Status-quo policy gradient in Multi-Agent Reinforcement Learning

Pinkesh Badjatiya*
Microsoft, India
pbadjatiya@microsoft.com

Mausoom Sarkar
Media and Data Science Research Lab, Adobe

Nikaash Puri
Media and Data Science Research Lab, Adobe

Jayakumar Subramanian
Media and Data Science Research Lab, Adobe

Abhishek Sinha*
Waymo
a7b23@stanford.edu

Siddharth Singh†
University of Maryland
siddharth9820@gmail.com

Balaji Krishnamurthy
Media and Data Science Research Lab, Adobe

{msarkar,jasubram,nikpuri,kbalaji}@adobe.com

Abstract

Individual rationality, which involves maximizing expected individual return, does not always lead to high-utility individual or group outcomes in multi-agent problems. For instance, in multi-agent social dilemmas, Reinforcement Learning (RL) agents trained to maximize individual rewards converge to a low-utility mutually harmful equilibrium. In contrast, humans evolve useful strategies in such social dilemmas. Inspired by ideas from human psychology that attribute this behavior to the status-quo bias, we present a status-quo loss ($SQLoss$) and the corresponding policy gradient algorithm that incorporates this bias in an RL agent. We demonstrate that agents trained with $SQLoss$ learn high-utility policies in several social dilemma matrix games (Prisoner's Dilemma, Stag Hunt matrix variant, Chicken Game). We show how $SQLoss$ outperforms existing state-of-the-art methods to obtain high-utility policies in visual input non-matrix games (Coin Game and Stag Hunt visual input variant) using pre-trained cooperation and defection oracles. Finally, we show that $SQLoss$ extends to a 4-agent setting by demonstrating the emergence of cooperative behavior in the popular Braess' paradox.

1 Introduction

In sequential social dilemmas, individually rational behavior can lead to outcomes that are sub-optimal for each individual in the group Hardin [1968], Ostrom [1990], Ostrom et al. [1999], Dietz et al. [2003]. Current state-of-the-art Multi-Agent Deep Reinforcement Learning (MARL) methods that train agents independently lead to agents that play individualistically, thereby receiving poor rewards, even in simple social dilemmas [Foerster et al., 2018, Lerer and Peysakhovich, 2017].

*Work done while at Media and Data Science Research Labs, Adobe
†Work done during the internship at Adobe

35th Conference on Neural Information Processing Systems (NeurIPS 2021), Sydney, Australia.

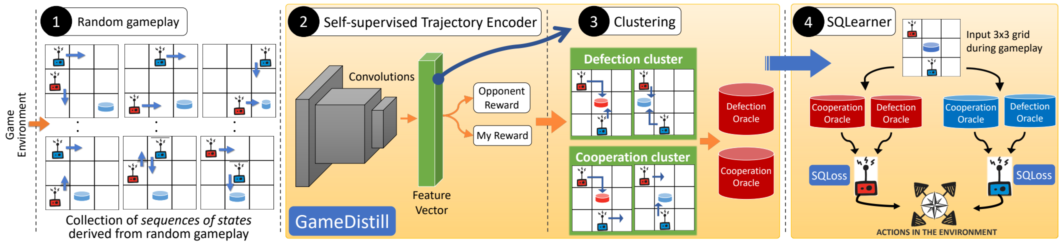

Figure 1: High-level architecture illustrated using coin game. Each agent runs $GameDistill$ by performing steps $(1), (2), (3)$ individually to obtain two oracles per agent. During game-play(4), each agent (with $SQLoss$) takes either the action suggested by the cooperation or the defection oracle

To illustrate why it is challenging to learn optimal policies in such dilemmas, we consider the Coin Game, which was first explored in [Foerster et al., 2018]. Each agent can play either selfishly (pick all coins) or cooperatively (pick only coins of its color). Regardless of the other agent's behavior, the individually rational choice for an agent is to play selfishly, either to minimize losses (avoid being exploited) or to maximize gains (exploit the other agent). However, when both agents behave rationally, they try to pick all coins and achieve an average long term reward of -0.5. In contrast, if both cooperate, then the average long term reward for each agent is 0.5. Therefore, when agents cooperate, they are both better off. Training Deep RL agents independently in the Coin Game using state-of-the-art methods leads to mutually harmful selfish behavior (Section 2.2).

The problem of how independently learning agents develop optimal behavior in social dilemmas has been studied by researchers through human studies and simulation models [Fudenberg and Maskin, 1986, Green and Porter, 1984, Fudenberg et al., 1994, Kamada and Kominers, 2010, Abreu et al., 1990]. A large body of work has looked at the mechanism of evolution of cooperation through reciprocal behaviour and indirect reciprocity [Trivers, 1971, Axelrod, 1984, Nowak and Sigmund, 1992, 1993, 1998], through variants of reinforcement using aspiration [Macy and Flache, 2002], attitude [Damer and Gini, 2008] or multi-agent reinforcement learning [Sandholm and Crites, 1996, Wunder et al., 2010], and under specific conditions [Banerjee and Sen, 2007] using different learning rates [de Cote et al., 2006] similar to WoLF [Bowling and Veloso, 2002] as well as using embedded emotion [Yu et al., 2015], social networks [Ohtsuki et al., 2006, Santos and Pacheco, 2006].

