# 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.

# Introduction

The game of poker is already a pretty complex game, put that together with artificial intelligence and you have a situation that is very challenging. To create an AI player that is producing good results against other players bring many problematics along the way. As far as implementing some artificial intelligence we are facing problems regarding the hidden information which is our opponents cards. Furthermore deception, and how many players we are playing against will also be obstacles that we will have to overcome.  
 If we want to produce the best possible results we will have to player model each of the players that we are playing against, to make sure that the bots gameplay will be the most optimal, considering all the opponents and their gameplay.  
One of the reasons why it is nesscary to player model every player is because we need to have the opputionity to exploit a weak player. If one keeps making weak moves that we can predict it would be a wise consideration to exploit this and beat the player.   
Even if we are playing a game of poker there will always good ways of handling each situation and thereby each opponent in the best way possible. The best players in the world are very efficient at adapting to each and every player of the board, even if the opponents changes their gameplay throughout the game, one must be able to adapt to this to be able to beat them.

# Texas Hold’em

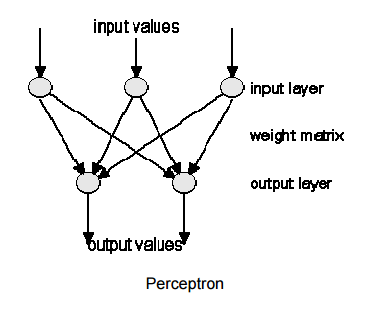
As there are different kinds of poker, there is one which is the most common, which is Texas Hold’em. Therefore this will be the kind of poker this bachelor thesis will be evolving around. This type of poker is also considered to be the type that is the most strategically complex type of the game.  
The rules of the game however are fairly simple.  
Each player is dealt 2 *hole cards*, which only the one receive them must be able to see.   
After the players have been dealt their cards, the two players who are sitting next to the dealer must be then ones to lay the blinds.  
The players after that can then either *fold*, *call* the bet or *raise* with either big blind or if we are playing no-limit then he can go all the way up to *all-in*.   
When the first round of betting are finished the dealer will deal 3 *community cards* which is also known as the *flop*. These cards are dealt so every player can see what their value is, and then we have another round of betting. When that is done, the dealer will *burn* a card, which means that that card will not be in play, and then he will draw another and put it face up like the rest of the community cards. This card is known as the *turn*. The next betting round begins and then the dealer will burn another card and then deal the *river* card with face up.  
Another round of betting, and then finally the players try to make the best hand they can with 5 cards with a combination of their hole cards and the community cards. The player with the best hand will be the winner.

# Types of neural networks

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.

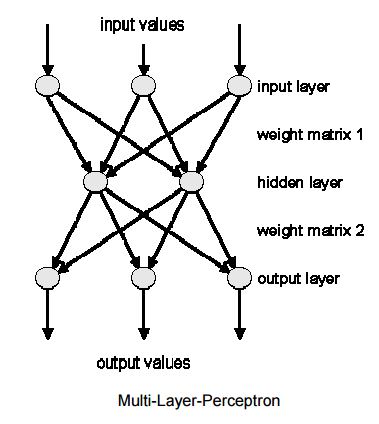
## Perceptron

This neural network is considered to be one of the simplest as it has two neuron layers which only accepts binary values for the input and output. With only two layers (input/output) there are no hidden layers.  
The learning process of this neural network is supervised[[1]](#footnote-1). The network can be used to solve basic logical operations.



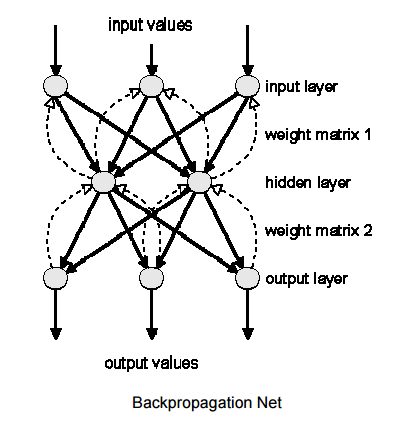
## Multi-layer-perceptron

This neural network is an extended version of the first discussed perceptron. The difference is that this network has one or more of the hidden layers which are in between the input and output layers.  
This network is also supervised but has a different learning algorithm as you can either choose delta learning rule or the backpropagation which is the one that is most common.



