

Reinforcement Learning

Lecture 2. Model-based Methods

Sungjoon Choi, Korea University

Markov Process

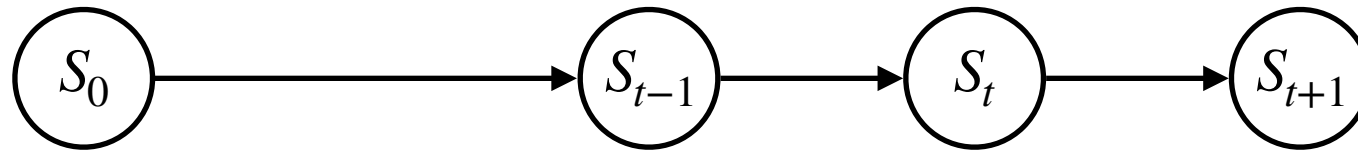
Introduction



When can we formulate our problem as reinforcement learning?

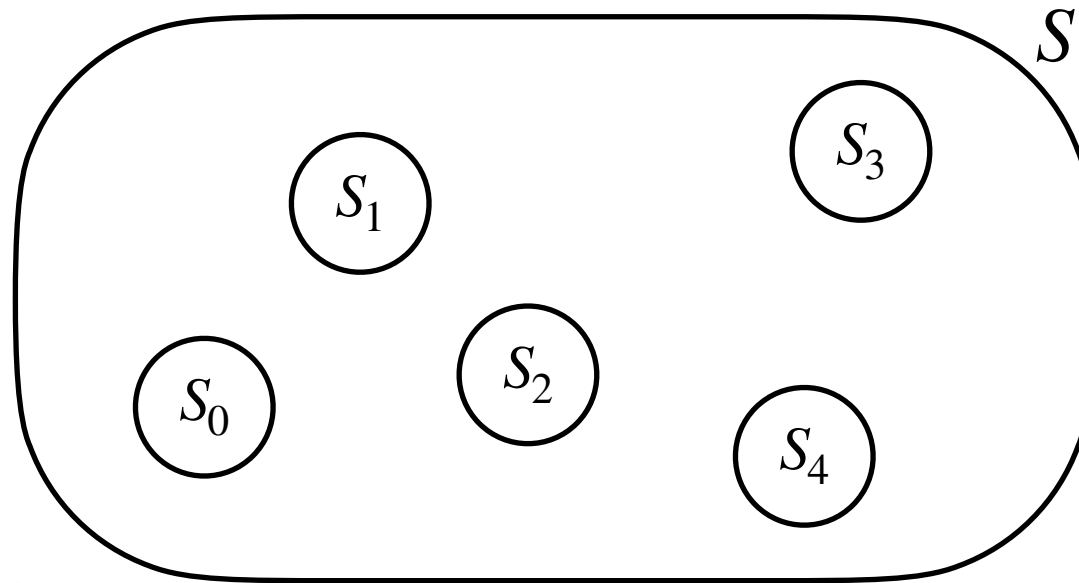
Markov Process

- A **Markov chain** is a stochastic model describing a sequence of possible events in which the probability of each event depends only on the state attained in the previous event (aka **Markov** property).
- Random variable: S_t (state)



Markov Process

- State space
 - All possible state values



- Random variable: S_t
- Outcome: $S_t = s_t$

Markov Property



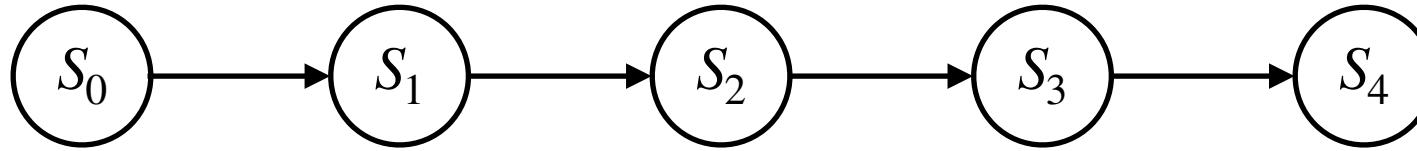
Andrey Markov (1856-1922)

- "Markov" (or Markov property) means that given the present state, the future and the past are independent.

$$P(\underbrace{S_{t+1} = s'}_{\text{future}} | \underbrace{S_t = s_t}_{\text{present}}, \underbrace{S_{t-1} = s_{t-1}, \dots, S_0 = s_0}_{\text{past}}) = P(\underbrace{S_{t+1} = s'}_{\text{future}} | \underbrace{S_t = s_t}_{\text{present}})$$

Markov Process

- Given a state sequence $\{s_0, s_1, s_2, s_3, s_4\}$, how **plausible** is this?



$$P(S_0 = s_0, S_1 = s_1, S_2 = s_2, S_3 = s_3, S_4 = s_4)$$

$$= p(s_0, s_1, s_2, s_3, s_4)$$

$$= p(s_4 | s_3, s_2, s_1, s_0) p(s_3 | s_2, s_1, s_0) p(s_2 | s_1, s_0) p(s_1 | s_0) p(s_0)$$

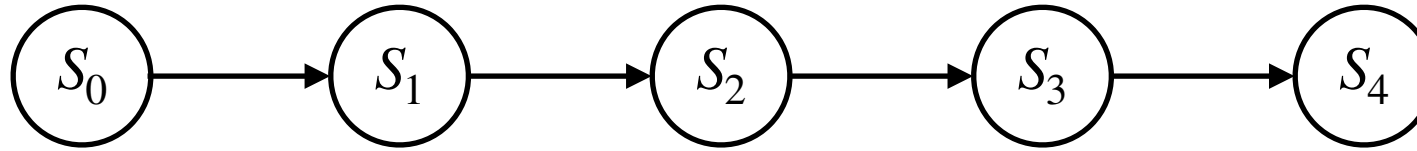
$$= p(s_4 | s_3) p(s_3 | s_2) p(s_2 | s_1) p(s_1 | s_0) p(s_0)$$

Chain Rule

Markov Property

Markov Process

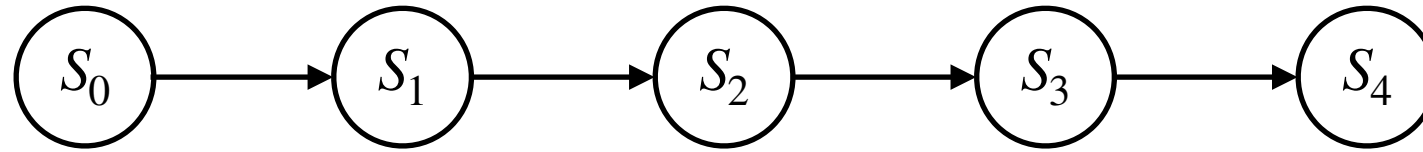
- Given a state sequence $\{s_0, s_1, s_2, s_3, s_4\}$, how **plausible** is this?



$$\begin{aligned} P(S_0 = s_0, S_1 = s_1, S_2 = s_2, S_3 = s_3, S_4 = s_4) \\ &= p(s_0, s_1, s_2, s_3, s_4) \\ &= p(s_4 | s_3, s_2, s_1, s_0) p(s_3 | s_2, s_1, s_0) p(s_2 | s_1, s_0) p(s_1 | s_0) p(s_0) \\ &= p(s_4 | s_3) p(s_3 | s_2) p(s_2 | s_1) p(s_1 | s_0) p(s_0) \end{aligned}$$

