DMQA Seminar 20211203

Basics of Reinforcement Learning

From Markov Decision Process To SARSA/Q-Learning

일반대학원 산업경영공학과
김재훈

고려대학교
KOREA UNIVERSITY

Introduction

발표자 소개

- 이름: 김재훈
- 학력
  - ✓ 2020.03 – 현재 | 석박사통합과정 | 고려대학교 산업경영공학과 (지도교수: 김성범)
- 연구분야
  - ✓ Self-supervised learning
  - ✓ Reinforcement learning
- e-mail : jhoon0418@korea.ac.kr

CONTENTs

Reinforcement Learning

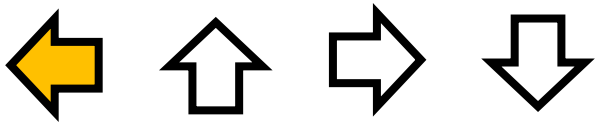
State, Action Reward

❖ What is Reinforcement Learning?

- 강화학습은 상태(state), 행동(action), 보상(reward)으로 구성되어 있음
- 강화학습은 순차적인 의사결정 문제에서 누적 보상을 최대화 하기 위해 시행착오를 거쳐서 상황에 따른 행동 정책을 학습

State

Action

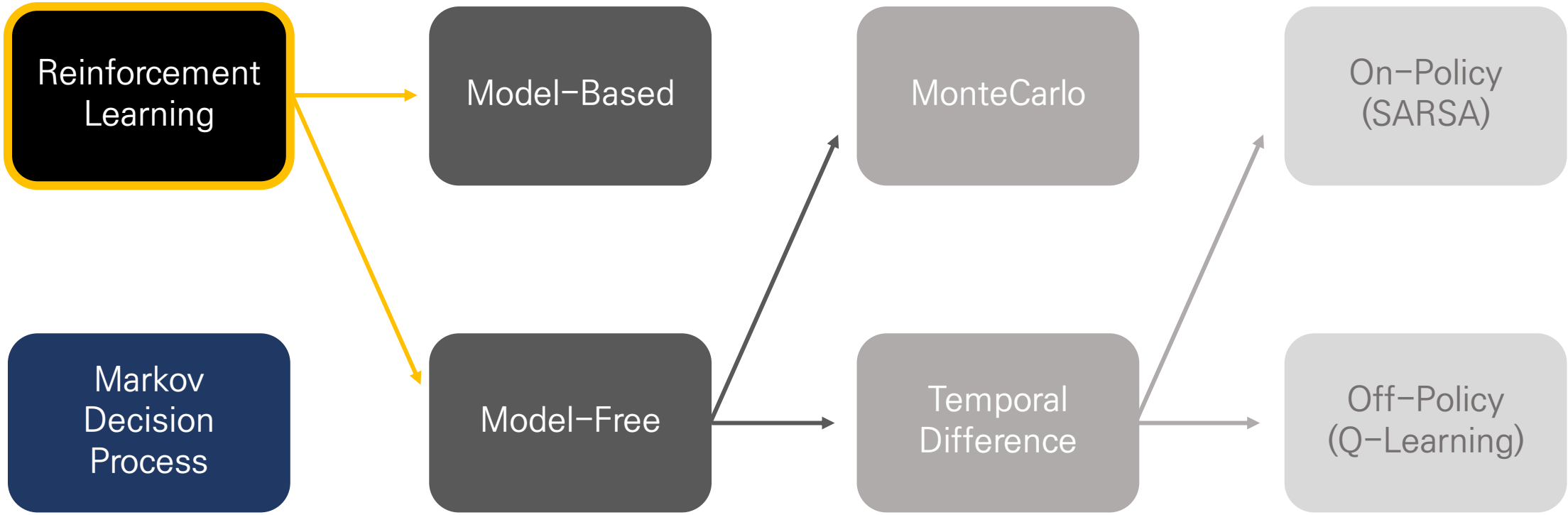


Reward

-10 / 0 / 1

Reinforcement Learning

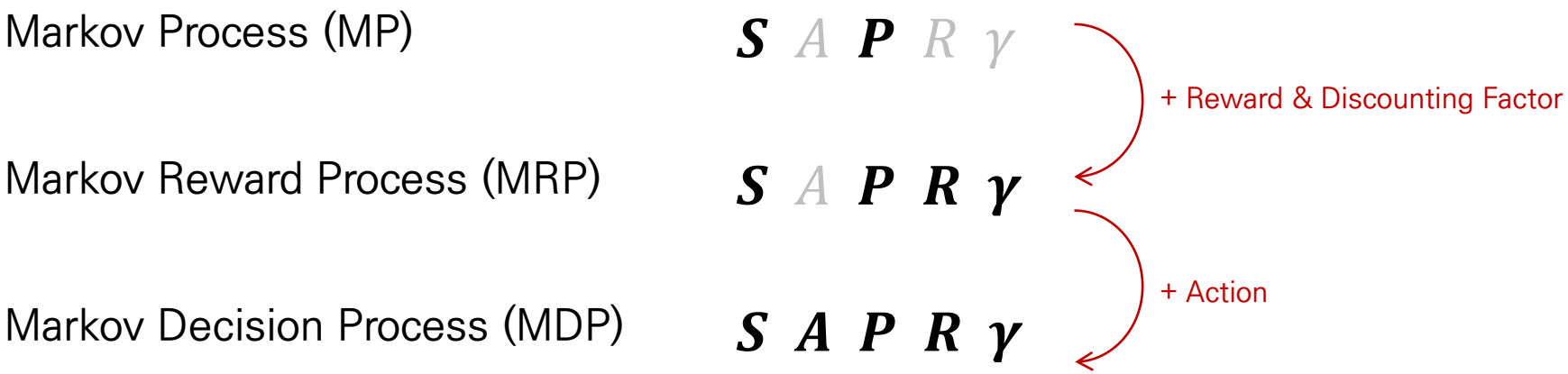
❖ What is in the Reinforcement Learning?



Formulating Sequential Problem

❖ Markov Property

- 미래의 상태(s_{t+1})는 현재의 상태(s_t)에만 영향을 받으며 과거의 상태($s_1 \ldots s_{t-1}$)의 영향을 받지 않음
- $\mathbb{P}[S_{t+1} \mid S_t] = \mathbb{P}[S_{t+1} \mid S_1, S_2, \ldots, S_t]$



Formulating Sequential Problem

Markov Process (MP)

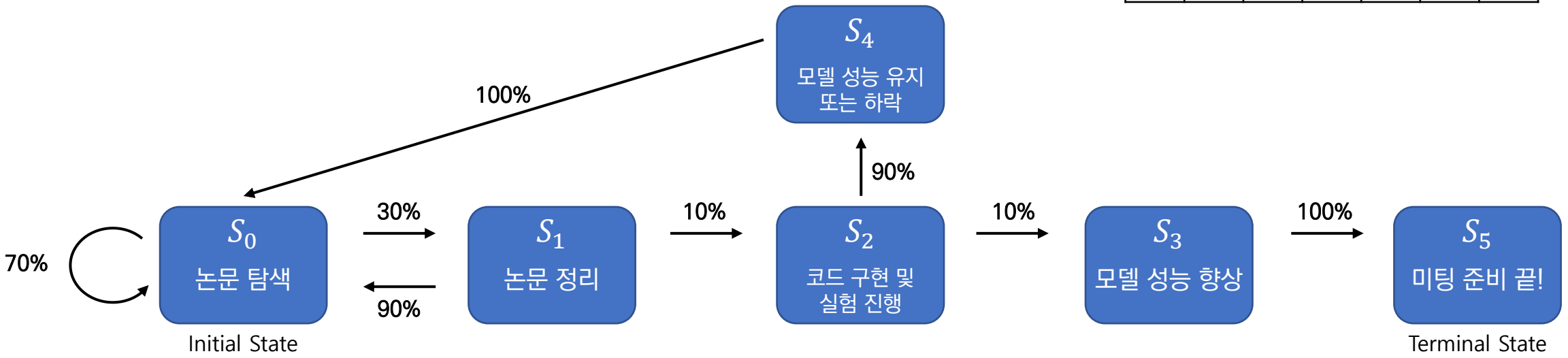
❖ Markov Process (MP)

- $\boldsymbol{S}$ A $\boldsymbol{P}$ R γ
- 상태와 상태전이확률로 구성되며 상태 간 순차적인 관계를 확인할 수 있음

- 상태전이확률 (state transition probability matrix)
  - ✓ 상태 s에서 상태 s'로 전이할 확률의 행렬을 의미함
  - ✓ $P_{ss'} = P[S_{t+1} = s' | S_t = s]$

S_t → S_{t+1}

$P_{ss`}$	S_0	S_1	S_2	S_3	S_4	S_5
S_0	0.7	0.3				
S_1	0.9		0.1			
S_2				0.1	0.9	
S_3						1.0
S_4	1.0					
S_5						



Formulating Sequential Problem

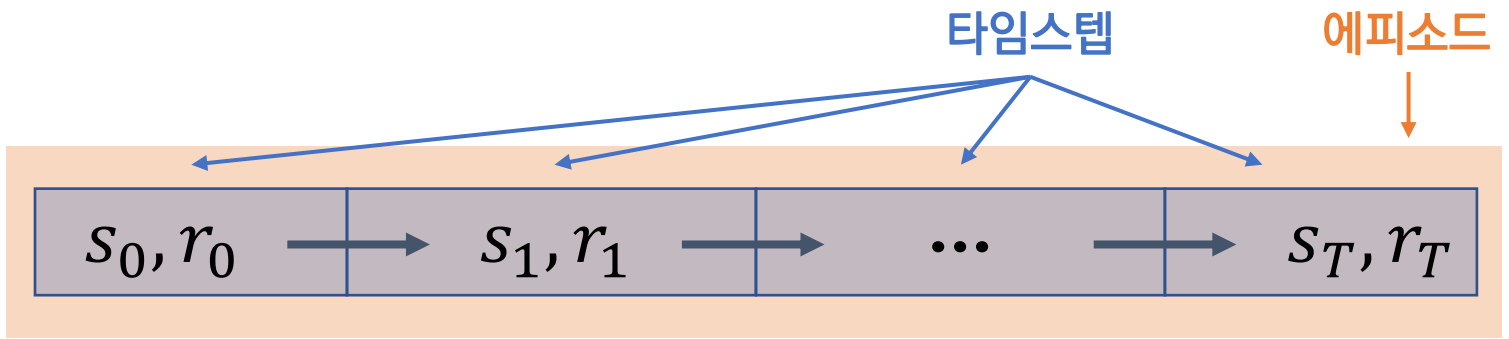
Markov Reward Process (MRP)

❖ Markov Reward Process (MRP)

- $\boldsymbol{S}$ A $\boldsymbol{P}$ $\boldsymbol{R}$ $\boldsymbol{\gamma}$
- 보상함수 및 감쇠 인자가 추가로 구성되며 상태 간 이동 따른 보상을 계산할 수 있음

- 보상함수 (reward function)
  : 상태 s에 도달함으로써 받게 되는 보상
  $R = \mathbb{E}[R_t \mid S_t = s]$

- 감쇠인자 (discounting factor, γ)
  : 미래 보상의 중요도를 조절하는 인자



Formulating Sequential Problem

Markov Reward Process (MRP)

❖ Markov Reward Process (MRP)

- 리턴 (return)
  - ✓ 시점 t에서부터 기대할 수 있는 미래의 보상
  - ✓ $G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots$

- 감쇠인자를 통해서 보상 시점의 선호도를 반영할 수 있음
- 감쇠인자를 0과 1사이의 값으로 하여 리턴이 무한대가 되는 것을 방지함

> “ 리턴(G_t)를 최대화하는 것이 강화학습의 최종 목적 ”

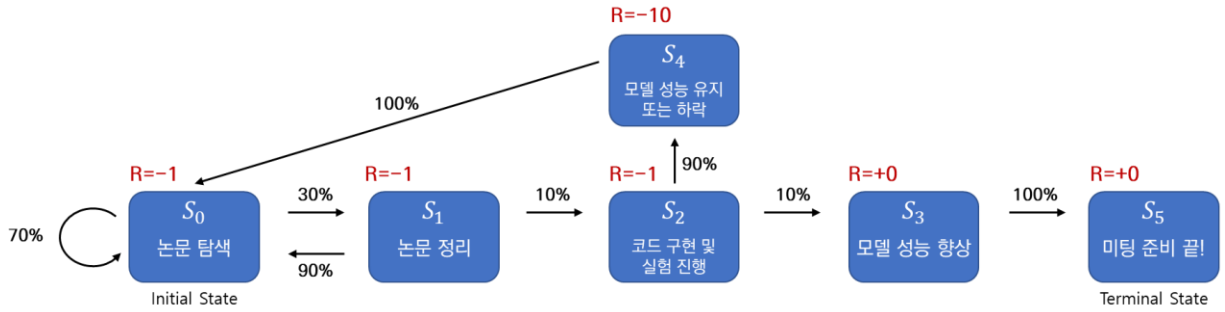
Formulating Sequential Problem

Markov Reward Process (MRP)

❖ State Value Function (V)

- 상태가치함수 (state value function)
  - ✓ 특정 상태에 도달하는 것의 가치를 책정하는 함수
  - ✓ $v(s) = \mathbb{E}[G_t|S_t = s]$

- 환경의 확률적 요소로 인해 에피소드별 리턴의 기댓값을 이용함
- 같은 상태에서 출발하여도 에피소드마다 리턴이 달라지므로 주어진 상태의 가치를 리턴의 기댓값을 통해 정의함



Sampling Episodes

EP.1	논문 탐색	논문 탐색	논문 탐색	논문 정리	구현/실험	성능향상	준비 끝!			
EP.2	논문 탐색	논문 정리	구현/실험	유지/하락	논문 탐색	논문 정리	구현/실험	성능향상	준비 끝!	
	-1	$-1 * 0.9$	$-1 * 0.9^2$	$-10 * 0.9^3$	$-1 * 0.9^4$	$-1 * 0.9^5$	$-1 * 0.9^6$	$+0 * 0.9^7$	$+0 * 0.9^8$	

$G(s_0) = -4.095$

$G(s_0) = -11.778$

$\mathbf{v(s_0) = -7.937}$

Formulating Sequential Problem

Markov Decision Process (MDP)

❖ Markov Decision Process (MDP)

- $\boldsymbol{S}$ $\boldsymbol{A}$ $\boldsymbol{P}$ $\boldsymbol{R}$ $\boldsymbol{\gamma}$
- MRP에 에이전트를 추가하여 각 상태에서의 의사결정을 모델에 포함시킴

- 행동 (action, a)
  - ✓ 에이전트가 각 상태에서 취하는 의사결정

- 정책함수(policy function)
  - ✓ 현재 상태에서 특정 행동을 취할 확률을 나타냄
  - ✓ $\pi(a|s) = \mathbb{P}[A_t = a|S_t = s]$

- 상태전이확률의 조건에 행동이 추가됨
  - ✓ 현재 상태에서 특정 행동을 취했을 때 다음 상태에 도달할 확률
  - ✓ $P_{ss'}^{a} = \mathbb{P}[S_{t+1} = s'|S_t = s,\ A_t = a]$

- 보상함수의 조건에 행동이 추가됨
  - ✓ 특정 상태에서 특정 행동을 함으로써 받는 보상을 평가
  - ✓ $R_s^a = \mathbb{E}[R_{t+1}|S_t = s,\ A_t = a]$

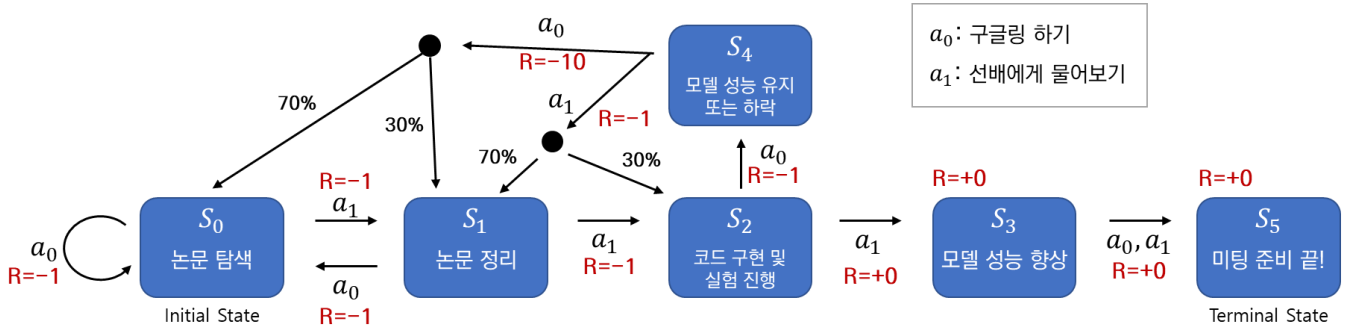
Formulating Sequential Problem

Markov Decision Process (MDP)

❖ State-Action Value Function (Q)

- 상태-행동가치함수 (state-action value function)
  : 현재 상태에서 특정 행동 취하는 것의 가치를 책정하는 함수
  $q(s,a) = \mathbb{E}[G_t|S_t = s,\ A_t = a]$

- 환경의 확률적 요소로 인해 에피소드별 리턴의 기댓값을 이용함
- 같은 상태에서 같은 행동을 하여도 에피소드마다 리턴이 달라지므로 상태-행동의 가치를 리턴의 기댓값을 통해 정의함



