CS230

Learning Explainable Policy For Playing Blackjack Using Deep Reinforcement Learning (Reinforcement Learning)

Ziang Liu
Department of Computer Science
Stanford University
ziangliu@cs.stanford.edu

Gabriel Spil
Department of Computer Science
Stanford University
gspil@stanford.edu

Abstract

Our attempt was to learn an optimal Blackjack policy using a Deep Reinforcement Learning model that has full visibility of the state space. We implemented a game simulator and various other models to baseline against. We showed that the Deep Reinforcement Learning model could learn card counting and exceed standard card counting methodologies and achieved a positive average reward (e.g, "beat the house"). We showed the model was explainable and could be used to derive a human-understandable strategy for Blackjack that exceeds current best practices.

1 Introduction

Blackjack, or twenty-one, is a popular casino card game. Although a near-optimal policy was proposed decades ago, the problem itself remains an interesting testbed for AI algorithms [1]. The goal of the project is to learn Blackjack gameplay policy with Deep Reinforcement Learning. We encode the rule-based optimal strategy with a neural network and learn a policy with vanilla Q-learning without supervision with only information about the current hand of both the player and the dealer. We then extend the state encoding to include information of past hands from the deck and train with Proximal Policy Optimization (PPO) to implicitly learn "card counting", the use of prior knowledge of dealt cards to optimize actions, and thus increase win rate. Finally, we hope to infer human-understandable knowledge from our trained agent to improve human game performance.

2 Related work

2.1 The Blackjack Game

Blackjack is a simple card game where a player plays against the dealer or house. The goal is to get as close to a total of 21 with a set of cards without going over. There are several variations of the rules with regard to valid actions and dealer policy.

2.2 Human Strategies and Learning Methods

Various strategies exist for human players. The best-known strategies include the Basic Strategy, the Hi-Lo Card Counting, and the Hi-Opt 1 card counting. Several works attempt to train Blackjack playing agents using learning-based approaches. Dawson [2] learned to approximate the basic strategy using Monte Carlo reinforcement learning. Granville[3] used traditional Q-learning and was able

CS230: Deep Learning, Autumn 2021, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

to achieve an average return close to the basic strategy. Wu [4] applied DQN with experience relay buffer to Blackjack in order to learn the basic strategy, but was unable to achieve similar performance. Sommerville [5] implemented the Genetic Algorithm, also to learn and approximate the basic strategy table, and was able to generate a strategy highly similar to the ground truth. Vidami et al. [6] learned a real-valued card counting strategy to allow more accurate computation of winning probability.

2.3 Deep Reinforcement Learning and PPO

Standard RL methods have the limit that the memory required to store the states grows exponentially with the state space dimension. Deep RL methods remedy this issue by implicitly representing the value table with a neural network. Proximal Policy Optimization (PPO), proposed by Schulman et al.[7], is a state-of-the-art algorithm used widely in benchmarks due to its simplicity and good sample complexity. Empirically it outperforms other state-of-the-art methods including DQN[8] and A2C[9].

2.4 Our Contributions

Compared to prior work, our main contributions are as follows: 1) Extended state space to enable learning implicit card counting. 2) Benchmark learned policy with best known human strategy (card counting). 3) Investigate the influence of multiple decks and the learning process. 4) Explain learned policy and propose methods to extract human-understandable strategy.

3 Dataset and Features

We have built a Blackjack game simulator that performs deck shuffling, card dealing, dealer policy, and player actions. Using the simulator, we gather 50000 episodes of gameplay, and split into individual intermediate game states as our dataset. For each game state, we encode the player's current hand, the dealer's current hand, and if the player has a usable Ace. We also add the historical deck state to allow for learning card counting.

4 Methods

4.1 The Blackjack RL Environment

In this section, we describe the Blackjack RL environment with regards to states, actions, and rewards. We implemented a Blackjack framework that supports the following features: Early Surrender, Double Down, Dealer Hit on 17, and Dynamic Betting.

4.1.1 Action Space

We support 4 Blackjack actions.

Stand(ST) – The player requests that no more cards be dealt to their hand.

Hit(HT) – The player requests another card.

Surrender(SR) – The player can surrender after the first 2 cards with a loss of 1/2.

Double Down(DD) – The player can double the initial bet after receiving the first 2 cards.

4.1.2 Rewards

The goal of the agent is to maximize episode rewards. By default we model the initial wager as $w = 1.0$. The final reward r is calculated by

$$r = b \times w$$

where b is the bet ratio. If the player loses then the loss is the wager, or $b = -1.0$. If the player surrenders the loss is half the wager, $b = -0.5$. If the player has a Natural Blackjack (Ace and 10 value card) the reward is 1.5 times the wager, or $b = 1.5$. Otherwise if the player wins, $b = 1.0$.

