Impact of deck size Q-Learning: Blackjack

Avish Buramdoyal
Supervisor: Assoc. Prof Tim Gebbie

Department of Statistical Science
University of Cape Town

November 3, 2020

Table of Contents

1. Introduction
2. Blackjack
3. Optimal strategy
4. Reinforcement Learning and blackjack
5. MC performance comparison
6. Temporal difference and Q
7. Q-Learning: Hyperparameter fine-tuning
8. Q-Learning performance
9. Perfect card counter
10. Further extensions
11. Appendix
12. Q & A

Introduction

Title

Impact of deck size Q-Learning blackjack

Objectives

1. Implement a Q-learning algorithm, and reinforcement learn a simulated blackjack game.
2. Visualise and investigate the rate-of-learning convergence for the algorithms as a function of simulation size, deck size and number of players .
3. Consider possible extensions to base Q-learning algorithm to enhance convergence for a large number of decks.
4. Compare performance of a single RL agent to the multi-RL agents based off house edge.

Blackjack game

Game settings

1. Players compete against the house rather than each other.
2. Game consists of a dealer, 1-7 players and played with 1-8 decks.
3. Cards 2 to 10: worth their face values. Jacks, Queens and Kings: counted as 10 and ace: worth 1 or 11.
4. Ace valued as 11 is a "soft hand" and all other hands are considered "hand hands".
5. Objective of game: Get a score as close or equal to 21 without busting[1].
6. Blackjack: A starting hand of an Ace and a 10-valued card.

The deal

1. Each player places an initial bet.
2. Players are next dealt with two cards face up each and the dealer also gets two cards, one face up and one face down (called a hole card).

[1] busting: exceeding a score of 21

Blackjack game

Actions

1. Possible actions at play include standing, hitting, splitting, doubling down and insurance.
2. Hitting (H): Player asks for another card from the dealer.
3. Standing (S): Player no additional cards.
4. Doubling down (D): Player doubles his initial bet in the middle of a hand on the condition that he draws one and only one more card.
5. Splitting: Player separates two cards of the same face value and make another bet of equal size as the initial bet made and play each card as a separate hand.
6. Dealer can be bounded to either hit or stand on a soft 17 depending on the rule set by the casino.

Blackjack game

Settlement

1. A tie: No money exchanged if both the player and the dealer have the same total hand value.
2. The one with the higher total hand value wins the bet if neither the player nor the dealer busts.
3. The player wins the bet if the dealer busts and the player's total is below 21 and vice-versa.
4. If the player wins (other than from a blackjack), he receives a reward equal to his initial size bet.
5. Assuming blackjack pays 3:2, the player receives 1.5 times his initial bet for having a blackjack and the dealer having a less favorable score.
6. If the dealer wins (irrespective from a blackjack or not), the player pays out his total bet made in that round.

Optimal playing strategy

Optimal actions

1. Challenge posed to the player is choosing the optimal action at each hand, given his current total and the dealer's face up card.
2. We follow the blackjack basic strategy, initially determined by [Baldwin et al., 1956] which is simply a proper playing decision for every possible hand against the dealer.

	Dealer's Card		
Player total	2	3	4
4-8	H	H	H
9	H	D	D
10	D	D	D

Figure: Strategy Table

Optimal playing strategy

Optimal bets

1. Harvey Dubner introduced a simplified variation of Thorp's strategy, called the "Hi-Lo" card counting strategy, which we incorporate in one of the Q-Learning formulations.

Hi-Lo system framework

Card	2,3,4,5,6	7,8,9	A, K, Q J, 10
Value Assigned	1	0	-1

Table: Harvey Dunbner's Hi-Lo system

1. The player starts with a count of 0 and based on the above indices, he will add or subtract for every single card revealed and store this cumulative sum for each round, known as the running count.
2. $\text{True Count} = \frac{\text{Running Count}}{\text{Decks Remaining}}$ is then computed

The general idea is to bet little or nothing when player advantage is low and to bet proportionately high when player advantage is high, i.e. in multiples of the player's betting unit.

