Towards Cooperative Learning Equilibrium in Reinforcement Learning

Jesse Clifton **Maxime Riché**

Abstract

Suppose that several actors are going to deploy learning agents to act on their behalf. What principles should guide these actors in designing their agents, given that they may have competing goals? We argue that the ideal solution concept in this setting is *welfare-optimal learning equilibrium*. This means that the profile of learning agents should constitute a Nash equilibrium whose payoff profile is optimal according to some measure of total welfare (welfare function). We construct learning algorithms which maximize a welfare function, and punish their counterpart when they detect that they are deviating from this scheme. In the "perfect policy monitoring" case, where agents can verify whether their counterpart is following a cooperative learning algorithm, we show that our construction yields a welfare-optimal learning equilibrium. In the *im*perfect policy monitoring case, agents must instead infer whether their counterpart is following a cooperative learning algorithm. For this purpose, we construct learning algorithms based on a new class of hypothesis tests. We illustrate the behavior of these learning algorithms in two social dilemmas.

1 Introduction

As the capabilities of artificial agents improve, ensuring that these agents interact cooperatively is an increasingly important research direction. In the setting we envision, multiple learning agents are deployed in an unknown environment to act on behalf of their respective principals. These principals will generally have differing goals, and so need to design their learners to avoid conflict while still advancing their own goals as well as possible.

A natural criterion for rational behavior on part of the principals *learning equilibrium (LE)* [Brafman and Tennenholtz, 2003]: are their agents best responses to each other? Beyond that, the learning equilibrium should be in some sense socially optimal. It should be Pareto-optimal, should reflect some combination of total utility gained by all of the agents, the fairness of the distribution of utility, and other normative considerations. Lastly, equilibrium must be enforced with punishment, but we want the punishment of deviations to be forgiving. In particular, punishments should only be severe enough to deter deviations from the equilibrium, be no more. Altogether, this can be thought of as a formal model of principals specifying a contract and a method for enforcing that contract.

Here we work towards these desiderata by sketching a new framework for multi-agent reinforcement learning. To measure the quality of different payoff profiles, we rely on a *welfare function*. The learning equilibrium we construct will converge to a policy profile that maximizes this welfare function. It is beyond the scope of this article to argue for a particular welfare function, as this is ultimately an ethical judgement to be made by the human principals. For simplicity, we will use the utilitarian welfare function (the sum of the agents' individual value functions) in our experiments.

There are several benefits to this approach. First, in this setting, LE is a better account of individual rationality than other solution concepts used in multi-agent learning. The principals will want to choose their learners strategically. This means that it is not enough for a learner to perform well

Working draft. Last updated June 30, 2020.

against a set of non-strategic opponents (e.g. Shoham and Leyton-Brown 2008). It is also not enough that a learner converges to Nash equilibrium *of the game which is being learned* – which we call the "base game" – in self-play (e.g. Bowling and Veloso 2001, Conitzer and Sandholm 2007, Balduzzi et al. 2018, Letcher et al. 2018), as this does not guarantee that a player cannot benefit by deviating from that profile (i.e., submitting a different learning algorithm).

Moreover, our construction draws attention to the advantages of coordination by the principals on the learning algorithms used by the agents they deploy. The equilibrium selection problem can be solved if a welfare function is agreed upon beforehand, even if the environment in which the reinforcement learners will be deployed is initially unknown.

Learning equilibrium was first discussed (under that name) by Brafman and Tennenholtz [2003]. Their setting differs in several ways from ours, however. For stochastic games, their approach involves first finding a cooperative policy profile offline, and then deploying learning algorithms which enforce the usage of the cooperative policies. As we are interested in the setting where the environment is initially unknown (and therefore an initial offline planning step is not available), we instead consider online reinforcement learning in which learning consists in incremental stochastic updates towards the optimizer of a value function. Second, unlike Brafman and Tennenholtz we are motivated by a normative concern with constructing learning equilibria that are optimal with respect to an appropriate welfare function.

Several authors have recently studied cooperation among deep reinforcement learners[1]. Peysakhovich and Lerer [2017], Lerer and Peysakhovich [2017], and Wang et al. [2018] all study the setting in which agents are able to plan offline, and train punishment policies to deter defection. In the online learning setting, Zhang and Lesser [2010], Foerster et al. [2018], and Letcher et al. [2018] have studied "opponent-aware learning", in which players can see the parameters of their counterparts' policies. These authors find that their algorithms lead to higher rates of cooperation in iterated Prisoner's Dilemmas than naive learners. And, Letcher et al. show that their learning algorithm guarantees convergence to a stable fixed point in self play. However, they do not guarantee either that the learning algorithms themselves constitute a learning equilibrium, or that the stable fixed point which is converged to constitutes a reasonable pair of policies (in the sense of scoring high on some measure of total welfare).

In Section 3, we show how to construct learning equilibria by punishing deviations from the optimization of the welfare function in the perfect policy monitoring case. These equilibria involve finite-time punishments. The duration of punishment phases depends on the amount of data it takes to accurately estimate a punishment policy and on the amount of time it takes for the finite-horizon value of the learned punishment policy to approach its infinite-horizon average value. Thus, the theory presented here differs from other theoretical work in similar settings (e.g. Lerer and Peysakhovich 2017, Letcher et al. 2018) by making explicit how the enforcement of cooperation depends on the mixing time of the stochastic game and the accuracy in the estimation of the punishment policy. In Section 4 we present experiments illustrating the construction of strategies in the spirit of learning equilibrium. In so doing, we develop some new methods for making inferences about another player's learning algorithm.

2 Learning equilibrium in stochastic games

Consider a situation in which multiple agents interact with their environment over time, incrementally updating the policies which they use to choose actions. These updates are controlled by their learning algorithms, which are fixed by their principals before the agents begin interacting with the environment. It is helpful to distinguish between two different games that are being played here. There is the **base game**, which is defined by the dynamics of the environment, the reward functions of the agents, and the policies available to each agent. In this paper, the "base game" will be a stochastic game (described below). We can ask whether a particular profile of policies is a Nash equilibrium in the base game. However, as discussed in point 1 in the introduction, this would not guarantee that the *learning algorithms* which converge to that profile of policies are in equilibrium. item Then there is the **learning game**, in which the principals simultaneously submit learning algorithms to be

[1] See also the survey of cooperation in deep reinforcement learning in Hernandez-Leal et al. [2019, Section 3.5].

deployed in the base game environment, and attain payoffs equal to the limit of the average rewards they generate.

We work with stochastic games [Littman, 1994], a generalization of Markov decision processes to the multi-agent setting. We will assume only two players, $i = 1, 2$. We'll refer to player i's counterpart with the index j. In a stochastic game, players simultaneously take actions at each time $t \in \mathbb{N}$; observe a reward depending on the current state; and the state evolves according to a Markovian transition function. Formally, a 2-player stochastic game is a tuple $(\mathcal{S}, \mathcal{A}_1, \mathcal{A}_2, r_1, r_2, P)$, where $\mathcal{S}$ is the state space, with the state at time t denoted S^t; $\mathcal{A}_i$ is the space of actions (assumed to be finite) available to player i, with the action taken by player i at time t denoted as A_i^t; r_i is the reward function of player i, with the reward gained by player i at time t given by $R_i^t = r_i(S^t, A_i^t, A_j^t)$; and P is a Markovian transition function, such that the conditional distribution over states at time $t+1$ is given by $P(S^{t+1} \in \mathcal{B} \mid \{S^v, A_1^v, A_2^v\}_{v=0}^t) = P(S^{t+1} \in \mathcal{B} \mid S^t, A_1^t, A_2^t)$ for measurable $\mathcal{B} \subseteq \mathcal{S}$.

