Reinforcement Learning in Blackjack: Q-Learning Strategy Experiments

Suraj Varma
MAI, Faculty of Computer Science
Technical University of Applied Science Wurzburg-Schweinfurt
Würzburg, Germany
surajkumardineshbhai.varma@study.thws.de

***Abstract*—Blackjack, a popular casino game, challenges players to achieve a hand value less than or equal to 21 while surpassing the dealer's hand value. This study investigates the use of Reinforcement Learning (RL) to enhance Blackjack strategies, focusing on the Q-learning algorithm. By training an RL agent in a simulated Blackjack environment, we explore the effectiveness of various strategies, including Basic Strategy and Complete Point Count, along with additional rule variations. The research demonstrates that RL can significantly improve decision-making in Blackjack, revealing optimal strategies through extensive game simulations. The findings highlight the potential of RL for advancing strategic gameplay and decision-making in complex scenarios.**

***Index Terms*—Blackjack,Decision-making; Reinforcement learning, Q- learning algorithm, Basic strategy, Complete Point Count system**

I. Introduction

Reinforcement Learning (RL) is a type of artificial intelligence. It helps machines learn from their experiences. One exciting use of RL is in playing casino games, like Blackjack. Blackjack is interesting because it mixes skill and luck. By studying how RL can play Blackjack, we can learn more about making smart decisions. This paper will look at how RL can help make better strategies for playing Blackjack. It will also look at using point-counting and changing the rules of the game. This research can help us understand RL better and use it in other areas too.

A. *Blackjack Game Overview and challenges*

Blackjack, also known as 21, is a casino card game where the aim is to get a hand total as close to 21 as possible without exceeding it. Number cards are worth their face value, face cards are worth 10, and Aces can be 1 or 11. Players start with two cards and can 'hit' (take another card), 'stand' (keep current cards), 'double down' (double their bet for one more card), or 'split' (divide two cards of the same value into two hands). Dealers usually hit until they reach 17 or higher. There are soft and hard hands: a soft hand includes an Ace counted as 11 (e.g., Ace-6), while a hard hand either lacks an Ace or has one counted as 1 (e.g., 10-8 or Ace-6-10). Soft hands are more flexible, as the Ace's value can change without busting.

Blackjack's challenges include choosing the best actions based on the hand and dealer's card, managing uncertainty, and adapting to rule variations. Reinforcement Learning (RL) can aid in optimizing strategies despite these challenges.

B. *Reinforcement Learning and Suitable Methods*

Reinforcement Learning (RL) helps models learn to make better decisions by interacting with an environment and receiving feedback. Applying RL to Blackjack involves a model that learns to play better by trying different actions and seeing the outcomes. In this system, the agent (player) makes choices like hit or stand based on their hand and the dealer's card, receiving rewards for winning, losing, or drawing. Over time, the agent develops a policy to maximize these rewards. To enhance this process, point-counting systems like the Hi-Lo strategy can be used. These systems assign values to cards, helping track the proportion of high to low cards left in the deck, which improves decision-making. Additionally, varying game rules, such as the number of decks or the dealer's behavior on a soft 17, tests the RL model's adaptability to different conditions.

This paper explores the application of Reinforcement Learning techniques to Blackjack, focusing on strategies informed by point-counting methods and adapting to various rule variations common in both virtual and physical casino settings. It discusses challenges posed by different Blackjack rules, and presents experimental results demonstrating the effectiveness of RL in learning optimal strategies. Through a detailed analysis, this study aims to contribute to the broader understanding of RL applications in strategic decision-making within the domain of card games.

Also used most simple formula of exploration decay rate which controls how the exploration probability ϵ decreases over time, balancing exploration and exploitation. It is updated as $\epsilon_{t+1} = \epsilon_t \times \text{decay\_rate}$.

II. METHODOLOGY

A. *Blackjack Environment for Q-learning*

The `BlackjackEnvironment` class is tailored for Q-learning experiments in Blackjack. This environment serves as a controlled setting for training reinforcement learning agents, excluding actual betting or real gameplay scenarios. Key features include the simulation of game states, player actions, and rewards based on the agent's strategy.