RL problem to blackjack

1. RL problem involves relying on responses from the environment to learn and more specifically to map situations to actions. [S.Sutton and Barto, 2014].
2. Playing blackjack is naturally formulated as an episodic finite MDP.
3. Each game of blackjack is an episode. Rewards of +1, 1, and 0 are given for winning, losing, and drawing, respectively. .
4. All rewards within a game are zero, and are therefore not discounted.
5. The states depend on the player's cards and the dealer's showing card.

Reinforcement Learning

Markov Decision Processes (MDP)

1. We aim at estimating "how good" it might be for an agent to be in a given state or for taking performing an action in any particular state, i.e. "how good" it might be for an agent to take an action given his hand total and the dealer's face up card.
2. The estimates of "how good" it might be for an agent to either be in a state or take an action will be estimated from experience using Monte Carlo methods.

State-value function

1. The idea of "how good" is captured through the expected returns of future rewards for the agent and is represented by the value function $\mathrm{v}_\pi(\mathrm{s})$. $\mathrm{v}_\pi(\mathrm{s})$ represents the value of a state s under policy π and is given by:

$$\mathrm{v}_\pi(\mathrm{s}) = \mathbb{E}_\pi[\mathrm{G_t} \mid \mathrm{S_t} = \mathrm{s}] = \mathbb{E}_\pi[\sum_{\mathrm{k}=0}^{\infty} \gamma^{\mathrm{k}} \mathrm{R_{t+k+1}} \mid \mathrm{S_t} = \mathrm{s}] \tag{1}$$

$\mathrm{G_t}$ is the reward sequence of the agent, given by the sum of all rewards $\mathrm{R_t}$ for each time step. γ^k represents the discounting factor.

Action-value function

1. We can also compute the value of performing action a, being in state s under policy π, represented by the action-value function. $\mathrm{q}_\pi(\mathrm{s}, \mathrm{a})$ represents the expected return of starting from state s, performing action a and subsequently following policy π and is given by:

$$\mathrm{q}_\pi(\mathrm{s}, \mathrm{a}) = \mathbb{E}_\pi[\mathrm{G_t} \mid \mathrm{S_t} = \mathrm{s}, \mathrm{A_t} = \mathrm{a}] = \mathbb{E}_\pi[\sum_{\mathrm{k}=0}^{\infty} \gamma^{\mathrm{k}} \mathrm{R_{t+k+1}} \mid \mathrm{S_t} = \mathrm{s}, \mathrm{A_t} = \mathrm{a}] \tag{2}$$

Optimal Value functions

1. Optimal state-value function and optimal action-value function are defined as $v_*(s)$ and q_* respectively, and given by:

$$v_*(s) = \max_\pi \left\{ v_\pi(s) \right\} \text{ for all } s \in \mathcal{S} \tag{3}$$

$$q_*(s, a) = \max_\pi \left\{ q_\pi(s, a) \right\} \text{ for all } s \in \mathcal{S} \tag{4}$$

Optimal policy

1. Using the optimal v_*, actions appearing best after a one-step search will be optimal ones.
2. With q_*, no one-step ahead search is required from the agent. The action-value function $q_*(s, a)$ effectively stores results of all one-step ahead searched and the agent can simply find that action maximizing $q_\pi(s, a)$ for any given states.

Reinforcement Learning

Monte Carlo methods

1. Monte Carlo methods only requirement is experience, i.e. sample sequences of actions, states and rewards from an actual or simulated environment.
2. Value estimates and policies are only changed upon the completion of an episode.
3. To estimate the value states, we could adopt either the Monte Carlo on-policy or Monte Carlo off-policy.
4. Any state s can be visited multiple times for the same episode but we adopt the first-visit MC to formulate the blackjack problem.
5. A major problem here is that many state-action pairs may never be visited and learning is thus affected as all states cannot be explored. Episodes are therefore generated with exploring starts[2].
6. The agent has to therefore make sure to continue selecting states over time and we therefore adopt the MC on-policy and MC off-policy approach to learn the value states.