Define a (stationary) *policy* π_i to be a mapping from states to random variables taking values in $\mathcal{A}_i$, such that $P\{\pi_i(s) = a\}$ is the probability that player i takes action a in state s when following policy π_i. Players wish to learn a policy which (in some sense) leads to large cumulative reward. In the online setting, each player is learning over a space of policies $\Pi_\Theta = \{\pi_\theta : \theta \in \Theta\}$ for some parameter space Θ. Correspondingly, define for parameter profiles $\theta = (\theta_1, \theta_2)$ (and arbitrary initial state s^0) the policy-value functions $V_i(\theta) = \lim_{T\to\infty} T^{-1}\mathbb{E}_{\pi_{\theta_1},\pi_{\theta_2}}\left(\sum_{t=1}^T R_i^t \mid S^0 = s^0\right)$, where $\mathbb{E}_{\pi_{\theta_1},\pi_{\theta_2}}$ denotes the expectation with respect to trajectories in which players follow policies $\pi_{\theta_1}, \pi_{\theta_2}$ at each time step. That is, $V_i(\theta)$ is player i's expected average return when player i follows π_{θ_i} and player j follows π_{θ_j}. This limit is guaranteed to exist by the ergodicity assumption which is made in our construction of learning equilibria (Theorem 3.1 below).

Let H^t be the history of observations until time t. We will assume that both agents fully observe the state and each others' actions and rewards, such that $\{(S^v, A_1^v, A_2^v, R_1^v, R_2^v)\}_{v=0}^t \in H^t$. Then we define a learning algorithm.

Definition 2.1 (Learning algorithm)**.** A learning algorithm for player i is a mapping σ_i from histories H^t to parameters, i.e., $\sigma_i\left(H^t\right) \in \Theta$.

Define the value to player i of learning algorithm profile $\sigma = (\sigma_1, \sigma_2)$ starting from history H^t as $V_i(H^t, \sigma) = \liminf_{T\to\infty} T^{-1}\mathbb{E}_\sigma\left(\sum_{v=t}^{t+T} R_i^v \mid H^t\right)$, where $\mathbb{E}_\sigma$ is the expectation taken with respect to trajectories in which agents follow learning algorithms σ_1, σ_2 at each step. Let the initial history be $H^0 = \{(S^0, \theta_1^0, \theta_2^0)\}$. Then, the game described above may be written in normal form as a game with strategy spaces Σ_1, Σ_2 corresponding to spaces of possible learning algorithms, and player i's utilities at each profile (σ_1, σ_2) given by $V_i(H^0, \sigma_1, \sigma_2)$. This a *learning game*. A learning equilibrium is a Nash equilibrium of a learning game:

Definition 2.2 (Learning equilibrium)**.** Learning algorithm profile (σ_1, σ_2) is an learning equilibrium of the learning game with learning algorithm spaces Σ_1, Σ_2 if

$$\sup_{\sigma_1' \in \Sigma_1} V_1(H^0, \sigma_1', \sigma_2) \leq V_1(H^0, \sigma_1, \sigma_2) \text{ and } \sup_{\sigma_2' \in \Sigma_2} V_2(H^0, \sigma_1, \sigma_2') \leq V_2(H^0, \sigma_1, \sigma).$$

Lastly, we classify histories based on whether agents can verify whether their counterpart's actions are consistent with a particular learning algorithm.

Definition 2.3 (Policy monitoring)**.** Let $\chi_i^v = \mathbb{1}\left[A_i^v = \pi_{\sigma_i(H^v)}\right]$. The set of histories $\mathcal{H}$ satisfies *perfect policy monitoring* for algorithms σ_1, σ_2 if for each $H = \{H^0, H^1, \dots\} \in \mathcal{H}$ each t, $\{\chi_1^v, \chi_2^v\}_{v=1}^t \in H^t$. $\mathcal{H}$ exhibits *imperfect policy monitoring* if this is not satisfied for some H^t.

Opponent-aware algorithms [Zhang and Lesser, 2010, Foerster et al., 2018, Letcher et al., 2018] are an example of learning algorithms which make use of perfect policy monitoring.

3 Construction of forgiving and welfare-optimal learning equilibria

The learning algorithms we construct will converge to the optimizer of a welfare function in self-play. The welfare function is intended to encode a compromise between the principals' individual payoffs. One intuitive welfare function is the utilitarian welfare (sometimes just

called the "social welfare"). For player utility functions u_i and outcomes x, this function calculates welfare according to $w^{\text{Util}}(x) = u_1(x) + u_2(x)$. Another well-known welfare function from the cooperative bargaining literature welfare function is the Nash welfare [Nash, 1950], $w^{\text{Nash}}\{u_1(x), u_2(x)\} = \{u_1(x) - u_1^d\} \times \{u_2(x) - u_2^d\}$. (This welfare function depends on "disagreement" utilities u_i^d, which represent the value a player might get if the parties fail to compromise.)

The welfare function w will act on policy profiles θ, i.e., write $w(\theta) = w\{V_1(\theta), V_2(\theta)\}$. Write the set of (locally) welfare-optimal policy profiles as $\Theta^{2,C}$. As discussed below, we expect to be able to construct algorithms which converge to a policy profile in $\Theta^{2,C}$ using standard reinforcement learning methods[2]. However, in order to construct a learning equilibrium, we also need to be able to disincentivize defections from an algorithm which optimizes w. For this reason, we introduce a *punishment point* V_j^p for player i. To construct an equilibrium, we will need to assume that a player's punishment payoffs are worse than their payoffs under the welfare-optimal policy. That is, for each i there exists a punishment policy parameter θ_i^p such that $\max_{\theta_j} V_j(\theta_i^p, \theta_j) = V_j^p < \inf_{\theta \in \Theta^{2,C}} V_j(\theta)$.

Now we show how to construct welfare-optimal learning equilibria in the case of perfect policy monitoring (Definition 2.3). This is accomplished by optimizing a welfare function and punishing deviations from the optimization of this welfare function Let $\{\delta^t\}_{t=1}^{\infty}$ be a sequence of step sizes for the policy parameter updates. Let $\widehat{w}$ be an estimator of the welfare function. Let $\mathcal{O}^C$ be a base policy learning algorithm which maps histories H^t to candidate updates (θ_1^t, θ_2^t). For instance, $\mathcal{O}^C$ might be a policy gradient algorithm $\mathcal{O}^C(H^t) = \theta^t + \delta^t \nabla_\theta w^t(\theta^t)$, where $\nabla_\theta w^t$ refers to a policy gradient step on the welfare function.