However, these approaches do not directly apply to Deep RL agents [Leibo et al., 2017]. Recent work in this direction [Kleiman-Weiner et al., 2016, Julien et al., 2017, Peysakhovich and Lerer, 2018] focuses on letting agents learn strategies through interactions with other agents in a multi-agent setting. Leibo et al. [2017] defines the problem of social dilemmas in the Deep RL framework and analyzes the outcomes of a fruit-gathering game [Julien et al., 2017]. They vary the abundance of resources and the cost of conflict in the fruit environment to generate degrees of cooperation between agents. Hughes et al. [2018] defines an intrinsic reward (inequality aversion) that attempts to reduce the difference in obtained rewards between agents. The agents are designed to have an aversion to both advantageous (guilt) and disadvantageous (unfairness) reward allocation. This handcrafting of loss with mutual fairness develops cooperation, but it leaves the agent vulnerable to exploitation. LOLA [Foerster et al., 2018] uses opponent awareness to achieve high cooperation levels in the Coin Game and the Iterated Prisoner's Dilemma game. However, the LOLA agent assumes access to the other agent's network architecture, observations, and learning algorithms. This access level is analogous to getting complete access to the other agent's private information and therefore devising a strategy with full knowledge of how they are going to play. Wang et al. [2019] proposes an evolutionary Deep RL setup to learn cooperation. They define an intrinsic reward that is based on features generated from the agent's past and future rewards, and this reward is shared with other agents. They use evolution to maximize the sum of rewards among the agents and thus learn cooperative behavior. However, sharing rewards in this indirect way enforces cooperation rather than evolving it through independently learning agents.

Interestingly, humans develop individual and socially optimal strategies in such social dilemmas without sharing rewards or having access to private information. Several ideas in human psychology [Samuelson and Zeckhauser, 1988, Kahneman et al., 1991, Kahneman, 2011, Thaler and Sunstein, 2009] have attributed this cooperative behavior to the status-quo bias [Guney and Richter, 2018]. The status-quo bias is a decision-making bias that encourages humans to stick to the status-quo unless doing so causes significant harm. This preference in humans for sticking to 'comfortable' states rather than seeking short-term exploitative gains is one explanation for emergent cooperative behavior in human groups. Inspired by this idea, we present the status-quo loss ($SQLoss$) and the corresponding status-quo policy gradient formulation for RL. Agents trained with $SQLoss$ learn optimal policies in multi-agent social dilemmas without sharing rewards, gradients, or using a communication channel. Intuitively, $SQLoss$ encourages an agent to stick to past actions provided these actions did not cause significant harm. Therefore, mutually cooperating agents stick to cooperation since the status-quo yields higher individual reward, while unilateral defection by any agent leads to the other agent also switching to defection since the status-quo is very harmful for the exploited agent. Subsequently, for each agent, the short-term reward of exploitation is overcome by the long-term cost of mutual defection, and agents gradually switch to cooperation.

To apply $SQLoss$ to games where a sequence of non-trivial actions determines cooperation and defection, we present $GameDistill$, an algorithm that reduces a dynamic game with visual input to a matrix game. $GameDistill$ uses self-supervision and clustering to extract distinct policies from a sequential social dilemma game automatically.

Our key contributions can be summarised as:

1. We introduce a **Status-Quo** loss ($SQLoss$, Sec. 2.3) and an associated policy gradient-based algorithm to learn optimal behavior in a decentralized manner for agents playing iterated matrix games where agents can choose between a cooperative and selfish policy at each step. We empirically demonstrate that agents trained with $SQLoss$ learn optimal behavior in several social dilemma iterated matrix games (Sec. 4).

2. We extend $SQLoss$ to social dilemma game with visual observations (Sec. 2.4) using $GameDistill$. We empirically demonstrate that $GameDistill$ extracts cooperative and selfish policies for the Coin Game and Stag Hunt (Sec. 4.2). We further show that incorporating $GameDistill$ in LOLA accelerates learning and achieves higher cooperation.

3. We also demonstrate that $SQLoss$ extends in a straight forward manner to games with more than two agents due to its ego-centric nature (Sec. 6).

2 Approach

2.1 Social Dilemmas modeled as Iterated Matrix Games

To remain consistent with previous work, we adopt the notations from Foerster et al. [2018]. We model social dilemmas as general-sum Markov (simultaneous move) games. A multi-agent Markov game is specified by $G = \langle S, A, U, P, r, n, \gamma \rangle$. S denotes the state space of the game. n denotes the number of agents playing the game. At each step of the game, each agent $a \in A$, selects an action $u^a \in U$. $\vec{u}$ denotes the joint action vector that represents the simultaneous actions of all agents. The joint action $\vec{u}$ changes the state of the game from s to s' according to the state transition function $P(s'|\vec{u}, s) : S \times \mathbf{U} \times S \to [0, 1]$. At the end of each step, each agent a gets a reward according to the reward function $r^a(s, \vec{u}) : S \times \mathbf{U} \to \mathbb{R}$. The reward obtained by an agent at each step is a function of the actions played by all agents. For an agent a, the discounted future return from time t is defined as $R_t^a = \sum_{l=0}^{\infty} \gamma^l r_{t+l}^a$, where $\gamma \in [0, 1)$ is the discount factor. Each agent independently attempts to maximize its expected discounted return.

Matrix games are the special case of two-player perfectly observable Markov games [Foerster et al., 2018]. Table 1 shows examples of matrix games that represent social dilemmas. Consider the Prisoner's Dilemma game in Table 1a. Each agent can either cooperate (*C*) or defect (*D*). Playing *D* is the rational choice for an agent, regardless of whether the other agent plays *C* or *D*. Therefore, if both agents play rationally, they each receive a reward of -2. However, if each agent plays *C*, then it will obtain a reward of -1. This fact that individually rational behavior leads to a sub-optimal group (and individual) outcome highlights the dilemma.

