[redacted]

ADAPTIVE POKER BOT BASED ON PLAYER-MODELLING

Bjørn Hasager Vinther & Nicolai Guldbæk 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 Hold'em 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.

Texas hold'em poker

Poker is arguably one of the most popular gambling games. It combines psychology, statistic, and luck into a single game with simple rules. Even though the rules of poker are simple, the game itself is hard to master. It is played with a standard deck of 52 cards. There are multiple variances of poker but this section only describes the basic rules of Texas hold'em limit poker. For a more in-depth description of the rules see [**?**].

Player actions

Fold The player gives up and the player is out until the start of the next round. This action is always possible.

Check The player does not bet any chips. This action is only possible if no other player has placed a bet.

Bet The player bets an amount equal to the blind. All opponents have to match his bid in order to stay in the game.

Call The player place a bet that matches the highest bid of the opponents.

Raise The player place bet higher than the bet of the opponents. All opponents have to match this bid in order to stay in the game. This action is only possible if another player has placed a bet.

All-in The player bets all his chips. The player is still in the game but he will not be able to act any more. This action is only possible if you cannot afford to call, bet or raise.

Keywords

Dealer	The players takes turn being the dealer in the beginning of each round. The player to the left of the dealer is always the first to take action in the bidding round.
Blind	A fixed amount that the two players has to pay in the start of each round.
Hole cards	A pair of private cards that is dealt to each player.
Community cards	The five public cards that are dealt to the table and are shared between all players.
Bidding round	In the bidding round the players takes turn performing an action. A bidding round will occur in every game state.
Action	An action can be performed when it is the players turn to act during the bidding round.
Pot	The sum of all the bids that have been placed in the round. The winner of the round wins the pot.
Chips	The amount of money a player has.
Round	A round consists of 4 game states: pre-flop, flop, turn, and river. In the beginning of each round the blind are paid.
Pre-flop	The first game state. In this state the hole cards are dealt.
Flop	The second game state. In this state the first three community cards are dealt.
Turn	The third game state. In this state the fourth community card is dealt.
River	The final game state. In this state the fifth community card is dealt. After the bidding round a showdown will take place.
Showdown	The phase where all players who have not folded show their whole cards and the winner is determined.
Hand	The best combination of the players hole cards and the community cards. See table 1.

Game play

A Texas hold‘em poker game consists of multiple rounds. A round consists of four game states: pre-flop, flop, turn and river. Each game state starts with cards being dealt and then a bidding round begins. In the bidding round the players will take turn performing their actions. If only one player is left after a bidding round that player wins the pot, otherwise the game continues to the next game state. If multiple players are still left after the river a showdown will start. During showdown each player left will reveal their hole cards and a winner will be found. The winner wins the pot. In case of a draw the pot is split between the winners.

The amount of chips is the main indicator of how well the player is doing. The main goal for the players is to increase their amount of chips.

The bidding round

The bidding round is where the players in turn perform their actions. The actions include: call, bet, raise, fold, check, all-in. The player to the left of the dealer is always the first to act.

A player can play aggressively by betting or raising which will increase the cost for the other players. Likewise a player can also play defensively by calling or checking which won't increase the cost for other players. If a player decides to fold he will lose what he betted that round. Whenever a player chooses to play aggressively all other players must either fold or call in order for the bidding round to stop. The bidding round continues until all players have called the aggressor or folded.

Rules for determining the winner

The rules for finding the best hand is quite simple. We refer to the rank of a card (2-A) as the card rank and the rank of the hand (see table 1) as the rank. First the rank of the hand is found. The order of the cards is insignificant. In case two players has the same rank the winner will be determined by the card ranks of the given hand. For instance a pair of kings is better than a pair of jacks. No card suit is better than another.

Starting from the top (the best hand) in table 1 we have the explanations: **Royal Flush** is having a straight all in the same suit. The highest card also have to be an ace.

