[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

Nothing yet...

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 computer Deep Blue which won against Garry Kasparov, the world champion of chess at the 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. Poker is an example of a game with imperfect 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 calculate the probability of ending up with the winning hand?
2. How can we develop a default strategy without having information about the opponents?
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 find a solution to problem statement one. 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 3 ...

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. In order to find the probability without having to check all possible outcomes, we need to find an estimate rather than an true 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. This method is straight forward but the disadvantage is that it can only be used for pre-flop 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 the distribution of outcomes. This distribution can then be used to find the probability for any of the outcomes.

For a human player a simplified formula would work best but in our case the Monte Carlo method is best suited. This is because, a computer can perform thousands of simulations in no time. This method also gives a trade-off between accuracy and the number of simulations. This allows us to decide how accurate an estimate we need by regulating the number of simulations. The major poker sites also use the Monte Carlo method to determine each players probability of winning.

The solution is implemented as a subsystem that uses the Monte Carlo method. 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 true probability by simulating every possible outcome. We will compare the results from the calculator with the ones from caniwin. Caniwin's results are limited to the pre-flop game state, so we can only test the calculator for this game.

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 by 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. The true probability is 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 5 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.

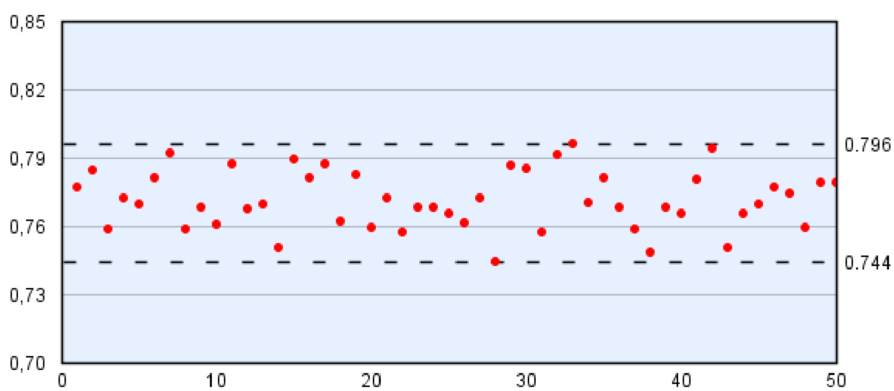

Figure 1: Result of the calculator with 1000 simulations

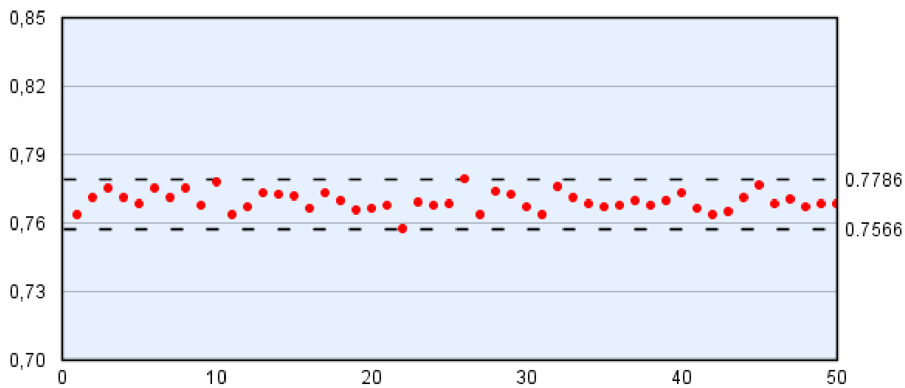

Figure 2: Result of the calculator with 10.000 simulations

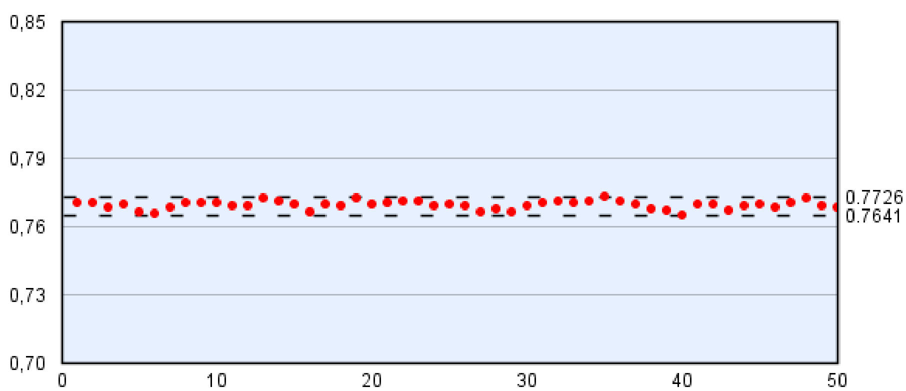

Figure 3: Result of the calculator with 50.000 simulations

The three graphs clearly shows that when using a higher amount of simulations, the more accurate the probability becomes. The error percentage in table 2 is found using the formula: $P_O - P_T$. Here P_O is the probability found by the calculator and P_T is the true probability. We settled for 50.000 as the number of simulations for our calculator. This satisfies our requirements..

hole cards	P_O (%)	P_T (%)	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

The pair of fives in this table have a zero percent chance of winning at the river, because the monte carlo does not count a draw as a win.

hole cards	community cards	pre-flop (%)	flop (%)	turn (%)	river (%)
A♣ K♢	3♠ 7♣ T♣ 8♢ 2♠	64,5	54,4	41,7	35,6
2♢ 5♠	3♠ 6♢ K♢ 4♡ 7♠	29,7	28,2	92,8	94,2
8♠ 9♠	2♢ K♠ 3♠ A♣ T♠	49,1	53,5	38,5	99,7
5♠ 5♣	A♢ K♡ Q♠ J♢ T♡	58,5	48,0	34,4	0,0

