BlackJack Simulation through lens of Monte Carlo

Soumiya Anantha Padmanaban 21200802

The project report is submitted to University College Dublin
in part fulfilment of the requirements for the degree of
MSc Data and Computational Science

School of Mathematics and Statistics
University College Dublin

Supervisor: Dr. Sarp Akcay

July 31, 2022

Contents

1 Abstract **2**

2 Acknowledgement **3**

3 Introduction **4**

4 Methodology **7**

4.0.1 Blackjack Game 7

4.0.2 Monte Carlo Prediction 8

4.0.3 Monte Carlo Control 11

5 Results **15**

6 Recommendations and Conclusions **17**

6.0.1 Future Work 17

6.0.2 Summary 17

Chapter 1

Abstract

Background: Players all over the world have attempted to beat the house in the game of Blackjack since its inception. American mathematician Edward O. Thorp was the first of many to discover that it was possible to beat the house by 'counting cards' (Thorp 2016) . Counting cards helps to determine whether the player or the dealer has the advantage in each round, by assigning point scores to each card which estimate its value. Since then, technology has gotten highly sophisticated. The higher computational powers and advancements in statistical theory has led to devising strategies to beat the house, despite the change in the rules. One such method is by reinforcement learning or Monte Carlo. The purpose of this study is to determine optimal methods in favour of the player.

Purpose: The main objective of this research is to design a simple online version of the Blackjack game. Then, proceed to predict the value of the current hand, based on the dealers show card. Lastly, monte carlo methods are employed to evaluate multiple games, in order to learn the optimal strategy for Blackjack.

Methodology: Using random and gym library in python arsenal the online version of the game is designed and the environment is set for prediction and control of the game respectively. The First Visit monte carlo algorithm will determine the value of the holding a certain hand (player) based on the dealer's showing card and a pre-determined policy. Discounted variable is factored in to take away the priority of an action as we move away from it. Then, Constant-α First visit algorithm is implemented to determine optimal

policy for blackjack (Watkins & Dayan 1992). It is a model free algorithm where agent explores the environment and outcomes are learnt directly based on actions taken. It employs an exploration-exploitation trade-off. Results of both the algorithms are understood by simulating thousands of games.

Findings and Conclusion: The results reveal that, the state values are higher, when the player sum is around 20 or 21. For usable ace optimal strategy was derived. It indicated that, in most of the cases, hit seems to be the strategy. Further policy iterations need to be adopted to improve the strategy for non-usable ace. Blackjack inadvertently favours the dealer. Hence, the optimal strategy results in negative reward. A combination of blackjack and betting policy could yield better results (Werthamer 2006).

Key words: Blackjack, environment, state, space, action, monte carlo, reinforcement learning

Chapter 2

Acknowledgement

Firstly, I would like to thank my module coordinator Dr Sarp Akcay for his guidance and support. I could not have completed this research without his unwavering support and motivation.

I would like to express my gratitude to my partner for motivating me and showing continuous faith in me. Finally, I thank University College Dublin for providing me the opportunity to work on this project.

Chapter 3

Introduction

"I wondered how my research into the mathematical theory of a game might change my life. In the abstract, life is a mixture of chance and choice. Chance can be thought of as the cards you are dealt in life. Choice is how you play them." — **Edward O. Thorp** (Thorp 2017)

Blackjack's major idea is the house having the edge in most of the cases. The discovery of probabilistic nature of the Black Jack by Edward O. Thorp combined with, the ability to model a card game on the computers for large samples based on Monte carlo method, there has been extensive exploration of the game in order to find an optimal solution to beat the house. One such method is card counting. Each card in the deck is assigned values: -1,0, 1 based on combination of cards dealt. The values are added together. Based on this value, the player decides to hit if the count is higher and stand if the count is lower.

The optimal strategy for the player to play a hand dealt based on the player's cards and dealer's showing card. Based on this strategy, the house edge reduces from 8% to 2.2%. However, this does not include the play – split or double. Unfortunately, the house still has an edge, since the player makes the first move. So, invariably the player goes bust first despite the chances of dealer going bust later.

