Actor-Critic Policy Optimization in Partially Observable Multiagent Environments

Sriram Srinivasan^{*,1} srsrinivasan@ **Marc Lanctot**^{*,1} lanctot@ **Vinicius Zambaldi**^{1} vzambaldi@ **Julien Pérolat**^{1} perolat@

Karl Tuyls^{1} karltuyls@ **Rémi Munos**^{1} munos@ **Michael Bowling**^{1} bowlingm@

...@google.com. ^{1}DeepMind. *These authors contributed equally.

Abstract

Optimization of parameterized policies for reinforcement learning (RL) is an important and challenging problem in artificial intelligence. Among the most common approaches are algorithms based on gradient ascent of a score function representing discounted return. In this paper, we examine the role of these policy gradient and actor-critic algorithms in partially-observable multiagent environments. We show several candidate policy update rules and relate them to a foundation of regret minimization and multiagent learning techniques for the one-shot and tabular cases, leading to previously unknown convergence guarantees. We apply our method to *model-free* multiagent reinforcement learning in adversarial sequential decision problems (zero-sum imperfect information games), using RL-style function approximation. We evaluate on commonly used benchmark Poker domains, showing performance against fixed policies and empirical convergence to approximate Nash equilibria in self-play with rates similar to or better than a baseline model-free algorithm for zero-sum games, without any domain-specific state space reductions.

1 Introduction

There has been much success in learning parameterized policies for sequential decision-making problems. One paradigm driving progress is deep reinforcement learning (Deep RL), which uses deep learning [52] to train function approximators that represent policies, reward estimates, or both, to learn directly from experience and rewards [84]. These techniques have learned to play Atari games beyond human-level [60], Go, chess, and shogi from scratch [82, 81], complex behaviors in 3D environments [59, 96, 37], robotics [27, 73], character animation [67], among others.

When multiple agents learn simultaneously, policy optimization becomes more complex. First, each agent's environment is non-stationary and naive approaches can be non-Markovian [58], violating the requirements of many traditional RL algorithms. Second, the optimization problem is not as clearly defined as maximizing one's own expected reward, because each agent's policy affects the others' optimization problems. Consequently, game-theoretic formalisms are often used as the basis for representing interactions and decision-making in multiagent systems [17, 79, 64].

Computer poker is a common multiagent benchmark domain. The presence of partial observability poses a challenge for traditional RL techniques that exploit the Markov property. Nonetheless, there has been steady progress in poker AI. Near-optimal solutions for heads-up limit Texas Hold'em were found with tabular methods using state aggregation, powered by policy iteration algorithms based on regret minimization [101, 86, 12]. These approaches were founded on a basis of counterfactual

32nd Conference on Neural Information Processing Systems (NeurIPS 2018), Montréal, Canada.

regret minimization (CFR), which is the root of recent advances in no-limit, such as Libratus [16] and DeepStack [61]. However, (i) both required Poker-specific domain knowledge, and (ii) neither were model-free, and hence are unable to learn directly from experience, without look-ahead search using a perfect model of the environment.

In this paper, we study the problem of multiagent reinforcement learning in adversarial games with partial observability, with a focus on the model-free case where agents (a) do not have a perfect description of their environment (and hence cannot do a priori planning), (b) learn purely from their own experience without explicitly modeling the environment or other players. We show that actor-critics reduce to a form of regret minimization and propose several policy update rules inspired by this connection. We then analyze the convergence properties and present experimental results.

2 Background and Related Work

In this section, we briefly describe the necessary background. While we draw on game-theoretic formalisms, we choose to align our terminology with the RL literature to emphasize the setting and motivations. We include clarifications in Appendix A. For details, see [79, 84].

2.1 Reinforcement Learning and Policy Gradient Algorithms

An agent acts by taking actions $a \in \mathcal{A}$ in states $s \in \mathcal{S}$ from their policy $\pi : s \rightarrow \Delta(\mathcal{A})$, where $\Delta(X)$ is the set of probability distributions over X, which results in changing the state of the environment $s_{t+1} \sim \mathcal{T}(s_t, a_t)$; the agent then receives an observation $o(s_t, a_t, s_{t+1}) \in \Omega$ and reward R_t.[1] A sum of rewards is a **return** $G_t = \sum_{t'=t}^{\infty} R_{t'}$, and aim to find π^* that maximizes expected return $\mathbb{E}_\pi[G_0]$.[2]

Value-based solution methods achieve this by computing estimates of $v_\pi(s) = \mathbb{E}_\pi[G_t \mid S_t = s]$, or $q_\pi(s, a) = \mathbb{E}_\pi[G_t \mid S_t = s, A_t = a]$, using temporal difference learning to bootstrap from other estimates, and produce a series of ϵ-greedy policies $\pi(s, a) = \epsilon/|\mathcal{A}| + (1-\epsilon)\mathbb{I}(a = \text{argmax}_{a'}\, q_\pi(s, a'))$. In contrast, policy gradient methods define a score function $J(\pi_\theta)$ of some parameterized (and differentiable) policy π_θ with parameters θ, and use gradient ascent directly on $J(\pi_\theta)$ to update θ.

There have been several recent successful applications of policy gradient algorithms in complex domains such as self-play learning in AlphaGo [80], Atari and 3D maze navigation [59], continuous control problems [76, 54, 21], robotics [27], and autonomous driving [78]. At the core of several recent state-of-the-art Deep RL algorithms [37, 22] is the advantage actor-critic (A2C) algorithm defined in [59]. In addition to learning a policy (*actor*), A2C learns a parameterized *critic*: an estimate of $v_\pi(s)$, which it then uses both to estimate the remaining return after k steps, and as a control variate (*i.e.* baseline) that reduces the variance of the return estimates.

2.2 Game Theory, Regret Minimization, and Multiagent Reinforcement Learning

In multiagent RL (MARL), $n = |\mathcal{N}| = |\{1, 2, \cdots, n\}|$ agents interact within the same environment. At each step, each agent i takes an action, and the joint action $\mathbf{a}$ leads to a new state $s_{t+1} \sim \mathcal{T}(s_t, \mathbf{a}_t)$; each player i receives their own separate observation $o_i(s_t, \mathbf{a}, s_{t+1})$ and reward $r_{t,i}$. Each agent maximizes their own return $G_{t,i}$, or their expected return which depends on the joint policy π.

Much work in classical MARL focuses on Markov games where the environment is fully observable and agents take actions simultaneously, which in some cases admit Bellman operators [55, 102, 70, 69]. When the environment is partially observable, policies generally map to values and actions from agents' observation histories; even when the problem is cooperative, learning is hard [65].

We focus our attention to the setting of zero-sum games, where $\sum_{i \in \mathcal{N}} r_{t,i} = 0$. In this case, polynomial algorithms exist for finding optimal policies in finite tasks for the two-player case. The guarantees that Nash equilibrium provides are less clear for the $(n > 2)$-player case, and finding one is hard [20]. Despite this, regret minimization approaches are known to filter out dominated actions, and have empirically found good (*e.g.* competition-winning) strategies in this setting [74, 26, 48].

[1] Note that in fully-observable settings, $o(s_t, a_t, s_{t+1}) = s_{t+1}$. In partially observable environments [39, 65], an observation function $\mathcal{O} : \mathcal{S} \times \mathcal{A} \rightarrow \Delta(\Omega)$ is used to sample $o(s_t, a_t, s_{t+1}) \sim O(s_t, a_t)$.

[2] We assume finite episodic tasks of bounded length and leave out the discount factor γ to simplify the notation, without loss of generality. We use $\gamma(= 0.99)$-discounted returns in our experiments.

The partially observable setting in multiagent reinforcement learning requires a few more key definitions in order to properly describe the notion of state. A **history** $h \in \mathcal{H}$ is a sequence of actions from all players *including the environment* taken from the start of an episode. The environment (also called "nature") is treated as a player with a fixed policy, such that there is a deterministic mapping from any h to the actual state of the environment. Define an **information state**, $s_t = \{h \in \mathcal{H} \mid$ player i's sequence of observations, $o_{i,t'<t}(s_{t'}, \mathbf{a}_{t'}, s_{t'+1})$, is consistent with $h\}$[3]. So, s_t includes histories leading to s_t that are indistinguishable to player i; *e.g.* in Poker, the $h \in s_t$ differ only in the private cards dealt to opponents. A joint policy π is a **Nash equilibrium** if the incentive to deviate to a best response $\delta_i(\pi) = \max_{\pi'_i} \mathbb{E}_{\pi'_i, \pi_{-i}}[G_{0,i}] - \mathbb{E}_\pi[G_{0,i}] = 0$ for each player $i \in \mathcal{N}$, where π_{-i} is the set of $i's$ opponents' policies. Otherwise, ϵ-equilibria are approximate, with $\epsilon = \max_i \delta_i(\pi)$. Regret minimization algorithms produce iterates whose average policy $\bar{\pi}$ reduces an upper bound on ϵ; convergence is measured using $\textsc{NashConv}(\pi) = \sum_i \delta_i(\pi)$. Nash equilibrium is minimax-optimal in two-player zero-sum games, so using one minimizes worst-case losses.

There are well-known links between learning, game theory and regret minimization [9]. One method, counterfactual regret (CFR) minimization [101], has led to significant progress in Poker AI. Let $\eta^\pi(h_t) = \prod_{t'<t} \pi(s_{t'}, a_{t'})$, where $h_{t'} \sqsubset h_t$ is a prefix, $h_{t'} \in s_{t'}, h_t \in s_t$, be the **reach probability** of h under π from all policies' action choices. This can be split into player i's contribution and their opponents' (including nature's) contribution, $\eta^\pi(h) = \eta^\pi_i(h)\eta^\pi_{-i}(h)$. Suppose player i is to play at s: under **perfect recall**, player i remembers the sequence of their own states reached, which is the same for all $h \in s$, since they differ only in private information seen by opponent(s); as a result $\forall h, h' \in s, \eta^\pi_i(h) = \eta^\pi_i(h') := \eta^\pi_i(s)$. For some history h and action a, we call h a **prefix history** $h \sqsubset ha$, where ha is the history h followed by action a; they may also be smaller, so $h \sqsubset ha \sqsubset hab \Rightarrow h \sqsubset hab$. Let $\mathcal{Z} = \{z \in \mathcal{H} \mid z \text{ is terminal}\}$ and $\mathcal{Z}(s,a) = \{(h,z) \in \mathcal{H} \times \mathcal{Z} \mid h \in s, ha \sqsubseteq z\}$. CFR defines **counterfactual values** $v^c_i(\pi, s_t, a_t) = \sum_{(h,z) \in \mathcal{Z}(s_t,a_t)} \eta^\pi_{-i}(h)\eta^\pi_i(z)u_i(z)$, and $v^c_i(\pi, s_t) = \sum_a \pi(s_t, a)v^c_i(\pi, s_t, a_t)$, where $u_i(z)$ is the return to player i along z, and accumulates regrets $\text{REG}_i(\pi, s_t, a') = v^c_i(\pi, s_t, a') - v^c_i(\pi, s_t)$, producing new policies from cumulative regret using *e.g.* regret-matching [28] or exponentially-weighted experts [6, 15].

CFR is a policy iteration algorithm that computes the expected values by visiting every possible trajectory, described in detail in Appendix B. Monte Carlo CFR (MCCFR) samples trajectories using an exploratory behavior policy, computing unbiased estimates $\hat{v}^c_i(\pi, s_t)$ and $\widehat{\text{REG}}_i(\pi, s_t)$ corrected by importance sampling [49]. Therefore, MCCFR is an *off-policy Monte Carlo* method. In one MCCFR variant, **model-free outcome sampling** (MFOS), the behavior policy at opponent states is defined as π_{-i} enabling online regret minimization (player i can update their policy independent of π_{-i} and $\mathcal{T}$).

There are two main problems with (MC)CFR methods: (i) significant variance is introduced by sampling (off-policy) since quantities are divided by reach probabilities, (ii) there is no generalization across states except through expert abstractions and/or forward simulation with a perfect model. We show that actor-critics address both problems and that they are a form of *on-policy* MCCFR.

2.3 Most Closely Related Work

There is a rich history of policy gradient approaches in MARL. Early uses of gradient ascent showed that cyclical learning dynamics could arise, even in zero-sum matrix games [83]. This was partly addressed by methods that used variable learning rates [13, 11], policy prediction [98], and weighted updates [1]. The main limitation with these classical works was scalability: there was no direct way to use function approximation, and empirical analyses focused almost exclusively on one-shot games.

Recent work on policy gradient approaches to MARL addresses scalability by using newer algorithms such as A3C or TRPO [76]. However, they focus significantly less (if at all) on convergence guarantees. Naive approaches such as independent reinforcement learning fail to find optimal stochastic policies [55, 32] and can overfit the training data, failing to generalize during execution [50]. Considerable progress has been achieved for cooperative MARL: learning to communicate [51], Starcraft unit micromanagement [24], taxi fleet optimization [63], and autonomous driving [78]. There has also been significant progress for mixed cooperative/competitive environments: using a centralized critic [57], learning to negotiate [18], anticipating/learning opponent responses in social dilemmas [23, 53], and control in realistic physical environments [3, 7]. In this line of research, the most common evaluation methodology has been to train centrally (for decentralized execution), either

[3] In defining s_t, we drop the reference to acting player i in turn-based games without loss of generality.

having direct access to the other players' policy parameters or modeling them explicitly. As a result, assumptions are made about the form of the other agents' policies, utilities, or learning mechanisms.

There are also methods that attempt to model the opponents [36, 30, 4]. Our methods do no such modeling, and can be classified in the "forget" category of the taxonomy proposed in [33]: that is, due to its on-policy nature, actors and critics adapt to and learn mainly from new/current experience.

We focus on the *model-free* (and online) setting: other agents' policies are inaccessible; training is not separated from execution. Actor-critics were recently studied in this setting for multiagent games [68], whereas we focus on partially-observable environments; only tabular methods are known to converge. Fictitious Self-Play computes approximate best responses via RL [31, 32], and can also be model-free. Regression CFR (RCFR) uses regression to estimate cumulative regrets from CFR [92]. RCFR is closely related to Advantage Regret Minimization (ARM) [38]. ARM [38] shows regret estimation methods handle partial observability better than standard RL, but was not evaluated in multiagent environments. In contrast, we focus primarily on the multiagent setting.

