Inducing Cooperation in Multi-Agent Games Through Status-Quo Loss

Pinkesh Badjatiya[1], Mausoom Sarkar[1], Abhishek Sinha[1], Siddharth Singh[2]*,
Nikaash Puri[1], Balaji Krishnamurthy[1]
[1]Media and Data Science Research Lab, Adobe
[2]IIT Kharagpur

ABSTRACT

Social dilemma situations bring out the conflict between individual and group rationality. When individuals act rationally in such situations, the group suffers sub-optimal outcomes. The Iterative Prisoner's Dilemma (IPD) is a two-player game that offers a theoretical framework to model and study such social situations. In the Prisoner's Dilemma, individualistic behavior leads to mutual defection and sub-optimal outcomes. This result is in contrast to what one observes in human groups, where humans often sacrifice individualistic behavior for the good of the collective. It is interesting to study how and why such cooperative and individually irrational behavior emerges in human groups. To this end, recent work models this problem by treating each player as a Deep Reinforcement Learning (RL) agent and evolves cooperative behavioral policies through internal information or reward sharing mechanisms. We propose an approach to evolve cooperative behavior between RL agents playing the IPD game without sharing rewards, internal details (weights, gradients), or a communication channel. We introduce a Status-Quo loss (SQLoss) that incentivizes cooperative behavior by encouraging policy stationarity. We also describe an approach to transform a two-player game (with visual inputs) into its IPD formulation through self-supervised skill discovery (IPDistill). We show how our approach outperforms existing approaches in the Iterative Prisoner's Dilemma and the two-player Coin game.

KEYWORDS

Reinforcement Learning; multi-agents; emergent cooperation

1 INTRODUCTION

Social dilemma situations often bring out the contradiction between individual and group rationality. In such situations, individual rationality suggests outcomes that lead to worse outcomes for the group [7, 13, 32, 33]. For instance, when sharing a pool of resources (such as a fishing pond or forest), individual rationality suggests overuse since the costs incurred through a single agent's selfishness are distributed equally across all agents while the rewards accrue to the selfish agent. Such collective individual rationality (or selfishness) leads to outcomes that are harmful to the group (eg. deforestation, draining the resource pool). Further, several studies have shown that humans do not exhibit such individual rationality in similar situations [33, 34]. Therefore, it is unclear why and how humans have evolved this individually irrational (or cooperative) behavior.

Game theorists have been studying the development of cooperation between rational agents [2, 9, 10, 12, 18] and the Prisoner's Dilemma [36] provides a sound theoretical framework to study this conflict between individual and group rationality. The question is: How can one incentivize agents to evolve cooperative behavior in these kinds of situations? A large body of work has looked at the mechanism of emergence of cooperation through reciprocal behaviour and indirect reciprocity [3, 28–30, 43], though variants of reinforcement using aspiration [26], attitude [5] or multi-agent reinforcement learning [38, 47], and under specific conditions [4] using different learning rates [6] similar to WoLF [1] as well as using embedded emotion [48], social netrworks [31, 39].

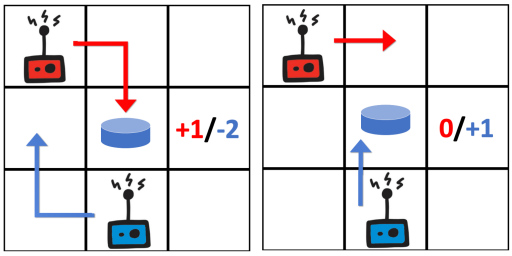


Figure 1: In Coin Game, 2 agents (Red and Blue) appear at random positions in a 3x3 grid. A coin of color Red or Blue appears randomly in a location. Agents traverse the grid to pick coins which maximises their reward. For each agent, eating a coin of any color gives +1 reward, but eating a coin of opponent's color penalizes the opponent with −2 reward. The best strategy (which maximizes long-term reward) to play requires cooperating with the opponent.

However, these approaches do not directly apply to Deep Reinforcement Learning agents. Recent work [15, 20, 35], in this direction, focuses on letting agents learn strategies in multi-agent settings through interaction with other agents. Leibo et al. [22] define the problem of social dilemma in the deep reinforcement learning framework and analyze the outcome of the 2d Fruit gathering game [15] as a general sum matrix game. They vary the abundance of resources and the cost of conflict to generate degrees of cooperation between agents. Hughes et al. [14] define an intrinsic reward (inequality aversion) that attempts to reduce the difference in rewards among agents by having an aversion to both advantageous (guilt) and disadvantageous (unfairness) inequity. This handcrafting of loss with mutual fairness leads to better cooperation but also makes the agent vulnerable to exploitation. LOLA [8] achieves high levels of cooperation in the Coin Game (see Figure 1) and in the Iterative Prisoner's Dilemma. However, LOLA assumes access to the opponent's policy parameters and gradients. This access is analogous to getting complete access to the opponent agent's

*Work done during the Adobe MDSR research internship Program

brain and devising a strategy with complete knowledge of how they are going to play. Wang et al. [44] propose an evolutionary deep reinforcement learning setup to evolve 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 achieve cooperation. However, sharing rewards in this indirect way enforces cooperative behavior rather than evolving it through individual behavior.

We **propose** an approach to evolve cooperative behavior between RL agents **without** sharing internal details (weights, gradients, etc.) or a communication channel. We introduce a **Status-Quo** (*SQLoss*) loss that can be thought of as a stable strategy loss which encourages agents to explore stationary policies. By stationary policy, we mean a policy (which defines the learning agent's way of behaving at a given time) where agents have an incentive to stick to their current actions as opposed to changing actions.

We evaluate the policy gradient learner with Status-Quo Loss on the IPD & Iterated Matching Pennies (IMP) and show that RL agents trained with the Status Quo loss converge to a cooperative behavior. This result is particularly interesting because behavioral economists [16, 17, 37, 42] have studied the effect of the status-quo bias in decision making in humans. Our work shows that it is possible that the status-quo bias helps to evolve cooperative behavior in human groups.

To work with games that have visual input [8], we extend the idea in [45]. They start with two policies, cooperation (denoted by π_C) and defection (denoted by π_D). They use these policies to generate intermediate policies with varying degrees of Cooperation and Defection. During gameplay, each agent predicts the degree of cooperation of its opponent and chooses a policy of the same degree of cooperation. We propose an IPD-Distilling architecture (*IPDistill*) to transform a multi-agent game to its IPD formulation. *IPDistill* uses self-supervision and clustering to learn cooperation and defection policies automatically. We use *IPDistill* to transform a game to an IPD, followed by *SQLoss* enhanced agents to evolve cooperative behavior for RL agents in the Coin game [8]. We show that RL agents trained using our approach achieve higher levels of cooperation and average reward than those trained with existing approaches (Section 4).

