The Formulation Of Blackjack As A Markov Decision Process

Jacob Caesar and Stephen Cool

I. Introduction

BLACKJACK is an ubiquitous casino card game that can be traced back to 18th Century France. Blackjack is played between a dealer and a player. The goal of the game is to have the highest card count without going over 21 (e.g. "busting"). The player can add to his total by hitting or staying. Played optimally, Blackjack provides some of the best odds in the casino to the player. However there are 290 possible game states for the player to memorize. The outcome is also stochastic in nature with the dealer having a slight advantage. For centuries, gamblers and researchers have attempted to find optimal strategies to beat the odds and make a Blackjack profitable endeavor. In this paper, we explore the application of reinforcement learning algorithms to determine optimal player strategies and compare those to known expert strategies.

II. Background And Related Work

A. The Blackjack Problem

The objective of each participant (e.g. player, dealer) is to gain the highest card total without exceeding 21 (e.g. "busting"). Each card has a count value; number cards have a count equal to their number and face cards are equal to 10. Aces, however, can be either 1 or 11, whichever yields the highest value without busting. If the player has an ace which can be counted as an 11 without busting the hand is considered "soft" and the ace is considered "useable". If this is not the case, the hand is considered "hard" and there is no "useable" ace. The flow of the game is such:

1) The player places his bet.
2) One face up card and one face down card are dealt to the dealer.
3) Two face up cards are dealt to the player.
4) The player then takes an action of hitting (another card is dealt), staying (player's turn ends), splitting (another hand is played if the player holds a pair, or doubling down.
5) The player continues to play unless he busts or stays. If the player busts, he automatically loses his bet and the round is over.
6) The dealer then plays next if the player has stayed.
7) The dealer must hit until his count is greater than 17 or he busts. If he busts, the dealer loses. [1]
8) If the player has a higher total, he collects his winnings. A tie is called a push and no winnings are collected. If the dealer wins, the player loses his bet. The round is over.

Blackjack can easily be modeled as a Markov Decision Process. For each hand or state, the player is trying to make an optimal choice in the face of an uncertain reward where the state history prior to the current state does not affect the next state. These two details make Reinforcement Learning an ideal method to apply to this problem.

B. Previous Work

Many have approached Blackjack strategies from a variety of perspectives[1]. The most famous group was the MIT Blackjack team that used "card counting" techniques to beat the odds and make millions of dollars from that late 1970's to the early 21st century [2]. This method exploits the finite amount of cards in the deck by looking for times when the deck is potentially "stacked" in the players favor. The player then places large bets during advantageous times and small bets otherwise. Others have explored ways machine learning could be leveraged. Because Blackjack can be modeled as an MDP[3], Q Learning has been used to find optimal strategies that provide similar returns as common strategies [4]. In this paper, Monte Carlo Tree Search, SARSA, SARSA-λ, Q Learning, and Double-Q Learning methods will be explored as approaches to find optimal strategies.

C. Model Based Reinforcement Learning Algorithms

Model based reinforcement learning algorithms attempt to estimate the transition $T(s'|s,a)$ and reward $R(s,a)$ functions to solve the MDP directly. $T(s'|s,a)$ and $R(s,a)$ can be approximated by having the agent interact with the environment by visiting state-action pairs.

Monte Carlo Tree Search (MCTS) is a form of model based, but only needs to generate transition samples by having the agent interact with the environment. MCTS is also a surprisingly effective learner with sparse knowledge of the entire MDP.

[1]There are variations of this rule where the dealer must hit on "soft 17". This slightly increases the house odds and is bad for the player. Player strategies change depending on if this rule is in effect.

D. Model Free Reinforcement Learning Algorithms

Model free methods attempt to learn the action-value function $Q(s,a)$ directly to find an optimal policy without explicit knowledge of $T(s'|s,a)$ and $R(s,a)$. Q-Learning and SARSA are both examples of Model-Free learning algorithms.

The major classification between Q-Learning and SARSA is that they are considered off-policy an on-policy RL algorithms, respectively. Off-policy algorithms are distinguished by using some other exploration strategy in order to estimate the optimal policy. For Q-Learning, $Q(s,a)$ is incrementally estimated and convergences given a suitable exploration strategy. Whereas SARSA is considered on-policy

E. Blackjack Alterations and Assumptions

For the purposes of this paper, a few alterations and assumptions have been made compared to casino Blackjack. Assumptions

1) The dealer draws from an infinite deck.
2) Bets are are always the same for each round.
3) The player action space has been reduced to include just hitting and staying. Doubling down and splitting are not considered.