3 Multiagent Actor-Critics: Advantages and Regrets

CFR defines policy update rules from thresholded cumulative counterfactual regret: $\textsc{tcreg}_i(K, s, a) = (\sum_{k \in \{1, \cdots, K\}} \textsc{reg}_i(\pi_k, s, a))^+$, where k is the number of iterations and $(x)^+ = \max(0, x)$. In CFR, regret matching updates a policy to be proportional to $\textsc{tcreg}_i(K, s, a)$.

On the other hand, REINFORCE [94] samples trajectories and computes gradients for each state s_t, updating $\boldsymbol{\theta}$ toward $\nabla_{\boldsymbol{\theta}} \log(s_t, a_t; \boldsymbol{\theta}) G_t$. A baseline is often subtracted from the return: $G_t - v_\pi(s_t)$, and policy gradients then become actor-critics, training π and v_π separately. The log appears due to the fact that action a_t is sampled from the policy, the value is divided by $\pi(s_t, a_t)$ to ensure the estimate is properly estimating the true expectation [84, Section 13.3], and $\nabla_{\boldsymbol{\theta}} \pi(s_t, a_t; \boldsymbol{\theta}) / \pi(s_t, a_t, \boldsymbol{\theta}) = \nabla_{\boldsymbol{\theta}} \log \pi(s_t, a_t; \boldsymbol{\theta})$. One could instead train q_π-based critics from states *and* actions. This leads to a q-based Policy Gradient (QPG) (also known as Mean Actor-Critic [5]):

$$\nabla_{\boldsymbol{\theta}}^{\mathrm{QPG}}(s) = \sum_a [\nabla_\theta \pi(s, a; \boldsymbol{\theta})] \left(q(s, a; \mathbf{w}) - \sum_b \pi(s, b; \boldsymbol{\theta}) q(s, b, \mathbf{w}) \right), \tag{1}$$

an advantage actor-critic algorithm differing from A2C in the (state-action) representation of the critics [56, 95] and summing over actions similar to the all-action algorithms [85, 71, 19, 5]. Interpreting $a_\pi(s, a) = q_\pi(s, a) - \sum_b \pi(s, b) q_\pi(s, b)$ as a regret, we can instead minimize a loss defined by an upper bound on the thresholded cumulative regret: $\sum_k (a_{\pi_k}(s, a))^+ \geq (\sum_k (a_{\pi_k}(s, a))^+$, moving the policy toward a no-regret region. We call this Regret Policy Gradient (RPG):

$$\nabla_{\boldsymbol{\theta}}^{\mathrm{RPG}}(s) = -\sum_a \nabla_\theta \left(q(s, a; \mathbf{w}) - \sum_b \pi(s, b; \boldsymbol{\theta}) q(s, b; \mathbf{w}) \right)^+. \tag{2}$$

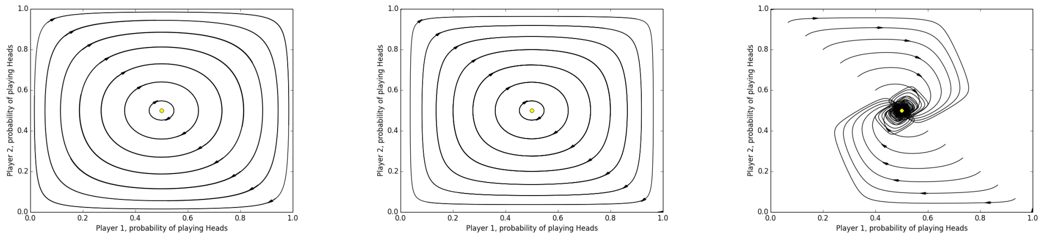
The minus sign on the front represents a switch from gradient ascent on the score to *descent* on the loss. Another way to implement an adaptation of the regret-matching rule is by weighting the policy gradient by the thresholded regret, which we call Regret Matching Policy Gradient (RMPG):

$$\nabla_{\boldsymbol{\theta}}^{\mathrm{RMPG}}(s) = \sum_a [\nabla_\theta \pi(s, a; \boldsymbol{\theta})] \left(q(s, a; \mathbf{w}) - \sum_b \pi(s, b; \boldsymbol{\theta}) q(s, b, \mathbf{w}) \right)^+. \tag{3}$$

In each case, the critic $q(s_t, a_t; \mathbf{w})$ is trained in the standard way, using ℓ_2 regression loss from sampled returns. The pseudo-code is given in Algorithm 2 in Appendix C. In Appendix F, we show that the QPG gradient is proportional to the RPG gradient at s: $\nabla_{\boldsymbol{\theta}}^{\mathrm{RPG}}(s) \propto \nabla_{\boldsymbol{\theta}}^{\mathrm{QPG}}(s)$.

3.1 Analysis of Learning Dynamics on Normal-Form Games

The first question is whether any of these variants can converge to an equilibrium, even in the simplest case. So, we now show phase portraits of the learning dynamics on Matching Pennies: a two-action version of Rock, Paper, Scissors. These analyses are common in multiagent learning as they allow visual depiction of the policy changes and how different factors affect the (convergence)



(a) Replicator Dynamics (b) RPG Dynamics (c) Average RPG Dynamics

Figure 1: Learning Dynamics in Matching Pennies: (a) and (b) show the vector field for $\partial\pi/\partial t$ including example particle traces, where each point is each player's probability of their first action; (c) shows example traces of policies following a discrete approximation to $\int_0^t \partial\pi/\partial t$.

behavior [83, 91, 13, 90, 11, 93, 1, 98, 97, 8, 88]. Convergence is difficult in Matching Pennies as the only Nash equilibrium $\pi^* = ((\frac{1}{2}, \frac{1}{2}), (\frac{1}{2}, \frac{1}{2}))$ requires learning stochastic policies. We give more detail and results on different games that cause cyclic learning behavior in Appendix D.

In Figure 1, we see the similarity of the regret dynamics to replicator dynamics [87, 75]. We also show the *average policy dynamics* and observe convergence to equilibrium in each game we tried, which is a known to be guaranteed in two-player zero-sum games using CFR, fictitious play [14], and continuous replicator dynamics [35]. However, computing the average policy is complex [31, 101] and potentially worse with function approximation, requiring storing past data in large buffers [32].

3.2 Partially Observable Sequential Games

How do the values $v_i^c(\pi, s_t, a_t)$ and $q_{\pi,i}(s_t, a_t)$ differ? The authors of [38] posit that they are approximately equal when s_t rarely occurs more than once in a trajectory. First, note that s_t cannot be reached more than once in a trajectory from our definition of s_t, because the observation histories (of the player to play at s_t) would be different in each occurrence (*i.e.* due to perfect recall). So, the two values are indeed equal in deterministic, single-agent environments. In general, counterfactual values are conditioned on *player i playing to reach* s_t, whereas q-function estimates are conditioned on *having reached* s_t. So, $q_{\pi,i}(s_t, a_t) = \mathbb{E}_{\rho\sim\pi}[G_{t,i} \mid S_t = s_t, A_t = a_t]$

$$
\begin{aligned}
&= \sum_{h,z\in\mathcal{Z}(s_t,a_t)} \Pr(h \mid s_t)\eta^\pi(ha,z)u_i(z) && \text{where } \eta^\pi(ha,z) = \frac{\eta^\pi(z)}{\eta^\pi(h)\pi(s,a)} \\
&= \sum_{h,z\in\mathcal{Z}(s_t,a_t)} \frac{\Pr(s_t \mid h)\Pr(h)}{\Pr(s_t)}\eta^\pi(ha,z)u_i(z) && \text{by Bayes' rule} \\
&= \sum_{h,z\in\mathcal{Z}(s_t,a_t)} \frac{\Pr(h)}{\Pr(s_t)}\eta^\pi(ha,z)u_i(z) && \text{since } h \in s_t, h \text{ is unique to } s_t \\
&= \sum_{h,z\in\mathcal{Z}(s_t,a_t)} \frac{\eta^\pi(h)}{\sum_{h'\in s_t}\eta^\pi(h')}\eta^\pi(ha,z)u_i(z) \\
&= \sum_{h,z\in\mathcal{Z}(s_t,a_t)} \frac{\eta_i^\pi(h)\eta_{-i}^\pi(h)}{\sum_{h'\in s_t}\eta_i^\pi(h')\eta_{-i}^\pi(h')}\eta^\pi(ha,z)u_i(z) \\
&= \sum_{h,z\in\mathcal{Z}(s_t,a_t)} \frac{\eta_i^\pi(s)\eta_{-i}^\pi(h)}{\eta_i^\pi(s)\sum_{h'\in s_t}\eta_{-i}^\pi(h')}\eta^\pi(ha,z)u_i(z) && \text{due to def. of } s_t \text{ and perfect recall} \\
&= \sum_{h,z\in\mathcal{Z}(s_t,a_t)} \frac{\eta_{-i}^\pi(h)}{\sum_{h'\in s_t}\eta_{-i}^\pi(h')}\eta^\pi(ha,z)u_i(z) \;=\; \frac{1}{\sum_{h\in s_t}\eta_{-i}^\pi(h)}v_i^c(\pi, s_t, a_t).
\end{aligned}
$$

The derivation is similar to show that $v_{\pi,i}(s_t) = v_i^c(\pi, s_t)/\sum_{h\in s_t}\eta_{-i}^\pi(h)$. Hence, counterfactual values and standard value functions are generally not equal, but are scaled by the Bayes normalizing constant $\mathcal{B}_{-i}(\pi, s_t) = \sum_{h \in s_t} \eta^{\pi}_{-i}(h)$. If there is a low probability of reaching s_t due to the environment or due to opponents' policies, these values will differ significantly.

This leads to a new interpretation of actor-critic algorithms in the multiagent partially observable setting: the advantage values $q_{\pi,i}(s_t, a_t) - v_{\pi,i}(s_t, a_t)$ are immediate counterfactual regrets scaled by $1/\mathcal{B}_{-i}(\pi, s_t)$. This then determines requirements for convergence guarantees in the tabular case.

Note that the standard policy gradient theorem holds: gradients can be estimated from samples. This follows from the derivation of the policy gradient in the tabular case (see Appendix E). When TD bootstrapping is not used, the Markov property is not required; having multiple agents and/or partial observability does not change this. For a proof using REINFORCE (G_t only), see [78, Theorem 1]. The proof trivially follows using $G_{t,i} - v_{\pi,i}$ since $v_{\pi,i}$ is trained separately and does not depend on ρ.

Policy gradient algorithms perform gradient ascent on $J^{PG}(\pi_{\boldsymbol{\theta}}) = v_{\pi_\theta}(s_0)$, using $\nabla_{\boldsymbol{\theta}} J^{PG}(\pi_{\boldsymbol{\theta}}) \propto \sum_s \mu(s) \sum_a \nabla_{\boldsymbol{\theta}} \pi_\theta(s,a) q_\pi(s,a)$, where μ is on-policy distribution under π [84, Section 13.2]. The actor-critic equivalent is $\nabla_{\boldsymbol{\theta}} J^{AC}(\pi_{\boldsymbol{\theta}}) \propto \sum_s \mu(s) \sum_a \nabla_{\boldsymbol{\theta}} \pi_\theta(s,a)(q_\pi(s,a) - \sum_b \pi(s,b) q_\pi(s,b))$. Note that the baseline is unnecessary when summing over the actions and $\nabla_{\boldsymbol{\theta}} J^{AC}(\pi_{\boldsymbol{\theta}}) = \nabla_{\boldsymbol{\theta}} J^{PG}(\pi_{\boldsymbol{\theta}})$ [5]. However, our analysis relies on a projected gradient descent algorithm that does not assume simplex constraints on the policy: in that case, in general $\nabla_{\boldsymbol{\theta}} J^{AC}(\pi_{\boldsymbol{\theta}}) \neq \nabla_{\boldsymbol{\theta}} J^{PG}(\pi_{\boldsymbol{\theta}})$.

Definition 1. *Define* ***policy gradient policy iteration*** *(PGPI) as a process that iteratively runs $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha \nabla_{\boldsymbol{\theta}} J^{PG}(\pi_{\boldsymbol{\theta}})$, and* ***actor-critic policy iteration*** *(ACPI) similarly using $\nabla_{\boldsymbol{\theta}} J^{AC}(\pi_{\boldsymbol{\theta}})$.*

In two-player zero-sum games, PGPI/ACPI are gradient ascent-descent problems, because each player is trying to ascend their own score function, and when using tabular policies a solution exists due to the minimax theorem [79]. Define player i's **external regret** over K steps as $R_i^K = \max_{\pi'_i \in \Pi_i} \left(\sum_{k=1}^K \mathbb{E}_{\pi'_i}[G_{0,i}] - \mathbb{E}_{\pi^k}[G_{0,i}] \right)$, where Π_i is the set of deterministic policies.