Bellman Equation

Introduction

❖ Bellman Equation

- 벨만 방정식 (bellman equation)
  - ✓ 현재 시점의 가치와 다음 시점의 가치 사이의 관계를 다루는 방정식
  - ✓ 벨만 방정식을 통해서 상태 및 행동의 가치를 책정할 수 있음

- 벨만 방정식은 두 종류로 나뉨
  - ✓ 벨만 기대방정식 (bellman expectation equation) : 주어진 정책을 평가할 때 사용
  - ✓ 벨만 최적방정식 (bellman optimality equation) : 최적의 상태 가치를 찾을 때 사용

Bellman Equation

Bellman Expectation Equation

❖ Bellman Expectation Equation

- 벨만 기대방정식 (bellman expectation equation)
  - ✓ 정책이 고정되어 있는 상황에서 상태 또는 상태-행동의 가치를 책정하는 방법으로서 주어진 정책에 대해 평가

- 보상함수와 상태전이확률을 알고 있는지 여부에 따라 표현하는 방법이 나뉨
  ** 보상함수와 상태전이확률을 안다고 할 때 MDP를 안다고 표현함

	MDP를 모르는 경우	MDP를 아는 경우
State Value	$v_\pi(\mathrm{s_t}) = \mathbb{E}_\pi[r_{t+1} + \gamma v_\pi(s_{t+1})]$	$v_\pi(s) = \sum_{a \in A} \pi(a\|s) \left(\boldsymbol{r_s^a} + \gamma \sum_{s' \in S} \boldsymbol{P_{ss'}^a}\, v_\pi(s') \right)$
State-Action Value	$q_\pi(\mathrm{s_t}, \mathrm{a_t}) = \mathbb{E}_\pi[r_{t+1} + \gamma q_\pi(s_{t+1}, a_{t+1})]$	$q_\pi(s,a) = \mathbf{r_s^a} + \gamma \sum_{s' \in S} \boldsymbol{P_{ss'}^a} \sum_{a' \in A} \pi(a'\|s')\, q_\pi(s', a')$

Bellman Equation

Bellman Expectation Equation

	MDP를 모르는 경우	MDP를 아는 경우
State Value	$v_\pi(s_t) = \mathbb{E}_\pi[r_{t+1} + \gamma v_\pi(s_{t+1})]$	$v_\pi(s) = \sum_{a \in A} \pi(a\|s) \left(r_s^a + \gamma \sum_{s' \in S} P_{ss'}^a v_\pi(s') \right)$
State-Action Value	$q_\pi(s_t, a_t) = \mathbb{E}_\pi[r_{t+1} + \gamma q_\pi(s_{t+1}, a_{t+1})]$	$q_\pi(s,a) = r_s^a + \gamma \sum_{s' \in S} P_{ss'}^a \sum_{a' \in A} \pi(a'\|s') \, q_\pi(s', a')$

❖ State Value Function (V)

- 주어진 환경에서 샘플링 된 에피소드의 리턴 값에 평균을 취하여 해당 상태의 가치를 평가함
- MDP를 모르는 상황에서의 상태가치함수

$$
\begin{aligned}
v_\pi(s_t) &= \mathbb{E}_\pi[G_t] \\
&= \mathbb{E}_\pi[r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots] \\
&= \mathbb{E}_\pi[r_{t+1} + \gamma(r_{t+2} + \gamma r_{t+3} + \cdots)] \\
&= \mathbb{E}_\pi[r_{t+1} + \gamma G_{t+1}] \\
&= \mathbb{E}_\pi[r_{t+1} + \gamma v_\pi(s_{t+1})]
\end{aligned}
$$

Bellman Equation

Bellman Expectation Equation

	MDP를 모르는 경우	MDP를 아는 경우
State Value	$v_\pi(s_t) = \mathbb{E}_\pi[r_{t+1} + \gamma v_\pi(s_{t+1})]$	$v_\pi(s) = \sum_{a \in A} \pi(a\|s) \left(r_s^a + \gamma \sum_{s' \in S} P_{ss'}^a v_\pi(s') \right)$
State-Action Value	$q_\pi(s_t, a_t) = \mathbb{E}_\pi[r_{t+1} + \gamma q_\pi(s_{t+1}, a_{t+1})]$	$q_\pi(s,a) = r_s^a + \gamma \sum_{s' \in S} P_{ss'}^a \sum_{a' \in A} \pi(a'\|s') \, q_\pi(s', a')$

❖ State Value Function (V)

- 주어진 환경에서 샘플링 된 에피소드의 리턴 값에 평균을 취하여 해당 상태의 가치를 평가함
- MDP를 알고 있는 상황에서의 상태가치함수

$$v_\pi(s) = \sum_{a \in A} \pi(a|s) \left(r_s^a + \gamma \sum_{s' \in S} P_{ss'}^a v_\pi(s') \right)$$

현재 상태에서 특정 행동을 선택할 확률

현재 상태에서 특정 행동을 통해 얻는 보상

현재 상태에서 특정 행동을 하였을 때 특정 다음 상태로 넘어갈 확률

특정 다음 상태의 가치

Bellman Equation

Bellman Expectation Equation

	MDP를 모르는 경우	MDP를 아는 경우
State Value	$v_\pi(s_t) = \mathbb{E}_\pi[r_{t+1} + \gamma v_\pi(s_{t+1})]$	$v_\pi(s) = \sum_{a \in A} \pi(a\|s) \left(r_s^a + \gamma \sum_{s' \in S} P_{ss'}^a v_\pi(s') \right)$
State-Action Value	$q_\pi(s_t, a_t) = \mathbb{E}_\pi[r_{t+1} + \gamma q_\pi(s_{t+1}, a_{t+1})]$	$q_\pi(s,a) = r_s^a + \gamma \sum_{s' \in S} P_{ss'}^a \sum_{a' \in A} \pi(a'\|s') \, q_\pi(s', a')$

❖ State Value Function (V)

- MDP를 모르는 상황에서 상태가치함수에 상태전이확률 및 보상함수를 고려할 경우 MDP를 아는 상황의 상태가치함수로 풀어 쓸 수 있음

$$v_\pi(s) = \sum_{a \in A} \pi(a|s) \left(r_s^a + \gamma \sum_{s' \in S} P_{ss'}^a v_\pi(s') \right)$$

$$= \sum_{a \in A} \pi(a|s) r_s^a + \gamma \sum_{a \in A} \pi(a|s) \sum_{s' \in S} P_{ss'}^a v_\pi(s')$$

현재 상태에서 특정 행동을 선택할 확률 X 현재 상태에서 특정 행동을 통해 얻는 보상

현재 상태에서 특정 행동을 하였을 때 특정 다음 상태로 넘어갈 확률 X 특정 다음 상태의 가치

현재 상태에서 특정 행동을 할 확률 X 특정 행동을 할 때 다음 상태의 평균 가치

$$= \mathbb{E}_\pi[r_{t+1}] + \mathbb{E}_\pi[\gamma v_\pi(s_{t+1})]$$

$$= \mathbb{E}_\pi[r_{t+1} + \gamma v_\pi(s_{t+1})]$$

Bellman Equation

Bellman Expectation Equation

	MDP를 모르는 경우	MDP를 아는 경우
State Value	$v_\pi(s_t) = \mathbb{E}_\pi[r_{t+1} + \gamma v_\pi(s_{t+1})]$	$v_\pi(s) = \sum_{a \in A} \pi(a\|s) \left(r_s^a + \gamma \sum_{s' \in S} P_{ss'}^a v_\pi(s') \right)$
State-Action Value	$q_\pi(s_t, a_t) = \mathbb{E}_\pi[r_{t+1} + \gamma q_\pi(s_{t+1}, a_{t+1})]$	$q_\pi(s,a) = r_s^a + \gamma \sum_{s' \in S} P_{ss'}^a \sum_{a' \in A} \pi(a'\|s') \, q_\pi(s', a')$

❖ State-Action Value Function (Q)

- 주어진 환경에서 샘플링 된 에피소드의 리턴 값에 평균을 취하여 해당 상태-행동의 가치를 평가함
- MDP를 모르는 상황에서의 상태-행동가치함수이며

$$v_\pi(s_t) = \mathbb{E}_\pi[r_{t+1} + \gamma v_\pi(s_{t+1})] \quad \Rightarrow \quad q_\pi(s_t, a_t) = \mathbb{E}_\pi[r_{t+1} + \gamma q_\pi(s_{t+1}, a_{t+1})]$$

Bellman Equation

Bellman Expectation Equation

	MDP를 모르는 경우	MDP를 아는 경우
State Value	$v_\pi(s_t) = \mathbb{E}_\pi[r_{t+1} + \gamma v_\pi(s_{t+1})]$	$v_\pi(s) = \sum_{a \in A} \pi(a\|s) \left(r_s^a + \gamma \sum_{s' \in S} P_{ss'}^a v_\pi(s') \right)$
State-Action Value	$q_\pi(s_t, a_t) = \mathbb{E}_\pi[r_{t+1} + \gamma q_\pi(s_{t+1}, a_{t+1})]$	$q_\pi(s,a) = r_s^a + \gamma \sum_{s' \in S} P_{ss'}^a \sum_{a' \in A} \pi(a'\|s')\, q_\pi(s', a')$

❖ State-Action Value Function (Q)

- MDP를 모르는 상황에서 상태-행동가치함수에 상태전이확률 및 보상함수를 고려할 경우 MDP를 아는 상황의 함수로 풀어 쓸 수 있음

$$q_\pi(s,a) = \mathbf{r_s^a} + \gamma \sum_{s' \in S} \boldsymbol{P_{ss'}^a} \underbrace{\sum_{a' \in A} \pi(a'|s')\, q_\pi(s', a')}_{v_\pi(s')}$$

현재 상태에서 특정 행동을 하였을 때 특정 다음 상태로 넘어갈 확률 X 특정 다음 상태의 가치

$$= \mathbb{E}_\pi[r_{t+1}] + \mathbb{E}_\pi[\gamma q_\pi(s_{t+1}, a_{t+1})]$$

$$= \mathbb{E}_\pi[r_{t+1} + \gamma q_\pi(s_{t+1}, a_{t+1})]$$

Bellman Equation

Bellman Optimality Equation

❖ Bellman Optimality Equation

- 벨만 최적방정식 (bellman optimality equation)
  - ✓ 주어진 MDP에서 가장 좋은 정책(최적정책)을 선택하여 최적의 상태 혹은 상태-행동 가치를 평가함
- 보상함수와 상태전이확률을 알고 있는지 여부에 따라 표현하는 방법이 나뉨

  ** 보상함수와 상태전이확률을 안다고 할 때 MDP를 안다고 표현함

	MDP를 모르는 경우	MDP를 아는 경우
State Value	$v_*(\mathrm{s_t}) = \max_a \mathbb{E}\left[r_{t+1} + \gamma v_*(s_{t+1})\right]$	$v_*(s) = \max_{\mathrm{a}} \left[\boldsymbol{r_s^a} + \gamma \sum_{s' \in S} \boldsymbol{P_{ss'}^a}\, v_*(s') \right]$
Action-State Value	$q_*(\mathrm{s_t}, a_t) = \mathbb{E}\left[r_{t+1} + \gamma \max_{a'} q_*(s_{t+1}, a')\right]$	$q_*(s,a) = \mathbf{r_s^a} + \gamma \sum_{s' \in S} \boldsymbol{P_{ss'}^a} \max_{\mathrm{a}'} q_*(s', a')$

Bellman Equation

Bellman Optimality Equation

❖ Bellman Optimality Equation

- 벨만최적방정식은 행동을 선택할 때 확률적으로 선택하는 것이 아니라 최댓값 연산자를 통해 제일 좋은 액션을 선택함
- 따라서 확률적 요소 중 상태전이확률은 유지되지만 행동 선택 확률은 사용되지 않음

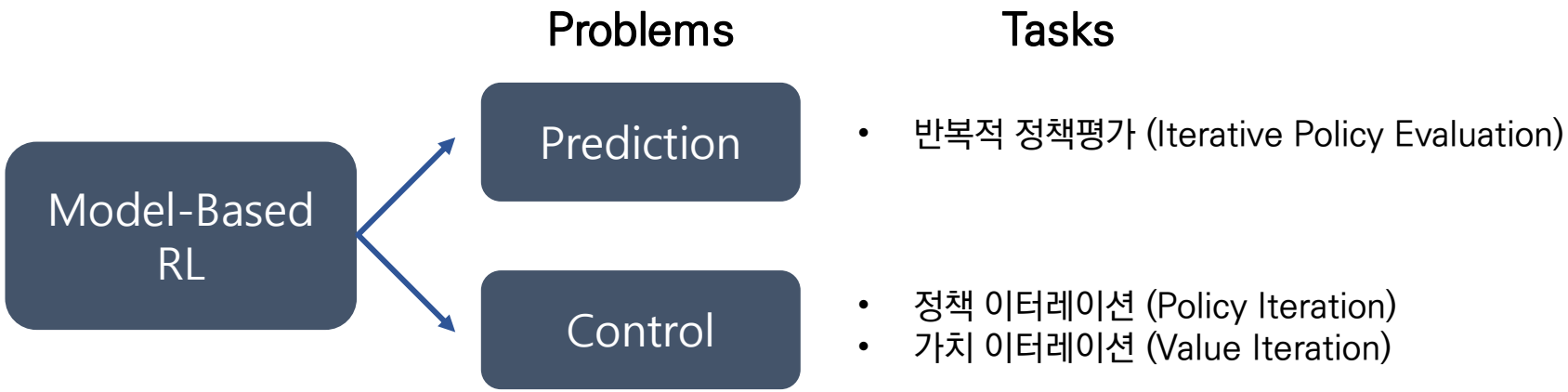
	MDP를 모르는 경우	MDP를 아는 경우
Bellman Expectation Equation	$v_\pi(s_t) = \mathbb{E}_\pi[r_{t+1} + \gamma v_\pi(s_{t+1})]$	$v_\pi(s) = \sum_{a \in A} \pi(a\|s) \left(\boldsymbol{r_s^a} + \gamma \sum_{s' \in S} \boldsymbol{P_{ss'}^a}\, v_\pi(s') \right)$
Bellman Optimality Equation	$v_*(s_t) = \max_a \mathbb{E}\,[r_{t+1} + \gamma v_*(s_{t+1})]$	$v_*(s) = \max_a \left[\boldsymbol{r_s^a} + \gamma \sum_{s' \in S} \boldsymbol{P_{ss'}^a}\, v_*(s') \right]$

Model-Based Reinforcement Learning

Introduction

❖ Model-Based Reinforcement Learning(Planning)

- MDP에 대한 모든 정보를 알 때 이를 이용하여 정책을 평가 및 개선해 나가는 과정
- Prediction problem
  - ✓ 정책이 주어졌을 때 각 상태의 가치를 평가하는 문제
- Control problem
  - ✓ 최적의 정책 함수를 찾는 문제



Model-Based Reinforcement Learning

Introduction

❖ Example Environment) Grid World 1

- Action : Left(25%), Right(25%), Up(25%), Down(25%)
- Reward : −1 per step
- State : Total 16 states including terminal states
- Decay Ratio : 1
- State Transition Probability : 1

Grid World

End	S_1	S_2	S_3
S_4	S_5	S_6	S_7
S_8	S_9	S_{10}	S_{11}
S_{12}	S_{13}	S_{14}	End

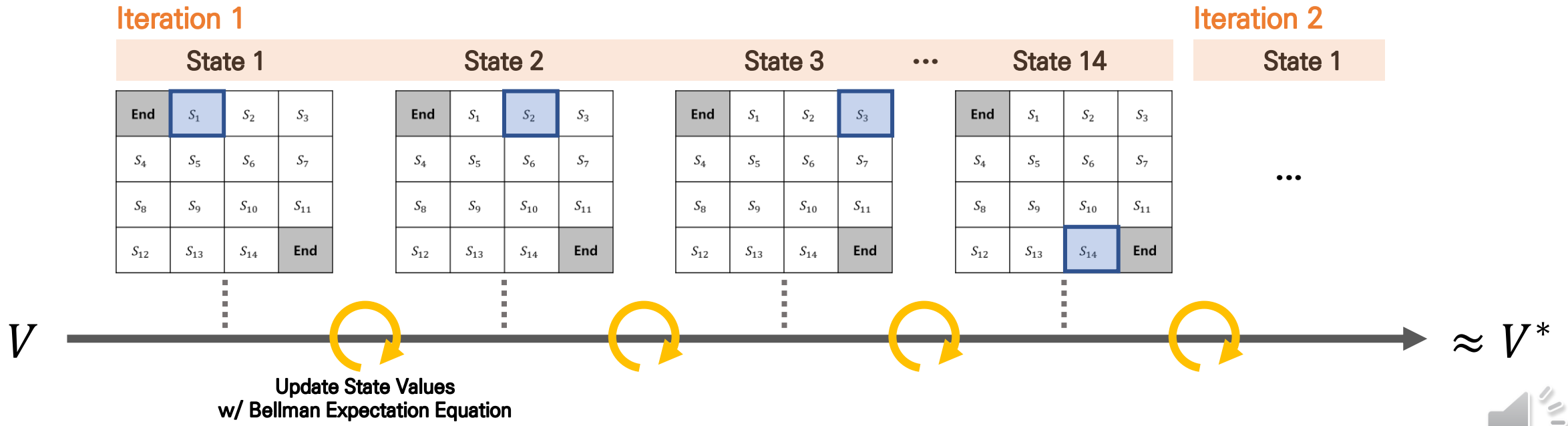
Model-Based Reinforcement Learning

Iterative Policy Evaluation

❖ Iterative Policy Evaluation (반복적 정책 평가)

