Converging to Unexploitable Policies in Continuous Control Adversarial Games

Max Goldstein *
New York University

Noam Brown
Meta AI

Abstract

Fictitious Self-Play (FSP) is an iterative algorithm capable of learning approximate Nash equilibria in many types of two-player zero-sum games. In FSP, at each iteration, a best response is learned to the opponent's meta strategy. However, FSP can be slow to converge in continuous control games in which two embodied agents compete against one another. We propose Adaptive FSP (AdaptFSP), a deep reinforcement learning (RL) algorithm inspired by FSP. The main idea is that instead of training a best response only against the meta strategy, we additionally train against an adaptive deep RL agent that can adapt to the best response. In four test domains, two tabular cases–random normal-form matrix games, Leduc poker–and two continuous control tasks–Thou Shall Not Pass and a soccer environment–we show that AdaptFSP achieves lower exploitability more quickly than vanilla FSP.

1 Introduction

For many two-player zero-sum games such as poker [3, 23, 5, 7], variants of the counterfactual regret minimization (CFR) algorithm [30, 28, 6, 8, 20, 27, 13] are state of the art. However, a major limitation of CFR is that it is not clear how to extend it to continuous action spaces. Unlike CFR, Fictitious Self Play (FSP) [16] extends trivially to continuous action spaces. FSP is a iterative algorithm for solving two-player zero-sum (2p0s) games. At every iteration, each player computes a best response to the opponent's *meta-strategy*. In vanilla FSP, the *meta-strategy* is a uniform average of all the policies computed at previous iterations (i.e., on iteration k it is the uniform average of all the policies from iterations 1 through $k-1$). FSP is guaranteed to eventually converge to a Nash equilibrium in which both players' meta-strategy is an optimal response to the other [4].

FSP, unlike CFR, easily extends to continuous action spaces because each iteration can be viewed as solving a single-agent reinforcement learning problem, of which there are many options for continuous action space environments.

We observe that in large games, an issue with vanilla FSP is that learning the best response to the meta-strategy with deep RL can lead to overfitting to the meta-strategy policies which consequently slows down convergence to a low exploitability policy. For example, in a game of soccer the kicker might adversarially figure out how to move in such a way that the population of previous goalie policies has not seen before, causing degenerate behavior.

To address these issues, we propose the algorithm AdaptFSP. AdaptFSP augments the meta-strategy of a player on iteration k with an adaptive opponent that proactively updates its policy against the best response. The adaptive opponent acts as a regularizer that reduces overfitting to the previous population of policies.

We first prove that AdaptFSP falls into a category of algorithms known as Generalized Weakened Fictitious Play, all of which provably converge to Nash equilibria in 2p0s games. We then evaluate

*mag1038@nyu.edu

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

AdaptFSP in four tabular and continuous games. In our first set of experiments, we compare AdaptFSP to baselines on random normal-form matrix games as well as Leduc poker. For continuous games, we study the "you-shall-not-pass" [2] game from OpenAI gym as well as a MuJoCo soccer penalty kick scenario [21] based on a dm-control environment. We introduce partial observability into these environments by delaying the observation of the opponent by ten time steps. We demonstrate that AdaptFSP produces policies that are more robust and less exploitable than self play of FSP in these settings.

2 Related Work

Bansal et al. [2] study fully observable 2p0s games in which the agents have to perform continuous control. They use a method they call *opponent sampling*. In opponent sampling, when training a policy the opponent's behavior is sampled from a buffer of past policies. Instead of keeping all policies, they use a sliding window of past policies. This is very similar to standard FSP. However, they do not study the exploitability of their learned agents. Al-Shedivat et al. [1] study the question of meta learning in two-player competitive continuous control tasks. Two players play several rounds against each other, and they show that the meta-learning agent performs the best out of several candidate algorithms. The environments in these papers are fully observable, whereas we introduce partial observability to the environments in this paper. Many algorithms that converge to optimal policies in fully observable environments do not maintain that property in partially observable environments.

Neural Fictitious Self-Play [15] uses deep reinforcement learning to mimic fictitious play. It is able to learn a Nash equilbrium in 2p0s imperfect-information games from self-play. NFSP uses a deep neural network to approximate the average policy as well as a deep neural network to learn the policy with a RL style reward, but has only ever been shown to be effective in games with small action spaces.

Liu et al. [21] introduce the MuJoCo Soccer domain where teams of players can compete against each other in a soccer match. They use decentralized population-based training to evolve agents that show capabilities of cooperation and exploitation. We modify this environment for our penalty kick domain. Liu et al. [22] present a population-based learning method that converges to a Nash Equilbrium in 2p0s including in the same soccer environment from Liu et al. [21]. This soccer environment is fully observable whereas we introduce partial observability.