Theorem 1. *In two-player zero-sum games, when using tabular policies and an ℓ_2 projection $P(\boldsymbol{\theta}) = \operatorname{argmin}_{\boldsymbol{\theta}' \in \Delta(\mathcal{S},\mathcal{A})} \|\boldsymbol{\theta} - \boldsymbol{\theta}'\|_2$, where $\Delta(\mathcal{S},\mathcal{A}) = \{\boldsymbol{\theta} \mid \forall s \in \mathcal{S}, \sum_{b \in \mathcal{A}} \boldsymbol{\theta}_{s,b} = 1\}$ is the space of tabular simplices, if player i uses learning rates of $\alpha_{s,k} = k^{-\frac{1}{2}} \eta_i^{\pi^k}(s) \mathcal{B}_{-i}(\pi, s_t)$ at s on iteration k, and $\theta^k_{s,a} > 0$ for all k and s, then projected PGPI, $\theta^{k+1}_{s,\cdot} \leftarrow P(\{\theta^k_{s,a} + \alpha_{s,k} \frac{\partial}{\partial \theta^k_{s,a}} J^{PG}(\pi_{\boldsymbol{\theta}^k})\}_a)$, has regret $R_i^K \leq \frac{1}{\eta_i^{\min}} |\mathcal{S}_i| \left(\sqrt{K} + (\sqrt{K} - \frac{1}{2}) |\mathcal{A}| (\Delta r)^2 \right)$, where $\mathcal{S}_i$ is the set of player i's states, Δr is the reward range, and $\eta_i^{\min} = \min_{s,k} \eta_i^k(s)$. The same holds for projected ACPI (see appendix).*

The proof is given in Appendix E. In the case of sampled trajectories, as long as every state is reached with positive probability, Monte Carlo estimators of $q_{\pi,i}$ will be consistent. Therefore, we use exploratory policies and decay exploration over time. With a finite number of samples, the probability that an estimator $\hat{q}_{\pi,i}(s,a)$ differs by some quantity away from its mean is determined by Hoeffding's inequality and the reach probabilities. We suspect these errors could be accumulated to derive probabilistic regret bounds similar to the off-policy Monte Carlo case [46].

What happens in the sampling case with a fixed per-state learning rate α_s? If player i collects a batch of data from many sampled episodes and applies them all at once, then the *effective* learning rates (expected update rate relative to the other states) is scaled by the probability of reaching s: $\eta_i^\pi(s) \mathcal{B}_{-i}(\pi, s)$, which matches the value in the condition of Theorem 1. This suggests using a globally decaying learning rate to simulate the remaining $k^{-\frac{1}{2}}$.

The analysis so far has concentrated on establishing guarantees for the optimization problem that underlies standard formulation of policy gradient and actor-critic algorithms. A better guarantee can be achieved by using stronger policy improvement (proof and details are found in Appendix E):

Theorem 2. *Define a state-local $J^{PG}(\pi_{\boldsymbol{\theta}}, s) = v_{\pi_{\boldsymbol{\theta}},i}(s)$, composite gradient $\{\frac{\partial}{\partial \theta_{s,a}} J^{PG}(\pi_{\boldsymbol{\theta}}, s)\}_{s,a}$,* ***strong policy gradient policy iteration*** *(SPGPI), and* ***strong actor-critic policy iteration*** *(SACPI) as in Definition 1 except replacing the gradient components with $\frac{\partial}{\partial \theta_{s,a}} J^{PG}(\pi_{\boldsymbol{\theta}}, s)$. Then, in two-player zero-sum games, when using tabular policies and projection $P(\boldsymbol{\theta})$ as defined in Theorem 1 with learning rates $\alpha_k = k^{-\frac{1}{2}}$ on iteration k, projected SPGPI, $\theta^{k+1}_{s,\cdot} \leftarrow P(\{\theta^k_{s,a} + \alpha_k \frac{\partial}{\partial \theta^k_{s,a}} J^{PG}(\pi_{\boldsymbol{\theta}}, s)\}_a)$, has regret $R_i^K \leq |\mathcal{S}_i| \left(\sqrt{K} + (\sqrt{K} - \frac{1}{2}) |\mathcal{A}| (\Delta r)^2 \right)$, where $\mathcal{S}_i$ is the set of player i's states and Δr is the reward range. This also holds for projected SACPI (see appendix).*

4 Empirical Evaluation

We now assess the behavior of the actor-critic algorithms in practice. While the analyses in the previous section established guarantees for the tabular case, ultimately we want to assess scalability and generalization potential for larger settings. Our implementation parameterizes critics and policies using neural networks with two fully-connected layers of 128 units each, and rectified linear unit activation functions, followed by a linear layer to output a single value q or softmax layer to output π. We chose these architectures to remain consistent with previous evaluations [32, 50].

4.1 Domains: Kuhn and Leduc Poker

We evaluate the actor-critic algorithms on two n-player games: Kuhn poker, and Leduc poker.

Kuhn poker is a toy game where each player starts with 2 chips, antes 1 chip to play, and receives one card face down from a deck of size $n + 1$ (one card remains hidden). Players proceed by betting (raise/call) by adding their remaining chip to the pot, or passing (check/fold) until all players are either in (contributed as all other players to the pot) or out (folded, passed after a raise). The player with the highest-ranked card that has not folded wins the pot.

In **Leduc poker**, players have a limitless number of chips, and the deck has size $2(n + 1)$, divided into two suits of identically-ranked cards. There are two rounds of betting, and after the first round a single public card is revealed from the deck. Each player antes 1 chip to play, and the bets are limited to two per round, and number of chips limited to 2 in the first round, and 4 in the second round.

The rewards to each player is the number of chips they had after the game minus before the game. To remain consistent with other baselines, we use the form of Leduc described in [50] which does not restrict the action space, adding reward penalties if/when illegal moves are chosen.

4.2 Baseline: Neural Fictitious Self-Play

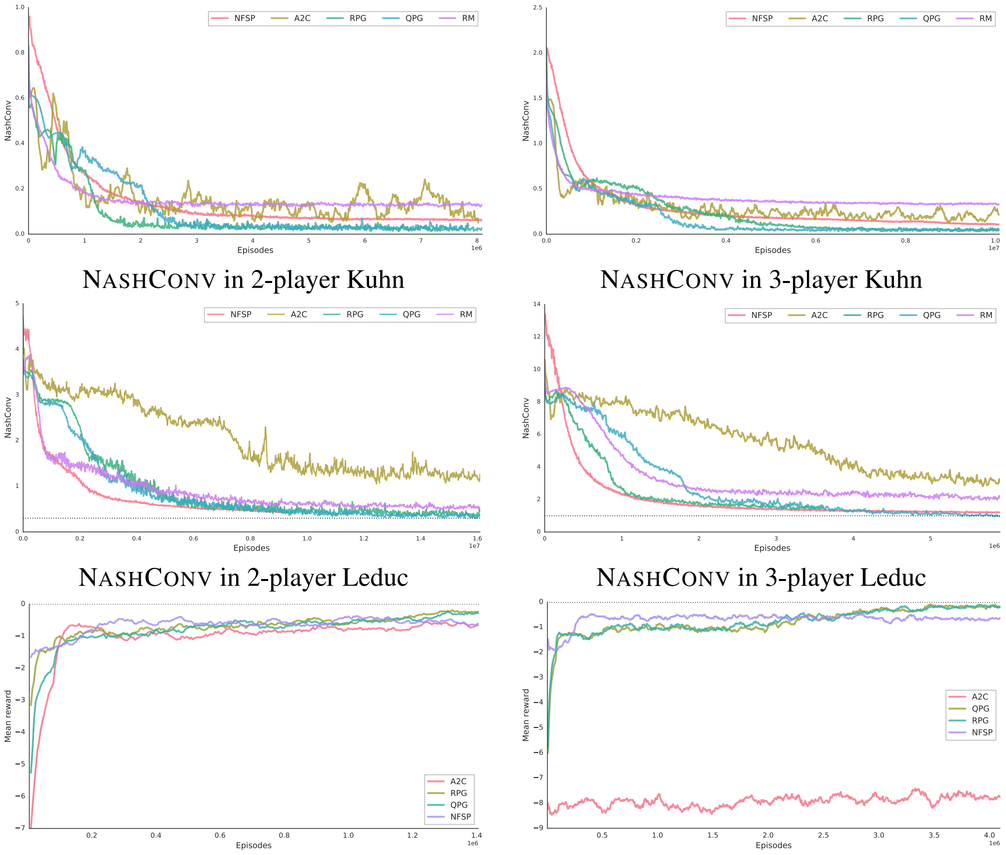
We compare to one main baseline. **Neural Fictitious Self-Play** (NFSP) is an implementation of fictitious play, where approximate best responses are used in place of full best response [32]. Two transition buffers of are used: $\mathcal{D}^{RL}$ and $\mathcal{D}^{ML}$; the former to train a DQN agent towards a best response π_i to $\bar{\pi}_{-i}$, data in the latter is replaced using reservoir sampling, and trains $\bar{\pi}_i$ by classification.

4.3 Main Performance Results

Here we show the empirical convergence to approximate Nash equlibria for each algorithm in self-play, and performance against fixed bots. The standard metric to use for this is $\textsc{NashConv}(\pi)$ defined in Section 2.2, which reports the accuracy of the approximation to a Nash equilibrium.

Training Setup. In the domains we tested, we observed that the variance in returns was high and hence we performed multiple policy evaluation updates (q-update for ∇^{QPG} , ∇^{RPG}, and ∇^{RMPG}, and v-update for A2C) followed by policy improvement (policy gradient update). These updates were done using separate SGD optimizers with their respective learning rates of fixed 0.001 for policy evaluation, and annealed from a starting learning rate to 0 over 20M steps for policy improvement. (See Appendix G for exact values). Further, the policy improvement step is applied after N_q policy evaluation updates. We treat N_q and batch size as a hyper parameters and sweep over a few reasonable values. In order to handle different scales of rewards in the multiple domains, we used the streaming Z-normalization on the rewards, inspired by its use in Proximal Policy Optimization (PPO) [77]. In addition, the agent's policy is controlled by a(n inverse) temperature added as part of the softmax operator. The temperature is annealed from 1 to 0 over 1M steps to ensure adequate state space coverage. An additional entropy cost hyper-parameter is added as is standard practice with Deep RL policy gradient methods such as A3C [59, 77]. For NFSP, we used the same values presented in [50].

Convergence to Equilibrium. See Figure 2 for convergence results. Please note that we plot the NASHCONV for the average policy in the case of NFSP, and the current policy in the case of the policy gradient algorithms. We see that in 2-player Leduc, the actor-critic variants we tried are similar in performance; NFSP has faster short-term convergence but long-term the actor critics are comparable. Each converges significantly faster than A2C. However RMPG seems to plateau.



NashConv in 2-player Kuhn

NashConv in 3-player Kuhn

NashConv in 2-player Leduc

NashConv in 3-player Leduc

2-player Leduc vs. CFR500

3-player Leduc vs CFR500

Figure 2: Empirical convergence rates for NashConv(π) and performance versus CFR agents.

Performance Against Fixed Bots. We also measure the expected reward against fixed bots, averaged over player seats. These bots, CFR500, correspond to the average policy after 500 iterations of CFR. QPG and RPG do well here, scoring higher than A2C and even beating NFSP in the long-term.

5 Conclusion

In this paper, we discuss several update rules for actor-critic algorithms in multiagent reinforcement learning. One key property of this class of algorithms is that they are model-free, leading to a purely online algorithm, independent of the opponents and environment. We show a connection between these algorithms and (counterfactual) regret minimization, leading to previously unknown convergence properties underlying model-free MARL in zero-sum games with imperfect information.

Our experiments show that these actor-critic algorithms converge to approximate Nash equilibria in commonly-used benchmark Poker domains with rates similar to or better than baseline model-free algorithms for zero-sum games. However, they may be easier to implement, and do not require storing a large memory of transitions. Furthermore, the current policy of some variants do significantly better than the baselines (including the average policy of NFSP) when evaluated against fixed bots. Of the actor-critic variants, RPG and QPG seem to outperform RMPG in our experiments.

As future work, we would like to formally develop the (probabilistic) guarantees of the sample-based on-policy Monte Carlo CFR algorithms and/or extend to continuing tasks as in MDPs [41]. We are also curious about what role the connections between actor-critic methods and CFR could play in deriving convergence guarantees in model-free MARL for cooperative and/or potential games.

Acknowledgments. We would like to thank Martin Schmid, Audrūnas Gruslys, Neil Burch, Noam Brown, Kevin Waugh, Rich Sutton, and Thore Graepel for their helpful feedback and support.

References

[1] Sherief Abdallah and Victor Lesser. A multiagent reinforcement learning algorithm with non-linear dynamics. *JAIR*, 33(1):521–549, 2008.

[2] Abbas Abdolmaleki, Jost Tobias Springenberg, Yuval Tassa, Nicolas Heess Remi Munos, and Martin Riedmiller. Maximum a posteriori policy optimisation. *CoRR*, abs/1806.06920, 2018.

[3] Maruan Al-Shedivat, Trapit Bansal, Yuri Burda, Ilya Sutskever, Igor Mordatch, and Pieter Abbeel. Continuous adaptation via meta-learning in nonstationary and competitive environments. In *Proceedings of the Sixth International Conference on Learning Representations*, 2018.

[4] Stefano V. Albrecht and Peter Stone. Autonomous agents modelling other agents: A comprehensive survey and open problems. *Artificial Intelligence*, 258:66–95, 2018.

[5] Cameron Allen, Melrose Roderick Kavosh Asadi, Abdel rahman Mohamed, George Konidaris, and Michael Littman. Mean actor critic. *CoRR*, abs/1709.00503, 2017.

[6] P. Auer, N. Cesa-Bianchi, Y. Freund, and R. E. Schapire. Gambling in a rigged casino: The adversarial multi-armed bandit problem. In *Proceedings of the 36th Annual Symposium on Foundations of Computer Science*, pages 322–331, 1995.

[7] Trapit Bansal, Jakub Pachocki, Szymon Sidor, Ilya Sutskever, and Igor Mordatch. Emergent complexity via multi-agent competition. In *Proceedings of the Sixth International Conference on Learning Representations*, 2018.

[8] Daan Bloembergen, Karl Tuyls, Daniel Hennes, and Michael Kaisers. Evolutionary dynamics of multi-agent learning: A survey. *J. Artif. Intell. Res. (JAIR)*, 53:659–697, 2015.

[9] A. Blum and Y. Mansour. Learning, regret minimization, and equilibria. In *Algorithmic Game Theory*, chapter 4. Cambridge University Press, 2007.

[10] Branislav Bošanský, Viliam Lisý, Marc Lanctot, Jiří Čermák, and Mark H.M. Winands. Algorithms for computing strategies in two-player simultaneous move games. *Artificial Intelligence*, 237:1—-40, 2016.

[11] Michael Bowling. Convergence and no-regret in multiagent learning. In *Advances in Neural Information Processing Systems 17 (NIPS)*, pages 209–216, 2005.