Define $\theta^C = \arg\max_\theta w(\theta)$ and $\theta_j^p = \arg\max_{\theta_j} V_2(\theta_1^p, \theta_j)$. Write the updated parameters returned by $\mathcal{O}^C$ as $\theta^{C,t+1} = \mathcal{O}^C(H^t)$. We say that player i defects at time t if $\theta_i^t \neq \theta_i^{C,t}$, and construct a strategy which punishes defections. The learning equilibrium construction presented here divides time points into blocks, and deploys either a cooperative or punishment learning algorithm in each block depending on whether there was a defection in the previous block. (This block construction is similar to that used in Wiseman [2012]'s construction of equilibria for repeated games in which players only see noisy observations of the underlying stage game.) Note that, while equilibrium could be accomplished with a "grim-trigger" strategy which never stops punishing after a defection, we deliberately use limited-time punishments to make our construction more forgiving. Limited-time punishments require additional care, as we have to account for estimation error in the cooperative and punishment policies in ensuring that they yield payoffs sufficiently close to their target values in any particular block. Specifically, in the b^{th} block of length $M^b + N^b$:

1. If the other player cooperated at each time point in the previous block, then
   - Deploy cooperative policy learner $\mathcal{O}^C$ for M^b steps to obtain a good estimate $\theta_i^{b,C}$ of the welfare-optimal policy;
   - Deploy policy $\theta_i^{b,C}$ for N^b time steps to generate payoffs close to V_i^C.
2. If the other player defected at any time point in the previous block, then
   - Deploy punishment policy learner $\mathcal{O}^p$ for M^b steps to obtain a good estimate $\theta_i^{b,p}$ of the punishment policy;
   - Deploy policy $\theta_i^{b,p}$ for N^b time steps to generate payoffs close to V_i^p.

This algorithm is summarized in algorithmic form in Appendix A.

Theorem 3.1 (Construction of forgiving and welfare-optimal learning equilibrium)**.** Consider a learning game with perfect policy monitoring. Fix a welfare function w, cooperative policy optimizer $\mathcal{O}^C$, and punishment policy optimizer $\mathcal{O}^p$. Consider the corresponding learning algorithms $(\sigma_1^{\text{LE}}, \sigma_2^{\text{LE}})$ constructed as in Algorithm 3.

For $i = 1, 2$, let $\theta_i^C \in \Theta^{2,C}$ index a (locally) welfare-optimal policy and let θ_i^p be a punishment policy parameter satisfying $\max_{\theta_j} V_j(\theta_i^p, \theta_j) = V_j^p < \inf_{\theta \in \Theta^{2,C}} V_j(\theta)$. Let $t^b = \sum_{b' \leq b}(M^{b'} + N^{b'})$, i.e., t^b is the time at which the b^{th} block starts. Write $V^{t:t'}(H^t, \theta) = (t' - t)^{-1}\mathbb{E}_\theta\left(\sum_{v=t}^{t'} R^v \mid H^t\right)$.

[2]Note that optimizing a welfare function still does not fully solve the equilibrium selection problem, as there may be several welfare-optimal policy profiles. In our experiments (Section 4) we use entropy regularization to select a unique welfare-optimal policy among several policies which give equivalent welfare.

Use a similar notation for the average value attained in over time points $t, \ldots, t'$ under combinations of stationary policies θ_i and learning algorithms $\widetilde{\sigma}_i$. To make notation more compact, define the following quantities:

$$\widehat{V}_j^{b,C} = V^{t^b+M^b:t^b+M^b+N^b}(H^{t^b}, \theta_i^{b,C}, \theta_j^{b,C}) \qquad V_j^{b,C} = V^{t^b+M^b:t^b+M^b+N^b}(H^{t^b}, \theta_i^{C}, \theta_j^{C})$$
$$\widehat{V}_j^{b,p} = V^{t^b+M^b:t^b+M^b+N^b}(H^{t^b}, \theta_i^{b,p}, \widetilde{\sigma}_j) \qquad V_j^{b,p} = V^{t^b+M^b:t^b+M^b+N^b}(H^{t^b}, \theta_i^{p}, \theta_j^{p}).$$

Further assume that

1. The value of the estimated cooperative (punishment) policy in a block $\widehat{V}_j^{b,C}$ ($\widehat{V}_j^{b,p}$) converges in probability to the true value in that block $V_j^{b,C}$ ($V_j^{b,p}$) at a polynomial rate;
2. The stochastic game is uniformly ergodic [Ortner, 2018];
3. Player j's reward function is bounded by $\Delta_j < \infty$;
4. Almost surely, for each t the distribution over states at time t is dominated by a finite measure. Similarly, the stationary distribution over states under each policy parameter θ, ν_θ, is dominated by a finite measure.
5. $\sum_{b \leq B} M^b \left\{ \sum_{b \leq B} \left(M^b + N^b \right) \right\}^{-1} \longrightarrow 0.$
6. There exist $K^p < \infty, \gamma^p \in (0,1)$ such that $\sup_{\widetilde{\sigma}_j} \limsup V_j(\theta_i^p, \widetilde{\sigma}_j) - V_j^p \leq K^p(\gamma^p)^b$;

Under Assumptions 1-6, $(\sigma_1^{\mathrm{LE}}, \sigma_2^{\mathrm{LE}})$ is a learning equilibrium and $w(H^0, \sigma_1^{\mathrm{LE}}, \sigma_2^{\mathrm{LE}})$ converges to a locally welfare-optimal policy profile.

Proof. In Appendix A. □

It is beyond the scope of this article to analyze the conditions under which the assumptions of Theorem 3.1 hold. But, we note that the requirements of Assumption 1 are standard desiderata for learning algorithms in single-agent reinforcement learning and zero-sum games. The latter can be used to construct a punishment policy by setting the punishment policy equal to the one which minimizes the other player's best-case payoff. For instance, Yang et al. [2019] give rates for the convergence of deep Q-network (DQN) Mnih et al. 2013 estimators. This result extends to and minimax DQN [Littman, 1994], which can be used to construct a punishment policy.

Uniform ergodicity is a standard assumption in the theoretical analysis of reinforcement learning; see e.g. Ortner [2018] and references therein. Lastly, it suffices for Assumption 6 to hold that V_j^p is player j's minimax value and that there exists a stationary minimax policy for player j and that this can be learned at the required rate. See González-Trejo et al. [2002] on the existence of stationary minimax policies in stochastic games.

4 Illustrative experiments

We now describe simulations illustrating the construction of learning algorithms in the spirit of σ^{LE}. We call our family of algorithms L-TFT for "learning tit-for-tat", to emphasize that they involve limited-time punishment learning algorithms in the spirit of tit-for-tat. We consider the case of imperfect policy monitoring (IPM; Definition 2.3), where the agents must *infer* whether their counterpart is cooperating. Throughout, the class of base cooperative learning algorithms will be policy gradient algorithms of the form $\theta_i^{C,l+1} = \theta_i^{C,l} + \delta^t \nabla_{\theta_i} w^l$. Here, l indexes blocks of episodes and $\nabla_{\theta_i} w^l$ is a policy gradient step on the welfare function. In our experiments this corresponds to either REINFORCE [Williams, 1992] or proximal policy optimization (PPO; Schulman et al. 2017). In Sections 4.1 and 4.2 we assume that the sequence δ^t is known, and then in Section 4.3 allow it to be unknown. In all cases, the counterpart's initial parameters are unknown, so the other player's policy parameters under the hypothesis that they are cooperating are never known exactly.