Straight Flush is the same as a royal flush except there is no requirement for the highest card.
Four of a kind is having four cards with the same card rank
Full house is having three of a kind and a pair
Flush is having five cards with the same suit
Straight is having five cards in a row (e.g 10-A or 4-9)
Three of a kind is having three cards with the same card rank
Two pairs is having two pairs
One pairs is having two cards with the same card rank.
High card is the highest card.

rank	name	example hand
1	Royal flush	A♣ K♣ Q♣ J♣ 10♣
2	Straight flush	7♣ 6♣ 5♣ 4♣ 3♣
3	Four of a kind	K♣ K♠ K♢ K♡ 10♣
4	Full house	K♣ K♠ K♢ Q♡ Q♣
5	Flush	K♡ Q♡ 5♡ 3♡ 2♡
6	Straight	A♣ K♠ Q♢ J♡ 10♣
7	Three of a kind	A♣ A♢ A♠ J♣ 10♣
8	Two pairs	A♣ A♢ 5♠ 5♣ 4♣
9	One pair	A♣ A♡ J♣ 9♠ 2♡
10	High card	A♣ K♢ 6♡ 5♡ 3♡

Table 1: Ranks of different hands in Texas hold'em poker sorted best to worst.

Preface

This bachelor thesis consists of 15 ETCS points. The thesis has been written exclusively by Nicolai Guldbæk Holst and Bjørn 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 Støy.

Introduction

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

Reading a player refers to the process of figuring out the players strategy and is a major element of poker. The top players are able to read their opponents and adapt to 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 the players do not know the cards that will be dealt throughout the game, players must calculate the likelihood of their hand ending up being the winning hand. The more likely the player is to win the more aggressive he can play to maximize his profit.

In the field of artificial intelligence research games are interesting because of their well-defined game rules and success criteria. Computers have already mastered some of the popular games. One example is the chess bot Deep Blue which won against Garry Kasparov, the world champion of chess at that time. Chess is a game of perfect information, as no information is hidden from the players. Since then the interest of the artificial intelligence research has shifted towards games with imperfect information. These types of games presents new challenges such as deception and hidden information.

Developing an algorithm capable of playing poker is not only limited to the domain of poker but can end up having a future applications in other domains as well. In essence a game presents a challenge for the players to solve. How the player approaches the challenge and what strategy uses to solve it, are what determines the players success. Likewise in any other domain a person will be presented with a number of challenges which each requires the person to develop a strategy.

In may 2015 a contest was held with four of the best professional poker players in the world. Each of the players had to play 20.000 hands of heads-up no-limit Texas Hold'em against the currently best poker bot in the world Claudico. Claudico were able to end up with more chips than one of the four players which proves that artificial intelligence in regards to poker have come a long way.

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 solve the following problem statements:

Problem statements

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?

Our thesis is divided into three sections each of which focuses on one problem statement.

In section 1 we develop a subsystem which is able to estimate the probability of winning for any set of hole cards in any poker state. The subsystem can calculate the probability of winning with an error percentage of one percent and it takes less than a second on average.

In section 2 ...

In section **??** ...

Contents

1 Determine the strength of a poker hand

Our first step towards developing an adaptive poker bot is to find a way to determine the strength of any given hand in any game state. In this chapter we will answer the question:

Problem statement 1

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

Since a player do not know the outcome of the community cards during a round of poker we have to estimate odds of winning based on the possible outcomes of the community cards. It would be too time consuming to check every outcome as there are more than 250 millions different outcomes of community cards alone. Additionally, the poker rules are quite complex when it comes to determining the wining hand, therefore it is almost impossible to make a formula to calculate the exact probability. In order to find the probability without having to check the outcome of 250 million hands we need to find an estimate rather than an exact probability.

1.1 Design

When trying to find an estimate we have two options.

One option is to create a simplified formula to estimate the strength of the hand. The Chen formula is an example of this. This method is straight forward but the disadvantage is that it can only be used for preflop and would limit the implementation further in the project.

Another option is to use the Monte Carlo method to simulate a lot of games and get an estimate of the probability. The Monte Carlo method is also considered to be more precise than methods similar to the it.

For a human player the simplified formula would work best but in our case the Monte Carlo method is best suited. This is because, a computer has no problem performing calculating thousand of games, but a human would struggle to keep track of the different results from each simulation of a game. Not to mention, that a computer can calculate mathematical formulas many times faster than a human. This method also gives a trade-off between accuracy and computations which allows us to decide how accurate an estimate we need. The major poker sites also use this method to calculate the probability of ending up with the winning hand.

