Reinforcement Learning - BlackJack Player

Anagha Manikathuparambil Baby
(Student, MAI program)
Technische Hochschule Würzburg-Schweinfurt
Würzburg, Germany
anagha.manikathuparambilbaby@study.thws.de

***Abstract*—This study explores the application of reinforcement learning (RL) techniques to optimize gameplay strategies in blackjack, a popular casino game requiring players to balance their hand values below 21 while outscoring the dealer. Q-learning, SARSA, and Monte Carlo Control algorithms train the agent to make dynamic and informed decisions through iterative self-play. The performance of the basic strategy and complete point count system within a simulated blackjack environment is thoroughly evaluated using the reinforcement learning methods. The results show that the Monte-Carlo Control method performs the best in both the basic strategy and complete point count system scenarios, achieving the highest win rate of 43.83%, the loss rate of 48.68% and the tie rate of 7.49%. This study highlights the potential for reinforcement learning to address complex decision-making challenges and provides a foundation for advancing AI-driven strategies in Blackjack and similar decision-intensive games.**

***Index Terms*—Blackjack, Decision Making, Reinforcement Learning, Machine Learning, Q-learning, SARSA, Monte Carlo Control, Basic Strat egy, Complete Point Count System, Soft hand and Hard hand rules,Epsilon greedy scenarios.**

I. Introduction

Black Jack is a card game consisting of more than one number of players and a single dealer. The players aim to have a hand value closest to 21 without exceeding it. During each turn, the player can choose to stand (stop receiving additional cards and choose to outplay the dealer) or hit(receive an extra card). Additionally, there are other actions apart from the basic ones: double down (receive exactly one more card and choose to stand) or split down (if the player has cards worth the same value, then the player can split cards and play each hand as a separate game). If the player's hand is more than 21, this is referred to as a 'bust' and the player loses the game. If the player opts to "stand" before busting, the dealer's hand is dealt out until their hand sums to at least 17 or they bust. If neither the player nor the dealer busts, the winner is chosen by whose hand is closer to 21. An Ace can count as a 1 or an 11 in a player's hand, all face cards (king/queen/Jack) represent a value of 10, and all other cards represent a value equal to their numeric value.[1]

The game involves randomness due to card drawing, making it an ideal case for reinforcement learning. Reinforcement learning [2] is a branch of machine learning where it focuses on training agents to learn a particular task based on the reward-punishment system in a well-defined environment. By observing the outcomes of its actions, a reinforcement learning agent can autonomously learn optimal strategies, even in the absence of predefined rules. Blackjack is often considered as a test bed for existing reinforcement learning, and deep learning algorithms because of its comparatively limited state and action spaces, as well as its clearly stated rules and reward systems.

This paper looks to answer the following questions: Can a reinforcement learning agent autonomously learn optimal strategies for Blackjack? What profits can be expected for the different scenarios in a greedy evaluation of reinforcement learning approaches?

The rest of this paper is structured as follows. Section II describes the description of the Reinforcement Learning player. Section III presents the Experiments and Evaluation. The conclusion is discussed in Section IV.

II. Description of Reinforcement Learning player

A. Basic strategy

The game setup includes a single-deck Blackjack game where the dealer stands on a soft 17. The agent can choose from three actions: hit, stand, or double down. The agent learns strategies through interactions with the environment and adjusts its policy to improve performance over time.

The three reinforcement learning approaches: Q-learning, SARSA, and Monte Carlo Control were used to train the agent to learn optimal strategies for the blackjack game. Q learning [3], an off-policy method, updates its action-value estimates using the maximum expected reward for the next state, enabling it to focus on learning the optimal policy using the Bellman equation. However, SARSA, an on-policy algorithm [4], updates its action-value estimates using the agent's current policy, directly incorporating exploration effects using the Bellman equation. Monte Carlo Control methods [5], which can be on-policy or off-policy, update their action-value estimations by averaging the actual returns observed from complete episodes, using the law of large numbers rather than the Bellman equation to evaluate state-action pairs. An on-policy Monte Carlo Control method was implemented to evaluate the performance of the simulated blackjack environment. The key difference between approaches is the update rule.

