Neural Replicator Dynamics:
Multiagent Learning via Hedging Policy Gradients

Daniel Hennes*†
hennes@google.com

Dustin Morrill*‡
morrill@ualberta.ca

Shayegan Omidshafiei*†
somidshafiei@google.com

Rémi Munos†
munos@google.com

Julien Perolat†
perolat@google.com

Marc Lanctot†
lanctot@google.com

Audrunas Gruslys†
audrunas@google.com

Jean-Baptiste Lespiau†
jblespiau@google.com

Paavo Parmas‡
paavo.parmas@oist.jp

Edgar Duéñez-Guzmán†
duenez@google.com

Karl Tuyls†
karltuyls@google.com

ABSTRACT

Policy gradient and actor-critic algorithms form the basis of many commonly used training techniques in deep reinforcement learning. Using these algorithms in multiagent environments poses problems such as nonstationarity and instability. In this paper, we first demonstrate that standard softmax-based policy gradient can be prone to poor performance in the presence of even the most benign nonstationarity. By contrast, it is known that the replicator dynamics, a well-studied model from evolutionary game theory, eliminates dominated strategies and exhibits convergence of the time-averaged trajectories to interior Nash equilibria in zero-sum games. Thus, using the replicator dynamics as a foundation, we derive an elegant one-line change to policy gradient methods that simply bypasses the gradient step through the softmax, yielding a new algorithm titled *Neural Replicator Dynamics (NeuRD)*. NeuRD reduces to the exponential weights/Hedge algorithm in the single-state all-actions case. Additionally, NeuRD has formal equivalence to softmax counterfactual regret minimization, which guarantees convergence in the sequential tabular case. Importantly, our algorithm provides a straightforward way of extending the replicator dynamics to the function approximation setting. Empirical results show that NeuRD quickly adapts to nonstationarities, outperforming policy gradient significantly in both tabular and function approximation settings, when evaluated on the standard imperfect information benchmarks of Kuhn Poker, Leduc Poker, and Goofspiel.

KEYWORDS

multiagent; reinforcement learning; regret minimization; games

---

*Equal contributors.
†DeepMind.
‡Work done during an internship at DeepMind.

1 INTRODUCTION

Policy gradient (PG) algorithms form the foundation of many scalable approaches driving the field of deep reinforcement learning (RL) [19, 40, 50, 59, 60]. Agents using PG-based algorithms have learned to navigate in complex 3D worlds, play a wide array of video games, and learned to simulate humanoid locomotion on high-dimensional continuous control problems. The problem of *multiagent reinforcement learning* (MARL) [15, 54, 72], which involves several agents acting and learning simultaneously, is significantly more challenging because each agent perceives its environment as nonstationary [45, 72]. There have been several extensions or applications of PG algorithms to the multiagent setting, with remarkable success [1, 3, 20, 21, 43]. Given the wide use of PG and related variants in practice, it is paramount to understand its behavior and potential failure modes.

In partially-observable zero-sum games, such as poker, traditional approaches using expert knowledge or search with a perfect model have scaled to very large games [8, 11, 12, 51]. However, there have been RL-inspired advances as well: for example Regression Counterfactual Regret Minimization (RCFR) [18, 75], Neural Fictitious Self-Play (NFSP) [26], Policy-Space Response Oracles (PSRO) [38], Deep Counterfactual Regret Minimization [10], Double Neural CFR [39], and Exploitability Descent (ED) [42]. Recent work proposed Regret PGs (RPG) [64], an entirely model-free RL approach with formal relationships established between action-values and counterfactual values used in tabular regret minimization algorithms for partially observable zero-sum games. With a tabular representation, we can achieve regret minimization guarantees with the *Hedge* algorithm [22, 41][1] or its bandit version, *Exp3* [2], but they are not trivially extended to the function approximation case.

Evolutionary game theory (EGT) studies how populations of individuals interact strategically [30, 46, 76]. EGT requires minimal knowledge of opponent strategies or preferences, and has been important for the analysis and evaluation of MARL agents [53, 57, 68–70]. Formal connections have been made between MARL and EGT [6, 71], and population-based training regimes have been critical in scaling MARL to complex multiagent environments [4, 31, 73]. Central to EGT is the standard *replicator dynamics* (RD): a dynamical system that increases/decreases the tendency of the population

---

[1]This algorithm has many names including exponential weights, multiplicative weights, and entropic mirror descent [5].

toward playing strategies that would give high/low payoffs, compared to the average population value.

In this paper, we introduce a novel algorithm, *Neural Replicator Dynamics* (NeuRD), that corresponds to a parameterization of RD. Specifically, NeuRD is a fully sample-based and model-free algorithm for sequential nonstationary environments which is not restricted to the case of perfect information Markov/stochastic (simultaneous move) games [27], does not require sequence-form representations [55], and is fully compatible with general function approximation (e.g., deep neural networks). The new algorithm is an elegant one-line modification of the standard softmax PG, which effectively skips the gradient through the final softmax layer. An important result of this transformation is the ability to respond more dynamically to environmental nonstationarities. NeuRD is provably no-regret, which yields time-average convergence to approximate Nash equilibria in partially-observable zero-sum games under standard softmax policy representations. Our results validate these claims empirically in several partially observable games, even with compounded nonstationarity in the environment in the form of a dynamically-changing reward function.

1.1 Motivating Example

Softmax Policy Gradient (SPG) is usually deployed in complex domains, implemented with function approximation, and trained via stochastic gradients. Here we detail a simple case where SPG fails to be reasonably adaptive even with these complexities removed.

Consider *matching pennies*, a two-player zero-sum game with two actions, $\mathcal{H}$ and $\mathcal{T}$. If both players choose the same action, the "even" player receives a reward of 1 and the "odd" player receives a reward of -1, and vice versa if the players choose opposing actions. We will simplify the problem further by considering only the policy for the "even" player and fixing the "odd" player's policy in each playout: given a number of rounds to play in advance, the "odd" player will play $\mathcal{H}$ for the first 40% of the rounds and play $\mathcal{T}$ for the remaining 60%. We will consider the all-actions version of the repeated game, where players are evaluated by their cumulative expected reward and observe their entire reward functions.

What would be a good score for the "even" player to achieve? A starting point is to look at the value of fixed pure strategies. Playing $\mathcal{H}$ over T-rounds achieves a value of $-0.2T$, while playing $\mathcal{T}$ achieves $0.2T$. The difference between the player's value and the cumulative reward of the best fixed action in hindsight is called *regret*, and ensuring that this difference does not increase with the horizon length is evidence that an algorithm can robustly learn to distinguish between good and bad actions in a reasonable time.

Let us now compare SPG to Hedge [22, 41], a well-known no-regret algorithm. While Hedge has near-constant regret[2], SPG's regret grows with the horizon length, even if the step size is independently tuned for each T. The growth rate is roughly a function of $\sqrt{T}$ though, which matches the order of the best worst-case performance bound.

