Neural Replicator Dynamics

Shayegan Omidshafiei[*,1] somidshafiei@...
Daniel Hennes[*,1] hennes@...
Dustin Morrill[*,2] dmorrill@...
Remi Munos[1] munos@...
Julien Perolat[3] perolat@...
Marc Lanctot[2] lanctot@...
Audrunas Gruslys[3] audrunas@...
Jean-Baptiste Lespiau[1] jblespiau@...
Karl Tuyls[1] karltuyls@...

...@google.com.

[1]DeepMind Paris, [2]DeepMind Edmonton, [3]DeepMind London. [*]Equal contributors.

Abstract

In multiagent learning, agents interact in inherently nonstationary environments due to their concurrent policy updates. It is, therefore, paramount to develop and analyze algorithms that learn effectively despite these nonstationarities. A number of works have successfully conducted this analysis under the lens of evolutionary game theory (EGT), wherein a population of individuals interact and evolve based on biologically-inspired operators. These studies have mainly focused on establishing connections to value-iteration based approaches in stateless or tabular games. We extend this line of inquiry to formally establish links between EGT and policy gradient (PG) methods, which have been extensively applied in single and multiagent learning. We pinpoint weaknesses of the commonly-used softmax PG algorithm in adversarial and nonstationary settings and contrast PG's behavior to that predicted by replicator dynamics (RD), a central model in EGT. We consequently provide theoretical results that establish links between EGT and PG methods, then derive Neural Replicator Dynamics (NeuRD), a parameterized version of RD that constitutes a novel method with several advantages. First, as NeuRD reduces to the well-studied no-regret Hedge algorithm in the tabular setting, it inherits no-regret guarantees that enable convergence to equilibria in games. Second, NeuRD is shown to be more adaptive to nonstationarity, in comparison to PG, when learning in canonical games and imperfect information benchmarks including Poker. Thirdly, modifying any PG-based algorithm to use the NeuRD update rule is straightforward and incurs no added computational costs. Finally, while single-agent learning is not the main focus of the paper, we verify empirically that NeuRD is competitive in these settings with a recent baseline algorithm.

1 Introduction

In multiagent reinforcement learning (MARL), agents interact in a shared environment and aim to learn policies that maximize their returns [15, 48, 62]. The associated core challenge is that the agents' concurrent learning implies that they each perceive the environment as nonstationary [37, 62]. It has been suggested that enabling agents to *adapt* to nonstationary environments, rather than merely *learn* static policies, is the paramount objective in multiagent learning [54, 61]. Recent

Preprint. Under review.

works have made considerable progress in increasing the scalability of MARL algorithms and the complexity of application domains. These include approaches relying on self-play [23], regret minimization [9, 10, 12, 34, 43, 69], and a large body of works that assume a standard centralized learning, decentralized execution paradigm [18, 35, 44, 51]. However, approaches such as Neural Fictitious Self-Play (NFSP) [23] and Deep CFR [12] rely on maintenance of extremely large data buffers; continual re-solving approaches such as DeepStack [43] and Libratus [10] are restricted to finite-horizon domains with enumerable belief spaces; and centralized learning approaches assume team-wide shared knowledge of experiences or policies. Most other works that focus on nonstationary multiagent learning are limited to repeated and stochastic games [1, 15, 24, 46, 47].

This paper focuses on model-free learning in the context of nonstationary games. Specifically, we leverage insights from evolutionary game theory (EGT) [26, 38, 70] to develop our proposed algorithm. EGT models the interactions of a population of individuals that have minimal knowledge of opponent strategies or preferences, and has proven vital for not only the analysis and evaluation of MARL agents [45, 50, 66, 67], but also the development of novel algorithms [64]. Links between EGT and MARL have been primarily identified between the Replicator Dynamics (RD), a standard model in EGT, and simple policy iteration algorithms such as Cross Learning and Learning Automata [4, 5, 65], value-iteration algorithms such as Q-learning in stateless cases [27, 63], and no-regret learning [29].

In this paper, we go beyond these simple settings and focus our analysis on the connections between EGT and Policy Gradient (PG) algorithms, which have been extensively applied to MARL [2, 7, 8, 18, 35, 55, 57]. We first establish links between the RD, PG methods, and online learning, enabling new interpretations of these approaches. We use these links to identify a key limitation in commonly-used softmax PG-based methods that prevents their rapid adaptation in nonstationary settings, in contrast to RD. Given this insight, we derive a new PG method, called Neural Replicator Dynamics (NeuRD). We prove that in the tabular setting, NeuRD reduces to Hedge, a well-studied no-regret learning algorithm. We then introduce a variant of NeuRD that enables model-free learning in sequential imperfect information games. We empirically evaluate NeuRD in canonical and sequential games. By including variants of each where reward spaces are nonstationary, we demonstrate that NeuRD is not only more stable when learning in fixed games, but also more adaptive under nonstationarity compared to standard PG. Moreover, though not the focus of the paper, we demonstrate that NeuRD matches state-of-the-art performance in a suite of complex stationary single-agent tasks. Critically, the conversion of standard PG-based algorithms to the NeuRD update rule is simple, involving changes to few lines of code. These features make NeuRD a strong alternative to usual PG algorithms in nonstationary environments, such as in MARL or for computing equilibria in games.

2 Preliminaries

This section briefly introduces the necessary game-theoretic and reinforcement learning background.

2.1 Game Theory

Game theory studies strategic interactions of players. A *normal-form game* (NFG) specifies the simultaneous interaction of K players with corresponding action sets $\{\mathcal{A}^1, \ldots, \mathcal{A}^K\}$. The payoff function $\boldsymbol{u} : \prod_{k=1}^{K} \mathcal{A}^k \mapsto \mathbb{R}^K$ assigns a numerical utility to each player for each possible joint action $\mathbf{a} \doteq (a^1, \ldots, a^K)$, where $a^k \in \mathcal{A}^k$ for all $k \in [K] \doteq \{1, \ldots, K\}$. Let $\boldsymbol{\pi}^k \in \Delta^{|\mathcal{A}|}$ denote the k-th player's mixed strategy. The expected utility for player k given strategy profile $\boldsymbol{\pi} \doteq (\boldsymbol{\pi}^1, \ldots, \boldsymbol{\pi}^K)$ is then $\bar{u}^k(\boldsymbol{\pi}) \doteq \mathbb{E}_{\boldsymbol{\pi}}[u^k(\mathbf{a}) | \mathbf{a} \sim \boldsymbol{\pi}]$. The best response for player k given $\boldsymbol{\pi}$ is $\mathrm{BR}^k(\boldsymbol{\pi}^{-k}) = \arg\max_{\boldsymbol{\pi}^k}[\bar{u}^k((\boldsymbol{\pi}^k, \boldsymbol{\pi}^{-k}))]$, where $\boldsymbol{\pi}^{-k}$ is the set of opponent policies. Profile $\boldsymbol{\pi}_*$ is a Nash equilibrium if $\boldsymbol{\pi}_*^k = \mathrm{BR}^k(\boldsymbol{\pi}_*^{-k})$ for all $k \in [K]$. A useful metric for evaluating policies is $\textsc{NashConv}(\boldsymbol{\pi}) = \sum_k \max_{\boldsymbol{\pi}^k}[\bar{u}^k((\boldsymbol{\pi}^k, \boldsymbol{\pi}^{-k}))] - \bar{u}^k(\boldsymbol{\pi})$ [32].

2.2 Replicator Dynamics

Replicator Dynamics (RD) are a key concept from EGT, which describe how a population evolves through time based on biologically-inspired operators, such as selection and mutation [26, 59, 60, 71, 72]. The single-population RD are defined by the following system of differential equations:

$$\dot{\pi}_t(a) = \pi_t(a)\big[u(a, \boldsymbol{\pi}_t) - \bar{u}(\boldsymbol{\pi}_t)\big] \qquad \forall i \in \mathcal{A}, \tag{1}$$

Each component of $\boldsymbol{\pi}_t$ determines the proportion of an action (or pure strategy) a being played in the population at time t. The time derivative of each component a is proportional to the difference in its expected payoff, $u(a, \boldsymbol{\pi}_t)$, and the average population payoff, $\bar{u}(\boldsymbol{\pi}_t) = \sum_{a \in \mathcal{A}} \pi_t(a) u(a, \boldsymbol{\pi}_t)$.

2.3 Online Learning

Online learning examines the performance of learning prediction algorithms in potentially adversarial environments. On each round, t, the learner samples an action, $a_t \in \mathcal{A}$ from a discrete set of actions, $\mathcal{A}$, according to a policy, $\boldsymbol{\pi}_t \in \Delta^{|\mathcal{A}|}$, and receives utility, $u_t(a) \in \mathbb{R}$, where $\boldsymbol{u}_t \in \mathbb{R}^{|\mathcal{A}|}$ is a bounded vector provided by the environment. A typical objective is for the learner to minimize its expected *regret* in hindsight for not committing to $a \in \mathcal{A}$ after T rounds; the regret is defined as

$$R_T(a) \doteq \sum_{t=1}^{T} u_t(a) - \boldsymbol{\pi}_t \cdot \boldsymbol{u}_t.$$

Algorithms that guarantee their average worst-case regret goes to zero as the number of rounds increases, i.e., $R_T \in o(T)$, are called *no-regret*; these algorithms learn optimal policies under fixed or stochastic environments. According to a folk theorem, the average policies of no-regret algorithms in self-play or against best responses converge to a Nash equilibrium in two-player zero-sum games (e.g., see Waugh [68, Section 2.2.1]). This result can be extended from matrix games to sequential imperfect information games by composing learners in a tree and defining utility as counterfactual value [25, 73]. We provide background on sequential games in Appendix A for completeness.

The family of no-regret algorithms known as Follow the Regularized Leader (FoReL) [39, 40, 52, 53] generalizes well-known decision making algorithms and population dynamics. For a discrete action set, $\mathcal{A}$, FoReL is defined through the following updates:

$$\begin{aligned} \boldsymbol{\pi}_t &\doteq \operatorname*{arg\,max}_{\boldsymbol{\pi}' \in \Delta^{|\mathcal{A}|}} [\boldsymbol{\pi}' \cdot \boldsymbol{y}_{t-1} - h(\boldsymbol{\pi}')] \\ \boldsymbol{y}_t &\doteq \boldsymbol{y}_{t-1} + \eta_t \boldsymbol{u}_t, \end{aligned}$$

where $\eta_t > 0$ is the learning rate at timestep t, $\boldsymbol{u}_t \in \mathbb{R}^{|\mathcal{A}|}$ is the vector of action utilities observed at t, and the regularizer h is a convex function. Note that this algorithm assumes that the learner observes the entire action utility vector at each timestep rather than only the reward for taking a particular action. This is known as the full information or *all-actions* setting.

Under negative entropy regularization $h(\boldsymbol{\pi}) = \sum_a \pi(a) \log \pi(a)$, policy $\boldsymbol{\pi}_t$ reduces to a softmax function $\boldsymbol{\pi}_t \doteq \Pi(\boldsymbol{y}_{t-1})$, where $\Pi(\boldsymbol{z}) \propto \exp(z), \forall z \in \mathbb{R}^{|\mathcal{A}|}$. This yields the well-known *Hedge* [19] algorithm:

$$\boldsymbol{\pi}_T \doteq \Pi\left(\sum_{t=1}^{T-1} \eta_t \boldsymbol{u}_t\right). \tag{2}$$

Hedge is no-regret as long as $\eta_t \in O(1/\sqrt{t})$. Likewise, the continuous-time FoReL dynamics [41] are

$$\begin{aligned} \boldsymbol{\pi}_t &\doteq \operatorname*{arg\,max}_{\boldsymbol{\pi}' \in \Delta^{|\mathcal{A}|}} \left[\boldsymbol{\pi}' \cdot \boldsymbol{y}_t - h(\boldsymbol{\pi}')\right] \\ \dot{\boldsymbol{y}}_t &\doteq \boldsymbol{u}_t \end{aligned} \tag{3}$$

which in the case of entropy regularization yields RD as defined in (1) (e.g., see [41]). This implies that RD is no-regret, thereby enjoying equilibration to Nash and convergence to the optimal prediction in the time-average.

