\}$, where $r(-1, -) = 1$ and $r(2, +) = 3$. The next state when taking action a at state s is the state which the corresponding arrow head is pointing at.



Example: Life MDP



- Is the optimal action $\pi_*(-1)$ at state (-1) always (-: give-up) for any discount factor $\gamma \in [0, 1)$?
- What is optimal policy with discount factor $\gamma = 0.999$?
- What is optimal policy with discount factor $\gamma = 0.001$?

Outline

Markov Decision Process (MDP) and planning

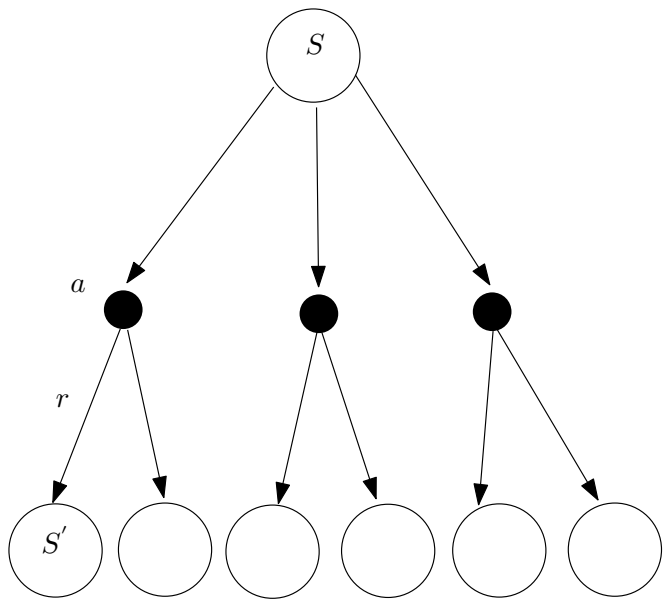
Recursion in Value Function

An interesting observation on the infinite summation in the value function: recursive relationship among $v_\pi(s)$'s: for all $s \in \mathcal{S}$,

$$
\begin{aligned}
v_\pi(s) &= \mathbb{E}_\pi \left[\sum_{k=1}^{T-t} \gamma^{k-1} R_{t+k} \mid S_t = s \right] = \mathbb{E}_\pi \left[R_{t+1} + \sum_{k=2}^{T-t} \gamma^{k-1} R_{t+k} \mid S_t = s \right] \\
&= \mathbb{E}_\pi \left[R_{t+1} + \gamma \sum_{k=1}^{T-(t+1)} \gamma^{k-1} R_{(t+1)+k} \mid S_t = s \right] \\
&= \mathbb{E}_\pi \left[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s \right] \\
&= \sum_{a \in \mathcal{A}(s)} \pi(a \mid s) \, \mathbb{E} \left[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s, A_t = a \right] \\
&= \sum_{a \in \mathcal{A}(s)} \pi(a \mid s) \sum_{s', r} p(s', r \mid s, a) \left(r + \gamma v_\pi(s') \right)
\end{aligned}
$$

Bellman Equation

$$v_\pi(s) = \sum_{a \in \mathcal{A}(s)} \pi(a \mid s) \sum_{s', r} p(s', r \mid s, a) \left(r + \gamma v_\pi(s')\right)$$



Backup diagram for v_π

Bellman Equation

The recursion among $v_\pi(s)$'s is often called Bellman equation:

$$v_\pi(s) = (\mathcal{B}^\pi v_\pi)(s) := \sum_{a \in \mathcal{A}(s)} \pi(a \mid s) \sum_{s', r} p(s', r \mid s, a) \left(r + \gamma v_\pi(s')\right) \ ,$$

where $\mathcal{B}^\pi$ is Bellman operator w.r.t. π.

In fact, Bellman equation for π is a linear equation:

$$v_\pi = \underbrace{R^\pi + \gamma P^\pi v_\pi}_{= \mathcal{B}^\pi v_\pi}$$

- Value vector $v_\pi \in \mathbb{R}^{|\mathcal{S}| \times 1}$ with $v_\pi(s)$'s
- Reward vector $R^\pi \in \mathbb{R}^{|\mathcal{S}| \times 1}$ s.t. $R^\pi(s) = \sum_a \pi(a \mid s) r(s, a)$
- Transition matrix $P^\pi \in \mathbb{R}^{|\mathcal{S}| \times |\mathcal{S}|}$ s.t. $P^\pi(s, s') = \sum_a \pi(a|s) p(s'|s, a)$