Monte Carlo policy improvement

1. Policy improvement is done by making the policy greedy with respect to the current value function such that:

$$\pi(\mathrm{s}) = \arg\max_{\mathrm{a}} \mathrm{q}(\mathrm{s}, \mathrm{a}) \tag{5}$$

[2] exploring starts: each episode begins with a randomly chosen state and action and then follows the current policy

Monte Carlo methods

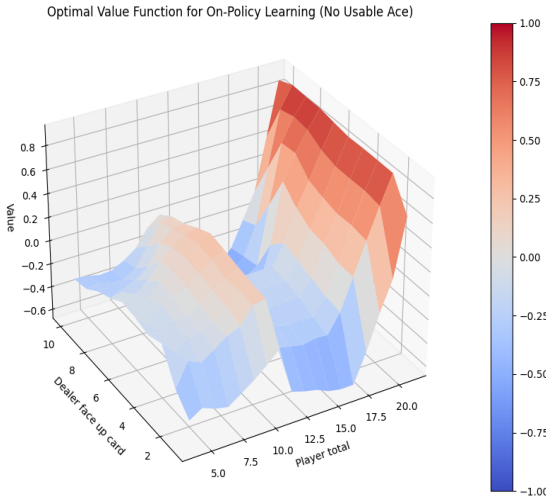
MC on-policy/off-policy and environment

1. Multiplayer game with an infinite deck
2. Actions: Hitting and standing only
3. Target policy: Policy we are following/learning is an epsilon greedy policy[3] that becomes the optimal policy
4. Behavior policy: Policy that is more exploratory and used to generate behavior and used under MC off-policy.
5. We adapt a technique called importance sampling to estimate the Q-function for the target policy π for which the agent behaves using a different policy, the behavior policy μ.
6. Number of samples (rounds): 1000
7. Number of simulations: 100, 000

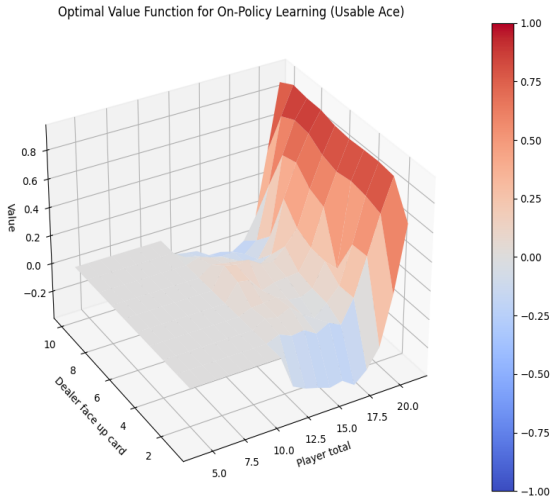
[3] epsilon greedy policy:not only guarantees the maximum probability of the optimal action, but also allows the probability of taking other actions to be a small non-zero value. This ensures the exploration of unknown states.

Monte Carlo performance comparison

MC on-policy: Value states learnt



(a) MC on-policy (No UA)

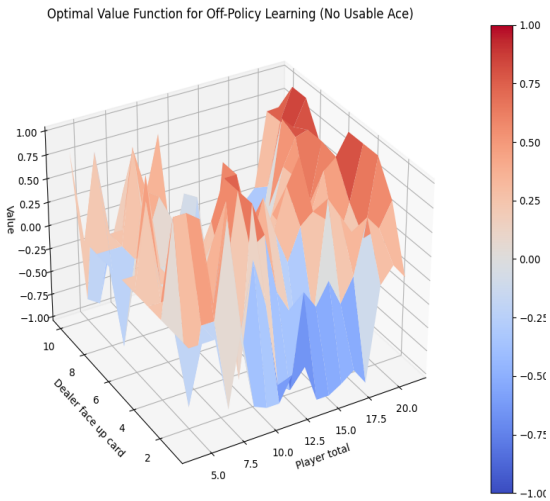


(b) MC on-policy (UA)

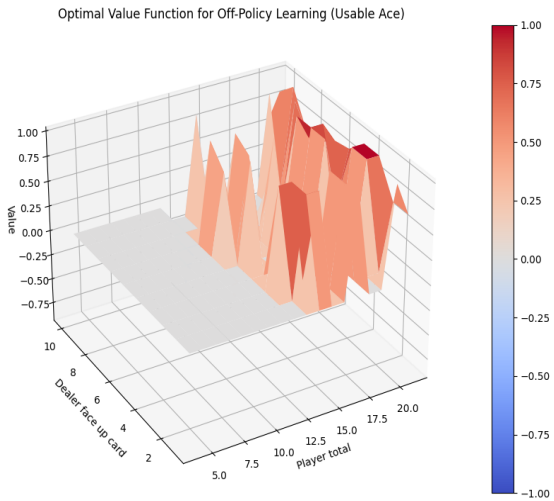
As expected, when a player has value 20 or 21, which is at the rear of the plot, one is more likely to win indicated by higher value states. For the case of UA, states for which scores are beyond a total of 13 are mostly explored.