The environment utilizes a standard 52-card deck, shuffled at the beginning of each game. Cards are valued as follows: numerical cards from 2 to 10 hold their face values, face cards (Jack, Queen, King) are worth 10, and Aces can be worth either 1 or 11. The deck is used until it is exhausted.

- `deal_card`: Draws a card from the deck.
- `hand_value`: Computes the total value of a player's hand, accounting for the flexibility of Aces.
- `step`: Executes player actions (hit, stand, double down), updates the game state, and adjusts Q-values based on rewards.
- `complete_point_count_action`: Determines the optimal action using the Hi-Lo card counting system.

There is also an environment for the Hi-Lo card counting system, as described in the book *Beat the Dealer* by Edward O. Thorp, implemented to aid the agent's decision-making process. In this system, cards are categorized as high or low to keep a running count that influences strategy:

- High cards (10s, Jacks, Queens, Kings, Aces) are assigned a negative count.
- Low cards (2s, 3s, 4s, 5s, 6s) are assigned a positive count.
- The running count is updated as cards are dealt, and this value is used to adjust strategies dynamically.

This system helps the agent make informed decisions about actions such as hitting, standing, or doubling down, based on the composition of the remaining deck.

The `BlackjackEnvironment` class supports various gameplay strategies and adjustments, including handling a soft 17 rule for the dealer and allowing the player to double down. It provides a robust framework for experimenting with and optimizing reinforcement learning strategies in Blackjack.

In the experimental setup, the environment is adjusted to include specific rules like the soft 17 condition for the dealer and the double down option for the player. These adjustments ensure that the environment aligns with the requirements of different Q-learning experiments.

B. *Q-Learning Workflow and Interaction with Environment*

Q-Learning is a model-free reinforcement learning algorithm designed to learn the optimal policy for decision-making problems. It estimates the value of action-state pairs and updates these estimates based on the agent's interactions with the environment. The objective is to find a policy that maximizes the cumulative reward over time.

In Q-Learning, the *Q-value* or action-value function $Q(s, a)$ represents the expected utility of taking action a in state s and following the optimal policy thereafter. The core of Q-Learning is the *Bellman equation*, which updates the Q-values iteratively:[2]

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left[r + \gamma \max_{a'} Q(s',a') - Q(s,a) \right]$$

where:

- α is the learning rate, which controls how much new information overrides old information.
- γ is the discount factor, which determines the importance of future rewards.
- r is the reward received after taking action a in state s.
- s' is the new state after action a is executed.

C. *Workflow in the Blackjack Environment*

The Q-Learning algorithm operates in the Blackjack environment as follows:

1) **Initialization:** Q-values for all state-action pairs are initialized to zero, and parameters such as the learning rate (α), discount factor (γ), and exploration rate (ϵ) are set.
2) **Exploration and Exploitation:** The agent uses an ϵ-greedy policy to decide whether to explore (choose a random action) or exploit (choose the best-known action). The exploration rate ϵ decays over time to decrease exploration as the agent gains more knowledge about the environment.
3) **Interaction with the Environment:** The agent selects an action based on its policy. The action is executed in the Blackjack environment, resulting in a new state and a reward. The Q-value for the state-action pair is updated using the *Bellman equation*.
4) **Updating Q-values:** The Q-values are updated after each action based on the observed reward and the estimated value of the next state. This process is repeated across numerous episodes to improve the Q-values and converge towards the optimal policy.
5) **Learning and Policy Improvement:** Over many episodes, the Q-values become more accurate, leading to better decision-making and an improved policy.

In the Blackjack environment, the Q-Learning algorithm learns to make optimal decisions such as hit, stand, or double down based on the game state, which includes hand values and the dealer's upcard. The addition of double down and the handling of soft hands introduce complexity to the learning process, providing a more robust training scenario for the agent.