In Infinitely Iterated Matrix Games, agents repeatedly play a particular matrix game against each other. In each iteration of the game, each agent has access to actions played by both agents in the previous iteration. Therefore, the state input to an RL agent consists of both agents' actions in the previous iteration of the game. We adopt this state formulation as is typically done in such games

[Press and Dyson, 2012, Foerster et al., 2018]. The infinitely iterated variations of the matrix games in Table 1 represent sequential social dilemmas. We refer to infinitely iterated matrix games as iterated matrix games in subsequent sections for convenience.

2.2 Learning Policies in Iterated Matrix Games: The Selfish Learner

The standard method to model agents in iterated matrix games is to model each agent as an RL agent that independently attempts to maximize its expected total discounted reward. Several approaches to model agents in this way use policy gradient-based methods [Sutton et al., 2000, Williams, 1992]. Policy gradient methods update an agent's policy, parameterized by θ^a, by performing gradient ascent on the expected total discounted reward $\mathbb{E}[R_0^a]$. Formally, let θ^a denote the parameterized version of an agent's policy π^a and $V^a_{\theta^1,\theta^2}$ denote the total expected discounted reward for agent a. Here, V^a is a function of the policy parameters (θ^1, θ^2) of both agents. In the i^{th} iteration of the game, each agent updates θ^a_i to θ^a_{i+1}, such that it maximizes it's total expected discounted reward. θ^a_{i+1} is computed using Eq. 1. For agents trained using reinforcement learning, the gradient ascent rule to update θ^1_{i+1} is given by Eq. 2.

$$\theta^1_{i+1} = \arg\max_{\theta^1} V^1(\theta^1, \theta^2_i) \quad \text{and} \quad \theta^2_{i+1} = \arg\max_{\theta^2} V^2(\theta^1_i, \theta^2) \tag{1}$$

$$f^1_{nl} = \nabla_{\theta^i_1} V^1(\theta^1_i, \theta^2_i) \cdot \delta \quad \text{and} \quad \theta^1_{i+1} = \theta^1_i + f^1_{nl}(\theta^1_i, \theta^2_i) \tag{2}$$

where δ is the step size of the updates. In the Iterated Prisoner's Dilemma (IPD) game, agents trained with the policy gradient update method converge to a sub-optimal mutual defection equilibrium (Figure 3, Lerer and Peysakhovich [2017]). This sub-optimal equilibrium attained by Selfish Learners motivates us to explore alternative methods that could lead to a desirable cooperative equilibrium. We denote the agent trained using policy gradient updates as a Selfish Learner (SL).

2.3 Learning Policies in Iterated Matrix Games: The Status-Quo Aware Learner ($SQLoss$)

2.3.1 $SQLoss$: Motivation and Theory

The status-quo bias instills in humans a preference for the current state provided the state is not harmful to them. Inspired by this idea, we introduce a status-quo loss ($SQLoss$) for each agent, derived from the idea of imaginary game-play (Figure 2). The $SQLoss$ encourages an agent to

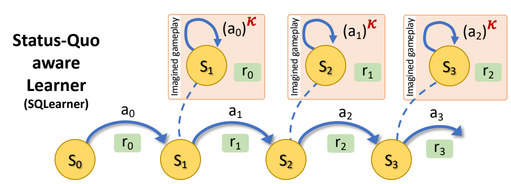

Figure 2: Intuition behind *Status-Quo*-aware learner. At each step, the $SQLoss$ encourages an agent to imagine the consequences of sticking to the status-quo by imagining an episode where the status-quo is repeated for κ steps. Section 2.3 describes $SQLoss$ in more detail.

imagine a future episode where the status-quo (current situation) is repeated for several steps. If an agent has been exploited in the previous iteration of the game (state *DC*), then $SQLoss$ will encourage the agent to imagine a continued risk of exploitation and subsequently switch to defection and move to state *DD*. Hence, for the exploiting agent, the short-term gain from exploitation (*DC*) is overcome by the long-term loss from mutual defection (*DD*). Conversely, if both agents cooperated in the previous iteration of the game (state *CC*), then $SQLoss$ will encourage the agent to imagine a continued gain from mutual cooperation and subsequently stick to state *CC*. Since state *CC* is more beneficial to both agents than *DD*, and exploitation (*DC* and *CD*) also leads to *DD*, agents move away from defection and converge to mutual cooperation (*CC*). Appendix C describes in detail how cooperation emerges between $SQLoss$ agents.

2.3.2 $SQLoss$: Formulation

We describe below the formulation of SQLoss with respect to agent 1. The formulation for agent 2 is identical to that of agent 1. Let $\tau_a = (s_0, u^1_0, u^2_0, r^1_0, \cdots s_T, u^1_T, u^2_T, r^1_T)$ denote the collection

	C	*D*
C	(-1, -1)	(-3, 0)
D	(0, -3)	(-2, -2)

(a) Prisoners' Dilemma (PD)

	H	*T*
H	(+1, -1)	(-1, +1)
T	(-1, +1)	(+1, -1)

(b) Matching Pennies (MP)

	C	*D*
C	(0, 0)	(-4, -1)
D	(-1, -4)	(-3, -3)

(c) Stag Hunt (SH)

Table 1: Payoff matrices for the different games used in our experiments. (X, Y) in a cell represents a reward of X to the row and Y to the column player. *C*, *D*, *H*, and *T* denote the actions for the row and column players. In the iterated versions of these games, agents play against each other over several iterations. In each iteration, an agent takes an action and receives a reward based on the actions of both agents. Each matrix represents a different kind of social dilemma.

