Deep Reinforcement Learning from Self-Play in Imperfect-Information Games

Johannes Heinrich
University College London, UK
j.heinrich@cs.ucl.ac.uk

David Silver
University College London, UK
d.silver@cs.ucl.ac.uk

Abstract

Many real-world applications can be described as large-scale games of imperfect information. To deal with these challenging domains, prior work has focused on computing Nash equilibria in a handcrafted abstraction of the domain. In this paper we introduce the first scalable end-to-end approach to learning approximate Nash equilibria without prior domain knowledge. Our method combines fictitious self-play with deep reinforcement learning. When applied to Leduc poker, Neural Fictitious Self-Play (NFSP) approached a Nash equilibrium, whereas common reinforcement learning methods diverged. In Limit Texas Hold'em, a poker game of real-world scale, NFSP learnt a strategy that approached the performance of state-of-the-art, superhuman algorithms based on significant domain expertise.

1 Introduction

Games have a tradition of encouraging advances in artificial intelligence and machine learning (Samuel, 1959; Tesauro, 1995; Campbell et al., 2002; Gelly et al., 2012; Bowling et al., 2015). A game is a domain of conflict or cooperation between several entities (Myerson, 1991). One motivation for studying recreational games is to develop algorithms that will scale to more complex, real-world games such as airport and network security, financial and energy trading, traffic control and routing (Lambert III et al., 2005; Nevmyvaka et al., 2006; Bazzan, 2009; Tambe, 2011; Urieli and Stone, 2014). Most of these real-world games involve decision making with imperfect information and high-dimensional information state spaces. An optimal (and in principle learnable) solution to these games would be a Nash equilibrium, i.e. a strategy from which no agent would choose to deviate. While many machine learning methods have achieved near-optimal solutions to classical, perfect-information games, these methods fail to converge in imperfect-information games. On the other hand, many game-theoretic approaches for finding Nash equilibria lack the ability to learn abstract patterns and use them to generalise to novel situations. This results in limited scalability to large games, unless the domain is abstracted to a manageable size using human expert knowledge, heuristics or modelling. However, acquiring human expertise often requires expensive resources and time. In addition, humans can be easily fooled into irrational decisions or assumptions (Selten, 1990). This motivates algorithms that learn useful strategies end-to-end.

Fictitious play (Brown, 1951) is a popular method for learning Nash equilibria in normal-form (single-step) games. Fictitious players choose best responses (i.e. optimal counter-strategies) to their opponents' average behaviour. Fictitious Self-Play (FSP) (Heinrich et al., 2015) extends this method to extensive-form (multi-step) games. In this paper we introduce NFSP, a deep reinforcement learning method for learning *approximate* Nash equilibria of imperfect-information games. NFSP combines FSP with neural network function approximation. An NFSP agent consists of two neural networks. The first network is trained by reinforcement learning from memorized experience of play against fellow agents. This network learns an approximate best response to the historical behaviour of other agents. The second network is trained by supervised learning from memorized experience of the agent's own behaviour. This network learns a model that averages over the agent's own historical strategies. The agent behaves according to a mixture of its average strategy and best response strategy.

We empirically evaluate our method in two-player zero-sum computer poker games. In this domain, current game-theoretic approaches use heuristics of card strength to abstract the game to a tractable size (Johanson et al., 2013). Our approach does not rely on engineering such abstractions or any other prior domain knowledge. NFSP agents leverage deep reinforcement learning to learn directly from their experience of interacting with other agents in the game. When applied to Leduc poker, NFSP approached a Nash equilibrium, whereas common reinforcement learning methods diverged. We also applied NFSP to Limit Texas Hold'em (LHE), learning directly from the raw inputs. NFSP learnt a strategy that approached the performance of state-of-the-art, superhuman methods based on handcrafted abstractions.

2 Background

In this section we provide a brief overview of reinforcement learning, extensive-form games and fictitious self-play. For a more detailed exposition we refer the reader to (Sutton and Barto, 1998), (Myerson, 1991), (Fudenberg, 1998) and (Heinrich et al., 2015).

2.1 Reinforcement Learning

Reinforcement learning (Sutton and Barto, 1998) agents typically learn to maximize their expected future rewards from interaction with an environment. The environment is usually modelled as a **Markov decision process (MDP)**. An agent behaves according to a **policy** that specifies a distribution over available actions at each state of the MDP. The agent's goal is to improve its policy in order to maximize its **gain**, $G_t = \sum_{i=t}^{T} R_{i+1}$, which is a random variable of the agent's cumulative future rewards starting from time t.

Many reinforcement learning algorithms learn from sequential **experience** in the form of transition tuples, $(s_t, a_t, r_{t+1}, s_{t+1})$, where s_t is the state at time t, a_t is the action chosen in that state, r_{t+1} the reward received thereafter and s_{t+1} the next state that the agent transitioned to. A common objective is to learn the **action-value function**, $Q(s,a) = \mathbb{E}^{\pi}\left[G_t \mid S_t = s, A_t = a\right]$, defined as the expected gain of taking action a in state s and following policy π thereafter. An agent is learning **on-policy** if it learns about the policy that it is currently following. In the **off-policy** setting an agent learns from experience of another agent or another policy, e.g. a previous policy.

