파이썬과 케라스로 배우는 강화학습
2장, 3장 -강화학습 기초

2018. 01. 10

권도형

LINK-lab

강화학습은 **순차적으로 행동을 계속 결정해야 하는 문제**를 푸는 것이다.
이 문제를 풀기 위해, 수학적인 모델링이 필요하다.
그것이 MDP(Markov Decision Process)다.

강화학습의 목표는 보상을 최대화할 수 있는 최적의 정책을 찾는 것이다.

MDP(Markov Decision Process)의 구성은 다음과 같다.

상태(state), 행동(action), 감가율(discount factor), 정책(policy) 은 Agent가 결정하고

보상(reward), 상태 변환 확률(state transition probability) 은 Environment가 Agent에게 알려준다.

상태(state)와 행동(action)은 집합이며, 집합 기호로 표현된다.

$$S_t = s$$
$$A_t = a$$

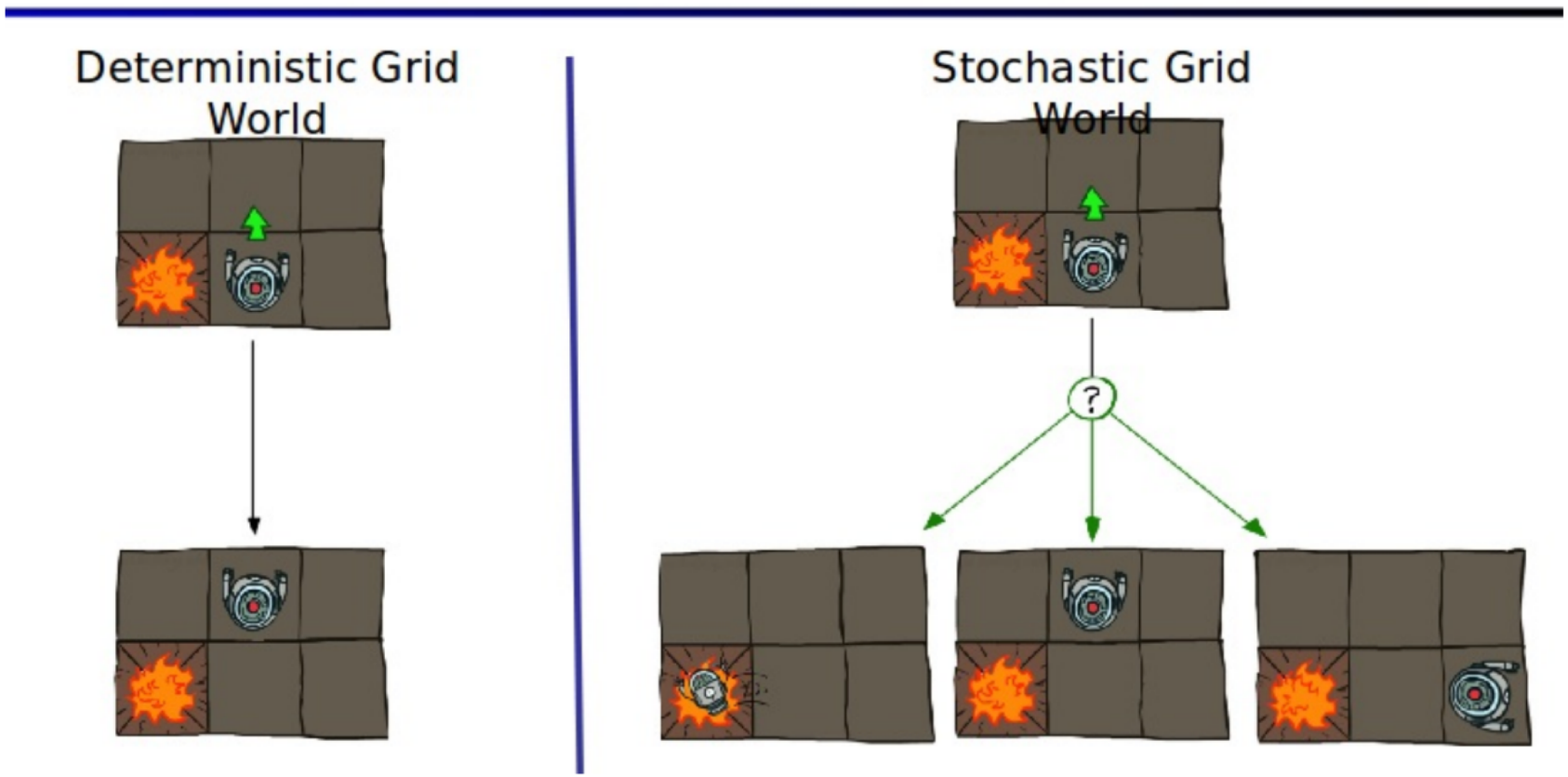
그리고 **정책(policy)**은 확률이며, "에이전트가 어떤 State s에서 Action a를 할 확률"로 표현할 수 있다.

