데이터마이닝 - black jack 과제

컴퓨터공학과

12161558 김혜윤

20.06.22

1. 목표

- 주어진 코드로부터 최적 policy를 학습시킨 agent를 사용해, 블랙 잭 게임을 시뮬레이션한 뒤 승률을 계산.

2. 초기 설정

- 플레이어는 1,000,000번의 에피소드로부터 최적 policy를 학습
- 플레이어의 초기 자금 : 10,000 달러
- 게임 참가 시 10 달러를 지불, 결과에 따라 금액을 획득
    -> 승리 시 : 20 달러 획득
    -> 무승부 시 : 10달러 획득
    -> 패배 시 : 0 달러 획득

3. 요구사항

1) 플레이어와 딜러가 1,000번의 게임을 진행
2) 1,000번의 게임 후 플레이어의 승률을 계산
3) 매 게임 별 플레이어의 소지 금 변화를 그래프로 시각화

4. 코드 작성

- 2가지 코드를 작성했음.

```
for i in range(1000):
    deck.reset()
    episodes = mc_es.generate_episode(dealer,agent,deck)
    agent.update_qval(episodes)
    p_money-=10
    if episodes[-1][-1]==1:
        p_money+=20
        win+=1
    elif episodes[-1][-1] == 0:
        p_money+=10
        draw+=1
    else:
        lose+=1
    if p_money < 0:
        p_money = 0
    money_ch.append(p_money)
```

i. train 시키지 않고 그냥 게임 만 진행해서 하고, 각 reward를 확인하여 획득 또는 잃은 금액을 변경 해주었다.

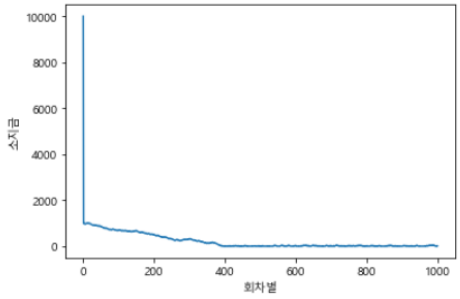
```
############## 코드 작성 ############################
print("Total win rate : {:.3f}%" .format(win/(win+lose)*100))
print("--- TOTAL Games WIN :" , win, "DRAW:",draw, "LOSS:", lose)

#####################################################
```

```
Total win rate : 34.292%
--- TOTAL Games WIN : 322 DRAW: 61 LOSS: 617
```

승률 계산은 각 에피소드를 진행하면서 reward를 인해 win, lose, draw 횟수를 저장했고 그를 이용해 win / (win + lose) *100 으로 확인했다. 그 결과 34.292% 의 승률이 나왔고,

```
############## 코드 작성 ############################
import matplotlib
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd

plt.rc('font', family='Malgun Gothic')

MoneyChange = pd.Series(money_ch)
plt.xlabel("회차별")
plt.ylabel("소지금")
MoneyChange.plot()

#####################################################
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x25861286348>
```



매 에피소드에서 저장한 소지금을 money_ch 라는 list에 저장하여 pd.Series(money_ch)로 그래프를 출력했다.

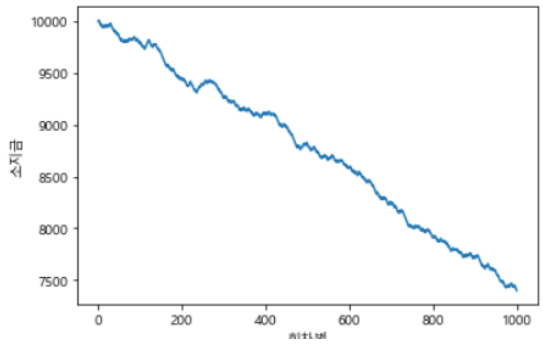
ii. 1000번의 게임또한 train method로 학습을 시켜 승률을 구하고, 소지금 그래프를 출력했다. 여기서는 train method를 약간 변경하였다.

```
    if episode[-1][-1] == 1:
        win += 1
        p_money += 10

    elif episode[-1][-1] == 0:
        draw += 1

    else:
        loss += 1
        p_money -= 10

    if p_money < 0:
        p_money = 0

    l_money.append(p_money)

    if count % 1000 == 0 and verbose == True:
        total_win += win
        total_loss += loss
        total_draw += draw

        #print("========== Training : Episode ",
        #print("Recent 1000 games win rate :{:.
        #print(" -- 1000 Games WIN :", win, "DR
        #print("Total win rate : {:.3f}%".forma
        #print(" -- TOTAL Games WIN :", total_w

        win = 0
        loss = 0
        draw = 0

return total_win,total_loss,total_draw,l_money
```

소지금 변경과 return 코드를 넣어 주었다.

```
win, loss, draw,l_money  = mc_es.train(dealer, agent, deck, it=1000)
print("Total win rate : {:.3f}%".format(win / (win + loss) * 100))
print(" -- TOTAL Games WIN :", win, "DRAW :", draw, "LOSS :", loss)

import matplotlib
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd

plt.rc('font', family='Malgun Gothic')

MoneyChange = pd.Series(l_money)
plt.xlabel("회차별")
plt.ylabel("소지금")
MoneyChange.plot()
```

```
Total win rate : 36.258%
 -- TOTAL Games WIN : 343 DRAW : 54 LOSS : 603

<matplotlib.axes._subplots.AxesSubplot at 0x2586117fcc8>
```



그 결과 첫번째보다 높은 승률인 36.258% 를 보였고, 소지금 그래프 또한 첫번째보다 떨어지는 수준이 낮아졌다. 그것으로 지는 확률이 낮아짐을 알 수 있다.

5. 추가 과제 : 플레이어의 승률 높이기

i) state를 수정하여 승률이 더 높은 policy를 찾기

ii) 딜러는 게임이 끝났을 때, 남은 카드의 수를 확인

iii) 15장 이상이라면 해당 덱을 다음 게임에서 그대로 사용하고, 미만이라면 셔플된 새로운 덱을 사용

6. 코드 작성

-
```
# 카드 덱, 딜러, Agent를 초기화
#deck.reset()
dealer.reset()
agent.reset()
agent.hit(deck)
agent.hit(deck)
dealer.hit(deck)
dealer.hit(deck)
```
MonteCarlo class 내의 generate_episode method에서 deck.reset() 을 제외해주고,

```
dealer_num = 0
agent_num = 0
p_money = 10000
l_money = [10000, ]
for i in range(it):
    count += 1
    dealer_num += dealer.num()
    agent_num += agent.num()

    if i==0 or (52-(dealer_num+agent_num)) < 15:
        deck.reset()
        dealer_num = 0
        agent_num = 0
    else:
        pass
```
train method에서 dealer_num은 딜러가 소지한 카드의 개수, agent_num은 player가 소지한 카드의 개수로 설정하고, 매 에피소드마다 deck의 개수를 확인하고, 15개 미만이라면 deck을 reset해주었다. 더 이상의 방법을 찾지 못했고 이렇게만 해본 결과, 승률은 35.466%로 변경하기 전과 별다름이 없었다. State 변경을 하였어야 했지만 못한 부분 때문에 별다르지 않았던 것 같다.

```
Total win rate : 35.466%
 -- TOTAL Games WIN : 333311 DRAW : 60206 LOSS : 606483

<matplotlib.axes._subplots.AxesSubplot at 0x258051ca088>
```