Monte Carlo comparison

MC off-policy: Value states learnt



(a) MC off-policy (No UA)



(b) MC off-policy (UA)

Similar convergence in terms of where the high and low value states are located. For the case of UA however, only states for which the player's total higher than 15 are explored.

Average payoff

An average payoff of -131.68 was achieved under the off-policy compared to an average payoff of -100.25 under the on-policy.

Temporal difference

TD Prediction

The TD(0) control algorithm implementation by [S.Sutton and Barto, 2014] is an approach to learning how to predict a quantity that depends on future values of a given signal [Paul", 2019] and unlike MC methods only need to wait only the next time step and not the end of an episode to update $\mathrm{V}(\mathrm{S_t})$, TD(0) does not.

One step q-learning

One of the most important breakthroughs in RL was the development of an off-policy TD control algorithm known as Q-Learning Sutton. The simplest form of q-learning is a one-step Q-Learning as given by:

$$\mathrm{Q_t}(\mathrm{s_t}, \mathrm{a_t}) \Leftarrow \mathrm{Q_t}(\mathrm{s_t}, \mathrm{a_t}) + \alpha \left[\mathrm{R_t} + \gamma \max_{\mathrm{a'}} \mathrm{Q_{t+1}}\left(\mathrm{s_{t+1}}', \mathrm{a'}\right) - \mathrm{Q_{t+1}}(\mathrm{s_{t+1}}, \mathrm{a_t}) \right] \tag{6}$$

α is the learning rate, allowing for the determination of the update made on each time-step and γ is the discount rate, allowing for the determination of the value of future rewards.

The learned action-value function, Q, directly approximates $\mathrm{q_*}$, independent of the policy being followed which dramatically simplifies our analysis and fasten convergence.

Q-Learning: Hit and stick only

Q-Learning: Hit and stick only

1. Multiplayer game with an infinite deck
2. Actions: Hitting and standing only
3. Decaying epsilon greedy policy [4] adopted.
4. Number of samples (rounds): 1000
5. Number of simulations: 100, 000

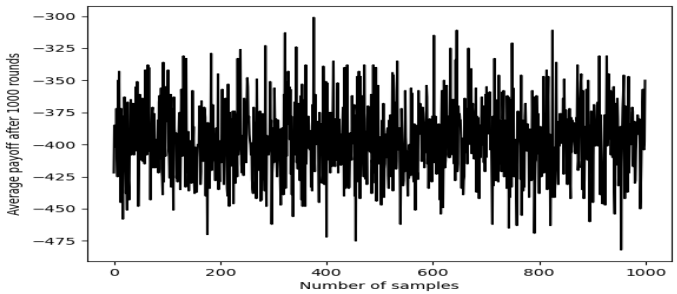
Fine-tuning parameters

1. The "number of episodes to train" parameter determines the rate of decay of parameter ϵ.
2. ϵ initially set to a maximum of 1.
3. ϵ drops to 90% of its initial value in the first 30% of the number of episodes to train, then drops to 10% of its initial value in the next 40% of the number of episodes to train and finally becomes 0 in the final 30% of the number of episodes to train.
4. α chosen to be a value 0.5.
5. γ chosen to be 0.1 to keep the agent short-sighted.

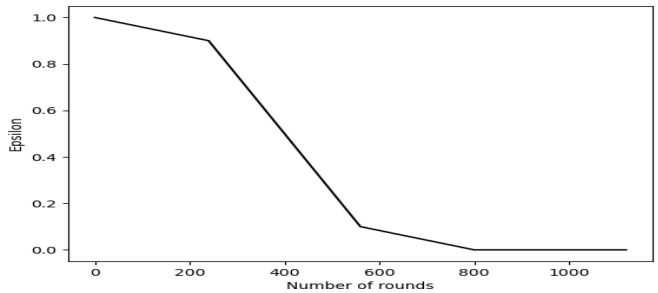
[4] Decaying epsion greedy policy: policy used to balance exploration and exploitation in many reinforcement learning settings.