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

상태 변환 확률(State-transition probability)은 확률이며, "시간 t일 때 state s에서 action a를 했을 때, 그 다음 state가 s'이 될 확률을 나타내는 함수"로 표현할 수 있다. (non-determinstic을 고려하기 위해)

$$P_{ss'}^{a} = P[S_{t+1} = s'|S_t = s, A_t = a]$$

robot이 움직인다고 생각해보자. robot이 왼쪽으로 움직이면 위치가 변할텐데, 어떠한 외부요인에 의해 (예를 들면, 바람이 분다던지 하는 등의 요인에 의해) robot이 왼쪽으로 가려했지만, 오른쪽으로 가는 경우가 발생할 수 있다. 로봇은 앞으로 간다고 갔지만 왼쪽으로 가서 불에 빠질 수도 있고 오른쪽으로 갈 수도 있다. 그 확률을 표현하는 것이 "state transition probability"이다. 이렇게 어떠한 action을 취했을 경우 state가 deterministic하게 딱 정해지는 것이 아니고 확률적으로 정해지게 된다.



Deterministic

```python
# Register FrozenLake with is_slippery False
register(
    id='FrozenLake-v3',
    entry_point='gym.envs.toy_text:FrozenLakeEnv',
    kwargs={'map_name' : '4x4', 'is_slippery': False}
)

env = gym.make('FrozenLake-v3')
```

```
SFFF
FHFH
FFFH
HFFG

SFFF
FHFH
FFFH
HFFG
  (Right)
('State: ', 1, 'Action: ', 2,
SFFF
FHFH
FFFH
HFFG
  (Right)
('State: ', 2, 'Action: ', 2,
SFFF
FHFH
FFFH
HFFG
```

```
  (Down)
('State: ', 6, 'Action: ', 1,
SFFF
FHFH
FFFH
HFFG
  (Down)
('State: ', 10, 'Action: ', 1,
SFFF
FHFH
FFFH
HFFG
  (Down)
('State: ', 14, 'Action: ', 1,
SFFF
FHFH
FFFH
HFFG
  (Right)
('State: ', 15, 'Action: ', 2,
('Finished with reward', 1.0)
```

Stochastic (non-deterministic)

```
# is_slippery True
env = gym.make('FrozenLake-v0')
```

```
SFFF
FHFH
FFFH
HFFG

SFFF
FHFH
FFFH
HFFG
  (Right)
('State: ', 0, 'Action: ', 2,
SFFF
FHFH
FFFH
HFFG
  (Right)
('State: ', 4, 'Action: ', 2,
SFFF
FHFH
FFFH
HFFG
  (Down)
('State: ', 5, 'Action: ', 1,
('Finished with reward', 0.0)
```

```
SFFF
FHFH
FFFH
HFFG
  (Right)
('State: ', 0, 'Action: ', 2,
SFFF
FHFH
FFFH
HFFG
  (Right)
('State: ', 1, 'Action: ', 2,
SFFF
FHFH
FFFH
HFFG
  (Right)
('State: ', 1, 'Action: ', 2,
SFFF
FHFH
FFFH
HFFG
  (Right)
('State: ', 5, 'Action: ', 2,
('Finished with reward', 0.0)
```

일종의 noise다.

