ADAPTIVE POKER BOT BASED ON PLAYER-MODELLING

Bjrn Hasager Vinther & Nicolai Guldbk Holst
bhas@itu.dk ngul@itu.dk

A thesis submitted for the degree of

Bachelor Softwaredevelopment

May 2014

Abstract

Poker is a game which have a lot of different aspects which makes it a very interesting game to implement artificial intelligence (AI) into. The game is unlike games like chess and backgammon as poker has the problem that the information is imperfect. This is also one of the reasons why poker is interesting topic in regards to artificial intelligence as it opens up for some problematics as how to handle the information we have in regards to what we do not have. Because of this hidden information there are many decision to decide from, as we have to take into account all the information that we have access to and the possible information which is hidden from us. Another thing that makes poker interesting to have a look at is that we now have opponents who has the opportunity to trick us into thinking that he is in another state than what he actually is. In this bachelor thesis we will present a way to implement a neural network in a poker game to teach the bots how to play Texas Holdem Poker. The neural network will keep evolving along the opponents to ensure that the bot has the best possible chances of winning or minimize the loses. We will test the bots against real life human players in many suitable situations and from these results we can determine how effective our neural network is at teaching our bots how to play the game.

Preface

This is a bachelor thesis consisting of 15 ETCS points. The thesis has been written exclusively by Nicolai Guldbk Holst and Bjrn Hasager Vinther, both studying *Software development* at the IT University of Copenhagen. It has been written in the spring semester 2015. Our supervisor was Kasper Sty.

Introduction

Poker is the most popular card game in the world [9]. It involves intelligence, psychology and luck. Poker has a lot of hidden informations which requires the players to make decisions based on qualified guesses.

In a game of poker analysing your opponents is a major part. Being able to predict your opponents hands, future actions and reactions can give you a huge advantage. Top players are able to read their opponents and adapt their strategy in order to gain an advantage. Most rounds are won before the showdown simply by one player outplaying the others. Another big element of poker is statistics. Since you don't know the cards that will be dealt throughout the game players must calculate the likelihood of their hand ending up being the winning hand.

For humans it is often easy to recognise various patterns whereas it can be very challenging for a computer. On the other hand computers can perform thousands of computations in a matter of seconds. Even for most humans players it is really hard to read your opponents and it is even harder for a computer.

The goal of this thesis is to see if it is possible to train a computer to play poker and adapt to its opponents strategy. In order to achieve this goal we first need to answer the following questions:

1. How can we predict the probability of ending up with the winning hand?

2. Can the computer learn to play poker by observing humans playing poker, and if so how can we implement it?

3. How can the computer adapt to the opponents strategies?

Contents

1 Default play style

1.1 Introduction

As we cannot have a bot which can outplay the opponents from first round we will have to create a default player. The default player will have the task of lowering our costs while the system gathers information about the opponent so the system can adapt to his style of play. There are different ways of how to achieve this, firstly we could simply just hardcore a decent way of playing into the game and make decisions based on the cards only. The other method is to implement a neural network and train it till we are satisfied with the results. The neural network seemed like the best idea for further development so this will be our way to go.

1.2 test

1.3 Discussion

When our artificial intelligence starts out by playing the games of poker it doesn't have any kind of information about the opponents. So to make sure that the bot wont just go keep throwing away the bank roll. Instead we wanted the bot to minimizes our losses so that we would still have a decent bank roll when we have gathered information about the opponents so that the bot could make qualified guesses at what move would be the most appropriate in terms of the current opponent. The default player was never meant to be on the same level as a human player, but humans are only in a slightly better position than the default bot. The human player can like the bot only see the hole and community cards, but a human is also able to make a profile of the bot in their head. This means that they can learn how the bot decides and exploit this. This is one of the reasons that we chose to go with a neural network. In a neural network we are able to of course train the network to make correct decisions based on the targeted output that we give it. But as the game proceeds the neural network has an input which will weight when we have enough information about the opponents to shift our gameplay from the default play style to a more adaptive one. The time that it takes a human player to learn about the bot and adapt to it, should be the same for the bot. So if we imagine a human player who is good enough to decipher the way the bot is playing and adapt to it. When the human player does that, the bot should also have started to change its ways. Slowly as the bot learn it will adapt more and more to the opponent so when we have enough information the bot will shift completely and disregard the default play style.

1.4 Conclusion

2 Adapt to opponents

2.1 Introduction

In attempt to make an artificial intelligence that is able to play along human players, we will have to adapt to opponents by modeling each player we are playing against. How can we use neural networks to model a player? And how can we use this to make our artificial intelligence a better poker player?

2.2 What is player-modeling?

If we want to make the best possible moves we will have to player model each player, to make the system adapt to their gameplay. Player modeling is basically when we model a player characteristics and the way they behave throughout the game to gain knowledge about the individual player. In poker there are many aspects of the game which could alter the players behaviour, such as position relatively to the dealer, the pot size, active players etc. Therefore it will be crucial to take atleast some of these things into perspective when trying to model a player. One of the reasons why it is necessary to player model every player is because we need to have the opportunity to exploit a weak player. If one keeps making weak moves that we can predict it would be a wise consideration to exploit it. In a game of poker there will always be good and bad ways of handling a given situation, and this means there will also be an optimal way of handling each players moves. The pro players has to be good at adapting to an opponent. Even if the opponent changes their gameplay throughout the game, one must be able to adapt to this, and that is why we need to constantly player model each player.