The Monte Carlo method has been implemented in as a subsystem in a playable poker game to perform the simulations and return the probability of winning given a set of hole cards. We will refer to this subsystem as the calculator. The calculator takes three arguments: the hole cards of the player, the community cards (optional) and the number of opponents. The calculator then performs the simulations and returns an object containing the distribution of outcomes. Caniwin [1] is a website that has calculated the probability of winning with each of the 169 different hole cards combination. We consider caniwins results to be the actual probability of winning with a given combination of hole cards. This sites results is limted to the preflop game state, but is still a good indicator to prove that the Monte Carlo method has been implemented correctly.

Requirements for the calculator:

- It shall be able to return the probability of winning for any poker game state with up to ten players.
- It must have a maximum error percentage of one percent. (deviation from caniwin)
- It shall calculate the probability in less than five seconds.

1.1.1 Monte Carlo method

The Monte Carlo method can be used to calculate a distribution of results for any given set of hole cards. The system developed were given a specific combination of hole cards to calculate an estimate, on what the probability of winning with that hand was. The Monte Carlo simulation then ran x number of games with the same hand, each time giving the opponents a random set of hole cards. The distribution of results we get can be used to find the likelihood of possible outcomes by looking at how many wins and loses compared to the number of games. The more simulations that are performed the more accurate the probabilities will get when comparing to caniwins results.

1.2 Test

To test if the calculator can calculates the correct probabilities we find the probability of a number of different pre-flop scenarios and compare the result with the results of caniwin. Every test have been preformed with 50.000 simulations against one opponent. The result can be seen in table 2. From the results we can see that for all hole cards the error percentage is less than one.

To test the accuracy of the calculator we try to find a number of simulations that fulfil the requirements. This is done by comparing the error percentage we get with a different number of simulations. Each test is performed with a pair of jacks in pre-flop with one opponent. Caniwin found the probability to be 77,1 %. Figure 1, 2 and 3 shows the distribution of results from 50 tests. Each test result is indicated with a red dot. In figure 1 we can see the results ranges from 74,4 % to 79,6 % (5,2 %). In figure 2 the range is only 75,7 % to 77,9 % (2,2 %) and finally in figure 3 the range is down to 76,4 % to 77,3 % (0,9 %).

In table 3 you can see the combined result. The range is the difference between the lowest and highest result and the max error is the maximum deviation from caniwin.