L-TFT maintains a cooperative network trained to optimize welfare function, which in our experiments will mean using $R_1^t + R_2^t$ as the reward signal. It also maintains a punishment network trained on

$-R_j^t$. Finally, due to imperfect policy monitoring, L-TFT maintains estimates of the sequence of estimates of the other agent's policy under the hypothesis that they are following a cooperative algorithm, denoted $\{\widehat{\theta}_j^{C,l}\}_{l=L-L'}^{L}$ and an estimate under the hypothesis that they are not following a cooperative algorithm, denoted $\{\widehat{\theta}_j^{D,l}\}_{l=L-L'}^{L}$ (for some memory L').

In each case, the sequence of policy networks $\{\widehat{\theta}_j^{D,l}\}_{l=L-L'}^{L}$ corresponding to the hypothesis of defection is estimated using supervised learning on agent j's actions. That is, the sequence representing the hypothesis of defection is represented by the sequence of policies which best fit the other player's actions, without any additional constraints. The estimation of the cooperative networks $\{\widehat{\theta}_j^{C,l}\}_{l=L-L'}^{L}$ depends on what is assumed about the other player's learning algorithm; in our experiments, this depends on whether the learning rate is assumed to be known.

Defections are detected via a hypothesis test, where the null hypothesis is that the other player is following a cooperative learning algorithm. This hypothesis test is conducted by bootstrapping [Efron, 1992] the log likelihood ratio of the other player's actions under $\widehat{\theta}_j^{C,t}$ and $\widehat{\theta}_j^{D,t}$. See Appendix B for details.

The environments we use are the iterated Prisoner's Dilemma (IPD) and the Coin Game environment [Lerer and Peysakhovich, 2017, Foerster et al., 2018][3].

4.1 Known learning rate: Iterated Prisoner's Dilemma

Our version of the IPD involves repeated play of the matrix game with expected payoffs as in Table 1. Policy π_{θ_i} gives the probability of each action given player j's action at the previous timestep. In particular, $\theta_i = (\theta_{i,CC}, \theta_{i,CD}, \theta_{i,DC}, \theta_{i,DD})$, and $P\{\pi_{\theta_i}(A_j^{t-1}) = a\} \propto \exp(\theta_{i,A_j^{t-1}a})$.

		Player 2 C	Player 2 D
Player 1	C	$-1, -1$	$-3, 0$
	D	$0, -3$	$-2, -2$

Table 1: Prisoner's Dilemma expected payoffs.

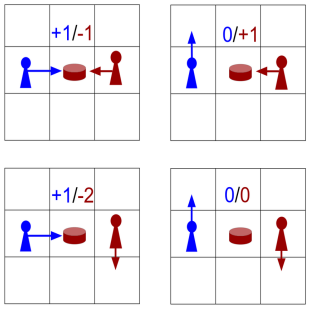

Figure 1: Coin Game payoffs.

Write the (expected) reward function, corresponding to the payoffs in Table 1, as $r_i(a_i, a_j)$. We introduce randomness by drawing rewards $R_i^t \sim N\left\{r_i(A_i^t, A_j^t), 0.1\right\}$.

For the base cooperative policy optimization algorithm we use REINFORCE Williams [1992]. When player i detects a defection, they follow the punishment policy for one episode and then return to the cooperative policy.

Figure 2 shows the results of L-TFT in self-play and against a naive learner. This learning algorithm profile is still able to converge to welfare-optimal payoffs in self-play and punish defections by a naive learner, resulting in mutual defection in the stage game. This indicates that our hypothesis test successfully distinguishes a cooperative from a naive counterpart.

4.2 Known learning rate: Coin Game

In the Coin Game environment [Lerer and Peysakhovich, 2017, Foerster et al., 2018], depicted in Figure 1, a Red and Blue player navigate a grid and pick up randomly-generated coins. Each player

[3]These experiments were implemented using extensions of OpenAI Gym [Brockman et al., 2016] and the SLM Lab framework for reinforcement learning [Keng and Graesser, 2017]. The code for reproducing the experiments can be found here: `https://github.com/Manuscrit/SLM-Lab`. This depends on a fork of the Gym repository here: `https://github.com/Manuscrit/gym`.

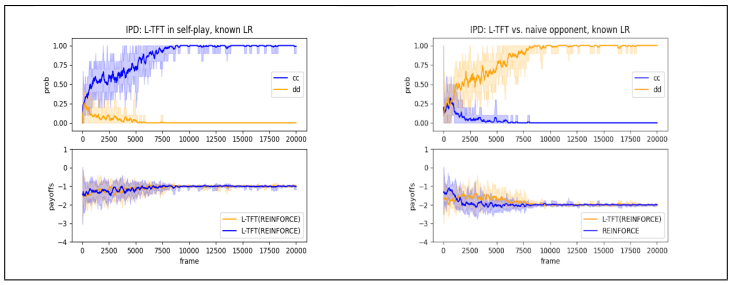

Figure 2: Performance of L-TFT against itself (left) and against a naive REINFORCE learner (right) in the iterated Prisoner's Dilemma. The `prob` axis is the probability of mutual cooperation (`cc`) and mutual defection (`dd`), and the `payoffs` axis is the reward for each player at the corresponding time steps. These values are averaged over 10 replicates and bands are (min, max) values.

gets a reward of 1 for picking up a coin of any color. But, a player gets a reward of -2 if the other player picks up their coin. This creates a social dilemma in which the socially optimal behavior is to only get one's own coin, but there is incentive to defect and try to get the other player's coin as well.

The state space consists of encodings of each player's location and the location of the coin. We use networks with two convolutional layers and two fully connected feedforward networks. L-TFT works as before, except that networks are trained with proximal policy optimization (PPO) [Schulman et al., 2017]. L-TFT again converges to mutual cooperation in self-play (picking own coin 100% of the time). This is significantly better than the performance of naive PPO in self-play (`Coin Game: baseline`), which does not converge to mutual cooperation. And against a naive PPO learner, L-TFT successfully punishes the naive learner, with the naive learner's converging to lower values than are achieved in self-play. However, this comes at the expense of very low payoffs for L-TFT.

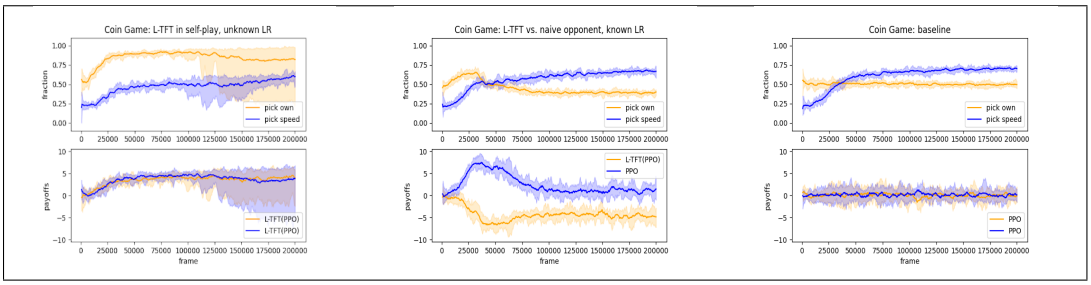

Figure 3: Coin Game performance of L-TFT self-play, versus a naive learner, and naive learner in self-play. Payoffs are moving averages over the past 100 time steps. "Pick speed" refers to what fraction of steps into an episode an agent picks a coin. "Pick own" refers to what fraction of episodes an agent picks the coin of their own color. These values are averaged over 10 replicates and bands are (min, max) values.