However, the difference is amplified, if we simply add a nuisance action choice for the "even" player. This action only has the effect of forfeiting the round to the "odd" player, so its reward is always -1. Figure 1 shows that this drastically increases the regret growth-rate

[2]In fact, we can ensure that Hedge achieves at most a constant regret of 1 for any horizon length T as the "even" player by simply choosing a large step size.
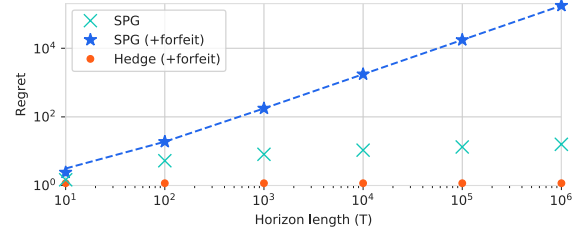

Figure 1: The regret of SPG with and without a forfeit action in repeated matching pennies compared to Hedge. The dashed line is a linear least-squares fit.

for SPG to linear, which is larger than worst-case bounds provided by other algorithms (including Hedge) by a factor of $\sqrt{T}$. In practice, Hedge's regret remains *constant*. This example, trivially, shows that even in the simplest of non-stationary environments SPG does not respond to changes in a reasonable manner, while other simple algorithms perform well. We will revisit this example in Section 3.1 and further explore why SPG's regret grows so rapidly.

2 PRELIMINARIES

We first establish prerequisite definitions and notation before presenting our technical analysis of SPG and the proposed algorithm.

2.1 Game Theory

Game theory studies strategic interactions of players. A *normal-form game* (NFG) specifies the interaction of K players with corresponding action sets $\{\mathcal{A}^1, \dots, \mathcal{A}^K\}$. 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 $\mathbf{a} \doteq (a^1, \dots, a^K)$, where $a^k \in \mathcal{A}^k$ for all $k \in [K] \doteq \{1, \dots, K\}$. 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 $\boldsymbol{\pi} \doteq (\boldsymbol{\pi}^1, \dots, \boldsymbol{\pi}^K)$ 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 $k \in [K]$. We use the Nash Convergence metric (NASHCONV) [38] to evaluate learned policies:

$$\mathrm{NASHCONV}(\boldsymbol{\pi}) = \sum_k \bar{u}^k((\mathrm{BR}^k(\boldsymbol{\pi}^{-k}), \boldsymbol{\pi}^{-k})) - \bar{u}^k(\boldsymbol{\pi}) \quad (1)$$

Roughly speaking, NASHCONV measures 'distance' of $\boldsymbol{\pi}$ to a Nash equilibrium (i.e., lower NASHCONV is better).

2.2 Replicator Dynamics (RD)

Replicator Dynamics (RD) is a concept from EGT that describe a population's evolution via biologically-inspired operators, such as selection and mutation [30, 66, 67, 77, 78]. The single-population RD are defined by the following system of differential equations:

$$\dot{\pi}(a) = \pi(a)\big[u(a, \boldsymbol{\pi}) - \bar{u}(\boldsymbol{\pi})\big] \qquad \forall a \in \mathcal{A}, \quad (2)$$

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

2.3 Online Learning

Online learning examines the performance of learning 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, $\pi_t \in \Delta^{|\mathcal{A}|}$, and receives utility, $u_t(a) \in \mathbb{R}$, where $u_t \in \mathbb{R}^{|\mathcal{A}|}$ 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 observing T rounds; the regret is defined as $R_T(a) \doteq \sum_{t=1}^{T} u_t(a) - \pi_t \cdot u_t$. Algorithms that guarantee their average worst-case regret goes to zero as the number of rounds increases, i.e., $R_T = o(T)$, are called *no-regret*; these algorithms learn optimal policies under fixed or stochastic environments. According to a folk theorem, <u>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</u> [7]. This result can be extended to sequential imperfect information games by composing learners in a tree and defining utility as counterfactual value [28, 79]. We refer the interested reader to [64] for additional background on sequential games.

The family of no-regret algorithms known as Follow the Regularized Leader (FoReL) [47, 48, 61, 62] generalizes well-known decision making algorithms and population dynamics. For a discrete action set, $\mathcal{A}$, FoReL is defined through the following updates:

$$\pi_t \doteq \underset{\pi' \in \Delta^{|\mathcal{A}|}}{\arg\max} \left[\pi' \cdot y_{t-1} - h(\pi') \right], \qquad y_t \doteq y_{t-1} + \eta_t u_t,$$

where $\eta_t > 0$ is the learning rate at timestep t, $u_t \in \mathbb{R}^{|\mathcal{A}|}$ is the utilities vector observed at t, $y_t \in \mathbb{R}^{|\mathcal{A}|}$ are the accumulated values at t, and regularizer h is a convex function. Note that FoReL 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 *all-actions* setting.

Under negative entropy regularization $h(\pi) = \sum_a \pi(a) \log \pi(a)$, policy π_t reduces to a softmax function $\pi_t \doteq \Pi(y_{t-1})$, where $\Pi(z) \propto \exp(z), \forall z \in \mathbb{R}^{|\mathcal{A}|}$. This yields the Hedge algorithm:

$$\pi_T \doteq \Pi \left(\sum_{t=1}^{T-1} \eta_t u_t \right). \tag{3}$$

Hedge is no-regret when η_t is chosen carefully, e.g., $\eta_t \in \Theta(1/\sqrt{t})$ [52]. Likewise, the continuous-time FoReL dynamics [49] are

$$\pi \doteq \underset{\pi' \in \Delta^{|\mathcal{A}|}}{\arg\max} \left[\pi' \cdot y - h(\pi') \right], \qquad \dot{y} \doteq u, \tag{4}$$

which in the case of entropy regularization yield RD as defined in (2) (e.g., see Mertikopoulos et al. [49]). This implies that RD is no-regret, thereby enjoying equilibration to Nash and convergence to the optimal prediction in the time-average.

2.4 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 $\pi : s \to \Delta^{|\mathcal{A}|}$, which maximizes the expected return $v^{\pi}(s) = \mathbb{E}_{\pi}[\sum_{k=t}^{\infty} \gamma^{k-t} r_k | s_t = s]$, with discount factor $\gamma \in [0, 1)$. In actor-critic algorithms, one generates trajectories according to some parameterized policy $\pi(\cdot|s; \theta)$ while learning to estimate the action-value function $q^{\pi}(s, a) = \mathbb{E}_{\pi}[\sum_{k=t}^{\infty} \gamma^{k-t} r_k | s_t = s, a_t = a]$. Temporal difference learning [65] can be used to learn an action-value
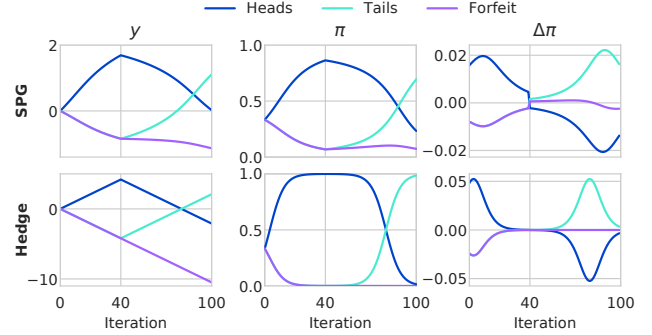
Figure 2: The logit and policy trajectories of SPG and Hedge in all-actions, 100-round, repeated matching pennies with a forfeit action. The vertical lines mark the change in the opponent's policy at 40-rounds. The step size $\eta = 0.21$ was optimized in a parameter sweep for SPG with $T = 100$.

function estimator, $q(s, a; w)$, which is parameterized by w. A PG algorithm then updates policy π parameters θ in the direction of the gradient $\nabla_{\theta} \log \pi(a|s; \theta) \left[q(s, a; w) - v(s; w) \right]$ for a given state-action pair (s, a), where the quantity in square brackets is defined as the *advantage*, denoted $A(a; \theta, w)$, and $v(s; w) \doteq \sum_{a'} \pi(s, a'; \theta) q(s, a'; w)$. <u>The advantage is analogous to regret in the online learning literature.</u> In sample-based learning, the PG update incorporates a $(\pi(a|s; \theta))^{-1}$ factor that accounts for the fact that a was sampled from π. The all-actions PG update without this factor is then

$$\theta_t = \theta_{t-1} + \eta_t \sum_a \nabla_{\theta} \pi(a|s; \theta_{t-1}) \left[q(s, a; w) - v(s; w) \right]. \tag{5}$$

While different policy parameterizations are possible, the most common choice for discrete decision problems is a softmax function over the *logits* y: $\pi_t(\theta_t) = \Pi(y(\theta_t))$. Thus, we focus the rest of our analysis on Softmax Policy Gradient (SPG).

