CS230

AlphaPokerZero? Deep reinforcement learning for imperfect information games

Edgard Bonilla
Department of Physics
Stanford University
edgard@stanford.edu

Divyanshu Murli
Department of Physics
Stanford University
divyansh@stanford.edu

Geoffrey Penington
Department of Physics
Stanford University
geoffp@stanford.edu

Abstract

In this paper we propose a novel algorithm for learning approximate Nash equilibria in imperfect information games with no prior domain knowledge. This algorithm draws from the success of MCTS-based deep reinforcement learning algorithms in perfect information games and is shown to approach Nash equilibrium at Leduc poker. Preliminary tests are conducted for Limit Texas Hold Em. Our approach never requires a complete traversal of the game tree and never stores an entire game strategy. As a result it should scale successfully to very complex games.

1 Introduction

In recent years, deep reinforcement learning has revolutionised AI performance at perfect information games. The same algorithm, AlphaZero, achieved superhuman, state-of-the-art performance in Go, Chess and Shogi [1] [2]. Imperfect information games, where players have hidden private information about the state of the game, pose a more difficult challenge. However, they more accurately reflect real life, which, at least according to Von Neumann, consists mostly of bluffing.

In this paper we develop a novel algorithm that successfully uses deep reinforcement learning to find an approximate Nash equilibrium in a simplified version of poker known as Leduc. Once trained, a single neural network takes in the players' information set and returns an approximate Nash equilibrium strategy. Furthermore, we applied our algorithm to Heads Up Limit Texas Hold Em and found positive preliminary results after limited testing.

Our main objective in this project is not to achieve a better approximation of the Nash equilibrium in Leduc poker compared to some baseline state-of-the-art technique. Classical algorithms can achieve a far higher precision with far less computational power than any deep reinforcement learning based method could ever hope to [3]. The point is to demonstrate a proof-of-concept for our algorithm and show that it successfully converges.

The main value in our algorithm is that it never performs a complete traversal of the game tree and that we never store an entire strategy. Additionally, similar to Alpha Zero, most of its components are highly parallelisable. As a result it should scale successfully to games such as 6 Max No Limit Texas Hold Em, where computers still cannot achieve expert human level performance, even if the computational power required to do so is beyond our resources. Our performance at Leduc was comparable to the main existing deep reinforcement learning algorithm for imperfect information games; however the exact performance at Leduc is not necessarily a good barometer for performance at much more complex games.

CS230: Deep Learning, Winter 2018, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

2 Related work

The dominant technique used in computer poker is counterfactual regret minimisation (CFR) [4], a classical reinforcement learning algorithm. This has achieved great success - recently achieving superhuman performance in Heads Up No limit (HUNL) Texas Hold Em [5]. However CFR relies on a complete traversal of the game tree, which becomes computationally challenging, or even impossible, as the game becomes larger. In particular, superhuman performance has yet to be achieved in No Limit Texas Hold Em with more than two players, which is probably the most widely played version of poker. DeepStack, which is essentially the state of the art in computer poker, does use a deep neural network, pretrained using supervised learning of CFR searches of randomly chosen poker positions, to provide a policy heuristic that allows them to depth-limit the CFR search [6]. Overall, however, imperfect information games have seen far less impact from the deep learning revolution than their perfect information cousins.

The most significant implementation of deep reinforcement learning for imperfect-information games was done by Heinrich and Silver [3][7] with considerable success. Their approach, which they call Neural Fictitious Self-Play (NFSP), calculated a "best response" - an ϵ-greedy strategy based on a deep Q network, as well as an "average strategy" trained to approximate the average best response policy over the entire training time. During the self-play agents played a mixture of the best response and the average policy so as to increase the stability of the learning. The average policy was found to successfully converge to a Nash equilibrium in a simple version of poker known as Leduc and achieved close to state of the art performance on heads up limit Texas Hold Em. Our algorithm aims to build on this success.

Our algorithm makes use of the insights of both NFSP and AlphaZero. Monte Carlo Tree Searches (MCTS), which lie at the heart of the success of AlphaZero, do not in general converge to Nash equilibrium for imperfect information games, although they can quickly learn relatively strong strategies [8]. However here we only use it for the much more limited task of finding a best response for a given fixed strategy. In this sense, the task is much more similar to planning in Partially Observable Markov Decision Processes [9], where an agent samples a probability distribution of the unseen information prior to sampling the game tree.

