Fictitious Self-Play in Extensive-Form Games

Johannes Heinrich J.HEINRICH@CS.UCL.AC.UK
University College London, UK

Marc Lanctot LANCTOT@GOOGLE.COM
Google DeepMind, London, UK

David Silver DAVIDSILVER@GOOGLE.COM
Google DeepMind, London, UK

Abstract

Fictitious play is a popular game-theoretic model of learning in games. However, it has received little attention in practical applications to large problems. This paper introduces two variants of fictitious play that are implemented in behavioural strategies of an extensive-form game. The first variant is a full-width process that is realization equivalent to its normal-form counterpart and therefore inherits its convergence guarantees. However, its computational requirements are linear in time and space rather than exponential. The second variant, Fictitious Self-Play, is a machine learning framework that implements fictitious play in a sample-based fashion. Experiments in imperfect-information poker games compare our approaches and demonstrate their convergence to approximate Nash equilibria.

1. Introduction

Fictitious play, introduced by Brown (1951), is a popular game-theoretic model of learning in games. In fictitious play, players repeatedly play a game, at each iteration choosing a best response to their opponents' average strategies. The average strategy profile of fictitious players converges to a Nash equilibrium in certain classes of games, e.g. two-player zero-sum and potential games. Fictitious play is a standard tool of game theory and has motivated substantial discussion and research on how Nash equilibria could be realized in practice (Brown, 1951; Fudenberg, 1998; Hofbauer & Sandholm, 2002; Leslie & Collins, 2006). Furthermore, it is a classic example of self-play learning from experience that has inspired artificial intelligence algorithms in games.

Despite the popularity of fictitious play to date, it has seen use in few large-scale applications, e.g. (Lambert III et al., 2005; McMahan & Gordon, 2007; Ganzfried & Sandholm, 2009; Heinrich & Silver, 2015). One possible reason for this is its reliance on a normal-form representation. While any extensive-form game can be converted into a normal-form equivalent (Kuhn, 1953), the resulting number of actions is typically exponential in the number of game states. The extensive-form offers a much more efficient representation via behavioural strategies whose number of parameters is linear in the number of information states. Hendon et al. (1996) introduce two definitions of fictitious play in behavioural strategies and show that each convergence point of their variants is a sequential equilibrium. However, these variants are not guaranteed to converge in imperfect-information games.

The first fictitious play variant that we introduce in this paper is full-width extensive-form fictitious play (XFP). It is realization equivalent to a normal-form fictitious play and therefore inherits its convergence guarantees. However, it can be implemented using only behavioural strategies and therefore its computational complexity per iteration is linear in the number of game states rather than exponential.

XFP and many other current methods of computational game theory (Sandholm, 2010) are full-width approaches and therefore require reasoning about every state in the game at each iteration. Apart from being given state-aggregating abstractions, that are usually hand-crafted from expert knowledge, the algorithms themselves do not generalise between strategically similar states. Leslie & Collins (2006) introduce generalised weakened fictitious play which explicitly allows certain kinds of approximations in fictitious players' strategies. This motivates the use of approximate techniques like machine learning which ex-

Proceedings of the 32^{nd} International Conference on Machine Learning, Lille, France, 2015. JMLR: W&CP volume 37. Copyright 2015 by the author(s).

cel at learning and generalising from finite data.

The second variant that we introduce is Fictitious Self-Play (FSP), a machine learning framework that implements generalised weakened fictitious play in behavioural strategies and in a sample-based fashion. In FSP players repeatedly play a game and store their experience in memory. Instead of playing a best response, they act cautiously and mix between their best responses and average strategies. At each iteration players replay their experience of play against their opponents to compute an approximate best response. Similarly, they replay their experience of their own behaviour to learn a model of their average strategy. In more technical terms, FSP iteratively samples episodes of the game from self-play. These episodes constitute data sets that are used by reinforcement learning to compute approximate best responses and by supervised learning to compute perturbed models of average strategies.

1.1. Related work

Efficiently computing Nash equilibria of imperfect-information games has received substantial attention by researchers in computational game theory and artificial intelligence (Sandholm, 2010; Bowling et al., 2015). The most popular modern techniques are either optimization-based (Koller et al., 1996; Gilpin et al., 2007; Miltersen & Sørensen, 2010; Bosansky et al., 2014) or perform regret minimization (Zinkevich et al., 2007). Counterfactual regret minimization (CFR) is the first approach which essentially solved an imperfect-information game of real-world scale (Bowling et al., 2015). Being a self-play approach that uses regret minimization, it has some similarities to the utility-maximizing self-play approaches introduced in this paper.

Similar to full-width CFR, our full-width method's worst-case computational complexity per iteration is linear in the number of game states and it is well-suited for parallelization and distributed computing. Furthermore, given a long-standing conjecture (Karlin, 1959; Daskalakis & Pan, 2014) the convergence rate of fictitious play might be $\mathcal{O}(n^{-\frac{1}{2}})$, which is of the same order as CFR's.

Similar to Monte Carlo CFR (Lanctot et al., 2009), FSP uses sampling to focus learning and computation on selectively sampled trajectories and thus breaks the curse of dimensionality. However, FSP only requires a black box simulator of the game. In particular, agents do not require any explicit knowledge about their opponents or even the game itself, other than what they experience in actual play. A similar property has been suggested possible for a form of outcome-sampling MCCFR, but remains unexplored.

2. Background

In this section we provide a brief overview over common game-theoretic representations of a game, fictitious play and reinforcement learning. For a more detailed exposition we refer the reader to (Myerson, 1991), (Fudenberg, 1998) and (Sutton & Barto, 1998).

