# Exercise 1

Yoon-gu Hwang

August 5, 2022

## 1 Installation

다음과 같이 필요한 패키지를 설치합니다.

```
(base)$ conda create --name=rl python=3.8
(base)$ conda activate rl
(rl)  $ conda install jupyter notebook matplotlib
(rl)  $ pip install gym==0.20
```

## 2 About Environment

첨부된 파일의 `frozenlake.py`에서 `FrozenLakeEnv`를 생성합니다.

```
from frozenlake import FrozenLakeEnv
env = FrozenLakeEnv()
```

`print(env.desc)`를 실행하면 다음과 같이 $4 \times 4$ Grid가 출력됩니다.

```
[['S' 'F' 'F' 'F']
 ['F' 'H' 'F' 'H']
 ['F' 'F' 'F' 'H']
 ['H' 'F' 'F' 'G']]
```

이 Environment는 `'S'`에서 시작해서 `'G'`에 도착하면 되는 MDP입니다. `'F'`는 얼음(Frozen)이고 `'H'` 구멍(Hall)입니다.

### 2.1 State space $\mathcal{S}^+$

다음과 같이 16개의 State가 있습니다.

$$\mathcal{S}^+ = \{ 0, 1, 2, \cdots, 15 \} \tag{1}$$

1행부터 차례대로 $0, 1, 2, \cdots, 15$ 순서입니다. 각 칸에 알파벳 대신 State를 쓰면 다음과 같습니다.

```
# Alphabet
[['S' 'F' 'F' 'F']
 ['F' 'H' 'F' 'H']
 ['F' 'F' 'F' 'H']
 ['H' 'F' 'F' 'G']]

# State
[[ 0   1   2   3 ]
 [ 4   5   6   7 ]
 [ 8   9  10  11 ]
 [12  13  14  15 ]]
```

### 2.2 Action space $\mathcal{A}$

Action space는 다음과 같이 4가지 입니다. 0은 Left, 1은 Down, 2는 Right, 그리고 3은 Up입니다.

$$\mathcal{A} = \{ 0, 1, 2, 3 \} \tag{2}$$

## 2.3 Reward $R$

Reward는 최종 목적지 'G'에 도착한 경우에만 1이 주어집니다. 이러한 Task를 Sparse Reward Task라고 합니다.

## 2.4 Transition Probability, P

```
state = 1
action = 0 # Left
possible = env.P[state][action]
```

print(possible)을 실행하면 다음과 같이 Tuple의 List가 나옵니다.

```
[(0.3333333333333333, 1, 0.0, False),
 (0.3333333333333333, 0, 0.0, False),
 (0.3333333333333333, 5, 0.0, True )]
```

Remark 2.1. List안의 각 원소들은 (확률, 다음 State, Reward, 에피소드 종료 여부) 총 4가지로 구성되어 있습니다.

이것을 확률로 나타내면 다음과 같습니다.

$$\mathrm{P}(S_{t+1}=s', R_{t+1}=r|S_t=1, A_t=1) = \begin{cases} \frac{1}{3} \text{ if } s'=1, r=0 \\ \frac{1}{3} \text{ if } s'=0, r=0 \\ \frac{1}{3} \text{ if } s'=5, r=0 \\ 0 \text{ else} \end{cases} \tag{3}$$

이번엔 같은 위치에서 아래쪽($A_t=1$)으로 이동한 경우를 살펴보겠습니다.

```
state = 1
action = 1
possible = env.P[state][action]
```

다시 한번 print(possible)을 실행하면 다음과 같은 출력이 나옵니다.

```
[(0.3333333333333333, 0, 0.0, False),
 (0.3333333333333333, 5, 0.0, True ),
 (0.3333333333333333, 2, 0.0, False)]
```

이 때, 확률 표현은 다음과 같습니다.

$$\mathrm{P}(S_{t+1}=s', R_{t+1}=r|S_t=1, A_t=1) = \begin{cases} \frac{1}{3} \text{ if } s'=0, r=0 \\ \frac{1}{3} \text{ if } s'=5, r=0 \\ \frac{1}{3} \text{ if } s'=2, r=0 \\ 0 \text{ else} \end{cases} \tag{4}$$

env에는 다음과 같은 변수들이 있습니다. State와 Action에 관한 정의와  입니다.

```
print(env.observation_space)
print(env.action_space)
print(env.nS)
print(env.nA)
```

출력은 다음과 같습니다.

```
Discrete(16)
Discrete(4)
16
4
```

# 3 Problem 1. Policy Evaluation

다음과 같이 주어진 Random Policy에 대해서 Policy Evaluation을 하는 코드를 작성합니다.

```
import numpy as np
random_policy = np.ones([env.nS, env.nA]) / env.nA
```

**Algorithm 1:** Policy Evaluation

Input: MDP, policy $\pi$, small positive number $\theta$
Output: $V \approx v_\pi$
Initialize $V$ arbitrarily (e.g., $V(s) = 0$ for all $s \in \mathcal{S}^+$)
repeat
  $\Delta \leftarrow 0$
  for $s \in \mathcal{S}$ do
    $v \leftarrow V(s)$
    $V(s) \leftarrow \sum_{a \in \mathcal{A}(s)} \pi(a|s) \sum_{s' \in \mathcal{S}, r \in \mathcal{R}} p(s', r|s, a)(r + \gamma V(s'))$
    $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
  end
until $\Delta < \theta$;
return $V$

```python
def policy_evaluation(env, policy, gamma=1, theta=1e-8):
    V = np.zeros(env.nS)

    # TODO

    return V
```

아마 다음과 같은 그림을 확인할 수 있습니다. 현재는 Random Policy이므로 Figure 1과 다를 수 있습니다.