2 APPROACH

In this section, we formally describe our problem setup by borrowing the notation from [8]. A multi-agent game is specified by a tuple $G = \langle S, U, P, r, n, \gamma \rangle$. In the game, n agents, $a \in A \equiv \{1, ..., n\}$, select actions, $u^a \in U$, to reach state $s \in S$ in the environment. The action vector $\mathbf{u} \in \mathbf{U} \equiv U^n$ results in a state transition according to the transition function $P(s'|s, \mathbf{u}) : S \times \mathbf{U} \times S \rightarrow [0, 1]$. The reward functions $r^a(s, \mathbf{u}) : S \times \mathbf{U} \rightarrow \mathbb{R}$ denotes the rewards and $\gamma \in [0, 1)$ is the discount factor. For each agent, a, the discounted future return from time t is defined as $R_t^a = \sum_{l=0}^{\infty} \gamma^l r_{t+l}^a$.

Each agent independently attempts to maximize its expected total discounted reward. Several approaches to solve this use policy gradient based methods [41] (example, REINFORCE [46]). Policy gradient methods update an agent's policy, parameterized by θ^a, by performing gradient ascent on the expected discounted total

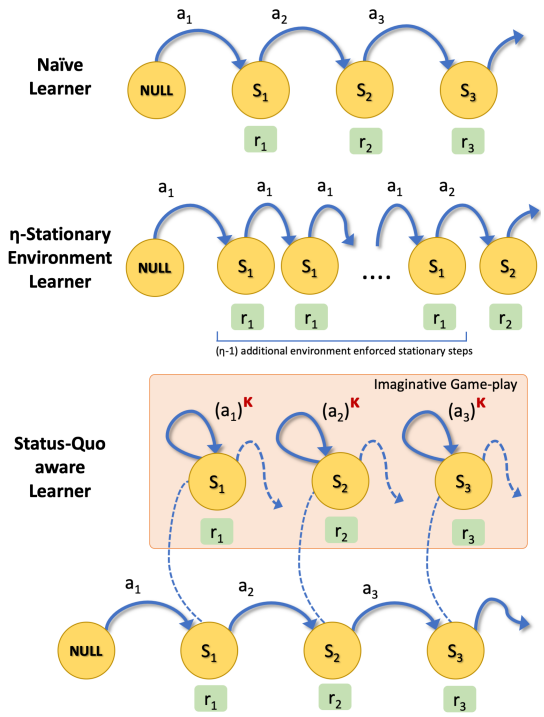


Figure 2: Converting a Naïve Leaner into a Status-Quo aware Learner using the imaginary η-Stationary Environment intuition. In the figure, K refers to parameter κ as used in the paper Section 2.1.3

reward $\mathbb{E}[R_0^a]$. In the Iterative Prisoner's Dilemma (IPD), agents trained with this update method converge to a sub-optimal mutual defection equilibrium [23]. The IPD game has two agents a_1 and a_2. Each agent has two possible actions, Cooperate (denoted by C) and Defect (denoted by D). Table 1a shows the reward matrix for all possible combinations of actions. The Nash Equilibrium (NE) in a single-step Prisoner's Dilemma is (D, D) [11]. The notation, (A,B) denotes the first agent takes an action A and the second agent takes an action B. This equilibrium describes the scenario when both players always defect. According to the Folk theorem [10, 27], the IPD has infinitely many Nash equilibria, out of which (D, D) is the most commonly occurring NE in deep reinforcement learning setup [8]. From the reward matrix, it is clear that both players would do better if they played (C, C). However, this is not a stable equilibrium because a_1 can improve its reward by playing D instead of C since $R_1(D, C) > R_1(C, C)$. Here, R_1 denotes the reward for a_1. The same reasoning holds for a_2. Therefore, both players have an incentive to switch from (C, C) to (D, D). In contrast, if the two players are playing (D, D), then neither has an incentive to change their action. This sub-optimal equilibrium attained by policy gradients motivates us to explore other techniques that could lead to a mutually beneficial equilibrium.

When RL agents optimize for individual rewards in an IPD, they reach a state of mutual defection, as discussed earlier. Here, we propose an approach to evolve cooperative behavior in two-player IPD games. Our approach has two components.

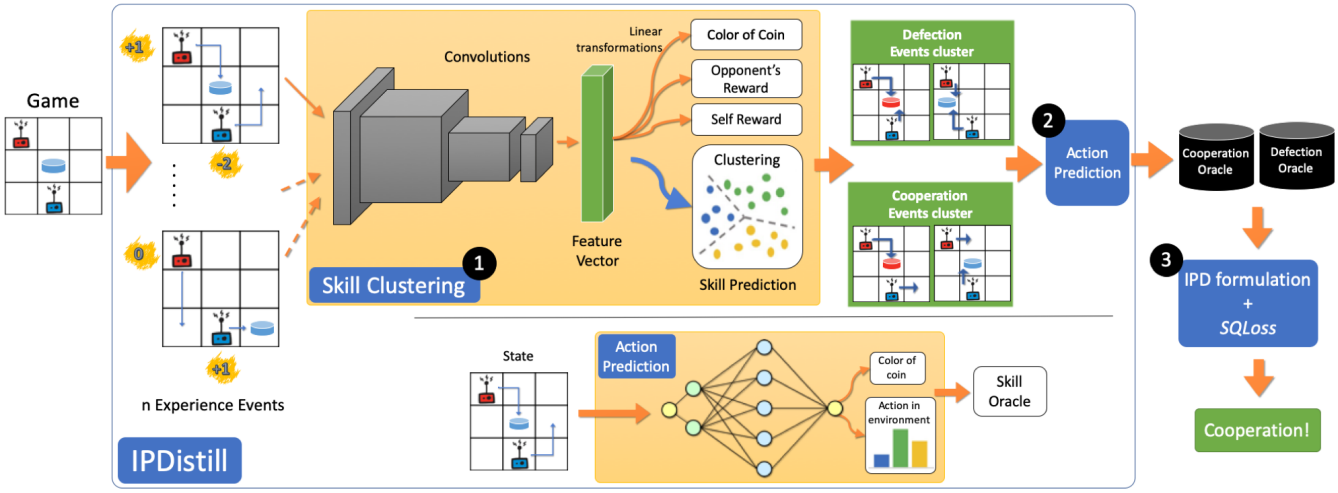


Figure 3: Framework Overview. Some parts of the network are representational, actual parameters are provided in Section 3

- In Section 2.1, we describe the Status-Quo loss (*SQLoss*). We show how agents trained with an additional *SQLoss* evolve cooperative behavior in an IPD setting.
- In section 2.2, we describe the architecture of the *IPDistill* component which uses self-supervised skill discovery to transform a game into its IPD formulation.