III. Formulation As MDP

Several simplifications are made in the formulation of our Blackjack MDP to simplify implementation and reduce the size of the state space. Firstly, our state does not consist of the entire content of the player's hand. Instead it keeps track only of the running total and whether or not the player has a useable ace, i.e. whether the current count is "soft" or "hard". If at any point during play, the players count would go over 21 and the player has a useable ace, then the value of that ace is reduced from 11 to 1 and the player's useable ace state transitions to false. Defining an explicit state transition model is non-trivial, so a generative model of Blackjack is formulated instead. The cards are randomly drawn from an infinite deck, with replacement, to simplify implementation of the MDP in software. This eliminates the possibility of card counting and guarantees that the game will, on average, return a negative payout.

Each game begins at state $s_0 = (playertotal = 0, dealershowing = 0, useableace = false)$. After the players first action (stay and hit produce the same outcome for the first turn), the player is dealt two cards randomly from the deck and the dealer is dealt one card visible to the player. The game has now been appropriately initialized and play may begin. The player has two possible actions: stay or hit. The stay action immediately ends play. The players total is frozen and the dealer follows a defined behavior, hitting until they reach a total of 17 or bust. If the dealer total is greater than the player's total (without busting) the dealer wins. If the dealer achieves a natural Blackjack (i.e. a total of 21 in 2 draws) the dealer wins regardless of the player's total. If the dealer and the player have equal counts the game ends in a tie and a reward of 0 is received. The hit action deals the player a new card, randomly selected from the deck, increasing the players total. If the players total goes over 21, the player busts and automatically loses. All player totals over 21 are capped at 22 to further reduce the size of the state space. If the player total is higher than the final dealer total and the player did not themselves bust, then the player wins. The reward for a player win is $+1$, the reward for a player loss is -1 and the reward for a player-dealer tie is 0. After play has completed the state is transitioned to $s_{terminal} = (playertotal = -1, dealershowing = -1, useableace = false)$ which is a terminal, consuming state. In total, there are 292 states and 2 actions summarized below:

$$\mathcal{S} = \{playertotal : \{-1, 0, 4, 5, ..., 20, 21, 22\}, dealershowing : \{ace, 2, 3, ..., 8, 9, 10\}, useableace : \{true, false\}\} \quad (1)$$

$$\mathcal{A} = \{hit, stay\} \quad \mathcal{R} = \{1, 0, -1\} \quad \gamma = 1.0 \quad (2)$$

$$cards = \{ace, 2, 3, 4, 5, 6, 7, 8, 9, 10, 10, 10, 10\} \quad (3)$$

s	a	s'	Description	$R(s,a)$
$(0, 0, false)$	all	$(rand(cards)^2, rand(cards), any(cards_{player}) = ace)$	Player draws 2 cards, dealer draws one card.	0
$(22, any, false)$	all	$(-1, -1, false)$	Player has busted.	-1
$(< 21, any, any)$	hit	$(total + rand(cards), dealershowing, useableace)$	Draw new card, use ace or gain ace based on total and new card.	0
$(< 21, any, any)$	stay	s	Dealer follows set strategy, evaluate win, loss, or tie.	$w : +1, l : -1, d : 0$
$(-1, -1, false)$	any	$(-1, -1, false)$	Terminal state.	0

IV. Solution And Results

Several methods were applied to the Blackjack MDP to determine a strategy for optimal play. These strategies were compared against an "expert" strategy for a game configuration of 4-8 decks[5]. This strategy approximates an optimal strategy for our simplified MDP model. Our generative model formulation rules out methods such as value iteration which require explicit state transition models. We have applied four model free learning algorithms and one model based algorithm.