This theory has also been worked upon where the game is modelled using softmax selection agent (Kakvi 2009). This method involves altering rewards and studying its impact on the policy change. It was noted that there has

been a higher impact on the policy change due to altering negative rewards rather than positive rewards. However, due to the house edge it was observed that an optimal policy alone is not enough to win the game.

Aims and Objectives:
The aim of this study is to perform exploratory analysis on various plays and find which one provides an edge to the player. Constant-α First visit algorithm helps access the hands dealt and helps determine what the next action is to be taken by the player.

The objectives of this research –

- To design an online version of the blackjack game.
- Dealer moves are based on simple logic – if hand dealt is greater than 17, then dealer stands.
- Predict the value of player's hand taking into consideration the dealer's open hand based on multiple simulations.
- Optimal strategy and current hand value prediction study is split based on Ace's ability to bust the game.

Limitations:
There are certain limitations to the study. Firstly, the decision to proceed with for the player takes into consideration only hit or stand. Double or split is not taken into consideration. This makes the study biased.The gym environment used in python involves usage of infinite deck, combined with inherent nature of the blackjack to favour the dealer, makes player lose (Lin B. & N. 2021).The initial policy adopted for the game is very rudimentary. Hence, not all states are considered.

Organisation of Study:
The rest of the research is organised as follows:

- Chapter 2 reviews the literature related to Blackjack game policy and monte carlo method-based modelling.
- Chapter 3 discusses the methodology of research used
- Chapter 4 discusses the results of the tests
- Chapter 5 concludes the study and provides further recommendations.

Chapter 4

Literature Review

Blackjack is a widely played casino game. Despite the origins of the game being disputed, the records date back to the 16th century (Wu 2018). Since then, the game and rules have evolved. The goal of the player is to attain a sum of dealt cards to be higher than that of a dealer, yet still under 21 with partially known cards. In the present day, the game can also be played on mobile devices, which was made possible through modern mathematics and technology. Over the years, players have come up with ways, to beat the house such as counting cards. One such way to do this is through the Monte-Carlo method.

The Monte Carlo method is a statistical sampling technique that has been applied extensively. The idea was conjured by Stanislaw Ulam, who collaborated with John Von Neuman to develop this method. His inspiration was the game of cards – Solitaire. He became interested in plotting the outcome of each of these games to observe their distribution and determine the probability of winning (Kenton 2021). The probability of different outcomes for a given problem is modelled using this method, where the prediction gets difficult due to the intervention of random variables. Its application lies in virtually every field.

The application of Monte Carlo for blackjack lies in the decision-making process of whether to hit or stand in a way that benefits the player. The player's hand and the dealer's showing card are taken as sample space. Prediction function is developed to evaluate the value of current hand taking into consideration the dealer's showing card and the total number of decks. The simulation is run multiple times to determine effective strategies.

Chapter 5

Methodology

The study is split into three parts –

1. Single experiment of online Blackjack game
2. Hit/Stand strategy prediction using First visit Monte Carlo
3. Optimal policy determination using Constant alpha First Visit

5.0.1 Blackjack Game

The game design begins with creating a deck of cards and relevant variables – suit, notation, and card value. The three variables are used with card display design for aesthetics using unicodes. Next, arrays are created to store the notations for each round of the cards for both dealer and the player. Cards are dealt using random.choice method in the python library random. The value of the cards dealt, are stored separately for dealer and player in order to determine the strategy. The cards are dealt twice. However, only the first card dealt to the dealer can be shown.

The value of the cards 2-10 is the same as the number on the card. The face cards – king, queen and jack has a value of 10 each. The ace card takes values of 11 or 1 depending on the number of aces dealt and is applicable to both dealer and player. Multiple strategies exist, but this study takes into consideration hit and stand. Based on this strategy, the dealer moves are automated based on a simple 'if' condition – if the score of the hands dealt exceeds 17, then the dealer chooses to stand. The goal of the blackjack game

is to not exceed 21 points.