- 반복적 정책 평가 (Iterative Policy Evaluation)
  - ✓ 주어진 정책으로 상태의 가치를 평가하는 문제에 사용
  - ✓ 벨만기대방정식을 반복적으로 사용하여 상태의 가치를 점진적으로 업데이트하는 방법론

$$v_\pi(s) = \sum_{a \in A} \pi(a|s) \left(r_s^a + \gamma \sum_{s' \in S} P_{ss'}^a \, v_\pi(s') \right)$$



Model-Based Reinforcement Learning

Iterative Policy Evaluation

❖ Iterative Policy Evaluation

Step 1. 상태 가치 초기화

✓ 상태 가치의 초기값을 0으로 설정함

Grid World

End	S_1	S_2	S_3
S_4	S_5	S_6	S_7
S_8	S_9	S_{10}	S_{11}
S_{12}	S_{13}	S_{14}	**End**

State Value(k=0)

0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

Model-Based Reinforcement Learning

Iterative Policy Evaluation

Action	: Left(25%), Right(25%), Up(25%), Down(25%)
Reward	: −1 per step
State	: Total 16 states including a terminal state
Decay Ratio	: 1
State Transition Probability	: 1

❖ Iterative Policy Evaluation

Step 2. 상태 가치 업데이트 (반복)

✓ 벨만기대방정식을 사용하여 상태의 가치를 업데이트

State Value(k=1)

0	0	0	0
0	0	s_6 **0**	0
0	s_9 **0**	s_{10} **-1**	s_{11} **0**
0	0	s_{14} **0**	0

$$v_\pi(s_{10}) = \sum_{a \in A} \pi(a|s_{10}) \left(r_s^a + \gamma \sum_{s' \in S} P_{ss'}^a \, v_\pi(s') \right)$$

(0.25: $\pi(a|s_{10})$, -1: r_s^a, 1: γ, 1: $P_{ss'}^a$)

$$= 0.25 * (-1 + 1 * (1 * 0 + 0 * 0 + 0 * 0 + 0 * 0) \quad \text{\# Left}$$
$$+ 0.25 * (-1 + 1 * (0 * 0 + 1 * 0 + 0 * 0 + 0 * 0) \quad \text{\# Right}$$
$$+ 0.25 * (-1 + 1 * (0 * 0 + 0 * 0 + 1 * 0 + 0 * 0) \quad \text{\# Up}$$
$$+ 0.25 * (-1 + 1 * (0 * 0 + 0 * 0 + 0 * 0 + 1 * 0) \quad \text{\# Down}$$

$$= -1.0$$

Model-Based Reinforcement Learning

Iterative Policy Evaluation

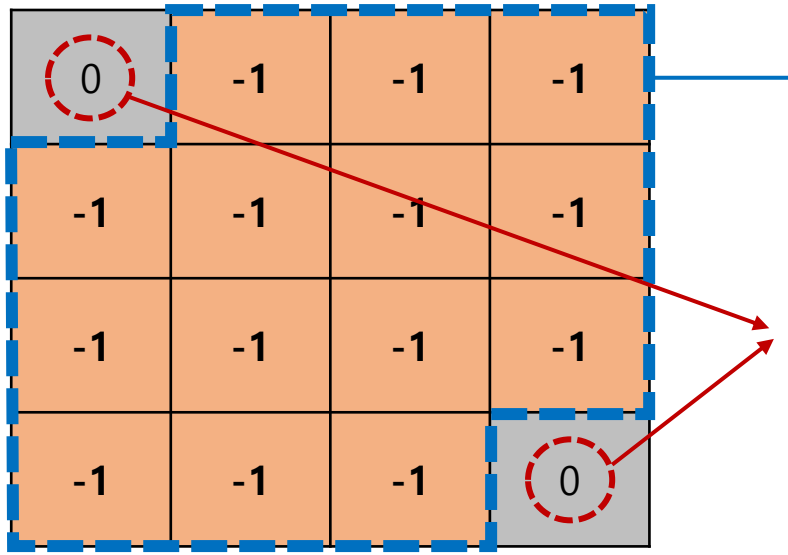
Action	: Left(25%), Right(25%), Up(25%), Down(25%)
Reward	: −1 per step
State	: Total 16 states including a terminal state
Decay Ratio	: 1
State Transition Probability	: 1

❖ Iterative Policy Evaluation

Step 2. 상태 가치 업데이트 (반복)

✓ 벨만기대방정식을 사용하여 상태의 가치를 업데이트

State Value(k=1)



s_{10}을 업데이트한 것과 동일한 방식으로 다른 state도 업데이트를 진행함

Terminal state의 경우 episode의 마지막이기 때문에 상태 가치를 업데이트하지 않음

Model-Based Reinforcement Learning

Iterative Policy Evaluation

Action	: Left(25%), Right(25%), Up(25%), Down(25%)
Reward	: −1 per step
State	: Total 16 states including a terminal state
Decay Ratio	: 1
State Transition Probability	: 1

❖ Iterative Policy Evaluation

Step 2. 상태 가치 업데이트 (반복)

✓ 벨만기대방정식을 사용하여 상태의 가치를 업데이트

State Value(k=2)

0	-1	-1	-1
-1	-1	s_6 **-1**	-1
-1	s_9 **-1**	s_{10} **-2**	s_{11} **-1**
-1	-1	s_{14} **-1**	0

$$v_\pi(s_{10}) = \sum_{a \in A} \pi(a|s_{10}) \left(r_s^a + \gamma \sum_{s' \in S} P_{ss'}^a \, v_\pi(s') \right)$$

(0.25: $\pi(a|s_{10})$, -1: r_s^a, 1: γ, 1: $P_{ss'}^a$)

$$\begin{aligned} &= 0.25 * (-1 + 1 * (1 * -1 + 0 * -1 + 0 * -1 + 0 * -1) \\ &\quad + 0.25 * (-1 + 1 * (0 * -1 + 1 * -1 + 0 * -1 + 0 * -1) \\ &\quad + 0.25 * (-1 + 1 * (0 * -1 + 0 * -1 + 1 * -1 + 0 * -1) \\ &\quad + 0.25 * (-1 + 1 * (0 * -1 + 0 * -1 + 0 * -1 + 1 * -1) \\ &= -2.0 \end{aligned}$$

Model-Based Reinforcement Learning

Iterative Policy Evaluation

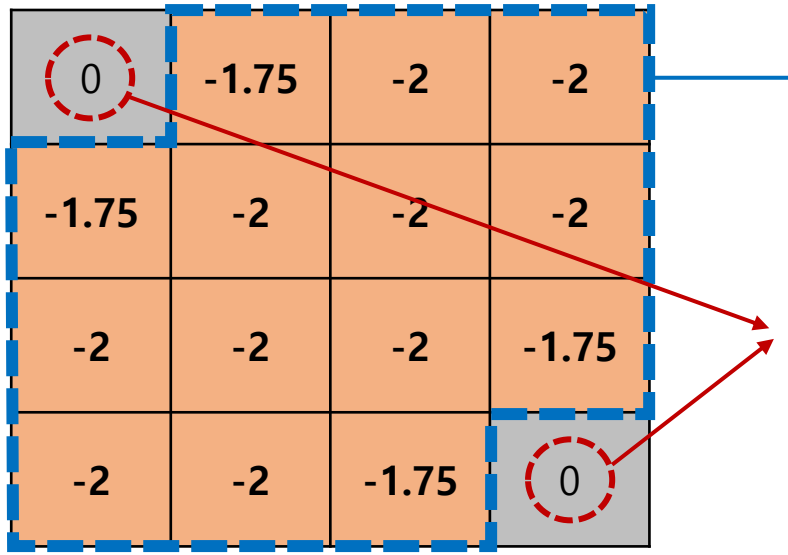
Action	: Left(25%), Right(25%), Up(25%), Down(25%)
Reward	: −1 per step
State	: Total 16 states including a terminal state
Decay Ratio	: 1
State Transition Probability	: 1

❖ Iterative Policy Evaluation

Step 2. 상태 가치 업데이트 (반복)

✓ 벨만기대방정식을 사용하여 상태의 가치를 업데이트

State Value(k=2)



s_{10}을 업데이트한 것과 동일한 방식으로 다른 state도 업데이트를 진행함

Terminal state의 경우 episode의 마지막이기 때문에 상태 가치를 업데이트하지 않음

Model-Based Reinforcement Learning

Iterative Policy Evaluation

Action	: Left(25%), Right(25%), Up(25%), Down(25%)
Reward	: −1 per step
State	: Total 16 states including a terminal state
Decay Ratio	: 1
State Transition Probability	: 1

❖ Iterative Policy Evaluation

Step 2. 상태 가치 업데이트 (반복)

✓ 벨만기대방정식을 사용하여 상태의 가치를 업데이트

State Value(k=2)

0	-1.75	-2	-2
-1.75	-2	-2	-2
-2	-2	-2	-1.75
-2	-2	-1.75	0

...

State Value(k=3)

0	-2.4	-2.9	-3
-2.4	-2.9	-3	-2.9
-2.9	-3	-2.9	-2.4
-3	-2.9	-2.4	0

...

State Value(k=∞)

0	-14	-20	-22
-14	-18	-20	-20
-20	-20	-18	-14
-22	-20	-14	0

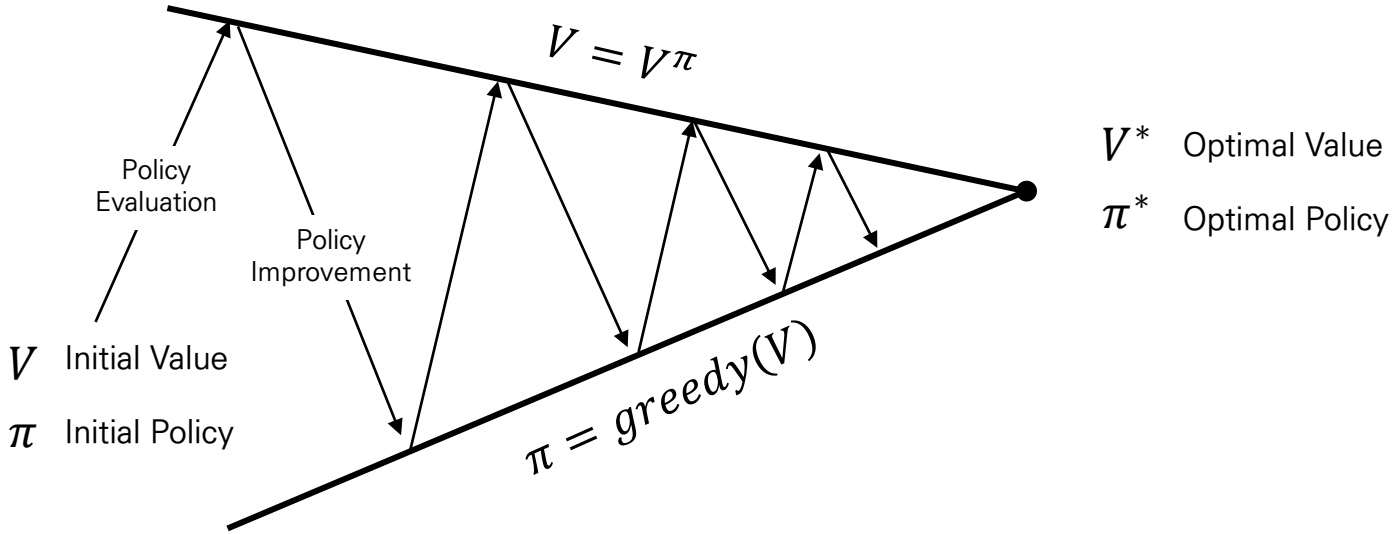
Model-Based Reinforcement Learning

Policy Iteration

❖ Policy Iteration

- 정책 이터레이션 (Policy Iteration)
  - ✓ 정책 평가와 정책 개선을 번갈아 수행하여 정책이 수렴할 때까지 반복하는 방법론

- 정책 평가: 반복적 정책 평가
- 정책 개선: 그리디 정책 개선



Model-Based Reinforcement Learning

Policy Iteration

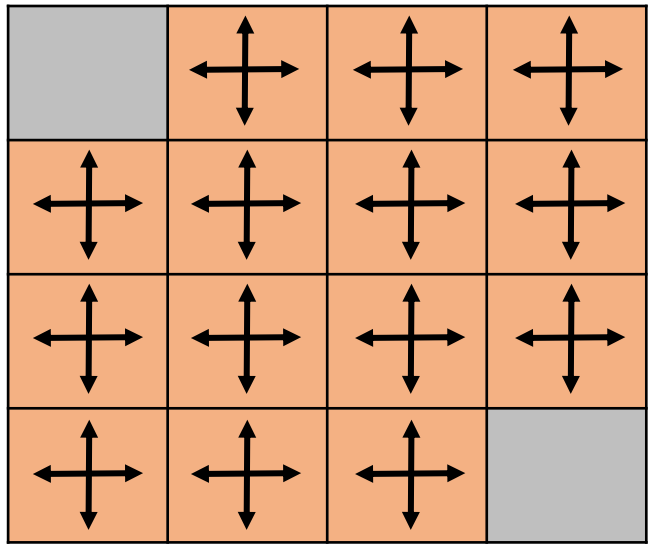
❖ Greedy Policy Improvement

- 현재 상태에서 갈 수 있는 다음 상태 중 가치가 가장 높은 쪽으로만 행동을 취하도록 하는 정책

State Value (k=1)

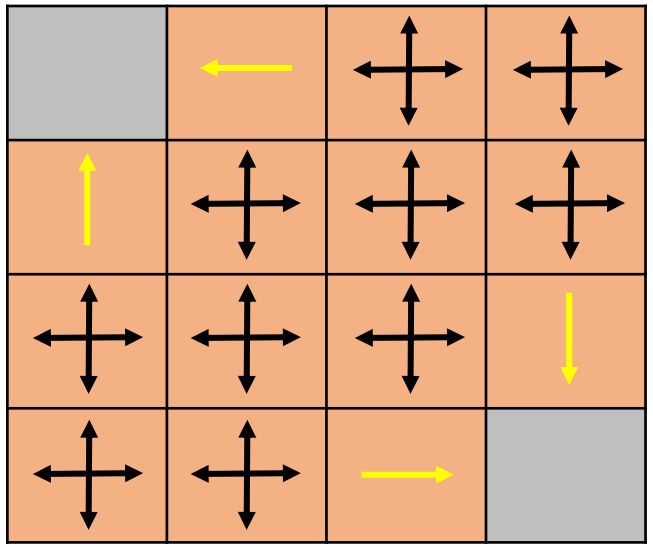
0	-1	-1	-1
-1	-1	-1	-1
-1	-1	-1	-1
-1	-1	-1	0

Random Policy (k=0)



Policy Improvement

Greedy Policy (k=1)

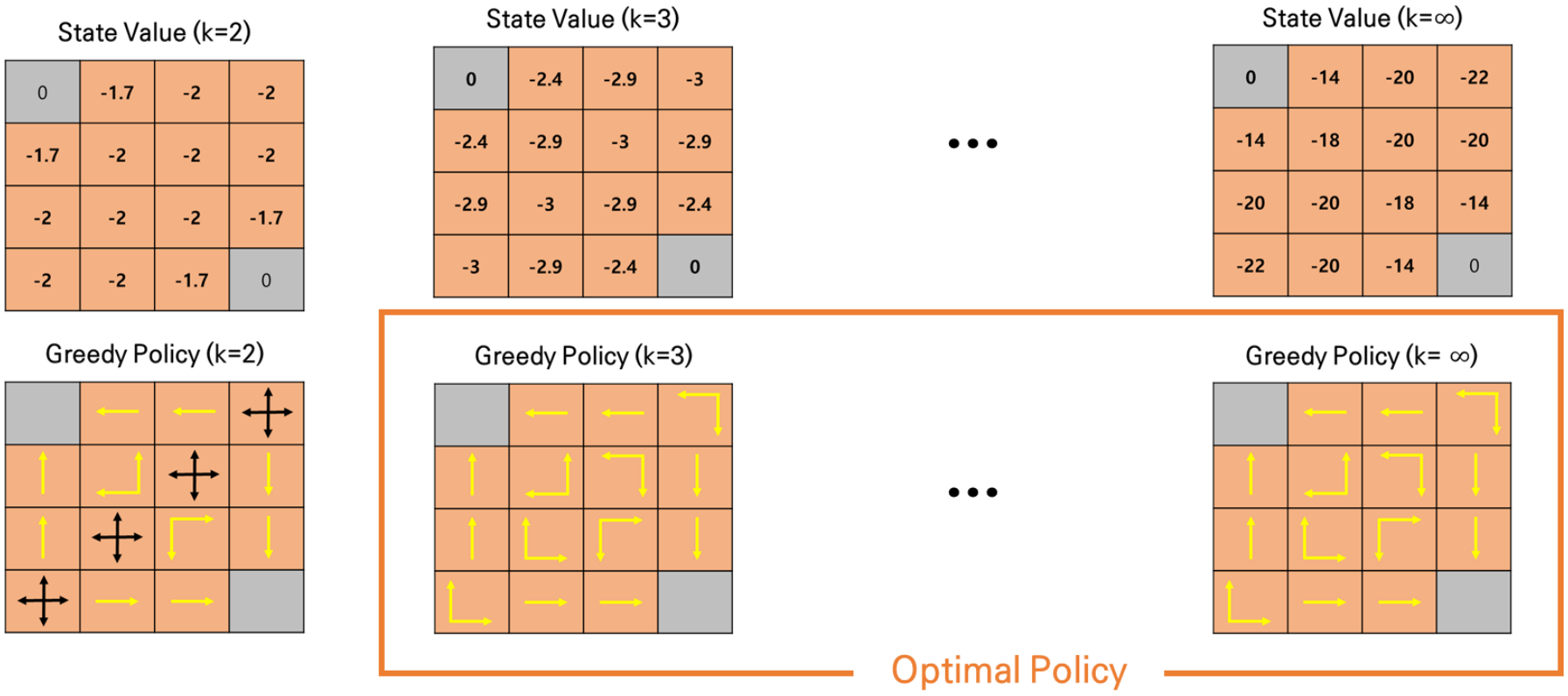


Model-Based Reinforcement Learning

Policy Iteration

❖ How many iterations are needed to finish the policy evaluation?