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

[13] Michael Bowling and Manuela Veloso. Multiagent learning using a variable learning rate. *Artificial Intelligence*, 136:215–250, 2002.

[14] G. W. Brown. Iterative solutions of games by fictitious play. In T.C. Koopmans, editor, *Activity Analysis of Production and Allocation*, pages 374–376. John Wiley & Sons, Inc., 1951.

[15] Noam Brown, Christian Kroer, and Tuomas Sandholm. Dynamic thresholding and pruning for regret minimization. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2017.

[16] Noam Brown and Tuomas Sandholm. Superhuman AI for heads-up no-limit poker: Libratus beats top professionals. *Science*, 360(6385), December 2017.

[17] L. Busoniu, R. Babuska, and B. De Schutter. A comprehensive survey of multiagent reinforcement learning. *IEEE Transaction on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 38(2):156–172, 2008.

[18] Kris Cao, Angeliki Lazaridou, Marc Lanctot, Joel Z Leibo, Karl Tuyls, and Stephen Clark. Emergent communication through negotiation. In *Proceedings of the Sixth International Conference on Learning Representations (ICLR)*, 2018.

[19] Kamil Ciosek and Shimon Whiteson. Expected policy gradients. In *Proceedings of the Thirty-Second AAAI conference on Artificial Intelligence (AAAI-18)*, 2018.

[20] Constantinos Daskalakis, Paul W. Goldberg, and Christos H. Papadimitriou. The complexity of computing a nash equilibrium. In *Proceedings of the Thirty-eighth Annual ACM Symposium on Theory of Computing*, STOC '06, pages 71–78, New York, NY, USA, 2006. ACM.

[21] Yan Duan, Xi Chen, Rein Houthooft, John Schulman, and Pieter Abbeel. Benchmarking deep reinforcement learning for continuous control. *CoRR*, abs/1604.06778, 2016.

[22] Lasse Espeholt, Hubert Soyer, Rémi Munos, Karen Simonyan, Volodymyr Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, Shane Legg, and Koray Kavukcuoglu. IMPALA: scalable distributed deep-rl with importance weighted actor-learner architectures. *CoRR*, abs/1802.01561, 2018.

[23] Jakob N. Foerster, Richard Y. Chen, Maruan Al-Shedivat, Shimon Whiteson, Pieter Abbeel, and Igor Mordatch. Learning with opponent-learning awareness. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2017.

[24] Jakob N. Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. Counterfactual multi-agent policy gradients. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, 2017.

[25] N. Gatti, F. Panozzo, and M. Restelli. Efficient evolutionary dynamics with extensive-form games. In *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence*, pages 335–341, 2013.

[26] Richard Gibson. Regret minimization in non-zero-sum games with applications to building champion multiplayer computer poker agents. *CoRR*, abs/1305.0034, 2013.

[27] Shixiang Gu, Ethan Holly, Timothy P. Lillicrap, and Sergey Levine. Deep reinforcement learning for robotic manipulation. *CoRR*, abs/1610.00633, 2016.

[28] S. Hart and A. Mas-Colell. A simple adaptive procedure leading to correlated equilibrium. *Econometrica*, 68(5):1127–1150, 2000.

[29] Elad Hazan. Introduction to online convex optimization. *Foundations and Trends in Optimization*, 2(3–4):157–325, 2015.

[30] He He, Jordan L. Boyd-Graber, Kevin Kwok, and Hal Daumé III. Opponent modeling in deep reinforcement learning. In *Proceedings of The 33rd International Conference on Machine Learning (ICML 2016)*, 2016.

[31] Johannes Heinrich, Marc Lanctot, and David Silver. Fictitious self-play in extensive-form games. In *Proceedings of the 32nd International Conference on Machine Learning (ICML 2015)*, 2015.

[32] Johannes Heinrich and David Silver. Deep reinforcement learning from self-play in imperfect-information games. *CoRR*, abs/1603.01121, 2016.

[33] Pablo Hernandez-Leal, Michael Kaisers, Tim Baarslag, and Enrique Munoz de Cote. A survey of learning in multiagent environments: Dealing with non-stationarity. *CoRR*, abs/1707.09183, 2017.

[34] Josef Hofbauer and Karl Sigmund. *Evolutionary Games and Population Dynamics*. Cambridge University Press, 1998.

[35] Josef Hofbauer, Sylvain Sorin, and Yannick Viossat. Time average replicator and best-reply dynamics. *Mathematics of Operations Research*, 34(2):263–269, 2009.

[36] Zhang-Wei Hong, Shih-Yang Su, Tzu-Yun Shann, Yi-Hsiang Chang, and Chun-Yi Lee. A deep policy inference q-network for multi-agent systems. *CoRR*, abs/1712.07893, 2017.

[37] Max Jaderberg, V. Mnih, W. M. Czarnecki, T. Schaul, J. Z. Leibo, D. Silver, and K Kavukcuoglu. Reinforcement learning with unsupervised auxiliary tasks. In *Proceedings of the International Conference on Representation Learning*, 2017.

[38] Peter H. Jin, Sergey Levine, and Kurt Keutzer. Regret minimization for partially observable deep reinforcement learning. *CoRR*, abs/1710.11424, 2017.

[39] Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101:99–134, 1998.

[40] Sham Kakade and John Langford. Approximately optimal approximate reinforcement learning. In *Proceedings of the Nineteenth International Conference on Machine Learning*, ICML '02, pages 267–274, San Francisco, CA, USA, 2002. Morgan Kaufmann Publishers Inc.

[41] Ian A. Kash and Katja Hoffman. Combining no-regret and Q-learning. In *European Workshop on Reinforcement Learning (EWRL) 14*, 2018.

[42] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[43] Vojtech Kovarík and Viliam Lisý. Analysis of hannan consistent selection for monte carlo tree search in simultaneous move games. *CoRR*, abs/1509.00149, 2015.

[44] H. W. Kuhn. Extensive games and the problem of information. *Contributions to the Theory of Games*, 2:193–216, 1953.

[45] Shapley L. Some topics in two-person games. In *Advances in Game Theory*. Princeton University Press., 1964.

[46] M. Lanctot, K. Waugh, M. Bowling, and M. Zinkevich. Sampling for regret minimization in extensive games. In *Advances in Neural Information Processing Systems (NIPS 2009)*, pages 1078–1086, 2009.

[47] Marc Lanctot. *Monte Carlo Sampling and Regret Minimization for Equilibrium Computation and Decision-Making in Large Extensive Form Games*. PhD thesis, Department of Computing Science, University of Alberta, Edmonton, Alberta, Canada, June 2013.

[48] Marc Lanctot. Further developments of extensive-form replicator dynamics using the sequence-form representation. In *Proceedings of the Thirteenth International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 1257–1264, 2014.

[49] Marc Lanctot, Kevin Waugh, Martin Zinkevich, and Michael Bowling. Monte Carlo sampling for regret minimization in extensive games. In Y. Bengio, D. Schuurmans, J. Lafferty, C. K. I. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems 22*, pages 1078–1086, 2009.

[50] Marc Lanctot, Vinicius Zambaldi, Audrunas Gruslys, Angeliki Lazaridou, Karl Tuyls, Julien Perolat, David Silver, and Thore Graepel. A unified game-theoretic approach to multiagent reinforcement learning. In *Advances in Neural Information Processing Systems*, 2017.

[51] Angeliki Lazaridou, Alexander Peysakhovich, and Marco Baroni. Multi-agent cooperation and the emergence of (natural) language. In *Proceedings of the International Conference on Learning Representations (ICLR)*, April 2017.

[52] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521:436–444, 2015.

[53] Adam Lerer and Alexander Peysakhovich. Maintaining cooperation in complex social dilemmas using deep reinforcement learning. *CoRR*, abs/1707.01068, 2017.

[54] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *CoRR*, abs/1509.02971, 2015.

[55] Michael L. Littman. Markov games as a framework for multi-agent reinforcement learning. In *In Proceedings of the Eleventh International Conference on Machine Learning*, pages 157–163. Morgan Kaufmann, 1994.

[56] Hao Liu, Yihao Feng, Yi Mao, Dengyong Zhou, Jian Peng, and Qiang Liu. Action-dependent control variates for policy optimization via stein identity. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2018.

[57] Ryan Lowe, YI WU, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 6379–6390. Curran Associates, Inc., 2017.

[58] L. Matignon, G. J. Laurent, and N. Le Fort-Piat. Independent reinforcement learners in cooperative Markov games: a survey regarding coordination problems. *The Knowledge Engineering Review*, 27(01):1–31, 2012.

[59] Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *Proceedings of the 33rd International Conference on Machine Learning (ICML)*, pages 1928–1937, 2016.

[60] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518:529–533, 2015.

[61] Matej Moravčík, Martin Schmid, Neil Burch, Viliam Lisý, Dustin Morrill, Nolan Bard, Trevor Davis, Kevin Waugh, Michael Johanson, and Michael Bowling. Deepstack: Expert-level artificial intelligence in heads-up no-limit poker. *Science*, 358(6362), October 2017.

[62] Todd W. Neller and Marc Lanctot. An introduction to counterfactual regret minimization. In *Proceedings of Model AI Assignments, The Fourth Symposium on Educational Advances in Artificial Intelligence (EAAI-2013)*, 2013. `http://modelai.gettysburg.edu/2013/cfr/index.html`.

[63] Duc Thien Nguyen, Akshat Kumar, and Hoong Chuin Lau. Policy gradient with value function approximation for collective multiagent planning. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 4319–4329. Curran Associates, Inc., 2017.

[64] A. Nowé, P. Vrancx, and Y-M. De Hauwere. Game theory and multi-agent reinforcement learning. In *Reinforcement Learning: State-of-the-Art*, chapter 14, pages 441–470. 2012.

[65] Frans A. Oliehoek and Christopher Amato. *A Concise Introduction to Decentralized POMDPs*. Springer, 2016.

[66] Fabio Panozzo, Nicola Gatti, and Marcello Restelli. Evolutionary dynamics of q-learning over the sequence form. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, pages 2034–2040, 2014.

[67] Xue Bin Peng, Pieter Abbeel, Sergey Levine, and Michiel van de Panne. Deepmimic: Example-guided deep reinforcement learning of physics-based character skills. *CoRR*, abs/1804.02717, 2018.

[68] Julien Perolat, Bilal Piot, and Olivier Pietquin. Actor-critic fictitious play in simultaneous move multistage games. In Amos Storkey and Fernando Perez-Cruz, editors, *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*, volume 84 of *Proceedings of Machine Learning Research*, pages 919–928, Playa Blanca, Lanzarote, Canary Islands, 09–11 Apr 2018. PMLR.

[69] Julien Pérolat, Bilal Piot, Bruno Scherrer, and Olivier Pietquin. On the use of non-stationary strategies for solving two-player zero-sum markov games. In *The 19th International Conference on Artificial Intelligence and Statistics (AISTATS 2016)*, 2016.

[70] Julien Pérolat, Bruno Scherrer, Bilal Piot, and Olivier Pietquin. Approximate dynamic programming for two-player zero-sum markov games. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2015.

[71] Jan Peters. Policy gradient methods for control applications. Technical Report TR-CLMC-2007-1, University of Southern California, 2002.

[72] Yu Qian, Fang Debin, Zhang Xiaoling, Jin Chen, and Ren Qiyu. Stochastic evolution dynamic of the rock–scissors–paper game based on a quasi birth and death process. *Scientific Reports*, 6(1):28585, 2016.

[73] Deirdre Quillen, Eric Jang, Ofir Nachum, Chelsea Finn, Julian Ibarz, and Sergey Levine. Deep reinforcement learning for vision-based robotic grasping: A simulated comparative evaluation of off-policy methods. *CoRR*, abs/1802.10264, 2018.

[74] N. A. Risk and D. Szafron. Using counterfactual regret minimization to create competitive multiplayer poker agents. In *Proceedings of the International Conference on Autonomus Agents and Multiagent Systems (AAMAS)*, pages 159–166, 2010.

[75] William H. Sandholm. *Population Games and Evolutionary Dynamics*. The MIT Press, 2010.

[76] John Schulman, Sergey Levine, Philipp Moritz, Michael I. Jordan, and Pieter Abbeel. Trust region policy optimization. *CoRR*, abs/1502.05477, 2015.

[77] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[78] Shai Shalev-Shwartz, Shaked Shammah, and Amnon Shashua. Safe, multi-agent, reinforcement learning for autonomous driving. *CoRR*, abs/1610.03295, 2016.

[79] Y. Shoham and K. Leyton-Brown. *Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations*. Cambridge University Press, 2009.

[80] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529:484—-489, 2016.

[81] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy P. Lillicrap, Karen Simonyan, and Demis Hassabis. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *CoRR*, abs/1712.01815, 2017.

[82] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy Lillicrap, Fan Hui, Laurent Sifre, George van den Driessche, Thore Graepel, and Demis Hassabis. Mastering the game of go without human knowledge. *Nature*, 530:354–359, 2017.

[83] Satinder P. Singh, Michael J. Kearns, and Yishay Mansour. Nash convergence of gradient dynamics in general-sum games. In *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence*, UAI '00, pages 541–548, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.

[84] R. Sutton and A. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 2nd edition, 2018.

[85] Richard S. Sutton, Satinder Singh, and David McAllester. Comparing policy-gradient algorithms, 2001. Unpublished.

[86] Oskari Tammelin, Neil Burch, Michael Johanson, and Michael Bowling. Solving heads-up limit Texas Hold'em. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence*, 2015.

[87] Taylor and Jonker. Evolutionarily stable strategies and game dynamics. *Mathematical Biosciences*, 40:145–156, 1978.

[88] Karl Tuyls, Julien Perolat, Marc Lanctot, Joel Z Leibo, and Thore Graepel. A Generalised Method for Empirical Game Theoretic Analysis . In *AAMAS*, 2018.

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

[90] W. E. Walsh, D. C. Parkes, and R. Das. Choosing samples to compute heuristic-strategy Nash equilibrium. In *Proceedings of the Fifth Workshop on Agent-Mediated Electronic Commerce*, 2003.

