Reinforment Learning for BlackJack

Carlos Miguel Veloso[1], Luís Henriques[1], Paulo Portela[1]

[a]*Faculty of Engineering of University of Porto (FEUP)*
Master's of Engineering and Data Science
Advanced Topics in Computational Learning
Porto, December 15, 2023

Abstract

Card games, celebrated for their versatility, span from rapid, luck-dependent contests to strategic, skill-based challenges. This study ventures into applying reinforcement learning (RL) techniques to one of the most enduring card games, Blackjack. Focusing on evaluating the efficacy of RL methods, including Q-Learning, Monte Carlo, and SARSA, the research aims to master Blackjack strategies. It examines the adaptability, resilience, and learning efficiency of RL agents within Blackjack's unique gaming environment, investigating the impact of hyperparameters and training methods on agent performance. A novel aspect of this study is the introduction of a sophisticated reward structure based on player types —Aggressive, Conservative, and Neutral — providing a tailored learning experience for each agent. The findings reveal that Neutral agents, across all methods, outperformed the average human performance in Blackjack. For instance, Neutral agents achieved win rates of 43.21% in Monte Carlo On-Policy First Visit, 43.37% in Monte Carlo On-Policy Every Visit, and 43.35% in Monte Carlo Off-Policy. Similarly, the Q-Learning Neutral and SARSA Neutral models showed promising results with 43.25% and 43.36% win rates, respectively. Despite not surpassing human baselines, Aggressive and Conservative agents offered valuable insights, with the Aggressive approach proving more effective than the Conservative. The study concludes that the SARSA Neutral model emerged as the most effective, closely followed by the Q-Learning Neutral and Monte Carlo Off-Policy Neutral models. These results represent a significant advancement in applying RL to Blackjack, demonstrating these agents' ability to optimize strategies and adapt to various scenarios, potentially outperforming rule-based approaches. This research not only highlights the potential and challenges of applying RL to the complex dynamics of Blackjack but also offers a thorough analysis of various RL methodologies and their outcomes, contributing to a deeper understanding of AI's application in strategic card games.

Keywords: Blackjack, Reinforcement Learning, Reward Structure, Artificial Intelligence

1. Introduction

Card games, celebrated for their versatility, encompass a spectrum ranging from fast-paced, chance-driven contests to intricate challenges driven by strategic acumen. This study immerses itself in the domain of reinforcement learning techniques and algorithms, applying them to one of the most enduring and iconic card games: Blackjack.

Blackjack, steeped in tradition, has consistently presented a formidable challenge to players as they navigate the delicate equilibrium between the unpredictability of card draws and meticulously crafted strategies. In contrast to traditional approaches anchored in fixed rules, our study ventures into the realm of reinforcement learning, aiming to dynamically unveil optimal game strategies.

At the heart of our exploration lies the assessment of the effectiveness of reinforcement learning techniques in mastering the art of Blackjack. Our research employs a diverse array of reinforcement learning methodologies, encompassing various Monte Carlo techniques, Q-Learning, and SARSA approaches. These strategies exhibit remarkable adaptability across diverse game scenarios, hinting at the potential to surpass more traditional, rule-bound strategies. Moreover, our study delves into the nuanced exploration of the impact of hyper-parameters and training methods on the performance of reinforcement learning agents. This analysis seeks to provide insights into fine-tuning and enhancing the application of these techniques within the intricate landscape of Blackjack.

This paper serves as a guide through the exploration of key stages, unveiling the results and implications of our research. We aim to shed light on the promises and challenges inherent in the realm of reinforcement learning, especially when applied to the complex dynamics of Blackjack.

2. Problem Description

The challenge inherent in the game of Blackjack lies in the pursuit of optimal strategies that delicately balance the unpredictability of card draws with the necessity for well-informed decisions. Historically, game strategies have been constructed upon fixed rules, prescribing predetermined actions for specific scenarios. However, this study acknowledges the potential of reinforcement learning techniques to dynamically adapt and optimize strategies in real-time gameplay.

The core problem we address unfolds on two fronts: firstly, we embark on a quest to comprehend how reinforcement learning methodologies, encompassing Monte Carlo techniques, Q-Learning, and SARSA approaches, can effectively learn and implement strategies for playing Blackjack. Secondly, our aim is to scrutinize the influence of various hyper-parameters and training methods on the performance of reinforcement learning agents.

Our investigation stems from the realization that the dynamic and evolving nature of Blackjack necessitates strategies capable of adapting to a myriad of scenarios. Traditional rule-based approaches may falter in capturing the complexity and adaptability demanded for success in Blackjack. Hence, our problem statement revolves around the exploration of whether reinforcement learning can furnish superior strategies, and if so, understanding the nuances of their effectiveness.

