Research and General Information

Bachelor Thesis, 2017

Fabian Moik

Contents

1 State of the Art **3**
1.1 Requirements for a Poker Playing Agent . . . 3
1.1.1 Hand Strength Estimation . . . 3
1.1.2 Hand Potential . . . 3
1.1.3 Bluffing . . . 3
1.1.4 Unpredictability . . . 3
1.1.5 Opponent Modeling . . . 3
1.2 Different types of poker bots . . . 4
1.2.1 Knowledge-based poker agents . . . 4
1.2.2 Monte-Carlo Simulation . . . 4
1.2.3 e-Nash Equilibrium based poker agents . . . 5
1.2.4 Exploitive poker agents . . . 5
1.2.5 Bayesian poker agents and evolutionary algorithms . . . 5

2 Evolutionary Neural Network **7**
2.1 Evolutionary algorithms . . . 7

3 Testbed **8**

4 Structure and Input of Neural Network **9**
4.1 General Structure . . . 9
4.2 Feature-List . . . 9
4.3 Effective Hand Strength (EHS) . . . 9
4.4 Chip count . . . 9
4.5 Chips in pot . . . 9
4.6 Chips to call . . . 10
4.7 Number of opponents . . . 10
4.8 Chip counts of all players . . . 10
4.9 Position of hero . . . 10
4.10 Opponent model . . . 10

5 Calculating the EHS **11**
5.1 Pre-flop . . . 11

6 Basic Structure of Bachelor Thesis **12**

1 State of the Art

1.1 Requirements for a Poker Playing Agent

The main reason why creating a poker-playing system which plays at a high level is hard, is the fact that poker is a game of imperfect information [6]. A reasonably good poker bot needs to fulfill a set of requirements which includes **hand strength evaluation**, **hand potential**, **betting strategy**, **bluffing** and **opponent modeling**. Furthermore it needs to stay **unpredictable** for the opponents.[1]
Some of these requirements may be handled indirectly by the poker-playing system, rather than explicitly by design. Nonetheless all of the requirements must be solved at least in some way for the system to succeed against the best human players. There are many approaches for solving each requirement and it is yet not possible to say which approach is best for which requirement. [2]

1.1.1 Hand Strength Estimation

In its simplest form it is used to determine the probability of winning a hand, given the current board texture (community cards) and the hole cards of the player. The **most successful** methods are using a **Monte Carlo Sampling** technique. The idea behind this method is to sample the remaining unknown cards and play the hand to the end. After a sufficient number of iterations the wins are counted and the hand strength is therefore estimated. In addition more factors could be taken into account such as the *number of players still in the game*, *hero's position at the table* and the *betting action* until the current point in the game [1].

1.1.2 Hand Potential

Hand potential describes the probability of a hand improving or getting beat as further community cards get into play on later streets. At it's core hand potential is a function of hero's hole cards and the already dealt community cards. For an even better calculation all factors described in the hand strength estimation could be used in addition [1].

1.1.3 Bluffing

Bluffing is essential to succeed in the game of poker. From a game theoretical point of view, there is an optimal bluffing frequency in certain situations. A system with a minimal bluffing strategy would bluff exactly that percentage of hands randomly. However, a better bluffing strategy would identify profitable bluffing opportunities by predicting the probability of an opponent to fold. [2]

1.1.4 Unpredictability

This part of the system allows to hide information from the opponents by varying strategies for a given situation over time. [2]

1.1.5 Opponent Modeling

Opponent modeling describes the method of modeling a likely probability distribution of the opponent's hand. In it's simplest form a single generic model could be used for all opponents.

However, to improve the system the probabilities can be modified based on the collected data of each opponent. [2]

All of these requirements combined form a **betting strategy**, which determines the optimal play whether to *fold*, *call* or *raise* in a certain situation.