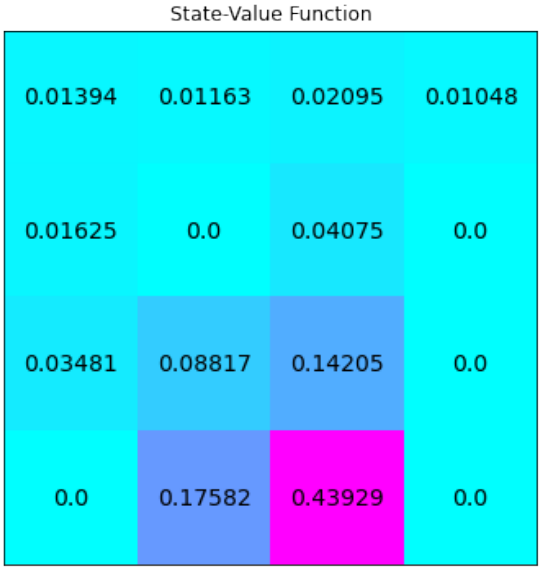


Figure 1: Random Policy의 Evaluation

# 4 Problem 2. Value function $v_\pi(s)$ and Action-value function $q_\pi(s, a)$

Value Function을 이용하여 Action-Value function를 구하는 코드를 작성합니다.

**Algorithm 2:** Estimation of Action Values

Input: MDP, state-value function $V$
Output: action-value function $Q$
for $s \in \mathcal{S}$ do
  for $a \in \mathcal{A}(s)$ do
    $Q(s, a) \leftarrow \sum_{s' \in \mathcal{S}, r \in \mathcal{R}} p(s', r|s, a)(r + \gamma V(s'))$
  end
end
return $Q$

```python
def q_from_v(env, V, s, gamma=1):
```

```
    q = np.zeros(env.nA)

    # TODO

    return q
```

작성한 코드를 다음과 같이 출력해봅니다.

```
Q = np.zeros([env.nS, env.nA])
for s in range(env.nS):
    Q[s] = q_from_v(env, V, s)
print(Q)
```

# 5 Problem 3. Policy Improvement

아래 Pseudo Code를 참고하여 Policy Improvement를 작성합니다.

**Algorithm 3:** Policy Improvement

Input: MDP, value function $V$
Output: policy $\pi'$
for $s \in \mathcal{S}$ do
  for $a \in \mathcal{A}(s)$ do
    $Q(s,a) \leftarrow \sum_{s' \in \mathcal{S}, r \in \mathcal{R}} p(s', r|s, a)(r + \gamma V(s'))$
  end
  $\pi'(s) \leftarrow \arg\max_{a \in \mathcal{A}(s)} Q(s,a)$
end
return $\pi'$

```
def policy_improvement(env, V, gamma=1):
    policy = np.zeros([env.nS, env.nA])

    # TODO

    return policy
```

# 6 Problem 4. Policy Iteration

위에서 작성한 2가지 함수를 활용하여 Policy Iteration을 구현합니다.

**Algorithm 4:** Policy Iteration

Input: MDP, small positive number $\theta$
Output: policy $\pi \approx \pi_*$
Initialize $\pi$ arbitrarily (e.g., $\pi(a|s) = \frac{1}{|\mathcal{A}(s)|}$ for all $s \in \mathcal{S}$ and $a \in \mathcal{A}(s)$)
$policy\text{-}stable \leftarrow false$
repeat
  $V \leftarrow$ Policy_Evaluation(MDP, $\pi, \theta$)
  $\pi' \leftarrow$ Policy_Improvement(MDP, $V$)
  if $\pi = \pi'$ then
    $policy\text{-}stable \leftarrow true$
  end
  $\pi \leftarrow \pi'$
until $policy\text{-}stable = true$;
return $\pi$

```
def policy_iteration(env, gamma=1, theta=1e-8):
    policy = np.ones([env.nS, env.nA]) / env.nA

    # TODO

    return policy, V
```

작성 후에 다음 코드를 실행하면 Optimal policy

```
policy_pi, V_pi = policy_iteration(env)
print(policy_pi)
plot_values(V_pi)
```