Agent가 Environment를 탐험하면서 얻는 보상을 **반환값(return)**이라고 한다.

$$G_t = R_{t+1} + R_{t+2} + R_{t+3} + R_{t+4} + R_{t+5} + \cdots$$

그러나, 지금 100만원 받는 것과 한 달 후에 100만원을 받는 것 중 지금 100만원을 받는 것이 더 큰 가치가 있으므로, 미래에 받게 될 보상에 대해서는 가치를 감가시킨다. (감가율(Discount Factor); $\boldsymbol{\gamma} \in [\mathbf{0}, \mathbf{1}]$)

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots$$

이는 한 에피소드(시행, 학습의 한 단위)가 끝나기 전에 어떤 값이 나오리라고 예측/기대할 수 있다.
즉 '나 agent는 앞으로 얼마의 **보상**을 받을 것 같다'라고 할 때, 이 값은 기대값이며
이를 **가치함수(value function)**라 한다.

$$v(s) = E[\boldsymbol{G_t}|S_t = s]$$

가치함수(value function)에는 두 가지 종류가 있다.

state-value function 그리고 action-value function

가치함수: $\boldsymbol{v(s) = E[G_t|S_t = s]}$

현재 상태의 가치 ($v_\pi(S_t)$) 와 다음 상태의 가치 ($v_\pi(S_{t+1})$)의 관계를 나타낸 식이 벨만 방정식이다.
더 정확하게는 state-value function에 대한 벨만 방정식이다.

$$\boldsymbol{v(s) = E[R_{t+1} + \gamma R_{t+2} + \ \gamma^2 R_{t+3} + \cdots |S_t = s]}$$
$$\boldsymbol{\Leftrightarrow v(s) = E[R_{t+1} + \gamma(R_{t+2} + \ \gamma R_{t+3} + \cdots)|S_t = s]}$$
$$\boldsymbol{\Leftrightarrow v(s) = E[R_{t+1} + \gamma G_{t+1}|S_t = s]}$$

하지만,
상태뿐만이 아닌, 행동에 대한 가치까지도 판단하기 위해 action-value function을 이용해야 한다.
action-value function을 다른 말로는 **Q-function**이라고 부른다.

$$\boldsymbol{Q_\pi(s,a) = \ E[R_{t+1} + \gamma Q_\pi(S_{t+1}, A_{t+1})|S_t = s, a_t = a]}$$
$$\boldsymbol{Q_\pi(s,a) = \ E[R_{t+1} + \gamma R_{t+2} + \ \gamma^2 R_{t+3} + \cdots |S_t = s, a_t = a]}$$

state-value function을 구하기 위해서는 action-value function을 먼저 구한 다음에,
그 구한 값을 이용해 state-value function을 구해야 한다. 이를 위해 Q-function에 정책을 곱한 것의 합을 구한다.

$$v_\pi(s) = \sum_{a \in A} \pi(a|s) Q_\pi(s,a)$$

$$v_\pi(s) = \sum_{a \in A} \pi(a|s)\, Q_\pi(s,a)$$

$$v_\pi(s) = \sum_{a \in A} \pi(a|s)\, \boldsymbol{E}[\boldsymbol{R_{t+1}} + \boldsymbol{\gamma Q_\pi}(\boldsymbol{S_{t+1}}, \boldsymbol{A_{t+1}}) | \boldsymbol{S_t = s}, \boldsymbol{a_t = a}]$$

기대값을 정확한 값으로 수치화시키기 위해, 컴퓨터가 계산할 수 있도록 수식을 변형시킨다.

$$v_\pi(s) = \sum_{a \in A} \pi(a|s)\, (R_{t+1} + \gamma \sum_{s' \in S} P^a_{ss'}\, v_\pi(s'))$$

상태 변환 확률($\sum_{s \in S} P^a_{ss'}$)이 1이라면 deterministic 환경이고,
그 외라면 stochastic 환경이라 할 수 있다.