of an agent's experiences after T time steps. Let $R^1_t(\tau_1) = \sum_{l=t}^{T} \gamma^{l-t} r^1_l$ denote the discounted future return for agent 1 starting at s_t in actual game-play. Let $\hat{\tau}_1$ denote the collection of an agent's **imagined** experiences. For a state s_t, where $t \in [0, T]$, an agent imagines an episode by starting at s_t and repeating u^1_{t-1}, u^2_{t-1} for κ_t steps. This is equivalent to imagining a κ_t step repetition of already played actions. We sample κ_t from a Discrete Uniform distribution $\mathbb{U}\{1, z\}$ where z is a hyper-parameter ≥ 1. To simplify notation, let $\phi_t(s_t, \kappa_t)$ denote the ordered set of state, actions, and rewards starting at time t and repeated κ_t times for imagined game-play. Let $\hat{R}^1_t(\hat{\tau}_1)$ denote the discounted future return starting at s_t in imagined status-quo game-play.

$$\phi_t(s_t, \kappa_t) = \big[(s_t, u^1_{t-1}, u^2_{t-1}, r^1_{t-1})_0, \; \cdots (s_t, u^1_{t-1}, u^2_{t-1}, r^1_{t-1})_1, (s_t, u^1_{t-1}, u^2_{t-1}, r^1_{t-1})_{\kappa_t - 1}\big] \tag{3}$$

$$\hat{\tau}_1 = \big(\phi_t(s_t, \kappa_t), (s_{t+1}, u^1_{t+1}, u^2_{t+1}, r^1_{t+1})_{\kappa_t+1}, \; \cdots, (s_T, u^1_T, u^2_T, r^1_T)_{T+\kappa_t-t}\big) \tag{4}$$

$$\hat{R}^1_t(\hat{\tau}_1) = \Big(\frac{1-\gamma^\kappa}{1-\gamma}\Big) r^1_{t-1} + \gamma^\kappa R^1_t(\tau_1) = \Big(\frac{1-\gamma^\kappa}{1-\gamma}\Big) r^1_{t-1} + \gamma^\kappa \sum_{l=t}^{T} \gamma^{l-t} r^1_l \tag{5}$$

$V^1_{\theta^1,\theta^2}$ and $\hat{V}^1_{\theta^1,\theta^2}$ are approximated by $\mathbb{E}[R^1_0(\tau_1)]$ and $\mathbb{E}[\hat{R}^1_0(\hat{\tau}_1)]$ respectively. These V values are the expected rewards conditioned on both agents' policies (π^1, π^2). For agent 1, the regular gradients and the Status-Quo gradient-like correction term, $\nabla_{\theta^1} \mathbb{E}[R^1_0(\tau_1)]$ and f^1_{sqcor}, can be derived from the policy gradient formulation as

$$\nabla_{\theta^1} \mathbb{E}[R^1_0(\tau_1)] = \mathbb{E}[R^1_0(\tau_1) \nabla_{\theta^1} \log \pi^1(\tau_1)] = \mathbb{E}\Big[\sum_{t=1}^{T} \nabla_{\theta^1} \log \pi^1(u^1_t | s_t) \gamma^t \big(R^1_t(\tau_1) - b(s_t)\big)\Big] \tag{6}$$

$$\begin{aligned} f^1_{sqcor} &= \mathbb{E}\left[\sum_{t=1}^{T} \nabla_{\theta^1} \log \pi^1(u^1_{t-1} | s_t) \times \gamma^t \times \left(\Big(\frac{1-\gamma^\kappa}{1-\gamma}\Big) r^1_{t-1} + \gamma^\kappa \sum_{l=t}^{T} \gamma^{l-t} r^1_l - b(s_t)\right)\right] \\ &= \mathbb{E}\Big[\sum_{t=1}^{T} \nabla_{\theta^1} \log \pi^1(u^1_{t-1} | s_t) \times \gamma^t \times \big(\hat{R}^1_t(\hat{\tau}_1) - b(s_t)\big)\Big] \end{aligned} \tag{7}$$

Here, f^1_{sqcor} in Eq. 7 is a biased term that has policy gradient-like structure and incorporates the prior action taken through the $u_{t-1} | s_t$ term. It should be noted that the f^1_{sqcor} is not a gradient of the expected return of the imagined trajectory in Eq. 5, but the form of this expression is arrived at considering the policy gradient expression. Furthermore, $b(s_t)$ is a baseline-like term added for variance reduction in the implementation.

The update rule $f_{sql,pg}$ for the policy gradient-based Status-Quo Learner (SQL-PG) is defined by,

$$f^1_{sql,pg} = \big(\alpha \cdot \nabla_{\theta^1} \mathbb{E}[R^1_0(\tau_1)] + \beta \cdot f^1_{sqcor}\big) \cdot \delta \tag{8}$$

where α, β are the loss scaling factor for REINFORCE and imaginative game-play respectively.

2.4 Learning policies in Dynamic Non-Matrix Games using $SQLoss$ and $GameDistill$

The previous section focused on learning optimal policies in iterated matrix games that represent sequential social dilemmas. In such games, an agent can take one of a discrete set of policies at each step. For instance, in IPD, an agent can either cooperate or defect at each step. However, in social dilemmas such as the Coin Game (Appendix A), cooperation and defection policies are composed of a sequence of state-dependent actions. $SQLoss$, proposed above, works for matrix games but is not directly applicable to games with visual input to yield mutual cooperation. To apply the Status-Quo policy gradient to these games, we present $GameDistill$, a self-supervised algorithm that reduces a dynamic infinitely iterated game with visual input to a matrix game. $GameDistill$ takes as input game-play episodes between agents with random policies and learns oracles (or policies) that lead to distinct outcomes. $GameDistill$ (Figure 1) works as follows.