Q-learning (Watkins and Dayan, 1992) is a popular off-policy reinforcement learning method. It learns about the greedy policy, which at each state takes the action of the highest estimated value. Storing and replaying past experience with off-policy reinforcement learning from the respective transitions is known as experience replay (Lin, 1992). Fitted Q Iteration (FQI) (Ernst et al., 2005) is a batch reinforcement learning method that replays experience with Q-learning. Neural Fitted Q Iteration (NFQ) (Riedmiller, 2005) and Deep Q Network (DQN) (Mnih et al., 2015) are extensions of FQI that use neural network function approximation with batch and online updates respectively.

2.2 Extensive-Form Games

Extensive-form games are a model of sequential interaction involving multiple players. Assuming rationality, each player's goal is to maximize his payoff in the game. In imperfect-information games, each player only observes his respective **information states**, e.g. in a poker game a player only knows his own private cards but not those of other players. Each player chooses a **behavioural strategy** that maps information states to probability distributions over available actions. We assume games with **perfect recall**, i.e. each player's current information state s_t^i implies knowledge of the sequence of his information states and actions, $s_1^i, a_1^i, s_2^i, a_2^i, ..., s_t^i$, that led to this information state. The **realization-probability** (Von Stengel, 1996), $x_{\pi^i}(s_t^i) = \prod_{k=1}^{t-1} \pi^i(s_k^i, a_k^i)$, determines the probability that player i's behavioural strategy, π^i, contributes to realizing his information state s_t^i. A **strategy profile** $\pi = (\pi^1, ..., \pi^n)$ is a collection of strategies for all players. π^{-i} refers to all strategies in π except π^i. Given a fixed strategy profile π^{-i}, any strategy of player i that achieves optimal payoff performance against π^{-i} is a **best response**. An approximate or ϵ-best response is suboptimal by no more than ϵ. A **Nash equilibrium** is a strategy profile such that each player's strategy in this profile is a best response to the other strategies. Similarly, an approximate or ϵ-Nash equilibrium is a profile of ϵ-best responses. In a Nash equilibrium no player can gain by deviating from his strategy. Therefore, a Nash equilibrium can be regarded as a fixed point of rational self-play learning. In fact, Nash equilibria are the only strategy profiles that rational agents can hope to converge on in self-play (Bowling and Veloso, 2001).

2.3 Fictitious Self-Play

Fictitious play (Brown, 1951) is a game-theoretic model of learning from self-play. Fictitious players choose best responses to their opponents' average behaviour. The average strategies of fictitious players converge to Nash equilibria in certain classes of games, e.g. two-player zero-sum and many-player potential games (Robinson, 1951; Monderer and Shapley, 1996). Leslie and Collins (2006) introduced generalised weakened fictitious play. It has similar convergence guarantees as common fictitious play, but allows for approximate best responses and perturbed average strategy updates, making it particularly suitable for machine learning.

Fictitious play is commonly defined in normal form, which is exponentially less efficient for extensive-form games. Heinrich et al. (2015) introduce **Full-Width Extensive-Form Fictitious Play (XFP)** that enables fictitious players to update their strategies in behavioural, extensive form, resulting in linear time and space complexity. A key insight is that for a convex combination of normal-form strategies, $\hat{\sigma} = \lambda_1 \hat{\pi}_1 + \lambda_2 \hat{\pi}_2$, we can achieve a realization-equivalent behavioural strategy σ, by setting it to be proportional to the respective convex combination of realization-probabilities,

$$\sigma(s,a) \propto \lambda_1 x_{\pi_1}(s)\pi_1(s,a) + \lambda_2 x_{\pi_2}(s)\pi_2(s,a) \quad \forall s,a, \tag{1}$$

where $\lambda_1 x_{\pi_1}(s) + \lambda_2 x_{\pi_2}(s)$ is the normalizing constant for the strategy at information state s. In addition to defining a full-width average strategy update of fictitious players in behavioural strategies, equation (1) prescribes a way to sample data sets of such convex combinations of strategies. Heinrich et al. (2015) introduce **Fictitious Self-Play (FSP)**, a sample- and machine learning-based class of algorithms that approximate XFP. FSP replaces the best response computation and the average strategy updates with reinforcement and supervised learning respectively. In particular, FSP agents generate datasets of their experience in self-play. Each agent stores its experienced transition tuples, $(s_t, a_t, r_{t+1}, s_{t+1})$, in a memory, $\mathcal{M}_{RL}$, designated for reinforcement learning. Experience of the agent's own behaviour, (s_t, a_t), is stored in a separate memory, $\mathcal{M}_{SL}$, designated for supervised learning. Self-play sampling is set up in a way that an agent's reinforcement learning memory approximates data of an MDP defined by the other players' average strategy profile. Thus, an approximate solution of the MDP by reinforcement learning yields an approximate best response. Similarly, an agent's supervised learning memory approximates data of the agent's own average strategy, which can be learned by supervised classification.