A. Model Free Reinforcement Learning Algorithms & Parameters

1) SARSA: SARSA is an on-policy reinforcement learning algorithm which attempts to estimate the value of the exploration policy as it follows it. The exploration policy used for our application is an ϵ-greedy policy. The SARSA update equations are as follows[6]:

$$Q(s,a) \longleftarrow Q(s,a) + \alpha[r + \gamma Q(s',a') - Q(s,a)] \tag{4}$$

2) SARSA-λ: SARSA can exhibit slow learning in environments with sparse rewards due to difficulty in assigning credit or blame for a update to the value estimate. SARSA-λ attempts to address this by maintaining a state trajectory history and updating the value estimate through the trajectory history. The backwards propagation of credit or blame is decayed within the trajectory history by λ, the exponential decay parameter. The modified update equations are as follows[6]:

$$\delta = r + \gamma Q(s',a') - Q(s,a) \tag{5}$$

$$Q(s,a) \longleftarrow Q(s,a) + \alpha\delta N(s,a) \tag{6}$$

$$N(s,a) \longleftarrow \gamma\lambda N(s,a) \tag{7}$$

3) Q-learning: Q-learning differs from SARSA in that it directly attempts to estimate the optimal value function Q^*, regardless of the policy being followed. It's update equation is as follows[6]:

$$Q(s,a) \longleftarrow Q(s,a) + \alpha[r + \gamma \operatorname*{argmax}_{a'} Q(s',a') - Q(s,a)] \tag{8}$$

4) Double Q-learning: Q-learning suffers from maximization bias, which is the tendency to overestimate the value of when there is uncertainty in the transition dynamics. Double Q-learning attempts to eliminate the harmful effects of maximization bias by utilizing two estimates of the value function, $Q_1(s,a)$ and $Q_2(s,a)$. These two estimates are updated as follows, where Q_1 and Q_2 are swapped with 50% probability[3]:

$$Q_1(s,a) \longleftarrow Q_1(s,a) + \alpha[r + \gamma Q_2(s', \operatorname*{argmax}_{a'} Q_1(s',a')) - Q_1(s,a)] \tag{9}$$

Algorithm	Q_0	ϵ	α	λ
SARSA	0.0	0.35	0.01	-
SARSA-λ	0.0	0.35	0.01	0.95
Q-learning	0.0	0.35	0.01	-
Double Q-learning	0.0	0.35	0.01	-

B. Model Based Reinforcement Learning Algorithms & Parameters

1) Monte Carlo Tree Search: MCTS is an online planning algorithm characterized by it's combination of Monte Carlo simulation of state trajectories and application of heuristic policy[6]. Our MCTS implementation was permitted 1 second to select an action and utilized a rollout simulation with a heuristic policy of staying on any $playertotal$ greater than 17, and otherwise hitting.

Algorithm	Time (s)	Max Depth	Exploration Constant c	Rollout Policy
MCTS	1	5	4.0	Hit on hard player totals less than 17

C. Summary Of Algorithm Performance

Overall, each tested algorithm yielded a strong although unprofitable Blackjack strategy. Table I summarizes the performance of each algorithm against the reference Expert policy and a Random policy from 10,000 Blackjack simulations. MCTS is the only method that outperformed the Expert policy by slim margin. All algorithms produced strategies far superior to the Random policy which is an expected result.

Figure 1 details the Expert policy based on the players hand and the dealer's showing card. MCTS produced a similar although not identical policy (Figure 2). Besides MCTS, the remaining algorithms performed worse than the Expert policy because of incorrect "stay" actions played when the player counts were small or "hit" actions on high count "soft" hands. The

reason for this is these states were rarely visited by each algorithm during training and further exacerbated by the stochastic and sparse nature of $R(s, a)$. SARSA-λ performed the worst because the sparse and stochastic rewards caused over-weighting of poor outcomes. It is also interesting to note that even with a less optimal strategy like SARSA-λ (Figure 6), the difference in win rate with the Expert policy is only 0.1%.

Figures 8 and 7 depict the learning rates the RL algorithms used compared to the Expert policy with 10k and 100k steps in the environment, respectively. Each shown algorithm reaches an asymptotic policy around 100k steps in the environment suggesting there is much room for improvement with more training. The Expert policy still maintains an advantage over these trained algorithms.

Algorithm	Average Reward $\pm$ 95%	Win Rate (%)
Expert	-0.0434 ± 0.006	43.05
MCTS	-0.0427 ± 0.019	43.64
Q-learning	-0.0485 ± 0.019	43.23
Double Q-learning	-0.0495 ± 0.019	43.10
SARSA	-0.0564 ± 0.019	42.95
SARSA-λ	-0.0647 ± 0.019	42.63
Random	-0.355 ± 0.018	30.09