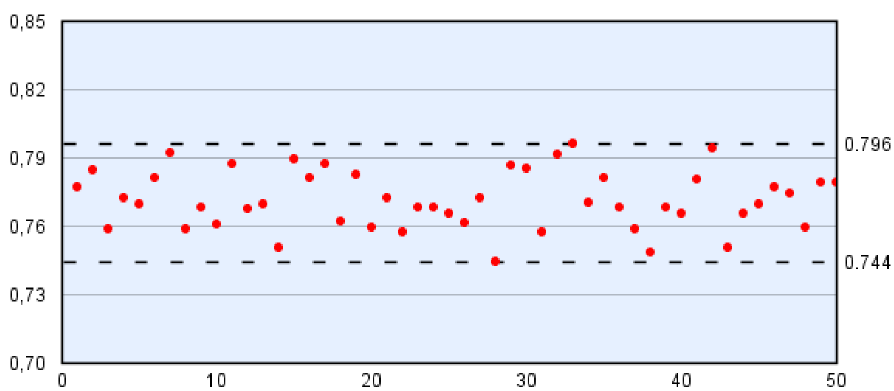

Figure 1: Result of the calculator with 1000 simulations

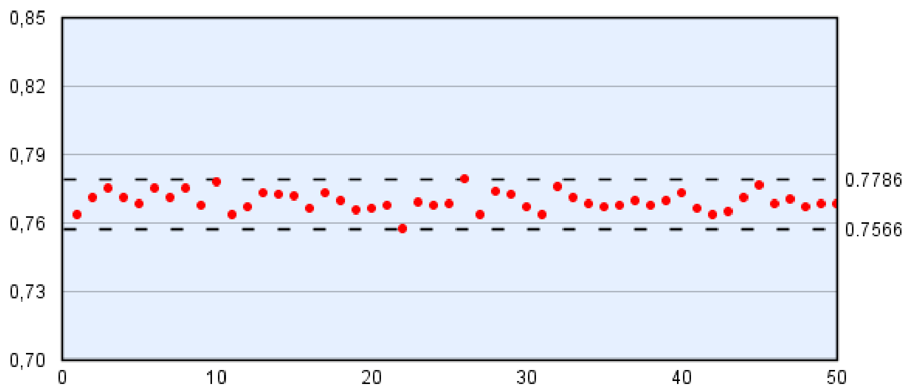

Figure 2: Result of the calculator with 10.000 simulations

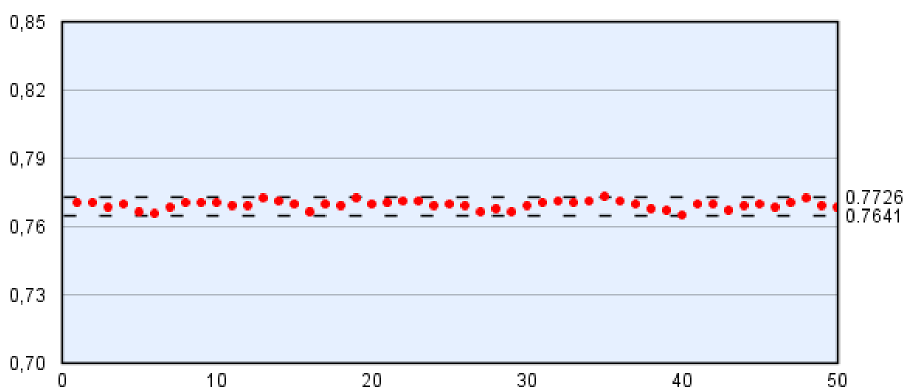

Figure 3: Result of the calculator with 50.000 simulations

The 3 graphs clearly states that when using a higher amount of simulations, the more accurate the probability will become. The error percentage in the table have been calculate by comparing our result to caniwin:

$$Ourresult - caniwin$$

We settled for 50.000 as the number of simulations for our calculator. This satisfies our requirements..

hole cards	our result (%)	caniwin (%)	error (%)
A♣ A♢	85,2	84,9	0,3
8♣ 8♢	67,9	68,7	0,8
Q♣ k♣	62,8	62,4	0,4
A♡ 8♠	58,8	60,5	0,7
J♠ Q♢	57,2	56,9	0,3
10♡ J♡	56,7	56,2	0,5
3♢ 3♠	53,0	52,8	0,2
2♢ 2♡	49,5	49,4	0,1
9♢ 3♠	37,8	37,4	0,4
2♢ 7♢	35,5	35,4	0,1
2♢ 7♡	31,9	31,7	0,2

Table 2: Test results for different hole cards in pre-flop with one opponent

simulations	range (%)	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

Table 3: Combined test results from running the calculator with different numbers of simulations.

1.3 Discussion

For the implementation of the calculator the Monte Carlo method was. This method works well for the needs. The calculator can calculate the probability with an error percentage of less than one percent when using 50.000 simulations. This solution can be used for any poker state with up to ten players. At first it did not seem necessary to optimise the calculator any further, as the running time is acceptable. But in regards to later use or if were to scale the project, we decided to make an obvious optimisation by making it multi-threaded which speeded up the calculations by.................

Alternatively we could have created our own formula to calculate a rank, or used an existing one, for instance the Chen formula. By using this method we will get a less accurate result but it will be easier to calculate. We would also have to create a formula for each game state which would cause even more work. Since performance is not a problem for our calculator we chose to use the Monte Carlo method.

1.4 Conclusion

In this section we have answered the question:

Problem statement 1

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

We have implemented a subsystem called the calculator that can estimate the probability of winning with a set of hole cards. The calculator uses the Monte Carlo method and it works for every poker state with up to ten players. We have found that 50.000 simulations is a good number of simulations for our calculator.

We compare our results to caniwin, which is a website that calculates the actual probabilities of all the 169 combinations of hole cards. The calculator has a maximal error percentage of one percent and performs the simulation in less than a second.