[91] William E Walsh, Rajarshi Das, Gerald Tesauro, and Jeffrey O Kephart. Analyzing Complex Strategic Interactions in Multi-Agent Systems. In *AAAI*, 2002.

[92] Kevin Waugh, Dustin Morrill, J. Andrew Bagnell, and Michael Bowling. Solving games with functional regret estimation. In *Proceedongs of the AAAI Conference on Artificial Intelligence*, 2015.

[93] Michael P. Wellman. Methods for empirical game-theoretic analysis. In *Proceedings, The Twenty-First National Conference on Artificial Intelligence and the Eighteenth Innovative Applications of Artificial Intelligence Conference*, pages 1552–1556, 2006.

[94] R.J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3):229–256, 1992.

[95] Cathy Wu, Aravind Rajeswaran, Yan Duan, Vikash Kumar, Alexandre M. Bayen, Sham Kakade, Igor Mordatch, and Pieter Abbeel. Variance reduction for policy gradient with action-dependent factorized baselines. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2018.

[96] Yuxin Wu and Yuandong Tian. Training agent for first-person shooter game with actor-critic curriculum learning. In *Proceedings of the International Conference on Representation Learning*, 2017.

[97] Michael Wunder, Michael Littman, and Monica Babes. Classes of multiagent q-learning dynamics with ϵ-greedy exploration. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, pages 1167–1174, 2010.

[98] Chongjie Zhang and Victor Lesser. Multi-agent learning with policy prediction. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence*, pages 927–934, 2010.

[99] M. Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. In *Proceedings of Twentieth International Conference on Machine Learning (ICML-2003)*, 2003.

[100] M. Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. Technical Report CMU-CS-03-110, Carnegie Mellon University, 2003.

[101] M. Zinkevich, M. Johanson, M. Bowling, and C. Piccione. Regret minimization in games with incomplete information. In *Advances in Neural Information Processing Systems 20 (NIPS 2007)*, 2008.

[102] Martin Zinkevich, Amy Greenwald, and Michael L. Littman. Cyclic equilibria in markov games. In *Proceedings of the 18th International Conference on Neural Information Processing Systems*, NIPS'05, pages 1641–1648, Cambridge, MA, USA, 2005. MIT Press.

Appendices

A Some Notes on Notation and Terminology

Here we clarify some notational differences between the work on computational game theory and (multiagent) reinforcement learning.

There are some analogues between approximate dynamic programming and RL to counterfactual regret minimization in zero-sum games.

CFR is a policy iteration technique: it implements generalized policy iteration: policy evaluation computes the (counterfactual) values v^c_π. Policy "improvement" is implemented by the regret minimizers at each information state, such as regret-matching which yields the next policy $\pi_{k+1}(s)$ by assigning probabilities to each action $\pi_{k+1}(s, a)$ proportional to its thresholded cumulative regret $\text{TCREG}_i(k, \pi, a)$. There is one main difference: this improvement step is not (necessarily) a contraction on any distance to an optimal policy. However, the average policy $\bar{\pi}_k$ *does* converges due to the Folk theorem, so in some sense the policy update operator on π_k is improving $\bar{\pi}_k$. We give more detail on CFR in the following subsection (Appendix B).

Like standard value/policy iteration, CFR requires a full state space sweep each iteration. Intead, Monte Carlo sampling can be applied to get estimated values $\tilde{v}^c$ [49]. Then the equivalent policy update operator can be applied and there are probabilistic bounds on convergence to equilibrium.

One main crticial point is that temporal difference bootstrapping from values recursively is not possible as the Markov property does not hold in general for information states in partially-observable multiagent environments: the optimal policy $\pi_i(s, a)$ at some state s_t *does* generally depend on the policies at other information states.

POMDPs represent hidden state using belief states. They are different from information states, as they are paired with an associated distribution over the histories.

The following table shows a mapping between most-used terms that are analogous (mostly equivalent) but used within the two separate communities:

Computational Game Theory	Reinforcement Learning	This paper	Prev. paper(s)
Player	Agent	i	i
Information set	Information state	s	I
Action (or move)	Action	a	a
History	State	h	h
Utility	Reward	u, G	u
Strategy	Policy	π	σ
Reach probability	[4]	η	π
Chance event probability	Transition probability	$\mathcal{T}$	σ_c
Chance	Nature		
Imperfect Information	Partial Observability		
Extensive-form game	Multiagent Environment[5]		
Simultaneous-move/Stochastic Game	Markov/Stochastic Game		

Table 1: A mapping of analogous terms across fields. The last two columns show nomenclature used for instances of each, compared to the previous papers from computational game theory.

B Counterfactual Regret Minimization

As mentioned above, Counterfactial Regret Minimization (CFR) is a policy iteration technique with a different policy improvement step. In this section we describe the algorithm using the terminology as defined in this paper. Again, it is presented in a slightly different way than from previous papers to

[4] There is no precise equivalent. The closest is the on-policy distribution in episodic tasks $\mu(s)$ described in [84, Section 9.2].

[5] Also: finite-horizon Dec-POMDP, in the cooperative setting.

emphasize the elements of policy iteration. For an overview with more background, see [47, Chapter 2]. For a thorough introductory tutorial, including backgound and code exercises see [62].

C Regret-based Policy Gradients: Algorithm Pseudo-Code

The algorithm is similar in form to A2C [59]. The differences are:

1. Gradient *descent* is used with $\nabla^{\text{RPG}}_{\boldsymbol{\theta}}$ instead of gradient ascent in $\nabla^{\text{QPG}}_{\boldsymbol{\theta}}$, $\nabla^{\text{RMPG}}_{\boldsymbol{\theta}}$, and A2C.
2. An (action,value) q-function critic is used in place of the usual state baseline v.

The pseudo-code is presented in Algorithm 2.

In this paper we focus on episodes of bounded length and t_{max} is greater than the maximum number of moves per episode. So there is no TD-style bootstrapping from other values. In environments with longer episodes, it might be necessary to truncate the sequence lengths as is common in Deep RL.

D Analysis of Regret Dynamics in Matrix Games

In Tables 2, 3, and 4 we show the three games under study, i.e., matching pennies (MP), rock-paper-scissors (RPS), and a skewed version of the latter, called bias rock-paper-scissors (bRPS) from [10]. In Figures 3, 4 and 5 we illustrate several dynamics in these respective games. More precisely, we show classical Replicator Dynamics (RD) as a reference point (row a), RPG Dynamics (row b), and time average RPG Dynamics plots (row c), sorted row by row. As can be observed from the figures, and as is well known, the RD cycle around the mixed Nash equilibrium (indicated by a yellow dot) in all three games, see row (a). The RPG dynamics cycle as well, though in a slightly different manner than RD as can be seen from row (b). Finally, in row (c) we show the average time RPG dynamics. Interestingly these plots show that in all three cases the learning dynamics converge to the mixed Nash equilibrium. These final plots illustrate that the average intended behavior of RPG converges to the mixed Nash equilibrium and that the RPG algorithm is regret-minimizing in these specific normal-form games [35].

D.1 Normal Form Games Dynamical Systems

Here we present the dynamical systems that describe each policy gradient update rule in two-player matrix games. For futher detail on the construction and analysis of these dynamical systems, see [8, 34].

Let us recall the updated we consider in this paper.

QPG:

$$\nabla^{\text{QPG}}_{\boldsymbol{\theta}}(s) = \sum_a [\nabla_\theta \pi(s,a;\boldsymbol{\theta})] \left(q(s,a;\mathbf{w}) - \sum_b \pi(s,b;\boldsymbol{\theta}) q(s,b,\mathbf{w}) \right). \tag{4}$$

RPG:

$$\nabla^{\text{RPG}}_{\boldsymbol{\theta}}(s) = -\sum_a \nabla_\theta \left(q(s,a;\mathbf{w}) - \sum_b \pi(s,b;\boldsymbol{\theta}) q(s,b;\mathbf{w}) \right)^+. \tag{5}$$

RMPG:

$$\nabla^{\text{RMPG}}_{\boldsymbol{\theta}}(s) = \sum_a [\nabla_\theta \pi(s,a;\boldsymbol{\theta})] \left(q(s,a;\mathbf{w}) - \sum_b \pi(s,b;\boldsymbol{\theta}) q(s,b,\mathbf{w}) \right)^+. \tag{6}$$

Let us consider that the game is in normal form and let us suppose that the policy is only parametrized by logits. The parameter will be $\boldsymbol{\theta} = (\theta_a)_a$ and $\pi(a;\boldsymbol{\theta}) = \frac{\exp(\theta_a)}{\sum_a \exp(\theta_a)}$ in a state less game. It follows that:

$$\frac{d\pi(a;\boldsymbol{\theta})}{d\theta_b} = 1_{a=b}\pi(a;\boldsymbol{\theta}) - \pi(a;\boldsymbol{\theta})\pi(b;\boldsymbol{\theta}) \tag{7}$$

Algorithm 1: Vanilla CFR

```
input  :  $K$ – number of iterations; $\pi^0$ – initial uniform joint policy
Initialize table of values $v_i^c(\pi, s, a) = v_i^c(\pi, s) = 0$ for all $s, a$
Initialize cumulative regret tables CREG$(s, a) = 0$ for all $s, a$
Initialize average policy tables $S(s, a) = 0$ for all $s, a$

PolicyEvalTreeWalk(joint policy $\pi$, history $h$, player reach probs $\vec{\eta}$, chance reach $\eta_c$):
if $h$ is terminal then
    return utilites (episode returns) $\vec{u} = (u_i(h))$ for $i \in \mathcal{N}$
else if $h$ is a chance node then
    return $\sum_{a \in \mathcal{A}(h)} \Pr(ha|h) \cdot \textsc{PolicyEvalTreeWalk}(ha, i, \vec{\eta}, \Pr(ha|h) \cdot \eta_c)$
else
    Let $i$ be the player to play at $h$
    Let $s$ be the information state containing $h$
    $\vec{u} \leftarrow \vec{0}$
    for legal actions $a \in \mathcal{A}(h)$ do
        $\eta_{-i} \leftarrow \eta_c \cdot \Pi_{j \neq i} \eta_j$
        $\vec{\eta}' \leftarrow \vec{\eta}$
        $\eta_i' \leftarrow \eta_i' \cdot \pi_i(s, a)$
        $\vec{u}_a \leftarrow \textsc{PolicyEvalTreeWalk}(ha, \vec{\eta}', \eta_c)$
        $v_i^c(\pi, s, a) \leftarrow v_i^c(\pi, s, a) + u_{a,i}$     ($i^{th}$ component)
        $S(s, a) \leftarrow S(s, a) + \eta_i \cdot \pi(s, a)$     (policy improvement on average policy $\bar{\pi}$)
        $\vec{u} \leftarrow \vec{u} + \pi(s, a)\vec{u}_a$
    return $\vec{u}$

PolicyEvaluation(iteration $k$):
Let $h$ be the initial empty history
PolicyEvalTreeWalk($\pi^k$, $h$, $\vec{1}$, $\vec{1}$)
for all $s$ do
    Let $i$ be the player to play at $s$
    $v_i^c(\pi^k, s) = \sum_b \pi^k(s, b) v_i^c(\pi^k, s, b)$

RegretMatching(information state $s$):
Define thresholded cumulative TCREG$(s, a) = ($CREG$(s, a))^+$
$d \leftarrow \sum_b$ TCREG$(s, b)$
for $a \in \mathcal{A}(s)$ do
    $\pi(s, a) \leftarrow \frac{\text{TCREG}(s,a)}{d}$ if $d > 0$ otherwise $\frac{1}{|\mathcal{A}(s)|}$
return $\pi(s)$

PolicyUpdate(iteration $k$):
for all $s$ do
    Let $i$ be the player to play at $s$
    for $a \in \mathcal{A}(s)$ do
        CREG$(s, a) \leftarrow$ CREG$(s, a) + (v_i^c(\pi^k, s, a) - v_i^c(\pi^k, s))$
    $\pi^{k+1}(s) \leftarrow$ RegretMatching$(s)$

for $k \in \{1, 2, \cdots, K\}$ do
    Set all the counterfactual values $v_i^c(\pi^k, s, a) = v_i^c(\pi^k, s) = 0$
    PolicyEvaluation($k$)
    PolicyUpdate($k$)
for all $s$ do
    $\bar{\pi}^T(s, a) = \frac{S(s,a)}{\sum_b S(s,b)}$
return $\bar{\pi}^T$
```

Algorithm 2: Generalized Advantage Actor-Critic with (state,action) critics.

```
input  : $\pi$ – policy; $s_0$ – initial state
repeat
    Reset gradients: $d\boldsymbol{\theta} \leftarrow 0$, and $d\mathbf{w} \leftarrow 0$.
    $t_{start} \leftarrow t$
    repeat
        Sample $a_t \sim \pi(\cdot \mid s_t, \boldsymbol{\theta})$
        Take action $a_t$ and receive reward $r_t$ and $s_{t+1}$
        $t \leftarrow t + 1$
        $T \leftarrow T + 1$
    until terminal $s_t$ or $t - t_{start} = t_{max}$
    $G \leftarrow \begin{cases} 0 & \text{if } s_t \text{ is terminal;} \\ \sum_{a \in A} \pi(a \mid s_t, \boldsymbol{\theta}) Q(s, a; \mathbf{w}) & \text{otherwise.} \end{cases}$
    for $i \in \{t - 1, \dots, t_{start}\}$ do
        $G \leftarrow r_i + \gamma G$
        Acc. policy gradients: $d\boldsymbol{\theta} \leftarrow d\boldsymbol{\theta} + \delta$, where $\delta$ is one of $\{\nabla_{\boldsymbol{\theta}}^{\text{QPG}}, \nabla_{\boldsymbol{\theta}}^{\text{RPG}}, \nabla_{\boldsymbol{\theta}}^{\text{RMPG}}\}$ from Sec. 3
        Acc. $q$-value function gradients: $d\mathbf{w} \leftarrow d\mathbf{w} + \nabla_{\mathbf{w}}(G - q(s_i, a_i; \mathbf{w}))^2$
    Update critic: $\mathbf{w} \leftarrow \mathbf{w} - \alpha d\mathbf{w}$
    Update actor: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha d\boldsymbol{\theta}$
until $T > T_{max}$
```

$$\dot{\pi}(a;\boldsymbol{\theta}) = \sum_b \frac{d\pi(a;\boldsymbol{\theta})}{d\theta_b} \dot{\theta}_b = \pi(a;\boldsymbol{\theta})(\dot{\theta}_a - \sum_b \pi(b;\boldsymbol{\theta})\dot{\theta}_b) \tag{8}$$