2.4 Reinforcement Learning and Policy Gradient (PG)

In a Markov Decision Process, at each timestep t, an agent in state $s_t \in \mathcal{S}$ selects an action $a_t \in \mathcal{A}$, receives a reward $r_t \in \mathbb{R}$, then transitions to a new state $s_{t+1} \sim \mathcal{T}(s, a, s')$. In the discounted infinite-horizon regime, the reinforcement learning (RL) objective is to learn a policy $\boldsymbol{\pi} : s \to \Delta^{|\mathcal{A}|}$, which maximizes the expected return $v^{\boldsymbol{\pi}}(s) = \mathbb{E}_{\boldsymbol{\pi}}[\sum_{k=t}^{\infty} \gamma^{k-t} r_k | s_t = s]$, with discount factor $\gamma \in [0, 1)$.

In actor-critic algorithms, one generates trajectories according to some parameterized policy $\boldsymbol{\pi}(\cdot|s; \boldsymbol{\theta})$ while learning to estimate the action-value function $q^{\boldsymbol{\pi}}(s, a) = \mathbb{E}_{\boldsymbol{\pi}}[\sum_{k=t}^{\infty} \gamma^{k-t} r_k | s_t = s, a_t = a]$.

Temporal difference learning [58] can be used to learn an action-value function estimator, $q^{\boldsymbol{\pi}}(s, a; \boldsymbol{w})$, which is parameterized by $\boldsymbol{w}$. A PG algorithm then updates policy $\boldsymbol{\pi}$ parameters $\boldsymbol{\theta}$ in the direction of the gradient $\nabla_{\boldsymbol{\theta}} \log \pi(a_t|s_t; \boldsymbol{\theta})\big[q^{\boldsymbol{\pi}}(s_t, a_t; \boldsymbol{w}) - v^{\boldsymbol{\pi}}(s_t; \boldsymbol{w})\big]$, where the quantity in square brackets is defined as the *advantage*, denoted $A(a_t; \boldsymbol{\theta}, \boldsymbol{w})$, and $v^{\boldsymbol{\pi}}(s_t; \boldsymbol{w}) \doteq \sum_b \pi(s_t, b_t; \boldsymbol{\theta}) q^{\boldsymbol{\pi}}(s_t, b_t; \boldsymbol{w})$. The advantage is analogous to regret in the online learning literature. In sample-based learning, the PG update incorporates a $(\pi(a_t|s_t; \boldsymbol{\theta}))^{-1}$ factor that accounts for the fact that a_t was sampled from $\boldsymbol{\pi}$. The all-actions PG algorithm without this factor is then $\nabla_{\boldsymbol{\theta}} \boldsymbol{\pi}(\cdot|s_t; \boldsymbol{\theta})\big[\boldsymbol{q}^{\pi}(s_t, \cdot; \boldsymbol{w}) - v^{\pi}(s_t; \boldsymbol{w})\mathbf{1}\big]$, where $\mathbf{1}$ is a column vector of 1s. While different policy parameterizations are possible, the most common choice for discrete decision problems is a softmax function over the logits, $\boldsymbol{\pi}(\cdot|s_t; \boldsymbol{\theta}) \doteq \Pi(\boldsymbol{y}(\cdot|s_t; \boldsymbol{\theta}))$, and we focus the rest of our analysis on this form of PG.

PG-based methods learn policies directly, handle continuous actions seamlessly, and combine readily with deep learning to solve high-dimensional tasks [58]. These benefits have, in part, led to the success of recent PG-based algorithms (e.g., A3C [42], IMPALA [17], MADDPG [35], and COMA [18]).

3 A Unifying Perspective on Replicator Dynamics and Policy Gradient

This section motivates and introduces a novel algorithm, Neural Replicator Dynamics (NeuRD), then presents unifying theoretical results for NeuRD, online learning, RD, and PG.

3.1 Learning Dynamics under RD and PG

Let us consider the strengths and weaknesses of the algorithms described thus far. While RD and the closely-related FoReL are no-regret and enable learning of equilibria in games, they are limited in application to tabular settings. By contrast, PG is applicable to high-dimensional single and multiagent RL domains. Unfortunately, PG suffers from the fact that increasing the probability of taking an action which already has low probability mass can be very slow, in contrast to the considered no-regret algorithms. We can see this by writing out the single state, tabular, all-actions PG update explicitly, using the notation of online learning to identify the correspondences to that literature. On round t, PG updates its logits and policy as

$$\begin{aligned} \boldsymbol{\pi}_t &\doteq \Pi(\boldsymbol{y}_{t-1}) \\ \boldsymbol{y}_t &\doteq \boldsymbol{y}_{t-1} + \eta_t \nabla_{\boldsymbol{y}_{t-1}} \boldsymbol{\pi}_t \cdot \boldsymbol{u}_t. \end{aligned}$$

As there is no action or state sampling in this setting, shifting all the payoffs by the expected value $\bar{u}$ (or $v^{\boldsymbol{\pi}}$ in RL terms) has no impact on the policy, so this term is omitted here. Noting that ${}^{\partial \pi_t(a')}/_{\partial y_{t-1}(a)} = \pi_t(a')\left[\mathbb{1}_{a'=a} - \pi_t(a)\right]$ [58, Section 2.8], we observe that the update direction, $\nabla_{\boldsymbol{y}_{t-1}} \boldsymbol{\pi}_t \cdot \boldsymbol{u}_t$, is actually the instantaneous regret scaled by $\boldsymbol{\pi}_t$, yielding the concrete update:[1]

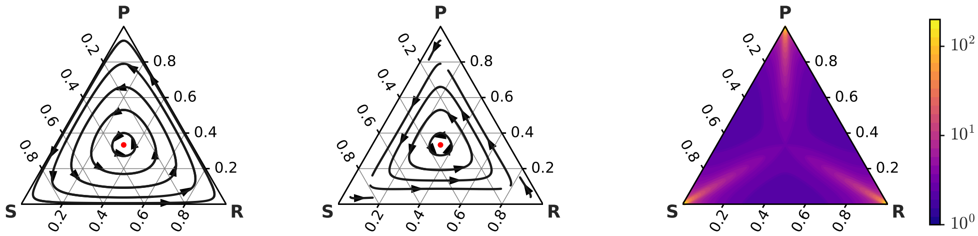
$$y_t(a) = y_{t-1}(a) + \eta_t \pi_t(a)\big[u_t(a) - \sum_{a' \in \mathcal{A}} \pi_t(a') u_t(a')\big]. \tag{4}$$

See Appendix C.1 for details. Scaling the regret by $\boldsymbol{\pi}_t$ leads to an update that can prevent PG from achieving reasonable performance in even simple environments.

The fact that the update includes a $\boldsymbol{\pi}_t$ scaling factor can hinder learning in nonstationary settings (e.g., in games) when an action might be safely disregarded at first, but later the value of this action improves. We illustrate this adaptability issue in the game of *Rock–Paper–Scissors*, by respectively comparing the dynamics of RD and PG in Figs. 1a and 1b; note the differences near the vertices, where a single action retains a majority of policy mass. Figure 1c compares the speeds of the dynamics by plotting the ratio $\|\dot{\boldsymbol{\pi}}_{\text{RD}}\|/\|\dot{\boldsymbol{\pi}}_{\text{PG}}\|$. This example illustrates the practical issues that arise when using PG in settings where learning has converged to a near-deterministic policy and then must adapt to a different policy given, e.g., dynamic payoffs or opponents. While PG fails to adapt rapidly to the game at hand, RD does not exhibit this issue.

Given these insights, our objective is to derive an algorithm that combines the best of both worlds, in that it is theoretically-grounded and adaptive in the manner of RD, while still enjoying the practical benefits of the parametric PG update rule in RL applications.

[1] This scaling by $\boldsymbol{\pi}_t$ is also apparent when taking the continuous-time limit of PG dynamics (see Appendix B).



(a) Replicator dynamics. (b) PG dynamics. (c) Comparison of dynamics speeds.

Figure 1: Learning dynamics of (a) RD and (b) PG dynamics in *Rock–Paper–Scissors*. In (c) we compare the the speed of RD and PG, i.e., $\|\dot{\boldsymbol{\pi}}_{\text{RD}}\|/\|\dot{\boldsymbol{\pi}}_{\text{PG}}\|$. When policies are near-deterministic (i.e., near the simplex boundaries), RD is notably more adaptive than PG.

3.2 NeuRD: Neural Replicator Dynamics

While we have highlighted key limitations of PG in comparison to RD, the latter has limited practicality when computational updates are inherently discrete-time or a parameterized policy is desired for generalizability. To address these limitations, we derive a discrete-time parameterized policy update rule, titled Neural Replicator Dynamics (NeuRD), which is later compared against PG. For seamless comparison of our update rule to standard PG, we next switch our nomenclature from the utilities used in online learning, $u(a)$, to the analogous action-values used in RL, $q^{\pi}(a)$. We write the RD (i.e., FoReL with entropy regularization) logit dynamics (3) as

$$\dot{y}(a) = q^{\boldsymbol{\pi}}(a) - v^{\boldsymbol{\pi}}, \tag{5}$$

where $v^{\boldsymbol{\pi}}$ is the standard variance-reducing baseline [58]. Let $y_t(a;\boldsymbol{\theta})$ denote the logits parameterized by $\boldsymbol{\theta}$. A natural way to derive a parametric update rule is to compute the Euler discretization[2] of (5),

$$y_t(a) \doteq y_{t-1}(a;\boldsymbol{\theta}) + \eta\big(q^{\boldsymbol{\pi}_t}(a) - v^{\boldsymbol{\pi}_t}\big),$$

and consider $y_t(a)$ a fixed *target value* that the parameterized logits $y_{t-1}(a;\boldsymbol{\theta})$ are adjusted toward. Specifically, one can update $\boldsymbol{\theta}$ to minimize some choice of metric $d(\cdot,\cdot)$,

$$\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} - \sum_a \nabla_{\boldsymbol{\theta}} d(y_t(a), y_{t-1}(a;\boldsymbol{\theta})).$$

In particular, minimizing the Euclidean distance yields,

$$\begin{aligned}
\boldsymbol{\theta}_t &= \boldsymbol{\theta}_{t-1} - \sum_a \nabla_{\boldsymbol{\theta}} \frac{1}{2}\|y_t(a) - y_{t-1}(a;\boldsymbol{\theta})\|^2 \\
&= \boldsymbol{\theta}_{t-1} + \sum_a (y_t(a) - y_{t-1}(a;\boldsymbol{\theta}))\nabla_{\boldsymbol{\theta}} y_{t-1}(a;\boldsymbol{\theta}) \\
&= \boldsymbol{\theta}_{t-1} + \eta \sum_a \nabla_{\boldsymbol{\theta}} y_{t-1}(a;\boldsymbol{\theta})\big(q^{\boldsymbol{\pi}}(a) - v^{\boldsymbol{\pi}}\big), \qquad (6)
\end{aligned}$$

which we later prove has a rigorous connection to Hedge and, thereby, inherits the no-regret guarantees that are useful in nonstationary settings such as games. As our experiments use a neural network parameterization, we refer to the update rule (6) as Neural Replicator Dynamics (NeuRD).