In essence, our objective is to unveil the potential of reinforcement learning to revolutionize our approach to solving strategic challenges in the game of Blackjack. The problem transcends merely achieving a high win rate or losing less money; it delves into comprehending the adaptability, robustness, and learning efficiency of reinforcement learning agents in this specific gaming environment.

3. Reinforcement Learning

Reinforcement Learning, a subset of machine learning, emulates human decision-making by rewarding favorable actions and penalizing undesirable ones. This paradigm involves providing the model with inputs that encapsulate the current state and potential actions within a scenario. The model then receives rewards based on its output, reinforcing positive behaviors and discouraging detrimental ones. Over time, reinforcement learning models become proficient through trial-and-error interactions with their environment, autonomously adapting and optimizing strategies without direct human intervention.

3.1. Fundamental Concepts

Understanding key concepts is crucial for grasping the structures and processes underlying our study of how reinforcement learning agents interact with their environment. Here are the fundamental concepts:

- **Agent**: The reinforcement learning model, serving as a decision-maker and learner;
- **Environment**: The world around the agent, where it learns and acts. The environment returns the next state and reward based on the agent's actions; **state** and the **reward**;
- **State**: A complete description of the conditions of the environment;
- **Action**: The way an agent interacts with the environment. Action Space is the set of all possible actions the agent can perform;
- **Reward**: Feedback from the environment, indicating success or failure. Rewards impact the agent's behavior, and agents learn to estimate rewards correctly;
- **Policy**: A rule used by an agent to decide actions based on the state. It maps from state to action or provides probabilities for each action in the action space;
- **Value Function**: The function returning the expected total reward an agent can get from following a specific policy. In this work, we use the state-action value function *Q(S, A)* estimating rewards for taking action *A* from state *S*.

3.2. Process

As illustrated in Figure 1, the continuous cycle involves the agent playing multiple games, selecting actions, and updating its strategies based on the obtained rewards. This cyclical learning process is fundamental to the adaptive nature of reinforcement learning and forms the cornerstone of our exploration into intelligent decision-making in the context of the chosen environment.

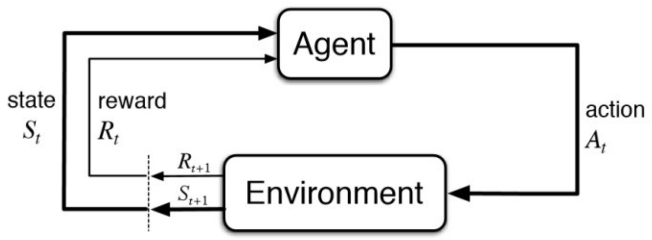

Figure 1: Reinforcement Learning Process

The process behind reinforcement learning, highlighting how the agent evolves and improves its strategies over time, will be explored and described below. By understanding the essential workflow of reinforcement learning, it will be easier to appreciate the impact of the steps mentioned above and how they fit into the overall context of our study.

1. The agent plays multiple games;
2. In each game, the agent selects an action based on its policy and action-value function;
3. The action impacts the environment, resulting in a reward or penalty and a new state, returned to the agent;
4. The agent tracks rewards and penalties received for each action;
5. After completing the game, the agent updates estimated rewards for each state and action based on actual values obtained;
6. This cycle repeats continuously, illustrating how the agent evolves and improves its strategies over time. Understanding this workflow is crucial to appreciating the impact of the steps in the overall context of our study.

3.3. Exploring Reinforcement Learning Algorithms for Intelligent Decision-Making

Considering the adaptability inherent in these algorithms for learning and improving their game strategies, one could anticipate their effectiveness in a seemingly uncomplicated card game like Blackjack. However, the question arises: how do we measure success in this scenario? Is it determined by achieving a superior average win rate when compared to an algorithm that depends solely on random gameplay?

Initially, that was what we set out to do, but we realized that the learning capacity of the models was far superior, and that, with an adjustment made to the relevant metric to assess the agents, we could be more ambitious with our goal. So we investigated what the win rate, lose rate, and draw rate of a human in a one versus one game of blackjack would be, and then set that as our baseline for our agents to achieve and surpass.

On the other hand, we considered that win rate wasn't a good enough metric to evaluate the agents. This happens because even with a higher win rate the player might be losing more money than a player that wins less. Notice that a player with a win rate of 47% and a loss rate of 52% loses more money than a player with a win rate of 42% and a loss rate of 46%. So, the assessment metric we came up with was the difference between the win rate and the loss rate. Now, because this value will always be negative, given the odds margin the house has, the closer the difference rate gets to zero the better.

According to [1], this average winning percentage is around 42.22%, 8.48% in the case of draws, and the remaining 49.30% for the average loss. These figures are therefore an interesting target to aim for. The fact that we can create a model that at least keeps up with human performance would be quite satisfying. The models chosen have different approaches, which makes it very interesting, because experiments can be carried out with each of these models to study which performs best in Blackjack.