4.3 Unknown learning rate: Iterated Prisoner's Dilemma

In the next experiments we relax the assumption on what is known about the other player's learning algorithm. Under the hypothesis that the other player is cooperating, players assume that their counterpart has been following a sequence of policies $\theta_j^{C,2}, \theta_j^{C,2}, \ldots$ updated according to REINFORCE with an unknown learning rate. These policies are updated after blocks of size U. Index the u^{th} observation in the l^{th} policy block by lu. To conduct the hypothesis test, we thus solve

$$\widehat{\theta}_j^{C,1}, \widehat{\theta}_j^{C,2}, \ldots, \widehat{\theta}_j^{C,L}, \widehat{\delta} = \arg\max_{\theta_j, \delta} \sum_{l=1}^{L-1} \sum_{u=1}^{U} \ell(A^{lu}, S^{lu}, \theta_j^l) \\ \text{s.t. } \theta_j^l + \delta \nabla_{\theta_j} w^l(\widehat{\theta}_i^l, \theta_j^l) = \theta_j^{l+1}, l = 1, \ldots, L-1. \tag{1}$$

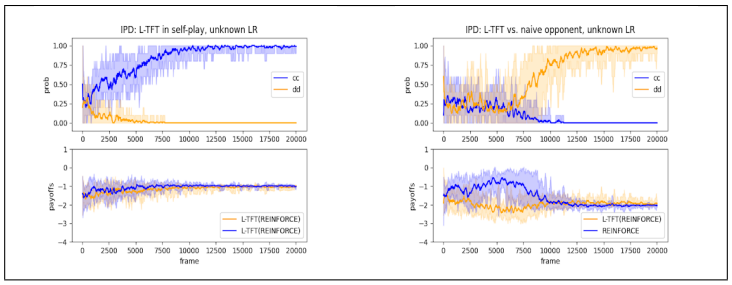

Figure 4: Performance of L-TFT in the iterated Prisoner's Dilemma, in the case where the other player's learning rate is unknown.

Here, $\ell(A^{lu}, S^{lu}, \theta_j^l)$ is the supervised loss of the predictions under policy θ_j^l at observations (A^{lu}, S^{lu}); $\nabla_{\theta_j} w^l(\widehat{\theta}_i^l, \theta_j^{C,l})$ refers to a cooperative policy gradient step, and $\widehat{\theta}_i^l$ is the unconstrained estimate of player i's policy in policy block l. Thus this a surrogate for the maximum likelihood sequence of policies θ_j^l under the assumption that updates $\delta \nabla_{\theta_j} w^l$ are taken at each policy block for some δ. We then proceed as before, using $\theta_j^{C,l}$ as the estimated cooperative policies and the (penalized) maximum likelihood policy sequence when conducting our hypothesis tests. The results are presented in Figure 4.

Remark 4.0.1. Constructing algorithms which gave the desired results required a fair amount of hyperparameter tuning. In particular, in cases where the networks used to estimate the counterpart's policy gave low-entropy policies, the hypothesis test would be destabilized. For this reason we ensured that the policies were high-entropy in the initial steps of the experiments. Moreover, in the Coin Game, there might be several policies which are welfare-optimal (e.g., several equivalent paths to the coin). In the case that the actual counterpart policy $\theta_j^{C,l}$ and the and estimated counterpart policy $\widehat{\theta}_j^{C,l}$ use different welfare-optimal paths, the counterpart would be treated as defecting. To avoid this problem, we used entropy regularization throughout to prevent opponent policies from taking on extreme probabilities. This can be loosely interpreted as choosing the maximum-entropy welfare-optimal policy.

5 Discussion

It is of paramount importance that powerful machine intelligences avoid conflict when they are eventually deployed. One way to avoid such outcomes is to construct profiles of agents which avoid conflict and which rational principals prefer to deploy. We have argued that the appropriate notion of rationality in this context is learning equilibrium (rather than mere convergence to Nash equilibrium of the base game, for instance). We have taken initial steps towards showing how learning equilibria which are cooperative — in the sense of optimizing a reasonable welfare function — can be constructed.

Many open questions remain. First, while we have relied on a welfare function, we have said little about which welfare function should be used. The ideal scenario would be for the principals of the reinforcement learning systems in question to coordinate on a welfare function before deploying these systems. Another direction worth exploring is the design of learning algorithms which are themselves capable of "bargaining" to agree on a welfare function.

Cases such as partial state observability, more than two agents, and criteria other than average-reward should be studied in theory and experiment. We also have assumed that the agents' reward functions are known, but a complete framework should address the problem of incentives to misrepresent one's reward function to improve one's payoffs in the welfare-optimal policy profile. One direction would be to allow for uncertainty about the other players' reward functions, analogously to Bayesian games [Harsanyi, 1967] but in a way that is tractable in complex environments. Another direction would be to use mechanism design techniques Nisan et al. [2007, Ch. 9] to incentivize the truthful reporting of the principals' utility functions.

In terms of implementation, it may be necessary to develop novel reinforcement learning methods tailored for the optimization of different welfare functions. For instance, we have focused on policy-based reinforcement learning, but the development of value-based methods would likely broaden the space of practical algorithms available for implementing our approach. This is nontrivial, as the welfare functions other than the utiliarian welfare (such as the Nash welfare [Nash, 1950]) do not admit a Bellman equation (see the discussion of nonlinear scalarization functions in Roijers et al. [2013]'s review of multi-objective reinforcement learning).

Finally, perhaps the most important shortcoming for the applicability of our framework to real-world agents is the unrealistic assumption about ergodicity used to construct equilibria. In future work, we plan to develop model-based learning equilibrium concepts which apply to arbitrary environments, ergodic or otherwise.

Acknowledgements

Many thanks to Tobias Baumann, Flo Dorner, Jakob Foerster, Daniel Kokotajlo, and Johannes Treutlein for helpful comments on drafts of this document.

References

David Balduzzi, Sebastien Racaniere, James Martens, Jakob Foerster, Karl Tuyls, and Thore Graepel. The mechanics of n-player differentiable games. *arXiv preprint arXiv:1802.05642*, 2018.

Michael Bowling and Manuela Veloso. Rational and convergent learning in stochastic games. In *International joint conference on artificial intelligence*, volume 17, pages 1021–1026. Lawrence Erlbaum Associates Ltd, 2001.

Ronen I Brafman and Moshe Tennenholtz. Efficient learning equilibrium. In *Advances in Neural Information Processing Systems*, pages 1635–1642, 2003.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016.

Vincent Conitzer and Tuomas Sandholm. Awesome: A general multiagent learning algorithm that converges in self-play and learns a best response against stationary opponents. *Machine Learning*, 67(1-2):23–43, 2007.