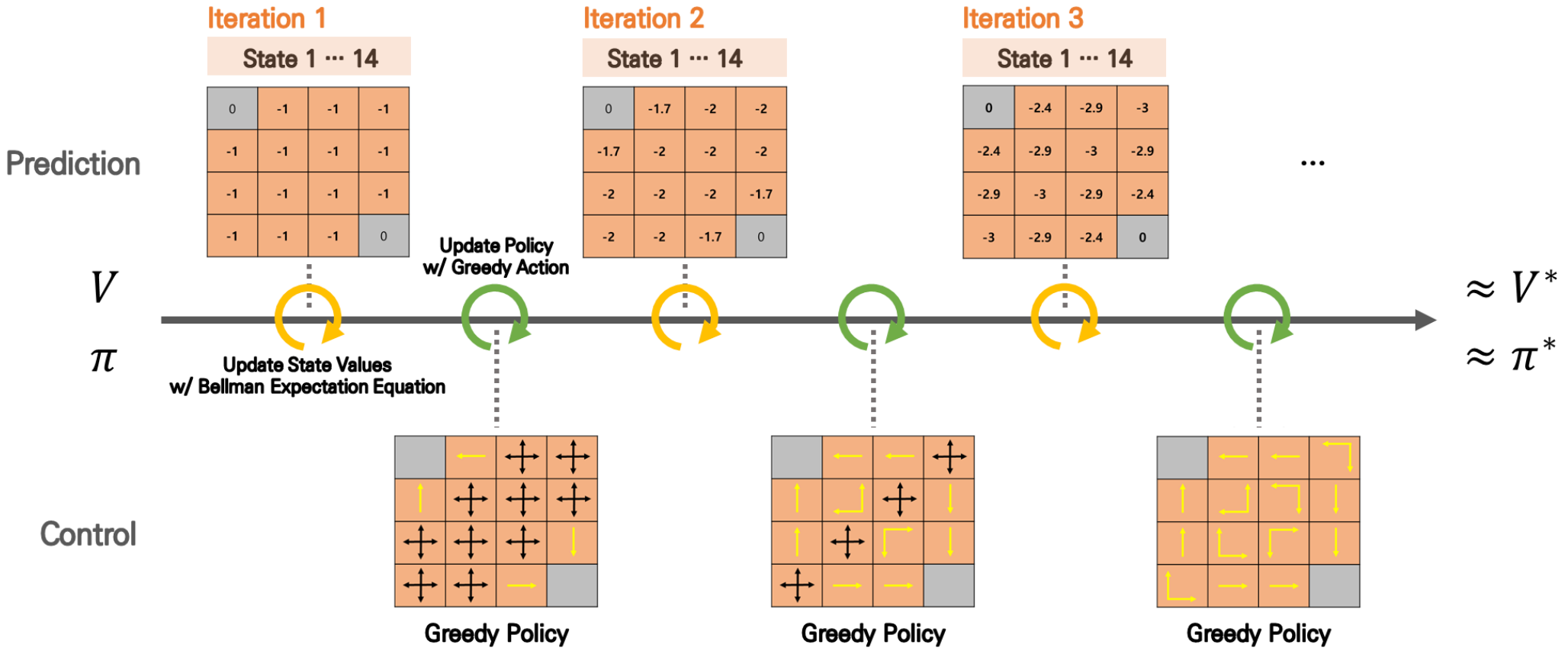
- 매 정책 평가 때마다 상태 가치가 수렴할 필요는 없으며, 정책 개선이 발생할 수 있을 만큼 루프를 진행해도 충분함
- 최적 정책을 찾는 것이 정책 이터레이션의 목적이기 때문에 정책이 수렴한 경우에는 업데이트 진행 필요 없음



Model-Based Reinforcement Learning

Policy Iteration

❖ Overall Process Visualization

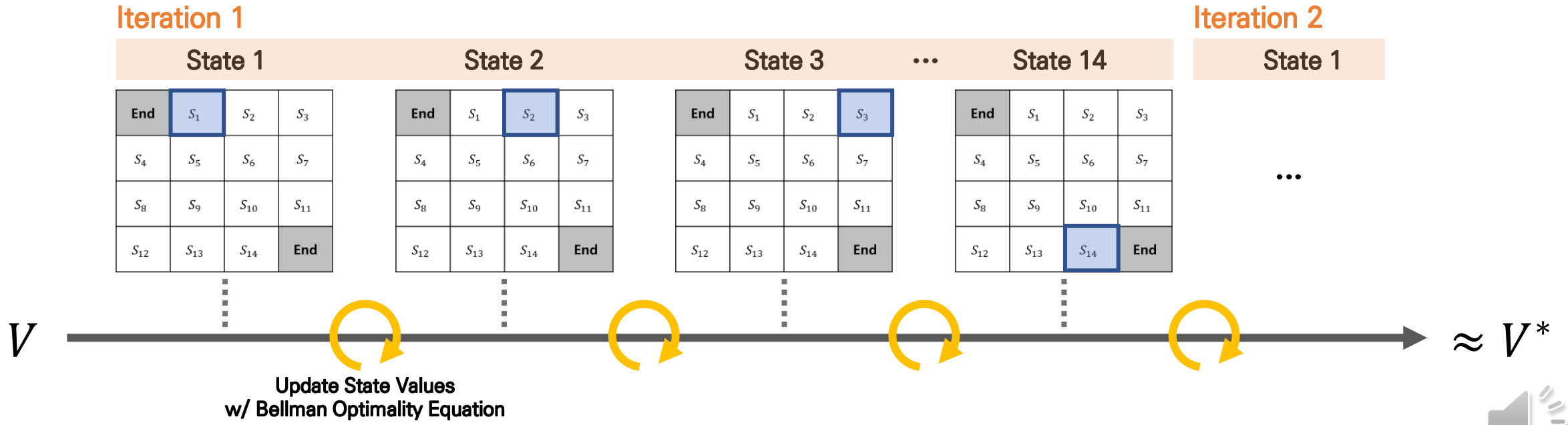


Model-Based Reinforcement Learning

Value Iteration

❖ Value Iteration

- 가치 이터레이션 (Value Iteration)
  - ✓ 환경에 의존적인 정책 평가를 진행하여 최적 가치를 구함으로써 최적 정책을 도출하는 방법론
- 벨만최적방정식을 사용하여 상태가치를 반복적으로 업데이트 함 ⟶ $v_*(s) = \max_a \left[r_s^a + \gamma \sum_{s' \in S} P_{ss'}^a v_*(s') \right]$



Model-Based Reinforcement Learning

Value Iteration

❖ Value Iteration

Step 1. 상태 가치 초기화

✓ 상태 가치의 초기값을 0으로 설정함

Grid World

End	S_1	S_2	S_3
S_4	S_5	S_6	S_7
S_8	S_9	S_{10}	S_{11}
S_{12}	S_{13}	S_{14}	**End**

⟷

State Value(k=0)

0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

Model-Based Reinforcement Learning

Value Iteration

Action	: Left(25%), Right(25%), Up(25%), Down(25%)
Reward	: −1 per step
State	: Total 16 states including a terminal state
Decay Ratio	: 1
State Transition Probability	: 1

❖ Value Iteration

Step 2. 상태 가치 업데이트 (반복)

✓ 벨만최적방정식을 사용하여 상태의 가치를 업데이트

State Value(k=1)

0	-1	-1	-1
-1	-1	s_6 -1	-1
-1	s_9 -1	s_{10} -1	s_{11} -1
-1	-1	s_{14} -1	0

$$v_*(s_{10}) = \max_a \left[\underset{\text{-1}}{r_s^a} + \underset{\text{1}}{\gamma} \sum_{s' \in S} \underset{\text{1}}{P_{ss'}^a} v_*(s') \right]$$

$$= \max(-1 + 1 * (1 * 0 + 0 * 0 + 0 * 0 + 0 * 0) \quad \text{\# Left}$$
$$-1 + 1 * (0 * 0 + 1 * 0 + 0 * 0 + 0 * 0) \quad \text{\# Right}$$
$$-1 + 1 * (0 * 0 + 0 * 0 + 1 * 0 + 0 * 0) \quad \text{\# Up}$$
$$-1 + 1 * (0 * 0 + 0 * 0 + 0 * 0 + 1 * 0)) \quad \text{\# Down}$$

$$= -1.0$$

Model-Based Reinforcement Learning

Value Iteration

Action	: Left(25%), Right(25%), Up(25%), Down(25%)
Reward	: −1 per step
State	: Total 16 states including a terminal state
Decay Ratio	: 1
State Transition Probability	: 1

❖ Value Iteration

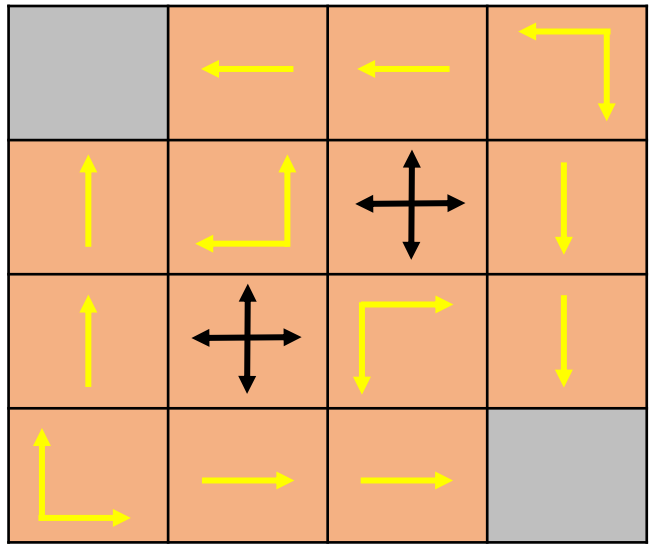
Step 2. 상태 가치 업데이트 (반복)

- ✓ 벨만최적방정식을 사용하여 상태의 가치를 업데이트
- ✓ 업데이트를 반복하여 최적 가치를 함으로써 최적 정책을 가짐

State Value(k= ∞)

0	-1	-2	-3
-1	-2	-3	-2
-2	-3	-2	-1
-3	-2	-2	0

Optimal Policy (k= ∞)



Model-Based Reinforcement Learning

Summary

❖ Prediction : 정책이 주어졌을 때 각 상태의 가치를 평가하는 문제

- 반복적 정책 평가
  - ✓ 벨만기대방정식을 사용함
  - ✓ 주어진 정책이 고정되어 있음
  - ✓ 주어진 정책으로 각 상태의 가치를 평가함

❖ Control : 최적의 정책 함수를 찾는 문제

- 정책 이터레이션
  - ✓ 정책 평가와 정책 개선이 번갈아 가며 시행됨
  - ✓ 정책 평가는 Prediction 문제로 접근하며 마찬가지로 반복적 정책 평가를 사용함 (벨만기대방정식을 사용)
  - ✓ 정책 개선은 정책 평가를 통해 업데이트 된 상태가치에 따라서 그리디 행동을 취하도록 정책을 변경함

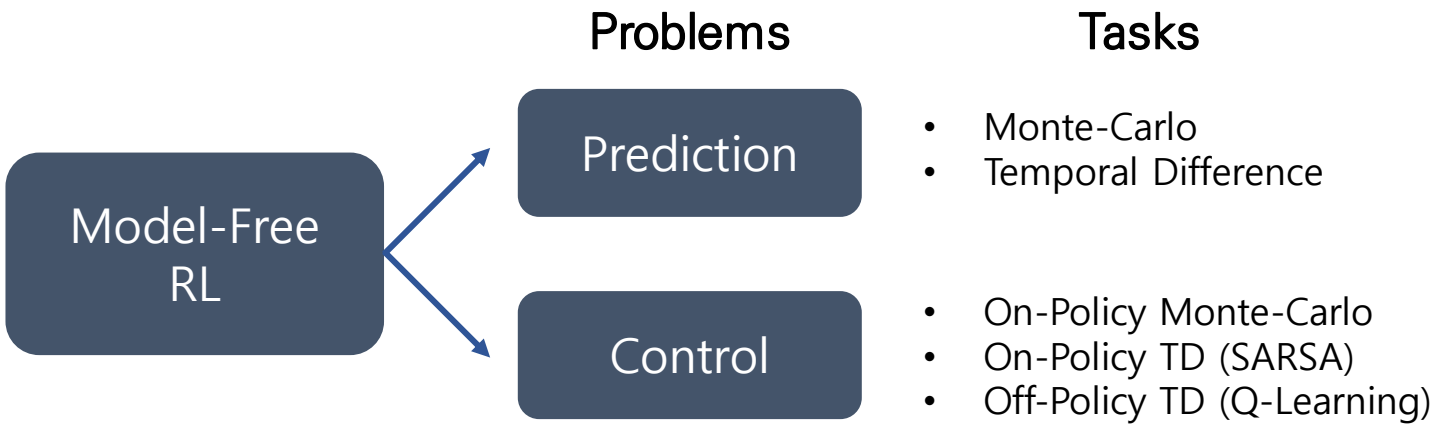
- 가치 이터레이션
  - ✓ Prediction 문제를 수행할 때 벨만최적방정식을 사용함
  - ✓ 벨만최적방정식에서 이미 그리디 행동을 취하기 때문에 최적 가치를 구함으로써 최적 정책함수를 찾게 됨

Model-Free Reinforcement Learning

Introduction

❖ Model-Free Reinforcement Learning이란?

- MDP를 알 수 없을 때 정책을 평가 및 개선해 나가는 과정
- Prediction problem
  - ✓ 정책이 주어졌을 때 각 상태의 가치를 평가하는 문제
- Control problem
  - ✓ 최적의 정책 함수를 찾는 문제



Model-Free Reinforcement Learning

Introduction

❖ Example Environment) Grid World 2

- Action : Left(25%), Right(25%), Up(25%), Down(25%)
- Reward : ? ~~-1 per step~~
- State : Total 16 states including terminal states
- Decay Ratio : 1
- State Transition Probability : ? ~~1~~

Grid World

Start	S_1	S_2	S_3
S_4	S_5	S_6	S_7
S_8	S_9	S_{10}	S_{11}
S_{12}	S_{13}	S_{14}	**End**

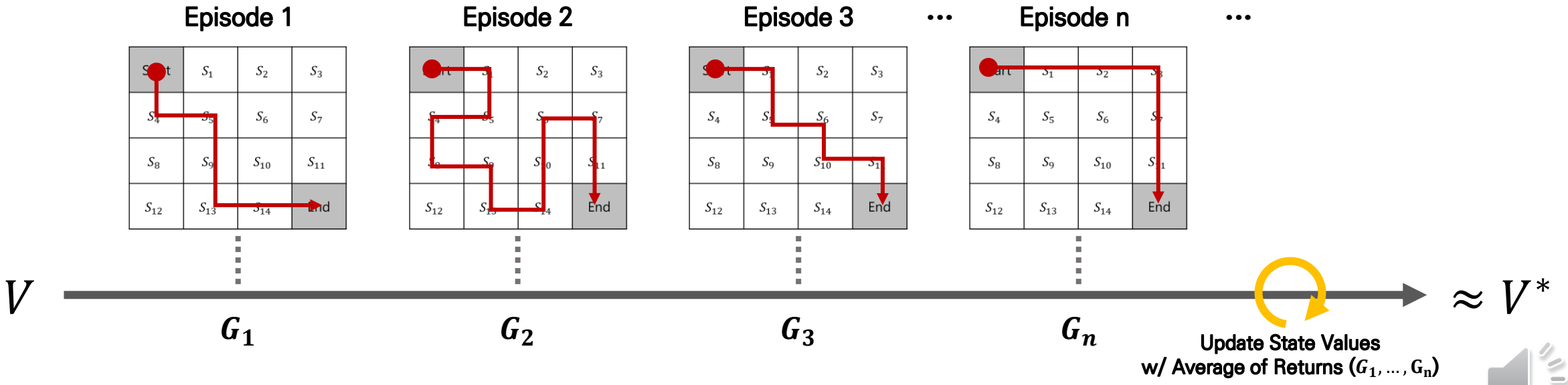
Model-Free Reinforcement Learning

Model-Free Prediction (Monte-Carlo)

❖ Monte-Carlo(MC)