If the player chooses the Double Down action, the initial bet is doubled, thus $b = 2.0$. We have also added the option for dynamic betting in order to support card counting strategies. The player can choose an initial bet of any positive value.

Feature Value	Range
Player Hand	3-31
Dealer Face Card	1-10
Player Usable Ace	0-1

Table 1: Standard State Space Bounds

Feature Value	Range
Aces Dealt	0-4
2-9, one for each	0-4
10 or Face Card Dealt	0-16

Table 2: Additional Extended State Space Bounds

4.1.3 State Space

The minimal attributes are the player's total, the dealer's face card, and if the player has an Ace with optional values 1 or 11. There are 580 permutations. This is a reasonably small number and can be represented in a Q-value table. It is also reasonably represented in lookup tables for various rules-based approaches.

$$|S| = 29 * 10 * 2 = 580$$

We extend the state space to include a count of each card value (Ace, 2-10/Face) observed since the last shuffle. This is represented as an additional 10 features in the state space. For simplicity, we describe the state space for one deck below. (See Table 1 & 2 for details.)

$$|S| = 29 * 10 * 2 * 5 * 5^8 * 17 = 1,093,750,000$$

4.2 Algorithms

4.2.1 The Optimal Strategy and Fixed-Distribution Random Choice

As a simple baseline, we choose a random action with a similar probability to an action chosen by a rules-based strategy. 40% Hit, 40% Stand, 10% Double Down, and 10% Surrender.

4.2.2 Hi-Lo Card Counting with Dynamic Betting

The Hi-Lo method makes decisions based on the probability of drawing high (Ace,10/Face) vs low (2-6) cards. A counter increases when low cards are seen and decreases when high cards are seen. Dynamic Betting increases the initial bet based on the current count.

4.2.3 Neural Network Represented Optimal Strategy

To investigate the representational power of Neural Networks in Blackjack games, we simulated 50000 random states in our Standard State Space, and labeled each state with a corresponding action according to the Optimal Strategy. We then designed a network to learn the state-to-action mapping with the following architecture: Input layer, 10 Neuron ReLU, 20 Neuron ReLU, 10 Neuron ReLU, and Sigmoid as the output layer.

4.2.4 Reinforcement Learning and Deep RL

We use a standard Q-Learning algorithm to learn a policy that maximizes the episode rewards. Each entry in the Q-value table corresponds to one state in the Standard State Space. In the Extended State Space, due to increased complexity, standard Q-Learning is unable to encode the value function in a table. Thus, we employ Proximal Policy Optimization (PPO), which uses internally a Multi-Layer Perceptron architecture to implicitly represent the value function. With the help of deep networks, PPO is able to learn policies on extremely large state spaces that were intractable for standard policy optimization methods.

5 Results

For PPO, we set $\gamma = 0.99$, and first train for 1e4 epochs with $lr = 0.0001$. In experience, this allows the agent to quickly approach a good policy with an average reward $r_{avg} \approx -0.2$. Then, as the rewards curve starts to fluctuate, we train with $lr = 0.00001$ for 1e6 epochs, and $lr = 0.000001$ for 1e7 epochs. Although we don't have theoretical reasoning behind this training scheme, empirically

Method	Avg Reward
Random	-0.483
Q-Learning	-0.296
Rule-Based	-0.058
Hi-Lo	-0.036
NN	-0.008
PPO	**0.0397**

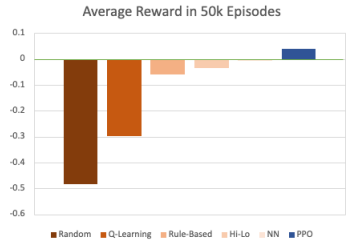


Figure 1 & Table 3: Average reward per episode evaluated with 50,000 game plays against the dealer. PPO with the Extended State Space was the only method that could gain positive reward and exceed the theoretical margin.

it gives the best result. After training, we evaluate each model with 50,000 gameplay episodes and compute the average reward per episode as the metric.