3.3.1. Random Agent

The use of a random agent in Blackjack involves employing a strategy where actions are chosen entirely at random, without any consideration of the current game state or historical information. While this approach might seem counter-intuitive in a game that requires strategic decision-making, it serves as a baseline for comparison when evaluating the effectiveness of more sophisticated reinforcement learning algorithms. In Blackjack, the random agent's role is to make decisions such as hit or stand without having any criteria about the cards in hand, both its own cards and those of the dealer. For this research, the purpose of this agent would be to establish a performance benchmark for the more complex models to be implemented later.

3.3.2. Human Performance

The use of a random agent in Blackjack can be very limited when we look at the rational decision-making point-of-view of this issue. So, becomes obvious that an agent that doesn't use rational thinking is an unrealistic baseline if we want to actually check the agent's performance against the one in a practical environment. When possible, a very used baseline is human performance. Therefore, although we will use the random agent, the ultimate baseline for the agents developed will be the human performance in a one versus one game of Blackjack.

3.3.3. Monte Carlo

Since the inception of the project, Monte Carlo has consistently been our preferred choice. The Monte Carlo Algorithm, a probabilistic approach relying on random sampling for numerical results, operates on the principle of exploring a system in a uniform and random sense. This method makes predictions by considering a range of values in the problem's domain rather than a specific input.

$$S_1, A_1, R_1 \rightarrow S_2, A_2, R_2 \rightarrow \ldots \rightarrow S_n \sim \pi \tag{1}$$

Applied to Blackjack, the Monte Carlo process can be simplified as follows:

1. **Problem Definition**: Identify the optimal strategy for playing Blackjack, considering factors such as the player's hand and the dealer's card;
2. **Simulation of Random Scenarios**: Generate random scenarios, including various combinations of hands for both the player and dealer, and potential card counts;
3. **Player Strategy Simulation**: For each scenario, simulate the player's strategy by deciding whether to hit or stand, and record the results for each decision;
4. **Results Collection**: Gather the results of each simulation, categorized into three possible outcomes: win, draw, or loss;
5. **Analysis and Strategy Determination**: Analyze the collected results to determine the strategy that maximizes winning and minimizes losses.

This represents the general approach of Monte Carlo Simulation. While there are standard techniques for its use, such as different types of policy evaluation and variations like First-Visit Monte Carlo or Every-Visit Monte Carlo, a crucial decision point is whether the learning agent explores the environment and updates its policy (on-policy or off-policy).

3.3.3.1. On Policy First Visit and Exploring Starts

In the categorization of agents into On Policy and Off Policy, the focus on On Policy methods is noteworthy. These methods aim to evaluate or enhance the policy guiding decision-making, utilizing the agent's current policy for both exploration and updating based on experiences. Unlike Off Policy methods, where the policy for generating data differs from the one being improved, On Policy methods maintain a direct relationship between exploration and policy update.

In the categorization of agents into On Policy and Off Policy, the focus on On Policy methods is noteworthy. These methods aim to evaluate or enhance the policy guiding decision-making, utilizing the agent's current policy for both exploration and updating based on experiences. Unlike Off Policy methods, where the policy for generating data differs from the one being improved, On Policy methods maintain a direct relationship between exploration and policy update.

The specific agent under consideration falls into the On Policy group, signifying that its exploration and policy updates are based on the same decision-making framework. In addition to being On Policy, this agent incorporates two key adjustments. The first adjustment involves setting First Visit to true. In this context, First Visit implies that during the learning process, the agent places significance on the initial occurrence of a state-action pair within a given scenario.

This unique configuration adds a layer of sophistication to the agent's learning process. By prioritizing the first occurrence of state-action pairs, the agent gains a nuanced understanding of the strategic intricacies involved in decision-making, contributing to a more refined and adaptive policy over time. This adjustment, coupled with the inherent nature of On Policy methods, positions the agent to comprehensively explore and improve its decision-making capabilities in the context of the Blackjack game.

3.3.3.2. On Policy Every Visit and Exploring Starts

Within the On Policy Every Visit category, the designated agent embraces a more comprehensive strategy by accounting for every instance of a state-action pair during simulations. This deliberate modification aims to cast a wider net, capturing a diverse spectrum of experiences. The objective is to foster a more extensive and varied dataset, ultimately leading to a more robust evaluation of the agent's policy.

By considering every visit, the agent gains exposure to a richer set of scenarios, facilitating a deeper understanding of the nuances in decision-making within the Blackjack game. This broader perspective contributes to a more informed and adaptable policy. The integration of the Exploring Starts technique in this context is pivotal. It ensures that the agent deliberately initiates simulations with a diverse array of initial state-action pairs, further enhancing its adaptability and strategic acumen.

