국민대학교 KPSC & AIM 스터디 – 강화학습을 이용한 체스 AI 만들기

Introduction to RL, Week 2

Chris Ohk
utilForever@gmail.com

벨만 기대 방정식

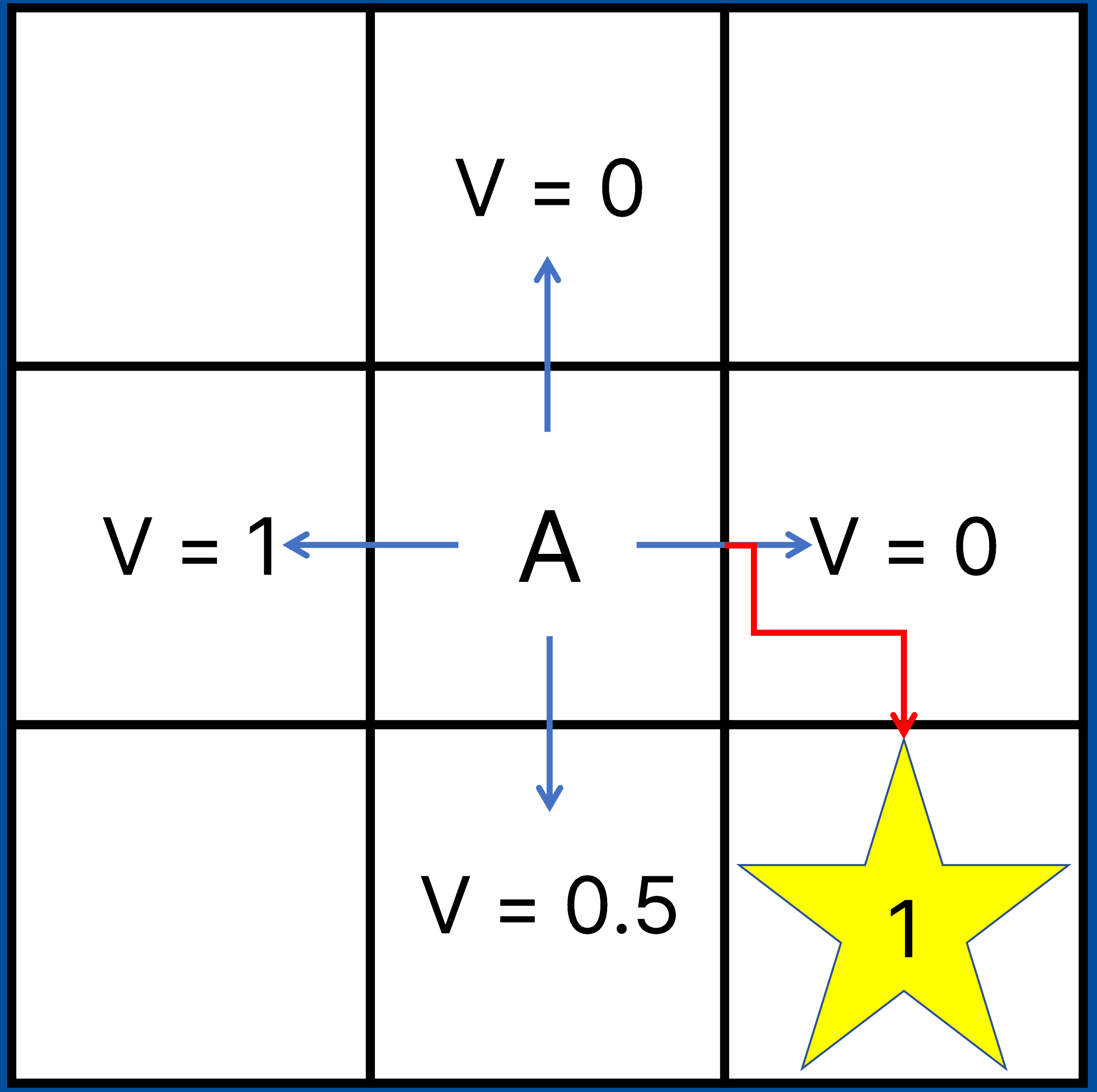
$$v_\pi(s) = E_\pi[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s]$$

- 위 함수는 현재 가치함수 값을 갱신한다.
  하지만 갱신하려면 기댓값을 계산해야 하는데 어떻게 계산할까?
  - 기댓값에는 어떤 행동을 할 확률(정책 $\pi(a \mid s)$)과
    그 행동을 했을 때 어떤 상태로 가게 되는 확률(상태 변환 확률 $P_{ss'}^a$)이 포함되어 있다.
  - 따라서 정책과 상태 변환 확률을 포함해서 계산하면 된다.

$$v_\pi(s) = \sum_{a \in A} \pi(a \mid s) \left(R_{t+1} + \gamma \sum_{s' \in S} P_{ss'}^a \cdot v_\pi(s') \right)$$