1. We initialize agents with random weights and play them against each other in the game. In these **random game-play** episodes, whenever an agent receives a reward, we store the sequence of states along with the rewards for both agents.
2. This collection of state sequences is used to train the $GameDistill$ network, which is a **self-supervised trajectory encoder**. It takes as input a sequence of states and predicts the rewards of both agents during training.
3. We now **cluster the embeddings** extracted from the penultimate layer of the trained $GameDistill$ network using Agglomerative Clustering [Friedman et al., 2001]. Each embedding is a finite-dimensional representation of the corresponding state sequence. Each cluster represents a collection of state sequences or transitions, that lead to a consistent outcome (w.r.t rewards). For CoinGame, when we use the number of clusters as two, we observe that one cluster consists of transitions that represent cooperative behavior (cooperation cluster) while the other contains transitions that lead to defection (defection cluster).
4. Using the state sequences in each cluster, we **train an oracle** to predict the next action given the current state. For the Coin Game, the oracle trained on state sequences from the cooperation cluster predicts the cooperative action for a given state. Similarly, the oracle trained on the defection cluster predicts the defection action for a given state. Each agent uses $GameDistill$ independently to extract a cooperation and a defection oracle. Figure 15 (Appendix J) illustrates the cooperation and defection oracles extracted by the Red agent.

During game-play, an agent can consult either oracle at each step. In CoinGame, this is equivalent to either cooperating (consulting the cooperation oracle) or defecting (consulting the defection oracle). In this way, an agent reduces a dynamic game to its matrix equivalent using $GameDistill$. We then utilize the Status-Quo policy gradient to learn optimal policies in the reduced matrix game. For the Coin Game, this leads to agents who cooperate by only picking coins of their color (Figure 5). It is important to note that for games like CoinGame, we could have learned cooperation and defection oracles by training agents using the sum-of-rewards for both agents and individual reward, respectively [Lerer and Peysakhovich, 2017]. However, $GameDistill$ learns distinct policies without using hand-crafted reward functions. $GameDistill$ is applicable only in games where cooperation and defection policies are clearly defined and lead to distinct payoffs (rewards) for both players.

Algorithm 1 in Appendix provides the pseudo-code for $GameDistill$ with additional details in Appendix B.3. Appendix B.2 provides additional details about the architecture and the different components of $GameDistill$. Further, we have verified $GameDistill$ empirically on the Coin Game and the Stag Hunt (results in Appendix G). Additional clustering visualizations for the trajectories and the experimental plots are provided in Appendix F and J.

3 Experimental Setup

In order to compare our results to previous work, we use the Normalized Discounted Reward or $NDR = (1-\gamma)\sum_{t=0}^{T}\gamma^t r_t$. A higher NDR implies that an agent obtains a higher reward in the environment. We compare our approach (Status-Quo Aware Learner or $SQLearner$) to Learning with Opponent-Learning Awareness (Lola-PG) [Foerster et al., 2018] and the Selfish Learner (SL) agents. For all experiments, we perform 20 runs and report average NDR, along with variance across runs. The bold line in all the figures is the mean, and the shaded region is the one standard deviation region around the mean.

3.1 Iterated Matrix Game Social Dilemmas

For our experiments with social dilemma matrix games, we use the Iterated Prisoners Dilemma (IPD) [Luce and Raiffa, 1989], Iterated Matching Pennies (IMP) [Lee and Louis, 1967], and the

Iterated Stag Hunt (ISH) [Fang et al., 2002]. Each matrix game in Table 1 represents a different dilemma. In the Prisoner's Dilemma, the rational policy for each agent is to defect, regardless of the other agent's policy. However, when each agent plays rationally, each is worse off. In Matching Pennies, if an agent plays predictably, it is prone to exploitation by the other agent. Therefore, the optimal policy is to randomize between *H* and *T*, obtaining an average NDR of 0. The Stag Hunt game represents a coordination dilemma. In the game, given that the other agent will cooperate, an agent's optimal action is to cooperate as well. However, each agent has an attractive alternative at each step, that of defecting and obtaining a guaranteed reward of -1. Therefore, the promise of a safer alternative and the fear that the other agent might select the safer choice could drive an agent to select the safer alternative, thereby sacrificing the higher reward of mutual cooperation.

In iterated matrix games, at each iteration (iter), agents take an action according to a policy and receive the rewards in Table 1. To simulate an infinitely iterated game, we let agents play 200 iters of game against each other, and do not provide an agent with any information about the number of remaining iters. In an iter, state for an agent is the actions played by both agents in the previous iter.

3.2 Iterated Dynamic Game Social Dilemmas

For our experiments on a social dilemma with extended actions, we use the Coin Game (Figure 7) [Foerster et al., 2018] and the non-matrix variant of the Stag Hunt (Figure 8). We provide details of the games in Appendix A due to space considerations.

4 Results

4.1 Learning optimal policies in Iterated Matrix Dilemmas

Iterated Prisoner's Dilemma (IPD): We train different learners to play the IPD game. Figure 3 shows the results. For all learners, agents initially defect and move towards an NDR of -2.0. This initial bias towards defection is expected, since, for agents trained with random game-play episodes, the benefits of exploitation outweigh the costs of mutual defection. For Selfish Learner (SL) agents, the bias intensifies, and the agents converge to mutually harmful selfish behavior (NDR of -2.0). Lola-PG agents learn to predict each other's behavior and realize that defection is more likely to lead to mutual harm. They subsequently move towards cooperation, but occasionally defect (NDR of -1.2). In contrast, $SQLearner$ agents quickly realize the costs of defection, indicated by the small initial dip in the NDR curves. They subsequently move towards close to 100% cooperation, with an NDR of -1.0. Finally, it is important to note that $SQLearner$ agents have close to zero variance, unlike other methods where the variance in NDR across runs is significant.