In essence, the On Policy Every Visit agent adopts a holistic approach, leveraging a wealth of experiences to refine its decision-making policy. The synergy between considering every visit and incorporating the Exploring Starts technique establishes a framework for a nuanced, adaptable, and strategically adept learning process.

3.3.3.3. Off Policy

In the realm of Off Policy methods, a crucial advantage lies in their inherent flexibility, stemming from the separation of exploration and policy update phases. This design allows for a tailored exploration policy, specifically crafted for thorough and diverse scenario exploration. Simultaneously, the update policy concentrates on refining decision-making strategies based on learned experiences. This dichotomy not only facilitates adaptability but also offers a nuanced approach to balancing exploration and exploitation, a pivotal aspect in the intricate landscape of reinforcement learning.

In practical terms, Off Policy methods, by maintaining this separation, exhibit robust performance across diverse environments. The ability to adapt the exploration strategy independently from the update mechanism provides a dynamic learning framework. This approach proves especially beneficial in scenarios where a fine-tuned exploration strategy is necessary for uncovering the nuances of a complex game like Blackjack, ultimately contributing to more effective policy updates

3.3.4. SARSA (Temporal Difference control)

SARSA, standing for "State-Action-Reward-State-Action," represents an on-policy reinforcement learning algorithm. This distinctive approach allows SARSA to evaluate the value of state-action pairs in real-time, dynamically adjusting its policy as it engages with the environment. What sets SARSA apart is its innate proficiency in handling sequential decision-making processes. This capability makes it a fitting choice for dynamic games like Blackjack, where each decision's outcome significantly influences subsequent choices. SARSA's sequential decision-making prowess aligns seamlessly with the dynamic nature of the Blackjack environment, enhancing its adaptability and learning efficiency.

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[R_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t) \right] \tag{2}$$

In application, SARSA navigates the complexities of Blackjack by considering not only immediate rewards but also the long-term implications of its decisions. This forward-thinking approach enables the algorithm to grasp the intricate interplay of actions, consequences, and rewards in sequential scenarios. As a result, SARSA excels in scenarios where understanding the sequential nature of decision-making is paramount, showcasing its suitability for dynamic and evolving environments like those encountered in Blackjack.

Comparing with the previous model "Monte Carlo", these both models have different key aspects that affect the learning process and can also produce different results. Sara, for example, can perform better in some cases due to its sample efficiency as it updates value estimates more frequently, it updates its estimates after each time step. Another interesting key aspect is the computational efficiency where SARSA, due to its online nature, can perform better even in large state or action space - in our research it this computational difference of efficiency does not exists due to the fact of the nature of the Blackjack game.

The learning rate (α) determines how much the q-values are updated in each iteration. The discount factor (γ) influences the importance of future rewards in the q-value updates. A higher discount factor gives more weight to future rewards, influencing the agent to consider long-term consequences. The epsilon (ϵ) plays a crucial role in balancing exploration and exploitation. It determines the agent's strategy for selecting actions during the learning process. There are 2 epsilons, the initial and the final, and we have also implemented an epsilon decay, so that in the first episodes the agent explores more the environment by having a higher probability of making a random action, and then the epsilon decays. This epsilon decay, therefore, depends directly on the number of episodes for which the agent is trained.

Because the fine tuning was requesting great computational power as it was dealing with a high number of possible combinations, it can be said that task of finding the best combination requested much longer time than the task of training a specific model, even for much higher episodes during training.

3.3.5. Q-Learning

Q-Learning [4], distinguished as an off-policy reinforcement learning algorithm, focuses on learning the value of state-action pairs. Unlike SARSA, Q-Learning exhibits a distinctive quality during the learning phase by not necessitating a predefined policy. This characteristic grants Q-Learning the autonomy to explore and glean insights from experiences independently. Central to its functioning is the Q-table, a dynamic repository that encapsulates the anticipated cumulative rewards associated with each conceivable action within a given state. This comprehensive record empowers the algorithm to make informed decisions, capitalizing on learned experiences and paving the way for the optimization of decision-making strategies over time.

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[R_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t) \right] \tag{3}$$

In the context of Blackjack, Q-Learning's approach aligns seamlessly with the game's dynamic and evolving nature. The absence of a predetermined policy during the learning phase allows the algorithm to adapt organically to the shifting dynamics of the game. The Q-table, acting as a repository of accumulated knowledge, becomes a valuable asset in steering decision-making toward more rewarding outcomes. As we embark on a thorough exploration of these reinforcement learning algorithms, each model contributes a unique perspective, collectively enhancing our intelligent agent's ability to navigate the nuanced challenges posed by the intricate game of Blackjack. Comparing with SARSA, Q-learning tends to focus more on exploitation than SARSA as this last involves more exploration through is ϵ-greedy strategy. In terms of convergence, Q-learning is in general converging faster to the optimal policy and this is due to the fact that it considers (always) the maximum Q-value for the next state [5].