FSP can be viewed as a type of curriculum learning, in which there has been much work for continuous control. Eysenbach et al. [11] show they can maximize an information theoretic objective to learn a diverse set of skills that are generalizable to downstream tasks. Yarats et al. [29] present a self-supervised technique for learning different skills based on prototypical representations that shows strong generalization to downstream tasks.

Ghosh et al. [12] study generalization in RL in the context of POMDPs (Partially Observable Markov Decision Process) and show classic RL algorithms do not perform as well in POMDPs as in MDPs. They propose an ensemble-based approach and show that it generalizes (i.e. solves the POMDP) more effectively than vanilla RL. This does not apply directly to 2p0s games and requires training N policies for each agent, whereas our approach requires only two policies for each agent.

3 Background and Notation

3.1 Reinforcement Learning

We consider a finite horizon MDP defined by a tuple $(\mathcal{S}, \mathcal{A}, b, r)$ where $\mathcal{S} \subset \mathbb{R}^d$ is the state space and $\mathcal{A} \subset \mathbb{R}^k$ is the action space $\mathcal{A}$. $b : \mathcal{S} \times \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is a probability density function over the next state s_{t+1} given a state and action tuple (s_t, a_t). $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ specifies the reward for taking action a_t at state s_t. A policy $\pi : \mathcal{S} \to \mathcal{A}$ is a function mapping states to actions. In continuous RL, a state is typically mapped to a distribution over actions. A trajectory is a sequence $\tau = (s_1, a_1, r_1, s_2, \ldots, s_N, a_N, r_N)$. The reward for the trajectory τ is given by $R = \sum_{i=1}^{N} \gamma^i r_i$. A value function $V : \mathcal{S} \to \mathbb{R}$ is a function that maps states to future rewards, $V(s_t) \approx \sum_{i=t}^{N} \gamma^i r_i$, where γ is the discount factor. The discount factor is introduced to incentivize immediate rewards

over future rewards. The value function is typically trained with the bellman equation

$$V(s_t) = r(s_t, a_t) + \gamma V(s_{t+1})$$

A reinforcement learning algorithm is called on-policy if it updates its policy from trajectories that were collected using its current policy, whereas an off-policy algorithm can update its policy with data coming from arbitrary behavior trajectories.

For continuous games, we make use of two state-of-the-art reinforcement learning algorithms, namely Proximal Policy Optimization (PPO) and Soft-Actor Critic (SAC). These two algorithms have shown that they are capable of solving a wide variety of reinforcement learning tasks, and have achieved state-of-the-art in several continuous control tasks and games [17, 24].

3.1.1 Proximal Policy Optimization

PPO [25] is an actor-critic policy gradient algorithm. Given an MPD, it learns an approximately optimal policy π_θ and a value function V_θ. It is an on-policy algorithm, and as such it alternates from sampling data through on-policy interaction with the environment, and using that data to maximize its objective function, J_{PPO}.

$$\begin{aligned} J_{\mathrm{PG}}(\theta) &= \sum_{a \in \mathcal{A}} \pi_\theta(a|s) \hat{A}_{\theta_{\mathrm{old}}}(s,a) \\ &= \mathbb{E}_{a \sim \pi_{\theta_{\mathrm{old}}}} \left[\frac{\pi_\theta(a|s)}{\pi_{\theta_{\mathrm{old}}}(a|s)} \hat{A}_{\theta_{\mathrm{old}}}(s,a) \right], \end{aligned}$$

where $\hat{A}(\cdot)$ is an estimate of the advantage function $Q_\theta(s,a) - V_\theta(s)$ and $\pi_{\theta_{\mathrm{old}}}$ is the behavior policy used to collect trajectories. The current policy parameters might be different than the behavior policy, so we must use importance sampling to appropriately weigh the advantage terms. The full PPO objective is the sum of a clipped version of the PG objective, a value function loss and a policy entropy bonus.

3.1.2 Soft Actor Critic

SAC is an off-policy actor-critic algorithm, meaning it uses a buffer of past data to continually update its actor and critic functions. SAC learns a value function $V(s)$, a state-action value function $Q_\theta(s,a)$ and a stochastic policy $\pi_\theta(s)$ via the following objectives [14]. The $Q-$function is trained from trajectories an experience buffer $\mathcal{D}$ via the squared Bellman error

$$\mathcal{L}(Q_\theta) = \mathbb{E}_{\tau \sim \mathcal{D}} \left[\left(Q_\phi(s_t, a_t) - (r_t + \gamma V(s_{t+1})) \right)^2 \right]$$

and the value function V_θ is trained to compute the value of the $Q-$function plus an entropy bonus

$$V_\theta(s_{t+1}) = \mathbb{E}_{a \sim \pi_\theta} \left[Q_{\hat{\theta}}(s_{t+1}, a) - \alpha \log \pi_\theta(a|s_{t+1}) \right]$$

where $Q_{\hat{\theta}}$ is a moving average of the weights of the Q network. Finally, the policy is trained to minimze the loss

$$L(\pi_\theta) = \left[Q_\theta(s_t, a) - \alpha \log \pi_\theta(a|s_t) \right]$$

and α is a learned parameter.