Iterated Matching Pennies (IMP): We train different learners to play the IMP game. Figure 4 shows the results. $SQLearner$ agents learn to play optimally and obtain an NDR close to 0. Interestingly, Selfish Learner (SL) and Lola-PG agents converge to an exploiter-exploited equilibrium where one agent consistently exploits the other agent. This asymmetric exploitation equilibrium is more pronounced for SL agents than for Lola-PG agents. As before, we observe that $SQLearner$ agents have close to zero variance across runs, unlike other methods where the variance in NDR is significant.

Iterated Chicken Game (ICG): Appendix H shows additional results for the ICG game.

4.2 Learning Optimal Policies in Iterated Dynamic Dilemmas

GameDistill: To evaluate the clustering step in $GameDistill$, we make two t-SNE [Maaten and Hinton, 2008] plots of the 100-dimensional feature vectors extracted from the penultimate layer of the trained $GameDistill$ network in Figure 9 of Appendix F. In the first plot, we color each point (or state sequence) by the rewards obtained by both agents in the format $r_1|r_2$. In the second, we color each point by the cluster label output by the clustering technique. $GameDistill$ correctly discovers two clusters, one for transitions that represent cooperation (Red cluster) and the other for transitions that represent defection (Blue cluster). We experiment with different values for feature vector dimensions and clustering techniques, and obtain similar results (see Appendix B). Results on Stag Hunt using $GameDistill$ are presented in Appendix F and G. To evaluate the trained oracles that represent cooperation and a defection policy, we modify the CoinGame environment to contain only a single agent (the Red agent). We then play two variations of the game. In the first variant, the

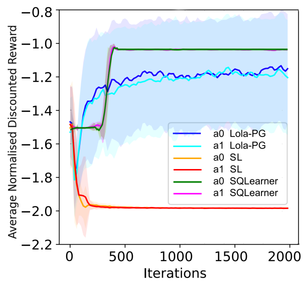

Figure 3: Average NDR values for different learners in IPD. $SQLearner$ agents obtain a near-optimal NDR value (-1) for this game.

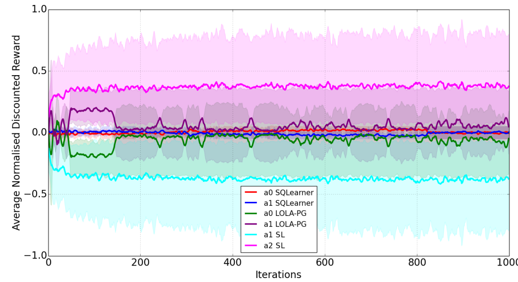

Figure 4: Average NDR values for different learners in IMP. $SQLearner$ agents avoid exploitation by randomising between H and T to obtain a near-optimal NDR value (0).

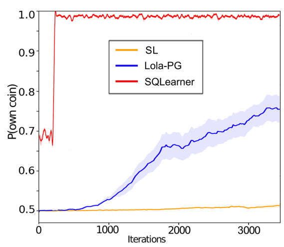

Figure 5: Probability that an agent will pick a coin of its color in CoinGame. $SQLearner$ agents achieve a cooperation rate close to 1.0 while Lola-PG agents achieve rate close to 0.8.

Figure 6: Impact of using pretrained $GameDistill$ oracles in CoinGame. $SQLearner$ agents (w/ $GameDistill$ oracles) achieve close to 100% cooperation. Lola-PG agents eventually (after roughly 4000 epochs) achieve close to 80% cooperation. Interestingly, Lola-PG agents trained w/ $GameDistill$ oracles converge to higher cooperation rate of 85% in 300 epochs.

Red agent executes the actions as suggested by the first oracle. We observe that Red agent picks only 8.4% of Blue coins, indicating a high cooperation rate, which represents a cooperation policy. In the second variant, Red agent executes the actions suggested by the second oracle. We observe that Red agent picks 99.4% of Blue coins, indicating high defection rate, which represents a defection policy.

SQLoss: During game-play, at each step, an agent follows either the action suggested by its cooperation oracle or defection oracle. We compare approaches using the degree of cooperation between agents, measured by the probability that an agent will pick the coin of its color [Foerster et al., 2018]. Figure 5 shows the results. The probability that an $SQLearner$ agent will pick the coin of its color is close to 1.0. This high probability indicates that the other $SQLearner$ agent is cooperating with this agent and only picking coins of its color. In contrast, the probability that a Lola-PG agent will pick a coin of its color is close to 0.8, indicating higher defection rates. As expected, the probability of an agent picking its own coin is the smallest for Selfish Learner.

To evaluate the impact of $GameDistill$, we compare $SQLearner$ agents to Lola-PG agents trained using $GameDistill$ oracles. Figure 6 shows the results. Interestingly, Lola-PG agents trained with $GameDistill$ oracles converge to a higher cooperation rate (0.88) than Lola-PG agents trained directly on visual input.