We then show how applying *IPDistill* along with *SQLoss* evolves cooperative behavior between agents in the Coin game.

2.1 SQLoss: Encouraging a Stationary Policy

2.1.1 ***Naïve Learner (NL).*** Following the notation of [8], let θ^a denote a parameterized version of an agent's policy π^a and $V^a_{\theta^1,\theta^2}$ denote the expected total discounted reward for agent a. Here V^a is a function of both agent's policy parameters (θ^1, θ^2). The objective of a Naïve Learner (NL) at i^{th} iteration, is to update θ^a_i to θ^a_{i+1}, such that it maximizes it's expected total discounted reward .

θ^a_{i+1} is updated as follows:

$$\begin{aligned}\theta^1_{i+1} &= argmax_{\theta^1} V^1(\theta^1, \theta^2_i)\\ \theta^2_{i+1} &= argmax_{\theta^2} V^2(\theta^1_i, \theta^2)\end{aligned} \tag{1}$$

The gradient ascent rule to update θ^1_{i+1} in reinforcement learning is given by:

$$\begin{aligned}f^1_{nl} &= \nabla_{\theta^1_i} V^1(\theta^1_i, \theta^2_i) \cdot \delta\\ \theta^1_{i+1} &= \theta^1_i + f^1_{nl}(\theta^1_i, \theta^2_i)\end{aligned} \tag{2}$$

Where δ is the step size of the updates. Agents trained to optimize for individual rewards evolve mutually harmful behavior and subsequently obtain a lower average reward.

2.1.2 ***η-Stationary Environment Learner.*** Here we introduce the intuition behind our Status-Quo loss. A Naïve Learner plays the game in an environment and gathers the experience for updates defined by Equation 2. If the environment enforces a *stationary-policy* (defined earlier in Section 1) for η horizons, then the action an agent takes at time-step t will be executed for the next η future time-steps. Since the environment is fair to all agents, the η time-step enforced *stationariness* is synchronous to both agents. This means both the agents are forced to stick with the same action every η steps. Figure 2 shows the states, rewards, and actions for such a scenario. A Naïve learner playing in an η-stationary environment is incentivized to make decisions that optimize long-term rewards while having a stationary policy.

The stationary policy reduces the likelihood that an agent will defect in a mutually cooperative situation. This is because the policy encourages sticking to the current action. Further, optimizing the long-term reward with a stationary policy reduces the impact of the short-term gains introduced by defection. Therefore, agents playing an IPD game in an η-stationary environment are more likely to evolve cooperative behavior.

However, in general, it might not be possible to alter environment dynamics to enforce stationarity. Therefore, we introduce a Status-Quo loss that encourages a stationary policy and, subsequently, cooperative behavior through a separate loss term.

2.1.3 ***Status-Quo aware Learner (SQLoss).*** Let τ denote an episode of actual experiences of the agent in the environment (T is the horizon of the episode) as $\tau = (s_0, u^1_0, u^2_0, r^1_0, r^2_0, \cdots, r^1_T, r^2_T)$, while $\hat{\tau}$ denotes the imaginary episode of the agent as a result of maintaining the Status-quo. $\hat{\tau}_t$ denotes the trajectory from time t, and $\phi_t(\kappa_t)$ is states, actions and rewards at time t repeated κ_t times, and κ (>=1) denotes the length of the imaginary game-play of the agents.

$$\begin{aligned}\hat{\tau}_t = (\phi_t(\kappa_t), (s_{t+1}, u^1_{t+1}, u^2_{t+1}, r^1_{t+1}, r^2_{t+1})_{\kappa_t+1}, \cdots,\\ (s_T, u^1_T, u^2_T, r^1_T, r^2_T)_{T+\kappa_t-t})\end{aligned} \tag{3}$$

$$\phi_t(\kappa_t) = \big[(s_t, u^1_{t-1}, u^2_{t-1}, r^1_{t-1}, r^2_{t-1})_0, \cdots, (s_t, u^1_{t-1}, u^2_{t-1}, r^1_{t-1}, r^2_{t-1})_{\kappa_t - 1}\big] \quad (4)$$

here, κ_t is κ sampled from Discrete Uniform distribution $\mathbb{U}\{1, z\}$ (both inclusive, z is a hyper-parameter ≥ 1) at time t.

The discounted return for agent a at time-step t is $R^a_t(\tau)$. The imaginary status-quo discounted return is $\hat{R}^a_t(\hat{\tau})$, which is defined by:

$$\begin{aligned} R^a_t(\tau) &= \sum_{l=t}^{T} \gamma^{l-t} r^a_l \\ \hat{R}^a_t(\hat{\tau}) &= \Big(\frac{1-\gamma^\kappa}{1-\gamma}\Big) r^a_{t-1} + \gamma^\kappa R^a_t(\tau) \\ &= \Big(\frac{1-\gamma^\kappa}{1-\gamma}\Big) r^a_{t-1} + \gamma^\kappa \sum_{l=t}^{T} \gamma^{l-t} r^a_l \end{aligned} \quad (5)$$