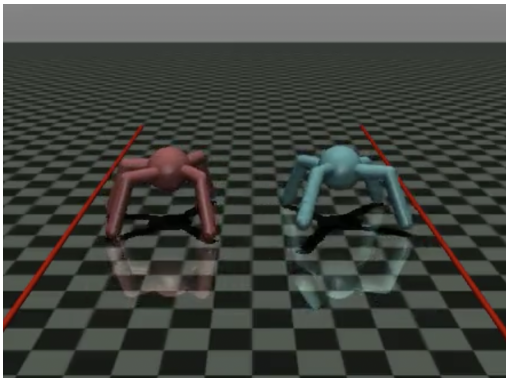
SAC updates a replay buffer $\mathcal{D}$ by collecting trajectories and performs gradient descent using ADAM on these objective functions.

3.2 Normal form games

A *normal-form game (NFG)* is a 2p0s game that can be specified by a matrix $P \in \mathbb{R}^{m \times n}$. Each row in the matrix corresponds to an action of player 1 and each column corresponds to an action of player 2. The entry P_{ij} specifies the payoff if player 1 plays i and player 2 plays j. A mixed strategy

$\pi^1 \in \mathbb{R}^\kappa$ for player 1 is a vector satisfying $\sum \pi_i^1 = 1, \pi_i^1 \geq 0$. The Nash equilibrium of a normal form game can be computed by solving the following linear program

$$\begin{aligned} &\max s \\ &\text{subject to } s \leq a_i^T P y \\ &\sum_{j=1}^{m} y_j = 1 \\ &y_j \geq 0 \end{aligned}$$



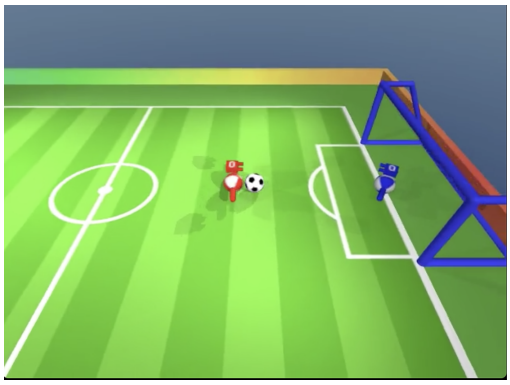


Figure 1: On the left, an image of the you-shall-not-pass environment. It consists of two gym ant creatures who are competing to pass each other. On the right, an image of the soccer environment [21]. There is a kicker (red), a goalie (blue), a soccer ball, and a goal. The objective of the kicker is to put the ball into the goal and the objective of the goalie is to keep the ball out of the goal.

The utility for player 1 of a policy profile $\pi = (\pi^1, \pi^2)$ is defined as $u^1(\pi^1, \pi^2)$ and is the expected payoff for player 1 if it follows π^1 and the opponent plays π^2. A two player game is called zero-sum if $u^1(\pi^1, \pi^2) = -u^2(\pi^1, \pi^2)$.

If player i plays according to π^i, a best response $BR(\pi^i) \in \Pi^{-i}$ is defined as

$$BR(\pi^i) \in \arg\max_{\pi^{-i}} u(\pi^{-i}, \pi^i)$$

which is the policy that does the best against player i who plays π^i. There may be multiple different best responses to a given policy. We may also denote a best response policy by β. A policy profile $\pi = (\pi^1, \pi^2)$ is a Nash Equilibrium if $\pi^1 = BR(\pi^2)$ and $\pi^2 = BR(\pi^1)$. The exploitability of a policy π^1 is defined to be how much worse the policy does against $BR(\pi^1)$ compared to how a Nash equilibrium strategy π_*^1 does against a best response to it, formally

$$E(\pi^1) = u^1(\pi_*^1, BR(\pi_*^1)) - u^1(\pi^1, BR(\pi^)$$

An ϵ-*best response* is an approximate best response. Formally, a policy π^2 is an ϵ-best response to π^1 if $u^2(\pi^1, BR(\pi^1)) - u^2(\pi^1, \pi^2) \leq \epsilon$.

3.3 Fictitious Play

Fictitous Play (FP) is an iterative algorithm that is very simple and also provably converges to a Nash equilbrium in 2p0s games. At the first iteration agents choose a uniform policy β_u^1, β_u^2 and initialize their average policy to this uniform policy $\pi^1 = \beta_u^1, \pi^2 = \beta_u^2$. At each subsequent iteration, the agent computes a best response to the opponent's average policy $\beta_t^p = \arg\max_{\beta^p}(\beta^p, \pi_{t-1}^{-p})$ and update their average policy via $\pi_t^p = \frac{t-1}{t}\pi_{t-1}^p + \frac{1}{t}\beta_t^p$. Brown et al. [9] introduce a variant of fictitious play in which the policies are weighed linearly rather than uniformly (i.e., π_t is given a relative weight of t). This has been shown to converge more quickly than FSP and as such we use linear weighting for our experiments in this paper. In this case, the update formula for the average policy is $\pi_t^p = \frac{t-1}{t+1}\pi_{t-1}^p + \frac{2}{t+1}\beta_t^p$.

