[redacted]

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.

Introduction

Being a good poker player requires that one are able to adapt to opponents and decide when it will be beneficial to play aggressive, defensive or fold. In order to do that one must take a lot of different inputs into consideration such as cards dealt, chips, opponents strategy etc. a lot of these inputs will be hidden and not available to the player during most of the game.

In the games of poker a perfect player will never be anything more than a theoretical term. Such a player would have to be able to know all the information about the game including the hidden information so basically read the opponents mind and the future. We think of the perfect player as an unreachable goal instead.

To overcome the challenge about the hidden information players will instead make some decisions based on qualified guesses. They will try to figure out the opponents strategy based on prior actions and use statistics to calculate the odds of winning.

This thesis will focus on the design and implementation of artificial intelligence in poker. Our poker bot is targeted to Texas Hold'em Limit. We will make use of existing algorithms such as neural networks and Monte Carlo Simulation.

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.

Contents

Part I
Design

1 Design overview

2 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.

2.1 Why use player-modeling?

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.

2.2 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.

3 Determine the strength of a poker hand

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.

3.1 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.

3.2 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.

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

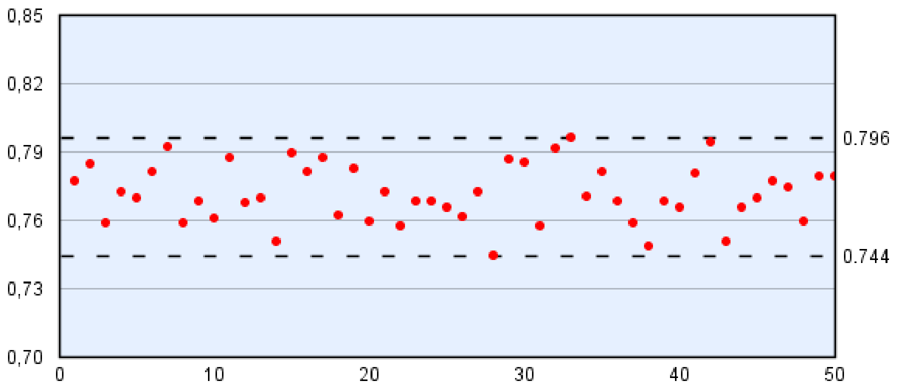

Figure 1: Result of Monte Carlo methods with 1000 simulations

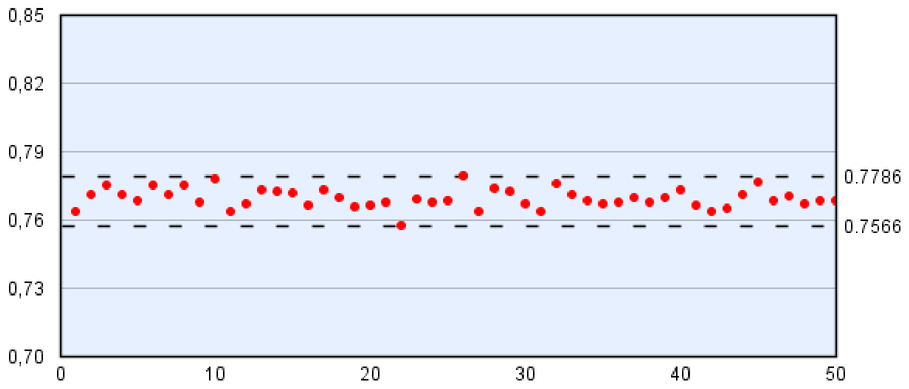

Figure 2: Result of Monte Carlo methods with 10.000 simulations

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.

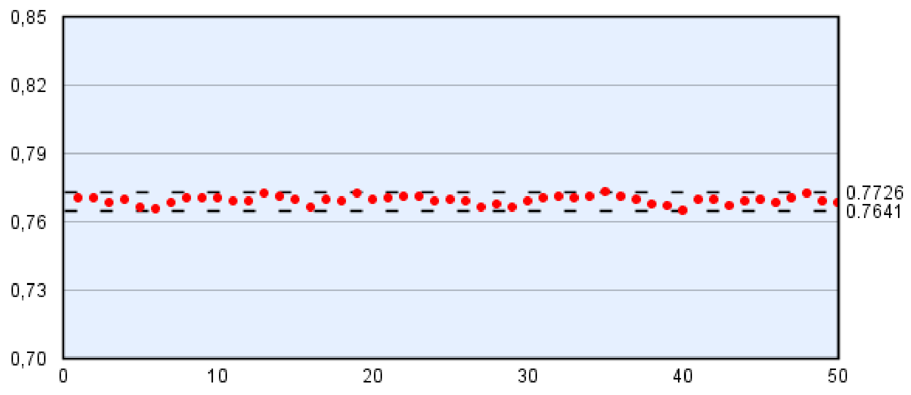

Figure 3: Result of Monte Carlo methods with 50.000 simulations

Part II
Implementation

4 Conclusion