3 A UNIFYING PERSPECTIVE ON RD AND PG

This section motivates and presents a novel algorithm, Neural Replicator Dynamics (NeuRD), and unifying theoretical results.

3.1 A Close-up on Learning Dynamics

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, SPG is applicable to high-dimensional single and multiagent RL domains. Unfortunately, SPG suffers from the fact that increasing the probability of taking an action that 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 SPG update explicitly, using the notation of online learning to identify correspondences to that literature. On round t, SPG updates its logits and policy as

$$\pi_t \doteq \Pi(y_{t-1}), \qquad y_t \doteq y_{t-1} + \eta_t \nabla_{y_{t-1}} \pi_t \cdot u_t.$$

As there is no action or state sampling in this setting, shifting all the payoffs by the expected value $\bar{u}$ (or v^{π} in RL terms) has no impact on the policy, so this term is omitted above. Noting that $\partial \pi_t(a') / \partial y_{t-1}(a) = \pi_t(a') [\mathbb{1}_{a'=a} - \pi_t(a)]$ [65, Section 2.8], we
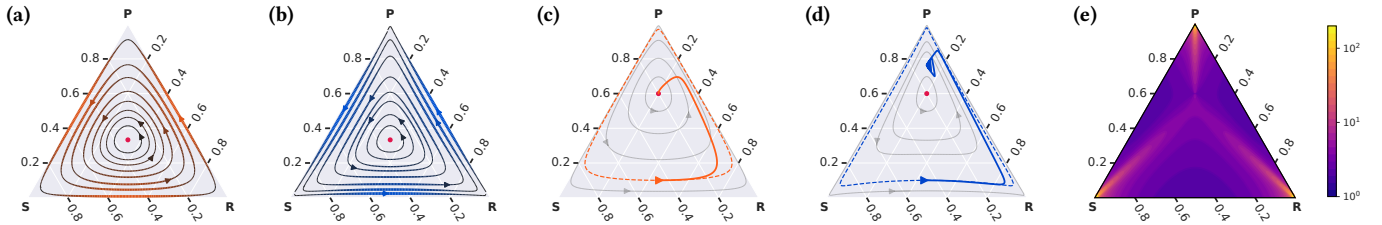
Figure 3: Learning dynamics of (a) RD and (b) SPG in *Rock–Paper-Scissors* (RPS). Time-averaged trajectories (solid lines) are shown in (c) for RD and in (d) for SPG in the biased-RPS game. In (e) we compare their rate of adaptation, i.e., $\|\dot{\boldsymbol{\pi}}_{\text{RD}}\|/\|\dot{\boldsymbol{\pi}}_{\text{PG}}\|$.

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:

$$y_t(a) = y_{t-1}(a) + \eta_t \pi_t(a)\big[u_t(a) - \bar{u}_t\big] \quad \forall a \in \mathcal{A}. \quad (6)$$

See Section A.1 for details. Scaling the regret by $\boldsymbol{\pi}_t$ leads to an update that can prevent SPG from achieving reasonable performance.

Specifically, the additional scaling factor $\boldsymbol{\pi}_t$ 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. This is precisely why SPG fails to adapt in our motivating example (Section 1.1): at the switch point $t = 0.4T$, the rate of increase of the logit corresponding to $\mathcal{T}$ is modulated by a low action probability. Likewise, the rate of decrease of the logit corresponding to $\mathcal{H}$ is modulated by a low instantaneous regret, i.e., $\bar{u}_t$ is close to -1 as the probability of playing $\mathcal{H}$ is close to 1 at the switching point. Thus, we clearly see a difference in the slopes of the logit trajectories for SPG and Hedge in Fig. 2, where SPG reacts slower. We also observe that the logit for the forfeit action is decreasing very slowly after the switch, as both the action probability and the instantaneous regret are small. This decreases the logit gap (i.e., the range between largest and smallest logit), thereby effectively increases probability mass on the undesirable forfeit action. Without a nuisance action, the logit gap would quickly reduce which in turn would reduce the probability mass on $\mathcal{H}$ due to the softmax projection. With the added asymmetry caused by the presence of a forfeit action, the logit gap remains larger, thus leaving more probability mass on $\mathcal{H}$. Notice that this is in stark contrast to the behavior of Hedge, where the logit for the forfeit action keeps decreasing, thus maintaining a constant logit gap until $\mathcal{T}$ becomes the preferred action.

The scaling by $\boldsymbol{\pi}_t$ is also apparent when taking the continuous-time limit of SPG dynamics. Consider the continuous-time q-value based policy gradient dynamics [64, Section D.1.1, QPG], which are amenable for comparison against RD:

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

conducted for all $a \in \mathcal{A}$. In contrast to RD (2), the SPG dynamics in (7) have an additional $\pi(a; \boldsymbol{\theta})$ term that modulates learning.

The issue arising due to regret scaling manifests itself also when considering convergence to Nash equilibria in games, even under continuous-time dynamics. We illustrate this in the game of *Rock–Paper–Scissors*, by respectively comparing the continuous-time dynamics of RD and SPG in Figs. 3a and 3b. The game is intransitive, thus resulting in trajectories that cycle around the Nash equilibrium. In Fig. 3c and Fig. 3d, we show time-averaged trajectories in a biased version of RPS (see Table 1 with $\nu = 3$). The

time-averaged trajectories of RD converge to interior Nash equilibria in zero-sum games Hofbauer et al. [29]. However, even in this simple game, SPG does not converge to Nash in the time-average. As a further comparison, Fig. 3e plots the ratio of their speeds, i.e., $\|\dot{\boldsymbol{\pi}}_{\text{RD}}\|/\|\dot{\boldsymbol{\pi}}_{\text{PG}}\|$ in the biased version of the game. The differences in updates are compounded near the simplex vertices, where a single action retains a majority of policy mass. This difference causes practical issues when using SPG in settings where learning has converged to a near-deterministic policy and then must adapt to a different policy given (e.g., under dynamic payoffs or opponents). While SPG fails to adapt rapidly to the game at hand due to its extra downscaling by factor $\boldsymbol{\pi}$, 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 SPG update rule in RL applications.