As with other models, Q-learning was also tuned with different parameters using a grid-search-like code to perform it, using the same parameters as in the SARSA.

4. Blackjack Environment

Within the realm of the Blackjack environment [3] tailored for reinforcement learning, we explore its intricacies, elucidating the rules, dynamics, and essential components that define the learning landscape for agents.

4.1. Rules and Dynamics

The Blackjack environment, implemented through the **gym** library [2], adheres to the traditional rules of the card game. Understanding the dynamics, including the card distribution, player options (hit and stand), and the decision-making process of the dealer, is paramount for comprehending the learning challenges faced by reinforcement learning agents.

In adherence to standard rules, each participant, including the dealer, is initially dealt two cards. The primary objective for players is to attain a card value higher than the dealer's without exceeding 21. Aces hold a dual value of 1 or 11, offering strategic flexibility, while face cards (such as 10, Jack, Queen, and King) carry a fixed value of 10. Numeric cards retain their face value. The dealer is mandated to hit until their hand totals 17 or more, opting to stand otherwise. This succinct portrayal captures the core elements of Blackjack, presenting a well-defined environment conducive to exploration in the realm of reinforcement learning

4.2. State and Action Space

The Blackjack environment is characterized by the available observations and the actions that the agent can take.

The observation is represented as a **3-tuple**, containing:

- The player's current score;
- The dealer's visible card;
- Information on whether or not the player has a usable ace.

This is expressed as Tuple(player's score, dealer's revealed card, usable ace), where each element is drawn from discrete spaces. Specifically, Tuple(Discrete(32), Discrete(11), Discrete(2)) is interpreted as follows:

- The player's score can range from 1 to 32, considering the possibility of having an ace with a value of 11;
- The dealer reveals only one card, which can have any value from 1 to 11;
- The 'usable ace' space is binary, indicating whether the player's hand includes an ace that can be used as 11. This results in a space of size 2.

Thus, the environment encompasses a total of $32 \times 11 \times 2 = 704$ possible states.

4.3. Rewards and Penalties

In the Blackjack environment, the reward system plays a pivotal role in shaping the behavior of reinforcement learning agents. Victories, defeats, and ties result in specific rewards, influencing the agent's strategy development. This sub-section delves into the intricacies of the reward and penalty system, shedding light on how it guides the learning process.

The reward structure is designed to reflect the outcomes of the game:

- Winning yields a positive reward, typically set to +1;
- Losing results in a negative reward, commonly set to -1;
- Ties or draws, where both the player and dealer have the same score, usually yield a reward of 0.

These rewards serve as essential signals for the reinforcement learning agent, guiding it to learn optimal strategies through trial and error. The explicit and immediate feedback provided by the reward system enables the agent to adjust its policies over time, ultimately converging towards effective decision-making strategies in the context of the Blackjack game.

4.4. Exploration-Exploitation: Epsilon-Greedy

In the dynamic landscape of the Blackjack environment, effective exploration of various strategies is crucial for the learning process. One prominent technique employed to strike a balance between exploration and exploitation is the epsilon-greedy strategy.

As previously discussed, a policy is a function guiding the agent on which action to take given the current state. In our Blackjack environment, a straightforward deterministic policy π for a specific state, such as $(15, 10, 0)$, could appear as illustrated below in the Figure 2:

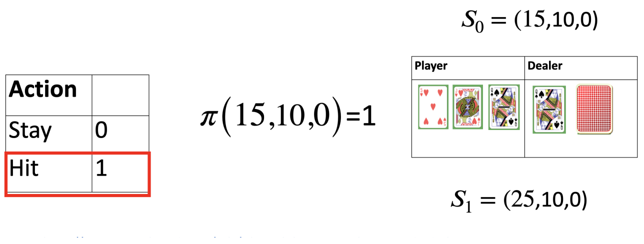

Figure 2: Epsilon-Greedy Policy

The epsilon-greedy strategy introduces an exploration parameter, epsilon (ϵ), which determines the agent's probability of choosing a random action versus exploiting the current best-known action. This approach is instrumental in preventing the agent from becoming overly reliant on suboptimal strategies and promotes the discovery of potentially superior actions.

During the training episodes, at each decision point, the agent selects the optimal action with a probability of $(1 - \epsilon)$ or explores a random action with a probability of (ϵ). This controlled randomness facilitates a more comprehensive exploration of the state-action space, contributing to a more robust learning process.