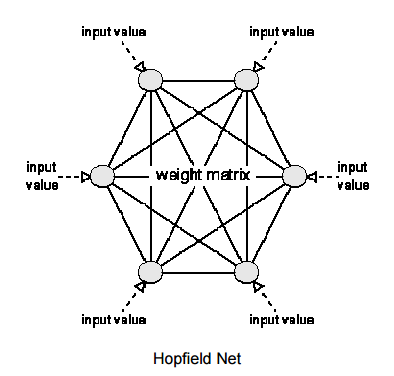
## Backpropagation

The backpropagation is considered to be one of the most powerful neural network types. It is very similar to the multi-layer-perceptron as it has the same structure with the use of the backpropagation learning algorithm.



## Hopfield

This type of neural network is called “thermodynamical models”.  
Unlike the other network discussed earlier we now have that each neuron is connected to eachother and we no longer have any differentiation between our input and output neurons.  
A Hopfield network would be used in situations where we would need to recognize patterns.  
The learning method of this network is unsupervised[[2]](#footnote-2).



## Kohonen Feature Map

When you want the learning process that is the closest to the human brain as possible then this neural network would be the right way to go.   
This is a neuron layer where every neuron in the network is organizing themselves in the right way according to the input values that it is given.  
(skal der lige læses mere om)

# Learning process

## Forwardpropagation

This learning process is one of the supervised learning algorithms. This means that we have an expected output and the system should learn how to get to that value by adjusting the weights accordingly.  
Forwardpropagation will randomly set all the weights in the range of -1.0 to +1.0 and adjust repeatly until the networks error is zero approximately 0.

Example:

Lets say we want to make our network learn this pattern:  
0 1 -> 0  
1 1 -> 1  
First the values of the weights are randomly set to 0.20 and 1.0.  
Our learning rate will be set to 0.30.  
Now we input the values of the first pattern which is 0 1.  
We get the following:  
Input 1: 0 \* 0.20 = 0  
Input 2: 1 \* 1.0 = 1  
Add the inputs: 0 + 1.0 = 1  
Compute our error value: 0 – 1.0 = -1.0  
Value to change weight 1: 0.30 \* 0 \* (-1.0) = 0  
Value to change weight 2: 0.30 \* 1 \* (-1.0) = -0.3  
Change weight 1: 0.20 + 0 = 0.20  
Change weight 2: 1.0 + (-0.3) = 0.7  
  
So now after changing the weights (only weight 2 was changed) we will input the next values 1 1  
Input 1: 1 \* 0.20 = 0.2  
Input 2: 1 \* 0.7 = 0.7  
Add the inputs: 0.2 + 0.7 = 0.9  
Compute our error value: 1 – 0.9 = 0.1  
Value to change weight 1: 0.30 \* 1 \* 0.1 = 0.03  
Value to change weight 2: 0.30 \* 1 \* 0.1 = 0.03  
Change weight 1: 0.20 + 0.03 = 0.23  
Change weight 2: 0.7 + 0.03 = 0.73

This shows one learning step. Each of our inputs have been inserted into the net and have changed the weights in the network.  
We are now able to calculate the error of our network, which is done by taking the squared values of our output errors of each of our input patterns:  
  
Computation of the network error: (-1.0)2 + (0.1)2 = 1.01  
As this process is repeated the network error will get lower and lower and is done when reaching zero.

## Backpropagation

Like the learning process before this is also a supervised learning algorithm and is often used to change the weights of the networks hidden layers in a multi-layer-perceptron.  
A computed output error is used in this algorithm in order to correct the weights in a backwards operation.  
Before we can get the networks error value, we will first have to do a forwardpropagation (the one explained earlier). The neurons in the network are being activated while we are in a forwardpropagation and using the sigmoid activation function.  
  
Sigmoid activation:



The steps of this algorithm is as so:  
First we run the forwardpropagation for some input pattern and hereby calculate what the output error is.  
Then we change all of the weights accordingly by using the formula:   
The old weight + LR \* Output error \* Output(neurons i) \* output(neurons i+1) \* (1 – output(neurons i+1))  
This keeps on being repeated until all of our outputs are matching our expected values.

## Selforganization

Selforganization is unlike the others discussed earlier a unsupervised learning algorithm which means that we have no expected value. This learning algorithm is used by the neural network Kohnen Feature Map.

1. Supervised learning is when we have an exptected output [↑](#footnote-ref-1)
2. Unsupervised – when we dont have an expected output [↑](#footnote-ref-2)