Model-free Q-learning of Blackjack

Austin Finn*
Stanford University, Stanford, California, 94305

Blackjack is a common casino game whose goal is to have more points than the dealer without exceeding 21. Because of the simplicity of its rules, blackjack is popular for testing the implementation of machine learning algorithms. Using three different representations of the game state and a random exploration strategy, a Q-learning method was used to obtain optimal blackjack playing policies, and their performance was compared to a random actor and a known optimal policy. All three state space definitions were able to outperform the random player, but could not match the known optimum. The least robust state space had the best performance of the three due to the simple exploration, but the use of a more sophisticated exploration could obtain better performing policies.

I. Introduction

A. Problem Description

BLACKJACK is a popular casino card game where the objective is to have a point total higher than the dealer's without exceeding 21 points. In blackjack, each card is assigned a number of points: cards 2 through 10 are worth points equal to their face value; Jacks, Queens, and Kings are each worth 10 points; Aces are worth either 1 point or 11, whichever is better. When a game starts, the dealer and player are each dealt two cards, with all but one of the dealer's cards face up. After hands are dealt, the player can choose to hit, and be dealt another card from the deck, or stand. The player can hit as many times as they wish, as long as they do not exceed 21 points (going 'bust'). After choosing to stand, the dealer hits until they bust or exceed some predetermined point threshold, typically 17. If the player does not bust and ends with a higher point total than the dealer, the player wins, and otherwise loses.

As a stochastic game, optimal play in blackjack requires consideration of both the likelihood of being beaten by the dealer and the likelihood of going bust. Because blackjack is easily represented as a Markov Decision Process (MDP), it is commonly used to test the implementation of various learning and decision making algorithms, including Q-learning [1], reinforcement learning [2], multi-agent learning [3], neural networks [4], and quantum computing techniques [5]. Various exploration methods are also tested using the blackjack environment, such as greedy policies or Monte Carlo searches. Performance of a computed policy is compared to an already known optimal policy that minimizes player losses, called the blackjack basic strategy. It is then expected that our method of interest, being the model-free Q-learning described in the next section, should be suitable to find an optimal or near-optimal policy for blackjack.

B. Q-Learning

Reinforcement learning is a form of decision-making algorithm which uses rewards gained from exploration of the problem space to determine an optimal policy, in comparison to methods which directly calculate an optimal policy from precalculated values for the transition probabilities and rewards from any state s for an action a. Model-based reinforcement learning uses data collected from this exploration to construct approximations of the reward and transition models, and is particularly suited for problems with small state and action spaces. Model-free methods, meanwhile, do not build estimations for the transition and reward matrices, and instead work directly with the action value function Q, defined as

$$Q(s,a) = R(s,a) + \gamma \sum_{s'} T(s'|s,a)U(s'). \tag{1}$$

R represents the reward model, γ the discount factor of future rewards, T the probability of transitioning from state s to s' following action a, and U the utility of being in a certain state. The action value function represents the expected return when starting in state s, taking action a, and then following the greedy policy given by Q. One can get an optimal policy π from Q with

$$\pi(s) = \arg\max_a Q(s,a). \tag{2}$$

*M.S Candidate, Department of Aeronautics and Astronautics, 496 Lomita Mall

Q-learning is a specific type of model-free reinforcement learning that incrementally updates Q with exploration data. For a sample that contains information on the state, action, currently obtained reward, and the next state, the action value function can be updated by

$$Q(s,a) \leftarrow Q(s,a) + \alpha(r + \gamma \max_{a'} Q(s',a') - Q(s,a)). \tag{3}$$

The learning rate α determines how large the update to Q is. With sufficient data on each state and action combination, this update process will converge to the optimum $Q*$.

II. Implementation

Many state space representations are possible for the blackjack environment. The most accurate would be a state space with every possible combination of player hand and known dealer card, which could then allow for a direct calculation of the transition probabilities between states. Such a state space would be incredibly large (in the millions), however, when simpler definitions could be used, such as state spaces defined by point totals rather than exact cards. Three different state spaces were used in exploring the results of Q-learning for this system. The first state space was defined solely by the point total of the player's hand with no other considerations. The second included the dealer's known card alongside the player's point total. The final state space considered the dealer's visible card, the player's point total, and whether or not the player's hand was 'soft' (meaning the hand contains an Ace with a value of 11 rather than 1). The number of states for these three representations are displayed in Table 1. These state definitions keep the size of the state space small, which reduces the number of calculations necessary. For similar reasons, the action space was limited to hit or stand only, though other actions are available (depending on where the game is being played).