D. *Q-Value Updation*

The Q-value table is central to the Q-Learning algorithm and is updated iteratively as the agent interacts with the environment. The table maintains estimates of the expected utility of taking various actions from different states, which guides the agent's decision-making process.

1) Update Process:

The update of the Q-value table involves the following steps:

1) **Action Selection:** The agent selects an action based on its current policy, which is determined by the Q-values for the given state.

2) **Execute Action:** The selected action is executed, resulting in a transition to a new state and receiving a reward.
3) **Q-Value Update:** The Q-value for the state-action pair is updated based on the observed reward and the estimated future rewards. This update is crucial for refining the agent's policy. The specific update formula used for adjusting the Q-values is illustrated in Figure 1.[2]
4) **Repeat:** The process of action selection, execution, and Q-value updating continues across multiple episodes, allowing the agent to improve its policy over time through repeated interactions with the environment.

The iterative updates to the Q-value table help the agent learn an optimal strategy by continuously adjusting its estimates of the value of different actions in various states. By doing so, the agent increasingly favors actions that lead to higher cumulative rewards.

```
Initialize $Q(s, a), \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$, arbitrarily, and $Q(terminal\text{-}state, \cdot) = 0$
Repeat (for each episode):
  Initialize $S$
  Repeat (for each step of episode):
    Choose $A$ from $S$ using policy derived from $Q$ (e.g., $\epsilon$-greedy)
    Take action $A$, observe $R$, $S'$
    $Q(S, A) \leftarrow Q(S, A) + \alpha\big[R + \gamma \max_a Q(S', a) - Q(S, A)\big]$
    $S \leftarrow S'$;
  until $S$ is terminal
```

Fig. 1: *Q-learning algorithm is shown in procedural form*

III. Experiments

A. *Random Action Selection Comparison*

In the initial experiment, we employed random action selection by the agent within a standard Blackjack environment, featuring only the actions "hit" and "stand" without the soft hand 17 stand condition. The agent's actions were purely random, without reinforcement learning. Subsequently, we incorporated Q-learning with decaying epsilon, utilizing the random action selection as the exploration strategy. A comparative graph Figure 2 was plotted to analyze the differences, demonstrating the impact of reinforcement learning on the agent's performance.

B. *Basic Strategy Comparison*

An experiment using the basic strategy outlined in the book "Beat the Dealer" instead of random action selection was conducted. The agent's actions were based on the basic strategy rules. The performance of this basic strategy without Q-learning was compared, with Q-learning, and with SARSA. Comparative graphs of the win performance were plotted for this analysis. The results indicated that Q-learning outperformed SARSA in implementing the basic strategies.

C. *Complete Point Count Strategy*

For the next set of experiments, we added variations to the Blackjack environment by including the "double down" action and the "soft hand 17 stand" condition, where the dealer must stand on a soft 17. The complete point count technique from the book "Beat the Dealer" in this modified environment was employed. The agent's actions were guided by the complete point count strategy.

The experiments with these variations were repeated, comparing the performance in environments with different numbers of decks, such as 1, 2, and 3decks. The environment still only allowed "hit" and "stand" actions.

D. *Summary of counducted experiments*

The experiments encompassed several configurations:

- Random Action Selection Comparison without Q-learning vs. with Q-learning.
- Basic Strategy Comparison without Q-learning vs. with Q-learning vs. SARSA.
- Complete Point Count Strategy in a standard environment.
- Basic Strategy with variations, including Double Down and Soft hand 17.
- Complete Point Count Strategy with variations, including Double Down and Soft hand 17.
- Complete Point Count Strategy with different numbers of decks.

These experiments provided comprehensive insights into the effectiveness of various strategies and reinforcement learning techniques in different Blackjack environments. Experiments were chose to analyse the influence and potential of reinforcement learning in optimizing decision-making processes in Blackjack and draw conclusion different different learning algorithms performs in different environment.

