Poker Project - Team 07

Arielyte Tsen Chung Ming, Devarajan Preethi, James Pang Mun Wai, Lee Yi Wei Joel, Yip Seng Yuen
National University of Singapore
chungming.tsen@u.nus.edu, e0203237@u.nus.edu, jamespang@nus.edu.sg, lywjoel@u.nus.edu, yip@u.nus.edu

1 Introduction

The game of poker has long been a field of interest for artificial intelligence (AI) researchers. This is not surprising as developing an AI poker agent capable of competing with humans at the highest level is challenging, given that poker is a game that forces players to make decisions with incomplete and imperfect information. This paper talks about the steps we took in designing an AI Poker Player agent in Heads Up Limit Texas Hold'em. An agent subjected to reinforcement learning must learn to interact with the environment in order to maximize its reward. Reinforcement learning was chosen as a way to get the best possible action based on the current state.

The problem with Limit Texas Hold'em is that there is no input or output training data sets available for the agent to gather feedback from. Hence, the only way an agent can learn is by trial and error based on feedback from its own actions and experiences. A reward function where actions that led to positive outcomes are rewarded, and actions that led to negative outcomes are penalized is created, with the goal of maximizing its reward. The design of a reward function good enough for the agent to master poker is the basis for this project. In a game of poker, the agent does not have a complete model of the environment, nor does it know the states that its actions will lead to. Thus, a Q-learning agent, which compares the expected utilities of the remaining choices without the need to know the outcomes nor the model of the environment [Russell and Norvig, 2014], is our agent design of choice as it fits the scenario of a poker game.

2 Pre-Flop Hand Strength

To begin with, we wanted to decide how the agent will determine the strength of its hand, just from the pre-flop round. Due to the limited amount of information that we can manipulate during the pre-flop round, the strength of a hand, HS, at pre-flop is determined by the probability of winning, $Pr(Win)$, based on the prior knowledge of the two hole cards currently in hand.

To estimate HS, we simulated 10,000 games for each possible combination of starting hand. The total combination of starting hands is $13 + \binom{13}{2} = 91$, obtained by summing up the combination of pocket pairs and the combination of unsuited hands. For each game, we first pick the two starting cards for the agent, followed by randomly picking two starting cards for the opponent and five community cards. Both players' hands are then revealed and the game result determined as a win or a loss for the agent.

Card 2 \ Card 1	2	3	4	5	6	7	8	9	T	J	Q	K	A
2	0.4986	0.2853	0.2921	0.3138	0.3334	0.3345	0.3702	0.3878	0.4033	0.427	0.47	0.503	0.5371
3		0.531	0.32	0.3365	0.3566	0.3624	0.3728	0.3941	0.4283	0.4583	0.4762	0.5164	0.559
4			0.5686	0.3556	0.3867	0.3834	0.3926	0.4091	0.4433	0.4712	0.504	0.5295	0.5595
5				0.5973	0.3972	0.4139	0.4093	0.4302	0.4456	0.4831	0.5121	0.5386	0.5728
6					0.6327	0.4298	0.4442	0.4552	0.4715	0.4913	0.521	0.5593	0.5883
7						0.6737	0.4608	0.4792	0.4847	0.5081	0.5341	0.5669	0.5929
8							0.6944	0.4863	0.5063	0.5276	0.5403	0.5749	0.6107
9								0.7235	0.5283	0.5423	0.5696	0.5931	0.6254
T									0.7621	0.5734	0.5924	0.6125	0.6406
J										0.7804	0.593	0.6283	0.6407
Q											0.8019	0.6368	0.6612
K												0.8373	0.6697
A													0.8556

Figure 1: Estimated winning probabilities with starting hands

We then calculate $Pr(Win)$ using the formula:

$$Pr(Win) = \frac{\#\ of\ wins}{Total\ games\ played}$$

Where the total games played is fixed at 10,000 in our case. The results of our simulation are seen in Figure 1.

At every pre-flop street, the agent chooses an action to be performed based on the starting hand's estimated probabilities of winning, which are:

$$\left.\begin{array}{l} raise\text{: } Pr(Win) > k \\ call\text{: } Pr(Win) > j \\ fold\text{: } Pr(Win) \leq j \end{array}\right\} \quad \text{where } 0 < j < k \leq 1$$

Where j and k are pre-determined thresholds set for the agent.

3 Post-Flop Rounds

In subsequent post-flop rounds, reinforcement learning is employed. Specifically, the technique of Q-learning, which was described in the Introduction, was used to train our agent.

In Q-learning, we maintain a table of states and actions as inputs for the agent. Each state has a Q-value that corresponds to each action. An illustration of it can be seen in Table 1.