Table 1 State space definitions and corresponding number of states.

Space Definition	# States
Player Points	20
Player + Dealer Card	200
Player + Dealer + Soft/Hard Ace	270

Exploration was done by randomly choosing actions and storing the corresponding (s, a, r, s') tuples. When reaching a terminal state, a reward of +1 was given if the player won the hand, and a reward of -1 given if the player lost. The blackjack simulator played with the following rules:

- 4 decks
- Dealer stands on soft 17
- Player can only hit or stand
- Ties are considered a loss for the player

For each of the three state spaces, 10000 learning hands were played. For every 500 learning hands played, the Q-learning update algorithm was performed 10 times to ensure the update was converging, with $\alpha = 1/k$ where k was the iteration number, and a discount factor $\gamma = 0.95$. After updating Q with Eq. (3), 10000 testing hands were played. The win percentage corresponding to that policy was calculated, along with the total win percentage for that state space, to observe how more learning hands influenced performance of the policy.

III. Results

Performance of the learning agent was done by comparing its win rate to two hard coded players. One player took random actions, which corresponded to a win rate around 30%, which serves as a lower bound for any reasonable policy. The second player followed what the loss-minimizing basic strategy, which wins around 43% of the time, and serves as the upper bound for blackjack win rate (excluding strategies like counting cards, which are not being considered). For the three state spaces defined, the win rate for each iteration of the policy is shown in Fig. (1), and the overall win rate as the policy evolved in Fig. (2). The bounding cases are also displayed as constant lines in both figures.

Notably, policies for all three state spaces are able to perform significantly better than a random policy with only 500 learning hands played, despite the exploration strategy being entirely random. The smallest (point only) state space may perform better at this stage because there are fewer states that need to be visited, and the random sampling therefore

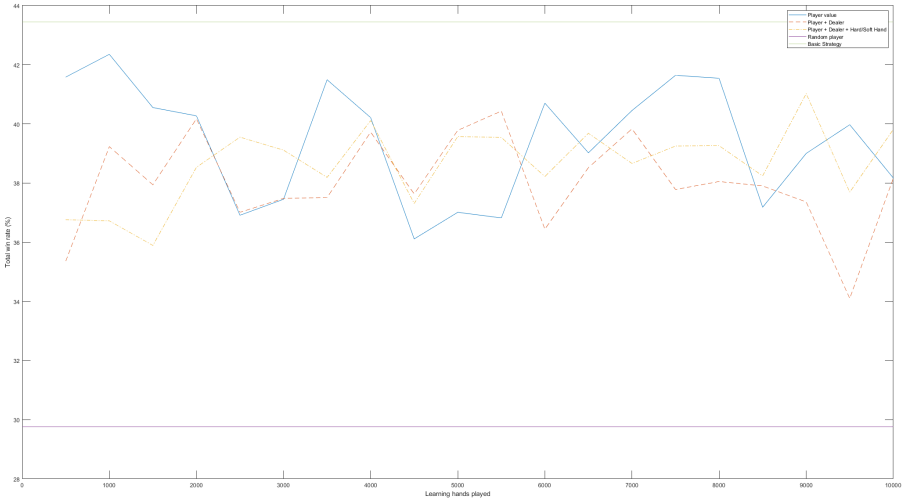

Fig. 1 Policy win rates versus number of learning hands.

better explores the entire space, while the other two spaces need more exploration to refine the policy. As can be seen in Fig. (2), the point-only space actually performs the highest early on, but starts performing worse as the policy evolves, while the other two policies improve as more data is included. However, the point-only policy still manages to have the best performance of the three over the 10000 learning hand period, contrary to expectations that the point-dealer-ace space would perform the best.