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

$$\begin{aligned} \nabla_{\theta^1}\mathbb{E}[R^1_0(\tau)] &= \mathbb{E}\left[R^1_0(\tau)\nabla_{\theta^1} log\pi^1(\tau)\right] \\ &= \mathbb{E}\Big[\sum_{t=0}^{T} \nabla_{\theta^1} log\pi^1(u^1_t|s_t) \cdot \sum_{l=t}^{T} \gamma^l r^1_l\Big] \\ &= \mathbb{E}\Big[\sum_{t=0}^{T} \nabla_{\theta^1} log\pi^1(u^1_t|s_t)\gamma^t \big(R^1_t(\tau) - b(s_t)\big)\Big] \end{aligned} \quad (6)$$

$$\begin{aligned} \nabla_{\theta^1}\mathbb{E}[\hat{R}^1_0(\hat{\tau})] &= \mathbb{E}\left[\hat{R}^1_0(\hat{\tau})\nabla_{\theta^1} log\pi^1(\hat{\tau})\right] \\ &= \mathbb{E}\left[\sum_{t=0}^{T} \nabla_{\theta^1} log\pi^1(u^1_{t-1}|s_t) \cdot \left(\sum_{l=t}^{t+\kappa} \gamma^l r^1_{t-1} + \sum_{l=t}^{T} \gamma^{l+\kappa} r^1_l\right)\right] \\ &= \mathbb{E}\left[\sum_{t=0}^{T} \nabla_{\theta^1} log\pi^1(u^1_{t-1}|s_t) \cdot \left(\Big(\frac{1-\gamma^\kappa}{1-\gamma}\Big)\gamma^t r^1_{t-1} + \gamma^\kappa \sum_{l=t}^{T} \gamma^l r^1_l\right)\right] \\ &= \mathbb{E}\Big[\sum_{t=0}^{T} \nabla_{\theta^1} log\pi^1(u^1_{t-1}|s_t)\gamma^t \big(\hat{R}^1_t(\hat{\tau}) - b(s_t)\big)\Big] \end{aligned} \quad (7)$$

where $b(s_t)$ is a baseline for variance reduction, γ is the discount rate.

Then the update rule $f_{sql,pg}$ for the policy gradient-based Status-Quo-learner (SQL-PG) is,

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

where α and β denotes the loss scaling factor for the reinforce and the imaginative gameplay, respectively.

2.2 IPDistill: Reducing a Game to an IPD formulation

In the previous section, we focus on the IPD formulation of the games, where agents are only allowed to either cooperate or defect at each step. Table 1 tabulates the payoff matrices of the iterated versions of different games. In iterated games, agents interact by playing a stage game (such as the prisoner's dilemma) numerous times.

We propose an approach (*IPDistill*) to transform a two-player game into its IPD formulation. We explain the idea using the coin game described in paper [8]. Figure 3 shows a high-level description of *IPDistill*. First, we initialize RL agents with random weights and pit them against each other in the Coin game (Figure 1). In the game, whenever an agent captures a coin, it obtains a reward, depending on its color and the color of the coin it captured. A cooperative strategy in the coin game is one where agents only capture coins of their color. In contrast, a defection strategy is one where agents capture all the available coins. Therefore, the notion of cooperation or defection in the Coin game is closely linked to the color of the coins. Whenever an agent receives a reward (when it picks a coin), we store a sequence of the last three states up to the current state. This collection of sequences is used to train the *IPDistill* network. The network takes as input a sequence of states and predicts the rewards of the agents and the environment attributes. Predicting the environment variables provides an implicit understanding of the environment to the agents while knowing the rewards of the opponent is needed as it is essential in differentiating defection from cooperation. We then use Agglomerative Clustering on the embeddings of the final layer to create two clusters, which leads to the discovery of cooperation and defection automatically. Figure 3 shows examples of the events clustered in the cooperation and defection skills discovered by *IPDistill* for the Coin game.

We also train an action prediction network for each discovered strategy cluster. Each agent trains neural networks, one for each skill/strategy such that given a state and skill as input, it can predict the action which needs to be taken. Figure 3 shows the network diagram with appropriate input & output of the network. More details about the specific architecture parameters are specified in Section 3. The network gets a single state as input, which is passed by a set of convolution layers that extract features from the states. Along with predicting the action, we also predict the environment attributes, which helps in training the action prediction network better. We train the network with three losses, action prediction loss, color-of-coin prediction loss, and the L2 weight regularization loss (with weight 1e−3) that ensures stable learning. Therefore given a state, the action prediction network for a strategy returns the action that an agent should take to play according to that strategy. The RL agents consult these networks during gameplay. Therefore, at each step, the agent can either take an action recommended by the cooperation or the defection oracles (trained networks). Hence, *IPDistill* reduces the game to an IPD formulation. Figure 4 shows the schematic diagram of the proposed reduction.

Hence, *IPDistill* reduces the game to an IPD formulation.

3 EXPERIMENTAL SETUP

In this section, we provide details of our experiments and values of hyper-parameters used in our experiments.

3.1 IPDistill: Skill Clustering

We define an Experience Event for the Coin Game as a sequence of states when either agents receive a positive or a negative reward.

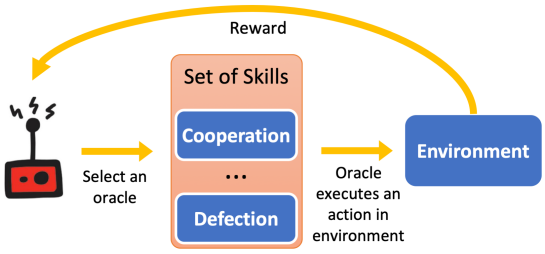


Figure 4: Reduction of a game to an IPD formulation. We summarize complex skills using oracles and the agent has to learn a meta-policy to switch between skills.

Our architecture for Skill Clustering consists of two components. First, the Experience Event encoder that takes as input an Experience event and outputs a feature representation of the sequence. The Event encoder encodes each of the states from the sequence using a series of standard Convolution layers with a kernel of size 3. This is followed by a fully-connected layer with 100 neurons that outputs a unified representation of the Experience Event.

The Color-of-Coin, self-Reward, opponent-Reward, and Skill prediction branches consist of a series of dense layers with no non-linear activations enabling us to cluster the feature vectors using a Linear Clustering algorithm, Agglomerative Clustering. We also experiment with K-means clustering algorithm which gives similar results. We use *Binary-Cross-Entropy (BCE)* loss for classification, *mean-squared error (MSE)* loss for regression and *Adam* [19] as our optimizer with a learning-rate of $3e-3$.

For the Action Prediction Network, input is the game state, encoded using a series of Convolutions with kernel-size 3 and *relu* activations. The Action prediction and Color-of-Coin prediction branches consist of a series of fully-connected layers with *relu* activation with *weighted*-BCE and BCE as the loss functions respectively. We use *L2* regularization and *Gradient Descent* (learning-rate 0.001) optimizer.

3.2 SQLoss

For all our experiments with Learners, we use a policy gradient-based learning where we train agents with the Actor-Critic method [40]. Each agent is parameterised with a policy actor and -critic for variance reduction in policy updates.