3 Neural Fictitious Self-Play

NFSP 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, $\mathcal{M}_{RL}$ and $\mathcal{M}_{SL}$. 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 $\mathcal{M}_{RL}$ using off-policy reinforcement learning. The resulting network defines the agent's approximate best response strategy, $\beta = \epsilon\text{-greedy}(Q)$, 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 $\mathcal{M}_{SL}$. 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. NFSP also makes use of two technical innovations in order to ensure the stability of the resulting algorithm as well as enable simultaneous self-play learning. First, it uses reservoir sampling (Vitter, 1985) to avoid windowing artifacts due to sampling from a finite memory. Second, it uses anticipatory dynamics (Shamma and Arslan, 2005) to enable each agent to both sample its own best response behaviour and more effectively track changes to opponents' behaviour.

Algorithm 1 Neural Fictitious Self-Play (NFSP) with fitted Q-learning

```
Initialize game $\Gamma$ and execute an agent via RunAgent for each player in the game
function RunAgent($\Gamma$)
    Initialize replay memories $\mathcal{M}_{RL}$ (circular buffer) and $\mathcal{M}_{SL}$ (reservoir)
    Initialize average-policy network $\Pi(s, a \mid \theta^{\Pi})$ with random parameters $\theta^{\Pi}$
    Initialize action-value network $Q(s, a \mid \theta^{Q})$ with random parameters $\theta^{Q}$
    Initialize target network parameters $\theta^{Q'} \leftarrow \theta^{Q}$
    Initialize anticipatory parameter $\eta$
    for each episode do
        Set policy $\sigma \leftarrow \begin{cases} \epsilon\text{-greedy}\,(Q)\,, & \text{with probability } \eta \\ \Pi, & \text{with probability } 1-\eta \end{cases}$
        Observe initial information state $s_1$ and reward $r_1$
        for $t = 1, T$ do
            Sample action $a_t$ from policy $\sigma$
            Execute action $a_t$ in game and observe reward $r_{t+1}$ and next information state $s_{t+1}$
            Store transition $(s_t, a_t, r_{t+1}, s_{t+1})$ in reinforcement learning memory $\mathcal{M}_{RL}$
            if agent follows best response policy $\sigma = \epsilon$-greedy $(Q)$ then
                Store behaviour tuple $(s_t, a_t)$ in supervised learning memory $\mathcal{M}_{SL}$
            end if
            Update $\theta^{\Pi}$ with stochastic gradient descent on loss
                $\mathcal{L}(\theta^{\Pi}) = \mathbb{E}_{(s,a)\sim\mathcal{M}_{SL}}\left[-\log \Pi(s, a \mid \theta^{\Pi})\right]$
            Update $\theta^{Q}$ with stochastic gradient descent on loss
                $\mathcal{L}\left(\theta^{Q}\right) = \mathbb{E}_{(s,a,r,s')\sim\mathcal{M}_{RL}}\left[\left(r + \max_{a'} Q(s', a' \mid \theta^{Q'}) - Q(s, a \mid \theta^{Q})\right)^2\right]$
            Periodically update target network parameters $\theta^{Q'} \leftarrow \theta^{Q}$
        end for
    end for
end function
```

Fictitious play usually keeps track of the average of normal-form best response strategies that players have chosen in the game, $\hat{\pi}^i_T = \frac{1}{T}\sum_{t=1}^{T}\hat{\beta}^i_t$. Heinrich et al. (2015) propose to use sampling and machine learning to generate data on and learn convex combinations of normal-form strategies in extensive form according to equation (1). E.g. we can generate a set of extensive-form data of π^i_T by sampling whole episodes of the game, using $\beta^i_t, t = 1, \dots, T$, in proportion to their weight, $\frac{1}{T}$, in the convex combination. NFSP agents use reservoir sampling (Vitter, 1985; Osborne et al., 2014) to memorize experience of their historical best responses. The agent's supervised learning memory, $\mathcal{M}_{SL}$, is a reservoir to which it only adds experience when following its approximate best response policy, $\beta = \epsilon\text{-greedy}(Q)$. An NFSP agent regularly trains its average policy network, $\Pi(s, a \mid \theta^{\Pi})$, to match its average behaviour stored in its supervised learning memory, e.g. by optimizing the log-probability of past actions taken.