The update rule for Q-learning is as follows [6]:

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left(r + \gamma \max_{a'} Q(s',a') - Q(s,a) \right) \quad (1)$$

In (1), $Q(s,a)$ represents the current Q-value for a given state s and action a. The parameter α is the learning rate. The term

Scenario	ϵ final	α	γ	Win Rate	Description
1	0.05	0.02	0.5	42.48%	Balanced exploration and learning.
2	0.1	0.02	0.5	41.75%	Higher exploration retained, by increasing the final epsilon value.
3	0.05	0.002	0.5	41.93%	Slower learning rate, by decreasing the learning factor.
4	0.05	0.2	0.5	40.02%	Faster learning rate, by increasing the learning factor.
5	0.05	0.02	0.1	41.72%	Short-term reward focus, by decreasing the discount factor.
6	0.05	0.02	1	41.23%	Long-term reward focus, by increasing the discount factor.

TABLE I: The ϵ-greedy scenarios and their influence in the implementation of the Basic Strategy in Q-learning reinforcement learning approach. The scenario 1, which is a balanced approach between exploration and learning gives the best result.

r stands for the reward obtained from performing action a in state s. The discount factor γ adjusts the importance of future rewards, and $\max_{a'} Q(s', a')$ identifies the highest Q-value obtainable from the next state s'.

The update rule for SARSA is as follows [6]:

$$Q(s, a) \leftarrow Q(s, a) + \alpha \left(r + \gamma Q(s', a') - Q(s, a)\right) \quad (2)$$

In (2), $Q(s, a)$ represents the current Q-value for a given state s and action a. The parameter α is the learning rate. The term r is the reward received after performing the action a in state s. The discount factor γ controls the importance of future rewards in the update. Lastly, $Q(s', a')$ is the Q-value corresponding to the next state-action pair, selected according to the policy being followed.

The update rule of Monte Carlo Control method is as follows [6]:

$$Q(s, a) \leftarrow Q(s, a) + \alpha \left[G - Q(s, a)\right] \quad (3)$$

In (3), $Q(s, a)$ represents the action-value estimate for a state s and action a. The term G denotes the return, which is the cumulative reward observed after visiting the state-action pair (s, a) in an episode. The parameter α is the learning rate, which adjusts the influence of new information on the updated value. In some variations, α may not be used, and the average is updated directly.

The state space contains the player total card value (1-21), the dealer's visible card (1-10, where face cards are treated as 10), a binary flag indicating whether the player has a soft hand or not, and a binary flag indicating whether it has doubled down or not. This representation ensures computational efficiency while preserving key information for decision-making. The action space consists of three actions: hit, stand, and double down. Split was excluded to reduce the complexity of the game. Each action directly impacts the continuation of the game and influences the agent's reward.

The size of the state-action space for a reinforcement learning agent is estimated. For Q learning method, the action space is three (hit, stand, and double down), dealer's displayed card ranges from 1 to 10 (10 states), the player's hand value runs from 1 to 21 (21 states), "Soft hand" adds two states (True or False), and "Doubled down" adds two states (True or False). The approximate total number of state-action pairs in total is 840 (21 * 10 * 2 * 2 = 840). So, the method has a total state action space of 2,520 states (total states × 3 = 2,520). Despite its size, the state-action space in blackjack is finite in Q-learning and SARSA. Stable estimates of $Q(\cdot, \cdot)$ are obtained with sufficient exploration and learning episodes, but convergence takes some time. However, because Monte Carlo updates values based on entire episodes, it takes a lot of episodes to provide accurate state-action value estimations. Simplified state space or advanced exploration and learning techniques can be used in agent training to address the convergence time.