We use parameters similar to Foerster et al. [8]. During training, we use gradient descent with step size, $\delta = 0.005$ for the actor and 1 for the critic. We use a batch size of 4000 for LOLA [8] and 500 for *SQLearner* for rollouts. We keep the trace-length (length of an episode) to 200 for both IPD and IMP in our experiments. The discount rate γ is set to 0.96 for the prisoners' dilemma. The high value of γ allows for long time horizons, thereby incentivising long-term reward.

We randomly sample κ from $\mathbb{U} \in (0, 10)$ for each step, independently for each agent.

In order to compare our results with previous work, we use the Normalized Discounted Reward (NDR) defined as,

$$NDR = (1-\gamma)\sum_{t=0}^{T}\gamma^t r_t \tag{9}$$

Section 4 describes the results obtained in more detail with additional discussion and analysis in Section 5.

4 RESULTS

We present the results first from the proposed SQLoss, and then we show results of the Skill clustering strategy on Coin Game. We also provide the Probability of Cooperation plots for both agents for the IPD game and the IMP game.

For iterated games, we consider two classic infinitely iterated games, the iterated prisoners dilemma (IPD), [24] and iterated matching pennies (IMP) [21]. Each round in these two environments requires a single action from each agent. We can obtain the discounted future return of each player given both players policies. For CoinGame, a more difficult two-player environment, where each round requires the agents to take a sequence of actions and exact discounted future reward can not be calculated.

4.1 Developing cooperation in Iterated Games

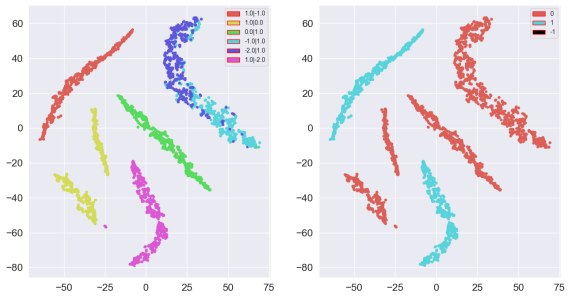
4.1.1 ***IPD.*** Figure 7 shows the average reward and rate of defection for two Naïve learners in the IPD game with a η-stationary environment. As expected, we observe that enforcing stationarity in the environment induces cooperative behavior.

The results obtained for the game of IPD are tabulated in Figures 6a & 6b. Figure 6b shows the Probability of Cooperation for both the agents for the game of IPD with environment enforcing the stationary policy with $\eta = 20$. Agents initially decide to defect, but eventually, both the agents choose to cooperate. They occasionally do defect and end up in the Defect-Defect state, but they again switch to cooperation for the majority of the game.

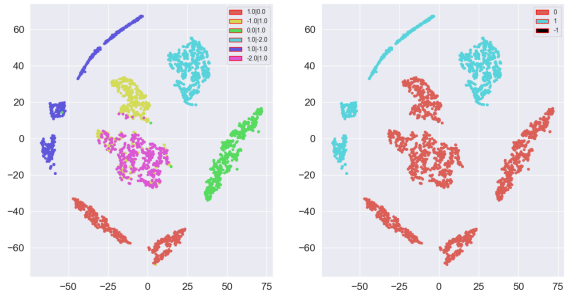
Having $\eta = 10$ has a similar effect on the agents, but the convergence to cooperation takes longer. Setting the parameter η to 5 did not result in cooperation, as visible in Figure 6a, which aligns with our understanding that there is a minimum threshold, a minimum policy stationarity, that the agents require in order to overcome the loss from the rewards when taking C as compared to D.

We experiment with the proposed changes with the learning policy, as shown in Section 2.1.3 for the traditional IPD game. Figure 7 shows the results obtained on training the agents using the proposed SQLoss. The SQLearner agents attempt to defect for a few epochs, but then switch to cooperation resulting in CC state. The SQLearner agents also give high NDR scores on average (-1.05 vs. -1.2). The Naive Learners trained using Policy gradient (NL-PG) end up defecting, giving a very low NDR of -2.0. For this experiment, we sample κ from the Discrete Uniform distribution $\mathbb{U} \in (0, 10)$ randomly for each step. This ensures that κ for both the agents is not synchronous which makes it more challenging. With synchronous κ, it would collapse to a setting similar to the η-Stationary environment.

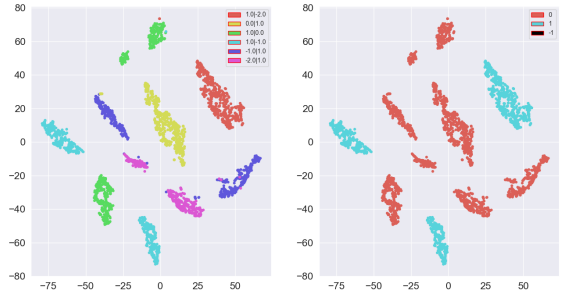
4.1.2 ***IMP.*** Figure 8 shows the average Normalized Discounted Reward of 5 independent trials for a pair of SQLearner agents in the IMP game. The Nash Equilibrium for IMP game is 50% Heads/50% Tails. As visible in Figure 8, the Naïve Learners trained using policy-gradient (denoted by NL-PG) end up diverging from the equilibrium. The LOLA-PG agents still manage to stay near the equilibrium but farther as compared to the pair of SQLearner agents. SQLearner agents converge at the Nash Equilibrium of the IMP game.



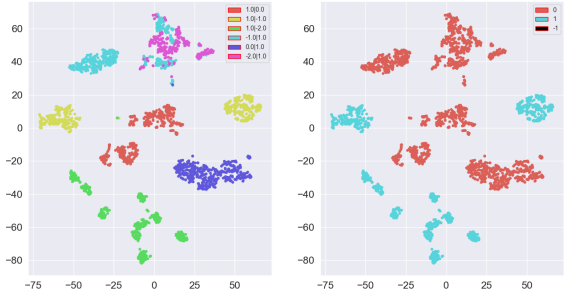
(a) 3-dimensional feature vector



(b) 4-dimensional feature vector



(c) 10-dimensional feature vector



(d) 100-dimensional feature vector

Figure 5: In each subfigure, Left figures represent the actual rewards obtained by both the agents (the ground-truth), while the figures on the Right represent the clusters obtained by Agglomerative Clustering (when no-of-clusters was set to 2). For feature vectors of higher dimensions (>2), we show their t-SNE [25] projections. In the figures on the left, the Legend shows the events that give $Agent_1$ a reward of r_1 and $Agent_2$ a reward of r_2 denoted in the format $r_1|r_2$. In the plots on the right, from the legend, Class 0 and Class 1 represent the Cooperation and Defection clusters respectively. From the figures, all the events that resulted in the opponent getting penalized belong to the same cluster, namely the Defection cluster.