3.2 NeuRD: Neural Replicator Dynamics

While we have highlighted key limitations of SPG in comparison to RD, the latter has limited practicality when computational updates are inherently discrete-time or a parameterized policy is useful 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 SPG. For seamless comparison of our update rule to SPG, 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 start by unifying notations, specifically reformulating the RD dynamics (2) in RL terms as

$$\dot{\pi}(a) = \pi(a)[q^{\boldsymbol{\pi}}(a) - v^{\boldsymbol{\pi}}].$$

As RD aligns with FoReL with entropy regularization, we can further write the RD logit dynamics using (4) as

$$\dot{y}(a) = q^{\boldsymbol{\pi}}(a) - v^{\boldsymbol{\pi}}, \quad (8)$$

where $v^{\boldsymbol{\pi}}$ is the variance-reducing baseline [65]. Let $y(a; \boldsymbol{\theta}_t)$ denote the logits parameterized by $\boldsymbol{\theta}_t$. A natural way to derive a parametric update rule is to compute the Euler discretization[3] of (8),

$$y_t(a) \doteq y(a; \boldsymbol{\theta}_{t-1}) + \eta_t \big(q^{\boldsymbol{\pi}_t}(a) - v^{\boldsymbol{\pi}_t}\big), \quad (9)$$

and consider $y_t(a)$ a fixed *target value* that the parameterized logits $y(a; \boldsymbol{\theta}_{t-1})$ are adjusted toward. Namely, one can update $\boldsymbol{\theta}$ to

---

[3]Given a tabular softmax policy, this definition matches the standard discrete-time RD. See Section A.5.

Algorithm 1 Neural Replicator Dynamics (NeuRD)

1: Initialize policy weights $\boldsymbol{\theta}_0$ and critic weights $\boldsymbol{w}_0$.
2: **for** $t \in \{1, 2, \dots\}$ **do**
3:    $\boldsymbol{\pi}_{t-1}(\boldsymbol{\theta}_{t-1}) \leftarrow \Pi(\boldsymbol{y}(\boldsymbol{\theta}_{t-1}))$
4:    **for all** $\tau \in \textsf{SampleTrajectories}(\boldsymbol{\pi}_{t-1})$ **do**
5:      **for** $s, a \in \tau$ **do**   ▷ POLICY EVALUATION
6:        $R \leftarrow \textsf{Return}(s, \tau, \gamma)$
7:        $\boldsymbol{w}_t \leftarrow \textsf{UpdateCritic}(\boldsymbol{w}_{t-1}, s, a, R)$
8:      **for** $s \in \tau$ **do**   ▷ POLICY IMPROVEMENT
9:        $v(s; \boldsymbol{w}_t) \leftarrow \sum_{a'} \pi(s, a'; \boldsymbol{\theta}_{t-1}) \, q_t(s, a'; \boldsymbol{w}_t)$
10:       $\boldsymbol{\theta}_t \leftarrow \boldsymbol{\theta}_{t-1} + \eta_t \sum_{a'} \nabla_{\boldsymbol{\theta}} y(s, a'; \boldsymbol{\theta}_{t-1})\big(q_t(s, a'; \boldsymbol{w}_t) - v(s; \boldsymbol{w}_t)\big)$

---

minimize a choice of metric $d(\cdot, \cdot)$,

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

In particular, minimizing the Euclidean distance yields,

$$\begin{aligned}
\boldsymbol{\theta}_t &= \boldsymbol{\theta}_{t-1} - \sum_a \nabla_{\boldsymbol{\theta}} \frac{1}{2}\|y_t(a) - y(a; \boldsymbol{\theta}_{t-1})\|^2 \\
&= \boldsymbol{\theta}_{t-1} + \sum_a (y_t(a) - y(a; \boldsymbol{\theta}_{t-1})) \nabla_{\boldsymbol{\theta}} y(a; \boldsymbol{\theta}_{t-1}) \\
&\overset{(9)}{=} \boldsymbol{\theta}_{t-1} + \eta_t \sum_a \nabla_{\boldsymbol{\theta}} y(a; \boldsymbol{\theta}_{t-1})(q^{\boldsymbol{\pi}}(a) - v^{\boldsymbol{\pi}}),
\end{aligned} \tag{10}$$

which we later prove has a rigorous connection to Hedge and, thus, inherits no-regret guarantees that are useful in nonstationary settings such as games. Update rule (10) applies to all differentiable policy parameterizations. However, as our core experiments use neural networks, we henceforth refer to the update rule (10) as Neural Replicator Dynamics (NeuRD), with pseudocode shown in Algorithm 1. NeuRD effectively corresponds to a 'one-line fix' of SPG, in that replacing Algorithm 1 line 10 with (5) yields SPG.

3.3 Properties and Unifying Theorems

Overall, NeuRD is not only practical to use as it involves a simple modification of SPG 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 the appendix.

We first state a well-known relationship[4] between Hedge and replicator dynamics [34, 58, 74]:

STATEMENT 1. *The following are equivalent: a) Hedge [22, 41], b) discrete-time RD [17], and c) single state, all-actions, tabular NeuRD.*

We now extend this equivalence to the sequential decision-making setting, providing convergence guarantees in the tabular setting. We first need a few definitions; see Srinivasan et al. [64] for reference. First, for any given joint policy, $\boldsymbol{\pi} = \{\boldsymbol{\pi}_i\}_{i=1}^N$, define the reach probability $\rho^{\boldsymbol{\pi}}(h) = \rho_i^{\boldsymbol{\pi}}(h)\rho_{-i}^{\boldsymbol{\pi}}(h)$ as a product of probabilities of all agents' policies along the history of actions h (including chance/nature). Reach probabilities can be split up into agent i's contribution, $\rho_i^{\boldsymbol{\pi}}$, and those of the other agents, $\rho_{-i}^{\boldsymbol{\pi}}$. We make the standard assumption of *perfect recall*, so no agent ever forgets any information they have observed along h; an *information state*, s, is a set of histories consistent with the observations of the agent to act at s. Thus, agent i's policy must be defined on information states, $\boldsymbol{\pi}_i(s) \in \Delta^{|\mathcal{A}(s)|}$, where $\mathcal{A}(s)$ is the set of actions available at

---

[4]This relationship also holds for the continuous-time variants as well [29].

---

information state s, but we can overload this notation to say that agent i's policy at history $h \in s$ is $\boldsymbol{\pi}_i(h) = \boldsymbol{\pi}_i(s)$.

Due to perfect recall, if agent i is to act at h, $\rho_i^{\boldsymbol{\pi}}(h)$ is the same for all $h \in s$, so we refer to it as $\rho_i^{\boldsymbol{\pi}}(s)$. With $\beta_{-i}(\boldsymbol{\pi}, s) = \sum_{h \in s} \rho_{-i}^{\boldsymbol{\pi}}(h)$ as the normalizing denominator, we define:

$$q_i^{\boldsymbol{\pi}}(s, a) = \frac{1}{\beta_{-i}(\boldsymbol{\pi}, s)} \sum_{h \in s} \rho_{-i}^{\boldsymbol{\pi}}(h) q_i^{\boldsymbol{\pi}}(h, a),$$

where $q_i^{\boldsymbol{\pi}}(h, a)$ is the expected value of playing a at history h and following $\boldsymbol{\pi}$ afterward. Let $v_i^{\boldsymbol{\pi}}(s) = \sum_{a \in \mathcal{A}} \pi_i(a|s) q_i^{\boldsymbol{\pi}}(s, a)$ be the expected value at information state s for agent i.

