Bachelor Thesis - Poker Simulator

Thesis Relevant Blocks

Fabian Moik

Inhaltsverzeichnis

1 Structure of the Thesis

- Abstract
- Introduction
  - Objective of thesis
  - Outline of thesis
- What is No-Limit Texas Hold'em
  - What is poker
  - Poker language
  - Basic Poker Strategy
  - How will the Neural network try to apply and learn strategies?
- Test Environment (How the testbed was constructed and what is it capable off?)
- Neural Network Agents trained with EA
  - structure of NN
  - implementation of EA
    - improvements (Hall of Fame, varying the fitness function)
- results of the experiment
- future improvement and work

2 Introduction

3 No-Limit Texas Hold'Em Poker

3.1 What is Poker

Talk about what poker is in general and also talk about, why it is a perfect candidate for testing unsupervised machine learning techniques such as evolutionary learning.

3.2 Poker Lingua

Is that a scientific work or just call it language?

3.3 Basic profitable Poker Strategy

3.3.1 Different Types of Poker Players

Talk about a general sound poker strategy and shortly describe the game theoretic standpoint and how it compares to more practicable playing strategy

LAG

TAG

...

Which strategy is known to be the best one against a wide variety of opponents and which strategy outperforms against which? -> **Keep that as short as possible**

3.4 What does this machine learning program try to learn the agent and what is the desired strategy if there is any.

Talk about what the expected behavior of the agents were beforehand and what would be the optimal behavior if that can even be said at this point. Would it be a game theoretic approach or would other strategies work even better for the test environment?

4 Test Environment

4.1 What it consists of

4.2 How it was implemented

Why C++ was choosen as the programming language? -> **because of the speed for calculations**

4.3 What does it provide

4.4 How it is used to train the agents

4.5 What are remarkable features of the testbed compared to already existing ones

4.5.1 Some alternative Testbeds and why they were not used

5 Evolution of Neural Network Agents

5.1 Overview

5.2 The Neural Network Structure

5.2.1 The Input & Hidden Layer

The Winning Chance

Round of The Game (Preflop, Flop, Turn, River

How many Turns have passed

Hero Chip Count

Chips in Pot

Chips to Call

Number of Opponents

number of agents currently in the hand, not counting ourself

Hero's Position

Chip Count of all Opponents

The first one is always the chip count of the first opponent immediately to the left of the current active player. For empty seats a value of 0 is assigned to this feature.

Opponent model

//TODO not yet implemented -> **optional**

5.2.2 The Output Layer

5.3 Evolutionary learning

5.3.1 Evolution Phase

5.3.2 Evaluating the Fitness

5.3.3 Pitfalls

5.3.4 Countermeasures against common pitfalls

6 Results of this Study

7 Future Improvements and Work

8 Thesis Blocks

8.1 Calculating Hand Strength

How is EHS calculated:
https://en.wikipedia.org/wiki/Poker_Effective_Hand_Strength_(EHS)_algorithm

Interesting read on EHS implementation techniques:
http://poker-ai.org/archive/www.pokerai.org/pf3/viewtopicfdcf.html?f=3&t=444&st=0&sk=t&sd=a&start=40

Github of EHS implementation algo:
https://github.com/Pip3r4o/Bluffasaurus

Paper on calculating the EHS: [1]

A full consideration of the whole remaining deck against two opponants would take $47^6 = 7.73{\cdot}10^9$ calculations, and hence a lot of time. For that reason the solution to this calculation problem is to use the *Monte Carlo Method*. It considers a small random subset of card sequences. The result though is just an approximation, but experience has shown that the error of the calculation is very small, so the Monte Carlo Method is a reliable solution. [1]

To make the EHS calculation even better one need to consider a hand distribution for each player. This has the effect, that the calculation considers the players strength and observing enough hand of his opponents.

Further improvements would be to generate an opponentn model for each opponent and also give it as input to the EHS calculation.

8.1.1 Hand Strength

Calculating the hand strength can be done by comparing hero's hand against every combination of opponent hands on a given board. **pseudo-code on wiki page**. This technique can also be used to calc the probability against multiple opponents, by raising the resulting probability to the power of opponents. [1]

To take into account the opponent models one could use this algorithm but instead of iterating over every possible hand one only uses the **Sklansky Groups**. [1]

8.1.2 Calculating the EHS - Preflop

What about the *Chen Formula*? Can only be used for Preflop but it is fast for preflop! What are the **alternatives**?