Q-Learning: Hit and stick only

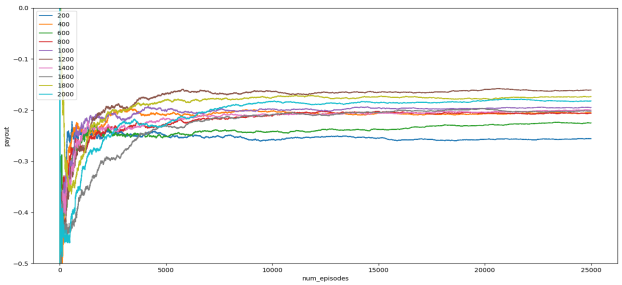
Q-learning performance



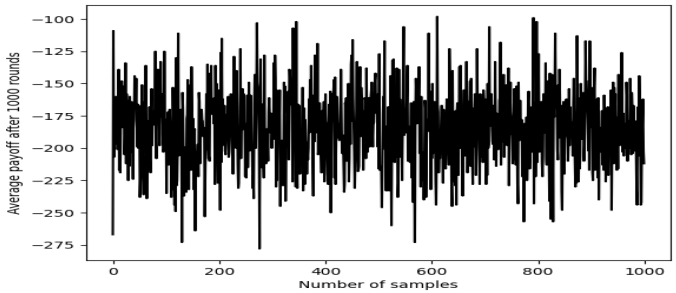
(a) Average payoff (Random actions): -395.93.



(b) Validation analysis: Epsilon decay



(c) Optimal number of episodes: 800



(d) Average payoff (Final model): -198.54

Q-Learning: Hit and stick only

Analyzing strategy learnt

	Dealer's Face up card (No UA)										Dealer's Face up card (UA)									
Player Total	1	2	3	4	5	6	7	8	9	A	1	2	3	4	5	6	7	8	9	A
1	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
2	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
3	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
4	*	H	H	*	H	S	H	H	S	H	*	*	*	*	*	*	*	*	*	*
5	S	H	H	H	H	S	*	S	S	H	*	*	*	*	*	*	*	*	*	*
6	H	S	H	H	S	S	H	H	S	H	*	*	*	*	*	*	*	*	*	*
7	H	S	S	S	S	H	S	H	S	H	*	*	*	*	*	*	*	*	*	*
8	H	H	H	H	S	H	S	S	S	H	*	*	*	*	*	*	*	*	*	*
9	H	H	S	H	S	S	H	H	H	H	*	*	*	*	*	*	*	*	*	*
10	H	H	S	H	S	H	H	S	H	H	*	*	*	*	*	*	*	*	*	*
11	H	H	H	H	H	S	H	H	H	H	*	*	*	*	*	*	*	*	*	*
12	H	H	H	H	H	S	S	H	S	H	H	*	*	H	S	H	H	S	*	H
13	H	S	H	H	H	H	H	H	H	H	S	H	H	H	S	H	S	H	S	H
14	S	H	H	S	S	S	H	H	S	H	S	H	H	H	H	S	H	H	S	H
15	H	H	S	H	S	S	S	S	H	H	H	S	H	H	H	H	S	S	H	H
16	S	H	S	S	S	H	S	H	H	H	S	H	H	H	H	H	S	S	H	H
17	H	H	H	S	S	H	H	H	H	H	S	H	H	H	H	S	S	S	H	S
18	H	S	S	H	S	S	S	S	H	S	H	H	H	H	H	S	H	S	H	S
19	S	S	S	S	S	S	S	S	S	S	H	H	S	S	H	S	S	H	S	H
20	S	S	S	S	S	S	S	S	S	S	H	S	S	H	S	H	H	S	S	H
21	H	S	S	S	S	S	S	S	S	S	S	S	S	H	S	S	S	S	H	S

Table: Strategy Table (Hit and stick)

The agent seems to have learned when to 'hit' when it's hand is not close to higher hand totals. The agent however seems to favour 'sticking' a lot possibly to avoid going a bust. This is a good strategy.