Table 3: Test results for different hole cards in pre-flop, flop, turn, and river with one opponent

Despite a pair of aces is a good hand, the number of opponents makes the chances of winning a lot less

opponents	1	2	3	4	5	6	7	8	9
win probability (%)	85,2	73,4	64,0	56,1	49,5	43,9	39,1	35,0	31,6

Table 4: Chance of winning with the hole cards A♠ A♣in pre-flop

The time have been calculated by finding an average of 100 calculations of the same hole cards.

simulations	range (%)	max error (%)	time (seconds)
1000	5,2	2,7	~ 0,01
10.000	2,2	1,5	~ 0.04
50.000	0,9	0,7	~ 0.15

Table 5: 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 roughly 3 times.

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 a default strategy

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 and 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:

> **Problem statement 2**
>
> How can we develop a default strategy without having information about the opponents?

Even though players have different strategies they often have some decisions in common. Most players tend to play more aggressively the better their chances are of winning and likewise most players will fold if they have a weak hand. These tendencies can be used in a default strategy. The more popular decisions are more likely to be good. For instance most players agree that it is unwise to fold a pair of aces in pre-flop.

The default strategy has to work against every strategy so it is impossible for it to be better than all of them. The goal is not for the default strategy to win, although that would be preferable, but instead to reduce the loses while it gathers information about the opponent.

2.1 Design

To develop a default strategy we have two options.

The first option is to directly program how the computer shall act in all situations. In this case the programmers need a deep insight in how to play poker in order to decide which decisions are optimal in the different situations. One can also use the expertise of professional poker players in case one lack the insight.

The second option is to develop a self-learning algorithm for the computer. Such an algorithm uses the concept *watch and learn* by observing other players playing poker and trying to learn the strategy behind the decisions. This method requires that the computer has something to observe.

Since we do not have any particular insight in how to play poker and do not have expertise from any pro poker players, we will implement a self-learning algorithm. Another advantage by using this method is that the computer is not limited by our understanding of the game.

The University of Alberta has a research group that specializes in the field of artificial intelligence in poker. They have released a dataset containing data from 18.000 real-life rounds of Texas hold'em limit poker. The dataset only contains the hole cards of the players who make it to the showdown, so there is no data about the hole cards of the players who fold. This data will be used to develop the default strategy.

To implement the self-learning algorithm we use an artificial neural networks (ANN), see section 2.1.1. The algorithm needs to recognise patterns in the players decisions and ANNs are very good at this.

We use an iterative development method to design the ANN for the algorithm. We start by designing a simple ANN and then move on to more complex ANNs.