$\pi_*$와 Figure 2가 다음과 같이 나옵니다.

```
[[1.   0.   0.   0.  ]
 [0.   0.   0.   1.  ]
 [0.   0.   0.   1.  ]
 [0.   0.   0.   1.  ]
 [1.   0.   0.   0.  ]
 [0.25 0.25 0.25 0.25]
 [0.5  0.   0.5  0.  ]
 [0.25 0.25 0.25 0.25]
 [0.   0.   0.   1.  ]
 [0.   1.   0.   0.  ]
 [1.   0.   0.   0.  ]
 [0.25 0.25 0.25 0.25]
 [0.25 0.25 0.25 0.25]
 [0.   0.   1.   0.  ]
 [0.   1.   0.   0.  ]
 [0.25 0.25 0.25 0.25]]
```

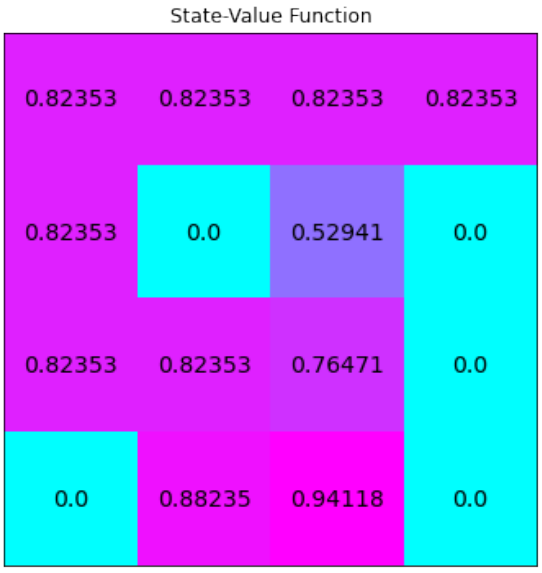


Figure 2: Policy Iteration으로 구한 $v_*(s)$

# 7 Problem 5. Value Iteration

Value Iteration을 활용해서 Optimal policy와 Optimal value function을 구합니다.

**Algorithm 5:** Value Iteration

**Input:** MDP, small positive number $\theta$
**Output:** policy $\pi \approx \pi_*$
Initialize $V$ arbitrarily (e.g., $V(s) = 0$ for all $s \in \mathcal{S}^+$)
repeat
  $\Delta \leftarrow 0$
  for $s \in \mathcal{S}$ do
    $v \leftarrow V(s)$
    $V(s) \leftarrow \max_{a \in \mathcal{A}(s)} \sum_{s' \in \mathcal{S}, r \in \mathcal{R}} p(s', r|s, a)(r + \gamma V(s'))$
    $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
  end
until $\Delta < \theta$;
$\pi \leftarrow$ Policy_Improvement(MDP, $V$)
return $\pi$

```python
def value_iteration(env, gamma=1, theta=1e-8):
    V = np.zeros(env.nS)

    # TODO
    policy = policy_improvement(env, V, gamma)
    return policy, V
```

다음 코드를 실행하면

```python
policy_vi, V_vi = value_iteration(env)
print(policy_vi)

plot_values(V_vi)
```

$\pi_*$와 Figure 3가 다음과 같이 나옵니다.

```
[[1.   0.   0.   0.  ]
 [0.   0.   0.   1.  ]
 [0.   0.   0.   1.  ]
 [0.   0.   0.   1.  ]
 [1.   0.   0.   0.  ]
 [0.25 0.25 0.25 0.25]
 [0.5  0.   0.5  0.  ]
 [0.25 0.25 0.25 0.25]
 [0.   0.   0.   1.  ]
 [0.   1.   0.   0.  ]
 [1.   0.   0.   0.  ]
 [0.25 0.25 0.25 0.25]
 [0.25 0.25 0.25 0.25]
 [0.   0.   1.   0.  ]
 [0.   1.   0.   0.  ]
 [0.25 0.25 0.25 0.25]]
```

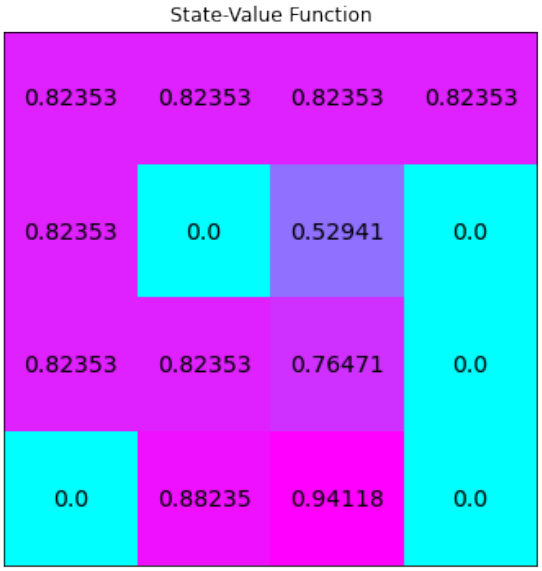


Figure 3: Value Iteration 방법으로 구한 $v_*(s)$

# 8 Optional

1. 더 큰 사이즈로 해보기: `env = FrozenLakeEnv(map_name='8x8')`
2. Deterministic으로 변경해서 해보기: `env = FrozenLakeEnv(is_slippery=False)`
3. 위의 2가지 섞어서 해보기: `env = FrozenLakeEnv(map_name='8x8', is_slippery=False)`