# CS 238 Project
# Agent Q Plays Texas Hold'em

**Ammar Alqatari**
Computer Science
ammarq@stanford.edu

**Ben Gaiarin**
Symbolic Systems
bgaiarin@stanford.edu

**Michael Vobejda**
Computer Science
mvobejda@stanford.edu

## Abstract

Multi-agent games like card games offer us with outlets for testing foundational reinforcement learning algorithms and techniques. Currently, there exists uncertainty regarding how Q-Learning might perform when applied to the problem of Texas Hold'em. We investigate this uncertainty first by building a multi-agent Markov Decision Process model of the game Texas Hold'em. We then evaluate the performance of Q-Learning on the model by assigning specific Q-Learning strategies to specific agents, and comparing those agents by playing them against each other and by playing them against baseline agents (agents that follow fixed policies and do not learn). We find that while Q-Learning performs better than random in some cases, it consistently performs worse than a fixed policy that elects to call on every bet, suggesting that Q-Learning alone is insufficient for the problem of Texas Hold'em. We also find that Q-Learning performs better when it extracts more salient information about the game environment.

## 1 Introduction

Texas Hold'em is a classic multi-agent game. Games like Texas Hold'em present us with interesting problems on which to model and to test new reinforcement learning (RL) algorithms and techniques. Q-Learning, a foundational RL algorithm, has been applied to multi-agent games and card games. There exists, however, some doubt regarding how Q-Learning might perform on the problem of Texas Hold'em.

In this paper, in an effort to recognize that doubt, we consider how Q-Learning might best be applied to the problem of Texas Hold'em. We build a Markov Decision Process (MDP) model of the game of Texas Hold'em, and, to help further a better understanding of what the most salient points of information are in Texas Hold'em, we run Q-Learning with five different feature extraction strategies. We simulate several hundred thousand games between seven players playing against each other, with each player either employing Q-Learning with a unique feature extractor or employing a fixed baseline policy without learning. We find that while Q-Learning does, in some cases, perform better than random (a fixed policy of randomly choosing actions), it does not perform better than a fixed policy of calling on every bet. This suggests that Q-Learning alone is not sufficient to handle the state and action complexity of the Texas Hold'em problem with good results. We also notice that Q-Learning performs better when it extracts more information, connoting that including more salient information is preferred to generalizing more Q values across the state and action space.

CS238: Decision Making Under Uncertainty, Fall 2018, Stanford University, CA.

We hope that our MDP model might inform future simulators of Texas Hold'em, and that our results with Q-Learning may contribute to the growing body of research and public interest in applying RL techniques to zero-sum, multi-agent games.

## 2 Background

For the scope of this paper, it will be helpful to know the basic rules of the game Texas Hold'em, as all of the rules are built into our model. We will assume that the reader is familiar with basic card terminology (e.g. "folding"). The game is played as follows:

- (ROUND 0) Each player is dealt two cards. Three communal cards are placed in the middle of the table face-down. The ultimate goal of the game is to make the best five-hand card possible using any combination of the two cards in your hand and the communal cards. With that in mind, and knowing only your two cards for the moment, you take one of three types of actions: fold, call, or raise. Your bet goes into the "pot" in the middle of the table. If you raise, all other players who wish to remain in the game will need to at least match your raise. Similarly, you must always at least call the bets of other players to remain in the game.
- (ROUND 1) The three communal cards are flipped over. Now you can consider those communal cards to be a part of your own hand. The second round of betting takes place, with the play moving clockwise around the table. (Players take turns betting first between rounds.)
- (ROUND 2) An additional card is flipped over, and there is a third round of betting.
- (ROUND 3) An additional card is flipped over, and there is a fourth and final round of betting. At this point, whoever is still in the game and holds the five-card combination of highest value wins the pot. Alternatively, if all but one folds before reaching Round 3, the game would end early. Note that if you folded earlier in the game, you would have lost all of the money you bet before folding.