D.1.1 QPG

The dynamical system followed by QPG on a normal form game can be written as follow:

$$\dot{\boldsymbol{\theta}} = \sum_a [\nabla_\theta \pi(a;\boldsymbol{\theta})] \left(q(a;\mathbf{w}) - \sum_b \pi(b;\boldsymbol{\theta}) q(b, \mathbf{w}) \right) \tag{9}$$

$$\dot{\boldsymbol{\theta}} = \sum_a \nabla_\theta \pi(a;\boldsymbol{\theta}) q(a;\mathbf{w}) \textit{ because} \sum_a \pi(a;\boldsymbol{\theta}) = 1 \tag{10}$$

$$\dot{\theta}_b = \sum_a \frac{d\pi(a;\boldsymbol{\theta})}{d\theta_b} q(a;\mathbf{w}) = \pi(b;\boldsymbol{\theta}) \left(q(b;\mathbf{w}) - \sum_a \pi(a;\boldsymbol{\theta}) q(a;\mathbf{w}) \right) = \pi(b;\boldsymbol{\theta}) A(b, \boldsymbol{\theta}, \mathbf{w}) \tag{11}$$

Final dynamical system:

$$\dot{\pi}(a;\boldsymbol{\theta}) = \pi(a;\boldsymbol{\theta}) \left(\pi(a;\boldsymbol{\theta}) A(a, \boldsymbol{\theta}, \mathbf{w}) - \sum_b \pi(b;\boldsymbol{\theta})^2 A(b, \boldsymbol{\theta}, \mathbf{w}) \right) \tag{12}$$

$$\textit{where } A(a, \boldsymbol{\theta}, \mathbf{w}) = q(a;\mathbf{w}) - \sum_b \pi(b;\boldsymbol{\theta}) q(b;\mathbf{w}) \tag{13}$$

D.1.2 RPG

$$\nabla_{\boldsymbol{\theta}}^{\text{RPG}}(s) = -\sum_a \nabla_\theta \left(q(s, a;\mathbf{w}) - \sum_b \pi(s, b;\boldsymbol{\theta}) q(s, b;\mathbf{w}) \right)^+ \tag{14}$$

$$\nabla_{\boldsymbol{\theta}}^{\text{RPG}}(s) = -\sum_a \nabla_\theta 1_{A(a,\boldsymbol{\theta},\mathbf{w}) \geq 0} \left(q(s, a;\mathbf{w}) - \sum_b \pi(s, b;\boldsymbol{\theta}) q(s, b;\mathbf{w}) \right) \tag{15}$$

$$\nabla_{\boldsymbol{\theta}}^{\text{RPG}}(s) = -\sum_a 1_{A(a,\boldsymbol{\theta},\mathbf{w})\geq 0}\nabla_\theta \left(q(s,a;\mathbf{w}) - \sum_b \pi(s,b;\boldsymbol{\theta})q(s,b;\mathbf{w})\right) \tag{16}$$

$$\nabla_{\boldsymbol{\theta}}^{\text{RPG}}(s) = \sum_a 1_{A(a,\boldsymbol{\theta},\mathbf{w})\geq 0} \sum_b \nabla_\theta \pi(b;\boldsymbol{\theta})q(b;\mathbf{w}) \tag{17}$$

$$\nabla_{\boldsymbol{\theta}}^{\text{RPG}}(s) = \underbrace{\left(\sum_a 1_{A(a,\boldsymbol{\theta},\mathbf{w})\geq 0}\right)}_{n_{a+}} \underbrace{\sum_b \nabla_\theta \pi(b;\boldsymbol{\theta})q(b;\mathbf{w})}_{\nabla_{\boldsymbol{\theta}}^{\text{QPG}} \text{ from 10}} \tag{18}$$

$$\nabla_{\boldsymbol{\theta}}^{\text{RPG}}(s) = n_{a+}\nabla_{\boldsymbol{\theta}}^{\text{QPG}} \tag{19}$$

it falls that the dynamical system of RPG is:

$$\dot{\pi}(a;\boldsymbol{\theta}) = n_{a+}\left[\pi(a;\boldsymbol{\theta})\left(\pi(a;\boldsymbol{\theta})A(a,\boldsymbol{\theta},\mathbf{w}) - \sum_b \pi(b;\boldsymbol{\theta})^2 A(b,\boldsymbol{\theta},\mathbf{w})\right)\right] \tag{20}$$

	H	T
H	$1,-1$	$-1,1$
T	$-1,1$	$1,-1$

Table 2: Matching Pennies.

	R	P	S
R	0	-1	1
P	1	0	-1
S	-1	1	0

Table 3: Rock-Paper-Scissors.

	R	P	S
R	0	-0.25	0.5
P	0.25	0	-0.05
S	-0.5	0.05	0

Table 4: Bias RPS.

D.2 Generalised Rock-Paper-Scissors Game

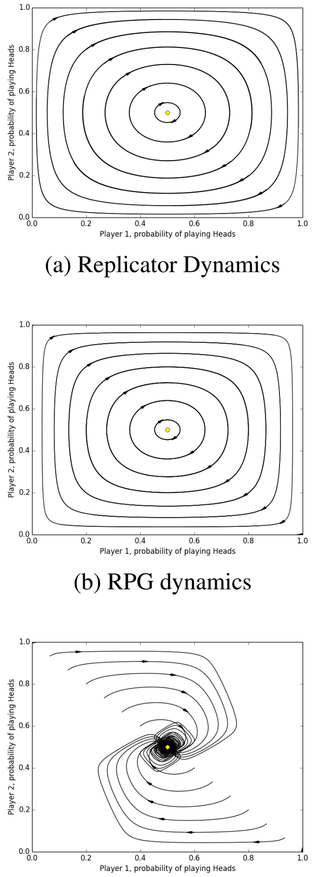
For the sake of completeness we also looked at the behavior of the dynamics in the generalised Rock-Paper-Scissors (gRPS) game [72, 45]. More precisely, the gRPS game can be described as illustrated in Table 5.

	R	P	S
R	1	0	2
P	2	1	0
S	0	2	1

Table 5: Generalized Rock-Paper-Scissors.

We describe the dynamics in this game for replicator dynamics, RPG dynamics, and both replicator and RPG dynamics as average time dynamics plots. As in the RPS game, the replicator dyanmis and RPG dynamics cycle around the Nash equilibrium, and the average time replicator dynamics and average time RPG dynamics converge to the Nash equilibrium, as illustrated in Figures 6, 7, 8 and 9. A more detailed description on the convergence properties of replicator equations in this game can be found in [72].

Figure 3: Matching Pennies

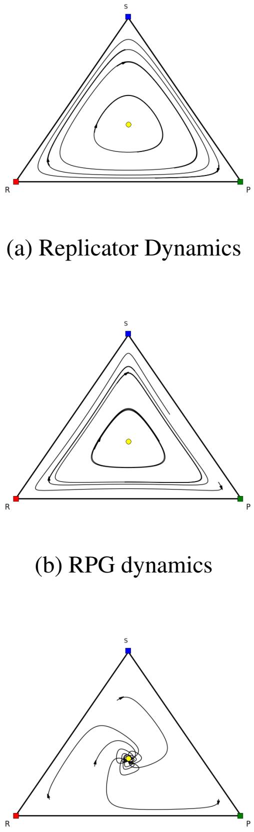


(a) Replicator Dynamics

(b) RPG dynamics

(c) Average RPG dynamics

Figure 4: Rock-Paper-Scissors

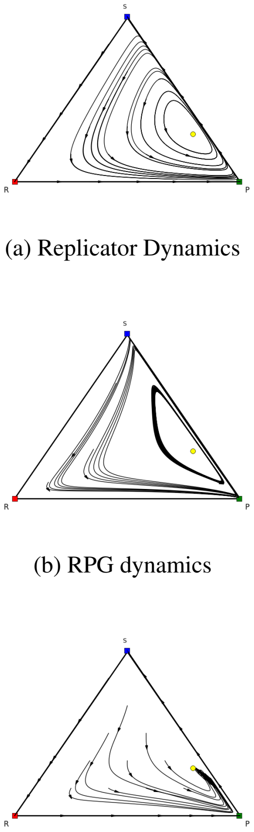


(a) Replicator Dynamics

(b) RPG dynamics

(c) Average RPG dynamics

Figure 5: Bias Rock-Paper-Scissors



(a) Replicator Dynamics

(b) RPG dynamics

(c) Average RPG dynamics

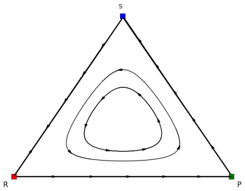


Figure 6: Replicator Dynamics in the gRPS Game

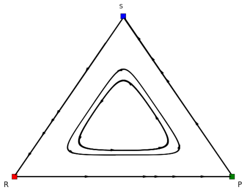


Figure 7: RPG dynamics in the gRPS Game

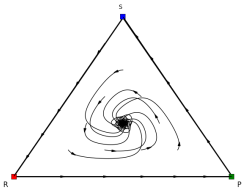


Figure 8: Avg. Replicator Dynamics in the gRPS Game

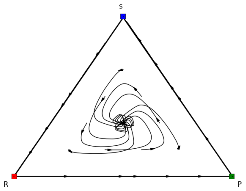


Figure 9: Avg. RPG Dynamics in the gRPS Game

E Sequential Partially-Observable Case

Let P be defined as in Theorem 1. We first define the four update rules that we will discuss in this section. On iteration k, at state s, the update to the policy parameters are:

$$\text{Projected PGPI}: \theta_{s,\cdot}^{k+1} \leftarrow P(\{\theta_{s,a}^k + \alpha_{s,k}\frac{\partial}{\partial \theta_{s,a}^k} J^{PG}(\pi_{\boldsymbol{\theta}^k})\}_a)$$

$$\text{Projected ACPI}: \theta_{s,\cdot}^{k+1} \leftarrow P(\{\theta_{s,a}^k + \alpha_{s,k}\frac{\partial}{\partial \theta_{s,a}^k} J^{AC}(\pi_{\boldsymbol{\theta}^k})\}_a)$$

$$\text{Projected Strong PGPI}: \theta_{s,\cdot}^{k+1} \leftarrow P(\{\theta_{s,a}^k + \alpha_{k}\frac{\partial}{\partial \theta_{s,a}^k} J^{PG}(\pi_{\boldsymbol{\theta}^k}, s)\}_a)$$

$$\text{Projected Strong ACPI}: \theta_{s,\cdot}^{k+1} \leftarrow P(\{\theta_{s,a}^k + \alpha_{k}\frac{\partial}{\partial \theta_{s,a}^k} J^{AC}(\pi_{\boldsymbol{\theta}^k}, s)\}_a)$$

Tabular policies are represents in behavioral strategy form: a probability is a weight $\theta_{s,a}$ per state-action, where the weights obey simplex constraints: $\forall s, \sum_a \theta_{s,a} = 1$.

In turn-based games, the gradient of a tabular policy, $\nabla_{\boldsymbol{\theta}} \pi_\theta$ is then simply a sum of partial derivates with respect to each specific weight $\theta_{s,a}$.

The score function $J^{PG}(\pi_{\boldsymbol{\theta}}) = \mathbb{E}_{\rho \sim \pi}[G_0 \mid S_0 = s_0] = \sum_{z \in \mathcal{Z}} \eta^\pi(z) G_{i,z}$. The contribution of some $\theta_{s,a}$ to the gradient is:

$$\begin{aligned}
\frac{\partial J^{PG}(\pi_\theta)}{\partial \theta_{s,a}} &= \frac{\partial}{\partial \theta_{s,a}} \sum_{z \in \mathcal{Z}} \eta^\pi(z) G_{i,z} \\
&= \frac{\partial}{\partial \theta_{s,a}} \left(\sum_{h, z \in \mathcal{Z}(s,a)} \eta^\pi(z) G_{i,z} \right) \qquad \text{since other terminal histories do not contain } \theta_{s,a} \\
&= \sum_{h, z \in \mathcal{Z}(s,a)} \frac{\partial}{\partial \theta_{s,a}} \eta^\pi(z) G_{i,z} \\
&= \sum_{h, z \in \mathcal{Z}(s,a)} \eta^\pi(h) \eta^\pi(ha, z) G_{i,z} \\
&:= v_{\eta,\pi}(s,a) \quad \text{(definition)}
\end{aligned}$$

Here, we define $v_{\eta,\pi}(s,a)$ as the (reach-weighted) portion of the overall expected value contributed by action a at information state s. The weight $\sum_{h,z \in \mathcal{Z}(s,a)} \eta^\pi(h)$ is analogous to the on-policy distribution μ in the standard policy gradient theorem. Each component of the gradient treats the other $\theta_{s',a'}$ as constant, increasing the local expected value contributed at s, which is just $\pi(s, \mathbf{a}) \cdot v_{\eta,\pi}(s, \mathbf{a})$, can be optimized independently for the purposes of taking a single gradient step. The result is that the problem can be decomposed into a per-state optimization problem, for the purposes of a single policy gradient update. This is a direct consequence of the tabular representation and perfect recall.

We then observe that ACPI can implement some form of Generalized Infinitesimal Gradient Ascent (GIGA) algorithm of Zinkevich [99], an application of greedy projection or now also called online gradient descent [29]. The idea here is that there is an online convex program with a convex cost function c_k at each step k. The optimization proceeds by moving the point x (*i.e.* the policy) following the gradient of c_k at at step k and projecting back into the feasible set (of simplices) greedily after each gradient step. GIGA is an application of online gradient descent to repeated games; Zinkevich shows that GIGA minimizes regret against this adversary, defining a new OCP after each play.