- Monte-Carlo
  - ✓ 반복된 무작위 추출을 이용하여 함수의 값을 수리적으로 근사하는 알고리즘
  - ✓ 주어진 정책을 반복적으로 실행하여 얻은 리턴의 값으로 상태 가치를 근사하는 알고리즘, 이 때 대수의 법칙에 의해 예측치가 점점 정확해짐

- 종료된 에피소드에서만 학습할 수 있음, 따라서 Non-terminating Episode에서는 사용할 수 없음
- 각 상태의 가치를 평가할 때 Return 값을 사용함 $V(s_t) \leftarrow V(s_t) + \alpha\big(\boldsymbol{G_t} - V(s_t)\big)$



Model-Free Reinforcement Learning

Model-Free Prediction (Monte-Carlo)

Action	: Left(25%), Right(25%), Up(25%), Down(25%)
Reward	: ? ~~-1 per step~~
State	: Total 16 states including terminal states
Decay Ratio	: 1
State Transition Probability	: ? ~~1~~

❖ Monte-Carlo(MC)

Step 1. 상태 가치 초기화

✓ 상태 가치의 초기값을 0으로 설정함

Grid World

Start	S_1	S_2	S_3
S_4	S_5	S_6	S_7
S_8	S_9	S_{10}	S_{11}
S_{12}	S_{13}	S_{14}	**End**

State Value

0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	**End**

Model-Free Reinforcement Learning

Model-Free Prediction (Monte-Carlo)

Action	: Left(25%), Right(25%), Up(25%), Down(25%)
Reward	: ? ~~-1 per step~~
State	: Total 16 states including terminal states
Decay Ratio	: 1
State Transition Probability	: ? ~~1~~

❖ Monte-Carlo(MC)

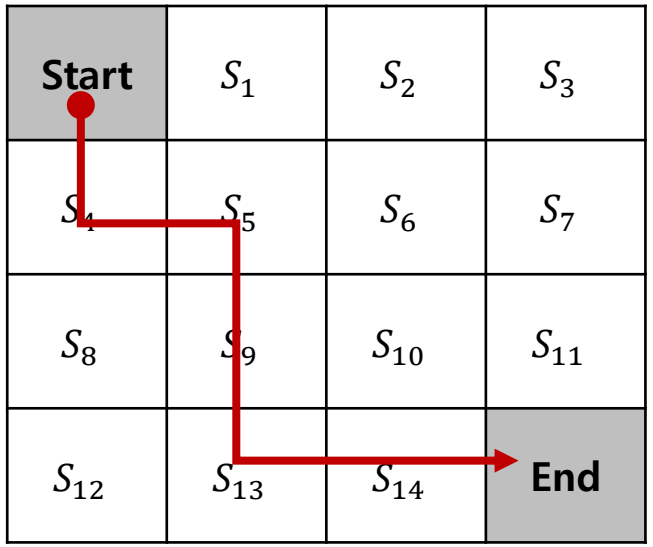
Step 2. 상태 가치 업데이트 (반복)

✓ 획득한 리턴의 값에 따라 상태의 가치를 업데이트 (수식 참고)

MC / update equation

$$V(s_t) \leftarrow V(s_t) + \alpha * \left(\boldsymbol{G_t} - V(s_t)\right)$$

Episode 1



총 6 step으로 구성된 episode

Time Step(t)	State(s_t)	Action(a_t)	Reward(r_t)	Next State(s_{t+1})	Return(G_t)
1	Start	Down	-1	State 4	-6
2	State 1	Right	-1	State 5	-5
3	State 5	Down	-1	State 9	-4
4	State 9	Down	-1	State 13	-3
5	State 13	Right	-1	State 14	-2
6	State 14	Right	-1	End	-1

$$G_1 = r_1 + \gamma r_2 + \gamma^2 r_3 + \gamma^3 r_4 + \gamma^4 r_5 + \gamma^5 r_6$$
$$= -1 + 1 * -1 + 1 * -1 + 1 * -1 + 1 * -1 + 1 * -1$$
$$= -6 \quad \text{\# Return of Start State}$$

Model-Free Reinforcement Learning

Model-Free Prediction (Monte-Carlo)

Action	: Left(25%), Right(25%), Up(25%), Down(25%)
Reward	: ? ~~-1 per step~~
State	: Total 16 states including terminal states
Decay Ratio	: 1
State Transition Probability	: ? ~~1~~

❖ Monte-Carlo(MC)

Step 2. 상태 가치 업데이트 (반복)

✓ 획득한 리턴의 값에 따라 상태의 가치를 업데이트 (수식 참고)

MC / update equation

$$V(s_t) \leftarrow V(s_t) + \alpha * \left(\boldsymbol{G_t} - V(s_t)\right)$$

Episode 1

Start	S_1	S_2	S_3
S_4	S_5	S_6	S_7
S_8	S_9	S_{10}	S_{11}
S_{12}	S_{13}	S_{14}	End

Return(G_t)

-6	0	0	0
-5	-4	0	0
0	-3	0	0
0	-2	-1	End

Model-Free Reinforcement Learning

Model-Free Prediction (Monte-Carlo)

Action	: Left(25%), Right(25%), Up(25%), Down(25%)
Reward	: ? ~~-1 per step~~
State	: Total 16 states including terminal states
Decay Ratio	: 1
State Transition Probability	: ? ~~1~~

❖ Monte-Carlo(MC)

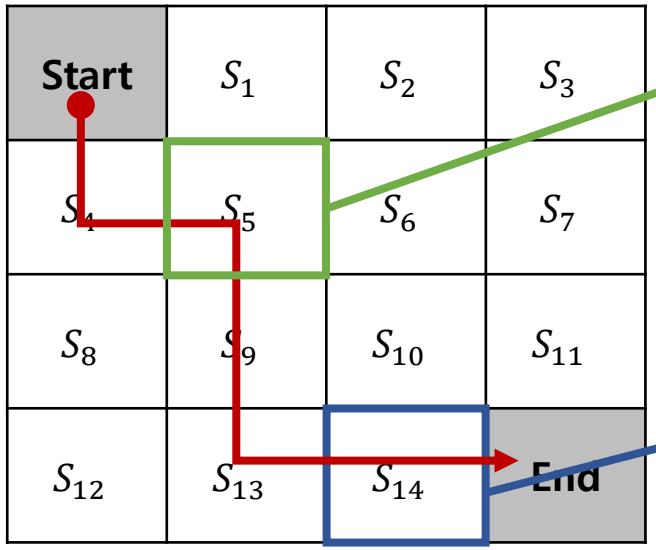
MC / update equation

$$V(s_t) \leftarrow V(s_t) + \alpha * \left(\boldsymbol{G_t} - V(s_t)\right)$$

Step 2. 상태 가치 업데이트 (반복)

✓ 획득한 리턴의 값에 따라 상태의 가치를 업데이트 (수식 참고)

Episode 1



-4 리턴(G) : -4

$$V(s_3) \leftarrow V(s_3) + \alpha\left(\boldsymbol{G_3} - V(s_3)\right)$$
$$= 0 + 0.001(\boldsymbol{-4} - 0) = -0.004$$

** α가 0.001 일 때

-1 리턴(G) : -1

$$V(s_6) \leftarrow V(s_6) + \alpha\left(\boldsymbol{G_6} - V(s_6)\right)$$
$$= 0 + 0.001(\boldsymbol{-1} - 0) = -0.001$$

** α가 0.001 일 때

Model-Free Reinforcement Learning

Model-Free Prediction (Monte-Carlo)

Action	: Left(25%), Right(25%), Up(25%), Down(25%)
Reward	: ? ~~-1 per step~~
State	: Total 16 states including terminal states
Decay Ratio	: 1
State Transition Probability	: ? ~~1~~

❖ Monte-Carlo(MC)

Step 2. 상태 가치 업데이트 (반복)

✓ 획득한 리턴의 값에 따라 상태의 가치를 업데이트 (수식 참고)

MC / update equation

$$V(s_t) \leftarrow V(s_t) + \alpha * \left(\boldsymbol{G_t} - V(s_t)\right)$$

Episode 1

Start	S_1	S_2	S_3
S_4	S_5	S_6	S_7
S_8	S_9	S_{10}	S_{11}
S_{12}	S_{13}	S_{14}	End

State Value

-0.006	0	0	0
-0.005	-0.004	0	0
0	-0.003	0	0
0	-0.002	-0.001	End

Model-Free Reinforcement Learning

Model-Free Prediction (Monte-Carlo)

Action	: Left(25%), Right(25%), Up(25%), Down(25%)
Reward	: ? ~~-1 per step~~
State	: Total 16 states including terminal states
Decay Ratio	: 1
State Transition Probability	: ? ~~1~~

❖ Monte-Carlo(MC)

MC / update equation

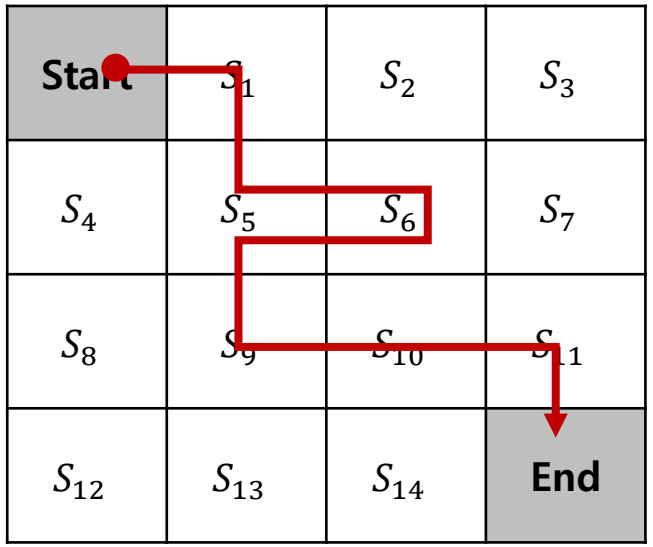
$$V(s_t) \leftarrow V(s_t) + \alpha * \left(\boldsymbol{G_t} - V(s_t)\right)$$

Step 2. 상태 가치 업데이트 (반복)

✓ 만일 같은 상태를 두 번 이상 방문하게 될 경우, 리턴 값을 계산하는 방법은 사용자가 지정함

ex) First-Visit / **Every-Visit / Incremental**

Episode 2



총 8 step으로 구성된 episode

Time Step(t)	State(s_t)	Action(a_t)	Reward(r_t)	Next State(s_{t+1})	Return(G_t)
1	Start	Right	-1	State 1	-8
2	State 1	Down	-1	State 5	-7
3	State 5	Right	-1	State 6	-6
4	State 6	Left	-1	State 5	-5
5	State 5	Down	-1	State 9	-4
6	State 9	Right	-1	State 10	-3
7	State 10	Right	-1	State 11	-2
8	State 11	Down	-1	End	-1

Model-Free Reinforcement Learning

Model-Free Prediction (Monte-Carlo)

Action	: Left(25%), Right(25%), Up(25%), Down(25%)
Reward	: ? ~~-1 per step~~
State	: Total 16 states including terminal states
Decay Ratio	: 1
State Transition Probability	: ? ~~1~~

❖ Monte-Carlo(MC)

Step 2. 상태 가치 업데이트 (반복)

✓ 획득한 리턴의 값에 따라 상태의 가치를 업데이트 (수식 참고)

MC / update equation

$$V(s_t) \leftarrow V(s_t) + \alpha * \left(\boldsymbol{G_t} - V(s_t)\right)$$

Episode 2

Start	S_1	S_2	S_3
S_4	S_5	S_6	S_7
S_8	S_9	S_{10}	S_{11}
S_{12}	S_{13}	S_{14}	End

Return(G_t)

-8	-7	(0, 0)	(0, 0)
(0, 0)	(-4, -6)	-5	(0, 0)
(0, 0)	-3	-2	-1
(0, 0)	(0, 0)	(0, 0)	End

Model-Free Reinforcement Learning

Model-Free Prediction (Monte-Carlo)

Action	: Left(25%), Right(25%), Up(25%), Down(25%)
Reward	: ? ~~-1 per step~~
State	: Total 16 states including terminal states
Decay Ratio	: 1
State Transition Probability	: ? ~~1~~

❖ Monte-Carlo(MC)

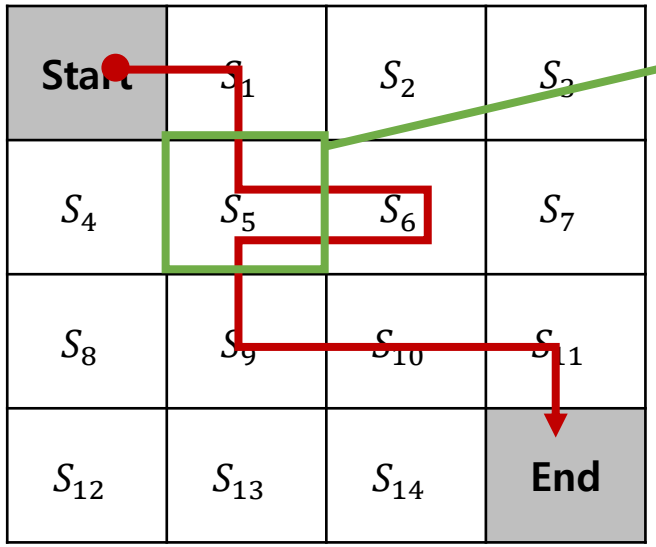
MC / update equation

$$V(s_t) \leftarrow V(s_t) + \alpha * \left(\boldsymbol{G_t} - V(s_t)\right)$$

Step 2. 상태 가치 업데이트 (반복)

✓ 획득한 리턴의 값에 따라 상태의 가치를 업데이트 (수식 참고)

Episode 2



-4, -6 리턴(G) : −4, −6

$$V(s_5) \leftarrow V(s_5) + \alpha\left(\boldsymbol{G_5} - V(s_5)\right)$$
$$= -0.004 + 0.001(-4 + 0.004) = -0.008$$

** α가 0.001 일 때

$$V(s_3) \leftarrow V(s_3) + \alpha\left(\boldsymbol{G_3} - V(s_3)\right)$$
$$= -0.008 + 0.001(-6 + 0.008) = -0.014$$

** α가 0.001 일 때

Model-Free Reinforcement Learning

Model-Free Prediction (Monte-Carlo)

Action	: Left(25%), Right(25%), Up(25%), Down(25%)
Reward	: ? ~~-1 per step~~
State	: Total 16 states including terminal states
Decay Ratio	: 1
State Transition Probability	: ? ~~1~~

❖ Monte-Carlo(MC)

Step 2. 상태 가치 업데이트 (반복)

✓ 획득한 리턴의 값에 따라 상태의 가치를 업데이트 (수식 참고)

MC / update equation

$$V(s_t) \leftarrow V(s_t) + \alpha * \left(\boldsymbol{G_t} - V(s_t)\right)$$

State Value(Episode 1)

-0.006	0	0	0
-0.005	-0.004	0	0
0	-0.003	0	0
0	-0.002	-0.001	End

State Value(Episode 2)

-0.006	-0.007	0	0
-0.005	-0.014	-0.005	0
0	-0.006	-0.002	-0.001
0	-0.002	-0.001	End

Model-Free Reinforcement Learning

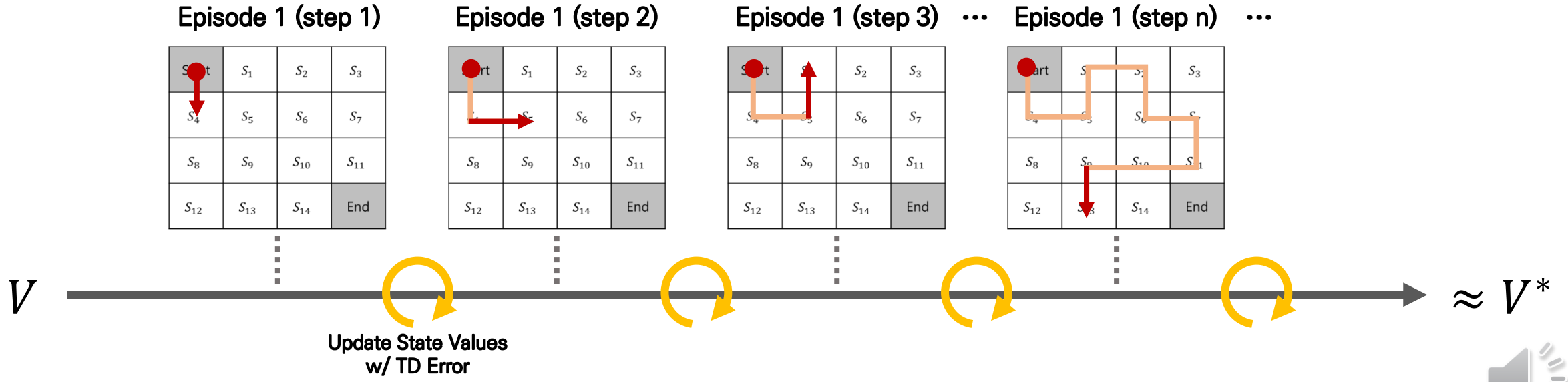
Model-Free Prediction (Temporal Difference)

$$V(s_t) \leftarrow V(s_t) + \alpha * \underbrace{\big(\underbrace{r_{t+1} + \gamma V(s_{t+1})}_{\text{TD Target}} - V(s_t)\big)}_{\text{TD Error}}$$

❖ Temporal Difference(TD)

- Temporal Difference
  - ✓ 미래의 추측으로 현재의 추측을 업데이트하는 방법론
  - ✓ 조금이라도 시간이 흐르면 좀 더 정확한 추측을 할 수 있다고 가정하여 미래의 추측을 일종의 정답지로 활용하는 것