IV. Results and Insights

A. *Performance with Random Action Selection*

In the first experiment, the Blackjack agent used random actions and achieved a win percentage of 27.94% after playing 10,000 games [Figure 2]. When Q-learning was introduced, the agent started with random actions and had an epsilon value of 1.0, which decreased to 0.1 within just 2,000 episodes, representing a 50% reduction in exploration. After about 6,000 episodes, epsilon stabilized around 10%, and the win percentage increased to 31.70%. This improvement indicates that Q-learning allowed the agent to focus more on exploiting its learned strategies rather than exploring new ones, leading to better overall performance. Figure 2 indicates that Q-learning significantly outperforms Random Action Selection. Q-learning demonstrates higher win rates and cumulative rewards, showcasing the benefits of learning-based strategies over random decision-making.

$$\epsilon_{t+1} = \epsilon_t \times \text{decay\_rate}$$

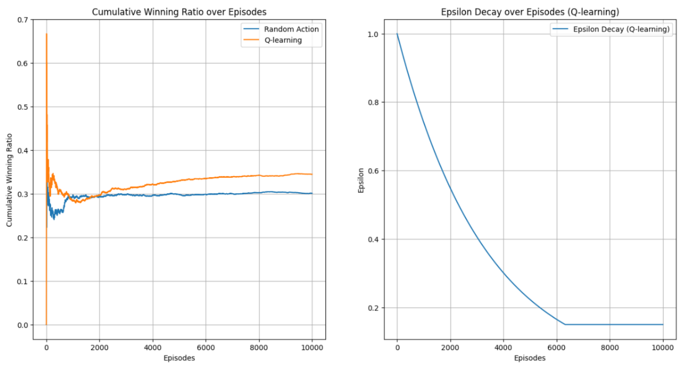

Fig. 2: *Comparison of Performance Between Random Action Selection and Q-learning over 10,000 Episodes with a Specific Decay Rate*

B. *Performance with Basic Strategy*

In the first experiment, the Blackjack agent was tested in a standard environment with only two actions: hit and stand. The agent used a basic strategy, which involved predefined rules for taking actions, resulting in a win percentage of 43.05% after playing 10,000 games.[Figure 3] When Q-learning was introduced, the agent initially explored using given basic strategies actions, with epsilon starting at 1.0 and decreasing to 0.1 with a decay rate of 0.99999. This approach allowed the agent to leverage learned experiences from previous episodes and utilize learned Q-values for most of the games. However, the improvement was marginal, with a win percentage of 43.43%.

In the second experiment, we compared the performance of Q-learning with that of the SARSA algorithm in the same standard environment. The SARSA algorithm achieved a win accuracy of only 37.29%, which is notably less than the 43.43% win rate achieved by Q-learning.[Figure 4] In the third experiment, we modified the basic strategy environment

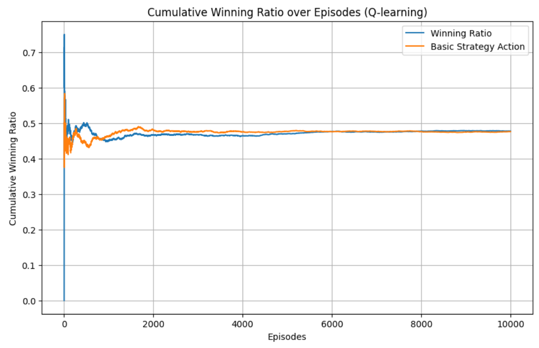

Fig. 3: *Comparison of Basic Strategy Action Selection and Q-learning over 10,000 Episodes with a Specific Decay Rate*

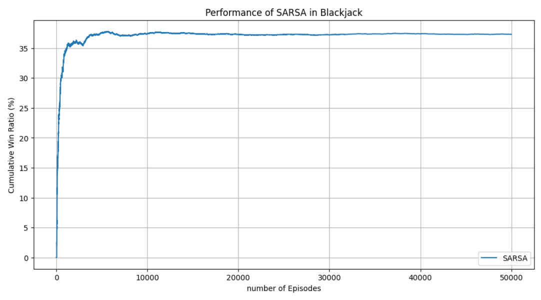

Fig. 4: *Sarsa win percent under basic strategy*

by adding a ”double down” action and introducing the ”soft hand 17” rule for the dealer. With these changes, the win percentage was 41.22%. During this experiment, epsilon was decayed with a rate of 0.9997, and it was observed that epsilon decreased rapidly within the first 15% of episodes, leading to more exploitation rather than exploration for the remaining 85% of episodes.[Figure 5] The total number of episodes for this experiment was 100,000. The winning rate remained relatively constant as training progressed.