2.1. Extensive-Form

Extensive-form games are a model of sequential interaction involving multiple agents. The representation is based on a game tree and consists of the following components: $\mathcal{N} = \{1, ..., n\}$ denotes the set of players. $\mathcal{S}$ is a set of **states** corresponding to nodes in a finite rooted game tree. For each state node $s \in \mathcal{S}$ the edges to its successor states define a set of **actions** $\mathcal{A}(s)$ available to a player or chance in state s. The player function $P : \mathcal{S} \to \mathcal{N} \cup \{c\}$, with c denoting chance, determines who is to act at a given state. Chance is considered to be a particular player that follows a fixed randomized strategy that determines the distribution of chance events at chance nodes. For each player i there is a corresponding set of **information states** $\mathcal{U}^i$ and an information function $I^i : \mathcal{S} \to \mathcal{U}^i$ that determines which states are indistinguishable for the player by mapping them on the same information state $u \in \mathcal{U}^i$. Throughout this paper we assume games with **perfect recall**, i.e. each player's current information state u^i_k implies knowledge of the sequence of his information states and actions, $u^i_1, a^i_1, u^i_2, a^i_2, ..., u^i_k$, that led to this information state. Finally, $R : \mathcal{S} \to \mathbb{R}^n$ maps terminal states to a vector whose components correspond to each player's payoff.

A player's **behavioural strategy**, $\pi^i(u) \in \Delta\left(\mathcal{A}(u)\right), \forall u \in \mathcal{U}^i$, determines a probability distribution over actions given an information state, and Δ^i_b is the set of all behavioural strategies of player 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. Based on the game's payoff function R, $R^i(\pi)$ is the expected payoff of player i if all players follow the strategy profile π. The set of **best responses** of player i to their opponents' strategies π^{-i} is $b^i(\pi^{-i}) = \arg\max_{\pi^i \in \Delta^i_b} R^i(\pi^i, \pi^{-i})$. For $\epsilon > 0$, $b^i_\epsilon(\pi^{-i}) = \{\pi^i \in \Delta^i_b : R^i(\pi^i, \pi^{-i}) \geq R^i(b^i(\pi^{-i}), \pi^{-i}) - \epsilon\}$ defines the set of **ϵ-best responses** to the strategy profile π^{-i}. A **Nash equilibrium** of an extensive-form game is a strategy profile π such that $\pi^i \in b^i(\pi^{-i})$ for all $i \in \mathcal{N}$. An **ϵ-Nash equilibrium** is a strategy profile π such that $\pi^i \in b^i_\epsilon(\pi^{-i})$ for all $i \in \mathcal{N}$.

2.2. Normal-Form

An extensive-form game induces an equivalent **normal-form game** as follows. For each player $i \in \mathcal{N}$ their deterministic strategies, $\Delta^i_p \subset \Delta^i_b$, define a set of normal-form actions, called **pure strategies**. Restricting the extensive-form payoff function R to pure strategy profiles yields a payoff function in the normal-form game.

Each pure strategy can be interpreted as a full game plan that specifies deterministic actions for all situations that a player might encounter. Before playing an iteration of the game each player chooses one of their available plans and commits to it for the iteration. A **mixed strategy** Π^i for player i is a probability distribution over their pure strategies. Let Δ^i denote the set of all mixed strategies available to player i. A mixed strategy profile $\Pi \in \times_{i \in \mathcal{N}} \Delta^i$ specifies a mixed strategy for each player. Finally, $R^i : \times_{i \in \mathcal{N}} \Delta^i \to \mathbb{R}$ determines the expected payoff of player i given a mixed strategy profile.

Throughout this paper, we use small Greek letters for behavioural strategies of the extensive-form and large Greek letters for pure and mixed strategies of a game's normal-form.

2.3. Realization-equivalence

The sequence-form (Koller et al., 1994; Von Stengel, 1996) of a game decomposes players' strategies into sequences of actions and probabilities of realizing these sequences. These realization probabilities provide a link between behavioural and mixed strategies.

For any player $i \in \mathcal{N}$ of a perfect-recall extensive-form game, each of their information states $u^i \in \mathcal{U}^i$ uniquely defines a **sequence** σ_{u^i} of actions that the player is required to take in order to reach information state u^i. Let $\Sigma^i = \{\sigma_u : u \in \mathcal{U}^i\}$ denote the set of such sequences of player i. Furthermore, let $\sigma_u a$ denote the sequence that extends σ_u with action a.

Definition 1. A **realization plan** of player $i \in \mathcal{N}$ is a function, $x : \Sigma^i \to [0, 1]$, such that $x(\emptyset) = 1$ and $\forall \sigma_u \in \mathcal{U}^i$: $x(\sigma_u) = \sum_{a \in \mathcal{A}(u)} x(\sigma_u a)$.

A behavioural strategy π induces a realization plan $x_\pi(\sigma_u) = \prod_{(u',a) \in \sigma_u} \pi(u', a)$, where the notation (u', a) disambiguates actions taken at different information states. Similarly, a realization plan induces a behavioural strategy, $\pi(u, a) = \frac{x(\sigma_u a)}{x(\sigma_u)}$, where π is defined arbitrarily at information states that are never visited, i.e. when $x(\sigma_u) = 0$. As a pure strategy is just a deterministic behavioural strategy, it has a realization plan with binary values. As a mixed strategy is a convex combination of pure strategies, $\Pi = \sum_i w_i \Pi_i$, its realization plan is a similarly weighted convex combination of the pure strategies' realization plans, $x_\Pi = \sum_i w_i x_{\Pi_i}$.

The following definition and theorems connect an extensive-form game's behavioural strategies with mixed strategies of the equivalent normal-form representation.

Definition 2. Two strategies π_1 and π_2 of a player are **realization-equivalent** if for any fixed strategy profile of the other players both strategies, π_1 and π_2, define the same probability distribution over the states of the game.

Theorem 3 (compare also (Von Stengel, 1996))**.** *Two strategies are realization-equivalent if and only if they have the same realization plan.*

Theorem 4 (Kuhn's Theorem (Kuhn, 1953))**.** *For a player with perfect recall, any mixed strategy is realization-equivalent to a behavioural strategy, and vice versa.*

2.4. Fictitious Play

In this work we use a general version of fictitious play that is due to Leslie & Collins (2006) and based on the work of Benaïm et al. (2005). It has similar convergence guarantees as common fictitious play, but allows for approximate best responses and perturbed average strategy updates.