1.2 Different types of poker bots

The Annual Computer Poker Competition takes place each summer at the *AAAI Conference on Artificial Intelligence* or at the *International Joint Conference on Artificial Intelligence* and attracts competitors from all around the world to compete in the challenge to improve existing methods or develop new ideas and algorithms for beating complex variants of poker. [5]

Following link redirects to the AAAI official website where results of recent competitions can be found.
`http://www.computerpokercompetition.org/`

The official University of Alberta Research Team website offers a vast majority of poker AI research and is the starting point for most novices in poker AI. Many years of research have produced a number of precious articles and papers which can be found on their website.
`http://poker.cs.ualberta.ca/index.html`

1.2.1 Knowledge-based poker agents

Knowledge-based systems can be broken into two categories, namely *rule-based expert systems* and *formula-based methods*. In general knowledge-based systems require the knowledge of an expert player to design the system[11].A **rule-based expert system** in its simplest form is a sequence of *if-else* statements for frequently occurring scenarios of the game. Human players describe their poker hands in a very similar way when they break them down in a discussion [7]. **Formula-based methods** on the other hand are a more generalized system which takes a collection of weighted inputs that describe the current game state and then outputs a probability triple upon which a betting decision is made. An example for a typical input to the formula is the hand strength and pot odds.

1.2.2 Monte-Carlo Simulation

In general simulation based methods rely on repeatedly simulating an outcome in order to obtain a statistical average [7]. A frequently used simulation method applied to the game of poker is the so called *Monte-Carlo simulation or Monte-Carlo Tree Search*, which is a procedure that searches the game tree by drawing random samples for choices in a game state and simulates the play until a leaf node (last betting round when five community cards are present) is reached [11]. At this point the payoff value for one path is known and the procedure is repeated until the values for intermediate nodes in the tree converge to a robust evaluation value [11]. In poker this approach can be highly volatile because the outcome of the simulation highly depend on the quality of the simulated play. Future actions in a hand are not only determined by the knowledge of the opponent's hole cards and the community cards but also on the type of opponent one is facing, hence biased values for certain situations might lead to inaccurate simulation outcomes. [7]

1.2.3 e-Nash Equilibrium based poker agents

The currently best performing poker-playing programs are approximating a *Nash equilibrium* [4]. In game theory, the Nash equilibrium is a solution concept of a non-cooperative game involving two or more players in which each player is assumed to know the equilibrium strategies of the other players, and no player has anything to gain by changing only his own strategy [14].

The most successful bots for Heads-up No-Limit (HUNL) Hold'em poker all use the same basic structure. They are based on using a smaller abstract version of the game in which they then approximate the nash equilibrium and execute the result in the original game via a translation method. In the abstract version of the game the bot abstracts the card information by clustering hands with similar strength and potential. Furthermore it abstracts the betting information by restricting the available bets to a small number of crafted bets, dependent on the current size of the pot. [4]

The currently most used and most successful algorithm for approximating the Nash equilibrium in the abstract game is called **Counterfactual Regret Minimization**. All three top performing players in ACPC 2016 used this algorithm and outperformed the rest of the competition. [4]

1.2.4 Exploitive poker agents

Eventhough Nash-equilibrium approaches are static and robust strategies, they will never identify and exploit opponents' weaknesses because equilibrium strategies limit their own exploitability by assuming that they play against a perfect opponent who also plays an equilibrium style.[11] To maximize the winnings against a weak opponent good poker agents attempt to adapt to their opponents' playing style by constructing accurate opponent models which alters their equilibrium play, therefore making them exploitable too. attempt to adapt to their opponents' playing style by constructing accurate opponent models

1.2.5 Bayesian poker agents and evolutionary algorithms

Bayesian poker agents

A Bayesian network is a probabilistic graphical model. Each node in the directed acyclic graph represents a random variable and edges between nodes represent the conditional dependencies of variables. Nodes are associated with a probability function which returns the conditional probability values based on the values of the parent node. Initializing nodes within the network with different values results in propagated probabilities throughout the network which yield a probability distribution over the random variables. Compared to other poker playing agents, bayesian agents performed badly in the AAAI Computer Poker Competitions in the last years. There is still a lot of room for improvement in Bayesian based networks and further research in this field has to be done to stand a chance in upcoming competitions. [11]

Evolutionary Algorithms

Evolutionary algorithms evolve a population of agents over successive generations. A *fitness function* typically decides which agents are more likely to proceed to the next generation.The offspring is then created via a *crossover* procedure. Additionally a procedure called *coevolution* can be used to restrict the evolution process of a population. For this purpose separate genetic populations are maintained and evolution can only occur within these separated populations.