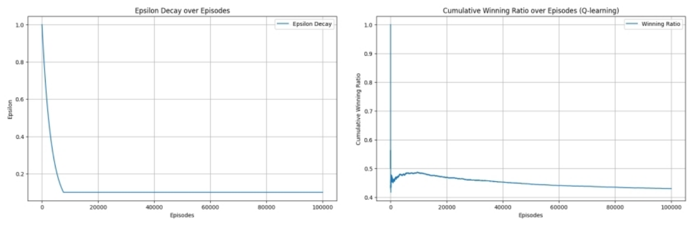

Fig. 5: *The image displays 100,000 runs with epsilon decay for the first around 10,000 runs and 10% exploration for the next 90,000 runs, highlighting win percentages with basic strategy and variations.*

The results reveal that Q-learning consistently outperforms SARSA. Q-learning’s adaptability and learning capabilities lead to superior performance in terms of win rates and rewards compared to SARSA. The 3D plotting of Q-values for actions ”hit” and ”stand” in the soft and hard hand scenarios showed significant changes and improvements compared to the ”stand” action. It appears that the agent had more opportunities to ”hit” rather than ”stand,” which likely contributed to a higher win rate, as the dealer is required to stand on a total of 17 or higher.[Figure 6]

C. *Performance with Complete Point Count Strategy*

In the first experiment with the complete point count system, the Blackjack agent operated in a standard environment with only two actions: hit and stand. The agent employed Q-learning with epsilon starting at 1.0 and decaying to 0.1 over 10,000 episodes. Initially, the agent used the complete point count system for action selection. As epsilon decreased, the exploration ratio dropped to 50%, meaning that for the remaining episodes, over 40% of the actions were based on

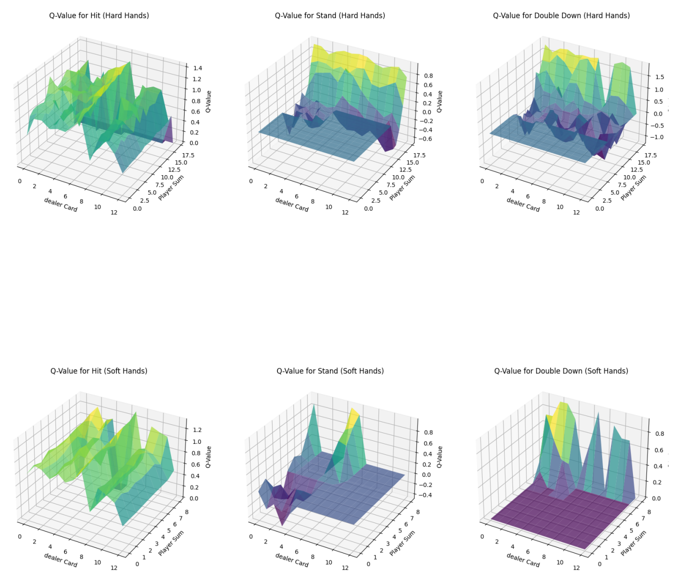

Fig. 6: *The 3D plots of updated Q-values for "hit" vs. "stand"*

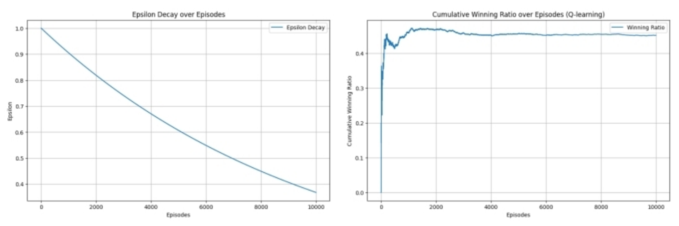

Fig. 7: *The plot shows a Blackjack agent's win rate of 40.25% with Q-learning and epsilon decaying from 1.0 to 0.1 over 10,000 episodes, indicating room for improvement.*

