Chapter. 02

동적 계획법

더 효율적인 DP:
비동기적 동적계획법

FAST CAMPUS
ONLINE
강화학습 A-Z I

강사. 박준영

l 지난 이야기…

- 동적 계획법 (Dynamic programming: DP)
  - 최적 하위구조 / 반복 하위구조
- 정책 반복 알고리즘
  - 정책 평가 알고리즘
  - 정책 개선 알고리즘
    - 정책 개선 정리
- 정책 평가 알고리즘의 수렴 증명

I 굳이 정책 평가가 수렴할 때 까지 반복해야 할까?

입력: 임의의 정책 π_0

출력: 최적 정책 π^*

반복: $(k = 0, \ldots,)$

- $\pi_{k+1} \leftarrow$ **정책 반복** (π_k)
- 만약 $\pi_{k+1} \sim \pi_k$, 반복문 탈출

최적 정책 $\pi^* \leftarrow \pi_{k+1}$

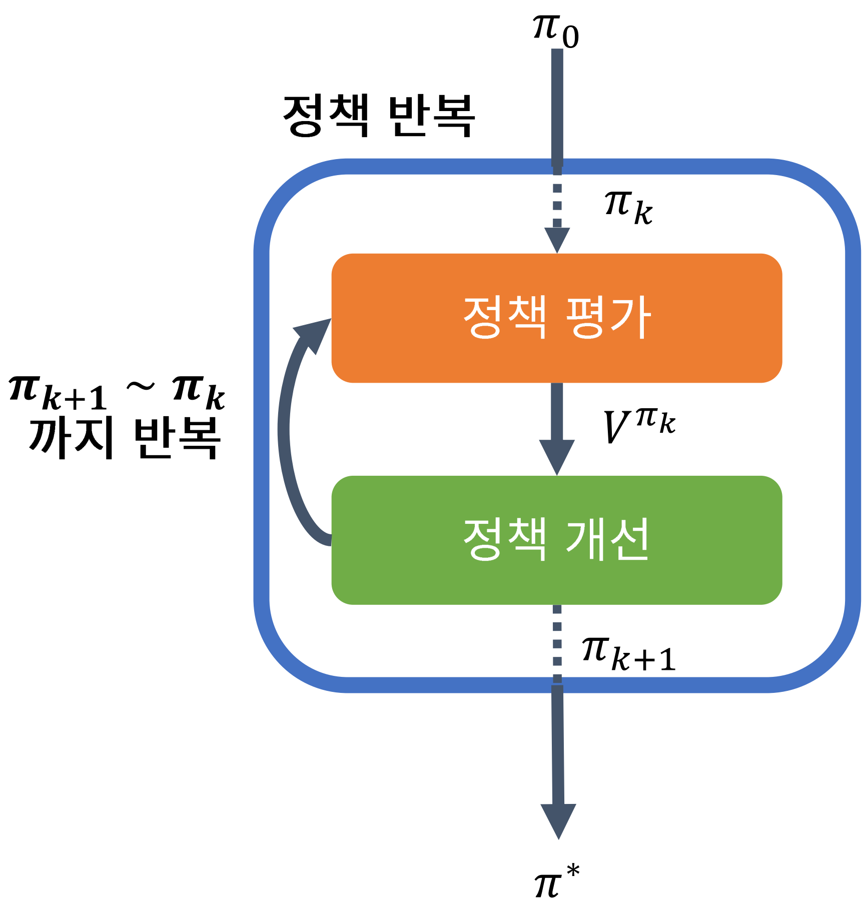
정책 반복 (Policy iteration)

입력: 임의의 정책 정책 π

출력: 개선된 정책 π'

(정책에 대한 가치함수 V^π 가 수렴할 때까지 반복!)

1. 정책 평가 (PE) 를 적용해 $V^\pi(s)$ 계산
2. 정책 개선 (PI) 를 적용해 π' 계산



| 가치 반복 (Value Iteration: VI)

가치 반복 (Value iteration) 꼭 0 이 아니라 어떤 값으로 시작해도 하나의 값으로 알고리즘의 결과는 수렴함.

입력: 임의의 가치 함수 $V_0(s) \leftarrow 0$ 모든 $s \in \mathcal{S}$

출력: 최적 가치 함수 $V^*(s)$

반복: $(k = 0, \dots,)$:

<Bellman optimal backup>

- 모든 상태 s 에 대해서, $V_{k+1}(s) = \max_{a \in \mathcal{A}} (R_s^a + \gamma \sum_{s' \in \mathcal{S}} P_{ss'}^a V_k(s'))$ 적용
- 모든 상태 s 에 대해서, $V_{k+1}(s) \sim V_k(s)$ 이면, 반복문 탈출

반환: $V^*(s) \leftarrow V_{k+1}$

$$\text{행렬표현: } V_{k+1} = \max_{a \in \mathcal{A}} (R^a + \gamma P^a V_{k+1})$$

Q) 이 알고리즘이 과연 유일한 하나의 값으로 수렴하나요?
A) 네, 수렴합니다. 저번 강의의 마지막 슬라이드와 동일한 원리입니다.