현재, 상태 변환 확률을$P^a_{ss'}$= 1로 가정하므로

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

벨만 기대 방정식

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

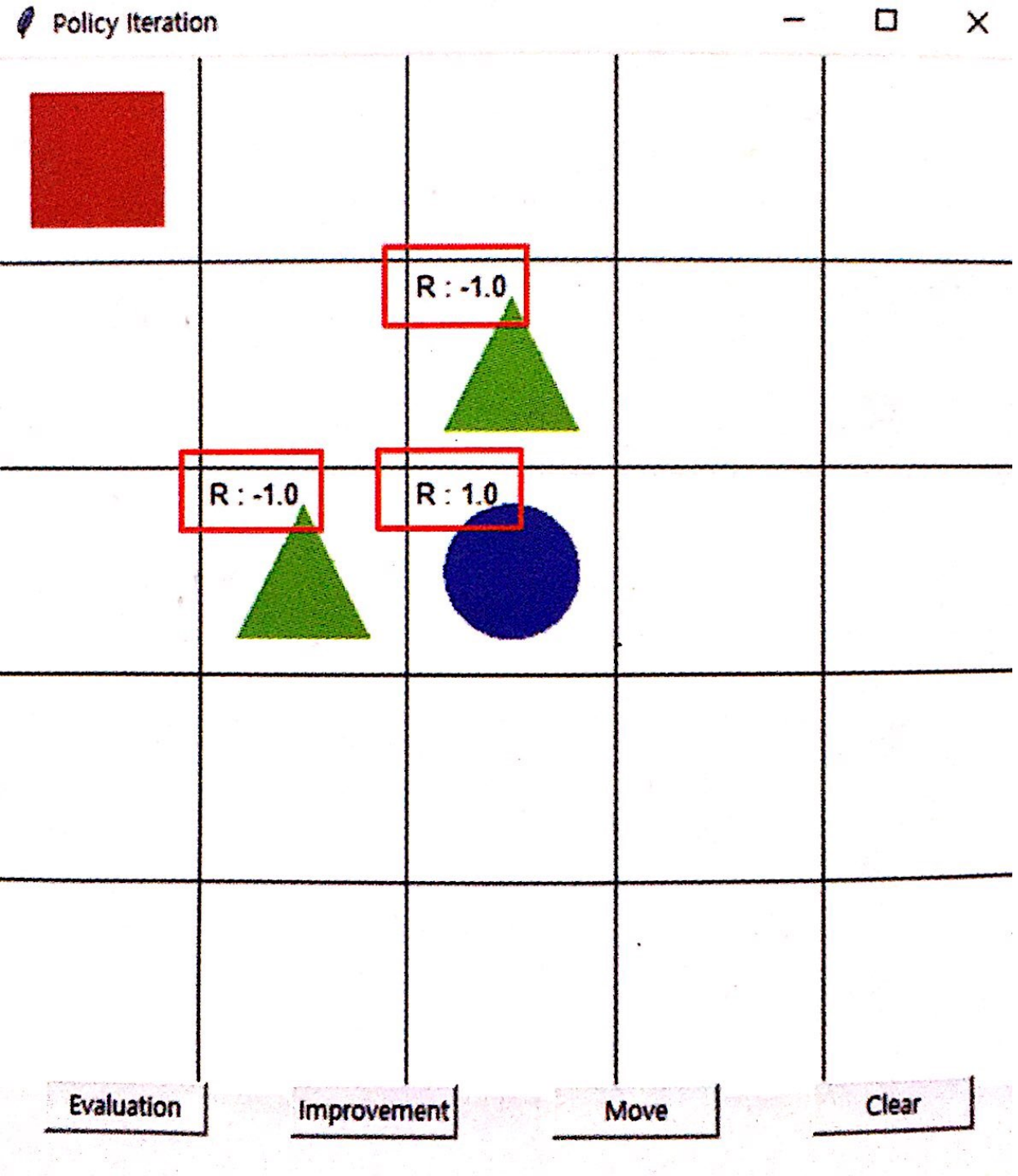
- 바로 주변 상태들에 대한 가치(보상)의 합을 구한다.
- 미래의 가치에 감가율을 적용한다.
- 현재 state에서 그 다음 state로 가는 action을 했을 때의 보상을 구한다.
- 해당 action a을 할 확률을 곱한다.
- 모든 선택 가능한 행동들에 대해 반복하여 그 값을 더한다.
- 모든 상태에 대해 반복한다.

위 과정을 여러 번 반복하여 수렴값을 구한다.