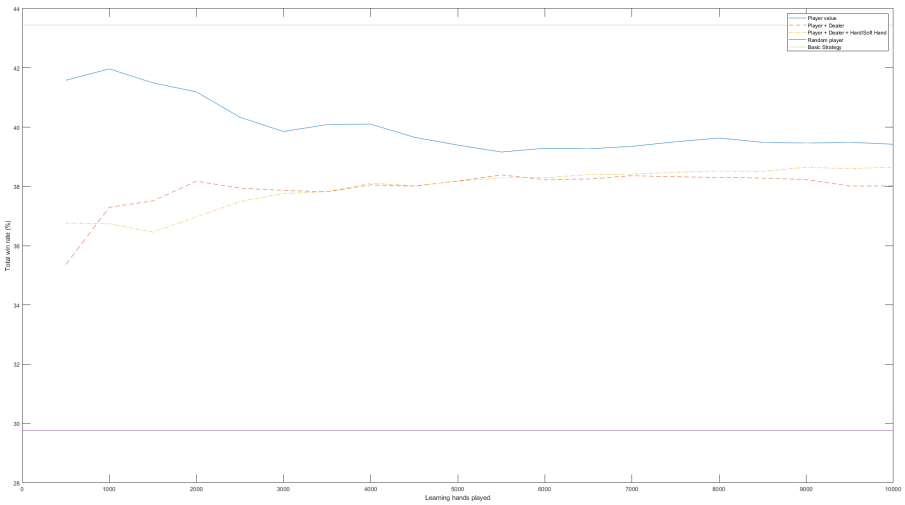

Fig. 2 Cumulative win rate as the policies evolve.

The win rates that these policies are converging to have been listed in Table 2. Although all do better than the random policy, none of the three are converging to the optimal basic strategy. This, in part, can be attributed to the random exploration strategy, which makes it possible for the data samples to be taken state-action pairs that do not allow convergence to the true optimum, but to a near-optimal policy. A more sophisticated exploration strategy allow for closer convergence to basic strategy.

Table 2 Overall win rate for each state space's policies.

Space Definition	Overall Win Rate (%)
Player Points	39.4
Player + Dealer Card	38.0
Player + Dealer + Soft/Hard Ace	38.7
Random	29.8
Basic Strategy	43.5

IV. Future Work

The results of Q-learning for the three state spaces considered are able to get well-performing policies, but does not approach the optimal goal of basic strategy. To better determine the effectiveness of each state space definition, the exploration strategy could be upgraded to a method like ϵ-greedy exploration to balance exploration with taking greedy actions. Such a strategy should guide the policy towards the optimum more so than the purely random exploration strategy employed here.

Additionally, the effect of more learning data can be better analyzed by calculating the performance of the policy at more instances of the learning process. Ideally, this could be calculated for each new learning hand played; though such a method requires significantly more computational time, more insight could be gained into how quickly each state space definition converges, and when it surpasses the random player's performance.

Lastly, the simulator and action space can be expanded to include actions like splitting hands or doubling down, to better represent a realistic game of blackjack. Such actions allow an experienced player to reduce the casino's edge (the expected amount of money the house gains on average) to 0.5%, and by fully simulating the game, the possibility of a policy that can 'beat the house' and push the odds to the player's favor can be explored.

V. Conclusion

The blackjack environment provides a simple scenario to test the performance of learning algorithms by comparing to known optimal policies. Using a Q-learning method for three state spaces of increasingly complexity, the resulting policies were compared, in terms of win rate, to a random player and the optimal basic strategy. Using a random exploration strategy, policies were determined that considerably outperformed the random player even with only 500 learning hands, but the methodology used was unable to reach the performance of the basic strategy. Implementation of a more sophisticated exploration strategy and expansion of the blackjack simulator should allow for closer convergence to the optimal policy, and possibly find a policy that can exceed it.

Acknowledgments

I would like to thank Professor Kochenderfer and the rest of the AA228 teaching team for their excellent instruction in the course this quarter, and for being accommodating during these stressful times.

References

[1] De Granville, C., "Applying reinforcement learning to blackjack using q-learning," *University of Oklahoma.*, n.d.

[2] Kakvi, S. A., "Reinforcement Learning for Blackjack," *Entertainment Computing – ICEC 2009*, edited by S. Natkin and J. Dupire, Springer Berlin Heidelberg, Berlin, Heidelberg, 2009, pp. 300–301.

[3] Gan, X., Guo, H., and Li, Z., "A New Multi-Agent Reinforcement Learning Method Based on Evolving Dynamic Correlation Matrix," *IEEE Access*, Vol. 7, 2019, pp. 162127–162138. https://doi.org/10.1109/ACCESS.2019.2946848.

[4] Perez-Uribe, A., and Sanchez, E., "Blackjack as a test bed for learning strategies in neural networks," *1998 IEEE International Joint Conference on Neural Networks Proceedings. IEEE World Congress on Computational Intelligence (Cat. No.98CH36227)*, Vol. 3, 1998, pp. 2022–2027 vol.3. https://doi.org/10.1109/IJCNN.1998.687170.

[5] Lockwood, O., and Si, M., "Reinforcement Learning with Quantum Variational Circuits," , 2020.