Also have a look at the oopoker project it has the malmuth/Sklansky group implemented plus the other method.

One way is to use the so-called **all-in equities** of all distinct starting hands (169) vs #of-Opponents. I found a table online with all the equity percentage points for 1 to 8 opponents. In this paper the process of roll-out simulation is explained. [3]

8.1.3 Calculating the EHS - Flop

We can use the 2+2 evaluator for 5-6-7 card evaluations!!! need to implement it!! A really good explaination is given in [2]
We need a HandRanker for 5cards, 6cards and 7cards (2+2) can do that i guess...

8.1.4 Calculating the EHS - Turn

A really good explaination is given in [2]

8.1.5 Calculating the EHS - River

on the river only the handstrength against opponents is needed

8.1.6 Calculating the EHS - Flop

How to calculate the EHS vs multiple opponents? have a read hear:
http://www.poker-ai.org/archive/pokerai.org/pf3/viewtopiceb2e

8.2 Betting Decision for Neural Network Agents

The neural networks output layer has three neurons representing FOLD, CALL and RAISE. Furthermore the RAISE action was later divided into 3 sub-categories namely *small raise medium raise* and *large raise*. the decider of wether a raise is small, medium or large is the players own chip count. The implemtentation used the proposed betting system, which can be found here [4]. The normal distributions and there sigma calculation can be found at page 55 and 56 of this thesis.

8.3 Evolving the Neural Network Agents

To improve the fitness function from only evaluating the agents by there placing another fitness function is introduced. The combined weight of them is our new fitness function:

$$HandsWonScore = plainWeight \cdot plainRanking + HandsWonWeight \cdot HandsWonRanking \tag{8.1}$$

where plain weight is the weight of the placement fitnessfunction (also called *plain fitness function*).

Talk in the paper about the try against random agents without hallOfFame and

with only the plain fitnessfunction of average placement in tournement! -> Quickly found the strategy of always folding

9 Open Questions & Thoughts

9.0.1 Open Bugs and Fixes

- is Raise action for > stack a valid allInAction? -> **for now it is**

- if on flop two all in and two active players, and the first active player folds -> second active player shouldn't have the opportunity to doTurn

- check the logic behind settleBets() and betsSettled() because if one player can decide and the rest is out or all in while in settleBets() the one player can still bet -> **but here it should stop and go to showdown**

  if there are two guys in the betting round with stack greater 0 and if one goes all in and the ai should decide now, there would only be one active player left because the other guys just went all-in. To provide input we should now say there is one opponent.

  there is a lot of improvement when it comes to multiway all in on different betting rounds. E.g. if there where 3 guys all in preflop and two guys are on the flop where now the one guy moves all in, the ai should rather only consider his all-in action therefor as input it should only get 1 opponent for the calculation, because the others only contribute to the sidepot

- **BUGS found via debugging with logfile**

  AI: RAISE command with 0 amount?

  cardDealer: is lastRaiseAmount calculated correctly? (sb - wager: 13, raise 131 (highesW: 23 - lastRaiseAmount: 121) ... bb - wager: 23, raise 755 (highestW: 131 - lastRaiseAmount: 634))

  FOLD action should not be possible when CHECK is possible - should also not be possible when only checks in front of you! Then change FOLD to CHECK instead

  lastRaiseAmount on unopened flop should be 0 not the bigblind amount!

  two people before the last dealer busted and one guy joined to the right of the dealer and now the old dealer also is the new dealer. new guy at table was also old dealer on other table!! -> **expected result?** shouldn't it be that the new guy is on cutoff and next guy is now official dealer? (most fair!) – also what happens is that on the table where the new player is taken from on player is overjump and the next player is dealer (e.g player 10 was dealer and gets reseated, next hand player 12 is dealer but player 11 should be)

  At a second example the seating behaviour was as expected! -> DIFFERENCE only one busted in this e.g, didn't work when 2 busted simultanously

  AI: if it wants to raise but has less then the calling chips, just change RAISE action to CALL action

  AI: implement betting system that never raises smaller than bigblind

  cardDealer: when two players left and bigblind has less then bigblind but more than small blind, then the small blind does not have to cover the bet and showdown is done... -> bet should be covered -> because now there is a but that the guy how didnt cover bet only wins double his money and not everything when only one guy left, one more hand is player where he plays against himself.... get rid of this extra game

raise amount is not calculated correctly!