The reward structure was designed in a way to maximize the agent's long-term winnings. A reward of +1 is given for a win, -1 for a loss, and 0 for a tie. The reward for winning a game that has previously been double down is 4, which helps the agent to prioritize doubling down winning.

An ϵ-greedy policy was used for exploration, with ϵ initialized to 1.0. The exponential decay formula was used to calculate the required epsilon decay. The exponential decay formula is:

$$N(t) = N_0 e^{-\lambda t} \quad (4)$$

where $N(t)$ is the quantity at time t, $N_0 = N(0)$ is the initial quantity (i.e., the quantity at time $t = 0$), and λ is the exponential decay constant. This approach ensured high exploration during the early stages while promoting exploitation as the agent converged toward optimal strategies. Optuna [7], an open- source hyperparameter optimization framework is used to find the best hyperparameters for the agent in Q-learning and SARSA approach. The final $epsilon$ was searched between 0.05 and 0.1, the learning rate between 0.01 and 0.5, and the discount factor between 0.5 and 1.0 to find the best hyperparameters.

One challenge I faced during agent training was finding the right balance between the exploration and exploitation phases. To address this, I implemented a decaying exploration rate and also found the best hyperparameters for agent training.

1) Rule variations implemented: Soft and hard hand rule variations are used to assess how agent adapts to the policy. The agent was asked to stand on a soft hand with a total value greater than 18 regardless of the dealer's up card (Soft 19). The action stand was asked to perform on 17+ regardless of the dealer's up card (Hard 17+). The agent was asked to perform frequent double-down actions for soft totals of 13 to 18 (Soft Doubling Rule).

B. Complete Point-Count System

The complete point count system, or high-low system, is a method designed to guide betting and playing decisions by assigning a specific value to each card. This allows players

State (player_total, dealer_up_card, is_soft, has_doubled_down)	Before Action	After Action	Explanation of Change
(15, 4, True, False)	'h'	'd'	Soft 15 vs. dealer 4 now favors doubling instead of hitting.
(16, 5, True, False)	's'	'd'	Soft 16 vs. dealer 5 now favors doubling down instead of standing.
(18, 2, True, False)	's'	'd'	Soft 18 vs. dealer 2 shifts to doubling for higher potential rewards.
(16, 10, False, False)	'h'	's'	Agent prefers standing on hard 16 vs. dealer 10, possibly minimizing losses.
(16, 4, False, False)	'h'	's'	Standing on hard 16 vs. dealer 4 to avoid busting.

TABLE II: Tells the impact of rule variations on policy changes in the implementation of the basic strategy scenario using the Q-learning reinforcement learning approach.

Parameter	SARSA	Q-Learning
Learning Rate	0.02785245521472777	0.022095606128790385
Discount Factor	0.5484400129288652	0.527737516246394
Final Epsilon	0.05027826788413271	0.05205650810936523

TABLE III: The optimal hyperparameters for the SARSA and Q-Learning reinforcement learning algorithms in basic strategy scenario. These hyperparameters, including the learning rate, discount factor, and final epsilon, were fine-tuned using Optuna to ensure the best possible performance for each approach.The intial ϵ is set to 1 to get full exploration in the beginning.

Parameter	SARSA	Q-Learning
Learning Rate	0.012300769827864733	0.01130525029479314
Discount Factor	0.5484400129288652	0.7101134217374672
Final Epsilon	0.05098351347681501	0.0504862898296134

TABLE IV: The optimal hyperparameters for the SARSA and Q-Learning reinforcement learning algorithms in complete point count system scenario. The initial ϵ is set to 1 to get full exploration in the beginning.

to estimate whether the remaining cards in the deck are more favorable to the player or the dealer [1]. The complete point count system is implemented by integrating the Hi-Lo card counting method [1] into the blackjack environment. In this method, cards ranked 2 through 6 are given a count of +1, cards ranked 7 through 9 are given a count of 0, and cards ranked 10, J, Q, K, and A are given a count of -1.