벨만 기대 방정식

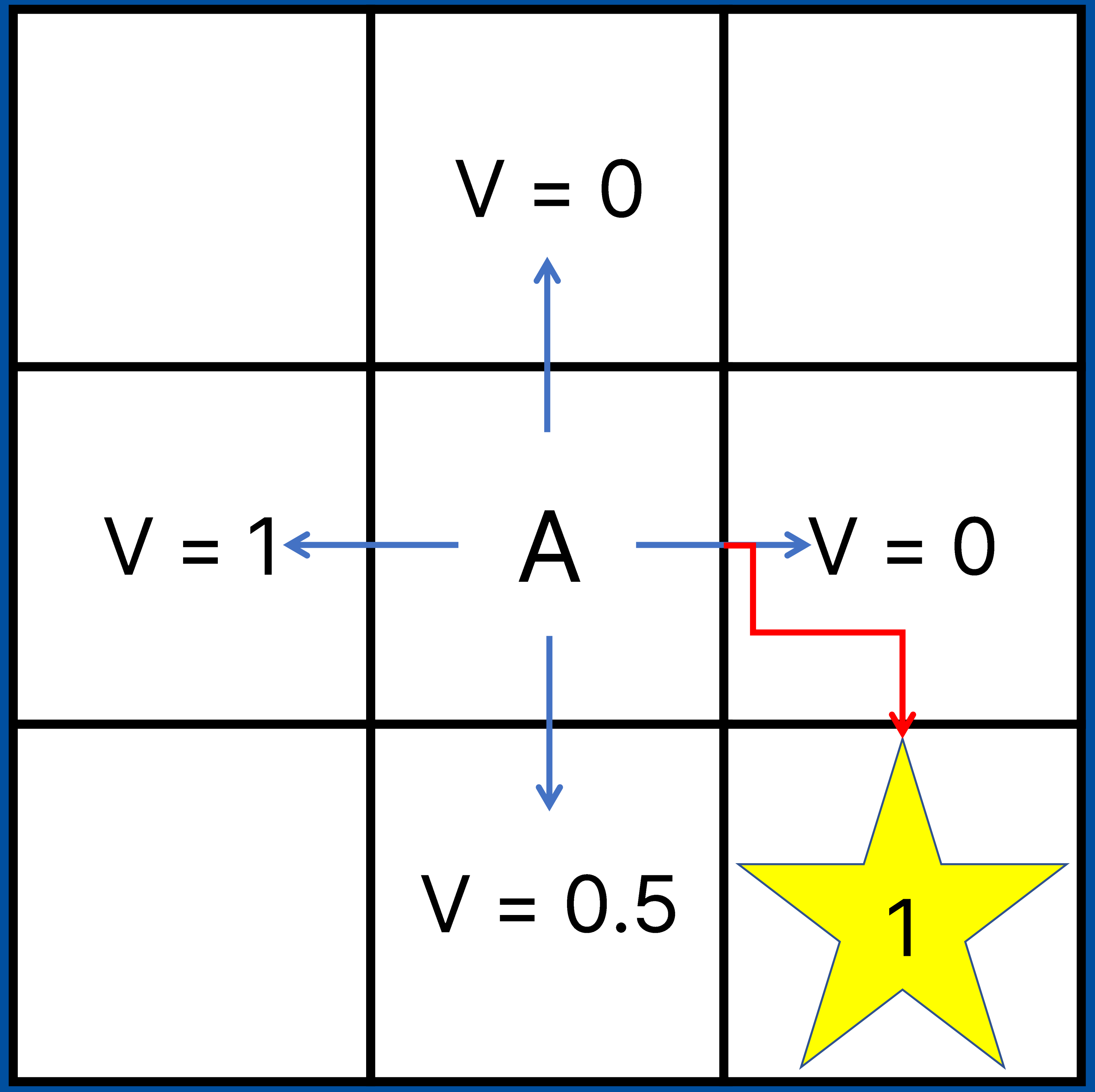
- 상태 변환 확률을 모든 s와 a에 대해 1이라고 가정하자.
  그리고 다음 예제를 한 번 생각해 보자.
  - 행동은 “상, 하, 좌, 우” 4가지
  - 초기 정책 : 무작위로 각 행동을 선택할 확률 25%
  - 현재 에이전트 상태에 저장된 가치함수는 0
    - 왼쪽 상태의 가치함수는 1, 밑쪽 상태의 가치함수는 0.5
      위쪽 상태의 가치함수는 0, 오른쪽 상태의 가치함수는 0
  - 감가율은 0.9
  - 오른쪽으로 행동을 취할 경우
    노란색 별로 표현된 1의 보상을 받음



벨만 기대 방정식

$$v_\pi(s) = \sum_{a \in A} \pi(a \mid s)\big(R_{t+1} + \gamma v_\pi(s')\big)$$

- 행동 = 상 : $0.25 \times (0 + 0.9 \times 0) = 0$
- 행동 = 하 : $0.25 \times (0 + 0.9 \times 0.5) = 0.1125$
- 행동 = 좌 : $0.25 \times (0 + 0.9 \times 1) = 0.225$
- 행동 = 우 : $0.25 \times (1 + 0.9 \times 0) = 0.25$
- 기댓값 = $0 + 0.1125 + 0.225 + 0.25 = 0.5875$



벨만 최적 방정식

$$v_\pi(s) = E_\pi[R_{t+1} + \gamma v_\pi(S_{t+1}) | S_t = s]$$

- 벨만 기대 방정식을 통해 계속 계산을 진행하다 보면
  언젠가 식의 왼쪽 항과 오른쪽 항이 동일해지는 순간이 온다.
  → $v_\pi(s)$ 값이 수렴 → 현재 정책 π에 대한 **참 가치함수**를 구한 것

벨만 최적 방정식

- 하지만 참 가치함수와 최적 가치함수는 다르다.
  - 참 가치함수는 "어떤 정책"을 따라서 움직였을 경우에 받게 되는 보상에 대한 참값
  - 가치함수란 "현재로부터 미래까지 받을 보상의 총합"인데 이 값이 얼마가 될 지에 대한 값
  - 하지만 최적의 가치함수는 수많은 정책 중에서 **가장 높은 보상을 주는 가치함수**다.

$$v_{k+1}(s) = \pi(a \mid s)\left(R_s^a + \gamma v_k(s')\right)$$

  - $v_{k+1}(s)$: 현재 정책에 따라 $k+1$번째 계산한 가치함수 (그 중에서 상태 s의 가치함수)
  - $k+1$번째의 가치함수는 k번째 가치함수 중에서 주변 상태들 s'을 이용해 구한다. 그리고 이 계산은 모든 상태에 대해 동시에 진행한다.