We can now state the following corollaries to Statement 1 that give regret and Nash equilibrium approximation bounds in the tabular case:

COROLLARY 3.1. *Consider a sequential decision making task with finite length histories and N-agents. Assume that agent i acts according to a softmax tabular policy, $\boldsymbol{\pi}_{i,t}(s) \propto \exp(\boldsymbol{y}_{i,t}(s))$, where $\boldsymbol{y}_{i,t}(s) \in \mathbb{R}^{|\mathcal{A}(s)|}$ is a vector of logits for the actions available at information state s, and the other agents act arbitrarily.*

Using NeuRD as the local learning algorithm in counterfactual regret minimization (CFR) *[79] results in the following local logit updates for agent i and all s, a:*

$$y_{i,t}(s, a) = y_{i,t-1}(s, a) + \eta_t(s)\beta_{-i}(\boldsymbol{\pi}_{t-1}, s)\left(q_i^{\boldsymbol{\pi}_{t-1}}(s, a) - v_i^{\boldsymbol{\pi}_{t-1}}(s)\right),$$

where $\beta_{-i}(\boldsymbol{\pi}_{t-1}, s)\left(q_i^{\boldsymbol{\pi}_{t-1}}(s, a) - v_i^{\boldsymbol{\pi}_{t-1}}(s)\right)$ are counterfactual regrets, and $\eta_t(s)$ is the stepsize for information state s at time t. These updates and the resulting local policies are identical to those of Hedge by Statement 1. Therefore, CFR(NeuRD) is equivalent to CFR(Hedge) [9, 79].

By the CFR Theorem [79] and the Hedge regret bound [16, 22], when the time-constant learning rate $\eta_t(s) = \sqrt{2\ln(|\mathcal{A}(s)|)T^{-1}}$ is used at each information state, a CFR(NeuRD) agent has its regret with respect to any fixed policy after T-updates, $R_{i,T}$, upper-bounded by $R_{i,T} \leq |\mathcal{S}_i|\Delta_u\sqrt{2\ln|\mathcal{A}|\,T}$, where $\mathcal{S}_i$ is the set of information states for agent i, $\Delta_u = \max_{z,z'} |u(z) - u(z')|$ for any two terminal histories, z, z', and $|\mathcal{A}|$ is the maximum number of actions at any of agent i's information states.

COROLLARY 3.2. *As stated in Section 2.3, Corollary 3.1 implies that in a two-player, zero-sum game, if each player i employs CFR(NeuRD), the policy generated from their average sequence weights, $\bar{\boldsymbol{\pi}}_{i,T}(s) \propto \frac{1}{T}\sum_{t=1}^T \rho_i^{\boldsymbol{\pi}_t}(s)\boldsymbol{\pi}_{i,t}(s)$, converges to an ε-Nash equilibrium, with ε upper-bounded by the maximum $R_{i,T}$ across players by the folk theorem [7].*

Corollary 3.1 and Corollary 3.2 show that NeuRD can be used to solve a broad class of problems where SPG may fail. Note that in the function approximation case, where we use $y(s, a; \boldsymbol{\theta}_t)$ to approximate the tabular logit for information state s and action a, one can still establish regret guarantees [18].

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 [32, 71].

THEOREM 1. *SPG 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) [33].

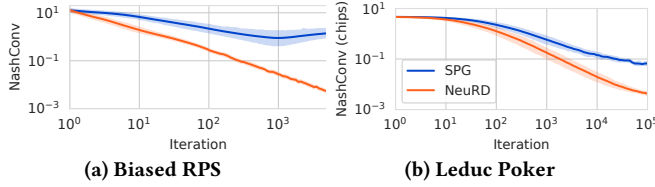(a) Biased RPS   (b) Leduc Poker

Figure 4: (a) NASHCONV of the average NeuRD and SPG policies in biased RPS. (b) NASHCONV of the sequence-probability average policies of tabular, all-actions, counterfactual value NeuRD and SPG in two-player Leduc Poker.

THEOREM 2. *The NeuRD update rule* (10) *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 [44].

REMARK. *As, on average, the logits $\boldsymbol{y}$ get incremented by the advantage, they may diverge to $\pm\infty$. To avoid numerical issues, one can stop increasing/decreasing the logits if the logit-gap exceeds a threshold. We apply this by using a clipping gradient, $\widehat{\nabla}_{\boldsymbol{\theta}}$, as follows:*

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

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 yields the update,

$$\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} + \widehat{\nabla}_{\boldsymbol{\theta}}\big(y(s, a; \boldsymbol{\theta}_{t-1})(q^{\boldsymbol{\pi}}(s, a) - v^{\boldsymbol{\pi}}(s)), \eta_t, \beta\big).$$

Note that logits 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.

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 added reward nonstationarity. As NeuRD involves only a simple modification of the SPG update rule to improve adaptivity, we focus our comparisons against SPG as a baseline, noting that additional benefits can be analogously gained by combining NeuRD with more intricate techniques that improve SPG (e.g., variance reduction, improved exploration, or off-policy learning).

We consider several domains. **Rock–Paper–Scissors (RPS)** is a canonical NFG involving two players. In **Kuhn Poker**, 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 pot. In **Leduc Poker** [63], 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 so at a showdown, the player whose card pairs the public card or has the highest-ranked card wins the pot. In **Goofspiel**, players try to obtain point cards by bidding simultaneously. We use an imperfect information variant
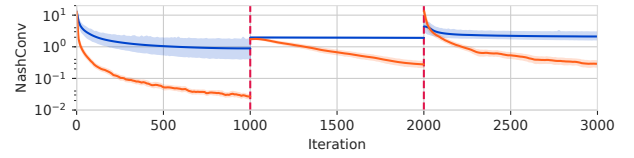

Figure 5: Time-average policy NASHCONV in nonstationary RPS, with the game phases separated by vertical red dashes.

with 5 cards where bid cards are not revealed [35]. The number of information states for the Kuhn, Leduc, and Goofspiel variants evaluated is 12, 936, and 2124, respectively; we use the OpenSpiel [37] implementations of these three games in our experiments.

We first show that the differences between NeuRD and SPG detailed in Section 3.3 are more than theoretical. Consider the NASHCONV of the time-average NeuRD and SPG tabular policies in the game of RPS, shown in Fig. 4a. 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 SPG eventually plateaus. The continuous time dynamics of NeuRD (8) and SPG (7) are integrated over time with step size $\Delta t = 0.1$ (equals 1 iteration). The figure shows the mean NASHCONV (1) 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.

Consider next a more complex imperfect information setting, where Fig. 4 shows that tabular, all-actions, counterfactual value NeuRD[5] more quickly and more closely approximates a Nash equilibrium in two-player Leduc Poker than tabular, all-actions, counterfactual value PG. 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 SPG algorithms in this case is the information-state logit update rule, and the only difference here–as described by (6)–is that SPG scales the NeuRD update by the current policy. The performance displayed is that of the sequence probability time-average policy for both algorithms. 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 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 v in Table 1.

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

Table 1: RPS payoffs.

Payoffs are shown for the first player only. We set v to 20, 0, and 20, respectively, for the three phases, with payoff switches happening every 1000 iterations. This effectively biases the Nash towards one of the simplex corners (see Fig. 3c), then to the center (Fig. 3a), then again towards the corner. Figure 5 plots the NASHCONV of NeuRD and SPG with respect to the Nash for that particular phase.

---

[5]This can be seen as counterfactual regret minimization (CFR) [79] with Hedge [9].

Despite the changing payoffs, the NeuRD NᴀꜱʜCᴏɴᴠ decreases in each of the phases, while SPG plateaus. We use the same setup detailed above for Fig. 4a for the nonstationary case.

