Achieving Optimal Blackjack Play Through Double Q-Learning

COMP3106 Final Project
Qayam Damji (101287631)
Shri Vaibhav Mahesh Kumar(101287156)
Daniel Tam(101267869)

Carleton University
Prof. Holden
December 6, 2024

Contents

1 Statement of Contribution **2**

2 Introduction **2**
2.1 Background and Motivation . . . 2
2.2 Related Prior Work . . . 2
2.3 Statement of Objectives . . . 2

3 A Formal Definition of Blackjack **3**
3.1 Basic Blackjack Strategy . . . 3
3.2 How to Get Kicked Out of a Casino . . . 3

4 Methodology **4**
4.1 The Environment . . . 4
4.2 Q-Learning . . . 5
4.3 Double Q-Learning . . . 5
4.4 Warm Start & the Reward Function . . . 5
4.5 Epsilon-Greedy Exploration . . . 6
4.6 Prioritized Experience Replay . . . 6
4.7 Hyperparameter Optimization Implementation . . . 6
4.8 Training Algorithm . . . 7
4.9 Validation . . . 7
4.9.1 Training Validation . . . 7
4.9.2 Strategy Analysis . . . 7

5 Results **7**
5.1 The Search for Optimal Hyperparameters . . . 7
5.2 Training Metrics . . . 8
5.3 Reviewing the Learned Strategy . . . 10
5.3.1 Hard Totals Strategy . . . 10
5.3.2 Soft Totals Strategy . . . 10
5.3.3 Pairs Strategy . . . 11

6 Discussion **11**
6.1 Limitations . . . 11
6.1.1 Performance Ceiling . . . 11
6.1.2 Splitting Strategy . . . 11
6.1.3 Casino Simulation Fidelity . . . 11
6.2 Implications . . . 11
6.2.1 Strategic Learning Beyond Basic Rules . . . 11
6.2.2 Card Counting Integration . . . 12

1 Statement of Contribution

Each of us made an approximately equal contribution to the project.

- Qayam researched and implemented the core of the learning process for the agent, created the demo application and wrote the technical portion of the report.
- Daniel implemented the environment and state representation used by the agent, as well as writing the discussion and results section of the report.
- Shri implemented the code that performs statistical analysis of model strategy, as well as generating the plots used in the report. He also made the largest contribution to the proposal, and wrote the intro to the paper.

2 Introduction

2.1 Background and Motivation

Blackjack represents an intriguing intersection of skill and chance, making it an ideal environment for exploring reinforcement learning techniques. This project was motivated by the challenge of developing an intelligent agent capable of making optimal decisions in a probabilistic game environment. The inherent complexity of Blackjack—with its nuanced rules, strategic decision-making, and element of randomness—provides a compelling test for advanced machine learning algorithms. The fun of this project lies in its potential to create an AI system that can learn and adapt to the intricate strategies of a casino game. Unlike deterministic environments, Blackjack requires balancing risk, probability, and strategic thinking, which makes it an excellent domain for testing and refining intelligent decision-making algorithms.

2.2 Related Prior Work

Some prior work that relates to our selected model, algorithm, and task is Q Learning, Double Q learning, and poker research conducted by FAIR(Facebook AI Research). Q Learning was a reinforcement learning technique introduced by Christoper Watkins. It allowed for agents to learn optimal action-value functions through iterative experience. This was based on estimating the value of state-action pairs and determining the best move. This was then further developed by Hado van Hasselt in 2010 to introduce double Q learning. By introducing two separate Q Value estimates, the algorithm mitigates the overestimation bias found in traditional Q learning algorithms. This was very important in stochastic environments, such as our current environment of Blackjack, since overly optimistic value estimates can affect the decision making. FAIRS poker research also helped in pioneering applying reinforcement learning to strategic games.Their work on Poker demonstrated the potential of AI in information-imperfect games. In 2017, FAIR and CMU researches helped develop an AI that was able to defeat professional poker players, showcasing how these advances reinforcement leaning technique can navigate complex, probabilistic and stochastic environments, such as Blackjack.