- 에피소드가 진행되는 중간에도 학습할 수 있음, 따라서 Non-terminating Episode에서도 사용할 수 있음
- 각 상태의 가치를 평가할 때 TD error 값을 사용함



Model-Free Reinforcement Learning

Model-Free Prediction (Temporal Difference)

Action	: Left(25%), Right(25%), Up(25%), Down(25%)
Reward	: ? ~~-1 per step~~
State	: Total 16 states including terminal states
Decay Ratio	: 1
State Transition Probability	: ? ~~1~~

❖ Temporal Difference(TD)

Step 1. 상태 가치 초기화

✓ 상태 가치의 초기값을 0으로 설정함

Grid World

Start	S_1	S_2	S_3
S_4	S_5	S_6	S_7
S_8	S_9	S_{10}	S_{11}
S_{12}	S_{13}	S_{14}	**End**

⟷

State Value

0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	**End**

Model-Free Reinforcement Learning

Model-Free Prediction (Temporal Difference)

Action	: Left(25%), Right(25%), Up(25%), Down(25%)
Reward	: ? ~~-1 per step~~
State	: Total 16 states including terminal states
Decay Ratio	: 1
State Transition Probability	: ? ~~1~~

❖ Temporal Difference(TD)

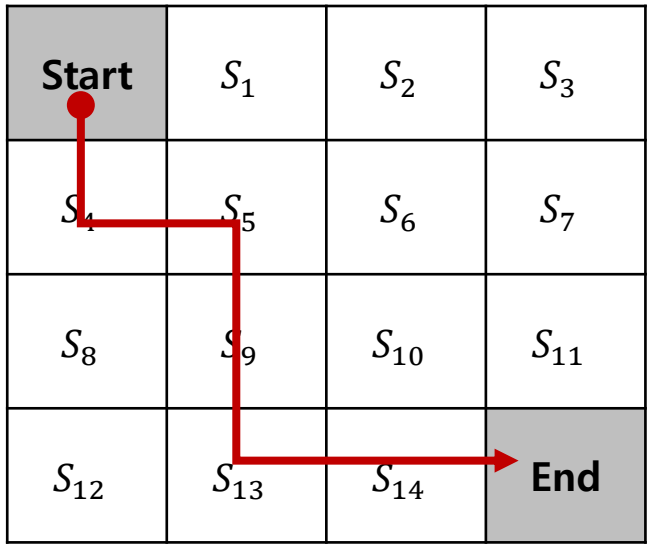
Step 2. 상태 가치 업데이트 (반복)

✓ 획득한 리턴의 값에 따라 상태의 가치를 업데이트 (수식 참고)

TD / update equation

$$V(s_t) \leftarrow V(s_t) + \alpha * \left(\boldsymbol{r_t + \gamma V(s_{t+1})} - V(s_t)\right)$$

Episode 1



총 6 step으로 구성된 episode

Time Step(t)	State(s_t)	Action(a_t)	Reward(r_t)	Next State(s_{t+1})
1	Start	Down	-1	State 4
2	State 1	Right	-1	State 5
3	State 5	Down	-1	State 9
4	State 9	Down	-1	State 13
5	State 13	Right	-1	State 14
6	State 14	Right	-1	End

Model-Free Reinforcement Learning

Model-Free Prediction (Temporal Difference)

Action	: Left(25%), Right(25%), Up(25%), Down(25%)
Reward	: ? ~~-1 per step~~
State	: Total 16 states including terminal states
Decay Ratio	: 1
State Transition Probability	: ? ~~1~~

❖ Temporal Difference(TD)

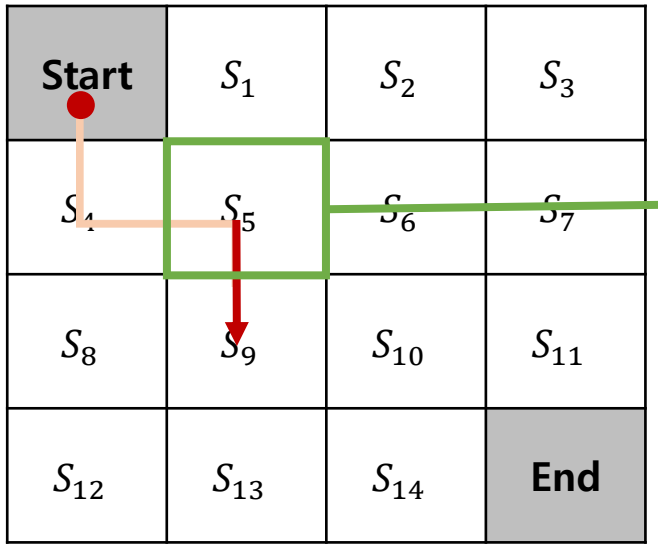
TD / update equation

$$V(s_t) \leftarrow V(s_t) + \alpha * \left(r_{t+1} + \gamma V(s_{t+1}) - V(s_t)\right)$$

Step 2. 상태 가치 업데이트 (반복)

✓ 획득한 TD error에 따라 상태의 가치를 업데이트 (수식 참고)

Episode 1 (step 3)



$$V(s) \leftarrow V(s) + \alpha\left(r' + \gamma V(s') - V(s)\right)$$
$$= 0 + 0.01(-1 + 1 * 0 - 0) = -0.01$$

** α가 0.01 일 때

Model-Free Reinforcement Learning

Model-Free Prediction (Temporal Difference)

Action	: Left(25%), Right(25%), Up(25%), Down(25%)
Reward	: ? ~~-1 per step~~
State	: Total 16 states including terminal states
Decay Ratio	: 1
State Transition Probability	: ? ~~1~~

❖ Temporal Difference(TD)

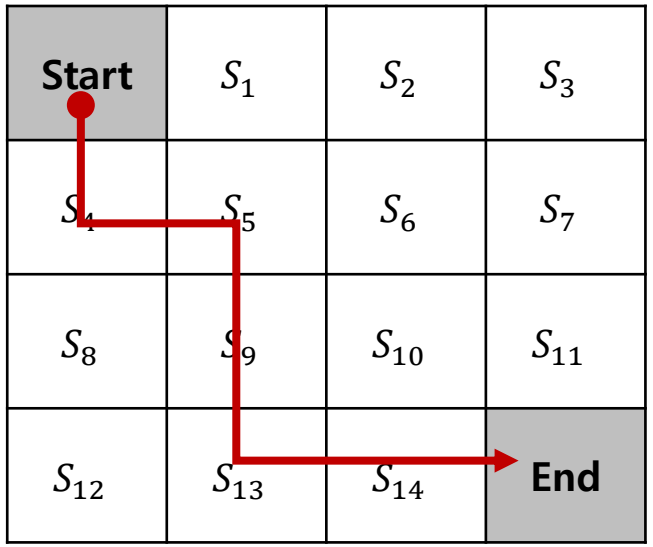
Step 2. 상태 가치 업데이트 (반복)

✓ 획득한 TD error에 따라 상태의 가치를 업데이트 (수식 참고)

TD / update equation

$$V(s_t) \leftarrow V(s_t) + \alpha * \left(\boldsymbol{r_{t+1}} + \boldsymbol{\gamma V(s_{t+1})} - V(s_t)\right)$$

Episode 1 (steps 1 ~ 6)



State Value

-0.01	0	0	0
-0.01	-0.01	0	0
0	-0.01	0	0
0	-0.01	-0.01	End

Model-Free Reinforcement Learning

Model-Free Prediction (MC vs. TD)

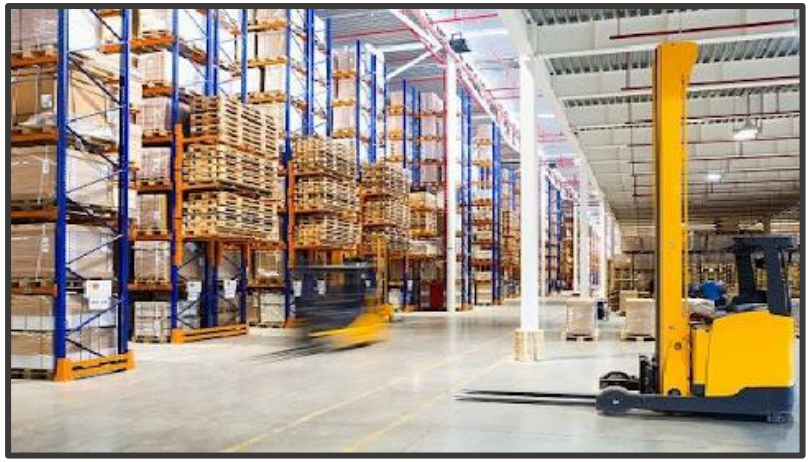
❖ MC, TD 방법론 간의 비교 (학습 시점)

- MC 방법론은 에피소드 단위로 상태 가치를 업데이트함, 에피소드가 종료되어야 학습이 가능함
- TD 방법론은 스텝 단위로 상태 가치를 업데이트함, 에피소드가 종료되지 않아도 학습이 가능함
- 따라서 학습 시점 측면에서는 TD 방법론이 더 효과적임

Terminating Environments

ex) 전략 시뮬레이션 게임

Non-Terminating Environments



ex) 24시간 가동되는 물류센터/공장

Model-Free Reinforcement Learning

Model-Free Prediction (MC vs. TD)

❖ MC, TD 방법론 간의 비교 (편향성)

- MC 방법론은 에피소드 단위로 상태 가치를 업데이트함, 여러 번의 업데이트로 얻은 리턴의 평균은 실제 가치에 수렴하게 됨
- TD 방법론은 스텝 단위로 상태 가치를 업데이트, 실제 가치 함수를 알 수 없어서 주어진 상태의 가치를 사용하기 때문에 수렴이 보장 안 됨
- 따라서 학습의 편향성 측면에서는 MC 방법론이 더 효과적임

Updating Equation

MC method

$$V(s_t) \leftarrow V(s_t) + \alpha * \left(\boldsymbol{G_t} - V(s_t)\right)$$

$$v_\pi(s_t) = \mathbb{E}[\boldsymbol{G_t}]$$

가치 함수의 정의

TD method

$$V(s_t) \leftarrow V(s_t) + \alpha * \left(\boldsymbol{r_t + \gamma V(s_{t+1})} - V(s_t)\right)$$

$$v_\pi(s_t) = \mathbb{E}[\boldsymbol{r_t + \gamma v_\pi(s_{t+1})}]$$

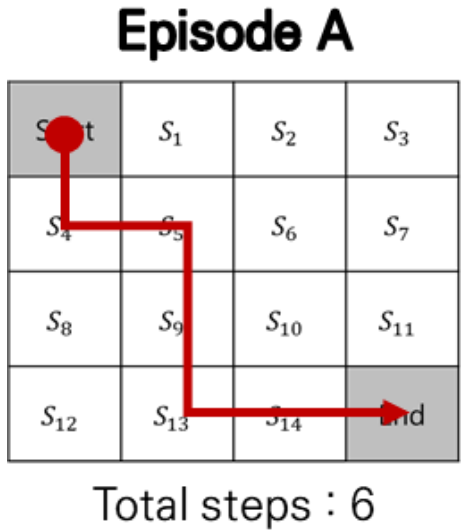
벨만 기대 방정식

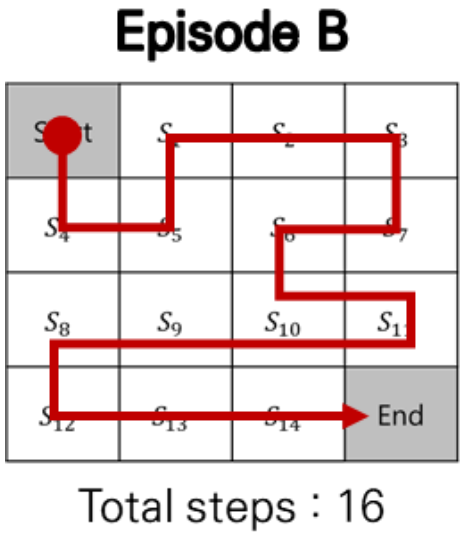
Model-Free Reinforcement Learning

Model-Free Prediction (MC vs. TD)

❖ MC, TD 방법론 간의 비교 (분산)

- MC 방법론은 에피소드 단위로 상태 가치를 업데이트함, 업데이트 주기가 유동적이므로 학습의 분산이 큼
- TD 방법론은 스텝 단위로 상태 가치를 업데이트, 업데이트 주기가 고정되어 있으므로 학습의 분산이 작음
- 따라서 학습의 분산 측면에서는 TD 방법론이 더 효과적임





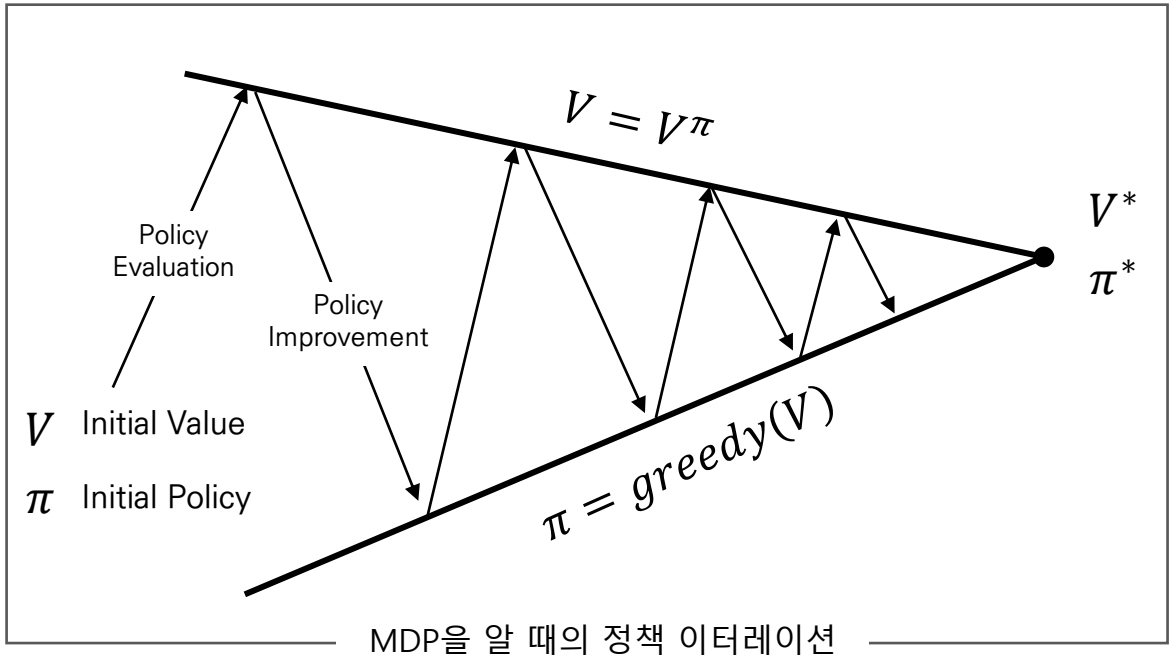
Model-Free Reinforcement Learning

Model-Free Control

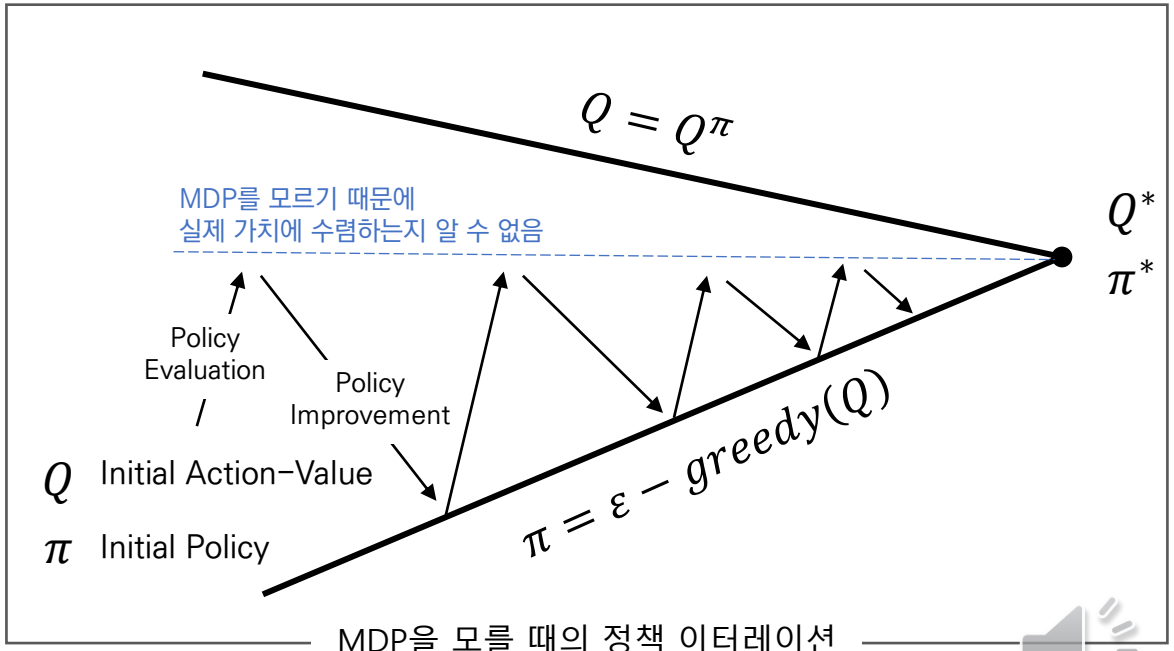
❖ Improving Policy in Model-Free RL

- MDP를 모르는 상황에서 기존의 반복적 정책 평가 방법론은 사용할 수 없음
  → MDP를 모르는 상황에서 상태 또는 상태-행동의 가치를 업데이트하는 방법론인 Monte-Carlo와 Temporal Difference를 정책 평가에 사용함