5.1 Average Reward

We computed the average relative reward per episode over 50,000 gameplays. The reward values are the fraction relative to the initial bet value $B = 1.0$, with the exception that $B_n = 1.5$ when the player hits a natural blackjack. Surprisingly, the Hi-Lo Card Counting strategy achieved a negative reward even with information about cards dealt in the past. Our Deep RL method achieved the highest average reward and was the only method to "beat the house" and receive positive earnings. (Figure 1)

		DEALER FACE CARD									
		Ace	2	3	4	5	6	7	8	9	10
	20	ST	ST	ST	ST	ST	ST	ST	ST	ST	ST
	19	ST	ST	ST	ST	ST	ST	ST	ST	ST	ST
P	18	ST	ST	ST	ST	ST	ST	ST	ST	ST	ST
L	17	SR	ST	ST	ST	ST	ST	ST	ST	ST	ST
A	16	SR	ST	ST	ST	ST	ST	HT	HT	SR	SR
Y	15	SR	ST	ST	ST	ST	ST	HT	HT	HT	SR
E	14	SR	ST	ST	ST	ST	ST	HT	HT	SR	SR
R	13	SR	ST	ST	ST	ST	ST	HT	HT	HT	SR
	12	HT	HT	HT	ST	ST	ST	HT	HT	SR	SR
H	11	DD	DD	DD	DD	DD	DD	DD	DD	DD	DD
A	10	HT	DD	DD	DD	DD	DD	DD	DD	DD	HT
N	9	HT	HT	DD	DD	DD	DD	HT	HT	HT	HT
D	8	HT	HT	HT	HT	HT	HT	HT	HT	HT	HT
	7	HT	HT	HT	HT	HT	HT	HT	HT	HT	HT
	6	HT	HT	HT	HT	HT	HT	HT	HT	HT	HT
	5	HT	HT	HT	HT	HT	HT	HT	HT	HT	HT
	4	HT	HT	HT	HT	HT	HT	HT	HT	HT	HT

RULES BASED STRATEGY (NO ACE)

(a) Rule-based Hi-Lo Counting

		DEALER FACE CARD									
		Ace	2	3	4	5	6	7	8	9	10
	20	ST	ST	ST	ST	ST	ST	ST	ST	ST	ST
	19	ST	ST	ST	ST	ST	ST	ST	ST	ST	ST
P	18	ST	ST	ST	ST	ST	ST	ST	ST	ST	ST
L	17	SR	ST	ST	ST	ST	ST	ST	ST	ST	SR
A	16	SR	ST	ST	ST	ST	ST	HT	SR	SR	SR
Y	15	SR	ST	ST	ST	ST	ST	HT	HT	SR	SR
E	14	SR	ST	ST	ST	ST	ST	HT	HT	HT	SR
R	13	SR	HT	HT	HT	ST	ST	HT	HT	HT	HT
	12	SR	HT	HT	HT	HT	HT	HT	HT	HT	HT
H	11	HT	DD	DD	DD	DD	DD	DD	DD	DD	HT
A	10	HT	DD	DD	DD	DD	DD	DD	DD	DD	HT
N	9	HT	DD	DD	DD	DD	DD	DD	DD	HT	HT
D	8	HT	HT	HT	DD	DD	DD	HT	HT	HT	HT
	7	HT	HT	HT	HT	HT	HT	HT	HT	HT	HT
	6	HT	HT	HT	HT	HT	HT	HT	HT	HT	HT
	5	HT	HT	HT	HT	HT	HT	HT	HT	HT	HT
	4	HT	HT	HT	HT	HT	HT	HT	HT	HT	HT

LEARNED STRATEGY (NO ACE)

(b) PPO Learned Policy

Figure 2: Action comparison of Hi-Lo counting vs. learned strategy in different states

5.2 Learned Policy

We generated the action table (Figure 2) from the deep RL learned policy by fixing the state

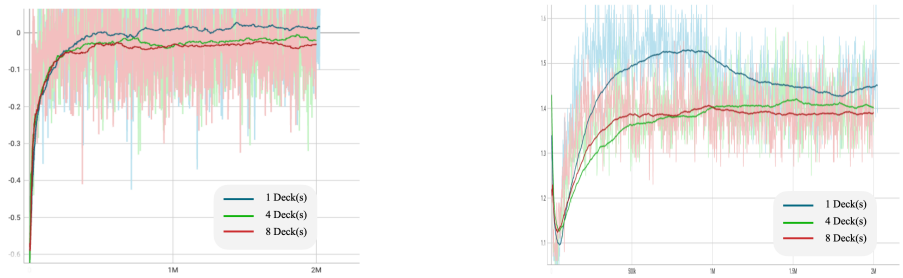
$$s = [p, d, 0, 2, 2, 2, 2, 2, 2, 2, 2, 2, 8]$$

and varying only the player hand value $p \in [4, 20]$ and the dealer face card value $d \in [1, 10]$. The deep RL method learned a policy very similar to Hi-Lo Counting. Out of the total 170 states, the learned strategy has the same action choice in 148 states. Unlike methods in previous work which attempted to learn the mappings of the optimal strategy under supervision, we show that even without supervision, an RL agent can learn to play Blackjack with sufficient experience and even exceed the performance of state-of-the-art baseline algorithms.