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, $Q^i(s, a) \approx \mathbb{E}_{\beta^i, \pi^{-i}}\left[G^i_t \,\middle|\, S_t = s, A_t = a\right]$, 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, β, which is needed to train its average policy network, $\Pi(s, a \mid \theta^{\Pi})$, that approximates the agent's average of past best responses. To address this problem, we suggest using an approximation of *anticipatory dynamics* of continuous-time dynamic fictitious play (Shamma and Arslan, 2005). In this variant of fictitious play players choose best responses to a short-term prediction of their opponents' average normal-form strategies, $\hat{\pi}^{-i}_t + \eta\frac{d}{dt}\hat{\pi}^{-i}_t$, where $\eta \in \mathbb{R}$ is termed the **anticipatory parameter**. The authors show that for appropriate, game-dependent choice of η stability of fictitious play at equilibrium points can be improved. NFSP uses $\hat{\beta}^i_{t+1} - \hat{\pi}^i_t \approx \frac{d}{dt}\hat{\pi}^i_t$ as a discrete-time approximation of the derivative that is used in these anticipatory dynamics. Note that $\Delta\hat{\pi}^i_t \propto \hat{\beta}^i_{t+1} - \hat{\pi}^i_t$ is the normal-form update direction of common discrete-time fictitious play.

NFSP agents choose their actions from the mixture policy $\sigma \equiv (1-\eta)\hat{\pi} + \eta\hat{\beta}$. This enables each agent to compute an approximate best response, β^i, to its opponents' anticipated average strategy profile, $\sigma^{-i} \equiv \hat{\pi}^{-i} + \eta(\hat{\beta}^{-i} - \hat{\pi}^{-i})$, by iteratively evaluating and maximizing its action values, $Q^i(s,a) \approx \mathbb{E}_{\beta^i,\sigma^{-i}}\left[G_t^i \,\middle|\, S_t = s, A_t = a\right]$. 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.

4 Experiments

We evaluate NFSP and related algorithms in Leduc and Limit Texas Hold'em poker games. Most of our experiments measure the exploitability of learned strategy profiles. In a two-player zero-sum game, the exploitability of a strategy profile is defined as the expected average payoff that a best response profile achieves against it. An exploitability of 2δ yields at least a δ-Nash equilibrium.

4.1 Leduc Hold'em

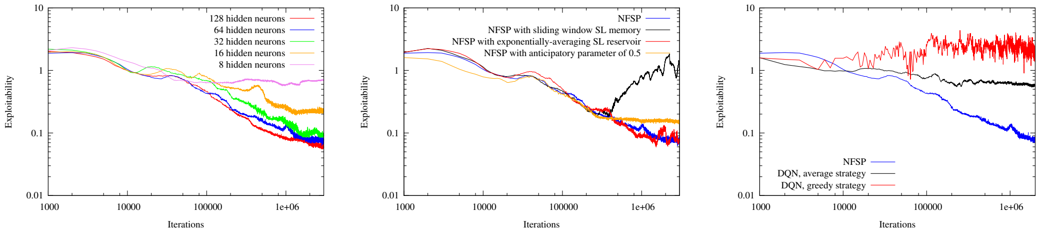
We empirically investigate the convergence of NFSP to Nash equilibria in Leduc Hold'em. We also study whether removing or altering some of NFSP's components breaks convergence.

One of our goals is to minimize reliance on prior knowledge. Therefore, we attempt to define a domain-independent encoding of information states in poker games. Contrary to other work on computer poker (Zinkevich et al., 2007; Gilpin et al., 2007; Johanson et al., 2013), we do not engineer any higher-level features. Poker games usually consist of multiple rounds. At each round new cards are revealed to the players. We represent each rounds' cards by a k-of-n encoding. E.g. LHE has a card deck of 52 cards and on the second round three cards are revealed. Thus, this round is encoded with a vector of length 52 and three elements set to 1 and the rest to 0. In Limit Hold'em poker games, players usually have three actions to choose from, namely {fold, call, raise}. Note that depending on context, calls and raises can be referred to as checks and bets respectively. Betting is capped at a fixed number of raises per round. Thus, we can represent the betting history as a tensor with 4 dimensions, namely {player, round, number of raises, action taken}. E.g. heads-up LHE contains 2 players, 4 rounds, 0 to 4 raises per round and 3 actions. Thus we can represent a LHE betting history as a $2 \times 4 \times 5 \times 3$ tensor. In a heads-up game we do not need to encode the fold action, as a two-player game always ends if one player gives up. Thus, we can flatten the 4-dimensional tensor to a vector of length 80. Concatenating with the card inputs of 4 rounds, we encode an information state of LHE as a vector of length 288. Similarly, an information state of Leduc Hold'em can be encoded as a vector of length 30, as it contains 6 cards with 3 duplicates, 2 rounds, 0 to 2 raises per round and 3 actions.