벨만 최적 방정식

- 최적 정책은 각 상태 s에서의 큐함수 중에서 가장 큰 큐함수 값을 갖는 행동을 하는 것
  → 선택 상황에서 판단 기준은 큐함수이며,
  최적 정책은 언제나 이 큐함수 중에서 가장 높은 행동을 하나 하는 것

$$\pi_*(s,a) = \begin{cases} 1, & if\ a = \underset{a \in A}{\operatorname{argmax}}\, q_*(s,a) \\ 0, & otherwise \end{cases}$$

벨만 최적 방정식

- 최적의 가치함수 = **최적의 큐함수 중에서 최대를 선택하는 것**

$$v_*(s) = \max_a [q_*(s,a) \mid S_t = s, A_t = a]$$

- 큐함수를 가치함수로 고쳐보자.

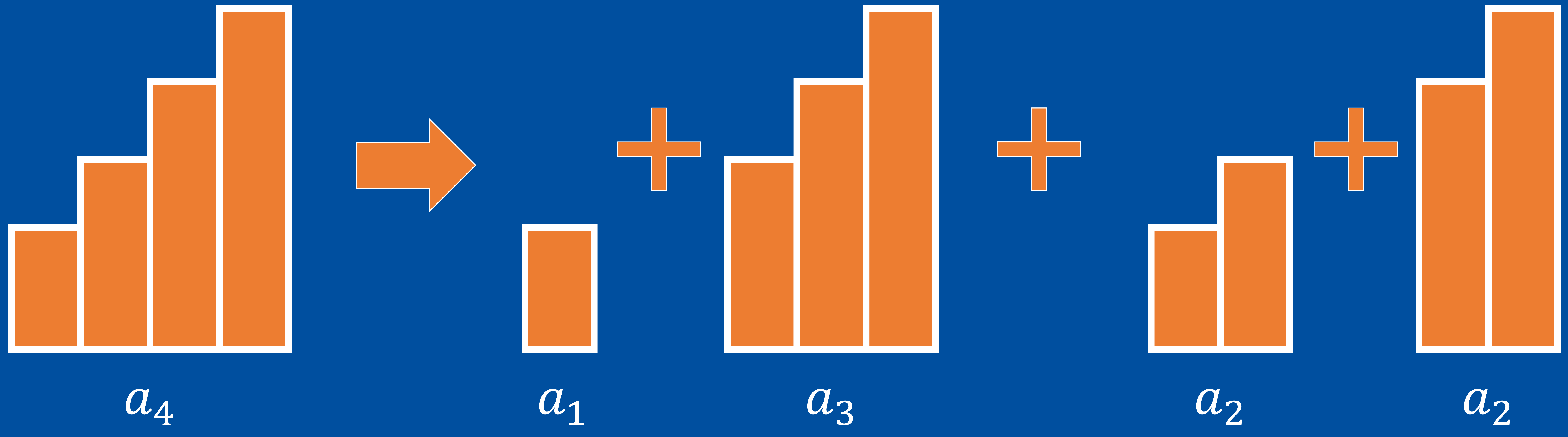
$$v_*(s) = \max_a E[R_{t+1} + \gamma v_*(S_{t+1}) \mid S_t = s, A_t = a]$$

- 이를 **벨만 최적 방정식(Bellman Optimality Equation)**이라고 한다.

$$q_*(s,a) = E\left[R_{t+1} + \gamma \max_{a'} q_*(s',a') \mid S_t = s, A_t = a\right]$$

다이내믹 프로그래밍

- 한번에 풀기 어려운 문제를 여러 개의 작은 문제로 나눠 푸는 것
  - 작은 문제들 사이에서 공유하는 계산 결과들을 재사용해 총 계산량을 줄일 수 있음
  - 예) 총 4칸인 계단을 한 번에 1칸, 2칸씩 오르는 경우의 수는?