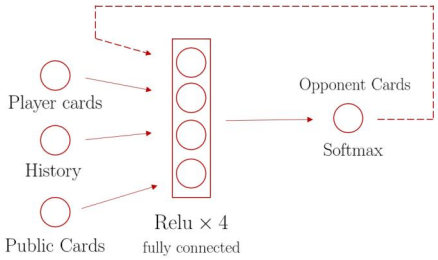
3 Dataset, Features and Methods

Our algorithm makes use of two neural networks: a policy-value network (Fig 1a), similar to the one used in AlphaZero, and an opponent range network (Fig 1b). The policy-value network has two outputs: a policy p, a normalized vector containing the probabilities of performing an action (raise, check or fold), and a value v for the given information state. This value is normalised so that the value of folding is zero and winning the pot corresponds to a value of one.

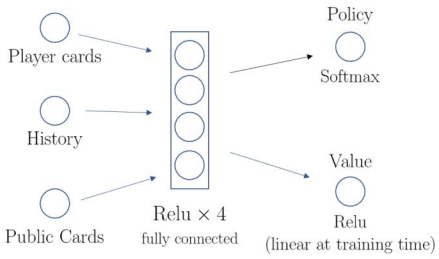
The opponent range network outputs a normalized vector giving probability that the opponent holds each card. This is a vector of length 3 for Leduc and a vector of length 52 for Texas Hold Em.

The inputs for the neural networks are representations of the information states for each player. The games of poker consist of multiple rounds where the players can take different actions, namely {raise,check,fold}. At the end of each round public card(s) are revealed. In Limit hold'em games the number of bets are constrained to be a set amount per round. We therefore encode the input data as follows:

- **Betting History**: Contains the action made by a player during a round after some number of bets have happened. For Leduc it is two players, two rounds, the raises per round can go from zero to two, and three actions. Since these are heads-up games, the game ends as soon as a player folds, so we don't need to encode that action in the history of play. The betting history is therefore stored in a multi-hot array of size $2 \times 2 \times 3 \times 2 \times 2 = 24$. For Limit Texas Hold Em, there are now four rounds with at most four raises per round. The history array size is therefore $2 \times 4 \times 5 \times 2 = 80$.
- **Public cards**: One (or multi-) hot vector for the public card(s). For Leduc there are three different cards, while for Limit Texas Hold Em there are 52. If no public card has been revealed, all elements of the public card array are zero.



(a) Opponent range network, used to predict a probability distribution of the opponent's card(s) given the player's card, history and public cards.



(b) Policy value network, used to predict a player average strategy given the player cards, history and public cards.

Figure 1: Opponent range network (left) and policy value network (right) as implemented for Limit Texas Hold Em

- **Player cards**: This is similar to the public cards, one hot vector of size 3 for Leduc, and a two hot vector of size 52 for Texas Hold Em.

If we flatten and concatenate this information, the input of the neural networks is a vector of length 30 for Leduc and a vector of length 184 for Texas Hold Em.

In Texas Hold Em the opponent range network contains an additional input vector of length 52, This input vector is set to zero when predicting the first card held by the opponent, and is a one hot vector corresponding to the first card that was sampled when predicting the second card. This technique is similar to the techniques used in language sampling models, where the sampled word is used as an input predict the next word. However we do not use a memory state, as in an RNN; instead we just input the player information set to the network on both occasions.

The dataset used was generated from self play between poker agents, all using the same neural nets. Most of the time agents play according to the neural net policy p; however with some probability ($\eta = 0.1$) they play an approximate "best response" strategy to the policy p. This is known as anticipatory learning [10] and is used similarly in NFSP.

When playing a "best response" strategy, an agent uses a policy π recommended by a Monte Carlo tree search (MCTS) adapted to imperfect information states:

Within the MCTS, the opponent card(s) are sampled using the opponent range network, and the opponent plays according by Monte Carlo sampling of the policy p. Meanwhile, the agent plays according to an upper confidence bound strategy guided by p, a greedy strategy that maximises

$$Q(s,a) + \sqrt{N(s)}\frac{p(s,a)}{1+N(s,a)}, \tag{1}$$

where $Q(s,a)$ is the average value achieved by doing action a in state s, $N(s)$ is the number of times the state has been visited and $N(s,a)$ is the number of times action a has been taken. The returned strategy π is proportional to $N(s_0,a)^{1/\tau}$, where τ is a tunable temperature parameter and s_0 is the root node where MCTS was called. The value v from the neural network is used to bootstrap the search.

If the temperature is τ less than one, and if the values produced by each strategy are similar (as should be the case close to Nash equilibrium), then this best response strategy would be excessively biased in favour of the action with the largest p and could push the average strategy away from Nash equilibrium. As a result the initial policy at the root node was replaced with p^τ.