Bradley Efron. Bootstrap methods: another look at the jackknife. In *Breakthroughs in statistics*, pages 569–593. Springer, 1992.

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

JI González-Trejo, Onésimo Hernández-Lerma, and Luis F Hoyos-Reyes. Minimax control of discrete-time stochastic systems. *SIAM Journal on Control and Optimization*, 41(5):1626–1659, 2002.

John C Harsanyi. Games with incomplete information played by "bayesian" players, i–iii part i. the basic model. *Management science*, 14(3):159–182, 1967.

Pablo Hernandez-Leal, Bilal Kartal, and Matthew E Taylor. A survey and critique of multiagent deep reinforcement learning. *Autonomous Agents and Multi-Agent Systems*, 33(6):750–797, 2019.

Wah Loon Keng and Laura Graesser. Slm lab. `https://github.com/kengz/SLM-Lab`, 2017.

Adam Lerer and Alexander Peysakhovich. Maintaining cooperation in complex social dilemmas using deep reinforcement learning. *arXiv preprint arXiv:1707.01068*, 2017.

Alistair Letcher, Jakob Foerster, David Balduzzi, Tim Rocktäschel, and Shimon Whiteson. Stable opponent shaping in differentiable games. *arXiv preprint arXiv:1811.08469*, 2018.

Michael L Littman. Markov games as a framework for multi-agent reinforcement learning. In *Machine learning proceedings 1994*, pages 157–163. Elsevier, 1994.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

John F Nash. The bargaining problem. *Econometrica: Journal of the Econometric Society*, pages 155–162, 1950.

Noam Nisan et al. Introduction to mechanism design (for computer scientists). *Algorithmic game theory*, 9:209–242, 2007.

Ronald Ortner. Regret bounds for reinforcement learning via markov chain concentration. *arXiv preprint arXiv:1808.01813*, 2018.

Alexander Peysakhovich and Adam Lerer. Consequentialist conditional cooperation in social dilemmas with imperfect information. *arXiv preprint arXiv:1710.06975*, 2017.

Diederik M Roijers, Peter Vamplew, Shimon Whiteson, and Richard Dazeley. A survey of multi-objective sequential decision-making. *Journal of Artificial Intelligence Research*, 48:67–113, 2013.

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

Yoav Shoham and Kevin Leyton-Brown. *Multiagent systems: Algorithmic, game-theoretic, and logical foundations*. Cambridge University Press, 2008.

Weixun Wang, Jianye Hao, Yixi Wang, and Matthew Taylor. Towards cooperation in sequential prisoner's dilemmas: a deep multiagent reinforcement learning approach. *arXiv preprint arXiv:1803.00162*, 2018.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

Thomas Wiseman. A partial folk theorem for games with private learning. *Theoretical Economics*, 7 (2):217–239, 2012.

Zhuora Yang, Yuchen Xie, and Zhaoran Wang. A theoretical analysis of deep q-learning. *arXiv preprint arXiv:1901.00137*, 2019.

Chongjie Zhang and Victor Lesser. Multi-agent learning with policy prediction. In *Twenty-Fourth AAAI Conference on Artificial Intelligence*, 2010.

Appendix A: Proof of Theorem 3.1

Proof. For brevity, write $V_j(\theta^C) = V_j^C$ and $V_j(\theta^p) = V_j^p$. Let χ^b indicate whether player j cooperated in block b (setting $\chi^{-1} = 1$).

Notice that, under Assumption 2, we can bound the difference between V_j^C and the finite-time average reward under the cooperative policy in block b as follows:

$$V_j^{t^b:t^b+M^b+N^b}(H^{t^b}, \sigma_i^C, \sigma_j^C) - V_j^C \leq (M^b + N^b)^{-1} \left\{ M^b \Delta_j + N^b \left(\widehat{V}_j^{b,C} - V_j^C \right) \right\}.$$

Let μ be the measure on $\mathcal{S}$ given by Assumption 4. By Assumption 3, take K and γ such that

$$\sup_\theta \| P_\theta^t \nu - \nu_\theta \| \leq K \gamma^t.$$

Let $P^t_{\theta^{b,C}}\nu(S^{t^b+M^b})$ refer to the distribution of states t steps after the state $S^{t^b+M^b}$ is encountered, under the policy profile indexed by $\theta^{b,C}$. By Assumptions 1, 2, and 4, there exist constants K^C, r_b^C, and $r_\epsilon^C < r_\delta^C$ such that

$$\begin{aligned}
\widehat{V}_j^{b,C} - V_j^C &= \left(\widehat{V}_j^{b,C} - V_j^{b,C}\right) + \left(V_j^{b,C} - V_j^C\right) \\
&\leq (t^b)^{-r_\epsilon^C} P\left\{\widehat{V}_j^{b,C} - V_j^{b,C} < (t^b)^{-r_\epsilon^C}\right\} \\
&\quad + \Delta_j P\left\{\widehat{V}_j^{b,C} - V_j^{b,C} > (t^b)^{-r_\epsilon^C}\right\} \\
&\quad + (N^b)^{-1} \sum_{t=t^b+M^b}^{t^b+M^b+N^b} \int r_j\left\{S^t, \pi_{\theta^{b,C}}(S^t)\right\}\left\{\mathrm{d}P^t_{\theta^{b,C}}\nu(S^{t^b+M^b})(S^t) - \mathrm{d}\nu_{\theta^C}(S^t)\right\} \\
&\leq (t^b)^{-r_\epsilon^C}\left\{1 - (t^b)^{-r_b^C(r_\delta^C - r_\epsilon^C)}K^C\right\} \\
&\quad + \Delta_j (t^b)^{-r_b^C(r_\delta^C - r_\epsilon^C)}K^C \\
&\quad + K\Delta_j\mu(\mathcal{S})\left\{N^b(1-\gamma)\right\}^{-1}\left(1 - \gamma^{-N^b}\right)
\end{aligned}$$

The same holds for the corresponding punishment point quantities with some constants K^p, r_b^p, and $r_\epsilon^p < r_\delta^p$ such that making the necessary changes. Moreover, by Assumption 5,

$$\begin{aligned}
(t^B)^{-1}\sum_{b\leq B}\left(1 - \gamma^{-N^b}\right) &\leq (t^B)^{-1}B \\
&\longrightarrow 0.
\end{aligned}$$

We will now write the value attained by player 2 in the first B blocks in a way that allows us to apply these convergence statements. Then we will take expectations and apply Assumption 2 to show that each term goes to 0. Letting Ω be a set of measure 1 and $\chi^b(\omega)$ the value of χ^b along an arbitrary sample path $\omega \in \Omega$, write $\overline{B}^D = \sup_{\omega\in\Omega}\limsup_{B\to\infty} B^{-1}\sum_{b\leq B}\left\{1 - \chi^b(\omega)\right\}$. And, write

$$\begin{aligned}
\mathbb{1}_<^{b,C} &= \mathbb{1}\left\{\widehat{V}_j^{b,C} < V_j^{b,C} + (t^b)^{-r_\epsilon^C}\right\}; \\
\mathbb{1}_<^{b,p} &= \mathbb{1}\left\{\widehat{V}_j^{b,p} < V_j^{b,p} + (t^b)^{-r_\epsilon^p}\right\}.
\end{aligned}$$