2.1.1 Artificial neural network (ANN)

An artificial neural network (ANN) is inspired by the human brain. It is well suited for finding an approximation to a non-linear function. An ANN can take any number of inputs and return any number of outputs.

An ANN is made up of neurons that are connected into a network. Each neuron takes a set of inputs and give one output. The output of a neuron is sent to the connected neurons. Each input has a weight that determines influence of the input. The neuron uses an input function to calculate the net input, usually the sum of all weighted inputs, and pass it on to the transfer function. The type of transfer function determines the output. A step function returns zero or one if the net input is above a certain threshold. This is useful for logical functions. If one needs a value between zero and one a sigmoid function can be used instead.

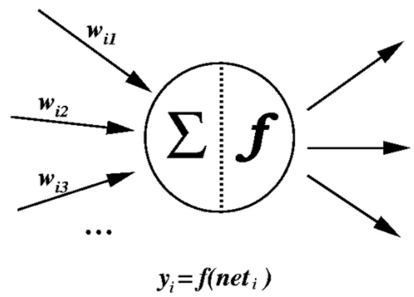

Figure 4: Model of a neuron.

The neurons in an ANN are distributed in layers as shown in figure 6. The coloured circles represents neurons and the arrows represents the connections between the neurons. There are three types of layers, an input layer, a hidden layer, and an output layer. An ANNs consists of one input layer and one output layer but may contain any number of hidden layers. The simplest type of ANN is the perceptron which has no hidden layers. It is used for single calculations. A multilayer perceptron is another type of ANN which contains hidden layers. It is used for more complex domains with multiple layers of computations.

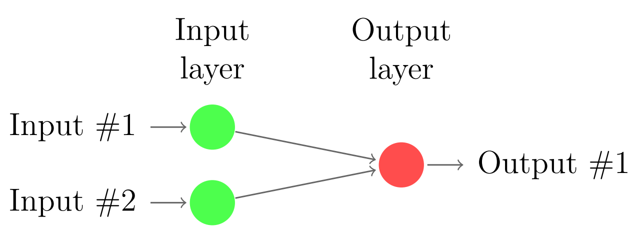

Figure 5: Perceptron with two inputs and one output.

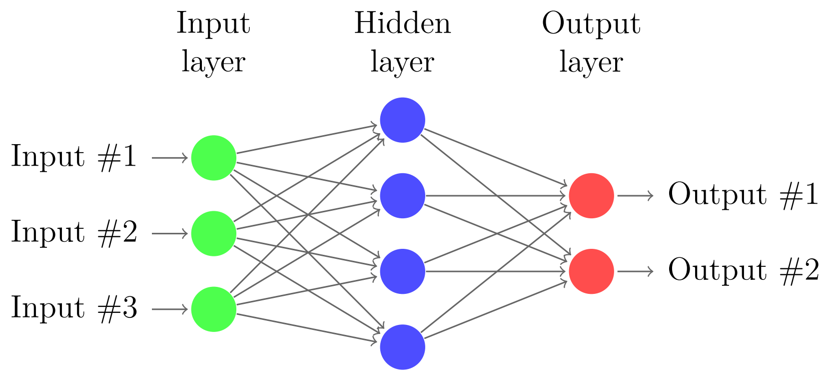

Figure 6: Multilayer Perceptron with three inputs and two outputs.

The ANN can be trained using a training set of inputs. Using supervised learning, in contrast to unsupervised learning, one must also supply an expected output. For each input it will adjust the weights in order to get closer to the expected output. It will continue doing this until the outputs are close enough to the expected outputs. Different algorithms exist for this.

Backpropagation is the most common algorithm for supervised ANNs. It adjust the weights from the end (the output neurons) back to the start (the input neurons).

After the training the ANN can be validated to see if it works. This is done using a test set different from the training set and see if the results of the ANN matches the expected results of the test set.

2.1.2 First ANN design

For the first attempt we design a simple perceptron. The perceptron takes two inputs, and gives two outputs.

The first input is the hand strength. We use the calculator from section 1 to calculate the probability of winning against a single opponent. The reason we always find the probability against a single opponent is because the probability of winning decreases drastically as the number of opponents increases. We use the absolute hand strength rather than a hand strength relative to the number of players. This makes it easier to compare the hands strengths in situations with different numbers of players. The number of players is instead the second input.