For learning in Leduc Hold'em, we manually calibrated NFSP for a fully connected neural network with 1 hidden layer of 64 neurons and rectified linear activations. We then repeated the experiment for various network architectures with the same parameters. In particular, we set the sizes of memories to 200k and 2m for $\mathcal{M}_{RL}$ and $\mathcal{M}_{SL}$ respectively. $\mathcal{M}_{RL}$ functioned as a circular buffer containing a recent window of experience. $\mathcal{M}_{SL}$ was updated with reservoir sampling (Vitter, 1985). The reinforcement and supervised learning rates were set to 0.1 and 0.005, and both used vanilla Stochastic Gradient Descent (SGD) without momentum for stochastic optimization of the neural networks. Each agent performed 2 stochastic gradient updates of mini-batch size 128 per network for every 128 steps in the game. The target network of the DQN algorithm was refitted every 300 updates. NFSP's anticipatory parameter was set to $\eta = 0.1$. The ϵ-greedy policies' exploration started at 0.06 and decayed to 0, proportionally to the inverse square root of the number of iterations.

Figure 1a shows NFSP approaching Nash equilibria for various network architectures. We observe a monotonic performance increase with size of the networks. NFSP achieved an exploitability of 0.06, which full-width XFP typically achieves after around 1000 full-width iterations.

In order to investigate the relevance of various components of NFSP, e.g. reservoir sampling and anticipatory dynamics, we conducted an experiment that isolated their effects. Figure 1b shows that these modifications led to decremental performance. In particular, using a fixed-size sliding window to store experience of the agents' own behaviour led to divergence. NFSP's performance plateaued for a high anticipatory parameter of 0.5, that most likely violates the game-dependent stability conditions of dynamic fictitious play (Shamma and Arslan, 2005). Finally, using exponentially-averaged reservoir sampling for supervised learning memory updates led to noisy performance.



(a) Varying network sizes (b) Removed essential components (c) Comparison to DQN

Figure 1: Learning performance of NFSP in Leduc Hold'em.

4.2 Comparison to DQN

Several stable algorithms have previously been proposed for deep reinforcement learning, notably the DQN algorithm (Mnih et al., 2015). However, the empirical stability of these algorithms was only previously established in single-agent, perfect (or near-perfect) information MDPs. Here, we investigate the stability of DQN in multi-agent, imperfect-information games, in comparison to NFSP.

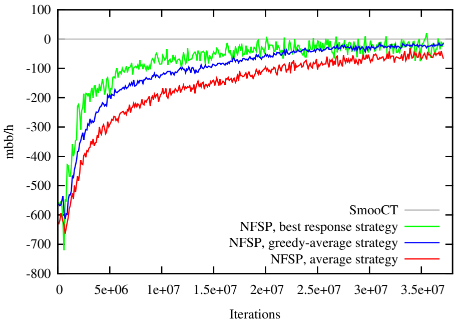
DQN learns a deterministic, greedy strategy. Such strategies are sufficient to behave optimally in single-agent domains, i.e. MDPs for which DQN was designed. However, imperfect-information games generally require stochastic strategies to achieve optimal behaviour. One might wonder if the average behaviour of DQN converges to a Nash equilibrium. To test this, we augmented DQN with a supervised learning memory and train a neural network to estimate its average strategy. Unlike NFSP, the average strategy does not affect the agent's behaviour in any way; it is passively observing the DQN agent to estimate its evolution over time. We implement this variant of DQN by using NFSP with an anticipatory parameter of $\eta = 1$. We trained DQN with all combinations of the following parameters: Learning rate $\{0.2, 0.1, 0.05\}$, decaying exploration starting at $\{0.06, 0.12\}$ and reinforcement learning memory $\{\text{2m reservoir}, \text{2m sliding window}\}$. Other parameters were fixed to the same values as NFSP; note that these parameters only affect the passive observation process. We then chose the best-performing result of DQN and compared to NFSP's performance that was achieved in the previous section's experiment. DQN achieved its best-performing result with a learning rate of 0.1, exploration starting at 0.12 and a sliding window memory of size 2m.

Figure 1c shows that DQN's deterministic strategy is highly exploitable, which is expected as imperfect-information games usually require stochastic policies. DQN's average behaviour does not approach a Nash equilibrium either. In principle, DQN also learns a best-response to the historical experience generated by fellow agents. So why does it perform worse than NFSP? The problem is that DQN agents exclusively generate self-play experience according to their ϵ-greedy strategies. These experiences are both highly correlated over time, and highly focused on a narrow distribution of states. In contrast, NFSP agents use an ever more slowly changing (anticipated) average policy to generate self-play experience. Thus, their experience varies more smoothly, resulting in a more stable data distribution, and therefore more stable neural networks. Note that this issue is not limited to DQN; other common reinforcement learning methods have been shown to exhibit similarly stagnating performance in poker games (Heinrich and Silver, 2015).

4.3 Limit Texas Hold'em

We applied NFSP to LHE, a game that is popular with humans. Since in 2008 a computer program beat expert human LHE players for the first time in a public competition, modern computer agents are widely considered to have achieved superhuman performance (Newall, 2013). The game was *essentially solved* by Bowling et al. (2015). We evaluated our agents against the top 3 computer programs of the most recent (2014) Annual Computer Poker Competition (ACPC) that featured LHE. Learning performance was measured in milli-big-blinds won per hand, mbb/h, i.e. one thousandth of a big blind that players post at the beginning of a hand.