Definition 5. A generalised weakened **fictitious play** is a process of mixed strategies, $\{\Pi_t\}$, $\Pi_t \in \times_{i \in \mathcal{N}} \Delta^i$, s.t.

$$\Pi^i_{t+1} \in (1 - \alpha_{t+1}) \Pi^i_t + \alpha_{t+1} (b^i_{\epsilon_t}(\Pi^{-i}_t) + M^i_{t+1}), \ \forall i \in \mathcal{N},$$

with $\alpha_t \to 0$ and $\epsilon_t \to 0$ as $t \to \infty$, $\sum_{t=1}^\infty \alpha_t = \infty$, and $\{M_t\}$ a sequence of perturbations that satisfies $\forall T > 0$

$$\lim_{t \to \infty} \sup_k \left\{ \left\| \sum_{i=t}^{k-1} \alpha_{i+1} M_{i+1} \right\| \text{ s.t.} \sum_{i=t}^{k-1} \alpha_{i+1} \leq T \right\} = 0.$$

Original fictitious play (Brown, 1951; Robinson, 1951) is a generalised weakened fictitious play with stepsize $\alpha_t = \frac{1}{t}$, $\epsilon_t = 0$ and $M_t = 0 \ \forall t$. Generalised weakened fictitious play converges in certain classes of games that are said to have the fictitious play property (Leslie & Collins, 2006), e.g. two-player zero-sum and potential games.

2.5. Reinforcement Learning

Reinforcement learning (Sutton & Barto, 1998) agents typically learn to maximize their expected future reward from interaction with an environment. The environment is usually modelled as a **Markov decision process (MDP)**. A MDP consists of a set of Markov states $\mathcal{S}$, a set of actions $\mathcal{A}$, a transition function $\mathcal{P}^a_{ss'}$ and a reward function $\mathcal{R}^a_s$. The transition function determines the probability of transitioning to state s' after taking action a in state s. The reward function $\mathcal{R}^a_s$ determines an agent's reward after taking action a in state s. An agent behaves according to a **policy** that specifies a distribution over actions at each state.

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. An agent is learning **on-policy** if it gathers these transition tuples by following its own policy. In the **off-policy** setting an agent is learning from experience of another agent or another policy.

Q-learning (Watkins & Dayan, 1992) is a popular off-policy reinforcement learning method that can be used to learn an optimal policy of a MDP. Fitted Q Iteration (FQI) (Ernst et al., 2005) is a batch reinforcement learning method that applies Q-learning to a data set of transition tuples from a MDP.

3. Extensive-Form Fictitious Play

In this section, we derive a process in behavioural strategies that is realization equivalent to normal-form fictitious play.

The following lemma shows how a mixture of normal-form strategies can be implemented by a weighted combination of their realization equivalent behavioural strategies.

Lemma 6. *Let π and β be two behavioural strategies, Π and B two mixed strategies that are realization equivalent to π and β, and $\lambda_1, \lambda_2 \in \mathbb{R}_{\geq 0}$ with $\lambda_1 + \lambda_2 = 1$. Then for each information state $u \in \mathcal{U}$,*

$$\mu(u) = \pi(u) + \frac{\lambda_2 x_\beta(\sigma_u)}{\lambda_1 x_\pi(\sigma_u) + \lambda_2 x_\beta(\sigma_u)} (\beta(u) - \pi(u))$$

defines a behavioural strategy μ at u and μ is realization equivalent to the mixed strategy $M = \lambda_1 \Pi + \lambda_2 B$.

Theorem 7 presents a fictitious play in behavioural strategies that inherits the convergence results of generalised weakened fictitious play by realization-equivalence.

Theorem 7. *Let π_1 be an initial behavioural strategy profile. The extensive-form process*

$$\beta^i_{t+1} \in b^i_{\epsilon_{t+1}}(\pi^{-i}_t),$$

$$\pi^i_{t+1}(u) = \pi^i_t(u) + \frac{\alpha_{t+1} x_{\beta^i_{t+1}}(\sigma_u) \left(\beta^i_{t+1}(u) - \pi^i_t(u)\right)}{(1-\alpha_{t+1}) x_{\pi^i_t}(\sigma_u) + \alpha_{t+1} x_{\beta^i_{t+1}}(\sigma_u)}$$

for all players $i \in \mathcal{N}$ and all their information states $u \in \mathcal{U}^i$, with $\alpha_t \to 0$ and $\epsilon_t \to 0$ as $t \to \infty$, and $\sum_{t=1}^{\infty} \alpha_t = \infty$, is realization-equivalent to a generalised weakened fictitious play in the normal-form and therefore the average strategy profile converges to a Nash equilibrium in all games with the fictitious play property.

Algorithm 1 implements XFP, the extensive-form fictitious play of Theorem 7. The initial average strategy profile, π_1, can be defined arbitrarily, e.g. uniform random. At each iteration the algorithm performs two operations. First it computes a best response profile to the current average strategies. Secondly it uses the best response profile to update the average strategy profile. The first operation's computational requirements are linear in the number of game states. For each player the second operation can be performed independently from their opponents and requires work linear in the player's number of information states. Furthermore, if a deterministic best response is used, the realization weights of Theorem 7 allow ignoring all but one subtree at each of the player's decision nodes.