Let $\beta^i(\pi^{-i})$ denote a ϵ_t approximate best response for player i to the policy π^{-i} Here we state the Generalized Weakened Fictitious Play theorem, as it appears in [19].

Definition 1 *A **generalised weakened fictitious play** is a sequence of mixed strategies $\{\Pi_t\}$ such that*

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

where $\alpha_t \to 0, \epsilon_t \to 0$ as $t \to \infty$ and M_t is a sequence of perturbations such that for all $\delta > 0$

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

Algorithm 1 Fictitious Play

Initialize π^1_1, π^2_1 to uniform policies
for $t = 1, \ldots, N$ **do**
  for $p = 1, 2$ **do**
    $\beta^p_{t+1} \leftarrow \arg\max_{\beta^p}(\beta^p, \pi^{-p}_t)$
    $\pi^i_{t+1} \leftarrow \frac{t-1}{t+1}\pi^p_{t-1} + \frac{2}{t+1}\beta^p_t$
  end for
end for

4 Description of Adaptive Fictitious Play

Algorithm 2 Adaptive Fictitious Play

Input: N number of iterations, M number of inner loop iterations
Initialize π^1_0, π^2_0 to be uniform policies
for $i = 1, 2, \ldots, N$ **do**
  for $j = 1, 2, \ldots, M$ **do**
    for $p = 1, 2$ **do**
      $\psi^p_j \leftarrow AVG(\{\beta^p_1, \ldots, \beta^p_{j-1}\})$
      $\beta^p_j \leftarrow BR(AVG_j(\{\pi^{-p}_i, \psi^p_j\})$
    end for
  end for
  for $p = 1, 2$ **do**
    $\pi^p_i \leftarrow AVG(\{\pi^p_{i-1}, \beta^p_M\})$
  end for
end for
Output: Approximate Nash equilibrium
$\pi = (\pi^1_N, \pi^2_N)$

4.1 Exact case

We first present our algorithm in the tabular case where it is possible to compute exact best responses (algorithm 2). Similar to vanilla fictitious play, we run N iterations (or until convergence). Rather than computing a best response to the meta strategy of the opponent, we compute a best response to a linear combination of the meta strategy and a regularizer policy. Suppose that we are at iteration i of AdaptFP, i.e. that we have meta strategies for each player π^1_i, π^2_i. We initialize the regularizer policy to be the null policy, and in each iteration j of the inner loop, we compute

$$\psi^p_j = AVG(\{\beta^p_1, \ldots, \beta^p_{j-1}\})$$
$$\beta^p_j = BR(AVG_j(\{\pi^{-p}_i, \psi^p_j\})$$

where AVG is a uniform average and $AVG_i(x, y) = \frac{i-1}{i}x + \frac{1}{i}y$. From player 1's perspective (it is symmetric for player 2), the inner loop for player 2 is adversarially finding a best response to the opponent's, and player 1 must learn a best response to an opponent that is allowed to adapt.

At the end of the inner loop the most recent best response is added to the meta policy. It is not fair to compare exploitability purely based on iterations since in the adaptive case we are computing many more best responses than in the non-adaptive case. Thus in our figures we show results based on the total number of best responses computed for each player.

Proposition 1 proves that AFP converges to a Nash equilibrium in 2p0s games. The crux of the proof is that AFP is an instance of a broader category of algorithms called Generalized Weakened Fictitious Play [19] (GWFP). All instances of GWFP converge to a Nash equilibrium in 2p0s games.

Proposition 1 *AFP converges to a Nash equilibrium in two-player zero-sum games.*

Proof 1 *Assume that the range of payoffs in the game is M. On each iteration t, $\beta_t^p = BR(\frac{t-1}{t}\pi_{t-1}^{-p} + \frac{1}{t}\psi_t^{-p})$, so β_t^p is an ϵ_t-best response to π_{t-1}^{-p} where $\epsilon_t \leq \frac{M}{t}$ and $\epsilon_t \to 0$ as $t \to \infty$. Thus, FSP is a form of GWFP with $\alpha_t = \frac{1}{t}$.*

4.2 Adaptive Fictitious Self Play

The approximate case is similar, but instead of the inner loop of finding best responses we just continuously train a player 2 regularizer agent during the process of approximating the best response for player 1. At each iteration i, we train a policy β_i^p for player p as well as a regularizer π_r^p, which is reinitialized each iteration. This policy is trained with a reinforcement learning algorithm (e.g. SAC [14] or PPO [25]) independently of β_i^{-p}. In training, the opponent's behavior is sampled from the average of the first i best responses $\beta_1^p, \ldots, \beta_i^p$ as well as the regularizer π_r^p.