In combination with evolutionary algorithms, neural networks are often used for the poker playing agents which compete against each other over a number of generations. The best ranking agents are then selected to act as parents for the next generation. In the crossover procedure the weights of the evolved neural networks are calculated using a biased sum over the parents' weights.[11]

2 Evolutionary Neural Network

Unlike other games such as Chess, Checkers or Go, Poker is a game of imperfect information. This means, that a player can not see all relevant information at a given game state at all times. The game of Poker involves hidden information and deception. and therefore players must be willing to take risks, based on the information they have [8]. Imperfect information includes a very large decision space and therefore efficient computational calculations which search for the optimal decision path may be unfeasible for many problems, including the No-Limit Hold'em version of poker. Unlike other methods, which reduce the decision space by transforming the game into a smaller abstract version in which they then approximate the optimal path and execute the result in the original game via a translation method, my approach is to use an evolutionary algorithm on agents improving their game by training a neural network through iterative play. [8]

2.1 Evolutionary algorithms

"Evolutionary algorithms mimic natural evolution, and reward good decisions while punishing less desirable ones.[8]" Agents make betting decisions based on given input features to the neural network, which represent the current game state as detailed as possible. The evolutionary algorithm is then applied to the neural network itself. It may add or remove certain features and change the weights of the parameters [8]. Adding random noise to the weights provides a crucial feature of natural evolution - Mutation. A small percentage of random mutation allows agents not to get stuck in local optima. Before the evolution of a population can occur the population first needs to be evaluated. To find a number of suitable parents to form the next generation all agents of the current population compete against each other in a number of tournaments where their finishing place is directly correlated to their fitness. The fittest agents are then chosen for reproduction. Together with the best agents of the previous generation the newly born agents for the new generation and once again compete against each other in a number of tournaments. The exact procedure of evolving a population has not yet been established since it **will not** be the aim of this work. The goal is rather to set up a test environment where agents can be evaluated and sorted by their strength of play.

3 Testbed

In order to test a population of agents against each other, a suited test bed is needed. Creating a Poker playing agent requires a lot of computation and simulation. It starts with calculating the chances of winning a hand given the current state of the game considering the positive and negative hand potential for all possible opponent's hand combinations. Certain probabilities can be precalculated and saved as lookup tables others require real time calculations and simulations. Some open source libraries can be found, which help with the hand evaluation process and also offer some other handy features for creating a dynamic test bed. Such libraries and APIs are, for example, linked on the *University of Alberta Computer Poker Research Group* website. There already exist some test beds for poker playing agents but most of them do not provide simulations for tournaments or are too slow and their architecture make it hard to create or implement new agents [13]. Unfortunately most of the libraries are Java libraries but since they are open source it should not be too hard to port them to other languages such as C++. On reason for building the test bed in C++ is the speed of the language and my programming knowledge in it, compared to Java. But I have not yet decided if I want to fully implement my test bed in C++ (or Python) or try to use the existing libraries to some extend in their native language and combine both languages.

With the evolutionary approach of creating a Poker playing agent, an efficient test bed which can simulate hundreds of thousands of games per hour is needed. The general idea is to write a program which simulates a Poker tournament with *x tables* and *9 agents* per table. Each agent uses a neural network for its betting decisions. The inputs for the neural network are provided by a class which holds all the table information. The test bed then runs thousands of tournaments and keeps track of the final placements of each agent. After x number of tournaments, the fittest agents are selected to form a new generation. Some other features can be implemented in the simulation environment for instance a table seat permutation would reduce the variance of the end results and could hence speed-up the simulations by reducing the number of needed simulations [13]. A precise explanation on the workflow of the evolutionary algorithm can not be given at this moment, because the idea of this bachelor thesis is to setup the test bed and the neural networks of the playing agents including the intense calculations of the input features to the neural network. The final goal is to use this thesis as a cornerstone for the future work on a fully functional and autonomous poker playing agent.

4 Structure and Input of Neural Network

4.1 General Structure

At this point it is not yet decided if two neural networks will be used, one for the pre-flop game and one for the post-flop game, or if one neural network for both game stages would be sufficient. Under the assumption of two separate networks for each stage of the game the preliminary structure of the post-flop neural network consists of an input layer with 22 neurons, one hidden layer with a yet unknown number of neurons and an output layer with three neurons. Following inputs are necessary to keep track of the evolution of the table and are listed in 4.1. The three output neurons represent the so called **probability triple** (f, c, r), which specifies the probability distribution for the actions *fold, check/call* or *bet/raise* at the current state of the game [11]. According to this distribution an action is then chosen by the agent.