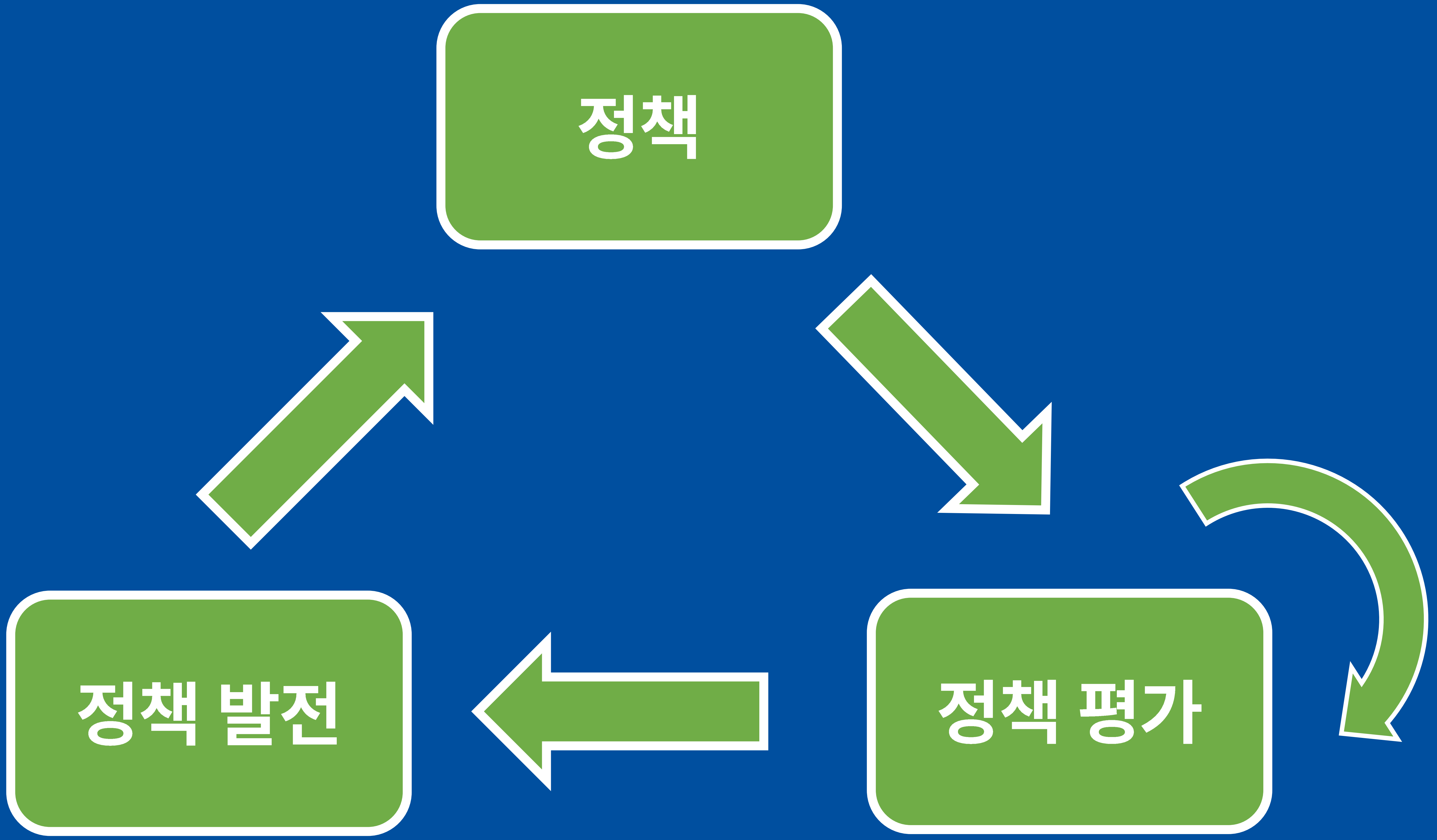
다이내믹 프로그래밍

$$v_\pi(s) = E_\pi[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots \mid S_t = s]$$

- 가치함수($v_\pi(s)$)를 기준으로 정책이 얼마나 좋은 지 평가한다.
- 문제는 정책을 평가하려면 $v_\pi(s)$가 필요한데 이게 뭔 지 모른다.

  → 다이내믹 프로그래밍으로 해결

정책 이터레이션



정책 이터레이션

정책 평가

- 가치함수 : (지금 받은 보상 + 미래에 받을 보상)의 기댓값
- 가치함수의 정의에 맞게 점화식으로 만들면

$$v_\pi(s) = E_\pi[R_{t+1} + \gamma v_\pi(S_{t+1}) | S_t = s]$$

정책 평가

- 기댓값 : 사건이 일어날 확률과 사건으로 얻을 이득의 곱의 총합

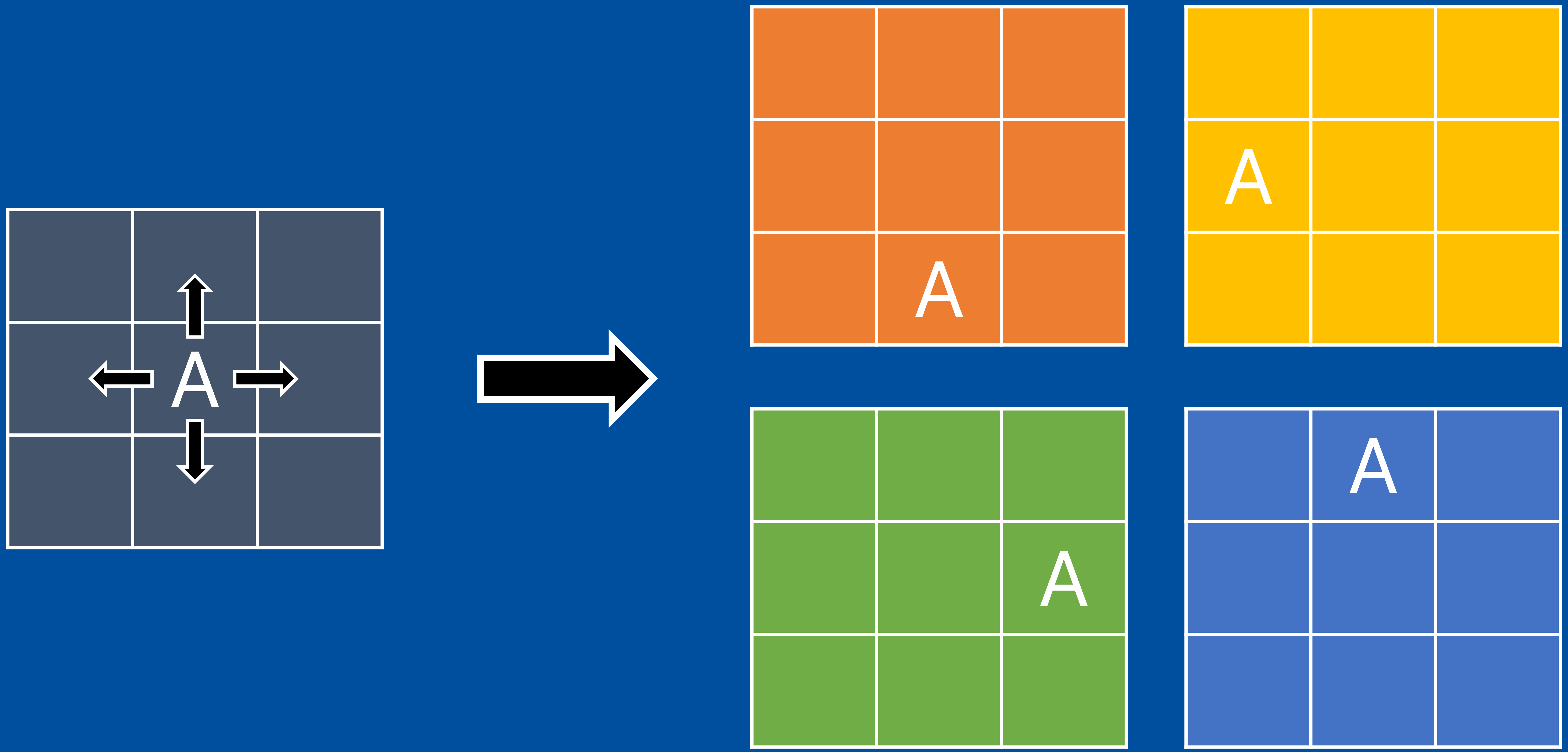
$$v_\pi(s) = E_\pi[R_{t+1} + \gamma v_\pi(S_{t+1}) | S_t = s]$$