3.3 Properties and Unifying Theorems

Overall, NeuRD is not only practical to use as it involves a simple modification of PG with no added computational expense, but also benefits from rigorous links to algorithms with no-regret guarantees, as shown in this section. All proofs are in Appendix C.

We first consider the single state, tabular case to make a key connection to no-regret algorithms.

Theorem 1. *Single state, all-actions, tabular NeuRD is Hedge.*

[2]Given a tabular softmax policy, this definition matches the standard discrete-time RD. See Appendix C.5.

As a reminder, Hedge (and therefore tabular NeuRD) is no-regret, so NeuRD can be used to find optimal policies or Nash equilibria. In sequential decision making settings, ensuring no-regret requires independent learners at every decision point, and additionally a counterfactual weighting of utilities in imperfect information games (IIGs) [73]; for completeness, we provide the necessary background on this in Appendix A. At a high level, in IIGs the set of information states $\mathcal{S}^k$ for player $k \in [K]$ corresponds to a partitioning of action histories, and counterfactual values are defined as the expected player utility weighted by the product of opponent sequence probabilities. With these changes, applying NeuRD to imperfect information settings is trivial, as shown in our experiments.

These facts ensure that tabular NeuRD can be used to solve a broad class of problems where PG may fail. While these guarantees are limited to the tabular case, they constitute a principled theoretical grounding on which parameterized NeuRD is constructed.

We next formalize the connection between RD and PG, expanding beyond the scope of prior works that have considered only the links between EGT and value-iteration based algorithms [27, 63].

Proposition 1. *PG is a policy-level Euler discretization approximation of continuous-time RD (i.e., computing $\boldsymbol{\pi}_{t+1}$ using $\boldsymbol{\pi}_t$), under a KL-divergence minimization criterion.*

Next we establish a formal link between NeuRD and Natural Policy Gradient (NPG) [28].

Proposition 2. *The NeuRD update rule* (6) *corresponds to a naturalized policy gradient rule, in the sense that NeuRD applies a natural gradient only at the policy output level of softmax function over logits, and uses the standard gradient otherwise.*

Unlike NPG, NeuRD does not require computation of the inverse Fisher information matrix, which is especially expensive when, e.g., the policy is parameterized by a large-scale neural network [36].

Empirical Remarks. As in average, the logits $\boldsymbol{y}$ get incremented by the advantage, they may diverge to $\pm\infty$. To avoid numerical issues, in practice one can stop updating the logits if the gap between them exceeds a large threshold. We apply this by using the following clipped gradient in lieu of a standard gradient:

$$\widetilde{\nabla}_{\boldsymbol{\theta}}(z(\boldsymbol{\theta}), \eta, \beta) \doteq [\eta \nabla_{\boldsymbol{\theta}} z(\boldsymbol{\theta})] \mathbb{I}\{z(\boldsymbol{\theta}) + \eta \nabla_{\boldsymbol{\theta}} z(\boldsymbol{\theta}) \in [-\beta, \beta]\},$$

where $\mathbb{I}\{\cdot\}$ is the indicator function, $\eta > 0$ is a learning rate, and $\beta \in \mathbb{R}$ controls the allowable logits gap. This thresholding is not problematic at the policy representation level, since actions can have a probability arbitrarily close to 0 or 1 given a large enough β. Moreover, while we have so far assumed an all-actions NeuRD update, a corresponding sample-based variant, where $a_t \sim \pi(a; \boldsymbol{\theta})$, is given by

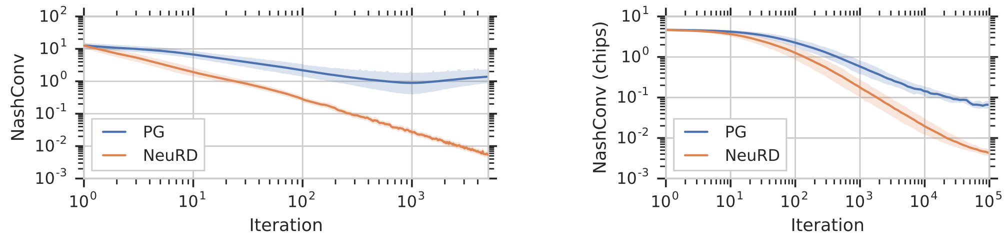
$$\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} + \frac{\widetilde{\nabla}_{\boldsymbol{\theta}}\big(y(a_t; \boldsymbol{\theta})(q^\pi(a) - v^\pi), \eta, \beta\big)}{\pi(a_t; \boldsymbol{\theta}) + \varepsilon},$$

where $\varepsilon > 0$ is a variance-reduction term, useful since the update rule scales inversely with action selection probability $\pi(a_t; \boldsymbol{\theta})$, which may be close to 0 for certain actions.

4 Evaluation

We conduct a series of evaluations demonstrating the effectiveness of NeuRD when learning in nonstationary settings such as NFGs, standard imperfect information benchmarks, and variants of each with additional reward nonstationarity. As NeuRD involves only a simple modification of the PG update rule to improve adaptivity, we focus our comparisons against standard PG as a baseline, noting that additional benefits can be analogously gained by combining NeuRD with more intricate techniques that improve PG (e.g., variance reduction, improved exploration, or off-policy learning). Experimental procedures are detailed in Appendix D.1.

We consider several domains. **Rock–Paper–Scissors (RPS)** is a well-known canonical NFG involving two players, with a cyclic dominance among the three strategies. **Goofspiel** is a card game where players try to obtain point cards by bidding simultaneously. We use an imperfect information variant with 4 cards where bid cards are not revealed [30]. **Kuhn Poker** is a game wherein each player starts with 2 chips, antes 1 chip to play, receives a face-down card from a deck of $K + 1$ such that one card remains hidden, and either bets (raise or call) or folds until all players are in (contributed equally to the pot) or out (folded). Amongst those that are in, the player with the highest-ranked card wins the

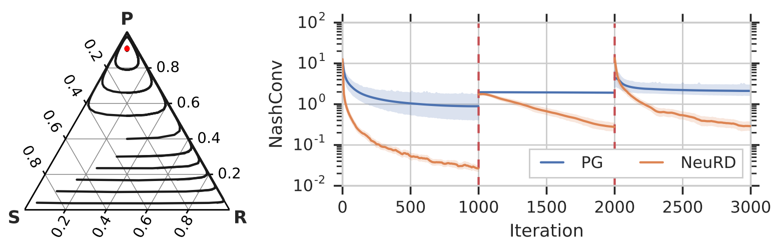


(a) Biased RPS (b) Leduc Poker

Figure 2: (a) NashConv of the average NeuRD and PG policies in biased RPS. (b) NashConv of the sequence-probability average policies of tabular, all-actions, counterfactual value NeuRD and PG in two-player Leduc Poker.

	R	P	S
R	0	-1	ν
P	1	0	-1
S	$-\nu$	1	0

(a) RPS payoff table



(b) RD dynamics (c) Nonstationary biased RPS

Figure 3: (a) RPS payoffs, where the choice of $\nu \in \mathbb{R}$ induces payoff bias. (b) RD trajectories in biased RPS with $\nu = 20$, Nash shown as red dot. (c) Time-average policy NashConv in nonstationary RPS, with the three game phases separated by vertical dashed red lines.

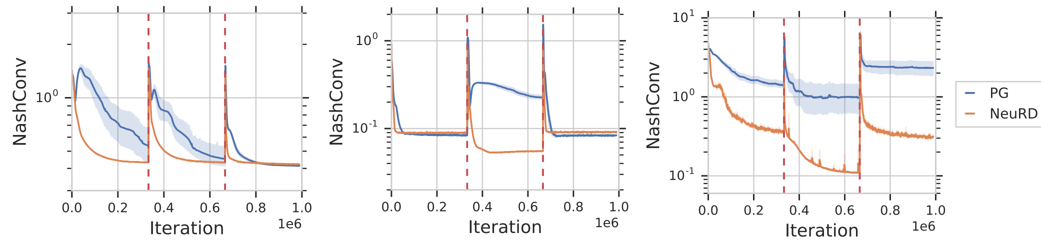
pot. In **Leduc Poker** [56], players instead have limitless chips, one initial private card, and ante 1 chip to play. Bets are limited to 2 and 4 chips, respectively, in the first and second round, with two raises maximum in each round. A public card is revealed before the second round. Though not the primary focus of the paper, we also provide empirical results for single-agent stationary RL tasks in Appendix D.3, with the key observation being that updating the recently-introduced IMPALA [17] agent to use the NeuRD update rule matches state-of-the-art performance.

We first show that the differences between NeuRD and PG detailed in Section 3.3 are more than theoretical. Consider the NashConv of the time-average NeuRD and PG tabular policies in the game of RPS, shown in Fig. 2a. Note that by construction, NeuRD and RD are equivalent in this tabular, single-state setting. NeuRD not only converges towards the Nash equilibrium faster, but PG eventually plateaus. Consider next a more complex imperfect information setting, where Fig. 2 shows that tabular, all-actions, counterfactual value NeuRD[3] more quickly and more closely approximates a Nash equilibrium in two-player Leduc Poker than tabular, all-actions, counterfactual value PG.

We next consider modifications of our domains wherein reward functions change at specific intervals during learning, compounding the usual nonstationarities in games. Specifically, we consider games with three phases, wherein learning commences under a particular reward function, after which it switches to a different function in each phase while learning continues *without* the policies being reset. In biased RPS, each phase corresponds to a particular choice of the parameter ν in payoff tables Fig. 3a; specifically, we set ν to 20, 0, and 20, respectively, for the three phases. This has the effect of biasing the Nash equilibrium towards one of the simplex corners (see Fig. 3c), then to the simplex center (Fig. 1a), then back again towards the corner. In Fig. 3c, we plot the NashConv of NeuRD and PG with respect to the Nash for that particular phase. Despite the changing payoffs, the NeuRD NashConv decreases towards 0 in each of the phases, while PG again plateaus.

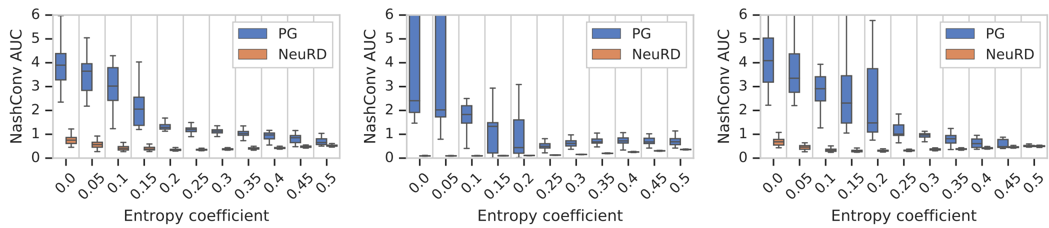
Finally, we consider imperfect information games, with the reward function being iteratively negated in each game phase for added nonstationarity, and policies parameterized using neural networks. Due to the complexity of maintaining a time-average neural network policy to ensure no-regret, we use entropy regularization to induce realtime policy convergence towards the Nash (e.g., as done by Srinivasan et al. [57]). Figure 4 illustrates the NashConv for NeuRD and PG for the

[3]This can be seen as counterfactual regret minimization (CFR) [73] with Hedge, also see Brown et al. [11].