Algorithm 1 Full-width extensive-form fictitious play

```
function FictitiousPlay($\Gamma$)
    Initialize $\pi_1$ arbitrarily
    $j \leftarrow 1$
    while within computational budget do
        $\beta_{j+1} \leftarrow$ ComputeBRs($\pi_j$)
        $\pi_{j+1} \leftarrow$ UpdateAvgStrategies($\pi_j, \beta_{j+1}$)
        $j \leftarrow j + 1$
    end while
    return $\pi_j$
end function

function ComputeBRs($\pi$)
    Recursively parse the game's state tree to compute a
    best response strategy profile, $\beta \in b(\pi)$.
    return $\beta$
end function

function UpdateAvgStrategies($\pi_j, \beta_{j+1}$)
    Compute an updated strategy profile $\pi_{j+1}$ according
    to Theorem 7.
    return $\pi_{j+1}$
end function
```

4. Fictitious Self-Play

FSP is a machine learning framework that implements generalised weakened fictitious play in a sample-based fashion and in behavioural strategies. XFP suffers from the curse of dimensionality. At each iteration, computation needs to be performed at all states of the game irrespective of their relevance. However, generalised weakened fictitious play only requires approximate best responses and even allows some perturbations in the updates.

FSP replaces the two fictitious play operations, best response computation and average strategy updating, with machine learning algorithms. Approximate best responses are learned by reinforcement learning from play against the opponents' average strategies. The average strategy updates can be formulated as a supervised learning task, where each player learns a transition model of their own behaviour. We introduce reinforcement learning-based best response computation in section 4.1 and present supervised learning-based strategy updates in section 4.2.

4.1. Reinforcement Learning

Consider an extensive-form game and some strategy profile π. Then for each player $i \in \mathcal{N}$ the strategy profile of their opponents, π^{-i}, defines an MDP, $\mathcal{M}(\pi^{-i})$ (Silver & Veness, 2010; Greenwald et al., 2013). Player i's information states define the states of the MDP. The MDP's dynamics are given by the rules of the extensive-form game, the chance function and the opponents' fixed strategy profile. The rewards are given by the game's payoff function. An ϵ-optimal policy of the MDP, $\mathcal{M}(\pi^{-i})$, therefore yields an ϵ-best response of player i to the strategy profile π^{-i}. Thus the iterative computation of approximate best responses can be formulated as a sequence of MDPs to solve approximately, e.g. by applying reinforcement learning to samples of experience from the respective MDPs. In particular, to approximately solve the MDP $\mathcal{M}(\pi^{-i})$ we sample player i's experience from their opponents' strategy profile π^{-i}. Player i's strategy should ensure sufficient exploration of the MDP but can otherwise be arbitrary if an off-policy reinforcement learning method is used, e.g. Q-learning (Watkins & Dayan, 1992).

While generalised weakened fictitious play allows ϵ_k-best responses at iteration k, it requires that the deficit ϵ_k vanishes asymptotically, i.e. $\epsilon_k \to 0$ as $k \to \infty$. Learning such a valid sequence of ϵ_k-optimal policies of a sequence of MDPs would be hard if these MDPs were unrelated and knowledge could not be transferred. However, in fictitious play the MDP sequence has a particular structure. The average strategy profile at iteration k is realization-equivalent to a linear combination of two mixed strategies, $\Pi_k = (1-\alpha_k)\Pi_{k-1} + \alpha_k B_k$. Thus, in a two-player game, the MDP $\mathcal{M}(\pi_k^{-i})$ is structurally equivalent to an MDP that initially picks between $\mathcal{M}(\pi_{k-1}^{-i})$ and $\mathcal{M}(\beta_k^{-i})$ with probability $(1-\alpha_k)$ and α_k respectively. Due to this similarity between subsequent MDPs it is possible to transfer knowledge. The following corollary bounds the increase of the optimality deficit when transferring an approximate solution between subsequent MDPs in a fictitious play process.

Corollary 8. *Let Γ be a two-player zero-sum extensive-form game with maximum payoff range $\bar{R} = \max_{\pi \in \Delta} R^1(\pi) - \min_{\pi \in \Delta} R^1(\pi)$. Consider a fictitious play process in this game. Let Π_k be the average strategy profile at iteration k, B_{k+1} a profile of ϵ_{k+1}-best responses to Π_k, and $\Pi_{k+1} = (1-\alpha_{k+1})\Pi_k + \alpha_{k+1}B_{k+1}$ the usual fictitious play update for some stepsize $\alpha_{k+1} \in (0,1)$. Then for each player i, B_{k+1}^i is an $[\epsilon_k + \alpha_{k+1}(\bar{R} - \epsilon_k)]$-best response to Π_{k+1}.*

This bounds the absolute amount by which reinforcement learning needs to improve the best response profile to achieve a monotonic decay of the optimality gap ϵ_k. However, ϵ_k only needs to decay asymptotically. Given $\alpha_k \to 0$ as $k \to \infty$, the bound suggests that in practice a finite amount of learning per iteration might be sufficient to achieve asymptotic improvement of best responses.

In this work we use FQI to learn from data sets of sampled experience. At each iteration k, FSP samples episodes of the game from self-play. Each agent i adds its experience to its replay memory, $\mathcal{M}_{RL}^i$. The data is stored in the form of episodes of transition tuples, $(u_t, a_t, r_{t+1}, u_{t+1})$. Each episode, $\mathcal{E} = \{(u_t, a_t, r_{t+1}, u_{t+1})\}_{0 \le t \le T}$, $T \in \mathbb{N}$, contains a finite number of transitions. We use a finite memory of fixed size. If the memory is full, new episodes replace existing episodes in a first-in-first-out order. Using a finite memory and updating it incrementally can bias the underlying distribution that the memory approximates. We want to achieve a memory composition that approximates the distribution of play against the opponents' average strategy profile. This can be achieved by using a self-play strategy profile that properly mixes between the agents' average and best response strategy profiles.

4.2. Supervised Learning

Consider the point of view of a particular player i who wants to learn a behavioural strategy π that is realization-equivalent to a convex combination of their own normal-form strategies, $\Pi = \sum_{k=1}^n w_k B_k$, $\sum_{k=1}^n w_k = 1$. This task is equivalent to learning a model of the player's behaviour when it is sampled from Π. Lemma 6 describes the behavioural strategy π explicitly, while in a sample-based setting we use samples from the realization-equivalent strategy Π to learn an approximation of π. Recall that we can sample from Π by sampling from each constituent B_k with probability w_k and if B_k itself is a mixed strategy then it is a probability distribution over pure strategies.