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

(a) Prisoner's Dilemma (IPD)

	Head	*Tail*
Head	(+1,-1)	(-1,+1)
Tail	(-1,+1)	(+1,-1)

(b) Matching Pennies (IMP)

	C_{skill}	D_{skill}
C_{skill}	(1.0, 1.0)	(0.0, 2.8)
D_{skill}	(2.8, 0.0)	(-1.0, -1.0)

(c) Coin Game in IPD formulation

Table 1: Tables above show the payoff matrices of different games. The payoff for Coin game was inferred by averaging out the returns for a sequence of states when the agent plays the game using a skill in the environment. The absolute payoffs for such a game can vary, but the relative value of each skill remains the same. Here the C_{skill} & D_{skill} denote the Cooperation & Defection skills respectively.

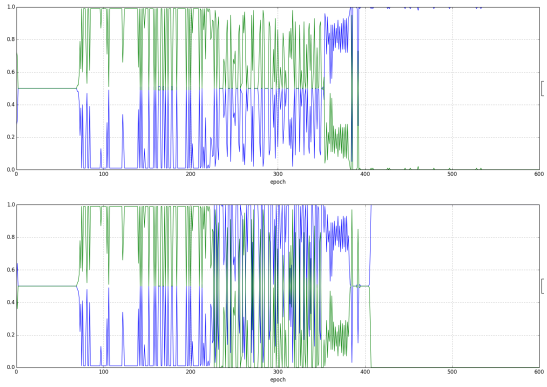
4.2 IPDistill: Skill Clustering

Figure 5 shows results obtained from the clustering of the Experience Events for Coin Game. From Figure 5a, the Left figure represents the Experience Events with similar rewards representing the ground truths while the Right figure shows the same set of events now labeled with the class prediction from the Skill Clustering network. As visible in the Figure, the events where the Red agent defected are clustered in Cluster 0 (denoted in red, they represent Defection), while the remaining events belong to the other cluster (Cluster 1), representing Cooperation strategy.

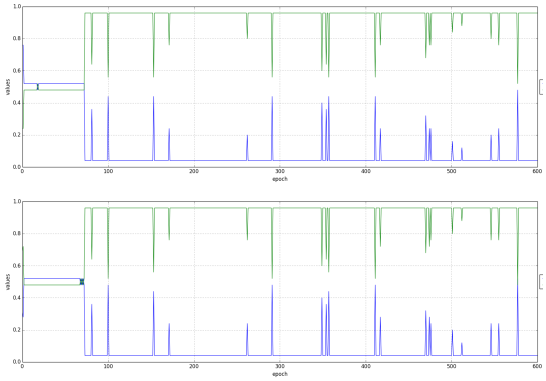
We also experiment with different dimensions for the feature vectors and obtain similar clustering results with sufficient training. Figures 5a, 5b, 5c & 5d show the results obtained for 2-D, 3-D, 10-D and 100-D feature vectors respectively.

Once the cooperation and defection skills are clustered in the feature space, we train an Action Prediction Network to predict the action at any state of the game.

We train the networks, for each strategy, using the discovered skills and then evaluate the oracles by executing it in the environment keeping the opponent frozen. We initiate 5000 independent games with each game-play lasting for a maximum of 2 steps (For CoinGame an agent can pick the coin in a maximum of 2 steps, irrespective of both the agent's and coin's positions). We terminate the game once an agent picks the coin. Table 2 shows the percentage of picking coins of a particular color for a given strategy when guided



(a) ($\eta = 5$)-Stationary environment. Agents first cooperate but eventually end up defecting



(b) ($\eta = 20$)-Stationary environment

Figure 6: Instantaneous probability of cooperation for both the agents in the IPD game when the environment enforces the stationary criterion. Green line indicates probability of cooperation while Blue line indicates probability of defection.

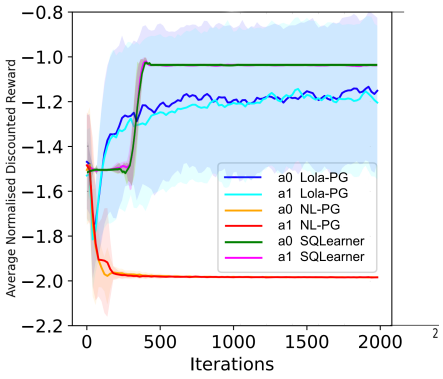


Figure 7: Results of 10 independent trials for the IPD game with SQLearner agents along with the results from [8] for different pairs of agents as baselines. For SQLearner agents, κ varies from 0 to $z = 10$. $a0$ SQLearner's reward graph is hidden because of overlap with $a1$ SQLearner's reward shown in magenta. Here x-axis is epochs while y-axis shows the probability

Color of Coin	Max steps	Percentage of coins picked	
		Defection Skill	**Cooperation Skill**
Red	Max 1 steps	12.7%	46.7%
	Max 2 steps	22.4%	95.8%
Blue	Max 1 steps	48.4%	4.7%
	Max 2 steps	99.4%	8.4%

Table 2: For Coin game. Percentage of coins picked by the *Red* agent when the actions are predicted by the oracle trained from the events obtained by Skill clustering.

by the Action Prediction Network. The network can learn to either pick or leave the coin based on the strategy.