2 Learning default play style

In the previous chapter we created a calculator which can calculate the probability of winning with any hole cards in any game state. We will use the calculator in this section to estimate the strength of our hole cards.

Before we can learn our poker bot to adapt to opponents strategy we first need it to learn a default one. When the poker bot first joins a poker game it has no information about the opponent. In this case it must use a default strategy while it gathers more information. In this chapter we will find a solution to the problem statement:

> How can we make a default strategy without having information about the opponents?

Because the default strategy has to work against any type of opponent, we don't expect it to be able to win against every type of opponent. The focus of the default strategy is not to make the poker bot win, although that would be preferable, its goal is instead to reduce the loses while the system gathers information about the opponent. But how can we make sure that we create a bot that plays on a sufficient level that it wont loose unnecessary?

2.1 Design

To create a default strategy we have two options which satisfies our requirements.

One way is to use a professional players guideline on how to handle every situation that our bot could reach. This could be achieved by hard coding the decisions into the program and take every scenario into account, so the bot wont ever have to think for itself. If we were to hardcore every situation this would both be time consuming compared to implementing a self learning bot, but it would also make the strategy vulnerable as an intelligent opponent to some extend predict the future behaviour of the bot.

To implement a self learning bot, we would have to create a system that is able to read, understand, and learn from the poker data that were provided by University of Alberta. The University of Alberta has a research group that specializes in computer poker players. Real life poker players have been observed and the data regarding the players behaviours throughout the game have been collected and inserted into that dataset. This data will be used to develop the default strategy, by making the bot learn from the real life players behaviours. In our case this seems like the best way to go, as the default strategy will be hybrid as it learns many different ways of playing the game by various players and putting them together as one creating a more general player.

There are various ways to implement artificial intelligence in a game of poker but they are all imperfect as there is a lot of imperfect information in a game of poker. The way that was chosen to be used in this thesis is a neural network. A supervised neural network is when the neural network has a targeted output. This means that when the neural network is trying to learn from the dataset we know what the system should output. Given the data from our dataset has the action of the players, a supervised learning neural network can be created, by targeting the output to the given action the player performed. To create a neural network a framework called Neuroph. Neuroph was used as it was well documented.

A perceptron is a simple neural network which has inputs and outputs, but no hidden layers in which calculations could be made. Each neuron has a weight that is an indicator on how much a given input means for the output of the network. The sum of each neurons weight is passed into a transfer function, that is a mathematical representation for constructing the curve that fits the data points best. To see if a a simple neural network could solve our problem statement, a simple perceptron was created. The perceptron takes two inputs, and gives two outputs. The inputs are given by looking at each hand in the dataset that is visible and following that given players behaviour. The first input being the probability, that the current player who the perceptron is looking at in the dataset, has the winning hand. The second input being how many opponents the current player are playing against. The two outputs tell us if the strategy of the current player is aggressive or defensive. The perceptron uses a sigmoid transfer function to give us us the desired output 1 or 0 as to whether the strategy of the current player is aggressive or defensive.

This approach did not seem to suit our needs as after training the neural network it did not make reasonable decisions. The neural network somewhat found a context between the number of opponents versus as to if we should be aggressive or not. The probability of the player having the winning hand

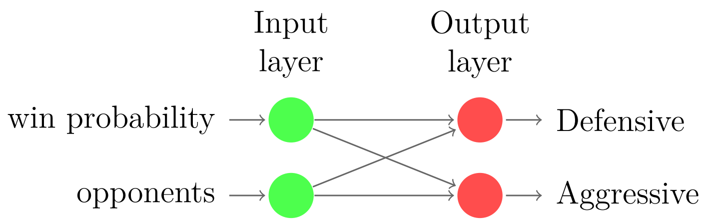

Figure 4: Neural network 1

did not weight as much as the number of opponents did. Therefore we chose to implement a more complex neural network as the results from the perceptron perhaps were not the most preferable decisions in the given situations.