Then,

$$
\begin{aligned}
V_j^{t^B} - V_j^C &= (t^B)^{-1} \sum_{b=0}^{B-1} (t^{b+1} - t^b) \mathbb{E}(\widehat{V}_j^b - V_j^C) \\
&\leq (t^B)^{-1} \sum_{b=0}^{B-1} \left[M^b \Delta_j + N^b \mathbb{E} \left\{ \chi^{b-1} \left(\widehat{V}_j^{b,C} - V_j^C \right) + (1 - \chi^{b-1}) \left(\widehat{V}_j^{b,p} - V_j^C \right) \right\} \right] \\
&= (t^B)^{-1} \sum_{b=0}^{B-1} \Big[M^b \Delta_j + N^b \mathbb{E} \Big\{ \chi^{b-1} \left(\widehat{V}_j^{b,C} - V_j^C \right) \\
&\qquad + (1 - \chi^{b-1}) \left(\widehat{V}_j^{b,p} - V_j^p + V_j^p - V_j^C \right) \Big\} \Big] \\
&= (t^B)^{-1} \sum_{b=0}^{B-1} \Big[M^b \Delta_j + N^b \mathbb{E} \Big\{ \chi^{b-1} \left(\widehat{V}_j^{b,C} - V_j^C \right) \\
&\qquad + (1 - \chi^{b-1}) \left(\widehat{V}_j^{b,p} - V_j^p \right) \Big\} \Big] + \overline{B}^D \left(V_j^p - V_j^C \right) \\
&= (t^B)^{-1} \sum_{b=0}^{B-1} M^b \Delta_j \\
&\quad + (t^B)^{-1} \sum_{b=0}^{B-1} N^b \mathbb{E} \Big\{ \chi^{b-1} \left(\widehat{V}_j^{b,C} - V_j^C \right) + (1 - \chi^{b-1}) \left(\widehat{V}_j^{b,p} - V_j^p \right) \Big\} \Big] \\
&\quad + \overline{B}^D \left(V_j^p - V_j^C \right) \\
&\leq (t^B)^{-1} \sum_{b=0}^{B-1} M^b \Delta_j \\
&\quad + (t^B)^{-1} \sum_{b=0}^{B-1} N^b \mathbb{E} \Big[\left\{ \left((t^b)^{-r_\epsilon^C} + \rho(N^b) \right) \mathbb{1}_<^{b,C} + \Delta_j \left(1 - \mathbb{1}_<^{b,C} \right) \right\} \\
&\qquad + \left\{ \left((t^b)^{-r_\epsilon^p} + \rho(N^b) \right) \mathbb{1}_<^{b,p} + \Delta_j \left(1 - \mathbb{1}_<^{b,p} \right) \right\} \Big] \\
&\quad + \overline{B}^D \left(V_j^p - V_j^C \right).
\end{aligned} \tag{2}
$$

Now, the quantity on the right hand side of in display 2 is bounded above by

$$
\begin{aligned}
&(t^B)^{-1} \sum_{b=0}^{B-1} M^b \Delta_j + \\
&(t^B)^{-1} \sum_{b=0}^{B-1} N^b \Big(\Big[\Big\{ \left((t^b)^{-r_\epsilon^C} + K\gamma^{-N^b} \right) \left(1 - (t^b)^{-r_\delta^C} \right) \Big\} + \Delta_j K^C (t^b)^{-r_b^C (r_\delta^C - r_\epsilon^C)} \Big] \\
&\qquad + \Big[\left((t^b)^{-r_\epsilon^p} + K\gamma^{-N^b} \right) \left(1 - (t^b)^{-r_\delta^p} \right) + \Delta_j K^p (t^b)^{-r_b^p (r_\delta^p - r_\epsilon^p)} \Big] \Big) \\
&+ \overline{B}^D (V_j^p - V_j^C) \\
&\longrightarrow \overline{B}^D (V_j^p - V_j^C).
\end{aligned}
$$

Thus, $V_j(H^t, \sigma_1, \widetilde{\sigma}_j) \leq 0$, with strict inequality if $\overline{B}^D > 0$.

Moreover, Assumption 1 implies that $V_j(H^0, \sigma_1, \sigma_2) = V_j^C$. □

This algorithm is summarized in the following displays.

Algorithm 1: Cooperative learning algorithm for player 1, σ_1^C

Input: Welfare function estimator $H^t \mapsto \widehat{w}$, cooperative policy learner $\mathcal{O}^C$, history H^t, current block index b, sub-block sizes $\{M^{b'}, N^{b'}\}_{b' \leq b}$

$\chi^b \leftarrow 1$;

`/* Estimate cooperative policy */`

for $t = \sum_{b'<b}(M^{b'} + N^{b'})$ **to** $\sum_{b'<b}(M^{b'} + N^{b'}) + M^b$ **do**

  $\theta_1^t \leftarrow \mathcal{O}^C(H^{t-1})$;

  $\theta_2^{C,t} \leftarrow \mathcal{O}^C(H^{t-1})$;

  $A_1^t \leftarrow \pi_{\theta_1^t}(S^t)$;

  Observe A_2^t;

  if $A_2^t = \pi_{\theta_2^{C,t}}(S^t)$ *and* $\chi^b = 1$ **then**

    $\chi^b \leftarrow 1$;

  else

    $\chi^b \leftarrow 0$;

  $S^{t+1} \sim P(\cdot \mid S^t, A_1^t, A_2^t)$;

  $H^t \leftarrow H^{t-1} \cup \{A_1^t, A_2^t, R_1^t, R_2^t, S^{t+1}\}$;

$\theta^{b,C} \leftarrow \mathcal{O}^C(H^t)$;

`/* Deploy cooperative policy */`

for $t = \sum_{b'<b}(M^{b'} + N^{b'}) + M^b$ **to** $\sum_{b'<b}(M^{b'} + N^{b'}) + M^b + N^b$ **do**

  $A_1^t \leftarrow \pi_{\theta_1^{b,C}}(S^t)$;

  Observe A_2^t;

  if $A_2^t = \pi_{\theta_2^{b,C}}(S^t)$ *and* $\chi^b = 1$ **then**

    $\chi^b \leftarrow 1$;

  else

    $\chi^b \leftarrow 0$;

  $S^{t+1} \sim P(\cdot \mid S^t, A_1^t, A_2^t)$;

  $H^t \leftarrow H^{t-1} \cup \{A_1^t, A_2^t, R_1^t, R_2^t, S^{t+1}\}$;

return H^t, χ^b

Algorithm 2: Punishment learning algorithm for player 1, σ_1^p

Input: Welfare function estimator $H^t \mapsto \widehat{w}$, punishment policy learner $\mathcal{O}^p$, history H^t, current block index b, sub-block sizes $\{M^{b'}, N^{b'}\}_{b' \leq b}$

`/* Estimate punishment policy */`

for $t = \sum_{b'<b}(M^{b'} + N^{b'})$ **to** $\sum_{b'<b}(M^{b'} + N^{b'}) + M^b$ **do**

  $\theta_1^t \leftarrow \mathcal{O}^p(H^{t-1})$;