In the continuous control setting we cannot simply average the policies as in the tabular case. As such, we adopt the same method as used in Single-Deep CFR [8] and [27] and sample the average policy as follows: At the start of each episode, we sample a policy from the set of policies according to the weights and run the entire episode with that policy. The pseudocode is in algorithm 3.

Algorithm 3 Adaptive Fictitious Self Play

Input: N number of iterations, T number of inner loop episodes
Initialize β_1^1, β_1^2 to be random policies
for $i = 1, 2, \ldots, N$ **do**
    Initialize β_r^1, β_r^2 regularizer policies
    Initialize data buffers D_1, D_2
    Initialize β_i^1, β_i^2
    for $t = 1, 2, \ldots, T$ **do**
        $\tilde{\pi}^1 = AVG_i(\{\pi^1, \pi_r^1\})$
        $\tilde{\pi}^2 = AVG_i(\{\pi^2, \pi_r^2\})$
        Collect episode sampled from $\pi^1, \tilde{\pi}^2$ and add to D_1
        Collect episode sampled from $\tilde{\pi}^1, \pi^2$ and add to D_2
        $\beta_i^1 \leftarrow$ ReinforcmentLearning(β_i^1, D_1); $\beta_r^1 \leftarrow$ ReinforcmentLearning(β_r^1, D_1)
        $\beta_i^2 \leftarrow$ ReinforcmentLearning(β_i^2, D_2); $\beta_r^2 \leftarrow$ ReinforcmentLearning(β_r^2, D_2)
    end for
    for $p = 1, 2$ **do**
        $\pi_i^p \leftarrow AVG(\{\beta_1^p, \ldots, \beta_i^p\})$
    end for
end for
Output: Approximate Nash equilibrium
$\pi = (\pi_N^1, \pi_N^2)$

5 Experiments

We run all experiments on a machine with two Nvidia RTX 3090 GPUs.

5.1 Normal form matrix games

First, we compare FP and AFP in normal-form games. As a test bed, we use 500x500 random matrix games where the matrix entries are drawn from a uniform distribution on $[0, 1)$. We run the algorithm with 10 different random matrices and plot average exploitability at each iteration as well as error bars. We compute exact best responses at each iteration of FP and AFP with linear programming.

5.2 Leduc poker

Leduc poker is a simplified version of poker [26]. It is played with six cards: two jacks, two queens, and two kings. Each player gets dealt a single card, and places a one unit ante. There are two betting rounds, and each betting round the player can have a maximum of two raises. A community card is revealed after the first round of betting, and payouts are determined similarly to regular poker: paired cards beat unpaired cards, and otherwise K > Q > J.

We run experiments with Fictitious Play, Adaptive Fictitious Play, and Counterfactual Regret Minimization. All three implementations are either from or based on the Openspiel python code base [18].

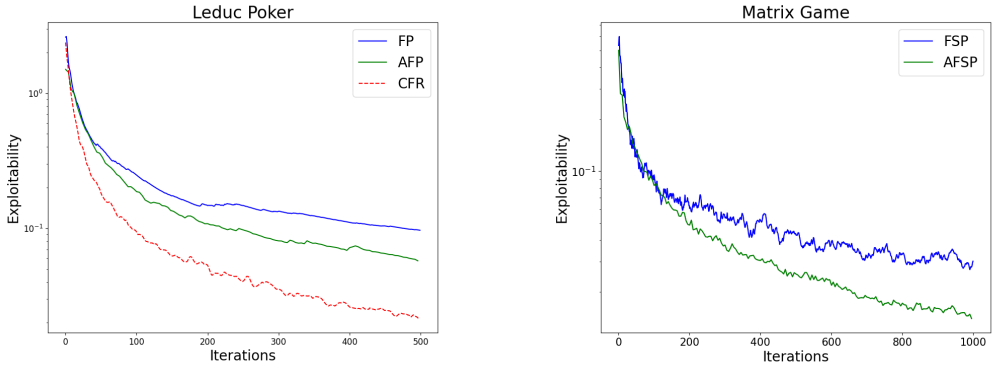


Figure 2: Results for tabular experiments. The y-axis is on a log scale. We plot exploitability agains the number of iterations of each algorithm. The left handside are the results for Leduc poker, where we run FP, AFP, and CFR. The right hand side are the results for random normal form games. We over ten matrices and compute error bars at a 95% confidence interval. TODO add titles !!

5.3 You-shall-not-pass

This environment consists of two ant-like creatures starting at opposite sides of an alleyway, pictured in the left side of figure 1. The winner is declared if the ant can reach the opposite side of the alley. We train all of the algorithms for 65 rounds. At the start of each round we initialize the new policy with the weights from the most recent round in order to speed up training. We find it does not make a significant difference in exploitability and significantly speeds up training.

