May the Flop be with you - Deep RL based No-Limit Texas Hold'em Pokerbot

Srivatsan Srinivasan SRIVATSANSRINIVASAN@G.HARVARD.EDU
Sebastien Baur SEBASTIENBAUR@G.HARVARD.EDU
Donghun Lee DONGHUNLEE@G.HARVARD.EDU

Abstract

Imperfect information games such as Poker have always proven challenging to master for an AI agent . In this work, we introduce a self-sufficient end-to-end approach to learning approximate Nash equilibria in such games without prior domain knowledge. For the learning agent, we have utilized the architecture of Neural Fictitious Self Play(NFSP) where each playing agent stores separate experience buffers and trains two neural networks that learns best action values and average strategy using off-policy deep Reinforcement Learning and supervised learning. We also develop sub-networks that learn abstracted states and employ frozen hidden layer weights to represent hand and board in a reduced dimension space. We then demonstrate the learning model's performance through structured experiments of the agent's game play against baseline agents and its own clones.

1. Introduction

Poker is an archetypal game of imperfect information. The challenges involved in poker include sophisticated decision sequences, large state-action space, imperfect information and adversarial behavior, presenting a perfect template for probabilistic inference and deep reinforcement learning to sequentially learn from the game states via non-linear models. Heads-up no-limit Texas Hold'em (HUNL) is a two-player version of poker in which two cards are initially dealt face down to each player(pre-flop), and additional cards are dealt face up in three subsequent rounds(flop(3),turn(1) and river(1)). No limit is placed on the size of the bets although there is an overall limit to the total amount wagered in each game.

Imperfect information games demand way more complex reasoning than perfect information games in the same state space. The main complication is due to recursive reasoning, where an appropriate decision at any point in the game is a function of the probability distribution over latent(private) information of your opponent(cards, strategy) and opponents' mechanism of revealing that information depends on the players' actions. This makes it impossible to reason about a game state in isolation, the backbone of usual heuristic search methods in perfect information games. Combination of imperfect information with the humongous state space of poker also renders classic supervised learning through simulation of all possible scenarios prohibitively large. Hence an off-policy deep Reinforcement Learning framework where the agent learns through self-play(learning by playing against its own clones) is a natural direction that this work adopted in the training procedure.

2. Background

In this section, we provide a brief overview of deep reinforcement learning, extensive form games and Fictitious Self-Play. We would suggest the readers to refer (Sutton & Barto, 1998) for fundamentals of Reinforcement Learning, (Schaul et al., 2015) and (Silver et al., 2016) for Experience Replay and for Double DQNs and (Heinrich et al., 2015) for extensive form games and fictitious self-play to assimilate a more in-depth background.

2.1. Extensive Form Games

Extensive form games(Wikipedia) allow explicit representation of pivotal game aspects, like players' move sequences, choices made, the information each player has about the other players' decision mechanism and the utilities for each resulting action. A perfect information game is characterized by a player knowing

Submitted as part of final project requirements in *CS281:Advanced Machine Learning*, Advisor: Prof. Alexander S. Rush, Fall-2017 SEAS, Harvard University..

exactly the history of happenings in the game and the information set is the same singleton for both players. Games that fall under the purview of this include tic-tac-toe, checkers and Go. Most card games such as Poker are imperfect information games. In **imperfect-information games**, each player only observes his own information states(only his own cards in poker) and generates a strategy profile that maps information states to a distribution over possible actions. A **strategy profile** (Heinrich & Silver, 2016) is a collection of strategies of all players and a best response is defined by a strategy that obtains optimal payoff against the other strategies in the strategy profile. In this context, a **Nash equilibrium** is a strategy profile such that each player's strategy in the profile is a best-response to the other strategies, essentially a deadlock where no player can gain by deviating from this strategy.

2.2. Reinforcement Learning

Reinforcement learning(RL) agents typically try to maximize the expected reward when reacting with its given environment (Sutton & Barto, 1998). Usually, the environment is modeled under the MDP framework, where transition from a state does not depend on history, and overall the agent tries to find a trajectory that maximizes its rewards. Many reinforcement learning algorithms learn from sequential experience in the form of transition tuples $(s_t; a_t; r_{t+1}; s_{t+1})$. The problem is setup to learn the action-value function Q(s,a) which is the value of an action taken from a state and adhering to the base policy from that step forward and can be setup to learn off-policy, i.e. learn all Q-values while the agent enacts a policy different from the one suggested by Q-values. **Q-Learning**(Watkins & Dayan, 1992) is an algorithm mostly to this effect where we learn from a batch of transitions. **Double-DQN** with experience replay(Silver et al., 2016) is a recent advancement in Q-learning where the agent learns Q-values from experience data by fitting neural networks in the state-action space.

