Limit Hold'em Poker Playing using Reinforcement Learning

Anmol Prasad
IIIT Delhi
2016012

Jai Mahajan
IIIT Delhi
2016154

Suryaansh Mata
IIIT Delhi
2016272

I. Problem Statement/Motivation

Game Playing has been a field of interest in Artificial Intelligence, as they have a clear set of predefined rules. However in order for AI to play them at a competent level, complex strategies may be required. The advancements in AI have mostly been in the area of deterministic games, like Chess or Checkers, where complete information is available. However, research is lacking for games with incomplete information.

In this project, we attempted to develop an agent for a game with incomplete information (Poker) . In poker, an agent should have the ability to estimate it's opponent's actions and the current state of the table to take an appropriate action to maximize it's chances of winning. The objective of this project was to :

- Estimate the opponent's 'hand-strength' by the means of their actions.
- Classification of a given player according to the number of hands played by them into categories "Loose Aggressive/Loose Passive/Tight Aggressive and Tight Passive"
- Learn a policy to win against an opponent using Reinforcement Learning

II. Literature Review

We have followed the "Building Poker Agent Using Reinforcement Learning with Neural Networks" [2] paper throughout and we took inspiration from them to understand various methods to solve various subparts arising in the game of poker. The authors of the paper worked on hand/table strength estimation, opponent type, hand strength classification and the current state evaluation.We tried to find more papers for the same topic, but we could not find many. In fact, one of the reasons we faced problems in the implementation was because we could not find enough information about the problem.

III. The Dataset

There were very few datasets which were relevant to limit hold'em poker playing and we found the IRC dataset[1] created by the Poker Research group at the University of Alberta to be the most relevant, given our problem. This dataset is large and consists of all the IRC poker games played between 1995 and 2001 which means more than 10 million games. Only the Limit hold'em folder was extracted from the dataset to be used for our problem.

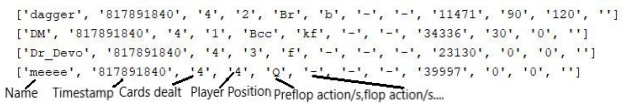
The dataset was created by recording the live games as they were going on, by an observer bot. Therefore only the players who haven't folded till the showdown (which is after the River) have their cards revealed ultimately and thereby shown to all the players, including the observer bot. As a consequence of this the hole cards, for all the players are not revealed in the dataset, and we only get to know the cards for the players that did not leave the game in that particular round.

We used the data of one such table(199512) from the given dataset with 93 players. We arranged the data in such a way that for a particular player all his games were linked to the table cards, pot size at each round and the actions taken up by the player. The player actions were read from the database and according to their history, their playing style was analyzed, according to which they were classified as "Loose Aggressive/Loose Passive/Tight Aggressive and Tight Passive" as described in the original paper[2].

We have classified for all the limit holdem tables, which ended up in classifying 3763,229,803,29 players as loose passive,loose aggressive,tight passive & tight aggressive respectively which is based on number of hands a player plays.

This dataset was also used to estimate the hand strength of a player on the basis of the actions taken by them such as Call, Raise, Check and so on and the value of the pot at that point in time to determine the estimates "hand strength" of the opponent.

Figure 1:Data of a particular game Extracted in Python.



IV. Methodology

We solved three subproblems of this game using RL and Neural Networks

- Determining Hand Strength of opponent given the actions of the opponent using Neural Networks.
- Classifying player type (passive/aggressive or tight/loose) which was done using the past history of the player games and maintaining counts of the required quantities.
- Trained a poker playing agent that learns given the table cards, player actions, agent's estimated hand strength and some past information about the opponent.

For the first objective. we extracted the individual players from games wherein the players' cards were revealed in that game, i.e the player played till the very end. A vector was made for each of these players representing features including the number of Calls, Raises and Checks made by them, as well as the amount of the pot of the table. This model was trained using a Deep Neural Network using Keras and Tensorflow backend, upon which we did some hyperparameter tuning improve the accuracy of the predictions. What was observed was aligned to the actual game wherein the Player Hand Strength was high when they made more number of raises and the Pot was also high.

The second objective was met by taking into consideration the player history where we counted the number of calls, raises and checks made by the player. The player was classified into different labels on the basis of different ratios taken using these data.