- 상태 가치만으로는 상태 전이 확률을 알 수 없기 때문에 어떤 행동으로 어떤 상태에 도달하는지 알 수 없기 때문에 정책 개선을 할 수 없음
  → 상태-행동 가치를 통해서 더 높은 상태-행동 가치를 가진 행동을 선택하도록 정책 개선을 함



MDP을 알 때의 정책 이터레이션



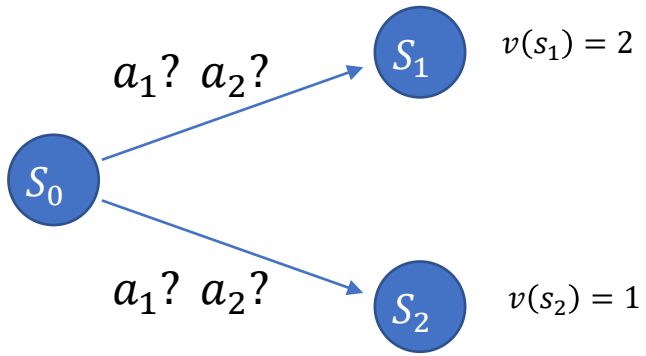
MDP을 모를 때의 정책 이터레이션

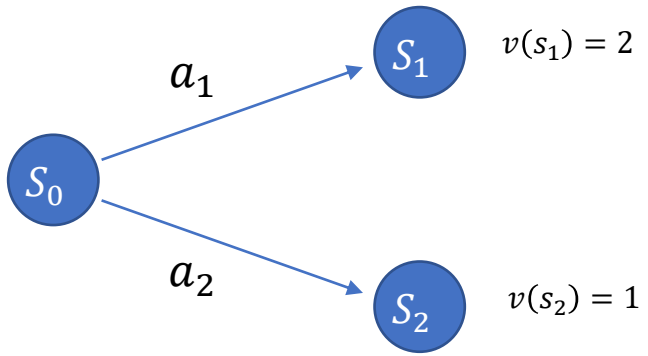
Model-Free Reinforcement Learning

Model-Free Control (Introduction)

❖ Greedy Action in Model Free Control

- MDP를 모르는 상황에서는 상태 전이 확률을 모르기 때문에 각 행동을 선택했을 때 다음 상태가 어디가 될 지 알 수 없음
- 행동에 따라 도착하는 상태를 비교할 수 없기 때문에 정책을 개선할 수 없음





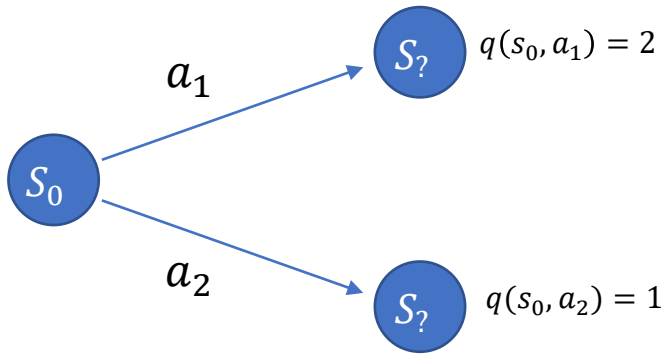
	MDP를 모르는 경우	MDP를 아는 경우
State Value	$v_\pi(\mathrm{s_t}) = \mathbb{E}_\pi[r_{t+1} + \gamma v_\pi(s_{t+1})]$	$v_\pi(s) = \sum_{a \in A} \pi(a\|s) \left(\boldsymbol{r_s^a} + \gamma \sum_{s' \in S} \boldsymbol{P_{ss'}^a}\, v_\pi(s') \right)$
State-Action Value	$q_\pi(\mathrm{s_t}, \mathrm{a_t}) = \mathbb{E}_\pi[r_{t+1} + \gamma q_\pi(s_{t+1}, a_{t+1})]$	$q_\pi(s,a) = \mathbf{r_s^a} + \gamma \sum_{s' \in S} \boldsymbol{P_{ss'}^a} \sum_{a' \in A} \pi(a'\|s')\, q_\pi(s', a')$

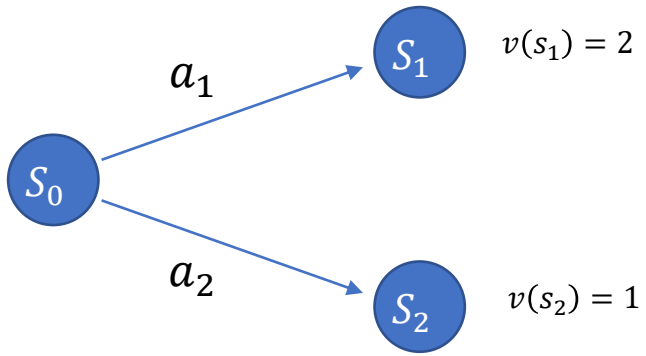
Model-Free Reinforcement Learning

Model-Free Control (Introduction)

❖ Greedy Action in Model Free Control

- 가치 함수 대신에 행동-가치 함수를 쓸 경우 MDP 정보를 몰라도 그리디 행동을 취할 수 있음
- 상태전이확률을 모르기 때문에 선택한 액션이 어느 상태로 데려가는지는 모르지만 가장 높은 가치에 도달하는 것은 알 수 있음





	MDP를 모르는 경우	MDP를 아는 경우
State Value	$v_\pi(s_t) = \mathbb{E}_\pi[r_{t+1} + \gamma v_\pi(s_{t+1})]$	$v_\pi(s) = \sum_{a \in A} \pi(a\|s) \left(\boldsymbol{r_s^a} + \gamma \sum_{s' \in S} \boldsymbol{P_{ss'}^a} \, v_\pi(s') \right)$
State-Action Value	$q_\pi(s_t, a_t) = \mathbb{E}_\pi[r_{t+1} + \gamma q_\pi(s_{t+1}, a_{t+1})]$	$q_\pi(s, a) = \mathbf{r_s^a} + \gamma \sum_{s' \in S} \boldsymbol{P_{ss'}^a} \sum_{a' \in A} \pi(a'\|s') \, q_\pi(s', a')$

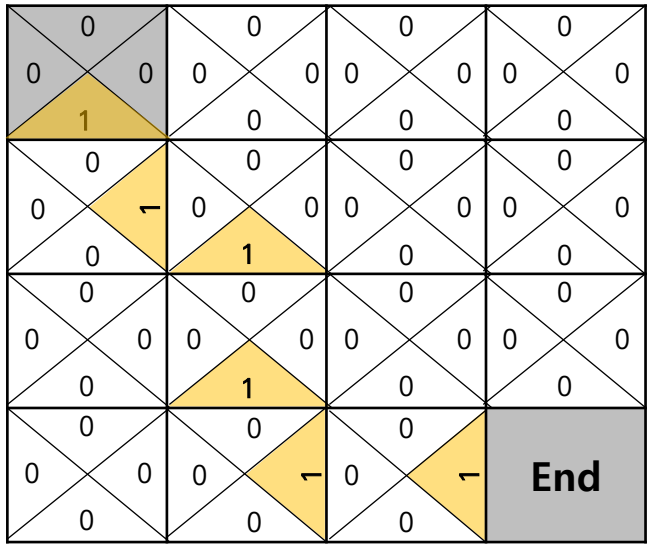
Model-Free Reinforcement Learning

Model-Free Control

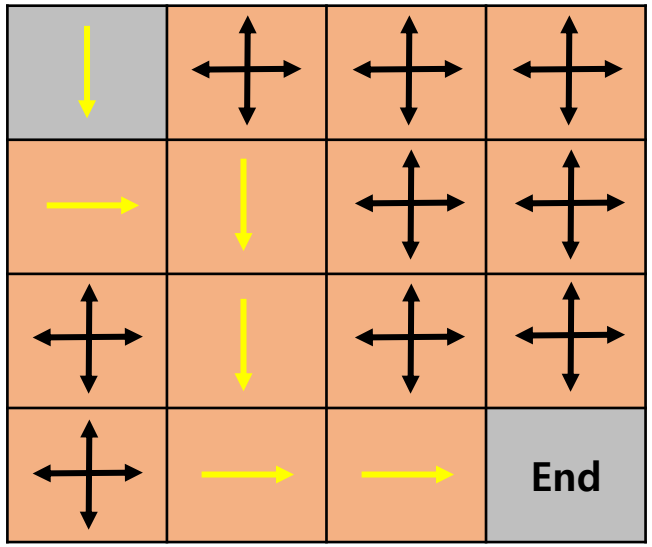
❖ ε-Greedy Policy

- Model-Free의 경우 경험한 상태-행동 가치에 대해서만 업데이트 됨
- 따라서 정책 개선 후에 특정 상태에서 특정 행동만 선택하도록 패턴이 고착될 수 있음
- 이를 타파하기 위해서 일정 확률로 무작위 행동을 선택하도록 하는 장치가 필요함

Action-State Value (Episode 1)



Greedy Policy



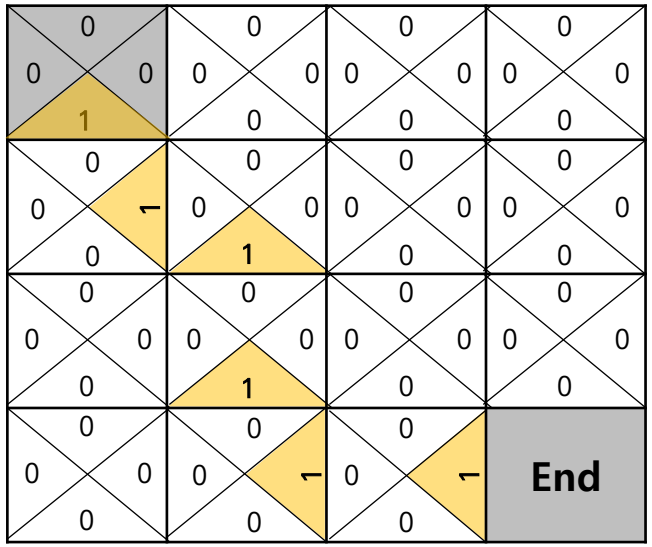
Model-Free Reinforcement Learning

Model-Free Control

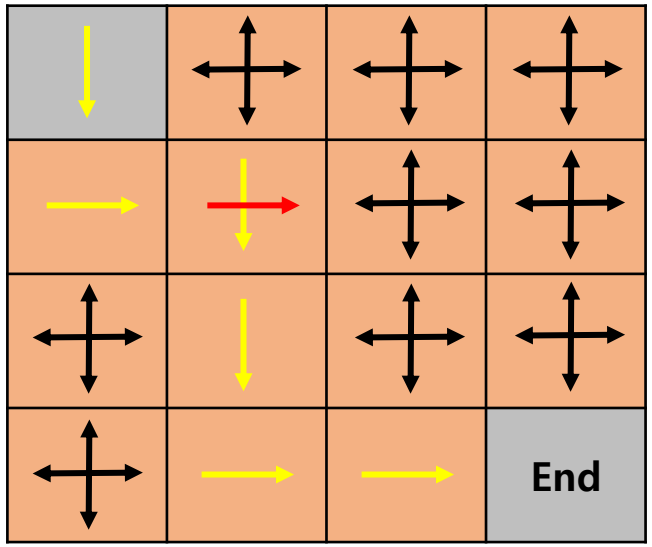
❖ ε-Greedy Policy

- 작은 확률(ε)로 랜덤한 액션을 선택하는 정책
- 그리디 행동을 취하면서도 다른 환경에 대한 정보를 탐색할 수 있음
- 시작할 때 ε 값을 높게 준 뒤 점점 줄이는 방식으로 초반에 환경에 대한 탐색에 중심을 두어 정보를 효과적으로 탐색하는 전략도 있음

Action-State Value (Episode 1)



ε-Greedy Policy



Model-Free Reinforcement Learning

Model-Free Control (On-policy vs. Off-Policy)

❖ On-Policy, Off-Policy 방법론 간의 비교

- Behavior Policy : 실제로 환경과 상호 작용하며 경험을 쌓는 정책, 정책 평가 단계에서 사용됨
- Target Policy : 강화하고자 하는 목표가 되는 정책, 정책 개선 단계에서 사용 됨

- Behavior Policy와 Target Policy가 같은 경우 → On-Policy Learning
- Behavior Policy와 Target Policy가 다른 경우 → Off-Policy Learning

Category 1	Monte-Carlo	Temporal Difference	
Category 2	On-Policy	On-Policy	Off-Policy
Learning Method	**On-Policy MC**	**SARSA**	**Q-Learning**
Behavior Policy	ε-Greedy	ε-Greedy	ε-Greedy
Target Policy	ε-Greedy	ε-Greedy	Greedy

Model-Free Reinforcement Learning

Model-Free Control (On-policy vs. Off-Policy)

❖ On-Policy, Off-Policy 방법론 간의 비교 (경험 재사용)

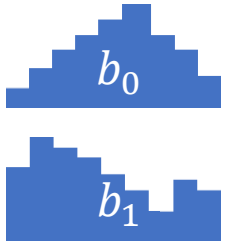
- On-Policy의 경우 현재 정책으로 쌓은 경험으로만 현재 정책을 개선할 수 있음
- Off-Policy의 경우 현재 정책 뿐만 아니라 과거에 쌓은 경험으로도 현재 정책을 개선할 수 있음
- 따라서 Off-Policy가 On-Policy보다 데이터 효율적인 학습이 가능함

On-Policy Learning

Iteration 1	Policy Evaluation	:	Experience(0) ~ b_0
	Policy Improvement	:	$\pi_0 \rightarrow \pi_1$ using Experience(0)
Iteration 2	Policy Evaluation	:	Experience(1) ~ b_1
	Policy Improvement	:	$\pi_1 \rightarrow \pi_2$ using Experience(1)

Off-Policy Learning

Iteration 1	Policy Evaluation	:	Experience(0) ~ b_0
	Policy Improvement	:	$\pi_0 \rightarrow \pi_1$ using Experience(0)
Iteration 2	Policy Evaluation	:	Experience(1) ~ b_1
	Policy Improvement	:	$\pi_1 \rightarrow \pi_2$ using Experience(0,1)

** 

개선 전, 후의 정책은 아주 조금의 업데이트를 했더라도 서로 다른 정책에 해당함

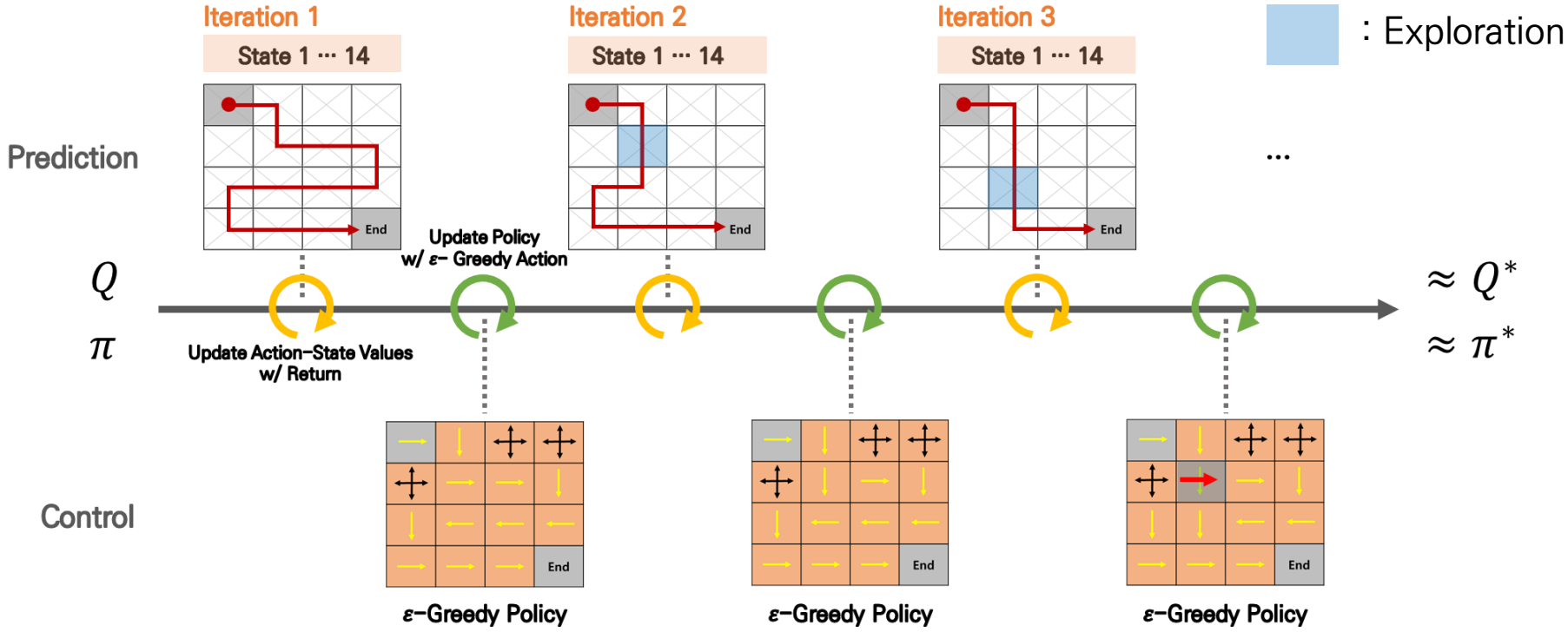
정책이 개선됨에 따라 샘플링 되는 경험의 분포도 달라짐

Model-Free Reinforcement Learning

Model-Free Control (On-Policy MC Control)