Corollary 9. *Let $\{B_k\}_{1 \le k \le n}$ be mixed strategies of player i, $\Pi = \sum_{k=1}^n w_k B_k$, $\sum_{k=1}^n w_k = 1$ a convex combination of these mixed strategies and μ^{-i} a completely mixed sampling strategy profile that defines the behaviour of player i's opponents. Then for each information state $u \in \mathcal{U}^i$ the probability distribution of player i's behaviour at u induced by sampling from the strategy profile (Π, μ^{-i}) defines a behavioural strategy π at u and π is realization-equivalent to Π.*

Hence, the behavioural strategy π can be learned approximately from a data set consisting of trajectories sampled from (Π, μ^{-i}). In fictitious play, at each iteration n we want to learn the average mixed strategy profile $\Pi_{n+1} = \frac{n}{n+1}\Pi_n + \frac{1}{n+1}B_{n+1}$. Both Π_n and B_{n+1} are available at iteration n and we can therefore apply Corollary 9 to learn for each player i an approximation of a behavioural strategy π_{n+1}^i that is realization-equivalent to Π_{n+1}^i. Let $\tilde{\pi}_{n+1}^i$ be such an approximation and $\tilde{\Pi}_{n+1}^i$ its normal-form equivalent. Then $\tilde{\pi}_{n+1}^i$ is realization-equivalent to a perturbed fictitious play update in normal-form, $\Pi_{n+1}^i + \frac{1}{n+1}M_{n+1}^i$, where $M^i_{n+1} = (n+1)(\tilde{\Pi}^i_{n+1} - \Pi^i_{n+1})$ is a normal-form perturbation resulting from the estimation error.

In this work we restrict ourselves to simple models that count the number of times an action has been taken at an information state or alternatively accumulate the respective strategies' probabilities of taking each action. These models can be incrementally updated with samples from β_k at each iteration k. A model update requires a set of sampled tuples, (u^i_t, ρ^i_t), where u^i_t is agent i's information state and ρ^i_t is the policy that the agent pursued at this state when this experience was sampled. For each tuple (u_t, ρ_t) the update accumulates each action's weight at the information state,

$$\forall a \in \mathcal{A}(u_t) : N(u_t, a) \leftarrow N(u_t, a) + \rho_t(a)$$
$$\forall a \in \mathcal{A}(u_t) : \pi(u_t, a) \leftarrow \frac{N(u_t, a)}{N(u_t)}$$

In order to constitute an unbiased approximation of an average of best responses, $\frac{1}{k}\sum_{j=1}^{k} B^i_j$, we need to accumulate the same number of sampled episodes from each B^i_j and these need to be sampled against the same fixed opponent strategy profile μ^{-i}. However, we suggest using the average strategy profile π^{-i}_k as the sampling distribution μ^{-i}. Sampling against π^{-i}_k has the benefit of focusing the updates on states that are more likely in the current strategy profile. When collecting samples incrementally, the use of a changing sampling distribution π^{-i}_k can introduce bias. However, in fictitious play π^{-i}_k is changing more slowly over time and thus this bias should decay over time.

Algorithm 2 General Fictitious Self-Play

```
function FictitiousSelfPlay($\Gamma, n, m$)
    Initialize completely mixed $\pi_1$
    $\beta_2 \leftarrow \pi_1$
    $j \leftarrow 2$
    while within computational budget do
        $\eta_j \leftarrow$ MixingParameter($j$)
        $\mathcal{D} \leftarrow$ GenerateData($\pi_{j-1}, \beta_j, n, m, \eta_j$)
        for each player $i \in \mathcal{N}$ do
            $\mathcal{M}^i_{RL} \leftarrow$ UpdateRLMemory($\mathcal{M}^i_{RL}, \mathcal{D}^i$)
            $\mathcal{M}^i_{SL} \leftarrow$ UpdateSLMemory($\mathcal{M}^i_{SL}, \mathcal{D}^i$)
            $\beta^i_{j+1} \leftarrow$ ReinforcementLearning($\mathcal{M}^i_{RL}$)
            $\pi^i_j \leftarrow$ SupervisedLearning($\mathcal{M}^i_{SL}$)
        end for
        $j \leftarrow j + 1$
    end while
    return $\pi_{j-1}$
end function

function GenerateData($\pi, \beta, n, m, \eta$)
    $\sigma \leftarrow (1-\eta)\pi + \eta\beta$
    $\mathcal{D} \leftarrow n$ episodes $\{t_k\}_{1\le k\le n}$, sampled from strategy profile $\sigma$
    for each player $i \in \mathcal{N}$ do
        $\mathcal{D}^i \leftarrow m$ episodes $\{t^i_k\}_{1\le k\le m}$, sampled from strategy profile $(\beta^i, \sigma^{-i})$
        $\mathcal{D}^i \leftarrow \mathcal{D}^i \cup \mathcal{D}$
    end for
    return $\{\mathcal{D}^k\}_{1\le k\le N}$
end function
```

4.3. Algorithm

This section introduces a general algorithm of FSP. Each iteration of the algorithm can be divided into three steps. Firstly, episodes of the game are simulated from the agents' strategies. The resulting experience or data is stored in two types of agent memory. One type stores experience of an agent's opponents' behaviour. The other type stores the agent's own behaviour. Secondly, each agent computes an approximate best response by reinforcement learning off-policy from its memory of its opponents' behaviour. Thirdly, each agent updates its own average strategy by supervised learning from the memory of its own behaviour.