4.2 Feature-List

Input	Feature
1	Effective Hand Strength
2	Chip count
3	Chips in pot
4	Chips to call
5	Number of opponents
6	Position of hero
7-14	Chip count of all opponents
15-22	Opponent model

Table 4.1: *Table of inputs (features) for the neural network*

4.3 Effective Hand Strength (EHS)

The EHS in conjunction with other components should be used to help selecting a suitable betting action. It is an indicator for how likely a hand is winning at showdown considering the current hand strength but also the positive and negative hand potential.

4.4 Chip count

The second feature is the number of chips the acting agent has.

4.5 Chips in pot

The third feature is the number of chips already in the pot, including all chips of previous betting rounds plus the number of chips betted by other agents in the current betting round. An agent may not be able to win all the chips in the pot if his chip count (stack) is smaller than a bet of

his opponents in the current round. Hence the value of this feature is the effective pot size an agent can actually win when his hand is the strongest at showdown.

4.6 Chips to call

The fourth feature is the number of chips an agent has to match if it wants to continue in the hand. Combined with the third feature this forms the so called **pot odds**, which represent the bet to pot ratio and is a commonly used term and tool in the Poker community to calculate the needed winning percentage of an agent's hand to make the call a profitable play.

4.7 Number of opponents

This feature is the number of opponents currently involved in the hand.

4.8 Chip counts of all players

In a 9-handed multi table tournament there are at most 9 players on a table. Therefore this feature has 9 inputs, one for each player on the table. The input is the number of chips an opponent has. Late in a tournament it will occur that hands are not always played 9-handed but with less then 9 players per hand. For this case the input of empty seats on the table is set to unknown and will be recognized by the neural network as an empty seat.

4.9 Position of hero

This feature is the relative position of the agent to the dealer. The dealer position is the most valuable position in Poker because the action of only two more players will follow in an unopened pot. In general it is desirable to be in a late position because many players have already acted before you have to do and hence more information is available for the agent in late position.

4.10 Opponent model

The opponent model should indicate which type of opponent the agent is facing. For this purpose a combination of *VPIP (Voluntarily Put Money In Pot), PFR (Pre-flop Raise)* and *AF (aggression factor)* is used. The VPIP of the player is the percentage of hands in which a player voluntarily puts money in the pot, PFR is the percentage of hands in which a player puts in a pre-flop raise and the AF is the number of times a player bets plus the number of times he raises divided by the number of times he calls (Bets + Raises)/Calls. All three values together give a good estimation of the overall playing style of the opponent. In general there are four basic Poker playing styles: Tight Passive, Loose Passive, Tight Aggressive (TAG) and Loose Aggressive (LAG). Aggression is key in winning at Poker. Passive playing styles will never win at the game of poker.

5 Calculating the EHS

To calculate the winning chances of a poker hand and comparing it with the cost of playing the hand is essential to make profitable long term decisions. A hand evaluation algorithm tries to approximate the chances of winning a hand at a given state in the current game and a given the opponent models. Simply calculating the current hand strength is insufficient if there are community cards still to come. Therefore a good hand evaluation algorithm for post-flop play takes the present hand strength and the hand potential on later betting streets into account.[9]

For the pre-flop evaluation the hand potential is not considered due to the computational complexity and hence a different evaluation is used for pre-flop play.[9]
At this point of time it is not yet clear which methods will be used for calculating the EHS. Since there are some open source projects which already have a hand evaluator implemented, not all calculations will be done manually. Some fast evaluators are from the GPL pokersource project and can be found on SourceForge.net. They offer hand evaluators for C, Java and Python. Though I am not yet sure how good and how well suited those 'ready to go' hand evaluators are. This is the reason why I can not yet tell how the hand evaluation will be done in this work.