Bellman optimality backup 오퍼레이터

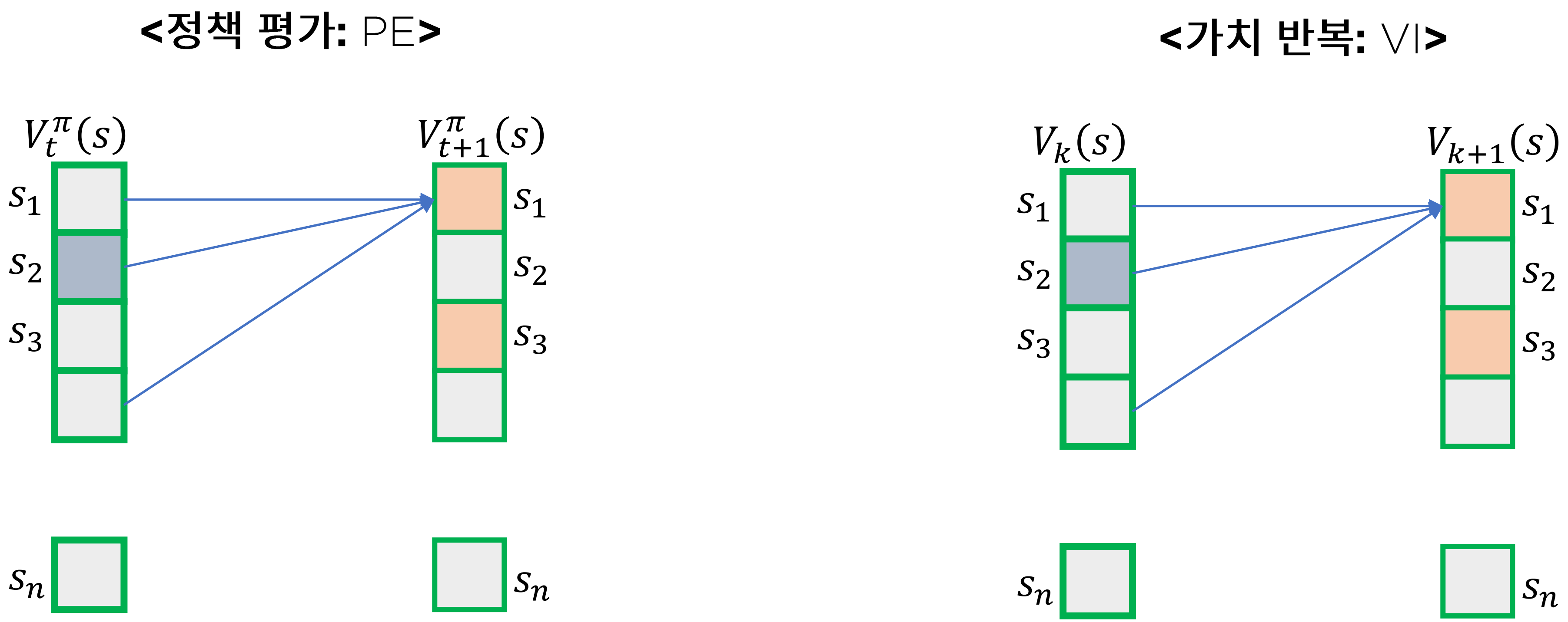
$$T^*(V) \overset{\text{def}}{=} \max_{a \in \mathcal{A}} (R^a + \gamma P^a V)$$

$T^\pi(V)$ 는 Bellman expectation backup 오퍼레이터 (연산자) 라고 불림.

$T^*(V)$ 은 $\gamma -$ 수축 사상이다. $\|T^*(u) - T^*(v)\|_\infty \leq \gamma \|u - v\|_\infty$

“**바나흐 고정점 정리**”에 의해 VI는 유일한 해 V^* 으로 수렴한다.

I 동기적 DP 알고리즘



- 모든 가치 Table이 동시에 업데이트 됨
- 만약의 $|S|$ (전체 s 의 개수) 가 커진다면, 계산하기 난감해짐. (바둑의 경우 $|S| = 3^{(19\times19)}$)

→ 비동기적 업데이트 알고리즘이 필요

I 비동기적 DP 알고리즘 (Asynchronous DP)

여러가지 비동기적 DP 알고리즘이 존재하지만, 세 가지 알고리즘을 설명:

비동기적 DP 알고리즘을 사용할 때 일반적으로 각 s 에 대해 임의의 순차적으로 진행함.

