An Adaptive Agent for No-Limit Texas Hold 'em Poker

David O'Steen
daost@stanford.edu
CS229 Project

ABSTRACT

Creating a computer agent to effectively play the game of no-limit Texas hold 'em is a formidable challenge for machine learning researchers, and to do date, existing techniques have failed to create an agent that is competitive with skilled human opponents. An effective agent must be able to assess the game state based upon incomplete information, model an opponent's playing style, and make economic decisions based upon these factors and random future events. Some researchers have had success using evolutionary neural networks in this setting, and I extend this approach by implementing more specialized opponent models. Rather than training one neural net to play against any random opponent type, I train separate models that are each specialized to play against a number of basic, hard-coded playing strategies. I find that the combination of these models into a final type performs against random opponent types as well as the average of the static types.

I. INTRODUCTION

In the game of no-limit Texas hold'em (NLH), a player's hand consists of the best five cards between his two concealed cards (hole cards), and five cards shared among all players (the board). There are four betting rounds: an initial round when each player has seen their hole cards, another after three shared cards have been revealed (the flop), the next after one more shared card (the turn), and the final round after the last shared card is revealed (the river). Unlike the limit variant of the game, NLH has no restrictions on maximum bet size, so a player is always at risk of losing all of their money (stack) when playing against an opponent with an equal or greater sized stack. This feature of the game changes the strategy significantly – it is very difficult (and usually ill-advised) to bluff in LH, but in NLH, it is a necessary component of any good player's strategy. Effective play becomes critically dependent on an assessment of the opponent's playing style.

With this in mind, I extend the approach employed by Nicolai and Hilderman [1], who use evolutionary neural networks to train a NLH agent. They model opponents with two "aggressiveness" features which calculate scores based on players' full or recent history of actions (fold, check, or bet). While this is a sensible feature type, they acknowledge this component of their model could be improved.

To that end, rather than modeling opponent playing style as a feature, I train a separate neural network for each of a number of hard-coded playing styles. I then combine those models into a final model, which uses a weighted combination of those models, with weights calculated as the relative distance between an opponent's playing style, and that of each of the static classes.

This extension has limited direct application because it is dependent on an artificial and simple classification of opponents, but it is my hope that this approach can be extended by training against more complex opponents (e.g., other no-limit bots) with distinct playing styles. Given a set of bots that roughly simulate the archetypal playing styles of real players, I believe my approach could yield an agent that is more likely to be successful against human players.

II. APPROACH

I use a number of different modules to generate the final adaptive agent. The first is the player class, which can be a hard-coded player, or an adaptive agent. I use five hard-coded types – a random player, a player that always calls, a player that always raises, and then two more sophisticated types which are meant to roughly simulate tight aggressive (TAG) and loose aggressive (LAG) players.

The primary model for the artificial agent is a fully connected feedforward neural net, with one hidden layer. There are four features used as inputs to the neural net, shown in *Table 1* below, 24 nodes in the input layer, 12 nodes in the hidden layer, and 6 outputs. The outputs correspond to the different actions a player can take, and they include folding, calling, or making a bet in one of four different ranges, relative to the pot size.

Table 1: Inputs to Neural Net

Feature	Description
win probability	% of wins, given the game state
# opponent bets	# times an opponent bet this stage
stage	pre-flop, flop, turn, river
position	position relative to dealer

The parameters are learned with an evolutionary algorithm. I begin with 1,000 players and select the parameters from a uniform random distribution, with range (-4, 4). Each player then plays against the selected static opponent 100 times, during which time data is collected about the opponent's observable "style", and also about the player's likelihood to win, given a certain game state. *Table 2* shows the playing style features; these become useful when trying to classify an opponent of an unknown type. There are certainly other observable features which could be useful here, but given the simplicity and consistency of the static player classes, these features are sufficient to obtain a decent classification.