Algorithm 2 presents the general framework of FSP. It does not specify particular off-policy reinforcement learning or supervised learning techniques, as these can be instantiated by a variety of algorithms. However, as discussed in the previous sections, in order to constitute a valid fictitious play process both machine learning operations require data sampled from specific combinations of strategies. The function GenerateData uses a sampling strategy profile $\sigma_k = (1-\eta_k)\pi_{k-1} + \eta_k\beta_k$, where π_{k-1} is the average strategy profile of iteration $k-1$ and β_k is the best response profile of iteration k. The parameter η_k mixes between these strategy profiles. In particular, choosing $\eta_k = \frac{1}{k}$ results in σ_k matching the average strategy profile π_k of a fictitious play process with stepsize $\alpha_k = \frac{1}{k}$. At iteration k, for each player i, we would simulate n episodes of play from (π^i_k, π^{-i}_k) and m episodes from (β^i_k, π^{-i}_k). All episodes can be used by reinforcement learning, as they constitute experience against π^{-i}_k that the agent wants to best respond to. For supervised learning, the sources of data need to be weighted to achieve a correct target distribution. On the one hand sampling from (π^i_k, π^{-i}_k) results in the correct target distribution. On the other hand, when performing incremental updates only episodes from (β^i_k, π^{-i}_k) might be used. Additional details of data generation, e.g. with non-full or finite memories are discussed in the experiments.

For clarity, the algorithm presents data collection from a centralized point of view. In practice, this can be thought of as a self-play process where each agent is responsible to remember its own experience. Also, extensions to an online and on-policy learning setting are possible, but have been omitted as they algorithmically intertwine the reinforcement learning and supervised learning operations.

5. Experiments

We evaluate the introduced algorithms in two parameterized zero-sum imperfect-information games. Leduc Hold'em (Southey et al., 2005) is a small poker variant that is similar to Texas Hold'em. With two betting rounds, a limit of two raises per round and 6 cards in the deck it is however much smaller. River poker is a game that is strategically equivalent to the last betting round of Limit Texas Hold'em. It is parameterized by a probability distribution over possible private holdings, the five publicly shared community cards, the initial potsize and a limit on the number of raises. The distributions over private holdings could be considered the players' beliefs that they have formed in the first three rounds of a Texas Hold'em game. At the beginning of the game, a private holding is sampled for each player from their respective distribution and the game progresses according to the rules of Texas Hold'em.

In a two-player zero-sum game, the exploitability of a strategy profile, π, is defined as $\delta = R^1\left(b^1(\pi^2), \pi^2\right) + R^2\left(\pi^1, b^2(\pi^1)\right)$. An exploitability of δ yields at least a δ-Nash equilibrium. In our experiments, we used exploitability to measure learning performance.

5.1. Full-Width Extensive-Form Fictitious Play

We compared the effect of information-state dependent stepsizes, $\lambda_{t+1} : \mathcal{U} \rightarrow [0,1]$, on full-width extensive-form fictitious play updates, $\pi_{t+1}(u) = \pi_t(u) + \lambda_{t+1}(u)(\beta_{t+1}(u) - \pi_t(u)), \forall u \in \mathcal{U}$, where $\beta_{t+1} \in b(\pi_t)$ is a sequential best response and π_t is the iteratively updated average strategy profile. Stepsize $\lambda^1_{t+1}(u) = \frac{1}{t+1}$ yields the sequential extensive-form fictitious play introduced by Hendon et al. (1996). XFP is implemented by stepsize $\lambda^2_{t+1}(u) = \frac{x_{\beta_{t+1}}(\sigma_u)}{t x_{\pi_t}(\sigma_u) + x_{\beta_{t+1}}(\sigma_u)}$.

The average strategies were initialized as follows. At each information state u, we drew the weight for each action from a uniform distribution and normalized the resulting strategy at u. We trained each algorithm for 400000 iterations and measured the average strategy profiles' exploitability after each iteration. The experiment was repeated 5 times and figure 1 plots the resulting learning curves. The results show noisy behaviour of each fictitious play process that used stepsize λ^1. Each XFP instance reliably reached a much better approximate Nash equilibrium.

5.2. Fictitious Self-Play

We tested the performance of FSP with a fixed computational budget per iteration and evaluated how it scales to larger games in comparison with XFP.

We instantiated FSP's reinforcement learning method with FQI and updated the average strategy profiles with a simple counting model. We manually calibrated FSP in 6-card Leduc Hold'em and used this calibration in all experiments and games. In particular, at each iteration, k, FQI replayed 30 episodes with learning stepsize $\frac{0.05}{1+0.003\sqrt{k}}$. It returned a policy that at each information state was determined by a Boltzmann distribution over the estimated Q-values, using temperature $(1 + 0.02\sqrt{k})^{-1}$. The state of FQI was maintained across iterations, i.e. it was initialized with the parameters and learned Q-values of the previous iteration. For each player i, FSP used a replay memory, $\mathcal{M}^i_{RL}$, with space for 40000 episodes. Once this memory was full, FSP sampled 2 episodes from strategy profile σ and 1 episode from (β^i, σ^{-i}) at each iteration for each player respectively, i.e. we set $n = 2$ and $m = 1$ in algorithm 2.

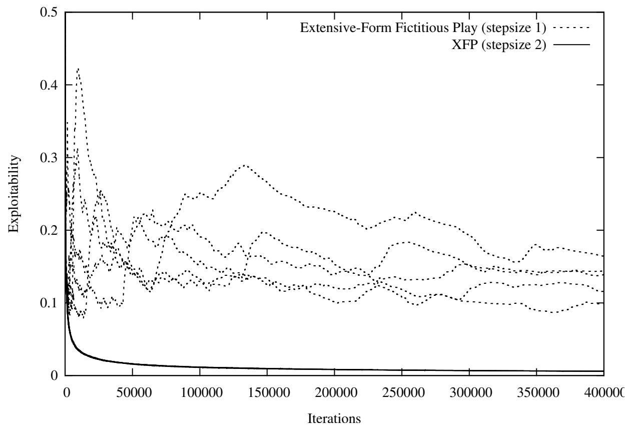

Figure 1. Learning curves of extensive-form fictitious play processes in Leduc Hold'em, for stepsizes λ^1 and λ^2.