## 3 Literature Review

Models for zero-sum, multi-agent games, poker games included, have made an important mark in both recent and early literature on reinforcement learning. Erev and Roth (1), in an early work from 1998, find that even a simple, single-parameter RL model can be used to conjure predictions and explanations for a "broad range" of games without "fitting parameters to each game." Their results moreover suggest that incorporating "responsiveness" into a model can improve predictions by allowing the model to change in accordance with changes in players' individual behaviors. We enable responsiveness in our Texas Hold'em model indirectly; the actions that players do affect the possible actions available to other players as well as their eventual outcomes, but players do not reference other players' decisions when making their own.

There exists some uncertainty in the literature regarding the appropriateness of employing Q-Learning to zero-sum, multi-agent games like Texas Hold'em. Shoham et al.(2) find that Q-Learning peforms well for zero-sum repeated games. Dahl (3), however, finds that the method is "not applicable to games of imperfect information." Similarly, Szita (4) notes that fundamental RL algorithms (such as Q-Learning) "are rarely sufficient for high-level gameplay," and that it is therefore essential to think of additional ways of "inserting domain knowledge" to allow such algorithms to scale to complex modeling problems.

Recognizing the uncertainty regarding the potential performance of Q-Learning for this problem, we conduct a thorough investigation of Q-Learning as applied to Texas Hold-em to unpack some uncertainty regarding how the algorithm performs. We take into account various means of incorporating domain knowledge into the algorithm by implementing different feature extractor strategies. And we extend Q-Learning to work in the multi-agent space by strictly associating our Q values with a given player's hand and their action, effectively ignoring the other agents' decisions and making the game space passive, as suggested by Shoham et al.(2).

## 4 Approach

For a more detailed illustration of our approach, our code can be found at:
`https://github.com/michaelvobejda/texas_holdem`.

We model the game of Texas Hold'em as a Markov Decision Process (MDP) with a given state (type: dictionary) containing the following information:

- state['board'] = a list of the communal cards
- state['pot'] = an integer representing the amount of money in the pot
- state['players'] = a list of tuples $(hand, bet)$, one for each player. If $bet = -1$, meaning the player folds, then their $hand$ is set to false.
- state['curBet'] = the current highest bet in the round (the bet to call or raise to)
- state['curPlayer'] = the index of the player who's turn it is to bet

We make use of the open-source python hand evaluation library $Deuces$ (source: `https://github.com/worldveil/deuces`) for keeping track of the card deck, for card representation, and for evaluating hands (by rank). The main methods of our MDP class run according to the rules of Texas Hold'em as described in Section 3. The only notable deviation from the standard rules we take is our method for deciding which player goes first in a given round of betting. Normally, the starting player would rotate clockwise around the table, but we code the starting player to be decided randomly. This ensures that when we run Q-Learning, aggregate earnings across games are not determined by the ordering of players in state['players'].

We run Q-Learning on our MDP according to a standard Algorithm 5.3 from Kochenderfer (5) with a tweaked update formula. For cases where $action == -1$ (fold), we perform no update to the associated $weight[state][action]$. If the reward is a loss (placing a bet but haven't won yet), then we set $reward = absoluteValue(1/(reward + 0.01)$ in our update function. Otherwise, if the reward is a win (win all the money in the pot), then we perform the standard update. This tweaking ensures that Q values are not negative; consistently negative Q values yield little learning (the agent would rarely revisit negative Q values).

We employ five different feature extractors for Q-Learning that all achieve different results of varying salience. This allows us to run a game of seven agents where five agents are simulating Q-Learning under the five different feature extractor strategies. These five agents pursue the following feature extraction strategies:

- Standard: Extracts `[state = board + hand]` and `[action = any action in mdp.getActions(state)]`.
- Action Agnostic: Extracts only `[state = board + hand]`.
- Binary Action: Extracts `[state = board + hand]` and `[action = 0 if fold; 1 otherwise]`.
- Hand Rank: Extracts only `[state = rank of (board + hand)]`.
- Hand Rank with Binary Action: Extracts `[state = rank of (board + hand)]` and `[action = 0 if fold; 1 otherwise]`.