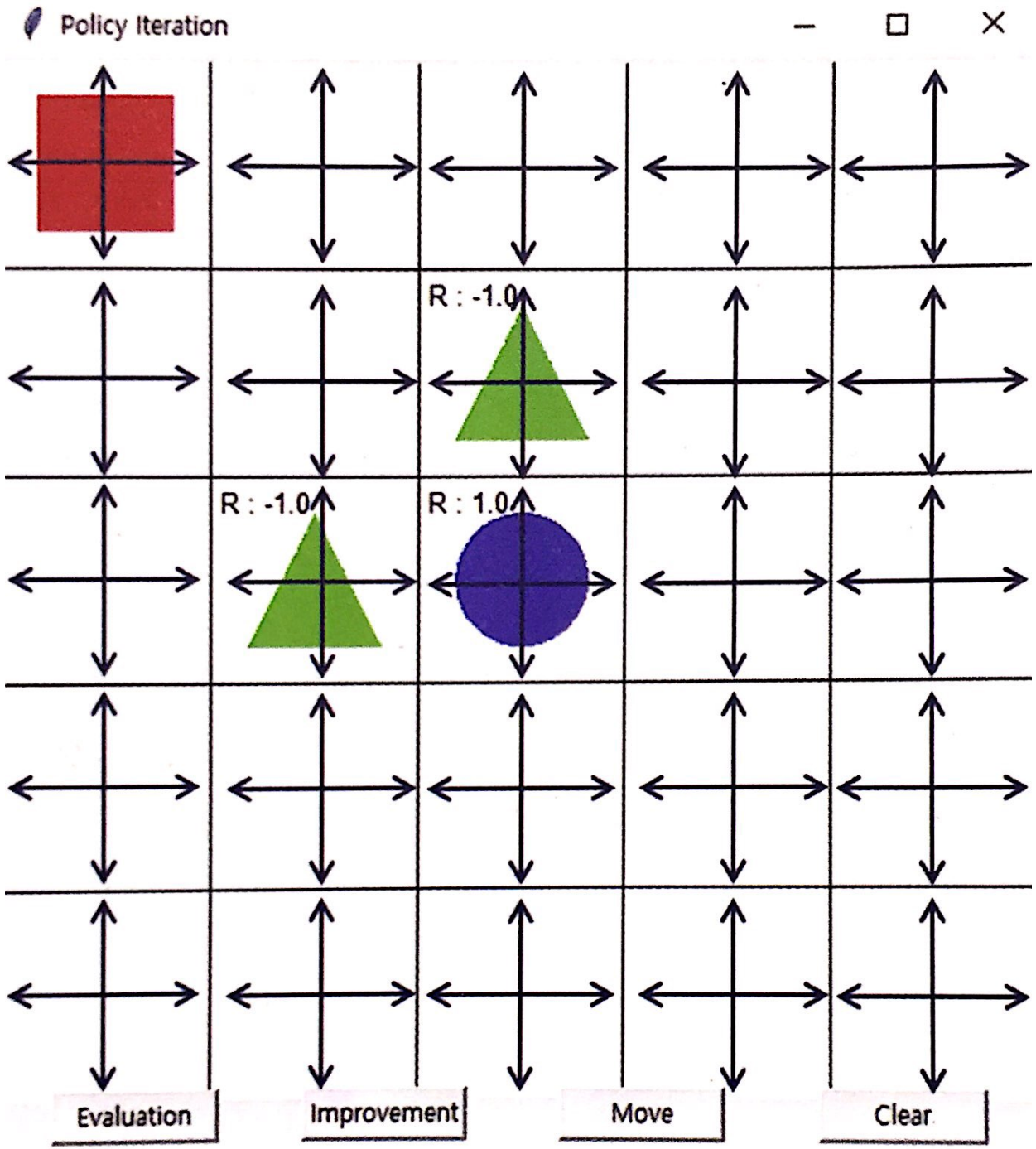
모든 상태에 대해서 가치 함수를 구하는 과정을 무한히 반복하면 최적의 벨만 방정식을 얻게 되며,

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

http://computingkoreanlab.com/app/jAI/jQLearning/



파란색에 도착하는 최적 정책을 찾는 것이 목표
즉 에이전트가 받을 보상의 합을 최대로 하는 것이 목표다.