$$v_\pi(s) = \sum_{a \in A} \underbrace{\pi(a|s)}_{\text{확률}} \underbrace{\big(R_{t+1} + \gamma v_\pi(s')\big)}_{\text{이득}}$$

$$v_{k+1}(s) = \sum_{a \in A} \pi(a|s)\big(R_s^a + \gamma v_k(s')\big)$$

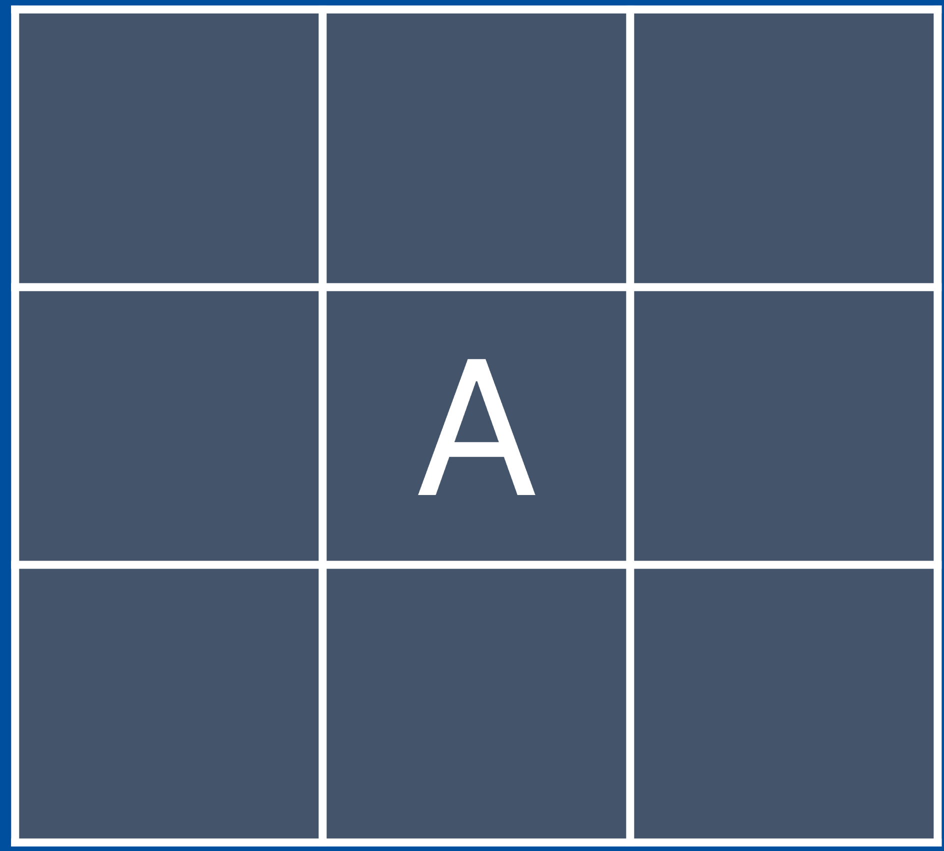
정책 평가

1. 현재 상태 s에서 가능한 행동 A를 통해 다음 상태 s'들을 구한다.

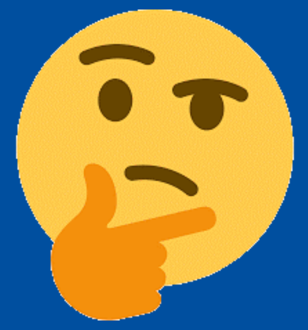


정책 평가

2. 현재 k번째 가치함수(v_k)로,
   다음 상태 s'에 대한 가치($v_k(s')$)를 구한다.
3. 다음 상태에 대한 가치 $v_k(s')$에
   감가율(γ)을 곱하고 그 상태로 가는 행동에
   대한 보상(R_s^a)을 더한다.
4. 위에서 구한 이득에 s에서 s'가 되도록
   행동할 확률, 즉 정책을 곱한다.