  $A_1^t \leftarrow \pi_{\theta_1^t}(S^t)$;

  Observe A_2^t;

  $S^{t+1} \sim P(\cdot \mid S^t, A_1^t, A_2^t)$;

  $H^t \leftarrow H^{t-1} \cup \{A_1^t, A_2^t, R_1^t, R_2^t, S^{t+1}\}$;

$\theta^{b,p} \leftarrow \mathcal{O}(H^t)$;

`/* Deploy punishment policy */`

for $t = \sum_{b'<b}(M^{b'} + N^{b'}) + M^b$ **to** $\sum_{b'<b}(M^{b'} + N^{b'}) + M^b + N^b$ **do**

  $A_1^t \leftarrow \pi_{\theta_1^{b,p}}(S^t)$;

  Observe A_2^t;

  $S^{t+1} \sim P(\cdot \mid S^t, A_1^t, A_2^t)$;

  $H^t \leftarrow H^{t-1} \cup \{A_1^t, A_2^t, R_1^t, R_2^t, S^{t+1}\}$;

return H^t

Algorithm 3: Learning equilibrium strategy

Input: Welfare function estimator $H^t \mapsto \widehat{w}$, cooperative policy learner $\mathcal{O}^C$, punishment policy learner $\mathcal{O}^p$

Initialize θ_1^0, state S^0. **for** $b = 0, 1, \ldots$ **do**

  if $\chi^{b-1} = 1$ **then**

    $H^{t^b}, \chi^b \leftarrow \texttt{Algorithm1}(H^{t^{b-1}})$;

  else if $\chi^{b-1} = 0$ **then**

    $H^{t^b} \leftarrow \texttt{Algorithm2}(H^{t^{b-1}})$;

    $\chi^b = 1$;

Appendix B: Test for cooperativeness under imperfect policy monitoring

We construct a procedure by which player i can test whether player j is playing the cooperative learning algorithm. For blocks of episodes $l = 1, \ldots, L$, let

$$\widehat{\theta}_j^{D,l} = \arg\max_{\theta_j} \prod_{u=1}^{U} P\{\pi_{\theta_j}(S^{lu}) = A_j^{lu}\}$$

be the maximum likelihood policy in the l^{th} block. This sequence of policies represents the hypotheses that player j is not playing a cooperative policy. We approximate the sequence $\widehat{\theta}_1^{D,l}, \widehat{\theta}_2^{D,l}, \ldots$ by a moving window of networks $\{\widehat{\theta}_j^{D,l}\}_{l=L-L'}^{L}$ updated by a stochastic gradient step on the supervised loss after each block l. We use a window size of $L' = 3$ for the iterated Prisoner's Dilemma (Sections 4.1 and 4.3) and $L' = 2$ for the Coin Game (Section 4.2)).

Let $\widehat{\theta}_j^{C,l}$ be an estimate of the cooperative policy. In the case where the learning rate δ is known (Sections 4.1 and 4.2), the sequence $\{\widehat{\theta}_j^{C,l}\}_{l=L-L'}^{L}$ is obtained by applying the base cooperative learning algorithm with the known learning rate. In the case where δ is unknown (Section 4.3), networks $\widehat{\theta}_j^{C,l}$ are obtained by an approximate solution to objective 1. We use a genetic algorithm, summarized in Algorithm 4.

We will use a bootstrap estimator of the sampling distribution of these log likelihoods to construct our test. To specify a hypothesis test, define the log likelihoods under cooperative and defecting learning algorithms (conditional on the sequence of states in H^t, and for window size t'):

$$\mathcal{L}_j^{C,t} = \sum_{v=t-t'}^{t} \log P\{\pi_{\widehat{\theta}_j^{C,v}}(S^v) = A_j^v\};$$

$$\mathcal{L}_j^{D,t} = \sum_{v=t-t'}^{t} \log P\{\pi_{\widehat{\theta}_j^{D,t}}(S^v) = A_j^v\}.$$

For $b = 1, \ldots, 50$, we then:

1. Draw a list H^{b_t} of 20 (S^v, A_j^v) pairs with replacement from the previous 200 observations;
2. Compute the log probabilities of each element of $H^{b,t}$ under the cooperative and defecting policy sequences to obtain the corresponding log likelihoods $\mathcal{L}_j^{C,b_t}$ and $\mathcal{L}_j^{D,b_t}$;
3. Take the difference in log likelihoods $\Delta_j^{b_t} = \mathcal{L}_j^{C,b_t} - \mathcal{L}_j^{D,b_t}$.

This results in a bootstrap distribution of values $\Delta_j^{b_t}$. If the 95^{th} percentile of this distribution, $\Delta_j^{(95),t}$, is less than 0, player i rejects the hypothesis that player j is cooperating, and punishes accordingly.

Notice that, to reduce computational costs, we do not re-fit the estimators $\{\widehat{\theta}_j^{C,l}\}_{l=L-L'}^{L}, \{\widehat{\theta}_j^{D,l}\}_{l=L-L'}^{L}$ on the bootstrapped data. For stability, we actually reject when the average of these percentiles over the past 20 steps is less than 0, i.e. reject if and only if $\frac{1}{20}\sum_{v=t-20}^{t} \Delta_j^{(95),v} < 0$.

Algorithm 4: Genetic algorithm for estimating $\theta_j^{C,L}$ under unknown learning rate δ

Input: Population size G

Initialize $\widehat{\delta}_1$;

$\widehat{\delta}^* \leftarrow \widehat{\delta}_1$;

$\widehat{\boldsymbol{\delta}} \leftarrow \{\widehat{\delta}_1\}$;

for $L = 1, 2, \ldots$ **do**

  if $|\widehat{\boldsymbol{\delta}}| < G$ **then**

    $\widehat{\boldsymbol{\delta}} \leftarrow \widehat{\boldsymbol{\delta}} \cup \{\widehat{\delta}^* \cdot \mathrm{Unif}(0.5, 2)\}$;

  for $\widehat{\delta}$ in $\widehat{\boldsymbol{\delta}}$ **do**

    $\widehat{\theta}_j^{C,L}(\widehat{\delta}) = \arg\min_{\theta_j} \sum_{l=1}^{L} \sum_{u=1}^{U} \ell(A^{lu}, S^{lu}, \theta_j^l)$ s.t. $\theta_j^{l+1} = \widehat{\delta} \nabla w^l(\widehat{\theta}_i^l, \theta_j^l)$;

    $\ell(\widehat{\delta}) \leftarrow \sum_{u=U-20}^{U} \ell\{A^{Lu}, S^{Lu}, \widehat{\theta}_j(\widehat{\delta})\}$;

  $\delta^* \leftarrow \arg\min_{\widehat{\delta} \in \widehat{\boldsymbol{\delta}}} \ell(\widehat{\delta})$;

  if $\mathrm{Unif}(0, 1) > 0.8$ **then**

    Kill $\arg\max_{\widehat{\delta} \in \widehat{\boldsymbol{\delta}}} \ell(\widehat{\delta})$;

  else

    Kill oldest element of $\widehat{\delta}$;

  return $\delta^*, \widehat{\theta}_j^{C,L}(\delta^*)$