A multi-layer-perceptron(will be refereed to as MLP) is closely related to our first described perceptron. The most simple structure is almost identical, but instead only having 2 layers, the input and output layers. The MLP can have multiple hidden layers inbetween the two layers of inputs and outputs. The inputs in the MLP is taking the same as the earlier discussed perceptron, but also the chips the current player has, how much it will cost the player to call the current bid, and the total pot(a.k.a the profit) for that game state. The MLP has 1 hidden layer with 2 hidden neurons. One hidden neuron for the probability of having the winning hand and the number of opponents. The second hidden neuron is for the chips, the cost and the pot. As with the perceptron the sum of each weight in the MLP will be passed into a transfer function. In this case we chose to use a STEP function.

2.2 Test

To test our first simple neural network a.k.a the perceptron, we ran through x games in the dataset. The neural network is not able to get particular smarter when running through an additional set of entries as the data is from different players and x games should be sufficient for the neural network to gain some kind of knowledge on how to play as the goal of this strategy is not to have a bot that can beat the opponents but minimize the loses. The neural network now had to learn from the dataset

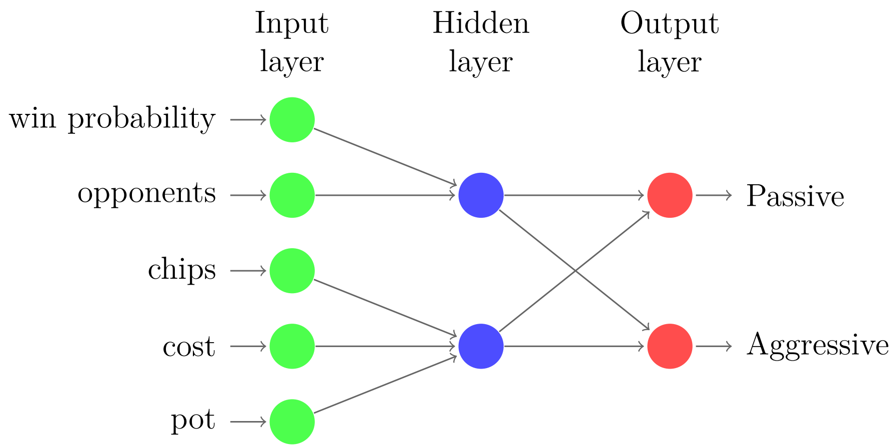

Figure 5: Neural network 2

2.3 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

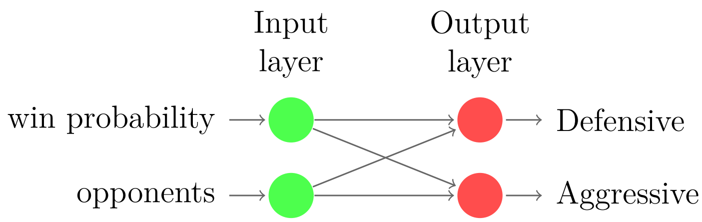

Figure 6: Neural network 1

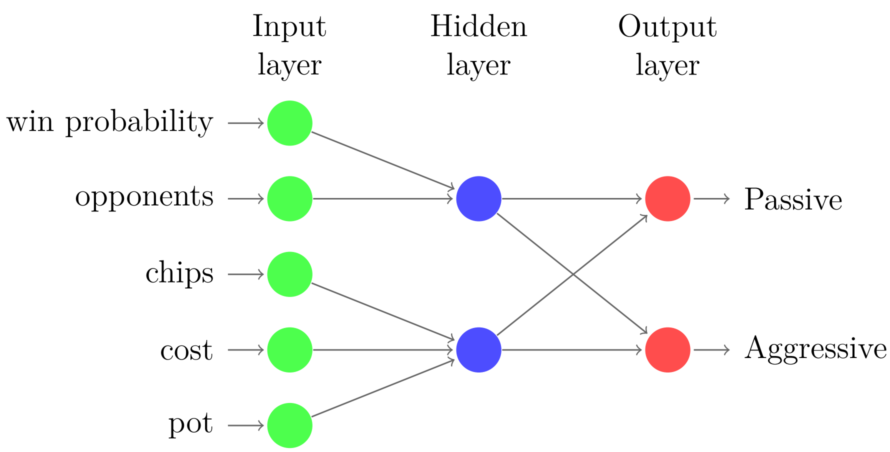

Figure 7: Neural network 2

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.