The "best response" strategies for the entire history of self play are stored on a reservoir $\mathcal{R}_p$, that is used to train the policy p of the neural network with a cross entropy loss. This approximates fictional self play (FSP) where agents play the true best response to the previous average strategy, and where convergence of the average strategy to a Nash equilibrium is guaranteed [3]. Reservoir sampling [11] is used to ensure $\mathcal{R}_p$, contains an even distribution of the training data.

The value estimate v and opponent range estimate are trained using a smaller dataset $\mathcal{D}_{v,c}$ based on all recent games (regardless of which policy was used). The value estimate is trained to predict the

actual outcome of the game and uses an L2 loss. L2 regularisation was used to prevent overfitting, giving an overall loss function

$$J_{p,v} = -\alpha \sum_m \pi^T \log p + \sum_n (v - v_\star)^2 + \lambda_1 \sum \|w\|^2 \tag{2}$$

where p and v are the outputs of the neural network, π is the best response strategy produced by the tree search, $v_\star$ is the value based on the actual game result and α and λ_1 are hyperparameters. m and n range over a minibatch of examples from $\mathcal{R}_p$ and $\mathcal{D}_{v,c}$ respectively.

The opponent range network is trained to predict the actual opponent card(s) and uses a cross entropy loss. For Texas Hold Em, each minibatch contains both samples where one opponent card is given and the network is trained to predict the other card and samples where the network is only given the player information set and is trained to predict either of the actual cards. L2 regularisation was used to prevent overfitting, giving an overall loss function

$$J_{op} = -\sum_n c^T \log \hat{c} + \lambda_2 \sum \|w\|^2 \tag{3}$$

where $\hat{c}$ is the neural network estimate of the opponent cards and c is the ground truth opponent cards.

4 Experiments/Results/Discussion

The neural networks were trained from an updated $\mathcal{R}_p$ and $\mathcal{D}_{v,c}$, by performing 128 steps of training with minibatch size 128 after each episode of self play. An episode consisted of 128 full games between the agents. We used the Adam optimisation algorithm with different learning rates depending on the experiments. The L2 regularisation parameters were set to $\lambda_1 = \lambda_2 = 0.01$, and $\alpha = 10$.

4.1 Leduc Poker

For Leduc Poker the policy value network both had two hidden layers with 256 and 128 units. The learning rate for the policy value network was set to 0.0005, and to 0.005 for the opponent range network after some initial tests.

We use exploitability as a metric to evaluate the performance of our algorithm. In a two player zero-sum game, the exploitability of a player strategy is the maximum average value that any opposing strategy can make from it. It is a reliable metric for convergence; a strategy with an exploitability of 2ϵ is a ϵ-Nash Equilibrium strategy.

We performed three experiments as shown in Fig 2a, two with constant temperatures ($\tau \approx 0$ and $\tau = 1$), and one with a temperature decay such that $\tau \leftarrow \tau / 1.005$ on each episode of training. As expected, the zero-temperature case converges the fastest; this is equivalent the player playing best-response selecting the most probable action rather than sampling from its entire strategy distribution. The necessity for exploration in the MCTS is leveraged by using p^τ in the root node.

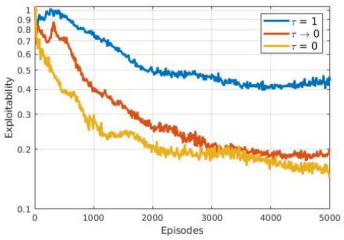
We observe that for low temperatures the algorithm is successfully converging to a Nash equilibrium strategy. More specifically, the $\tau = 0$ achieves an exploitability of $\lesssim 0.15$ after 5000 episodes of play. This performance is comparable to the results from NSFP ([7], Fig 1a) after the same amount of self play game data, which is the closest thing we have to a baseline model given that Leduc poker itself is a game where classical algorithms will always vastly outperform deep reinforcement learning.

4.2 Heads Up Limit Texas Hold Em (LHE)

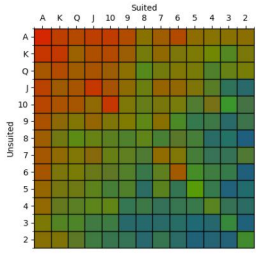
For this experiment, the policy value network both had four hidden layers: three with 256 units and the last one with 128 units. The learning rate for the policy value network was set to 0.0005, and to 0.0005 for the opponent range network after some initial tests.

The results shown in Fig 2b correspond to the preflop strategies learned after 2000 episodes of training. The algorithm is already able to discern which cards have the highest value in the game, also has an increased propensity to raise when it has pairs, shown in the diagonal of the diagram. This is achieved despite no preprocessing in the data to indicate the card number or suit.