Q-Learning: All actions implemented correctly

Analyzing strategy learnt

Player Total	Dealer's Face up card									
HARD	2	3	4	5	6	7	8	9	10	11
4	H	H	H	H	H	H	H	H	H	H
5	H	H	H	H	H	H	H	H	H	H
6	H	H	H	H	H	H	H	H	H	H
7	H	H	H	H	H	H	H	H	H	H
8	H	H	H	H	H	H	H	H	H	H
9	H	D	D	D	D	D	D	H	H	H
10	D	D	D	D	D	D	D	D	H	H
11	D	D	D	D	D	D	D	D	D	H
12	H	H	H	H	D	H	H	H	H	H
13	H	H	H	S	S	H	H	H	H	H
14	S	S	S	S	S	H	H	H	H	H
15	S	S	S	S	S	H	H	H	H	H
16	S	S	S	S	S	H	H	H	H	H
17	S	S	S	S	S	S	S	S	S	H
18	S	S	S	S	S	S	S	S	S	S
19	S	S	S	S	S	S	S	S	S	S
20	S	S	S	S	S	S	S	S	S	S
21	S	S	S	S	S	S	S	S	S	S
SOFT										
13	H	H	D	H	H	H	H	H	H	H
14	H	H	H	D	D	H	H	H	H	H
15	H	H	H	H	D	H	H	H	H	H
16	H	H	H	H	D	H	H	H	H	H
17	H	H	D	D	D	H	H	H	H	H
18	S	S	D	S	S	S	S	H	H	H
19	S	S	S	S	S	S	S	S	S	H
20	S	S	S	S	S	S	S	S	S	S
21	S	S	S	S	S	S	S	S	S	S
SPLITS										
2,2	Sp	Sp	Sp	Sp	Sp	Sp	Sp	Sp	Sp	Sp
3,3	Sp	Sp	Sp	Sp	Sp	Sp	Sp	Sp	Sp	Sp
4,4	H	H	H	H	H	H	H	Sp	Sp	Sp
5,5	D	D	D	D	D	D	D	D	H	H
6,6	Sp	Sp	Sp	Sp	Sp	Sp	Sp	Sp	Sp	Sp
7,7	Sp	Sp	Sp	Sp	Sp	Sp	Sp	Sp	Sp	Sp
8,8	Sp	Sp	Sp	Sp	Sp	Sp	Sp	Sp	Sp	Sp
9,9	S	S	S	S	S	S	S	Sp	Sp	Sp
10,10	S	S	S	S	S	S	S	S	S	S
A,A	Sp	Sp	Sp	D	D	Sp	Sp	Sp	Sp	Sp

Table: Strategy Table (All actions implemented)

We compare the strategy learnt to the strategy table adapted by [Shackleford, 2019] and find the similarities in 90% of the case which is a satisfactory result.

Q-Learning: All actions implemented correctly

Impact of deck size on learning

1. We compare the performance of the player over different deck size using winning odds as metric.
2. 1 player against the dealer, 4-8 decks and Hi-lo system used by player.
3. Actions: Hitting, standing, doubling down and splitting.

Deck size	Winning odds %
4	43.83 %
5	42.28 %
6	43.43 %
7	43.45 %
8	43.89 %

Table: Performance vs deck size

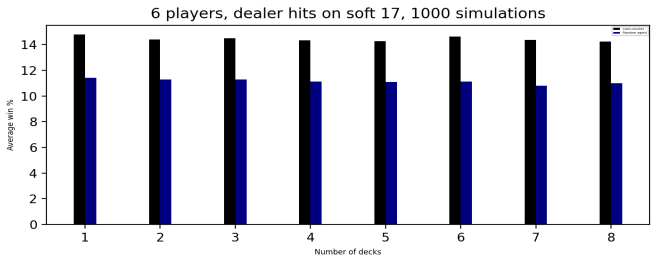
1. The discrepancy in winning odds % across varying deck size is not significant.
2. A variation in deck size between 4 to 8 decks will not therefore influence the learning efficiency and outcome of an agent following the q-learning policy.