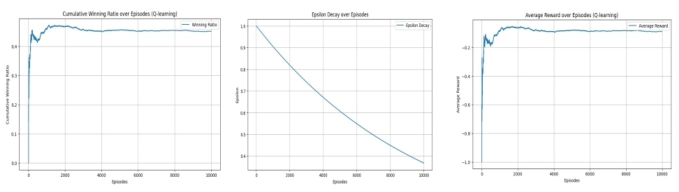

Fig. 8: *The plot shows a 40.95% win rate with "double down," "soft hand 17," and Hi-Lo rules, using an epsilon decay of 0.9997 over 10,000 episodes. Epsilon decreased steadily, and win rates remained stable.*

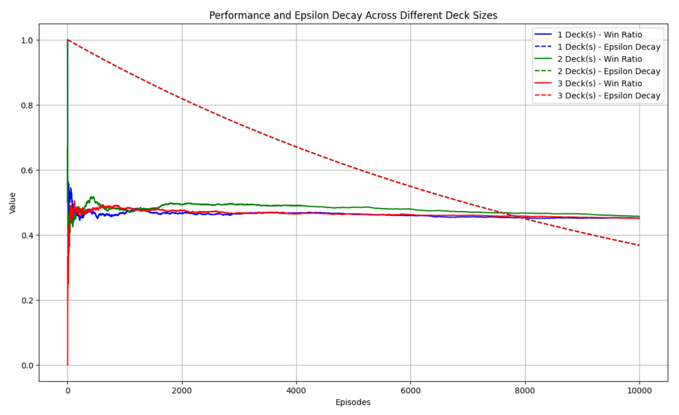

Fig. 9: *The plot illustrates win percentages across different numbers of decks, showing a slight edge with fewer decks, but could be affected by episode count and decay rate.*

Q-values rather than random exploration. The resulting win rate was 40.25%.[Figure 7] Although this win rate was not as high as that achieved with the basic strategy, it indicates that there is room for improvement through hyperparameter tuning.

In the second experiment, additional features were introduced: the "double down" action and the "soft hand 17" rule for the dealer. With these changes, the win percentage was 40.95%.[Figure 8] The epsilon decay rate was set at 0.9997. Analysis of the results showed that epsilon decreased monotonically over approximately 50% of the episodes, and for the remaining episodes, the agent engaged in more exploitation than exploration. The win rate remained relatively constant as training progressed over a total of 10,000 episodes.

For the third experiment, the standard environment was modified to include multiple decks to observe performance differences. The results were as follows: with 1 deck, the win percentage was 40.61%; with 2 decks, it was 41.18%; and with 3 decks, it was 40.65%.[Figure 9] These results indicate that the win percentage varied slightly with the number of decks used. Although the differences were not highly significant, the trend suggests that fewer decks yield sligtly higher win ratio, likely due to easier card counting and tracking. The relatively modest results may be attributed to the number of episodes and the need for further hyperparameter tuning. The decay rate was also applied in this experiment, which could have affected the performance.

D. *Overall Results*

Significant differences were observed in strategy performance. Basic strategies showed minimal variance compared to the complete count system, suggesting effective learning by the agent. Although the epsilon decay rate in the complete count system affected results, the impact was not substantial. Performance decreased in basic strategies due to variations like doubling down and the dealer's soft 17 rule.

The primary finding is that all agents, regardless of strategy, are losing money, indicating no winning strategy was achieved. However, Q-learning agents outperformed the Random Player, reflecting better gameplay knowledge. Despite this, the Q-learning agent did not reach the optimal policy.