In our case, we have *local online convex programs* OCP(s,k), one at each s, and a separate instantiation of GIGA(s) at s that solves this local OCP at each s. Each problem is locally convex, and the adversary is the policy π_{-i} for states outside of s. We use this as a basis to prove Theorem 1.

Our construction essentially shows that ACPI is equivalent to CFR except with the policy update rule (RegretMatching on Algorithm 1, line 43) replaced by GIGA(s). We then show that PGPI can be treated as a special case of this overall argument.

Definition 2. *State-local Generalized Infinitesimal Gradient Ascent (*$\mathrm{GIGA}(s)$*) (an adaptation of [99, Algorithm 2] proceeds as follows. Let* $\boldsymbol{\theta}_s = (\theta_{s,a'}$ *for* $a' \in \mathcal{A}$ *) be the policy parameters at s. Initialize* $\boldsymbol{\theta}_s$ *arbitrarily according to simplex constraints. Choose a sequence of learning rates* $\{\alpha_1, \alpha_2, \cdots, \alpha_K\}$*, and repeat for* $k \in \{1, 2, \cdots, K\}$*:*

1. *Choose* a *according to* $\pi^k(s)$
2. *Observe* π_{-i}*, and update the local policy* $\pi^k(s)$*:*
   (a) $y^{k+1} = \boldsymbol{\theta}_s + \alpha_{s,k}\mathbf{v}_{\eta,\pi}(s,\cdot)$
   (b) $\boldsymbol{\theta}_s = P(y^{t+1})$,

where the projection P *is defined as in Theorem 1.*

Note that $v_{\eta,\pi}(s)$ here is the gradient wrt $\theta_{s,a}$ of the score function from above. To prove Theorem 1, we will make use of a few lemmas.

Lemma 1. *The value of the local component* $\frac{\partial J^{PG}(\pi_\theta)}{\partial \theta_{s,a}} = v_{\eta,\pi}(s,a) = \eta_i^\pi(s) v_i^c(\pi, s, a)$.

Proof.

$$\begin{aligned}
v_{\eta,\pi}(s,a) &= \sum_{h,z \in \mathcal{Z}(s,a)} \eta^\pi(h)\eta^\pi(ha,z)G_{i,z} && \text{as defined above} \\
&= \sum_{h,z \in \mathcal{Z}(s,a)} \eta_i^\pi(h)\eta_{-i}^\pi(h)\eta^\pi(ha,z)G_{i,z} \\
&= \eta_i^\pi(s) \sum_{h,z \in \mathcal{Z}(s,a)} \eta_{-i}^\pi(h)\eta^\pi(ha,z)G_{i,z} && \text{from perfect recall} \\
&= \eta_i^\pi(s) v_i^c(\pi, s, a)
\end{aligned}$$

□

The following lemma shows how the advantage is related to the local value $\mathrm{GIGA}(s)$ has in its update rule:

Lemma 2. *The advantage*

$$q_\pi(s,a) - \sum_b \pi(s,b) q_\pi(s,b) = \frac{v_{\eta,\pi}(s,a)}{\eta_i^\pi(s)\mathcal{B}_{-i}(\pi,s)} - \frac{v_i^c(\pi,s)}{\mathcal{B}_{-i}(\pi,s)}.$$

Proof.

$$\begin{aligned}
q_\pi(s,a) - \sum_b \pi(s,a) q_\pi(s,b) &= \frac{v_i^c(\pi,s,a) - v_i^c(\pi,s)}{\mathcal{B}_{-i}(\pi,s)} && \text{from Section 3} \\
&= \frac{v_i^c(\pi,s,a)}{\mathcal{B}_{-i}(\pi,s)} - \frac{v_i^c(\pi,s)}{\mathcal{B}_{-i}(\pi,s)} \\
&= \frac{v_{\eta,\pi}(s,a)}{\eta_i^\pi(s)\mathcal{B}_{-i}(\pi,s)} - \frac{v_i^c(\pi,s)}{\mathcal{B}_{-i}(\pi,s)} && \text{by Lemma 1}
\end{aligned}$$

□

We require one more property about projections onto simplices:

Lemma 3. *Define the simplex* $\Delta = \{\mathbf{x} \in \Re^N : \sum_i x_i = 1\}$*, and for some* $\mathbf{y} \in \Re^N$ *the* ℓ_2 *projection* $P(\mathbf{y}) = \operatorname{argmin}_{\mathbf{x} \in \Delta} \|\mathbf{x} - \mathbf{y}\|_2$*. If* k *is any real constant, then*

$$P(\mathbf{y} - k\mathbf{1}) = P(\mathbf{y}).$$

Proof.

$$\begin{aligned}
P(\mathbf{y} - k\mathbf{1}) &= \operatorname*{argmin}_{\mathbf{x}\in\Delta} \|\mathbf{x} - (\mathbf{y} - k\mathbf{1})\|_2 \\
&= \operatorname*{argmin}_{\mathbf{x}\in\Delta} \|\mathbf{x} + k\mathbf{1} - \mathbf{y}\|_2 \\
&= \operatorname*{argmin}_{\mathbf{x}\in\Delta} \sqrt{\sum_i^N (x_i + k - y_i)^2} \\
&= \operatorname*{argmin}_{\mathbf{x}\in\Delta} \sqrt{\sum_i^N (x_i^2 - 2x_iy_i + y_i^2 + k^2 + 2kx_i - 2ky_i)} \\
&= \operatorname*{argmin}_{\mathbf{x}\in\Delta} \sqrt{\sum_i^N ((x_i - y_i)^2 + k^2 + 2kx_i - 2ky_i)} \\
&= \operatorname*{argmin}_{\mathbf{x}\in\Delta} \sqrt{\sum_i^N (x_i - y_i)^2 + \sum_i^N k^2 + \sum_i^N 2kx_i - \sum_i^N 2ky_i} \\
&= \operatorname*{argmin}_{\mathbf{x}\in\Delta} \sqrt{\sum_i^N (x_i - y_i)^2 + Nk^2 + 2k - 2k\sum_i^N y_i} \qquad \text{since } \mathbf{x} \in \Delta \\
&= \operatorname*{argmin}_{\mathbf{x}\in\Delta} \sqrt{\sum_i^N (x_i - y_i)^2}.
\end{aligned}$$

The last line follows because $\sum_i^N y_i$ is constant when minimizing over $\mathbf{x}$, and the functions $\sqrt{f(x)}$ and $\sqrt{f(x) + c}$, for some constant c, are minimized at the same point. □

We can now relate ACPI and GIGA(s) using the lemmas above.

Lemma 4. *Running projected ACPI (or projected PGPI) with learning rates*

$$\alpha_{s,k} = k^{-\frac{1}{2}} \mathcal{B}_{-i}(\pi, s)\eta_i^\pi(s),$$

is equivalent to running GIGA(s) at each state s with its required learning rate of $k^{-\frac{1}{2}}$; as a result, the total (local) regret at each state s after K steps is at most $\sqrt{K} + \left(\sqrt{K} - \frac{1}{2}\right) |\mathcal{A}|(\Delta r)^2$, where $\Delta r = \max r_i - \min r_i$.

Proof. We first prove the statement for ACPI. If we rewrite the ACPI update equations by substituting the advantage from Lemma 2 and the learning rate in the statement of the theorem, after cancelling terms we get:

$$\theta_{s,\cdot} \leftarrow P(\{\theta_{s,a} + k^{-\frac{1}{2}} v_{\eta,\pi}(s,a) - k^{-\frac{1}{2}} \eta_i^\pi(s) v_i^c(\pi, s)\}_a)$$

Notice that the right-most term is a constant value added to all the components of the vector $\boldsymbol{\theta}_s$. These constants shift each component of the vector $\boldsymbol{\theta}_s$ by the same amount, which by Lemma 3 does not affect the resulting local policy $\pi^{k+1}(s)$ when projected back onto the simplex, leaving an equivalent update to the step 2(a) in Definition 2. The total regret is then obtained by [100, Theorem 4].

To prove the statement is true for PGPI, we simply need to retrace our steps. In Lemma 2, the rightmost term is missing, leaving:

$$q_\pi(s,a) = \frac{v_{\eta,\pi}(s,a)}{\eta_i^\pi(s)\mathcal{B}_{-i}(\pi, s)}.$$

Now, when we rewrite the PGPI update equations, but substituting $q_\pi(s,a)$ we obtain the same update equation except missing the constant shift:

$$\theta_{s,\cdot} \leftarrow P(\{\theta_{s,a} + k^{-\frac{1}{2}} v_{\eta,\pi}(s,a)\}_a).$$

and the same logic holds as before without the constant shift over all actions a at s. □

We are now ready to prove Theorem 1.

E.1 Proof of Theorem 1

As a result of Lemma 4, the regret is minimized locally. Formally, for K steps the total local regret for playing the sequence of policies $\pi^0(s), \pi^1(s), \cdots, \pi^K(s)$ at s is:

$$R_i^k(s) = \max_{a \in \mathcal{A}} \sum_{k=1}^{K} v_{\eta,\pi}(s,a) - \pi^k(s) \cdot \mathbf{v}_{\eta,\pi^k}(s,\mathbf{a}) \leq \sqrt{K} + (\sqrt{K} - \frac{1}{2})|\mathcal{A}|(\Delta r)^2$$

That is, the total regret over k steps is sublinear in K, so the average regret locally at each s approaches 0 as $k \to \infty$.

Using Lemma 1, and noticing that the second term is a dot product over a vector whose components are $v_{\eta,\pi^k}(s,a)$ for each $a \in \mathcal{A}$ at s, we can rewrite the above in terms of counterfactual values:

$$\max_{a \in \mathcal{A}} \sum_{k=1}^{K} \eta_i^{\pi^k}(s) v_i^c(\pi^k, s, a) - \eta_i^{\pi^k}(s) v_i^c(\pi^k, s) \leq \sqrt{K} + (\sqrt{K} - \frac{1}{2})|\mathcal{A}|(\Delta r)^2,$$

Let $\eta_i^{\min} = \min_k \eta_i^{\pi^k}(s)$. We can divide both sides by this value to get an expression in terms of counterfactual values only:

$$\max_{a \in \mathcal{A}} \sum_{k=1}^{K} v_i^c(\pi^k, s, a) - v_i^c(\pi^k, s) \leq \frac{1}{\eta_i^{\min}} \left(\sqrt{K} + (\sqrt{K} - \frac{1}{2})|\mathcal{A}|(\Delta r)^2 \right).$$

This means that average immediate counterfactual regret is also minimized at s, and similarly to [101, Theorem 4] we can plug this into [101, Theorem 3] and get an upper bound on the overall average regrets by summing over all states. Then by the Folk theorem ([101, Theorem 2]) this results in an approximate Nash equilibrium with approximations that become more accurate as the number of steps grows.

There are several observations worth noting from the regret bound above. The probability of reaching a state (from one's own policy), *i.e.* $\eta_i^{\pi}(s)$, plays a role in the bound. Specifically, a lower bound on exploration might help ensure $1/\eta_i^{\min}$ is not too large. Also, the choice of $\eta_i^{\min}$ is conservative and the upper-bound might be loose as a result; a more practical consideration might be to keep the reach probability high to states where the average counterfactual regrets are high, and slowly lower exploration to subtrees that consistently have low or zero regret (being sure to always sample with positive probability in case a change in policy above triggers new changes further down the tree.)

Adding exploration in this way does not cause any problems for *asymptotic convergence in the limit*. When using tabular policies in games with perfect recall, an exploratory behavior policy can be decomposed into a mixture of main policy that is learning and some exploration policy due to Kuhn's theorem [44]. Then, the total regret can be decomposed into the regret from the policy that's learning and the regret of the exploration policy, and the latter is upper-bounded by the reward range (see [43] for details). This introduces a source of constant regret and hence the exploration must be decayed over time for it to vanish in the limit.

However, adding exploration in this way will affect the finite-time regret bound. The problem of $\frac{1}{\eta_i^{\min}}$ can be overcome by using a stronger form of policy improvement, which we show in the next section.

E.2 Proof of Theorem 2

The original optimization problem concentrates on ascending the score function $J(\pi_\theta) = v_\pi(s_0)$. The change in policy at every state is focused on increasing only the value of the initial state, which leads to the changes at each state to be weighted by their reach probability. Hence this update is a kind of incremental policy improvement with small/careful improvement steps, which is closer in spirit to Conservative Policy Iteration [40] and Maximum aposteriori Policy Optimisation [2]. This is in contrast to other tabular methods that perform a greedy policy improvement steps at every state.

Instead, strong ACPI changes the optimization problem to modify the policy in the direction of ascent at all the state-local values simultaneously, which is closer in form to assigning (or moving toward) $\pi(s) = \text{argmax}_a\, q(s,a)$ at every state s as is done in value-based methods. This changes the policy updates, so we now re-derive the gradient components from the start of this section, but using state-local gradients at each state s.

Here, $J^{PG}(\pi_{\boldsymbol{\theta}}, s) = \mathbb{E}_{\rho\sim\pi}[G_t \mid S_t = s]$. The contribution of some $\theta_{s,a}$ to the gradient is:

$$\begin{aligned}
\frac{\partial J^{PG}(\pi_\theta, s)}{\partial \theta_{s,a}} &= \frac{\partial}{\partial \theta_{s,a}} \left(\sum_{h,z \in \mathcal{Z}(s,a)} \Pr(h \mid s) \eta^\pi(h,z) u_i(z) \right) \\
&= \frac{\partial}{\partial \theta_{s,a}} \sum_{h,z \in \mathcal{Z}(s,a)} \frac{\eta^\pi_{-i}(h)}{\mathcal{B}_{-i}(\pi, s)} \eta^\pi(h,z) u_i(z) \qquad \text{from Section 3.2} \\
&= \sum_{h,z \in \mathcal{Z}(s,a)} \frac{\eta^\pi_{-i}(h)}{\mathcal{B}_{-i}(\pi, s)} \eta^\pi(ha,z) u_i(z).
\end{aligned}$$

which differs from $v_{\eta,\pi}(s,a)$ by replacing the weight $\eta^\pi(h)$ by $\frac{\eta^\pi_{-i}(h)}{\mathcal{B}_{-i}(\pi,s)}$ inside the sum.