5 $SQLoss$ for social dilemma matrix games

For a given matrix game, $SQLoss$ will work on an equivalent version of the game in which all rewards have been transformed to non-positive values. For the matrix games described in our paper, we have used their variants with negative rewards to remain consistent with the LOLA paper. For a general matrix game, we can subtract the maximum reward (or any number larger than it) from each reward value to make rewards negative and then use $SQLoss$. We consider the social dilemma class of matrix games from Leibo et al. [2017], $\begin{pmatrix} & C & D \\ C & R, R & S, T \\ D & T, S & P, P \end{pmatrix}$,where the first row corresponds to cooperation for the first player and the first column corresponds to cooperation for the second player. Leibo et al. [2017] define the following rules that describe different categories of social dilemmas: **(i)** If $R > P$, then mutual cooperation is preferred to mutual defection. **(ii)** If $R > S$, then mutual cooperation is preferred to being exploited by a defector. **(iii)** If $2R > T + S$, this ensures that mutual cooperation is preferred to an equal probability of unilateral cooperation and defection. **(iv)** Either greed ($T > R$: Exploit a cooperator preferred over mutual cooperation) or fear ($P > S$: Mutual defection preferred over being exploited) should hold. These rules have been reproduced from Leibo et al. [2017],

1. When "greed" ($T > R$) as well as "fear" ($P > S$) conditions hold, we have $T > R > P > S$. The Iterated Prisoner's Dilemma (IPD) is an example of this game. If we subtract T from each reward, we get the matrix game, $\begin{pmatrix} & C & D \\ C & R-T, R-T & S-T, 0 \\ D & 0, S & P-T, P-T \end{pmatrix}$,where all entries are non-positive and therefore Lemma 3 and Lemma 4 hold as before.
2. When "greed" holds but not "fear" we have $T > R > S >= P$. The Chicken Game (CG) is an example of this game. If we subtract T from each reward as before, we get the equivalent matrix game with non-positive entries and Lemmas 3 and 4 hold.
3. When "fear" holds but not "greed" we have $R >= T > P > S$. The Iterated Stag Hunt is an example of this game. If we subtract R from each reward, we get the equivalent matrix game with non-positive entries then Lemmas 3 and 4 (Appendix C) hold.

In our experiments, we have considered games from each of the classes mentioned above, and the use of $SQLoss$ leads to cooperation in all these examples.

6 Games with more than 2 players

Our formulation of $SQLoss$ has the distinct advantage of being fully ego-centric, i.e., the learning agent does not require any information regarding its opponents. This feature enables a straightforward extension of $SQLoss$ beyond the two agent setting, without any change in each agent's learning algorithm. In order to test this extension of $SQLoss$ beyond 2-players, we consider as an example, the problem described in the popular Braess' paradox[Braess, 1968], which is a well-known extension of the Prisoner's Dilemma problem to more than 2 agents. We performed additional experiment in this game with 4 agents. We simulated the result when all agents are selfish learners (the SL agent(s)) and also when all agents use $SQLoss$ (the $SQLearner$(s)). We observe that when using selfish learners, all agents converge to Defection and when using $SQLoss$, all agents converge to Cooperation. Detailed explanation of the game, experimental setup and results are shown in Appendix D.

7 Conclusion

We presented a status-quo policy gradient that encourages an agent to imagine the consequences of sticking to the status quo. We demonstrated how $SQLoss$ outperforms LOLA on standard benchmark matrix games. To work with dynamic games, we further proposed $GameDistill$, an algorithm that reduces a dynamic game with visual input to a matrix game. We combined $GameDistill$ and $SQLoss$ to demonstrate how agents learn optimal policies in dynamic social dilemmas with visual observations. We empirically demonstrated that $SQLoss$ obtains near-optimal rewards in various social-dilemma games such as IPD, IMP, Chicken, Stag-Hunt and Coin Game (both matrix and variant with visual observations).

References

Dilip Abreu, David Pearce, and Ennio Stacchetti. Toward a theory of discounted repeated games with imperfect monitoring. *Econometrica*, 58(5):1041–1063, 1990. ISSN 00129682, 14680262. URL `http://www.jstor.org/stable/2938299`.

Robert Axelrod. *Robert Axelrod's (1984) The Evolution of Cooperation*. Basic Books, 1984.

Dipyaman Banerjee and Sandip Sen. Reaching pareto-optimality in prisoner's dilemma using conditional joint action learning. *Autonomous Agents and Multi-Agent Systems*, 15(1), August 2007. ISSN 1387-2532.

Michael Bowling and Manuela Veloso. Multiagent learning using a variable learning rate. *Artificial Intelligence*, 136(2):215–250, April 2002. ISSN 0004-3702.

D. Braess. Über ein paradoxon aus der verkehrsplanung. *Unternehmensforschung*, 12(1):258–268, Dec 1968.

Anonymous Code. MARL with SQLoss. `https://github.com/user12423/MARL-with-SQLoss/`, 2019.

Steven Damer and Maria Gini. Achieving cooperation in a minimally constrained environment. volume 1, pages 57–62, 01 2008.

Enrique Munoz de Cote, Alessandro Lazaric, and Marcello Restelli. Learning to cooperate in multi-agent social dilemmas. In *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems*, AAMAS '06, 2006.

Thomas Dietz, Elinor Ostrom, and Paul C. Stern. The struggle to govern the commons. *Science*, 302 (5652):1907–1912, 2003. doi: 10.1126/science.1091015.

Christina Fang, Steven Orla Kimbrough, Stefano Pace, Annapurna Valluri, and Zhiqiang Zheng. On adaptive emergence of trust behavior in the game of stag hunt. *Group Decision and Negotiation*, 11(6):449–467, 2002.

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.

Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics New York, 2001.

Drew Fudenberg and Eric Maskin. The folk theorem in repeated games with discounting or with incomplete information. *Econometrica*, 54(3):533–554, 1986. ISSN 00129682, 14680262.

Drew Fudenberg, David Levine, and Eric Maskin. The folk theorem with imperfect public information. *Econometrica*, 62(5):997–1039, 1994. ISSN 00129682, 14680262. URL `http://www.jstor.org/stable/2951505`.

Edward J Green and Robert H Porter. Noncooperative Collusion under Imperfect Price Information. *Econometrica*, 52(1):87–100, 1984.

Begum Guney and Michael Richter. Costly switching from a status quo. *Journal of Economic Behavior & Organization*, 156:55–70, 2018.