State	Action		
	fold	*call*	*raise*
1			
2			
3			
...			
S			

Table 1: Q-learning Training

After each training round, the Q-value of each state is updated based on the formula given below:

$$Q\left(s',a'\right) = Q\left(s,a\right) + \sum_{0}^{t} \lambda^t R_t\left(s,a\right)$$

Where t is the number of turns between the move and the terminal node, R_t is the reward for round t, λ is the discount factor, s is the state and a is the action.

$$R_t = \begin{cases} +\frac{pot}{2} & \text{if win,} \\ -\frac{pot}{2} & \text{otherwise} \end{cases}$$

$\lambda =$ discount rate for future reward

Additionally, we applied the technique of experience replay to achieve more efficient use of previous experiences. We store the agent's experiences based on the formula

$$e_t = (s_t, a_t, R_t, s_{t+1})$$

The agent stores the data discovered for each round in a table, and reinforcement learning takes place based on random sampling from the table. This reduces the amount of experience required to learn, replacing it with more computation and memory which are cheaper resources than the agent's continuous interactions with the environment [Schaul *et al.*, 2016].

4 ε-Greedy Algorithm

At every turn, with a probability of ε, a random action is chosen to be performed, otherwise an action with the best expected reward is chosen.

The value of ε will slowly decrease as the agent acquires more training. A relatively large initial value ensures that all possible paths will be explored before the agent settles into a sub-optimal pattern. The i-th *State* and *Action* for the ε-Greedy Algorithm is

$$State_i = \{EHS_i, S_i, P_i, \#OR_i, \#SR_i, OPS_i\}$$

$$Actions = \{fold, call, raise\}$$

Where

EHS refers to the current Expected Hand Strength of a given player. *EHS* is grouped in increments of 0.01. More details can be found in Section 4.1.

S refers to the current Street that the game is in, which is the start of each new round. *S* three groups that are evaluated are Flop, Turn and River.

P refers to the Pot Size, which is the total amount in the player's bet. *P* is grouped by the number of big blinds.

#OR refers to the number of Opponent Raises per street. *OR* is grouped into nine different groups, i.e. from 0 to 8.

#SR refers to the number of Self Raises per street. *SR* is grouped into nine different groups, i.e. from 0 to 8.

OPS refers to the Opponent Playing Style. *OPS* is grouped into four categories. More details can be found in Section 4.3

Based on the groupings, the total size of the state space will be

4.1 Expected Hand Strength

The Expected Hand Strength, *EHS* is the probability of the current hand of a given player winning if the game reaches a showdown. It factors in all possible combinations of the opponents' hands, the remaining hidden board cards, and performing a comparison between the agent's hand and the hands in the enumeration to see which is better. The quality of the hand is then measured based on the number of times the hand turns out to be better. For our ε-Greedy Algorithm, the *EHS* is grouped in increments of 0.01. The remaining cards, *Rem* is given by:

$$Rem = [\alpha \backslash \beta]^5$$

Where α is the set of all cards in the deck, and β is the set of all hole cards of a particular player. The formula to calculate the rank of each hand, $Rank(h)$ is:

$$Rank(h) = max(\forall x \in [\beta \cup \Omega]^5 : s(x))$$

Where Ω is the set of community cards. Having *Rem* and $Rank(h)$, it is now possible to calculate *HS* through the formulas below:

$$Ahead(h) = \#\{\forall x \in Rem : s(x) > Rank(h)\}$$

$$Tied(h) = \#\{\forall x \in Rem : s(x) = Rank(h)\}$$

$$Behind(h) = \#\{\forall x \in Rem : s(x) < Rank(h)\}$$

Thus, the *EHS* for player h against 1 opponent is given by:

$$EHS(h) = \frac{Ahead(h) + \frac{Tied(h)}{2}}{Ahead(h) + Tied(h) + Behind(h)}$$

The above formulas used to derive the *EHS* was referenced from Teofilo *et al.* [2013a]. Note that the *EHS* may be used at any round of the game. However, computing an accurate *EHS* would exceed the time constraint for each round as the number of iterations needed to compute it for a single hand at the pre-flop street is very high. This issue is addressed by using the Average Rank Strength technique which is explained in the next section.

4.2 Average Rank Strength

A technique developed by Teofilo *et al.* [2013b] called Average Rank Strength (*ARS*), is used to improve the efficiency of *EHS*. *ARS* consists of using the hand score to estimate the future outcome of the match, without having to generate all card combinations. This is simply done by storing the average *EHS* of a hand for each score in three lookup tables, one for Flop, one for Turn and one for River. Since we are not considering suits, the number of possible scores is reduced. The pre-computation of a given score is as follows:

$$ARS_5(C_1,C_2) = (\frac{\sum_i X_i \in [\alpha \backslash \beta]^5 : Rank(X_i \cup \{C_1,C_2\})}{\#[\alpha \backslash \beta]^5}$$

Where X_i is a distinct subset of size 5 of the deck except the pocket cards.