처음에는 높은 보상을 얻게 하는 정책이 무엇인지 모른다.
따라서 처음에는 무작위로 행동을 정하는 정책부터 시작한다.

무작위 정책은 우리가 찾으려는 최적 정책이 아니다.

최적의 정책을 찾기 위해서는 현재의 정책을 **평가**하고
더 나은 정책으로 **발전**시키는 과정을 무한히 반복하면
정책은 결국 최적 정책으로 수렴한다.

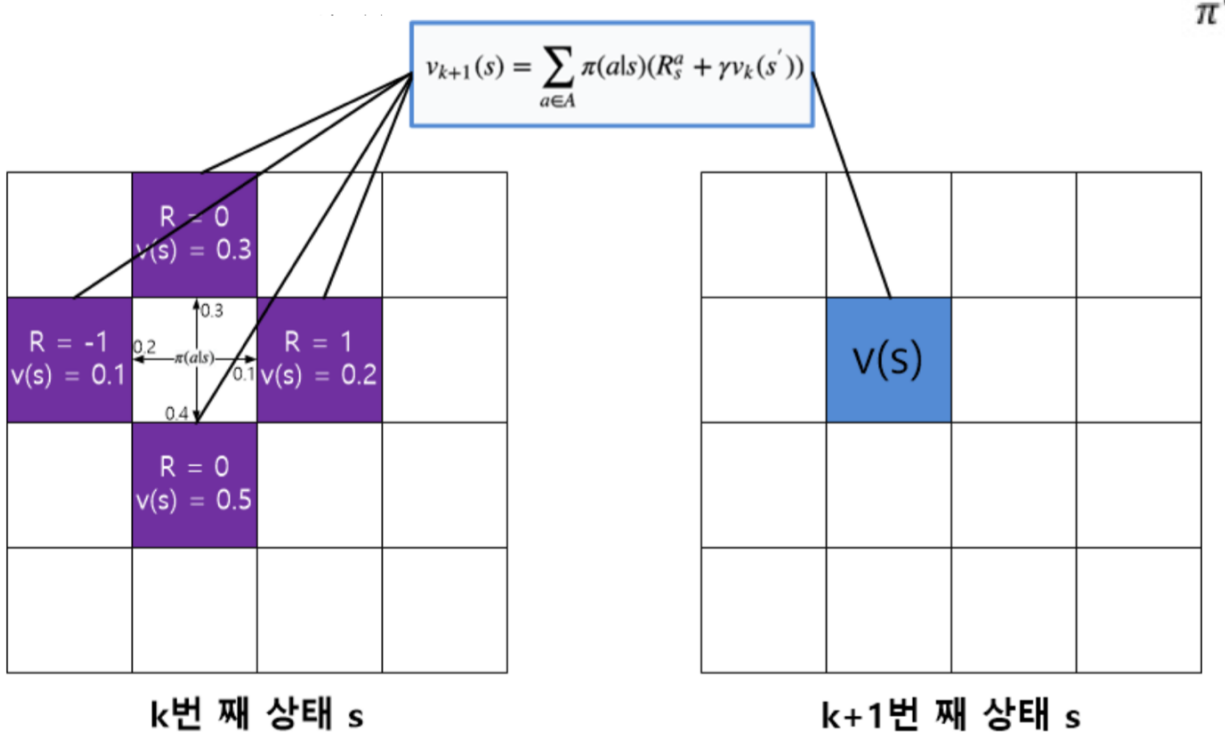
이를 풀려면
정책, 보상, 감가율, 다음 상태의 가치 함수를 알아야 한다.

감가율은 사용자가 정하기 나름
보상은 환경이 제공한다.

최적의 정책이 되려면 정책 발전을 해야 한다. 탐욕적인 방법(일정한 확률로 랜덤하게 행동 선택)을 사용하여 최대가 되는 방향으로 가는 정책을 선택한다.

