Exercise 2

Yoon-gu Hwang

August 11, 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==0.9.6 scipy tqdm
```

2 About Environment

BlackJack 게임이 Environment로 구현되어 있습니다.

```
import sys
import gym
import numpy as np
from collections import defaultdict
from plot_utils import plot_blackjack_values, plot_policy

env = gym.make('Blackjack-v0')

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

State와 Action은 다음과 같은 구조로 되어 있습니다.

```
Tuple(Discrete(32), Discrete(11), Discrete(2))
Discrete(2)
```

State와 Action에 대한 상세한 내용은 다음 docstring을 통해 자세히 알 수 있습니다.

```
"""
Blackjack is a card game where the goal is to beat the dealer by obtaining cards
that sum to closer to 21 (without going over 21) than the dealers cards.
### Description
Card Values:
- Face cards (Jack, Queen, King) have a point value of 10.
- Aces can either count as 11 (called a 'usable ace') or 1.
- Numerical cards (2-9) have a value equal to their number.
This game is played with an infinite deck (or with replacement).
The game starts with the dealer having one face up and one face down card,
while the player has two face up cards.
The player can request additional cards (hit, action=1) until they decide to stop (stick, action=0)
or exceed 21 (bust, immediate loss).
After the player sticks, the dealer reveals their facedown card, and draws
until their sum is 17 or greater.  If the dealer goes bust, the player wins.
If neither the player nor the dealer busts, the outcome (win, lose, draw) is
decided by whose sum is closer to 21.
### Action Space
```

```
There are two actions: stick (0), and hit (1).
### Observation Space
The observation consists of a 3-tuple containing: the player's current sum,
the value of the dealer's one showing card (1-10 where 1 is ace),
and whether the player holds a usable ace (0 or 1).
This environment corresponds to the version of the blackjack problem
described in Example 5.1 in Reinforcement Learning: An Introduction
by Sutton and Barto (http://incompleteideas.net/book/the-book-2nd.html).
### Rewards
- win game: +1
- lose game: -1
- draw game: 0
- win game with natural blackjack:
    +1.5 (if <a href="#nat">natural</a> is True)
    +1 (if <a href="#nat">natural</a> is False)
"""
```

게임을 진행하는 코드는 다음과 같습니다. 매 실행마다 다른 결과가 나옵니다.

```
for i_episode in range(3):
    state = env.reset()
    while True:
        print(state)
        action = env.action_space.sample()
        state, reward, done, info = env.step(action)
        if done:
            print('End game! Reward: ', reward)
            print('You won :)\n') if reward > 0 else print('You lost :(\n')
            break
```

3 Problem 1. MC Prediction

이 문제에서는 다음 `generate_episode_from_limit_stochastic()`로 Episode를 생성해주는 함수를 이용합니다.

```
def generate_episode_from_limit_stochastic(bj_env):
    episode = []
    state = bj_env.reset()
    while True:
        probs = [0.8, 0.2] if state[0] > 18 else [0.2, 0.8]
        action = np.random.choice(np.arange(2), p=probs)
        next_state, reward, done, info = bj_env.step(action)
        episode.append((state, action, reward))
        state = next_state
        if done:
            break
    return episode
```

`generate_episode_from_limit_stochastic()`가 입력으로 들어가서 다음 함수를 채우는 것이 1번 문제입니다. First-Visit을 사용해도 되고, Every-Visit을 사용해도 됩니다.

```
def mc_prediction_q(env, num_episodes, generate_episode, gamma=1.0):
    # initialize empty dictionaries of arrays
    returns_sum = defaultdict(lambda: np.zeros(env.action_space.n))
    N = defaultdict(lambda: np.zeros(env.action_space.n))
    Q = defaultdict(lambda: np.zeros(env.action_space.n))
    for i_episode in tqdm(range(1, num_episodes+1)):
        # TODO
        pass

    return Q
```

Algorithm 1: First-Visit MC Prediction (for action values)