It is now easy to verify that this value is more desirable than $v_{\eta,\pi}(s,a)$ by looking again at Lemmas 1 and 2 using this new definition of value. Continuing from above,

$$\begin{aligned}
\frac{\partial J^{PG}(\pi_\theta, s)}{\partial \theta_{s,a}} &= \ldots \\
&= \sum_{h,z \in \mathcal{Z}(s,a)} \frac{\eta^\pi_{-i}(h)}{\mathcal{B}_{-i}(\pi, s)} \eta^\pi(ha,z) u_i(z) \\
&= \frac{1}{\mathcal{B}_{-i}(\pi, s)} \sum_{h,z \in \mathcal{Z}(s,a)} \eta^\pi_{-i}(h) \eta^\pi(ha,z) u_i(z) \\
&= \frac{v^c_i(\pi, s, a)}{\mathcal{B}_{-i}(\pi, s)} \qquad \text{(which is an analog to Lemma 1)} \\
&= \frac{v_{\eta,\pi}(s,a)}{\eta^\pi_i(s)\mathcal{B}_{-i}(\pi, s)} = \frac{\frac{\partial}{\partial \theta_{s,a}} J^{PG}(\pi_{\boldsymbol{\theta}})}{\eta^\pi_i(s)\mathcal{B}_{-i}(\pi, s)},
\end{aligned}$$

where the last line shows how the learning rates relate between Theorems 1 and 2. The advantage works out to be: $q_\pi(s,a) - \sum_b \pi(s,b) q_\pi(s,b)$

$$\begin{aligned}
&= \frac{v^c_i(\pi, s, a) - v^c_i(\pi, s)}{\mathcal{B}_{-i}(\pi, s)} \qquad \text{from Section 3} \\
&= \frac{v^c_i(\pi, s, a)}{\mathcal{B}_{-i}(\pi, s)} - \frac{v^c_i(\pi, s)}{\mathcal{B}_{-i}(\pi, s)} \\
&= \frac{\partial}{\partial \theta_{s,a}} J^{PG}(\pi_\theta, s) - \sum_b \pi(s,b) \frac{\partial}{\partial \theta_{s,b}} J^{PG}(\pi_\theta, s) \qquad \text{by the derivation above.}
\end{aligned}$$

Now, when GIGA(s) uses this new value, we can state a lemma that analogous to Lemma 4 with but with a weaker requirement on the learning rate:

Lemma 5. *Running SACPI (or SPGPI) with learning rate $\alpha_k = k^{-\frac{1}{2}}$ is equivalent to running GIGA(s) at each state s with its required learning rate of $k^{-\frac{1}{2}}$; as a result, the total (local) regret at each state s after K steps is at most $\sqrt{K} + \left(\sqrt{K} - \frac{1}{2}\right) |\mathcal{A}|(\Delta r)^2$, where $\Delta r = \max r_i - \min r_i$.*

The proof follows the same logic as in the proof of Lemma 4.

The proof of Theorem 2 then follows very closely the steps of proof of Theorem 1, but instead of the counterfactual values being weighted by $\eta_i^{\pi^k}(s)$, they are instead weighted by $\frac{1}{\mathcal{B}_{-i}(\pi,s)}$. This is better because then we can multiply both sides by $\mathcal{B}_{-i}(\pi, s)$ and remove it from the bound because it is upper-bounded by 1. In particular, there is no occurence of $\frac{1}{\eta_i^{\min}}$.

F On the similarity of QPG and RPG

As discussed in Section 3.2 and in [84, Chapter 13], subtracting the baseline does not affect the gradient. Therefore, the QPG gradient at state s can be written as:

$$\nabla_{\boldsymbol{\theta}}^{\text{QPG}}(s) = \nabla_{\boldsymbol{\theta}} \left(\sum_a \pi_{\boldsymbol{\theta}}(s,a) q(s,a;\mathbf{w}) \right) = \sum_a \nabla_{\boldsymbol{\theta}} \pi_{\boldsymbol{\theta}}(s,a;\boldsymbol{\theta}) q(s,a;\mathbf{w}).$$

The RPG gradient $\nabla_{\boldsymbol{\theta}}^{\text{RPG}}(s) = -\nabla_{\boldsymbol{\theta}} \sum_a \left(q(s,a;\mathbf{w}) - \sum_b \pi(s,b;\boldsymbol{\theta}) q(s,b;\mathbf{w})\right)^+$

$$\begin{aligned}
&= \sum_a \mathbb{I}\left[q(s,a;\mathbf{w}) > \sum_b \pi(s,b;\boldsymbol{\theta}) q(s,b;\mathbf{w})\right] \nabla_{\boldsymbol{\theta}} \sum_b \pi(s,b;\boldsymbol{\theta}) q(s,b;\mathbf{w}) \\
&= \sum_a \mathbb{I}\left[q(s,a;\mathbf{w}) > \sum_b \pi(s,b;\boldsymbol{\theta}) q(s,b;\mathbf{w})\right] \sum_b \nabla_{\boldsymbol{\theta}} \pi(s,b;\boldsymbol{\theta}) q(s,b;\mathbf{w}) \\
&= \sum_a \mathbb{I}\left[q(s,a;\mathbf{w}) > \sum_b \pi(s,b;\boldsymbol{\theta}) q(s,b;\mathbf{w})\right] \nabla_{\boldsymbol{\theta}}^{\text{QPG}}(s) \\
&= n_{a+}(s) \nabla_{\boldsymbol{\theta}}^{\text{QPG}}(s),
\end{aligned}$$

where the first line follows because $\frac{d}{dx}(x)^+ = 0$ for $x < 0$, and $n_{a+}(s)$ is the number of actions at s with positive advantage. Therefore for any state s, the RPG gradient is proportional to the QPG gradient.

In the special case of two-action games, at any state s, either the advantages are both 0 or there is one negative-advantage action and one positive-advantage action, so $\nabla_{\boldsymbol{\theta}}^{\text{RPG}}(s) = \nabla_{\boldsymbol{\theta}}^{\text{QPG}}(s)$.

The similarity between RPG and QPG shown here could be (partly) responsible the similar behavior of QPG and RPG that we observed in our experiments.

G Additional details on the experiments

For the policy gradient algorithm experiments, we tried both Adam and SGD optimizers and found SGD to be better performing. We ran sweeps over learning rates and found 0.01 to be the best performing one. The other hyper parameters in the search included N_q and $batch_size$ and we found that the values of 128 and 4 were the best performing in all the games. The entropy cost was swept in the range [0, 0.2] and the best performing value was found to be 0.1 for all the domains. The experiments were run over 5 different seeds and we noted that the exploitability across all 5 seeds was very close (standard deviation = ±0.02).

The Nq and batch size hyper parameters correspond to the number of q-updates (critic) and batch size used to compute actor and critic updates. Note that the critic (q) is updated more times (Nq) in order to perform accurate policy evaluation before performing one policy improvement (actor update). The hyper parameters were swept over a grid: $\{16, 32, 64, 128, 256, 512\}$ for Nq and $\{4, 8, 16, 32\}$ for batch size. The best performing values were Nq = 128 and batch size = 4. We performed the experiments over 5 random seeds and found that the results for the chosen hyper parameters were very close for all 5 seeds (standard deviation of 0.02). While we only performed a simple grid search, we found that the chosen hyper parameters worked best across all games.

H Other Reductions to Counterfactual Regret Minimization

The reduction of ACPI to CFR shown in Section 3.2 is analogous to the reduction from another existing algorithm, sequence-form replicator dynamics (SFRD) [25], to CFR [48].

In the common setting of estimating gradients from samples, the algorithm then becomes analogous to the on-policy Monte Carlo sampling case in RL.

There is also a model-free sampled version of SFRD called sequence-form Q-learning (SFQ) [66]. In SFQ, each step samples a deterministic policy requiring time linear in size of the policy. In contrast, actor-critic algorithms can work directly in the behavioral representation (tables indexed by s, a). Also, RL-style function approximation can be easily used in the standard way to generalize over the state space.

I Negative Results: Monte Carlo Regression CFR (Retracted Baseline)

In this section, we describe a baseline that we would have liked to include in the comparison: Monte Carlo RCFR, a version of Regression CFR [92] built from sampled trajectories. Unfortunately, we were unable to get stable results with this algorithm (with Monte Carlo sampling) and more work needs to be done in order to investigate the cause of the instability.

CFR produces, at each iteration k, a joint policy π_k, and *cumulative regrets* $\text{CREG}(K, s, a) = \sum_{k \in \{1, \cdots, K} \text{REG}(\pi_k, s, a)$, and average cumulative regrets $\text{ACREG}(K, s, a) = \frac{1}{K}\text{CREG}(K, s, a)$. CFR uses *thresholded* cumulative regrets $\text{TCREG}(K, s, a) = \text{CREG}(K, s, a)^+$ (or $\text{TACREG}(K, s, a) = \text{ACREG}(K, s, a)^+$) values to determine the next policy π_{k+1} at each information state using regret matching.

Regression CFR (RCFR) is a policy iteration algorithm that, like CFR, does full tree passes at each iteration. However, it uses a regressor to approximate the $\text{CREG}(K, s, a)$ for all information states. The policies are still derived from these approximate regrets using regret matching. The original RCFR used input features and regression trees. Our implementation uses raw input and neural networks, with the same architectures as for our actor-critic experiments. We ran some experiments for our implementation of RCFR, which are shown in Figure 10. We found SGD to be unstable for this problem, and had better results using Adam [42].

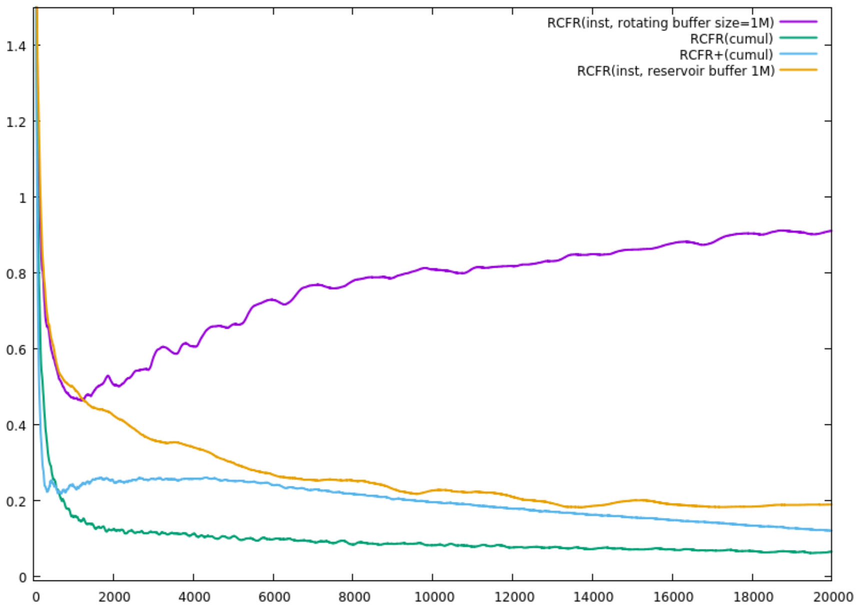


Figure 10: Regression CFR convergence results; x-axis represents iterations k and y-axis represents $\text{NASHCONV}(\pi_k)$. RCFR(inst, ?) refers to using instantaneous (immediate) regrets REG as the regression targets, whereas RCFR(cumul) refers to the average cumulative regrets ACREG. Rotating buffer refers to a circular buffer where new entries delete the oldest entries when the buffer is full. Reservoir sampling seems to be important when using immediate regret as targets, since it is predicting an average cumualtive regret. RCFR+ accumulates regret only if the immediate regrets are positive.

RCFR is not a baseline in our experiments for the same reason CFR is not: it performs policy iteration which requires free access to the environment, and a full state space sweep per iteration. So, we define a sampling version, **Monte Carlo RCFR** (MCRCFR) that is entirely model-free (independent of the environment and other players' policies). MCRCFR differs from RCFR in the following ways:

- A value critic is also trained from samples: $v_\pi(s)$. This is done in exactly the same way as in the actor-critic methods.
- A data set (of size 1 million) is retained to store data. Each data point stored in this data set is a tuple $(s, R, \widehat{\text{REG}}(s, a), a)$" the input encoding of the information state s, the return obtained R, the sampled immediate regret $\widehat{\text{REG}}(s, a) = R - v_\pi$, and the action that was chosen a. Reservoir sampling [89] is used to replace data in this buffer, so in expectation the data retained in the buffer is a uniform sample over all the data that was seen.
- Instead of running a full state space sweep, it samples a trajectory ρ using an explorative policy $\mu = \epsilon \text{UNIFORM}(\mathcal{A}(s)) + (1 - \epsilon)\pi_i(s)$ at each s. At the beginning, $\epsilon = 1$ and is decayed (multiplied) by 0.995 every 1000 episodes to a minimum of 0.0001.
- An average policy $\bar{\pi}_i$ is predicted via classification using the actions a that were taken at each s stored in the data set, similar to NFSP. This is the policy we use to assess exploitability.

To train, we use Adam [42] a constant learning rate of 0.0001 and batch size of 128. For every 100 sampled episodes, for each network we assembled 10 mini-batches of 128 and run a training step.

MCRCFR seems similar to Advantage Regret Minimization (ARM) [38]. We outline the differences below:

- ARM does not maintain a data set: it learns values and (thresholded) advantage values online using moving average of the parameters (whose targets are composed of two separate approximators), but does not predict the average policy.
- The q_k^+ values are bootstrapped from previous values, which is possible when using CFR+. (CFR average cumulative regrets cannot be bootstrapped in this way.)
- The q_k^+ values predict *cumulative sums* rather than average cumulative values (see [38, Equation 13]).