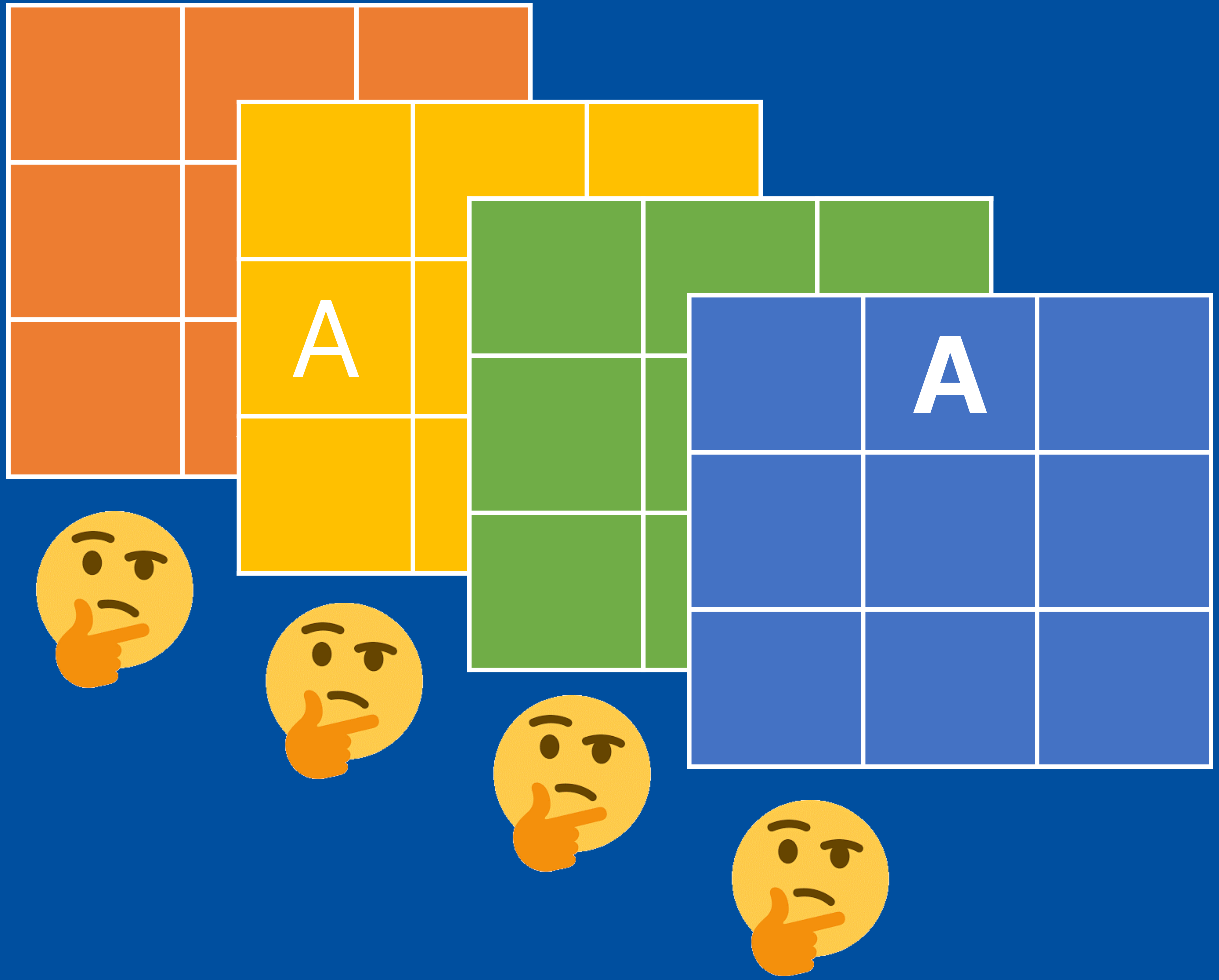


$$\pi(a|s)(R_s^a + \gamma v_k(s'))$$



정책 평가

5. 현재 k번째 가치함수(v_k)로,
   2~4번 과정을 아까 구했던 모든 행동들에 대해 반복하고 더한다.



$$\sum_{a \in A} \pi(a|s)(R_s^a + \gamma v_k(s'))$$

정책 평가

6. 위 과정을 통해 구한 값을 $k+1$번째 가치함수 행렬에 저장한다.
7. 1~6 과정을 모든 $s \in S$에 대해 반복한다.
8. 이 과정을 무한히 반복하면 실제 $v_\pi(s)$에 수렴한다.

$$v_{k+1}(s) = \sum_{a \in A} \pi(a|s)\left(R_s^a + \gamma v_k(s')\right)$$

정책 발전

- 앞에서 진행했던 정책 평가를 이용해 정책을 발전시킨다.
- 최초의 정책은 무작위 정책이었다.
- 정책 평가를 통해 무작위 정책들에 대한 가치를 알았으므로,
  큐함수를 이용해 어떤 행동이 좋은 지 알 수 있다.

$$q_\pi(s,a) = E_\pi[R_{t+1} + \underbrace{\gamma v_\pi(S_{t+1})}_{\text{가치}} | S_t = s, A_t = a]$$

정책 발전

$$q_\pi(s,a) = E_\pi[R_{t+1} + \gamma v_\pi(S_{t+1}) | S_t = s, A_t = a]$$

- '오른쪽으로 이동했는데 일정 확률로 움직이지 않는다.'와 같은 무작위성이 없다면, 위 큐함수 정의에서 확률이 1이라고 볼 수 있다.
- 그러므로, 큐함수 정의를 계산 가능한 아래 형태로 바꿀 수 있다.

$$q_\pi(s,a) = R_s^a + \gamma v_\pi(s')$$

정책 발전

- 현재 상태 s에서 선택 가능한 행동 a들의 $q_\pi(s,a)$를 비교하고, 가장 큰 행동을 선택하도록 새로운 정책을 구하면 된다.