In training the best response, we use several snapshots of the parameters from various points in training and train all of them against the policy in order to ensure we are getting as close to a true best response as possible. Each agent gets a reward of 1000 if they win and there is an auxiliary reward that encourages forward movement (i.e. movement toward the goal).

We run self play [10], FSP and AFSP each for five seeds in this environment. We use the PPO reinforcement learning algorithm since we find it to be most effective in this environment. Observations consist of data on the agent's current position as well as the opposing agent. the To introduce imperfect information into the game, the opponent's agent is lagged by 10 time steps.

At the start of each episode, we sample a policy from the set according to the given weights and use that policy for the whole episode. We repeat this for the entire training of the best response, which assures that the policy is trained against the average policy.

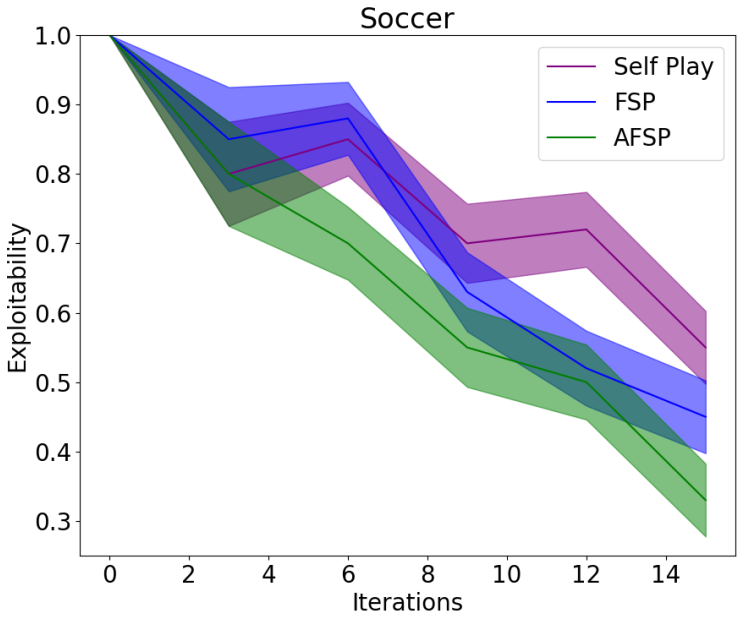


Figure 3: Results for soccer task. The y-axis is on a log scale. We plot exploitability against the number of iterations for each algorithm. We compare self-play, FSP, and AFSP. For each environment we run five seeds.

5.4 Soccer

The penalty kick environment consists of two players, a kicker and a goalie and an object which is the soccer ball, pictured in the right side of figure 1. The kicker tries to maneuver the soccer ball into the goal, and the goalie tries to not let this happen. The kicker has a time limit of 2500. If it does not succeed, the goalie gets reward of 1000 and the kicker gets reward of -1000. If the kicker does succeed, these rewards are swapped.

In particular, the kicker gets a reward for moving the ball toward the goal, and the goalie gets a reward for moving toward the ball. Having auxiliary rewards has been shown to significantly speed up training and is widely used [2]. We run self play, FSP, and AFSP for 5 seeds each. In this environment, we find that the agents train most efficiently with SAC. We train the best response the same as we do in you-shall-not-pass.

Observations consist of data on the agent's current position, the position relative to the ball, the position relative to the goal, and the opponent's position. Similar to the thou-shall-not-pass environment, we introduce imperfect information to the game by lagging the observation of the opponent's state by 10 time steps.

6 Discussion

While there has been tremendous progress in finding equilibria in games with discrete action spaces, there has not been as much progress in doing so with continuous action space games. Particularly, CFR is the state of the art in games with discrete action spaces, but it is not currently possible to apply CFR to continuous action space games. FSP works both in discrete and continuous action spaces, but we show in this paper that it is possible to converge to a Nash equilibrium more quickly than FSP. In continuous control where training best responses is expensive it is important that the algorithm is as efficient as possible. Similarly, pure self-play is not optimal in the presence of imperfect information or intransitivity. We show that AdaptFSP is more efficient than both FSP and self-play.