Perfect card counter

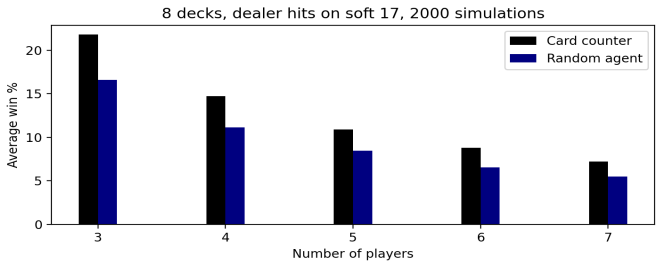
Game settings

1. Multiple player game with multiple decks, dealer can either hit/stand on soft 17.
2. Other players follow random but intuitive actions.

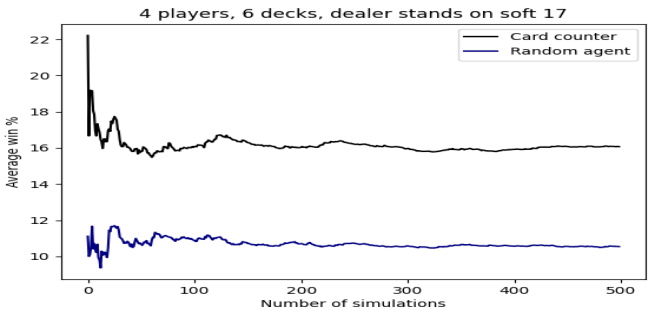
Performance



(a) Average payoff vs deck size



(b) Average payoff vs number of players



(c) Average payoff vs simulation size

4 players, 6 Decks, Dealer stands on soft 17, 10000 simulations

Players	Win %	Draw %	Loss %	Bankroll
Card counter	14.55%	1.35%	3.21%	R14,647,394
Random agent 1	11.03%	1.43%	7.52%	R (6,835,650)
Random agent 2	11.01%	1.44%	7.52%	R(6,989,950)
Random agent 3	11.02%	1.47%	7.42%	R(6,949,200)
Dealer	3.21%	1.35%	14.55%	R(14,622,385)

(d) Average payoff and Bankroll level

Further extensions

Deep Reinforcement Learning

The Deep-Q network also introduced the idea of an experience buffer which stores and samples from the agent's experiences and the neural network learns the Q-matrix rather than storing it in memory.

Complex Dynamics

We consider a multiplayer game involving a type of reinforcement learning called Experience-Weighted Attraction (EWA) which assumes a numerical attraction to each strategy. [Galla and Farmer, 2013].

Replicating "21" movie

1. Multiple tables and universal blackjack rules applied.
2. 6 perfect card counters on each table.
3. Big player switches across table with highest counts.
4. Big player only player to optimally use Hi-Lo scheme.
5. Big player and passed count through mnemonic signals.

Appendix

4-8 Decks, Dealer Stands on Soft 17

Player hard	2	3	4	5	6	7	8	9	10	A
4-8	H	H	H	H	H	H	H	H	H	H
9	H	Dh	Dh	Dh	Dh	H	H	H	H	H
10	Dh	Dh	Dh	Dh	Dh	Dh	Dh	Dh	H	H
11	Dh	Dh	Dh	Dh	Dh	Dh	Dh	Dh	Dh	H
12	H	H	S	S	S	H	H	H	H	H
13	S	S	S	S	S	H	H	H	H	H
14	S	S	S	S	S	H	H	H	H	H
15	S	S	S	S	S	H	H	H	Rh	H
16	S	S	S	S	S	H	H	Rh	Rh	Rh
17+	S	S	S	S	S	S	S	S	S	S

soft	2	3	4	5	6	7	8	9	10	A
13	H	H	H	Dh	Dh	H	H	H	H	H
14	H	H	H	Dh	Dh	H	H	H	H	H
15	H	H	Dh	Dh	Dh	H	H	H	H	H
16	H	H	Dh	Dh	Dh	H	H	H	H	H
17	H	Dh	Dh	Dh	Dh	H	H	H	H	H
18	S	Ds	Ds	Ds	Ds	S	S	H	H	H
19+	S	S	S	S	S	S	S	S	S	S