Transition Probability
Initial Probability

Markov Process



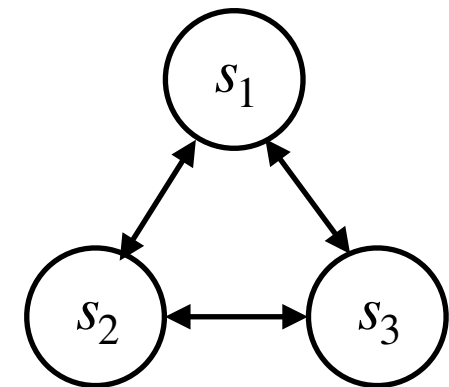
- A **Markov process** is specified by the **initial probability** and the **transition probabilities**.
 - Initial probability
 - $P(S_0 = s)$
 - Transition probability
 - $P(S_{t+1} = s' | S_t = s)$

Markov Process

- Initial distribution vector

- $d_i = P(S_0 = s_i)$

$$\mathbf{d} = \begin{matrix} S_1 \\ S_2 \\ S_3 \end{matrix} \begin{bmatrix} P(S_0 = s_1) \\ P(S_0 = s_2) \\ p(S_0 = s_3) \end{bmatrix}$$

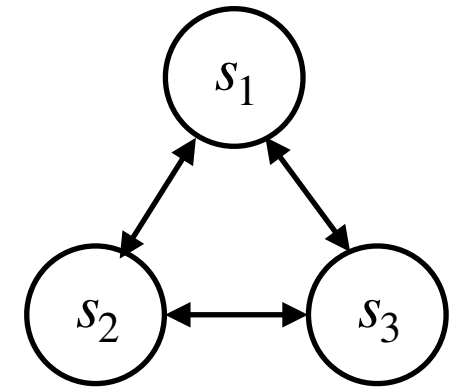


Markov Process

- Transition probability matrix

- $P_{(i,j)} = P(S_{t+1} = s_j | S_t = s_i)$

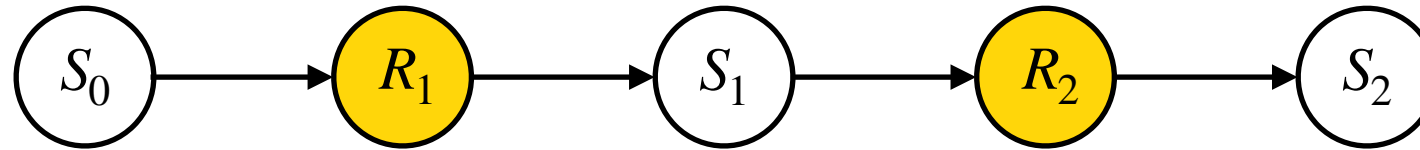
$$P = \begin{array}{c} \begin{array}{c} S_1 \\ S_2 \\ S_3 \end{array} \begin{array}{|c|c|c|} \hline S_1 & S_2 & S_3 \\ \hline p(s_1 | s_1) & p(s_2 | s_1) & p(s_3 | s_1) \\ \hline p(s_1 | s_2) & p(s_2 | s_2) & p(s_3 | s_2) \\ \hline p(s_1 | s_3) & p(s_2 | s_3) & p(s_3 | s_3) \\ \hline \end{array} \end{array}$$



Markov **Reward** Process

Markov Reward Process

- Now, we obtain rewards as we move to states.



- We have two random variables.
 - State: S_t
 - Reward: R_t
- Since the reward is a random variable, we take expectation to compute the reward function.
 - Reward function: $r(s) = \mathbb{E}[R_{t+1} | S_t = s]$

Return

- **Return** G_t is the (discounted) sum of future rewards, and hence, also a random variable.

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots = \sum_{k=0}^{\infty} \gamma^k R_{t+1+k}$$

- Discount factor γ :



1

Worth Now



γ

Worth Next Step

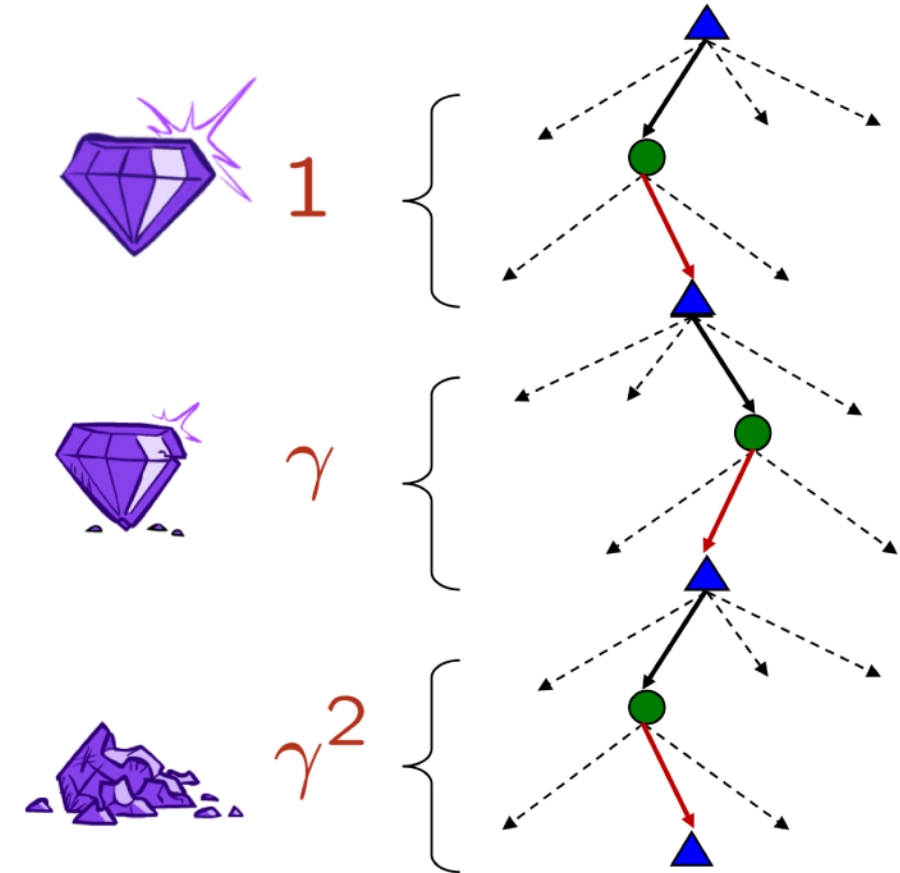


γ^2

Worth In Two Steps

Discount Factor

- How to **discount**?
 - Each time we descend a level, we multiply the d
- Why **discount**?
 - Sooner rewards may have higher utility than late
 - It also helps the algorithms to converge.