Next, the player moves are taken in as the user input, provided the total value of the cards dealt is not already 21. The player is also given two choices – hit or stand. The player decides based on the total value of the hands dealt and the dealers showing hand. A validation for user choice is in place. Multiple scenarios arise –

1. If the dealer reaches 21 or closer before player then dealer wins.
2. If dealer or the player exceeds 21, they are busted.
3. If the dealer and player have the same score, then the game is a tie.
4. If the score of the player/dealer is less than 21, but greater than dealer/player respectively then player/dealer wins.
5. During initial hand if the player has a value of 21 already without deciding to hit or stand, the player automatically wins.
6. The above condition is applicable to dealer as well.

For each round, of the card dealt the variables – player_score and dealer_score, are updated on continuous basis. The cards dealt are appended to the arrays: player cards and dealer cards. In order to moderate the speed of the game, an input for the user is taken before proceeding to next step.

5.0.2 Monte Carlo Prediction

Reinforcement Algorithm

Black jack can be modelled using reinforcement learning as it can solve tasks without having any prior knowledge (De Granville 2005). Learning happens based on the feedback obtained from its own actions and experiences. The main goal of reinforcement learning is to maximise the total cumulative reward of the agent. Figure 1 (Sutton & Barto 2018) is a generic version of a RL algorithm with an action-reward feedback loop. RL algorithm has some basic elements within it:

1. Environment – Space where agent operates

In the game of blackjack, environment is the gaming environment. State is the cards dealt to players and dealers. Action taken is hit or stand. Based on the action taken, player could win or go bust based on which positive or negative points are rewarded respectively. The value is the rewards given for each game or iteration based on which player or dealer's edge is determined. Since, the major goal behind this study is find ways for the player to beat the house, we are going to take a monte carlo approach of RL. It is a model free approach that overcomes the difficulty in estimating strategies caused by models that are unknown (Dearden R. & S. 1999).

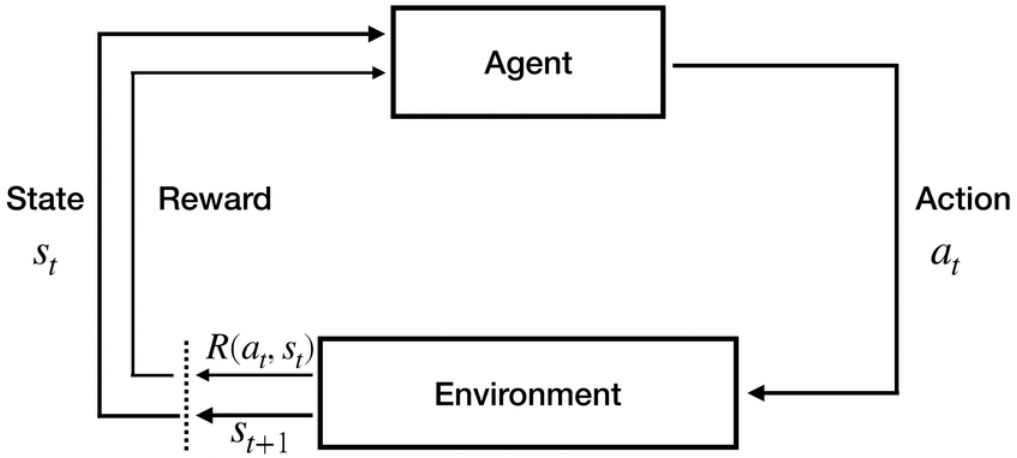


Figure 5.1: Reinforcement learning algorithm with an action-reward feedback loop

2. State – Current information about the agent's situation
3. Action – Step taken by agent to move from state to state
4. Reward – Feedback based on action taken
5. Value – Future reward based on actions taken in states visited.

