Applying Reinforcement Learning to Kuhn Poker : With Gradient Reward

Manish Kumar, Prateek Bhat and Ritesh Agarwal
School of Informatics and Computing
Department of Computer Science
Indiana University, Bloomington, IN
[*kumar*20, *pratbhat and riteagar*] @ iu.edu

***Abstract*—Solving Poker using Reinforcement Learning is a difficult task, there are many challenges that need to be taken care of while learning poker like the uncertainties of the card being dealt, partial observability of the environment and dynamic opponent modelling. Though Kuhn poker is a toned down and heavily modified version of regular poker where the domain is restricted to only two players and a deck of three cards (A, K, Q), the agent still faces quite a difficult task while facing a non-stationary agent like ours. Our aim is to model an opponent based on human behaviour and examine how our agent learn from scratch, which is another challenge. In this paper, we try to evaluate some of the basic strategies employed in the field of Reinforcement Learning with the addition of discrete gradient reward/penalty which is a function of the amount of pot won/lost in a hand respectively.**

***Keywords*—Reinforcement Learning, Kuhn Poker, Opponent Modelling, Expected SARSA Lambda, Dyna-Q+**

I. Introduction

Artificial Intelligence researchers have always been fascinated with the Game theory field. This field has been an active area of research for the past 60 years. From backgammon to Go, every milestone in the field of game theory has been proven to be a cornerstone in the field of Artificial Intelligence. This field intrigues scientist all over the world because of the challenges involved in modelling the opponent. To model the opponent means predicting the future behaviour of an opponent from possibly incomplete observation of past behaviour of the opponent[5].Limited observations of the past, incomplete information of opponent's dynamic nature, stochasticity of the environment etc. are some of the major reasons that make opponent modelling a challenging task. Opponent modelling find it's usefulness in real time application as well, interaction of a robot with the humans is an example where human acts as an opponent.

Poker is an ideal example where opponent modelling presents all the above mentioned challenges along with challenging human emotions. Extremely large number of states results in very large game tree and increased stochasticity of the environment which makes opponent modelling in poker a very challenging task. Thereby, to reduce the game complexity but still keeping the basic intricacies of opponent modelling intact Kuhn poker plays an interesting role for this problem domain.

In Reinforcement Learning, opponent modelling can be seen as a specific instance of general problem domain for state estimation[5]. For each possible state of opponent, likelihood of an action out of the available actions can be estimated by the corresponding state action value pair learnt over the past behaviour of the opponent. So, our ultimate goal is to learn the true state action value for the opponent by our agent. In the past, work has been done in Kuhn poker by applying some of the well known reinforcement algorithms with decent success. In this paper, we try to evaluate some of the basic strategies employed in the field of Reinforcement Learning with the addition of discrete gradient reward/penalty which is a function of the amount of pot won/lost in a hand. We applied Expected SARSA Lambda and Dyna-Q+ with gradient reward+penalty and compared their performances in this context.

II. Related Work

Poker has been a relevant platform for research in artificial intelligence from past 60 years or so. Kuhn Poker maintains the crux of the real poker game by being a game of partial information (hidden cards) and stochastic environment (dealing cards).

Alessandro and his colleagues have explained in [3] about how the knowledge of the opponent strategies can effect the learning process of an agent. The results showed that agent having no prior information about the opponent strategies is better most of the times than having partial information. This urged us not to give our agent any prior information other than the rules of poker.

The authors in [2] give attention to parameter learning and strategy learning. The experiments were done for six opponent strategies. For parameter learning Nash equilibrium (safe strategy) was used for exploration and only after the opponent model is learned, the algorithm shifts to exploitation strategy. For strategy learning, an algorithm is implemented that combines exploration and exploitation by not making any preconceived notions about the opponent. The results still do not show significant improvements in learning for a small number of hands due to the stochastic nature of the game.

Nolan and Michael in their paper [5] have used particle filtering for both static and dynamic agent modelling. The results against static opponents were not that impressive as the approach was specifically designed for dynamic opponents. They use switching and drifting to distinguish between dynamic agents and the experiments were carried out on known motion models, unknown motion parameters and unknown model. The results for the unknown motion parameters against highly switching opponent (like ours) were similar to those of static model. The results for unknown model were reasonable but still very poor against a highly switching opponent where the total winnings is less than $10 over 1000 hands when each player can bet $1.