(State) Value Function

- The **state-value function** $V(s)$ is a function of a state and is the expected return starting from the state s .

$$\begin{aligned} V(s) &= \mathbb{E}[G_t | S_t = s] \\ &= \mathbb{E}[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} \cdots | S_t = s] \end{aligned}$$

(State) Value Function

- The (state) **value function** is the expected return (discounted sum of rewards) starting from state s .

$$\begin{aligned}\mathbb{E}[G_t | S_t = s] &= \mathbb{E}[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} \cdots | S_t = s] \\ &= \mathbb{E}[R_{t+1} + \gamma(R_{t+2} + \gamma R_{t+3} + \cdots) | S_t = s] \\ &= \mathbb{E}[R_{t+1} + \gamma G_{t+1} | S_t = s]\end{aligned}$$

Recursive equation

(State) Value Function

$$\begin{aligned} V(s) &= \mathbb{E}[G_t | S_t = s] = \mathbb{E}[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots | S_t = s] \\ &= \mathbb{E}[R_{t+1} + \gamma G_{t+1} | S_t = s] \\ &= \mathbb{E}[R_{t+1} + \gamma \mathbb{E}[G_{t+1} | S_{t+1}] | S_t = s] \\ &= \mathbb{E}[R_{t+1} + \gamma V(S_{t+1}) | S_t = s] \\ &= \mathbb{E}[R_{t+1} | S_t = s] + \gamma \mathbb{E}[V(S_{t+1}) | S_t = s] \\ &= r(s) + \gamma \sum_{s'} V(s') P(s' | s) \end{aligned}$$

Bellman Equation for Markov Reward Processes



$$V(s) = r(s) + \gamma \sum_{s'} V(s')P(s' | s)$$

Bellman Equation for Markov Reward Processes

$$V(s) = r(s) + \gamma \sum_{s'} V(s') P(s' | s)$$

Diagram illustrating the Bellman Equation for Markov Reward Processes, with components labeled:

- $V(s)$: Current Value
- $r(s)$: Current Reward
- γ : Discounted
- $V(s')$: Next Value
- $P(s' | s)$: Transition Probability

Bellman Equation for Markov Reward Processes

$$V(s) = r(s) + \gamma \sum_{s'} V(s') P(s' | s)$$



For multiple states s_1, s_2, s_3

$$V(s_1) = r(s_1) + \gamma (V(s_1)P(s_1 | s_1) + V(s_2)P(s_2 | s_1) + V(s_3)P(s_3 | s_1))$$

$$V(s_2) = r(s_2) + \gamma (V(s_1)P(s_1 | s_2) + V(s_2)P(s_2 | s_2) + V(s_3)P(s_3 | s_2))$$

$$V(s_3) = r(s_3) + \gamma (V(s_1)P(s_1 | s_3) + V(s_2)P(s_2 | s_3) + V(s_3)P(s_3 | s_3))$$

Bellman Equation for Markov Reward Processes

$$V(s) = r(s) + \gamma \sum_{s'} V(s') P(s' | s)$$

\downarrow (matrix form)

$$V = R + \gamma P V$$



$$V = R + \gamma P V$$

Bellman Equation for Markov Reward Processes



- The Bellman equation is a **linear equation**!

$$V = R + \gamma P V$$



Solve the equation w.r.t V

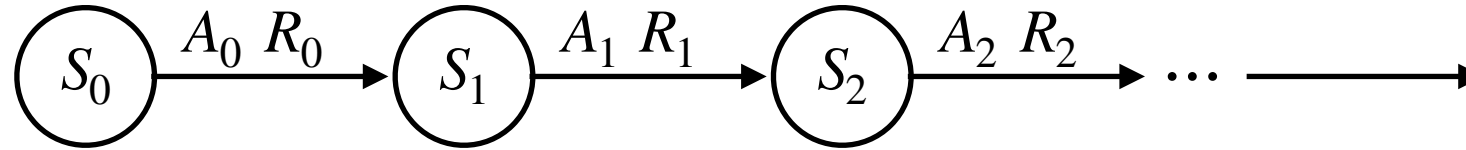
$$V = (I - \gamma P)^{-1} R$$

Markov Reward Process

- Summary
 - S is a finite set of states.
 - d is an initial state distribution.
 - P is a state transition probability matrix.
 - $P_{ss'} = P(S_{t+1} = s' | S_t = s)$
 - r is a reward function.
 - $r(s) = \mathbb{E} [R_{t+1} | S_t = s]$
 - Note that there is no notion of **action** here.

Markov Decision Process

Markov Decision Process



- Now, we have three random variables:
 - State: S_t
 - Reward: R_t
 - Action: A_t

Markov Decision Process

- Formally, an MDP is a tuple (S, A, P, R, d) :
 - A set of states $s \in S$.
 - A set of actions $a \in A$
 - A state transition function (or matrix)
 - $P(s' | s, a) = P(S_{t+1} = s' | S_t = s, A_t = a)$
 - $P_{sas'} = P(s' | s, a)$
 - A reward function
 - $r(s) = \mathbb{E}[R_{t+1} | S_t = s]$
 - It depends on both state and action.
 - An initial state distribution d

Policy

- A **policy** is a **distribution over actions** given state.

$$\pi(a | s) = P(A_t = a | S_t = s)$$

- In an MDP, a policy is a function of the current state s .
- A policy is stationary (time-independent).
- A deterministic policy can also be represented by a distribution.

Policy

- Given a fixed policy, a **Markov decision process** becomes and a **Markov reward process**.
- We denote $P_{ss'}^\pi$ a policy-conditioned state transition probability matrix:

$$\begin{aligned} P_{ss'}^\pi &= \sum_a P(S_{t+1} = s' | S_t = s, A_t = a) \pi(A_t = a | S_t = s) \\ &= \sum_a \pi(a | s) P_{sas'} \end{aligned}$$

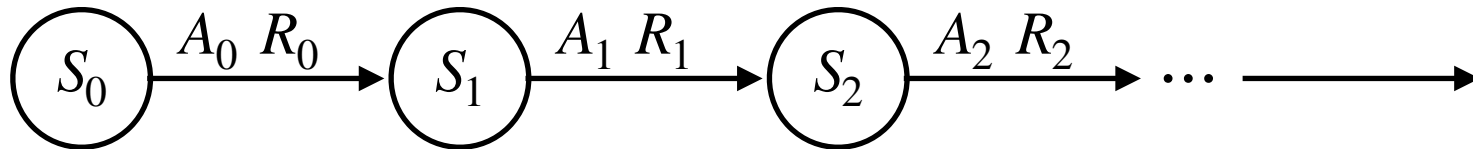
- Similarly, we can compute a policy-conditioned reward function

$$r^\pi(s) = \mathbb{E}[R_{t+1} | S_t = s] = \sum_a r(s, a) \pi(a | s)$$

Return

- Given a policy π , return G_t is determined.

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots = \sum_{k=0}^{\infty} \gamma^k R_{t+1+k}$$



State Value and State Action Value of Policy



- Given a fixed policy π
 - State value function, $V_\pi(s)$, is the expected return starting from state s , and then following policy

$$V_\pi(s) = \mathbb{E}[G_t | S_t = s]$$

- State-action value function, $Q_\pi(s, a)$, is the expected return starting from state s and action a , and then following policy

$$Q_\pi(s, a) = \mathbb{E}[G_t | S_t = s, A_t = a]$$

Bellman Equation (1)

- Given a fixed policy π

$$V_{\pi}(s) = \sum_a \pi(a | s) Q_{\pi}(s, a)$$

Relation between a **state value function**, $V_{\pi}(s) = \mathbb{E}[G_t | S_t = s]$, and a **state-action value function**. $Q_{\pi}(s, a) = \mathbb{E}[G_t | S_t = s, A_t = a]$.