TABLE I
EXPERT VS RL ALGORITHMS SUMMARY PERFORMANCE

		Expert Policy									
		Dealer Showing									
		A	2	3	4	5	6	7	8	9	10
Player Hand	21	S	S	S	S	S	S	S	S	S	S
	20	S	S	S	S	S	S	S	S	S	S
	19	S	S	S	S	S	S	S	S	S	S
	18	S	S	S	S	S	S	S	S	S	S
	17	S	S	S	S	S	S	S	S	S	S
	16	H	S	S	S	S	S	H	H	H	H
	15	H	S	S	S	S	S	H	H	H	H
	14	H	S	S	S	S	S	H	H	H	H
	13	H	S	S	S	S	S	H	H	H	H
	12	H	H	H	S	S	S	H	H	H	H
	11	H	H	H	H	H	H	H	H	H	H
	10	H	H	H	H	H	H	H	H	H	H
	9	H	H	H	H	H	H	H	H	H	H
	8	H	H	H	H	H	H	H	H	H	H
	7	H	H	H	H	H	H	H	H	H	H
	6	H	H	H	H	H	H	H	H	H	H
	5	H	H	H	H	H	H	H	H	H	H
	4	H	H	H	H	H	H	H	H	H	H
	Soft 20	S	S	S	S	S	S	S	S	S	S
	Soft 19	S	S	S	S	S	S	S	S	S	S
	Soft 18	H	S	S	S	S	S	S	S	H	H
	Soft 17	H	H	H	H	H	H	H	H	H	H
	Soft 16	H	H	H	H	H	H	H	H	H	H
	Soft 15	H	H	H	H	H	H	H	H	H	H
	Soft 14	H	H	H	H	H	H	H	H	H	H
	Soft 13	H	H	H	H	H	H	H	H	H	H
	Soft 12	H	H	H	H	H	H	H	H	H	H

Fig. 1. Policy of Expert

		MCTS Learning Policy									
		Dealer Showing									
		A	2	3	4	5	6	7	8	9	10
Player Hand	21	S	S	S	S	S	S	S	S	S	S
	20	S	S	S	S	S	S	S	S	S	S
	19	S	S	S	S	S	S	S	S	S	S
	18	S	S	S	S	S	S	S	S	S	S
	17	S	S	S	S	S	S	S	S	S	S
	16	H	S	S	S	S	S	H	H	H	S
	15	H	S	S	S	S	S	H	H	H	H
	14	H	S	S	S	S	S	H	H	H	H
	13	H	S	S	S	S	S	H	H	H	H
	12	H	S	S	S	S	S	H	H	H	H
	11	H	H	H	H	H	H	H	H	H	H
	10	H	H	H	H	H	H	H	H	H	H
	9	H	H	H	H	H	H	H	H	H	H
	8	H	H	H	H	H	H	H	H	H	H
	7	H	H	H	H	H	H	H	H	H	H
	6	H	H	H	H	H	H	H	H	H	H
	5	H	H	H	H	H	H	H	H	H	H
	4	H	H	H	H	H	H	H	H	H	H
	Soft 20	S	S	S	S	S	S	S	S	S	S
	Soft 19	S	S	S	S	S	S	S	S	S	S
	Soft 18	S	S	S	S	S	S	S	S	H	S
	Soft 17	H	H	H	H	H	H	H	H	H	H
	Soft 16	H	H	H	H	H	H	H	H	H	H
	Soft 15	H	H	H	H	H	H	H	H	H	H
	Soft 14	H	H	H	H	H	H	H	H	H	H
	Soft 13	H	H	H	H	H	H	H	H	H	H
	Soft 12	H	H	H	H	H	H	H	H	H	H

Fig. 2. Policy of MCTS

		Q Learning Policy									
		Dealer Showing									
		A	2	3	4	5	6	7	8	9	10
Player Hand	21	S	S	S	S	S	S	S	S	S	S
	20	S	S	S	S	S	S	S	S	S	S
	19	S	S	S	S	S	S	S	S	S	S
	18	S	S	S	S	S	S	S	S	S	S
	17	S	S	S	S	S	S	S	S	S	S
	16	H	S	S	S	S	S	S	H	H	S
	15	H	S	S	S	S	S	H	H	H	S
	14	H	S	S	S	S	S	S	H	H	H
	13	H	H	S	S	S	S	H	H	H	H
	12	H	S	H	S	S	S	H	H	H	H
	11	H	H	H	H	H	H	H	H	H	H
	10	H	H	H	H	H	H	H	H	H	H
	9	H	H	H	H	H	H	H	H	H	H
	8	H	H	H	H	H	H	H	H	H	H
	7	H	H	H	H	H	H	H	H	H	H
	6	H	H	H	H	H	H	S	H	H	H
	5	H	H	H	H	H	H	H	H	H	H
	4	H	H	H	H	H	S	H	H	H	H
	Soft 20	S	S	S	S	S	S	S	S	S	S
	Soft 19	S	S	S	S	S	S	S	S	S	S
	Soft 18	H	S	S	S	S	S	S	S	S	H
	Soft 17	H	S	H	H	H	S	H	H	H	H
	Soft 16	H	H	H	H	S	H	H	H	H	H
	Soft 15	H	H	H	H	H	H	H	H	H	H
	Soft 14	H	H	H	H	H	H	H	H	H	H
	Soft 13	H	H	H	H	H	H	H	H	H	H
	Soft 12	H	H	H	H	H	H	H	H	H	H