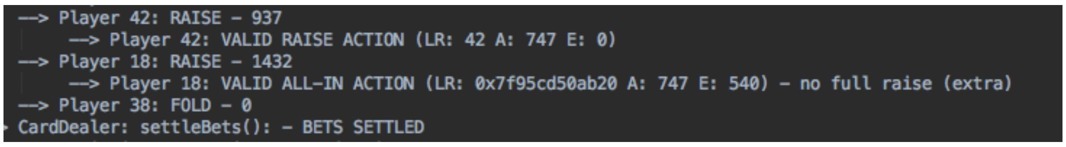
cap the noise added amount so total value not greater than 1 or less than -1

AI: how to sample from output vector? should i take the highest value or sample with distribution?

- AI: **Output of AI neural network is not always normed to 1 and has always the same outputs... something is wrong here.**

  somehow to handsWon seams not working correctly as is the neural network!!! -> Have a look!!!!

- check if the mean Money Won fitnessfunction makes sense
- **BUG** with the last raiser index! When all in action of no full size with extra at LR index there is an adress and the action stops there



```
--> Player 42: RAISE - 937
      --> Player 42: VALID RAISE ACTION (LR: 42 A: 747 E: 0)
--> Player 18: RAISE - 1432
      --> Player 18: VALID ALL-IN ACTION (LR: 0x7f95cd50ab20 A: 747 E: 540) - no full raise (extra)
--> Player 38: FOLD - 0
CardDealer: settleBets(): - BETS SETTLED
```

9.0.2 Things To Implement

- **A UnitTest like in oopoker_master which tests importent functions for correctness**
- **Structurize the code and make it more efficient**
- **approximate the execution time**
- **Neural Network:** how to normalize input values if distribution is unknown???

  Answer:

  Additional feature (totalchips in game as BB), and other chip features also in bb Additionally normalize all chip features with the totalchips feature in bb and normalize other features to 0...1.
- Last layer is a softmax activation function so it gets a classification characteristic
- **Evolutionary Algorithm:** How to evolve our agents?
- how to order agents by places when multiple agents bust within one hand?

  Answer: guy with less chips has worst position... -> **for now not treated**
- **Effective Hand Strength** as feature -> see how it is calculated and then try to do it with pokereval.h and pokereval2.h

  https://en.wikipedia.org/wiki/Poker_Effective_Hand_Strength_(EHS)_algorithm

  github for EHS calculation algo: https://github.com/Pip3r4o/Bluffasaurus
- interesting read on EHS for multiple opponents and general implementation strategies.. http://poker-ai.org/archive/www.pokerai.org/pf3/viewtopicfdcf.html?f=3&t=444&st=0&sk=t&sd=
- **Change the ordering of numerating opponent hands and all possible turn and river combinations -> performance improvement?**
- **Things to implement next:**

  betting strategy

  missing input features

  checking if game logic is correct
- make them train vs another population aswell and only evolve 24 players the rest stays the same
- check if the distributions for a high bet are right
- HALL OF FAME
- increase the number of players in a tournement
- **Change the betting strategy** because I will never reach the case of high raise because most of the time the neural network fixes the output values to a fixed number which only varies slightly!
- think of a way to make a better PREFLOP decision because bot fold too many hand preflop
- improve the debugging to better see what the network is doing
- implement a different fitness function where fitness is meassured via return of money %

9.0.3 Open Problems

- When agent decides to raise he will always raise until the other one folds or he is allin! There is no reevaluation after a betting round.

Literaturverzeichnis

[1] Luis Filipe Teofilo, *Estimating the Probability of Winning for Texas Hold'em Poker Agents*, Departamento de Engenharia Informatica, Faculdade de Engenharia da Universidade do Porto, Portugal, yeart unknown.

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

[3] Darse Billings, *Algorithms and Assesment in ComputerPoker*, Department of Computing Science, University of Alberta, 2006.

[4] Garret Joseph Nicolai, *Evolutionary Methods for Learning No-Limit Texas Hold'em Poker*, Department of Computing Science, University of Reginia, 2008.