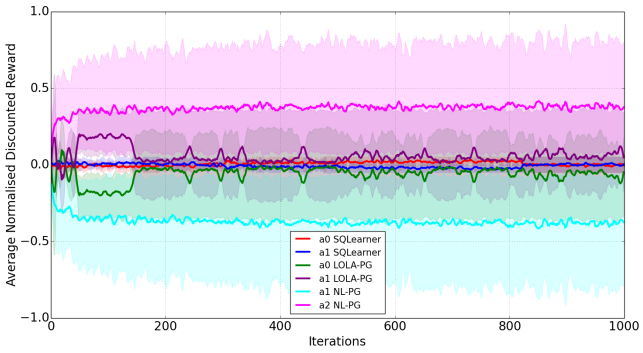
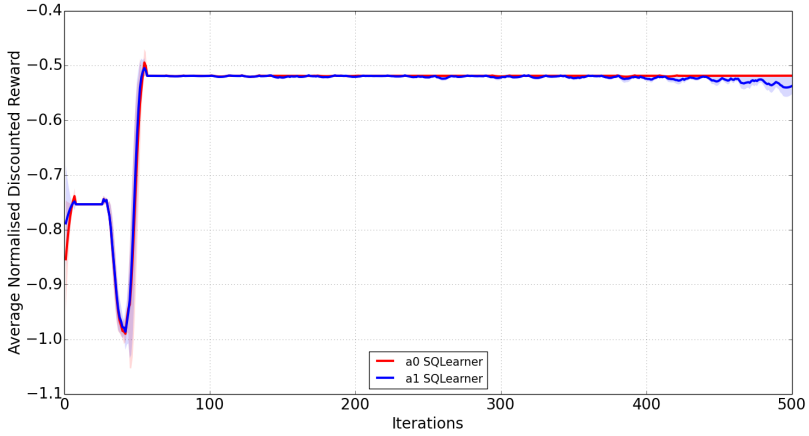


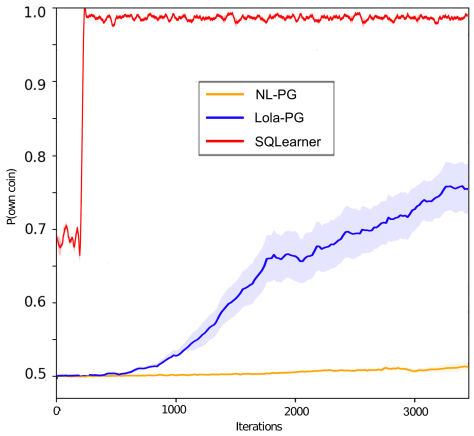
Figure 8: Results of 5 independent trials for the IMP game with pairs of Lola, NL and SQLearner agents. SQLearner agents converge on the Nash Equilibrium which is 50%/50% Heads and Tails resulting in NDR close to 0. Best viewed in color.

Action Prediction Network consists of a set of convolution layers to encode the game state, followed by a full-connected layer to obtain the state feature vector, followed by 2 branches consisting of a fully-connected layer to predict the coin-color and the action. We use the *relu* activation and the Gradient Descent optimizer for training the network with $lr = 1e-2$. We train the network for $\sim$ 700 epochs with a batch size of 128.

After obtaining the skills for Coin game, Cooperation, and Defection, agents play the skills in the environment and obtain the following reward values for the skills. The trained oracles with the reward values are then used in the IPD formulation of CoinGame with trained oracles. Figure 9 shows the average NDR of the pair of SQLearner agents along with *Probability(picking-own-coin)*. From the figure, the oracle-based SQLearner agents have *Probability(picking-own-coin)* higher than Naïve Learner and Lola agents. The Figure 9a shows the average Normalized Discounted Reward (NDR) which



(a) Average NDR for a pair of SQLearner



(b) Probability(picking-own-coin) for SQLearner agents with baseline results from [8]

Figure 9: In the figures above, we have (a) average NDR for a pair of SQLearner agents, and (b) *Probability(picking-own-coin)* for CoinGame. y-axis represents the average NDR results for 5 independent runs of the game.

reaches -0.5 when trained for sufficiently long time, which is much higher than Naïve Learner agents.

5 DISCUSSION AND ANALYSIS

In this section we perform various experiments to understand the effect of SQLoss on the agent's behaviour, the effect of various parameters on the results and explain some of the observations.

We observe that the SQLearner agents initial play randomly for a few epochs. Initial random play of agents forces the Status-Quo loss to penalize the agents heavily which results in the logits getting diverged and subsequently resulting in very high probabilities of taking an action given a state. Eventually the agents stabilize and thus the probabilities come near zero and they converge to CC.

We also observe certain sudden drops in Cooperation probabilities after every few epochs when the Naïve agents play in η-stationary environment. This trend is clearly visible in Figure 6b. We believe this is due to the inherent incentive to exploit the opponent which entices the agents to exploit, but realizing it won't help, then agents switch to CC again. The interval between consecutive drops in *Probability(Cooperation)* reduces as training proceeds.

5.0.1 ***Exploitability.*** Given the fact that the RL agents don't have any prior information about the opponent, its necessary that they evolve their strategies based on the opponent. For a SQLearner, we attempt to understand its adaptability when the opponent suddenly switches its strategy mid-game. In contrast to playing against an SQLearner, when the agent learns to Cooperate, the SQLearner agent when played against an *always-Cooperate* agent and against an *always-Defect*, it learns to Defect. This nature of exploiting the opponent, and prevent from being exploited are useful indications of the agent adapting to its opponent. The SQLearner agent when played against a Tit-for-tat (TFT) agent failed to converge to Cooperation possibly because of the difficulty to model instantaneous changes. Both agents converged to DD state.

5.0.2 ***Effect of κ, z & η on convergence.*** In this section we attempt to explore the effect of various new parameters introduced in the Section 2 on convergence to CC.

For η-Stationary Environment, η controls the amount of stationary criterion the environment enforces on the agents. Experimenting with varying η, we observed that starting with $\eta = 5$, the agents reach the CC state for quite some time, but each of the agents attempts to exploit the opponent and both eventually end up defecting. The agents end up in DD state if played for long. But as we increase η (to lets say 20), the frequency of exploitation reduces, and the agents stay in equilibrium in the CC state. Figures 6a & 6b show the results of varying η and the nature of agents when η is relatively low.

κ and z, both control the degree of imaginary self-play. Since, κ_t is sampled from Discrete Uniform distribution $\mathbb{U}\{1, z\}$ (both inclusive), larger z implies larger κ_t. Having higher z adds variance in the rewards and thus possible future outcomes, but the overall long-term expected reward is still higher for CC, allowing the agents to converge to cooperation. It also makes the imaginary game-play increasingly asynchronous for both the agents. Larger κ results in faster cooperation.

6 CONCLUSION

We have described an approach to evolve cooperative behavior between RL agents playing the IPD and IMP game without sharing rewards, internal details (weights, gradients, etc.) or a communication channel. We introduce a Status Quo loss (SQLoss) that incentivizes cooperative behavior by encouraging policy stationarity. Further, we have described an approach to transform a two-player game (with visual inputs) into its IPD formulation through self-supervised skill discovery (IPDistill). Finally, we showed how our approach (IPDistill + SQLoss) outperforms existing approaches in the IPD, IMP and the two-player Coin game.