The first output is whether or not to be defensive by checking or calling. The second output is whether or not to be aggressive by betting or calling.

The outputs will both be zero in case it is best to fold. The perceptron uses a step function as a transfer function in both output neurons.

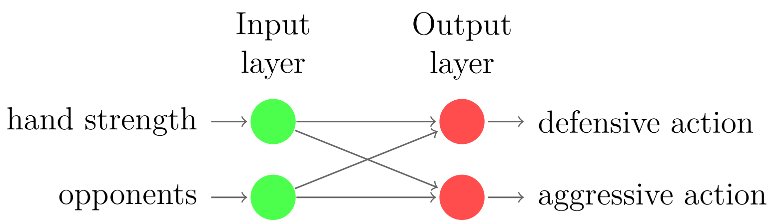

Figure 7: First ANN design.

The perceptron is more likely to be aggressive the higher the number of opponents is and the stronger the hand is. The problem is that the perceptron almost never acts aggressively. From our test set we expect the perceptron to be more aggressive.

2.1.3 Second ANN design

Since the first attempt designing a perceptron did not work properly we try to design a multilayer perceptron (MLP) with more inputs.

The MLP is taking the same inputs as the earlier discussed perceptron, but it now takes three additional inputs: The chips of the player, the cost for the player to call, and the pot.

The MLP has one hidden layer with two hidden neurons. One hidden neuron to calculate the likelihood of winning and another for the economically aspect.

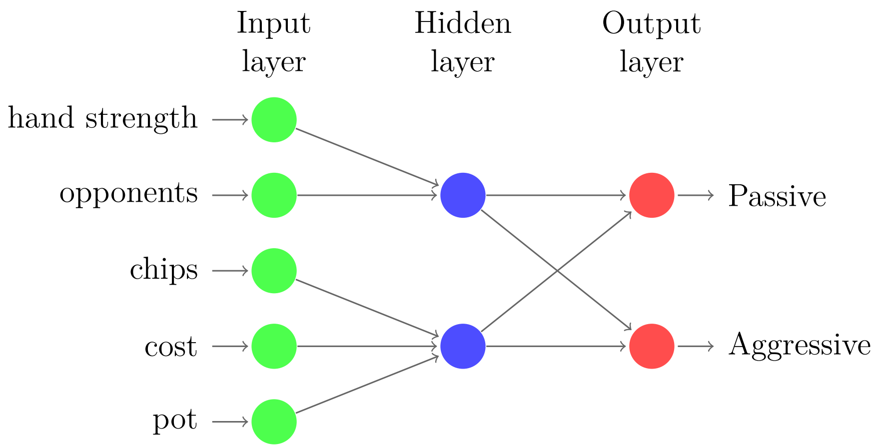

Figure 8: Second ANN design.

Vi har ikke nået mere så du behøver ikke læse videre :)

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

2.3 Discussion

When 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.

2.4 Conclusion

3 Learning to adapt to opponents

In the previous section we created an algorithm for learning the default strategy for the computer. The default strategy is one out the two strategies that will be used to determine the actions of the computer. The other strategy is the adaptive strategy.

The goal of this section is to learn the computer to create an algorithm to develop the adaptive strategy by solving problem statement 3.

Problem statement 3

How can the computer adapt to the opponents strategies?

In poker there is no such thing as a single optimal strategy. Every strategy has weaknesses and therefore the optimal strategy is one that take advantage of the weaknesses of the strategies of the opponents. In order to take advantage of the the opponents strategies one must first understand their strategy. In poker understanding the opponents is one of the most important key elements of the game.

3.0.1 What is player-modeling?

Player modeling is a loosely defined concept and may vary from one context to another. The concept of player modeling is to make a computational model of a player. This model includes game related attributes, such as play style and preferences, as well as non-game related attributes, such as cultural background, gender, and personality. All decisions of the player are ultimately made on the basis of these attributes.

Player modeling is used to describe or predict the players decisions, reasoning and reactions. In the field of artificial intelligences the human player is the most used model for developing computer players. Understanding the reason behind every choice of a player will not only bring a better understanding of the player but also a better understanding of the game and its mechanics.

Since the player model can easily become extremely extensive one normally only includes the relevant attributes of the player.

3.1 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.

3.2 Test

3.3 Discussion

3.4 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