The choice of an appropriate epsilon value is a nuanced decision, impacting the trade-off between exploration and exploitation. A higher epsilon encourages more exploration, while a lower epsilon emphasizes exploitation of the learned strategies.

Integrating the epsilon-greedy strategy into the Blackjack environment provides an avenue for the reinforcement learning agent to adapt dynamically to various scenarios, ensuring a well-rounded and adaptable decision-making process. This exploration-exploitation strategy, incorporating the principles of exploration and exploitation, is a key component in enhancing the overall effectiveness of reinforcement learning methodologies applied to the complexities of Blackjack.

5. Diverse Player Types and Customized Rewards

In the pursuit of optimizing the strategy in our Blackjack reinforcement learning framework, the incorporation of diverse agent types is paramount. Beyond the conventional reward system offered by the gym environment for winning or losing a game, our approach introduces a nuanced custom reward structure. This innovative strategy involves the categorization of players into 3 distinct types based on their playing styles.

We also tried to add an extra reward when the agent got a draw, given the thought that by forcing a draw the agents might be able to keep their win rate and, by increasing the draw rate, make the loss rate lower, and therefore getting the difference rate even closer to zero. That being said, the after several experiments we checked that the draw rate didn't change much, and so the difference metric, so we excluded this custom reward.

In our quest to optimize the strategy in Blackjack, we introduce a sophisticated approach to reward calculation that goes beyond the default gym rewards. This dual reward system, combining the standard gym reward with a customized reward based on the player's identified type, aims to provide a tailored and nuanced learning experience for each agent. The objective is to recognize and reinforce specific playing styles, fostering adaptability and excellence in a diverse range of Blackjack scenarios. The types of player shown below were the types considered:

1. **Aggressive (Agressivo)**: The aggressive player adopts a playing style characterized by embracing risk and demonstrating a willingness to take bold actions in pursuit of higher rewards. This proactive and risk-taking nature is reflected in specific rewards and penalties within our customized reward system. When opting to draw a card with a card sum above 15, aligning with their risk-taking inclination, they receive a substantial reward of 0.3. However, to discourage excessively risky behavior, a penalty of -0.3 is applied if the player exceeds a card sum of 21. Moreover, their inclination to take more actions is further acknowledged with a reward of 0.1 for choosing to draw a card. Conversely, a penalty of -0.1 is incurred when opting to stay, discouraging overly conservative decisions. This tailored reward structure aims to enhance the learning experience for aggressive players within the dynamic context of Blackjack;
2. **Conservative (Conservador)**: The conservative player exhibits a risk-averse playing style, prioritizing cautious decisions to minimize potential losses. This strategic approach translates into specific rewards and penalties within our modified reward system. When opting to stay with a card sum below 12, aligning with their conservative nature, they receive a reward of 0.3. Additionally, for choosing not to draw a card, reflecting their preference for less risky gameplay, they gain a further reward of 0.1. Conversely, a penalty of -0.2 is applied when the player decides to draw a card, serving as a deterrent against excessive risk-taking. This nuanced reward structure aims to finely tune the learning experience for conservative players in the Blackjack environment;
3. **Neutral (Neutro)**: The neutral player cultivates a playing style marked by a delicate balance between conservatism and aggression, showcasing adaptability in strategy based on the ever-evolving game context. This balanced and context-sensitive approach is encapsulated in specific rewards and penalties within our customized reward system. Opting to draw a card with a card sum of 15 or below, re-

flecting a moderately risk-taking stance, is rewarded with a notable 0.3. Conversely, to discourage overly conservative decisions, a penalty of -0.1 is incurred when choosing to stay. This refined reward structure aims to cater to the adaptive and versatile nature of neutral players, enriching their learning experience within the dynamic landscape of Blackjack.

As we explore this nuanced reward system, our aim is to observe its impact on agent behavior, cultivating strategic diversity and adaptability. Through this optimization process, we anticipate an enhancement in the overall performance of our intelligent agents within the intricate and dynamic landscape of Blackjack.

6. Results

Although winning more games on average is a very important metric, we need to also consider if the agent loses a lot or not. Notice that even an agent that wins more games that another agent might also lose more games, and therefore this first agent could lose more money in practice. So, a metric that we came up with in order to properly determine if an agent is better than another one is by subtracting the average wins by the average loses. The results present in this section refer to the tests done while using a grid-search to do multiple combinations of the hyper-parameters in cycles of 200.000 games. All results obtained with the final agents are presented in Table 1. All agents were trained for, more or less, 1 hour and 30 minutes. All agents except Monte Carlo Off-Policy were trained with 100,000,000 episodes, while this exception was trained for 20,000,000 given the training rate of all the other being around 18000-21000 episodes per second, while the Monte Carlo Off-Policy agents' training rate was 4000-5000 episodes per second.