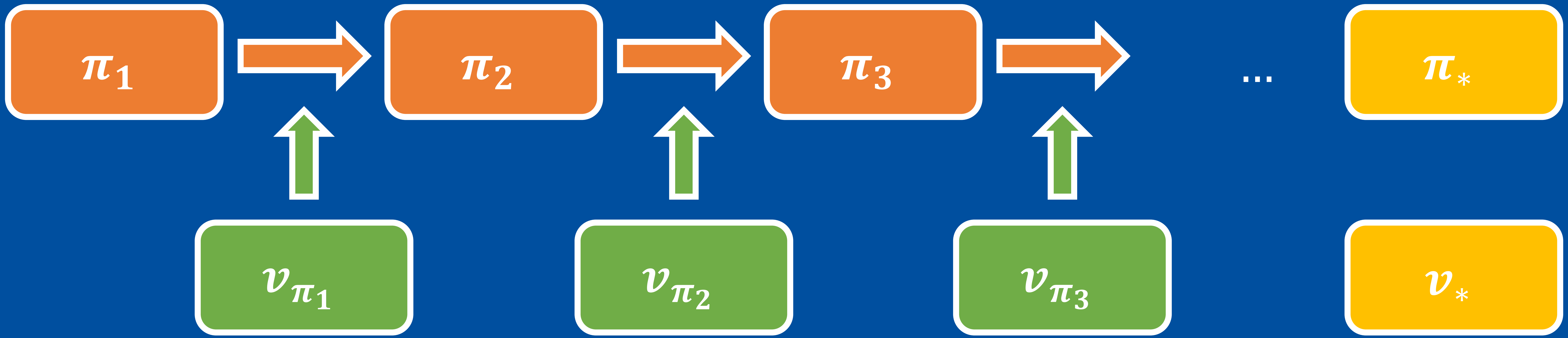
$$\pi'(s) = \operatorname*{argmax}_{a \in A} q_\pi(s,a)$$

정책 발전

- 이렇게 근시안적으로 현재에 가장 최적인 해를 구하는 것을 **탐욕(Greedy) 알고리즘**이라 한다. 그래서 큐함수의 값이 지금 가장 높은 행동을 선택하는 방법을 **탐욕 정책 발전**이라고 부른다.
  - 탐욕 정책 발전을 사용하면 업데이트된 가치함수가 이전보다 무조건 좋거나 같게 되므로, 언젠가는 가치함수가 가장 큰 최적 정책에 수렴한다.

정책 이터레이션?

- 정책 이터레이션에서는 정책과 가치함수가 명확히 분리된 상태로 발전한다.



정책 이터레이션?

- 정책이 독립적이므로, 결정적이지 않은 정책이 가능하다.
- 즉 정책이 확률적으로 여러 행동이 가능한 상태이고,
  이를 고려해서 가치함수를 계산하려면 기댓값을 구할 필요가 있다.

가치 이터레이션

- 하지만 현재 정책이 최적이라고 가정하고, 결정적 정책을 적용한다면?
- 이는 틀린 가정이지만, 이 방법으로 반복적으로 가치함수를 발전시켜 언젠가 최적 정책을 구할 수 있기 때문에 이는 문제가 되지 않는다.

가치 이터레이션

- 정책 이터레이션에서는 가치함수의 업데이트, 정책의 발전을 모두 다뤘다. 하지만 가치 이터레이션에서는 가치함수의 업데이트만을 다룬다.
- 그 이유는 가치 이터레이션에서는 가치함수 안에 정책이 내재적으로 포함되어 있어, 가치함수의 업데이트가 정책의 발전을 동반하기 때문이다.

가치 이터레이션

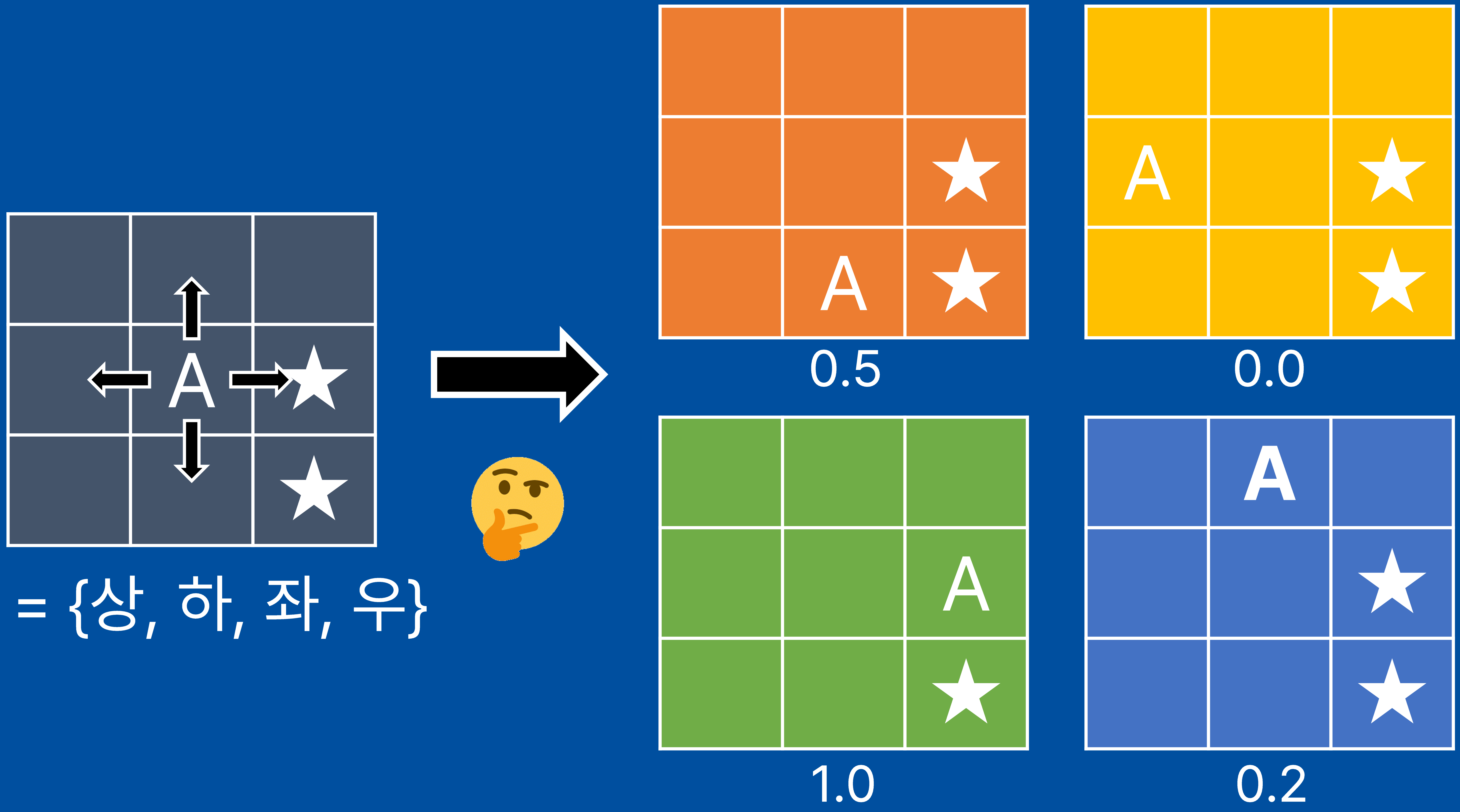
- 가치 이터레이션은 벨만 최적 방정식을 이용한다.