splits	2	3	4	5	6	7	8	9	10	A
2,2	Ph	Ph	P	P	P	P	H	H	H	H
3,3	Ph	Ph	P	P	P	P	H	H	H	H
4,4	H	H	H	Ph	Ph	H	H	H	H	H
6,6	Ph	P	P	P	P	H	H	H	H	H
7,7	P	P	P	P	P	P	H	H	H	H
8,8	P	P	P	P	P	P	P	P	P	P
9,9	P	P	P	P	P	S	P	P	S	S
A,A	P	P	P	P	P	P	P	P	P	P

wizardofodds.com

H	Hit
S	Stand
Dh	Double if allowed, otherwise hit
Ds	Double if allowed, otherwise stand
P	Split
Ph	Split if double after split is allowed, otherwise hit
Rh	Surrender if allowed, otherwise hit

4-8 Decks, Dealer Hits on Soft 17

Player hard	2	3	4	5	6	7	8	9	10	A
4-8	H	H	H	H	H	H	H	H	H	H
9	H	Dh	Dh	Dh	Dh	H	H	H	H	H
10	Dh	Dh	Dh	Dh	Dh	Dh	Dh	Dh	H	H
11	Dh	Dh	Dh	Dh	Dh	Dh	Dh	Dh	Dh	Dh
12	H	H	S	S	S	H	H	H	H	H
13	S	S	S	S	S	H	H	H	H	H
14	S	S	S	S	S	H	H	H	H	H
15	S	S	S	S	S	H	H	H	Rh	Rh
16	S	S	S	S	S	H	H	Rh	Rh	Rh
17	S	S	S	S	S	S	S	S	S	Rs
18+	S	S	S	S	S	S	S	S	S	S

soft	2	3	4	5	6	7	8	9	10	A
13	H	H	H	Dh	Dh	H	H	H	H	H
14	H	H	H	Dh	Dh	H	H	H	H	H
15	H	H	Dh	Dh	Dh	H	H	H	H	H
16	H	H	Dh	Dh	Dh	H	H	H	H	H
17	H	Dh	Dh	Dh	Dh	H	H	H	H	H
18	Ds	Ds	Ds	Ds	Ds	S	S	H	H	H
19	S	S	S	S	Ds	S	S	S	S	S
20+	S	S	S	S	S	S	S	S	S	S

splits	2	3	4	5	6	7	8	9	10	A
2,2	Ph	Ph	P	P	P	P	H	H	H	H
3,3	Ph	Ph	P	P	P	P	H	H	H	H
4,4	H	H	H	Ph	Ph	H	H	H	H	H
6,6	Ph	P	P	P	P	H	H	H	H	H
7,7	P	P	P	P	P	P	H	H	H	H
8,8	P	P	P	P	P	P	P	P	P	Rp
9,9	P	P	P	P	P	S	P	P	S	S
A,A	P	P	P	P	P	P	P	P	P	P

wizardofodds.com

H	Hit
S	Stand
Dh	Double if allowed, otherwise hit
Ds	Double if allowed, otherwise stand
P	Split
Ph	Split if double after split is allowed, otherwise hit
Rh	Surrender if allowed, otherwise hit
Rs	Surrender if allowed, otherwise stand
Rp	Surrender if allowed, otherwise split

Strategy Table

Acknowledgements
I want to thank Assoc Prof Tim for his constant guidance and support to allowing me to work on this project. I also appreciate the help provided by the Stats department during this trying year.

References

Baldwin, R. R., Cantey, W. E., Maisel, H., and McDermott, J. P. (1956).
The optimum strategy in blackjack.
Journal of the American Statistical Association, 51(275):429–439.

Galla, T. and Farmer, J. D. (2013).
Complex dynamics in learning complicated games.

Paul", S. (2019).
An introduction to q-learning: Reinforcement learning.

Shackleford, M. (2019).
4-deck to 8-deck blackjack strategy.
`http://wizardofodds.com/games/blackjack/strategy/4-decks/`.

S.Sutton, R. and Barto, A. G. (2014).
Reinforcement Learning: An Introduction.
MIT Press.

Q & A