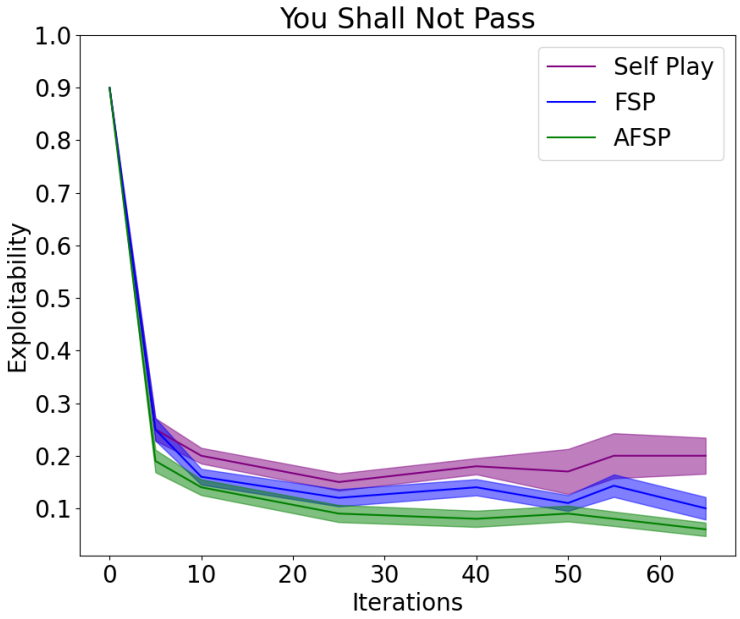


Figure 4: Results for you-shall-not-pass task. The y-axis is on a log scale. We plot exploitability against the number of iterations for each algorithm. We compare self-play, FSP, and AFSP. For each environment we run five seeds.

We can interpret fictitious play as a form of curriculum learning. The past policies of the opponent make up the curriculum, and the agent can never forget how to beat a previous policy in order to maintain good performance. However, one issue with this is that, especially early on in training, it is possible for the policy to overfit to the opponent's curriculum and learn behavior that is capable of beating the current opponents but is not robust to new opponents. We address this issue by introducing an adaptive regularizer agent to the curriculum which the best response agent also attempts to beat. Since this regularizer agent is adaptive, the best response training cannot overfit as much to the current policies.

We prove that AFSP converges to a Nash equilibrium. In our experiments, we first show in small games, specifically random matrix normal-form games and Leduc poker, that tabular AFSP converges to a Nash equilibrium more quickly than FSP. In two continuous control games, specifically the "you shall not pass" environment and the soccer environment, we use state-of-the-art reinforcement learning algorithms (PPO and SAC) and study the approximate version of AFSP. we introduce imperfect information into these environments by lagging the observation of the opponent's state, and we show here that AFSP converges more quickly and to a lower final exploitability than either pure self play or FSP.

In the future, much work remains to develop robust algorithms to solve continuous control games. This type of game is very important as many real-world applications involve continuous action spaces, including application involving robots in the world. While this work focuses on two-player zero-sum games, we view this as just an initial step toward developing agents that can more broadly act effectively in real-world interactions. A next step is to move toward more complex continuous state and action space environments, including environments that involve both cooperation and competition.

References

[1] M. Al-Shedivat, T. Bansal, Y. Burda, I. Sutskever, I. Mordatch, and P. Abbeel. Continuous adaptation via meta-learning in nonstationary and competitive environments. *CoRR*, abs/1710.03641, 2017. URL http://arxiv.org/abs/1710.03641.

[2] T. Bansal, J. Pachocki, S. Sidor, I. Sutskever, and I. Mordatch. Emergent complexity via multi-agent competition. *CoRR*, abs/1710.03748, 2017. URL http://arxiv.org/abs/1710.03748.

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

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

[5] N. Brown and T. Sandholm. Superhuman AI for heads-up no-limit poker: Libratus beats top professionals. *Science*, page eaao1733, 2017.

[6] N. Brown and T. Sandholm. Solving imperfect-information games via discounted regret minimization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 1829–1836, 2019.

[7] N. Brown and T. Sandholm. Superhuman AI for multiplayer poker. *Science*, page eaay2400, 2019.

[8] N. Brown, A. Lerer, S. Gross, and T. Sandholm. Deep counterfactual regret minimization. In *International Conference on Machine Learning*, pages 793–802, 2019.

[9] N. Brown, A. Bakhtin, A. Lerer, and Q. Gong. Combining deep reinforcement learning and search for imperfect-information games. *Advances in Neural Information Processing Systems*, 33:17057–17069, 2020.

[10] A. DiGiovanni and E. C. Zell. Survey of self-play in reinforcement learning. *CoRR*, abs/2107.02850, 2021. URL https://arxiv.org/abs/2107.02850.

[11] B. Eysenbach, A. Gupta, J. Ibarz, and S. Levine. Diversity is all you need: Learning skills without a reward function. *CoRR*, abs/1802.06070, 2018. URL http://arxiv.org/abs/1802.06070.

[12] D. Ghosh, J. Rahme, A. Kumar, A. Zhang, R. P. Adams, and S. Levine. Why generalization in RL is difficult: Epistemic pomdps and implicit partial observability. *CoRR*, abs/2107.06277, 2021. URL https://arxiv.org/abs/2107.06277.

[13] A. Gruslys, M. Lanctot, R. Munos, F. Timbers, M. Schmid, J. Perolat, D. Morrill, V. Zambaldi, J.-B. Lespiau, J. Schultz, et al. The advantage regret-matching actor-critic. *arXiv preprint arXiv:2008.12234*, 2020.