After this relatively small amount of training, the neural net was already able to beat very simple strategies such as always calling, although there was considerable noise given the number of games



(a) Performance at Leduc Poker with the MCTS temperature parameter set to $\tau = 1$ (blue), $\tau = 0$ (yellow) and gradually decaying from $\tau = 1$ (red). Zero exploitability corresponds to the Nash equilibrium solution.



(b) Preflop strategy for Heads Up Limit Texas Hold Em after 2000 iterations. Red indicates raise, green call and blue fold. The upper right triangle shows the strategy for suited cards, the lower left triangle shows the strategy for unsuited cards.

Figure 2: Exploitability over 5000 training episodes on Leduc (left), and colour plot strategy shown after 2000 episodes of training of Heads Up Limit Texas Hold Em (right).

that we were able to simulate. Gameplay against humans was done, but would require hundreds of times more games to be statistical significant.

Given the training required for NFSP to achieve high performance at LHE, we would need to train much longer on GPUs for our algorithm to have a chance of holding its own against a state-of-the-art poker bot such as the Cepheus project [12], even if we had the computational power to measure that performance with any degree of precision.

5 Conclusion/Future Work

- Our algorithm successfully converged to an approximate (0.07) Nash equilibrium in Leduc Poker. It provides an alternative approach to NFSP for deep reinforcement learning in imperfect information games.
- The performance at Leduc Poker was similar, yet not quite as high as for NFSP. However, as we have optimised hyperparameters and algorithmic details, the performance of our algorithm has steadily improved. We are confident that further performance gains are possible.
- MCTS algorithms are most effective for very deep games. This suggests our algorithm should be even more useful in 'real poker', and especially in even deeper imperfect information games such as various board games.
- Our algorithm appears to be learning Heads Up Limit Texas Hold Em successfully but we were limited in our ability to measure performance, because of the time required to simulate sufficient games given our limited computational power.
- Future plans include continuing to optimise performance on Leduc poker. Just in the last few days, various optimisations have improved the exploitability achieved by a factor of two.
- We will also look to test our algorithm's performance at Limit Texas Hold Em against state-of-the-art poker bots.
- We can apply our algorithm to Heads Up No Limit Texas Hold Em, where superhuman performance has only been achieved in the last year, as well as poker games with more than two players, where humans still reign supreme.
- Finally, the algorithm can be easily adapted to various other imperfect information games, particularly games where very deep tree searches are vital to success.

6 Contributions

Every team member contributed to the project in numerous ways. We did not keep track of who contributed exactly which piece of code and everyone was involved with every major part of the project.

References

[1] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354, 2017.

[2] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv preprint arXiv:1712.01815*, 2017.

[3] Johannes Heinrich, Marc Lanctot, and David Silver. Fictitious self-play in extensive-form games. In *International Conference on Machine Learning*, pages 805–813, 2015.

[4] Martin Zinkevich, Michael Johanson, Michael Bowling, and Carmelo Piccione. Regret minimization in games with incomplete information. In *Advances in neural information processing systems*, pages 1729–1736, 2008.

[5] Noam Brown and Tuomas Sandholm. Libratus: the superhuman ai for no-limit poker. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*, 2017.

[6] Matej Moravčík, Martin Schmid, Neil Burch, Viliam Lisỳ, Dustin Morrill, Nolan Bard, Trevor Davis, Kevin Waugh, Michael Johanson, and Michael Bowling. Deepstack: Expert-level artificial intelligence in heads-up no-limit poker. *Science*, 356(6337):508–513, 2017.

[7] Johannes Heinrich and David Silver. Deep reinforcement learning from self-play in imperfect-information games. *arXiv preprint arXiv:1603.01121*, 2016.

[8] Johannes Heinrich and David Silver. Smooth uct search in computer poker. In *IJCAI*, pages 554–560, 2015.

[9] David Silver and Joel Veness. Monte-carlo planning in large pomdps. In *Advances in neural information processing systems*, pages 2164–2172, 2010.

[10] Jeff S Shamma and Gürdal Arslan. Dynamic fictitious play, dynamic gradient play, and distributed convergence to nash equilibria. *IEEE Transactions on Automatic Control*, 50(3):312–327, 2005.

[11] Jeffrey S Vitter. Random sampling with a reservoir. *ACM Transactions on Mathematical Software (TOMS)*, 11(1):37–57, 1985.

[12] Michael Bowling, Neil Burch, Michael Johanson, and Oskari Tammelin. Heads-up limit hold'em poker is solved. *Science*, 347(6218):145–149, 2015.