Thus, compared to *EHS*, *ARS* had a three orders of magnitude faster response time when querying the lookup table, while also doing so with negligible error [Teofilo *et al.*, 2013b].

To find out the estimated *EHS* for the three post-flop streets, we simulated 2,500,000 rounds each. Additionally, for each round of simulation, we ran the Monte Carlo sampling algorithm 500 times to obtain the *EHS*. The results are displayed below:

Street	Number of Combinations
Flop	5133
Turn	13,408
River	17,470

Table 2: Result of *ARS* Simulation

Our reasoning behind such a high number of simulation is. so that we can sample as many combinations as possible. And in doing so, achieve a more accurate estimate.

Based on the results from simulating 100,000 rounds, approximately 0.13% of the scores could not be found in our table. This goes to show that our three *ARS* tables are highly reliable.

4.3 Opponent Playing Styles

According to Rupeneite [2010], the playing style of an opponent can be classified into four categories. Each style is distinct, in that it describes the opponent's frequency of play and how the player bets. The four categories of playing styles are Loose/Passive, Loose/Aggressive, Tight/Passive and Tight/Aggressive. A brief description of each style is shown in Table 3 below:

The Aggressive Factor, AF, is used to classify a player as either Aggressive or Passive. The formula for AF is as follows:

$$AF = \frac{\#\ raises}{\#\ calls}$$

Players can be classified into either Aggressive or Passive by the percentage of games they have played. Based on research by Rupeneite [2010], a threshold of 1 is used:

- Aggressive if $AF > 1$

Playing Styles	Description
Tight	Plays few hands and often folds.
Loose	Plays multiple and varied hands.
Aggressive	Bets and raises a lot, almost always never checking or call.
Passive	Usually checks and call, unlikely to take the lead.

Table 3: Description of Playing Styles

- Passive if $AF \leq 1$

The Player Tightness, PT, is used to classify a player as either Loose or Tight. The formula for PT is as follows:

$$PT = \frac{\#\ folds}{\#\ games}$$

Players can be classified into either Loose or Tight by the percentage of games they have played. Based on research by Rupeneite [2010], a threshold of 0.28 is used:

- Tight if $PT < 0.28$ hands
- Loose if $PT \geq 0.28$ hands

Later, a classification process, introduced by Dinis and Reis [2008], is conducted to classify the opponent's style of play into four categories, as seen in Table 4.

	AF $\leq$ 1	**AF $>$ 1**
PT $\geq$ 0.28	Loose Passive	Loose Aggressive
PT $<$ 0.28	Tight Passive	Tight Aggressive

Table 4: Style of Play Classification

Acknowledgments

The preparation of this report would not have been possible without the help of Dr. Yair Zick and Arka Maity, National University of Singapore, School of Computing.

References

[Dinis and Reis, 2008] Felix Dinis and Luis Paulo Reis. An experimental approach to online opponent modeling in texas hold'em poker. In *Advances in Artificial Intelligence*, pages 83–92, Savador, Brazil, October 2008. Brazilian Symposium on Artificial Intelligence.

[Rupeneite, 2010] Annija Rupeneite. *Building Poker Agent Using Reinforcement Learning with Neural Networks*. SCITEPRESS, 2010.

[Russell and Norvig, 2014] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2014.

[Schaul *et al.*, 2016] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. In *Proceedings of the International Conference on*

Learning Representations 2016, pages 1–21, San Juan, Puerto Rico, May 2016. Canada Institute for Scientific and Technical Information.

[Teofilo *et al.*, 2013a] Luis Filipe Teofilo, Luis Paulo Reis, and Henrique Lopes Cardoso. Computing card probabilities in texas hold'em. In *Proceedings of the 8th Iberian Conference on Information Systems and Technologies*, pages 988–993, Lisbon, Portugal, June 2013. Canada Institute for Scientific and Technical Information.

[Teofilo *et al.*, 2013b] Luis Filipe Teofilo, Luis Paulo Reis, and Henrique Lopes Cardoso. Speeding-up poker game abstraction computation: Average rank strength. In *Proceedings of the 27th Association for the Advancement of Artificial Intelligence Workshop*, pages 59–64, Bellevue, Washington, USA, July 2013. Association for the Advancement of Artificial Intelligence.