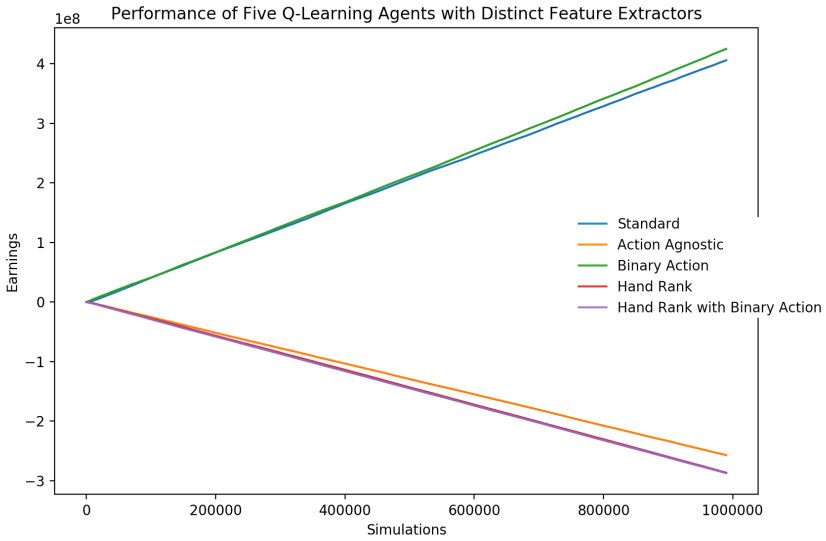
We also make use of two baseline agents, permitting us to run on a total of seven players that all make use of different methods. The two baseline agents perform the following:

- Random Action: No learning involved. Just choose a random action according to:
  `[action = randomly select an action a ∈ mdp.getActions(state)]`
- Uniform Call Action: No learning involved. Just uniformly choose whatever action calls (matches) the bet.

Testing five different feature extractor strategies for Q-Learning allows us to evaluate the tradeoff of between encapsulating enough salient information in a Q value and still allowing the Q value to generalize across the state and action space to produce meaningful results. We are interested, specifically, in discovering what features are most salient to the problem of Texas Hold'em, as that investigation has yet to be conducted for Texas Hold'em.

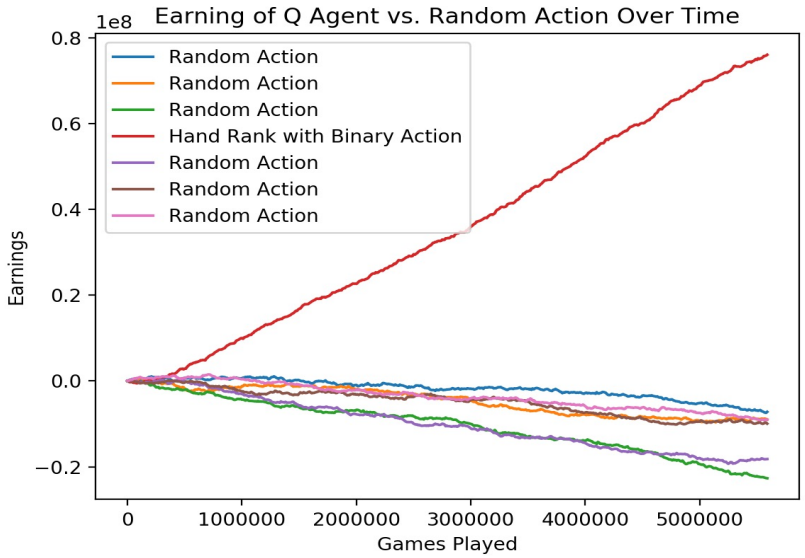
## 5 Results

We first run all five distinct Q-Learning strategies against one another as five separate agents in 1 million five-agent games of Texas Hold'em. Each agent maintains their own weights across all games. We achieve the following results:




We can see that Standard and Binary Action feature extraction strategies outperform the others. We hypothesize that this is because they are the two that extract the most detailed information. This comes at a loss of generalization (less feature extractions are sharing Q values), but it comes at a gain in precision (Q values are tailored to specific states and actions).