For all Monte Carlo algorithms the theta was set to 0 after a first round of hyper-parameters tuning, given that the average wins drooped more than 10% for any other value. Regarding Monte Carlo On-Policy First Visit & Exploring Starts, the best combination was with a discount factor of 0.9, an epsilon 0.5 and a theta of 0. Looking at the results, for the aggressive, conservative and neutral agents, the average wins, losses, draws, and difference were (40.02%, 50.96%, 9.02%, -10.94%), (41.30%, 52.67%, 6.03%, -11.37%), and (43.21%, 47.93%, 8.86%, -4.72%), respectively. The results were (39.25%, 52.05%, 8.70%, -12.80%), (40.75%, 53.32%, 5.93%, -12.57%), and (43.37%, 47.99%, 8.63%, -4.62%), respectively. For Monte Carlo On-Policy Every Visit & Exploring Starts the best combination was discount factor of 0.7 and an epsilon of 0.05. For Monte Carlo Off-Policy the best hyper-parameters were a discount factor of 0.7 and an epsilon of 0.05. The results were (41.45%, 49.44%, 9.11%, -7.99%), (40.76%, 53.30%, 5.93%, -12.54%), and (43.35%, 47.82%, 8.83%, -4,47%), respectively.

Regarding the Q-Learning algorithm, there were more possible combinations as it have more possible parameters to tune and also a epsilon decay. The best model has a learning rate of 0.01, start epsilon of 0.4, an epsilon decay of 8e-09, final epsilon of 0.4 and a discount factor of 0.95. The results were (41.28%, 49.48%, 9.24%, -8.20%), (41.40%, 52.52%, 6.07%, -11.12%), and (43.25%, 47.51%, 9.24%, -4.26%), respectively.

Agents	Wins	Losses	Draws	Diff
Random Agent (Initial Baseline)	28.33%	67.49%	4.17%	-39.16%
Human (Baseline)	42.22%	49.10%	8.48%	-6.88%
MC On-policy First Visit Aggressive	40.02%	50.96%	9.02%	-10.94%
MC On-policy First Visit Conservative	41.30%	52.67%	6.03%	-11.37%
MC On-policy First Visit Neutral	43.21%	47.93%	8.86%	**-4.72%**
MC On-policy Every Visit Aggressive	39.25%	52.05%	8.70%	-12.80%
MC On-policy Every Visit Conservative	40.75%	53.32%	5.93%	-12.57%
MC On-policy Every Visit Neutral	43.37%	47.99%	8.63%	**-4.62%**
MC Off-policy Aggressive	41.45%	49.44%	9.11%	-7.99%
MC Off-policy Conservative	40.76%	53.30%	5.93%	-12.54%
MC Off-policy Neutral	43.35%	47.82%	8.83%	**-4,47%**
Q-Learning Aggressive	41.28%	49.48%	9.24%	-8.20%
Q-Learning Conservative	41.40%	52.52%	6.07%	-11.12%
Q-Learning Neutral	43.25%	47.51%	9.24%	**-4.26%**
SARSA Aggressive	41.40%	49.49%	9.11%	-8.09%
SARSA Conservative	40.56%	53.64%	5.80%	-13.08%
SARSA Neutral	43.36%	47.60%	9.05%	**-4.24%**

Table 1: Final results

Concluding the results section with the SARSA agents, we highlight the most effective model, which boasts a learning rate of 0.0001, an initial epsilon of 0.4, an epsilon decay rate of 8e-09, a final epsilon of 0.4, and a discount factor of 0.70. The performance metrics for this model were 41.40% wins, 49.49% losses, and 9.11% draws, resulting in a net difference of -8.09%. Notably, the conservative strategy yielded 40.56% wins and 53.64% losses with a slightly lower draw percentage of 5.80%, culminating in a net difference of -13.08%. The neutral strategy emerged as the most successful, with 43.36% wins, 47.60% losses, and 9.05% draws, which translates to the most favorable net difference of -4.24%. In the context of these results, Figure 3 illustrates the policy matrix for the SARSA Neutral agent. This matrix visualizes the recommended actions—whether to 'hit' or 'stick'—based on the player's hand sum and the dealer's showing card when the player does not have a usable ace. Each cell in the matrix corresponds to a specific hand combination, with '1' indicating the action to 'hit' and '0' representing the decision to 'stick'.