(a) Goofspiel (b) Kuhn Poker (c) Leduc Poker

Figure 4: Comparison of NeuRD and PG NashConv in nonstationary imperfect information games. Vertical red dashed lines separate the phases with different reward functions for each game.



(a) Phase I (b) Phase II (c) Phase III

Figure 5: Comparison of NeuRD and PG NashConv AUC in the three phases of nonstationary-reward Leduc Poker, over 40 independent trials each.

imperfect information domains considered, for an intermediate level of entropy regularization. NeuRD converges faster than PG in all three domains. Appendix D.2 provides full sweeps over the entropy regularization parameter; as regularization increases, so does the rate of convergence, although to a fixed point further from the Nash. In Fig. 5, we plot the NashConv area-under-the-curve (AUC) for all game phases in nonstationary Leduc, and for all entropy regularization levels. Notably, NeuRD is significantly more stable in learning than PG, even without entropy regularization. Of particular note is Phase II of the game, wherein PG fails to match NeuRD's performance for any value of regularization.

5 Discussion

This paper rigorously investigated the links between RD and PG methods, extending prior inquiries between EGT and value-iteration based methods. The insights gained led to the introduction of a novel algorithm, NeuRD, that generalizes the no-regret Hedge algorithm and RD to utilize function approximation. NeuRD was empirically shown to better cope in highly nonstationary and adversarial settings than PG. While NeuRD represents an important extension of classical learning dynamics to utilize function approximation, Hedge and RD are also instances of the more general FoReL framework that applies to arbitrary convex decision sets and problem geometries expressed through a regularization function. This connection suggests that NeuRD could perhaps likewise be generalized to convex decision sets and various parametric forms, which would allow a general FoReL-like method to take advantage of functional representations. Moreover, as NeuRD is practical in that it involves a very simple modification of the standard PG update rule, a natural avenue for future work is to investigate NeuRD-based extensions of standard PG-based methods (e.g., A3C [42], DDPG [33], and MADDPG [35]), in addition to naturally nonstationary single-agent RL tasks such as intrinsic motivation-based exploration [22].

References

[1] Sherief Abdallah and Michael Kaisers. Addressing environment non-stationarity by repeating q-learning updates. *Journal of Machine Learning Research*, 17(46):1–31, 2016.

[2] Maruan Al-Shedivat, Trapit Bansal, Yuri Burda, Ilya Sutskever, Igor Mordatch, and Pieter Abbeel. Continuous adaptation via meta-learning in nonstationary and competitive environments. *arXiv preprint arXiv:1710.03641*, 2017.

[3] Charles Beattie, Joel Z Leibo, Denis Teplyashin, Tom Ward, Marcus Wainwright, Heinrich Küttler, Andrew Lefrancq, Simon Green, Víctor Valdés, Amir Sadik, et al. Deepmind lab. *arXiv preprint arXiv:1612.03801*, 2016.

[4] 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.

[5] Tilman Börgers and Rajiv Sarin. Learning through reinforcement and replicator dynamics. *Journal of Economic Theory*, 77(1):1–14, 1997.

[6] Vivek S Borkar. *Stochastic approximation: a dynamical systems viewpoint*, volume 48. Springer, 2009.

[7] Michael Bowling. Convergence and no-regret in multiagent learning. In *Advances in neural information processing systems*, pages 209–216, 2005.

[8] Michael Bowling and Manuela Veloso. Multiagent learning using a variable learning rate. *Artificial Intelligence*, 136(2):215–250, 2002.

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

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

[11] 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.

[12] Noam Brown, Adam Lerer, Sam Gross, and Tuomas Sandholm. Deep counterfactual regret minimization. *CoRR*, abs/1811.00164, 2018. URL `http://arxiv.org/abs/1811.00164`.

[13] Neil Burch. *Time and Space: Why Imperfect Information Games are Hard*. PhD thesis, University of Alberta, 2017.

[14] Neil Burch, Matej Moravcik, and Martin Schmid. Revisiting cfr+ and alternating updates. *Journal of Artificial Intelligence Research*, 64:429–443, 2019.

[15] Lucian Busoniu, Robert Babuska, and Bart De Schutter. A comprehensive survey of multiagent reinforcement learning. *IEEE Trans. Systems, Man, and Cybernetics, Part C*, 38(2):156–172, 2008.

[16] Ross Cressman. *Evolutionary Dynamics and Extensive Form Games*. The MIT Press, 2003.

[17] Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Volodymir Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, et al. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures. *arXiv preprint arXiv:1802.01561*, 2018.

[18] Jakob N Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. Counterfactual multi-agent policy gradients. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[19] Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139, 1997.

[20] 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.

[21] Nicola Gatti and Marcello Restelli. Sequence-form and evolutionary dynamics: Realization equivalence to agent form and logit dynamics. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI-16)*, pages 509–515, 2016.

[22] Alex Graves, Marc G Bellemare, Jacob Menick, Remi Munos, and Koray Kavukcuoglu. Automated curriculum learning for neural networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1311–1320. JMLR. org, 2017.

[23] Johannes Heinrich and David Silver. Deep reinforcement learning from self-play in imperfect-information games. *arXiv preprint arXiv:1603.01121*, 2016.

[24] 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.

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

[26] J. J. Hofbauer and K. Sigmund. Evolutionary games and population dynamics. *Cambridge University Press*, 1998.

[27] Michael Kaisers and Karl Tuyls. Frequency adjusted multi-agent q-learning. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1-Volume 1*, pages 309–316. International Foundation for Autonomous Agents and Multiagent Systems, 2010.

[28] Sham M Kakade. A natural policy gradient. In *Advances in neural information processing systems*, pages 1531–1538, 2002.

[29] Tomas Klos, Gerrit Jan van Ahee, and Karl Tuyls. Evolutionary dynamics of regret minimization. In *Machine Learning and Knowledge Discovery in Databases, European Conference, ECML PKDD 2010, Barcelona, Spain, September 20-24, 2010, Proceedings, Part II*, pages 82–96, 2010.

[30] 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.

[31] 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.

[32] 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 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 4190–4203. 2017.

[33] 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. *arXiv preprint arXiv:1509.02971*, 2015.

[34] Edward Lockhart, Marc Lanctot, Julien Pérolat, Jean-Baptiste Lespiau, Dustin Morrill, Finbarr Timbers, and Karl Tuyls. Computing approximate equilibria in sequential adversarial games by exploitability descent. *CoRR*, abs/1903.05614, 2019. URL `http://arxiv.org/abs/1903.05614`.

[35] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. In *Advances in Neural Information Processing Systems*, pages 6379–6390, 2017.

[36] James Martens and Roger Grosse. Optimizing neural networks with kronecker-factored approximate curvature. In *International conference on machine learning*, pages 2408–2417, 2015.

[37] Laetitia Matignon, Guillaume J Laurent, and Nadine Le Fort-Piat. Independent reinforcement learners in cooperative markov games: a survey regarding coordination problems. *The Knowledge Engineering Review*, 27(1):1–31, 2012.

[38] J. Maynard Smith and G. R. Price. The logic of animal conflicts. *Nature*, 246:15–18, 1973.

[39] H Brendan McMahan. Follow-the-regularized-leader and mirror descent: Equivalence theorems and l1 regularization. 2011.

[40] H Brendan McMahan, Gary Holt, David Sculley, Michael Young, Dietmar Ebner, Julian Grady, Lan Nie, Todd Phillips, Eugene Davydov, Daniel Golovin, et al. Ad click prediction: a view from the trenches. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1222–1230. ACM, 2013.

[41] Panayotis Mertikopoulos, Christos Papadimitriou, and Georgios Piliouras. Cycles in adversarial regularized learning. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2703–2717. SIAM, 2018.

[42] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937, 2016.

[43] 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.

[44] Frans A Oliehoek, Christopher Amato, et al. *A concise introduction to decentralized POMDPs*, volume 1. Springer, 2016.

[45] Shayegan Omidshafiei, Christos Papadimitriou, Georgios Piliouras, Karl Tuyls, Mark Rowland, Jean-Baptiste Lespiau, Wojciech M Czarnecki, Marc Lanctot, Julien Perolat, and Remi Munos. α-rank: Multi-agent evaluation by evolution. *arXiv preprint arXiv:1903.01373*, 2019.

[46] Gregory Palmer, Rahul Savani, and Karl Tuyls. Negative update intervals in deep multi-agent reinforcement learning. *CoRR*, abs/1809.05096, 2018.

[47] Gregory Palmer, Karl Tuyls, Daan Bloembergen, and Rahul Savani. Lenient multi-agent deep reinforcement learning. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '18, pages 443–451, 2018.

[48] Liviu Panait and Sean Luke. Cooperative multi-agent learning: The state of the art. *Autonomous Agents and Multi-Agent Systems*, 11(3):387–434, 2005.

[49] 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.

[50] Marc Ponsen, Karl Tuyls, Michael Kaisers, and Jan Ramon. An evolutionary game-theoretic analysis of poker strategies. *Entertainment Computing*, 1(1):39–45, 2009.

[51] Tabish Rashid, Mikayel Samvelyan, Christian Schroeder de Witt, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning. *arXiv preprint arXiv:1803.11485*, 2018.

[52] Shai Shalev-Shwartz and Yoram Singer. Online learning: Theory, algorithms, and applications. 2007.

[53] Shai Shalev-Shwartz and Yoram Singer. A primal-dual perspective of online learning algorithms. *Machine Learning*, 69(2-3):115–142, 2007.

[54] Yoav Shoham, Rob Powers, and Trond Grenager. If multi-agent learning is the answer, what is the question? *Artificial Intelligence*, 171(7):365–377, 2007.

[55] Satinder Singh, Michael Kearns, and Yishay Mansour. Nash convergence of gradient dynamics in general-sum games. In *Proceedings of the Sixteenth conference on Uncertainty in artificial intelligence*, pages 541–548. Morgan Kaufmann Publishers Inc., 2000.

[56] Finnegan Southey, Michael Bowling, Bryce Larson, Carmelo Piccione, Neil Burch, Darse Billings, and Chris Rayner. Bayes? bluff: Opponent modelling in poker. In *Proceedings of the Twenty-First Conference on Uncertainty in Artificial Intelligence (UAI-05)*, 2005.

[57] Sriram Srinivasan, Marc Lanctot, Vinicius Zambaldi, Julien Pérolat, Karl Tuyls, Rémi Munos, and Michael Bowling. Actor-critic policy optimization in partially observable multiagent environments. In *Advances in Neural Information Processing Systems*, pages 3422–3435, 2018.

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

[59] P. Taylor. Evolutionarily stable strategies with two types of players. *Journal of Applied Probability*, 16:76–83, 1979.

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

[61] Karl Tuyls and Simon Parsons. What evolutionary game theory tells us about multiagent learning. *Artif. Intell.*, 171(7):406–416, 2007.

[62] Karl Tuyls and Gerhard Weiss. Multiagent learning: Basics, challenges, and prospects. *AI Magazine*, 33(3):41–52, 2012.

[63] Karl Tuyls, Katja Verbeeck, and Tom Lenaerts. A selection-mutation model for q-learning in multi-agent systems. In *The Second International Joint Conference on Autonomous Agents & Multiagent Systems, AAMAS 2003, July 14-18, 2003, Melbourne, Victoria, Australia, Proceedings*, pages 693–700, 2003.

