Project Proposal

Pepijn van Teeffelen, Julius Mannes & Casper van Aarle

Problem:

We want to make a bot that can play the game No-Limit Texas Holdem Poker and is created using evolutionary computing. While we do not expect that the bot will be performing better than a somewhat experienced human poker player, we do hope that the bot will outperform other bots, which would be following rather simple heuristic strategies.

Methods/Approach:

In Poker, a player has to make a decision at several stages at the game. Simplified, these decisions are (a) folding the hand, (b) paying the price to continue or (c) raise the stakes. Our main approach will make use of neural networks that will output this decision. A similar approach has been taken by [1].

Thus, a bot will have such networks as components of its system. Our final bot will be the result of an evolution of such bots. While all weights (and potentially the architecture) of the networks will be randomly initialized, they will be tuned during the evolution process. For the crossover part we envision offspring whose networks have weights that correspond to the weighted sum of the best individuals of previous generations. For the mutation part we will randomly change one or more weights in a bot's network by multiplying it with a (random) scalar. For the fitness function we envision two possibilities, which we will potentially test both. First, the quantity of poker-chips the bot has after a certain number of rounds of poker. Two disadvantages to this approach are that one cannot evaluate the bots after each round played and also the inherent stochasticity of poker. A bot could have made the best possible moves but ended up losing regardless. To resolve these issues, we propose the second fitness function: A measure of how much the taken decision is *worth*, related to the notion of Expected Value. An example: The bot holds a flush draw and has a 33% chance of winning a pot worth of 40 chips. To continue playing the bot has to pay 10 chips, which would make the pot worth 50 chips. He thus has to pay 20%, while he would win in 33% of the cases. In this case the bot would be deemed fit, if at least he continues to play. A bot that would continue playing with a flush draw and a pot of 10 chips would be unfit, due to playing unfavorable odds.

Initially we will focus on simple neural networks, but we will possibly investigate how an RNN approach, similar to what is done in [3], would perform.

Instead of using neural networks, a formula-based approach is also possible, an approach taken by [2]. If time allows it, we will also investigate performance of that approach.

We will make use of Python and PyPokerEngine library which provides a poker playing environment and allows us to fully focus on the development of the bot itself.

During the research process, we will experiment with and compare different approaches.

References:

[1] Nicolai, G., & Hilderman, R.J. (2010). Algorithms for Evolving No-Limit Texas Hold'em Poker Playing Agents. *IJCCI*.

[2] L. Barone and L. While, "An adaptive learning model for simplified poker using evolutionary algorithms," *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406)*, Washington, DC, USA, 1999, pp. 153-160 Vol. 1.

[3] P. J. Angeline, G. M. Saunders and J. B. Pollack, "An evolutionary algorithm that constructs recurrent neural networks," in *IEEE Transactions on Neural Networks*, vol. 5, no. 1, pp. 54-65, Jan. 1994.
doi: 10.1109/72.265960