1 Poker Simulator

1.1 Basic Structure

1.1.1 Currently working on

- lets players make moves and check when a betting round has finished

  TODO: think of all scenarios where different people are the last raiser (bigblind, small blind, no one, did some go all-in? etc...)
- next check for action to be valid
- the **Action** class should check if the action is valid -> see rules for raising

  test the isValidAction method for all scenarios
- create an **Information class** which holds all the information that is accessible to an AI.
- reseating players if needed, to balance game
- make a concept of what you need for your neural network and how the training is done.

  the training is purely done by evolving the weights and features there for no backprop algorithm or gradient calculation is needed

1.1.2 Ideas and Implementation Hierarchy

- A **game** is run simultaneously on **x tables**

  each **table** holds **y players**, a **deck of cards** and a **dealer (observer)**
- the **dealer** deals the cards, and holds informations about the game state
- the **statsKeeper** class keeps track of the opponent models
- Information class, that holds all the information of the tables but also for each player

Preferably each table runs on it's own thread, a coordinator makes sure that tables wait if reseating has to be done.

1.1.3 Open Bugs and Fixes

- is Raise action for > stack a valid allInAction? -> **for now it is**
- allInAction for less than a real raise should be treated like **call + extra**, meaning, the lastRaiseAmount stays the same but a new Raise has to be twice the lastRaiseAmount + extra
- if on flop two all in and two active players, and the first active player folds -> second active player shouldn't have the opportunity to doTurn
- change the way and all-in action is treaded (for now a raise of > stack size is not treated as all-in, it has to be exactly the same)

1.1.4 Things To Implement

- A UnitTest likle in oopoker_master which tests importent functions for correctness
- porting the neural network from python to c++ (using tensorflow api or write own neural network?)
- **Neural Network:** how to normalize input values if distribution is unknown???

  Answer:

  Additional feature (totalchips in game as BB), and other chip features also in bb Additionally normalize all chip features with the totalchips feature in bb and normalize other features to 0...1.
- Last layer is a softmax activation function so it gets a classification characteristic
- **Evolutionary Algorithm:** How to evolve our agents?
- how to order agents by places when multiple agents bust within one hand?

  Answer: guy with less chips has worst position...
- **Effective Hand Strength** as feature -> see how it is calculated and then try to do it with pokereval.h and pokereval2.h

  https://en.wikipedia.org/wiki/Poker_Effective_Hand_Strength_(EHS)_algorithm

  github for EHS calculation algo: https://github.com/Pip3r4o/Bluffasaurus
- interesting read on EHS for multiple opponents and general implementation strategies.. http://poker-ai.org/archive/www.pokerai.org/pf3/viewtopicfdcf.html?f=3&t=444&st=0&sk=t&sd=

1.2 Game Play Plan

Pregame

- get number of players
- calculate number of tables and cardDealers
- give players a buy-in and assign them to tables

Once all players have their buy-in and seat on a table, the game can start

Game Start

Preflop

- **Dealer:** assign dealer button to player
- **Players:** post SB and BB and antes
- **Dealer:** shuffle deck of cards
- **Dealer:** deal 2 cards to each player
- **Dealer:** tell first player after BB to play action
  Players: make a move
  Dealer: check if action is valid
  Dealer: repeat with next player
- **Dealer:** after all players acted:
  calculate statistics
  update table

Flop

- **Dealer:** burn one card, and deal flop
- **Dealer:** tell first player (SB) to play action
  Players: make a move
  Dealer: check if action is valid
  Dealer: repeat with next player
- **Dealer:** after all players acted:
  calculate statistics
  update table

1 Thesis Blocks

1.1 Calculating Hand Strength

How is EHS calculated:
https://en.wikipedia.org/wiki/Poker_Effective_Hand_Strength_(EHS)_algorithm

Interesting read on EHS implementation techniques:
http://poker-ai.org/archive/www.pokerai.org/pf3/viewtopicfdcf.html?f=3&t=444&st=0
&sk=t&sd=a&start=40

Github of EHS implementation algo:
https://github.com/Pip3r4o/Bluffasaurus

Paper on calculating the EHS: [1]

A full consideration of the whole remaining deck against two opponants would take $47^6 = 7.73{\cdot}10^9$ calculations, and hence a lot of time. For that reason the solution to this calculation problem is to use the *Monte Carlo Method*. It considers a small random subset of card sequences. The result though is just an approximation, but experience has shown that the error of the calculation is very small, so the Monte Carlo Method is a reliable solution. [1]

To make the EHS calculation even better one need to consider a hand distribution for each player. This has the effect, that the calculation considers the players strength and observing enough hand of his opponents.