Input: policy π, positive integer $num\_episodes$
Output: value function Q ($\approx q_\pi$ if $num\_episodes$ is large enough)
Initialize $N(s,a) = 0$ for all $s \in \mathcal{S}, a \in \mathcal{A}(s)$
Initialize $returns\_sum(s,a) = 0$ for all $s \in \mathcal{S}, a \in \mathcal{A}(s)$
for $i \leftarrow 1$ **to** $num\_episodes$ **do**
  Generate an episode $S_0, A_0, R_1, \ldots, S_T$ using π
  for $t \leftarrow 0$ **to** $T-1$ **do**
    if (S_t, A_t) is a first visit (with return G_t) **then**
      $N(S_t, A_t) \leftarrow N(S_t, A_t) + 1$
      $returns\_sum(S_t, A_t) \leftarrow returns\_sum(S_t, A_t) + G_t$
  end
end
$Q(s,a) \leftarrow returns\_sum(s,a)/N(s,a)$ for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$
return Q

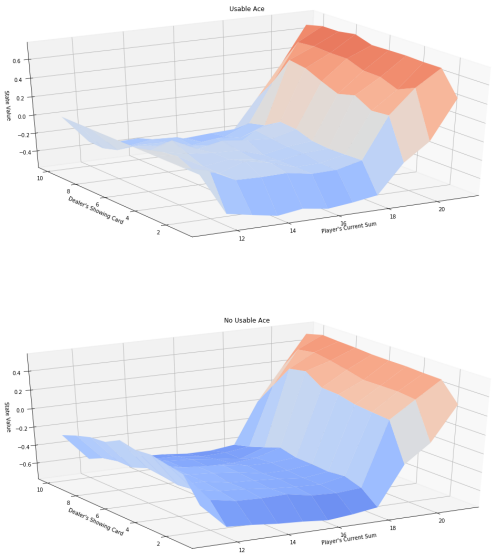

Figure 1: MC Prediction 결과

함수를 작성하고 다음 코드를 실행하면 Figure 1와 같은 그래프를 얻을 수 있습니다.

```
# obtain the action-value function
Q = mc_prediction_q(env, 500000, generate_episode_from_limit_stochastic)

# obtain the corresponding state-value function
V_to_plot = dict((k,(k[0]>18)*(np.dot([0.8, 0.2],v)) + (k[0]<=18)*(np.dot([0.2, 0.8],v))) \
         for k, v in Q.items())

# plot the state-value function
plot_blackjack_values(V_to_plot)
```

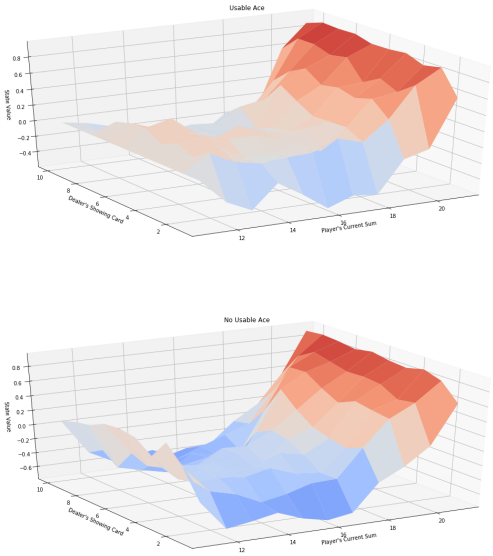

Figure 2: MC Control 결과(Value Function)

4 Problem2. MC Control

이번에는 MC Control을 구현하는 문제입니다. Episode를 생성은 ϵ-greedy 알고리즘을 통해 합니다.

Algorithm 2: First-Visit Constant-α (GLIE) MC Control

Input: positive integer $num\_episodes$, small positive fraction α, GLIE $\{\epsilon_i\}$
Output: policy π ($\approx \pi_*$ if $num\_episodes$ is large enough)
Initialize Q arbitrarily (e.g., $Q(s,a) = 0$ for all $s \in \mathcal{S}$ and $a \in \mathcal{A}(s)$)
for $i \leftarrow 1$ **to** $num\_episodes$ **do**
  $\epsilon \leftarrow \epsilon_i$
  $\pi \leftarrow \epsilon$-greedy(Q)
  Generate an episode $S_0, A_0, R_1, \ldots, S_T$ using π
  for $t \leftarrow 0$ **to** $T-1$ **do**
    if (S_t, A_t) is a first visit (with return G_t) **then**
      $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(G_t - Q(S_t, A_t))$
  end
end
return π