Opponent modeling in [6] is done for 'Leduc hold 'em', an advanced version of Kuhn poker, which increases the challenges faced by the agent. The authors make certain assumptions like having a stationary opponent and independent strategies. Probability of a showdown or a fold for an opponent strategy is calculated based on the information set which has the all the necessary information. Response strategies such as Bayesian Best Response (BBR) depends on observing past hands and maximizing the expected value based on opponent strategy and all possible hands, Max A Posterior (MAP) responds based on the most probable opponent strategy or Thompson response , based on calculating posterior probability from a set of opponent strategy were used. The results show that MAP and Thompson's performs better than BBR for a non-stationary agent but the average winning rate decreases gradually for both.

III. Requirements of a Kuhn Poker player

The basic requirements of a good poker player were first introduced in *Opponent Modelling in Poker*[1] and have been discussed below:

- **Hand Strength** : In Kuhn poker the hand strength will depend upon the face value of the card dealt i.e. A > K > Q.
- **Betting Strategy** : will depend upon the hand strength, the capital in hand, the state of the game which will include the the current value of the pot and opponent state of game play.
- **Bluffing** : is a strategy where a player can try to intimidate its opponent by betting thus giving a false impression of a high strength card in hand.
- **Opponent Modelling** : is an abbreviation for looking into your opponent's head. It is the process of learning the way the opponent plays for example when does the opponent bet does it lose or win? or when does it bluff and what are the consequences? By observing all these actions we try to form a transition model of our opponent which will improve as the game progresses.

IV. Kuhn Poker and Opponent Strategy

Kuhn Poker is an ideal environment to apply Reinforcement Learning. Though a toned down version of poker, by removing the complexities of the real game we still maintain it's essence and the problem still remains difficult to crack($Effective\ Short-Term\ Opponent\ Exploitation\ in\ Simplified\ Poker$[2]). The difficulty in making our agent learn to play Kuhn poker was to learn the nuances of the game like when to bluff given a card of low strength in hand.In order to learn the game of Kuhn poker we have to have an opponent which the agent will try to model. The designing of the adversary has been studied in a short paper, On the Usefulness of Opponent Modeling: the Kuhn Poker case study.

Our agent is always given the first option to play. The cards are dealt at random to each player out of a deck of three cards (A, K and Q), with A being the

highest card after K and Q respectively. The agent and the opponent start with a capital of 100 each. The minimum bet to play a single game for each player was set to 2 which is sometimes called as an ante.

A. Model of the Opponent

Previous research conducted in the field of opponent modelling tried to model fixed opponents and dynamic opponents. Fixed opponent in many papers was defined as an adversary that will take an action with some fixed probability, for example an adversary might bet with a probability of 0.8 and make a pass with a probability of 0.2. Dynamic opponent on the other hand was defined to change its policy after playing certain hands or for every hand discussed in related work above. We have tried our model against stationary and non-stationary opponents[5].

Stationary Agent: For stationary opponent the probability for pass and bet was kept constant and was tried with different values out of which the agent performance with opponent setting as Pass:40% and Bet:60% is shown in figure 3. The opponent wins most of the time in this setting, so we shift our focus towards modelling non-stationary agent.

Non-stationary Agent: We have tried to make the adversary take actions based on some human behaviour. The policy followed by the adversary will depend on the card dealt to it by the dealer(environment):

- Card in hand = A: The adversary will always make a bet.
- Card in Hand = K: The adversary will make a bet with a probability of $\frac{Capital\ in\ hand}{Max.\ amount\ a\ player\ can\ have}$. In the setup for this experiment the maximum amount any player can have is 200 (Players start with an amount of 100 each).
- Card in hand = Q: As the probability of winning with Q in hand is very less, the probability of bluffing is given as $\frac{Capital\ in\ hand}{2 * Max.\ amount\ a\ player\ can\ have}$

The adversary will play dynamically where the probability of betting is a function of the capital in hand. Thus, more the capital in hand higher will be the probability of bluffing and vice verse.

V. Opponent Modelling

We have tried three different strategies to model the fixed opponent. We define Gradient Reward as a constant function which depends on the amount of pot won or lost by our agent. The introduction of gradient reward proved to be quite an improvement over the constant assignment of reward no matter how much the agent wins or loses.

In our experiments we measured the performance of the agent by calculating how many times it bankrupted the opponent i.e. after playing a certain number of games with the opponent, we calculate the number of times the agent was able to win all the capital of it's opponent. We will define an event of bankruptcy as one episode and a hand as one step for the rest of the paper.

The different experimental setups have been discussed in detail below:

1) At first we tried to play against the opponent with a random agent. We defined the randomness of the agent actions independent of the card in hand or the opponent's action. The results obtained were pretty bad, as the agent wasn't able to bankrupt the opponent even once. This was to be considered as a base case for our agent performance with other strategies.