Fig. 3. Policy after 100k episodes of Q-learning

		Double Q Learning Policy									
		Dealer Showing									
		A	2	3	4	5	6	7	8	9	10
Player Hand	21	S	S	S	S	S	S	S	S	S	S
	20	S	S	S	S	S	S	S	S	S	S
	19	S	S	S	S	S	S	S	S	S	S
	18	S	S	S	S	S	S	S	S	S	S
	17	S	S	S	S	S	S	S	S	S	S
	16	H	S	S	S	S	S	H	S	H	H
	15	H	S	S	S	S	S	H	H	H	H
	14	H	H	S	S	S	H	H	H	H	S
	13	H	S	H	H	H	H	H	H	H	H
	12	H	H	H	H	H	H	H	H	H	H
	11	H	H	H	H	H	H	H	H	H	H
	10	H	H	H	H	H	H	H	H	H	H
	9	H	H	H	H	H	H	H	H	H	H
	8	H	H	H	H	H	H	H	H	H	H
	7	H	H	H	H	H	H	H	H	H	H
	6	H	H	H	H	H	H	H	H	H	H
	5	H	H	H	H	H	H	H	H	S	H
	4	H	H	H	H	H	H	H	H	H	H
	Soft 20	H	S	S	S	S	S	S	S	S	S
	Soft 19	S	H	S	S	S	S	S	S	H	S
	Soft 18	H	S	H	H	H	S	S	H	S	H
	Soft 17	H	H	H	H	H	H	H	H	H	H
	Soft 16	H	H	H	H	H	H	H	H	H	H
	Soft 15	H	H	H	H	H	H	H	H	H	H
	Soft 14	H	H	H	H	H	H	H	H	H	H
	Soft 13	H	H	H	H	H	H	H	H	H	H
	Soft 12	H	H	H	H	H	H	H	H	H	H

Fig. 4. Policy after 100k episodes of Double Q-learning

		SARSA Learning Policy									
		Dealer Showing									
		A	2	3	4	5	6	7	8	9	10
Player Hand	21	S	S	S	S	S	S	S	S	S	S
	20	S	S	S	S	S	S	S	S	S	S
	19	S	S	S	S	S	S	S	S	S	S
	18	S	S	S	S	S	S	S	S	S	S
	17	H	S	S	S	S	S	S	S	S	S
	16	S	S	S	S	S	S	H	H	H	H
	15	H	S	S	S	S	S	H	H	S	H
	14	H	S	S	S	S	S	H	H	H	H
	13	H	S	S	H	S	S	H	H	H	H
	12	H	H	H	H	H	H	H	H	H	H
	11	H	H	H	H	H	H	H	H	H	H
	10	H	H	H	H	H	H	H	H	H	H
	9	H	H	H	H	H	H	H	H	H	H
	8	S	H	H	H	H	H	H	H	H	H
	7	H	H	H	H	S	H	H	H	H	H
	6	H	H	H	H	H	S	H	S	H	H
	5	H	H	H	H	H	H	H	H	H	H
	4	H	H	H	H	H	H	H	H	H	H
	Soft 20	S	S	S	S	S	S	S	S	S	S
	Soft 19	S	S	S	S	S	S	S	S	S	S
	Soft 18	H	S	S	S	S	S	S	S	H	H
	Soft 17	H	H	H	H	S	H	H	H	H	H
	Soft 16	H	H	H	H	H	H	H	H	H	H
	Soft 15	H	H	H	H	H	H	H	H	H	H
	Soft 14	H	H	H	H	S	H	H	H	H	H
	Soft 13	H	H	H	H	H	H	H	H	H	H
	Soft 12	H	H	H	H	H	H	H	H	H	H