We next 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 instead use entropy regularization of the form introduced by Perolat et al. [56] to induce realtime policy convergence towards the Nash. Specifically, Perolat et al. [56] show that the last iterate policy of FoReL will converge to the Nash equilibrium in zero-sum two-player imperfect information games when an additional entropy cost is used. We note that the empirical evaluation of QPG in Srinivasan et al. [64] corresponds to SPG in our work in that both are all-actions policy gradient with a softmax projection. However, the form of entropy regularization in Srinivasan et al. [64] uses the common approach of adding an entropy bonus to the policy loss, whereas here we use entropy regularization applied to the action-values $q^{\boldsymbol{\pi}}$ [56].

For each game in Fig. 6, we randomly initialize a policy parameterized by a two-layer neural network (128 hidden units). Results are reported for 40 random neural network initializations for both NeuRD and PG. We update the policy once every 4 updates 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 0.33e6 learning iterations (with the three reward function phases separated by the vertical red stripes in plots, where applicable). Upon conclusion of each learning phase, policies are not reset; instead, learning continues given the latest policy (for each of the 40 trials).

Figures 6a to 6c illustrate the NᴀꜱʜCᴏɴᴠ for NeuRD and SPG in all imperfect information games considered, with an entropy regularization sweep conducted for each algorithm to yield the lowest *final* NᴀꜱʜCᴏɴᴠ. Notably, NeuRD converges faster than SPG in all three domains, and yields the lowest final NᴀꜱʜCᴏɴᴠ. To analyze the role of entropy regularization on the convergence rate, we next consider the average NᴀꜱʜCᴏɴᴠ over *all* training iterations and phases of each game. Figures 6d to 6f plot the average NᴀꜱʜCᴏɴᴠ for each game, across all considered entropy regularization levels. Notably, NeuRD consistently has lower average NᴀꜱʜCᴏɴᴠ and is less sensitive to the entropy regularization level, in contrast to PG.

Overall, the key takeaway of these experiments is SPG's weakness when learning in nonstationary domains. Critically, this weakness is effectively addressed through a simple 'one-line change' in using the NeuRD update rule (10) in lieu of the standard policy gradients update, as illustrated in Algorithm 1.

5 DISCUSSION

We established here unifying, rigorous links between replicator dynamics, policy gradient methods, and online learning. We began by demonstrating that commonly-used SPG methods face major adaptivity issues, even in the simplest of nonstationary domains. The insights gained led to development of a novel algorithm, NeuRD, which generalizes the no-regret Hedge algorithm and RD to utilize

function approximation. NeuRD is thus theoretically grounded and adaptive while still enjoying the practical benefits of parametric SPG methods. The key advantage of NeuRD is that it corresponds to a simple 'one-line fix' of standard PG algorithms.

NeuRD was established to have a number of theoretical properties, including regret guarantees in the sequential setting, in addition to concrete theoretical links to SPG and Natural PG. We empirically showed NeuRD to significantly outperform SPG in numerous highly nonstationary and adversarial settings, ranging from tabular to imperfect information games. Moreover, in contrast to existing sequence-form dynamics [23, 24, 36, 55], use of counterfactual values in NeuRD enables representation of policies in behavioral form, as is standard in RL. As a result, it is straight-forward to conduct sampling and function approximation via NeuRD.

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 involves a very simple modification of the SPG update rule, a promising avenue for future work is to investigate NeuRD-based extensions of standard PG-based methods (e.g., A3C [50], DDPG [40], and MADDPG [43]), opening the door to a large class of new and potentially performative algorithms. Additionally, it seems sensible to investigate nonstationary single-agent RL tasks such as intrinsic motivation-based exploration [25].

APPENDICES

A.1 Single state, tabular SPG update. Here, we fully derive the single state, tabular SPG update. On round t, SPG 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 $\bar{u}$ has no impact on the policy, so this term is omitted here. Noting that $\partial \pi_t(a') / \partial y_{t-1}(a) = \pi_t(a') [\mathbb{1}_{a'=a} - \pi_t(a)]$ [65, 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 $\boldsymbol{\pi}_t$:

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

Therefore, the concrete update is:

$$
\begin{aligned}
y_t(a) &= y_{t-1}(a) + \eta_t \pi_t(a) \Big[u_t(a) - \sum_{a'} \pi_t(a') u_t(a') \Big] \\
&= y_{t-1}(a) + \eta_t \pi_t(a) [u_t(a) - \bar{u}_t].
\end{aligned}
$$

A.2 Proof of Statement 1. In the single state, tabular case, $\nabla_{\boldsymbol{\theta}} y(a; \boldsymbol{\theta}_t)$ is the identity matrix, so unrolling the NeuRD update (10)
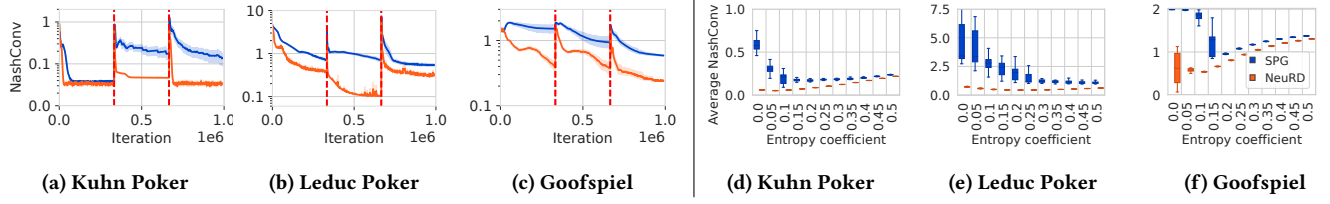
Figure 6: Comparison of NeuRD and SPG NashConv (left) and average NashConv over all iterations (right), for each game.

across $T - 1$ rounds, we see that the NeuRD policy is

$$\pi_T = \Pi\left(\sum_{t=1}^{T-1} \eta_t \left(\boldsymbol{u}_t - \boldsymbol{u}_t \cdot \pi_t\right)\right) = \Pi\left(\sum_{t=1}^{T-1} \eta_t \boldsymbol{u}_t\right), \quad (11)$$

as Π is shift invariant. As (11) is identical to (3), NeuRD and Hedge use the same policy on every round, thus are equivalent here.

A.3 Proof of Theorem 1. An Euler discretization of RD at the policy level is: $\boldsymbol{\pi}_{t+1} \doteq \boldsymbol{\pi}_t \odot \left[1 + \eta_t\left(\boldsymbol{q} - v^{\pi_t}\mathbf{1}\right)\right]$. Note that while $\sum_a \pi_{t+1}(a) = 1$, this Euler-discretized update may still be outside the simplex; however, π_{t+1} merely provides a target for our parameterized policy π_{θ_t} update, which is subsequently reprojected back to the simplex via $\Pi(\cdot)$.