V. Conceptual Discussion

In Blackjack, the game state is defined by your hand value and the dealer's face-up card. Hand values can be either "soft" or "hard." A **soft hand** includes an Ace counted as 11 (e.g., Ace and 6, totaling 17), while a **hard hand** doesn't include this flexibility (e.g., 10 and 6, totaling 16). The state space is determined by combining these hand values with the dealer's card. There are 18 possible hand values (from 4 to 21 for hard hands and 13 to 21 for soft hands) and 10 possible dealer cards (1 through 10). Therefore, the state space size is approximately $18 \times 10 = 180$ states. Each state involves 2 actions: Hit (request another card) or Stand (end your turn), making the total state-action space $180 \times 2 = 360$ pairs. Estimating the value of actions can be tricky due to the variability in game outcomes and infrequent exploration of some states. Effective strategies involve exploring various scenarios through simulations and using algorithms to refine the understanding of optimal actions, ensuring more stable and accurate estimates.

In blackjack, rule changes like the dealer hitting on a soft 17 or restrictions on doubling down impact the game's dynamics but don't directly alter the number of hits a player can take. When the dealer has a soft 17 (e.g., an Ace and a 6), requiring them to hit can significantly increase the house edge. For instance, if the dealer draws a 5, their total becomes 12. Drawing a 6 on the next hit results in a total of 18. This rule change means the dealer is more likely to end up with a stronger hand, which makes the game tougher for players. While your ability to hit remains unchanged, you face a higher chance of losing due to the dealer's improved hand.

If the rule limits doubling down (like allowing it only on a total of 11), you miss out on opportunities to double your bet in favorable situations. For example, if you have a total of 11 and double down, you could end up with a total of 21, a strong position. If doubling down is restricted to only totals of 11, you lose this strategic option for totals of 10. While this doesn't change how many hits you can take, it limits your chances to maximize winnings.

SARSA (State-Action-Reward-State-Action) considers the current state-action pair and the action to be taken next, updating its estimates accordingly.[4] For example, if you hit on 15 and then stand on 18, SARSA updates the value of hitting based on how standing with 18 turned out. It learns and adjusts as you play, relying on the actions you actually take and their results. This method is stable but can be slower to find the optimal strategy. Q-Learning, on the other hand, evaluates the best possible outcome for each action, regardless of the actions taken during play. By trying all actions in all states repeatedly, it learns which are best overall, judged by long-term discounted reward.[3] If you hit on 15 and end up with 18, Q-Learning updates the value of hitting by considering the best possible future rewards. It aims to learn faster by focusing on the best possible outcomes rather than just the actual experiences. This approach can find better strategies more quickly but might be less stable if not managed properly.

VI. Conclusion

This study demonstrates that Q-learning significantly enhances Blackjack strategies, achieving improved win rates compared to random action selection. The Q-learning agent refined its decision-making process effectively, surpassing random. However, the complete point count strategy, while promising, requires further optimization. The Study experiments conclude that Q-learning has potential to learn form the given inructions but also highlights the need for additional research to fully harness its benefits.

Further research is necessary on advance Blackjack strategies to beat the dealer consistently. Key areas include fine-tuning Q-learning's hyperparameters, integrating advanced strategies, and evaluating the impact of rule variations. Additionally, exploring Q-learning's performance with larger decks and comparing it with Deep Q-Learning— which utilizes neural networks for approximating Q-values—could provide valuable insights. Investigation on how Q-learning performs in multi-deck environments will also enhance understanding of its adaptability and effectiveness. This comprehensive approaches will help refine strategies, improve decision-making, and potentially lead to a more consistent winning edge against the dealer.

References

[1] Thorp, E. O. (1966). Beat the Dealer: A Winning Strategy for the Game of Twenty-One

[2] Richard S. Sutton and Andrew G. Barto. Reinforcement Learning: An Introduction. MIT Press, Cambridge, MA, 2 edition, 2018.

[3] Watkins, C.J.C.H., & Dayan, P. (1992). Q-learning. *Machine Learning*, 8(3), 279-292.

[4] Deep-sarsa: a reinforcement learning algorithm for autonomous navigation M. Andrecut,M. K. Ali,University of Lethbridge,01 Dec 2001