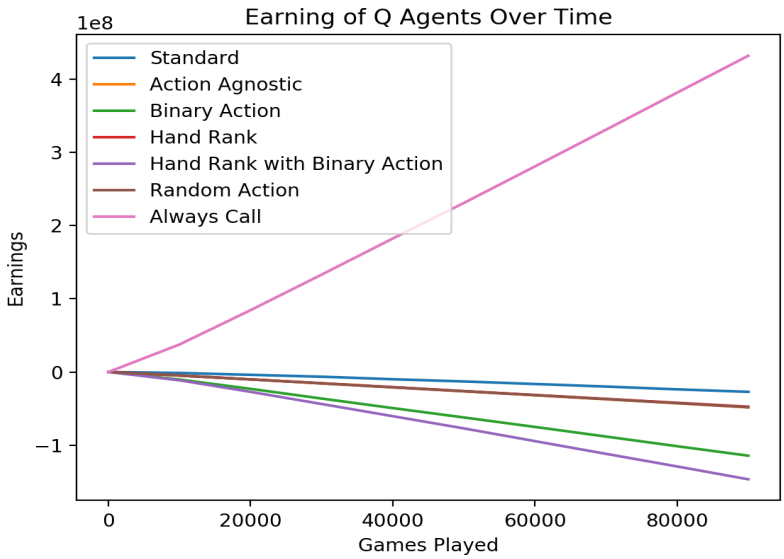
Next, we run the worst-performing Q-agent, Hand Rank, against agents that follow Random Action policies to ensure that all Q-Learning methods perform at least better than random. We achieve the following results:




As we can see in the graph above, Hand Rank performs significantly better than all Random Action agents. However, there is a delay in learning which causes a delay in seeing this superior performance;

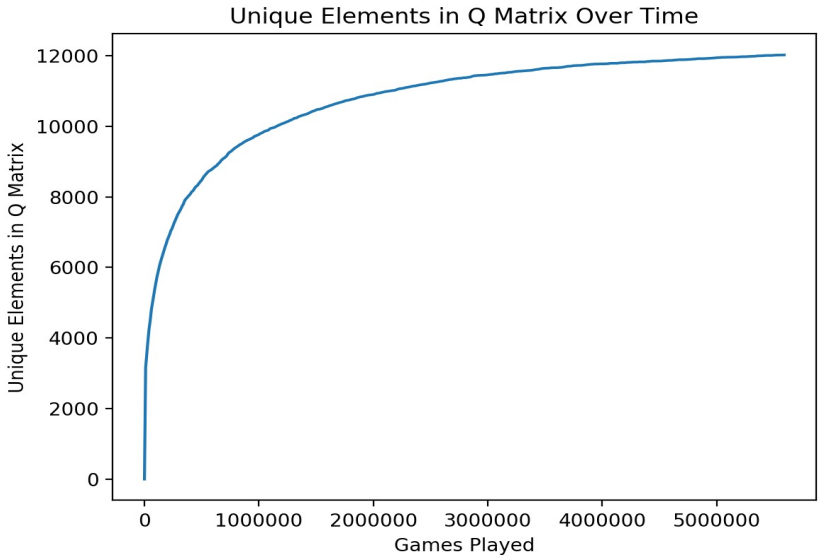
Hand Rank does not start performing better until a little under 1 million games. This suggests that Hand Rank, while it generalizes more than Standard and Binary Action, may require longer training periods to converge to useful Q values. Or, rather, that the state and action space of the MDP is big enough to require such a delay period in seeing improvement.

Next, we test all Q-agents against our two baselines: Random Action and Uniform Action Call (labeled as "Always Call").




We can see in this graph that while Q-Learning can outperform Random Action even in a smaller number of games, it woefully underperforms Uniform Action Call. This can be expected; always calling the bet makes the Uniform agent more likely to win ("you can't win without trying) than Agent-Q, which is more likely to fold.