2.3 How can we model a player dynamically?

To player model each player the system has to look at the games previous history while dynamic learning as the game proceeds. This can be implemented using a neural network and will help us reach our goal of predicting the opponents cards, next move or as a minimum what strength their dealt hand has. By using neural network we can model a player throughout the game. First the neural network of course take many inputs but one of them would be about a specific player and their history, the choices they made, their chips, cards etc. From that the neural network will continue to receive inputs about the player so that player modeling can be dynamic. Another big advantages of a neural network is that we are able to give a lot of inputs. These inputs will then be weighted by the system in order to come closer to our targeted output. This will help us cut out all of the noise that occur and leave us with data that is relevant.

2.3.1 Neural Network

To player model each player the system has to look at the games previous history while dynamic learning as the game proceeds. We decided to implement a neural network to achieve our goal of predicting the opponents cards, next move or as a minimum what strength their dealt hand had. This would be done by having the network look at many games played by that specific player. One of the big advantages of a neural network is that we are able to give a lot of inputs. These inputs will then be weighted by the system in order to come closer to our targeted output. This will help us cut out all of the noise that occur and leave us with data that is relevant.

Neural networks is models which are somewhat inspired by biological neural networks. Neural networks is considered to be one of the best methods to classify input-patterns. Neural network is for example used to recognize and reading handwriting and speech recognition. Humans are very good at recognizing visual patterns, but if we were to write a program that could do just that, it becomes alot more difficult. We have simple intuitions when we are trying to recognize different shapes. But to explain to a program how to recognize for example a Y would be something like a straight line with two lines pointing out from it at the top to each side at a 35 degree. When we try to make rules like this to let the program recognize letters we can quickly become lost in what we expect it to look like and the special cases.

Neural networks has a different approach to the program which is much more suitable than describing each letter in some mathematical algorithm formula. We give the neural network a very large amount of handwritten letters, which shall be used in the training of the neural network.

The neural network will then use the examples to automatically determine some rules for reading the handwritten letters. Of course this example doenst have a lot of different types of the letter A so if we want to let the neural network become better at reading the handwritten letter A we would have to give it an example with many more examples of the letter. It would improve the accuracy, therefore it is better to provide the neural network with a thorough example.

"...a computing system made up of a number of simple, highly interconnected processing elements, which process information by their dynamic state response to external inputs. A neural network consists of neurons which can send signals to each other. Each neuron processes its input signals and determines if the signal needs to be sent further. A neural network is not being fed with rules but instead examples that it can make rules from. With that a neural network is able to learn new skills, which is something that the traditionel computer cant do.

There are several types of neural networks and are being distinguished by their type (feedforward or feedback), their structure, and the learning algorithm that they use. Feedforward neural networks will only allow the neurons to connect between two different layers. The feedback type of neural network will have a connection between neurons which are of the same layer but also between the different layers.

2.4 Test

2.5 Discussion

2.6 Conclusion

3 Determine the strength of a poker hand

3.1 Introduction

There are different ways of evaluating the strength of a poker hand. One way is to create a formula (e.g. chen formula), another way could be to estimate the probability of the hand ending up being the winning hand. We chose to use probability since other major poker sites uses probability to determine the strength of a poker hand. To estimate the probability we use the Monte Carlo method as it is the most widely used method for calculating poker statistics.

3.2 Monte Carlo method

The Monte Carlo method can be used to calculate a probability distribution for domain. This is done by creating a large amount of simulations with random inputs within a range of allowed inputs. These simulations are also called Monte Carlo simulations. The more simulations that are performed the more accurate will the result be due to the large numbers law.

When designing our Monte Carlo method we had the following requirements:

1. It should be able to return the probability for any poker state with up to ten players.

2. It should have an error of one percent at max.

3. It should calculate the probability in less than five seconds.

To find out if the result is correct we compared our result to the results of other sources.

3.3 Determine the number of simulations

The only challenge we had when implementing the Monte Carlo method was to determine the number simulations. The number of simulations had a trade-off. The more simulations the calculations and time was needed but at the same time the more precise the result would get.

To find right amount of simulations we created a test where we changed the number of simulations to see how it would affect the time and probability.

It calculates the probability of winning with a pair of jacks against a single opponent. Other sources estimated the probability to 0,771. We plotted the results of running 50 test with 1000, 10.000 and 50.000 simulations.