2.3. Fictitious Self-Play

Fictitious Play is a game-theory framework where an agent chooses its best-response to the expected(average) opponent's response profile. It has been proven that the average strategy of the fictitious players converge to Nash equilibrium in two-player zero sum games(both for perfect and imperfect information games)(Leslie & Collins, 2006). Advances in machine learning and reinforcement learning have helped develop approximations to these two profiles using game events. Fictitious self-play, introduced by (Heinrich et al., 2015), replaces best response and average response computation with Q and π(policy) values learned from reinforcement learning and supervised learning respectively. These FSP agents generate their dataset of experiences via self-play in which they store their (s,a,r,s') memory in a RL reservoir buffer and the agents' own behavior (s,a) in a supervised learning circular buffer. The bot's RL memory approximates data of an MDP defined by the other players' average strategy profile, making the Q-values approximate the best response. Similarly, learning an average strategy becomes a classification task through the agent's own supervised learning experiences.

3. Related Work

4. Model

5. Inference

6. Methods

7. Results

8. Discussion

9. Conclusion

10. State-Action Space Representation

10.1. State Space

The cards are one-hot-encoded arrays. The board and the player's hand are separated so that they private and public information can be processed differently The board is decomposed in 3 parts (for 3 betting rounds), so that the neural network can process each part with the actions that were taken in a given betting round. We also take into account the actions that were taken in a given episode. They are represented as a 4x5x6x2 array (4 betting rounds, 5 types of action, at most 6 bets per player in a given betting round, 2 players). The action representation encodes allowed actions such as bet, call, check, going all-in, raise. To bound the possible number of actions per betting round, we forbid the players to do more two min-raise by betting round (after 2 they have to double). A last array contains the blinds, money stacks, pot, and dealer button values.

10.2. Hand Strength Evaluation

The number of possible combinations in the aforementioned state space representation is humongous and hence, the learning problem becomes computationally

intractable. We resort to the structure of the game in order to implicitly cluster the state-action possibilities based on possible outcomes. For instance, consider an extreme case of a board that has Ah, Ad, Qh, Jd, Kd and assume the agent has two non-face cards that don't form a pair. Irrespective of the the cards' numerical value, the agent is expected to take the same action in the game, which implicitly reduces potentially a $\binom{10}{2}$ hole cards possibilities into a single point estimate. We intend to capture this intuition through hand strength evaluation. With h being the set of hole cards and b being the set of board cards, HS(h,b)is defined as follows. Here, p(w) is the probability of the agent winning the game(assuming a final showdown) and the games being simulated through 100000 Monte Carlo simulations(w ith opponent cards and remaining portion of board being randomly sampled) for each combo of cards. $p(s_i)$ similarly suggests the probability of obtaining different final hands(Straight, flush etc.) from the current hole and board under MC simulations.

$$HS(h,b) = \left[p(w^{(hb)}), p(s_1^{(hb)}), ...p(s_n^{(hb)})\right]$$

10.3. State Abstraction through NN

We calculate the hand strengths of the pre-flop, flop, turn and river(trivial) cards using the methodology suggested above. Since the number of combinations is pretty huge(O(10^7)), we created a lookup table for the hand strength of these combinations. Then we build a fully-connected neural network that utilizes the hand strength evaluation table as the labeled dataset in order to predict these metrics, given a set of hole and board cards. The output of this NN is fed into our master NFSP network(to follow). Our Q network has an auxiliary output that tries to predict the hand strength based only on the board and the cards, so that similar situations get represented similarly at the last hidden layer of this sub-network - a representation that is then used to predict the Q values.

11. Neural Fictitious Self-Play(NFSP)

11.1. Prioritized Experience Replay

In the Reinforcement Learning setting, RL agents incrementally update their parameters (of the policy, value function or model) while they observe a stream of experience. In their simplest form, they discard incoming data immediately, after a single update. Two issues with this are (a) strongly correlated updates that break the i.i.d. assumption of many popular stochastic gradient-based algorithms, and (b) the rapid forgetting of possibly rare experiences that would be useful later on. Experience replay (Lin, 1992) addresses both of these issues: with experience stored in a replay memory, it becomes possible to break the temporal correlations by mixing more and less recent experience for the updates, and rare experience will be used for more than just a single update. This was demonstrated in the Deep Q-Network (DQN) algorithm (Mnih et al., 2013; 2015), which stabilized the training of a value function, represented by a deep neural network, by using experience replay.