Bellman Equation (2)

$$V_{\pi}(s) = \mathbb{E}[R_{t+1} + \gamma V(S_{t+1}) | S_t = s]$$

Bellman equation of
a Markov reward process

$$= r(s) + \sum_{s'} V_{\pi}(s') P_{\pi}(s' | s)$$

Definition of reward and
transition probability

$$= \sum_a r(s, a) \pi(a | s) + \sum_{s'} V_{\pi}(s') \sum_a \pi(a | s) P(s' | s, a)$$

Simple rearrange

$$= \sum_a \left[r(s, a) + \sum_{s'} V_{\pi}(s') P(s' | s, a) \right] \pi(a | s)$$

$$\begin{aligned} V(s) &= \mathbb{E}[G_t | S_t = s] = \mathbb{E}[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots | S_t = s] \\ &= \mathbb{E}[R_{t+1} + \gamma G_{t+1} | S_t = s] \\ &= \mathbb{E}[R_{t+1} + \gamma \mathbb{E}[G_{t+1} | S_{t+1}] | S_t = s] \\ &= \mathbb{E}[R_{t+1} + \gamma V(S_{t+1}) | S_t = s] \\ &= \mathbb{E}[R_{t+1} | S_t = s] + \gamma \mathbb{E}[V(S_{t+1}) | S_t = s] \\ &= r(s) + \gamma \sum_{s'} V(s') P(s' | s) \end{aligned}$$

Bellman Equation (3)

- Similarly, for $Q_\pi(s, a)$,

$$\begin{aligned} Q_\pi(s, a) &= \mathbb{E}[R_{t+1} + \gamma G_{t+1} | S_t = s, A_t = a] \\ &= \mathbb{E}[R_{t+1} | S_t = s, A_t = a] + \gamma \mathbb{E}[G_{t+1} | S_t = s, A_t = a] \\ &= r(s, a) + \gamma \mathbb{E}[V(S_{t+1}) | S_t = s, A_t = a] \\ &= r(s, a) + \gamma \sum_{s'} V(s') P(s' | s, a) \end{aligned}$$

Bellman Equations of an MDP

- Summary

$$V_{\pi}(s) = \sum_a \pi(a | s) Q_{\pi}(s, a)$$

$$V_{\pi}(s) = \sum_a \left[r(s, a) + \sum_{s'} V_{\pi}(s') P(s' | s, a) \right] \pi(a | s)$$

$$Q_{\pi}(s, a) = r(s, a) + \gamma \sum_{s'} V(s') P(s' | s, a)$$

Bellman (**Expectation**) Equation of Q_π

- State-action value function $Q_\pi(s, a) = \mathbb{E}[G_t | S_t = s, A_t = a, \pi]$
 - The expected return starting from state s , taking action a , and then follows the policy π

$$Q_\pi(s, a) = \sum_{s'} \left[r(s, a, s') + \gamma \sum_{a'} Q_\pi(s', a') \pi(a' | s') \right] P(s' | s, a)$$

Optimality

Optimality

What does it mean by **solving** an MDP?

Optimal Value and Policy

- Optimal state value

$$V^*(s) = \max_{\pi} V_{\pi}(s)$$

- Optimal state-action value

$$Q^*(s, a) = \max_{\pi} Q_{\pi}(s, a)$$

- Optimal policy

$$\pi^*(a | s) = 1 \text{ for } a = \arg \max_{a'} Q^*(s, a')$$

Bellman Optimality Equation



$$V^*(s) = \max_a Q^*(s, a)$$

$$Q^*(s, a) = \sum_{s'} [r(s, a, s') + \gamma V^*(s')] P(s' | s, a)$$

$$V^*(s) = \max_a \sum_{s'} [r(s, a, s') + \gamma V^*(s')] P(s' | s, a)$$

$$Q^*(s, a) = \sum_{s'} \left[r(s, a, s') + \gamma \max_{a'} Q^*(s', a') \right] P(s' | s, a)$$

Summary

- Bellman **Optimality** Equation
- Bellman Equation

$$V_{\pi}(s) = \sum_a \pi(a|s) Q(s, a)$$

$$Q_{\pi}(s, a) = \sum_{s'} [r(s, a, s') + \gamma V_{\pi}(s')] P(s'|s, a)$$

$$V_{\pi}(s) = \sum_a \pi(a|s) \sum_{s'} [r(s, a, s') + \gamma V_{\pi}(s')] P(s'|s, a)$$

$$Q_{\pi}(s, a) = \sum_{s'} \left[r(s, a, s') + \gamma \sum_{a'} Q_{\pi}(s', a') \pi(a'|s') \right] P(s'|s, a)$$

$$V^*(s) = \max_a Q^*(s, a)$$

$$Q^*(s, a) = \sum_{s'} [r(s, a, s') + \gamma V^*(s')] P(s'|s, a)$$

$$V^*(s) = \max_a \sum_{s'} [r(s, a, s') + \gamma V^*(s')] P(s'|s, a)$$

$$Q^*(s, a) = \sum_{s'} \left[r(s, a, s') + \gamma \max_{a'} Q^*(s', a') \right] P(s'|s, a)$$

$$\pi^*(a|s) = \arg \max_{a'} Q^*(s, a')$$

Value Iteration

Value Iteration



Given an MDP (S, A, P, R, d) , how can we **solve** an MDP?

Value Iteration

- Value iteration utilizes the **principle of optimality**.

$$V^*(s) = \max_{\pi} V_{\pi}(s)$$

- Then, the solution $V^*(s)$ can be found by one-step lookahead

$$V^*(s) = \max_a \sum_{s'} [r(s, a, s') + \gamma V^*(s')] P(s' | s, a)$$

- The main idea is to apply these updates iteratively, repeat until convergence.
 - It is guaranteed to converge to the unique optimal value.
- However, there is no explicit policy.

- Bellman **Optimality** Equation

$$V^*(s) = \max_a Q^*(s, a)$$

$$Q^*(s, a) = \sum_{s'} [r(s, a, s') + \gamma V^*(s')] P(s' | s, a)$$

$$V^*(s) = \max_a \sum_{s'} [r(s, a, s') + \gamma V^*(s')] P(s' | s, a)$$

$$Q^*(s, a) = \sum_{s'} \left[r(s, a, s') + \gamma \max_{a'} Q^*(s', a') \right] P(s' | s, a)$$

$$\pi^*(a | s) = \arg \max_{a'} Q^*(s, a')$$

Value Iteration

- Start from a random initial V_0
- Initialize V_{k+1} with $-\infty$
- Loop

- For all states $s \in S$:

- For all actions $a \in A$:

- For all next states $s' \in S$:

- If $V_{k+1}(s) \leq [r(s, a, s') + \gamma V_k(s')]P(s' | s, a)$

- $V_{k+1}(s) = [r(s, a, s') + \gamma V_k(s')]P(s' | s, a)$

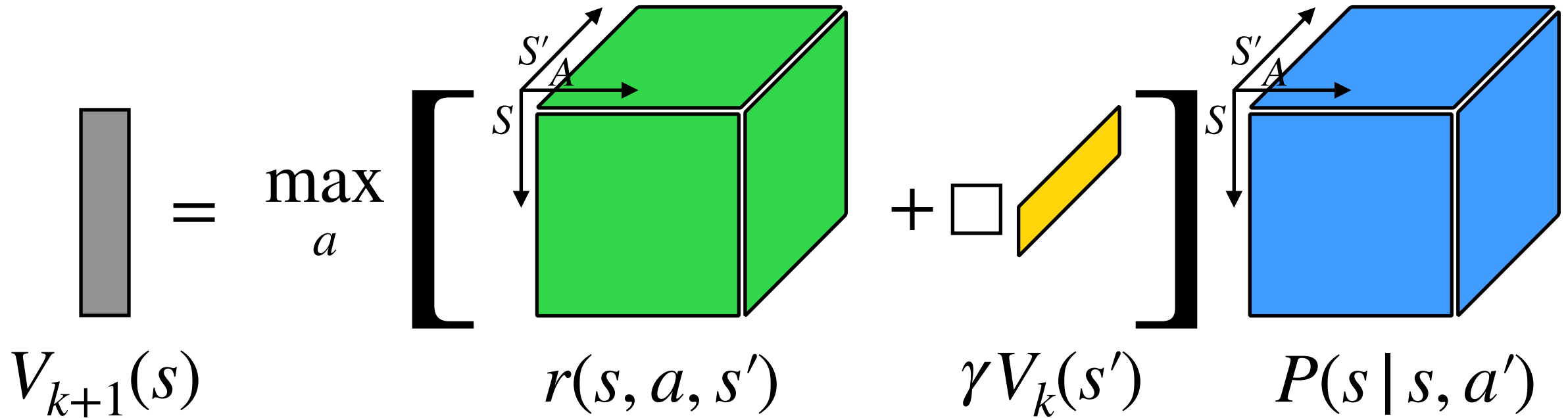
- If $|V_{k+1} - V_k|_\infty = \max_s |V_{k+1}(s) - V_k(s)| \leq \epsilon$
 - stop

Bellman Optimality Equation

$$V_{k+1}(s) = \max_a \sum_{s'} [r(s, a, s') + \gamma V_k(s')] P(s' | s, a)$$

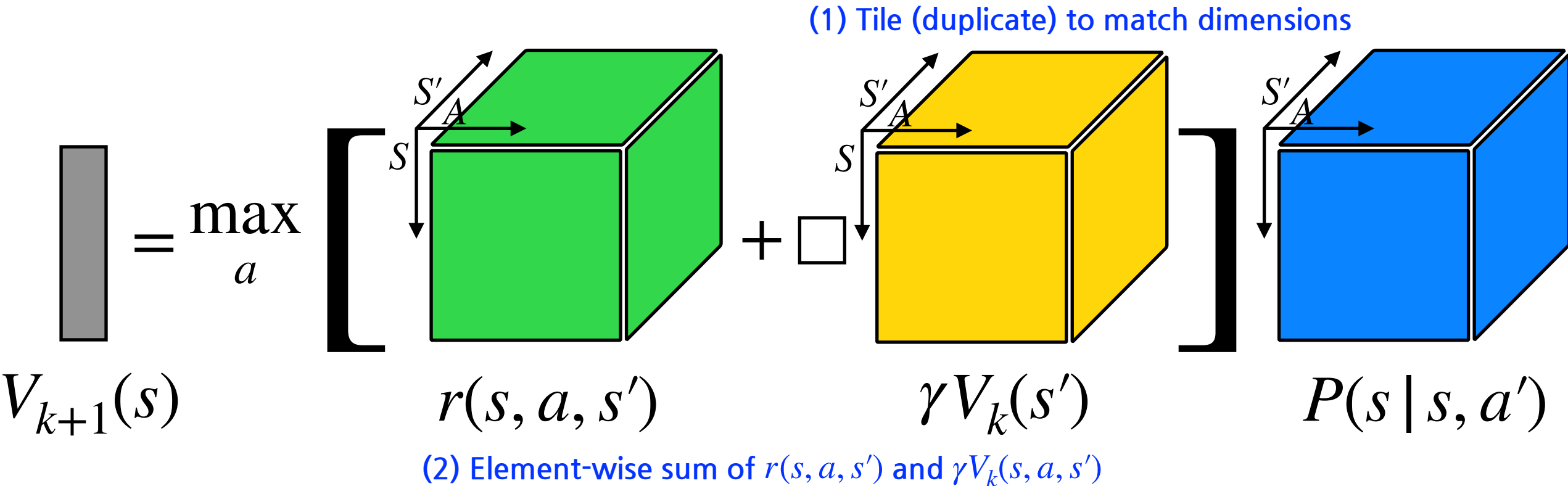
Value Iteration

$$V_{k+1}(s) = \max_a \sum_{s'} [r(s, a, s') + \gamma V_k(s')] P(s' | s, a)$$



Value Iteration

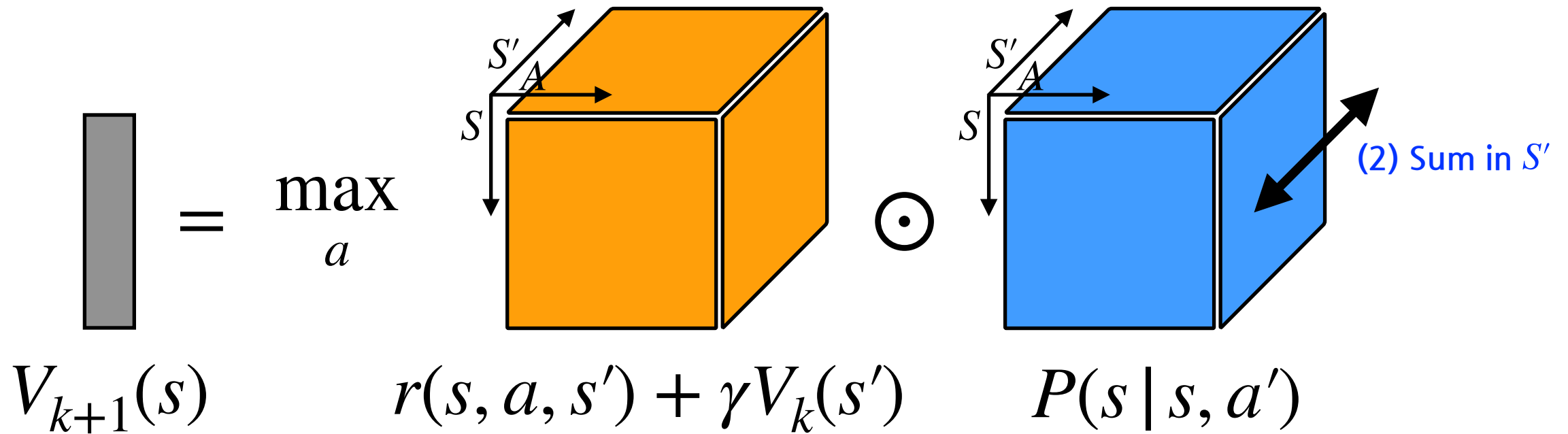
$$V_{k+1}(s) = \max_a \sum_{s'} [r(s, a, s') + \gamma V_k(s')] P(s' | s, a)$$