Further improvements would be to generate an opponentn model for each opponent and also give it as input to the EHS calculation.

1.1.1 Hand Strength

Calculating the hand strength can be done by comparing hero's hand against every combination of opponent hands on a given board. **pseudo-code on wiki page**. This technique can also be used to calc the probability against multiple opponents, by raising the resulting probability to the power of opponents. [1]

To take into account the opponent models one could use this algorithm but instead of iterating over every possible hand one only uses the **Sklansky Groups**. [1]

1.1.2 Calculating the EHS - Preflop

What about the *Chen Formula*? Can only be used for Preflop but it is fast for preflop! What are the **alternatives**?

Also have a look at the oopoker project it has the malmuth/Sklansky group implemented plus the other method.

One way is to use the so-called **all-in equities** of all distinct starting hands (169) vs #of-Opponents. I found a table online with all the equity percentage points for 1 to 8 opponents. In this paper the process of roll-out simulation is explained. [3]

1.1.3 Calculating the EHS - Flop

We can use the 2+2 evaluator for 5-6-7 card evaluations!!! need to implement it!! A really good explaination is given in [2]
We need a HandRanker for 5cards, 6cards and 7cards (2+2) can do that i guess...

1.1.4 Calculating the EHS - Turn

A really good explaination is given in [2]

1.1.5 Calculating the EHS - River

on the river only the handstrength against opponents is needed

1.1.6 Calculating the EHS - Flop

How to calculate the EHS vs multiple opponents? have a read hear:
http://www.poker-ai.org/archive/pokerai.org/pf3/viewtopiceb2e

1.2 Betting Decision for Neural Network Agents

The neural networks output layer has three neurons representing FOLD, CALL and RAISE. Furthermore the RAISE action was later divided into 3 sub-categories namely *small raise medium raise* and *large raise*. the decider of wether a raise is small, medium or large is the players own chip count. The implemtentation used the proposed betting system, which can be found here [4]. The normal distributions and there sigma calculation can be found at page 55 and 56 of this thesis.

1.3 Evolving the Neural Network Agents

To improve the fitness function from only evaluating the agents by there placing another fitness function is introduced. The combined weight of them is our new fitness function:

$$HandsWonScore = plainWeight \cdot plainRanking + HandsWonWeight \cdot HandsWonRanking \tag{1.1}$$

where plain weight is the weight of the placement fitnessfunction (also called *plain fitness function*).

Talk in the paper about the try against random agents without hallOfFame and

2 Open Questions & Thoughts

2.0.1 Open Bugs and Fixes

- is Raise action for > stack a valid allInAction? -> **for now it is**
- if on flop two all in and two active players, and the first active player folds -> second active player shouldn't have the opportunity to doTurn
- check the logic behind settleBets() and betsSettled() because if one player can decide and the rest is out or all in while in settleBets() the one player can still bet -> **but here it should stop and go to showdown**

  if there are two guys in the betting round with stack greater 0 and if one goes all in and the ai should decide now, there would only be one active player left because the other guys just went all-in. To provide input we should now say there is one opponent.

  there is a lot of improvement when it comes to multiway all in on different betting rounds. E.g. if there where 3 guys all in preflop and two guys are on the flop where now the one guy moves all in, the ai should rather only consider his all-in action therefor as input it should only get 1 opponent for the calculation, because the others only contribute to the sidepot

- **BUGS found via debugging with logfile**

  AI: RAISE command with 0 amount?

  cardDealer: is lastRaiseAmount calculated correctly? (sb - wager: 13, raise 131 (highesW: 23 - lastRaiseAmount: 121) ... bb - wager: 23, raise 755 (highestW: 131 - lastRaiseAmount: 634))

  FOLD action should not be possible when CHECK is possible - should also not be possible when only checks in front of you! Then change FOLD to CHECK instead

  lastRaiseAmount on unopened flop should be 0 not the bigblind amount!

  two people before the last dealer busted and one guy joined to the right of the dealer and now the old dealer also is the new dealer. new guy at table was also old dealer on other table!! -> **expected result?** shouldn't it be that the new guy is on cutoff and next guy is now official dealer? (most fair!) – also what happens is that on the table where the new player is taken from on player is overjump and the next player is dealer (e.g player 10 was dealer and gets reseated, next hand player 12 is dealer but player 11 should be)

  At a second example the seating behaviour was as expected! -> DIFFERENCE only one busted in this e.g, didn't work when 2 busted simultanously

  AI: if it wants to raise but has less then the calling chips, just change RAISE action to CALL action

  AI: implement betting system that never raises smaller than bigblind

  cardDealer: when two players left and bigblind has less then bigblind but more than small blind, then the small blind does not have to cover the bet and showdown is done... -> bet should be covered -> because now there is a but that the guy how didnt cover bet only wins double his money and not everything when only one guy left, one more hand is player where he plays against himself.... get rid of this extra game

raise amount is not calculated correctly!