[64] Karl Tuyls, Ann Nowe, Tom Lenaerts, and Bernard Manderick. An evolutionary game theoretic perspective on learning in multi-agent systems. *Synthese*, 139(2):297–330, 2004.

[65] Karl Tuyls, Pieter Jan't Hoen, and Bram Vanschoenwinkel. An evolutionary dynamical analysis of multi-agent learning in iterated games. *Autonomous Agents and Multi-Agent Systems*, 12(1): 115–153, 2006.

[66] Karl Tuyls, Julien Pérolat, Marc Lanctot, Joel Z. Leibo, and Thore Graepel. A generalised method for empirical game theoretic analysis. In *AAMAS*, pages 77–85. International Foundation for Autonomous Agents and Multiagent Systems Richland, SC, USA / ACM, 2018.

[67] Karl Tuyls, Julien Perolat, Marc Lanctot, Rahul Savani, Joel Leibo, Toby Ord, Thore Graepel, and Shane Legg. Symmetric decomposition of asymmetric games. *Scientific Reports*, 8(1): 1015, 2018.

[68] Kevin Waugh. Abstraction in large extensive games, 2009. URL `http://poker.cs.ualberta.ca/publications/waugh.msc.pdf`.

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

[70] Jorgen Weibull. Evolutionary game theory. *MIT press*, 1997.

[71] E.C. Zeeman. Population dynamics from game theory. *Lecture Notes in Mathematics, Global theory of dynamical systems*, 819, 1980.

[72] E.C. Zeeman. Dynamics of the evolution of animal conflicts. *Theoretical Biology*, 89:249–270, 1981.

[73] 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.

Appendices: Neural Replicator Dynamics

A Background on extensive-form games and counterfactual values

An *extensive-form game* (EFG) specifies the sequential interaction between players $k \in [K] \cup \{c\}$, where c is a special player called *chance* or *nature* (also denoted as player 0) who has a fixed stochastic policy that determines the transition probabilities at random events like dice rolls or the dealing of cards from a shuffled deck. Actions, $a \in \mathcal{A}(h)$, are played in turns according to a player function, $\tau\colon \mathcal{H} \to [K] \cup \{c\}$, and are recorded in a *history*, $h \in \mathcal{H}$, where $\mathcal{H}$ is the set of all possible action sequences. To model games like Poker that require some actions to be hidden from particular players, players do not observe the game's history directly. Instead, histories are partitioned into *information states*, $s \in \mathcal{S}^k, \mathcal{S}^k \subseteq \mathcal{H}$, for each player, $k \in [K] \cup \{c\}$, according to function $I\colon \mathcal{H} \to \mathcal{S}, \mathcal{S} \doteq \bigcup_{k=1}^{K} \mathcal{S}^k$ mapping sets of histories into information states. Player k must act from s without knowing which particular history led to s. This requires that the set of actions in each history within an information set must match, so we can define $\mathcal{A}(s) \doteq \mathcal{A}(h)$ with $s = I(h), h \in \mathcal{H}$. A *behavioral policy* for player k maps every $s \in \mathcal{S}^k$ to a probability distribution over actions: $\pi^k(s) \in \Delta^{|\mathcal{A}(s)|}$. Payoffs, $u^k(z), u^k\colon \mathcal{Z} \to \mathbb{R}$, are provided to each player upon reaching a *terminal history*, $z \in \mathcal{Z} \subseteq \mathcal{H}$. We assume finite games so a terminal history is always eventually reached. We denote the subset of terminal histories that share h as a prefix as $\mathcal{Z}(h)$. For any two histories, $h, h' \in \mathcal{H}$, we use the notation $h \sqsubset h'$ to indicate that history h is a prefix of h'.

We also define state-action values for joint policies. The value $q^{k,\boldsymbol{\pi}}(s, a)$ represents the expected return to player k starting at state s, taking action a, and playing $\boldsymbol{\pi}$. Formally, $q^{k,\boldsymbol{\pi}}(s, a) \doteq$

$$\mathbb{E}_{z\sim\boldsymbol{\pi}}[u^k(z) \mid h \in s, h \sqsubset z] = \sum_{h\in s, z\in\mathcal{Z}(h)} Pr(h|s)u^k(z) = \frac{\sum_{h\in s} \eta_{\boldsymbol{\pi}}(h) q^{k,\boldsymbol{\pi}}(h, a)}{\sum_{h\in s} \eta_{\boldsymbol{\pi}}(h)},$$

where $q^{k,\boldsymbol{\pi}}(h, a) = \mathbb{E}_{z\sim\boldsymbol{\pi}}[u^k(z) \mid ha \sqsubseteq z] = \sum_{h\in s, z\in\mathcal{Z}(h)} \sigma_{\boldsymbol{\pi}}(h, z)u^k(z)$ is the expected utility of the ground state-action pair (h, a), and $\sigma_{\boldsymbol{\pi}}(h)$ is the probability of reaching h under the profile $\boldsymbol{\pi}$. We make the common assumption that players have *perfect recall*, i.e., they do not forget anything they have observed while playing. Under perfect recall, the distribution of the states can be obtained only from the opponents' policies using Bayes' rule (see Srinivasan et al. [57, Section 3.2]). For convenience, in turn-based games, we define $q^{\boldsymbol{\pi}}(s, a) \doteq q^{\tau(s),\boldsymbol{\pi}}(s, a)$ and $v^{\boldsymbol{\pi}}(s) \doteq \sum_{a\in\mathcal{A}(s)} \pi^{\tau(s)}(s, a)q^{\boldsymbol{\pi}}(s, a)$.

The probability of reaching any history $h \in \mathcal{H}$ can be decomposed as the product of player *sequence probabilities*, $\sigma_{\boldsymbol{\pi}}(h) = \Pi_{k=0}^{K} \Pi_{h'a\sqsubseteq h, \tau(h')=k} \pi^k(a|I(h'))$. *Counterfactual value* [73] plays a critical role in many Nash equilibrium approximation methods and is defined as the expected utility weighted by the product of opponent sequence probabilities,

$$q^{k,\boldsymbol{\pi},c}(s, a) \doteq \sum_{h\in s} \sigma_{\boldsymbol{\pi}_{-1}}(h) q^{k,\boldsymbol{\pi}}(h, a),$$

with $v^{k,\boldsymbol{\pi},c}(s)$ defined analogously. Policy averaging is also part of many algorithms and is done in terms of sequence probabilities in EFGs.

In this regime, the policy for player $k \in [K]$ on round T in information state $s \in \mathcal{S}^k$, is

$$\pi_T^k(\cdot|s) = \Pi\left(\sum_{t=1}^{T-1} \eta_t \left(q_t^{k,\boldsymbol{\pi}_t,c}(s, a) - v_t^{k,\boldsymbol{\pi}_t,c}(s)\right)\right),$$

where $\eta_t > 0$ is the learning rate at round t, while $q_t^{k,\boldsymbol{\pi}_t,c}(s, a)$ and $v_t^{k,\boldsymbol{\pi}_t,c}(s)$ are, respectively, the counterfactual action-value and value.

B Comparison of RD and Continuous-time PG Dynamics

We can also consider the continuous-time q-value based policy gradient (QPG) dynamics [57, Section D.1.1], which are amenable for comparison against the RD and provide a reasonable approximation to the discrete-time PG models given a sufficiently small learning rate [6]:

$$\dot{\pi}(a;\boldsymbol{\theta}) = \pi(a;\boldsymbol{\theta})\Big(\pi(a;\boldsymbol{\theta})A(a,\boldsymbol{\theta},\boldsymbol{w}) - \sum_b \pi(b;\boldsymbol{\theta})^2 A(b,\boldsymbol{\theta},\boldsymbol{w})\Big). \tag{7}$$

In contrast to RD (1), the QPG dynamics in (7) have an additional $\pi(a;\boldsymbol{\theta})$ term that modulates learning and slows adaptation for actions that are taken with low probability under π.

C Proofs of theoretical results

C.1 Single state, tabular PG update

Here, we fully derive the single state, tabular PG update. On round t, PG updates its logits as

$$\boldsymbol{y}_t \doteq \boldsymbol{y}_{t-1} + \eta_t \nabla_{\boldsymbol{y}_{t-1}} \boldsymbol{\pi}_t \cdot \boldsymbol{u}_t.$$

As there is no action or state sampling in this setting, shifting all the payoffs by the expected value $\bar{u}$ (or $v^{\boldsymbol{\pi}}$ in RL terms) has no impact on the policy, so this term is omitted here. Noting that ${\partial \pi_t(a')}/{\partial y_{t-1}(a)} = \pi_t(a')\left[\mathbb{1}_{a'=a} - \pi_t(a)\right]$ [58, Section 2.8], we observe that the update direction, $\nabla_{\boldsymbol{y}_{t-1}} \boldsymbol{\pi}_t \cdot \boldsymbol{u}_t$, is actually the instantaneous regret scaled by $\boldsymbol{\pi}_t$:

$$\begin{aligned}
\frac{\partial \boldsymbol{\pi}_t}{\partial y_{t-1}(a)} \cdot \boldsymbol{u}_t &= \sum_{a' \in \mathcal{A}} \frac{\partial \pi_t(a')}{\partial y_{t-1}(a)} u_t(a') \\
&= \sum_{a' \in \mathcal{A}} \pi_t(a')\left[\mathbb{1}_{a'=a} - \pi_t(a)\right] u_t(a') \\
&= \pi_t(a)\left[1 - \pi_t(a)\right] u_t(a) - \sum_{a' \in \mathcal{A}, a' \neq a} \pi_t(a')\pi_t(a) u_t(a') \\
&= \pi_t(a)\Big[\left[1 - \pi_t(a)\right] u_t(a) - \sum_{a' \in \mathcal{A}, a' \neq a} \pi_t(a') u_t(a')\Big] \\
&= \pi_t(a)\big[u_t(a) - \pi_t(a) u_t(a) - \sum_{a' \in \mathcal{A}, a' \neq a} \pi_t(a') u_t(a')\big] \\
&= \pi_t(a)\big[u_t(a) - \sum_{a' \in \mathcal{A}} \pi_t(a') u_t(a')\big].
\end{aligned}$$

Therefore, the concrete update is:

$$y_t(a) = y_{t-1}(a) + \eta_t \pi_t(a)\big[u_t(a) - \sum_{a' \in \mathcal{A}} \pi_t(a') u_t(a')\big].$$

C.2 Proof of Theorem 1

Theorem 1. *Single state, all-actions, tabular NeuRD is Hedge.*

Proof. In the single state, tabular case, $\nabla_{\boldsymbol{\theta}_t} y_t(a;\boldsymbol{\theta})$ is the identity matrix, so unrolling the NeuRD update (6) across $T-1$ rounds, we see that the NeuRD policy is

$$\pi_T = \Pi\left(\sum_{t=1}^{T-1} \eta_t \left(\boldsymbol{u}_t - \boldsymbol{u}_t \cdot \pi_t\right)\right) = \Pi\left(\sum_{t=1}^{T-1} \eta_t \boldsymbol{u}_t\right), \tag{8}$$

since Π is shift invariant. But (8) is identical to (2), so NeuRD and hedge use the same policy on every round and are therefore equivalent in this setting. $\square$

C.3 Proof of Proposition 1

The following result unifies the Replicator Dynamics and Policy Gradient.