Value Iteration


$$V_{k+1}(s) = \max_a \sum_{s'} [r(s, a, s') + \gamma V_k(s')] P(s' | s, a)$$

(1) Element-wise multiplication



Value Iteration

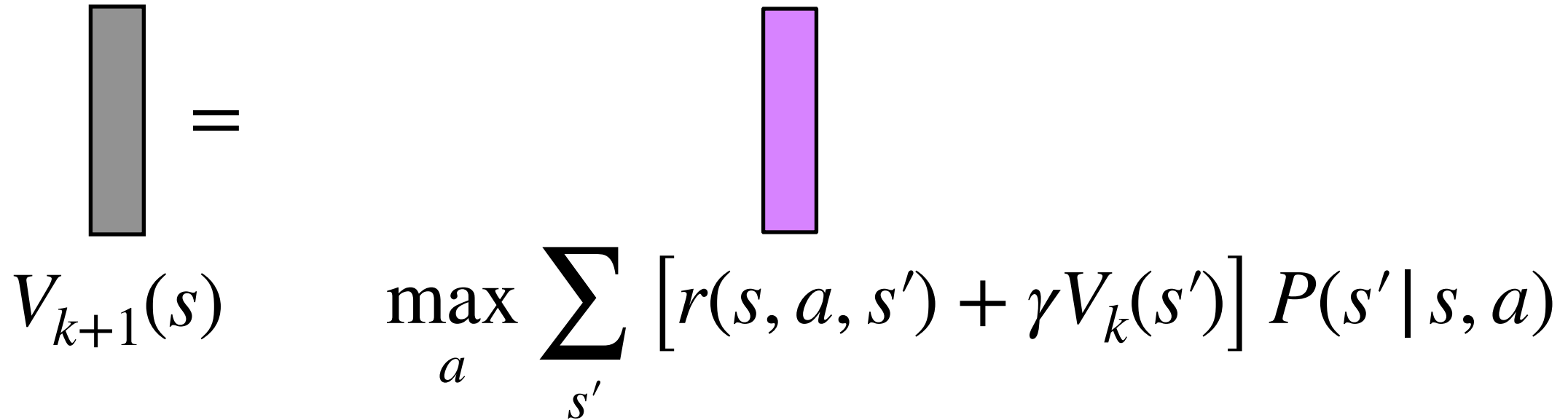
$$V_{k+1}(s) = \max_a \sum_{s'} [r(s, a, s') + \gamma V_k(s')] P(s' | s, a)$$



$V_{k+1}(s) = \max_a \sum_{s'} [r(s, a, s') + \gamma V_k(s')] P(s' | s, a)$

Value Iteration

$$V_{k+1}(s) = \max_a \sum_{s'} [r(s, a, s') + \gamma V_k(s')] P(s' | s, a)$$


$$V_{k+1}(s) = \max_a \sum_{s'} [r(s, a, s') + \gamma V_k(s')] P(s' | s, a)$$

Value Iteration

- Does **value iteration** converge?
 - Yes.
- **Why** does it converge?
 - The Bellman (Optimality) Backup operation is a **contraction** mapping.

Bellman Optimality Equation

$$V_{k+1}(s) = \max_a \sum_{s'} [r(s, a, s') + \gamma V_k(s')] P(s' | s, a)$$

Value Iteration

Bellman Optimality Backup Operator

$$TX(s) = \max_a \sum_{s'} [r(s, a, s') + \gamma X(s')] P(s' | s, a)$$

$$X \in \mathbb{R}^{|S|}$$

- Value iteration can be represented as $V_{k+1} = TV_k$.
- Bellman optimality equation becomes $V^* = TV^*$.
- Then, we can show that T is a γ -contraction mapping:
 - $|TU - TV|_\infty \leq \gamma |U - V|_\infty$

Value Iteration

- Value iteration can be represented as $V_{k+1} = TV_k$.

Bellman Optimality Backup Operator

- Bellman optimality equation becomes $V^* = TV^*$

- Then, we can show that T is a γ -contraction mapping:

$$X \in \mathbb{R}^{|S|}$$

$$\frac{|TU - TV|}{|TU - TV|_\infty} \leq \gamma \frac{|U - V|}{|U - V|_\infty} = \max_s |TU(s) - TV(s)|$$

$$\leq \max_s \left| \max_a \sum_{s'} \gamma |U(s') - V(s')| P(s'|s, a) \right|$$

$$\leq \max_s \left| \max_a \max_{s'} \gamma |U(s') - V(s')| \right|$$

$$= \gamma \max_{s'} |U(s') - V(s')| = \gamma |U - V|_\infty$$

Value Iteration

- Value iteration can be represented as $V_{k+1} = TV_k$.
- Bellman optimality equation becomes $V^* = TV^*$.
- Then, we can show that T is a γ -contraction mapping:

$$X \in \mathbb{R}^{|S|}$$

- $|TU - TV|_{\infty} \leq \gamma |U - V|_{\infty}$

- $|TU - TV|_{\infty} = \max_s |TU(s) - TV(s)|$

$$\leq \max_s \left| \max_a \sum_{s'} \gamma |U(s') - V(s')| P(s' | s, a) \right|$$

$$\leq \max_s \left| \max_a \max_{s'} \gamma |U(s') - V(s')| \right|$$

$$= \gamma \max_{s'} |U(s') - V(s')| = \gamma |U - V|_{\infty}$$

Q-Value Iteration

- However, it is not straightforward to come up with an **optimal policy** solely from $V^*(s)$.
- Hence, we use following relations between $V^*(s)$ and $Q^*(s, a)$ (aka the Bellman optimality equation).

$$V^*(s) = \max_a Q^*(s, a)$$

$$Q^*(s, a) = \sum_{s'} [r(s, a, s') + \gamma V^*(s')] P(s' | s, a)$$

• Bellman Equation

$$V_\pi(s) = \sum_a \pi(a | s) Q_\pi(s, a)$$

$$Q_\pi(s, a) = \sum_{s'} [r(s, a, s') + \gamma V_\pi(s')] P(s' | s, a)$$

$$V_\pi(s) = \sum_a \pi(a | s) \sum_{s'} [r(s, a, s') + \gamma V_\pi(s')] P(s' | s, a)$$

$$Q_\pi(s, a) = \sum_{s'} \left[r(s, a, s') + \gamma \sum_{a'} Q_\pi(s', a') \pi(a' | s') \right] P(s' | s, a)$$

• Bellman Optimality Equation

$$V^*(s) = \max_a Q^*(s, a)$$

$$Q^*(s, a) = \sum_{s'} [r(s, a, s') + \gamma V^*(s')] P(s' | s, a)$$

$$V^*(s) = \max_a \sum_{s'} [r(s, a, s') + \gamma V^*(s')] P(s' | s, a)$$

$$Q^*(s, a) = \sum_{s'} \left[r(s, a, s') + \gamma \max_{a'} Q^*(s', a') \right] P(s' | s, a)$$

$$\pi^*(a | s) = \arg \max_{a'} Q^*(s, a')$$

Q -Value Iteration

- Start from the random initial V_0
- For all states $s \in S$:

$$Q_k(s, a) = \sum_{s'} [r(s, a, s') + \gamma V_k(s')] P(s' | s, a)$$

$$V_{k+1}(s) = \max_{a'} Q_k(s, a')$$

- We now have an explicit form of the policy:

$$\pi_{k=1}(a | s) = 1 \text{ for } a = \arg \max_{a'} Q_k(s, a')$$

- Note that this policy is **deterministic**.

Policy Iteration

Policy Iteration

- Basic concept
 - Step1) Predict the value of the policy
 - Evaluate the expected return of the current policy
 - Step2) Improve the policy
 - Update the current policy to get a better expected return
- Iterate

Policy Iteration



- **Step 1: Policy evaluation**: calculate the value function for a fixed policy until convergence.
- **Step 2: Policy improvement**: update the policy using one-step lookahead using the converged value function.
- Repeat steps until the **policy** converges.
- It is guaranteed to converge to the optimal policy.
- It often converges much faster than value iteration.