Because we used finite memories and only partial replacement of episodes we had to make some adjustments to approximately correct for the expected target distributions. For a non-full or infinite memory, a correct target distribution can be achieved by accumulating samples from each opponent best response. Thus, for a non-full memory we collected all episodes from profiles (β^i, σ^{-i}) in alternating self-play, where agent i stores these in its supervised learning memory, $\mathcal{M}^i_{SL}$, and player $-i$ stores them in its non-full reinforcement learning memory, $\mathcal{M}^{-i}_{RL}$. However, when partially replacing a full reinforcement learning memory, $\mathcal{M}^i_{RL}$, that is trying to approximate experience against the opponent's $\Pi^{-i}_k = (1 - \alpha_k)\Pi^{-i}_{k-1} + \alpha_k B^{-i}_k$, with samples from Π^{-i}_k, we would underweight the amount of experience against the opponent's recent best response, B^{-i}_k. To approximately correct for this we set the mixing parameter to $\eta_k = \frac{\alpha_k}{\gamma p}$, where $p = \frac{n+m}{\text{MemorySize}}$ is the proportion of memory that is replaced and the constant γ controls how many iterations constitute one formal fictitious play iteration. In all our experiments, we used $\gamma = 10$. Both algorithms' average strategy profiles were initialized to a uniform distribution at each information state. Each algorithm trained for 300 seconds. The average strategy profiles' exploitability was measured at regular intervals.

Figure 2 compares both algorithms' performance in Leduc Hold'em. While XFP clearly outperformed FSP in the small 6-card variant, in the larger 60-card Leduc Hold'em it learned more slowly. This might be expected, as the computation per iteration of XFP scales linearly in the squared number of cards. FSP, on the other hand, operates only in information states whose number scales linearly with the number of cards in the game.

We compared the algorithms in two instances of River poker that were initialized with a potsize of 6, a maximum number of raises of 1 and a fixed set of community cards. The first instance assumes uninformed, uniform player beliefs that assign equal probability to each possible holding. The second instance assumes that players have inferred beliefs over their opponents' holdings. An expert poker player provided us with belief distributions that model a real Texas Hold'em scenario. The distributions assume that player 1 holds one of 16% of the possible holdings with probability 0.99 and a uniform random holding with probability 0.01. Similarly, player 2 is likely to hold one of 31% holdings. The exact distributions and scenario are provided in the appendix.

According to figure 3, FSP improved its average strategy profile much faster than the full-width variant in both instances of River poker. In River poker with defined beliefs, FSP obtained an exploitability of 0.11 after 30 seconds, whereas after 300 seconds XFP was exploitable by more than 0.26. Furthermore, XFP's performance was similar in both instances of River poker, whereas FSP lowered its exploitability by more than 40%. River poker has about 10 million states but only around 4000 information states. For a similar reason as in the Leduc Hold'em experiments, this might explain the overall better performance of FSP. Furthermore, the structure of the game assigns non-zero probability to each state of the game and thus the computational cost of XFP is the same for both instances of River poker. It performs computation at each state no matter how likely it is to occur. FSP on the other hand is guided by sampling and is therefore able to focus its computation on likely scenarios. This allows it to benefit from the additional structure introduced by the players' beliefs into the game.

6. Conclusion

We have introduced two fictitious play variants for extensive-form games. XFP is the first fictitious play algorithm that is entirely implemented in behavioural strategies while preserving convergence guarantees in games with the fictitious play property. FSP is a sample-based approach

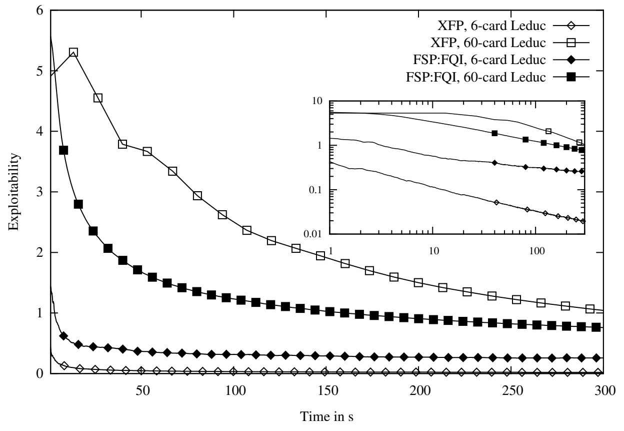

Figure 2. Comparison of XFP and FSP:FQI in Leduc Holdem. The inset presents the results using a logarithmic scale.

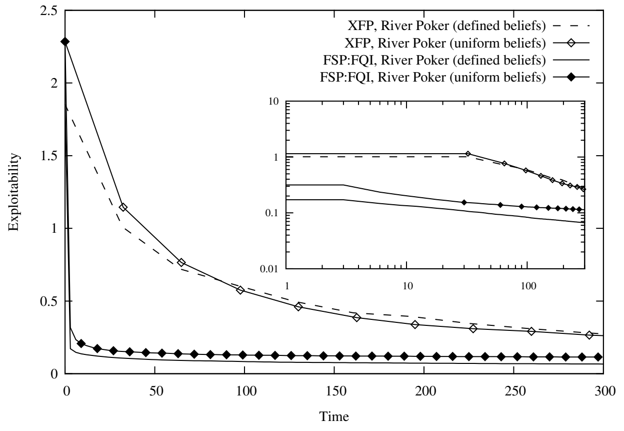

Figure 3. Comparison of XFP and FSP:FQI in River poker. The inset presents the results using a logarithmic scale for both axes.

that implements generalised weakened fictitious play in a machine learning framework. While converging asymptotically to the correct updates at each iteration, it remains an open question whether guaranteed convergence can be achieved with a finite computational budget per iteration. However, we have presented some intuition why this might be the case and our experiments provide first empirical evidence of its performance in practice.