2) In the second case we introduced Reinforcement Learning to our agent, in which the environment gives a reward of +1 when the agent wins a hand and a penalty of 0 for loosing a hand. It was our prior notion that playing the agent in this setting will boost the performance i.e. it will bankrupt the opponent significant amount of times. The result on the other hand was disappointing as the agent was still not able to bankrupt the opponent even once.

3) Getting some inspiration from a real poker game, were winning a pot of higher amount makes the player a lot happier. We gave the agent different positive rewards as forth called Gradient Reward, for different amounts of winnings. In the environment defined by us, the agent will win different amount of pots depending on the action chosen by the players.

   - **Pot = 4, Reward = +0.6**: This is the minimum amount of pot that an agent can win. This will happen when both agent

and the opponent pass in their first turn, thus the amount in the pot will be the ante that was placed in the beginning of the game.

- **Pot = 6, Reward = +0.8**: This is the intermediate amount that can be won by the agent. It will happen when the agent places a bet in the first turn and the opponent makes a pass in the second.
- **Pot = 8, Reward = +1**: This is the maximum amount that the agent can win in a hand. This will happen when both the agent and opponent make a bet in their respective turns.

The gradient positive reward proved to be a significant improvement over the base case as the agent was able to bankrupt the opponent in the first 50 experiments itself.

4) To further improve our agents performance, we introduced negative reward as forth called Gradient Reward+Penalty. The positive gradient reward remains the same for this setup as discussed in 3), in addition to that we give the negative gradient penalty depending on the amount of pot won by the opponent.
   - **Pot = 4, Penalty = 0**: This will be minimum amount of pot that the opponent can win, which will be the same as the ante placed by both agent and opponent to enter the game.
   - **Pot = 6, Penalty = -0.4**: This is the intermediate amount that the opponent can win in a hand. This will happen when agent passes in the first turn and the opponent makes a bet and in the third turn agent makes a pass.
   - **Pot = 8, Penalty = -0.6**: This is the highest amount of pot that an opponent can win in a hand. This will happen when both the agent and the opponent make a bet in their respective turns.

VI. Results

In this paper we have tried to emulate the emotions of a player in Kuhn poker with the help of Gradient Reward+Penalty, which played a significant role in improving the agent's performance for initial episodes. There was a huge boost in the winning percentage of the agent with the addition of gradient penalty. The penalization of agent's bad performance helps it to learn at a faster rate, but in the latter stages of the game, the strategy in case 3 outperforms the strategy in case 4.

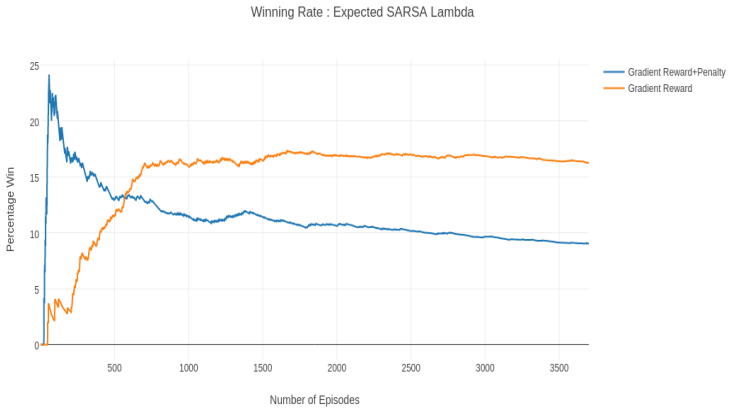

Fig. 1. Comparison of Gradient Reward with Gradient Reward+Penalty: Expected SARSA Lambda