[14] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *CoRR*, abs/1801.01290, 2018. URL http://arxiv.org/abs/1801.01290.

[15] J. Heinrich and D. Silver. Deep reinforcement learning from self-play in imperfect-information games. *CoRR*, abs/1603.01121, 2016. URL http://arxiv.org/abs/1603.01121.

[16] J. Heinrich, M. Lanctot, and D. Silver. Fictitious self-play in extensive-form games. In *International conference on machine learning*, pages 805–813. PMLR, 2015.

[17] I. Kostrikov, D. Yarats, and R. Fergus. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. *CoRR*, abs/2004.13649, 2020. URL https://arxiv.org/abs/2004.13649.

[18] M. Lanctot, E. Lockhart, J.-B. Lespiau, V. Zambaldi, S. Upadhyay, J. Pérolat, S. Srinivasan, F. Timbers, K. Tuyls, S. Omidshafiei, D. Hennes, D. Morrill, P. Muller, T. Ewalds, R. Faulkner, J. Kramár, B. D. Vylder, B. Saeta, J. Bradbury, D. Ding, S. Borgeaud, M. Lai, J. Schrittwieser, T. Anthony, E. Hughes, I. Danihelka, and J. Ryan-Davis. OpenSpiel: A framework for reinforcement learning in games. *CoRR*, abs/1908.09453, 2019. URL `http://arxiv.org/abs/1908.09453`.

[19] D. S. Leslie and E. J. Collins. Generalised weakened fictitious play. *Games and Economic Behavior*, 56(2):285–298, 2006.

[20] H. Li, K. Hu, S. Zhang, Y. Qi, and L. Song. Double neural counterfactual regret minimization. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=ByedzkrKvH`.

[21] S. Liu, G. Lever, J. Merel, S. Tunyasuvunakool, N. Heess, and T. Graepel. Emergent coordination through competition. *CoRR*, abs/1902.07151, 2019. URL `http://arxiv.org/abs/1902.07151`.

[22] S. Liu, L. Marris, D. Hennes, J. Merel, N. Heess, and T. Graepel. Neupl: Neural population learning. 2022. doi: 10.48550/ARXIV.2202.07415. URL `https://arxiv.org/abs/2202.07415`.

[23] M. Moravčík, M. Schmid, N. Burch, V. Lisỳ, D. Morrill, N. Bard, T. Davis, K. Waugh, M. Johanson, and M. Bowling. Deepstack: Expert-level artificial intelligence in heads-up no-limit poker. *Science*, 356(6337):508–513, 2017.

[24] R. Raileanu, M. Goldstein, D. Yarats, I. Kostrikov, and R. Fergus. Automatic data augmentation for generalization in deep reinforcement learning. *CoRR*, abs/2006.12862, 2020. URL `https://arxiv.org/abs/2006.12862`.

[25] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017. URL `http://arxiv.org/abs/1707.06347`.

[26] F. Southey, M. Bowling, B. Larson, C. Piccione, N. Burch, D. Billings, and D. C. Rayner. Bayes' bluff: Opponent modelling in poker. *CoRR*, abs/1207.1411, 2012. URL `http://arxiv.org/abs/1207.1411`.

[27] E. Steinberger, A. Lerer, and N. Brown. Dream: Deep regret minimization with advantage baselines and model-free learning. *arXiv preprint arXiv:2006.10410*, 2020.

[28] O. Tammelin. Solving large imperfect information games using cfr+. *arXiv preprint arXiv:1407.5042*, 2014.

[29] D. Yarats, R. Fergus, A. Lazaric, and L. Pinto. Reinforcement learning with prototypical representations. *CoRR*, abs/2102.11271, 2021. URL `https://arxiv.org/abs/2102.11271`.

[30] M. Zinkevich, M. Johanson, M. Bowling, and C. Piccione. Regret minimization in games with incomplete information. In *Advances in neural information processing systems*, pages 1729–1736, 2008.

A Appendix

A.1 Soccer hyper-parameters

Table 1: Hyperparameters for SAC in the soccer environment.

First column	Second column
Episodes	2500
Actor learning rate	3e-3
Critic learning rate	3e-3
Batch size	512
Hidden units	256
Replay buffer size	$1e6$
Non-linearity	ReLU
β_1	.9
β_2	.999
Optimizer	Adam
γ	.99

A.2 You shall not pass hyper-parameters

Table 2: Hyperparameters for PPO in the soccer environent

First column	Second column
Episodes	1000
Actor learning rate	3e-4
Critic learning rate	3e-4
Hidden units	512
Timesteps per rollout	512
Minibatches per rollout	32
Number of workers	16
Environments per worker	1
Optimizer	Adam
γ	.99
Entropy bonus	.01
Value loss coefficient	.5
PPO epochs	10
PPO clip range	.2