For the last model we trained the agent to come very close to playing the actual game of Poker by applying Reinforcement Learning and learning from it's past mistakes and rewards. For every game, the agent gets trained to a specific kind of player and it learns what it should do in cases where it gets a similar hand strength and/or a similar sort of player in the future. Some Opponent player types can be Safe Player, Player that always Calls a bet, a Random Player. Using the "PyPokerEngine" we simulated such playing styles of opponents and trained the RL agent against these players with different playing styles.

V. Game Environment Simulation

The Texas Hold'em Limit Poker consists of essentially two major components, which is the
1) Deck of Cards and it's evaluation and the
2) Betting Rounds that follow each community card reveal, ultimately ending with the show-down and the winner.

Our project makes use of the "Deuces"[3] module, whose Python compatible version is "Treys", originally written for the MIT Pokerbots Competition. It is lightweight and extremely fast Python library, used to effectively evaluate the hand strength of a set of cards varying form 3 to 7 cards. The poker betting round has been hard-coded for the different actions of Call, Raise, Check and Fold. Finally the Winner is the one amongst the players remaining on the table during the show-down with the strongest hand.

While the Hard Coded environment was used to solve the objectives 1 and 2 of classifying the Players as "Loose Aggressive/Loose Passive/Tight Aggressive and Tight Passive" as well as the Player Hand Strength Estimation on the actions taken by the Player during a betting round, we made use of the "PyPokerEngine" to simulate the Poker Game Environment during Reinforcement Learning for meeting objective 3. This is a highly effective library for the Game Simulation and one can easily define the States that are to be used for the RL Algorithm.

Figure2: Extracted from the proposal doc.

The Game

Poker has 4 stages in the game namely pre-flop, flop, turn, river.
1. Pre-flop- two cards are dealt for every player.
2. Flop- three table cards are shown.
3. Turn- fourth card is shown and finally
4. River- fifth card is shown and winner is determined.

Game winner is a player with the strongest five card combination as per the priority image given.

VI. Results

The first classifier we made was able to learn the value of each action such as Call, Raise, Check and so on and the value of the pot at that point in time in determining the hand strength of the game. For eg- The model was able to learn that a raise would mean a better set of cards generally compared to a call/check. When the value in the pot was high, thereby implying that the general cards the opponent held were of high rank, led to a better Strength.

The classification on the type of player was manually verified and we realized how a simple method could solve such an interesting problem, accurately with precision.

Lastly, the RL based agent the we made to learn the details was smart enough to learn opponent playing patterns on the basis of the Community Cards, the Betting Actions and the Hand Strength. We hope our RL based bot would become a top player soon !

VII. Some things not to be followed

We realized later in into the project that the data set is a very important parameter to determine the type of approach one can apply for a problem. In our case the biggest issue we faced was the face that the player cards are known to the bot only if a player stays in the game until the showdown. So when a game like poker that is already having a lot of incomplete information and unknowns, such information not being available makes it even tougher. We found only one dataset for the Texas Holdem Poker and since it was collected by an observer bot, most of the information was missing. Also one must not underestimate the data extraction step for the project as it might take a lot of time for refinement and making it usable by the algorithms.

Also, games such as poker with no certain fixed moves in spite of all the rules and limited number of Cards in the Deck, rely heavily on the human judgement and analysis, as there is always a tendency of bluffing. Such randomness and subtleties make it extremely difficult to model them into some fixed set of rules or boundaries. While designing the Poker environment, to train the RL bot thus it became difficult to model exact human behavior into the opponent's gameplay.

VIII. Takeaway from the results

We realized that subproblems of the game of poker were themselves were not easy to solve and even after solving there were various factors in the game including playing style change of a person which could affect the models trained. For instance if a "Safe Player " suddenly starts playing aggressively, the agent would not be able to recognise this transition in most of the cases.

IX. References

1. http://poker.cs.ualberta.ca/irc_poker_database.html (Link to Dataset)
2. http://www.scitepress.org/Papers/2014/51489/51489.pdf
3. https://github.com/worldveil/deuces
4. https://github.com/ishikota/PyPokerEngine
5. https://github.com/worldveil/deuces