Fig. 5. Policy after 100k episodes of SARSA

		SARSA-λ Learning Policy									
		Dealer Showing									
		A	2	3	4	5	6	7	8	9	10
Player Hand	21	S	S	S	S	S	S	S	S	S	S
	20	S	S	S	S	S	S	S	S	S	S
	19	S	S	S	S	S	S	S	S	S	S
	18	S	S	S	S	S	S	S	S	S	S
	17	S	S	S	S	S	S	S	S	S	S
	16	H	S	S	S	S	S	S	H	S	S
	15	H	S	S	S	S	S	H	S	H	H
	14	H	H	S	S	S	S	S	H	S	S
	13	H	S	S	H	S	S	H	H	H	S
	12	H	S	S	S	H	S	H	H	H	H
	11	H	H	H	H	H	H	H	H	H	H
	10	H	H	S	H	H	H	H	H	H	H
	9	S	H	H	H	S	S	H	H	H	H
	8	S	S	H	H	S	H	H	H	S	H
	7	H	H	S	H	H	H	H	H	H	H
	6	S	H	S	H	S	H	H	S	H	S
	5	H	S	S	S	S	S	H	S	H	H
	4	S	H	H	H	H	H	H	H	H	S
	Soft 20	S	S	S	S	S	S	S	S	S	S
	Soft 19	S	S	H	S	S	S	S	S	S	S
	Soft 18	H	S	H	S	S	S	H	H	S	H
	Soft 17	S	S	H	H	H	S	S	H	H	H
	Soft 16	S	S	S	H	H	S	S	H	H	H
	Soft 15	H	S	S	H	S	H	H	H	H	H
	Soft 14	S	H	H	S	H	H	H	S	S	H
	Soft 13	S	H	H	H	S	H	H	H	H	H
	Soft 12	H	H	S	H	H	H	S	H	H	H