5.3 Performance with Additional Decks

In practice, many casinos use more than one deck of cards to discourage and reduce the benefit of card counting. Despite our assumption of only using one deck, we explored the performance of our state space formulation and our method by training with four and eight decks. As shown in Figure 3, the trained policy only achieves a positive average reward with one deck of cards, and the performance decreases as the number of decks increases. Thus, the policy trained on one deck does not generalize well to multiple decks, proving that adding more decks does help reduce the benefit

of card counting. In Figure 3 (b), the algorithm first learns a simple strategy by always choosing to **Stand(ST)**, and thus approaches an average episode length of 1. As the training progresses, the algorithm then chooses to play more aggressively and obtain more cards in each episode, before finally converging to an intermediary strategy that balances risks well.



(a) Average episode reward

(b) Average episode length

Figure 3: Training reward and episode length for different number of decks used in the game. As the number of decks increases, the performance of the policy decreases.

5.4 Human-Understandable Policy Inference

Although deep RL learns a high-performance strategy to play Blackjack, it's not at all obvious how the learned policy looks like. To help understand the learned policy, for each possible player hand value and dealer face card value, we sampled 10,000 random deck states and computed the frequency of each action taken in each state. The center section of Figure 4 shows that our method indeed learns to perform different actions depending on the deck state. To infer a better human-understandable policy for card counting, we propose to treat the count value of each card as an optimization variable, then optimize the L2 loss of action table generated with the current count values and the action table of the learned policy, through 10,000 random instantiations. Due to the scope and time constraint, we leave this to future work.

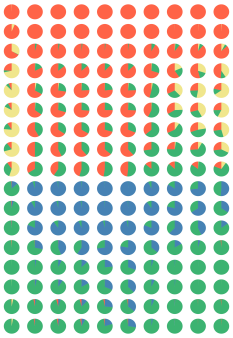


Figure 4: Action frequency of player hand value and dealer face card.

6 Conclusion

In this work, we trained a deep RL agent with PPO without supervision to learn Blackjack. To enable learning implicit card counting, we extended the state space to include information about the cards dealt. Compared to the baselines, ours was the only method to achieve a positive average reward.

For future work, we plan to use a more advanced card counting method as a baseline, train our agent with different number of decks to increase generalizability, and continue our work on inferring better human-understandable card counting strategy through optimization.

7 Contributions

Topic	Contributor
Design, Research and Documentation	Ziang Liu, Gabriel Spil
Initial Blackjack implementation	Ziang Liu, Gabriel Spil
card counting, Double Down, Surrender support	Gabriel Spil
Q-Learning	Ziang Liu
card counting implementations	Gabriel Spil
Neural Network	Gabriel Spil
Deep Learning	Ziang Liu

Table 4: Contributions

References

[1] A. Perez-Uribe and E. Sanchez, "Blackjack as a test bed for learning strategies in neural networks," vol. 3, 06 1998, pp. 2022 – 2027 vol.3.

[2] "Playing blackjack with machine learning." [Online]. Available: https://codebox.net/pages/reinforcement-learning-blackjack

[3] C. de Granville, "Applying reinforcement learning to blackjack using q-learning." [Online]. Available: https://www.cs.ou.edu/~granville/paper.pdf

[4] A. Wu, "Playing blackjack with deep q-learning." [Online]. Available: https://cs230.stanford.edu/files_winter_2018/projects/6940282.pdf

[5] G. Sommerville, "Winning blackjack using machine learning," Feb 2019. [Online]. Available: https://towardsdatascience.com/winning-blackjack-using-machine-learning-681d924f197c

[6] M. Vidámi, L. Szilágyi, and D. Iclanzan, "Real valued card counting strategies for the game of blackjack," in *ICONIP*, 2020.

[7] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *CoRR*, vol. abs/1707.06347, 2017. [Online]. Available: http://arxiv.org/abs/1707.06347

[8] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. A. Riedmiller, "Playing atari with deep reinforcement learning," *CoRR*, vol. abs/1312.5602, 2013. [Online]. Available: http://arxiv.org/abs/1312.5602

[9] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," *CoRR*, vol. abs/1602.01783, 2016. [Online]. Available: http://arxiv.org/abs/1602.01783