2.3 Statement of Objectives

Develop a Q-Learning agent capable of playing Blackjack at a optimal level, integrating advanced techniques like Double Q-Learning. Implement a comprehensive learning environment that captures the nuanced rules and strategic complexity of Blackjack, including features like card counting and multiple action spaces. Systematically optimize the agent's performance through rigorous hyperparameter tuning and validation against standard Blackjack strategies. Analyze the learned strategy to understand how the agent develops decision-making patterns that approach or potentially exceed traditional basic Blackjack strategy.

3 A Formal Definition of Blackjack

Blackjack is the most widely played casino game in the world, due to it's intoxicating combination of skill and chance. The fact that it is incredibly simple to play, (but hard to master), quick, and very social make it a staple in casinos worldwide.

Blackjack is played between individual players and a dealer, with each player having their own independent game against the house. The game begins with players placing their bets, after which the dealer deals two cards to each player and themselves. Players' cards are typically dealt face-up, while the dealer takes one card face-up (known as the "upcard") and one face-down (the "hole card").

The goal is to beat the dealer by building a hand that totals closer to 21 than the dealer's hand, without exceeding 21. Cards 2 through 10 are worth their face value, while face cards (Jacks, Queens, and Kings) are worth 10. Aces possess flexibility, being worth either 1 or 11, depending on which value benefits the hand more. A hand containing an Ace counting as 11 is called a "soft" hand, as its total can drop by 10 (by converting the Ace to a 1) without busting.

After receiving their initial two cards, players must decide how to play their hand. They can "hit" to request additional cards, or "stand" to keep their current total. Players may continue hitting until they either decide to stand or their total exceeds 21, resulting in a "bust" and immediate loss of their bet. "splitting" permits players to divide a pair of matching cards into two separate hands, each requiring an additional bet equal to the original wager.

The dealer plays their hand last, following strict rules that eliminate any decision-making. The dealer must hit on any total of 16 or below and stand on any total of 17 or above (though some casinos require dealers to hit on a "soft 17"). If the dealer busts, all remaining players win. If the dealer does not bust, each player's hand is compared to the dealer's, with the higher total winning. Equal totals result in a "push," and the player's bet is returned.

3.1 Basic Blackjack Strategy

- Always hit on hard totals of 8 or below, and stand on hard totals of 17 and above.
- Hit on soft totals of 17 or below, and stand on soft totals of 19 or above.
- For hard totals of 12-16, stand if the dealer's upcard is 6 or lower (except hit 12 against 2 or 3), and hit if the dealer shows 7 or higher.
- For pairs, always split aces and 8s, never split 5s or 10s, split 2s and 3s against dealer's 4-7, and split 6s against dealer's 2-6.
- (SOURCE)

3.2 How to Get Kicked Out of a Casino

In order to gain an edge, skilled Blackjack players employ techniques classified under the umbrella term "card counting". There are many different techniques for card counting, varying in complexity and effectiveness. The general idea is to track which cards remain in the deck being dealt from, in order to make educated predictions on when to hit or stand. Our agent is provided with the information for Hi-lo card counting during training:

- Hi-lo card counting assigns values to cards seen during play:
- low cards (2-6) are worth 1
- mid-range cards (7-9) are 0
- high cards (10s through Aces) are -1

The player keeps a running count of these values as cards are dealt. When divided by the number of remaining decks, this gives a "true count." A positive count suggests many high-value cards remain, increasing the chances of strong hands and blackjacks, so players should bet more. A negative count indicates many low cards remain, favoring the dealer, so players should minimize bets. (SOURCE)

4 Methodology

4.1 The Environment

The state space S is represented as a 6-tuple:

$$(player\_value, has\_usable\_ace, dealer\_upcard, count\_bucket, is\_pair, pair\_value)$$

Each component provides information for decision-making:

1. **player_value** ($\in [4, 21]$): The current total value of the player's hand.
2. **has_usable_ace** ($\in \{0, 1\}$): Indicates whether the player has an Ace that can be counted as 11 without causing a bust.
3. **dealer_upcard** ($\in [1, 10]$): The visible card of the dealer's hand.
4. **count_bucket** ($\in \{-1, 0, 1\}$): Represents the true count of the deck using the Hi-Lo card counting system, bucketed into three categories:
   - -1: Negative count (deck rich in low cards)
   - 0: Neutral count
   - 1: Positive count (deck rich in high cards)
5. **is_pair** ($\in \{0, 1\}$): Indicates whether the player's initial two cards are a pair, enabling split decisions.
6. **pair_value** ($\in [0, 10]$): The value of each card in the pair if is_pair is true, 0 otherwise. This allows for specific pair-based strategies.