Proposition 1. *PG is a policy-level Euler discretization approximation of continuous-time RD (i.e., computing $\boldsymbol{\pi}_{t+1}$ using $\boldsymbol{\pi}_t$), under a KL-divergence minimization criterion.*

Proof. A discrete-time Euler discretization of the RD at the policy level is:

$$\boldsymbol{\pi}_{t+1} \doteq \boldsymbol{\pi}_t \odot \Big[\mathbf{1} + \eta\big(\boldsymbol{q} - v^{\pi_t}\mathbf{1}\big)\Big].$$

Note that while $\sum_a \pi_{t+1}(a) = 1$, this Euler-discretized update may still be outside the simplex; however, $\boldsymbol{\pi}_{t+1}$ merely provides a target for our parameterized policy $\boldsymbol{\pi}_{\boldsymbol{\theta}_t}$ update, which is subsequently reprojected back to the simplex via $\Pi(\cdot)$.

Now if we consider parameterized policies $\boldsymbol{\pi}_t \approx \boldsymbol{\pi}_{\boldsymbol{\theta}_t}$, and our goal is to define dynamics on $\boldsymbol{\theta}_t$ that captures those of the RD, a natural way consists in updating $\boldsymbol{\theta}_t$ in order to make $\boldsymbol{\pi}_{\boldsymbol{\theta}_t}$ move towards $\boldsymbol{\pi}_{t+1}$, for example in the sense of minimizing their KL divergence, $KL(\boldsymbol{p}, \boldsymbol{q}) \doteq \boldsymbol{p} \cdot \log \boldsymbol{p} - \boldsymbol{p} \cdot \log \boldsymbol{q}, \boldsymbol{p}, \boldsymbol{q} \in \mathbb{R}^{+,n}, n > 0$.

Of course, the KL divergence is defined only when both inputs are in the positive orthant, $\mathbb{R}^{+,n}$, so in order to measure the divergence from $\boldsymbol{\pi}_{t+1}$, which may have negative values, we need to define a KL-like divergence. Fortunately, since the $\boldsymbol{p} \cdot \log \boldsymbol{p}$ is inconsequential from an optimization perspective and this is the only term that requires $\boldsymbol{p} > 0$, a natural modification of the KL divergence to allow for negative values in its first argument is to drop this term entirely, resulting in $\tilde{KL}(\boldsymbol{p}, \boldsymbol{q}) \doteq -\boldsymbol{p} \cdot \log \boldsymbol{q}, \boldsymbol{p} \in \mathbb{R}^n, \boldsymbol{q} \in \mathbb{R}^{+,n}$.

The gradient-descent step on the $\tilde{KL}(\boldsymbol{\pi}_{t+1}, \boldsymbol{\pi}_{\boldsymbol{\theta}_t})$ objective is:

$$\begin{aligned}
\boldsymbol{\theta}_{t+1} &= \boldsymbol{\theta}_t - \nabla_{\boldsymbol{\theta}} \tilde{KL}(\boldsymbol{\pi}_{t+1}, \boldsymbol{\pi}_{\boldsymbol{\theta}_t}) \\
&= \boldsymbol{\theta}_t + \sum_a \pi_{t+1}(a) \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}_t}(a) \\
&= \boldsymbol{\theta}_t + \sum_a \pi_t(a) \left[1 + \eta\big(q(a) - v^{\pi_t}\big)\right] \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}_t}(a).
\end{aligned}$$

Assuming $\pi_t = \pi_{\boldsymbol{\theta}_t}$,

$$\begin{aligned}
&= \boldsymbol{\theta}_t + \sum_a \pi_{\boldsymbol{\theta}_t}(a) \Big[1 + \eta\big(q(a) - v^{\pi_{\boldsymbol{\theta}_t}}\big)\Big] \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}_t}(a) \\
&= \boldsymbol{\theta}_t + \sum_a \Big[1 + \eta\big(q(a) - v^{\pi_{\boldsymbol{\theta}_t}}\big)\Big] \nabla_{\boldsymbol{\theta}} \pi_{\boldsymbol{\theta}_t}(a). \\
&= \boldsymbol{\theta}_t + \big(1 - \eta v^{\pi_{\boldsymbol{\theta}_t}}\big) \sum_a \nabla_{\boldsymbol{\theta}} \pi_{\boldsymbol{\theta}_t}(a) + \eta \sum_a q(a) \nabla_{\boldsymbol{\theta}} \pi_{\boldsymbol{\theta}_t}(a). \\
&= \boldsymbol{\theta}_t + \big(1 - \eta v^{\pi_{\boldsymbol{\theta}_t}}\big) \nabla_{\boldsymbol{\theta}} \underbrace{\sum_a \pi_{\boldsymbol{\theta}_t}(a)}_{=1} + \eta \sum_a q(a) \nabla_{\boldsymbol{\theta}} \pi_{\boldsymbol{\theta}_t}(a). \\
&= \boldsymbol{\theta}_t + \eta \nabla_{\boldsymbol{\theta}} \sum_a \pi_{\boldsymbol{\theta}_t}(a) q(a) \\
&= \boldsymbol{\theta}_t + \eta \nabla_{\boldsymbol{\theta}} v^{\pi_{\boldsymbol{\theta}_t}},
\end{aligned}$$

which is precisely a policy gradient step. □

C.4 Proof of Proposition 2

We detail here a unification of the Natural PG and NeuRD update rules.

Proposition 2. *The NeuRD update rule* (6) *corresponds to a naturalized policy gradient rule, in the sense that NeuRD applies a natural gradient only at the policy output level of softmax function over logits, and uses the standard gradient otherwise.*

Proof. Consider a policy $\pi(a)$ defined by a softmax over a set of logits $y(a)$: $\pi \doteq \Pi(y)$. Define the Fisher information matrix F of the policy π with respect to the logits y:

$$\begin{aligned}
F_{a,b} &= \sum_c \pi(c)(\partial_{y(a)} \log \pi(c))(\partial_{y(b)} \log \pi(c)) \\
&= \sum_c \pi(c)(\partial_{y(a)} y(c) - \sum_d \pi(d)\partial_{y(a)} y(d))(\partial_{y(b)} y(c) - \sum_d \pi(d)\partial_{y(b)} y(d)) \\
&= \sum_c \pi(c)(\delta_{a,c} - \sum_d \pi(d)\delta_{a,d})(\delta_{b,c} - \sum_d \pi(d)\delta_{b,d}) \\
&= \pi(b)(\delta_{a,b} - \pi(a)).
\end{aligned}$$

Note that

$$\begin{aligned}
(F\nabla y)(a) &= \sum_b F_{a,b} \nabla y(b) \\
&= \sum_b \pi(b)(\delta_{a,b} - \pi(a))\nabla y(b) \\
&= \pi(a)\nabla y(a) - \pi(a) \sum_b \pi(b)\nabla y(b) \\
&= \nabla \pi(a)
\end{aligned}$$

from the definition of π. This means that considering the variables y as parameters of the policy, the natural gradient $\widetilde{\nabla}_y \pi$ of π with respect to y is

$$\widetilde{\nabla}_y \pi = F^{-1}(\nabla_y \pi) = F^{-1}(F\nabla_y y) = I.$$

Now assume the logits $y_{\boldsymbol{\theta}}$ are parameterized by some parameter $\boldsymbol{\theta}$ (e.g., with a neural network). Let us define the *pseudo-natural gradient* of the probabilities π with respect to $\boldsymbol{\theta}$ as the composition of the natural gradient of π with respect to y (i.e., the softmax transformation) and the gradient of $y_{\boldsymbol{\theta}}$ with respect to $\boldsymbol{\theta}$:

$$\widetilde{\nabla}_{\boldsymbol{\theta}} \pi = (\nabla_{\boldsymbol{\theta}} y_{\boldsymbol{\theta}})(\widetilde{\nabla}_y \pi) = \nabla_{\boldsymbol{\theta}} y_{\boldsymbol{\theta}}.$$

From the above, we have that a natural policy gradient yields:

$$\sum_a \widetilde{\nabla}_{\boldsymbol{\theta}} \pi(a)\big(q(a) - v^\pi\big) = \sum_a \nabla_{\boldsymbol{\theta}} y(a)\big(q(a) - v^\pi\big),$$

which is nothing else than the NeuRD update rule in (6). $\square$

C.5 Equivalence to the standard discrete-time replicator dynamic

A common way to define discrete-time replicator dynamics is according to the so-called *standard discrete-time replicator dynamic* [16],

$$\pi_t(a) \doteq \pi_{t-1}(a) \frac{e^{q^{\pi_{t-1}}(a)}}{\boldsymbol{\pi}_{t-1} \cdot e^{\boldsymbol{q}^{\pi_{t-1}}}}.$$

The action values are exponentiated to ensure all the utility values are positive, which is the typical assumption required by this model. Since the policy is a softmax function applied to logits, we can rewrite this dynamic in the tabular case to also recover an equivalent to the NeuRD update rule in (6) with $\eta = 1$:

$$\begin{aligned}
\pi_{t-1}(a) \frac{e^{q^{\pi_{t-1}}(a)}}{\boldsymbol{\pi}_{t-1} \cdot e^{\boldsymbol{q}^{\pi_{t-1}}}} &= \frac{e^{y_{t-1}(a) + q^{\pi_{t-1}}(a)}}{\sum_b e^{y_{t-1}(b)}} \frac{\sum_b e^{y_{t-1}(b)}}{\sum_b e^{y_{t-1}(b) + q^{\pi_{t-1}}(b)}} \\
&= \frac{e^{y_{t-1}(a) + q^{\pi_{t-1}}(a)}}{\sum_b e^{y_{t-1}(b) + q^{\pi_{t-1}}(b)}}.
\end{aligned}$$

π_t is generated from logits, $y_t(a) = y_{t-1}(a) + q^{\pi_{t-1}}(a) = \sum_{\tau=0}^{t-1} q^{\pi_\tau}(a)$, which only differs from (6) in a constant shift of $v^{\pi_{t-1}}$ across all actions. Since the softmax function is shift invariant, the sequence of policies generated from these update rules will be identical.

Interestingly, there have been sequence-form extensions of these standard discrete-time replicator dynamics [20] that are also related to counterfactual regret minimization [31], but with a different (but similar) regret minimizer. There have also been a sampling variants, such as sequence-form Q-learning [49], and sequence-form logit dynamics [21], which seems similar to the NeuRD update. The main benefit of NeuRD over these algorithms is that using counterfactual values allows representing the policies in behavioral form, as is standard in reinforcement learning, rather than the sequence-form. As a result, it is straight-forward to do sampling and function approximation.

D Additional Experimental Remarks and Results

D.1 Experimental procedures and reproducibility notes

We detail the experimental procedures here.

For Fig. 1 we simulate the continuous time RD (1) and the continuous time PG dynamics (7) from regular grid points on the simplex. For Fig. 1a and Fig. 1b we plot 15 trajectories of length $t_{max} = 8$; arrows indicate the direction of evolution. For Fig. 1c we compute $\|\dot{\boldsymbol{\pi}}_{\text{RD}}\|/\|\dot{\boldsymbol{\pi}}_{\text{PG}}\|$ for 20,000 regularly spaced points on the policy simplex.

For Fig. 2a the continuous time dynamics of NeuRD (5) and PG (7) are integrated over time with step size $\Delta t = 0.1$ (equals 1 iteration). The figure shows the mean NASHCONV of 100 trajectories starting from initial conditions sampled uniformly from the policy simplex. The shaded area corresponds to the 95% confidence interval computed with bootstrapping from 1000 samples.