Here we use a modified version of experience replay, called PER - Prioritized Experience Replay((Schaul et.al., 2016)for storing our transition buffers. PER samples from the experience buffer with higher probability weight to transitions with larger TD error. The two classic approaches would include a proportional probability weighting scheme and a rank-based probability weighting scheme. Since the proportional weighting scheme is sensitive to outliers at times, we go ahead with the rank-based weighting scheme, where each transition is as probable as the inverse rank of TD-error. We use the same architecture for our RL buffer and SL buffer in the NFSP algorithm which follows.

11.2. NFSP

NFSP(Heinrich and Silver, 2016) combines FSP with neural network function approximation. In algorithm 1 all players of the game are controlled by separate NFSP agents that learn from simultaneous play against each other, i.e. self-play. An NFSP agent interacts with its fellow agents and memorizes its experience of game transitions and its own best response behaviour in two memories,M_{RL} and M_{SL}. NFSP treats these memories as two distinct datasets suitable for deep reinforcement learning and supervised classification respectively. The agent trains a neural network, $Q(s; a|\theta^Q)$, to predict action values from data in M_{RL} using off-policy reinforcement learning. The resulting network defines the agent's approximate best response strategy, ϵ-greedy(Q), which selects a random action with probability ϵ and otherwise chooses the action that maximizes the predicted action values. The agent trains a separate neural network, $\Pi(s; a|\theta^\Pi)$, to imitate its own past best response behaviour using supervised classification on the data in M_{SL}. This network maps states to action probabilities and defines the agent's average strategy, $\pi = \Pi$. During play, the agent chooses its actions from a mixture of its two strategies β and Π.

If we want all agents to learn simultaneously while playing against each other, we face the following

dilemma.In principle, each agent could play its average policy, Π, and learn a best response with off-policy Q-learning, i.e. evaluate and maximize its action values, $Q_i(s;a)$ of playing its best response policy β_i against its fellow agents' average strategy profile, Π_{-i}. However, in this case the agent would not generate any experience of its own best response behaviour, which is needed to train its average policy network that approximates the agent's average of past best responses. Hence, NFSP agents choose their actions from the mixture of these policies. This enables each agent to compute an approximate best response, β_i, to its opponents' anticipated average strategy profile by iteratively evaluating and maximizing its action values. Additionally, as each agent's own best response policy is now sampled in proportion to the anticipatory parameter, they can now train their average policy networks from that experience.

Algorithm 1 Neural Fictitious Self-Play

Initialize game G with 2 players.
Execute each player via runAgent.
function runAgent(G):
  Initialize replay memories M_{RL}, M_{SL}
  Initialize action-value network $Q(s,a|\theta^Q)$
  Initialize average-policy network $\Pi(s,a|\theta^\pi)$
  Initialize anticipatory hyper-parameter η
  Define L_{SL} as Softmax loss on π
  Define L_{RL} as TD Error loss on Q value.
  for each episode **do**
    σ: P($\sigma = \epsilon$ greedy Q) $= \eta$, else $\sigma = \Pi$
    Observe s_1, r_1
    for t=1,T **do**
      Sample action at from policy σ
      Execute a_t and Observe r_{t+1}, s_{t+1}
      Store $(s_t, a_t, r_{t+1}, s_{t+1}$ in $M_{RL})$
      if $\sigma = \epsilon$ greedy Q **then**
        Store (s_t, a_t) in M_{SL}
      end if
      Update θ^Π with SGD on loss L_{SL}
      Update θ^Q with SGD on loss L_{RL}
    end for
  end for

References

Heinrich, J. and Silver, D. Deep reinforcement learning from self-play in imperfect-information games. *arXiv: 1603.0112*, 2016.

Heinrich, J., Lanctot, M., and Silver, D. Fictitious self-play in extensive-form games. In *Proceedings of the 32nd International Conference on Machine Learning*, 2015.

Leslie, D. S. and Collins, E.J. Generalised weakened fictitious play. *Games and Economic Behavior*, 2006.

Schaul, T., Quann, J., Antanouglou, I., and Silver, D. Prioritized experience replay. *arXiv: 1511.05952*, 2015.

Silver, D., Guez, A., and van Hasselt, H. Deep reinforcement learning with double-q learning. *arXiv: 1509.06461*, 2016.

Sutton, R. S. and Barto, A. G. (eds.). *Reinforcement Learning - An Introduction, Volume-1*. MIT Printing Press, 1998.

Watkins and Dayan. Q-learning. In *Machine Learning*, volume 8, pp. 279–292, 1992.

Wikipedia. Extensive-form games. https://en.wikipedia.org/wiki/Extensive-form_game.