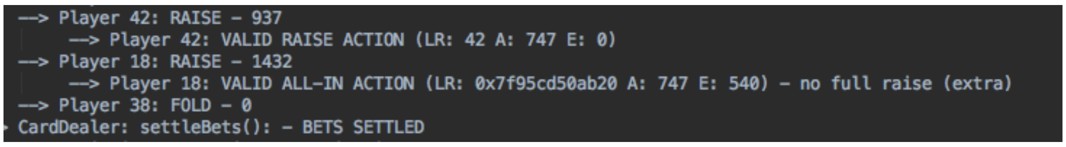
cap the noise added amount so total value not greater than 1 or less than -1

AI: how to sample from output vector? should i take the highest value or sample with distribution?

- AI: **Output of AI neural network is not always normed to 1 and has always the same outputs... something is wrong here.**

  somehow to handsWon seams not working correctly as is the neural network!!! -> Have a look!!!!

- check if the mean Money Won fitnessfunction makes sense
- **BUG** with the last raiser index! When all in action of no full size with extra at LR index there is an adress and the action stops there



```
--> Player 42: RAISE - 937
     --> Player 42: VALID RAISE ACTION (LR: 42 A: 747 E: 0)
--> Player 18: RAISE - 1432
     --> Player 18: VALID ALL-IN ACTION (LR: 0x7f95cd50ab20 A: 747 E: 540) - no full raise (extra)
--> Player 38: FOLD - 0
CardDealer: settleBets(): - BETS SETTLED
```

2.0.2 Things To Implement

- **A UnitTest like in oopoker_master which tests importent functions for correctness**
- **Structurize the code and make it more efficient**
- **approximate the execution time**
- **Neural Network:** how to normalize input values if distribution is unknown???

  Answer:

  Additional feature (totalchips in game as BB), and other chip features also in bb Additionally normalize all chip features with the totalchips feature in bb and normalize other features to 0...1.
- Last layer is a softmax activation function so it gets a classification characteristic
- **Evolutionary Algorithm:** How to evolve our agents?
- how to order agents by places when multiple agents bust within one hand?

  Answer: guy with less chips has worst position... -> **for now not treated**
- **Effective Hand Strength** as feature -> see how it is calculated and then try to do it with pokereval.h and pokereval2.h

  https://en.wikipedia.org/wiki/Poker_Effective_Hand_Strength_(EHS)_algorithm

  github for EHS calculation algo: https://github.com/Pip3r4o/Bluffasaurus
- interesting read on EHS for multiple opponents and general implementation strategies.. http://poker-ai.org/archive/www.pokerai.org/pf3/viewtopicfdcf.html?f=3&t=444&st=0&sk=t&sd=
- **Change the ordering of numerating opponent hands and all possible turn and river combinations -> performance improvement?**
- **Things to implement next:**

  betting strategy

  missing input features

  checking if game logic is correct
- make them train vs another population aswell and only evolve 24 players the rest stays the same
- check if the distributions for a high bet are right
- HALL OF FAME
- increase the number of players in a tournement
- **Change the betting strategy** because I will never reach the case of high raise because most of the time the neural network fixes the output values to a fixed number which only varies slightly!
- think of a way to make a better PREFLOP decision because bot fold too many hand preflop
- improve the debugging to better see what the network is doing

Literaturverzeichnis

[1] Luis Filipe Teofilo, *Estimating the Probability of Winning for Texas Hold'em Poker Agents*, Departamento de Engenharia Informatica, Faculdade de Engenharia da Universidade do Porto, Portugal, yeart unknown.

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

[3] Darse Billings, *Algorithms and Assesment in ComputerPoker*, Department of Computing Science, University of Alberta, 2006.

[4] Garret Joseph Nicolai, *Evolutionary Methods for Learning No-Limit Texas Hold'em Poker*, Department of Computing Science, University of Reginia, 2008.