We manually calibrated NFSP by trying 9 configurations. We achieved the best performance with the following parameters. The neural networks were fully connected with four hidden layers of $1024, 512, 1024$ and 512 neurons with rectified linear activations. The memory sizes were set to



(a) Win rates against SmooCT. The estimated standard error of each evaluation is less than 10 mbb/h.

Match-up	Win rate (mbb/h)
escabeche	-52.1 ± 8.5
SmooCT	-17.4 ± 9.0
Hyperborean	-13.6 ± 9.2

(b) Win rates of NFSP's greedy-average strategy against the top 3 agents of the ACPC 2014.

Figure 2: Performance of NFSP in Limit Texas Hold'em.

600k and 30m for $\mathcal{M}_{RL}$ and $\mathcal{M}_{SL}$ respectively. $\mathcal{M}_{RL}$ functioned as a circular buffer containing a recent window of experience. $\mathcal{M}_{SL}$ was updated with exponentially-averaged reservoir sampling (Osborne et al., 2014), replacing entries in $\mathcal{M}_{SL}$ with minimum probability 0.25. We used vanilla SGD without momentum for both reinforcement and supervised learning, with learning rates set to 0.1 and 0.01 respectively. Each agent performed 2 stochastic gradient updates of mini-batch size 256 per network for every 256 steps in the game. The target network was refitted every 1000 updates. NFSP's anticipatory parameter was set to $\eta = 0.1$. The ϵ-greedy policies' exploration started at 0.08 and decayed to 0, more slowly than in Leduc Hold'em. In addition to NFSP's main, average strategy profile we also evaluated the best response and greedy-average strategies, which deterministically choose actions that maximize the predicted action values or probabilities respectively.

To provide some intuition for win rates in heads-up LHE, a player that always folds will lose 750 mbb/h, and expert human players typically achieve expected win rates of 40-60 mbb/h at online high-stakes games. Similarly, the top half of computer agents in the ACPC 2014 achieved up to 50 mbb/h between themselves. While training, we periodically evaluated NFSP's performance against SmooCT from symmetric play for 25000 hands each. Figure 2a presents the learning performance of NFSP. NFSP's average and greedy-average strategy profiles exhibit a stable and relatively monotonic performance improvement, and achieve win rates of around -50 and -20 mbb/h respectively. The best response strategy profile exhibited more noisy performance, mostly ranging between -50 and 0 mbb/h. We also evaluated the final greedy-average strategy against the other top 3 competitors of the ACPC 2014. Table 2b presents the results. NFSP achieves winrates similar to those of the top half of computer agents in the ACPC 2014 and thus is competitive with superhuman computer poker programs.

4.4 Approximations to fictitious play

NFSP approximates fictitious play, both by using a neural network function approximator to represent strategies, and by averaging those strategies via gradient descent machine learning. In the appendix we provide experiments that illustrate the effect of these two approximations, compared to exact representations of strategies and perfect averaging procedures.

5 Related work

Reliance on human expert knowledge can be expensive, prone to human biases and limiting if such knowledge is suboptimal. Yet many methods that have been applied to games have relied on human expert knowledge. Deep Blue used a human-engineered evaluation function for chess (Campbell et al., 2002). In computer Go, Maddison et al. (2015) and Clark and Storkey (2015) trained deep neural networks from data of expert human play. In computer poker, current game-theoretic approaches use heuristics of card strength to abstract the game to a tractable size (Zinkevich et al., 2007; Gilpin et al., 2007; Johanson et al., 2013). Waugh et al. (2015) recently combined one of these methods with function approximation. However, their full-width algorithm has to implicitly reason about all information states at each iteration, which is prohibitively expensive in large domains. In contrast, NFSP focuses on the sample-based reinforcement learning setting where the game's states need not be exhaustively enumerated and the learner may not even have a model of the game's dynamics.

Nash equilibria are the only strategy profiles that rational agents can hope to converge on in self-play (Bowling and Veloso, 2001). TD-Gammon (Tesauro, 1995) is a world-class backgammon agent, whose main component is a neural network trained from self-play reinforcement learning. While its algorithm, based on temporal-difference learning, is sound in two-player zero-sum perfect-information games, it does not generally converge in games with imperfect information. DQN (Mnih et al., 2015) combines temporal-difference learning with experience replay and deep neural network function approximation. It achieved human-level performance in a majority of Atari games, learning from raw sensory inputs. However, these Atari games were set up as single-agent domains with potential opponents fixed and controlled by the Atari emulator. Our experiments showed that DQN agents were unable to approach a Nash equilibrium in Leduc Hold'em, where players were allowed to adapt dynamically. Yakovenko et al. (2016) trained deep neural networks in self-play in computer poker, including two poker games that are popular with humans. Their networks performed strongly against heuristic-based and simple computer programs. Expert human players were able to outperform their agent, albeit over a statistically insignificant sample size. It remains to be seen whether their approach converges in practice or theory.