First Visit Algorithm

The second part of the study estimate value function for a given policy. It is done by sampling a bunch of episodes based on the pre-determined policy. In our case, if the value exceeds 17 then the player opts to stand. Using the

Every occurrence of a state in an episode is coined as visit to that state-action pair. For our study we are opting the first-visit algorithm to calculate this value. Here, for each episode, despite multiple occurrences of a single state-action pair, only the first occurrence is taken into consideration. The algorithm is as follows:

First-visit MC prediction, for estimating $V \approx v_\pi$

Input: a policy π to be evaluated
Initialize:
    $V(s) \in \mathbb{R}$, arbitrarily, for all $s \in \mathcal{S}$
    $Returns(s) \leftarrow$ an empty list, for all $s \in \mathcal{S}$

Loop forever (for each episode):
    Generate an episode following π: $S_0, A_0, R_1, S_1, A_1, R_2, \ldots, S_{T-1}, A_{T-1}, R_T$
    $G \leftarrow 0$
    Loop for each step of episode, $t = T-1, T-2, \ldots, 0$:
        $G \leftarrow \gamma G + R_{t+1}$
        Unless S_t appears in $S_0, S_1, \ldots, S_{t-1}$:
            Append G to $Returns(S_t)$
            $V(S_t) \leftarrow$ average($Returns(S_t)$)

Figure 5.2: First-Visit Algorithm for Monte Carlo Prediction

episodes sampled, action-value function is estimated and tabulated called a Q-table.

1. Initially policy pi is decided.
2. Empty dictionaries are created for V(St): State-action pair values table, G: First-visit counts table and Returns(St): Sum of rewards table after first visit of each state-action pair.
3. Play a game of Blackjack using our current policy for multiple games
4. Loop through each turn in the game and check if the current State/Action pair has been seen in that game yet
5. If this is the first time we have seen that pair we increase the count for that pair and add the discounted rewards of each step from that turn onwards to our Returns(St) dictionary

Figure 3 and 4 demonstrates the state=action pair value calculation. In round 1, player has a total of 15 and dealer has a total of 10. There are no aces. Player decides to hit. Now, the updated player value in the new state is 26. Dealer wins and players loses. A negative point is awarded.

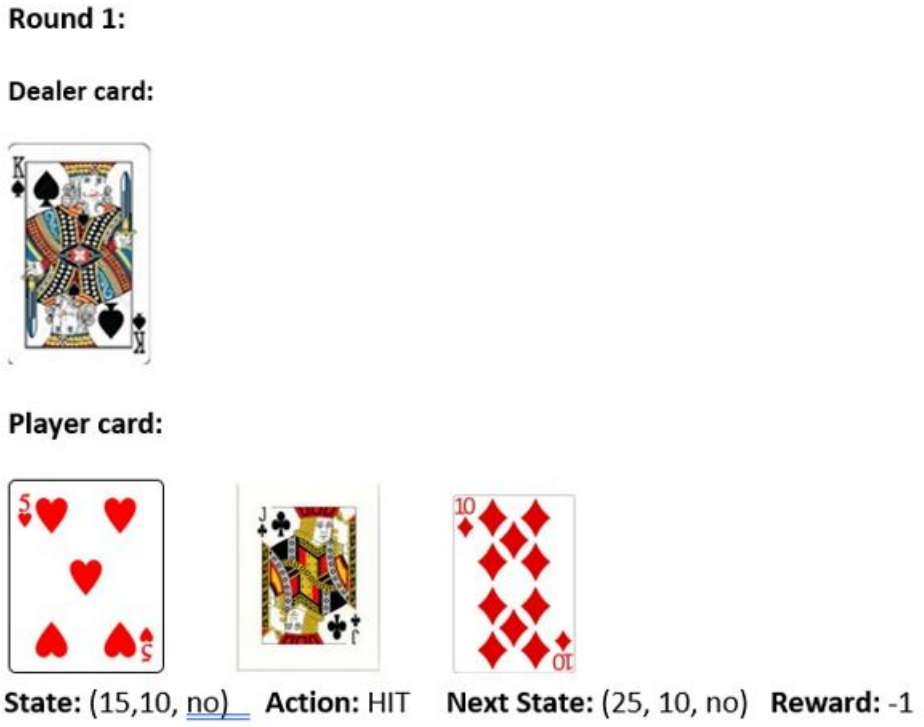


Figure 5.3: State-Action Calculation and Reward for first round of hands dealt. Player loses and Dealer wins