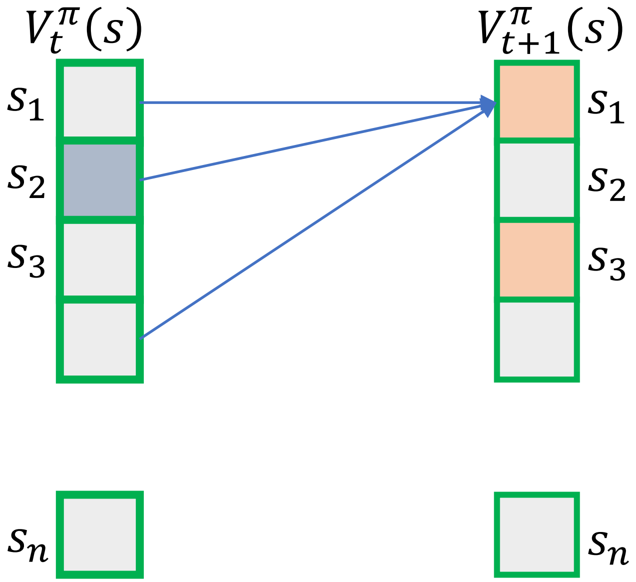
- In-place DP
- Prioritized sweeping
- Real-time DP

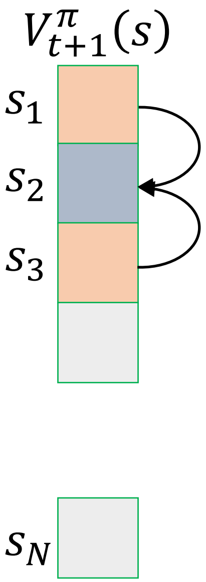
* 세 알고리즘은 PE/VI의 수렴성을 해치지 않는 것으로 증명됨.

** 세 가지 알고리즘은 강의 후반부에 다룰 <심층 강화학습 알고리즘> 의 빠른 수렴을 유도하기 위해서도 사용됨.

In-place DP

“In place” 연산





현재 알고 있는 가장 새로운 값 $V(s)$ 을 활용해 $V(s)$ 들을 업데이트한다.

+ 메모리 사용량 절감
+ 일반적으로 수렴속도가 더 빠름
+ 구현하기 쉬움

Prioritized sweeping

Bellman error의 크기가 가장 큰 s 부터 업데이트 한다

$$\textbf{Bellman error}(s) = \left|\max_{a\in\mathcal{A}}\left(R_s^a + \gamma \sum_{s'\in\mathcal{S}} P_{ss'}^a V(s')\right) - V(s)\right|$$

+ PE / VI 의 수렴속도가 빨라짐. ($T^\pi(V)$ 과 $T^*(V)$, $\|\cdot\|_\infty$ 에 대한 축약사상 + 바나흐 고정점 정리)

Real-time DP

강화학습의 에이전트가 현재 겪은 상황만 업데이트 하자!

현재 상태가 s_t 이라면,

$$V(s_t) = \max_{a \in \mathcal{A}} \left(R_{s_t}^{a} + \gamma \sum_{s' \in \mathcal{S}} P_{s_t s'}^{a} V(s') \right)$$

I 동적 계획법 한눈에 보기

정책 평가

반복 ($t = 1, \ldots$)

모든 상태 s에 대하여:

$$V_{t+1}^{\pi}(s) \leftarrow \sum_{a \in \mathcal{A}} \pi(a|s) \left(R_s^a + \gamma \sum_{s' \in \mathcal{S}} P_{ss'}^a V_t^{\pi}(s') \right)$$

정책 반복

정책 개선

모든 상태 s에 대하여:

$$\pi'(s) = \underset{a \in \mathcal{A}(s)}{\operatorname{argmax}}\, Q^{\pi}(s, a)$$

가치 반복

모든 상태 s에 대하여:

$$V_{t+1}(s) \leftarrow \max_{a \in \mathcal{A}} \left(R_s^a + \gamma \sum_{s' \in \mathcal{S}} P_{ss'}^a V_k(s') \right)$$

비동기적 가치 반복

하나의 상태 s 에 대해서:

$$V(s) \leftarrow \max_{a \in \mathcal{A}} \left(R_s^a + \gamma \sum_{s' \in \mathcal{S}} P_{ss'}^a V_k(s') \right)$$

I 마무리!

- 가치 반복 (Value Iteration: VI)
- 비동기적 동적계획법
  - In-place operation
  - Prioritized sweeping
  - Realtime DP

- 과연 우리는 진짜로 큰 **문제**를 주어진 시간 내에 DP 로 풀 수 있을 것인가?
- 현실에서 R, P 로 표현되는 문제의 정보를 언제나 알 수 있을까?

I In-place DP 알고리즘

In-place 가치 반복 (full-sweeping)

입력: 임의의 가치 함수 $V(s) \leftarrow 0$ 모든 $s \in \mathcal{S}$

출력: 최적 가치 함수 $V^*(s)$

반복: $(k = 0, \dots,)$:

<Bellman optimal backup>

- 모든 상태 s 에 대해서, $V(s) = \max_{a \in \mathcal{A}}(R_s^a + \gamma \sum_{s' \in \mathcal{S}} P_{ss'}^a V(s'))$ 적용
- 모든 상태 s 에 대해서, $V(s)$ 수렴하면, 반복문 탈출

반환: $V^*(s) \leftarrow V(s)$