For Fig. 2b, in every iteration, each information-state policy over the entire game was updated for both players in an alternating fashion, i.e., the first player's policy was updated, then the second player's [13, Section 4.3.6] [14]. The only difference between the NeuRD and PG algorithms in this case is the information-state logit update rule, and the only difference here–as described by (4)–is that PG scales the NeuRD update by the current policy. The performance displayed is that of the sequence probability time-average policy for both algorithms (see Appendix A). The set of constant step sizes tried were the same for both algorithms: $\eta \in \{0.5, 0.9, 1, 1.5, 2, 2.5, 3, 3.5, 4\}$. The shaded area corresponds to the 95% interval that would result from a uniform sampling of the step size from this set.

We use the same setup detailed above for Fig. 2a for the nonstationary case shown in Fig. 3. The payoff matrix is switched every 1000 iterations, i.e., at $t = 100$ and $t = 200$.

For each game in Fig. 4, we randomly initialize a policy parameterized by a two-layer neural neural network (128 hidden units). Results are reported for 40 random hyperparameter sweeps for both NeuRD and PG. We update the policy once every 4 update of the Q-function. The batch size of the policy update is 256 trajectories, with trajectory lengths of 5, 8, and 8 for Kuhn Poker, Leduc Poker, and Goofspiel. The Q-function update batch size is 4 trajectories (same lengths). A learning rate of 0.002 was used for policy updates, and 0.01 for Q-value function updates. Reward function negation occurs every $0.33\text{e}6$ learning iterations, with the three reward function phases separated by the vertical red stripes in the plots. Upon conclusion of each learning phase, policies are not reset; instead, learning continues given the latest policy (for each of the 40 trials).

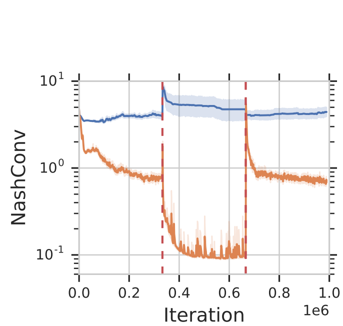
For Fig. 5, we simply compute the NASHCONV area-under-the-curve for each phase of learning in Leduc Poker, across all entropy regularization levels, using 40 hyperparameter seeds. I.e., this plot provides a concise summary of Fig. D.6.

With the exception of the experiments presented in Appendix D.3, all experiments were performed on local workstations. The single-agent DMLab-30 [3] experiments were conducted using a cloud computing platform with P100 GPUs. We performed 20 independent runs with 20 actors and one learner per run. The hyper-parameters were sampled from the same distributions as selected by Espeholt et al. [17] with the addition of NeuRD specific hyper-parameters ε log-uniformly sampled in the interval $[0.05, 0.5]$ and max-logit-gap β uniformly in the interval $[5, 50]$.

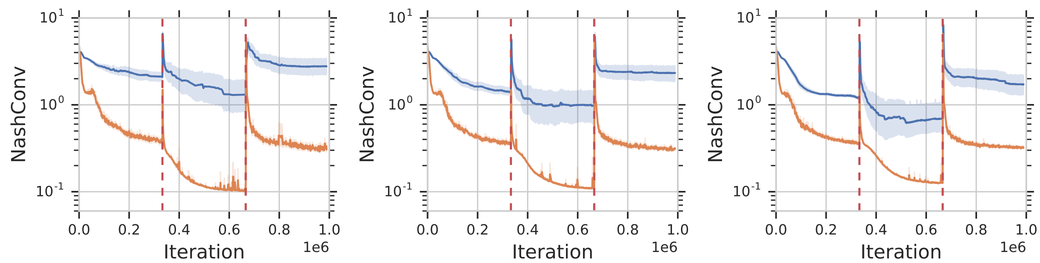
The sample mean is used as the central tendency estimator in plots, with variation indicated as the 95% confidence interval that is the default setting used in the `Seaborn` visualization library that generates our plots. No data was excluded and no other preprocessing was conducted to generate these plots.

D.2 Additional imperfect information game results

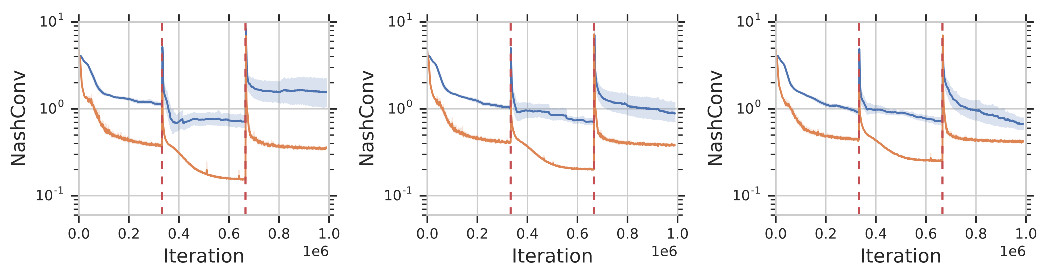
Full results for all considered nonstationary imperfect information domains, under varying levels of policy entropy regularization, are shown in Figs. D.6 to D.8. Increasing policy network regularization causes the NASHCONV to be biased away from the Nash equilibrium. In every learning phase of every domain, the lowest NASHCONV over the sweep of entropy regularization coefficients is achieved by NeuRD.



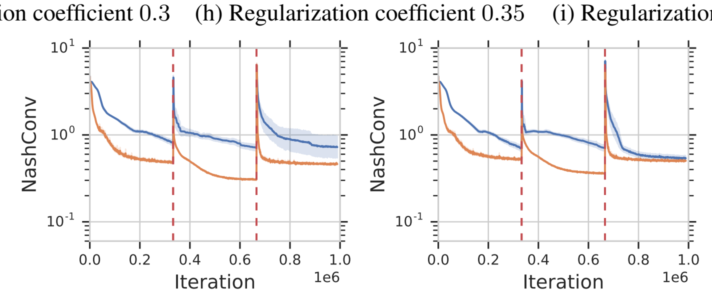
(a) Regularization coefficient 0 (b) Regularization coefficient 0.05 (c) Regularization coefficient 0.1



(d) Regularization coefficient 0.15 (e) Regularization coefficient 0.2 (f) Regularization coefficient 0.25

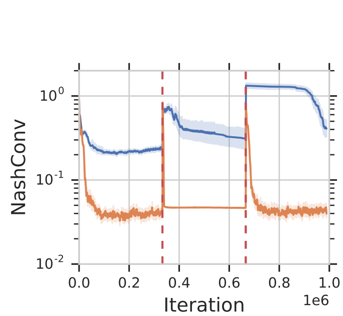


(g) Regularization coefficient 0.3 (h) Regularization coefficient 0.35 (i) Regularization coefficient 0.4

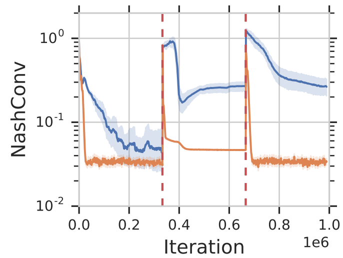


(j) Regularization coefficient 0.45 (k) Regularization coefficient 0.5

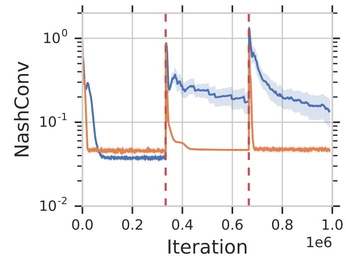
Figure D.6: Comparison of NeuRD and vanilla PG in Leduc Poker with different levels of entropy regularization. The three phases corresponding to Figs. 5a to 5c are separated by the vertical dashed red lines.