구한 가치함수를 토대로 최대의 보상을 얻게 하는 행동을 선택하는 탐욕 정책 발전을 사용

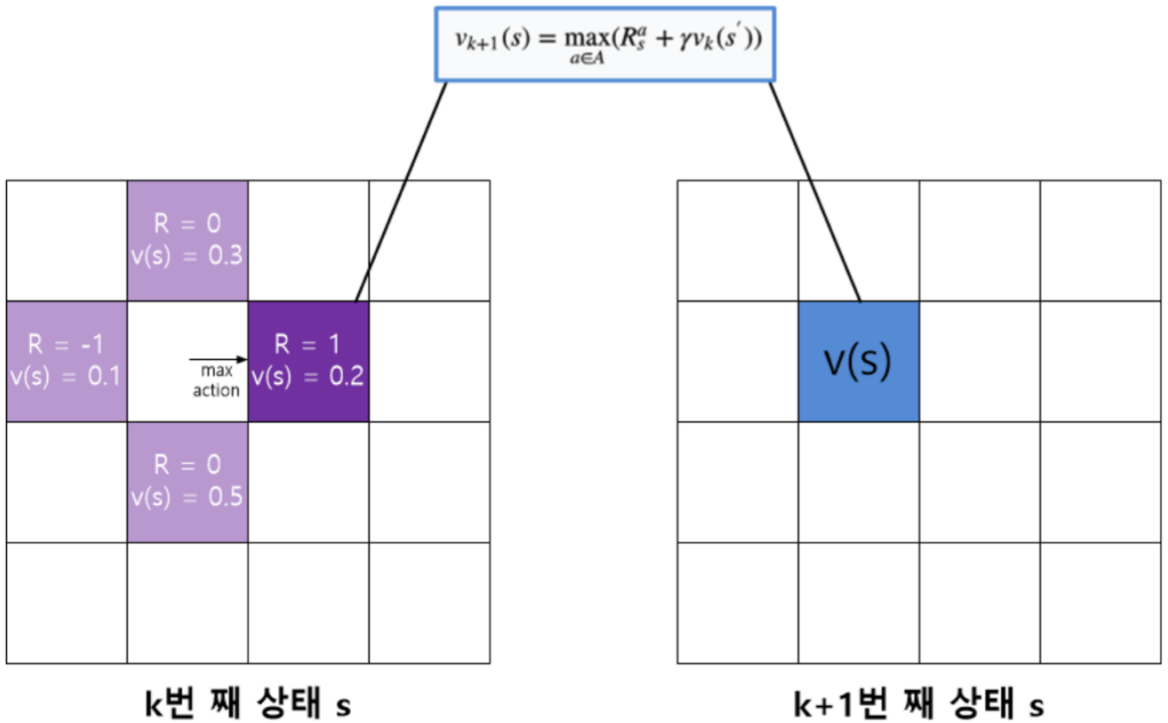
$$\pi'(s) = argmax_a \, q_\pi(s,a)$$



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

가치 이터레이션은 정책이 아닌 가치함수를 통해 다음 행동을 결정한다.

정책을 현재 가치함수가 최적이라고 가정



참고

파이썬과 케라스로 배우는 강화학습 (위키북스, 이용희, 양혁렬, 김건우, 이영무, 이의령 지음)

모두를 위한 강화학습 (김성훈, https://hunkim.github.io/ml/)

Q-learning Test (http://computingkoreanlab.com/app/jAI/jQLearning/)

Fundamental of Reinforcement Learning (https://www.gitbook.com/book/dnddnjs/rl/details)

상태

- “상태란, 에이전트의 현재 상황이다.”

에이전트가 관찰하는 자신의 상황, 위치 등의 정보

그리드월드(격자로 이루어진, 2차원으로 정보를 표현하는 공간)를 예로 들면, 에이전트의 현재 좌표가 에이전트의 현재 상황이 된다.

사람으로 치면, 눈을 통해 “나는 a방에 있다”라고 인식할 때의 “a방”이 상태가 될 수 있다.

“상태는 확률변수다.”

확률변수란? 매 시행마다 값이 달라지는 것

왜 상태가 확률변수일까? 어떤 에피소드(또는 시행. 학습의 한 단위, 학습 횟수)에서,

에이전트가 t=1일 때, 상태가 (1,3)일 때도 있고

또 다른 어떤 에피소드에서,

에이전트가 t=1일 때, 상태가 (4,2)일 수도 있다. (물론 같을 수도 있다.)

동일한 시간인 순간을 캡쳐한다면, 각 이미지마다 공의 위치가 매번 같을 수도, 다를 수도 있는 것.