다음 함수 `mc_control()`의 빈칸을 채우는 것 2번째 문제입니다.

```python
def mc_control(env, num_episodes, alpha, gamma=1.0, eps_start=1.0, eps_decay=.99999, eps_min=0.05):
    nA = env.action_space.n
    Q = defaultdict(lambda: np.zeros(nA))
    epsilon = eps_start
    # loop over episodes
    for i_episode in tqdm(range(1, num_episodes+1)):
        # TODO
        pass

    return policy, Q
```

코드를 다 작성한 후 다음 코드를 실행하면 Figure 2와 Figure 3과 비슷하게 나옵니다.

```python
policy, Q = mc_control(env, 500000, 0.02)
V = dict((k,np.max(v)) for k, v in Q.items())
plot_blackjack_values(V)
```

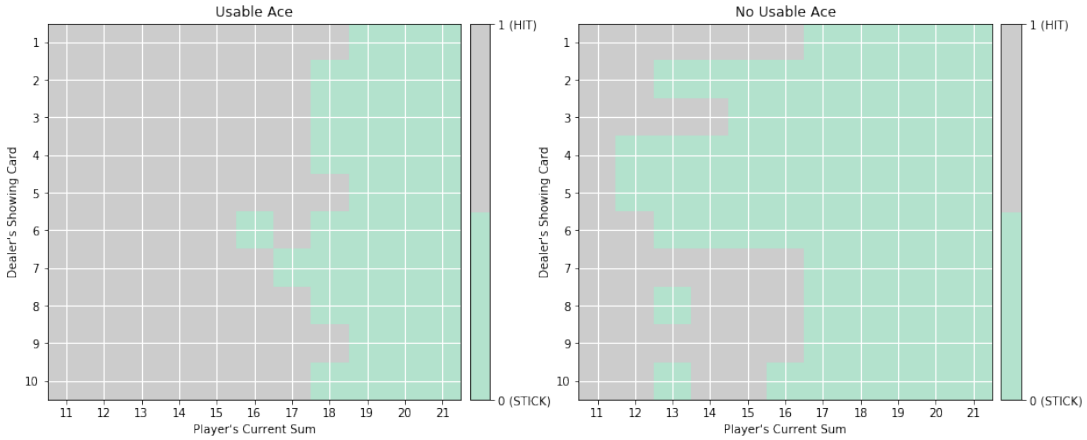

Figure 3: MC Control 결과 (Policy)

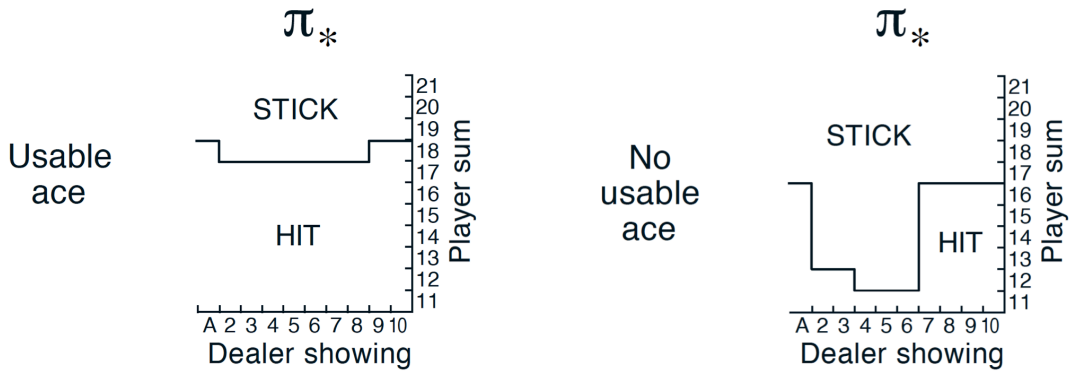

Figure 4: Optimal Policy