Fig. 6. Policy after 100k episodes of SARSA-λ

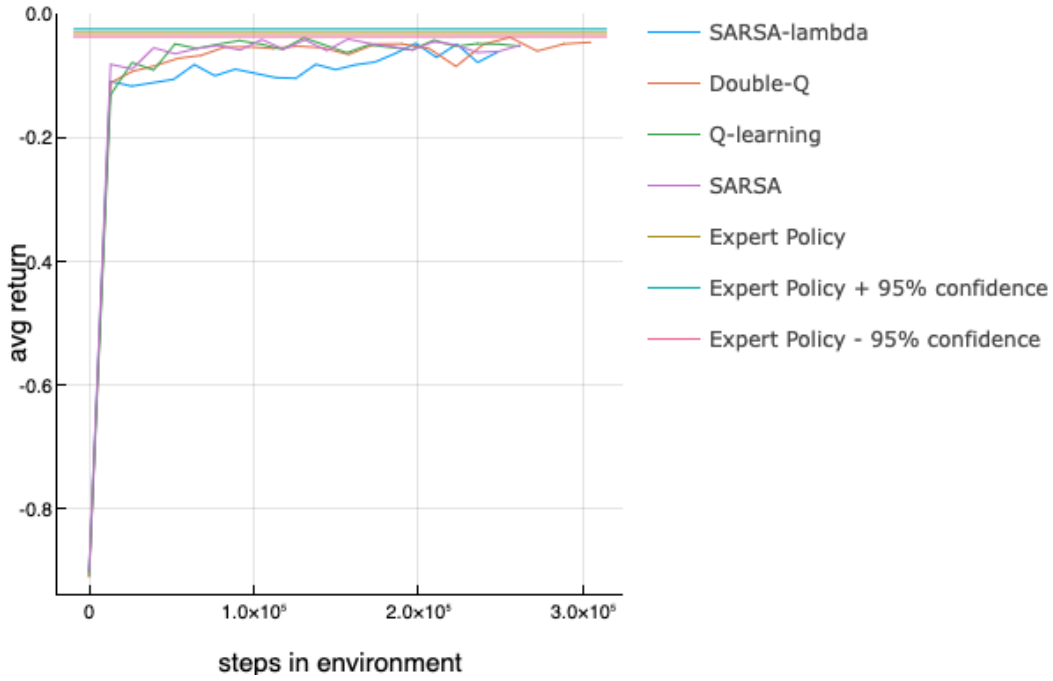

Fig. 7. Training results after 100k episodes

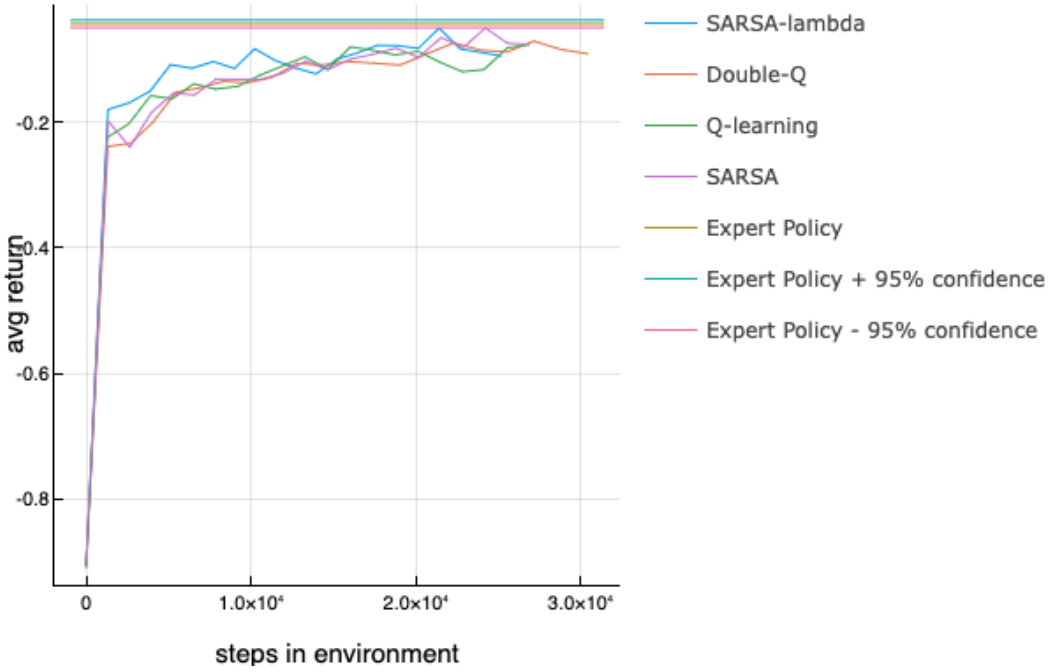

Fig. 8. Training results after 10k episodes

V. Conclusion

Multiple methods to determine optimal strategies of Blackjack were presented. Only MCTS yielded a superior strategy over the Expert strategy by a slim margin, but the remaining methods all had win rates within $<1\%$ and surpassed a Random strategy by $>12\%$. Future work could include the application of other methods such as tabular reinforcement learning and DQN. DQN may be especially well suited because similar hands should have similar state-action value functions, thus reducing the required sampling of very low probability hands, such as hands with low value player total. A more advanced MDP formulation with a finite deck, doubling, splitting, bet amounts, and tracking of individual cards could be added to compare to card counting and increasing the possibility of beating the house odds.

Appendix

Member Contributions

Jacob Caesar contributed to the report and coded the MDP and solver algorithms.
Stephen Cool contributed to the report and developed data visualizations. He also tested and verified the software MDP.

Release Statement

The authors grant permission for this report to be posted publicly.

Source Code

Source code is available at: https://github.com/jakemantiv/blackjackAI

REFERENCES

[1] E. O. Thorp, Beat the dealer. New York, NY: Random House, 1966.
[2] Jensen, Kamron, "The Expected Value of an Advantage Blackjack player" (2014). All Graduate Plan B and other Reports. 524.
[3] R. S. Sutton and A. G. Barto, "Example 5.3: Solving Blackjack," in Reinforcement learning: An introduction, 2nd ed., Cambridge, MA: The MIT Press, 2020, pp. 99–100.
[4] C. de Granville, "Applying Reinforcement Learning to Blackjack Using Q-Learning," https://www.cs.ou.edu/ granville/paper.pdf.
[5] M. Shackleford, "4-deck to 8-deck blackjack strategy," Wizard of Odds, 07-Mar-2021. [Online]. Available: https://wizardofodds.com/games/blackjack/strategy/4-decks/. [Accessed: 30-Apr-2022].
[6] M. J. Kochenderfer, T. A. Wheeler, and K. H. Wray, Algorithms for decision making. Cambridge, MA: Massachusetts Institute of Technology, 2022.