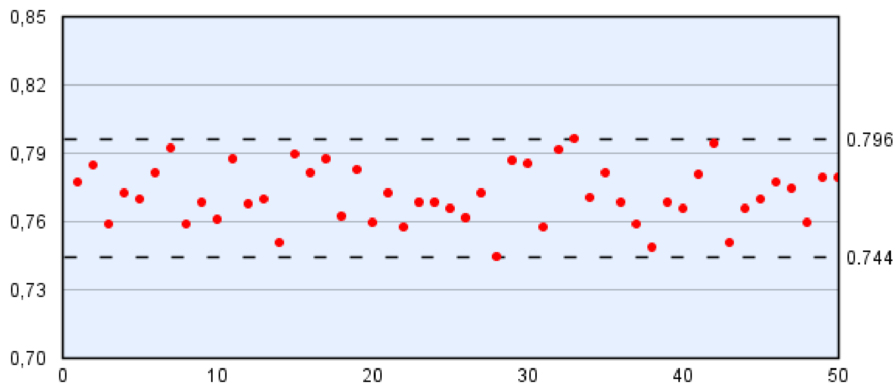

Figure 1: Result of Monte Carlo methods with 1000 simulations

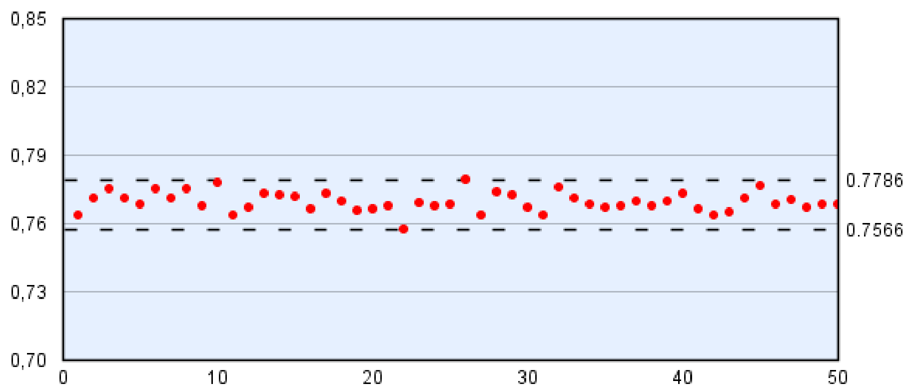

Figure 2: Result of Monte Carlo methods with 10.000 simulations

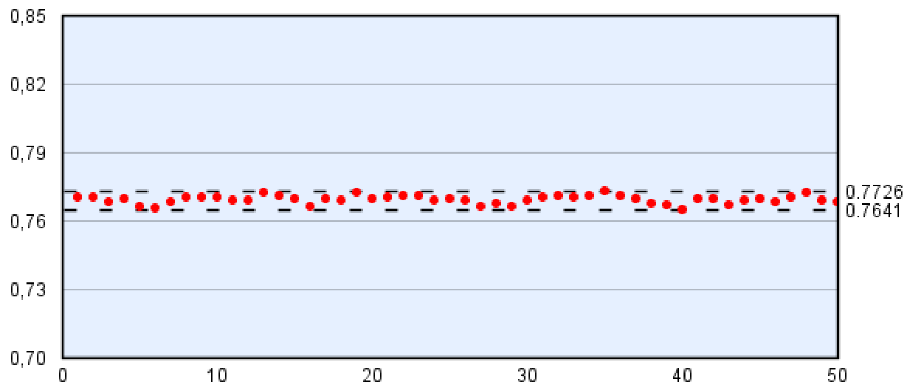

Figure 3: Result of Monte Carlo methods with 50.000 simulations

From the tests we found the maximum difference between the 50 results and the maximum difference from the expected result.

simulations	max difference (%)	max error (%)	time (seconds)
1000	5,2	2,7	0,03
10.000	2,2	1,5	0,22
50.000	0,9	0,7	0,82

We settled for 50.000 as the number of simulations for our Monte Carlo method. This satisfied our requirements.

3.4 Test

3.5 Discussion

3.6 Conclusion

4 Discussion

5 Conclusion

References

[1] Texas Holdem Heads-Up Preflop Odds *URL: https://caniwin.com/texasholdem/preflop/heads-up.php*

[2] Marlos C. Machado, Eduardo P. C. Fantini and Luiz Chaimowicz *Player Modeling: What is it? How to do it?*, SBC - Proceedings of SBGames 2011.

[3] Aaron Davidson, Darse Billings, Jonathan Schaeffer and Duane Szafron, *Improved Opponent Modeling in Poker*, Department of Computing Science, University of Alberta.

[4] Garrett Nicolai and Robert Hilderman, *Algorithms for evolving no-limit Texas Hold'em poker playing agents*, Department of Computer Science, University of Regina, Regina, and Dalhousie University, Halifax, Canada.

[5] Jochen Frhlich, *Neural Net Components in an Object Oriented Class Structure*, ebook pp. 11-28

[6] Hold'em Review, *http://www.holdemreview.com/texas-holdem-heads-up-rules/*

[7] Lily Hay Newman, *Using AI to Study Poker Is Really About Solving Some of the Worlds Biggest Problems*, Slat blog

[8]

[9] 10 most popular card games, *http://topyaps.com/top-10-popular-card-games*