The action space A consists of three possible actions:
$A = \{0$ (Stand)$, 1$ (Hit)$, 2$ (Split)$\}$
Each action has specific conditions:

1. **Stand (0)**: The player keeps their current hand and ends their turn. Always available.
2. **Hit (1)**: The player draws one additional card. Available unless the hand has busted.
3. **Split (2)**: Available only when:
   - The hand contains exactly two cards of equal value
   - The hand hasn't already been split (tracked by split_depth)
   - The maximum number of splits (3) hasn't been reached

   When executed, creates two separate hands, each receiving one additional card.

The environment implements several realistic constraints:

1. **Maximum Splits**: Limited to 3 splits per round to prevent infinite splitting
2. **Dealer Rules**: Dealer must hit on soft 17 or below, stand on hard 17 or above
3. **Natural Blackjack**: Special payouts (1.5x) for natural blackjacks (21 with first two cards)

4.2 Q-Learning

We have employed Q-Learning [5], a model-free reinforcement learning algorithm that learns a state-action value function $Q(s, a)$ representing the predicted cumulative future reward of taking action a in state s. The values of $Q(s, a)$ are generated through experimentation in the environment and updated through the following equation:

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

Where:

- α is the learning rate
- γ is the discount factor
- R is the immediate reward
- s' is the next state
- $\max_{a'} Q(s', a')$ is the maximum expected future reward

We make use of a dynamic learning rate:

$$\alpha(s, a) = \max(\alpha_0 \cdot \delta^{N(s,a)}, \alpha_{\min})$$

where $N(s, a)$ is the visit count and δ is the decay rate.

4.3 Double Q-Learning

The Double Q-Learning Technique [4] is employed here, as it has been shown to reduce overestimation bias in highly stochastic environments, like blackjack. This technique encorporates two Q-Tables (Q_1, Q_2) and the following update function:

$$Q_1(s, a) \leftarrow Q_1(s, a) + \alpha[R + \gamma Q_2(s', \arg\max_{a'} Q_1(s', a')) - Q_1(s, a)]$$

4.4 Warm Start & the Reward Function

The reward function we have created is nuanced, going beyond simple binary win/loss based rewards in order to encourage optimal play. The reward function is designed in order to teach behavior like encouraging safe play when at risk of busting, and hitting on lower totals to reach stronger hands.

$$R(p, d) = \begin{cases} -1.2b & \text{if } p > 21 \\ 1.1b & \text{if } d > 21 \\ 1.5b & \text{if } p = 21 \text{ (natural)} \\ 1.1b & \text{if } p > d \text{ and } p \geq 20 \\ b & \text{if } p > d \\ -b & \text{if } p < d \\ 0 & \text{if } p = d \end{cases} \tag{1}$$

Where p =the value of the player's hand, and d =the value of the dealer's hand.

4.5 Epsilon-Greedy Exploration

The implementation uses an adaptive epsilon-greedy strategy for action selection. [3] The probability of selecting action a in state s is given by:

$$P(a|s) = \begin{cases} 1 - \epsilon + \frac{\epsilon}{|A|} & \text{if } a = \arg\max_{a'} Q(s, a') \\ \frac{\epsilon}{|A|} & \text{otherwise} \end{cases} \tag{2}$$

The exploration rate ϵ follows a performance-based decay mechanism:

$$\epsilon = \max(\epsilon_{\min}, \epsilon \cdot \begin{cases} \delta_\epsilon \cdot 1.1 & \text{if improving} \\ \delta_\epsilon & \text{otherwise} \end{cases}) \tag{3}$$

Where:

- $\epsilon_{\min}$ is the minimum exploration rate
- δ_ϵ is the base decay rate
- "Improving" is determined by comparing recent average rewards to best seen

4.6 Prioritized Experience Replay

We use prioritized experience replay to focus learning on the most informative transitions.[1] Transitions are sampled with probability:

$$P(i) = \frac{p_i^\alpha}{\sum_k p_k^\alpha} \tag{4}$$

The priority p_i is the priority based on temporal-difference error:

$$p_i = |\delta_i| + \epsilon \tag{5}$$

Where:

- δ_i is the TD-error: $\delta_i = r + \gamma \max_{a'} Q(s', a') - Q(s, a)$
- ϵ is a small constant ensuring non-zero probability
- α determines the strength of prioritization

4.7 Hyperparameter Optimization Implementation

The Q-learning agent's hyperparameters are optimized through a systematic grid search framework. The optimization process explores the parameter space Θ through scikit-learn's ParameterGrid [2]:

$$\Theta = \{\theta_1 \times \theta_2 \times \cdots \times \theta_n\}$$

The system employs parallel processing to efficiently evaluate configurations.
The optimal configuration θ^* is selected based on win rate maximization:

$$\theta^* = \arg\max_{\theta \in \Theta} m_{\text{win}}(\theta)$$

4.8 Training Algorithm

Algorithm 1 Double Q-Learning with Prioritized Replay & Epsilon-Greedy Exploration

Initialize replay buffer $\mathcal{D}$
for episode = 1 to N **do**
    Initialize state s
    while not terminal **do**
        Select action a using epsilon-greedy
        Execute a, observe r, s'
        Store transition (s, a, r, s') in $\mathcal{D}$
        if time to update **then**
            Sample batch from $\mathcal{D}$ using priorities
            Update either Q_1 or Q_2 randomly
            Update transition priorities
        end if
        $s \leftarrow s'$
    end while
    Update ϵ based on performance
end for

4.9 Validation

The validation of the Q-learning agent's performance was conducted as follows:

4.9.1 Training Validation

During the training phase, several metrics were continuously monitored and logged:

- Win rate ($\frac{\text{wins}+0.5\cdot\text{draws}}{\text{total games}} \cdot 100\%$)
- Average reward per game
- Split rate ($\frac{\text{splits}}{\text{total games}} \cdot 100\%$)
- Blackjack rate ($\frac{\text{blackjacks}}{\text{total games}} \cdot 100\%$)
- Bust rate ($\frac{\text{busts}}{\text{total games}} \cdot 100\%$)

4.9.2 Strategy Analysis

The learned strategy was analyzed through three distinct matrices:

1. Hard totals: Player values (12-21) vs. Dealer up-card (2-11)
2. Soft totals: Player values with usable ace vs. Dealer up-card
3. Pairs: Split decisions for each possible pair vs. Dealer up-card

These matrices were compared against basic strategy to validate the agent's learning of optimal play patterns.

5 Results

5.1 The Search for Optimal Hyperparameters

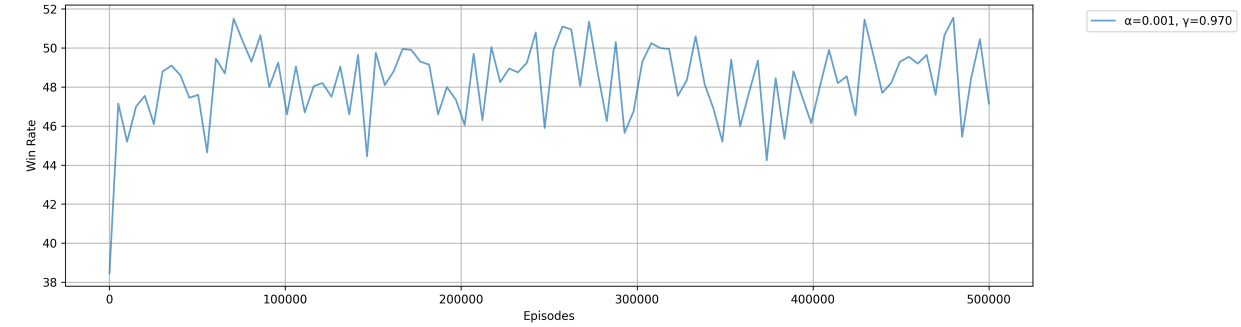
We trained close to 700 different models over 50,000 episodes, yielding the following hyperparameter configuration as "best":