Garrett Hardin. The tragedy of the commons. *Science*, 162(3859):1243–1248, 1968. doi: 10.1126/science.162.3859.1243.

Edward Hughes, Joel Z. Leibo, Matthew Phillips, Karl Tuyls, Edgar Dueñez Guzman, Antonio García Castañeda, Iain Dunning, Tina Zhu, Kevin McKee, Raphael Koster, Heather Roff, and Thore Graepel. Inequity aversion improves cooperation in intertemporal social dilemmas. In *Proceedings of the 32Nd International Conference on Neural Information Processing Systems*, NIPS'18, 2018.

Pérolat Julien, JZ Leibo, V Zambaldi, C Beattie, Karl Tuyls, and Thore Graepel. A multi-agent reinforcement learning model of common-pool resource appropriation. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, 12 2017.

Daniel Kahneman. *Thinking, fast and slow*. Macmillan, 2011.

Daniel Kahneman, Jack L Knetsch, and Richard H Thaler. Anomalies: The endowment effect, loss aversion, and status quo bias. *Journal of Economic perspectives*, 5(1):193–206, 1991.

Yuichiro Kamada and Scott Kominers. Information can wreck cooperation: A counterpoint to kandori (1992). *Economics Letters*, 107:112–114, 05 2010. doi: 10.1016/j.econlet.2009.12.040.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Max Kleiman-Weiner, Mark K Ho, Joseph L Austerweil, Michael L Littman, and Joshua B Tenenbaum. Coordinate to cooperate or compete: abstract goals and joint intentions in social interaction. In *CogSci*, 2016.

King Lee and K Louis. *The Application of Decision Theory and Dynamic Programming to Adaptive Control Systems*. PhD thesis, 1967.

Joel Z. Leibo, Vinicius Zambaldi, Marc Lanctot, Janusz Marecki, and Thore Graepel. Multi-agent reinforcement learning in sequential social dilemmas. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '17. International Foundation for Autonomous Agents and Multiagent Systems, 2017.

Adam Lerer and Alexander Peysakhovich. Maintaining cooperation in complex social dilemmas using deep reinforcement learning, 2017.

R Duncan Luce and Howard Raiffa. *Games and decisions: Introduction and critical survey*. Courier Corporation, 1989.

Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

Dr Macy and Andreas Flache. Learning dynamics in social dilemmas. *Proceedings of the National Academy of Sciences of the United States of America*, 99 Suppl 3:7229–36, 06 2002. doi: 10.1073/pnas.092080099.

Martin Nowak and Karl Sigmund. A strategy of win-stay, lose-shift that outperforms tit-for-tat in the prisoner's dilemma game. *Nature*, 364:56–8, 08 1993. doi: 10.1038/364056a0.

Martin A. Nowak and Karl Sigmund. Tit for tat in heterogeneous populations. *Nature*, 355(6357): 250–253, 1992.

Martin A. Nowak and Karl Sigmund. Evolution of indirect reciprocity by image scoring. *Nature*, 393 (6685):573–577, 1998.

Hisashi Ohtsuki, Christoph Hauert, Erez Lieberman, and Martin A. Nowak. A simple rule for the evolution of cooperation on graphs and social networks. *Nature*, 441(7092):502–505, 2006. ISSN 1476-4687. doi: 10.1038/nature04605. URL `https://doi.org/10.1038/nature04605`.

E. Ostrom. *Governing the commons-The evolution of institutions for collective actions*. Political economy of institutions and decisions, 1990.

Elinor Ostrom, Joanna Burger, Christopher B. Field, Richard B. Norgaard, and David Policansky. Revisiting the commons: Local lessons, global challenges. *Science*, 284(5412):278–282, 1999. doi: 10.1126/science.284.5412.278.

Alexander Peysakhovich and Adam Lerer. Consequentialist conditional cooperation in social dilemmas with imperfect information. In *International Conference on Learning Representations, ICLR 2018,Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*, 2018.

William H Press and Freeman J Dyson. Iterated prisoner's dilemma contains strategies that dominate any evolutionary opponent. *Proceedings of the National Academy of Sciences*, 109(26):10409–10413, 2012.

William Samuelson and Richard Zeckhauser. Status quo bias in decision making. *Journal of risk and uncertainty*, 1(1):7–59, 1988.

Tuomas W. Sandholm and Robert H. Crites. Multiagent reinforcement learning in the iterated prisoner's dilemma. *Bio Systems*, 37 1-2:147–66, 1996.

Felipe Santos and J Pacheco. A new route to the evolution of cooperation. *Journal of evolutionary biology*, 19:726–33, 06 2006. doi: 10.1111/j.1420-9101.2005.01063.x.

Richard S Sutton and Andrew G Barto. Reinforcement learning: An introduction. 2011.

Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, pages 1057–1063, 2000.

Richard H Thaler and Cass R Sunstein. *Nudge: Improving decisions about health, wealth, and happiness*. Penguin, 2009.

Robert Trivers. The evolution of reciprocal altruism. *Quarterly Review of Biology*, 46:35–57., 03 1971. doi: 10.1086/406755.

Jane X. Wang, Edward Hughes, Chrisantha Fernando, Wojciech M. Czarnecki, Edgar A. Duéñez Guzmán, and Joel Z. Leibo. Evolving intrinsic motivations for altruistic behavior. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '19, pages 683–692. International Foundation for Autonomous Agents and Multiagent Systems, 2019. ISBN 978-1-4503-6309-9.

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

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

C. Yu, M. Zhang, F. Ren, and G. Tan. Emotional multiagent reinforcement learning in spatial social dilemmas. *IEEE Transactions on Neural Networks and Learning Systems*, 26(12):3083–3096, 2015.