Solving Linear Equation

The value function v_π can be obtained from solving a linear equation

$$v_\pi = R^\pi + \gamma P^\pi v_\pi$$

- Matrix inversion: $v_\pi = (I - \gamma P^\pi)^{-1} R^\pi$
- Iterative policy evaluation, e.g., Richardson iteration repeating

$$V_{k+1} \leftarrow R^\pi + \gamma P^\pi V_k = \mathcal{B}^\pi V_k$$

where one can prove that $(I - \gamma P^\pi)$ is invertible, c.f., Perron–Frobenius theorem, and this implies that the value function is unique and well-defined

Contraction Mapping

Lemma

$\mathcal{B}^\pi$ s.t. $\mathcal{B}^\pi V = R^\pi + \gamma P^\pi V$ for a vector V is a contraction mapping, in particular, for any $V, V' \in \mathbb{R}^{|\mathcal{S}|\times 1}$

$$\|\mathcal{B}^\pi V' - \mathcal{B}^\pi V\|_\infty \leq \gamma \|V' - V\|_\infty \ ,$$

where $\|V\|_\infty := \max_s V(s)$.

Proof)

$$\begin{aligned}
\|\mathcal{B}^\pi V' - \mathcal{B}^\pi V\|_\infty &= \|R^\pi + \gamma P^\pi V' - R^\pi - \gamma P^\pi V\|_\infty \\
&= \|\gamma P^\pi (V' - V)\|_\infty \\
&\leq \gamma \|P^\pi\|_\infty \|(V' - V)\|_\infty && (\because \|VV'\| \leq \|V\|\|V'\|) \\
&= \gamma \|V' - V\|_\infty && (\because \sum_{s'} P^\pi(s, s') = 1 \ \forall s)
\end{aligned}$$

□

Convergence of Iterative Policy Evaluation

The contraction implies the convergence of the iterative policy evaluation to v_π:

$$V_{k+1} \leftarrow \mathcal{B}^\pi V_k$$

Proof) For any k,

$$\begin{aligned}
\|V_k - v_\pi\|_\infty &= \|\mathcal{B}^\pi V_{k-1} - \mathcal{B}^\pi v_\pi\|_\infty \\
&\leq \gamma \|V_{k-1} - v_\pi\|_\infty = \|\mathcal{B}^\pi V_{k-2} - \mathcal{B}^\pi v_\pi\|_\infty \\
&\leq \gamma^2 \|V_{k-2} - v_\pi\|_\infty = \|\mathcal{B}^\pi V_{k-3} - \mathcal{B}^\pi v_\pi\|_\infty \\
&\ldots \\
&\leq \gamma^k \|V_0 - v_\pi\|_\infty \ ,
\end{aligned}$$

where the inequalities are from the contraction. This implies $\|V_k - v_\pi\|_\infty \to 0$ as $k \to \infty$, i.e., $\lim_{k\to\infty} V_k = v_\pi$.

Unique Value Function

The contraction also implies the uniqueness of value function, i.e.,

V verifying Bellman equation for π iff V is value of π, i.e., $V = v_\pi$.

Proof by contradiction) Suppose there exist different V, V' verifying Bellman equation for π, i.e., $V = \mathcal{B}^\pi V$ and $V' = \mathcal{B}^\pi V'$. Then,

$$\begin{aligned} \|V - V'\|_\infty &= \|\mathcal{B}^\pi V - \mathcal{B}^\pi V'\|_\infty && (\because \text{supposition}) \\ &\leq \gamma \|V - V'\|_\infty \,, && (\because \text{contraction}) \end{aligned}$$

which implies $1 \leq \gamma$ due to the supposition that V, V' are different, i.e., $\|V - V'\|_\infty > 0$, and completes the proof by contradiction as $\gamma < 1$. $\square$

Outline

Markov Decision Process (MDP) and planning

Optimal Policy

- Our goal is to find an optimal policy π_* which maximizes the value function at every state, i.e., π_* is optimal iff for any policy π and state $s \in \mathcal{S}$,

$$v_{\pi_*}(s) \geq v_\pi(s) \ .$$

- The problem of finding optimal policy is often called control problem
- As we know how to evaluate a policy, an optimal policy can be obtained by comparing all possible policies in terms of value function
- However, this is intractable as the number of possible deterministic policies is $O(|\mathcal{A}|^{|\mathcal{S}|})$