The state space has been updated to incorporate information about card count values. The revised state space contains the player total card value (1-21), the dealer's visible card (1-10, where face cards are treated as 10), a binary flag indicating whether the player has a soft hand or not, a binary flag indicating whether it has doubled down or not, and current card count. The action space, reward system, and $epsilon$-greedy policy remain the same as those used in the basic strategy scenario.

Q-learning, SARSA, and on-policy Monte Carlo reinforcement learning methods are employed to evaluate the performance of the complete point count system within a simulated blackjack environment. The reinforcement learning class used for the basic strategy is utilized to analyze the impact of the modified blackjack environment. The rule variations implemented remain consistent with those used for the basic strategy. Optuna framework is used to find the best hyperparameters for the complete point count system scenario. The final $epsilon$ was searched between 0.05 and 0.1, the learning rate between 0.01 and 0.5, and the discount factor between 0.5 and 1.0 to find the best hyperparameters.

III. Experiments and Evaluation

A. Basic strategy

The experiments are designed to examine and compare the performance of Monte Carlo Control, Q-learning, and SARSA in learning optimal Blackjack policies. The focus is on convergence rates, learning stability, and the overall effectiveness of the learnt policies in maximizing win percentage.

The experiments were conducted in a single-deck Blackjack environment where the dealer stands on soft 17. The action space included *Hit*, *Stand*, and *Double Down*. Each algorithm was trained over 100,000 episodes using an ϵ-greedy policy. Six different scenarios were evaluated during a greedy assessment of Q-learning, leading to the conclusion that the agent performs best when there is a balance between exploration and exploitation.

Performance was assessed based on the win rate, which is calculated as the ratio of the total number of games won by the agent to the total number of games played. The stability of the agent was evaluated by monitoring the variance in rewards across training episodes, represented on a graph. The graph exhibited an upward trend, indicating promising results as it demonstrates the agent's ability to consistently learn and achieve positive rewards throughout the game.

Each algorithm was trained for 100,000 episodes. After training, the learned policies were evaluated over 10,000 games against the policy that we got from agent training. The Monto Carlo Control method performed the best, achieving the highest win rate of 44.3%, the loss rate of 48.9%, and the tie rate of 8.5%, with a computation time of 45.05 seconds.

1) Rule variations implemented: The rule variations impacted policy change and also increased the win rate from 42% to 44%. For many borderline totals—like '6 vs. 10', '16 vs. 9', '16 vs. 4' — the agent's actions flipped from 'h' to 's'.

Scenario	ϵ final	α	γ	Win Rate	Description
1	0.05	0.01	0.5	43.53%	Balanced exploration and learning.
2	0.1	0.01	0.5	41.42%	Higher exploration retained, by increasing the final epsilon value.
3	0.05	0.001	0.5	41.86%	Slower learning rate, by decreasing the learning factor.
4	0.05	0.1	0.5	42.99%	Faster learning rate, by increasing the learning factor.
5	0.05	0.01	0.1	42.55%	Short-term reward focus, by decreasing discount factor.
6	0.05	0.01	1	42.48%	Long-term reward focus, by increasing discount factor.

TABLE V: The ϵ-greedy scenarios and their influence in the implementation of the complete point count system in a Q-learning reinforcement learning agent. Scenario 1, where there is a balance between exploration and exploitation gives the best result.

State (player_value, dealer_up_card_value, is_soft, self.has_doubled_down, count_info)	Before Action	After Action	Explanation of Change
(12, 10, False, False, -1)	'h'	's'	Reflects a safer strategy under unfavorable advantage.
(21, 10, True, False, -4)	'h'	's'	Prioritization of standing on a high total, even with a soft hand and low advantage.

TABLE VI: Impact of rule variations on policy changes in the implementation of the complete point count system scenario using the Q-learning reinforcement learning approach.