6. Finally update the V(St) values with the average of the returns and amount of each State/Action pair.

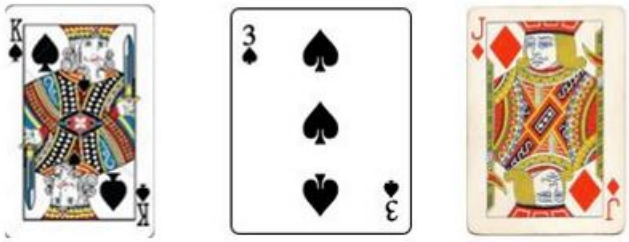
5.0.3 Monte Carlo Control

Each state is a $3'-'$tuple with players sum, dealer's sum and if there exists a usable ace. An ace is considered usable when the existence of ace does not lead to a bust. Discounted rewards are used to prioritise immediate reward over potential future rewards. Hence, cumulative discounted reward is considered while calculating action values. The discount factor is a constant

Now, in round 2, player has a total of 7 and dealer has a total of 10. Player decides to hit. Now, the updated player value in the new state is 17. After this, the player decides to stand. Next, the dealer reveals the other card. There is a total of 13 now for the dealer. Dealer decides to hit. The total for dealer is 23 hence he loses. In this first two states there is no reward yet as the game is not fully done. In the third row, a positive point is rewarded as we have an outcome.

Round 2 :

Dealer card:



Player card:

STATE	ACTION	NEXT STATE	REWARD
7, 10, no	HIT	17,10, no	0
17,10, no	HIT	17,13, no	0
17,13, no	HIT	17,23, no	1

Figure 5.4: State-Action Calculation and Reward for second round of hands dealt. Player wins and Dealer loses

number that is multiplied with reward. The power to which we multiply our discount factor is increased each time. This gives more priority to the immediate actions and less priority as we get further away from the action taken.Value of the discount factor lies between 0 and 1, depending on the task to be performed. To start with higher discount is taken in and then is lowered as the loop goes on so that immediate reward is prioritised.

Using $V(S_t)$ or Q table obtained above, an optimal policy to beat the house can be determined (Sutton & Barto 2018)(Reilly 2012). A better policy (need not be optimal) can be obtained by maximising the Q table. A policy constructed in such a manner is said to be greedy with respect to the Q table. The action selected is referred to as greedy action.

In order to arrive at optimal solution along with such an action we factor in epsilon ϵ which is a small positive number between zero and one. Hence, the policy used is Epsilon-greedy. On alternating between Q-table construction and determining ϵ-greedy policy, we arrive at an optimal policy Π^*. This is called monte carlo control method.

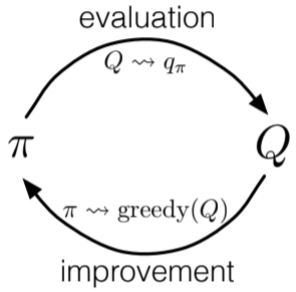


Figure 5.5: Epislon-Greedy for Optimal Policy

The above two steps are called - policy evaluation (Q table) and policy improvement (ϵ-greedy) shown in figure 5. Variation to this algorithm exists which updates the policy after every episode instead of updating after the values in Q table are fully converged is Constant- α. The main advantage is that learning happens from incomplete sequences.

1. Algorithm makes use of number of episodes, α and ϵ.

Algorithm 11: 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

Figure 5.6: First Visit Constant-alpha Monte Carlo Algorithm

2. A temporary policy is set with equal probability to select either action - [0.5,0.5]

3. Current best policy for the current state Q[s] and best action argmax(Q[s] based on this is determined.

4. Probability of selecting the best action is: 1- ϵ + the temporary policy value.

   In the beginning, epsilon will be large indicating the best action probability is 0.5. As the game goes on, epsilon will decrease which in turn increases the probability of best action. This makes it an epsilon greedy strategy as we are alternating between random action and best action.

5. As we go through, state, action and reward of each episode is recorded and passed to the update function.

6. Next Q Value is updated using the below function -

   Q[s][a] = Q[s][a] + α *(G - Q[s][a])

   Original Q value is added to the update. The update is made up of the cumulative reward of the episode (G) and old Q value subtracted from it. This is then all multiplied by alpha.This results in optimal policy.