Action-value Function a.k.a. Q-function

- Let $q_\pi(s, a)$ be the action-value of policy π with a perturbation selecting action a at the beginning state s, i.e.,

$$\underbrace{S_t = s, A_t = a, R_{t+1}, S_{t+1}}_{\text{perturbation}}, \underbrace{A_{t+1}, R_{t+2}, S_{t+2}, A_{t+2}, R_{t+3}, S_{t+3}, \cdots}_{\text{following } \pi}$$

$$\begin{aligned} q_\pi(s, a) &:= \mathbb{E}_\pi[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s, A_t = a] \\ &= \sum_{s', r} p(s', r \mid s, a)(r + \gamma v_\pi(s')) \end{aligned}$$

- Then, $q_\pi(s, a) > v_\pi(s)$ implies that it is better to select action a at state s than to follow $\pi(\cdot \mid s)$

Policy Iteration

Policy improvement: $\pi(s) \leftarrow \arg\max_a \underbrace{\sum_{s',r} p(s', r \mid s, a)(r + \gamma V(s'))}_{:=q_\pi(s,a)}$

Policy Iteration (using iterative policy evaluation) for estimating $\pi \approx \pi_*$

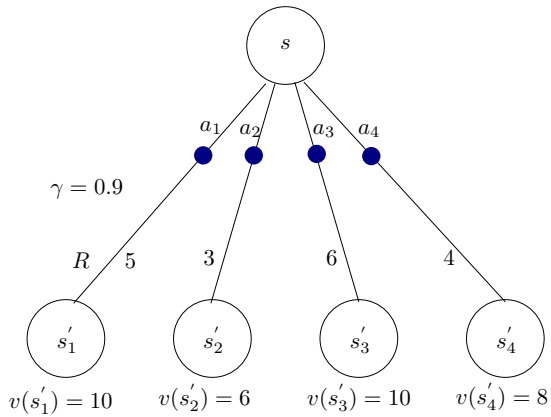
1. Initialization
   $V(s) \in \mathbb{R}$ and $\pi(s) \in \mathcal{A}(s)$ arbitrarily for all $s \in \mathcal{S}$

2. Policy Evaluation
   Loop:
   $\Delta \leftarrow 0$
   Loop for each $s \in \mathcal{S}$:
   $v \leftarrow V(s)$
   $V(s) \leftarrow \sum_{s',r} p(s', r|s, \pi(s))\left[r + \gamma V(s')\right]$
   $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
   until $\Delta < \theta$ (a small positive number determining the accuracy of estimation)

3. Policy Improvement
   policy-stable $\leftarrow$ *true*
   For each $s \in \mathcal{S}$:
   old-action $\leftarrow \pi(s)$
   $\pi(s) \leftarrow \arg\max_a \sum_{s',r} p(s', r|s, a)\left[r + \gamma V(s')\right]$
   If *old-action* $\neq \pi(s)$, then *policy-stable* $\leftarrow$ *false*
   If *policy-stable*, then stop and return $V \approx v_*$ and $\pi \approx \pi_*$; else go to 2

Example of Policy Iteration

Policy π is improved to π':



$p(s^{'}, r|s, a) = 1$ for these four cases.

$$
\begin{aligned}
q_\pi(s, a_1) : 5 + 0.9 \times 10 &= 14 \\
q_\pi(s, a_2) : 3 + 0.9 \times 6 &= 8.4 \\
q_\pi(s, a_3) : 6 + 0.9 \times 10 &= 15 \\
q_\pi(s, a_4) : 4 + 0.9 \times 8 &= 11.2
\end{aligned}
$$

Therefore,$\pi^{'}(s) = a_3$. (greedy method)

Monotonic Improvement

Let π_k, V_k, and Q_k be the greedy policy, value function, and action-value function, resp. in the k-th iteration of the policy iteration

Lemma (Monotone)

For the successive value functions V_k and V_{k+1},

$$V_{k+1} \geq V_k \ , \quad i.e., V_{k+1}(s) \geq V_k(s) \quad \forall s \in \mathcal{S} \ .$$