In this work, we focused on imperfect-information two-player zero-sum games. Fictitious play, however, is also guaranteed to converge to Nash equilibria in cooperative, potential games (Monderer and Shapley, 1996). It is therefore conceivable that NFSP can be successfully applied to these games as well. Furthermore, recent developments in continuous-action reinforcement learning (Lillicrap et al., 2015) could enable NFSP to be applied to continuous-action games, which current game-theoretic methods cannot deal with directly.

6 Conclusion

We have introduced NFSP, the first end-to-end deep reinforcement learning approach to learning approximate Nash equilibria of imperfect-information games from self-play. Unlike previous game-theoretic methods, NFSP is scalable without prior domain knowledge. Furthermore, NFSP is the first deep reinforcement learning method known to converge to approximate Nash equilibria in self-play. Our experiments have shown NFSP to converge reliably to approximate Nash equilibria in a small poker game, whereas DQN's greedy and average strategies did not. NFSP learned a strategy that is competitive with superhuman programs in a real-world scale imperfect-information game from scratch without using explicit prior knowledge.

Acknowledgments

We thank Peter Dayan, Marc Lanctot and Marc Bellemare for helpful discussions and feedback. This research was supported by the UK Centre for Doctoral Training in Financial Computing and by the NVIDIA Corporation.

References

Bazzan, A. L. (2009). Opportunities for multiagent systems and multiagent reinforcement learning in traffic control. *Autonomous Agents and Multi-Agent Systems*, 18(3):342–375.

Bowling, M., Burch, N., Johanson, M., and Tammelin, O. (2015). Heads-up limit hold'em poker is solved. *Science*, 347(6218):145–149.

Bowling, M. and Veloso, M. (2001). Rational and convergent learning in stochastic games. In *Proceedings of the 17th International Joint Conference on Artifical Intelligence*, volume 17, pages 1021–1026.

Brown, G. W. (1951). Iterative solution of games by fictitious play. *Activity analysis of production and allocation*.

Campbell, M., Hoane, A. J., and Hsu, F.-h. (2002). Deep blue. *Artificial intelligence*, 134(1):57–83.

Clark, C. and Storkey, A. (2015). Training deep convolutional neural networks to play go. In *Proceedings of the 32nd International Conference on Machine Learning*, pages 1766–1774.

Ernst, D., Geurts, P., and Wehenkel, L. (2005). Tree-based batch mode reinforcement learning. In *Journal of Machine Learning Research*, pages 503–556.

Fudenberg, D. (1998). *The theory of learning in games*, volume 2. MIT press.

Gelly, S., Kocsis, L., Schoenauer, M., Sebag, M., Silver, D., Szepesvári, C., and Teytaud, O. (2012). The grand challenge of computer go: Monte Carlo tree search and extensions. *Communications of the ACM*.

Gilpin, A., Hoda, S., Pena, J., and Sandholm, T. (2007). Gradient-based algorithms for finding Nash equilibria in extensive form games. In *Internet and Network Economics*, pages 57–69. Springer.

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

Heinrich, J. and Silver, D. (2015). Smooth UCT search in computer poker. In *Proceedings of the 24th International Joint Conference on Artifical Intelligence*.

Johanson, M., Burch, N., Valenzano, R., and Bowling, M. (2013). Evaluating state-space abstractions in extensive-form games. In *Proceedings of the 12th International Conference on Autonomous Agents and Multi-Agent Systems*, pages 271–278.

Lambert III, T. J., Epelman, M. A., and Smith, R. L. (2005). A fictitious play approach to large-scale optimization. *Operations Research*, 53(3):477–489.

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

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. (2015). Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*.

Lin, L.-J. (1992). Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine learning*, 8(3-4):293–321.

Maddison, C. J., Huang, A., Sutskever, I., and Silver, D. (2015). Move evaluation in go using deep convolutional neural networks. *The International Conference on Learning Representations*.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. (2015). Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533.

Monderer, D. and Shapley, L. S. (1996). Fictitious play property for games with identical interests. *Journal of economic theory*, 68(1):258–265.

Myerson, R. B. (1991). *Game Theory: Analysis of Conflict*. Harvard University Press.

Nevmyvaka, Y., Feng, Y., and Kearns, M. (2006). Reinforcement learning for optimized trade execution. In *Proceedings of the 23rd International Conference on Machine Learning*, pages 673–680. ACM.

Newall, P. (2013). *Further Limit Hold 'em: Exploring the Model Poker Game*. Two Plus Two Publishing, LLC.

Osborne, M., Lall, A., and Van Durme, B. (2014). Exponential reservoir sampling for streaming language models. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*.

Riedmiller, M. (2005). Neural fitted q iteration–first experiences with a data efficient neural reinforcement learning method. In *Machine Learning: ECML 2005*, pages 317–328. Springer.

Robinson, J. (1951). An iterative method of solving a game. *Annals of Mathematics*, pages 296–301.