행동

- “행동이란, 상태가 s인 에이전트가 취할 수 있는 행동의 집합이다.”

집합이란, 그 특징이 명확히 정의될 수 있는 것들의 모임.

예를 들면, x축과 y축으로 정보를 표현할 수 있는 공간상에서 방향성을 갖는 것들의 모임

“행동도 확률변수다.”

왜? 어떤 에피소드(또는 시행. 학습의 한 단위, 학습 횟수)에서,

현재 상태가 s인 에이전트가 a라는 행동을 한다고 할 때,

또 다른 어떤 에피소드의 같은 시간, 같은 상태에서 다른 행동을 할 수도, 같은 행동을 할 수도 있기 때문이다.

상태와 행동의 이러한 확률변수적인 특징을 non-determinstic(또는 stochastic)이라고 한다.

state-value function (가치함수)

- 상태를 고려한 가치함수 $v(s) = E[\boldsymbol{G_t}|S_t = s]$

$\Leftrightarrow v(s) = E[\boldsymbol{R_{t+1}} + \boldsymbol{\gamma R_{t+2}} + \boldsymbol{\gamma^2 R_{t+3}} + \cdots |S_t = s]$

$\Leftrightarrow v(s) = E[\boldsymbol{R_{t+1}} + \boldsymbol{\gamma}(\boldsymbol{R_{t+2}} + \boldsymbol{\gamma R_{t+3}} + \cdots)|S_t = s]$

$\Leftrightarrow v(s) = E[\boldsymbol{R_{t+1}} + \boldsymbol{\gamma G_{t+1}}|S_t = s]$

$v(\boldsymbol{S_{t+1}}) = E[\boldsymbol{R_{t+2}} + \boldsymbol{\gamma R_{t+3}} + \cdots)|S_{t+1} = s']$ 이므로

$\Leftrightarrow v(s) = E[\boldsymbol{R_{t+1}} + \boldsymbol{\gamma v}(\boldsymbol{S_{t+1}})|S_t = s]$

여기에 정책을 고려하면, $\boldsymbol{v_\pi(s) = E_\pi[R_{t+1} + \gamma v_\pi(S_{t+1})|S_t = s]}$ 또는 $\boldsymbol{v_\pi}\,(s) = E[\boldsymbol{G_t}|S_t = s]$

이를 '벨만 방정식 (다이내믹 프로그래밍 방정식)'이라고 함

벨만 방정식에선, 가치($v_\pi(S_{t+1})$)가 가장 높은 값을 선택하여 $v_\pi(s)$를 업데이트하게 되어 있다.

김성훈 교수님 강의에선, $\hat{Q}(s,a) = r + \gamma\, max_{a'}\hat{Q}(s',a')$

non-determinstic한 환경이라면, $\hat{Q}(s,a) = (1-\alpha)r + \alpha\,\gamma\, max_{a'}\hat{Q}(s',a')$.

기존의 Q값을α(α=0.1이면 10%)정도만 참고하겠다는 아이디어. 일종의 learning rate.

action-value function (가치함수)

- 상태와 함께 행동까지 고려한 가치함수

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

이 때, $q_\pi(s,a)$는 Q-function이며, 행동 각각에 대한 가치함수를 계산한다.
(즉 에이전트의 action에 대한 점수(quality)이다.)

"동적" : 대상이 시간에 따라 변하는 것
"프로그래밍" : 계획하는 것. 여러 프로세스가 다단계로 이루어지는 것

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

이 때,
정책이 얼마나 좋은지를 판단하는 근거로 쓰이기 위해
가치 함수라는 것이 쓰인다.

가치함수란,
현재의 정책 π를 따라갔을 때 받게 될 것이라고 기대되는 보상을 기대값으로 나타낸 것이다.

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

하지만,
더욱 멀리 있는 미래의 가치를 고려하면 고려할수록
경우의 수가 기하급수적으로 늘어나는 문제가 있다.
즉 가치 함수를 계산하기 어렵게 된다.

이 때, 벨만 방정식이 가치 함수의 계산량 문제를 해결해준다.

“주변 상태의 가치함수와 한 타임스텝의 보상만을 고려하여
현재 상태의 다음 가치 함수를 계산하겠다”