Proof of Monotonic Improvement

Proof) For any $s \in \mathcal{S}$,

$$
\begin{aligned}
V_k(s) &= \sum_a \pi_k(a \mid s) Q_k(s,a) \\
&\leq \max_a Q_k(s,a) \\
&= \sum_a \pi_{k+1}(a \mid s) \sum_{s',r} p(s', r \mid s, a)(r + \gamma \underbrace{V_k(s')}_{\leq (\mathcal{B}^{\pi_{k+1}} V_k)(s')}) = (\mathcal{B}^{\pi_{k+1}} V_k)(s) \\
&\leq (\mathcal{B}^{\pi_{k+1}} \mathcal{B}^{\pi_{k+1}} V_k)(s) \ldots
\end{aligned}
$$

where the last equality is from the definition of $\mathcal{B}^{\pi_{k+1}}$. Hence,

$$
V_k \leq \mathcal{B}^{\pi_{k+1}} V_k \leq (\mathcal{B}^{\pi_{k+1}})(\mathcal{B}^{\pi_{k+1}}) V_k \ldots \leq \lim_{n \to \infty} (\mathcal{B}^{\pi_{k+1}})^n V_k = V_{k+1}
$$

where the last equality is from the fact that the Bellman operator is contraction.

□

Convergence of Policy Iteration

Theorem

Policy iteration eventually finds optimal policy π_* s.t.

$$v_*(s) := v_{\pi_*}(s) = \max_{\pi} v_\pi(s) \ \forall s \in \mathcal{S} \ .$$

Proof sketch)

- To show convergence to the optimal policy, along with monotone improvement, we need to show that if there is no improvement in the value function at any state, then we are at optimality.
- The proof sketch is as follows. We consider k such that $V^{\pi_{k+1}}(s) = V^{\pi_k}(s), \forall s \in S$. We can show that such V^{π_k} satisfies the Bellman optimality equation, and hence $V^{\pi_k} = V^*$.

□

Value Iteration

- One complaint on policy iteration is that it requires the evaluation of policies many times
- However, do we really need to wait the complete evaluation of policies?
- Anyway, we want to find a solution of the optimal Bellman equation: for all $s \in \mathcal{S}$,

$$V(s) = (\mathcal{B}^* V)(s) := \max_a \sum_{s'} \sum_r p(s', r \mid s, a)(r + \gamma V(s'))$$

  where $\mathcal{B}^*$ is the optimal Bellman operator.
- Value iteration directly uses the optimal Bellman equation so that an iteration doesn't need to wait the convergence of policy evaluation.

Value Iteration

Value Iteration, for estimating $\pi \approx \pi_*$

Algorithm parameter: a small threshold $\theta > 0$ determining accuracy of estimation
Initialize $V(s)$, for all $s \in \mathcal{S}^+$, arbitrarily except that $V(terminal) = 0$

Loop:
| $\Delta \leftarrow 0$
| Loop for each $s \in \mathcal{S}$:
| $\quad v \leftarrow V(s)$
| $\quad V(s) \leftarrow \max_a \sum_{s',r} p(s', r \mid s, a)\big[r + \gamma V(s')\big]$
| $\quad \Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$

Output a deterministic policy, $\pi \approx \pi_*$, such that
$\pi(s) = \operatorname{argmax}_a \sum_{s',r} p(s', r \mid s, a)\big[r + \gamma V(s')\big]$

[from Sutton and Barto]

Convergence of Value Iteration

Lemma

$\mathcal{B}^*$ is a contraction mapping, in particular, for any $V, V' \in \mathbb{R}^{|\mathcal{S}|\times 1}$

$$\|\mathcal{B}^* V' - \mathcal{B}^* V\|_\infty \leq \gamma \|V' - V\|_\infty ,$$

where $\|V\|_\infty := \max_s V(s)$.

- Again, the contraction also implies the uniqueness of optimal value function, i.e.,

$$v_* = \mathcal{B}^* v_* \text{ iff } v_*(s) = \max_\pi v_\pi(s) \ \forall s \in \mathcal{S}.$$

- In addition, the contraction provides the convergence of π_n in value iteration to π_* as $\theta \to 0$

Summary

- Markov decision process is a popular model for the agent-environment interface with Markov property (for tractability/simplicity)
- A policy is a mapping from states to action distribution
- Best policy varies depending on the definition of value functions
- Bellman equation (from Markov property) provides tractable methods to compute value functions, e.g., iterative policy evaluation
- Policy iteration and value iteration with greedy improvement always find the optimal policy of MDP
- Note that such an efficient search of optimal policy, i.e., greedy improvement, is possible thanks to Markov property of MDP