Chapter 6

Results

OpenAI's gym environment (Blackjack – v1) is used to simulate the game of black jack. The First-Visit Monte Carlo Prediction algorithm contains state which is a $3-$tuple with players sum, dealer's sum and if there exists a usable ace. An ace is considered usable when the existence of ace does not lead to a bust. Actions is 0 (Stick) or 1 (Hit). Each game of blackjack is an episode with rewards -1, 0, 1 (lose, draw, win respectively).

Based on the updated Q-table, 3D plot of state value distribution is generated (figure 5.1). In both the cases, the state values are higher, when the player sum is around 20 or 21, and dealer sum is around 10. This is obvious because we are most likely to win the game.

The optimal policy obtained from the Constant-alpha First-Visit Monte carlo algorithm is plotted as shown in figure 5.2. The grey areas are hit and highlighted areas are stick. Despite the scope improvement for No usable Ace, we can see that for Usable ace the strategy is very good. In most of the cases, hit seems to be the strategy.

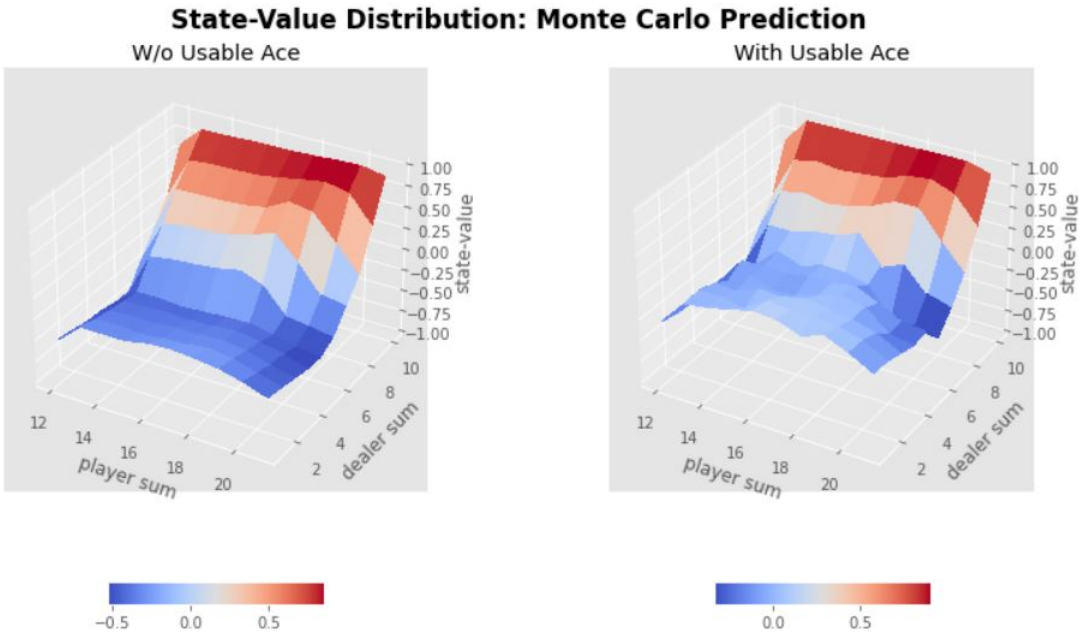


Figure 6.1: Monte Carlo Prediction's State-value distribution

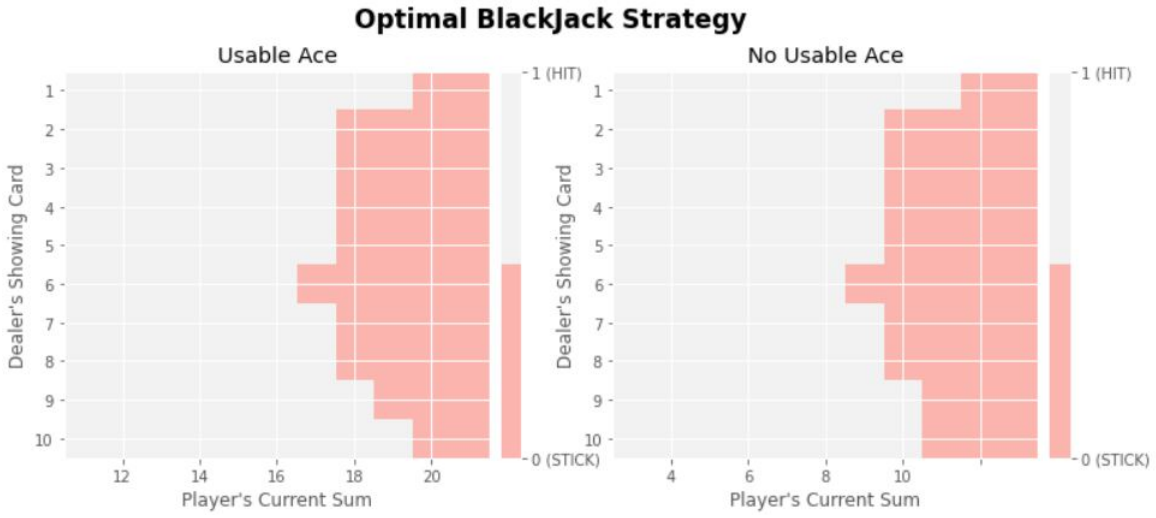


Figure 6.2: Monte Carlo Control's Optimal Strategy

Chapter 7

Recommendations and Conclusions

7.0.1 Future Work

While evaluating optimal policy, a successful agent cannot act greedily at every step; instead, it must continue to refine the estimated return for all state-action pairs. At the same time, the agent must maintain a certain level of greedy actions to maintain the goal of maximizing return as quickly as possible. The need to balance these requirements is the Exploration-Exploitation Dilemma (Wyatt 1998). Agent begins its interaction with the environment by opting for exploration and trying various strategies for maximizing return. At later stages,exploitation is opted, so that policy gradually becomes more-greedy with respect to the action-value function estimate.The current blackjack environment uses infinite deck. Since, blackjack inadvertently favours the dealer, optimal strategy results in negative rewards. As next step,the results for finite number of decks and other actions such as double or split can be considered.