REFERENCES

[1] 2002. Multiagent Learning Using a Variable Learning Rate. *Artificial Intelligence* 136, 2 (April 2002), 215–250.

[2] Dilip Abreu, David Pearce, and Ennio Stacchetti. 1990. Toward a Theory of Discounted Repeated Games with Imperfect Monitoring. *Econometrica* 58, 5 (1990), 1041–1063. http://www.jstor.org/stable/2938299

[3] Robert Axelrod. 1984. *Robert Axelrod's (1984) The Evolution of Cooperation.* Basic Books.

[4] Dipyaman Banerjee and Sandip Sen. 2007. Reaching Pareto-optimality in Prisoner's Dilemma Using Conditional Joint Action Learning. *Autonomous Agents and Multi-Agent Systems* 15, 1 (Aug. 2007).

[5] Steven Damer and Maria Gini. 2008. Achieving Cooperation in a Minimally Constrained Environment. *Proceedings of the National Conference on Artificial Intelligence* 1, 57–62.

[6] Enrique Munoz de Cote, Alessandro Lazaric, and Marcello Restelli. 2006. Learning to Cooperate in Multi-agent Social Dilemmas. In *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS '06).*

[7] Thomas Dietz, Elinor Ostrom, and Paul C. Stern. 2003. The Struggle to Govern the Commons. *Science* 302, 5652 (2003), 1907–1912. https://doi.org/10.1126/science.1091015

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

[9] Drew Fudenberg, David Levine, and Eric Maskin. 1994. The Folk Theorem with Imperfect Public Information. *Econometrica* 62, 5 (1994), 997–1039. http://www.jstor.org/stable/2951505

[10] Drew Fudenberg and Eric Maskin. 1986. The Folk Theorem in Repeated Games with Discounting or with Incomplete Information. *Econometrica* 54, 3 (1986), 533–554.

[11] Drew Fudenberg and Jean Tirole. 1991. *Game Theory.* MIT Press, Cambridge, MA. Translated into Chinesse by Renin University Press, Bejing: China.

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

[13] Garrett Hardin. 1968. The Tragedy of the Commons. *Science* 162, 3859 (1968), 1243–1248. https://doi.org/10.1126/science.162.3859.1243

[14] 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. 2018. Inequity Aversion Improves Cooperation in Intertemporal Social Dilemmas. In *Proceedings of the 32Nd International Conference on Neural Information Processing Systems (NIPS'18).*

[15] Pérolat Julien, JZ Leibo, V Zambaldi, C Beattie, Karl Tuyls, and Thore Graepel. 2017. 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).*

[16] Daniel Kahneman. 2011. *Thinking, fast and slow.* Macmillan.

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

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

[19] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

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

[21] King Lee and K Louis. 1967. *The Application of Decision Theory and Dynamic Programming to Adaptive Control Systems.* Ph.D. Dissertation.

[22] Joel Z. Leibo, Vinicius Zambaldi, Marc Lanctot, Janusz Marecki, and Thore Graepel. 2017. 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.

[23] Adam Lerer and Alexander Peysakhovich. 2017. Maintaining cooperation in complex social dilemmas using deep reinforcement learning. (2017). arXiv:arXiv:1707.01068

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

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

[26] Dr Macy and Andreas Flache. 2002. Learning Dynamics in Social Dilemmas. *Proceedings of the National Academy of Sciences of the United States of America* 99 Suppl 3 (06 2002), 7229–36. https://doi.org/10.1073/pnas.092080099

[27] ROGER B. MYERSON. 1991. *Game Theory: Analysis of Conflict.* Harvard University Press. http://www.jstor.org/stable/j.ctvjsf522

[28] Martin Nowak and Karl Sigmund. 1993. A Strategy of Win-Stay, Lose-Shift That Outperforms Tit-for-Tat in the Prisoner's Dilemma Game. *Nature* 364 (08 1993), 56–8. https://doi.org/10.1038/364056a0

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

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

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

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

[33] Elinor Ostrom, Joanna Burger, Christopher B. Field, Richard B. Norgaard, and David Policansky. 1999. Revisiting the Commons: Local Lessons, Global Challenges. *Science* 284, 5412 (1999), 278–282. https://doi.org/10.1126/science.284.5412.278

[34] Elinor Ostrom and Roy Gardner. 1993. Coping with Asymmetries in the Commons: Self-Governing Irrigation Systems Can Work. *The Journal of Economic Perspectives* 7, 4 (1993), 93–112. http://www.jstor.org/stable/2138503

[35] Alexander Peysakhovich and Adam Lerer. 2018. 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.*

[36] Anatol Rapoport. 1974. *Prisoner's Dilemma — Recollections and Observations.* Springer Netherlands.

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

[38] Tuomas W. Sandholm and Robert H. Crites. 1996. Multiagent reinforcement learning in the Iterated Prisoner's Dilemma. *Bio Systems* 37 1-2 (1996), 147–66.

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

[40] Richard S Sutton and Andrew G Barto. 2011. Reinforcement learning: An introduction. (2011).

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

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

[43] Robert Trivers. 1971. The Evolution of Reciprocal Altruism. *Quarterly Review of Biology* 46 (03 1971), 35–57. https://doi.org/10.1086/406755

[44] Jane X. Wang, Edward Hughes, Chrisantha Fernando, Wojciech M. Czarnecki, Edgar A. Duéñez Guzmán, and Joel Z. Leibo. 2019. Evolving Intrinsic Motivations for Altruistic Behavior. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS '19).* International Foundation for Autonomous Agents and Multiagent Systems, 683–692.

[45] Weixun Wang, Jianye Hao, Yixi Wang, and Matthew Taylor. 2018. Towards Cooperation in Sequential Prisoner's Dilemmas: a Deep Multiagent Reinforcement Learning Approach. (2018). arXiv:arXiv:1803.00162

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

[47] Michael Wunder, Michael Littman, and Monica Babes. 2010. 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).*

[48] C. Yu, M. Zhang, F. Ren, and G. Tan. 2015. Emotional Multiagent Reinforcement Learning in Spatial Social Dilemmas. *IEEE Transactions on Neural Networks and Learning Systems* 26, 12 (2015), 3083–3096.