5.1 Pre-flop

A hand evaluation for the pre-flop stage of the game is not very complex since there is a very limited number of possible holdings and there is little information before the flop that could influence a betting decision.Factors which play a role in the hand evaluation before the flop are the two hole cards of the player, previous player actions, opponent models from previous hands (not always given) and the position at the table relative to the dealer [2]. There are a total of $\binom{52}{2}) = 1326$ different hands in poker, of which many of them are equivalent before the flop [12]. For example a hand such as A♠ 2♠ and A♣ 2♣ have the same chance of making a flush and the same overall chance of winning. With this additional information there are 169 distinct hand types left (13 paired hands, $\binom{13}{2} = 78$ suited hands and $13 \cdot \binom{4}{2} = 78 unsuited hands$) [9]. With a method known as *roll-out simulation* tables can be calculated offline [12].
For each of the 169 distinct hand types a simple roll-out simulation of around 1 million hands can be done against each of one to nine opponents. The opponents simply call the big blind and then check all the way to the showdown. This methods yields a statistical measure of the approximate hand strength of each starting hand. Although this method assumes a simplified scenario where each player calls to the end, it however gives a good first approximation of how strong a starting hand is. [9] As described in [2] the simple *roll-out simulation* can be refined by letting the previous results govern the betting decision of each player. A net negative value of the income rate (IR) in the previous iteration would therefore mean, that the player should rather fold the hand instead of calling the big blind. This method would produce a more realistic distribution of opponents and would define a set of hands that can be played profitably against the blind and other unknown hands but it does nether take the betting position nor the know opponent actions into account.[2] Therefore a simple *roll-out simulation* could as well be sufficient for an adequate pre-flop play.

6 Basic Structure of Bachelor Thesis

1.) Introduction to the Game of Poker
   1.1) No-Limit Hold'em

2.) Poker as a Test Bed for AI
   2.1) History of Poker Bots
   2.2) Breakdown of Obstacles in the Creation of a Poker Bot

3.) Evolutionary Algorithms in Conjunction with Neural Networks
   3.1) What is an Evolutionary Algorithm
   3.2) What are Neural Networks

4.) Creating a Poker Bot Using AE and NN
   4.1) Implementation of the Test Bed
      4.1.1) Calculation of EHS
   4.2) Implementation of the NN Agents
      4.2.1) Feature Set
   4.3) Testing Agents by Self-Play

5.) Future work (Master Thesis)
   5.1) Using EA to Make a Strong Poker Playing Agent

Bibliography

[1] Darse Billings et al., *Using Selective-Sampling Simulations in Poker*, Department of Computing Science - University of Alberta, Edmonton, Alberta Canada, 1999.

[2] Darse Billings et al., *The Challenge of Poker*, Department of Computing Science - University of Alberta, Edmonton, Alberta Canada, June 22, 2001.

[3] Darse Billings , *Algorithms and assessment in computer poker*, Department of Computing Science - University of Alberta, Edmonton, Alberta Canada, 2006.

[4] Viliam Lisy and Michael Bowling , *Eqilibrium Approximation Quality of Current No-Limit Poker Bots*, Department of Computing Science - University of Alberta, Alberta Machine Intelligence Institute, Jan. 8, 2017.

[5] Super User(Dec. 27, 2012). *About the ACPC*. Retrieved from http://www.computerpokercompetition.org/index.php/about,

[6] Tjebbe Laurens Vlieg, *Computer Poker with Neural Networks*, University of Amsterdam - Faculty of Science, June 26, 2012.

[7] Nuno Miguel da Silva Passos, *Poker Learner: Reinforcement Learning Applied to Texas Hold'em Poker*, Faculdade de Engenharia da Universidade do Porto, July 18, 2011.

[8] Garrett Nicolai and Robert Hilderman, *Algorithms for Evolving No-Limit Texas Hold'em Poker Playing Agents*, Dalhousie University, Halifax, Canada; Department of Computer Science, University of Regina, Regina, Canada, Sep, 2010.

[9] Danis Richard Papp, *Dealing with Imperfect Information in Poker*, University of Alberta, Fall, 1998.

[10] Darse Billings, Denis Papp, Jonathan Schaeffer, Duane Szafron, *Opponent Modeling in Poker*, Department of Computing Science, University of Alberta, 1998.

[11] Jonathan Rubin, Ian Watson, *Computer poker: A review*, Department of Computing Science, University of Auckland, New Zealand, Jan. 21, 2011.

[12] Aaron Davidson, *Opponent Modeling in Poker: Learning and Acting in a Hostile and Uncertain Environment*, Department of Computing Science, University of Alberta, Summer, 2002.

[13] Francesca Giardini, Frederic Amblard, *Multi-Agent-Based Simulation XIII*, Springer, Revised Selected Papers, International Workshop MABS, June, 2012.

[14] Martin J. Osborne, Ariel Rubinstein, *A Course in Game Theory*, The MIT Press, Cambridge, Massachusetts, London, England, Jan. 07, 2014.