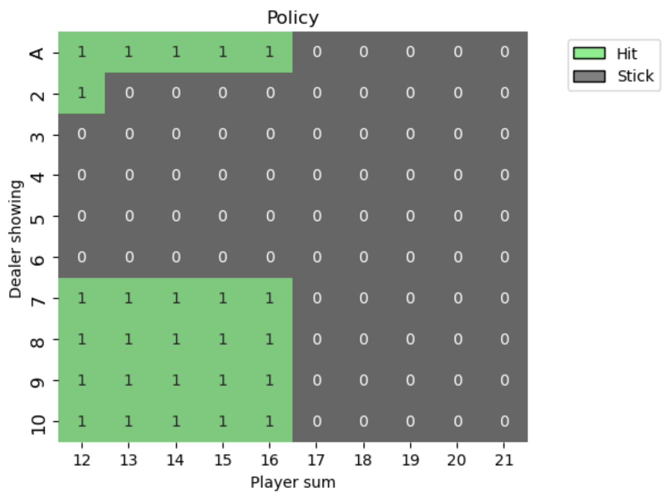

Figure 3: SARSA Neutral's policy without usable ace

Additionally, the Figure 4 extends this analysis by presenting the 3D graph of the SARSA Neutral agent's value function, which is a cornerstone of reinforcement learning. The graph's axes—dealer's showing card, player's hand sum, and expected return—align with the agent's learned experiences. The color gradient represents the nuanced understanding the agent has developed over time, with warmer colors indicating states with higher expected returns. This graph is not just a representation of data but a depiction of the agent's valuation of each state, guiding decisions in complex, uncertain scenarios. This visual tool encapsulates the learned efficacy of the neutral strategy, which outperforms the conservative and agresive strategies.

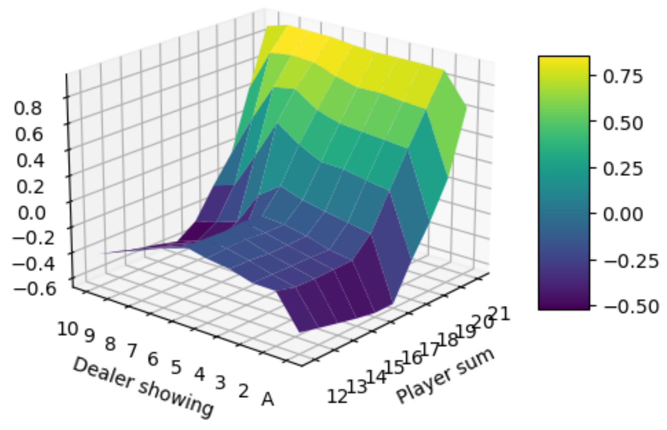

Figure 4: SARSA Neutral's value function without usable ace, after converting state-action values to state values

7. Summary and conclusions

All the different algorithms went through a hyper-parameter tuning face for a total of 200,000 episodes, and from there the final agents were trained for either 100,000,000 or 20,000,000 episodes. Monte Carlo off-policy had a training rate of around 4000-5000 episodes per second, while for all the others the training rate was between 18000-21000 episodes per second. By using those numbers of episodes, all agents had around the same time to train (1 hour and 30 minutes, more or less). From there, the final results were obtained by putting the agents in a test environment, where they always followed the final policy (without any greedy method with a certain epsilon). They all played 1,000,000 games to get the final results.

Comparing the difference metric of the human performance against all the agents, only (and all) the neutral versions of the agents were able to outperform humans in blackjack. On the other hand, although the aggressive and the conservative agents didn't outperform the baseline, it's interesting to mention that, in general, and considering our experiments and results, it's better to play aggressively than in a more conservative way. That being said, looking at the policy of these conservative models, we can see that by giving a relative small reward for conservative plays the agent always stands when it doesn't have a usable ace. However, even if always standing when it doesn't have a usable ace, the conservative agents still get average wins higher than 40%. The problem with this policy is that the agent loses more times because it never plays against the dealer by hitting, and this policy also makes the draw percentage decrease. Like we said previously when explaining the difference metric, although the conservative agent wins many games, being able to beat aggressive agents or getting very close regarding the win rates, the problem is that the aggressive agent loses much less games, making it the best strategy because it loses less money.

At the end, the best model was the SARSA Neutral agent, followed by the Q-Learning Neutral and Monte Carlo Off-Policy Neutral.

References

[1] Aneeca Younas, *Blackjack Odds and Probability*, `https://www.techopedia.com/gambling-guides/blackjack-odds-probability`.

[2] Artem Arutyunov, *Win at Blackjack with Reinforcement Learning*, `https://medium.com/the-power-of-ai/blackjack-with-reinforcement-learning-95f588dd670c`.

[3] Blackjack, *Gym Documentation*, `https://www.gymlibrary.dev/environments/toy_text/blackjack/`.

[4] Gymnasium Documentation, *Solving Blackjack with Q-Learning*, `https://gymnasium.farama.org/tutorials/training_agents/blackjack_tutorial/`.

[5] Sjoerd Vink, *Temporal Difference Learning: SARSA vs Q-Learning*, `https://medium.com/codex/temporal-difference-learning-sarsa-vs-q-learning-c367934b8bcc`.