Monte Carlo ES for Blackjack

Jingyi Zhao

October 2022

1 Problem Setup: State space and Action space

Let S denote the state space and A denote the action space.
Let $s = (i, j) \in S$ where i represents the player's sum and j represents the dealer's sum.
Let $a \in A = \{0, 1\}$ where $a = 1$ indicates *hit* and $a = 0$ indicates *stick*.

2 Pseudocode for Main Algorithms

Algorithm 1 Monte Carlo ES (Exploring Starts) for Blackjack

```
Initialize:
Create π, Q, Returns to be empty dictionaries
π[s] = 1 if player_sum ≥ 20 otherwise 0, for all s ∈ S
Q[(s, a)] = 0, for all s ∈ S, a ∈ A
Returns[(s, a)] = [ ] (empty list), for all s ∈ S, a ∈ A
loop for NUM_LOOP of times:
    Choose S_0 ∈ S, A_0 ∈ A randomly such that all pairs have possibility > 0
    Generate an episode from S_0, A_0 and π: S_0, A_0, S_1, A_1, ..., S_{T−1}, A_{T−1}, R_T
via Algorithm2
    G ← R_T
    for t = T-1, T-2, ..., 0 do
        Append G to Returns[(S_t, A_t)]
        Q[(S_t, A_t)] ← average(Returns[(S_t, A_t)])
        if Q[(S_t, 0)] > Q[(S_t, 1)] then
            π[S_t] = Q[(S_t, 0)]
        else
            if Q[(S_t, 0)] < Q[(S_t, 1)] then
                π[S_t] = Q[(S_t, 1)]
            end if
        end if
    end for
end loop
```

Algorithm 2 Generate Episode given S_0, A_0, π

Initialize:
$s = S_0$, $a = A_0$, *episode* $= [S_0, A_0]$
while $a = 1$ **do**
    player_sum $\leftarrow$ player_sum + a random card value
    if player_sum bursts **then**
        episode.append(-1)
        return *episode*
    end if
    Update s
    $a \leftarrow \pi[s]$
    Append s, a to *episode*
end while
while dealer_sum ≤ 16 **do**
    dealer_sum $\leftarrow$ dealer_sum + a random card value
    if dealer_sum bursts **then**
        episode.append(1)
        return *episode*
    end if
end while
Compare player_sum with dealer_sum, and append return to *episode* accordingly
return *episode*

3 Experiment Results

3.1 Case 1

After training n*1000 steps, run 100 episodes with the current policy (no random initial action) and calculate the average return for those 100 episodes.
Provide the average returns for n =1,2,...,20.

Figure 1 shows the average returns under different n's.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
average return	0.91	0.94	-0.17	-0.36	0.15	0.05	0.84	0.59	0.58	0.31	-0.25	-0.49	-0.33	-0.48	0.14	-0.46	-0.61	0.03	0.33	-0.12

Figure 1: Returns for Case 1

Figure 2 shows the optimal policy under this training.

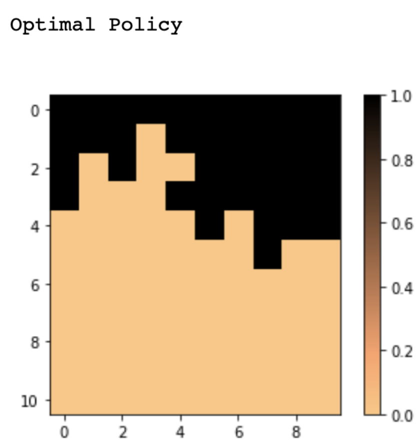

Figure 2: Optimal Policy for Case 1

Recall that 1 indicates *hit* and 0 indicates *stick*.

The horizontal direction shows the dealer's showing, where the index 0 here means dealer's showing is 2, the index 1 here means dealer's showing is 3, ..., the index 9 here means dealer's showing is 11.

The vertical direction shows the player's sum, where the index 0 here means player's sum is 12, the index 1 here means the player's sum is 13, ..., the index 10 here means the player's sum is 22.

For instance, the grid in the upper left corner indicates that under our policy, we choose *hit* when player's sum is 12 and dealer's showing is 2.

3.2 Case 2

- Train for 100,000 steps (instead of 20,000 steps)

- Every 1000 steps, evaluate the current policy by running 1000 episodes and averaging over the 1000 episodes

Figure 3 shows the average returns.

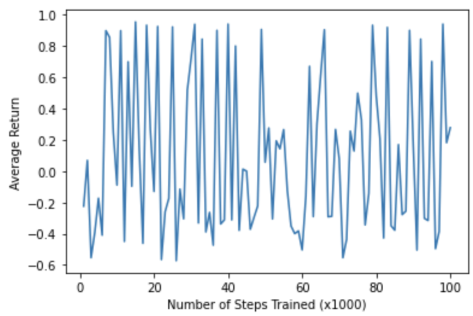

Figure 3: Returns for Case 2

Figure 4 shows the optimal policy under this training.

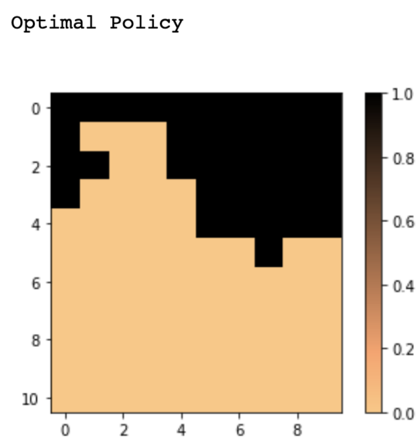

Figure 4: Optimal Policy for Case 2