FSP is a flexible machine learning framework. Its experiential and utility-maximizing nature makes it an ideal domain for reinforcement learning, which provides a plethora of techniques to learn efficiently from sequential experience. Function approximation could provide automated abstraction and generalisation in large extensive-form games. Continuous-action reinforcement learning could learn best responses in continuous action spaces. FSP has therefore a lot of potential to scale to large and even continuous-action game-theoretic applications.

Acknowledgments

We would like to thank Georg Ostrovski, Peter Dayan, Rémi Munos and Joel Veness for insightful discussions and feedback. This research was supported by the UK Centre for Doctoral Training in Financial Computing and Google DeepMind.

References

Benaïm, Michel, Hofbauer, Josef, and Sorin, Sylvain. Stochastic approximations and differential inclusions. *SIAM Journal on Control and Optimization*, 44(1):328–348, 2005.

Bosansky, B, Kiekintveld, Christopher, Lisy, V, and Pechoucek, Michal. An exact double-oracle algorithm for zero-sum extensive-form games with imperfect information. *Journal of Artificial Intelligence Research*, pp. 829–866, 2014.

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

Brown, George W. Iterative solution of games by fictitious play. *Activity analysis of production and allocation*, 13(1):374–376, 1951.

Daskalakis, Constantinos and Pan, Qinxuan. A counter-example to Karlin's strong conjecture for fictitious play. In *Foundations of Computer Science (FOCS), 2014 IEEE 55th Annual Symposium on*, pp. 11–20. IEEE, 2014.

Ernst, Damien, Geurts, Pierre, and Wehenkel, Louis. Tree-based batch mode reinforcement learning. In *Journal of Machine Learning Research*, pp. 503–556, 2005.

Fudenberg, Drew. *The theory of learning in games*, volume 2. MIT press, 1998.

Ganzfried, Sam and Sandholm, Tuomas. Computing equilibria in multiplayer stochastic games of imperfect information. In *Proceedings of the 21st International Joint Conference on Artifical Intelligence*, pp. 140–146, 2009.

Gilpin, Andrew, Hoda, Samid, Pena, Javier, and Sandholm, Tuomas. Gradient-based algorithms for finding Nash equilibria in extensive form games. In *Internet and Network Economics*, pp. 57–69. Springer, 2007.

Greenwald, Amy, Li, Jiacui, Sodomka, Eric, and Littman, Michael. Solving for best responses in extensive-form games using reinforcement learning methods. *The 1st Multidisciplinary Conference on Reinforcement Learning and Decision Making (RLDM)*, 2013.

Heinrich, Johannes and Silver, David. Smooth UCT search in computer poker. In *Proceedings of the 24th International Joint Conference on Artifical Intelligence*, 2015. In press.

Hendon, Ebbe, Jacobsen, Hans Jørgen, and Sloth, Birgitte. Fictitious play in extensive form games. *Games and Economic Behavior*, 15(2):177–202, 1996.

Hofbauer, Josef and Sandholm, William H. On the global convergence of stochastic fictitious play. *Econometrica*, 70(6):2265–2294, 2002.

Karlin, Samuel. *Mathematical methods and theory in games, programming and economics*. Addison-Wesley, 1959.

Koller, Daphne, Megiddo, Nimrod, and Von Stengel, Bernhard. Fast algorithms for finding randomized strategies in game trees. In *Proceedings of the 26th ACM Symposium on Theory of Computing*, pp. 750–759. ACM, 1994.

Koller, Daphne, Megiddo, Nimrod, and Von Stengel, Bernhard. Efficient computation of equilibria for extensive two-person games. *Games and Economic Behavior*, 14(2):247–259, 1996.

Kuhn, Harold W. Extensive games and the problem of information. *Contributions to the Theory of Games*, 2(28):193–216, 1953.

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

Lanctot, Marc, Waugh, Kevin, Zinkevich, Martin, and Bowling, Michael. Monte Carlo sampling for regret minimization in extensive games. In *Advances in Neural Information Processing Systems 22*, pp. 1078–1086, 2009.

Leslie, David S and Collins, Edmund J. Generalised weakened fictitious play. *Games and Economic Behavior*, 56(2):285–298, 2006.

McMahan, H Brendan and Gordon, Geoffrey J. A fast bundle-based anytime algorithm for poker and other convex games. In *International Conference on Artificial Intelligence and Statistics*, pp. 323–330, 2007.

Miltersen, Peter Bro and Sørensen, Troels Bjerre. Computing a quasi-perfect equilibrium of a two-player game. *Economic Theory*, 42(1):175–192, 2010.

Myerson, Roger B. *Game Theory: Analysis of Conflict*. Harvard University Press, 1991.

Robinson, Julia. An iterative method of solving a game. *Annals of Mathematics*, pp. 296–301, 1951.

Sandholm, Tuomas. The state of solving large incomplete-information games, and application to poker. *AI Magazine*, 31(4):13–32, 2010.

Silver, David and Veness, Joel. Monte-Carlo planning in large POMDPs. In *Advances in Neural Information Processing Systems*, pp. 2164–2172, 2010.

Southey, Finnegan, Bowling, Michael, Larson, Bryce, Piccione, Carmelo, Burch, Neil, Billings, Darse, and Rayner, Chris. Bayes bluff: Opponent modelling in poker. In *In Proceedings of the 21st Annual Conference on Uncertainty in Artificial Intelligence (UAI*, pp. 550–558, 2005.

Sutton, Richard S and Barto, Andrew G. *Reinforcement learning: An introduction*, volume 1. Cambridge Univ Press, 1998.

Von Stengel, Bernhard. Efficient computation of behavior strategies. *Games and Economic Behavior*, 14(2):220–246, 1996.

Watkins, Christopher JCH and Dayan, Peter. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.

Zinkevich, Martin, Johanson, Michael, Bowling, Michael, and Piccione, Carmelo. Regret minimization in games with incomplete information. In *Advances in Neural Information Processing Systems*, pp. 1729–1736, 2007.