$$v_*(s) = \max_a E[R_{t+1} + \gamma v_*(S_{t+1}) | S_t = s, A_t = a]$$

- 가치 이터레이션에서는 최적 정책이라고 가정했기 때문에 정책 발전이 필요 없고, 가치함수를 업데이트 할 때 정책을 고려할 필요가 없다.
- 그저 현재 상태에서 얻을 수 있는 이득의 값($R_{t+1} + \gamma v_k(S_{t+1})$) 중 최고의 값으로 업데이트 하면 된다. 이를 **가치 이터레이션**이라고 한다.

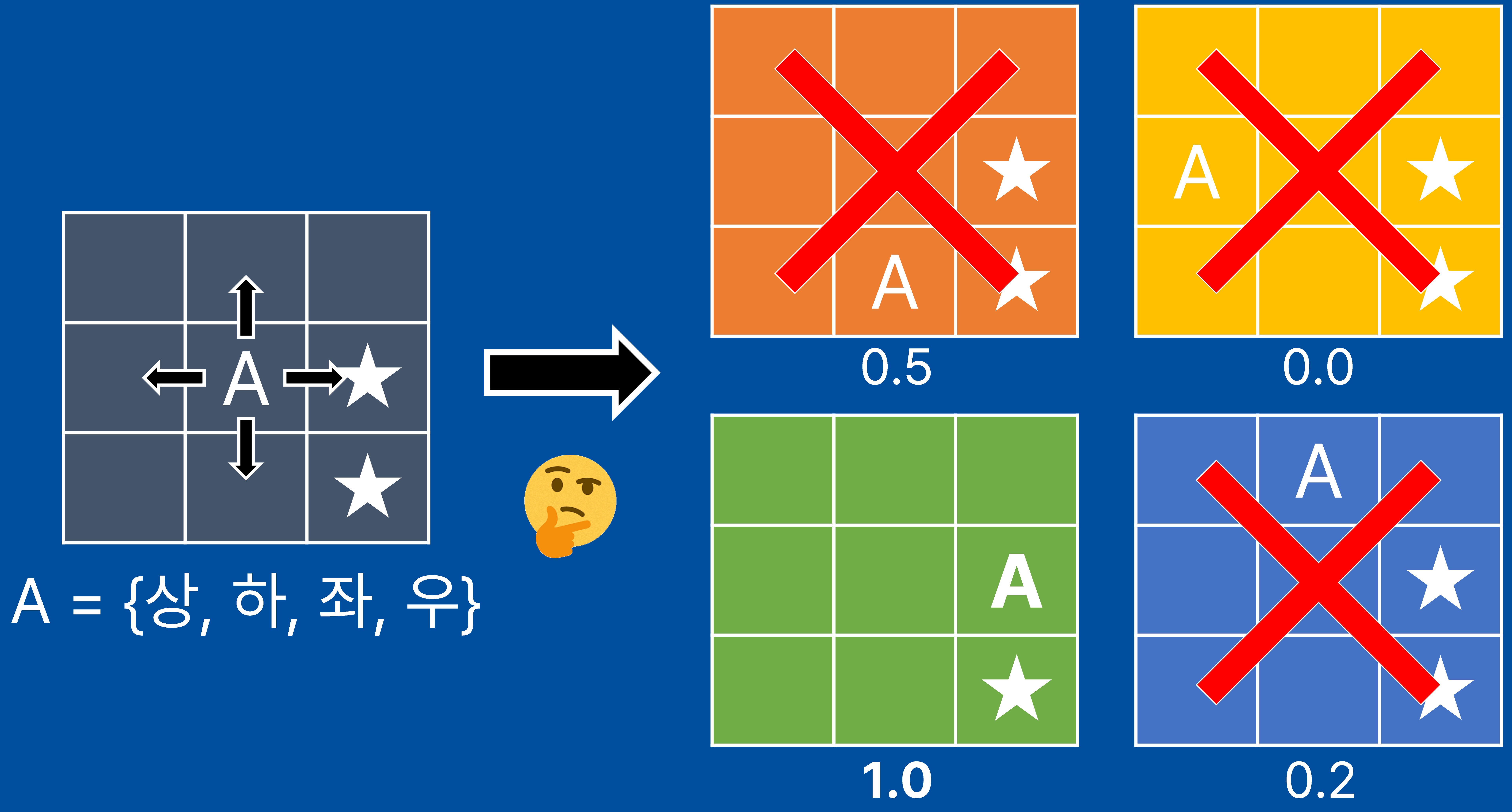
가치 이터레이션

1. 제일 높은 이득을 얻는 행동을 고른다.



가치 이터레이션

1. 제일 높은 이득을 얻는 행동을 고른다.



가치 이터레이션

2. 이번에도 무작위성이 없다 가정하면 확률이 1이다.
   그러므로 가치함수를 다음과 같이 업데이트하면 된다.

$$v_{k+1}(s) = \max_a (R_{t+1} + \gamma v_k(s'))$$

감사합니다.

utilForever@gmail.com

https://github.com/utilForever

X, Instagram: @utilForever