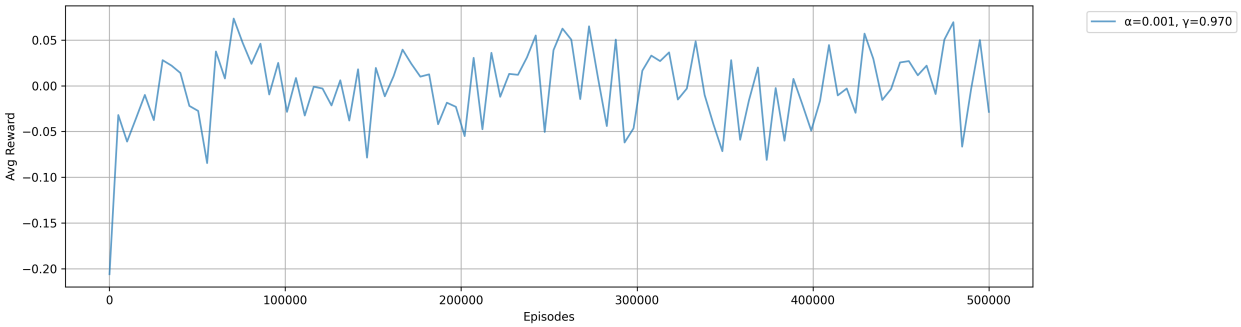
- $\alpha = 0.001$
- Batch Size=64
- Buffer Size=50000
- $\epsilon = 0.05$
- ϵ decay=0.99995
- $\min \epsilon = 0.01$
- $\gamma = 0.97$
- Learning Rate Decay: 0.9999

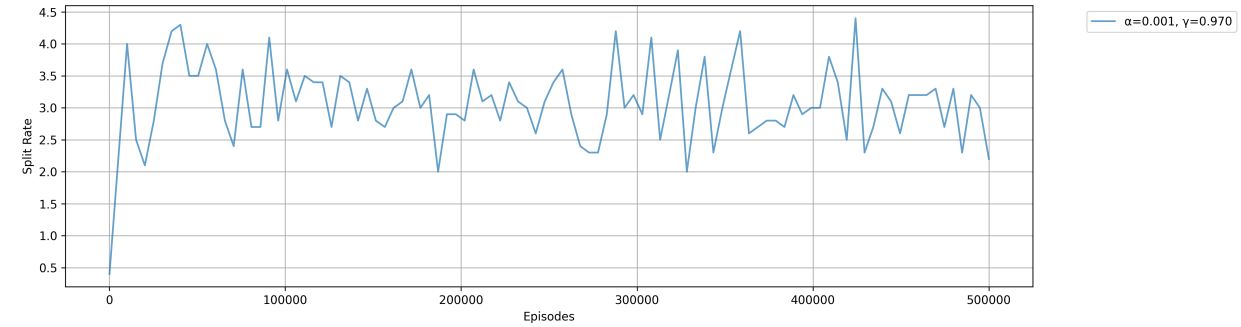
5.2 Training Metrics

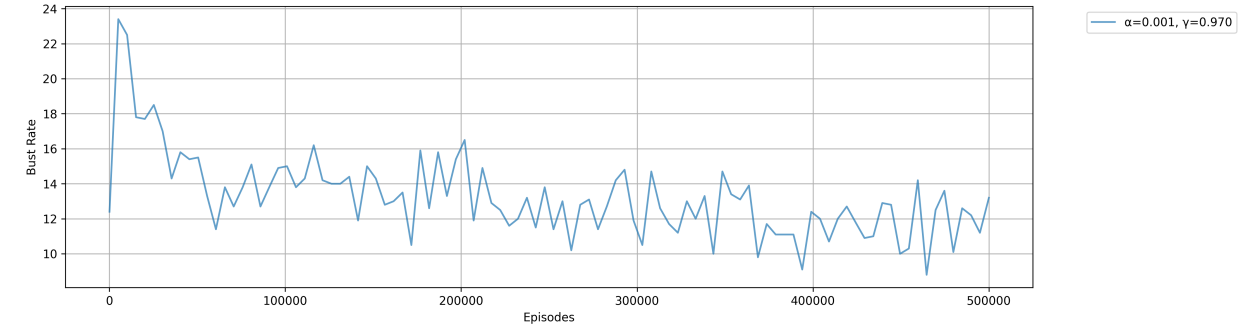
Our final agent was trained over 500,000 episodes, resulting in the following:

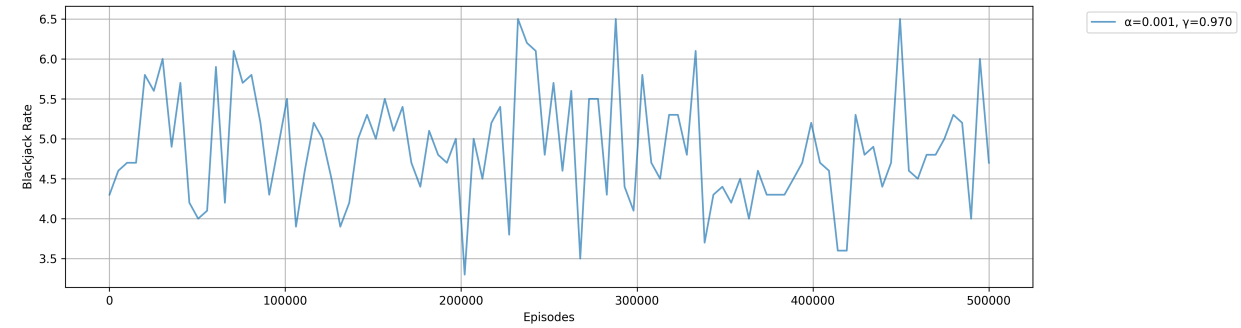
- Win rate: 48.05%
- Average reward per game: -0.003
- Split rate: 3.05%
- Blackjack rate: 5.3%
- Bust rate: 10.95%


Win Rate
Episodes
α=0.001, γ=0.970


Avg Reward
Episodes
α=0.001, γ=0.970


Split Rate
Episodes
α=0.001, γ=0.970


Bust Rate
Episodes
α=0.001, γ=0.970


Blackjack Rate
Episodes
α=0.001, γ=0.970

Upon analysis of the training metrics (Figure 5.2), we came to the following conclusions:

1. **Win Rate Stability**: The win rate stabilized around 48-50%, showing consistent performance against the house edge. This approaches the theoretical maximum for perfect basic strategy play.
2. **Average Reward Convergence**: The average reward per episode converged near zero (-0.003), indicating the agent learned to minimize losses effectively. The low reward variance (1.060491) suggests consistent play rather than high-risk strategies. Consistency is expected of Q-Learning!
3. **Strategic Behavior Evolution**:
   - The split rate settled around 3%, suggesting that splitting could have been "taught" more effectively.
   - The bust rate showed a drastic improvement from initial rates, moving from $> 20\%$ to a final rate of 10.95%
   - The blackjack rate sat around 5.3%, which is to be expected with theoretical probability

5.3 Reviewing the Learned Strategy

Analysis of the final strategy matrices (Figure 1) revealed decision-making patterns that mostly align with basic blackjack strategy.

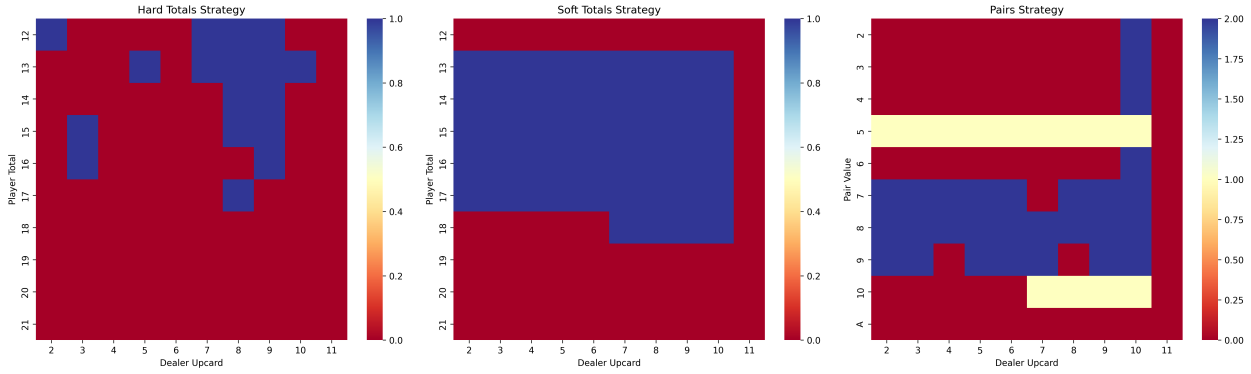


Figure 1: Final learned strategy matrices

5.3.1 Hard Totals Strategy

The hard totals matrix shows clear decision boundaries that follow conventional wisdom:

- Stand (red) dominates for player totals 17 and above
- Hit (blue) is preferred for most totals 16 and below against high dealer cards
- Safe play is adopted against dealer's 2-6, particularly with player totals 13-16

5.3.2 Soft Totals Strategy

The soft totals matrix demonstrates clever handling of ace-containing hands:

- Aggressive hitting on soft totals below 18
- Standing on soft 19 and above
- Strategic hitting on soft 18 against dealer's strong cards (9-A)
- The pattern suggests the agent learned to effectively make use of the flexible nature of aces

5.3.3 Pairs Strategy

The pairs matrix reveals selective splitting behavior that again, is close to optimal strategy:

- Always split aces and eights (blue regions)
- Never split fives and tens (red regions)

The combination of nearly optimal win rate, low bust rate, show that the agent has learned to balance risk and reward effectively while incorporating card counting information to make better decisions than basic blackjack strategy allows.

6 Discussion

6.1 Limitations

6.1.1 Performance Ceiling

Despite using advanced techniques like Double Q-Learning, Prioritized Experience Replay, and adaptive epsilon-greedy exploration, the agent's win rate plateaued/stabilized around 48-50%. This suggests inherent challenges in completely overcoming the house edge in Blackjack. This limiation makes sense with real world experience since Blackjack is inherrently desgined with a built in house advantage, making it challenging to achieve a state where the agernt can beat the casino.The agent does however approach the theoritical maximum for strategic play.

6.1.2 Splitting Strategy

The low split rate (3.05%) indicates that the agent did not fully learn the nuanced strategies for splitting pairs. Future work could focus on more explicitly teaching and incentivizing strategic splitting behaviors such as refining the reward function or training. The current reward structure might not inventizite splitting the pairs as a strageic option and instead shows as a high risk low reward action. Maybe modifying the reward function to inventivize splitting when neccessary can hepl increate the split rate and maybe even increase the win rate the agent has.

6.1.3 Casino Simulation Fidelity

The research environment, while comprehensive, may not perfectly replicate real-world casino conditions. Variations in dealer rules, deck composition, and other subtle factors could impact strategy transferability. We could have inputs to allow the model to train under various dealer rules, to help better accustom it to real world experience and hence find the best strategy that works over multitudes of dealer rules as well as deck compisition to even the mechanism used to shuffle a deck.

6.2 Implications

The research offers several significant implications for reinforcement learning and game strategy: Advanced Learning in Stochastic Environments: This work demonstrates the potential of combining techniques like Double Q-Learning, Prioritized Experience Replay, and adaptive exploration to effectively learn strategies in complex, probabilistic environments such as card games. This work showcases how these techniques work together and where they shine vs where they could use some improvement.

6.2.1 Strategic Learning Beyond Basic Rules

The agent's ability to approximate and sometimes exceed basic Blackjack strategy highlights the power of machine learning to discover nuanced decision-making patterns through iterative learning. The agent can pick the right actions to maximize winning chances based on its previous experience showcases that even though the game is in the House's Favor, strategic play can get close to 50 % win rate using the nuanced techniques.

6.2.2 Card Counting Integration

By incorporating the Hi-Lo card counting system into the state representation, the research showcases how additional contextual information can enhance machine learning performance in strategic games.By having the state space enriched with the bucket count of -1,0,1, the agent was able to expand trhe tradiantional state representation and acces more contexual information to help decide on which actions to take.This additional dimensiality helps transforms the state space from a simple game state description to a more dynamic appraoch.

References

[1] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. In *International Conference on Learning Representations (ICLR)*, 2016.

[2] Scikit-learn developers. Parametergrid. `https://scikit-learn.org/stable/modules/generated/sklearn.model\_selection.ParameterGrid.html`, n.d. Accessed: November 17, 2024.

[3] Michel Tokic. Adaptive epsilon-greedy exploration in reinforcement learning based on value differences. *Machine Learning*, 2011.

[4] Hado van Hasselt. Double q-learning. *Advances in Neural Information Processing Systems*, 23:2613–2621, 2010. Centrum Wiskunde & Informatica.

[5] Christopher J.C.H. Watkins and Peter Dayan. Q-learning. *Machine Learning*, 8:279–292, 1992.