Now if we consider parameterized policies $\pi_t \approx \pi_{\theta_t}$, and our goal is to define dynamics on θ_t that captures those of RD, a natural way consists in updating θ_t in order to make π_{θ_t} move towards π_{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 π_{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 $\tilde{KL}(\pi_{t+1}, \pi_{\theta_t})$ objective is:

$$\begin{aligned}
\theta_{t+1} &= \theta_t - \nabla_\theta \tilde{KL}(\pi_{t+1}, \pi_{\theta_t}) \\
&= \theta_t + \sum_a \pi_{t+1}(a)\nabla_\theta \log \pi_{\theta_t}(a) \\
&= \theta_t + \sum_a \pi_t(a)\left[1 + \eta_t\left(q(a) - v^{\pi_t}\right)\right]\nabla_\theta \log \pi_{\theta_t}(a).
\end{aligned}$$

Assuming $\pi_t = \pi_{\theta_t}$,

$$\begin{aligned}
\theta_{t+1} &= \theta_t + \sum_a \pi_{\theta_t}(a)\left[1 + \eta_t\left(q(a) - v^{\pi_{\theta_t}}\right)\right]\nabla_\theta \log \pi_{\theta_t}(a) \\
&= \theta_t + \sum_a \left[1 + \eta_t\left(q(a) - v^{\pi_{\theta_t}}\right)\right]\nabla_\theta \pi_{\theta_t}(a) \\
&= \theta_t + \left(1 - \eta_t v^{\pi_{\theta_t}}\right)\sum_a \nabla_\theta \pi_{\theta_t}(a) + \eta_t \sum_a q(a)\nabla_\theta \pi_{\theta_t}(a) \\
&= \theta_t + \eta_t \nabla_\theta \sum_a \pi_{\theta_t}(a)q(a) = \theta_t + \eta_t \nabla_\theta v^{\pi_{\theta_t}},
\end{aligned}$$

which is precisely a policy gradient step.

A.4 Proof of Theorem 2. 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:

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

Note that $(F\nabla y)(a) = \sum_b F_{a,b}\nabla y(b)$

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

from the definition of π. 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_θ are parameterized by some parameter θ (e.g., with a neural network). Let us define the *pseudo-natural gradient* of the probabilities π with respect to θ as the composition of the natural gradient of π with respect to y (i.e., the softmax transformation) and the gradient of y_θ with respect to θ, which is $\widetilde{\nabla}_\theta \pi = (\nabla_\theta y_\theta)(\widetilde{\nabla}_y \pi) = \nabla_\theta y_\theta$. Thus, we have that a natural policy gradient yields:

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

which is nothing else than the NeuRD update rule in (10).

A.5 Equivalence to standard discrete-time RD. A common way to define discrete-time replicator dynamics is according to the so-called *standard discrete-time replicator dynamic* [17], $\pi_t(a) \doteq \pi_{t-1}(a)e^{q^{\pi_{t-1}(a)}}(\boldsymbol{\pi}_{t-1} \cdot e^{q^{\pi_{t-1}}})^{-1}$. 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 (10) with $\eta = 1$:

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

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 (10) 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.

REFERENCES

[1] Maruan Al-Shedivat, Trapit Bansal, Yuri Burda, Ilya Sutskever, Igor Mordatch, and Pieter Abbeel. 2017. Continuous Adaptation via Meta-Learning in Nonstationary and Competitive Environments. *CoRR* abs/1710.03641 (2017). arXiv:1710.03641

[2] Peter Auer, Nicolo Cesa-Bianchi, Yoav Freund, and Robert E Schapire. 2002. The nonstochastic multiarmed bandit problem. *SIAM J. Comput.* 32, 1 (2002), 48–77.

[3] Trapit Bansal, Jakub Pachocki, Szymon Sidor, Ilya Sutskever, and Igor Mordatch. 2017. Emergent Complexity via Multi-Agent Competition. *CoRR* abs/1710.03748 (2017). arXiv:1710.03748

[4] Nolan Bard, Jakob N. Foerster, Sarath Chandar, Neil Burch, Marc Lanctot, H. Francis Song, Emilio Parisotto, Vincent Dumoulin, Subhodeep Moitra, Edward Hughes, Iain Dunning, Shibl Mourad, Hugo Larochelle, Marc G. Bellemare, and Michael Bowling. 2019. The Hanabi Challenge: A New Frontier for AI Research. *CoRR* abs/1902.00506 (2019). arXiv:1902.00506

[5] Amir Beck and Marc Teboulle. 2003. Mirror descent and nonlinear projected subgradient methods for convex optimization. *Operations Research Letters* 31, 3 (2003), 167–175.

[6] Daan Bloembergen, Karl Tuyls, Daniel Hennes, and Michael Kaisers. 2015. Evolutionary Dynamics of Multi-Agent Learning: A Survey. *J. Artif. Intell. Res. (JAIR)* 53 (2015), 659–697.

[7] A. Blum and Y. Mansour. 2007. Learning, Regret Minimization, and Equilibria. In *Algorithmic Game Theory*. Cambridge University Press, Chapter 4.

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

[9] Noam Brown, Christian Kroer, and Tuomas Sandholm. 2017. Dynamic Thresholding and Pruning for Regret Minimization. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*.

[10] Noam Brown, Adam Lerer, Sam Gross, and Tuomas Sandholm. 2018. Deep Counterfactual Regret Minimization. *CoRR* abs/1811.00164 (2018).

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

[12] Noam Brown and Tuomas Sandholm. 2019. Superhuman AI for multiplayer poker. *Science* 365, 6456 (2019), 885–890. https://doi.org/10.1126/science.aay2400

[13] Neil Burch. 2017. *Time and Space: Why Imperfect Information Games are Hard.* Ph.D. Dissertation. University of Alberta.

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

[15] Lucian Busoniu, Robert Babuska, and Bart De Schutter. 2008. A Comprehensive Survey of Multiagent Reinforcement Learning. *IEEE Trans. Systems, Man, and Cybernetics, Part C* 38, 2 (2008), 156–172.

[16] Nicolo Cesa-Bianchi and Gábor Lugosi. 2006. *Prediction, learning, and games.* Cambridge university press.

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

[18] Ryan D'Orazio, Dustin Morrill, James R. Wright, and Michael Bowling. 2020. Alternative Function Approximation Parameterizations for Solving Games: An Analysis of f-Regression Counterfactual Regret Minimization. In *International Conference on Autonomous Agents and Multi-Agent Systems.* arXiv:cs.AI/1912.02967

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

[20] Jakob Foerster, Richard Y Chen, Maruan Al-Shedivat, Shimon Whiteson, Pieter Abbeel, and Igor Mordatch. 2018. Learning with opponent-learning awareness. In *Proceedings of the 17th International Conference on Autonomous Agents and Multi-Agent Systems.* International Foundation for Autonomous Agents and Multiagent Systems, 122–130.

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

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

[23] N. Gatti, F. Panozzo, and M. Restelli. 2013. Efficient Evolutionary Dynamics with Extensive-form Games. In *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence.* 335–341.

[24] Nicola Gatti and Marcello Restelli. 2016. 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).* 509–515.

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

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

[27] Daniel Hennes, Karl Tuyls, and Matthias Rauterberg. 2009. State-coupled Replicator Dynamics. In *Proceedings of the 8th International Conference on Autonomous Agents and Multiagent Systems - Volume 2 (AAMAS '09).* International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 789–796.

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

[29] Josef Hofbauer, Sylvain Sorin, and Yannick Viossat. 2009. Time Average Replicator and Best-Reply Dynamics. *Mathematics of Operations Research* 34, 2 (2009), 263–269. https://doi.org/10.1287/moor.1080.0359

[30] J. J. Hofbauer and K. Sigmund. 1998. Evolutionary games and population dynamics. *Cambridge University Press* (1998).