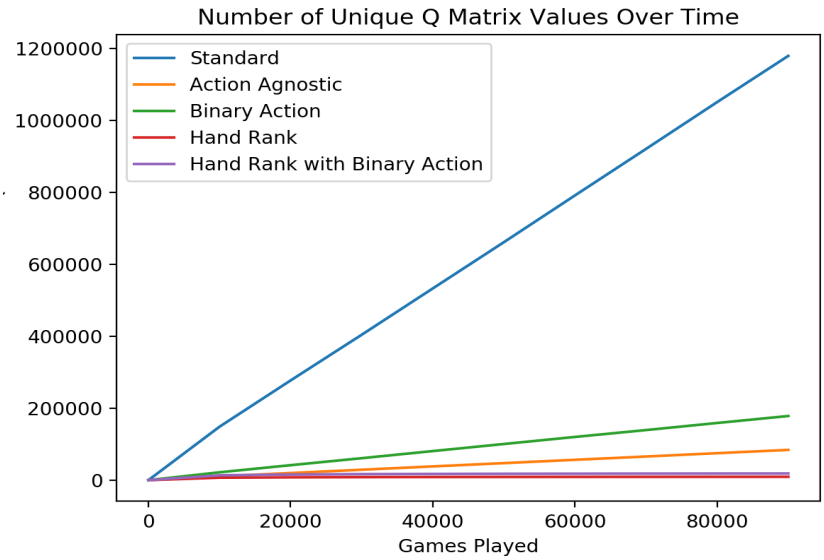
Next, we run Hand Rank with Binary Action, a feature extraction strategy that generalizes more than most others, and plot the number of Q values it establishes over time:




We can see in this graph that the number of Q values established plateaus over time, which is what we would expect. Hand Rank with Binary Action generalizes well, and as we run more games we

revisit more and more Q values as expected. But the plateau occurs only after approximately 4 million games have been played, suggesting we need greater computational power to run Q-learning successfully to achieve the generalization and revisit rate we would desire.

Finally, we run all Q-agents against one another and plot the number of unique Q values that each agent creates over time:




The results shown suggest that Standard, which extracts info from the largest state and action spaces, generalizes the least by a significant margin. This is to be expected, and suggests that there could indeed be a strong and positive relation between the number of Standard's distinct Q values and its success with respect to other Q-Learning strategies. We also do not see an asymptotic curve here, reaffirming the point that was made in response to the previous graph regarding computational power and performance.

## 6 Conclusions

Our investigation suggests that Q-Learning is not fully equipped to deal with complex, multi-agent problems with as large a state and action space as Texas Hold'em. While the model is able to learn some nuance about the value of making certain bets given certain hands, there is still too much randomness in its decision making and is beat out by strategies that never fold (e.g. Uniform Call Action).

Our results suggest that perhaps with more computational power, Q-Learning could still be effective against random action agents. We ran a simplified simulation with just one Q-Learning agent using the least-promising feature extraction algorithm and six Random Action agents (see Earning of Q Agent vs. Random Action Over Time), and the Q-Learning model did eventually outperform all of the Random Action agents. We believe this is a result of the Q matrix becoming almost fully populated, as shown in the graph "Unique Elements in Q Matrix Over Time". As discussed above, a more complete Q matrix results in much better performance of the model, so with more computational power it would be possible to populate the Q matrix more completely and for the model to make intelligent, informed decisions for a given state and action.

## 7 Contributions

Ammar Alqatari programmed much of the structure of our MDP and the Monte Carlo Tree Search algorithm. (Our Monte Carlo implementation is left out of this paper due to untimely errors.)

Ben Gaiarin wrote the skeleton for the MDP structure, programmed much of Q-Learning, prepared data visualizations and wrote this paper.

Michael Vobejda programmed much of the structure of the MDP and of Q-Learning, helped with Monte Carlo Tree Search, prepared data visualizations, and contributed to this paper.

## References


[1] Erev, Ido, and Alvin E. Roth. *Predicting How People Play Games: Reinforcement Learning in Experimental Games with Unique, Mixed Strategy Equilibria*. American Economic Review (1998): 848-881.

[2] Shoham, Yoav, Rob Powers, and Trond Grenager. *If Multi-Agent Learning is the Answer, What is the Question?*. Artificial Intelligence 171.7 (2007): 365-377.

[3] Dahl, Fredrik A. *A Reinforcement Learning Algorithm Applied to Simplified Two-player Texas Hold'em Poker*. European Conference on Machine Learning. Springer, Berlin, Heidelberg, 2001.

[4] Szita, István. *Reinforcement Learning in Games*. Reinforcement Learning. Springer, Berlin, Heidelberg, 2012. 539-577.

[5] Kochenderfer, Mykel J. *Decision Making Under Uncertainty: Theory and Application*. MIT press, 2015.