7.0.2 Summary

After performing MC prediction and control over 1000000 simulations, the algorithm converge to optimal strategy. Increasing the number of iterations, results in similar state-value distribution 3D plots for usable and non-usable cases (Reilly, 2012). This strategy alone is not enough to beat the house. A combination of blackjack and betting policy could yield better results.

Bibliography

De Granville, C. (2005), 'Applying reinforcement learning to blackjack using q-learning', *University of Oklahoma,Citeseer* .

Dearden R., F. N. & S., R. (1999), 'Bayesian q-learning.'.

Kakvi, S. (2009), 'Reinforcement learning for blackjack. in: Natkin, s., dupire, j. (eds) entertainment computing', *ICEC 2009. Lecture Notes in Computer Science, vol 5709. Springer, Berlin, Heidelberg* .

Kenton, W. (2021), 'What is a monte carlo simulation?', *Investopedia* .

Lin B., L. E. & N., L. (2021), 'Optimizing blackjack strategy with reinforcement learning.'.

Reilly, B. (2012), 'An mdp blackjack agent.'.

Sutton, R. & Barto, A. (2018), 'Reinforcement learning : an introduction.', *Cambridge,Lodon: The Mit Press.* .

Thorp, E. O. (2016), 'Beat the dealer: a winning strategy for the game of twenty one', *New York: Vintage* .

Thorp, E. O. (2017), 'A man for all markets: beating the odds, from las vegas to wall street.', *London,England: Oneworld Publications* .

Watkins, C. & Dayan, P. (1992), 'Q-learning. machine learning.', *Machine learning,Springer.* .

Werthamer, N. (2006), 'Optimal betting in casino blackjack ii: Back-counting. international gambling studies', *6(2),pp.111-122* .

Wu, A. (2018), 'Playing blackjack with deep q-learning'.

Wyatt, J. (1998), 'Exploration and inference in learning from reinforcement.'.