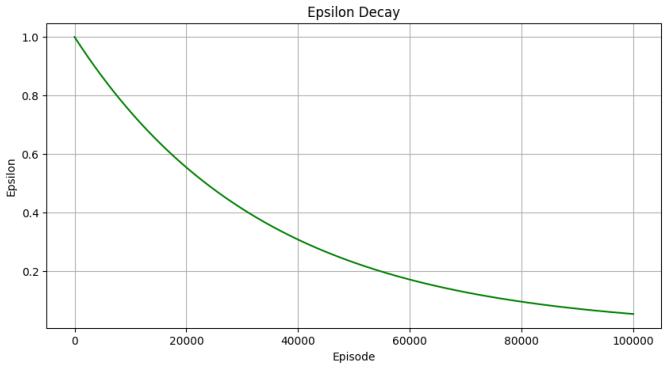

Fig. 1: The decay of epsilon values over episodes in the implementation of the Basic Strategy using the Q-learning reinforcement learning approach. Epsilon starts at 1.0, and decays to 0.05027826788413271

Fig. 2: The progression of rewards over episodes during the training of a Q-learning agent implementing the Basic Strategy.

Approach	Win Rate	Loss Rate	Tie Rate
Q-learning	42.48%	48.83%	8.69%
SARSA	42.64%	47.7%	8.38%
Monte-Carlo Control	**44.34%**	48.98%	8.52%

TABLE VII: The win rate, loss rate, and tie rate for the Q-learning, SARSA, and Monte Carlo Control approaches in the implementation of the basic strategy scenario.

For instance, the before policy: (16, 10, False, False) = 'h'. The After policy: (16, 10, False, False) = 's'. This strongly suggests the new rules changed the expected values enough that standing on 16 vs. 10 is now better than hitting.

In summary, Monte Carlo Control produced the most accurate and robust policy, with the highest win rate, by using the game's episodic structure to accurately link actions to their actual results, making it better at handling the randomness of the black jack environment. Q-learning exhibited faster convergence but struggled with rare states. SARSA offered stable performance but required more episodes to achieve optimal policy quality. These results highlight the trade-offs between on-policy and off-policy methods in reinforcement learning for Blackjack. Rule variations also influence the agent's decision-making process, enabling it to develop a more effective policy.

B. Complete Point Count System

The experiments are meant to understand the implications of a complete point count system in a blackjack game. The studies were carried out in a single-deck Blackjack table with the dealer standing on soft 17. The action space contained *Hit*, *Stand*, and *Double Down*. Each algorithm was trained over 100,000 episodes with a ϵ-greedy policy. Six different scenarios were evaluated during a greedy assessment of Q-learning, leading to the conclusion that the agent performs best when there is a balance between exploration and exploitation. The agent's performance was evaluated based on its win rate, with the graph displaying an upward trend, demonstrating improved learning over the episodes.

Each algorithm was trained for 100,000 episodes. After training, the learned policies were evaluated over 10,000 games against the policy that we got from agent training. The Monto Carlo Control method performed the best, achieving the highest win rate of 43.83%, the loss rate of 48.68%, and the tie rate of 7.49%. Compared to the basic strategy scenario,

each reinforcement learning approach increases by up to 1% in the win rate and by a reduction of up to 1% in the tie rate. This suggests that incorporating complete point count systems into the already simulated Blackjack environment allows the agent to adopt an efficient policy.

To enhance the system and achieve higher profits, multi-deck cards were implemented, and dynamic betting was introduced. This approach scales bets proportionally with the count instead of using a binary minimum or maximum. These adjustments enabled the agent to increase cumulative profits by 13.02%.