Policy Iteration

- **Step 1: Policy evaluation**: calculate the value function for a fixed policy until convergence.

- Start from a random initial V_0 .
- Iterate until value converges:

$$V_{k+1}(s) = \sum_a \pi_i(a | s) \sum_{s'} [r(s, a, s') + \gamma V_k(s')] P(s' | s, a)$$

- **Step 2: Policy improvement**: update the policy using one-step lookahead using the converged value function.

- One-step lookahead:

$$Q_{\pi_k}(s, a) = \sum_{s'} [r(s, a, s') + \gamma V_{\pi_i}(s')] P(s' | s, a)$$

$$\pi_{i+1}(a | s) = 1 \text{ for } a = \arg \max_{a'} Q_{\pi_i}(s, a')$$

Policy Evaluation

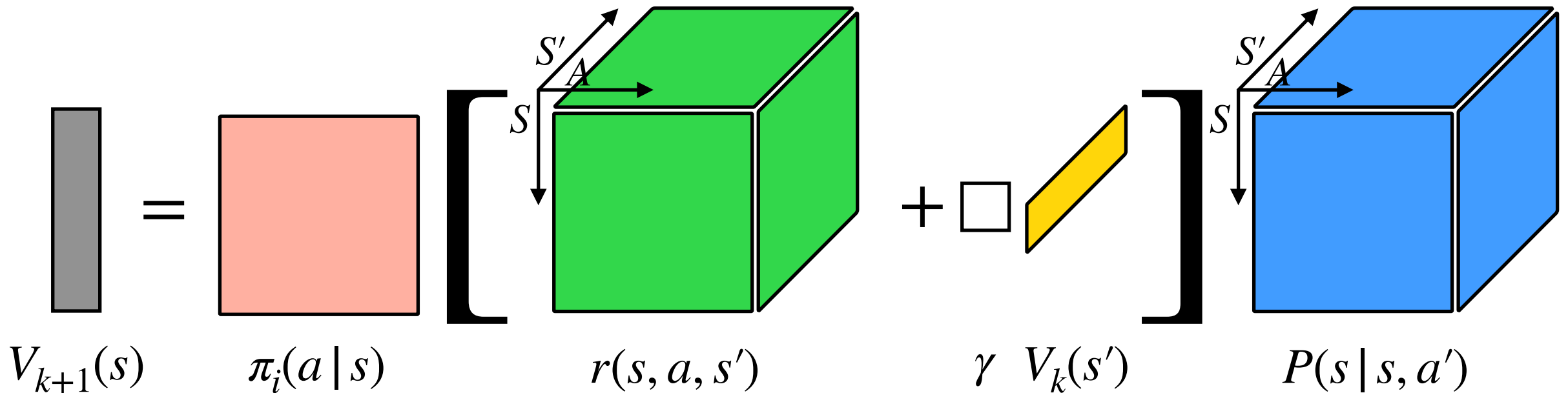
- Start from a random initial V_0
- Initialize V_{k+1} with a zero vector.
- Loop
 - For all states $s \in S$:
 - For all actions $a \in A$:
 - For all next states $s' \in S$:
 - $V_{k+1}(s) = V_k(s) + [r(s, a, s') + \gamma V_k(s')]P(s' | s, a)\pi_i(a | s)$
 - If $|V_{k+1} - V_k|_\infty = \max_s |V_{k+1}(s) - V_k(s)| \leq \epsilon$
 - stop

Bellman Equation

$$V_{k+1}(s) = \sum_a \pi_i(a | s) \sum_{s'} [r(s, a, s') + \gamma V_k(s')] P(s' | s, a)$$

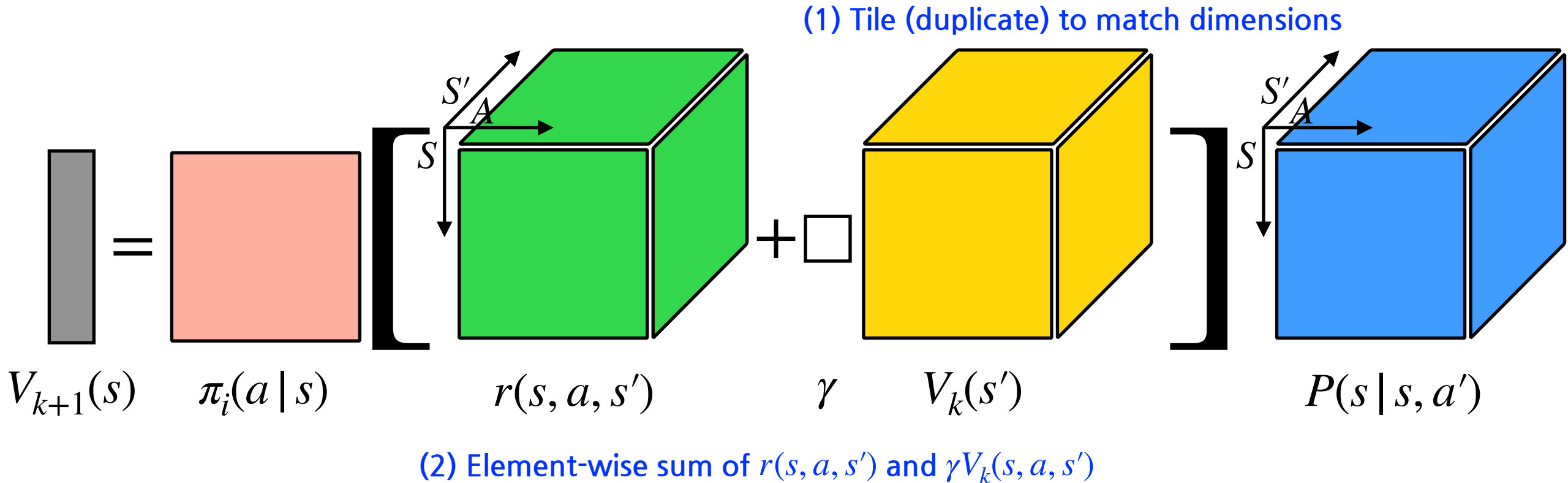
Policy Evaluation

$$V_{k+1}(s) = \sum_a \pi_i(a | s) \sum_{s'} [r(s, a, s') + \gamma V_k(s')] P(s' | s, a)$$