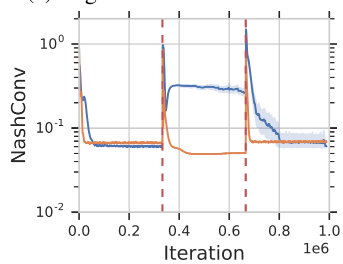
(a) Regularization coefficient 0



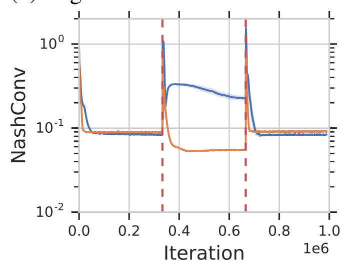
(b) Regularization coefficient 0.05



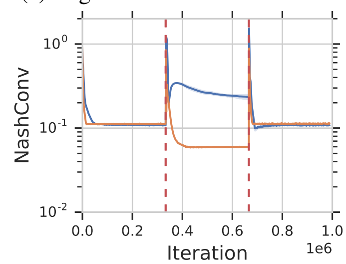
(c) Regularization coefficient 0.1



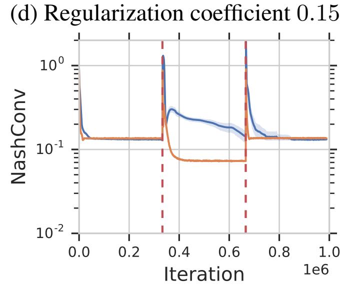
(d) Regularization coefficient 0.15



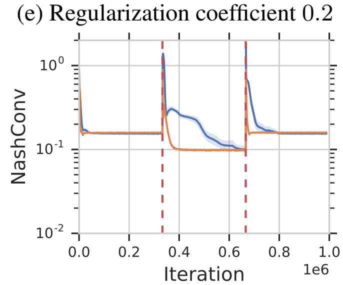
(e) Regularization coefficient 0.2



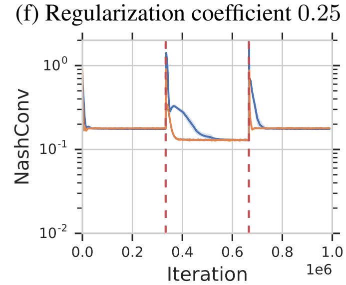
(f) Regularization coefficient 0.25



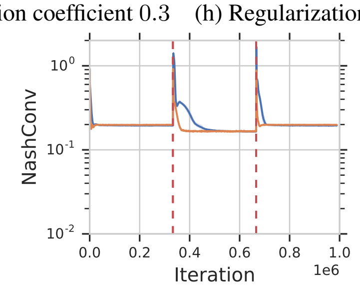
(g) Regularization coefficient 0.3



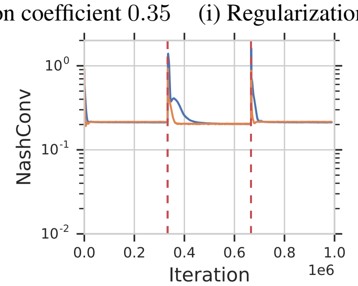
(h) Regularization coefficient 0.35



(i) Regularization coefficient 0.4

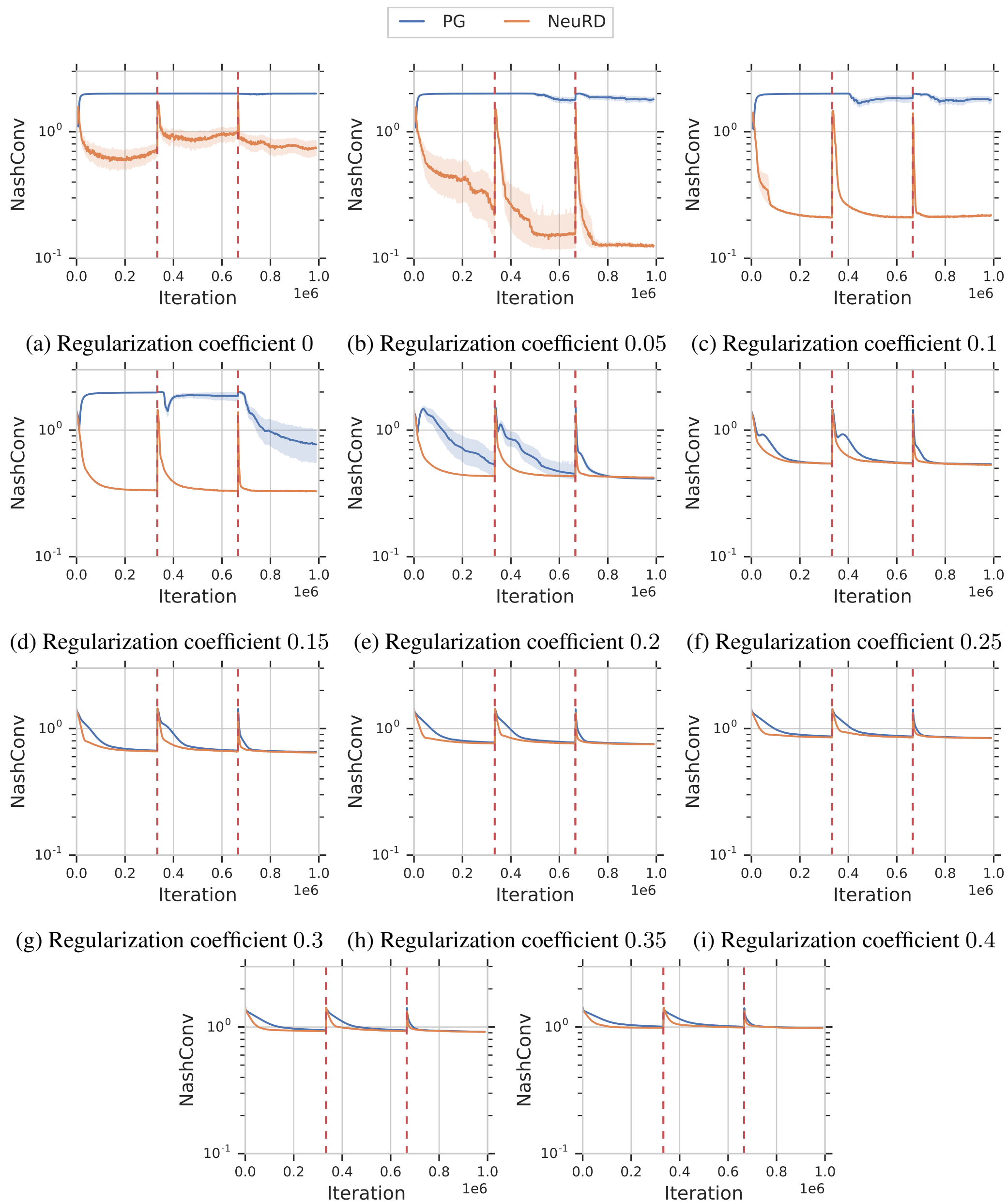


(j) Regularization coefficient 0.45



(k) Regularization coefficient 0.5

Figure D.7: Comparison of NeuRD and vanilla PG in Kuhn Poker with different levels of entropy regularization. The three game phases (each with a different reward function) are separated by the vertical dashed red lines.



(a) Regularization coefficient 0
(b) Regularization coefficient 0.05
(c) Regularization coefficient 0.1
(d) Regularization coefficient 0.15
(e) Regularization coefficient 0.2
(f) Regularization coefficient 0.25
(g) Regularization coefficient 0.3
(h) Regularization coefficient 0.35
(i) Regularization coefficient 0.4
(j) Regularization coefficient 0.45
(k) Regularization coefficient 0.5

Figure D.8: Comparison of NeuRD and vanilla PG in Goofspiel with different levels of entropy regularization. The three game phases (each with a different reward function) are separated by the vertical dashed red lines.

D.3 Single agent RL experiments

Though not considered a main focus in this paper, we also explore how NeuRD behaves in the single agent stationary RL setting. Specifically, we consider IMPALA [17], a state-of-the-art policy gradient based distributed learning agent, as a baseline. We replace the IMPALA policy update rule with that of NeuRD, otherwise keeping the training procedures the same as those used by Espeholt et al. [17]. We then compare the two algorithms in a suite of DMLab-30 [3] tasks. All IMPALA scores correspond to the 'experts' (i.e., non-multitask) results reported by Espeholt et al. [17, Table B.1].

Of particular note is the `rooms_select_nonmatching_object` domain, where NeuRD attains a score of 48.9 ± 4.0, which is significantly higher than IMPALA's score of 7.3. We conjecture this is due to the inherent adaptivity required on behalf of the agent to maximize its reward. In this domain, the agent spawns in a room, and views an out-of-reach object shown in one of two displays. If the agent reaches a specific pad in the room, it receives a $+1$ reward, and is spawned to a second room with two objects, one of which was the one in the previous room. The agent receives a reward of $+10$ (respectively, -10) if it collects object not matching (respectively, matching) the one in the previous room, then respawns to the first room. The episode ends after 12 seconds. This domain can be viewed as a contextual two-arm bandit problem, where the agent needs to learn to choose the non-matching object based on the context provided in the first room, rather than converge to a deterministic policy that always chooses the object that yielded the first reward. We, therefore, conjecture that the adaptivity properties of NeuRD enable it to attain the high reward in contrast to IMPALA. These preliminary results indicate that investigation of NeuRD's performance in related nonstationary single-agent settings may make for an interesting avenue of future work.

Level name	Human	Random	IMPALA	NeuRD
`rooms_select_nonmatching_object`	65.9	0.3	7.3	48.9±4.0
`rooms_watermaze`	54.0	4.1	26.9	29.8±0.4
`rooms_keys_doors_puzzle`	53.8	4.1	28.0	24.2±1.2
`lasertag_one_opponent_small`	18.6	-0.1	-0.1	0.0±0.0
`lasertag_three_opponents_small`	31.5	-0.1	19.1	0.0±0.0
`lasertag_one_opponent_large`	12.7	-0.2	-0.2	0.0±0.0
`lasertag_three_opponents_large`	18.6	-0.2	-0.1	0.0±0.0
`explore_goal_locations_small`	267.5	7.7	209.4	208.6±0.4
`explore_object_locations_small`	74.5	3.6	57.8	54.0±0.4
`explore_object_locations_large`	65.7	4.7	37.0	42.8±0.8
`natlab_fixed_large_map`	36.9	2.2	34.7	22.5±4.6
`natlab_varying_map_regrowth`	24.4	3.0	20.7	22.6±0.7
`natlab_varying_map_randomized`	42.4	7.3	36.1	37.2±1.3
`psychlab_arbitrary_visuomotor_mapping`	58.8	0.2	16.4	16.5±0.6
`psychlab_continuous_recognition`	58.3	0.2	29.9	30.7±0.1
`psychlab_sequential_comparison`	39.5	0.1	0.0	0.0±0.0

Table 1: Comparison of IMPALA and NeuRD performance in single-agent stationary RL, for various DMLab-30 domains. Both trained for $333e6$ environment frames in each domain. Human and random results are those reported by Espeholt et al. [17]. IMPALA scores correspond to the 'experts' (i.e., non-multitask) results reported by Espeholt et al. [17, Table B.1]. NeuRD scores are reported for top-3 of 20 hyperparameter seeds.

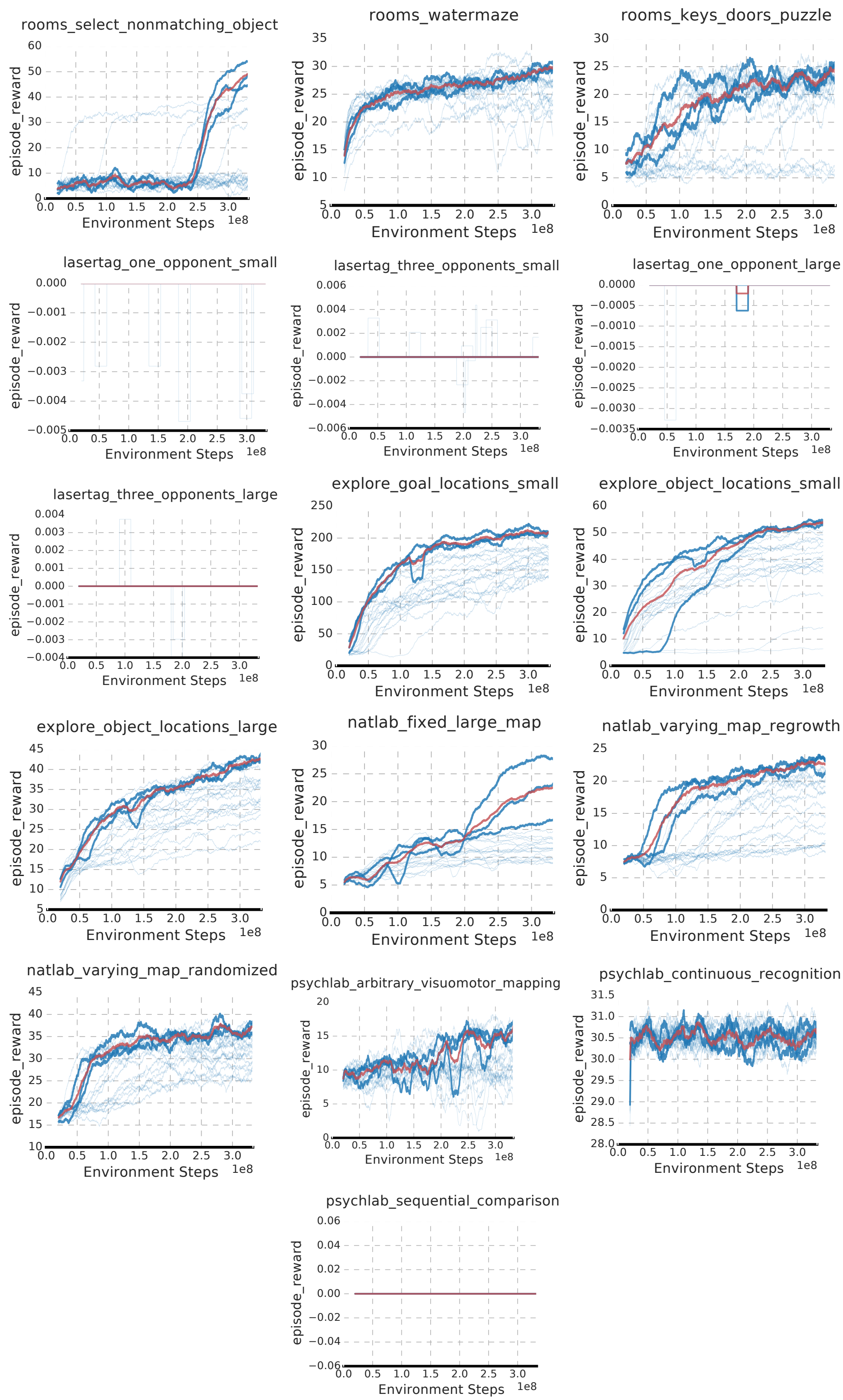


Figure D.9: NeuRD learning curves in single-agent stationary RL, for various DMLab-30 domains. Results reported for 20 hyperparameter seeds, with raw performance of top-3 seeds in dark blue, mean performance of top-3 seeds in red, and remaining seeds in light blue.