Table 2: Features for Playing Style

Feature	Description
flop%	Percent of flops seen
turn%	Percent of turns seen
river%	Percent of rivers seen
preraise%	Percent of raises before the flop
flopraise%	Percent of raises on the flop
turnraise%	Percent of raises on the flop
riverraise%	Percent of raises on the flop

Table 3 shows the representation of the game state for the opponent model. The number of occurrences of each state is counted, as is the number of times the opponent has the better hand at that point. This data is used later to calculate the win probability, which is in turn used as an input into the neural net. This is a fairly coarse representation of the win probability, and it is stage specific. For example, if the opponent bets preflop, that will have no effect on the win probability estimate on the

flop, turn, or river. The primary motivation for such a simple probability estimate was to keep the game state space manageable. Given the amount of time required to train a model with this evolutionary technique (~3 hours per 100 generations), calculating reliable estimates for a large state space would have required a prohibitive amount of time.

Table 3: Opponent Game State Discretization

Feature	Description
opponent's hand strength	e.g. pair, full house
# opponent bets	# times an opponent bet this stage
stage	pre-flop, flop, turn, river
position	position relative to dealer

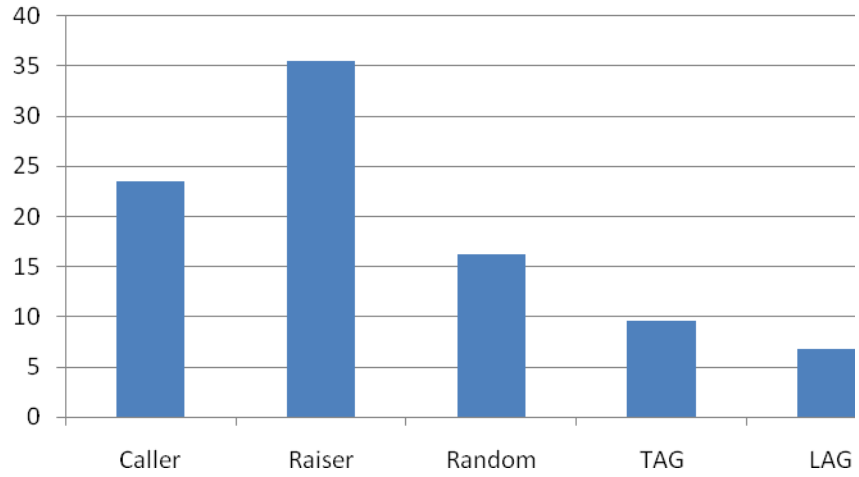
After collecting this data for the opponent model, I then play each of the 1,000 artificial agents against a static opponent 1,000 times each. Each player begins each hand with 100 times the big blind amount (a fairly standard buy-in for no-limit). The earnings for each hand are summed, and after all agents have finished playing, I select the top 10 agents with the highest earnings and discard the rest. With the 10 remaining, I create 24 children agents for each, which are generated by mutating a fraction of the parameters according to a N(0,1) distribution. I then generate 750 more opponents with randomized parameters, and the process repeats for 100 generations. The best model from the last generation is selected as the final model for that class of static player.

After models have been selected for each of the static playing classes, I combine them into a final model. This model estimates an opponent's class of play by measuring the distance of an opponent's observable playing style features to those recorded for each of the static classes. Each of the outputs for the static player types are weighted according to this distance and summed; the action with the highest sum is the final output of the model.

III. RESULTS

Figure 1 shows the results for each of the static player classes. The results are measured as the total amount won, divided by the total number of hands played, divided by the big blind amount (to normalize for different stakes). Unsurprisingly, the agents selected for the "dumb" models (caller, raiser, random) performed significantly better than the agents that played opponents with basic strategy (TAG, LAG). Generally speaking, the results are passable, albeit against weak competition, and still not comparable to what a human would achieve against these opponents. The returns per hand per big blind for the overall final model were 15.4, which is a little less than the average of the static types.

Figure 1: Performance against static player types



IV. CONCLUSION

While the final poker agent is still not competitive with a human opponent, I believe this approach could be significantly improved. The estimates for win probability are very crude, and there are many enhancements to the evolutionary algorithm which could improve its performance. In addition, it would be possible to create more agents by training against the models I developed for the static classes. By increasing the pool of player types, it is possible overall performance against more difficult opponents could improve.

V. REFERENCES

[1] G. Nicolai and R. Hilderman. No-limit Texas hold'em poker agents created with evolutionary neural networks. In 2009 IEEE Symposium on Computational Intelligence and Games, pages 125 – 131.