Policy Evaluation

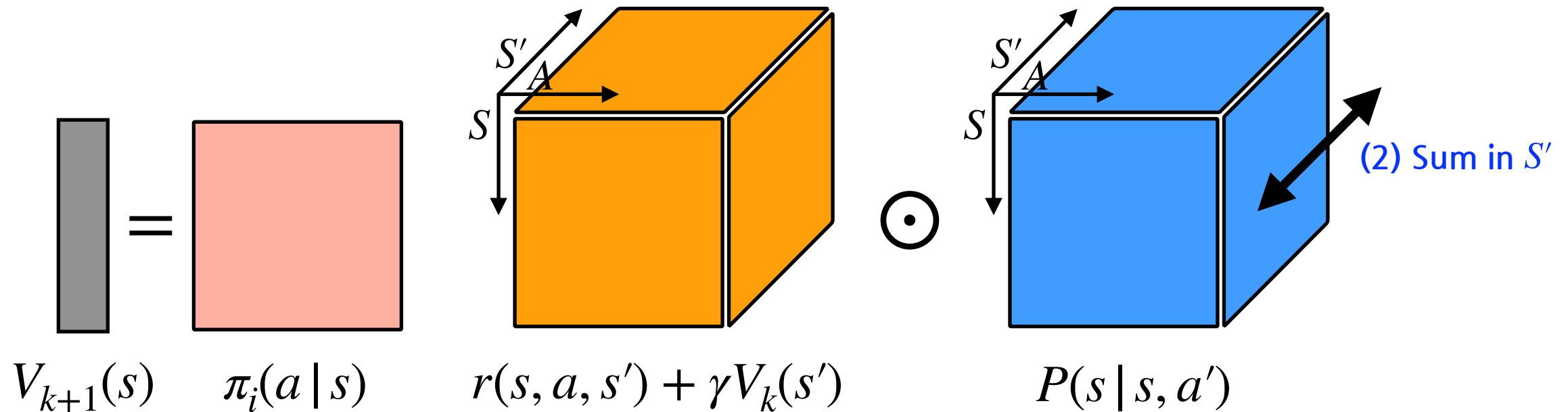
$$V_{k+1}(s) = \sum_a \pi_i(a | s) \sum_{s'} [r(s, a, s') + \gamma V_k(s')] P(s' | s, a)$$



Policy Evaluation

$$V_{k+1}(s) = \sum_a \pi_i(a | s) \sum_{s'} [r(s, a, s') + \gamma V_k(s')] P(s' | s, a)$$

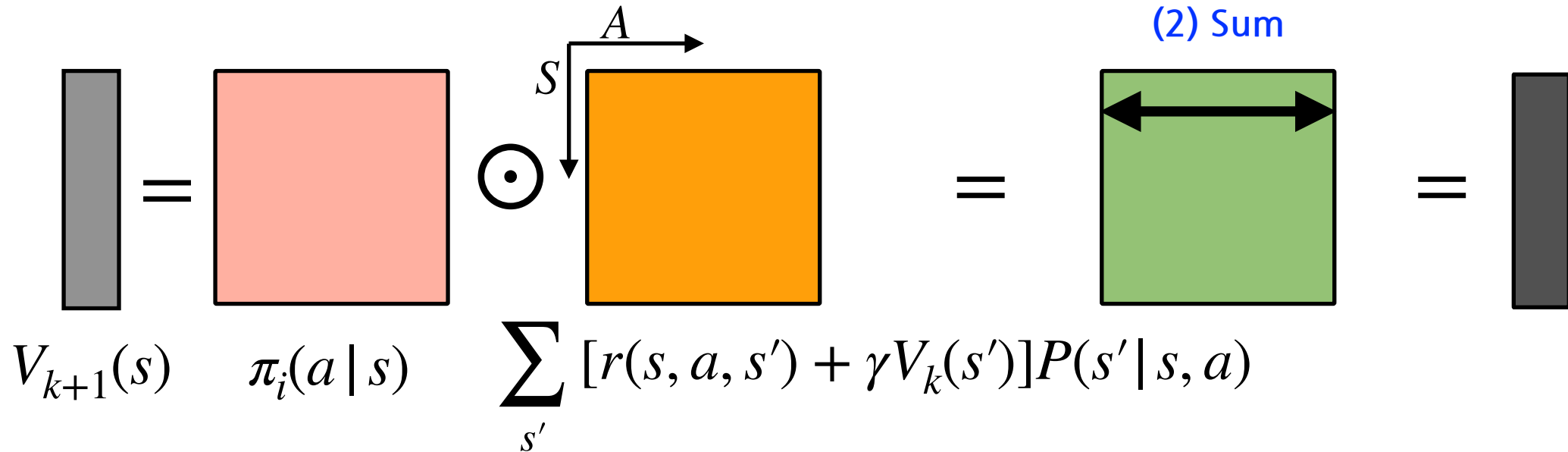
(1) Element-wise multiplication



Policy Evaluation

$$V_{k+1}(s) = \sum_a \pi_i(a | s) \sum_{s'} [r(s, a, s') + \gamma V_k(s')] P(s' | s, a)$$

(1) Element-wise multiplication



Policy Evaluation

- Does policy evaluation converge?
 - Yes.
- Why does it converge?
 - The Bellman Backup operation is a **contraction** mapping.

Policy Evaluation

Bellman Backup Operator

$$T_{\pi}X(s) = \sum_a \pi(a | s) \sum_{s'} [r(s, a, s') + \gamma X(s')] P(s' | s, a)$$

$$X \in \mathbb{R}^{|S|}$$

$$|T_{\pi}U - T_{\pi}V|_{\infty} = \max_s |T_{\pi}U(s) - T_{\pi}V(s)|$$

$$= \max_s \left| \sum_a \pi(a | s) \sum_{s'} [r(s, a, s') + \gamma U(s')] P(s' | s, a) - \sum_a \pi(a | s) \sum_{s'} [r(s, a, s') + \gamma V(s')] P(s' | s, a) \right|$$

$$= \max_s \left| \gamma \sum_a \sum_{s'} (U(s') - V(s')) P(s' | s, a) \pi(a | s) \right|$$

$$\leq \max_s \left| \gamma \sum_a \max_{s'} |U(s') - V(s')| \right|$$

$$= \gamma \max_{s'} |U(s') - V(s')| = \gamma |U - V|_{\infty}$$

Policy Improvement

- Let V_{π_i} be the value evaluated from the policy π_i .
- Our goal is to find a policy π_{i+1} .
- Compute the state-action value $Q_{\pi_i}(s, a)$ from state value $V_{\pi_i}(s)$:

$$Q_{\pi_i}(s, a) = \sum_{s'} [r(s, a, s') + \gamma V_{\pi_i}(s')] P(s' | s, a)$$

- The policy is updated via:

$$\pi_{i+1}(a | s) = 1 \text{ for } a = \arg \max_{a'} Q_{\pi_i}(s, a)$$

Policy Improvement

- It is guaranteed that the **policy improvement** improves the value.
- The value of the updated policy is

$$V_{\pi_{i+1}}(s) = \sum_a Q_{\pi_i}(s, a) \pi_{i+1}(a | s)$$

- We can show that $V_{\pi_{i+1}}(s) \geq V_{\pi_i}(s)$, hence it improves the value function.

$$\begin{aligned} V_{\pi_{i+1}}(s) &= \sum_a Q_{\pi_i}(s, a) \pi_{i+1}(a | s) \\ &= \max_{a'} Q_{\pi_i}(s, a') \\ &\geq \sum_s Q_{\pi_i}(s, a) \pi_i(a | s) \\ &= V_{\pi_i}(s) \end{aligned}$$

Policy Iteration

- **Step 1: Policy evaluation**: Evaluate V_{π} .
- **Step 2: Policy improvement**: Generate π' where $V_{\pi'} \geq V_{\pi}$.
- **Policy iteration** is often more effective than **value iteration**, why?
 - It is often the case that a policy function reaches the optimal policy (policy iteration) much sooner than a value function reaches the optimal value function (value iteration).
 - In many cases, we are more interested in finding the optimal policy function rather than the optimal value function.



ROBOT INTELLIGENCE LAB