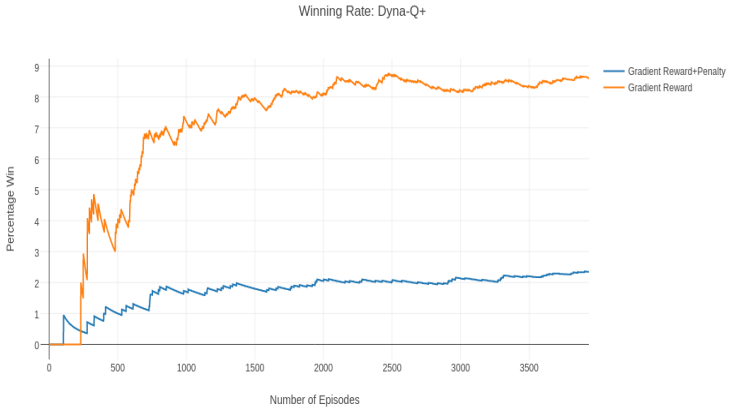

Fig. 2. Comparison of Gradient Reward with Gradient Reward+Penalty: Dyna-Q+

Figure 1 summarizes the performance of the agent using SARSA Lambda algorithm. Gradient Reward+Penalty outperforms our Gradient Reward in beginning but in latter stages the winning percentage becomes constant at around 9% whereas the agent playing with just Gradient Reward learns slowly and outperforms with a constant winning rate of 16%.

Figure 2 on the other hand shows the performance of the agent using Dyna-Q+. In this case for reward without penalty the agent wins 8.5% of the time while with penalty the winning percentage is 2.2%

In Figure 3 our agent is able to model a stationary opponent(Bet:60% and Pass:40%) and wins almost all the games which have been shown for first 500 episodes. Case with and without gradient penalty

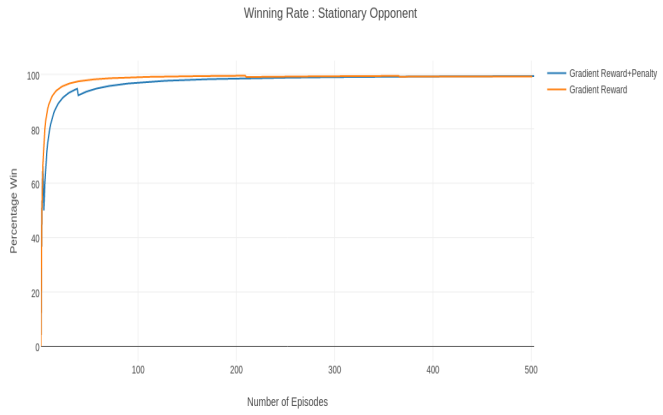

Fig. 3. Comparison of Gradient Reward with Gradient Reward+Penalty: Stationary Opponent

have been shown in the graph.

VII. Conclusion

In this paper, our work focuses on effectively learning state action value estimates for a non stationary opponent in Kuhn poker. First, we started our experiment with the stationary opponent model and the results show that learning stationary opponent model was very fast for our agent and it comprehensively beats the opponent. Later on, we experiment with non stationary opponent for Expected Sarsa (Lambda) and Dyna-Q+ with fixed reward to make agent learn the state value estimates for the opponent, results obtained were quite disappointing. Inspired by the poker game in realistic environment, we gave different reward, depending on the amount of pot won or lost. This change resulted in significant improvement in performance in comparison to the fixed reward. We can also draw conclude that Expected SARSA Lambda performs better than Dyna-Q+ in this problem setting. This gives us inspiration to experiment more with Kuhn poker and build a reasonable AI agent against any dynamic opponent.

VIII. Future work

We have implemented Dyna Q+ and Expected SARSA Lambda as Reinforcement Learning algorithms with only one set of parameter values but different learning rates etc. may be implemented for better results. Reward is a constant function of pot in our case but it can be a monotonically increasing function with respect to the amount won or lost. Comparison can be made between our algorithm and particle filtering[5] based on our opponent model and analysis can be done to improve the results. More algorithms can be implemented for better decision making based on previous hands which is not taken care of in this paper. To understand the stochastic nature of the opponent better opponent modeling needs to be done with faster learning as our state action pair currently has a large value and it may take a lot of time to learn.

References

[1] Darse Billings, Denis Papp, Jonathan Schaeffer, Duane Szafron, *Opponent Modeling in Poker.*

[2] Bret Hoehn Finnegan Southey Robert C. Holte and Valeriy Bulitko, *Effective Short – Term Opponent Exploitation in Simplified Poker.*

[3] Alessandro Lazaric, Mario Quaresimale and Marcello Restelli, *On the Usefulness of Opponent Modeling : the Kuhn Poker case study.*

[4] Duane Szafron, Richard Gibson and Nathan Sturtevant, *A Parameterized Family of Equilibrium Profiles for Three – Player Kuhn Poker.*

[5] Nolan Bard and Michael Bowling, *Particle Filtering for Dynamic Agent Modelling in Simplified Poker.*

[6] Finnegan Southey, Michael Bowling, Bryce Larson, Carmelo Piccione, Neil Burch, Darse Billings, Chris Rayner, *Bayes′ Bluff : Opponent Modelling in Poker.*