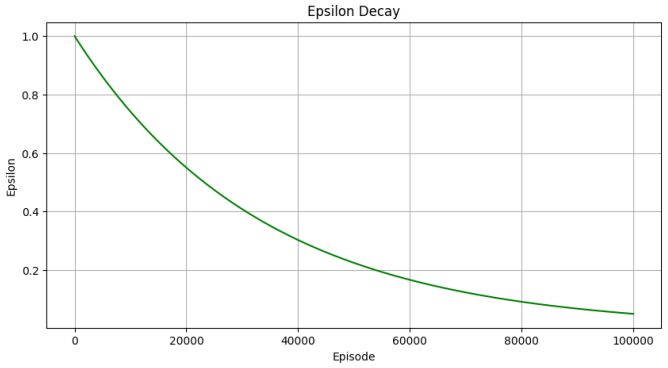

Fig. 3: The decay of epsilon values over episodes in the implementation of the complete point count system using the Q-learning reinforcement learning approach. Epsilon starts at 1.0, and decays to 0.0504862898296134 over 100,000 episodes.

Fig. 4: The progression of rewards over episodes during the training of a Q-learning agent implementing complete point count system scenario.

Approach	Win Rate	Loss Rate	Tie Rate
Q-learning	43.60%	48.51%	7.89%
SARSA	43.18%	49.29%	7.53%
Monte-Carlo Control	**43.83%**	48.68%	7.49%

TABLE VIII: The win rate, loss rate, and tie rate for the Q-learning, SARSA, and Monte Carlo Control approaches in the implementation of the complete point count system scenario.

1) Rule variations implemented: The rule differences had less of an impact on the win rate. The hard and soft hand decisions tend to be less impactful because complete point count systems provide precise information about the deck's composition, allowing players to make dynamic decisions based on the current count.

IV. Conclusion

In this study, reinforcement learning techniques, notably Q-learning, SARSA, and on-policy Monte Carlo, were applied to both the basic strategy and the complete point count system to increase blackjack winning rates under various rule modifications. The complete point count approach outperformed the basic strategy by a 1% winning rate, demonstrating the benefits of incorporating card counting into the agent's decision-making. However, altering hard and soft hand rules significantly improved the basic strategy but not the whole point count system. Monte Carlo control outperformed other approaches for maximizing agent performance.

Future research could look into integrating neural networks to apply methods across multiple blackjack scenarios. Developing individualized strategies based on individual player preferences and tendencies is another interesting area. Multi-agent reinforcement learning could replicate player interactions at the table, illustrating how collective decision-making affects optimal strategy. Furthermore, adopting explainable AI methodologies would improve model interpretability, revealing insights into decision-making in difficult circumstances.

References

[1] E. O. Thorp, *Beat the Dealer*, Vintage, New York, 1966.

[2] Leslie Pack Kaelbling, Michael L. Littman, and Andrew W. Moore, "Reinforcement Learning: A Survey," *CoRR*, vol. cs.AI/9605103, 1996. Available: https://arxiv.org/abs/cs/9605103.

[3] M. A. Wiering and M. van Otterlo, "Q-Learning Algorithms: A Comprehensive Classification and Applications," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 49, no. 12, pp. 2502–2525, Dec 2019. Available: https://ieeexplore.ieee.org/document/8836506.

[4] Prajval Mohan, Lakshya Sharma, and Pranav Narayan, "Optimal Path Finding using Iterative SARSA," in *2021 5th International Conference on Intelligent Computing and Control Systems (ICICCS)*, pp. 811-817, 2021. doi: 10.1109/ICICCS51141.2021.9432202.

[5] Magnus Gedda, Mikael Z. Lagerkvist, and Martin Butler, "Monte Carlo Methods for the Game Kingdomino," in *2018 IEEE Conference on Computational Intelligence and Games (CIG)*, pp. 1-8, 2018. doi: 10.1109/CIG.2018.8490419.

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

[7] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama, "Optuna: A Next-generation Hyperparameter Optimization Framework," *CoRR*, vol. abs/1907.10902, 2019. Available: http://arxiv.org/abs/1907.10902.

Appendix

I acknowledge the assistance provided by QuillBot, an AI-powered paraphrase tool. QuillBot was used to paraphrase technical and descriptive text to ensure it was brief, readable, and adhered to academic norms.