Samuel, A. L. (1959). Some studies in machine learning using the game of checkers. *IBM Journal of research and development*, 3(3):210–229.

Selten, R. (1990). Bounded rationality. *Journal of Institutional and Theoretical Economics*, pages 649–658.

Shamma, J. S. and Arslan, G. (2005). Dynamic fictitious play, dynamic gradient play, and distributed convergence to Nash equilibria. *IEEE Transactions on Automatic Control*, 50(3):312–327.

Sutton, R. S. and Barto, A. G. (1998). *Reinforcement learning: An introduction*, volume 1.

Tambe, M. (2011). *Security and game theory: algorithms, deployed systems, lessons learned*.

Tesauro, G. (1995). Temporal difference learning and td-gammon. *Communications of the ACM*, 38(3):58–68.

Urieli, D. and Stone, P. (2014). Tactex'13: a champion adaptive power trading agent. In *Proceedings of the 13th International Conference on Autonomous Agents and Multi-Agent Systems*, pages 1447–1448.

Vitter, J. S. (1985). Random sampling with a reservoir. *ACM Transactions on Mathematical Software*.

Von Stengel, B. (1996). Efficient computation of behavior strategies. *Games and Economic Behavior*.

Watkins, C. J. and Dayan, P. (1992). Q-learning. *Machine learning*, 8(3-4):279–292.

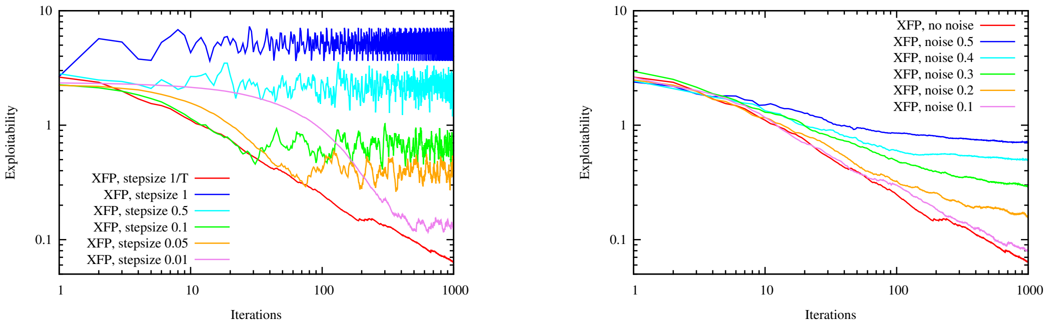
Waugh, K., Morrill, D., Bagnell, J. A., and Bowling, M. (2015). Solving games with functional regret estimation. In *29th AAAI Conference on Artificial Intelligence*.

Yakovenko, N., Cao, L., Raffel, C., and Fan, J. (2016). Poker-cnn: A pattern learning strategy for making draws and bets in poker games using convolutional networks. In *30th AAAI Conference on Artificial Intelligence*.

Zinkevich, M., Johanson, M., Bowling, M., and Piccione, C. (2007). Regret minimization in games with incomplete information. In *Advances in Neural Information Processing Systems*, pages 1729–1736.

A Robustness of XFP

To understand how function approximation interacts with FSP, we conducted some simple experiments that emulate approximation and sampling errors in the full-width algorithm XFP. Firstly, we explore what happens when the perfect averaging used in XFP is replaced by an incremental averaging process closer to gradient descent. Secondly, we explore what happens when the exact table lookup used in XFP is replaced by an approximation with epsilon error.



(a) Constant stepsizes in average policy updates

(b) Uniform-random noise in best response computation

Figure 3: Full-width XFP performance with added noise

Figure 3a shows the performance of XFP with default, $1/T$, and constant stepsizes for its strategy updates. We see improved asymptotic but lower initial performance for smaller stepsizes. For constant stepsizes the performance seems to plateau rather than diverge. With reservoir sampling we can achieve an effective stepsize of $1/T$. However, the results suggest that exponentially-averaged reservoir sampling can be a viable choice too, as exponential averaging of past memories would approximately correspond to using a constant stepsize.

XFP with stepsize 1 is equivalent to a full-width iterated best response algorithm. While this algorithm converges to a Nash equilibrium in finite perfect-information two-player zero-sum games, the results suggest that with imperfect information this is not generally the case. The Poker-CNN algorithm introduced by Yakovenko et al. (2016) stores a small number of past strategies which it iteratively computes new strategies against. Replacing strategies in that set is similar to updating an average strategy with a large stepsize. This might lead to similar problems as shown in Figure 3a.

Our NFSP agents add random exploration to their policies and use noisy stochastic gradient updates to learn action values, which determine their approximate best responses. Therefore, we investigated the impact of random noise added to the best response computation, which XFP performs by dynamic programming. At each backward induction step, we pass back a uniform-random action's value with probability ϵ and the best action's value otherwise. Figure 3b shows monotonically decreasing performance with added noise. However, performance remains stable and keeps improving for all noise levels.