[31] Max Jaderberg, Wojciech M. Czarnecki, Iain Dunning, Luke Marris, Guy Lever, Antonio Garcia Castañeda, Charles Beattie, Neil C. Rabinowitz, Ari S. Morcos, Avraham Ruderman, Nicolas Sonnerat, Tim Green, Louise Deason, Joel Z. Leibo, David Silver, Demis Hassabis, Koray Kavukcuoglu, and Thore Graepel. 2019. Human-level performance in 3D multiplayer games with population-based reinforcement learning. *Science* 364, 6443 (2019), 859–865. https://doi.org/10.1126/science.aau6249 arXiv:https://science.sciencemag.org/content/364/6443/859.full.pdf

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

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

[34] Robert Kleinberg, Georgios Piliouras, and Eva Tardos. 2009. Multiplicative Updates Outperform Generic No-regret Learning in Congestion Games: Extended Abstract. In *Proceedings of the Forty-first Annual ACM Symposium on Theory of Computing (STOC '09).* ACM, New York, NY, USA, 533–542. https://doi.org/10.1145/1536414.1536487

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

[36] Marc Lanctot. 2014. 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).* 1257–1264.

[37] Marc Lanctot, Edward Lockhart, Jean-Baptiste Lespiau, Vinicius Zambaldi, Satyaki Upadhyay, Julien Pérolat, Sriram Srinivasan, Finbarr Timbers, Karl Tuyls, Shayegan Omidshafiei, Daniel Hennes, Dustin Morrill, Paul Muller, Timo Ewalds, Ryan Faulkner, János Kramár, Bart De Vylder, Brennan Saeta, James Bradbury, David Ding, Sebastian Borgeaud, Matthew Lai, Julian Schrittwieser, Thomas Anthony, Edward Hughes, Ivo Danihelka, and Jonah Ryan-Davis. 2019. OpenSpiel: A Framework for Reinforcement Learning in Games. *CoRR* abs/1908.09453 (2019). arXiv:cs.LG/1908.09453 http://arxiv.org/abs/1908.09453

[38] Marc Lanctot, Vinicius Zambaldi, Audrunas Gruslys, Angeliki Lazaridou, Karl Tuyls, Julien Perolat, David Silver, and Thore Graepel. 2017. A Unified Game-Theoretic Approach to Multiagent Reinforcement Learning. In *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.). 4190–4203.

[39] Hui Li, Kailiang Hu, Shaohua Zhang, Yuan Qi, and Le Song. 2020. Double Neural Counterfactual Regret Minimization. In *Proceedings of the Eighth International Conference on Learning Representations (ICLR).*

[40] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. 2015. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971* (2015).

[41] N. Littlestone and M.K. Warmuth. 1994. The weighted majority algorithm. *Inform. and Comput.* 108 (1994), 212âĂŞ–261.

[42] Edward Lockhart, Marc Lanctot, Julien Pérolat, Jean-Baptiste Lespiau, Dustin Morrill, Finbarr Timbers, and Karl Tuyls. 2019. Computing Approximate Equilibria in Sequential Adversarial Games by Exploitability Descent. *CoRR* abs/1903.05614 (2019).

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

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

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

[46] J. Maynard Smith and G. R. Price. 1973. The Logic of Animal Conflicts. *Nature* 246 (1973), 15–18.

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

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

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

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

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

[52] Angelia Nedic and Soomin Lee. 2014. On stochastic subgradient mirror-descent algorithm with weighted averaging. *SIAM Journal on Optimization* 24, 1 (2014), 84–107.

[53] Shayegan Omidshafiei, Christos Papadimitriou, Georgios Piliouras, Karl Tuyls, Mark Rowland, Jean-Baptiste Lespiau, Wojciech M Czarnecki, Marc Lanctot, Julien Perolat, and Remi Munos. 2019. α-Rank: Multi-Agent Evaluation by Evolution. *arXiv preprint arXiv:1903.01373* (2019).

[54] Liviu Panait and Sean Luke. 2005. Cooperative Multi-Agent Learning: The State of the Art. *Autonomous Agents and Multi-Agent Systems* 11, 3 (2005), 387–434.

[55] Fabio Panozzo, Nicola Gatti, and Marcello Restelli. 2014. Evolutionary Dynamics of Q-Learning over the Sequence Form. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*. 2034–2040.

[56] Julien Perolat, Remi Munos, Jean-Baptiste Lespiau, Shayegan Omidshafiei, Mark Rowland, Pedro Ortega, Neil Burch, Thomas Anthony, David Balduzzi, Bart De Vylder, Georgios Piliouras, Marc Lanctot, and Karl Tuyls. 2020. From Poincaré Recurrence to Convergence in Imperfect Information Games: Finding Equilibrium via Regularization. (2020). arXiv:2002.08456

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

[58] Aldo Rustichini. 1999. Optimal Properties of Stimulus-Response Learning Models. *Games and Economic Behavior* 29 (1999), 244–273. Issue 1.

[59] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. 2015. Trust region policy optimization. In *International Conference on Machine Learning*. 1889–1897.

[60] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal Policy Optimization Algorithms. (2017). arXiv:cs.LG/1707.06347

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

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

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

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

[65] R. Sutton and A. Barto. 2018. *Reinforcement Learning: An Introduction* (2nd ed.). MIT Press.

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

[67] P. Taylor and L. Jonker. 1978. Evolutionarily Stable Strategies and Game Dynamics. *Mathematical Biosciences* 40 (1978), 145–156.

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

[69] Karl Tuyls, Julien Pérolat, Marc Lanctot, Joel Z. Leibo, and Thore Graepel. 2018. A Generalised Method for Empirical Game Theoretic Analysis. In *AAMAS*. International Foundation for Autonomous Agents and Multiagent Systems Richland, SC, USA / ACM, 77–85.

[70] Karl Tuyls, Julien Perolat, Marc Lanctot, Rahul Savani, Joel Leibo, Toby Ord, Thore Graepel, and Shane Legg. 2018. Symmetric Decomposition of Asymmetric Games. *Scientific Reports* 8, 1 (2018), 1015.

[71] Karl Tuyls, Katja Verbeeck, and Tom Lenaerts. 2003. 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*. 693–700.

[72] Karl Tuyls and Gerhard Weiss. 2012. Multiagent Learning: Basics, Challenges, and Prospects. *AI Magazine* 33, 3 (2012), 41–52.

[73] Oriol Vinyals, Igor Babuschkin, Wojciech M. Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H. Choi, Richard Powell, Timo Ewalds, Petko Georgiev, Junhyuk Oh, Dan Horgan, Manuel Kroiss, Ivo Danihelka, Aja Huang, Laurent Sifre, Trevor Cai, John P. Agapiou, Max Jaderberg, Alexander S. Vezhnevets, Rémi Leblond, Tobias Pohlen, Valentin Dalibard, David Budden, Yury Sulsky, James Molloy, Tom L. Paine, Caglar Gulcehre, Ziyu Wang, Tobias Pfaff, Yuhuai Wu, Roman Ring, Dani Yogatama, Dario Wünsch, Katrina McKinney, Oliver Smith, Tom Schaul, Timothy Lillicrap, Koray Kavukcuoglu, Demis Hassabis, Chris Apps, and David Silver. 2019. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature* (2019). https://doi.org/10.1038/s41586-019-1724-z

[74] M.K. Warmuth. 2016. *Evolutionary Games in Natural, Social, and Virtual Worlds*. Chapter 10.

[75] Kevin Waugh, Dustin Morrill, J. Andrew Bagnell, and Michael Bowling. 2015. Solving Games with Functional Regret Estimation. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

[76] Jorgen Weibull. 1997. Evolutionary game theory. *MIT press* (1997).

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

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

[79] M. Zinkevich, M. Johanson, M. Bowling, and C. Piccione. 2008. Regret Minimization in Games with Incomplete Information. In *Advances in Neural Information Processing Systems 20 (NIPS 2007)*.