2.4 Conclusion

3 Adapt to opponents

3.1 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?

3.2 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.

3.3 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.

3.3.1 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.

3.4 Test

3.5 Discussion

3.6 Conclusion

4 Discussion

5 Conclusion

Glossary

Poker

Texas Hold'em
The most popular variance of poker. Game rules [7]

Limit / No-limit
Restrictions about how much players are allowed to bet or raise. No-limit has no restrictions while in limit you are only allowed to bet and raise a fixed amount.

Poker game
A game is from the players start playing to all except one player have been knocked out or left the game.

Poker round
A round is from the poker get their hole cards dealt to the winners are found. In each round the players will have new hole cards.

Game state
A poker round is divided into four game states: Pre-flop, flop, turn, river. The difference between the states are the number of community cards dealt. The numbers of community cards dealt are zero, three, four, five for each state respectively.

Pre-flop
The first game state. Each player is dealt two cards and zero community cards are dealt. After the bidding round the flop will be dealt.

Flop

The second game state. In this state three community cards are dealt. After the bidding round the turn will be dealt.

Turn

The third game state. In this state on community card is dealt making it a total of four community cards. After the bidding round the river will be dealt.

River

The fourth and final game state. In this state on community card is dealt making it a total of five community cards. After the bidding round the winner of the round will be found.

Preflop: Preflop is the stage before the flop. Each player has been giving 2 cards each.

Flop: Flop is the stage when the flop cards are on the table (3 cards)

Turn: Turn is the stage when the turn card is on the table (1 card extra so 4 cards in total)

River: River is the stage when the river card is on the table (1 card extra so 5 cards in total) Game: A full game is when one of the players has lost and by lost we mean hitting 0 in the bankroll. Round: A round is when we have been through all stages, preflop, flop, turn, river and the end of the round when one of the players has won the round by having the best cards.

Hand: Hand is the 2 two cards a player is holding, therefore reach player has a hand. Pot: Pot is the sum of the total amount each player has betted in the given round. Deck: Deck is the deck of cards Bluff: Bluff is when a player is trying to make the opponent think that he has good cards on his hand by raising or calling when actually he has nothing or very bad cards. Aggressive: Aggressive is when a player or computer is playing aggressively Defensive: Defensive is when a player or computer is playing aggressively All-in: All-in is when a player is betting all of his bankroll Bankroll: Bankroll is the total amount that a player has to bet with, when this hits 0 the player has lost the game. Board: Board is the table that we are sitting and playing at. Small blind: Small blind is an amount that the player who has the small blind HAS to pay even though they want to fold their hand. Small blind is half the amount of big blind. Big blind: Big blind is an amount the player who has the small blind HAS to pay even though they want to fold their hand. Big blind is double the amount of small blind. Check: Check is when a player doesn't want to bet but just wants the game to go on. Call: Call is when the opponent has betted an amount and if we want to proceed the round we will have to call his bet and lay the same amount into the pot as he did. Raise: Raise is when the opponent has betted an amount and if we want to proceed with the round, we can either call his bet or raise the amount by laying more than what our opponent laid into the pot. Thereby our opponent will have to either call, fold or raise. Fold: Fold is if you don't want to play with the cards you have been dealt or the opponents bet is too high for you, given the cards you have on your hand. Limit: Limit is so that one playing cannot just bet 100 kr. Or whatever we set the limit to. But perhaps the limit is 10kr for each bet/raise. No-limit: No-limit means that a player can bet whatever he wants or raise with whatever amount he has left in his bankroll. Heads up: In a heads up we are only having 2 players who play against each other.

References

[1] caniwin. (no date known) Texas Holdem Heads-Up Preflop Odds [Online]. Available: https://caniwin.com/texasholdem/preflop/heads-up.php

[2] inferisx. (no date known) Top 10 popular card games [Online]. Available: http://topyaps.com/top-10-popular-card-games

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

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

[5] 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.

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

[7] Poker Listings, *http://www.pokerlistings.com/poker-rules-texas-holdem*

[8] Lily Hay Newman, *Using AI to Study Poker Is Really About Solving Some of the World's Biggest Problems*, Slat blog