❖ On-Policy Monte-Carlo

- 정책 평가(Behavior Policy) : ε-Greedy (MC Prediction)
- 정책 개선(Target Policy) : ε-Greedy
- 상태-행동 가치 함수 및 리턴을 사용함 : $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha * \boldsymbol{G_t}$

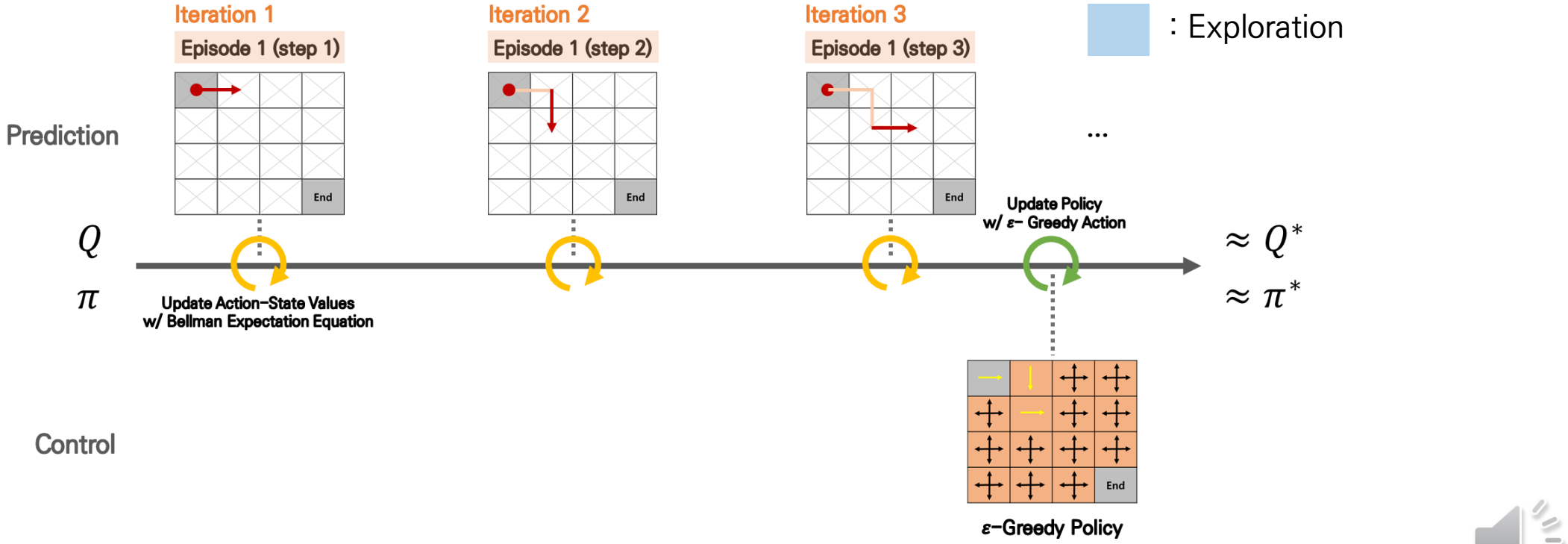


Model-Free Reinforcement Learning

Model-Free Control (On-Policy TD Control)

❖ SARSA

- 정책 평가(Behavior Policy) : ε-Greedy, TD Prediction
- 정책 개선(Target Policy) : ε-Greedy
- 상태-행동 가치 함수 및 벨만 기대 방정식을 사용함 : $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha * \left(\boldsymbol{r_{t+1} + \gamma Q(s_{t+1}, a_{t+1})} - Q(s_t, a_t)\right)$

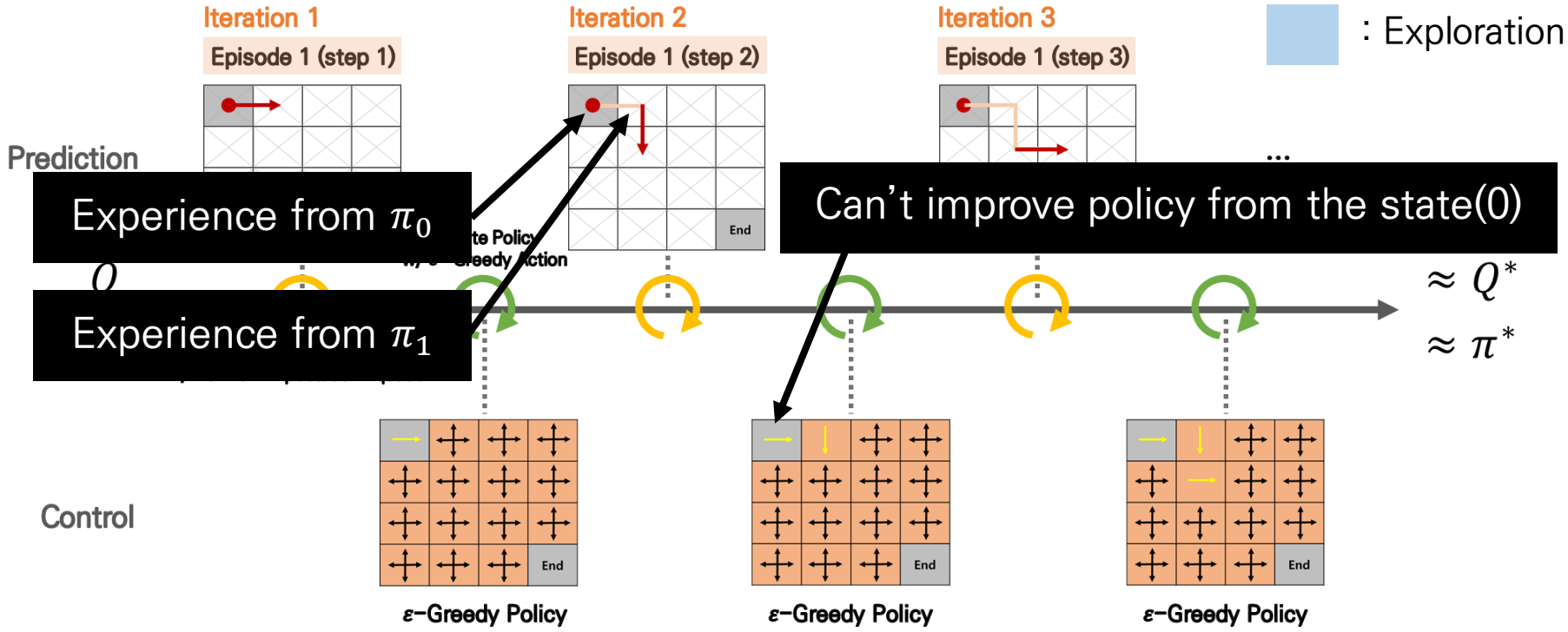


Model-Free Reinforcement Learning

Model-Free Control (On-Policy TD Control)

❖ SARSA

- 정책 평가(Behavior Policy) : ε-Greedy, TD Prediction
- 정책 개선(Target Policy) : ε-Greedy
- 상태-행동 가치 함수 및 벨만 기대 방정식을 사용함 : $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha * \left(\boldsymbol{r_{t+1} + \gamma Q(s_{t+1}, a_{t+1})} - Q(s_t, a_t)\right)$

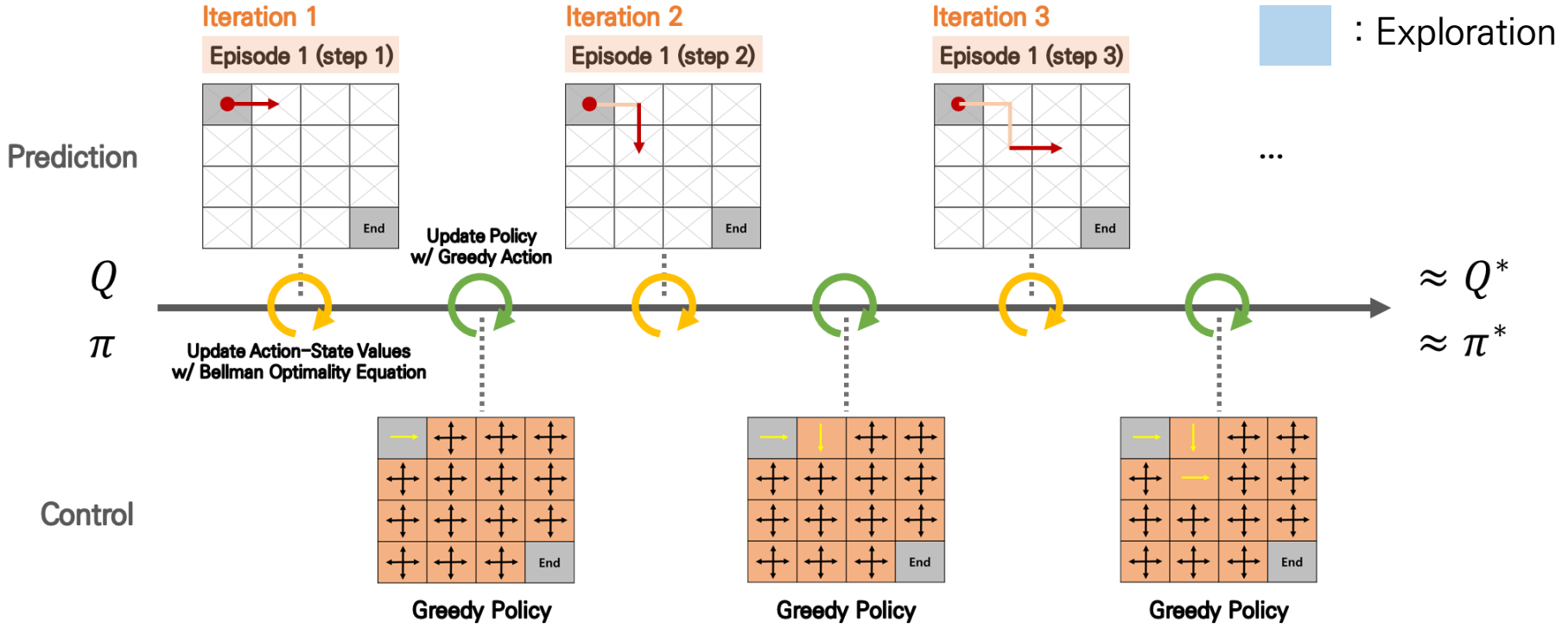


Model-Free Reinforcement Learning

Model-Free Control (Off-Policy TD Control)

❖ Q-Learning

- 정책 평가(Behavior Policy) : ε-Greedy, TD Prediction
- 정책 개선(Target Policy) : Greedy
- 상태-행동 가치 함수 및 벨만 최적 방정식을 사용함 : $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha * \left(r_{t+1} + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t) \right)$



Model-Free Reinforcement Learning

Model-Free Control (On-Policy TD Control)

벨만 기대 방정식	$q_\pi(\mathrm{s_t}, \mathrm{a_t}) = \mathbb{E}_\pi[r_{t+1} + \gamma q_\pi(s_{t+1}, a_{t+1})]$
벨만 최적 방정식	$q_*(\mathrm{s_t}, a_t) = \mathbb{E}\left[r_{t+1} + \gamma \max_{a'} q_*(s_{t+1}, a')\right]$

❖ SARSA가 서로 다른 정책을 사용하여 학습할 수 **없는** 이유

- SARSA는 행동-상태 가치를 계산할 때 벨만 기대 방정식을 사용함
- 벨만 기대 방정식은 주어진 정책 함수의 확률적인 요소에 따라서 행동을 취하고 이에 대한 리턴을 계산하도록 함

$$q_\pi(s,a) = \mathrm{r}_s^a + \gamma \sum_{s' \in S} \boldsymbol{P}_{ss'}^{\boldsymbol{a}} \sum_{a' \in A} \boldsymbol{\pi}(\boldsymbol{a'}|\boldsymbol{s'})\, q_\pi(s', a')$$

- 특정 정책의 확률이 고려된 리턴이기 때문에 향후 개선된 정책에게는 다른 확률분포에서 나온 데이터에 해당하여 정책 개선에 사용될 수 없음

Model-Free Reinforcement Learning

Model-Free Control (On-Policy TD Control)

벨만 기대 방정식	$q_\pi(s_t, a_t) = \mathbb{E}_\pi[r_{t+1} + \gamma q_\pi(s_{t+1}, a_{t+1})]$
벨만 최적 방정식	$q_*(s_t, a_t) = \mathbb{E}\left[r_{t+1} + \gamma \max_{a'} q_*(s_{t+1}, a')\right]$

❖ Q-Learning이 서로 다른 정책을 사용하여 학습할 수 **있는** 이유

- Q-Learning은 행동-상태 가치를 계산할 때 벨만 최적 방정식을 사용함
- 벨만 최적 방정식은 특정된 정책 함수가 없으며 정해진 환경에 따라서 최적 정책이 부여되어 이에 대한 리턴을 계산하도록 함

$$q_*(s,a) = \mathbf{r}_{\mathbf{s}}^{\boldsymbol{a}} + \gamma \sum_{s' \in S} \boldsymbol{P}_{\boldsymbol{ss'}}^{\boldsymbol{a}} \max_{a'} q_*(s', a')$$ ← 별도의 π가 명시되지 않음

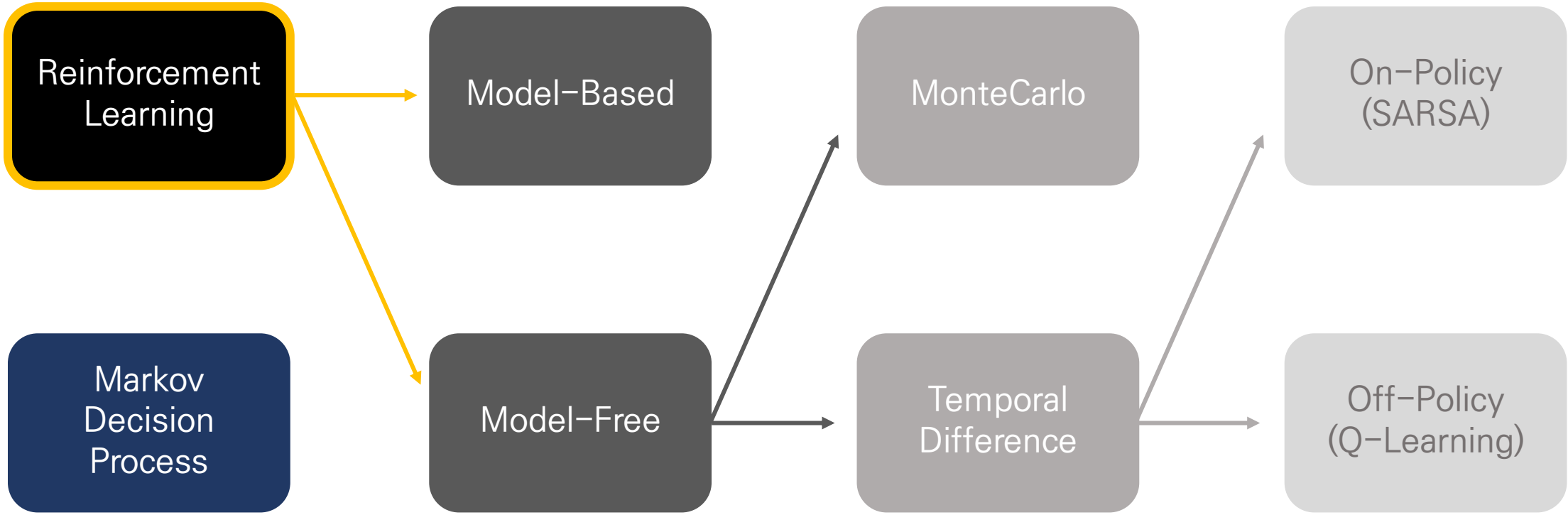
- 즉, 리턴을 계산하는 구조 상 특정 정책에 제한 되지 않으므로 Behavior Policy와 Target Policy가 달라도 학습이 가능함

Summary

Deep Reinforcement Learning

❖ Reinforcement Learning이란?

- 강화학습은 순차적인 의사결정 문제에서 누적 보상을 최대화 하기 위해 시행착오를 거쳐서 상황에 따른 행동 정책을 학습



Citation

노승은, 『바닥부터 배우는 강화학습』, 영진닷컴(2020)

Sutton, Richard S., and Andrew G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

https://www.youtube.com/playlist?list=PLqYmG7hTraZBiG_XpjnPrSNw-1XQaM_gB
(Introduction to Reinforcement learning with David Silver, DeepMind)

https://www.davidsilver.uk/wp-content/uploads/2020/03/DP.pdf
https://www.davidsilver.uk/wp-content/uploads/2020/03/control.pdf
https://www.davidsilver.uk/wp-content/uploads/2020/03/MC-TD.pdf